Fiabilité du signal des circuits logiques combinatoires sous fautes simultanées multiples

Denis Teixeira Franco

Fiabilité du Signal des Circuits Logiques Combinatoires sous Fautes Simultanées Multiples

Denis Teixeira Franco

Acknowledgments

Follows some acknowledgements, that have been written in French.

Et voilà, la thèse est finie. Cette section est consacrée aux remerciements aux personnes que d'une façon ou d'autre ont participé de ce parcours pendant trois années et quelques mois. Cette section représente aussi la possibilité d'une manifestation personnelle dans un document strictement technique et, alors, une précieuse opportunité.

Je remercie tous les membres de mon jury de thèse : M. Patrick Garda (président), MM. Alexandre Schmid et Patrick Girard (rapporteurs), MM. Fabrice Auzanneau et Yves Mathieu (examinateurs). Je remercie également mes directeurs M. Jean-François Naviner et Mme. Lirida Naviner pour l'évaluation publique très positive qu'ils ont faite du travail realisé.

Je voudrais remercier les personnes qui ont participé, directement ou indirectement, à la réussite de cette thèse. Je pense particulièrement aux professeurs du département de langues de l'ENST, qui m'ont appris à aimer la langue et la culture française. Les cours de FLE (Français Langue Étrangère) ont été une excellente opportunité de connaître des collègues du monde entier, avec leurs coutumes et richesses, qui m'ont fait voir le monde autrement.

Je remercie les interlocuteurs d'EGIDE, qui ont fait de leur mieux pour mener toutes les situations à bon terme. Un merci spécial à M. Bonenfant, qui s'est toujours battu avec EGIDE pour nous garantir les meilleures conditions de séjour possibles.

Je remercie les personnes du département COMELEC, dont quelques uns ont dépassé leurs attributions officielles et sont devenus des amis. Chantal et Karim, un grand merci à vous. Je remercie les membres du groupe SIAM : Mme. Desgreys et Messieurs Loumeau, Petit et Nguyen. Mon merci est extensible à Ali, Rachid et Hussein, qui ont rejoint le groupe. Les échanges et l'ambiance bonne humeur m'ont beaucoup aidé dans ce parcours.

Je remercie tout particulièrement mes directeurs de thèse, M. Jean-François Naviner et Mme. Lirida Naviner, pour la confiance qu'ils m'ont faite, pour leur manière de diriger ce travail de recherche. Vos conseils et suggestions m'ont permis de mener à bien ce travail. Je vous remercie aussi pour les opportunités que vous m'avez offertes, elles ont été fondamentales pour l'évaluation et maturation des recherches développées. J'espère pouvoir travailler en coopération avec vous dans l'avenir.

Je remercie les collègues en thèse et stage et les amis à l'école, qu'ils soient arrivés ou qu'ils soient partis pendant mon séjour, nous avons partagé des moments de sport, de pause café, de Butte aux Cailles, de karaokê, pique-nique ou simplement de rigolade au bureau. Sans ces moments de décontraction, je ne pense pas que la thèse se serait déroulée aussi bien que ça.

Je voudrais remercier ma famille, qui même au Brésil a eu un rôle très important dans cette conquête. Leur support logistique et psychologique a été très important et a beaucoup

aidé à réduire la distance qui parfois pesait énormément.

Je voudrais finalement remercier ma femme, qui a été la principale responsable pour l'achèvement de ce travail. Merci beaucoup Roberta, d'avoir cru à notre venue en France, d'avoir insisté quand j'avais déjà désisté, d'avoir compris mes moments de absence et de m'avoir supporté et soutenu dans les moments de tristesse et angoisse. Roberta, ce travail est dédié à toi.

Abstract

Integrated circuits have known a constant evolution in the last decades, with increases in density and speed that followed the rates predicted by *Moore's law*. The tradeoffs in area, speed and power, allowed by the CMOS technology, and its capacity to integrate analog, digital and mixed components, are key features to the dissemination of integrated circuits on the field of telecommunications. In fact, the progress of the CMOS technology is an important driver for telecommunications evolution, with the continuous integration of complex functions needed by demanding applications. The evolution scenario of the consumer electronics industry is a result of the fairly easy gains obtained with CMOS scaling, but this scenario is about to change, as scaling approaches some limits that make further improvements more difficult and even unpredictable.

The continuous reduction in the dimensions of integrated circuits has raised some serious problems to the implementation of nanometric circuits, like power consumption and dissipation, current leakage and parametric variations. Many of these problems lead to a reduction in the yield and the reliability of CMOS devices, what can seriously compromise the gains attained with technology scaling. To cope with these problems, work-around solutions are necessary, and more specifically, the sensibility of the circuits to transient faults must be improved.

Historically, transient faults were a concern in the design of memory and sequential elements only, and the protection of these elements is fairly simple and do not impose critical overheads, due to the static nature of the data involved. On the other way, the susceptibility of combinational blocks to transient faults, e.g., soft errors, thermal bit-flips, parametric variations, etc, increases as a side effect of technological scaling, and the protection of these blocks poses some problems to the designers. Known solutions are the use of fault-tolerant approaches, like modular redundancy and concurrent error detection but these solutions lead to important overheads in terms of implementation area, propagation time and power consumption. The referred solutions are traditionally targeted to mission critical applications, where reliability improvement and fault secureness are the main design objectives, and the resulting overheads can be accepted. In the case of mainstream applications, other design objectives are privileged and the implementation of these fault-tolerant approaches must be bounded by a detailed cost-benefit analysis.

Given the overheads associated with the traditional fault-tolerant approaches, alternative solutions based on partial fault tolerance and fault avoidance are also being considered as possible solutions to the reliability problem. These approaches are based on the application of fault tolerance to a restricted part of the circuit or hardening of individual cells, allowing fine-tuned improvements on the reliability and limiting the concerned overheads.

The choice of what reliability improvement method is better suited for a given application is not simple as the problem involves multi-criteria optimization. In this context, a fast and accurate evaluation of circuit's reliability is fundamental, to allow a reliability-aware

automated design flow, where the synthesis tool could rapidly cycle through several circuit configurations to assess the best option. Unfortunately, reliability analysis is a complex task, and computing its exact value is intractable for practical circuits.

Different methodologies have been proposed to evaluate the reliability of combinational circuits, where the computation of its exact value is limited by the size of the target circuits. For practical circuits, reliability analysis is generally bounded by simplified assumptions, like single-output, single-fault or single-path, what limits the results of the analysis. Furthermore, most of the proposed methodologies are based on external software packages or simulation, making difficult its integration to an automated design flow.

The current work proposes two new methods for reliability analysis, the first one based on a probabilistic binomial model and the second one based on signal probability propagation. The probabilistic binomial reliability analysis (PBR) method uses fault injection and functional simulation to determine an analytical model for the reliability of the circuit. Exploring the binomial probability distribution of multiple simultaneous faults, the proposed method allows an accurate estimation of the signal reliability of combinational logic circuits. The signal probability reliability analysis (SPR) uses a modified representation of signal probabilities to determine the cumulative effect of multiple simultaneous faults in the reliability of a circuit. Based on a straightforward signal probability computation algorithm, the methodology allows the evaluation of the logical masking capability of combinational logic circuits.

To validate the proposed methods, the reliability analysis of several fault-tolerant and hardened arithmetic circuits are presented and the bounds of applicability of these approaches are determined.

French Summary

Introduction

L'entrée de la technologie CMOS dans les dimensions nanométriques résulte de l'évolution prévue pour les circuits intégrés, déterminée par l'industrie des semi-conducteurs d'après les feuilles de route établies selon la loi de Moore [1]. Pourtant, la production des circuits nanométriques présente des défis de plus en plus critiques, qui demandent des efforts considérables de la communauté scientifique. Ces défis sont liés à des limitations d'ordre physique, économique et technologique, et se traduisent en un changement du comportement des structures fortement intégrées et en une difficulté pour les fabriquer avec la précision nécessaire [2].

La majorité des problèmes associés à la réduction des structures CMOS amène à une réduction du rendement de fabrication et de la fiabilité d'opération des circuits. Les technologies émergentes, conçues pour étendre, complémenter, voire substituer la technologie CMOS, seront très sensibles aux variations paramétriques des composants et aux défauts de fabrication. Les solutions proposées passent par les méthodes traditionnelles de tolérance aux fautes, tolérance aux pannes et durcissement des composants, mais passent aussi par un changement de la façon de concevoir et implémenter les fonctions intégrées, dans une méthode de conception ascendante. La méthode de conception ascendante est inspirée des architectures reconfigurables, telles que les FPGA, parfois avec une granularité beaucoup plus fine. Ces structures reconfigurables, appelées "nanotissus", sont des matrices de très haute densité qui peuvent être configurées pour réaliser des fonctions logiques diverses, en prenant en compte seulement les composants opérationnels.

La conception ascendante est une réponse envisagée pour la question du rendement de fabrication mais la fiabilité d'opération des circuits reste un problème critique, pour lequel les solutions proposées font appel aux techniques de tolérance aux pannes. Selon quelques études [2], la probabilité d'occurrence des fautes transitoires dans les systèmes nanométriques montera au fur et à mesure de l'augmentation de densité des composants intégrés, atteignant le même niveau observé dans les mémoires, où les fautes transitoires sont plus facilement traitées. Historiquement, les techniques de tolérance aux pannes étaient destinées aux circuits de mission critique, à cause des surcoûts matériels, de performance et de consommation d'énergie associés à son application. Son utilisation dans les circuits logiques non critiques dépendra directement de son rapport coût/bénéfice, ce qui n'est pas évident à déterminer, d'autant plus que l'occurrence de multiples fautes simultanées deviendra une réalité.

Alors, le problème qui se pose est la détermination de la fiabilité des circuits logiques, en amont dans le flot de conception, de façon à permettre l'application des méthodes de durcissement les plus adaptées à chaque système, et de permettre une exploration rapide de l'espace de projet disponible. La fiabilité en question est la fiabilité du signal et des

données, associée aux fautes transitoires et intermittentes. Les méthodes proposées pour ce type d'analyse sont généralement déconnectées du flot de conception des circuits intégrés et sont applicables à des parties spécifiques des circuits. Ce travail explore des méthodes d'estimation rapide de fiabilité, facilement intégrables dans le flot de conception, et qui peuvent permettre une comparaison préalable des différents choix architecturaux.

Fiabilité et technologie nanométrique

La réduction des dimensions des structures CMOS intégrés a une très forte répercussion sur la fiabilité des circuits. L'augmentation de la densité des circuits associée à la réduction de la tension d'alimentation fait augmenter la probabilité d'occurrence des fautes transitoires et aussi la probabilité d'occurrence de fautes multiples. Les fautes transitoires peuvent être originaires de plusieurs sources :

- Neutrons de haute énergie et particules alpha présentes dans l'atmosphère ;
- Bruit de couplage ou crosstalk ;
- Bruit thermique ;
- Fluctuations des tensions dues aux commutations des courants et IR drop (réduction de la tension d'alimentation dans un composant par la résistance des conducteurs) ;
- Bruit de substrat.

La prise en compte des ces problèmes est un défi urgent pour la communauté scientifique, ce qui représente la création et la validation de modèles et d'outils d'analyse adaptés aux besoins des concepteurs, intégrés dans le flot de conception de façon presque transparente. Parmi les problèmes mentionnés, les erreurs *soft* originaires des particules alpha et neutrons représentent un ancien défi, bien connu, mais qui prend une nouvelle dimension dans les structures nanométriques.

Les erreurs soft sont une menace classique pour les mémoires intégrées et pour les composants séquentiels, comme les bascules et flip-flops. L'interaction des particules alpha ou neutrons de haute énergie avec des composants CMOS peut faire commuter l'état d'un point mémoire ou d'une bascule, et cette erreur restera active jusqu'au moment où une nouvelle valeur est enregistrée. La prise en compte de ce type d'erreur est simple et les mémoires tolérantes aux fautes sont courantes dans les systèmes intégrés. Pourtant, les circuits logiques sont en train de devenir plus susceptibles aux soft erreurs à chaque nouvelle génération et ce niveau s'approche du niveau des mémoires non protégées [3]. L'équation (1) permet l'estimation du taux de soft erreurs (SER) dans les circuits CMOS dues aux neutrons de haute énergie.

$$SER \propto F \times A_{diff} \times exp(-\frac{Q_{crit}}{Q_{coll}}) \tag{1}$$

Dans ce modèle, F correspond au flot des particules, A_{diff} à la surface sensible à ces particules (surface des diffusions de la cellule), Q_{crit} au seuil de charge pour un changement de la valeur logique, et Q_{coll} correspond à la charge collectée par la cellule à l'occasion d'un impact de particule. La réduction des dimensions des composants amène à une réduction de la surface des diffusions et aussi de la capacité de collecte des charges. Par contre, la réduction de la tension d'alimentation correspond à une réduction du seuil Q_{crit}.

Quelques études ont prévu une augmentation du taux d'erreurs à cause de la réduction de la tension d'alimentation, mais des études plus récentes ont démontré que ce taux est en baisse pour les cellules individuelles, qu'il reste presque constant pour les matrices de mémoire et qu'il augmente pour les circuits logiques. La situation est plus difficile parce que la réduction des tensions d'alimentation n'a pas suivi le facteur prévu par les règles de *scaling*. Même si la sensibilité aux erreurs soft due aux neutrons n'est pas si critique que prévue, la sensibilité aux particules alpha est en train de monter. De plus, la probabilité d'erreurs multiples augmente beaucoup avec l'augmentation de la densité des circuits, où un même impact de particule peut générer des erreurs dans les cellules voisines [4].

Le bruit thermique est, peut-être, la plus grande menace à l'évolution des circuits intégrés [5]. L'augmentation de densité et de fréquence d'opération des circuits peut amener à l'occurrence d'erreurs logiques, ce qui est associé à des limites physiques incontournables. En fait, les fréquences d'horloges des circuits intégrés n'augmentent plus comme prévu par la loi de Moore, à cause des problèmes d'instabilité d'opération et aujourd'hui l'évolution technologique représente plutôt un gain en densité des circuits qu'un gain en fréquence d'opération.

Pour faire face aux erreurs transitoires, les concepteurs peuvent utiliser des méthodes de durcissement et les architectures tolérantes aux fautes et aux pannes. Ces méthodes représentent toujours un surcoût en surface, en consommation ou en vitesse et leur utilisation pour le sauvetage des circuits nanométriques n'est pas évidente. Pour cela, l'estimation de la fiabilité des circuits logiques est fondamentale pour déterminer les gains réels obtenus avec les méthodes de tolérance aux fautes et de durcissement des circuits.

Les architectures auto-contrôlables, tolérantes aux fautes et robustes

La prise en compte des erreurs transitoires dans les circuits logiques passe par l'utilisation de redondance, sous une forme spatiale ou temporelle. Alors, un circuit intégré robuste représente toujours un surcoût matériel, temporel et de consommation d'énergie. Pour les applications critiques, dans les domaines médical, militaire et spatial, entre autres, la fiabilité d'opération est la propriété la plus importante et ces surcoûts sont nécessaires. Par contre, pour les applications plus générales, l'évolution est guidée par les gains obtenus avec la réduction d'échelle des structures intégrées, et l'implémentation de circuits plus robustes sera limitée par cette règle.

Dans le présent travail, une bibliothèque de circuits arithmétiques robustes a été implémentée, dans le but d'étudier l'impact des différentes techniques dans la fiabilité des circuits. Même si les surcoûts sont bien connus et le gain en fiabilité en présence d'une seule faute est bien compris, l'augmentation de la probabilité d'erreurs multiples et la réduction de la fiabilité de circuits logiques nanométriques demande des nouveaux modèles d'estimation de fiabilité, de façon a mesurer le gains réels apportés par les méthodes de durcissement proposées.

Dans le domaine des circuits auto-contrôlables, deux méthodes ont été étudiées, la prédiction de parité et la duplication. La prédiction de parité permet la détection d'une erreur dans un circuit arithmétique du type additionneur [6] ou multiplicateur [7, 8] en suivant le modèle de parité en (2). Dans l'équation, Pa et Pb correspondent à la parité des données d'entrée, Pc est la parité des données de retenue et Ps est la parité prévue du résultat.

$$Ps = Pa \oplus Pb \oplus Pc \tag{2}$$

La Fig. 1 montre le schéma général de prédiction de parité dans un circuit arithmétique. La sortie ei, codée en représentation double rail, peut indiquer des erreurs dans le circuit qui ne sont pas détectables par le modèle en 2. Le contrôle de la parité est fait par les circuits externes à l'opérateur arithmétique. Le surcoût matériel pour la prédiction de parité est le plus bas parmi ceux des méthodes étudiées.

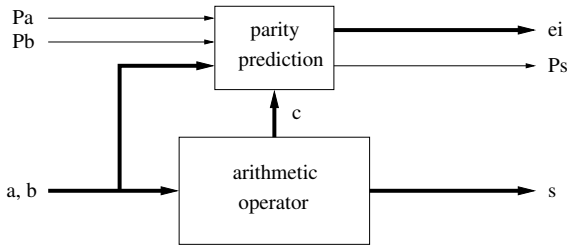


FIG. 1 – Schéma générale de la prédiction de parité.

La duplication est basée sur la réplication du circuit pour la détection des différences entre les résultats [9]. La sortie du deuxième circuit est complémentaire à celle du premier pour permettre l'utilisation d'un contrôleur double rail, envisageant aussi la détection des erreurs internes au contrôleur. Le surcoût matériel est plus que 100% mais la simplicité d'implémentation est un avantage de cette méthode. La Fig. 2 montre le schéma général de la duplication dans un circuit arithmétique.

La prédiction de parité et la duplication sont des méthodes auto-contrôlables, lesquelles signaleront l'occurrence d'une faute et déclencheront des actions de prise en compte de l'erreur, ayant un impact supplémentaire sur la performance.

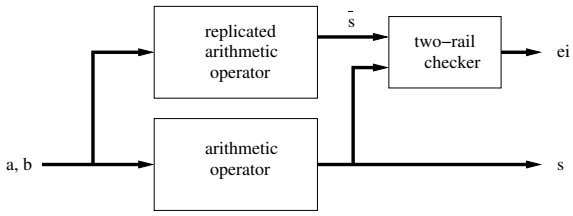


FIG. 2 – Schéma générale de la duplication.

La troisième méthode implémentée est la redondance modulaire triple (TMR), qui consiste en l'utilisation de trois circuits identiques pour l'exécution de l'opération et la détermination du résultat par l'application de la fonction majorité sur les trois résultats [10]. Le schéma général de la TMR est illustré à la Fig. 3. Le problème avec la TMR est le surcoût matériel de plus de 200%, mais en étant une méthode de tolérance aux fautes basée sur le masquage d'erreurs, la TMR permet une opération continue du système en présence d'une faute, sans impact sur la performance (seulement le retard de l'arbitre). La fiabilité de la méthode est fortement dépendente de la fiabilité de l'arbitre.

Les méthodes implémentées de tolérance aux fautes et d'auto-contrôle sont bien adaptées aux circuits d'applications critiques mais les surcoûts dérivés sont très élevés pour les circuits normaux et leur effectivité dans un environnement des fautes multiples n'est pas garanti. En considérant ces limitations, plusieurs études ont proposé l'application partielle de ces méthodes, comme une duplication des cellules plus importantes, ou la TMR d'une partie restreinte du circuit, mais aussi l'application des méthodes de prévention de fautes

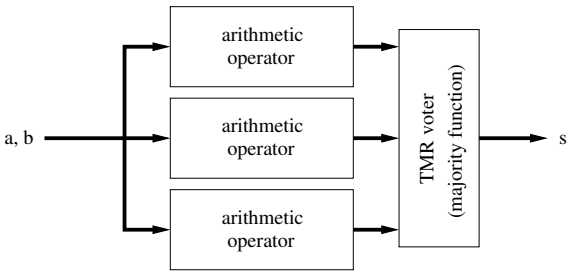


FIG. 3 – Schéma général de la redondance modulaire triple.

(durcissement des composants) [11, 12, 13, 14]. Ces méthodes sont basées sur une duplication des cellules critiques ou la redondance locale, avec l'utilisation des transistors plus robustes, moins sensibles aux variations des courants transitoires.

Ces méthodes ont des surcoûts beaucoup plus souples dans les circuits, et ces surcoûts peuvent être adaptés aux contraintes de conception. Le problème principal de ces méthodologies est la complexité de l'analyse nécessaire pour déterminer les composants critiques et sa nature extrinsèque au flot de conception. L'utilisation d'une méthodologie d'analyse de fiabilité intégrée dans le flot de conception permettrait la prise en compte en amont des effets des méthodes de durcissement et le management automatique de la fiabilité par les outils, en utilisant des bibliothèques de composants avec différents niveaux de durcissement.

La bibliothèque de composants robustes est composée d'additionneurs du type *ripple carry*, *carry-select*, *carry lookahead* et *signed digit*, et un multiplieur du type *Booth*. Tous les circuits ont été implémentés en version auto-contrôlable et quelques circuits en version TMR. Les circuits ont été décrits en langage VHDL et ont été synthétisés en technologie 130-nm pour permettre l'étude de la topologie résultante. Les circuits générés par cette méthode de conception sont la base pour l'étude de fiabilité et d'évaluation des méthodes de prévention de fautes.

L'Analyse de la fiabilité

La fiabilité (R) est un attribut de la sûreté de fonctionnement et correspond à la probabilité de fonctionnement d'un composant ou système, pendant une durée déterminée. La fiabilité est associée au taux de défaillance (λ) d'un composant, une mesure obtenue par des essais avec un lot de ces composants, sous différentes conditions d'opérations, pour permettre la définition d'un modèle de fiabilité plus précis [15]. Le taux de défaillance des composants intégrés est caractérisé par la "courbe en baignoire", qui est en fait la composition de trois courbes différentes, représentant les taux de défaillance initial, aléatoire et du au vieillissement. Les composants intégrés ont un taux de défaillance défini en FIT (*failures in time*), qui correspond au nombre de défaillances dans une période de 10^9 heures. Une autre métrique du taux de défaillance est le temps moyen entre pannes ($1/\lambda$) ou MTBF (*mean-time-between-failures*). L'équation 3 montre la fiabilité comme une fonction du taux de défaillance.

$$R(t) = e^{-\lambda t} \tag{3}$$

La fiabilité étudiée dans le présent travail concerne la fiabilité du signal, qui est associée au taux de défaillance aléatoire. La fiabilité du signal correspond à la probabilité d'occur-

rence d'une valeur correcte à la sortie du circuit, en considérant une probabilité donnée d'occurrence de fautes. La probabilité d'un résultat correct en présence de faute est une fonction de la capacité de masquage de fautes du circuit, ce qui peut être d'origine logique, électrique ou temporelle. La présente étude concerne plus spécifiquement la modélisation de la fiabilité du signal due au masquage logique, dépendante de la topologie du circuit.

Pour l'étude de la fiabilité du signal associée aux fautes transitoires, un modèle de faute d'inversion est utilisé, où une cellule ou porte logique en panne génère une valeur logique complémentaire à la valeur correcte. Chaque cellule dans un circuit a une probabilité d'erreur, et la fiabilité du circuit doit être calculée comme une fonction de la fiabilité de ses cellules et de la structure logique du circuit. Ce calcul a une complexité exponentielle par rapport au nombre d'entrées et sorties du circuit et des solutions approximatives sont nécessaires.

Trois méthodes d'estimation de la fiabilité ont été implémentées dans le présent travail, la méthode d'analyse de fiabilité par le modèle des matrices de transfert probabilistes (PTM), le modèle probabiliste binomial (PBR) et le modèle de la probabilité du signal (SPR). Ces méthodes sont présentées dans les sections qui suivent.

Le modèle des matrices de transfert probabilistes

La première méthode a être implémentée est basée sur le modèle des matrices de transfert probabilistes (PTM). Ce choix est du à la possibilité d'application directe de cette méthode, qui utilise des opérations matricielles courantes et qui permet la modélisation individuelle des portes logiques. Dans la proposition originelle de la méthode il n'y avait pas d'interface avec les outils de synthèse et nous voudrions exploiter cette possibilité dans l'implémentation courante.

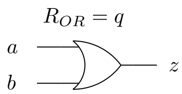
La PTM a été introduite dans les travaux de Patel et Krishnaswamy [16, 17, 18]. L'idée principale du modèle est de déterminer une matrice de probabilité d'occurrence d'une sortie par rapport aux probabilités des entrées. Cette matrice de corrélation entre entrées et sorties est appelée PTM. Elle est déterminée par la composition de la fiabilité des portes du circuit et sa topologie. La PTM peut être utilisée pour le calcul de la fiabilité du signal du circuit, en prenant en compte l'occurrence de fautes multiples.

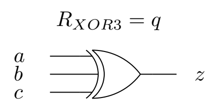
La PTM d'un circuit est une matrice de dimension $2^n \times 2^m$, où n est le nombre d'entrées et m est le nombre de sorties. La PTM d'une porte logique peut être définie à partir de sa table de vérité, aussi comme sa matrice de transfert idéale (ITM), qui correspond au modèle de fonctionnement sans faute de la porte. La figure 4 présente la PTM et l'ITM de deux portes logiques.

Comme montré dans la figure, les matrices représentent la probabilité d'occurrence d'entrées et sorties, et dans le cas de l'ITM, la probabilité est toujours 100% ou 0%. Les variables q et $1-q$ représentent respectivement la fiabilité et la probabilité d'erreur de la porte.

La détermination de la PTM d'un circuit est effectuée par la combinaison des PTMs des portes et de l'ITM des connexions. Le circuit doit être organisé par niveaux et la PTM d'un niveau est calculé par le tenseur des PTMs des composants du niveau. En multipliant les PTMs des différents niveaux, la PTM du circuit est déterminé. Les figures 5 et 6 montrent le calcul de la PTM d'un circuit simple, où $\bar{q} = 1 - q$.

Après le calcul de la PTM du circuit, sa fiabilité peut être déterminée à l'aide de l'ITM du circuit, en calculant la somme des probabilités de la PTM qui correspondent aux positions de probabilité **1** dans l'ITM, comme exprimé en (4), où I est l'ITM, $p(j|i)$ est la





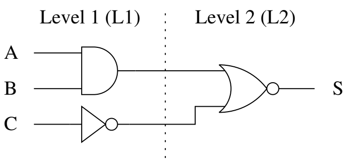
$$ITM_{OR} = \begin{array}{c|cc} & 0 & 1 \\ \hline 00 & 1 & 0 \\ 01 & 0 & 1 \\ 10 & 0 & 1 \\ 11 & 0 & 1 \end{array} \qquad PTM_{OR} = \begin{array}{c|cc} & 0 & 1 \\ \hline 00 & q & 1-q \\ 01 & 1-q & q \\ 10 & 1-q & q \\ 11 & 1-q & q \end{array}$$

(rows: *inputs*; columns: *outputs*)

$$ITM_{XOR3} = \begin{array}{c|cc} & 0 & 1 \\ \hline 000 & 1 & 0 \\ 001 & 0 & 1 \\ 010 & 0 & 1 \\ 011 & 1 & 0 \\ 100 & 0 & 1 \\ 101 & 1 & 0 \\ 110 & 1 & 0 \\ 111 & 0 & 1 \end{array} \qquad PTM_{XOR3} = \begin{array}{c|cc} & 0 & 1 \\ \hline 000 & q & 1-q \\ 001 & 1-q & q \\ 010 & 1-q & q \\ 011 & q & 1-q \\ 100 & 1-q & q \\ 101 & q & 1-q \\ 110 & q & 1-q \\ 111 & 1-q & q \end{array}$$

(rows: *inputs*; columns: *outputs*)

Fig. 4 – Exemple de modélisation PTM et ITM des portes logiques.



$$\text{PTM}_{\text{NOT}} = \begin{bmatrix} \bar{q} & q \\ q & \bar{q} \end{bmatrix} \qquad \text{PTM}_{\text{AND}} = \begin{bmatrix} q & \bar{q} \\ q & \bar{q} \\ q & \bar{q} \\ \bar{q} & q \end{bmatrix} \qquad \text{PTM}_{\text{NOR}} = \begin{bmatrix} \bar{q} & q \\ q & \bar{q} \\ q & \bar{q} \\ q & \bar{q} \end{bmatrix}$$

$$\text{PTM}_{\text{L1}} = \text{PTM}_{\text{AND}} \otimes \text{PTM}_{\text{NOT}} = \begin{bmatrix} q \cdot \begin{bmatrix} \bar{q} & q \\ q & \bar{q} \end{bmatrix} & \bar{q} \cdot \begin{bmatrix} \bar{q} & q \\ q & \bar{q} \end{bmatrix} \\ q \cdot \begin{bmatrix} \bar{q} & q \\ q & \bar{q} \end{bmatrix} & \bar{q} \cdot \begin{bmatrix} \bar{q} & q \\ q & \bar{q} \end{bmatrix} \\ q \cdot \begin{bmatrix} \bar{q} & q \\ q & \bar{q} \end{bmatrix} & \bar{q} \cdot \begin{bmatrix} \bar{q} & q \\ q & \bar{q} \end{bmatrix} \\ \bar{q} \cdot \begin{bmatrix} \bar{q} & q \\ q & \bar{q} \end{bmatrix} & q \cdot \begin{bmatrix} \bar{q} & q \\ q & \bar{q} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \bar{q}q & q^2 & \bar{q}^2 & \bar{q}q \\ q^2 & \bar{q}q & \bar{q}q & \bar{q}^2 \\ \bar{q}q & q^2 & \bar{q}^2 & \bar{q}q \\ q^2 & \bar{q}q & \bar{q}q & \bar{q}^2 \\ \bar{q}q & q^2 & \bar{q}^2 & \bar{q}q \\ q^2 & \bar{q}q & \bar{q}q & \bar{q}^2 \\ \bar{q}^2 & \bar{q}q & \bar{q}q & q^2 \\ \bar{q}q & \bar{q}^2 & q^2 & \bar{q}q \end{bmatrix}$$

Fig. 5 – Calcul de la PTM d'un niveau de circuit.

$$\text{PTM}_{\text{CIR}} = \text{PTM}_{\text{L1}} \cdot \text{PTM}_{\text{NOR}} = \begin{bmatrix} 2\bar{q}^2q + \bar{q}q^2 + q^3 & \bar{q}^2q + 2\bar{q}q^2 + \bar{q}^3 \\ \bar{q}^2q + 3\bar{q}q^2 & 2\bar{q}^2q + \bar{q}^3 + q^3 \\ 2\bar{q}^2q + \bar{q}q^2 + q^3 & \bar{q}^2q + 2\bar{q}q^2 + \bar{q}^3 \\ \bar{q}^2q + 3\bar{q}q^2 & 2\bar{q}^2q + \bar{q}^3 + q^3 \\ 2\bar{q}^2q + \bar{q}q^2 + q^3 & \bar{q}^2q + 2\bar{q}q^2 + \bar{q}^3 \\ \bar{q}^2q + 3\bar{q}q^2 & 2\bar{q}^2q + \bar{q}^3 + q^3 \\ 2\bar{q}q^2 + \bar{q}^3 + q^3 & 3\bar{q}^2q + \bar{q}q^2 \\ 2\bar{q}^2q + \bar{q}q^2 + q^3 & \bar{q}^2q + 2\bar{q}q^2 + \bar{q}^3 \end{bmatrix}$$

Fig. 6 – Détermination de la PTM du circuit.

probabilité d'occurrence d'une sortie j étant donnée l'entrée i. Cette expression est valable pour le cas où toutes les entrées ont la même probabilité d'occurrence

$$R(e) = 1/2^n \sum_{I(i,j)=1} p(j|i) \tag{4}$$

La figure 7 montre une analyse de fiabilité possible avec la méthode PTM, où la fiabilité du circuit est répresentée comme une fonction de la fiabilité de ses composants (q).

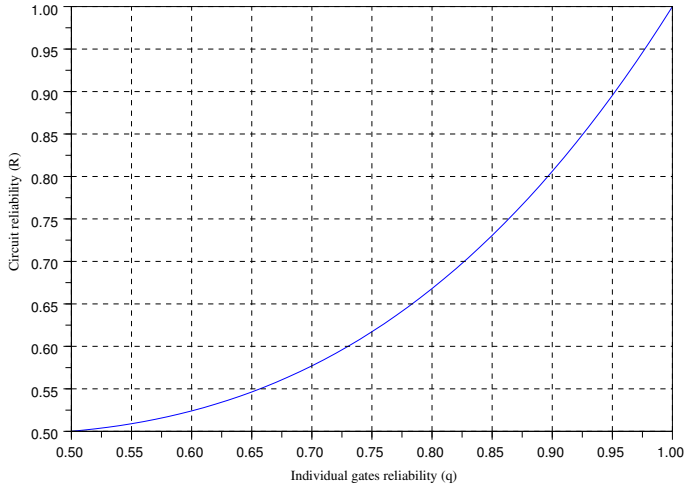


FIG. 7 – Courbe de fiabilité du circuit par rapport à la fiabilité de ses portes.

Malgré la simplicité associée a la méthode PTM, son utilisation est restreinte aux circuits de petite taille, à cause de sa complexité exponentielle par rapport au nombre d'entrées et sorties. La méthode fourni un résultat précis. Des outils nécessaires pour l'application de la PTM aux circuits issus de synthèse logique ont été implémentés et les résultats sont fournis dans le présent travail.

Le modèle probabiliste binomial

La méthode du modèle probabiliste binomial (PBR) est basée sur une analyse de la probabilité d'occurrence de fautes et leur masquage par rapport au taux de défaillance des cellules du circuit et leur nombre [19].

Si nous considérons que les portes d'un circuit ont une certaine probabilité de tomber en panne, la fiabilité R de ce circuit, vue comme une boîte noire, peut être déterminée comme en (5), où $p(\vec{y} = correct|\vec{x}_i)$ représente la probabilité d'une sortie correcte étant donnée un vecteur d'entrée $\vec{x}_i$ et sa probabilité d'occurrence $p(\vec{x}_i)$.

$$R = \sum_{i=0}^{2^m-1} p(\vec{y} = correct|\vec{x}_i)p(\vec{x}_i) \tag{5}$$

Pour modéliser la contribution de chaque porte dans la fiabilité totale, la probabilité de toutes les combinaisons d'entrées et vecteurs de fautes doit être déterminée. Soit Γ l'ensemble de toutes les portes du circuit (G portes), l'ensemble $\phi \subseteq \Gamma$ des portes qui tombent en panne et l'ensemble $\gamma \subseteq \Gamma$ des portes qui fonctionnent normalement. Dans ce cas, l'expression (6) modèle la fiabilité du circuit par rapport aux vecteurs d'entrées $\vec{x}_i$ et

de fautes $\vec{f_j}$. Le vecteur de fautes représente les portes en panne par une valeur **1** dans la position correspondante. La figure 8 illustre le vecteur de fautes.

$$R = \sum_{j=0}^{2^G-1} \sum_{i=0}^{2^m-1} p(\vec{y} = correct|\vec{x}_i, \vec{f_j}) p(\vec{x}_i) p(\vec{f_j}) \tag{6}$$

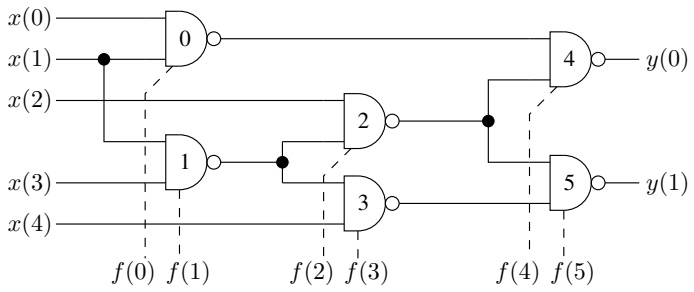


FIG. 8 – Circuit logique combinatoire générique.

L'expression générale de la fiabilité en (7) modèle le masquage de fautes pour les portes avec une fiabilité différente entre elles.

$$R = \sum_{j=0}^{2^G-1} \prod_{lc\in\gamma} q_{lc} \prod_{lf\in\phi} (1 - q_{lf}) \sum_{i=0}^{2^m-1} p(\vec{x}_i) \left(\overline{\vec{y}(\vec{x}_i, \vec{f_0}) \oplus \vec{y}(\vec{x}_i, \vec{f_j})} \right) \tag{7}$$

En considérant une probabilité égale des vecteurs d'entrées et les portes avec les mêmes fiabilités, l'expression (7) peut être ré-écrite comme en (8), en suivant un modèle probabiliste binomial, où k correspond au nombre des fautes simultanées.

$$R = \frac{1}{2^n} \sum_{k=0}^{G-1} (1-q)^k q^{G-k} \sum_{j\in\sigma} \sum_{i=0}^{2^i-1} \left(\overline{\vec{y}(\vec{x}_i, \vec{f_0}) \oplus \vec{y}(\vec{x}_i, \vec{f_j})} \right) \tag{8}$$

L'expression est dépendante du nombre de masquages en présence de fautes, ce qui est représenté par la fonction XOR. Le nombre de masquage doit être déterminé par simulation ou émulation, ce qui a été implémenté pendant le présent travail. Les outils développés déterminent le taux de masquage (coefficients c_k) des circuits cibles.

Le tableau 1 montre le taux de défaillance de 4 additionneurs différents de 4-bit, représenté par rapport au taux de défaillance de ses cellules composantes, obtenu avec la méthode PBR et l'équation (3). Les additionneurs sont des types *ripple-carry*, *carry-select*, *carry-lookahead* et *signed digit*. Les résultats obtenus avec la PRB sont exacts, et prennent en compte l'occurrence de fautes multiples simultanées.

TAB. 1 – MTBF des additionneurs 4-bit.

Adder type	Individual cell MTBF$_i$(hours)			
	10^{12}	10^9	10^6	10^3
RCA	$1.09 \cdot 10^{11}$	$1.10 \cdot 10^8$	$1.08 \cdot 10^5$	109
CSA	$1.03 \cdot 10^{11}$	$1.07 \cdot 10^8$	$1.06 \cdot 10^5$	106
CLA	$0.68 \cdot 10^{11}$	$0.66 \cdot 10^8$	$0.71 \cdot 10^5$	70
SD	$0.20 \cdot 10^{11}$	$0.18 \cdot 10^8$	$0.26 \cdot 10^5$	21

Étant donnée la complexité temporelle de la PBR, quelques approximations peuvent être faites pour permettre l'analyse des circuits plus importants. L'effet cumulatif des fautes dans la fiabilité du circuit permet l'application progressive de la méthode, par rapport au nombre des fautes simultanées considéré. La probabilité d'occurrence de fautes multiples est une fonction de la fiabilité et du nombre des cellules dans le circuit, régi par l'expression en (9).

$$P(k,G) = (C_k^G) q^{G-k} (1-q)^k \tag{9}$$

Si nous considérons que la fiabilité q des cellules est connue en avance, les coefficients c_k nécessaires pour un résultat avec une tolérance donnée peuvent être déterminés à travers cette expression. La Fig. 9 montre $P(k,G)$ pour un circuit composé de 100 cellules logiques, où la fiabilité de chaque cellule est $q = 0.995$. Dans ce cas, pour déterminer la fiabilité avec un erreur de moins de 1% il suffit d'utiliser les coefficients c_1 à c_4, i.e., le coefficients de masquage de fautes pour un nombre de fautes simultanées entre 1 et 4.

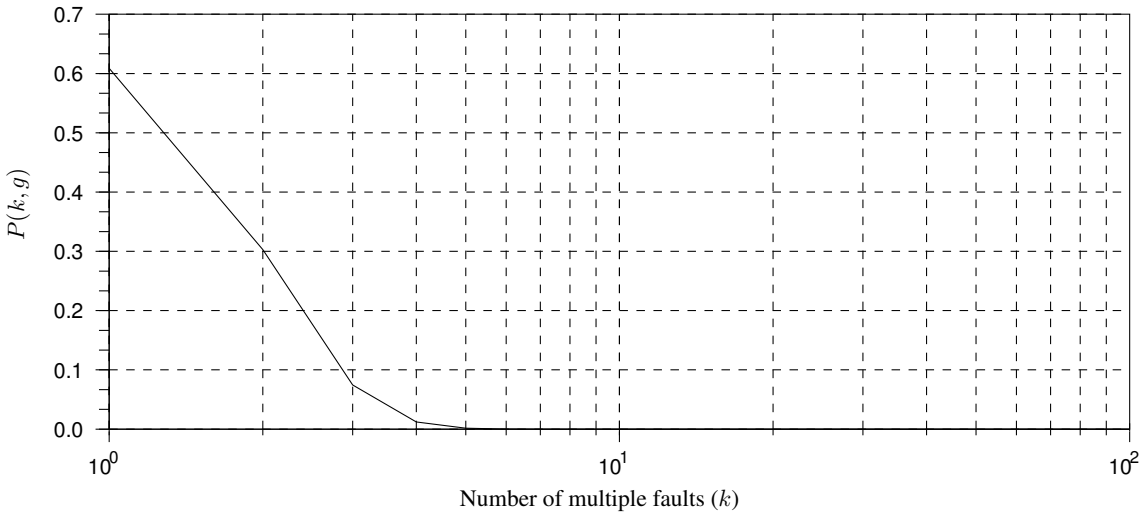


FIG. 9 – Probabilité d'occurrence de fautes dans un circuit avec 100 cellules ($q = 0,995$).

Une deuxième approximation peut être appliquée, en remplaçant la simulation exhaustive par une simulation pseudo-aléatoire, avec une durée déterminée. Dans ce cas, l'expression de fiabilité doit être ré-écrite comme en (10), où N_x est le nombre d'entrées appliquées et N_f le nombre de vecteurs de fautes du type $\vec{f}_k$.

$$R = \sum_{k=0}^{N_k-1} C_k^G (1-q)^k q^{G-k} \frac{\sum_{j=0}^{N_f-1} \sum_{i=0}^{N_x-1} \left(\overline{\vec{y}(\vec{x}_i, \vec{f}_0) \oplus \vec{y}(\vec{x}_i, \vec{f}_j)} \right)}{N_f N_x} \tag{10}$$

L'analyse de la fiabilité à travers le modèle probabiliste binomial nécessite une simulation fonctionnelle pour la détermination des coefficients de l'équation analytique. Pour permettre une implémentation générique et automatique du *testbench* des circuits logiques, l'outil d'analyse génère une version modifiée du circuit cible, en ajoutant des points d'injection de faute dans les sorties des cellules du circuit. Le testbench consiste à comparer les sorties des circuits sans et avec fautes pour déterminer le masquage des résultats et générer l'équation analytique de fiabilité.

En s'appuyant sur la capacité reconfigurable des FPGA, et en considérant la taille des circuits cibles, la simulation fonctionnelle peut être remplacée par une émulation des circuits, avec un gain intéressant en terme de performance. Cette possibilité a été exploitée dans le présent travail. La figure 10 montre le schéma implémenté pour l'émulation des circuits en présence de fautes.

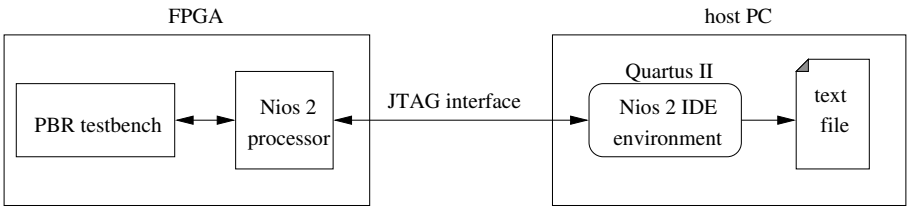


FIG. 10 – Schéma d'émulation pour l'analyse PBR.

La méthode PBR permet l'obtention de résultats très précis dans le calcul de la fiabilité des circuits, étant limitée par le temps disponible pour la simulation/émulation des fautes.

Le modèle de la probabilité du signal

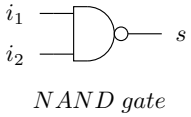
L'analyse de la fiabilité par la probabilité du signal est basée sur l'observabilité des fautes à la sortie du circuit, ce qui correspond à la probabilité de masquage de ces fautes par les entrées ou par l'effet cumulatif des fautes concurrentes [20, 21].

La probabilité de masquage d'une faute à l'entrée d'une porte logique est une fonction de la probabilité d'occurrence de quelques combinaisons des autres entrées. Dans ce cas, la détermination de la probabilité des signaux dans un circuit en présence de fautes permettrait la détermination de sa fiabilité à travers une comparaison avec la probabilité des signaux du circuit sans faute. Au lieu de faire une analyse de la probabilité des signaux pour chaque faute possible, une analyse directe de la fiabilité des signaux permettrait la prise en compte de l'effet cumulatif des fautes multiples. En considérant la fiabilité d'une cellule logique, la fiabilité du signal à la sortie est une fonction de la fiabilité de la cellule et de la fiabilité des signaux d'entrée. Alors, la propagation de la fiabilité de chaque signal à travers le circuit logique permet le calcul de la fiabilité des signaux à la sortie du circuit, en prenant en compte sa topologie. La fiabilité conjointe des signaux à la sortie du circuit correspond à la fiabilité du signal pour ce circuit.

Considérons une porte logique quelconque, dont la fiabilité de fonctionnement peut être représentée par sa matrice de transfert probabiliste (PTM). La fonction logique est représentée par sa matrice de transfert idéale (ITM). L'ITM et la PTM expriment la probabilité d'occurrence des paires entrées/sorties dans une cellule logique [16, 17]. La Fig. 11 montre la PTM et l'ITM d'une porte NAND, avec une fiabilité q, et une probabilité d'erreur $(1-q)$.

Considérons la probabilité d'un signal binaire comme étant représenté par 4 états possibles, à savoir, 0 correct, 0 incorrect, 1 correct et 1 incorrect. En organisant ces probabilités dans un format matriciel 2×2, la probabilité d'occurrence de chaque état d'un signal peut être écrite comme $signal_0$ (0 correct), $signal_1$ (1 incorrect), $signal_2$ (0 incorrect) et $signal_3$ (1 correct), et la probabilité du signal dans cette forme matriciel peut être écrite comme $SIGNAL_4$. La Fig. 12 montre la représentation matriciel de la probabilité du signal.

La probabilité conjointe des signaux peut être déterminée par le produit tensoriel (ou produit de Kronecker) des signaux. La Fig. 13 montre le calcul de la probabilité conjointe



NAND gate

i_1	i_2	s
0	0	1
0	1	1
1	0	1
1	1	0

truth table

$$ITM_{NAND} = \begin{array}{c} \\ 00 \\ 01 \\ 10 \\ 11 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \end{array} \quad \begin{array}{l} \\ inputs \\ \\ \\ \\ \end{array} \quad \text{output}$$

$$PTM_{NAND} = \begin{array}{c} \\ 00 \\ 01 \\ 10 \\ 11 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{bmatrix} 1-q & q \\ 1-q & q \\ 1-q & q \\ q & 1-q \end{bmatrix} \end{array}$$

FIG. 11 – L'ITM et PTM d'une porte NAND.

$$SIGNAL_4 = \begin{bmatrix} signal_0 & signal_1 \\ signal_2 & signal_3 \end{bmatrix}$$

$$P_{2\times 2}(signal) = \begin{bmatrix} P(signal = correct\ 0) & P(signal = incorrect\ 1) \\ P(signal = incorrect\ 0) & P(signal = correct\ 1) \end{bmatrix}$$

FIG. 12 – Représentation matricielle de la probabilité du signal.

de deux signaux, a et b. Si a et b sont les signaux à l'entrée d'une porte logique, la matrice I représente toutes les probabilités à l'entrée de cette porte.

$$I_g = \begin{bmatrix} a_0 & a_1 \\ a_2 & a_3 \end{bmatrix} \otimes \begin{bmatrix} b_0 & b_1 \\ b_2 & b_3 \end{bmatrix} = \begin{bmatrix} a_0b_0 & a_0b_1 & a_1b_0 & a_1b_1 \\ a_0b_2 & a_0b_3 & a_1b_2 & a_1b_3 \\ a_2b_0 & a_2b_1 & a_3b_0 & a_3b_1 \\ a_2b_2 & a_2b_3 & a_3b_2 & a_3b_3 \end{bmatrix}$$

FIG. 13 – Calcul de la probabilité conjointe de deux signaux.

La fiabilité du signal à la sortie d'une cellule logique est la composition de la fiabilité de la cellule avec la fiabilité des signaux d'entrée, ce qui peut être déterminé par la multiplication de la probabilité conjointe des entrées (I) par la PTM de la cellule. Alors, la probabilité de la sortie y de la porte g est donnée par $P(y) = I_g \otimes PTM_g$. La matrice $P(y)$ représente la probabilité d'occurrence des paires entrées/sortie de la porte g et en utilisant l'ITM de la porte, la probabilité Y_4 du signal peut être déterminée, à travers les expressions 11 à 14, où $0, r$ ou $1, r$ correspondent à la $colonne, ligne$ de l'ITM de la porte.

$$y_0 = \sum_{r=0}^{2^i-1} P(y)_{[0,r]} ITM_{[0,r]} \tag{11}$$

$$y_1 = \sum_{r=0}^{2^i-1} P(y)_{[1,r]} (1 - ITM_{[1,r]}) \tag{12}$$

$$y_2 = \sum_{r=0}^{2^i-1} P(y)_{[0,r]} (1 - ITM_{[0,r]}) \tag{13}$$

$$y_3 = \sum_{r=0}^{2^i-1} P(y)_{[1,r]} ITM_{[1,r]} \tag{14}$$

La Fig. 14 montre la détermination de la probabilité du signal à la sortie s d'une porte AND avec une fiabilité $q_{AND} = 0.9$. Le procédé de calcul de la probabilité du signal à la sortie est appelé **propagation** de la probabilité.

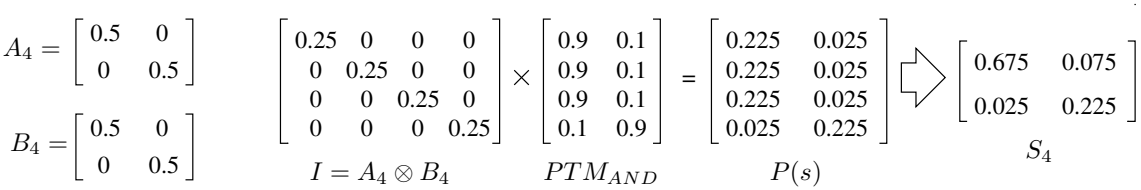


FIG. 14 – Propagation des probabilités des signaux d'entrée.

La fiabilité d'un signal peut être déterminée directement par $R(signal) = signal_0 + signal_3$ et la fiabilité du signal à la sortie d'un circuit logique correspond à la fiabilité du circuit. Pour un circuit avec plusieurs sorties, la fiabilité du circuit peut être déterminée par la multiplication de la fiabilité des sorties, comme montré à l'expression (15), où R_j correspond à la fiabilité de la sortie j.

$$R_{circuit} = \prod_{j=0}^{m-1} R_j \tag{15}$$

La Fig. 15 montre la détermination de la fiabilité d'un circuit à travers la propagation de la probabilité des signaux.

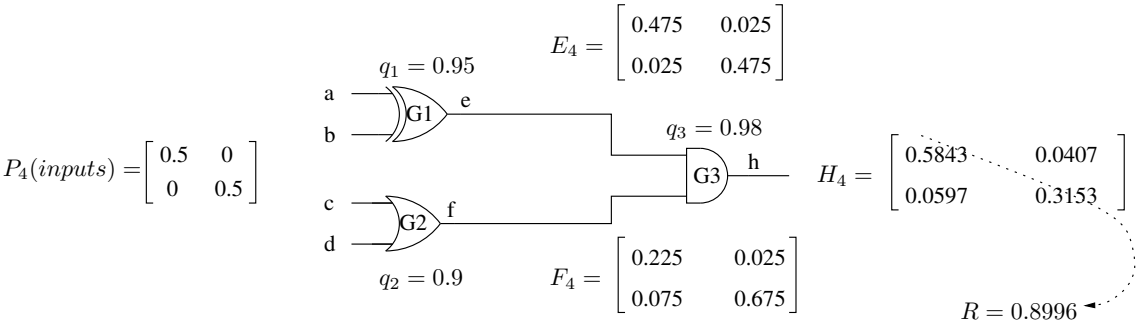


FIG. 15 – Détermination de la fiabilité d'un circuit à travers la propagation des probabilités des signaux.

Le calcul de la probabilité des signaux dans un circuit logique est un problème de haute complexité, du type *#P-complete* [22, 23], dans les circuits avec des signaux *reconvergents*. Dans ce cas, le calcul de la probabilité des signaux à travers une seule propagation (*single-pass*) ne conduit pas à un résultat exact. Plusieurs heuristiques peuvent être imaginées pour corriger ou minimiser cet erreur et deux heuristiques ont été implémentées dans le présent travail, la propagation *multi-pass* et l'algorithme des moyennes pondérées dynamique (DWAA) [23].

L'algorithme multi-pass correspond à l'implémentation successive de la propagation de la probabilité des signaux en prenant en compte à chaque itération un seul état de la probabilité des signaux *reconvergents*. A la fin de chaque propagation, la fiabilité du circuit est calculée et accumulée aux valeurs précédentes, comme montre l'équation (16), où k est le nombre des signaux *reconvergents*, et s l'état de chaque signal.

$$R = \sum_{j=1}^{k} \sum_{s=0}^{3} R(j * s) \tag{16}$$

La complexité de calcul pour l'algorithme multi-pass est exponentielle par rapport au nombre des signaux *reconvergents* du circuit, mais des échanges entre précision et temps de calcul sont possibles. La possibilité d'un choix explicite des signaux *reconvergents* à prendre en compte dans le calcul de l'algorithme multi-pass permet le choix entre précision et temps de calcul.

L'algorithme des moyennes pondérées dynamique est une heuristique qui exécute une approximation successive des probabilités des signaux. Cette heuristique a une complexité linéaire par rapport au nombre des signaux *reconvergents* et le nombre des portes logiques dans le circuit.

L'algorithme DWAA boucle la propagation des probabilités en remplaçant à chaque itération la probabilité d'un état du signal *reconvergent* par 1 et 0, successivement, et en effectuant la correction de la probabilité des signaux dépendants par rapport aux déviations des probabilités. L'algorithme débute par le calcul des probabilités de tous le signaux selon l'algorithme de propagation simple (algorithme-0). Après, les probabilités des signaux *reconvergents* (f) sont mis successivement à la valeur **1** et **0**, et des nouvelles valeurs de probabilité sont calculées pour les signaux j, dépendants de f, d'après l'expression (17).

$$p(j|f) = p(j|f = 0)p(f = 0) + p(j|f = 1)p(f = 1) \tag{17}$$

La déviation de la probabilité par rapport au signal *reconvergent* est calculée selon l'expression (18), où $p(j, 0)$ est la probabilité calculée par l'algorithme-0. Les valeurs des déviations sont accumulées pour permettre la pondération des corrections. L'expression (19) présente ce calcul, où $t-1$ représente les nombre de signaux *reconvergents* déjà calculés.

$$w_f(j) = |p(j|f) - p(j, 0)| \tag{18}$$

$$w_s(j, t-1) = \sum_{k=1}^{t-1} w_k(j) \tag{19}$$

Finalement, la correction des probabilités est calculée selon l'expression (20).

$$p(j, t) = \frac{p(j, t-1)w_s(j, t-1) + p(j|f)w_f(j)}{w_s(j, t-1) + w_f(j)} \tag{20}$$

A chaque itération de l'algorithme, les nouvelles probabilités sont calculées, et aussi la fiabilité, qui est corrigée par des expressions similaires à celles présentées. L'algorithme DWAA demande un ordonnancement des signaux par dépendance.

Résultats

Les outils d'analyse de fiabilité selon les modèles proposés dans le présent travail ont été implémentés en langage C. Ces outils permettent l'analyse des circuits décrits en langage VHDL ou Verilog, générés par les outils de synthèse, et calculent leur fiabilité du signal, en générant des fichiers de données au format Scilab, pour permettre la visualisation et le traitement des résultats. Ces mêmes outils peuvent être modifiés pour permettre la

communication de la valeur de fiabilité directement à l'outil de synthèse pour une synthèse dirigée vers la fiabilité.

Les outils d'analyses ont été utilisés dans l'évaluation des circuits tolérants aux pannes et l'évaluation des méthodes de prévention de fautes. Plusieurs analyses ont étés possibles, comme le rapport entre la fiabilité des cellules d'un circuit et la fiabilité du circuit. La Fig. 16 montre ce rapport pour les circuits additionneurs classiques de la bibliothèque.

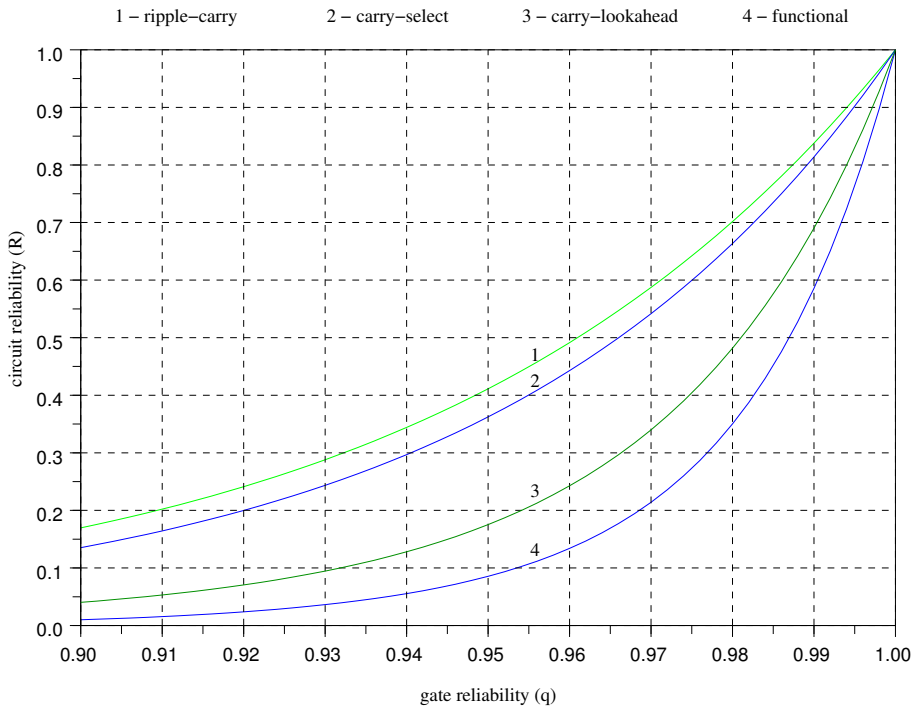


FIG. 16 – Fiabilité des additionneurs classiques 8-bits.

Une autre représentation possible pour les résultats obtenus est la comparaison du temps moyen entre fautes pour les différents circuits, ce qui peut être vu à la Fig.17.

D'autres types d'analyses possibles sont la détermination du taux d'erreurs, la détermination des cellules plus sensibles dans un circuit, l'amélioration de la fiabilité avec le durcissement des cellules spécifiques et le comportement des circuits tolérants aux fautes et auto-contrôlables en présence de fautes multiples.

La Fig.18 montre le comportement de la fiabilité fonctionnelle et du signal dans un additionneur du type *carry-lookahead* auto-contrôlable, par rapport à la fiabilité de ses cellules. La fiabilité fonctionnelle est une mesure de la probabilité de détection de fautes du circuit.

Même si la fiabilité fonctionnelle de l'additionneur est plus importante que la fiabilité du signal, les actions associées à la détection de fautes représentent une perte de performance pour le circuit. La Fig. 19 montre cette perte de performance qui peut être vue comme une réduction de la fréquence d'opération du circuit, ou une fréquence effective.

Le tableau 2 montre l'amélioration de la fiabilité de quelques circuits obtenue avec les méthodes de prévention de fautes. Dans ce cas, les cellules plus vulnérables des circuits ont été répliquées, et le tableau montre l'amélioration de la fiabilité (MTBF) et l'augmentation correspondante de la surface du circuit.

Tous ces types d'analyses ont été faits avec les modèles développés dans le présent

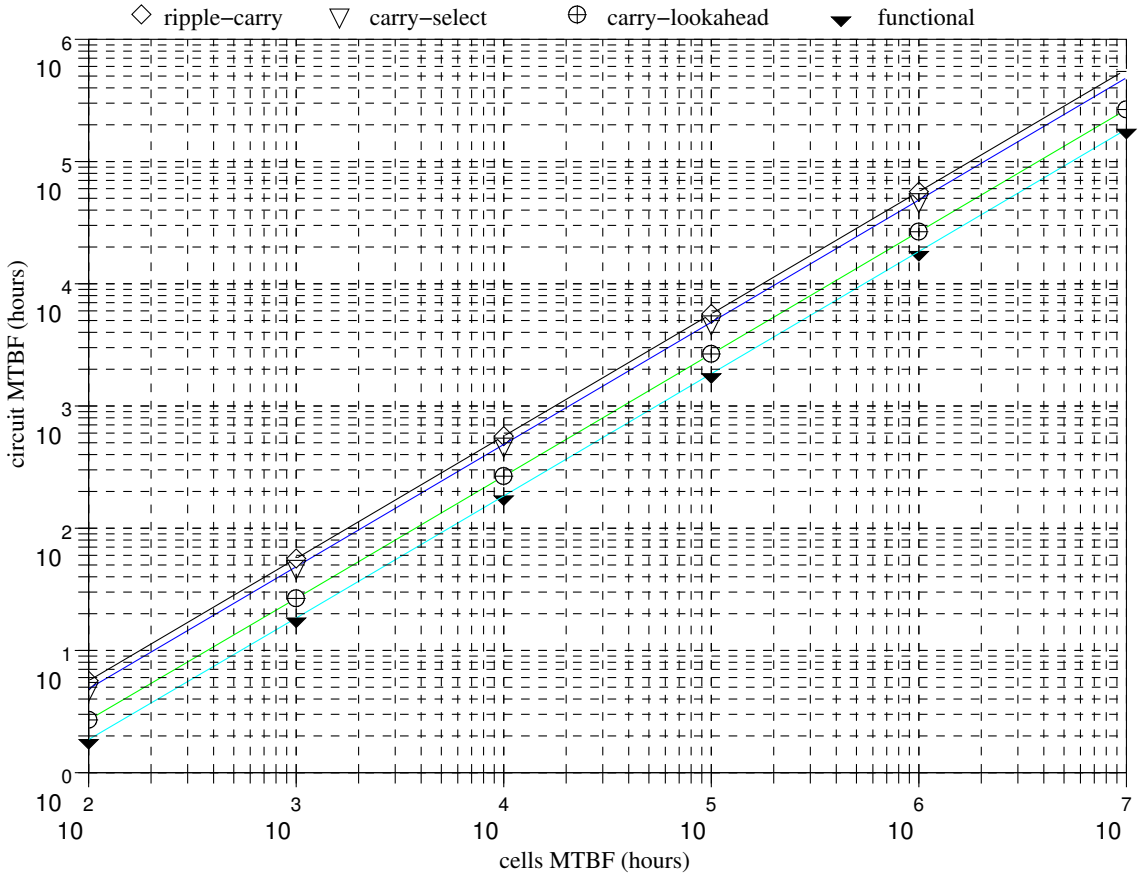


FIG. 17 – Temps moyen entre fautes pour les additionneurs classiques.

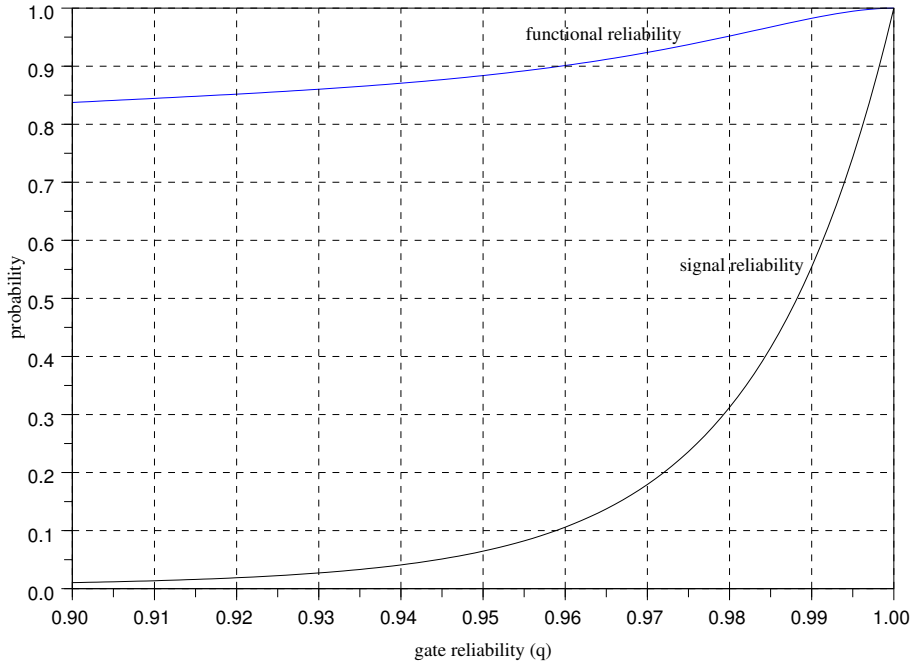


FIG. 18 – Fiabilité fonctionnel et du signal dans un additionneur du type *carry-lookahead* auto-contrôlable.

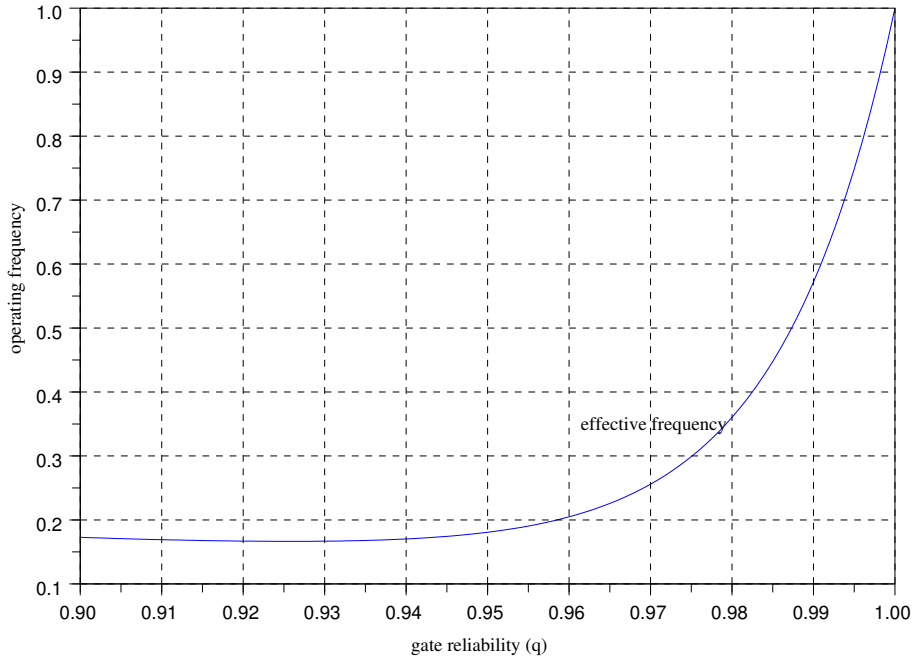


FIG. 19 – Fréquence effective de fonctionnement de l'additionneur auto-contrôlable.

TAB. 2 – Augmentation du MTBF et de la surface avec la réplication des cellules.

Circuit type	Cell replication approach	
	MTBF (%)	Area (%)
rca	67.5	61.1
csa	127	47.5
cla	143	66.0
fca	94.0	48.8
fcm	195	64.3
rca tmr	722	9.93

travail. Les résultats complets peuvent être vus dans le chapitre 4 de ce rapport.

Conclusions

L'estimation de la fiabilité des circuits logiques pendant les étapes initiales de projet est un pas fondamental pour la conception des circuits nanométriques. La réduction prévue pour la fiabilité des composants intégrés obligera les concepteurs à l'implémentation des méthodes de durcissement des circuits, mais avec un surcoût très limité. Pour permettre l'application de ces méthodes d'une façon adaptée aux contraintes de projet, l'estimation de la fiabilité doit être intégrée dans le flot de conception. Plusieurs méthodes ont été proposées dans la littérature pour l'estimation de la fiabilité, mais étant donnée la complexité de l'analyse, chaque méthode a des limitations d'application, comme la restriction à une seule faute, la restriction à une seule sortie, la restriction à un seul chemin logique ou la restriction à un sous-ensemble des entrées.

Le présent travail a proposé deux méthodes d'estimation de la fiabilité "flexifles" dans le sens où elles permettent de jouer sur un compromis rapidité et précision. Cette flexibilité

peut être utilisée de façon complémentaire tout au long de la conception. Ces méthodes prennent en compte l'occurrence de fautes multiples et sont alors adéquates pour l'étude des circuits nanométriques, plus susceptibles à ce type d'événement.

Symbols and Abbreviations

q	Gate reliability
$1-q$, $\bar{q}$	Gate unreliability
M, $M(t)$	Maintainability
μ, $\mu(t)$	Repair rate
λ, $\lambda(t)$	Failure rate
R, $R(t)$	Reliability
Y	Yield
ADD	Algebraic decision diagram
ASIC	Application specific integrated circuit
BDD	Binary decision diagram
BN	Bayesian network
BOX	Buried oxide
CCC	Custom configurable computer
CAEN	Chemically assembled electronic nanotechnology
CED	Concurrent error detection
CEG	Circuit equivalent graph
CMOL	CMOS-nanowire-molecular structure
CMOS	Complementary metal-oxide-semiconductor
CMF	Common-mode failure
CMP	Chemical mechanical polishing
CNN	Cellular nonlinear network
CNT	Carbon nanotube
CPT	Conditional probability table
CWSP	Code word state preserving
DAG	Directed acyclic graph
DFS	Depth-first search
DFM	Design for manufacturability
DRAM	Dynamic random access memory
DSP	Digital signal processing
DTMC	Discrete-time Markov chain
DWAA	Dynamic weighted averaging algorithm
DWC	Duplication with comparison

ECC	Error-correcting code
EDA	Electronic design automation
EDIF	Electronic design interchange format
EPP	Error propagation probability
EVIS	Evidence pre-propagated importance sampling
FASER	Fast and accurate SER analysis
FET	Field-effect transistor
FIN	Fault injection network
FIR	Finite impulse response
FIT	Failures in time
FPGA	Field programmable gate array
FS	Fault-secureness
FSM	Finite-state machine
GA	Genetic algorithm
GEG	Gate equivalent graph
HHE	Hybrid Hall effect
IC	Integrated circuit
ITRS	International technology roadmap of semiconductors
ISCAS	International symposium of circuit and systems
ITM	Ideal transfer matrix
LUT	Look-up table
LIFE-DAG	Logic induced fault encoded DAG
MAC	Multiply-accumulate
MDW	Moving domain wall
MOS	Metal-oxide semiconductor
MOSFET	Metal-oxide semiconductor FET
MPU	Microprocessor unit
MRE	Magneto resistive element
MRRNS	Modulus replication residue number system
MTBF	Mean-time-between-failures
MTTR	Mean-time-to-repair
MTBDD	Multi-terminal BDD
MXML	Micro-architectural XML
NEMS	Nano-electromechanical system
NMR	N-modular redundancy
NW	Nanowire
OPC	Optical proximity correction
PA	Availability
PBR	Probabilistic binomial reliability model
PGM	Probabilistic gate model
PLA	Programmable logic array
PLS	Probabilistic logic sampling
PMC	Probabilistic model checking
PSM	Phase-shift masking
PTM	Probabilistic transfer matrix

QCA	Quantum cellular automata
RESO	Recomputing with shifted operands
RET	Resolution enhancement technique
RSFQ	Rapid single flux quantum
RTD	Resonant tunneling diode
RTT	Resonant tunneling transistor
SBD	Signed binary digit
SCSL	Selective clock skewed-logic
SER	Soft error rate
SETRA	Scalable, extensible tool for reliability analysis
SET_1	Single-electron transistor
SET_2	Single event transient
SEU	Single event upset
SIP	System-in-package
SOC	System-on-chip
SOI	Silicon-on-insulator
SPICE	Simulation program with integrated circuit emphasis
SPR	Signal probability reliability model
SRAM	Static random access memory
TMR	Triple modular redundancy
UTB	Ultra-thin body
VHDL	Very high speed integrated circuit hardware description language
VLSI	Very large scale integration
XML	Extensible markup language
WAA	Weighted averaging algorithm

Contents

Chapter 1

Introduction

1.1 Motivations

The continuous development of computers and electronic systems, perceived in the last decades, is a consequence of the evolution of integrated circuits (ICs) and its design and manufacturing processes. The basic principle behind this evolution is the reduction, or scaling, in the dimensions of the integrated structures. The ability to create smaller devices enables them to switch faster, makes them cheaper to manufacture and allows improvements in processing power. Fig. 1.1 shows the historical evolution of density in integrated circuits [24], allowed by technology scaling. The improvement in the number of transistors in an IC can be considered as an increase in circuits density, by considering that ICs area remains the same.

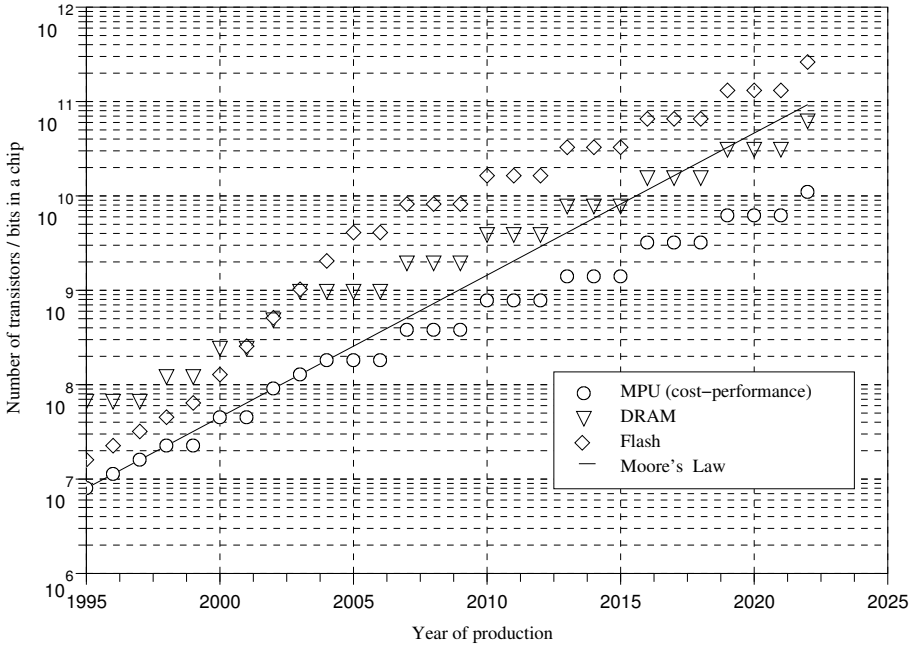


Figure 1.1: ICs density evolution.

The economical effect of transistors scaling is the reduction of the manufacturing cost of integrated circuits, considering that the area of the circuits can be reduced and the cost

is related with circuit's implementation area. Fig. 1.2 shows the historical evolution of the manufacturing cost of one million transistors.

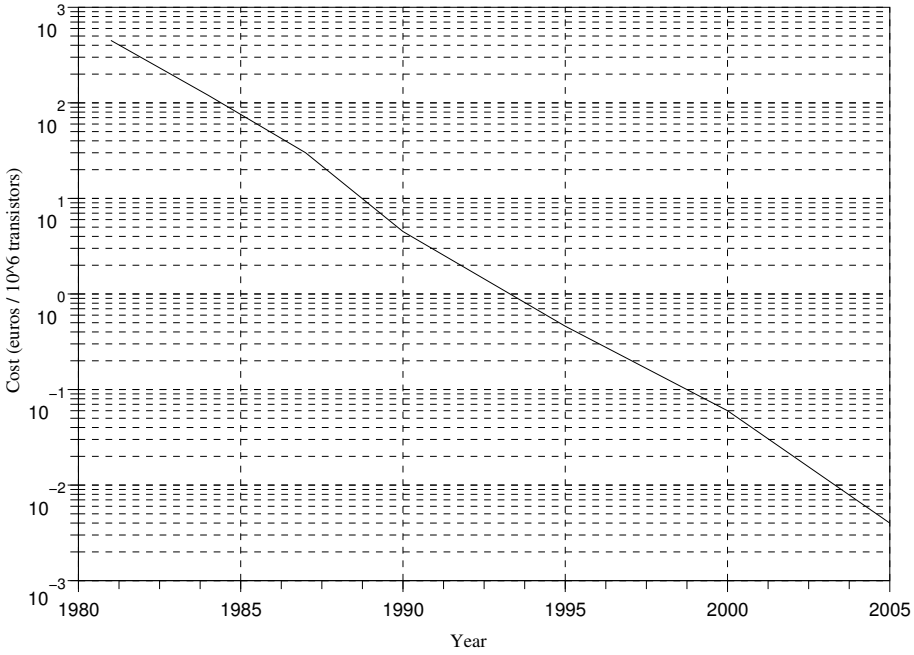


Figure 1.2: Historical evolution of integrated circuits manufacturing cost.

The ICs evolution is responsible for developments in all of the human activities. The tradeoffs between speed, area and power, allowed by the CMOS (complementary metal oxide semiconductor) IC design style, make it adaptable to the constraints of different systems and applications. In telecommunications, for example, low power circuits and large scale integration were fundamental for the establishment of the mobile telephony, and only IC technology scaling can cope with the growing processing and storage demands for this market.

As device-scaling enters the nanometer dimensions, words with the prefix *nano* are becoming mainstream, like nanoscale, nanotechnology and nanoelectronics. The word nanotechnology was introduced in 1974 by Norio Tanigushi [25], and nowadays is used to describe a large field of research that includes nanomaterials, nanoelectronics and nanobiotechnology. In a generic way, nanotechnology refers to the study/manipulation of structures/systems with less than 100-nm in at least one dimension of the space, and that have properties related with their nanometric size. That's the case in the nanoelectronics research field, where the size of the devices (nanodevices or nanoscale devices) brings many challenges to designers and researchers, due to different properties and behaviors associated with their operation at deep-submicrom dimensions.

The historical evolution pace of ICs development is seriously menaced, and in some areas it has already slowed down, since the traditional scaling process is reaching the limits of materials, processes and profitability [26]. As devices scale down into nanometric dimensions, design and manufacturing are becoming highly complex, some of the qualities and properties of the materials involved are no longer valid, and some of the manufacturing steps aren't as reliable as desired.

One of the main problems associated with nanoscale devices is the reduced yield expected from the future manufacturing processes [27, 28]. The variability in materials and

limited precision processes are translated into failures/imperfections in the circuits and wafers, leading to a reduction in the manufacturing yield. Another problem is the reliability of logic circuits related to intermittent and transient faults, that will play an increasing role in future technologies [29, 5, 30, 4], and may compromise the gains attained with device-scaling.

At this point, it's not possible to predict when the CMOS development will find its end, but some solutions and alternatives are being studied to guarantee a continuous evolution pace. Researchers are considering two general paths [31]: the first one is to extend CMOS scaling towards the 11-nm node predicted to arrive in 2022 [24], coping with performance, low power and standby power constraints; the second path is to develop new devices and architectures to replace or, most probably, complement the CMOS structures, aiming to maintain the evolution rates achieved in the last decades for performance, circuit density and power consumption. The areas of interest towards these two paths include emerging devices and materials, new architectures, new data representations and new computing models [32]. According to the ITRS association (International Technology Roadmap of Semiconductors), these two paths are defined as *more Moore* versus *more than Moore*.

Following the first path means to change considerably the way MOS transistors are designed, by using modified and new materials in a similar structure or by changing the structure itself. These changes in the traditional MOS transistor aim to improve switching speed, reduce static current leakage, limit short channel effects and to improve electrostatic control of the gate over the channel. These advanced CMOS structures are discussed in more detail in appendix A.1 [2], along with some definitions of the scaling process and a brief description of CMOS devices evolution and perspectives.

The introduction of new materials and process steps in the fabrication of CMOS devices also impacts negatively on the yield and reliability of these integrated circuits, since new fabrication methods, design tools and device models must be developed, and high precision manufacturing and test will be necessary.

The second path represents the use of breakthrough solutions as the introduction of new device types and new system architectures. The new devices being researched are based on carbon nanotubes (CNT), nanowires, single electron transistors (SETs), spin transistors and molecular devices, among others. These new devices are not considered for direct replacement of the CMOS technology but primarily as complementary structures to be integrated along with CMOS ones, to extend the capacities of the later. A detailed presentation of these emerging logic devices is available in appendix A.2 [2], where their characteristics and perspectives are summarized.

Along with these emerging logic devices, new circuit architectures have been proposed to explore alternative fabrication methodologies, data representation and computing models. These new architectures are mainly based on bottom-up assembly, on matrix topologies and non-volatile storage, in an FPGA-like organization targeted to a finer-grain level. These architectures promise low cost defect-tolerant fabrication and high device density and speed. Some examples of these new architectures and the concepts associated with bottom-up defect-tolerant approaches are presented in more detail in appendix A.3 [2].

Even if the proposed architectures are targeted to defect-tolerant approaches, the yield of these structures remain highly speculative and dependent on post-fabrication steps. On the side of the reliability of these emerging logic devices and architectures, their behavior remains unpredictable due to a lack of precise models, but considering parametric variations, reduced noise margins and the estimated circuit densities, a lower reliability must be expected.

Considering this evolution scenario, defect and fault-tolerant design approaches are unavoidable work-arounds in the development of nanoscale systems. These approaches are primarily based on hardware redundancy and can be applied to all of the abstraction levels, from spare transistors to spare processing blocks, from circuit replication to data coding methods. Despite the existence of many studies and proposals to deal with defects and faults in integrated circuits, as the semiconductor industry reaches the nanoscale era, many of the solutions are becoming application-specific [33] and the suitability of a given solution must be determined or adapted according to the application constraints.

As mentioned before, fault-tolerant approaches, e.g., modular redundancy and concurrent error detection (CED), are being considered to improve the reliability of nanoscale circuits. Modular redundancy has been traditionally targeted to mission critical systems, i.e., medical, spatial and military ones, where dependability measures are the main design objectives and the related overheads can be accepted. Concurrent error detection schemes are normally targeted to sequential logic parts of the circuits as memory blocks, register banks and latches, where the resulting overheads can be kept at a reduced level. The use of CED schemes in combinational logic is not as straightforward as in sequential logic but the resulting overheads are reduced compared to modular redundancy.

The use of these fault-tolerant approaches in mainstream applications is constrained by several parameters, i.e., power dissipation, battery autonomy, cost, among others, dependent on the target application. Given these limitations, the application of modular redundancy or CED schemes may not be the best option to improve reliability, and alternative approaches have been proposed, as the implementation of partial replication or the hardening of individual cells. These approaches allow a fine-grain improvement in the reliability figures, limiting their impact on area, propagation time and power consumption.

Given the expected reduction of the reliability and the referred range of possible solutions, an early estimation of circuit's reliability is a valuable feature in the design flow, allowing the implementation of a reliability-aware automated design process. The main obstacle to this workflow is the complexity associated with reliability analysis, since computing the exact values of reliability measures is intractable for practical circuits. Many reliability analysis approaches can be found in the literature but most of these are not compatible or adequate to design flow integration. Despite the promising results obtained in the referenced works, they are generally restricted to single-fault, single-path or single-output. Furthermore, the referred approaches are not focused to the same analysis.

Considering the presented context, this work is targeted to the development of methods and tools for reliability analysis of combinational logic circuits in nanometric technologies, i.e., in presence of multiple simultaneous faults. The objective being the integration of the proposed method or tool in an automated design flow, its desired characteristics are a straightforward (direct, simple) implementation and a parameterizable tradeoff between accuracy and processing time. The search for a straightforward method is related with the need for a method that could be integrated in the traditional multi-criteria synthesis process without an important impact on design cycle time.

Following these objectives, the current work developed two methods for reliability analysis, the first one based on a probabilistic binomial model of fault occurrence and the second one based on signal probability propagation of fault-prone logic signals. These methods have been used to evaluate some circuits representing the referred fault-tolerant approaches, allowing a better understanding of the concerned design space. The next section presents the organization of the current work.

1.2 Organization of the Thesis

The thesis is organized as follows:

Chapter 2 introduces the basic concepts related to reliability in logic circuits, like dependability, fault types and failure rate, allowing a better characterization of the scope of the work. The chapter presents the reasons for the improvement in the susceptibility of combinational logic circuits to transient faults, detailing the relation among circuit scaling, fault rates and the probability of occurrence of multiple simultaneous faults. A brief review of some fault-tolerant approaches is presented, concerning modular redundancy, self-checking circuits, partial fault tolerance and fault avoidance. A library of fault-tolerant arithmetic functions has been developed as case studies for the reliability analysis tools. This library is presented in appendix B.

Chapter 3 starts by presenting the state of the art in reliability analysis methods and tools. By examining the existent proposed solutions, advantages and drawbacks of each one are discussed. The probabilistic binomial model (PBR) is introduced, targeting an analytical modeling of the reliability behavior of logic circuits. Based on functional simulation of the fault-prone version of the target circuits, the PBR method allows an evaluation of the fault masking property of the circuits. Being a time consuming approach, the conditions to minimize computing time, keeping a high accuracy, are discussed. The reliability analysis based on signal probability computation (SPR) is also introduced, presenting the particularities that differentiate the method from an ordinary signal probability analysis. The heuristics implemented to deal with signal correlations are also presented, allowing a broad range of choices concerning processing time and accuracy.

Chapter 4 presents the results obtained with the proposed reliability analysis tools concerning the arithmetic circuits in the referred library. The results are targeted to characterize the design space concerning fault tolerant approaches, defining the limits of efficiency of the proposed solutions.

Chapter 5 presents the concluding remarks, reviews the thesis contributions and discusses further perspectives.

The appendices also present the auxiliary materials developed during the thesis as a support for the main subject, concerning the initial bibliographical research and fault-tolerant circuits implementation.

Chapter 2

Reliability in nanoelectronic technologies

2.1 Introduction

The reliability of integrated circuits has become a key consideration when dealing with nanoscale designs, as a consequence of many factors associated with technology scaling, like manufacturing precision limitations, devices parametric variations, supply voltage reduction, higher operation frequency and power dissipation concerns. These problems are a serious menace to the continuous evolution observed in the development of the integrated circuits industry. There exist many techniques to improve or to counteract the reduction of reliability in integrated circuits but, generally, these techniques reduce the gains achieved with scaling and there will be a point where scaling will be meaningless. The problem in determining this point is the complexity of reliability evaluation, that leads to the use of probabilistic and stochastic methods. The reliability of a circuit is dependent on too many variables and the growing complexity of the circuits themselves does not make the task easier.

This chapter presents some general definitions regarding the reliability of integrated circuits and related properties, defining the scope of the work. An overview of fault-tolerant schemes regarding nanoelectronic technologies is also presented, as well as a brief discussion concerning the fault-tolerant library implemented for the current work.

2.2 Basic concepts

An electronic system can be characterized by four properties, i.e., functionality, performance, cost and dependability [34]. Performance in this case concerns a given application. The *dependability* of a system is a property that reflects the reliance that can justifiably be expected from the operation delivered by such system. This is a qualitative definition and cannot be expressed by a single metric. An optional quantitative definition is referred in the work of Avizienis et al. [34], i.e., "dependability is the ability to avoid service failures that are more frequent and more severe than is acceptable to the user(s)".

Dependability is characterized by several attributes, and the reliability is one of them. Fig. 2.1 shows the taxonomy associated with the dependability property, listing its attributes, threats and means.

The *reliability* of a system is defined as the probability that such system will execute its

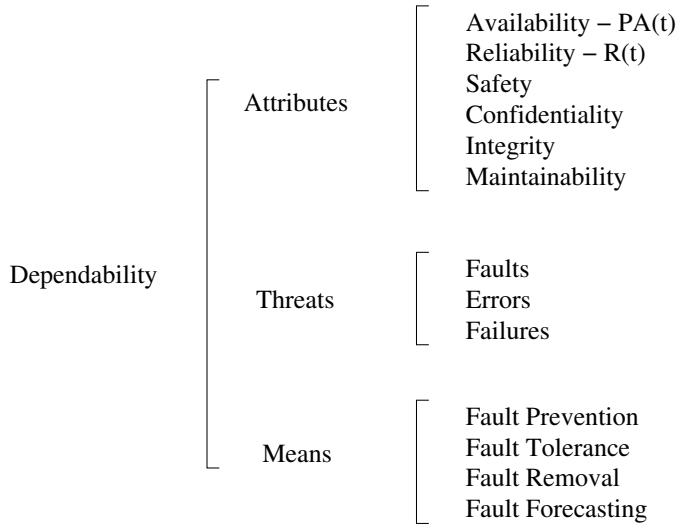


Figure 2.1: The taxonomy related to the dependability property.

specified function under given conditions for a given period of time [15], and is generally designated by R or $R(t)$. Considering that the function of a system has been specified correctly, the reliability can be seen as the probability of survival of the system [10] in the interval [0, t], where $R(0) = 1$.

An incorrect operation of a system is generated by a *fault*, an unexpected condition that can lead the system to achieve abnormal states. A fault can be originated from design flaws, physical defects or external interference and according to its duration can be classified as permanent, intermittent or transient. The presence of a fault can lead to an *error* that is an undesired change in the state of the system. The presence of an error can lead to an incorrect response of the system, what is known as a *failure*. These terms are interchangeably used according to the system level that they are referred to. An error in a transistor of a logic gate may represent a failure at the gate's output, what may represent a fault in the circuit where the gate is inserted.

Reliability can be predicted or assessed. Reliability assessment can be obtained by means of reliability tests or field data under known operating conditions. Reliability can be predicted by means of system structure, operating conditions and the reliability of system's components. The specification of reliability in electronic systems is generally presented in the form of a failure rate. The *failure rate* is the frequency with which an item fails and is an important metric in the determination of a system's reliability. Failure rate is generally designated by λ or $\lambda(t)$.

Given the number of factors that influence the failure rate of an item, a precise value cannot be determined and probabilistic methods are applied considering data from simulation, manufacturing, testing and other sources. For electronic components, the failure rate behavior is expressed by the *bathtub* curve, as shown in Fig. 2.2, where the individual contributions for the effective failure rate are presented. The time scale in the figure is not linearly distributed.

During the early life period (infant mortality failure), the failure rate of a system is mainly related to defects of the large number of new components that are used for the first time. To minimize the problems concerning the early life period, manufacturers apply burn-in tests (highly accelerated life tests), and the systems that survive these tests can be

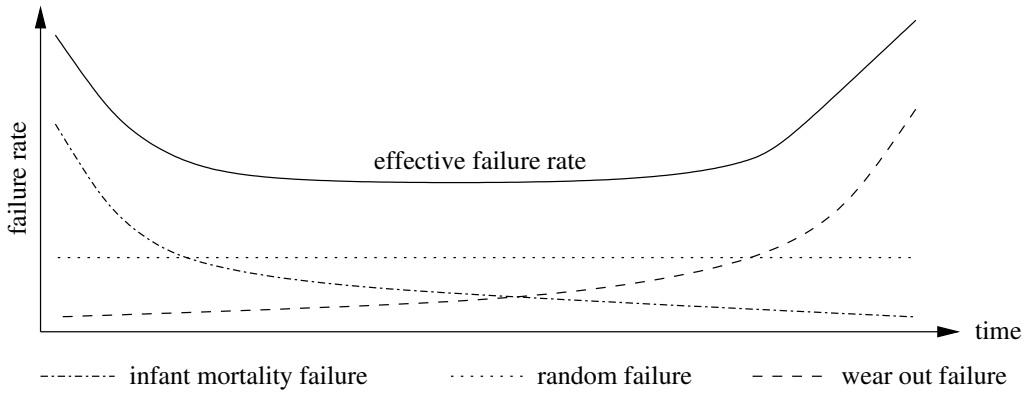


Figure 2.2: Example of a bathtub curve that represent the failure rate behavior.

considered in the useful life period. In this case, reliability figures are improved at the cost of yield reduction. Wear-out failure period concerns the problems associated to fatigue and degradation of the components and materials. In semiconductors, the main wear-out mechanisms are originated from electromigration, hot carrier injection, time dependent dielectric breakdown and negative bias temperature instability [35].

Fig. 2.3 presents the effect of operating conditions on the failure rate of a component. More stressful operating conditions (higher temperature, voltage and/or frequency) result in increased failure rate for components of the same fabrication batch. An increase in the failure rate can also be observed as an effect of technology scaling, as indicated by the work of Woods in [36].

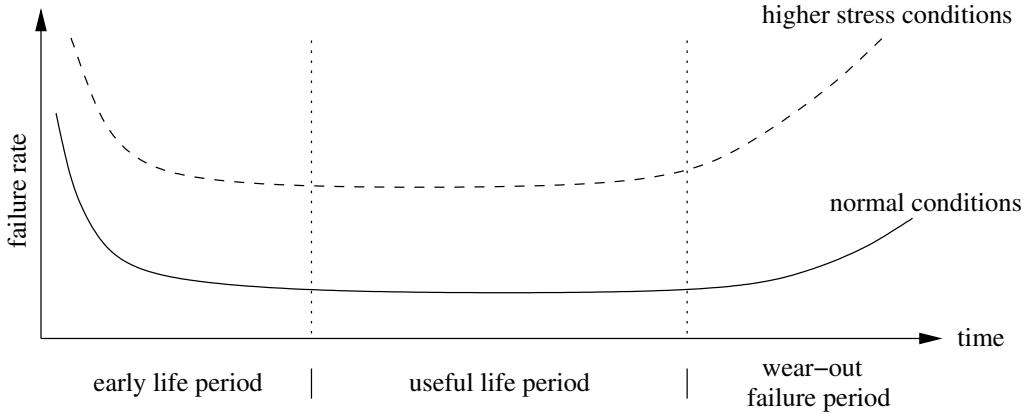


Figure 2.3: The effect of operating conditions on the failure rate.

The reliability of a component can be determined by its failure rate, as in (2.1).

$$R(t) = e^{-\int_0^t \lambda(x)dx} \tag{2.1}$$

The useful life period is considered to have a constant failure rate concerning random failures, e.g., noise, parametric variations, external interference (soft errors). In fact, the almost constant failure rate of the useful period results from the sum of all three failure mechanisms considered: defects, random failures and wear-out. During the useful life period, where the failure rate can be considered constant ($\lambda(t) = \lambda$) the reliability of a system or component can be expressed by the exponential failure law, as in (2.2).

$$R(t) = e^{-\lambda t} \tag{2.2}$$

For integrated devices, the characterization of the failure rate is usually done by a metric called *failures in time* or **FIT**, where 1 FIT means 1 failure in 10^9 device hours. The failure rate of a system with k different components, where failures are due to component failures, is shown in (2.3), where N_k is the number of components of k type.

$$\lambda_{System} = \sum_{1}^{k} N_k \lambda_k \tag{2.3}$$

Since reliability is dependent on the time of system or component operation, it is not of practical use, since it is variable. A more useful characterization is possible by means of the reciprocal of the failure rate, a metric known as *mean-time-between-failures* or **MTBF**, usually expressed in hours. Expression (2.4) relates failure rate to MTBF and in (2.5) the reliability is expressed as a function of the MTBF.

$$MTBF = \frac{1}{\lambda} \tag{2.4}$$

$$R(t) = e^{-\frac{t}{MTBF}} \tag{2.5}$$

Fig. 2.4 shows this relation between reliability and time, with the time expressed by means of the MTBF. As depicted, the MTBF corresponds to a period where the related reliability has dropped to 36.8% of its initial value.

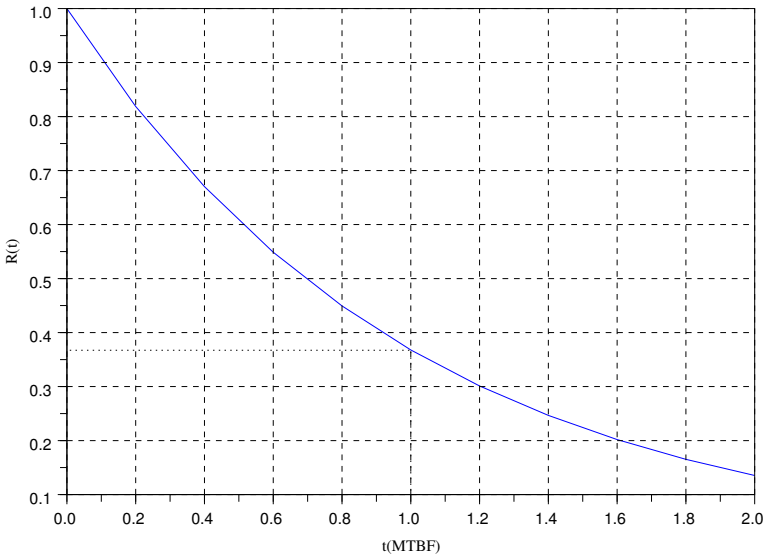


Figure 2.4: Reliability behavior over time, considering a constant failure rate.

In the case of fault-tolerant and defect-tolerant circuits, other dependability attributes may also be of interest, like the availability and the maintainability. The *maintainability* of a system or component is the probability that preventive maintenance or repair will be performed within a stated time [15] and is designated by $M(t)$. The maintainability is related to the *mean-time-to-repair* or **MTTR** and the repair rate μ, as in (2.6) and

(2.7) for a constant repair rate. The repair time includes the actions of failure detection, isolation, removal and system re-starting.

$$M(t) = 1 - e^{-\frac{t}{MTTR}} \tag{2.6}$$

$$MTTR = \frac{1}{\mu} \tag{2.7}$$

The *availability* is the probability that a system or component will perform its specified function at a given instant in time, and is designated by $PA(t)$. The availability can be expressed as a function of the MTBF and the MTTR of the system, as in (2.8).

$$PA(t) = \frac{MTBF}{MTBF + MTTR} \tag{2.8}$$

There exist different reliability measures, according to the type of faults considered, i.e., permanent, intermittent and transient, the life period considered, i.e., early, useful and wear-out, and the parameter of the circuit that is being considered, i.e., function or output data. Considering the referred parameters, we may have two reliability definitions: the *functional reliability*, that is the probability that the circuit will execute its specified function; the *signal reliability*, that is the probability of the output data being correct. The current work focuses on the latter, that is related with the fault masking capacity of the circuits, what is discussed on section 2.4.

As shown in the bathtub curves, reliability is highly dependent on the quality of the manufacturing process, that has a major contribution on the early life period of a system or component, but also contributes to random failures by means of parametric variations. The quality of the manufacturing process is expressed by the yield measure and keeping yield high in nanoscale technologies is one of the main challenges faced by the semiconductors industry. Next section presents an overview of the problems associated with nanoscale fabrication.

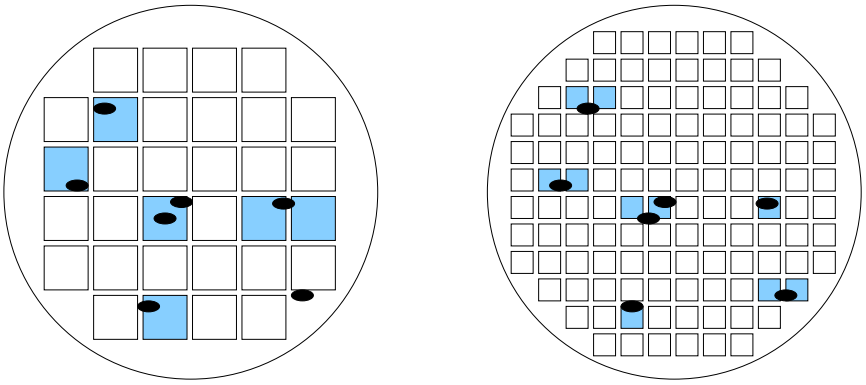
2.3 Yield at nanometric dimensions

The *yield* of a manufacturing process is a quality measure that represents the fraction of fabricated chips that has no defects [37]. To better understand the problem is important to characterize defects according to its source. There are four classes of yield loss [38]:

- *Systematic*, due to geometric or pattern variations in the manufacturing process, that affects a set of chips in the wafers;
- *Design-induced*, that are physical problems, caused by design-specific characteristics and manufacturing limitations;
- *Parametric*, associated with statistical variations in doping, channel length, gate oxide thickness, etc;
- *Defect-induced*, due to random shorts and opens, originated from scratches, particles, missing vias, contamination, etc.

Modeling yield loss is highly complex and some simplified models are available for estimation of defect-induced and parametric loss, i.e., Poisson, Exponential, Murphy, Bose-Einstein and Seeds [39]. These models consider a random distribution of defects and independent processes. For systematic and design-induced classes, specific models must be developed according to yield loss diagnosis.

Fig. 2.5 illustrates the influence of the die size in the defect-induced yield. The *gross die* value is the indication of the total number of dies in the referred wafer.



Wafer 1 : 32 *gross die*, 6 *lost die*, 81.25% *yield* *Wafer* 2 : 120 *gross die*, 10 *lost die*, 91.6% *yield*

Figure 2.5: Yield of different die sizes concerning the same defect pattern.

The Poisson yield model follows the expression in (2.9), where Y is the expected yield, A is the area of the die and D is the defect density per unit area.

$$Y = e^{-AD} \tag{2.9}$$

Fig. 2.6 illustrates the yield function by means of die size and defect density. The size of the die represents the side dimension of a square type die.

The size of the die has a significant influence on the yield, and if we consider that fault-tolerance leads to circuit's size increase, a tradeoff between reliability and yield can be established and must be considered at early design stages.

In earlier technology nodes, defect-induced problems were the main sources for yield losses, and the direct way to increase yield was to improve the manufacturing process, with better equipment and high quality materials. Thus, the yield problem was confined to the manufacturing process step and wasn't related to circuit designers. As industry moves to nanometer processes, the main sources of yield losses have become the systematic and design-induced ones. The effect of these changes in the ICs development can be seen on figure 2.7, that presents the numbers concerning real circuits case studies, where the percentage of failure in the initial designs shows a growth, according to technology scaling evolution. As shown in the figure, more than 60% of the 90-nm circuits had to be fully re-designed, affecting time-to-market and profitability.

Many factors contribute to this yield reduction. New materials, sub-wavelength lithography and process steps with statistical behavior are some of them. The use of copper wires, for example, poses some problems in chemical mechanical polishing (CMP) processes, where the thickness of the wires is affected differently along the wafer [41], leading to a mismatch between designed geometries and manufactured geometries. As another example, the implementation of sub-wavelength features, as shown in figure 2.8, reduces the

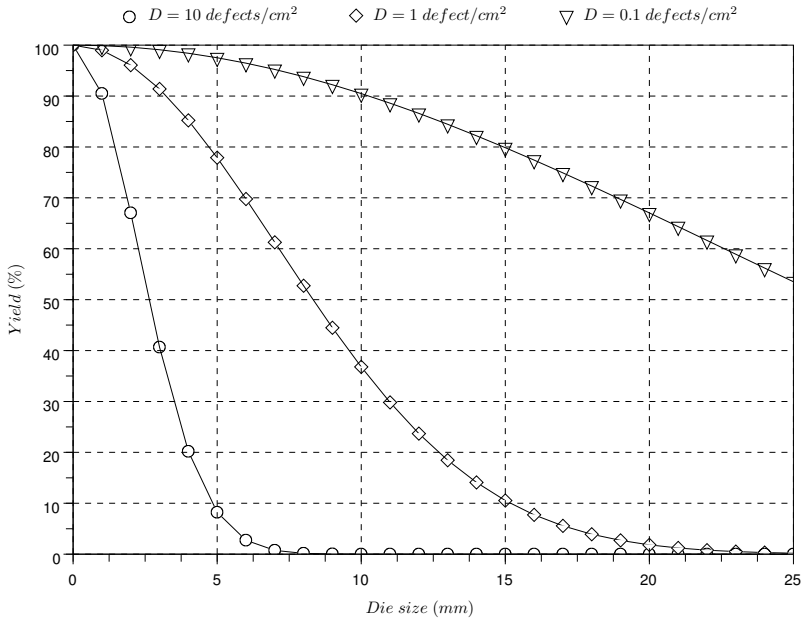


Figure 2.6: Poisson yield modeling.

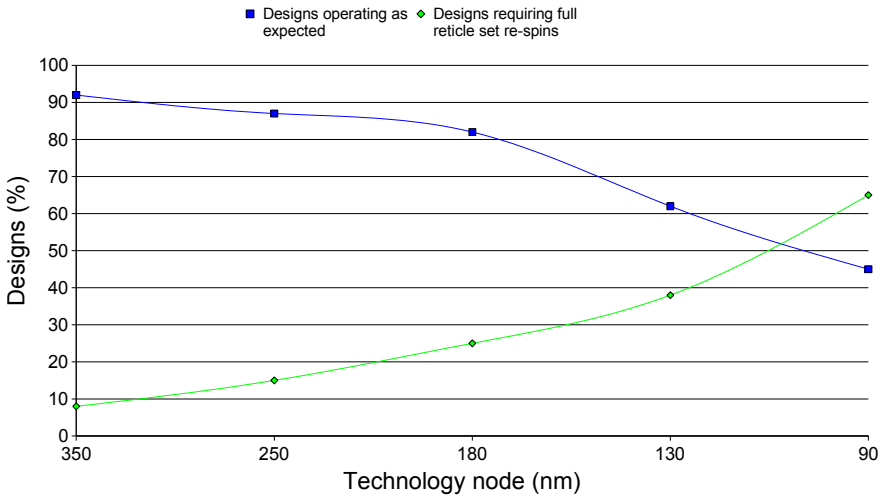


Figure 2.7: Yield evolution according to circuits scaling [40].

precision of the integrated structures, and the geometries designed for the circuit masks cannot be directly transferred, as traditionally, to wafer surface.

To deal with these lithographic problems, post-processing methods, called resolution enhancement techniques (RETs), must be applied by the mask generation tool, like optical proximity correction (OPC) or phase-shift masking (PSM). All of these problems are handled by electronic design automation (EDA) tools, in the form of improved design rules, but its increasingly complexity demands extremely high computing power and time-consuming design verification. As an example, the change from 180-nm to 130-nm processes have increased the number of these rules by a factor of 10, with some of them conflicting with others [42].

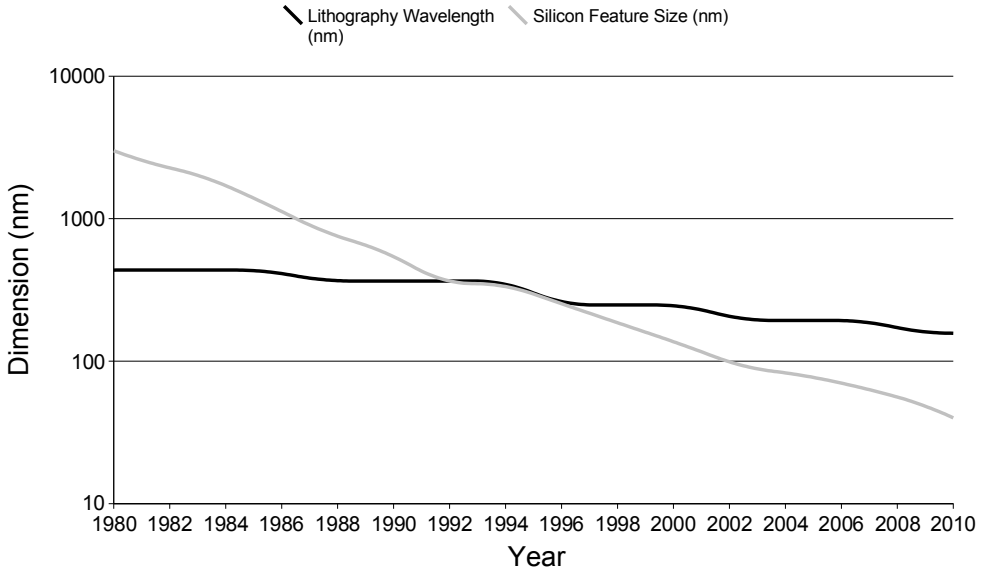


Figure 2.8: Lithography wavelength versus technology feature size [42].

Considering that traditional manufacturing improvements are not enough to increase yield, new methods must be applied in the development of nanoscale designs, like design for manufacturability (DFM), an approach that involves iterative procedures between different steps in the circuit development flow, from design and manufacture to test and diagnoses. This means that yield is becoming a new tradeoff that must be considered in ICs development by designers, along with speed, area and power. The corrective procedures to improve yield reflect in the cost of integrated circuits and there's a point where a perfect yield is not achievable, given cost constraints [43].

Considering that the yield loss sources are negatively affected by technology scaling, yield is a serious concern for future technologies, and the fabrication of perfect integrated circuits will be increasingly difficult to achieve. That's the reason for the growing interest of researches on the implementation of bottom-up assembly or self-assembly manufacturing, that may be able to produce simple regular structures with high device density, independently of lithography, and at a low cost [27]. Since bottom-up manufacturing concerns

molecular structures produced by chemical assembly, process variations and imprecision will lead to a high number of defects in the circuits, and specific yield enhancement methods will be necessary. Appendix A.3 details some defect-tolerant architecture proposals.

Given all of these facts, the semiconductors community considers that in future top-down or bottom-up technologies, defects will be unavoidable [43], and defect-tolerant design techniques will be necessary to guarantee yield and profitability. Although the manufacturing of non-defective structures is one of the main challenges in the nanoelectronics era, that's not a guarantee that the defect-free circuits will work as expected all the time, considering post-fabrication factors that may interfere with their operation. This concern is related with the reliability of the systems to transient and intermittent faults. Next section discusses this problem in more detail.

2.4 Transient faults in logic

The reliability of a circuit is associated with its capacity to sustain correct operation considering the occurrence of runtime failure(s) or interference(s), traditionally referred as faults. Reliability is related to three different classes of faults [44, 45]:

- *Permanent faults*, due to irreversible changes in circuit structure;
- *Intermittent faults*, that are characterized by repeated and sporadic occurrence, sometimes with burst behavior, and may affect circuits/devices with parametric variations, whose can act as shorts or opens under certain conditions (noise, hazards, aging, etc), and may become permanent faults;
- *Transient faults*, or soft errors, caused by temporary environmental conditions like alpha and neutron particles, electrostatic discharge, thermal noise, crosstalk, etc.

Permanent faults can be reduced by appropriate design and manufacturing methods, similar to the ones targeted to improve yield and in current processes, permanent faults can be considered negligible [45]. Intermittent faults are caused by variations in the manufactured structure that were not detected as defects, and that manifests themselves at specific conditions depending of supply voltage, temperature, etc. Intermittent faults behave like transient ones and a differentiation is difficult. Since they are related to parametric variations, these types of faults tend to increase with technology scaling. Permanent and intermittent faults are a concern for the test area, that must deal with the increasing density of integrated circuits. Intermittent faults can be reduced by careful design, quality manufacturing and effective test procedures but these procedures impact the manufacturing cost and a may lead to a reduction in profitability.

Transient faults represent a different challenge for the design of reliable integrated circuits, since they are random in nature or characterized by complex models. The focus of the present work is on the reliability of combinational logic circuits concerning transient faults. With technology entering into nanoscale dimensions, the transient fault rate of logic circuits is expected to increase and the present section details the reasons for this augmentation.

Transient faults, also called *soft errors*, may be originated from several mechanisms, like the following ones:

- *High energy neutron*, that account for most of the cosmic particles that reach sea level;

- *Alpha particles*, that are emitted by decaying radioactive impurities in packaging and interconnect materials;
- *Thermal noise*, or Johnson-Nyquist noise, that is a thermodynamical effect;
- *IR drop*, that corresponds to a reduction of the power supply voltage according to interconnect wire resistance;
- *Ground bounce* and Ldi/dt voltage fluctuations, also known as inductive noise;
- *Substrate noise*, that also translates to voltage fluctuations due to capacitive coupling of different circuit layers with the substrate;
- *Crosstalk*, due to capacitive and inductive coupling of switching signals;

Most of these mechanisms are well known and have been already modeled, but the complexity of these analysis grows with the growth in circuit's density, and some of the traditional models are not refined enough to take into account nanoscale behaviors. Transistor parametric variations combined with data values may also be a source of transient, intermittent and permanent faults, by means of delay faults, according to operating conditions and circuit escape rates in verification and test procedures.

Among the referred mechanisms, the most critical ones are considered neutron and alpha particles [46, 4], due to their strongly random behavior and also thermal noise [5] that already is the main obstacle to circuit's speed scaling. The remaining mechanisms won't be detailed, being considered as cumulative sources of transient faults.

A particle striking a sensitive region of an integrated circuit will generate a track of electron-hole pairs. The recombination of these charges will produce a current pulse whose duration and magnitude depend on the particle energy, circuit topology and transistor characteristics. According to magnitude and duration of this transient pulse, a soft error may occur, by means of a flipping in the logic value stored in a memory cell or the propagation of the transient pulse through the logic path in a circuit. A soft error in a storage cell is known as a single event upset (SEU) and in a logic cell as a single event transient (SET).

A model of the soft error rate (SER) in CMOS SRAM circuits due to atmospheric neutrons is shown in (2.10) [3, 47], where F is the neutron flux in $particles/(cm^2 s)$, A_{diff} is the area of the diffusions in cm^2, Q_{crit} is the critical charge in fC and Q_{coll} is the charge collection efficiency of the device in fC. The neutron flux is dependent on the altitude, geographic position and solar activity.

$$SER \propto F \times A_{diff} \times exp(-\frac{Q_{crit}}{Q_{coll}}) \tag{2.10}$$

The referred model has been used to predict the SER evolution due to technology scaling. In this case, the neutron flux is considered to be constant ($F = 0.00565\ part./(cm^2 s)$ in New York city) along different technology nodes. The critical charge depends on supply voltage and drain capacitances, meaning that Q_{crit} is reduced with scaling. Charge collection depends mainly on supply voltage and doping, meaning that Q_{coll} also reduces with scaling. According to Seifert et al. [4], the ratio Q_{crit}/Q_{coll} has remained approximately constant in the last technology generations, and since the diffusion area is reduced by scaling, the general trend is a reduction in the SER of individual cells due to neutron

particles. But with the improvement in the number of cells allowed by scaling, the SER of the overall circuit is expected to improve.

An important result of the work of Seifert et al. is the verification of the growing influence of alpha particles in the SER of CMOS circuits. In the 65-nm technology, neutron and alpha particle strikes equally contribute to the SER of logic and memory devices. Table 2.1 shows the ratio between neutron and alpha particle contributions to the SER of different devices and technologies [4].

Table 2.1: neutron and alpha particle SER ratio contributions.

Technology	SRAM	Latch	Combinational logic
130-nm	1.4	3.9	–
90-nm	1.1	5.6	> 10
65-nm	1.0	1.7	1.1

The probability of occurrence of multiple simultaneous faults has also been focused by the work of Seifert et al. [4], revealing an increase on this probability as circuits scale down. Fig. 2.9 shows the values determined in the referenced work, representing the probability of a 2-bit flip in a memory word according to the critical charge of the memory cells. The values have been computed considering multiple upsets originated from the same particle strike, what depends on the physical distance between memory cells.

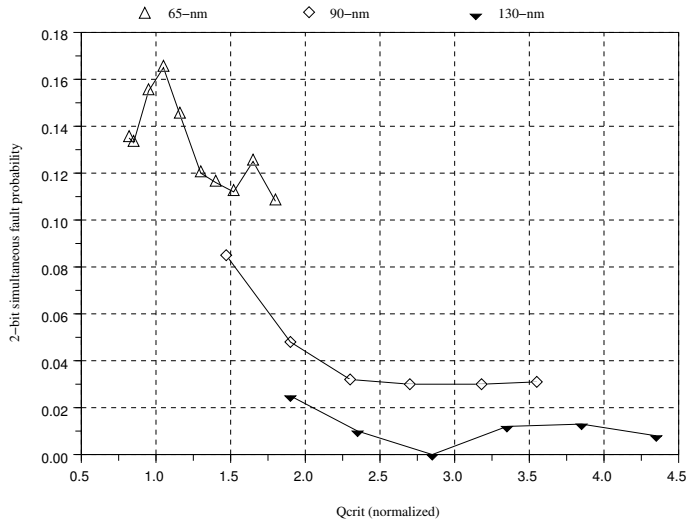


Figure 2.9: Probability of a 2-bit simultaneous fault in a memory word.

The metric that represents the SER in integrated circuits is the *failure in time* (FIT), with a FIT meaning one error in 10^9 device hours. In an IC, the fault rate for most fault sources is less than 150 FITs while for SER the value can be 50.000 FITs or more [46]. This level of SER represents almost a soft error every two years. Fig. 2.10 shows the behavior of SER in memories, according to technology scaling. In the case of SRAM SER, a comparison is shown with processes based on borophosphosilicate glass (BPSG) insulator, found to be more susceptible to soft errors.

As can be seen on Fig. 2.10, the actual sensitivity of DRAM memory cells is decreasing with scaling and, considering a complete DRAM memory array, the SER tends to remain the same. For SRAM, there's a tendency for stabilization in the SER of the cells, but with

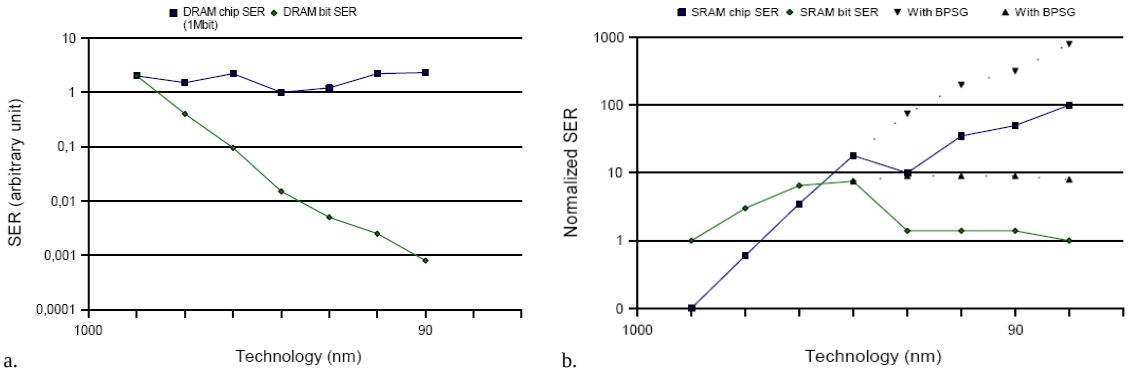


Figure 2.10: SER behavior in memories [46]; a. DRAM cell and array; b. SRAM cell and array.

increasing capacity and density, SRAM arrays are becoming more susceptible to soft errors [45]. SRAMs, flip-flops and latches have similar structures, but the former ones tend to be more robust than SRAMs [46]. These results confirm the trends of SER improvement found in similar works [3, 4, 47].

The protection of memory cells against SEUs is largely implemented by the industry, by means of error detection and sometimes correction mechanisms, generally based on data parity and Hamming codes. Parity based schemes are preferred due to the low overhead represented by the checking circuits but with multiple simultaneous bit upsets being more probable in scaled technologies, the use of parity may be insufficient for protection. Furthermore, simple Hamming encoding will not be able to correct multiple errors in the same data word. This way, a change in the traditional protection of memory cells against SEUs is one of the challenges of circuit designers in nanoelectronic technologies.

Once a concern only for memory arrays, soft errors are becoming an increasing threat to logic circuits and interconnections in nanotechnologies. Formerly negligible, SETs are gaining attention as circuits scales down and operating frequencies increase, due to a higher probability of a SET propagation to multiple paths in the circuit and capture of the voltage pulse by storage cells (latches and flip-flops). According to predictions, SETs would achieve the same probability of SEUs in the 65-nm technology [46].

The probability of SETs in earlier technologies has not been considered a menace because a soft error in a logic circuit must pass three types of filters to become a SET, known as the masking effects that follows:

- *Logical masking*, that occurs when the fault appears in a non-sensitized path of the circuit;
- *Electrical masking*, that occurs when the fault does not have enough duration or amplitude to propagate to the output of the circuit;
- *Latching-window masking* or *temporal masking*, that occurs when the fault arrives at the output of the circuit outside the storage window of the synchronous cells.

Figures 2.11, 2.12 and 2.13 show examples of logical, electrical and temporal masking, respectively, where the corresponding logical values are indicated. In Fig. 2.11, the errors are masked by the logical values present in the circuit nodes. In Fig. 2.12, the error is masked by the electrical characteristics of the gates in the logical chain, that attenuate the

magnitude and duration of the soft error. In Fig. 2.13, the error arrives at the output of the combinational circuit (S) outside the latching window, and its effect disappears shortly after, with the correct value being latched (Q) at the right moment.

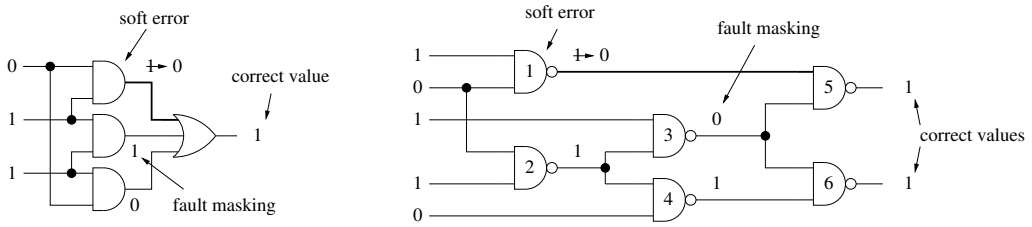


Figure 2.11: Example of logical masking.

These masking properties allowed combinational logic circuits to be much more reliable than sequential ones in earlier technologies but, as stated before, this vulnerability gap between logic and storage cells is closing with scaling. This is due to a reduction of the electrical and temporal masking effects. The works of Shivakumar et al. [3], Seifert et al. [4] and Baumann [46], whose results have already been discussed in the current section, are focused on demonstrating this reduction on the electrical masking effects originated by technological scaling. Temporal masking is dependent on operating frequency and when scaling translates into circuit speed improvement, temporal masking effects are reduced.

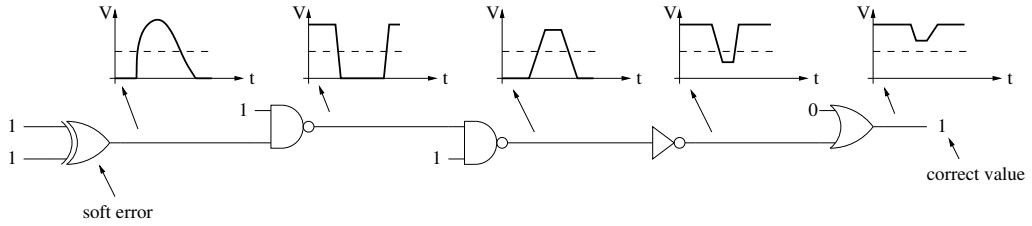


Figure 2.12: Example of electrical masking.

Logical masking is dependent on the topology of the circuit and technology scaling does not affect the logical masking capacity of a circuit. In fact, logical masking is the primary masking effect or inherent masking property of combinational logic circuits since electrical and temporal masking only take place if the fault is not logically masked. The effects of logical masking are not taken into account in the works of Shivakumar et al., Seifert et al. and Baumann. To evaluate the logical masking property of combinational logic circuits in a multiple simultaneous fault scenario, the focus of the current work is the development of logical masking models for this type of circuits.

Even if soft errors due to particle strikes are considered a critical problem, the most threatening source of transient faults in future circuits will be thermal noise. This electrical noise is generated by electrical charges thermal agitation, also known as the Johnson-Nyquist noise. Considering that logic circuits can be modeled as load capacitances, the root-mean-square thermal noise voltage (v_n) at this capacitor is modeled as in (2.11), where k_B is the Boltzmann constant and T is the temperature [5].

$$v_n = \sqrt{\frac{k_B T}{C}} \tag{2.11}$$

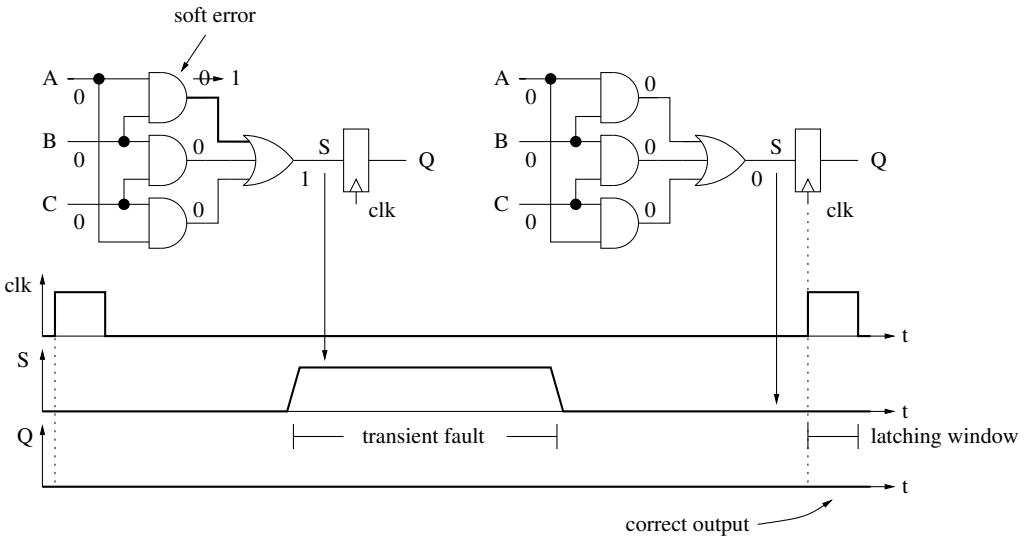


Figure 2.13: Example of temporal masking.

If the thermal noise contributes to generate a voltage that surpasses the threshold voltage v_{th} of a logic device, a bit flip occurs. The temperature T is dependent on the circuit density and operation frequency, and as circuits scale down the ratio v_{th}/v_n tends to a reduction, leading to a higher probability of false bit flips. This way, the main obstacle to technology scaling is power dissipation, what may compromise all types of advanced and emerging charge-based transport devices.

Despite the fact that the ITRS shows CMOS evolution until the 11-nm node, in 2022, scaling is strongly menaced by thermal noise and some studies [26, 5] predict the end of traditional scaling at around 2010 with nodes of 30-nm to 40-nm. Supply voltage reduction can help control thermal dissipation but in current CMOS circuits may seriously increase SER. Apparently, thermal limits have shown up earlier than predicted, when *bit-errors*, *heat* and current leakage prevented Intel of scaling its architectures from 90-nm to 60-nm in 2004 [26].

Table 2.2 shows a comparison of the operating frequency scaling trends defined by the ITRS reports of different years. As depicted, the 2007 ITRS report has reduced the expected frequency for microprocessors and ASICs, due to power dissipation limits.

Table 2.2: On-chip local frequency (MHz).

ITRS report	Implementation Year			
	2007	2010	2013	2016
2001	6,739	11,511	19,348	28,751
2003	9,285	15,079	22,980	39,683
2005	9,285	15,079	22,980	39,683
2007	4,700	5,875	7,344	9,180

Considering these facts, soft errors have become a fundamental concern for scaled technologies. In the current section, only thermal noise and particle strikes have been detailed, but it is important to remember that all of the previously referred fault mechanisms have a cumulative effect on the probability of soft error occurrence. Given this reliability reduction

scenario, and that operating frequencies and device densities must increase to guarantee profitability, fault-tolerant design techniques are being considered as one of the solutions to reduce SER in integrated circuits [45]. Traditionally applied in the design of memory arrays, as SETs increase, fault-tolerant methods would be also targeted to combinational logic in nanometric designs. Besides the application of the usual fault-tolerant methods, that are able to recover from a single-fault, new ones must be researched, to cope with multiple simultaneous transient faults that will be a reality in future technologies. The next sections presents some of the fault-tolerant methods to deal with unreliable devices.

2.5 Fault-tolerant design

Fault tolerance is a traditional research area, that arose from the need for reliable computation on dependable systems that had to operate without interruption and in critical applications and conditions. Fault tolerance can be achieved by error masking and error detection and correction. Error masking approaches are transparent to circuit operation and output, while error detection signs that incorrect operation has taken place and corrective actions must be applied. Although fault-tolerant methods have been studied for long time, many of the existent solutions were proposed to deal with a reduced number of faults, normally a single one. Thus, a fundamental challenge is the improvement of the traditional methods and the development of new ones, allowing correct operation of the circuits, in the presence of multiple transient faults, predicted to occur in the nanoscale architectures [4, 48, 49].

Fault-tolerant schemes can be focused to different levels of the system, and the focus on the current section is on methods intended to protect arithmetic and logic parts of combinational logic circuits.

N-modular redundancy (NMR) is a classical technique to design fault-tolerant circuits and consists in the concurrent operation of replicated circuits, where the output is defined by an arbiter block, or voter, based on the output of all replicated structures. In triple-modular redundancy (TMR), a particular case of NMR, three identical modules perform the same operation and the arbiter compares the three outputs to decide what is (probably) the correct result. TMR is an example of an error masking approach. NMR methods are based on the assumption that the arbiter is a reliable or hardened structure. The amount of redundancy demanded by this approach is justifiable only in specific applications where high reliability must be guaranteed at all costs, like in medical, spatial or military systems. The advantage of NMR approaches is that they are of straightforward implementation, without the need to change the original structure of the target circuit.

The use of data coding to detect and correct errors in logic circuits is another traditional approach to implement fault-tolerant applications [10, 29, 50]. Compared to replication methods, a lower area overhead can be achieved for error detection and sometimes error correction. However, its implementation depends on modifications on the topology of the target circuits and its efficiency and overhead depends on the application. Protecting operations other than storage demands different codes like the arithmetic ones or unordered. According to Chen and Vaciana [51], reliable computation with coded data is adequate to linear operations but quite complex in other cases. Like the NMR approach, the use of complex coding schemes is limited in practice to specific circuits, where reliability is more important than cost. Fig. 2.14 shows the general architecture of TMR and coded fault-tolerant approaches for combinational systems. Following is presented a brief overview of some fault-tolerant related researches.

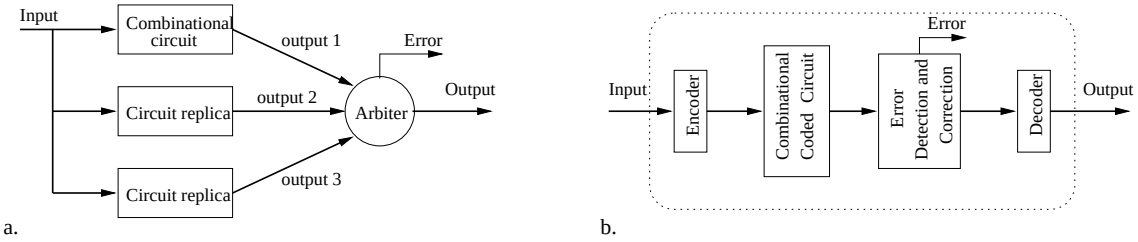


Figure 2.14: Traditional fault-tolerant schemes; a. Triple-Modular Redundancy, TMR; b. Error Detection and Correction.

A comparison of four defect/fault-tolerant schemes is presented by Nikolic et al. [52], showing that a great amount of redundancy must be integrated in the circuits to guarantee reliability, when defect rates are in the order of 10%. The schemes compared are NMR, cascaded TMR, von Neumann's NAND multiplexing and reconfiguration. The comparison shows the relation between the area overhead, the size of the target circuit, and the individual device probability failure. Considering area penalty, the worst techniques were found to be the NMR and cascaded TMR, and the best one is the reconfiguration. The work shows that modular redundancy is better suited for circuits with larger sizes and reconfiguration works better for small-sized circuits. Despite these results, reconfiguration is really effective as a defect-tolerant technique, that is able do deal with permanent errors, but to achieve fault tolerance against transient errors during circuit operation, another scheme must be used in the reconfigurable circuit.

Mitra and McCluskey [9] show a quantitative comparison of various concurrent error detection (CED) schemes, not only concerning area overhead but also fault coverage. Hardware-redundant CED techniques are based on duplicated function flows, where the output of the main circuit can be checked according to the output of a circuit that operates concurrently. In [9], five CED schemes are compared: duplex systems (identical and diverse), parity prediction and unidirectional error detection codes (Berger code and Bose-Lin code). Unidirectional error detection codes were found to be the schemes with the largest area overhead, and the parity prediction scheme achieved better results than duplex systems for this category. In the case of common-mode failures (CMF) and multiple faults, duplex systems with diverse logic have shown a higher degree of protection than parity checking and duplex systems with replicated logic. The work also discusses about transition codes and residue codes that are unable to detect some types of errors, or result in larger area overhead for higher fault coverage. Considering the system level design, the study indicates that the use of duplex system for combinational logic, and parity prediction for the other parts, may be a good approach to balance fault coverage and area overhead. A generic architecture of a CED scheme is represented at Fig. 2.15, that also shows the duplex and parity-checking organizations.

Lala and Walker [53] propose the use of signed binary digit (SBD) representation to implement a self-checking adder, that can detect a single error in the result. Parhami [54] presents the design of SBD arithmetic circuits protected from transient errors by parity checking. The conversion of the inputs to redundant representations allows the use of low overhead fault-tolerant circuits, with the reconversion occurring at the output. The overhead associated with the conversion of the data can be lowered if all of the processing datapath can be implemented with the redundant data representation, which is the case in high performance arithmetic circuits.

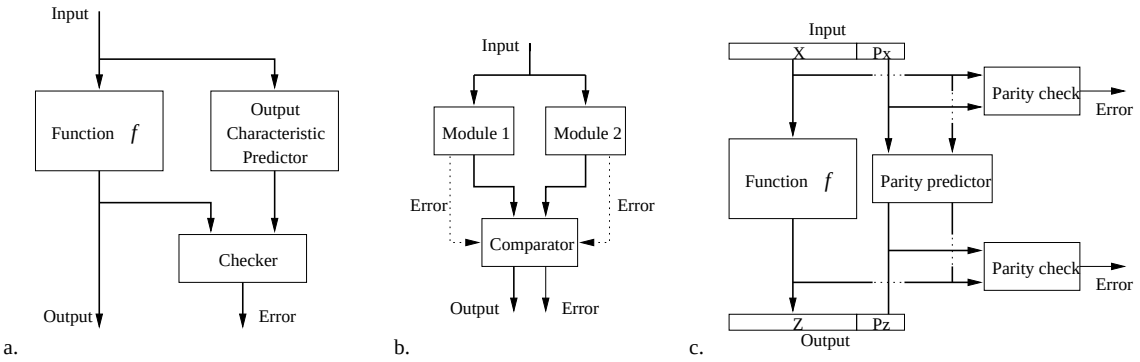


Figure 2.15: CED schemes; a. General architecture; b. Duplex system; c. Parity check system.

Cardarilli et al. [55] also explore signed digit representation to implement a self-checking adder that can locate a fault and apply some corrective actions. The use of signed digit representation allows the confinement of error propagation in the case of a fault. After a fault is located, the adder can calculate the correct output, based on two successive computations with shifted operands, or generate a degraded output (reduced number of bits) with only one computation.

Han and Jonker [56] propose von Neumann's NAND multiplexing method as an alternative to produce working hardware based on unreliable nanodevices. The work investigates the level of redundancy necessary to apply the multiplexing method in circuits with high defect rates. Despite the results that show significant area overhead, the authors consider that this problem can be compensated by the small dimensions and high densities allowed by nanodevices, like SETs. Fig. 2.16 shows the organization of a NAND multiplexing system, where the executive unit uses gate replication and signal multiplexing (U blocks) to implement the original logic function, and the restorative units restore the signals degraded by the executive unit.

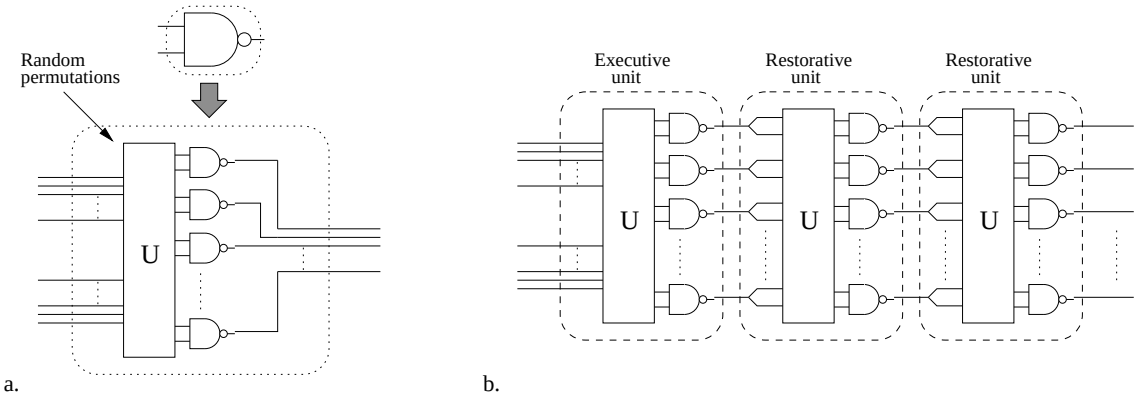


Figure 2.16: NAND multiplexing scheme; a. Function replication or executive unit; b. Complete system.

The repetitive organization of many proposed nanofabrics (circuit substrate based on nanodevices) benefits to the implementation of circuits where design regularity is a characteristic, like memories, and so, memory-based logic is a natural candidate for nanoarchitectures design style. Logic functions based on look-up tables (LUTs) are common in FPGAs,

and a similar proposal is done by KleinOsowski et al. [57], in the recursive NanoBox processor grid project. Like the Teramac computer (appendix A.3), the NanoBox processor is also organized as a hierarchy of programmable components. A NanoBox is the basic building block, that executes operations stored in a LUT, where data is protected by some fault-tolerant approach, like error-correcting codes (ECC) or TMR. Up in the hierarchy, a grid of NanoBoxes forms a module, and the system level is composed by a group of modules. Fault-tolerant schemes are also implemented in the module and system levels. The work presents the simulation of a NanoBox processor grid acting as an image co-processor, where faults are injected to determine the reliability of the system. The comparison of different fault-tolerant schemes in all the system levels, showed that a combination of TMR in the bit level and TMR in the module level presents higher reliability than coded data schemes. Considering that the applied fault-tolerant scheme was cascaded TMR, the area overhead was high, in the order of eight times. Fig. 2.17 shows an example of a NanoBox implemented with coded protection and the hierarchy of NanoBox architectures.

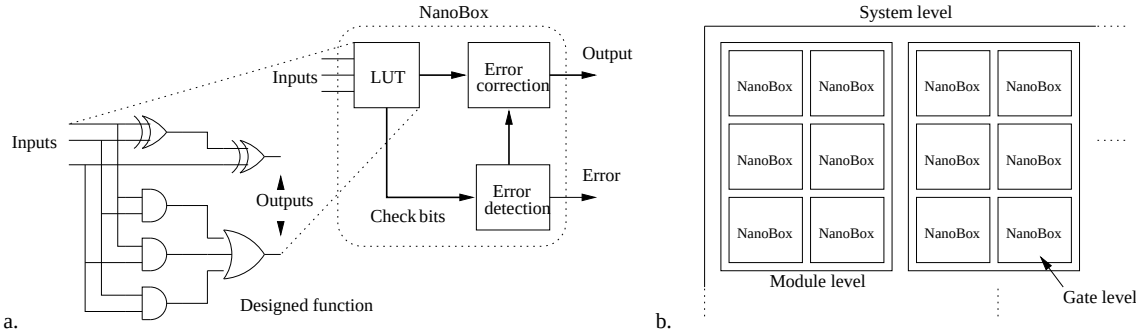


Figure 2.17: NanoBox approach; a. NanoBox with coded fault tolerance; b. Hierarchy of NanoBoxes.

In [58], Kastensmidt et al. shows the problems associated with single-event upsets (SEUs) in SRAM-based FPGAs, that can affect the combinational, sequential and wiring parts of the circuits. The occurrence of an upset in an ASIC circuit has a transient effect, but in memory-based configurable devices, a SEU may become a permanent error, similar to a manufacturing defect, as can be seen on Fig. 2.18.

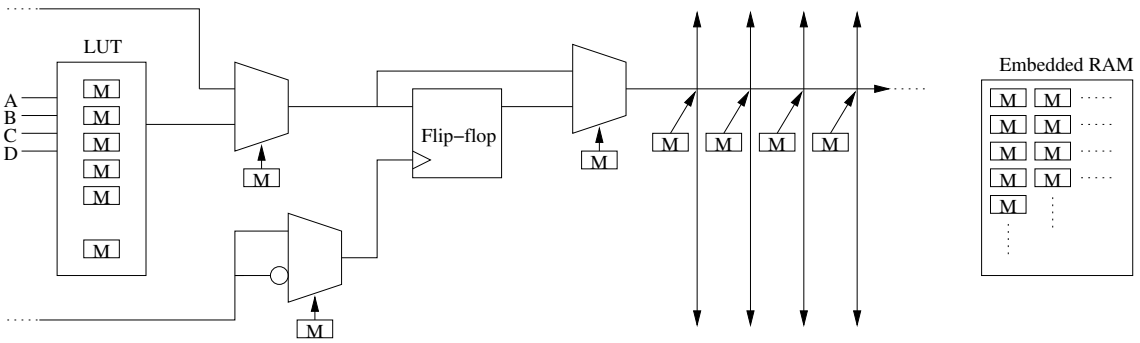


Figure 2.18: Points in a FPGA where a SEU may interfere: configuration bits (M), flip-flops and embedded memory.

The work presents a fault-tolerant design technique based on duplication with comparison (DWC) and another concurrent error detection (CED) scheme based on time redun-

dancy. The DWC part indicates when a mismatch in the output occurs, and after that, the fault-tolerant scheme of recomputing with shifted operands (RESO) determines the faulty block. A detailed study presents the impact of this approach in terms of area, speed an power, and shows good results when compared with the TMR approach.

In [59], Almukhaizim and Makris present a similar approach that uses CED to achieve error detection in combinational and sequential logic design. Instead of using a simple copy of the circuit to compare the outputs, a re-synthesized version of the circuit is implemented, based on parity check, allowing diagnosis by a parity predictor, and error correction actions. The circuits designed with this approach can generate correct outputs in the case of multiple faults in the original circuit, in the re-synthesized version, or in the parity predictor, but can't withstand faults in all of the circuits. Compared with TMR implementations, the parity check CED approach shows a reduction in the area overhead of the target circuits. The scheme proposed by Almukhaizim and Makris can be seen on the Fig. 2.19.

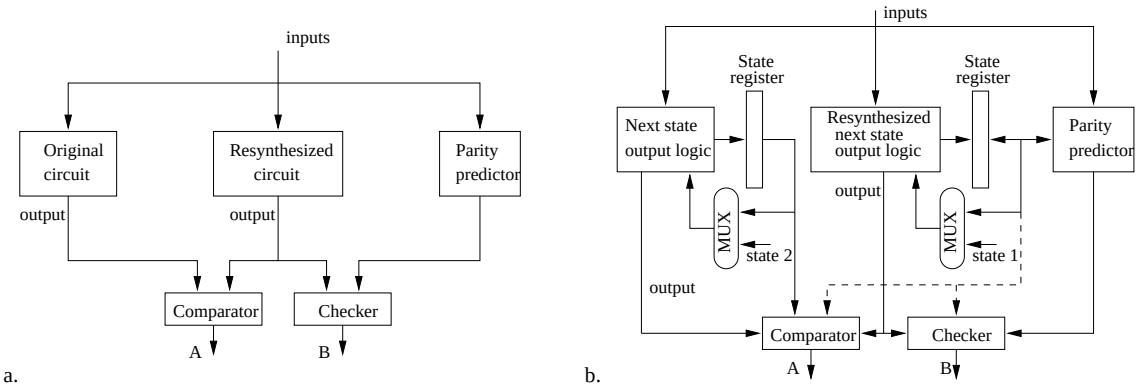


Figure 2.19: Fault tolerance with re-synthesized logic and parity check; a. Combinational circuits; b. Sequential circuits.

Lisboa and Carro [60] present the use of bitstream operators to cope with multiple soft errors, based on the theory of stochastic computation. By converting the binary values to data bitstreams, and using appropriate operators, the impact of transient errors on the output of the circuit may be reduced, and by incorporating redundant bits in the bitstream, error correcting capabilities are achieved. The level of fault coverage (number of simultaneous faults supported) is proportional to the number of extra bits incorporated in the bitstream, and so is the area overhead. Despite the sequential processing characteristic, the reduced area of the circuits allow the use of multiple units operating in parallel. A 5-taps filter implementation using the proposed architecture was compared with a TMR implementation, resulting in an area overhead of 18%, but with fault tolerance to multiple faults [48]. Following a similar idea, Schüler and Carro [61] explore the use of sigma-delta modulators, to generate bitstreams that are highly tolerant to multiple errors. The work presents arithmetic operators that, in the presence of multiple faults, may generate correct outputs or outputs that are very close to correct ones. This characteristic is not a problem for some applications, that can tolerate some level of imprecision, like the ones discussed on the error tolerance approach [28]. Fig. 2.20 shows the use of the sigma-delta approach in arithmetic operators. In another work concerning fault tolerance, Lisboa et al. [62] present an analog approach to increase robustness in the voter component of TMR and NMR schemes. By injecting faults in TMR circuits, the study shows that digital implementations of the voter can't withstand single transient failures, and the work

proposes the use of analog techniques to design the voter. The result is a solution that is really tolerant to single transient faults with low area cost.

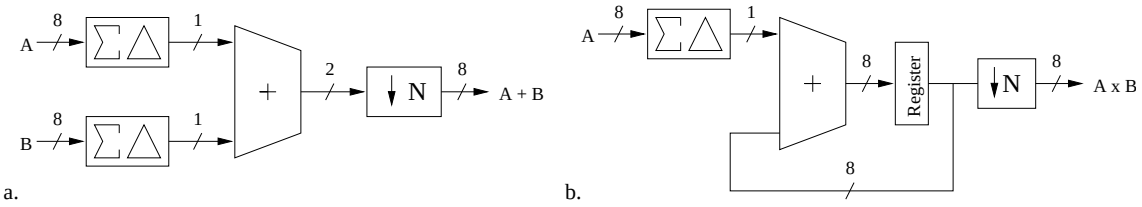


Figure 2.20: Sigma-delta modulation in robust arithmetic operators; a. Adder; b. Multiplier.

The work of Schmid and Leblebici [63] combines multiple-valued logic with a fixed-weight neural network architecture, to implement logic functions that present a high immunity to multiple faults. The work discusses the properties of SET devices, that can operate with discrete charge levels, and are well suited to integrate such architectures. According to the proposed approach, a logic block is divided in four layers: input, logic, averaging and decision. The logic layer is composed of replicated logic circuits that implement the desired function. The number of replicas in the logic layer is related with the fault immunity. The averaging layer receives the outputs of the logic layer, and generates a multi-valued logic, that will be compared to a threshold in the decision layer to define the output. The idea is to create basic building blocks that are inherently defect and fault-tolerant, and that can be used by system designers to generate more complex architectures, hiding from them the challenges of reliable design. Fig. 2.21 shows the layered architecture of the proposed approach and a logic function implementation example.

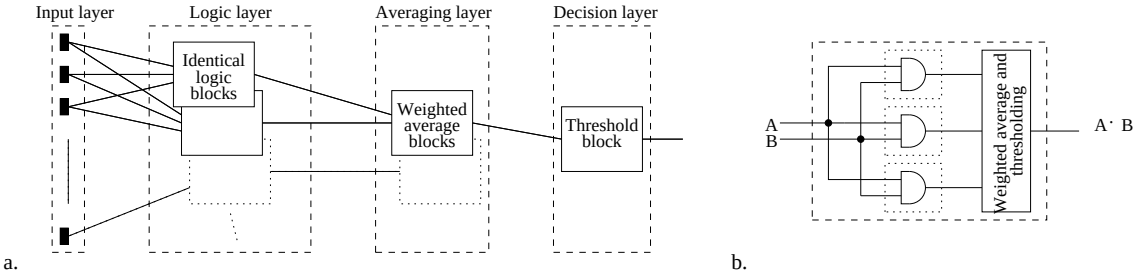


Figure 2.21: Multiple-valued logic approach; a. Layers organization; b. AND function example.

A cellular array scheme is presented by Peper et al. [64] as a candidate architecture to nanosystems, due to its asynchronous operation. The rules driving the design and computation of cellular arrays are discussed and the introduction of fault-tolerant capabilities in the form of error-correction codes is presented.

The work of Anghel [29] explores many fault-tolerant schemes to deal with transient faults. By comparing traditional techniques like TMR and duplication with data-coded ones, the work determines the drawbacks of each method, and proposes a combined approach of space and time redundancy that achieves high degrees of tolerance with small area and speed overhead. The scheme implemented is called code word state preserving (CWSP) and the logic circuits implemented with CWSP gates are inherently tolerant to transient faults. In the work of Lazzari et al. [65], an automated fault-tolerant design

process is demonstrated, where CWSP cells are used by the layout generator algorithm to create architectures with improved reliability to transient faults.

The work of Patel and Markov [66] is directed to fault protection of the wiring parts of the systems. The work proposes a new class of codes, called *boundary-shift codes*, that are intended to minimize the probability of crosstalk noise, and that are combined with error-correction capabilities. As a conclusion of the work, boundary-shift codes are found to introduce less overhead than other conventional codes and methods used to protect buses.

Chan et al. [67] implement a fault-tolerant design in a FIR filter using the modulus replication residue number system (MRRNS) to reduce the associated overhead. The method is applied in the design of a filter with 128 taps, and the result is an increase in the order of 83% in the area of the filter, considering a tolerance to a single fault.

Marculescu [68] presents the relation between the redundancy necessary to achieve fault tolerance in a given circuit and the cost of this redundancy in power consumption and delay. These factors must be taken into account, when energy efficiency and performance are constraints for the application. As an example, the work shows that, to achieve 99% of reliability in a given circuit, with 1% individual gate failure probability, the increase in the energy, due to fault-tolerant redundancy, is on the order of 40%.

As referred at the beginning of this section, fault-tolerant schemes can be targeted to higher levels of the system, i.e., more complex blocks controled by sequential logic, like filters, microprocessors, system-on-chip (SOC) and system-in-package (SIP). These higher levels of the hierarchy are protected by system-level fault-tolerant schemes like fault detection and correction codes, watch-dog timers, lock-step processing, simultaneous and redundant multithreading, active-stream/redundant-stream simultaneous multithreading and slipstream processors, among others. Most of these schemes explore the flexibility of software redundancy and the reduced overheads associated with system-level solutions.

Most of the fault-tolerant methods and architectures presented on the current section are designed following the traditional top-down approach, where the function or structure of the system is specified at some level, and successive steps of synthesis generate lower level descriptions that can be implemented on a given substrate. Whether these methods are compatible with the defect-tolerant approaches presented in appendix A.3 remains to be proved. Most of the referred defect-tolerant approaches are based on reconfigurable substrates or nanofabrics, with grain-sizes varying from self-assembled molecular switches to lithographically fabricated functional blocks. Reconfiguration is fundamental to allow the circumvention of defects in the substrate, making them transparent to the target system. In most cases, the compatibility of a given fault-tolerant design with some defect-tolerant nanofabric will depend on a synthesis step, but the specificities of the nanofabric like the grain-size and basic device type will determine the range of applicable methods.

Besides the presented fault-tolerant studies, fault avoidance or fault prevention approaches have also been proposed by the research community, as well as partial fault tolerance. These works are targeted to fine tuned improvements of the reliability of the circuits, allowing optimized tradeoffs among reliability, implementation area, power consumption and speed. Next section presents some of these alternative approaches.

2.6 Fault prevention

Most of the traditional fault-tolerant approaches, i.e., NMR, CED, self-checking, are designed to cope with specific types of faults like single stuck-at faults or unidirectional

multiple faults, and are designed with specific goals, like being fault-secure and capable of self-test [10]. Considering these objectives, combined with schemes targeted to system level application, leads to important overheads that are not compatible with mainstream applications and sometimes are not as effective considering transient faults [29].

An alternative to these methods is the hardening of individual logic cells, transistors inside the cells, or the implementation of partial fault tolerance, where only a subset of the circuit is protected by coding or redundancy. The approach of hardening individual components is known as fault prevention or fault avoidance, and allows a fine-grain improvement in circuit's reliability by targeting lower levels of the architecture.

The work of Mohanram and Touba [11] presents a partial fault-tolerant scheme, where only a subset of the circuit is replicated to CED purposes. The susceptibility of each node in the circuit to soft errors must be determined and considering a threshold of susceptibility the concerned gates are duplicated, as shown in Fig. 2.22. On the figure, the susceptibility of the gates of circuit C17 (from the ISCAS'85 benchmark) defines the duplicated gates on this partial CED scheme. The main drawback of the proposed approach is the computation of the soft error susceptibility of each node, that must be determined by fault injection and simulation. The proposed approach can be used for improving reliability according to the design constraints in area, speed and power consumption.

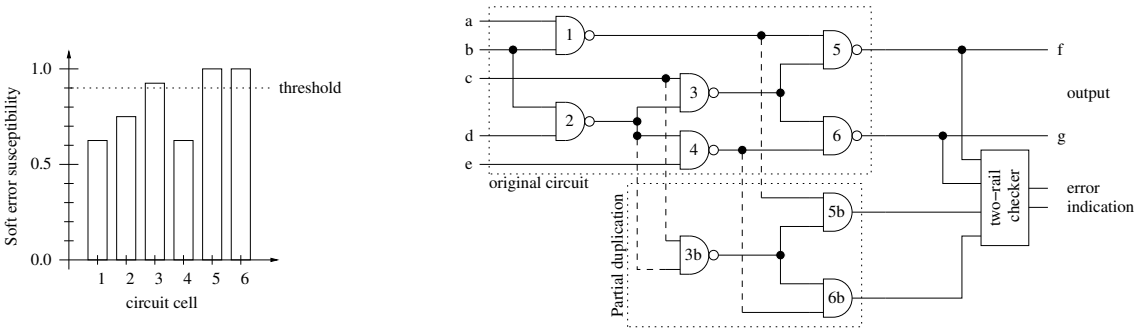


Figure 2.22: Susceptibility of circuit cells and partial duplication.

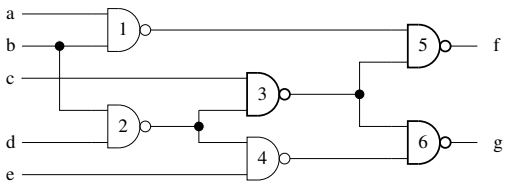
Zhou and Mohanram [12] propose the hardening of individual gates as a transient fault prevention approach. By determining the correlation between gate transistors size and gate soft error rate, the work explores the use of different sizes of equivalent cells by an iterative reliability improvement algorithm. The algorithm starts by ranking the cells by their susceptibility to soft errors (sensitization probability), according to logical masking computed by pseudo-random simulation. The work uses the *coverage* metric, as in (2.12), to determine the expected reduction in the soft error rate of the circuit, considering the sizing of the candidate cells (g_c), where $P_s(g_c)$ is the sensitization probability of the chosen cells set, and $P_s(g)$ is the sensitization probability of all cells.

$$Coverage(\%) = 100\left(\frac{\sum_{(candidates\ g_c)} P_s(g_c)}{\sum_{(all\ gates\ g)} P_s(g)}\right) \tag{2.12}$$

Fig. 2.23 shows the coverage obtained by sizing the more vulnerable gates on the C17 circuit. A coverage of 60% corresponds to a reduction of 2.5 times in the soft error rate.

The work present results for several circuits, covering different technology nodes, where reductions of an order of magnitude in the soft error rate of the circuits could be achieved with average overheads of 38.3% in area, 27.1% in power and 3.8% in speed. These over-

Gate	Sensitization Probability
1	0.625
2	0.75
3	0.9375
4	0.625
5	1
6	1



$$\text{Coverage} = 100\,\frac{2.9375}{4.9375} = 59.49\ \%$$

Figure 2.23: Coverage obtained with sizing the indicated gates of circuit C17.

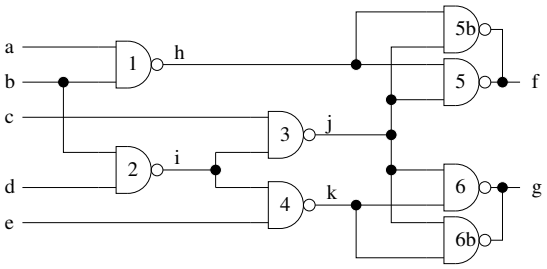
heads tend to increase with technology scaling. No details on the accounting of multiple simultaneous faults are shown in the work. The work does not detail the vulnerability of resized transistors concerning the increase in the sensible area.

The work of Dhillon et al. [69] proposes a similar approach of electrical masking improvement by gate sizing. The work presents the modeling of transient faults (particle strikes) propagation through logic gates according to SPICE simulations, and the generation of look-up tables for each logic cell, where several gate sizes, voltages and fault transfer functions are stored. Considering the cumulative probability of propagation of every possible single fault to primary outputs, the model determines the reliability of the circuits, and allows the optimization of this reliability by replacement of critical gates. Logical masking is computed by applying 10000 random inputs to the circuits. For practical circuits, logic partitioning must be executed to limit the computation time. The results of the application of the proposed method to ISCAS'85 circuit have shown an average reduction in the unreliability by 79.3% at the price of 6.2% increase in propagation delay and 30.9% increase in power consumption.

The work of Nieuwland et al. [13] discusses the problems of gate sizing and other fault-tolerant approaches, proposing the gate multiplication method for improving the reliability of logic circuits. The use of parallel gates reduces the probability of occurrence of a transient fault at the output of the gates, by improving the node drive strength. The method avoids redesigning library cells. The gates in the target circuit must be ranked according to their contribution to the soft error rate of the circuit, what is computed as the observability of the node in the outputs and is obtained by pseudo-random simulation. A simple algorithm guides the introduction of parallel gates until the reliability objective is met or other design constraints are violated.

Fig. 2.24 shows the reduction in the soft error rate by duplicating the output cells of the circuit, considering that a duplicated NAND gate has a reduction of 10 times on its soft error rate (in adimensional units). The proposed technique produces better results for larger size circuits.

The work of Sierawski et al. [14] uses approximate logic functions to improve the logical masking capacity of the target circuit. In a TMR implementation, each replica of the circuit will mask 100% of the transient faults, but by relaxing this masking rate, reduced overheads may lead to relevant reliability improvements. The original logic function G is replaced by an equivalent one $\hat{G}$, that is a generated by G and the approximate functions H and F, i.e., $\hat{G} = F + GH$. Fig. 2.25 shows the effect and construction of the equivalent logic function. This way, H protects G against transient **1**s and F protects G against transient **0**s. The work shows some results of the application of the proposed approach in several benchmark circuits, where good reliability improvements are achieved with reduced area overhead. The main drawback of this approach is the complexity involved in choosing the



SER gate	SER original circuit	SER hardened circuit	SER reduction
0.01	0.048	0.031	35.4%
0.05	0.216	0.145	32.9%
0.1	0.379	0.267	29.6%

Figure 2.24: Example of gate multiplication in the C17 circuit.

appropriate approximate functions, and comparing many different topologies.

Figure 2.25: Construction of the equivalent logic function and the resulting input space.

Tosun et al. [70] present a reliability-centric synthesis approach, targeted to high-level synthesis. The work is based on a library of arithmetic operators that has been characterized for soft error susceptibility. By having many alternatives of the same functional block, with different constraints, the synthesis process can cycle through different system configurations, choosing the library blocks that are adequate in terms of reliability, area, speed and power. Some results of possible tradeoffs between system constraints are presented for a FIR and an elliptic wave filter and for a differential equation solver. The characterization of the design library is done considering a simplified reliability model, not considering masking nor multiple faults.

To evaluate the functionality of some fault-tolerant approaches a library of arithmetic circuits, adders and multipliers, has been implemented in the current work. The reliability analysis of the circuits in the library should point out advantages and drawbacks of the proposed approaches. Next section presents the circuits available in the implemented library.

2.7 Fault-tolerant library

The main objective of the current work is developping reliability analysis tools for fast and accurate evaluation of logic circuits, and a secondary objective is the use of these tools to characterize the design space of fault-tolerant approaches. The present section briefly discusses some fault-tolerant approaches that have been implemented in the current work as study cases, concerning arithmetic circuits. These fault-tolerant circuits form a library that should evolve continuously. The chosen schemes of fault tolerance concern the most traditional ones, since some of the fault-tolerant approaches discussed in the previous sections are complex to reproduce or depend on yet unavailable devices and architectures.

Is worth mentioning that the implementation of some fault prevention approaches are not related to circuit architecture changes and their evaluation is possible considering the circuits already present in the library.

The circuits in the library have been described in the VHDL language, with parameterized data widths, and targeted to standard cell synthesis. All the circuit functions have been validated by VHDL testbenchs, executed on a Modelsim simulator.

Table 2.3 shows the circuits described in the library and the concerned fault-tolerant approaches implemented.

Table 2.3: Circuits versus fault-tolerant schemes implemented in the library.

Circuit type	Fault-tolerant scheme			
	DWC	Parity-prediction	TMR	1-out-of-3
Ripple-carry adder	X	X	X	
Carry-select adder	X	X	X	
Carry-lookahead adder	X	X	X	
Functional adder	X			
Signed digit adder (radix-2)		X		
Signed digit adder (1-out-of-3)				X
Booth multiplier		X		
Functional multiplier	X			

As referred before, the present library has not the objective of being an exhaustive source of fault-tolerant arithmetic circuits and new ones should be included as needed. A detailed presentation of the circuits is available at appendix B.Following is presented an overview of the circuits in the library and synthesis results.

The traditional adder topologies, i.e., ripple-carry, carry-select and carry-lookahead, represent well known tradeoffs between area, speed and power and a further characterization of these circuits by means of signal reliability is interesting for a more detailed characterization of the respective design space. The fault-tolerant approaches concerning these circuits are concurrent error detection and triple modular redundancy (TMR). The CED schemes implemented are duplication with comparison (DWC) and parity prediction, with these circuits also being referred as self-checking topologies. The DWC topology implies the duplication of the target circuit and the comparison of circuit's outputs by two-rail checkers [10]. The parity prediction scheme implies some redundancy inclusion in the carry chain logic to guarantee the fault secure property of the circuits. The TMR approach is based on the concurrent execution of the arithmetic function by three copies of the same circuit and the generation of the output by a majority function between the three results issued from these copies. Is worth mentioning that these three fault-tolerant approaches are generally compared according to their *functional reliability*, what is different from the *signal reliability* that is the focus of this work.

The functional reliability is the probability that the circuit will execute its specified function. This means that for CED schemes, the specified function is the detection of any single fault during operation (as well as the original function of the circuit). For TMR-based circuits, the function is the masking of any single fault during operation. The functional reliability of these schemes under single fault assumption is 100% ($R = 1$), if we consider perfect voters in the TMR scheme. In contrast, the signal reliability is related with the probability of a correct output of the circuit. If we consider that the auxiliary checking signals generated by the fault tolerant approaches are also output signals, then

the signal probability can be closely related to the number of logic cells, i.e., more cells have less probability of generating a correct output. Therefore, the improvement of the functional reliability is accompanied by a reduction of the signal reliability in self-checking circuits.

The referred fault-tolerant approaches must also be characterized with respect to their functionality under multiple simultaneous faults, which can be decisive in future technology nodes. The main objective is to determine the fault rate value where fault-tolerant circuits are less reliable than the unprotected ones. This characterization should take into account ordinary voters and checkers, instead of fault-free ones as traditionally considered.

2.8 Synthesis results

This section presents some early analysis on the circuits described in the fault-tolerant library detailed in appendix B, and the raw data obtained by synthesizing the parameterized circuits with Cadence BuildGates, concerning a 130-nm standard cells library from STMicroelectronics. This raw data contains the number of cells, area of implementation, propagation delay and power consumption for all the main circuits described, synthesized for 8, 12, 16 and 32-bit data widths. The raw data of the synthesis results can be consulted in the Appendix section B.10.

To analyze the synthesis results from the circuits in the fault-tolerant library is somehow difficult, since several different types of analysis are possible, considering circuit objectives, tradeoffs available and a series of factors that are difficult to quantify, like how much time the designer has to cycle through different solutions.

A first natural analysis is a comparison between the unprotected circuits. This comparison is a traditional one and allows the definition of the design space available in the library. Fig. 2.26 shows a relative comparison of all parameters concerning 12-bit circuits.

The nomenclature used corresponds to the following definitions: **rca**, ripple-carry adder; **csa**, carry-select adder; **cla**, carry-lookahead adder; **fca**, functional adder; **sd2**, radix-2 signed digit adder; **sd3**, 1-out-of-3 signed digit adder; **booth**, Booth multiplier; **fcm**, functional multiplier. For the fault-tolerant versions of the circuits, the nomenclature used is: **dup**, duplication-based fault tolerance; **par**, parity-based fault tolerance; **tmr2**, interlaced triple modular redundance; **1o3**, 1-out-of-3 checking. Functional adders and multipliers corresponds to proprietary circuits generated according to datapath functions available from the synthesis tool or standard cell library.

The values presented on Fig. 2.26 are relative to the ripple-carry adder (for adders) and to the Booth multiplier (for multipliers). The frequency value is computed as a function of the propagation delay obtained with the synthesis. The figure shows, for example, that despite the carry-free operation of the signed digit adders, they are not the fastest ones, with the functional adder being a better choice in terms of frequency, power consumption and number of cells. On the side of multipliers, the functional multiplier presents similar characteristics to the Booth multiplier but at the price of a higher number of cells.

The next analysis concerns the overheads associated with fault tolerance. Fig. 2.27 shows the area overhead concerning the different circuits and approaches, where the overhead value is relative to the unprotected version of the related circuit, i.e, the overhead of a duplication-based carry-lookahead adder is related to the unprotected version of the carry-lookahead adder. The better results are for parity prediction schemes, where the overhead depends on the area of the original circuit, with smaller circuits having a higher overhead. The parity-based Booth multiplier shows a reduced overhead compared to the

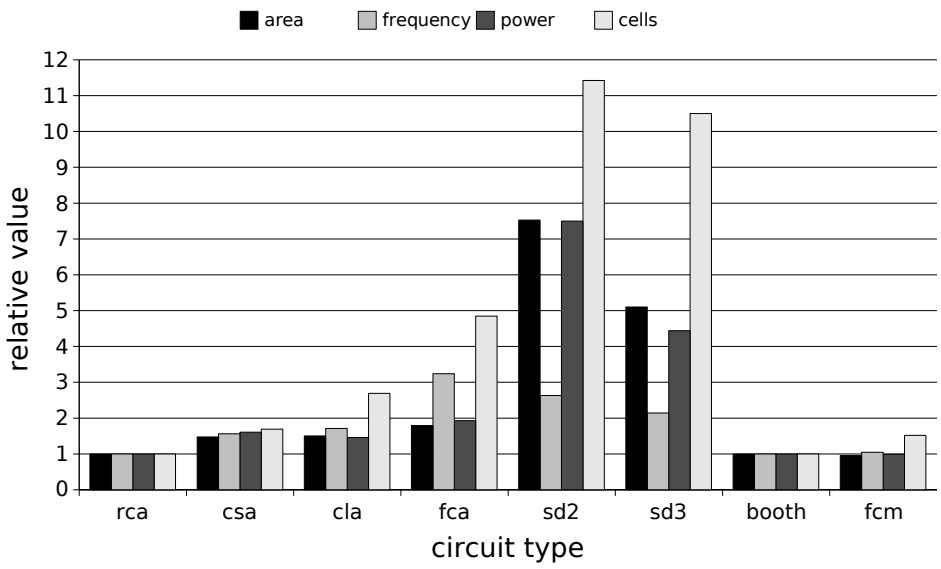


Figure 2.26: Overall results for 12-bit circuits.

duplication-based functional adder. The interlaced TMR structure of adder **rca tmr2** represents an area that is 4 times larger than the original but is expected since the circuit is targeted to a higher reliability based on fault masking.

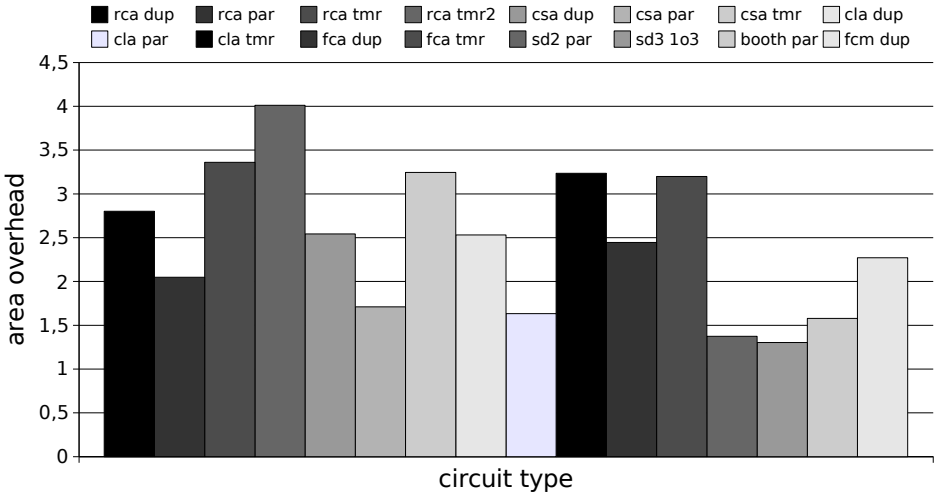


Figure 2.27: Area overhead for fault-tolerant versions of the circuits, 12-bit data width.

Fig. 2.28 shows the increase in the propagation delay of the fault-tolerant circuits, with the TMR approach presenting the better performance among the implemented approaches. The functional adder based on duplication and the signed digit adder with 1-out-of-3 code checking suffer from a great overhead on propagation delay, possibly associated with the checking tree, that introduces a serial propagation path into highly parallel structures.

The results for power consumption overhead are similar to the ones concerning area

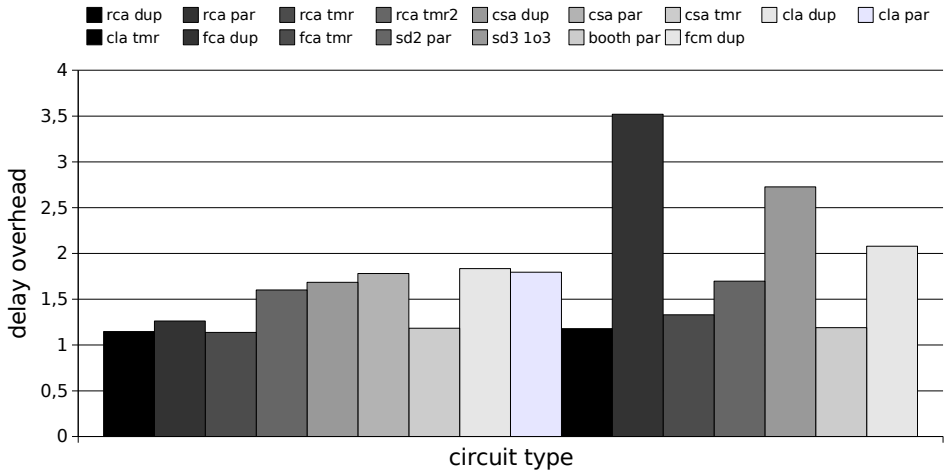


Figure 2.28: Propagation delay overhead for fault-tolerant versions of the circuits, 12-bit data width.

overhead and will not be detailed. The analysis on the number of cells will be discussed in chapter 4, where the impact of the number of cells on reliability is discussed.

The results obtained with the synthesis of the arithmetic circuits alone are an indication of their characteristics, but these circuits will probably integrate more complex structures and the impact of fault tolerance on these structures can be different from those of the standalone arithmetic blocks. As discussed in the Appendix section B.9, some multiply-accumulate filter blocks and FIR filters have been synthesized to evaluate the impact of fault tolerance on more realistic circuits. The raw data about the synthesis of filters and filter taps can be seen in the Appendix section B.10.

Fig. 2.29 shows the relative overhead obtained for different topologies of 8-bit 16-tap parallel filters. The overhead values are relative to the carry-lookahead version of the filter, with parity-based checking. The filters considered are based on the Booth multiplier block. The results show that the area (and power) overhead associated with different adders are not so critical on the filter structure as they were on the stand alone blocks. This can be explained by the optimizations associated with the signed digit adders and the predominant contribution of the multipliers and registers in the overall characteristics of the filters.

Fig. 2.30 shows the same analysis but considering the sequential version of the same filters. The results show a reduction of the area (and power) overhead levels when considering the parallel filters, a result that can be explained by the storage part of the sequential filters, that offsets the impact of the signed digit adders. Again, the main delay limitation is the multiplier block, that offsets the gains of the use of carry-free adders. A comparison of different types of multipliers is presented on the following analysis of filter taps.

Since the basic block of digital filters can be considered the multiply-accumulate block, i.e., a tap in a parallel filter, several topologies of filter taps were implemented and their synthesis results can be seen on Fig. 2.31. different combinations of multipliers and adders were synthesized and only some of them are presented here. The results show that the fastest option is the duplication-based fault-tolerant tap, with functional adder and multiplier, what is interesting for high level system description, and time-to-market purposes. Since functional circuits allow better degrees of optimization than topology-

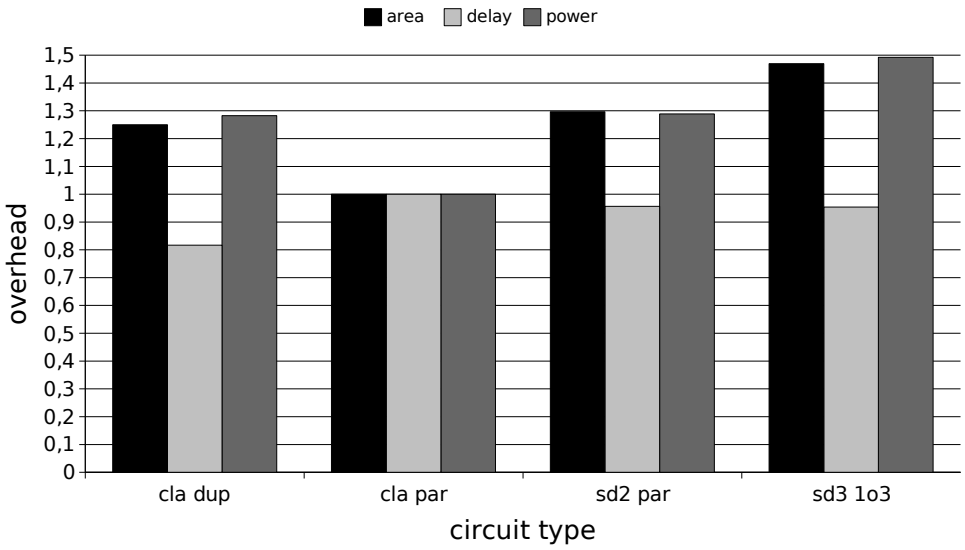


Figure 2.29: Overhead of fault-tolerant parallel filters, 8-bit data, 16-taps.

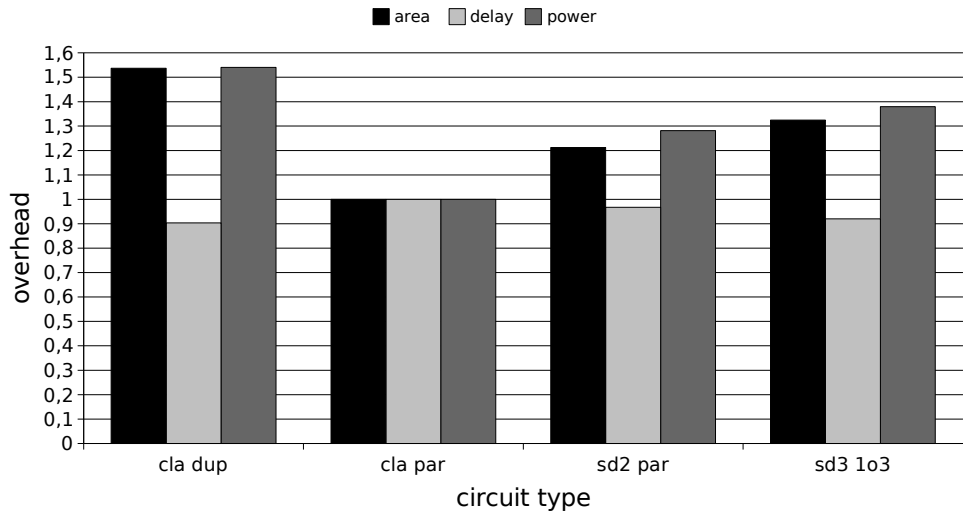


Figure 2.30: Overhead of fault-tolerant sequential filters, 8-bit data, 16-taps.

specific descriptions (as the ones made on the fault-tolerant library), the huge performance advantage of them can be traded off by some reduction on area and power consumption, representing the better option for fault-tolerant filter tap implementation.

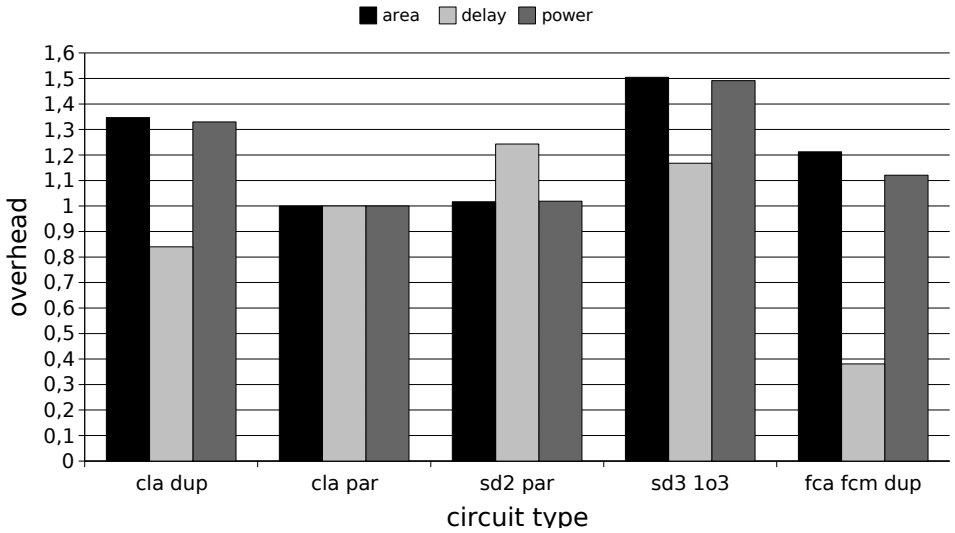


Figure 2.31: Overhead of fault-tolerant sequential filter taps.

The simplified analysis presented in the current section is focused on the traditional characteristics of logic circuits, i.e., area, delay and power, and the reliability aspect of each of the basic blocks can introduce a new understanding on what is the best option for each application. The next chapter presents the approaches that will be implemented to evaluate the reliability of the fault-tolerant block in the current library.

Chapter 3

Reliability analysis

3.1 Introduction

Back in the first days of semiconductor industry, simple tests assuring compliance of components and systems with specifications were considered enough to guarantee high quality products. With the continuous increase in the complexity of systems and the reduction in the profit margins, other factors have become as important as defect-free manufacturing, like reliability, maintainability and availability. It is well known that the cost of a defect increases by a factor of 10 according to where/when in the product chain the defect is found [71], i.e., chip level, board level, system level, user operation. With these values in mind, it is clear that reliability analysis in an earlier design step is becoming fundamental for product profitability.

Reliability analysis of electronic components is concerned with reliability prediction and reliability assessment. These two aspects of reliability analysis are equally important in the design process, since reliability assessment enables validation and refinements in the reliability models used for reliability prediction. The focus of the current work is on reliability prediction. Considering the bathtub curve presented in the previous chapter, the interest of the work is targeted to the useful life period, where fault occurrence is highly related to random nature sources. This eliminates the reliability analysis concerning infant mortality, that is related to manufacturing issues, and reliability analysis concerning wear-out mechanisms like electromigration, hot carriers, time-dependent dielectric breakdown, among others.

Even considering these restrictions in the scope of the analysis, different aspects of it can be focused, leading to different problems being treated under the same subject. The next section presents a revue in several works concerning reliability analysis of logic circuits, detailing their objectives, advantages and drawbacks and allowing a characterization of the proposed solutions and a definition of the focus of the current work.

3.2 Prior work

The book of Birolini [15] details all the aspects related to reliability, maintainability and availability analysis, from the point of view of modeling circuits according to the reliability of their components. The work is complete and treats specific aspects of electronic components but the approach is focused to classical reliability models, where systems can be characterized by series and parallel structures. The work does not focus on transient faults

and logic circuits but is mentioned here because details important aspects of reliability analysis and failure rate assessment.

The work of Ogus [72] focuses on the evaluation of the signal reliability, in contrast with the functional reliability. It proposes the functional equivalence class method for evaluating the probability of a correct output in a logic circuit. The classification of equivalent classes implies an exhaustive characterization of the state space of the circuits, leading to an intractable complexity. Another method proposed in the same work concerns a probabilistic model, where the probability of a correct output of a fault-prone circuit is computed as the signal reliability. Fig. 3.1 shows the general scheme of the probabilistic model, applied to a NAND gate (G).

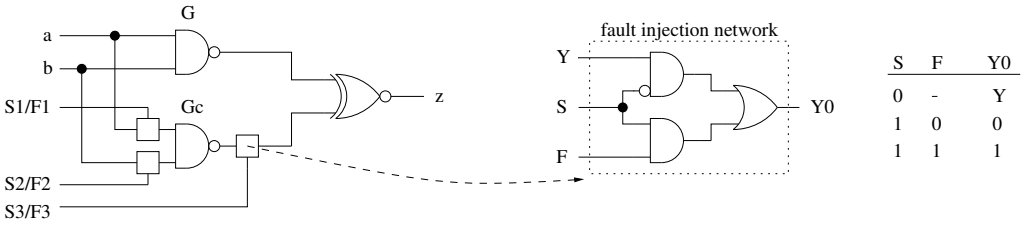


Figure 3.1: Probabilistic model applied to a NAND gate.

As shown in the figure, fault injection networks (FINs) are associated with the wires of the replica circuit (Gc), and the faults are injected to evaluate the reliability. The XNOR gate indicates when the fault-prone and the fault-free circuits have the same output value and the signal reliability of G is evaluated as the probability of z being 1, i.e., $R(G) = P(z = 1)$. The probabilistic model does not takes into account the individual contribution of circuit gates and computes the reliability by considering the circuit as a monolithic block.

The probabilistic model and the equivalent class method are targeted to stuck-at faults, diverging from the bit-flip fault model of interest for transient faults.

Stuck-at faults are also concerned in the work of Dokouzgiannis and Kontoleon [73], that models reliability by a circuit equivalent graph (CEG). The work determines the dominant fault vectors for some basic logic circuits and associates these full vectors to a gate equivalent graph (GEG). The GEGs of all the gates are interconnected according to the topology of the circuit and a path tracing procedure determines all the fault vectors related with correct functions, that contribute to the reliability of the circuit. The proposed methodology has been presented to single-output circuits and claimed to work with multiple output ones, but no benchmark results concerning circuit reliability were presented in the work. The use of stuck-at fault vectors allows the computation of dominant fault vectors but this concept is not compatible with the bit-flip fault model.

The work of Bogliolo et al. [74] proposes symbolic simulation to evaluate the reliability of fault-tolerant circuits. The work uses binary decision diagrams (BDDs) to model the original circuit and the fault-prone circuit, considering stuck-at faults. The method is limited by the number of nodes of the BDDs that can be handled adequately (10,000 nodes). As an alternative, an incremental evaluation is proposed, where the target circuit is partitioned in two sections that has the reliability incrementally computed. Further approximations are possible by modeling the circuit as independent sets, represented by smaller BDDs. The work presents some results on the reliability of quadded circuits, but no references are made concerning the error associated with the cuts and the approximations

applied to the target circuits.

As referred in the previous chapter, the reliability of a logic circuit is dependent on three masking mechanisms, i.e., logical, electrical and temporal. Accounting for the joint effect of the three effects is computationally intractable for practical circuits, and so, reliability analysis is dependent on model simplification. In section 2.4, the increase in the soft error rate of logic circuits has been discussed according to some fault models presented in the works of Hazucha and Svensson [47], Shivakumar et al. [3] and Seifert et al. [4]. These works have studied the reliability analysis from the point of view of electrical and temporal masking, considering faults that occur in a sensitized logic path. This approach is important to prove that the fault rate increases with circuit scaling but does not give a precise idea of the reliability of the circuits.

The works of Omaña et al. [75], Zhang et al. [76], Dhillon et al. [69] and Zivanov and Marculescu propose similar approaches for computing the probability of electrical masking as well as taking into account some degree of logical masking. Some details of these works follow.

In [75], Omaña et al. show that the probability of propagation of a glitch through a logic gate can be accurately modeled by the gate delay, and that the sensitized path can be modeled by a chain of inverters. The model is related with $\alpha-$particle strikes and is targeted to determine the susceptibility of combinational circuits to these transient faults. This susceptibility is calculated from a given node in the circuit to a given output and according to a specific input pattern. Despite the good correlation of the model with HSPICE simulations (evaluated as 90%), the extremely restricted scope of the analysis is not useful for practical purposes.

The work of Zhang et al. [76] also characterizes the propagation of a particle strike on the sensitive regions of gates in a methodology called FASER. This characterization is made according to SPICE simulations, with varying pulse widths, amplitudes and input patterns, what they call biasing conditions. According to their evaluations, considering worst-case conditions instead of biasing conditions can overestimate pulse propagation from 1.5× to 4×. The transfer functions determined with the SPICE models are stored in look-up tables. To account for logical masking the work uses BDDs, a *static* BDD being associated with a fault-free circuit and an *event* BDD being associate with particle strikes. Fig. 3.2 shows an example of an event BDD generated after a particle striking. Times $t0$, $t1$ and voltage V are obtained in the look-up table.

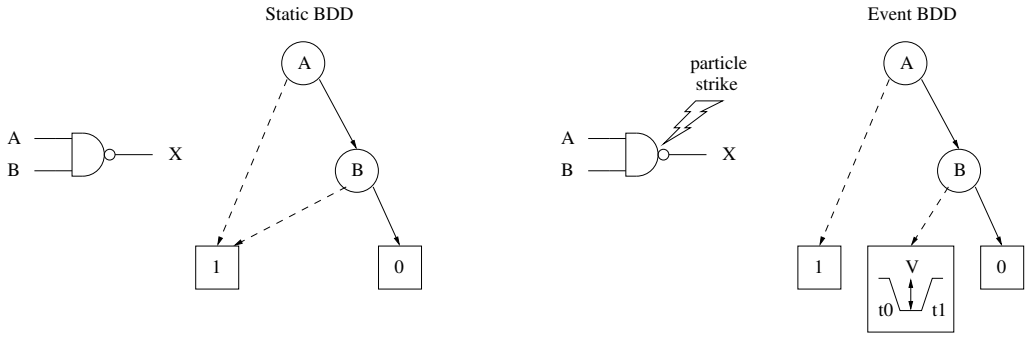


Figure 3.2: Particle strike on a NAND gate modeled by an event BDD.

The temporal masking is also taken into account by the proposed model, as in (3.1), where PL is the latching probability of a transient pulse, w is the latching window of the

latch, pw is the width of the transient pulse and T_{clk} is the clock period.

$$PL = \frac{pw - w}{T_{clk}} \tag{3.1}$$

To allow the application of the proposed analysis to practical circuits the authors implement a partition of the BDDs representing the circuits and thus, signal correlations among signals in different partitions are ignored. The FASER tool computes the probability of a transient fault being latched by considering single particle strike (single fault) at each node of the circuit. The work claims a $90,000\times$ speed-up over SPICE simulation and 12% average error. No reference is made whether the computing time includes the BDD generation and the results consider partition sizes of 15 gates, with a fast increase of computing time for increments on the partition size.

Another analysis based on symbolic techniques is proposed by Zivanov and Marculescu [77]. The probability of glitch propagation of each logic gate follows the same model presented by Omaña et al. [75]. To account for logical masking, BDDs and ADDs (algebraic decision diagrams) are constructed for each gate. The probability of fault propagation through a logic path from gate G_i to output F_j is computed by building all ADDs in the referred path. To evaluate the reliability of the circuit, two analysis are made, i.e., the probability of output F_j failing given faults at gates on its fanin cone, and the probability of error on the outputs given that gate G_i fails. The input patterns for the analysis are randomly generated. The proposed analysis is applied to some benchmark circuits and shows good results and confidence with HSPICE but no reference is made about the time necessary to build the BDDs and ADDs and the number of random inputs generated to evaluate logical masking. The overall reliability of the circuit cannot be computed since the values are related to single outputs.

Rao et al. [78] propose the use of parameterized descriptors to model the propagation probability of neutron strikes through logic gates. The work starts by discussing the problems of using BDDs and simplified glitch propagation models as the ones in the previous works of [76] and [77]. Rao et al. model the pulse generated after a neutron strike by the Weibull probability density function, instead of the trapezoidal models used on the cited works. The parameterized single event transient (SET) descriptors are generated by a SPICE characterization of the cells from the target library and some candidate waveforms are selected to represent a large range of particle energies, cell size, cell state, supply voltage, etc. The analysis runs by propagating the SET descriptors to the outputs of the circuit, where temporal masking can be computed. Logical masking is accounted for by workload simulation of 500,000 input vectors. The application of the proposed method to several benchmark circuits show interesting results for computing time but that does not include the workload simulation. The correlation with SPICE simulations is found to be 16.1% on average but this is related with just one random input vector. The work presents an interesting comparison of the SER of different output bits.

Other works, like the ones discussed in the following pages, focus on a reliability analysis targeted to the logical and temporal masking properties of the circuits. Despite the different number of methods and approaches, some similarities can be observed, meaning a different modeling of the same properties.

A straightforward approach for computing the reliability of logic circuits is presented in the works of Patel et al. [16] and krishnaswamy et al. [17, 18], by means of the probabilistic transfer matrices (PTMs). The PTM of a gate is a matrix that models the signal probability at the output according to the probability of input signals and gate

reliability. The method also defines the fault-free behavior of a gate by means of the ideal transfer matrix (ITM). Fig. 3.3 shows the ITM and PTM matrices of an OR logic gate, where q is the reliability of the gate and $1-q$ is the error probability of the gate.

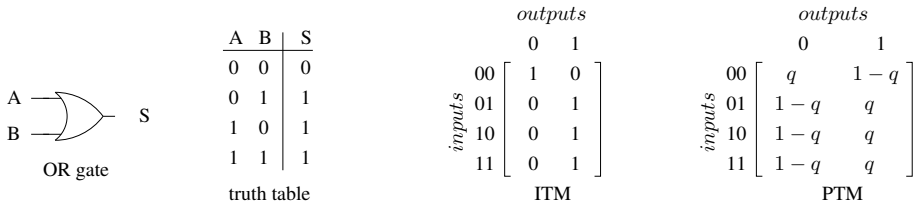


Figure 3.3: Example of ITM and PTM models for an OR logic gate.

The reliability of a logic circuit can be determined by its PTM model and its ITM model. The PTM of a combinational circuit can be determined by the adequate composition of the PTMs of its gates. The circuit must be organized in logic levels, the PTM of each level being determined by the Kronecker product (tensor product, $\otimes$) of the PTMs of the elements in that level. The PTM of the circuit is computed by multiplying the PTMs of all different levels (in topological order). Fig. 3.4 shows the computation of the PTM of the circuit C17 from the ISCAS'85 benchmark. The PTM of level n is identified as Ln and the *wiring* function is an ITM model of the wiring connections that takes into account signal correlations in the presence of reconvergent fanouts.

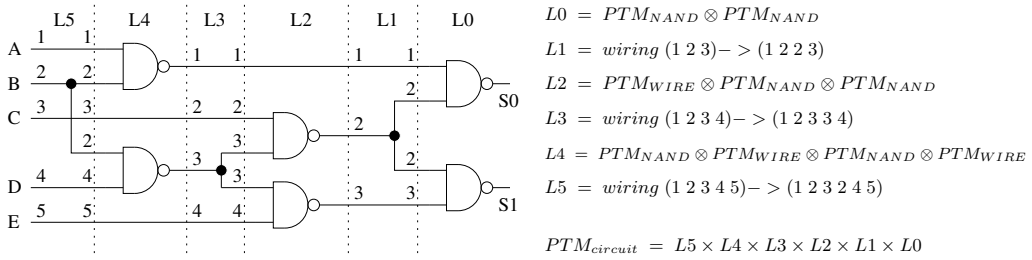


Figure 3.4: Computing the PTM of the circuit C17.

The ITM of the wiring levels represents the probability of occurrence of a given output vector (level output) considering a specified input vector (level input). An example of the *wiring* function for level $L1$ can be seen on Fig. 3.5, where the invalid outputs are indicated, according to the output values of level wire number 2.

The PTM approach allows the computation of the logical masking effect on the reliability of a circuit and can be also targeted to compute the electrical masking, concerning multiple simultaneous faults. The main advantages of the PTM model are the straightforward implementation, possibility of input pattern dependent reliability modeling of logic gates, exact logical masking modeling and signal correlations computation. The main problem is the scalability of the method. Since the PTM of a circuit is a monolithic matrix representation of input and output pattern probabilities, the size of this matrix increases exponentially with the number of inputs and outputs, what leads to intractable computing times and memory storage needs for practical circuits.

One approach for reducing the scalability problem is the use of algebraic decision diagrams (ADDs), based on the QuIDDPro library [17]. Despite the reduction on the size of the problem, the gains are still not enough for the computation of large circuits and the results presented are not the same as the ones calculated by the original PTM model,

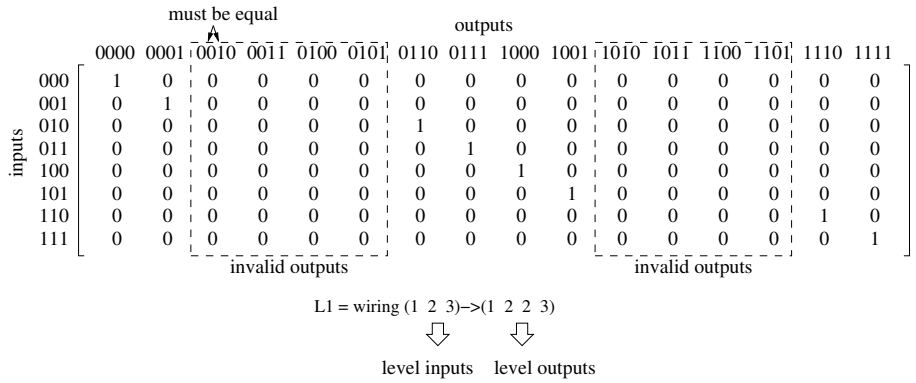


Figure 3.5: Example of the *wiring* mapping function.

indicating some problems in the use of the library package. A recent work from the same authors [79] proposes the partition of larger circuits into smaller blocks, ignoring signal correlations among blocks and also proposes the use of input vector sampling to reduce the complexity of the problem.

Despite the scalability problems of the PTM model, the proposal is considered important in the scope of the present work, and PTM evaluation tools have been implemented in the current work to exploit the characteristic of the method and verify the practical limits of the approach. These tools and more details on the PTM model are presented in section 3.3.

Rejimon and Bhanja propose the use of Bayesian Networks (BN) to compute the error detection probability in logic circuits [80, 81]. In the proposed model, each gate is represented by its conditional probability table (CPT) that models the output states according to input states. The overall joint probability model is determined by networking these individual gate CPTs, generating a directed acyclic graph (DAG). The error probability is computed by relating a fault-free network with the fault-prone one, as shown in Fig. 3.6, where Z_n are inputs, X_m are internal signals and Y_k are the outputs. The equivalent signals in the error-prone circuit are referred as X_m^e and Y_k^e.

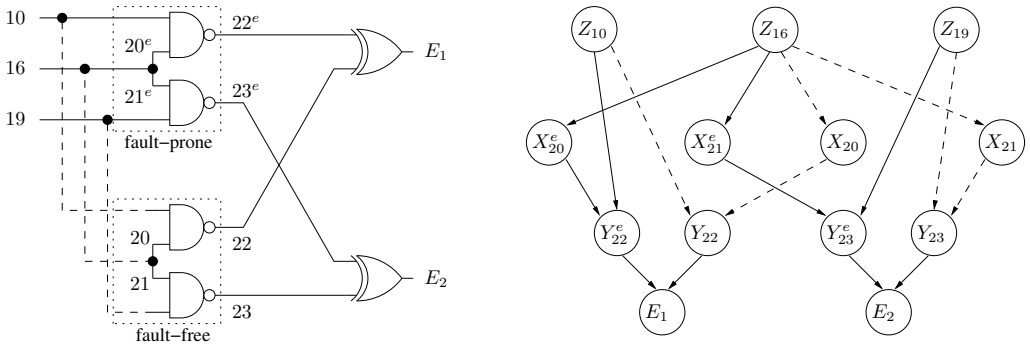


Figure 3.6: Example of Bayesian network representation.

This representation of the underlying error model is called a logic induced fault encoded directed acyclic graph (LIFE-DAG). With the proposed scheme, the probability of error at the ith output can be expressed as in (3.2).

$$P(E_i = 1) = P(Y_i^e \oplus Y_i = 1) \tag{3.2}$$

The authors claim that the proposed approach inherently models signal correlations and that the graphical model is the minimal optimally factorized representation of the joint probability functions. The results show three orders of magnitude improvement over the PTM approach but the reliability results for the small C17 circuit present a 26% difference from the exact value, indicating some problem with the implementation. A more recent work from the same authors [82] presents two stochastic sampling algorithms, i.e., probabilistic logic sampling (PLS) and evidence pre-propagated importance sampling (EPIS). The work presents results for large benchmark circuits obtained with the sampling schemes and 1000 and 3000 samples, what may not be that representative given the size of the target circuits.

The probabilistic gate model approach (PGM), proposed by Han et al. [83], uses the PGM of individual gates to iteratively build the reliability of the outputs. The PGM of an individual gate can be determined as in (3.3), where p_i is the sum of the minterms of inputs that produce a 1 and ε is the failure probability.

$$X_i = [p_i \;\; 1 - p_i] \times \begin{bmatrix} 1-\varepsilon \\ \varepsilon \end{bmatrix} \tag{3.3}$$

Signal correlations must be explicitly treated by treating the circuit as having different levels and computing the joint probabilities of each level, leading to a PTM-like modeling. The work does not detail the algorithm and the results are not exact. The method presents limitations on circuit's size and signal correlations.

The work of Asadi and Tahoori [84] proposes the error propagation probability (EPP) algorithm. The objective of the work is to compute the error probability at specific outputs by propagating the error probabilities through the gates of the circuit. The algorithm starts by determining the *on-path* signals (and on-path gates) from node n_i to primary output PO_j, what is achieved by executing the forward and backward depth-first search (DFS) algorithm. Fig. 3.7 shows an example of on-path and off-path signals in a logic circuit.

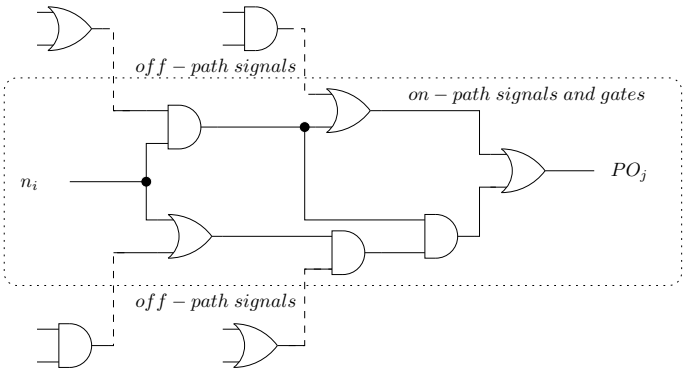


Figure 3.7: On-path signals and gates in a logic circuit.

Once the signal path is defined the error probability can be propagated to the output according to analytical expressions presented in the work, each gate having its own set of propagation expressions. The defined expressions take into account reconvergent signals but only considering the specific PO_j output. The signal probabilities of off-path signals must be determined by some adequate method, like simulation or BDD tools.

The work presents interesting performance results but for single-outputs and with no signal correlation correction among different outputs. Despite the good results, the execution time presented does not take into account computing signal probabilities, what can represent an important difference.

The work of Bhaduri et al. [85] introduces the probabilistic model checking (PMC), that is based on discrete time Markov chains (DTMCs) and the related transition probability matrices. Fig. 3.8 shows the DTMC of a NAND gate, where s is the level of the DTMC. The modeling of circuit states considering DTMCs suffers from exponential increase in the number of states according to the size of the circuits and the authors use a probabilistic models checker (PRISM) to analyze the DTMCs. Given the sparse nature of transition probability matrices, PRISM uses multi-terminal binary decision diagrams (MTBDDs) to represent them and to allow scalability to larger circuits.

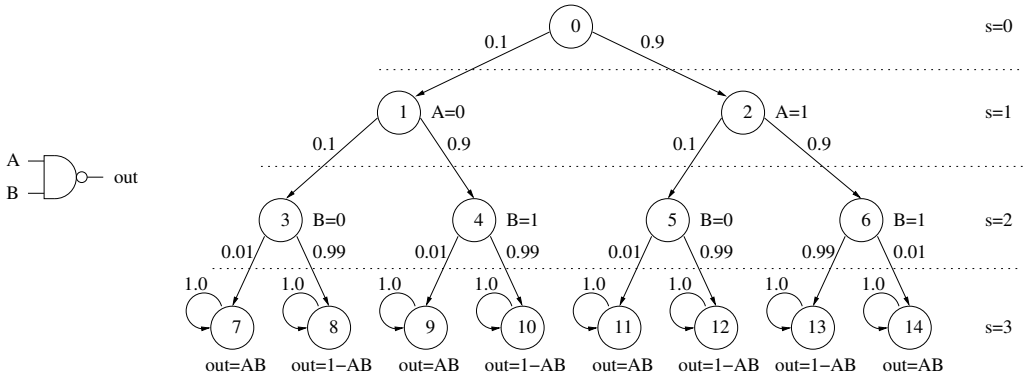


Figure 3.8: DTMC of a NAND gate.

The reliability of the circuit is computed by organizing the target circuit into different levels and iteratively building the DTMCs of each circuit level. Once the joint output probabilities for the current level has been computed the algorithm uses these probabilities as the input probabilities for the next level. The authors claim that the method fold space into time, reducing the problems of matrix storage but the limitation is in fact in the size of the levels themselves, and not only on the number of levels. The proposed iterative methodology is claimed to be adaptable to the PTM and PGM analysis models.

The work introduces the scalable, extensible tool for reliability analysis, called SETRA, that allows the proposed analysis to be used in an integrated design flow. The SETRA tool uses an external EDIF parser engine to translate the circuit into a Microarchitectural XML (MXML) representation, that is further translated to a C++ circuit structure for topological extraction. After this extraction the algorithm runs the iterative reliability analysis described before. Despite the fact that the results presented for large circuits represent fast solutions, no comparative values concerning other approaches are presented, nor the indication of what is included in the running time. The results presented for small circuits indicate that the method computes the probability of the outputs being 1 (0) but not the reliability itself, as indicated in Fig. 3.9. No references to signal correlation heuristics are detailed in the work.

The work of Choudhury and Mohanram [86] presents two methods for reliability modeling, one based on the observability and the other based on a single-pass error probability propagation. The work proposes the use of the observability metric to compute the reliability of the circuit. The intuitive idea is that a fault that has to propagate to several

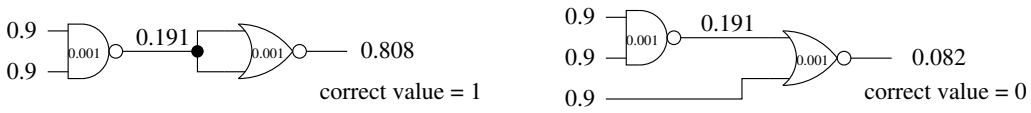


Figure 3.9: DTMC of a NAND gate.

Input vector	Weight	Weighted $0 \rightarrow 1$ input error
00	W_{00}	$\Pr(i_{0\rightarrow1})\Pr(j_{0\rightarrow1})W_{00}$
01	W_{01}	$\Pr(i_{0\rightarrow1})(1-Pr(j_{1\rightarrow0}))W_{01}$
10	W_{10}	$(1-Pr(i_{1\rightarrow0}))\Pr(j_{0\rightarrow1})W_{10}$
Total	W(0)	PW(0)
Input vector	Weight	Weighted $1 \rightarrow 0$ input error
11	W_{11}	$(\Pr(i_{1\rightarrow0})+\Pr(j_{1\rightarrow0})-\Pr(i_{1\rightarrow0})\Pr(j_{1\rightarrow0}))W_{11}$
Total	W(1)	PW(1)

Table 3.1: AND gate weighted input error vector.

levels of logic has a higher probability of being logically masked than a fault closer to the outputs. This probability can be determined by computing the observability of the signals, a metric traditionally related to the domain of test. The closed-form expression for the proposed model is presented in (3.4), where $\delta_y(\vec{\epsilon})$ is the probability of error, Ω is the set of all gates, ϵ_i the error probability of the ith gate and o_i the observability of this gate. The model is focused on single output circuits.

$$\delta_y(\vec{\epsilon}) = \frac{1}{2}\left(1 - \prod_{i\in\Omega}(1-2\epsilon_i o_i)\right) \tag{3.4}$$

The observability metrics on the proposed work are computed by BDDs. There are no specific results presented for this proposed approach and the authors claim that it is limited to small circuits and small values of error probability since no signal correlation effects are taken into account.

The single-pass reliability algorithm presented in the work of Choudhury and Mohanram is based on the idea that an error at the output of a gate can be generated by an error on the gate itself or can be propagated by the gate from its inputs. This way, the error probability in a circuit can be propagated through all the gates, and the output reliability can be evaluated considering the expected output and its error probability. This concept allows the reduction of the complexity of the algorithm to $O(n)$, with n being the number of gates in the circuit.

The expressions for the susceptibility of a gate to errors in its inputs is shown in table 3.1, where an AND gate is analyzed. The expressions represent the $0 \rightarrow 1$ ($\Pr(i_{0\rightarrow1})$, $\Pr(j_{0\rightarrow1})$) and $1 \rightarrow 0$ ($\Pr(i_{1\rightarrow0})$, $\Pr(j_{1\rightarrow0})$) input error probabilities. A $0 \rightarrow 1$ error probability considers that the error-free value is 0. The input error probabilities are combined with the input weight vector W_{ij} to determine the output error probability $Pr(g)$ due to input errors only, as shown in (3.5) and in (3.6). The input weight vector W_{ij} is the probability of the joint occurrence of the ij pattern at the fault-free circuit and can be computed by any known method like BDD-based symbolic techniques.

$$Pr(g_{0\rightarrow1}|g\ does\ not\ fail) = \frac{PW(0)}{W(0)} \tag{3.5}$$

$$Pr(g_{1\to 0}|g\ does\ not\ fail) = \frac{PW(1)}{W(1)} \tag{3.6}$$

The cumulative effect of an error on the gate itself is taken into account as in (3.7) and (3.8).

$$Pr(g_{0\to 1}) = (1-\epsilon)\left(\frac{PW(0)}{W(0)}\right) + \epsilon\left(1 - \frac{PW(0)}{W(0)}\right) \tag{3.7}$$

$$Pr(g_{1\to 0}) = (1-\epsilon)\left(\frac{PW(1)}{W(1)}\right) + \epsilon\left(1 - \frac{PW(1)}{W(1)}\right) \tag{3.8}$$

Given the output error probabilities, the *unreliability* of the output can be computed as in (3.9).

$$\delta_y = Pr(y=0)Pr(y_{0\to 1}) + Pr(y=1)Pr(y_{1\to 0}) \tag{3.9}$$

Fig. 3.10 shows an example of the single-pass analysis of a logic circuit. The W_{ij} components must be computed previously by an adequate tool.

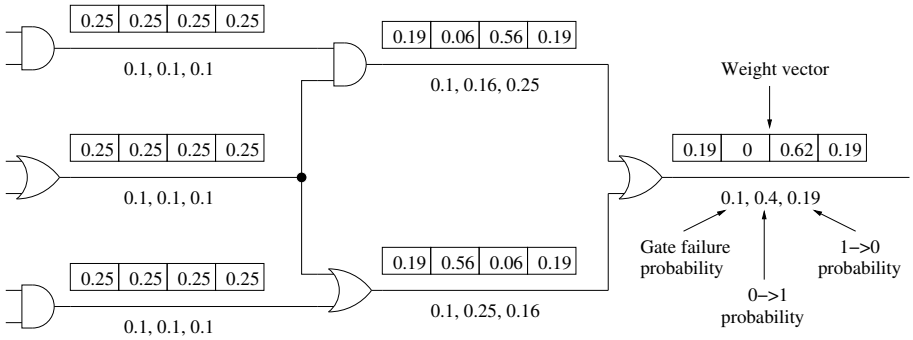


Figure 3.10: Example of the single-pass analysis in a logic circuit.

The main drawback of the single-pass methodology is related with reconvergent fanout signals. All of the precedent equations take into account an independence of the probabilities of the signals and a fanout interconnection means signals correlation. To deal with this problem the author applies a theory of correlation coefficients to correct the weighted input error expressions presented at the third column of table 3.1.

The results presented in the work show very good performance but some remarks can be made. First of all, the O(n) complexity of the algorithm does not consider the BDD running time and memory occupation, factors that can be representative for large circuits. Secondly, the results are for single-output error probability of the proposed benchmark circuits.

3.2.1 Comments

As can be seen in the works presented in the current section, there are many approaches to evaluate the reliability of combinational circuits. Some of the authors propose the modeling of all the masking factors, while others concentrate on specific factors.

Computing the electrical masking effect is achieved with the soft error rate (SER) modeling of the logic gates at the transistor-level and the attenuation modeling of the transient

glitches through the gates in the path to the primary outputs. Therefore, electrical masking is composed by two factors, the probability that a particle strike generates an electrical glitch at the output of the gate and the probability that this electrical glitch propagates to a primary output. The first factor, i.e., the SER of the gate, corresponds to the failure rate λ of the gate, and can be used to compute its individual reliability, as in (3.10).

$$R(t) = e^{-\lambda t} \tag{3.10}$$

The probability of attenuation of a transient glitch through a logic chain can be considered as a small contribution to the reliability of a circuit, related with the probability of occurrence of a sensitized path. The instinctive notion of observability of a logic signal leads to the assumption of a small probability of glitch attenuation in a highly observable node, i.e., nodes that are close to the outputs of the circuit, and of a higher probability of glitch propagation through a smaller logic chain. This statement is confirmed by the work of Baze and Buchner [87], that relates the length of a logic chain with the attenuation of a logic glitch.

The temporal masking effect is independent of the electrical and logical ones, representing a sample probability associated with operating frequency and latch timing characteristics, being easily computed by the expression in (3.1), as defined in the work of Zhang et al. [76].

These considerations confirm the fact that the logical masking is the most critical masking effect to be determined in combinational logic circuits, and a fundamental metric for an exact signal attenuation computation. By computing the signal attenuation factor based on simplified logical masking models, one can get a non negligible error. Furthermore, the glitch modeling of particle strikes is only one component of the glitch sources, as detailed in section 2.4, that have a cumulative effect over the same logical masking property.

As stated before, the main objective of the current work is to model the logical masking property of combinational logic circuit. For this purpose, we consider that most of the electrical masking effect of a circuit is implicitly modeled by the reliability of the gates. Instead of computing the probability of attenuation of a fault through a logic chain, we consider that the reliability of a gate corresponds to the probability of occurrence of a fault with enough strength to propagate through all the logic chain.

Among the works that model the reliability according to the logical masking property, a comment is necessary. It is worth mentioning that the logical masking computation problem of a circuit under transient faults is similar to the boolean satisfiability problem, which is a NP-complete one. This way, any model for computing the exact value of logical masking is intractable, and heuristic solutions are necessary. From this point of view, all of the referred works are equivalent, recurring to simplified models of logical masking. Some of the simplifications lead to more accurate results but generally at the price of an execution time penalty. Given the presented prior works on reliability analysis and given the objective of the current work, i.e., developing a fast and accurate method for reliability analysis that could be easily integrated into a design flow framework, some desired properties in a reliability analysis tool can be defined.

First of all, the algorithm must be simple, of straightforward application, to allow an easy integration into an automated design flow. The model must be accurate, as far as possible, to precisely evaluate the reliability of the circuits at final design steps. A tunable accuracy would be interesting, in a tradeoff with executing time. The possibility of taking into account different gate reliabilities is important, to allow the evaluation of

fault prevention techniques. Finally, the possibility of defining different gate reliabilities according to gate input patterns would allow a more realistic modeling of gate behaviors.

The need for a straightforward application is in contrast with the use of graph based data representations, like BDDs, ADDs, MTBDDs and LIFE-DAGs, whose generation and optimization represent a time and memory consuming task in itself. These types of data representation are adequate to represent static circuits but in the case of an automated reliability-aware design flow, the topology of the circuit may change after each optimization step leading to a re-built of the corresponding graphs. These data representations are more adequate to model the final version of the circuits, pointing to an understanding that different models should be targeted for different stages of circuit development.

Next sections introduce the methods developed and implemented in the current work, that are bounded by the desired properties defined here.

3.3 Probabilistic transfer matrices

The first method implemented along the current work was the method of the probabilistic transfer matrices (PTMs), given its desirable characteristics of straightforward application and individual gate and input pattern reliability modeling. The idea behind this implementation was to interface the method with the design flow of digital circuits, something that was not developed on the original PTM proposal. Optimizations of the algorithm were also the focus of the work. The present section details the PTM approach and the implementation characteristics.

The method of probabilistic transfer matrices was introduced by Patel et al. [16] based on the work of Levin [88], and was further explored in the works of Krishnaswamy et al. [17, 18]. The main idea is the determination of a matrix that correlates the inputs and outputs of a logic circuit considering its topology and the individual reliability of its gates. This correlation matrix is called a *probabilistic transfer matrix* (PTM) and must be built by the composition of basic gate's PTMs and the structure of the circuit. Given the PTM of the circuit, it can be used to determine dependability metrics like overall probability of errors, signal reliability, signal observability, among others.

The PTM of a circuit is a monolithic matrix representation of the probability of occurrence of all input and output combinations. For a circuit with n inputs and m outputs, its PTM is a $2^n \times 2^m$ matrix.

The PTM of an individual gate is derived from its truth table, as well as its *ideal transfer matrix* (ITM). The ITM is the matrix correlating inputs and outputs of the fault-free gate or circuit. Fig. 3.11 presents the modeling of the ITM and PTM for two logic gates.

As seen on the figure, the matrices map inputs to outputs according to a certain probability. In the ITMs the probabilities are 100% (value **1** in the matrix) or 0% (value **0** in the matrix). The ITM_{OR}, for example, express 100% probability of an output z=0, given an input ab=00. Similarly, the probability of an output z=0, given an input ab=11 is 0%. In the case of PTMs, the probability of an output given a certain input is defined by a value that reflects the reliability of the gate or cell. In the PTMs in Fig. 3.11, the reliability is represented by q and the unreliability is referred as $1 - q$. If different values of probabilities are associated to different inputs, they can be modeled by the PTM of the circuit. The sum of output probabilities (values in a PTM row) for a given input must always be **1**.

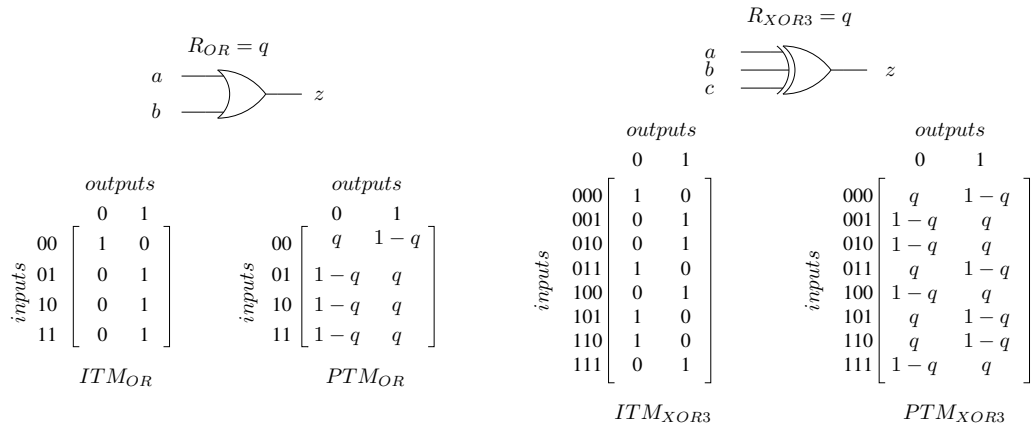


Figure 3.11: Examples of gate ITM and PTM modeling.

In mathematical notation, the (i,j)th entry of a PTM matrix is the probability of occurrence of value j at the output given the value i is applied at the input, and this probability is represented by $p(j|i)$. In Fig. 3.11, for example, the probability of an output **1** in the OR gate given an input **11** is $p(1|11) = q$.

For wire, wire permutation and fanout connection modeling, only ITMs are considered. Some examples of fanout connections and wire permutation ITMs can be seen in Fig. 3.12. Fanout connection ITMs are fundamental to take into account signal correlations, allowing exact joint probabilities computation.

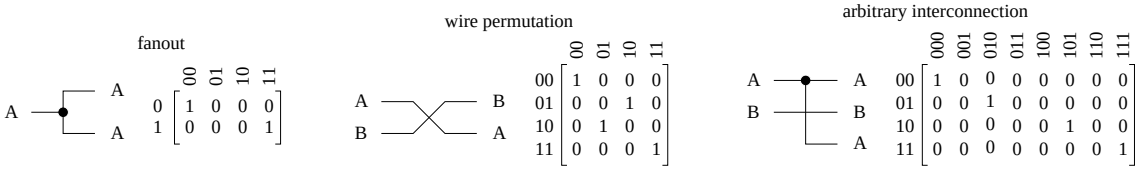


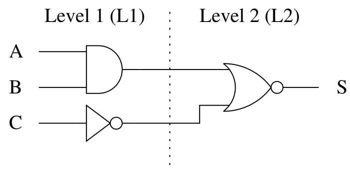
Figure 3.12: Examples of ITMs of fanout, wire permutations and connections.

The fault probability of a gate in a standard cell library should be modeled by the library developer, similarly to other operation parameters like speed and power consumption. As for now, no reliability figures are available for individual cells in commercial libraries and a characterization of the cells through SPICE simulations is necessary for determining these probabilities.

The generation of a circuit PTM involves the combination of gate PTMs and interconnection ITMs in series and parallel structures. To generate the PTM of elements connected in series, the matrix multiplication of element's PTMs must be executed. To generate the PTM of parallel elements in the same logic level, a tensor product (Kronecker product, operator $\otimes$) of the PTMs of these elements must be executed. These operations are presented in the figures that follow. Fig. 3.13 shows a small circuit with two logic levels, and the computation of the PTM of a circuit level by *tensoring* the PTM of its compound gates. To simplify the expressions, the unreliability of the gates is represented by $\bar{q}$.

Fig. 3.14 shows the determination of the PTM of the circuit by multiplying the PTMs of different levels.

As presented in Fig. 3.13, the circuit must be organized in different levels that are combined as series components (by matrix multiplication) and each level is composed



$$PTM_{NOT} = \begin{bmatrix} \bar{q} & q \\ q & \bar{q} \end{bmatrix} \quad PTM_{AND} = \begin{bmatrix} q & \bar{q} \\ q & \bar{q} \\ q & \bar{q} \\ \bar{q} & q \end{bmatrix} \quad PTM_{NOR} = \begin{bmatrix} \bar{q} & q \\ q & \bar{q} \\ q & \bar{q} \\ q & \bar{q} \end{bmatrix}$$

$$PTM_{L1} = PTM_{AND} \otimes PTM_{NOT} = \begin{bmatrix} q \cdot \begin{bmatrix} \bar{q} & q \\ q & \bar{q} \end{bmatrix} & \bar{q} \cdot \begin{bmatrix} \bar{q} & q \\ q & \bar{q} \end{bmatrix} \\ q \cdot \begin{bmatrix} \bar{q} & q \\ q & \bar{q} \end{bmatrix} & \bar{q} \cdot \begin{bmatrix} \bar{q} & q \\ q & \bar{q} \end{bmatrix} \\ q \cdot \begin{bmatrix} \bar{q} & q \\ q & \bar{q} \end{bmatrix} & \bar{q} \cdot \begin{bmatrix} \bar{q} & q \\ q & \bar{q} \end{bmatrix} \\ \bar{q} \cdot \begin{bmatrix} \bar{q} & q \\ q & \bar{q} \end{bmatrix} & q \cdot \begin{bmatrix} \bar{q} & q \\ q & \bar{q} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \bar{q}q & q^2 & \bar{q}^2 & \bar{q}q \\ q^2 & \bar{q}q & \bar{q}q & \bar{q}^2 \\ \bar{q}q & q^2 & \bar{q}^2 & \bar{q}q \\ q^2 & \bar{q}q & \bar{q}q & \bar{q}^2 \\ \bar{q}q & q^2 & \bar{q}^2 & \bar{q}q \\ q^2 & \bar{q}q & \bar{q}q & \bar{q}^2 \\ \bar{q}^2 & \bar{q}q & \bar{q}q & q^2 \\ \bar{q}q & \bar{q}^2 & q^2 & \bar{q}q \end{bmatrix}$$

Figure 3.13: Example of circuit level PTM computation.

by individual gate's PTMs, tensor products or permutation matrices. To preserve the input/output matrix correlation, the same order of gate's tensoring must be respected along the different levels of the circuits. In the case of Fig. 3.13, the tensor product applied to level 1 (PTM_{L1}) is done considering a top-down sequence of gates, that implies input A as being the most significant input of the circuit. If the tensor product had been applied bottom-up, input C would be the most significant input. The same reasoning applies to multiple output circuits.

$$PTM_{CIR} = PTM_{L1} \cdot PTM_{NOR} = \begin{bmatrix} 2\bar{q}^2q + \bar{q}q^2 + q^3 & \bar{q}^2q + 2\bar{q}q^2 + \bar{q}^3 \\ \bar{q}^2q + 3\bar{q}q^2 & 2\bar{q}^2q + \bar{q}^3 + q^3 \\ 2\bar{q}^2q + \bar{q}q^2 + q^3 & \bar{q}^2q + 2\bar{q}q^2 + \bar{q}^3 \\ \bar{q}^2q + 3\bar{q}q^2 & 2\bar{q}^2q + \bar{q}^3 + q^3 \\ 2\bar{q}^2q + \bar{q}q^2 + q^3 & \bar{q}^2q + 2\bar{q}q^2 + \bar{q}^3 \\ \bar{q}^2q + 3\bar{q}q^2 & 2\bar{q}^2q + \bar{q}^3 + q^3 \\ 2\bar{q}q^2 + \bar{q}^3 + q^3 & 3\bar{q}^2q + \bar{q}q^2 \\ 2\bar{q}^2q + \bar{q}q^2 + q^3 & \bar{q}^2q + 2\bar{q}q^2 + \bar{q}^3 \end{bmatrix}$$

Figure 3.14: Example of circuit PTM determination.

Once the circuit PTM is available, its reliability can be calculated considering an input probability distribution and the circuit ideal transfer matrix. In the present case, reliability is considered as the probability of a correct output. Given a circuit e (consider the one presented in Fig. 3.13), with PTM PTM_e and ITM ITM_e, the reliability can be determined according to the expression in (3.11).

$$R_e = \sum_{ITM_e(i,j)=1} p(j|i)p(i) \tag{3.11}$$

The $p(i)$ element represents the probability of an input value i, and $p(j|i)$ is the $(i,j)th$ element in PTM_e representing the probability correlation of output j given an input i. The (i,j) elements to be considered are those where the respective ITM_e matrix has a value of $\mathbf{1}$. To determine the probability of an error at the output of the circuit, the (i,j) elements to be considered are those where the respective ITM_e matrix has a value of $\mathbf{0}$.

If all inputs have the same occurrence probability, the reliability follows expression 3.12, where n is the number of inputs.

$$R_e = \frac{1}{2^n} \sum_{ITM_e(i,j)=1} p(j|i) \tag{3.12}$$

Fig. 3.15 shows the expressions (shaded) that are taken into account for computing the reliability of the circuit of Fig. 3.13, as indicated by (3.12).

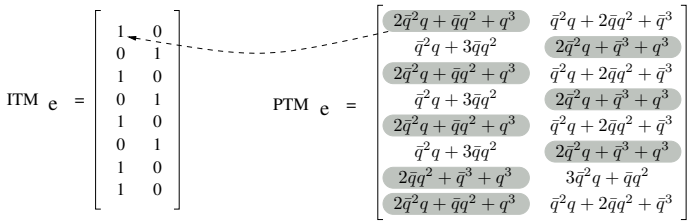


Figure 3.15: Computing reliability with the PTM and ITM models of the circuit.

Fig. 3.16 shows the circuit reliability (R) behavior, according to the reliability of its individual gates (q).

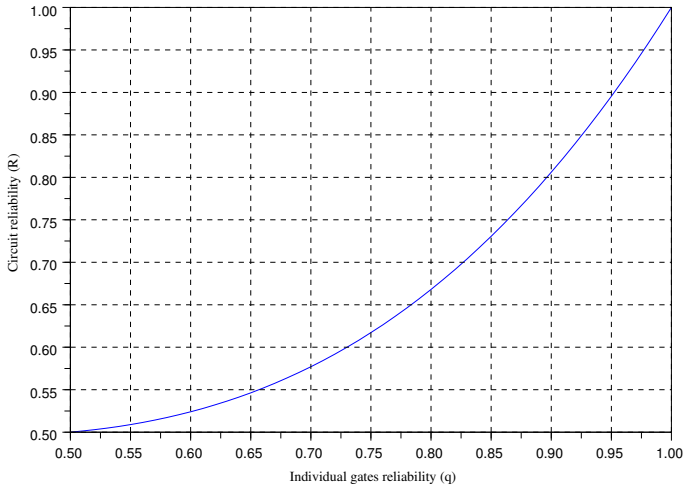


Figure 3.16: Reliability curve of the target circuit according to its gate's reliability.

The PTM method can be directly applied to model circuits with gates that have different error probabilities, or different input distributions. Permanent faults can also be modeled, as the examples shown at Fig. 3.17. The first PTM of the OR gate represents the modeling of a 10% probability of an output stuck-at **0** fault (**s-a-0**). The second PTM reflects a 7% probability of an stuck-at **1** (**s-a-1**) fault at the gate output.

Despite its straightforward application, some drawbacks in the PTM methodology can be observed. The first one is the exponential increase of matrices size according to the number of inputs and outputs in the circuit. This limits the processing times and demands a huge storage(memory) capacity, being useful only for small circuits. Another drawback is the need for a completely different circuit representation, that expresses the circuit as a sequence of levels with precise interconnection and component ordering.

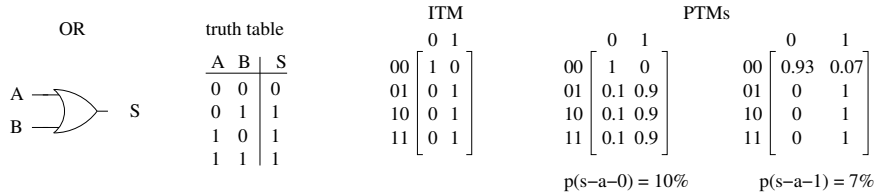


Figure 3.17: Stuck-at faults PTM modeling.

To explore the possibilities of the PTM methodology, some tools where developed in the current work to allow its integration to the design flow of integrated circuits and to optimize its execution. For this purpose, the target circuits must be considered as being issued from a synthesis tool, targeting the analysis to more realistic circuits.

3.4 The Probabilistic binomial model

This section presents the probabilistic binomial model (PBR) for reliability analysis, developed as an analytical approach to model the reliability of fault-prone logic circuits, and implemented by means of fault-simulation or fault-injection in a circuit emulation environment. The proposed analysis is capable of producing exact results or highly accurate ones for large circuits.

Let's consider a generic combinational logic circuit with an n-bit input vector $\vec{x}$ and an m-bit output vector $\vec{y}$, as shown in Fig. 3.18.

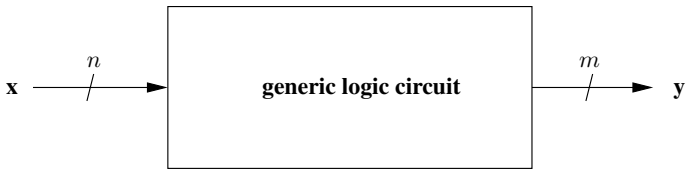


Figure 3.18: Generic combinational logic circuit.

If the gates in this circuit fail with a certain probability, the signal reliability R of such circuit, seen as a black box, can be determined as in (3.13), where $p(\vec{y} = correct|\vec{x}_i)$ represents the probability of occurrence of a correct output vector given the ith input vector $\vec{x}_i$ and $p(\vec{x}_i)$ represents the probability of occurrence of this input vector.

$$R_{cir} = \sum_{i=0}^{2^n-1} p(\vec{y} = correct|\vec{x}_i)p(\vec{x}_i) \tag{3.13}$$

To model the contribution of each gate of the generic circuit to the overall reliability, the probability of fault masking given all input and fault combinations must be determined. Let Γ be the set of all gates in the circuit (G gates), the set $\phi \subseteq \Gamma$ of gates that fail and the set $\gamma \subseteq \Gamma$ of gates that operate correctly. Consider the fault vector $\vec{f_j}$ as representing the jth fault vector, where each element of this vector represents the state of the corresponding gate in the circuit, $f(l) = 0$ meaning correct operation of gate l and $f(l) = 1$ meaning failure of gate l. Expression (3.14) models the reliability of the circuit according to the input and gate failure patterns.

$$R_{cir} = \sum_{j=0}^{2^G-1} \sum_{i=0}^{2^n-1} p(\vec{y} = correct|\vec{x}_i, \vec{f}_j) p(\vec{x}_i) p(\vec{f}_j) \tag{3.14}$$

Fig. 3.19 shows an illustrative representation of a fault-prone logic circuit, where the $f(l)$ signals indicate the operation state of the respective gates. A $f(l) = 1$ value means the occurrence of an error at the concerned gate, what results in a bit-flip error at the gate output signal.

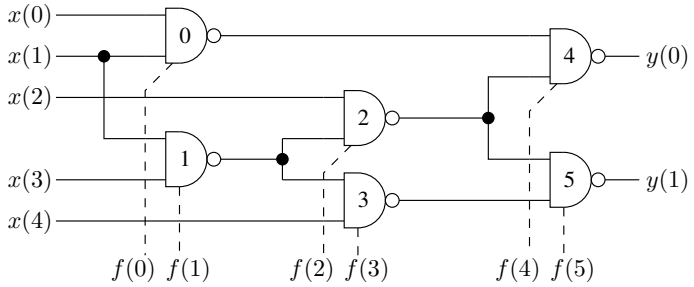


Figure 3.19: Generic combinational logic circuit.

Given an input $\vec{x}_i$ and a fault pattern $\vec{f}_j$, the probability of a correct output is an all-or-nothing function, representing the fault masking of the specified faults by the inputs applied to the circuit. This function can be modeled by a bit-wise XNOR function as expressed in (3.15), where $\vec{f}_0$ represents a fault-free circuit.

$$p(\vec{y} = correct|\vec{x}_i, \vec{f}_j) = p(\overline{\vec{y}(\vec{x}_i, \vec{f}_0) \oplus \vec{y}(\vec{x}_i, \vec{f}_j)} = 1) = \overline{\vec{y}(\vec{x}_i, \vec{f}_0) \oplus \vec{y}(\vec{x}_i, \vec{f}_j)} \tag{3.15}$$

Considering the reliability of a gate as q and its failure probability as $1 - q$, expression (3.16) determines the probability of occurrence of a fault pattern $\vec{f}_j$, given the individual reliability and unreliability of the gates on the circuit. Consider that the q_l and $1 - q_l$ elements are associated with different sets of gates, i.e., the index l represents different gates in these sets.

$$p(\vec{f}_j) = \prod_{l \in \gamma} q_l \prod_{l \in \phi} (1 - q_l) \tag{3.16}$$

The general expression for the reliability of a generic circuit is shown in (3.17), modeling the logical masking properties of the circuit according to input and fault patterns, considering each gate with a different reliability, and the occurrence of all combinations of inputs and faults.

$$R_{cir} = \sum_{j=0}^{2^G-1} \prod_{l \in \gamma} q_l \prod_{l \in \phi} (1 - q_l) \left[\sum_{i=0}^{2^n-1} p(\vec{x}_i) \left(\overline{\vec{y}(\vec{x}_i, \vec{f}_0) \oplus \vec{y}(\vec{x}_i, \vec{f}_j)} \right) \right] \tag{3.17}$$

Let q be the reliability of an individual logic gate (and $1 - q$ its unreliability), and consider that all the gates in the circuit have this same reliability. Let σ be the set of all fault vectors representing k simultaneous faults. Therefore, the probability of occurrence of k simultaneous faults ($F(q, k)$), can be modeled as a binomial distribution, as in (3.18).

$$F(q,k) = C_k^G (1-q)^k q^{G-k} = \frac{G!}{(G-k)!k!}(1-q)^k q^{G-k} \tag{3.18}$$

The general reliability expression in (3.17) can be rewritten as a weighted combination of all $F(q,k)$ functions, as in (3.19), where the coefficients c_k represent the percentage of fault masking for all input vectors $\vec{x}_i$ given all fault vectors with k simultaneous faults.

$$R = \sum_{k=0}^{G-1} F(q,k) c_k \tag{3.19}$$

The coefficients c_k can be computed according to expression (3.20).

$$c_k = \frac{\sum_{j\in\sigma} \sum_{i=0}^{2^n-1} p(\vec{x}_i) \left(\overline{\vec{y}(\vec{x}_i, \vec{f}_0) \oplus \vec{y}(\vec{x}_i, \vec{f}_j)} \right)}{C_k^G} \tag{3.20}$$

By considering a uniform distribution of the input vectors, $p(\vec{x}_i) = \frac{1}{2^n}$, and rewriting expression (3.19) according to expressions (3.18) and (3.20) gives the reliability expression in (3.21), that is the binomial model of the reliability.

$$R_{cir} = \frac{1}{2^n} \sum_{k=0}^{G-1} (1-q)^k q^{G-k} \sum_{j\in\sigma} \sum_{i=0}^{2^n-1} \left(\overline{\vec{y}(\vec{x}_i, \vec{f}_0) \oplus \vec{y}(\vec{x}_i, \vec{f}_j)} \right) \tag{3.21}$$

The reliability models represented by expressions (3.17) and (3.21) compute the exact reliability value of combinational logic circuits. The determination of coefficients c_k depends on an exhaustive fault simulation of the target circuit. This is intractable for large circuits but a closer examination of the binomial distribution can indicate how to alleviate the simulation workload. For large circuits, pseudo-random fault simulation can also be applied. The next subsection presents these optimization approaches.

3.4.1 Optimizing the model

With the assumption of gates with equal reliability and independence of fault occurrence, the probability of simultaneous faults can be modeled as a binomial distribution, as referred in (3.18). This model indicates that the probability of occurrence of multiple faults is dependent on the number of gates and the reliability of these gates, indicating that circuits with more reliable gates have a smaller probability of multiple faults. This can be seen in Fig. 3.20, that shows the probability of occurrence of k simultaneous faults in a circuit with a hundred gates. Three values of gate reliability are considered. Fig. 3.20 can be interpreted as follows: for an individual gate reliability of $q = 0.999$, there is almost 10% probability of occurrence of a single fault in a circuit with a hundred of these gates, and an almost negligible probability of occurrence of two simultaneous faults. For an individual gate reliability of $q = 0.99$, the probability of occurrence of two simultaneous faults rises to almost 20%, and the probability of a fault-free operation is reduced to less than 37%.

The assumption of equally reliable gates leads to the observation that a relaxation in the exhaustive simulation method can have a negligible impact in the accuracy obtained with the binomial model for reliability analysis. Given the number of gates in a circuit and their reliability, it is possible to determine the number of coefficients c_k necessary for an

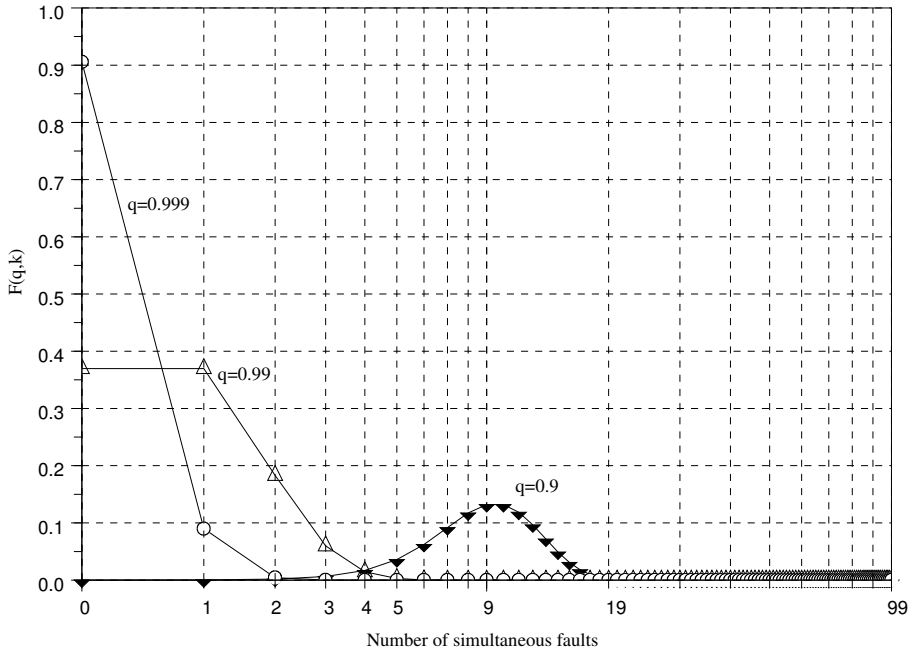


Figure 3.20: $F(q,k)$ of a circuit with a hundred gates as a function of q and k.

accurate analysis, i.e., the fault simulation can take into account only the relevant number of k simultaneous fault vectors. Considering that current CMOS gates have a reliability higher than 99.9999%, a great reduction in the complexity is allowed by the proposed approach of reliability analysis.

Consider the pseudo-random fault simulation, with N_x being the number of pseudo-random input vectors applied to the target circuit, and N_f the number of pseudo-random fault vectors of $\vec{f}_k$ type applied. The reliability model can be rewritten as in (3.22), where N_k defines de number of simultaneous faults that will be considered in the reliability computation.

$$R_{cir} = \sum_{k=0}^{N_k-1} C_k^G (1-q)^k q^{G-k} \frac{\sum_{j=0}^{N_f-1} \sum_{i=0}^{N_x-1} \left(\overline{\vec{y}(\vec{x}_i, \vec{f}_0) \oplus \vec{y}(\vec{x}_i, \vec{f}_j)} \right)}{N_f N_x} \tag{3.22}$$

The current work is targeted to the evaluation of the signal reliability of logic circuits but an important metric for evaluation of fault-tolerant approaches concerns the functional reliability. The next subsection shows the evaluation of the relevant metrics of self-checking circuits from the point of view of the PBR methodology.

3.4.2 Evaluating self-checking circuits

Another type of analysis that has been developed with the PBR approach is the functional evaluation of some self-checking fault-tolerant schemes, in the presence of multiple simultaneous faults. To evaluate these circuits, the circuit topology implemented for signal reliability analysis must be modified, individually comparing the data outputs of the circuits and the checking outputs of the circuits. This allows the evaluation of the functional

reliability of the referred circuits, when the original defined conditions (single-fault) are not respected.

Self-checking circuits, as presented in the previous chapter, are generally designed to deal with single faults during operation, where a checking circuit is responsible to sign the occurrence of a fault in the main circuit or a fault in the checking circuit itself. Every time the circuit detects a fault, a pre-defined action must take place, like re-computing the operation or marking the result as erroneous to avoid its usage by further operations. Independent of the action, a fault in a self-checking system can be seen as a performance penalty, where the data throughput is reduced, and the circuit can be considered as operating at an equivalent slower frequency. This penalty is one of the evaluations that we are interested in.

Besides the time penalty, by assuming more than one fault at a time, four possible events can be expected from the self-checking circuit:

- ξ, when the checker indicates correct operation and the circuit output is correct;
- τ, when the checker indicates an incorrect operation and the circuit output is correct;
- ψ, when the checker indicates a correct operation and the circuit output is incorrect;
- χ, when the checker indicates an incorrect operation and the circuit output is incorrect.

The functional reliability $\Re$ can be defined as in (3.23), and represents the signal reliability ξ plus the probability of the checker to detect an error, i.e., events τ and χ. An equivalent metric for the functional reliability is called fault-secureness (FS), that is the capacity of a circuit to indicate the faults of the assumed set.

$$\Re = p\left(\xi \cup \tau \cup \chi\right) \tag{3.23}$$

A time penalty metric Θ can be defined as the probability of detection of an error in a given circuit, considering the two events associated with it, τ and χ. This metric characterizes the probability of interruption of the normal operation of the circuit by the error detection logic.

The probability of occurrence of the defined events are expressed in (3.24), (3.25), (3.26), and (3.27), respectively, where $F(q,k) = C_k^G (1-q)^k q^{G-k}$.

$$E(q) = \frac{1}{2^n} \sum_{k=0}^{G-1} F(q,k) c_k(\xi) \tag{3.24}$$

$$T(q) = \frac{1}{2^n} \sum_{k=0}^{G-1} F(q,k) c_k(\tau) \tag{3.25}$$

$$U(q) = \frac{1}{2^n} \sum_{k=0}^{G-1} F(q,k) c_k(\psi) \tag{3.26}$$

$$X(q) = \frac{1}{2^n} \sum_{k=0}^{G-1} F(q,k) c_k(\chi) \tag{3.27}$$

This way, the functional reliability and the time penalty expressions can be rewritten as in (3.28) and (3.29).

$$\Re(q) = E(q) + T(q) + X(q) \tag{3.28}$$

$$\Theta(q) = T(q) + X(q) \tag{3.29}$$

3.5 The Signal probability model

The use of signal probability to evaluate the reliability of logic circuits is based on the assumption that the probability of occurrence of a correct output value can be determined by computing the cumulative effect of multiple faults on the signals of the circuit. It is widely accepted that fault masking occurs when one fault cannot propagate to the output of a circuit given the state of related signals on the circuit but, it is important to remember that fault masking also occurs when one fault interacting with another one results in a correct signal. The signal probability method models all possible interactions of fault-prone signals and gates to take into account these fault masking effects on the probability of circuit's signals.

Consider the signal probability at the output of a gate as a function of the signal probabilities at its inputs and the logic function of the gate. Fig. 3.21 shows the signal probabilities at the output of an AND gate, given the input probabilities $p(a)$ and $p(b)$.

truth table

a	b	y	
0	0	0	$p(y=0) = p(a=0)p(b=0) + p(a=0)p(b=1) + p(a=1)p(b=0)$
0	1	0	
1	0	0	
1	1	1	$p(y=1) = p(a=1)p(b=1)$

Figure 3.21: Signal probabilities at the output of an AND gate.

The traditional definition of *node signal probability* is the probability that the signal value of a given node will be **1**, under a random assignment of an input vector [22, 23, 89, 90]. According to this definition, the signal probability of output y can be rewritten as $p(y) = p(a)p(b)$. This definition is adequate for representing the signal probabilities in fault-free circuits. For fault-prone circuits, the computation of signal probabilities involves a higher number of signal states, gate function and gate reliability combinations.

Let's assume that in a fault-prone circuit every logic signal can be in one of four possible states, according to the corresponding error-free value of the signal. These four states are: a *correct 0*; a *correct 1*; an *incorrect 0*; and an *incorrect 1*. Define $0c$, $1c$, $0i$ and $1i$ as the respective representative values for these states. Consider that gate g can also be characterized by a correct operating state g_c and an incorrect operating state g_i. As stated before, a gate at an incorrect operating state generates the complement of the expected output value, i.e., an output bit-flip.

Fig. 3.22 shows the new truth tables that can be defined according to all possible states of the fault-prone inputs and fault-prone gate, where shadowed values correspond to fault-masked results, i.e., the results associated with a fault-free operation of the circuit.

truth table 1

a	b	$y(g_c)$	$y(g_i)$
0c	0c	0c	1i
0c	1c	0c	1i
1c	0c	0c	1i
1c	1c	1c	0i

truth table 2

a	b	$y(g_c)$	$y(g_i)$
0c	0i	0c	1i
0c	1i	0c	1i
1c	0i	0i	1c
1c	1i	1i	0c

truth table 3

a	b	$y(g_c)$	$y(g_i)$
0i	0c	0c	1i
0i	1c	0i	1c
1i	0c	0c	1i
1i	1c	1i	0c

truth table 4

a	b	$y(g_c)$	$y(g_i)$
0i	0i	0i	1c
0i	1i	0c	1i
1i	0i	0c	1i
1i	1i	1i	0c

Figure 3.22: Truth tables of a fault-prone AND gate with fault-prone input signals.

The probability of output signal y can be rewritten as in (3.30), where $p(signal_{state})$ corresponds to the probability of the specified signal being at the specified state.

$$p(y_{1c}) = p(a_{1c})p(b_{1c})p(g_c) + p(a_{1c})p(b_{0i})p(g_i) + p(a_{0i})p(b_{1c})p(g_i) + p(a_{0i})p(b_{0i})p(g_i) \tag{3.30}$$

The complete set of output probabilities can be seen on Fig. 3.23. The expressions shown in Fig. 3.23 are composed of all possible combinations of input and gate states, re-grouped according to the gate's logic function. The probability expressions for other combinational functions can be obtained in a similar way.

$$p(y_{1c}) = p(a_{1c})p(b_{1c})p(g_c) + p(a_{0i})p(b_{0i})p(g_i) + p(a_{0i})p(b_{1c})p(g_i) + p(a_{1c})p(b_{0i})p(g_i)$$

$$\begin{aligned} p(y_{0c}) = {} & p(a_{0c})p(b_{0c})p(g_c) + p(a_{0c})p(b_{1i})p(g_c) + p(a_{1i})p(b_{0c})p(g_c) + p(a_{1i})p(b_{1i})p(g_i) \\ & + p(a_{0c})p(b_{1c})p(g_c) + p(a_{0c})p(b_{0i})p(g_c) + p(a_{1i})p(b_{0i})p(g_c) + p(a_{1i})p(b_{1c})p(g_i) \\ & + p(a_{1c})p(b_{0c})p(g_c) + p(a_{0i})p(b_{0c})p(g_c) + p(a_{0i})p(b_{1i})p(g_c) + p(a_{1c})p(b_{1i})p(g_i) \end{aligned}$$

$$p(y_{1i}) = p(a_{1c})p(b_{1c})p(g_i) + p(a_{0i})p(b_{0i})p(g_c) + p(a_{0i})p(b_{1c})p(g_c) + p(a_{1c})p(b_{0i})p(g_c)$$

$$\begin{aligned} p(y_{0i}) = {} & p(a_{0c})p(b_{0c})p(g_i) + p(a_{0c})p(b_{1i})p(g_i) + p(a_{1i})p(b_{0c})p(g_i) + p(a_{1i})p(b_{1i})p(g_c) \\ & + p(a_{0c})p(b_{1c})p(g_i) + p(a_{0c})p(b_{0i})p(g_i) + p(a_{1i})p(b_{0i})p(g_i) + p(a_{1i})p(b_{1c})p(g_c) \\ & + p(a_{1c})p(b_{0c})p(g_i) + p(a_{0i})p(b_{0c})p(g_i) + p(a_{0i})p(b_{1i})p(g_i) + p(a_{1c})p(b_{1i})p(g_c) \end{aligned}$$

Figure 3.23: Output probabilities of the fault-prone AND gate with fault-prone input signals.

The reliability of the output signal y in this example can be computed by summing up the probabilities of this signal being in a correct state, i.e., $p(y_{0c})+p(y_{1c})$. The contribution of fault masking to this reliability is inherently modeled in the referred expressions, by means of the terms associated with incorrect signal/gate states that are included on the probability expression of a correct output state, e.g., $p(a_{1c})p(b_{0i})p(g_i)$ in expression (3.30).

In a logic circuit, the probabilities of all signals can be computed by signal probability propagation, considering the output signal probability of a gate as the input signal probability propagated to the gates on its fanout cone. By propagating the probability of all

signals through the gates of a circuit, the reliability of this circuit can be determined by computing the joint probability of correct **0**s and **1**s at all the output signals, as in (3.31), where m is the number of output signals. This reliability expression is correct for circuits without reconvergent fanout signals, that will be discussed further.

$$R_{circuit} = \prod_{j=0}^{m-1} (p(y(j)_{0c}) + p(y(j)_{1c})) \tag{3.31}$$

To simplify the computation of signal probabilities, a matrix organization of input and gate states seems to be more appropriate than the traditional probability expressions referred in previous works [22, 23, 86], similar to the ones in Fig. 3.21. The examination of signal probability expressions, like the ones in Fig. 3.23, shows a repetitive structure that can be modeled by matrix data representation and matrix operations. The presentation of this model follows.

Consider the representation of signal state probabilities as a 2×2 matrix, as shown in Fig. 3.24, where the conventions used for the probabilities are indicated by the lower matrix.

$$SIGNAL_4 = \begin{bmatrix} signal_0 & signal_1 \\ signal_2 & signal_3 \end{bmatrix}$$

$$P_{2\times2}(signal) = \begin{bmatrix} P(signal = correct\ 0) & P(signal = incorrect\ 1) \\ P(signal = incorrect\ 0) & P(signal = correct\ 1) \end{bmatrix}$$

Figure 3.24: Proposed 2×2 matrix representation of fault-prone signal probabilities.

The matrix representation of the fault-prone behavior of a gate can be obtained from its truth table, allowing a straightforward modeling of its operation probabilities. Considering their convenience, the probabilistic transfer matrices (PTMs) [16, 17] and ideal transfer matrices (ITMs) representation has been chosen to model the fault-prone and fault-free behavior of individual logic gates, but other representations are possible. Fig. 3.25 shows an example of the fault-free and fault-prone probability models for two logic gates, a 2-input XOR and a 3-input majority function (MAJ), with reliability q and unreliability $1 - q$.

Consider an i-input gate g, with input signals $x(0)$ to $x(i-1)$, and input probabilities $X(0)_4$ to $X(i-1)_4$. The input probabilities of gate g, defined as I_g, can be computed by tensoring the input signals probability matrices, as in (3.32). This results in a $2^i \times 2^i$ probability matrix.

$$I_g = X(0)_4 \otimes X(1)_4 \otimes ... \otimes X(i-1)_4 \tag{3.32}$$

Fig. 3.26 shows the input probabilities of a given gate g with inputs a and b. The fault-free values will be always represented by probabilities in the main diagonal of the concerned matrix.

To compute the output probabilities of a fault-prone gate, the input probabilities must be multiplied by the gate state matrix, i.e., the PTM of the gate. Thus, the signal probability of output y of gate g is computed by $p(y) = I_g \times PTM_g$. For a 2-input AND gate, with inputs a and b, reliability q and unreliability $\bar{q}$ ($\bar{q} = 1 - q$), the output probability $p(y)$ is computed as shown in Fig. 3.27.

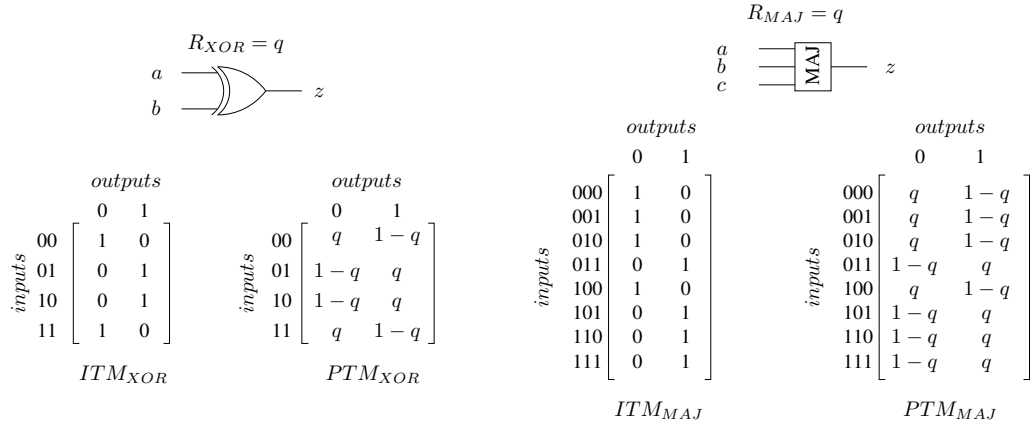


Figure 3.25: Fault-free and fault-prone probability models of two logic functions.

$$I_g = \begin{bmatrix} a_0 & a_1 \\ a_2 & a_3 \end{bmatrix} \otimes \begin{bmatrix} b_0 & b_1 \\ b_2 & b_3 \end{bmatrix} = \begin{bmatrix} a_0b_0 & a_0b_1 & a_1b_0 & a_1b_1 \\ a_0b_2 & a_0b_3 & a_1b_2 & a_1b_3 \\ a_2b_0 & a_2b_1 & a_3b_0 & a_3b_1 \\ a_2b_2 & a_2b_3 & a_3b_2 & a_3b_3 \end{bmatrix}$$

Figure 3.26: Input probabilities computation for a gate with inputs a and b.

$$p(y) = \underbrace{\begin{bmatrix} a_0b_0 & a_0b_1 & a_1b_0 & a_1b_1 \\ a_0b_2 & a_0b_3 & a_1b_2 & a_1b_3 \\ a_2b_0 & a_2b_1 & a_3b_0 & a_3b_1 \\ a_2b_2 & a_2b_3 & a_3b_2 & a_3b_3 \end{bmatrix}}_{I_{AND}} \times \underbrace{\begin{bmatrix} q & \bar{q} \\ q & \bar{q} \\ q & \bar{q} \\ \bar{q} & q \end{bmatrix}}_{PTM_{AND}}$$

Figure 3.27: Computing the output signal probability.

The $p(y)$ matrix models the probability of occurrence of input/output combinations at the concerned gate. Fig. 3.28 shows the output probability expressions for the gate referred in Fig. 3.27.

$$p(y) = \begin{array}{c} \begin{array}{cc} p(y=0) & p(y=1) \end{array} \\ \left[\begin{array}{c|c} a_0b_0q + a_0b_1q + a_1b_0q + a_1b_1\bar{q} & a_0b_0\bar{q} + a_0b_1\bar{q} + a_1b_0\bar{q} + a_1b_1q \\ a_0b_2q + a_0b_3q + a_1b_2q + a_1b_3\bar{q} & a_0b_2\bar{q} + a_0b_3\bar{q} + a_1b_2\bar{q} + a_1b_3q \\ a_2b_0q + a_2b_1q + a_3b_0q + a_3b_1\bar{q} & a_2b_0\bar{q} + a_2b_1\bar{q} + a_3b_0\bar{q} + a_3b_1q \\ a_2b_2q + a_2b_3q + a_3b_2q + a_3b_3\bar{q} & a_2b_2\bar{q} + a_2b_3\bar{q} + a_3b_2\bar{q} + a_3b_3q \end{array}\right] \end{array}$$

Figure 3.28: Output probability expressions of a gate with reliability q.

The output probabilities in Fig. 3.28 must be re-grouped in a 2×2 probability matrix, i.e., Y_4, to allow its use in further probability computations. To determine Y_4, the logic function of gate g must be considered, what is modeled by its ITM. Y_4 is computed by adding the adequate $p(y)$ probabilities according to the gate's ITM, as indicated in the general expressions (3.33), (3.34), (3.35) and (3.36). The indexes $0, r$ and $1, r$ in the equations correspond to $column, row$ indexes in the respective matrices.

$$y_0 = \sum_{r=0}^{2^i-1} p(y)_{[0,r]} ITM_{[0,r]} \tag{3.33}$$

$$y_1 = \sum_{r=0}^{2^i-1} p(y)_{[1,r]} (1 - ITM_{[1,r]}) \tag{3.34}$$

$$y_2 = \sum_{r=0}^{2^i-1} p(y)_{[0,r]} (1 - ITM_{[0,r]}) \tag{3.35}$$

$$y_3 = \sum_{r=0}^{2^i-1} p(y)_{[1,r]} ITM_{[1,r]} \tag{3.36}$$

Fig. 3.29 shows the $p(y)$ terms that have to be added to compute the probabilities of Y_4, concerning the AND gate of Fig. 3.27. As shown in the figure, the state probability y_3 perfectly matches the probability expressed in (3.30), computed by the fault-prone truth table expressions.

$$p(y) = \begin{array}{c} \begin{array}{cc} y_0 & y_1 \end{array} \\ \left[\begin{array}{c|c} \overbrace{a_0b_0q + a_0b_1q + a_1b_0q + a_1b_1\bar{q}} & \overbrace{a_0b_0\bar{q} + a_0b_1\bar{q} + a_1b_0\bar{q} + a_1b_1q} \\ a_0b_2q + a_0b_3q + a_1b_2q + a_1b_3\bar{q} & a_0b_2\bar{q} + a_0b_3\bar{q} + a_1b_2\bar{q} + a_1b_3q \\ \underbrace{a_2b_0q + a_2b_1q + a_3b_0q + a_3b_1\bar{q}} & \underbrace{a_2b_0\bar{q} + a_2b_1\bar{q} + a_3b_0\bar{q} + a_3b_1q} \\ \underbrace{a_2b_2q + a_2b_3q + a_3b_2q + a_3b_3\bar{q}} & \underbrace{a_2b_2\bar{q} + a_2b_3\bar{q} + a_3b_2\bar{q} + a_3b_3q} \end{array}\right] \\ \begin{array}{cc} y_2 & y_3 \end{array} \end{array}$$

Figure 3.29: Computing Y_4 from the output probability $p(y)$ of an AND gate.

The process of computing the 2×2 output probability matrix of a gate is defined here as the *propagation* of the input signal probabilities through the gate. Fig. 3.30 presents

the propagation of input signals a and b through a NAND gate with reliability $q = 0.92$. Concerning primary inputs, they are generally considered as fault-free signals and they are assumed to have the same probability of being **0** or **1**. This is represented by state probabilities $signal_0 = signal_3 = 0.5$.

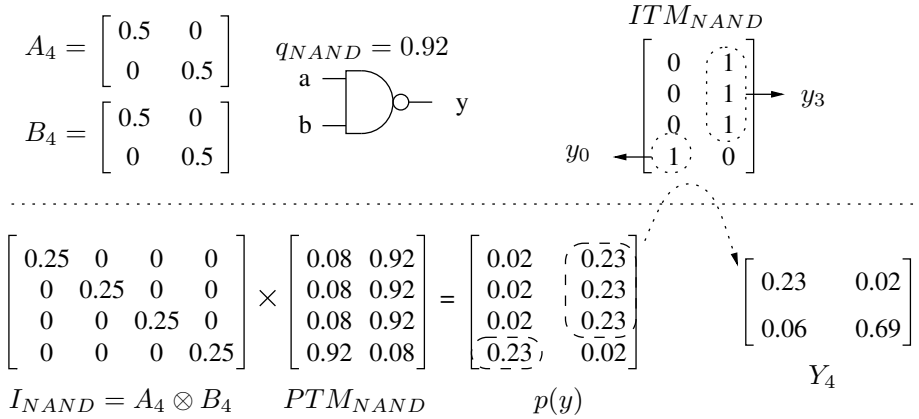


Figure 3.30: Propagation of signal probabilities through a NAND gate.

The reliability of a generic fault-prone signal is determined by its probability of being in a correct **0** or **1** state, as expressed in (3.37), being embedded in the 2×2 probability representation. The probability of error (unreliability) of the same signal can also be determined according to its probability matrix, as in (3.38).

$$R_{signal} = signal_0 + signal_3 = p(signal = correct\ 0) + p(signal = correct\ 1) \tag{3.37}$$

$$1 - R_{signal} = signal_2 + signal_1 = p(signal = incorrect\ 0) + p(signal = incorrect\ 1) \tag{3.38}$$

The signal reliability of a logic circuit can be obtained by propagating the signal probabilities of the primary input signals to the primary output signals. The reliability of a primary output signal corresponds to the signal reliability of the circuit that generates this output. The overall circuit reliability can be determined by computing the joint probability of the output signal reliabilities, as expressed in (3.39), where R_j corresponds to the reliability of the jth output node.

$$R_{circuit} = \prod_{j=0}^{m-1} R_j \tag{3.39}$$

This method for reliability analysis is defined as the signal probability reliability model (SPR), that is of straightforward application, based on simple matrix operations. Fig. 3.31 presents the application of the method to a simple circuit.

Fig. 3.32 shows another example of reliability analysis with the SPR approach. Despite the fact that the gate reliability is considered to be the same for all input patterns, different reliability values can be associated to different input patterns, what can be modeled by the gate transfer function, allowing a more realistic modeling of the logic gates.

The SPR model has a time complexity that is linear with the number of gates, i.e., $O(G)$, and generates exact reliability results for circuits without reconvergent fanouts,

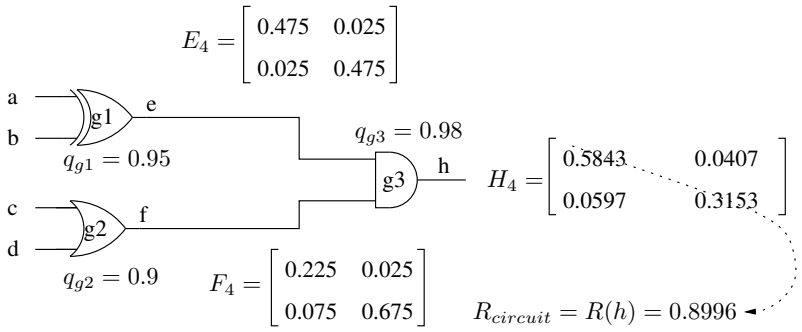


Figure 3.31: Example of application of the SPR approach.

where the assumption of independence among signal states holds. For practical circuits with reconvergent fanout signals, the main drawback concerns signal correlations, what invalidates the direct computation of joint signal probabilities as needed by the propagation process.

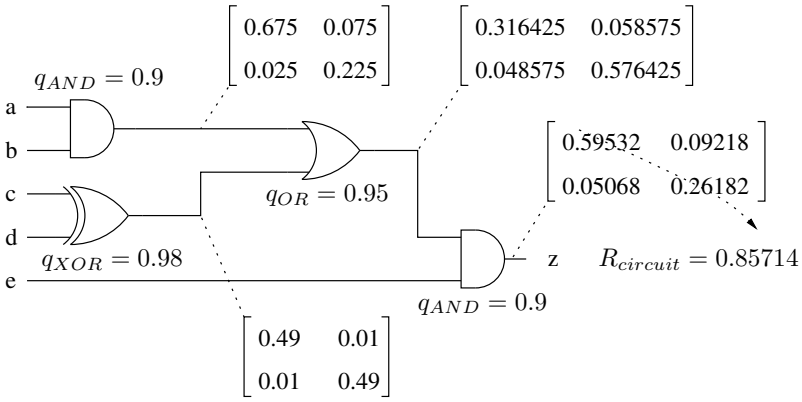
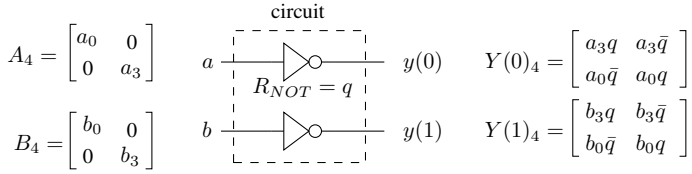


Figure 3.32: Another example of application of the SPR approach.

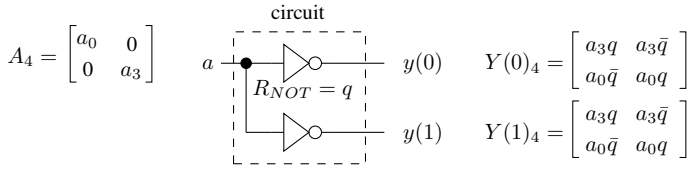
To detail the signal correlation problem, consider the example circuit in Fig. 3.33, without reconvergent fanout signals.



$$R(circuit) = R_{y(0)} R_{y(1)} = a_3 b_3 q^2 + a_3 b_0 q^2 + a_0 b_3 q^2 + a_0 b_0 q^2$$

Figure 3.33: Computing the reliability of a simple 2-output circuit.

By short-circuiting the inputs of the circuit in Fig. 3.33, we transform the single a input into a reconvergent fanout signal, as shown in Fig. 3.34. Apparently, the signal a does not reconverge in the referred circuit but the computation of the circuit reliability implies the joint probability of the primary output signal states, that represents a reconvergence point.



$$R(circuit) = R_{y(0)}R_{y(1)} = a_3^2q^2 + a_3a_0q^2 + a_0a_3q^2 + a_0^2q^2$$

Figure 3.34: Computing the reliability of a simple circuit with a reconvergent fanout.

As determined by the SPR model, the reliability expression for the circuit in Fig. 3.34 is composed of polynomials with inconsistent probabilities, that depend on different states of the same signal (a_0a_3, a_3a_0), and redundant probabilities, that appear twice (a_0^2, a_3^2). Inconsistent probabilities should result in null values and redundant probabilities should consider only one occurrence of each event, leading to a correct result of $R_{circuit} = a_3q^2 + a_0q^2$.

The problem of signal correlations is well known in domains that use the signal probability metric, like test generation and power consumption. This problem is similar to the boolean satisfiability problem and it is in the class of #P-complete (sharp P complete) ones [22, 23], possibly even harder than the NP-complete problems.

Fig. 3.35a shows an example of a circuit with two reconvergent fanouts and its exact reliability, for $q_{NAND} = 0.92$. The signal e is also considered a reconvergent fanout since the circuit reliability computation involves the joint probabilities of the output signals. Fig. 3.35b shows what is effectively being computed when signal correlations are not taken into account.

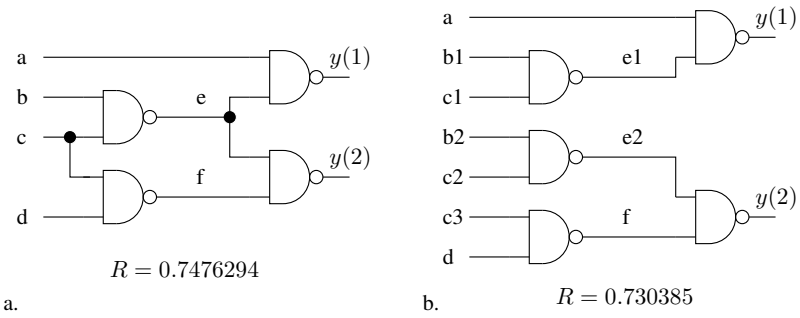


Figure 3.35: Effect of ignoring signal correlations.

Since the signal correlation problem is intractable for practical circuits, the search for better results leads to the use of heuristics, that aim a reduction in the complexity of the problem at the price of accuracy. The choice of a given heuristic must consider the tradeoffs that are allowed concerning execution time and accuracy. Considering the domain of signal probability, many heuristics have been proposed, whose are briefly presented here.

The work of Parker and McCluskey [91] applies exponent suppression in the probability polynomials to eliminate the effect of signal correlation. The method leads to exact results but requires exponential space for polynomials storage, and does not work for fault-prone signals. The method proposed by Brglez [92], referred as COP, does not consider signal correlations, assuming independence of all signals, directly computing the signal probabilities in linear time. The method PREDICT, proposed by Seth et al. [93], is based on a

graph approach to determine conditional probabilities, but the computational complexity of the approach increases exponentially. The *cutting* algorithm proposed by Savir et al. [94], transforms a combinational network into a tree-structured network, cutting reconvergent fanout branches. The accuracy of the algorithm depends on the choices of cuts, what is non-deterministic. The *cutting* algorithm is claimed to compute upper and lower bounds on signal probabilities but this remains to be proved. The Stafan approach is a work of Jain and Agrawal [95] and uses fault-free simulation and statistical analysis to determine testability metrics, not directly dealing with signal probability. Results tend to exact values as simulation tends to exhaustive experiments. The Weighted Averaging Algorithm (WAA) from Krishnamurthy and Tollis [89], aims to correct the signal probabilities by computing the effect of individual input probabilities on these values, providing a root mean square deviation from the exact values. The time complexity of the algorithm is linear on the number of gates and input signals. Ercolani et al. [23] propose two heuristics for correcting signal correlation effects, the dynamic WAA (DWAA) and the correlation coefficients method. The DWAA applies the same corrections of the WAA approach but controls the reconvergent fanout signal probabilities instead of the input signal ones, and updates the signal probabilities at each iteration of the algorithm. The correlation coefficient method computes along with signal probabilities the conditional dependence of these signals. The complexity of this method is dependent on the circuit structure. The results of these approaches are better than the ones obtained with the WAA approach. The work of Al-Kharji and Al-Arian [90] proposes the *possibilistic* algorithm, that is similar to the WAA heuristic but details even more the effects of input signals on probability deviations. It is claimed to be more accurate than the WAA approach, but some steps of the algorithm are not well explained in the original work.

Most of the heuristics are not directly targeted to signal probability determination and all of them have been developed for fault-free logic signals. Considering the original objective of the analysis, all of the cited heuristics can be adequate, but not all of them are useful for the proposed fault-prone analysis of signal probabilities. In the current work, the DWAA approach was chosen to be adapted to the SPR model in a fault-prone environment, because it is considered the more accurate approach among the ones with linear complexity [23].

Since accuracy can be generally traded with computing time, another heuristic was developed to allow the control of these tradeoffs in signal probability computation. This heuristic is referred as the SPR *multi-pass* algorithm, and exact results can be expected at the price of an exponential processing time, and a linear computing time can be obtained at the price of uncorrected SPR results. With the multi-pass algorithm the overall range of solutions can be explored.

Next subsections introduce the proposed heuristics, based on the SPR approach, to minimize the effects of signal correlations.

3.5.1 SPR DWAA

The weighted averaging algorithm (WAA) corrects the signal probability values by determining the influence of each primary input on these values. This is accomplished by iteratively setting each primary input probability to **0** and **1**, successively. Consider the node s in a logic circuit with n primary inputs x. Consider the computation of signal probabilities by direct probabilities propagation, ignoring signal correlations, as the *0-algorithm* [23]. Consider $p(s)$ as the signal probability computed with the 0-algorithm. By setting

the ith primary input to **0** and computing the signal probabilities we obtain $p(s|x(i)=0)$, and repeating the procedure with $x(i)$ set to **1** leads to $p(s|x(i)=1)$. The average signal probability of s can be computed as in (3.40), where the influence of multiple reconvergences of the input $x(i)$ on the signal s has been removed. The average expression assumes $p(x(i)=0)=p(x(i)=1)$ and an average probability of s being different of 0.

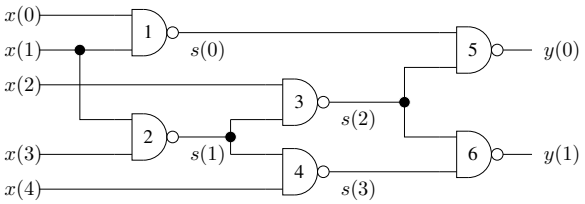
$$p(s|x(i)) = \frac{p(s|x(i)=0)+p(s|x(i)=1)}{2} \tag{3.40}$$

Since signal s can be influenced by reconvergence of every input signal, the algorithm must compute the signal probability average according to every primary input. Having determined the n probability averages, the corrected value of probability s is the weighted average of these n averages, where the weighting factor is dependent on the relative deviation of the average from the value obtained with the 0-algorithm. The correction on the value of $p(s)$ is computed as in (3.41), where $w_{x(i)}$ is the weight of the concerned average, calculated according to (3.42). These expressions must be computed for all s nodes and all i inputs.

$$p(s) = \sum_{i=0}^{n-1} w_{x(i)} p(s|x(i)) \tag{3.41}$$

$$w_{x(i)} = \frac{|p(s)-p(s|x(i))|}{\sum_{j=0}^{n-1}|p(s)-p(s|x(j))|} \tag{3.42}$$

Fig. 3.36 shows the signal probability values for the outputs of the C17 circuit, as computed by the 0-algorithm, the WAA heuristic and the exact result computed by exhaustive analysis. As can be seen, the WAA approximates the signal probabilities to their exact values. The precision of the correction depends on the reconvergence degree of the input signals into the target nodes and their interaction.



	$0-algorithm$	WAA	$Exact$
$p(y(0))$	0.53125	0.5625	0.5625
$p(y(1))$	0.609375	0.59375	0.5625

Figure 3.36: Signal probabilities at the output of the C17 circuit.

The dynamic weighted average algorithm (DWAA) is based on the WAA general idea, but instead of setting the primary inputs to **0** and **1**, it successively sets the signal probabilities of the reconvergent fanout signals to **0** and **1**. From now on, a reconvergent fanout signal will be referred as simply fanout. In order to apply the DWAA algorithm, the fanouts of the circuit must be ordered topologically from input to output. The algorithm processes one fanout signal at a time, successively setting its probability to **0** and **1**, and dynamically updating the signal probabilities on its fanout cone. This way, the algorithm updates signal probabilities according to a dependency order.

Consider a circuit with ϕ fanout signals, and the set Φ_f representing the signals on the fanout cone of the fth fanout. Compute the signal probability of all signals in the circuit by means of the 0-algorithm. These probabilities will be defined as $p(s,0)$, where s

is a node of the circuit. Successively set each fanout signal to **0** and **1**, and determine the corresponding signal probabilities of the nodes on its fanout cone, according to (3.43).

$$p(s|f) = p(s|f=0)p(f=0) + p(s|f=1)p(f=1); \quad \forall \ s \in \Phi_f \tag{3.43}$$

After determining $p(s|f)$, the corrections in $p(s)$ are computed according to (3.44), where $p(s,t)$ is the value of $p(s)$ after processing t fanouts, $w(s)_f$ is the weight correction to be applied and $ws(s,t-1)$ is the cumulative value of weight corrections from precedent steps.

$$p(s,t) = \frac{p(s,t-1)ws(s,t-1) + p(s|f)w(s)_f}{ws(s,t-1) + w(s)_f} \tag{3.44}$$

The weight of the corrections and its cumulative value are determined according to (3.45) and (3.46), respectively.

$$w(s)_f = |p(s,0) - p(s|f)| \tag{3.45}$$

$$ws(s,t-1) = \sum_{j=1}^{t-1} w(s)_j \tag{3.46}$$

Table 3.2 presents the output signal probabilities of the circuit C17 as computed by the method DWAA, where the column probabilities related to the fanout signals x_i, s_1 and s_2 correspond to the updated probability values of the outputs after processing each one of these fanout signals. As shown in the table, the result (bold values) is closer to the exact value than the signal reliability computed by the WAA heuristic.

Table 3.2: C17 output probabilities corrected by the DWAA method

	0-algorithm	Fanout signal			Exact
		x_1	s_1	s_2	
$p(y_0)$	0.53125	0.5625	0.5625	**0.5625**	0.5625
$p(y_1)$	0.609375	0.59375	0.5703125	**0.5703125**	0.5625

Since the DWAA can be considered the more accurate [23] heuristic among the linear complexity ones, it was chosen to be adapted to the SPR approach. The time complexity of the DWAA method is $O(G\phi)$, G being the number of gates and ϕ the number of fanouts.

The application of the DWAA heuristic to the SPR approach implies some modifications in the original expressions, to deal with the four-state probability representation of fault-prone signals. Besides that, not only the signal probabilities must be updated but also the reliability computed with these probabilities must be corrected by the same algorithm, since the joint probability of the output signals is a reconvergence point, as discussed before.

After determining the topological order of the fanouts in the circuit, the signal probabilities are computed based on the SPR algorithm, generating the probabilities $S_4(0)$, where s is a fault-prone node in the circuit. The individual probabilities of $S_4(0)$ are represented as $p(s_l, 0)$, where l is the lth signal state of matrix S_4. Successively, each probability state k of fanout node f is set to **1** (while the other states are set to **0**) and the probabilities of the nodes on its fanout cone are computed. The probability of signal s being a correct **1** given that the fanout node f on its fanin cone is a correct **0** is represented by $p(s_3|f_0 = 1)$. After

processing the fanout node f, the average signal probability of the signals on its fanout cone are computed according to (3.47), where $p(f_k = 1)$ corresponds to the probability of fanout signal f being in state k. The set of all $p(s_l|f)$ values corresponds to $S_4(f)$, or the signal probability of signal s given fanout f.

$$\begin{aligned} p(s_l|f) &= p(s_l|f_0 = 1)p(f_0 = 1) + p(s_l|f_1 = 1)p(f_1 = 1) \\ &+ p(s_l|f_2 = 1)p(f_2 = 1) + p(s_l|f_3 = 1)p(f_3 = 1) \quad \forall\ l \in \{0, 3\} \end{aligned} \tag{3.47}$$

The weighting factor is determined according to the deviation between $S_4(0)$ and $S_4(f)$ as in (3.48), and the sum of all previous weighting factors is computed as in (3.49), considering t iterations of the algorithm (t fanouts processed).

$$w(s_l)_f = |p(s_l, 0) - p(s_l|f)| \tag{3.48}$$

$$ws(s_l, t-1) = \sum_{j=1}^{t-1} w(s_l)_j \tag{3.49}$$

The weighted updates of signal probabilities are computed according to (3.50). All of the presented operations can be executed with the data in matrix form, but the multiplications and the division in expression (3.50) are *punctual* operations.

$$p(s_l, t) = \frac{p(s_l, t-1)ws(s_l, t-1) + p(s_l|f)w(s_l)_f}{ws(s_l, t-1) + w(s_l)_f} \tag{3.50}$$

The algorithm iterates through all the reconvergent fanouts of the circuit, and dynamically updates the signal probabilities and the signal reliability of the circuit. As explained before, instead of computing the reliability of the circuit at the end of the updating process of signal probabilities, the reliability analysis is also updated at the end of each iteration.

By means of the 0-algorithm, the reliability $R(0)$ of the circuit is determined. At each iteration of the algorithm, the related value of circuit reliability is computed, as in (3.51), where $p(y(j)_n|f_l = 1)$ is the value of the nth state of the jth output signal when the fanout signal probability f_l has been set to **1**.

$$R(f_l) = \prod_{j=0}^{m-1} p(y(j)_0|f_l = 1) + p(y(j)_3|f_l = 1) \quad \forall\ l \in \{0, 3\} \tag{3.51}$$

The average value of reliability after processing fanout f is determined according to (3.52).

$$R_f = \sum_{l=0}^{3} R(f_l)p(f_l = 1) \tag{3.52}$$

The deviation of the reliability from the initial value is computed by means of (3.53).

$$w(R)_f = |R(0) - R_f| \tag{3.53}$$

The cumulated deviations after processing t fanouts is determined as in (3.54).

$$ws(R, t-1) = \sum_{j=1}^{t-1} w(R)_j \tag{3.54}$$

The updated value after each iteration of the algorithm is computed according to expression (3.55).

$$R(t) = \frac{R(t-1)ws(R, t-1) + R_f w(R)_f}{ws(R, t-1) + w(R)_f} \tag{3.55}$$

Table (3.3) presents the reliability analysis of the circuit C17 according to the SPR DWAA heuristic, where $R(y)_{DWAA}$ is the reliability computed as a function of the corrected signal probabilities of the output signals and R_{DWAA} is the reliability computed according to expressions (3.51) through (3.55).

Table 3.3: Reliability of circuit C17 computed by the SPR DWAA method

	Gate reliability (q)				
	0.98	0.96	0.94	0.92	0.9
$R(y)_{DWAA}$	0.8931	0.8003	0.7199	0.6504	0.5902
R_{DWAA}	0.9052	0.8209	0.7278	0.6799	0.6213
$Exact$	0.9064	0.8225	0.7475	0.6806	0.6211

The multi-pass heuristic is presented in the next section, allowing the complete set of solutions for reliability analysis, exploring the concept of dominant fanout signals to reduce the time complexity of the algorithm with a reduced impact on the accuracy.

3.5.2 SPR multi-pass

As referred earlier in the current section, the assumption of independence among all the signals in a logic circuit leads to deviations at the signal probability in the presence of reconvergent fanout signals, when computed by the SPR method. As presented in Fig. 3.34, the assumption of independence among different branches of the same signal results on inconsistent and redundant probabilities. Apparently, it would be simple to correct these effects by setting the state values of inputs or fanout signals to **1** and **0** successively, as is the case for the WAA and DWAA heuristics. Despite the approximations allowed by these heuristics, the exact value cannot be deterministically determined by processing only one signal at a time. For exact results, all possible combinations of inputs or fanout states should be applied, i.e., an exhaustive simulation.

The multi-pass algorithm is a heuristic based on the fact that every fanout contributes differently to the probability of the signals and to the reliability of the circuit. By running an exhaustive propagation of a relevant subset of the fanout signal probabilities, it is possible to reduce the complexity of the problem and achieve a good level of accuracy. The set of relevant or *dominant* fanouts to be considered on the analysis is a function of the allowed time complexity, and so, the designer can choose the adequate tradeoff between execution time and accuracy by choosing the number of fanout signals that will be taken into account. The proposed heuristic covers the overall solution space, from a fast and not so accurate analysis (by choosing zero fanout signals), to an exact analysis (by choosing the complete set of fanout signals).

Once the number of fanout signals to be considered has been defined, the multi-pass algorithm cycles through all possible combinations of fanout states, following the topological order of fanouts, in a tree-like propagation algorithm. At each iteration a new value of circuit reliability is computed, as in (3.56), where F is the specified number of dominant fanouts.

$$R(circuit) = \sum_{f=1}^{4^F} R(circuit, f) \tag{3.56}$$

Fig. 3.37 shows an example of the multi-pass algorithm in the analysis of the circuit from Fig. 3.34.

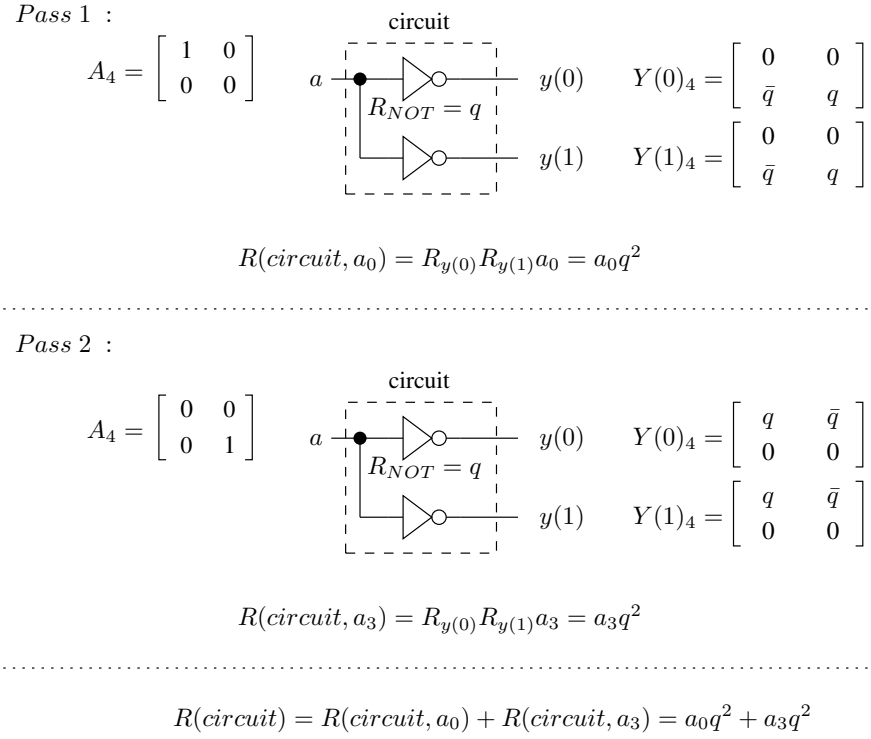


Figure 3.37: Example of application of the multipass algorithm.

The time complexity of the multipass algorithm is exponential with the number of fanouts but the concept of dominant fanouts allows a reduction in the complexity. Furthermore, other optimizations are possible. Since the signal probabilities of fanout signals are computed for each iteration according to a dependency order, when a fanout signal probability results in a null value, the corresponding circuit reliability will also have a null value and the concerned loops can be skipped. An example of this simplification approach follows.

Fig. 3.38 shows the topology of circuit C17, where the fanout signals are $x(1)$, $s(1)$ and $s(2)$.

The multi-pass algorithm cycles through all combinations of fanout states, in a tree-like processing flow. Every node in this tree represents a fanout state (probability of the fanout signal being on that state) or combination of fanout states. These fanout states are successively set to **1** and the signal probabilities of signals on its fanout cone are computed.

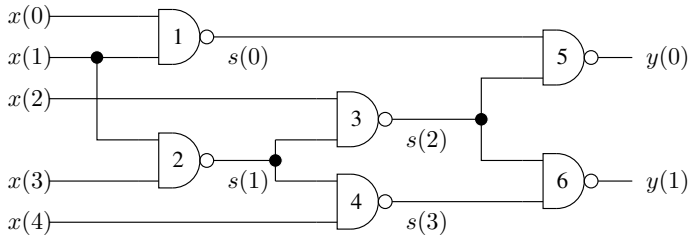


Figure 3.38: Topology of the C17 circuit.

The multi-pass tree representation for the circuit C17 can be seen on Fig. 3.39, where each branch represents a different combination of fanout states. Signal $x(1)$ is a primary input and does not have states representing incorrect logic values. The multi-pass tree in Fig. 3.39 shows shadowed nodes that are associated with null probability values, i.e., the points where the reliability computation for the current iteration can be skipped.

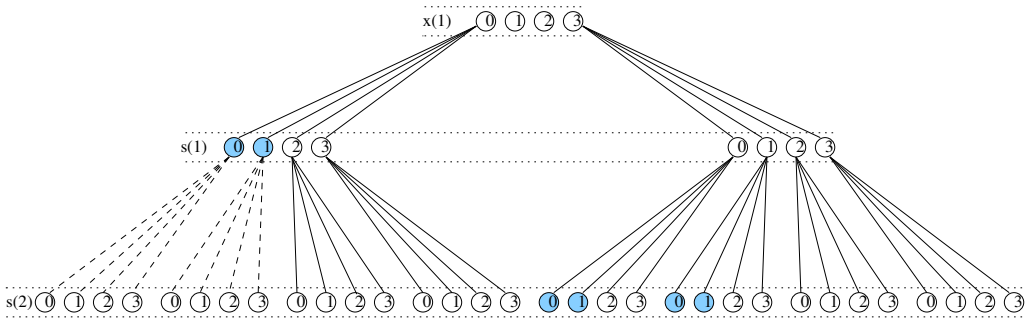


Figure 3.39: Multi-pass tree representation of circuit C17.

The approach of skipping propagation when a signal state has zero probability of occurrence can be extended to values next to zero, defined by a threshold probability value, represented as p_t. This expands the number of branches skipped on the multi-pass tree, reducing the running time of the algorithm. In fact, this approach is equivalent to skip the contribution of multiple faults masking to the reliability of the circuit. The threshold value in this case will determine how many simultaneous faults will be taken into account in the analysis.

Fig. 3.40 illustrates the threshold approach. The figure represents the multi-pass tree representation for circuit C17 considering logic gates with reliability $q = 0.99$ and skipping threshold of $t_p = 0.001$. This reduces the number of relevant paths to 6, from a set of 32 paths in the tree. The calculated circuit reliability is $R = 94.99\%$ and the exact value is $R = 95.19\%$. The threshold value must be defined for every node, as a function of the reliability of the gates.

As discussed before, the main optimization approach on the multi-pass algorithm is a reduction in the number of fanout signals that are considered for the multi-pass propagation. By computing the contribution of each fanout signal to the overall reliability, considerable differences can be observed, indicating the existence of *dominant* fanout signals. Applying the multi-pass algorithm considering only dominant fanout signals can reduce the computing time by orders of magnitude, with a small impact in the accuracy of the computed reliability.

Table 3.4 shows an example of the relation among the number of fanout signals taken

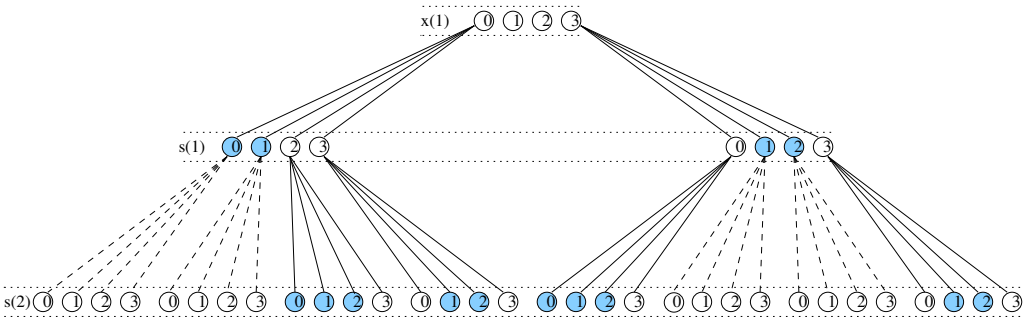


Figure 3.40: Multi-pass tree of circuit C17 with the skipping threshold approach.

into account by the multi-pass algorithm (column *Fanout signals* in the table) and the error (ϵ) in the reliability value, considerin a 130-nm standard cell 8-bit ripple carry adder. The exact value of reliability was obtained by exhaustive simulation.

Table 3.4: Fanout correlation effect: 8-bit ripple carry adder

Fanout signals	ϵ(%)	Elapsed Time (s)
0	21.17	0.004
1	17.84	0.004
3	11.04	0.004
5	4.01	0.012
6	0.98	0.052
9	0.003	1.748

The adder circuit analyzed in Table 3.4 is composed by 18 logic cells and has a total of 24 reconvergent fanout signals (16 primary inputs and 8 internal nodes). The choice of the more relevant ones to the multi-pass algorithm is defined by the logic distance of the branching node to the outputs and the number of branches, i.e., the size of its fanout cone. Experiments with several circuits have shown that this approach gives the best results most of the time.

3.6 Proposed methodologies

This chapter has detailed two methodologies that were proposed in the current work to evaluate the signal reliability of combinational circuits, the probabilistic binomial model PBR and the signal probability model SPR. It has also detailed the PTM approach, an interesting methodology for exact analysis of logic circuits under multiple simultaneous faults.

The development of the two methodologies focused on slightly different objectives. The PBR approach was aimed to compute the exact value of signal reliability of the target circuits, based on well known tools and simple modeling. The result is a methodology that automates the process of computing the logical masking capabilities of a circuit, using procedures that are already present in the design process. The main drawback of the PBR methodology is the need for fault simulation or fault emulation, tasks that are time consuming. Despite this drawback, the analytical solution proposed by the PBR approach is time saving from the point of view of reusability, allowing successive and repetitive analysis

of the simulated circuit by means of a simple reliability expression resolution. Furthermore, the presented optimizations allows a reduction of the simulation/emulation time according to the type of analysis to be made. This way, the PBR approach is indicated as an accurate reliability analysis methodology, that can be used for a detailed analysis of final versions of the circuits, and as a validation tool for other reliability analysis methodologies.

The SPR approach was aimed to a fast evaluation of the signal reliability of the target circuits, based on straightfoward operations and circuit modeling. The heuristics proposed to improve the accuracy of the SPR methodology are adequate for a fast evaluation of the signal reliability, as well as an accurate evaluation, at the price of a longer running time. The incremental accuracy available in the SPR multi-pass analysis is interesting to provide a mid-range solution between the DWAA linear approach and the PBR approach. Considering the characteristics of the SPR heuristics, the focus of the concerned analysis is the evaluation of the circuit reliability during the design process, where the synthesis tool can take the computed reliability value into account to optimize the target circuit, by means of the use of more reliable cells or different circuit topology. Considering the straightforward modeling of fault-masking in the SPR methodology, other heuristics are exploitable to optimize the accuracy of the proposed approach.

The PTM approach is also an interesting methodology but a simplification of the method is not so simple, and with the limitation on the size of the matrices that are necessary, its usage is highly restricted.

Next chapter shows the analysis of the circuits on the fault-tolerant library by the proposed methodologies.

Chapter 4

Results

4.1 Introduction

The main objectives of the current work are the development of new reliability analysis methodologies and the evaluation of some fault-tolerant approaches that are considered candidates for implementation in nanotechnology architectures. To validate the proposed reliability analysis methodologies, some of the circuits in the fault-tolerant library have been evaluated by the developed tools, allowing an analysis of the strengths and weaknesses of the methodologies. The analysis were also targeted to evaluate the efficiency of the fault tolerance and fault prevention approaches discussed in the current work.

One of the objectives of this study was to integrate the reliability analysis in the design flow of integrated circuits. To our understanding, this involves the development of stand alone tools and routines that can deal with files issued from the synthesis process and the possibility of guiding the synthesis process according to reliability parameters. This objective has been met by the means of the development of interfacing and parsing tools that generated the adequate reliability description models from the HDL files created by the synthesis tool.

The validation of the reliability analysis tools mainly concerns their performance and practical limits from the point of view of the implemented simplifications, since the proposed methodologies model the exact behavior of the signal reliability metric. The tradeoffs between accuracy and processing time will be evaluated considering the relaxation procedures discussed on the current work, in order to allow the application of the analysis to practical systems. The performance and characteristics of the proposed methodologies will be discussed on the sections that follow.

4.2 The analysis tools

The analysis tools have been applied in the evaluation of simple circuits, to verify their accuracy and computing time. The C17 circuit is not a circuit of the fault-tolerant library but it has been chosen as one of the basic circuits in our study because of its small size, that allows a fast and detailed evaluation of its characteristics and it is adequate for debugging purposes.

Fig. 4.1 shows the reliability behavior of the C17 circuit, as a function of the reliability of its gates, as computed by all the proposed methods. In the figure, the curve representing the PBR, PTM and multi-pass heuristic (SPR_MP) represents the exact signal reliability

of the circuit. The SPR model does shows the higher divergence from the correct value, were the reconvergent fanout signals are the source of inaccuracy, while the DWAA heuristic represents a good approximation of the exact value.

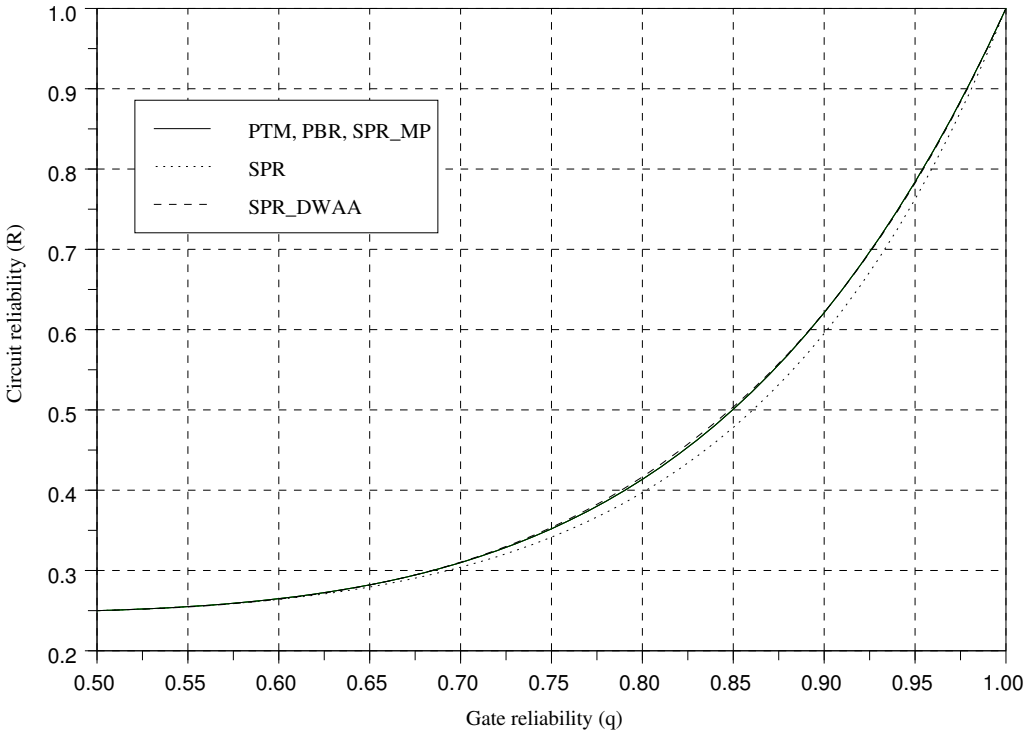


Figure 4.1: Reliability of circuit C17 as a function of gate's reliability.

Table 4.1 presents the analysis running time and relative error values (ϵ) for the referred analysis, with the values obtained with the PBR and PTM approaches being used as the reference. All the running times presented on the current section consider an Intel Core Duo 2.4GHz PC, 2GB of RAM, under Linux OS, as the standard platform. The execution time presented at table 4.1 concerns the computation of 50 points of circuit reliability, i.e., 50 different values of gate reliability q. In the case of the PBR approach, the presented execution time concerns the fault simulation time and the respective reliability computation by the Scilab tool. It's important to observe that once the coefficients c_k have been obtained for a circuit, the PBR analysis consists only in solving the adequate reliability expression in the Scilab tool, as presented in the previous section, that is generally faster than the SPR linear time.

Table 4.1: C17 analysis accuracy and execution time.

PBR		PTM		SPR		SPR_MP		SPR_DWAA	
$t(s)$	$\epsilon(\%)$	$t(s)$	$\epsilon(\%)$	$t(s)$	$\epsilon(\%)$	$t(s)$	$\epsilon(\%)$	$t(s)$	$\epsilon(\%)$
3.3	0.0	0.008	0.0	0.004	4.51	0.02	0.0	0.004	0.08

Table 4.2 shows the same evaluation parameters, concerning the analysis of a 4-bit ripple-carry adder. Unfortunately, this circuit size represents the limit of analysis for the PTM approach. Despite the fact that the PTM of the adder circuit can be represented by a $2^8 \times 2^5$ matrix, the intermediary matrices must be computed and the *height* of a

level is limited to 20 components (including wires). Circunventing techniques could be applied, but looking at the running time and the small size of the concerned circuit, the circunventing techniques would not be of much help. The PTM approach remains an interesting reliability analysis technique but with application limited to small logic blocks.

Table 4.2: 4-bit ripple-carry adder, analysis accuracy and execution time.

PBR		PTM		SPR		SPR_MP		SPR_DWAA	
$t(s)$	$\epsilon(\%)$	$t(s)$	$\epsilon(\%)$	$t(s)$	$\epsilon(\%)$	$t(s)$	$\epsilon(\%)$	$t(s)$	$\epsilon(\%)$
3.83	0.0	287.65	0.0	0.004	13.93	22.98	0.0	0.004	11.32

The error concerning the SPR approach is in the same level of most of the approaches discussed at section 3.2, with the SPR_DWAA heuristic leading to better results. It is important to consider that these error values are related to gate reliabilities around $q = 0.8$, a value that is not realistic for current technologies, and for this reliability range, the correlation effect is possibly maximized by the SPR algorithm. For a gate reliability of $q = 0.9$, the error of the SPR analysis is in the order of 7.56% and for $q = 0.99$ the error is 0.29%.

It is important to highlight this effect of the gate reliability range on the analysis, since this range has fundamental influence on the results. As detailed in section 3.4, the signal reliability attribute of a circuit can be modeled by the cumulative effect of the fault masking property of the circuit. This way, the signal reliability attribute can be characterized by a lower bound value, and the cumulative effect of the c_k coefficients, representing the fault masking capacity of the circuit to k simultaneous faults. As shown by Fig. 3.20, the influence of the fault masking capacity to the overall reliability is highly dependent on the value of the gate reliability range.

Consider a circuit with G gates or cells. Define the lower bound of the signal reliability (R_L) as in (4.1), where q_i is the reliability of gate i. The lower bound represents the probability of a fault-free operation of the circuit, and summing up to it the probability of fault masking (R_{FM}) results in the signal reliability (R), i.e., $R = R_L + R_{FM}$.

$$R_L = \prod_{i=1}^{G} q_i \tag{4.1}$$

None of the approaches proposed by other works (section 3.2) discusses the lower bound value of reliability, and some of the results computed as the reliability in these works are in reality below the lower bound. In the present work, we have considered the lower bound value as a limiting factor for the signal reliability computation, that is automatically taken into account by the implemented SPR tools. This indicates that depending on the gate reliability values considered on the analysis, the faster and more accurate way of reliability evaluation is to compute the lower bound value of the reliability.

Considering the implemented tools, that accept different gate reliability values, a simplified analysis of the contribution of each gate to the reliability of a given circuit is possible, with a simple modification in the basic algorithm. This type of analysis is executed by considering all gates as fault-free ($q = 1$), with the exception of the specific gate of interest, that is considered in error state ($q_{target} = 0$), and computing the reliability of the circuit by means of the SPR_MP or SRP_DWAA algorithms. A specific tool has been implemented to run this analysis for all gates in a circuit. Fig. 4.2 shows an example of the results obtained with this tool, where the values at each gate represent the reliability

of the circuit when the respective gate fails. This analysis indicates the gates that should be considered first for hardening approaches.

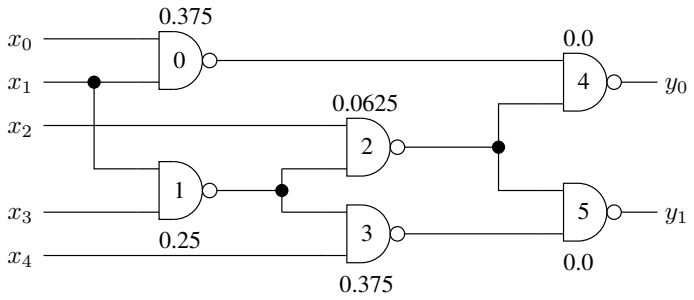


Figure 4.2: Reliability of circuit C17 as a function of single gate failure scenario.

Table 4.3 shows the analysis results for an 8-bit ripple-carry adder, with gate reliability varying on the range $0.9 \leq q \leq 1$, with 10 reliability values computed. This circuit surpasses the practical circuit size for the application of the *exhaustive* multi-pass heuristic, i.e., the analysis considering all of the reconvergent fanouts, that are 23 for this adder. It also represents the limit for the exhaustive fault simulation for the PBR approach. The SPR and the SPR_DWAA have obtained the same reliability values, indicating that the two approaches have arrived at the lower reliability bound. This is a particular characteristic of the ripple-carry adder, that is highly based on XOR functions, that do not mask faults, leading to reduced contribution of fault masking to the reliability of the circuit.

Table 4.3: 8-bit ripple-carry adder, analysis accuracy and execution time.

PBR		PTM		SPR		SPR_MP		SPR_DWAA	
$t(s)$	$\epsilon(\%)$	$t(s)$	$\epsilon(\%)$	$t(s)$	$\epsilon(\%)$	$t(s)$	$\epsilon(\%)$	$t(s)$	$\epsilon(\%)$
50684	0.0	-	-	0.004	10.19	29218	0.0	0.012	10.19

To allow the analysis of larger size circuits, the simplifications discussed in the previous chapter must be considered, i.e., a reduction in the number of coefficients c_k, pseudo-random simulation and fault emulation for the PBR approach, and the reduction of reconvergent fanouts taken into account in the SPR_MP approach. The analysis results with these simplifications are shown in table 4.4.

Table 4.4: 8-bit adder analysis based on simplified assumptions.

PBR		PBR emulation		PBR random		SPR_MP		SPR_DWAA	
$t(s)$	$\epsilon(\%)$	$t(s)$	$\epsilon(\%)$	$t(s)$	$\epsilon(\%)$	$t(s)$	$\epsilon(\%)$	$t(s)$	$\epsilon(\%)$
50684	0.0	326	0.0	159.67	1.47	2.27	2.3	0.012	10.19

As presented in the table, the emulation represents an speed-up of 155 times compared to the fault simulation approach but the pseudo-random fault-simulation also presents interesting results. In this case, the number of pseudo-random vectors applied to the circuit was 4,194,304 (1024 fault vectors, 4096 input vectors) for each set of fault vectors with k simultaneous faults, for the range $1 \leq k \leq 6$. The multi-pass heuristic has also shown very good accuracy and reasonable execution time, for 7 reconvergent fanout signals considered. The reduction of the number of fanout signals in the DWAA heuristic has not produced any effect on the result.

Table 4.5: 8-bit carry-select adder analysis.

PBR emulation		PBR random		SPR_MP		SPR_DWAA	
$t(s)$	$\epsilon(\%)$	$t(s)$	$\epsilon(\%)$	$t(s)$	$\epsilon(\%)$	$t(s)$	$\epsilon(\%)$
54976	0.0	252.73	0.21	185.77	0.0	0.02	7.08

The results of the analysis of a 8-bit carry-select adder can be seen on table 4.5. The fault emulation approach shows its practical limits, with 4 replicas of the target circuit operating in parallel at a frequency of 80 MHz. The pseudo-random fault simulation and the multi-pass heuristic show good results concerning execution time and accuracy, with the multi-pass approach resulting in exact values, based on 10 reconvergent fanout signals (out of 26). The error values concern the gate reliability of $q = 0.99$.

Considering the limitations of some of the implemented approaches, the ones that are candidates for larger circuit analysis are pseudo-random simulation (taken as the reference) and the SPR approach by means of the DWAA and multi-pass heuristics. Table 4.6 shows the analysis results for the 8-bit Booth multiplier, for 10 reliability values in the range $0.99 \leq q \leq 1$. As can be seen, the DWAA algorithm takes the advantage over the multi-pass heuristic, that is constrained by the execution time. An exact value can be obtained with the multi-pass approach but the execution time will exceed that of the pseudo-random simulation. As an example, by considering 11 fanout signals (instead of 8) for the multi-pass algorithm, the error drops to 31.7% but the execution time rises to 857s. On the other side, the DWAA algorithm gives the lower bound of the reliability, that is close to the exact reliability value for the computed range of q. The error values are related with $q = 0.999$. The booth multiplier presents a particular architecture, represented by the recoding mechanism, and this particular architecture is possibly the responsible for the bad results obtained with the multi-pass heuristic, that chooses the dominant fanout signals according to its *distance* to the primary outputs. The recoding mechanism possibly changes the contribution of fanout signals to primary inputs and another heuristic for choosing the dominant fanouts may be more adequate in this case. The multi-pass approach, in this case, did not take into account the lower bound value of reliability.

Table 4.6: 8-bit Booth multiplier analysis

PBR random		SPR_MP		SPR_DWAA	
$t(s)$	$\epsilon(\%)$	$t(s)$	$\epsilon(\%)$	$t(s)$	$\epsilon(\%)$
2199	0.0	172.36	36.29	0.39	0.62

Table 4.7 shows the analysis results for an 8-bit self-checking ripple-carry adder (RC_SC 8) and an 8-bit signed digit adder, for $q = 0.999$. The number of coefficients c_k considered on the pseudo-random simulation is shown at column **max** k, for 2^{24} (RC_SC 8) and 2^{20} (SD 8) fault and input vectors. The number of fanouts considered by the multi-pass algorithm is shown at column **nodes**. For these two circuits, both methodologies give similar results. Considering the size of the signed digit adder, that is larger than the 8-bit Booth multiplier, the error related with the multi-pass approach can be considered as more dependent on the circuit topology than its size.

The present section had the objective of determining the adequacy of the proposed reliability analysis approaches concerning the evaluation of logic circuits. The results obtained clearly show the advantages and drawbacks of each method, as follows:

Table 4.7: Reliability analysis results.

Circuit type	Cells	PBR			SPR_MP			SPR_DWAA	
		max k	$t(s)$	$\epsilon(\%)$	nodes	$t(s)$	$\epsilon(\%)$	$t(s)$	$\epsilon(\%)$
RC_SC 8	51	15	7865	0	7/49	1.73	0.37	0.016	0.34
SD 8	193	40	13253	0	9/80	5.69	0.02	0.032	7.88

- **PTM**. The PTM approach allows an exact reliability evaluation but is strongly limited in terms of circuit size, pointing to an utilization targeted to the study of small basic blocks.

- **PBR**. The dependence on fault simulation/emulation limits the scope of interest on the PBR approach, reducing its potential for usage in reliability-aware automated design. Nevertheless, it remains the best approach concerning accuracy, and is adequate for precise evaluation of the reliability of final versions of the circuits.

- **SPR**. The signal probability approach has shown promising features, but its usage is dependent on the heuristic for dealing with signal correlations. In the present work, two of these heuristics have been explored, and the results seem to be adequate for most of the evaluated circuits. The multi-pass approach represents a good compromise for small and medium size circuits, but with a dependence on the topology of the circuit. The DWAA approach presents better results for larger circuits, with the reliability approaching the lower bound. For larger circuits (several outputs), the contribution of fault masking to the reliability is reduced, and the computation of the lower bound of reliability is generally the best choice.

The application of the DWAA and multi-pass heuristics can be made in a complementary way, with the DWAA approach indicating the proximity of the multi-pass results from the lower bound. The strenght of the SPR approach remains its simplicity of application, allowing the exploration of other heuristics, that can contribute with some *intelligence* to a straightforward process.

Considering the integration of the proposed tools in the design flow, the objective has been accomplished, with target circuits generated by standard-cell-based synthesis tools. The implemented tools generate reliability report files that can be read by a synthesis script, controlling some aspects of the optimization of the circuits.

The next section discusses the reliability of the circuits of the fault-tolerant library, as determined by the implemented analysis tools.

4.3 The signal reliability

The signal reliability of some of the circuits in the fault-tolerant library was determined to allow a direct evaluation of the vulnerability of these circuits to transient faults. It's important to consider that signal reliability and functional reliability are different metrics, that focus on different properties of the circuits. The difference mainly concerns the CED fault-tolerant approaches, where the hardware redundancy impacts negatively on the signal reliability, while increases the functional reliability. It is up to the designer to decide whether this tradeoff is acceptable.

Fig. 4.3 shows the reliability of the conventional 8-bit adders, according to the reliability of its individual gates. As expected, the reliability of the circuits is primarily dependent on the number of cells, and the advantage is on the side of the smaller circuits.

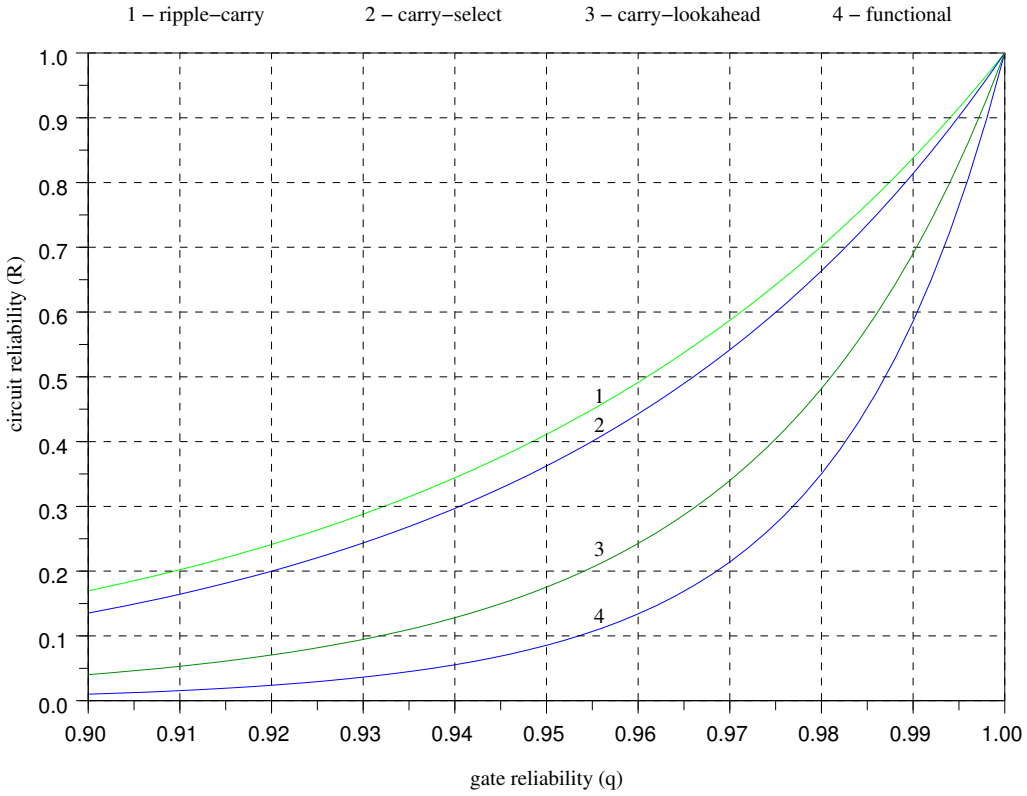


Figure 4.3: Reliability of the conventional adders.

This traditional representation of the reliability behavior, as presented in Fig. 4.3, is only adequate for a range of q that is much lower than the actual values for current technologies. Fig. 4.4 shows the same analysis data with the gate reliability represented in logarithmic scale.

As presented in Fig. 4.4, a linear representation of the reliability metric is not useful for more realistic reliability values ($q = 0.9999$ and over, i.e., 10000 FIT or less), since the reliability of the circuits cannot be distinguished in the linear graph for gate reliability (q) values next to 1. A better representation is the relation between the failure rate of the cells and the failure rate of the circuit. The MTBF of the circuits can be computed according to expression 4.2, as defined in section 2.2.

$$R(t) = e^{-\frac{t}{MTBF}} \tag{4.2}$$

Fig. 4.5 shows the same analysis presented in figures 4.3 and 4.4, but expressing the MTBF value of the circuits, allowing a direct comparison of the concerned values. The ripple-carry adder shows a better reliability (16% higher MTBF) than the carry-select adder, with the carry-lookahead and the functional adder presenting 2 to 3 times the failure rate of the ripple-carry adder.

Fig. 4.6 shows the MTBF of the carry-lookahead adder and its fault-tolerant versions, for 8-bit data width. As discussed before, the failure rate computed by the proposed

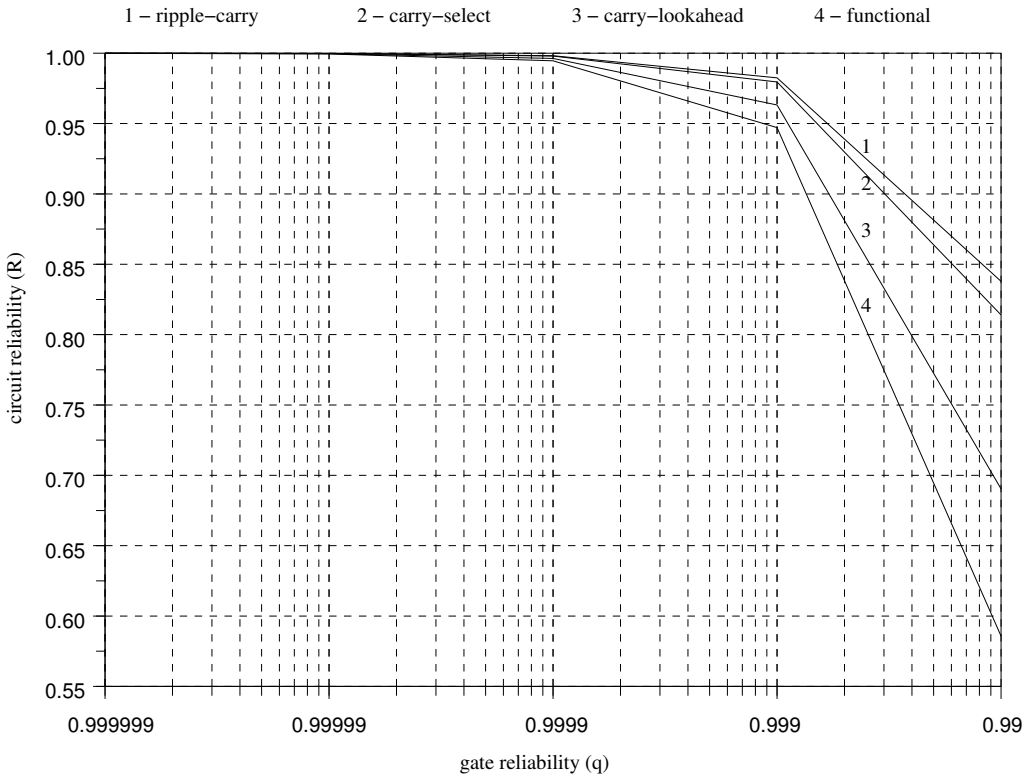


Figure 4.4: Reliability of the conventional adders with gate reliability in logarithmic scale.

methods is related to the signal reliability of the circuits, and the curves show that the self-checking approaches present a failure rate that is higher than the original circuit they are supposed to protect. This is a parameter that must be considered when evaluating fault-tolerant approaches. On the other way, the TMR version of the adder presents a relevant gain in the MTBF of the circuit, achieving a failure rate that is almost 3 times lower than the original circuit. These results consider two-rail checkers and TMR voters as reliable as the remaining cells.

Fig. 4.7 presents the MTBF of the 8-bit multipliers and their fault-tolerant versions. The failure rate of the fault-tolerant versions of the multipliers is a around 2 times higher in the worst case.

The raw data concerning the MTBF of the circuits in the fault-tolerant library can be seen on table 4.8, for 8-bit data widths. The nomenclature used in the table corresponds to the following definitions: **rca**, ripple-carry adder; **csa**, carry-select adder; **cla**, carry-lookahead adder; **fca**, functional adder; **sd2**, radix-2 signed digit adder; **sd3**, 1-out-of-3 signed digit adder; **booth**, Booth multiplier; **fcm**, functional multiplier. For the fault-tolerant versions of the circuits, the nomenclature used is: **dup**, duplication-based fault tolerance; **par**, parity-based fault tolerance; **tmr2**, interlaced triple modular redundance; **1o3**, 1-out-of-3 checking.

The results show the expected trend of reduced reliability related with cell count increase, with the exception of the TMR circuits, that have a much higher degree of fault masking than the other topologies. The effect of circuit topology has also a relevant impact on the reliability for the TMR versions of the circuits, as seen on the MTBF of the functional adder, the most reliable circuit among the ones listed, that has a number of cells

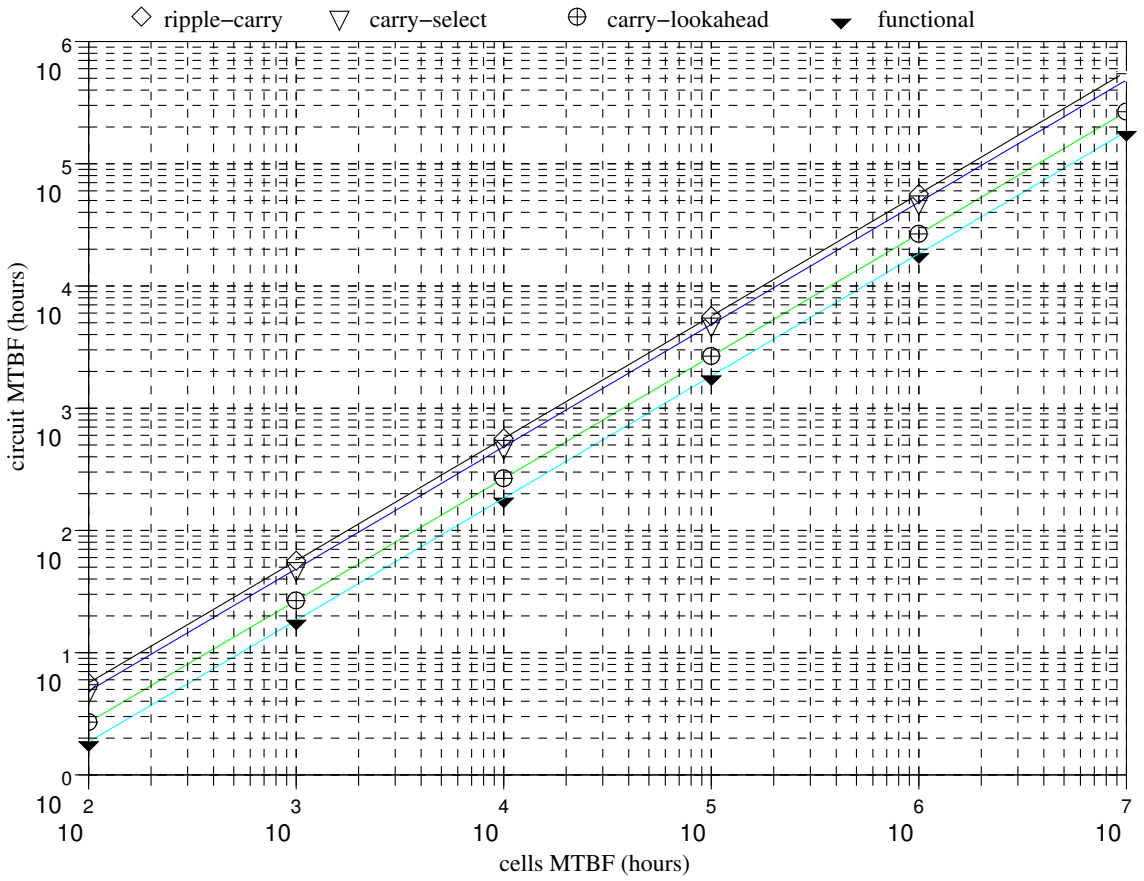


Figure 4.5: Failure rate of the conventional adders.

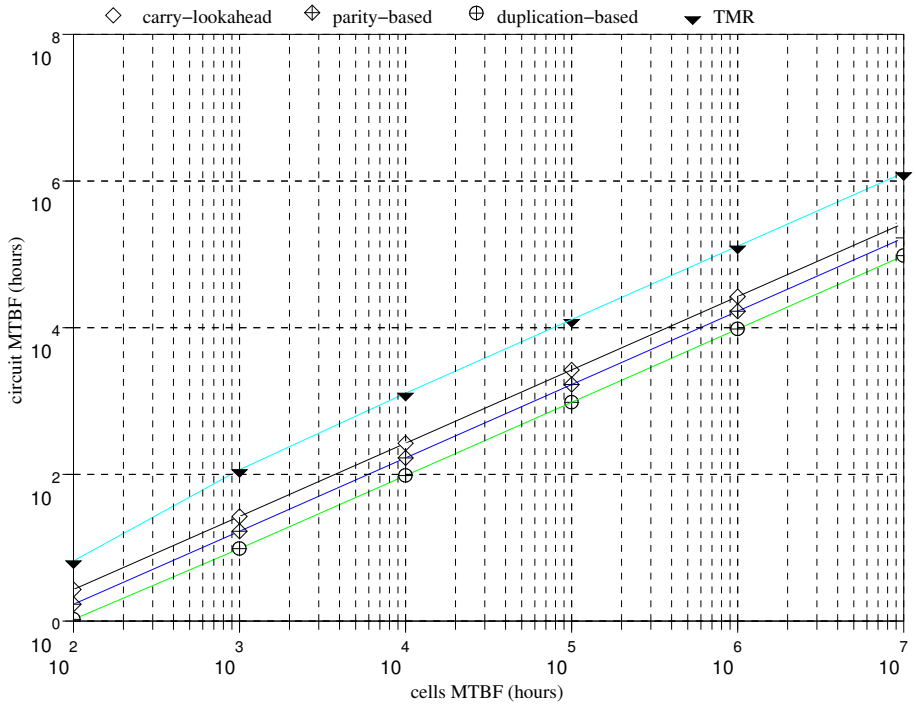


Figure 4.6: Failure rate of different versions of the carry-lookahead adder.

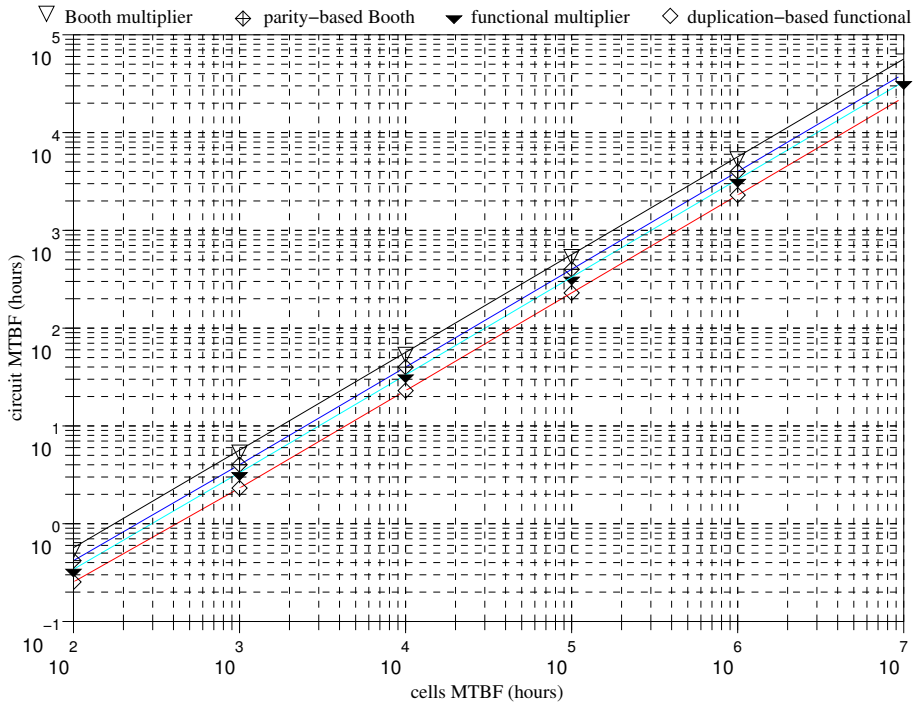


Figure 4.7: Failure rate of different versions of 8-bit multipliers.

that is superior to the other TMR adders.

The MTBF values have been determined with the Scilab tool, considering the expressions defined by the PBR model (section 3.4), based on pseudo-random fault simulation.

The raw data concerning the MTBF of some of the 12-bit circuits can be seen on table 4.9.

The raw data concerning the 16-bit adders can be seen on table 4.10. As a general trend, the MTBF of the 16-bit circuits is a litle bit less than half of the MTBF of the equivalent 8-bit circuits, in direct relation with the size of the circuits.

The results show that the self-checking fault-tolerant approaches have a negative impact on the signal reliability of the circuits, and that the TMR approach, by explicitly introducing logical masking can improve the signal reliability. The real impact of self-checking architectures is to improve the functional reliability of the circuits, targeting a given fault model, usually single stuck-at faults. As discussed in section 3.4, the behavior of self-checking circuits under multiple faults can be determined by computing a number of different events, defined as ξ (correct operation with fault masking), τ (checker error with fault detection), ψ (silent error) and χ (correct operation with error detection). These events have associated probabilities $E(q)$, $T(q)$, $U(q)$ and $X(q)$, respectively, and allow the determination of the functional reliability (fault-secureness) and the time penalty characteristics of the CED circuits.

Consider the self-checking version of the 8-bit carry-lookahead adder based on parity. The probability of occurrence of the defined events can be seen on Fig. 4.8. As shown in the figure, the probability of occurrence of a silent error increases with the increase of the probability of multiple faults.

Fig. 4.9, shows the signal and functional reliability behavior of the adder analyzed in Fig. 4.8. Although the functional reliability remains high, for $q < 0.99$ the majority of

Table 4.8: MTBF of 8-bit circuits of the fault-tolerant library.

Circuit type	Cells MTBF (hours)					
	10^2	10^3	10^4	10^5	10^6	10^9
rca	$5.65 \cdot 10^0$	$5.66 \cdot 10^1$	$5.66 \cdot 10^2$	$5.66 \cdot 10^3$	$5.66 \cdot 10^4$	$5.66 \cdot 10^7$
rca dup	$1.69 \cdot 10^0$	$1.57 \cdot 10^1$	$1.56 \cdot 10^2$	$1.56 \cdot 10^3$	$1.56 \cdot 10^4$	$1.56 \cdot 10^7$
rca par	$2.22 \cdot 10^0$	$2.18 \cdot 10^1$	$2.18 \cdot 10^2$	$2.18 \cdot 10^3$	$2.18 \cdot 10^4$	$2.18 \cdot 10^7$
rca tmr	$9.81 \cdot 10^0$	$11.3 \cdot 10^1$	$11.5 \cdot 10^2$	$11.5 \cdot 10^3$	$11.5 \cdot 10^4$	$11.5 \cdot 10^7$
rca tmr2	$10.1 \cdot 10^0$	$11.3 \cdot 10^1$	$11.5 \cdot 10^2$	$11.5 \cdot 10^3$	$11.5 \cdot 10^4$	$11.5 \cdot 10^7$
csa	$4.86 \cdot 10^0$	$4.84 \cdot 10^1$	$4.84 \cdot 10^2$	$4.84 \cdot 10^3$	$4.84 \cdot 10^4$	$4.84 \cdot 10^7$
csa dup	$1.53 \cdot 10^0$	$1.42 \cdot 10^1$	$1.41 \cdot 10^2$	$1.41 \cdot 10^3$	$1.41 \cdot 10^4$	$1.41 \cdot 10^7$
csa par	$2.04 \cdot 10^0$	$2.01 \cdot 10^1$	$2.00 \cdot 10^2$	$2.00 \cdot 10^3$	$2.00 \cdot 10^4$	$2.00 \cdot 10^7$
csa tmr	$8.67 \cdot 10^0$	$12.2 \cdot 10^1$	$12.9 \cdot 10^2$	$12.9 \cdot 10^3$	$13.0 \cdot 10^4$	$13.0 \cdot 10^7$
cla	$2.70 \cdot 10^0$	$2.67 \cdot 10^1$	$2.66 \cdot 10^2$	$2.66 \cdot 10^3$	$2.66 \cdot 10^4$	$2.66 \cdot 10^7$
cla dup	$1.05 \cdot 10^0$	$0.97 \cdot 10^1$	$0.96 \cdot 10^2$	$0.96 \cdot 10^3$	$0.96 \cdot 10^4$	$0.96 \cdot 10^7$
cla par	$1.70 \cdot 10^0$	$1.68 \cdot 10^1$	$1.68 \cdot 10^2$	$1.68 \cdot 10^3$	$1.68 \cdot 10^4$	$1.68 \cdot 10^7$
cla tmr	$6.61 \cdot 10^0$	$11.6 \cdot 10^1$	$12.8 \cdot 10^2$	$13.0 \cdot 10^3$	$13.0 \cdot 10^4$	$13.0 \cdot 10^7$
fca	$1.87 \cdot 10^0$	$1.84 \cdot 10^1$	$1.84 \cdot 10^2$	$1.84 \cdot 10^3$	$1.84 \cdot 10^4$	$1.84 \cdot 10^7$
fca dup	$0.78 \cdot 10^0$	$0.71 \cdot 10^1$	$0.71 \cdot 10^2$	$0.71 \cdot 10^3$	$0.71 \cdot 10^4$	$0.71 \cdot 10^7$
fca tmr	$4.51 \cdot 10^0$	$11.2 \cdot 10^1$	$14.2 \cdot 10^2$	$14.6 \cdot 10^3$	$14.7 \cdot 10^4$	$14.7 \cdot 10^7$
SD2	$1.04 \cdot 10^0$	$1.02 \cdot 10^1$	$1.02 \cdot 10^2$	$1.02 \cdot 10^3$	$1.02 \cdot 10^4$	$1.02 \cdot 10^7$
SD2 par	$0.50 \cdot 10^0$	$0.39 \cdot 10^1$	$0.39 \cdot 10^2$	$0.39 \cdot 10^3$	$0.39 \cdot 10^4$	$0.39 \cdot 10^7$
SD3	$0.75 \cdot 10^0$	$0.73 \cdot 10^1$	$0.73 \cdot 10^2$	$0.73 \cdot 10^3$	$0.73 \cdot 10^4$	$0.73 \cdot 10^7$
SD3 1o3	$0.60 \cdot 10^0$	$0.58 \cdot 10^1$	$0.58 \cdot 10^2$	$0.58 \cdot 10^3$	$0.58 \cdot 10^4$	$0.58 \cdot 10^7$
booth	$0.57 \cdot 10^0$	$0.57 \cdot 10^1$	$0.57 \cdot 10^2$	$0.57 \cdot 10^3$	$0.57 \cdot 10^4$	$0.57 \cdot 10^7$
booth par	$0.42 \cdot 10^0$	$0.40 \cdot 10^1$	$0.40 \cdot 10^2$	$0.40 \cdot 10^3$	$0.40 \cdot 10^4$	$0.40 \cdot 10^7$
fcm	$0.34 \cdot 10^0$	$0.33 \cdot 10^1$	$0.33 \cdot 10^2$	$0.33 \cdot 10^3$	$0.33 \cdot 10^4$	$0.33 \cdot 10^7$
fcm dup	$0.25 \cdot 10^0$	$0.23 \cdot 10^1$	$0.23 \cdot 10^2$	$0.23 \cdot 10^3$	$0.23 \cdot 10^4$	$0.23 \cdot 10^7$

Table 4.9: MTBF of 12-bit circuits of the fault-tolerant library.

Circuit type	Cells MTBF (hours)					
	10^2	10^3	10^4	10^5	10^6	10^9
rca	$3.87 \cdot 10^0$	$3.88 \cdot 10^1$	$3.88 \cdot 10^2$	$3.88 \cdot 10^3$	$3.88 \cdot 10^4$	$3.88 \cdot 10^7$
rca dup	$1.21 \cdot 10^0$	$1.07 \cdot 10^1$	$1.06 \cdot 10^2$	$1.06 \cdot 10^3$	$1.06 \cdot 10^4$	$1.06 \cdot 10^7$
rca par	$1.45 \cdot 10^0$	$1.39 \cdot 10^1$	$1.39 \cdot 10^2$	$1.39 \cdot 10^3$	$1.39 \cdot 10^4$	$1.39 \cdot 10^7$
rca tmr	$6.90 \cdot 10^0$	$8.98 \cdot 10^1$	$9.34 \cdot 10^2$	$9.37 \cdot 10^3$	$9.37 \cdot 10^4$	$9.37 \cdot 10^7$
csa	$3.11 \cdot 10^0$	$3.11 \cdot 10^1$	$3.11 \cdot 10^2$	$3.11 \cdot 10^3$	$3.11 \cdot 10^4$	$3.11 \cdot 10^7$
csa dup	$1.06 \cdot 10^0$	$0.95 \cdot 10^1$	$0.94 \cdot 10^2$	$0.94 \cdot 10^3$	$0.94 \cdot 10^4$	$0.94 \cdot 10^7$
csa par	$1.24 \cdot 10^0$	$1.21 \cdot 10^1$	$1.20 \cdot 10^2$	$1.20 \cdot 10^3$	$1.20 \cdot 10^4$	$1.20 \cdot 10^7$
csa tmr	$5.66 \cdot 10^0$	$8.42 \cdot 10^1$	$8.98 \cdot 10^2$	$9.05 \cdot 10^3$	$9.05 \cdot 10^4$	$9.05 \cdot 10^7$
cla	$1.73 \cdot 10^0$	$1.71 \cdot 10^1$	$1.71 \cdot 10^2$	$1.71 \cdot 10^3$	$1.71 \cdot 10^4$	$1.71 \cdot 10^7$
cla dup	$0.71 \cdot 10^0$	$0.63 \cdot 10^1$	$0.62 \cdot 10^2$	$0.62 \cdot 10^3$	$0.62 \cdot 10^4$	$0.62 \cdot 10^7$
cla par	$1.06 \cdot 10^0$	$1.05 \cdot 10^1$	$1.05 \cdot 10^2$	$1.05 \cdot 10^3$	$1.05 \cdot 10^4$	$1.05 \cdot 10^7$
cla tmr	$4.23 \cdot 10^0$	$8.09 \cdot 10^1$	$9.38 \cdot 10^2$	$9.55 \cdot 10^3$	$9.56 \cdot 10^4$	$9.57 \cdot 10^7$
fca	$1.14 \cdot 10^0$	$1.12 \cdot 10^1$	$1.12 \cdot 10^2$	$1.12 \cdot 10^3$	$1.12 \cdot 10^4$	$1.12 \cdot 10^7$
fca dup	$0.51 \cdot 10^0$	$0.44 \cdot 10^1$	$0.44 \cdot 10^2$	$0.44 \cdot 10^3$	$0.44 \cdot 10^4$	$0.44 \cdot 10^7$
fca tmr	$2.45 \cdot 10^0$	$7.21 \cdot 10^1$	$10.7 \cdot 10^2$	$11.3 \cdot 10^3$	$11.4 \cdot 10^4$	$11.4 \cdot 10^7$
booth	$0.28 \cdot 10^0$	$0.27 \cdot 10^1$	$0.27 \cdot 10^2$	$0.27 \cdot 10^3$	$0.27 \cdot 10^4$	$0.27 \cdot 10^7$

Table 4.10: MTBF of 16-bit circuits of the fault-tolerant library.

Circuit type	Cells MTBF (hours)					
	10^2	10^3	10^4	10^5	10^6	10^9
rca	$2.96 \cdot 10^0$	$2.96 \cdot 10^1$	$2.96 \cdot 10^2$	$2.96 \cdot 10^3$	$2.96 \cdot 10^4$	$2.96 \cdot 10^7$
rca dup	$0.96 \cdot 10^0$	$0.82 \cdot 10^1$	$0.81 \cdot 10^2$	$0.81 \cdot 10^3$	$0.81 \cdot 10^4$	$0.81 \cdot 10^7$
rca par	$1.08 \cdot 10^0$	$1.03 \cdot 10^1$	$1.03 \cdot 10^2$	$1.03 \cdot 10^3$	$1.03 \cdot 10^4$	$1.03 \cdot 10^7$
rca tmr	$5.04 \cdot 10^0$	$5.93 \cdot 10^1$	$6.05 \cdot 10^2$	$6.06 \cdot 10^3$	$6.06 \cdot 10^4$	$6.06 \cdot 10^7$
rca tmr2	$5.19 \cdot 10^0$	$6.01 \cdot 10^1$	$6.11 \cdot 10^2$	$6.12 \cdot 10^3$	$6.12 \cdot 10^4$	$6.12 \cdot 10^7$
csa	$2.34 \cdot 10^0$	$2.34 \cdot 10^1$	$2.34 \cdot 10^2$	$2.34 \cdot 10^3$	$2.34 \cdot 10^4$	$2.34 \cdot 10^7$
csa dup	$0.82 \cdot 10^0$	$0.70 \cdot 10^1$	$0.70 \cdot 10^2$	$0.70 \cdot 10^3$	$0.70 \cdot 10^4$	$0.70 \cdot 10^7$
csa par	$0.87 \cdot 10^0$	$0.85 \cdot 10^1$	$0.85 \cdot 10^2$	$0.85 \cdot 10^3$	$0.85 \cdot 10^4$	$0.85 \cdot 10^7$
csa tmr	$3.98 \cdot 10^0$	$6.29 \cdot 10^1$	$6.92 \cdot 10^2$	$6.93 \cdot 10^3$	$6.93 \cdot 10^4$	$6.93 \cdot 10^7$
cla	$1.26 \cdot 10^0$	$1.24 \cdot 10^1$	$1.24 \cdot 10^2$	$1.24 \cdot 10^3$	$1.24 \cdot 10^4$	$1.24 \cdot 10^7$
cla dup	$0.55 \cdot 10^0$	$0.47 \cdot 10^1$	$0.47 \cdot 10^2$	$0.47 \cdot 10^3$	$0.47 \cdot 10^4$	$0.47 \cdot 10^7$
cla par	$0.77 \cdot 10^0$	$0.74 \cdot 10^1$	$0.74 \cdot 10^2$	$0.74 \cdot 10^3$	$0.74 \cdot 10^4$	$0.74 \cdot 10^7$
cla tmr	$2.73 \cdot 10^0$	$5.99 \cdot 10^1$	$7.12 \cdot 10^2$	$7.28 \cdot 10^3$	$7.30 \cdot 10^4$	$7.30 \cdot 10^7$
fca	$0.82 \cdot 10^0$	$0.80 \cdot 10^1$	$0.80 \cdot 10^2$	$0.80 \cdot 10^3$	$0.80 \cdot 10^4$	$0.80 \cdot 10^7$
fca dup	$0.39 \cdot 10^0$	$0.33 \cdot 10^1$	$0.32 \cdot 10^2$	$0.32 \cdot 10^3$	$0.32 \cdot 10^4$	$0.32 \cdot 10^7$
fca tmr	$1.13 \cdot 10^0$	$4.88 \cdot 10^1$	$6.69 \cdot 10^2$	$7.00 \cdot 10^3$	$7.03 \cdot 10^4$	$7.04 \cdot 10^7$

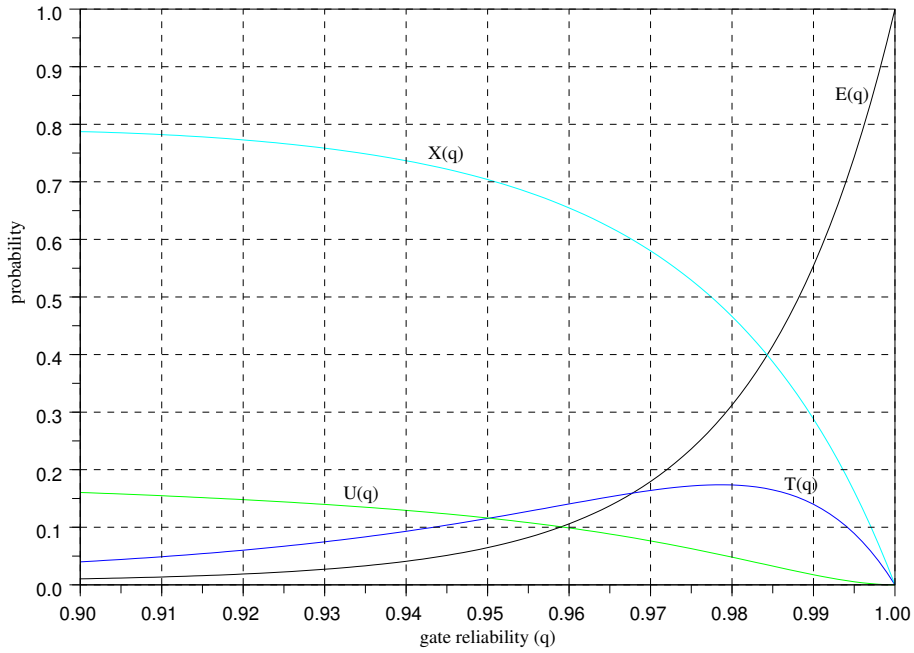


Figure 4.8: Output events on an 8-bit self-checking carry-lookahead adder.

the results are indeed incorrect (see Fig. 4.8), indicating that pre-defined actions must be applied, like re-computing and invalidation of the concerned computing step, and the effect of these actions is a reduction on the flow of useful results. This time penalty can be determined as detailed in section 3.4. Define the *effective frequency* F_{eff} as the frequency of results considered correct by the checker in a self-checking circuit. The effective frequency is a function of the time penalty Θ and the number of clock cycles for correcting or skipping an incorrect result. Assuming the data throughput as 1 data/clock cycle and the number of cycles for correction/skiping of incorrect results as 1, the effective frequency can be computed as in (4.3), where F_0 represents the circuit operating frequency.

$$F_{eff} = F_0 \cdot (1 - \Theta) \tag{4.3}$$

Fig. 4.10 shows the effective frequency behavior for the 8-bit self-checking carry-lookahead adder, as a function of cell's reliability. These results are shown for a reliability range that is not realistic for present day CMOS circuits but that is considered a possible range for future technology generations. These results are intended to show the compromises that are associated with the use of CED approaches with highly unreliable devices. It is also important to consider that the detection/correction overhead has been computed considering the best case time penalty, where only one clock cycle is assumed for correcting actions.

Table 4.11 shows the raw values of effective frequency for the 8-bit CED adder circuits, as a function of the MTBF of the individual cells. As mentioned before, the time penalty for current cell reliability is negligeable.

Table 4.12 shows the raw values of effective frequency for the 16-bit CED adder circuits, as a function of the MTBF of the individual cells.

As presented in the current section, the signal reliability of different fault-tolerant approaches can vary significatively and for multiple simultaneous errors, the functional relia-

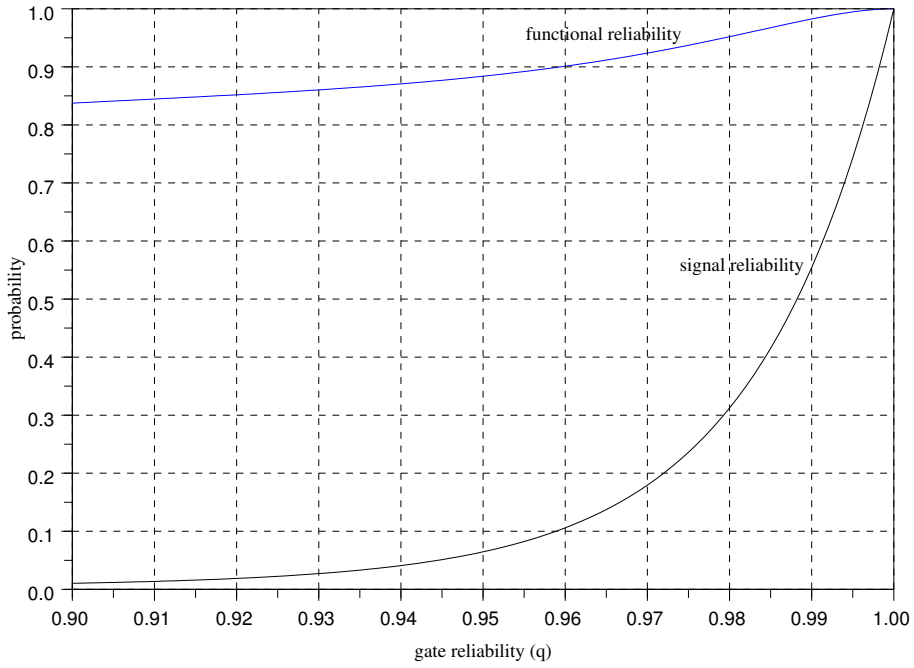


Figure 4.9: Signal and functional reliability of a self-checking carry-lookahead adder.

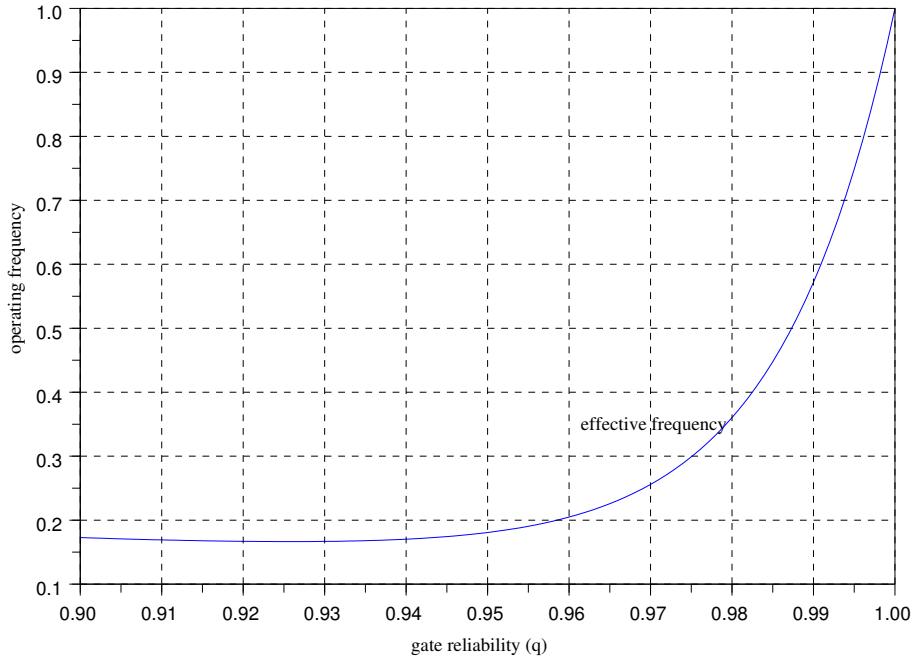


Figure 4.10: Effective frequency behavior of a self-checking carry-lookahead adder.

bility of CED fault-tolerant approaches is related with a time penalty, reducing even further the performance of the protected circuit. These overheads point to the search of alternative approaches, like fault prevention/avoidance, to improve the reliability of nanoscale architectures. Next section explores the characteristics of fault prevention approaches, where restricted redundancy is applied to harden the target circuits.

Table 4.11: Effective frequency for the 8-bit self-checking adders.

Circuit type	Cells MTBF (hours)				
	10^2	10^3	10^4	10^5	10^6
rca dup	$0.5706 \cdot F_0$	$0.9384 \cdot F_0$	$0.9954 \cdot F_0$	$0.9995 \cdot F_0$	$1.0 \cdot F_0$
rca par	$0.6437 \cdot F_0$	$0.9553 \cdot F_0$	$0.9954 \cdot F_0$	$0.9995 \cdot F_0$	$1.0 \cdot F_0$
csa dup	$0.5344 \cdot F_0$	$0.9321 \cdot F_0$	$0.9929 \cdot F_0$	$0.9993 \cdot F_0$	$0.9999 \cdot F_0$
csa par	$0.6210 \cdot F_0$	$0.9518 \cdot F_0$	$0.9951 \cdot F_0$	$0.9995 \cdot F_0$	$1.0 \cdot F_0$
cla dup	$0.4164 \cdot F_0$	$0.9025 \cdot F_0$	$0.9897 \cdot F_0$	$0.9990 \cdot F_0$	$0.9999 \cdot F_0$
cla par	$0.5718 \cdot F_0$	$0.9424 \cdot F_0$	$0.9940 \cdot F_0$	$0.9990 \cdot F_0$	$0.9999 \cdot F_0$
fca dup	$0.3227 \cdot F_0$	$0.8700 \cdot F_0$	$0.9860 \cdot F_0$	$0.9986 \cdot F_0$	$0.9999 \cdot F_0$

Table 4.12: Effective frequency for the 16-bit self-checking adders.

Circuit type	Cells MTBF (hours)				
	10^2	10^3	10^4	10^5	10^6
rca dup	$0.3915 \cdot F_0$	$0.8858 \cdot F_0$	$0.9877 \cdot F_0$	$0.9988 \cdot F_0$	$0.9999 \cdot F_0$
rca par	$0.4208 \cdot F_0$	$0.9078 \cdot F_0$	$0.9903 \cdot F_0$	$0.9990 \cdot F_0$	$0.9999 \cdot F_0$
csa dup	$0.3463 \cdot F_0$	$0.8685 \cdot F_0$	$0.9857 \cdot F_0$	$0.9986 \cdot F_0$	$0.9999 \cdot F_0$
csa par	$0.3352 \cdot F_0$	$0.8891 \cdot F_0$	$0.9883 \cdot F_0$	$0.9988 \cdot F_0$	$0.9999 \cdot F_0$
cla dup	$0.2656 \cdot F_0$	$0.8107 \cdot F_0$	$0.9788 \cdot F_0$	$0.9979 \cdot F_0$	$0.9998 \cdot F_0$
cla par	$0.3293 \cdot F_0$	$0.8753 \cdot F_0$	$0.9866 \cdot F_0$	$0.9987 \cdot F_0$	$0.9999 \cdot F_0$
fca dup	$0.2042 \cdot F_0$	$0.7371 \cdot F_0$	$0.9693 \cdot F_0$	$0.9969 \cdot F_0$	$0.9997 \cdot F_0$

4.4 Fault prevention

The interest of fault prevention techniques is the improvement of circuit reliability by means of restricted redundancy introduction. Instead of guaranteeing fault-secureness for a given fault model, the main idea is the hardening of the most critical cells of the circuits. Some fault prevention approaches have been discussed in section 2.6 and a more detailed evaluation will be developed with the application of the reliability analysis models proposed in the current work. Considering the objective of reliability improvement in an automated design process, the approaches that are better suited are based on cell replication and gate sizing, proposed on the works of Nieuwlan et al. [13] and Zhou and Mohanram [12], respectively.

The cell replication approach determines the most susceptible cells in the circuit and introduces replicas of the vulnerable cells in parallel with the original ones. The gate sizing approach increases the aspect ratio (W/L) of the transistors in vulnerable cells, increasing the capacitance of these cells and reducing their vulnerability to soft errors. This approach is similar to replacing the cells in the design by equivalent ones with higher drive strength, as the ones available on current standard cell libraries. According to the presented data, a vulnerability reduction of an order of magnitude of the cell can be obtained at the price of doubling the area and the power consumption, and with an increase in propagation delay.

To evaluate the effectiveness of these approaches, the PBR and the SPR approaches will be applied in the determination of the most vulnerable cells of the circuits and the reliability of these individual cells will be modified according to the proposed fault prevention techniques. It is important to observe that hardening individual cells does not change the logical masking characteristics of the circuit, but from a simplified point of view, it simply rises the lower bound of the signal reliability.

Consider some of the 8-bit circuits on the library. The analysis of these circuits by the SPR and PBR tools allows the determination of the most vulnerable cells, considering their contribution to the logical masking properties of the circuit, as presented in section 4.2. The SPR approach determines the reliability of the circuit for a single-fault on each one of its cells. A reliability **0** indicates that a fault in the specified cell is not masked by any of the input vectors. The lower the reliability associated to a failing cell, the more dependent the circuit reliability is on this cell, a metric that serves as an indication of the candidate cells for hardening actions.

Fig. 4.11 shows the analysis results for an 8-bit ripple-carry adder, that represents the dependency of the reliability of the circuit to its logic cells.

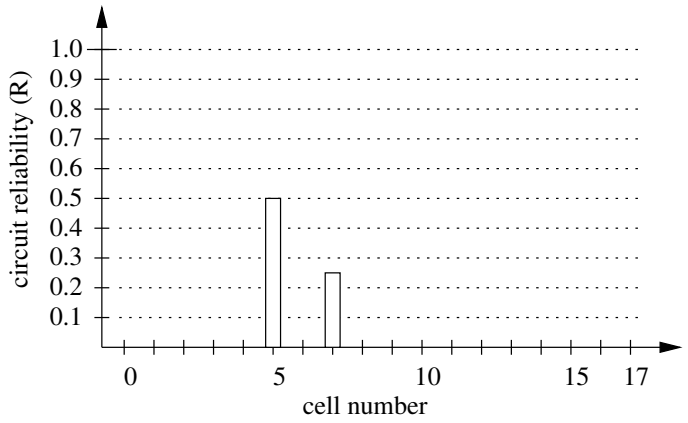


Figure 4.11: Reliability of the ripple-carry adder according to a single-fault on its cells.

As presented in the figure, the ripple-carry adder is highly dependent on almost all of its cells and directly applying the cell replication approach to the most vulnerable cells would lead to a doubling of the circuit area. To restrict the area overhead, the replication can be applied, for example, to the cells generating the primary outputs.

Fig. 4.12 shows the analysis results of cell vulnerability for an 8-bit carry-select adder.

As shown in the figure, the carry-select adder has a more restricted dependency on its cells and the replication approach can be applied to the cells associated with a reliability **0**, for example.

This analysis has been applied to some of the circuits in the library and the improve-

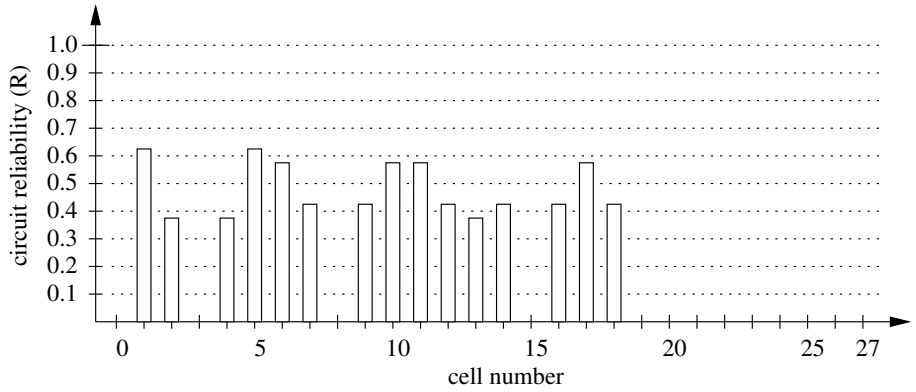


Figure 4.12: Reliability of the carry-select adder according to a single-fault on its cells.

ments in the signal reliability can be seen on table 4.13, where the results are expressed by MTBF values. Each circuit is shown with its original MTBF values, the MTBF for the cell replica approach (nomenclature indicated by *rep*), and the MTBF when the unreliability of each cell is reduced by half (nomenclature indicated by $2x$). This reduction of 50% in the probability of error can be associated with an increase of 50% in the cell area [96].

The results obtained with the cell sizing/replication approaches applied to the vulnerable cells of the target circuits do not show a relevant advantage over the hardening of all cells, and in some cases giving worse results. The exception is the TMR topology, where the reliability of the circuit is highly dependent on the reliability of the voter cells. By hardening the voter cells, the reliability improves considerably, by almost the same rate as the reliability improvement in the voter cells. Table 4.14 shows the improvements of the MTBF observed in the discussed analysis, as well as the area overhead associated with the fault prevention approach. Consider as a reference that hardening all cells in a circuit to reduce 50% of the probability of error imposes a 50% area overhead and leads to 100% MTBF improvement.

The advantage of hardening all cells in the circuit is the homogeneous treatment involved in the process. The determination of the vulnerable cells has a computing cost, generally associated with pseudo-random simulation, as does the iterative process of hardening individual cells and evaluating the new reliability. According to the results presented, this cost does not translate into a considerable advantage in terms of reliability and a better compromise would be the homogeneous hardening of the circuit cells.

4.5 Comments

This chapter has presented the characteristics of the reliability analysis methodologies proposed in the current work, as well as several types of analyses made with the concerned tools. The results show that each proposed methodology has its own advantages and limitations, and thus, must be directed to certain types of analyses and circuits.

The PTM methodology is the most restrict methodology considering the size of the analyzed circuits, but determines the exact value of signal reliability. The PTM models all signal correlations and fault patterns in a circuit and given the data structure of the matrices, a reduction in the complexity of the model is difficult to implement. This leads to a restriction of the PTM approach to the evaluation off small-sized circuits, with 10 to 16 components (wires and cells) in parallel at each level of the architecture.

Table 4.13: Results of the cell replication fault prevention approach.

Circuit type	Cells MTBF (hours)				
	10^3	10^4	10^5	10^6	10^7
rca	$5.66 \cdot 10^1$	$5.66 \cdot 10^2$	$5.66 \cdot 10^3$	$5.66 \cdot 10^4$	$5.66 \cdot 10^5$
rca rep	$9.47 \cdot 10^1$	$9.48 \cdot 10^2$	$9.48 \cdot 10^3$	$9.48 \cdot 10^4$	$9.48 \cdot 10^5$
rca 2x	$11.3 \cdot 10^1$	$11.3 \cdot 10^2$	$11.3 \cdot 10^3$	$11.3 \cdot 10^4$	$11.3 \cdot 10^5$
csa	$4.84 \cdot 10^1$	$4.84 \cdot 10^2$	$4.84 \cdot 10^3$	$4.84 \cdot 10^4$	$4.84 \cdot 10^5$
csa rep	$11.0 \cdot 10^1$	$11.0 \cdot 10^2$	$11.0 \cdot 10^3$	$11.0 \cdot 10^4$	$11.0 \cdot 10^5$
csa 2x	$9.68 \cdot 10^1$	$9.69 \cdot 10^2$	$9.69 \cdot 10^3$	$9.69 \cdot 10^4$	$9.69 \cdot 10^5$
cla	$2.67 \cdot 10^1$	$2.66 \cdot 10^2$	$2.66 \cdot 10^3$	$2.66 \cdot 10^4$	$2.66 \cdot 10^5$
cla rep	$6.46 \cdot 10^1$	$6.47 \cdot 10^2$	$6.47 \cdot 10^3$	$6.47 \cdot 10^4$	$6.47 \cdot 10^5$
cla 2x	$5.34 \cdot 10^1$	$5.34 \cdot 10^2$	$5.34 \cdot 10^3$	$5.34 \cdot 10^4$	$5.34 \cdot 10^5$
fca	$1.84 \cdot 10^1$	$1.84 \cdot 10^2$	$1.84 \cdot 10^3$	$1.84 \cdot 10^4$	$1.84 \cdot 10^5$
fca rep	$3.55 \cdot 10^1$	$3.57 \cdot 10^2$	$3.57 \cdot 10^3$	$3.57 \cdot 10^4$	$3.57 \cdot 10^5$
fca 2x	$3.68 \cdot 10^1$	$3.68 \cdot 10^2$	$3.68 \cdot 10^3$	$3.68 \cdot 10^4$	$3.68 \cdot 10^5$
fcm	$0.57 \cdot 10^1$	$0.57 \cdot 10^2$	$0.57 \cdot 10^3$	$0.57 \cdot 10^4$	$0.57 \cdot 10^5$
fcm rep	$1.67 \cdot 10^1$	$1.68 \cdot 10^2$	$1.68 \cdot 10^3$	$1.68 \cdot 10^4$	$1.68 \cdot 10^5$
fcm 2x	$1.14 \cdot 10^1$	$1.15 \cdot 10^2$	$1.15 \cdot 10^3$	$1.15 \cdot 10^4$	$1.15 \cdot 10^5$
rca tmr	$11.3 \cdot 10^1$	$11.5 \cdot 10^2$	$11.5 \cdot 10^3$	$11.5 \cdot 10^4$	$11.5 \cdot 10^5$
rca tmr rep	$43.8 \cdot 10^1$	$95.9 \cdot 10^2$	$109 \cdot 10^3$	$111 \cdot 10^4$	$111 \cdot 10^5$
rca tmr 2x	$22.6 \cdot 10^1$	$23.0 \cdot 10^2$	$23.0 \cdot 10^3$	$23.0 \cdot 10^4$	$23.0 \cdot 10^5$

Table 4.14: MTBF improvement and area overhead for cell replication.

Circuit type	Cell replication approach	
	MTBF (%)	Area (%)
rca	67.5	61.1
csa	127	47.5
cla	143	66.0
fca	94.0	48.8
fcm	195	64.3
rca tmr	722	9.93

The PBR methodology is the most useful considering the capacity to evaluate circuits of practical sizes. Despite being a fault-simulation-based approach, the relaxing conditions presented at section 3.4 help to reduce the timing complexity of this methodology, allowing its use in the evaluation of practical circuits with high accuracy. Even if the main drawback of this methodology is the time for fault simulation, functional simulation is a common practice for circuit validation, and fault simulation would be considered another iteration of the validation process. The analytical model generated by the PBR approach is adequate for repetitive analysis of the same circuit, suggesting its use in the final stages of the design process.

The SPR heuristics represent an intermediary solution between the PTM and the PBR methodologies, being capable of a highly accurate analysis of medium-sized circuits (100 cells, 16 outputs) and being capable of a fast evaluation of large circuits at the price of some lost in accuracy. The SPR approach provides the flexibility of individual values for gate reliability, and input-vector-dependent reliability values for each cell. The simple model behind the SPR methodology is flexible enough to allow the exploration of many heuristics, something that can be useful for other types of analyses, like the determination of observability and controllability metrics. The present work has targeted the SPR approach to the analysis of multiple-output circuits, and even better results can be obtained with the successive application of the SPR heuristics to single-output evaluation. The reliability analysis of a circuit based on the reliability of its individual outputs would allow a better accuracy of the results and better insights of the vulnerable elements of the circuit.

As shown in the previous section, the PBR and SPR methodologies have been applied in the evaluation of the fault-tolerant circuits described in the library presented in appendix B. This evaluation has brought new elements to the characterization of the design space of the target circuits, like the signal reliability, the time penalty and the effectiveness of fault prevention techniques. This characterization is fundamental for a better understanding of the compromises related to unreliable devices and systems and the available solutions. Next chapter presents some concluding remarks concerning the main objectives of the current work and the perspectives opened by the developed analyses.

Chapter 5

Concluding remarks

The present thesis explored the reliability attribute of combinational logic circuits, from the point of view of reliability models and analysis tools and also from the point of view of design space exploration of some fault-tolerant circuits of interest.

Considering the presented reliability models, the PBR and SPR methodologies, the results have shown that they are capable of estimating the signal reliability figures of practical circuits, allowing a direct comparison of different topologies concerning different operating properties. More than just computing the reliability value, these models allow a better understanding of the aspects involved in the reliability of a circuit, like fault masking, reconvergent fanout signals and the side-effects of fault-tolerance. To understand the contribution of these aspects helps the development of reliability-driven design automation tools and the exploration of new fault-tolerant architectures.

Considering that the PTM model is one of the best alternatives for reliability analysis of logic circuits, the results obtained with the PBR and SPR approaches can achieve a similar accuracy, with a computing time that is a fraction of the respective time for PTM execution. Comparing the PBR and SPR with other models is not simple, since many of the results presented on the related works are not directly comparable, focusing on restricted analysis conditions or other figures of merit.

It is important to consider that the PBR and SPR reliability models are based on the individual reliability of logic cells and, ideally, this information should be given by the foundries that provide the cell's library, since many of the factors that must be applied on the cell reliability model to compute its vulnerability are not present in the cell's datasheet. As mentioned before, the exact modeling of cell's reliability is impossible but with official data, the estimation provided by the models would be more reliable.

The proposed models still have some room left for improvements, like the development and implementation of other types of heuristics. This can be seen as a possible spin-off from the current work. Another natural extension of the current work is the inclusion of sequential logic in the models and tools for reliability analysis. This can be achieved by considering the reliability of the sequential elements in the circuits, associated to primary inputs and outputs of combinational blocks, and computing the cumulative effect of fault propagation through the data and control paths.

Considering the reliability improvement of logic circuits and its protection, the proposed analysis models have allowed a detailed characterization of the tradeoffs involved in the application of fault tolerance and avoidance schemes. Despite the reduced number of studied schemes, these schemes are considered representative of the universe of fault-tolerant approaches, with the other options not differing too much from the implemented

ones. The results of the analyses have shown that there exist clear compromises between fault tolerance and fault avoidance, and that the use of either solutions is not guaranteed in nanoscale systems.

First of all, the gains obtained with technology scaling are currently restricted to area of implementation, with speed gains being constrained by power consumption and thermal errors. This way, even small overheads in the propagation delay due to hardware redundancy have an important impact on the overall performance of the circuits.

The impact of the area overhead due to hardware redundancy cannot be evaluated as a general trend, and depends on the specific architecture of the target circuit and the blocks that will be protected. The protection of a reduced number of blocks in the circuit may not represent a considerable impact on the overall system area, or the overall power consumption.

The use of fault tolerance and prevention schemes for the protection of logic circuits depends on too many factors, that cannot be discussed in the present work. These factors range from circuit application and circuit architecture to the inertia of the design teams and time-to-market constraints. Considering this scenario, it's impossible to predict whether and what type of fault protection approach will be implemented in circuits of future technology nodes, and the reliability analysis presented on the current work was focused to a better characterization of the concerned design space.

From the point of view of difficulties found along the work, a relevant topic concerns the development of the analysis tools, more precisely the choice of the data structures to represent the proposed models. The translation of a physical model to an algorithmic model is not always simple and the choice of the more adequate data structure has a fundamental impact on the results obtained.

The studies developed along the thesis originated several publications, that are cited below:

- **Yield and reliability issues in nanoelectronic technologies** [2], a synthesis article published at the *annals of telecommunications* journal, representing the bibliographical review of state-of-the-art works on the related subjects;

- **Convolution blocks based on self-checking operators** [97], presented at the *14th International Conference on Mixed Design of Integrated Circuits and Systems*;

- **Efficient computation of logic circuits reliability based on Probabilistic Transfer Matrix** [98], presented at the *3rd International Conference on Design and Technology of Integrated Systems in Nanoscale Era*;

- **Reliability Analysis of Combinational Circuits Based on a Probabilistic Binomial Model** [19], presented at the *IEEE Northeast Workshop on Circuits and Systems*;

- **Reliability of logic circuits under multiple simultaneous faults** [20], presented at the *51st IEEE International Midwest Symposium on Circuits and Systems*;

- **Reliability analysis of logic circuits based on signal probability** [99] and **On the Output Events in Concurrent Error Detection Schemes** [100] to be presented at the *15th IEEE International Conference on Electronics, Circuits, and Systems*;

- **Signal probability for reliability evaluation of logic circuits** [21] and **Relevant metrics for evaluation of concurrent error detection schemes** [101] to be presented at the *19th European Symposium on Reliability of Electron Devices, Failure Physics and Analysis*, and to be published on a special edition of the *Microelectronics Reliability* journal.

The work developed along this thesis is meant to be a first attempt into the field of evolvable hardware and adaptive electronic systems. The development of architectures that could autonomously detect and correct manufacturing defects and could operate reliably with unreliable components is the ultimate objective of a higher level research project. To understand the compromises associated with reliable computing is fundamental to the development of robust systems, that could dynamically addapt to operating conditions and applications. We can imagine circuits that can temporarily change the supply voltage, change the load capacitance, change its topology to improve its reliability when necessary, and change back to a power saving mode and unreliable condition when executing no critical applications or when in idle operating mode.

The current work provided some of the answers and implemented some of the tools needed on the development of this type of evolvable hardware.

Appendix A

CMOS and beyond

A.1 CMOS evolution

CMOS is the most used technology in the design of VLSI (Very Large Scale Integration) circuits, mostly because of its power consumption properties and its planar fabrication process. CMOS properties are based on complementary field-effect transistors (FETs) operating together. In 1965, Gordon Moore [1] stated that the complexity of integrated circuits would improve by a factor of two per year, a value that Moore has reevaluated in 1975 to be a factor of two every two years, a rate that has been known as the *Moore's law*. Different formulations for Moore's law can be found in the literature and this empirical law is not related with devices density only, but with other circuit characteristics as operation frequency, for example. One popular form of Moore's law is the doubling in the density of the circuits every eighteen months. In the last decades, Moore's law has been used as a guiding measure and trend for device scaling. However, this constant evolution is reaching some limits and a "break" in the Moore's law is a matter of time. Figure A.1 shows a simplified structure of a MOSFET (metal oxide semiconductor FET) device, and the geometrical results of the traditional scaling ratio applied by industry to CMOS devices in an attempt to follow Moore's law. Looking at the figure, it is possible to compare the gain in area from one CMOS generation (technology node) to another.

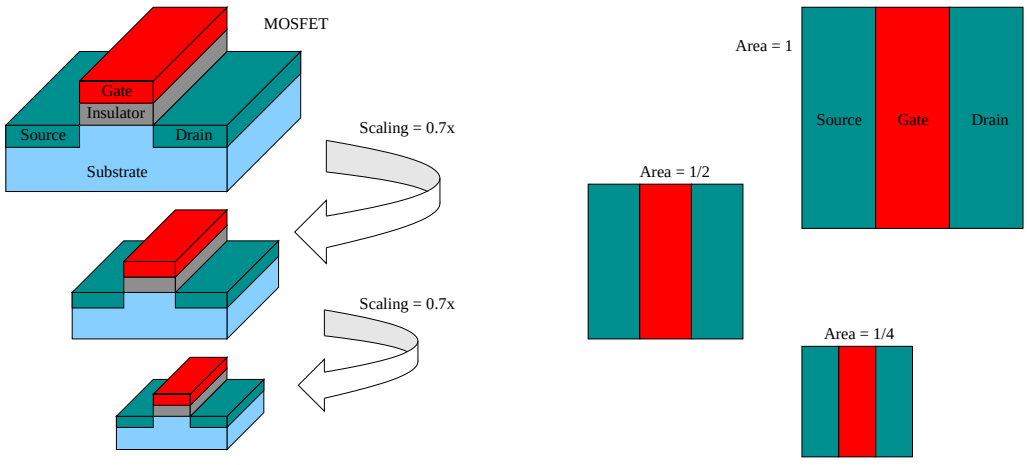


Figure A.1: Simplified MOSFET structure and scaling, with its effects in the area of integrated circuits. Actual layers proportions are not respected in the figures.

Scaling issues of CMOS devices are mainly related with supply and threshold voltages, short-channel effects, high-field effects, gate oxide leakage, dopant variation, interconnects delay and lithography [102]. There are known solutions for most of these issues but devices scaling has reached a point where some corrective actions to one issue reflect negatively in the others.

The increase in dynamic power consumption, for example, is a consequence of the increase in the switching speed of the circuits. To deal with it, one of the approaches is to reduce the supply voltage Vdd, but this reduces the signal/noise ratio of the circuit and the controllability of the transistor channel, making the device more susceptible to transient faults and uncontrolled operation. The power consumption is a serious constraint for portable or battery-supplied applications, one of the main system drivers for the industry. For high-performance applications, where consumption is not a constraint, power dissipation or heat removal is one of the main problems, with passive cooling techniques limited to dissipation of 100 W/cm^2 [32].

Another example of scaling issues is the reduction in the thickness of the gate oxide, that has followed the scaling ratio of the gate-length, resulting in an increase in the level of current leakage, and in an improvement in static power consumption. To reduce the leakage level and limit the resulting static power consumption, high-k dielectrics must be introduced in the place of the traditional *silicon oxy-nitride*.

The problems challenging the development of CMOS integrated circuits were previewed long time ago, since the semiconductors industry follows globally pre-defined trends. Nowadays, the device-scaling process is guided by the *International Technology Roadmap for Semiconductors* (ITRS), a worldwide cooperation group that determines in advance the goals and challenges for the semiconductor industry [103, 104]. The trends and challenges for the semiconductors evolution are presented in the ITRS *roadmaps*, that are a collection of tables and documents that shows a 15-year projection of semiconductors development. There are tables concerning all of the areas associated with the semiconductors industry, like design, test, process integration, lithography, yield enhancement, assembling and packaging. The roadmap tables show, along with the trends, the present level of knowledge associated with them. This reflects the fact that the achievement of the projected trends are not guaranteed, and many of them depend on solutions not known in advance. This set of unknown solutions is referred as the *Red Brick Wall* [24] because of the background color of these trends in the tables. According to the 2007 Roadmap, the *Red Brick Wall* is as close as 2008 for some areas of the CMOS manufacturing process, as lithography, test and yield enhancement. This means that a great research effort is expected from the microelectronics community, in a search for solutions to keep pace with the challenges to come. Table A.1 presents a summary of the most recent projections for the semiconductors development, extracted from the 2007 edition of the ITRS [24] roadmaps.

For many years, the scaling measure of the ICs evolution was associated with a single value called *technology node*, a term that, before the ITRS definition, indicated the smallest feature of a given manufacturing technology [103]. The ITRS defined the technology node as the "smallest feature printed in a repetitive array", and then, the half-pitch of the first-level of metal lines was chosen as the scaling measure for DRAM and MPU/ASIC circuits. For MPU/ASIC devices, the gate-length is another important metric and so, its value is indicated in the ITRS tables. Considering the rapid evolution of Flash products, the use of the technology node of DRAM circuits as the main scaling metric for the IC industry was no longer adequate. Thus, the 2005 ITRS addresses the technology pace by using different scaling measures for DRAM, MPU/ASIC and Flash products, as presented in table A.1.

Table A.1: Some trends of the ITRS 2007 reports.

Chip characteristics	Year of production					
	2007	2008	2011	2014	2017	2020
DRAM half-pitch (nm)	65	57	40	28	20	14
MPU/ASIC half-pitch (nm)	68	59	40	28	20	14
Flash half-pitch (nm)	57	51	36	25	18	13
MPU printed gate length (nm)	42	38	27	19	13	9
Maximum number wiring levels - minimum	11	12	12	13	14	14
Electrical defects						
DRAM Random Defect (faults/m^2)	2.791	3.516	3.516	3.516	3.516	3.516
Mask levels - DRAM	24	24	26	26	26	26
MPU Random Defect (faults/m^2)	1.395	1.757	1.757	1.757	1.757	1.757
Mask levels - MPU	33	35	35	37	39	39
Power supply (V)						
Vdd (high-performance)	1.1	1.0	1.0	0.9	0.7	0.7
Vdd (Low Operating Power)	0.8	0.8	0.7	0.6	0.5	0.5
Power dissipation (W)						
High-performance with heat sink	189	198	198	198	198	198
Cost-performance (cp)	104	111	119	137	151	157
Battery	3.0	3.0	3.0	3.0	3.0	3.0
Cost						
DRAM cost/bit (packaged microcents)	0.96	0.68	0.24	0.08	0.03	0.01
MPU-cp (microcents/transistor)	13.3	9.4	3.3	1.2	0.42	0.15

The representation of the metal pitch, gate-length and Flash poly pitch is shown in figure A.2.

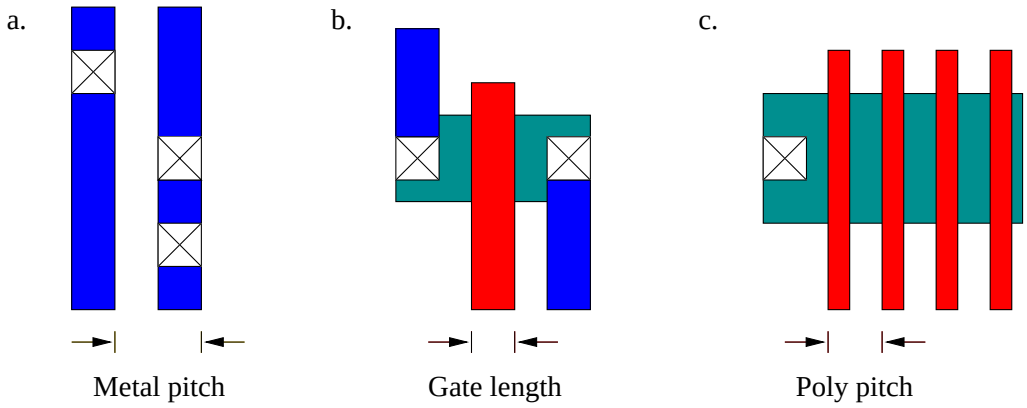


Figure A.2: Scaling metrics; a. Metal pitch; b. Gate-length; c. Flash poly pitch.

The ITRS addresses the problems faced by the industry to keep the evolution pace as the *Grand Challenges* [24]. These problems are related with the physical, technological and economical limits [105] of the scaling process. The physical limits, for example, can be understood as a change in the nanodevices behavior due to the atomic scales involved, in such a way that the laws of *classical mechanics* are no longer the only ones involved and *quantum mechanics* effects take place, like tunneling currents. This behavior cannot be changed with research effort but has to be completely understood to allow the creation of compatible models. Technological limits are associated with the quality of design, materials and equipments and there's a clear tradeoff between quality and cost in the semiconductor industry, and so, the end of the CMOS scaling era may be finally determined by economical limits.

Conventional MOSFET materials and structures are responsible for some of these issues

and new materials and new MOSFET structures have been proposed and demonstrated to guarantee the scaling of CMOS devices through the end of the roadmap. These new structures are called *nonclassical CMOS* [106, 31] or either *advanced CMOS*, and table A.2 shows a general description of these proposed devices. A detailed characterization and comparison can be seen in the work of Skotnicki et al. [31] and a brief description of each one is presented ahead.

Table A.2: Nonclassical CMOS devices.

Device Type	Concept
Transport-enhanced MOSFETs	Strained Si, Ge or SiGe channel; on bulk or SOI
Ultra-thin Body SOI MOSFETs	Fully depleted SOI with body thinner than 10 nm Ultra-thin channel and localized ultra-thin BOX
Source/Drain Engineered MOSFETs	Schottky source/drain Non-overlapped S/D extensions on bulk, SOI, or DG devices
Double-gate	Tied gates planar conduction Independently switched gates, planar conduction Vertical conduction
N-gate ($N > 2$)	Tied gates (number of channels > 2) Tied gates, side-wall conduction

One of the objectives in CMOS scaling is the increase in the switching speed, and thus, in system performance. The speed increase is a natural effect of the scaling process due to gate-length reduction, but this procedure is limited by the *short channel effects*, that can be understood as a reduction in the control of the channel by the gate. Another problem in reducing the gate length is that the supply voltage doesn't scale proportionally with the physical dimensions of the device, and thus, there's a reduction in the mobility of electrons and holes due to a *high field effect* (increase of the electric field in the channel) [102]. Transport-enhanced MOSFETs are implemented with strained semiconductors in the channel to achieve enhanced velocity and mobility of the carriers, without any change in the gate-length and device structure. Furthermore, the characteristics of strained semiconductors allow them to operate at a lower voltage when compared to bulk silicon and this property can be translated into a reduction in power consumption. Some types of transport-enhanced MOSFETs are already in production for high-performance logic [24].

Short channel effects will prevent planar bulk CMOS to scale beyond 65 nm technology and ultra-thin body (UTB) devices are scheduled to be introduced at this point. UTBs are CMOS structures with a very thin channel (or body), in the order of less than 10 nm. Such characteristic allows improvements in the electrostatic control of the channel by the gate in *off* state [31] and in threshold voltage setting, along with many other benefits. The fully depleted UTB have no floating body, like the partially depleted Silicon-on-Insulator (SOI) transistors in production [106]. The buried oxide (BOX) variation is a planar bulk UTB version where a thin buried dielectric layer isolates the channel from the bulk, and so, this device combines the features of bulk and SOI CMOS transistors. Figure A.3 shows examples of transport-enhanced MOSFETs and UTB devices.

Source/Drain engineered devices are intended to maintain the source and drain resistance to be a determined fraction of the channel resistance and to enhance the mobility of the carriers over the channel. Apart of the use of metallic electrodes (silicide) or non-overlapped gate implementation, the rest of the structure will follow the characteristics of transport-enhanced and/or UTB transistors, and so, source/drain engineered devices will probably be categorized as one of these previous devices.

The last proposed approach to CMOS evolution is based on the use of multiple gate MOSFETs, devices designed to achieve improved electrostatic control over the channel. With two or more gates to modulate the channel, short channel effects are greatly reduced and threshold voltage control is improved. Tied gates transistors have the advantage of

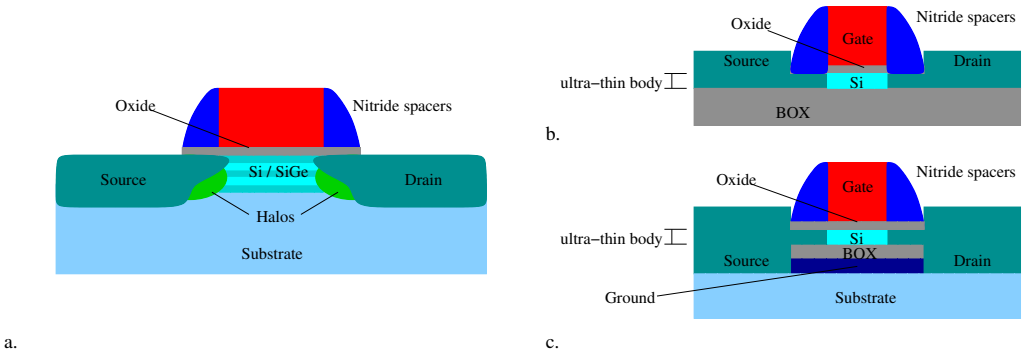


Figure A.3: Nonclassical CMOS; a. Transport-enhanced MOSFET; b. Fully depleted UTB SOI; c. UTB SOI-like ultra-thin BOX.

manufacturing and/or design compatibility with planar bulk CMOS, but the challenges are the need of sub-lithographic steps (side-wall devices) or gate doping (planar devices). Independently switched double-gate devices have a second gate in the opposite side of the channel that is capable to adjust the threshold voltage, and its operation is adequate for low-power applications. In the vertical transistor, current flows through a vertical channel controlled by two connected side gates. The advantage is that gate and channel lengths are defined by a deposition step in the manufacturing procedure, instead of a lithographic one. The challenge in this case is that the design process for vertical transistors is not compatible with the process for bulk transistors [31]. In the ITRS roadmap, double-gate devices are scheduled to enter production by the year 2011. Figure A.4 shows examples of multiple gate MOSFETs.

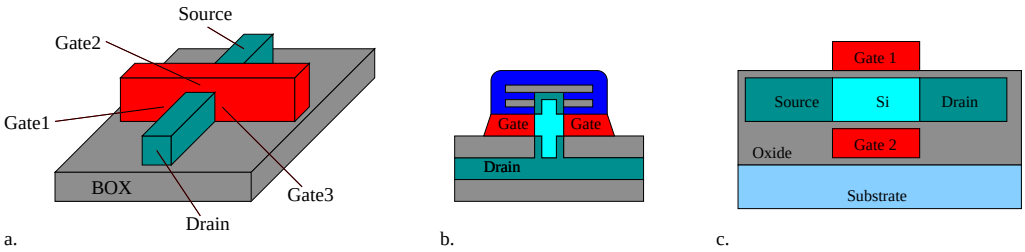


Figure A.4: Multiple gate MOSFETs; a. Tri-gate; b. Vertical transistor; c. Planar double-gate.

Many other aspects must be considered concerning these new devices like their suitability to high performance, low operation power, low standby power and scaling possibilities. As the grand challenges section in the ITRS roadmap shows, the production of nonclassical CMOS devices will depend on solutions in almost all areas of semiconductors research, like lithography, design automation, test, process integration, materials, modeling, simulation and so on. At some point, CMOS evolution will reach its end, and other devices and architectures must be introduced. The next section of this appendix presents some of the candidates that have been proposed as an alternative to extend integrated circuits evolution.

A.2 Emerging logic devices

Considering the predicted end of CMOS scaling, new devices, architectures and computing models are being studied to guarantee the evolution of electronics after the CMOS era. This appendix presents a brief introduction on topics related to these emerging logic devices, their characteristics and their limitations. Although they are being researched as CMOS alternatives, they are not direct substitutes to CMOS devices and will probably be targeted to specific applications and will be used as a complement to CMOS logic extending CMOS capabilities [106]. With this in mind, it's clear that one important property for these new devices must be the compatibility of their manufacturing processes with the CMOS process. A list of the emerging logic devices described ahead follows:

- Carbon Nanotubes and Nanowires;
- Resonant Tunneling Devices;
- Single Electron Transistors;
- Rapid Single Flux Quantum Logic;
- Quantum Cellular Automata;
- Molecular Devices;
- Ferromagnetic Devices;
- Spin transistors.

Carbon nanotubes (CNT) are cylindrical structures made of rolled up graphite sheets (*graphene*) [32, 107]. Depending on the orientation (chirality) of the graphene forming the tube, the structure may have semiconductor or metallic properties. CNTs have some important qualities like high electrical and thermal conductivity, high tolerance to chemical attack and electromigration, and can sustain much higher currents than metals [107]. CNT dimensions may vary from 1 to 20 nm in diameter, and from 100 nm to micrometers in length. CNTs have been studied in FET structures (CNT-FET) where the silicon channel of the transistor is replaced by a CNT. Figure A.5 shows a CNT-FET. The main challenges associated with CNTs are the non-deterministic chirality, placement and size of the fabricated tubes, and the high value of contact resistance [32] that limits the maximum current through the device. Recent research results show improvements in the precision of placement and in the control of chirality of CNTs [43].

Nanowires can be used in individual transistor structures or in array (crossbar) structures [32]. When used as the channel element connecting source and drain, the characteristics of nanowires are better than those of bulk silicon in terms of switching speed. When used in array structures, the resistance of the crossing points of nanowires can be configured, and architectures like programmable logic arrays (PLAs) can be implemented. Such array structures are conceptually simple, can achieve high density and are suitable to defect-tolerant designs [108, 28, 27]. Due to their electron waveguide behavior, nanowires have inspired propositions to wave interference devices like the Y-branch switch, the quantum interference transistor, the directional coupler and the stub transistor [109]. Figure A.5 shows transistor structures based on nanowires.

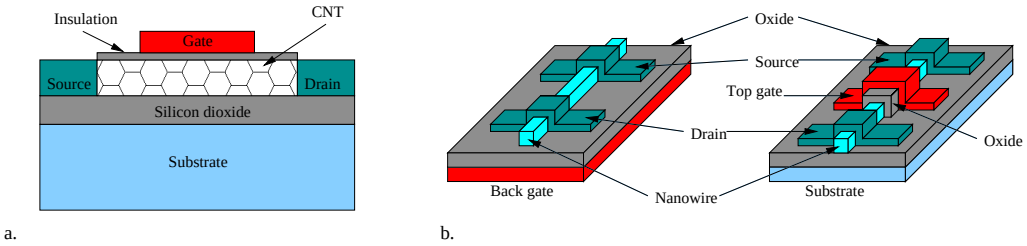


Figure A.5: 1D structures; a. CNT-FET; b. Nanowire devices.

CNTs and nanowires are classified as 1D structures [43] and are the most promising alternatives to be integrated with CMOS technology, but some problems remain unsolved. The first challenge is the comprehension and modeling of the mechanism behind the transport in quantum-confined structures. Another challenge is the improvement in the control of manufacturing and placement of this nanostructures.

Resonant tunneling devices are based on a double barrier structure separating source and drain, and at some voltage V, electrons tunnel from source to drain [105]. Resonant tunneling diodes (RTDs) are two-terminal devices that present a region with negative resistance behavior and have a high switching speed. Resonant tunneling transistors (RTTs) are RTDs that incorporate a third terminal acting as a gate. Resonant tunneling devices can be used in conventional logic architectures and in cellular nonlinear networks (CNNs) [43]. One of the main challenges is the complexity of the structure, requiring very good control of the manufacturing steps. Another problem is the on/off ratio of 10, that is far from the 10^5 ratio demanded by CMOS circuits [109]. Compatibility with the fabrication process of silicon structures has been demonstrated.

Single Electron Transistors (SETs) are three terminal devices based on the *Coulomb blockade* effect, where the number of electrons on an *island* (quantum dot) is controlled by a gate. The transfer of electrons from the source to the island, and from the island to the drain, flows one by one. Due to the low energy level involved in the operation of SETs, the frequency and temperature of operation are extremely limited [110]. Another problem with SET devices is the limited fan-out, considering that only one electron is transferred at a time. Because of these problems, the use of SETs in conventional logic circuits is not guaranteed and other architectures and computing models must be considered to explore the SETs characteristics, like CNNs. Figure A.6 shows an example of a SET structure.

Rapid Single Flux Quantum (RSFQ) is a superconduction quantum effect logic, where a flux quanta or *fluxon* is used as the basic data unit, like the *bit* unit [105]. Like SETs, one of the main drawbacks is the need for low temperature to correct operation. Another challenge is the scalability where the magnetic penetration depth limits the minimum dimensions. RSFQ technology was dropped from the tables of the 2005 ITRS roadmap because it's claimed to be already in production, and also because its applications are not the same of those targeted to CMOS devices.

Quantum Cellular Automata (QCA) is a device concept and an architecture concept, and represents a new approach for information processing. QCA devices can be divided in three different categories: molecular QCA, magnetic QCA and electrostatic QCA. The QCA basic block (electrostatic QCA) is a cell containing quantum dots that can be aligned in different ways representing binary information. Data is electrostatically propagated along the cells, that can be arranged in two-dimensional arrays to perform logic operations, or functions defined in the cellular automata theory (and cellular neural networks)

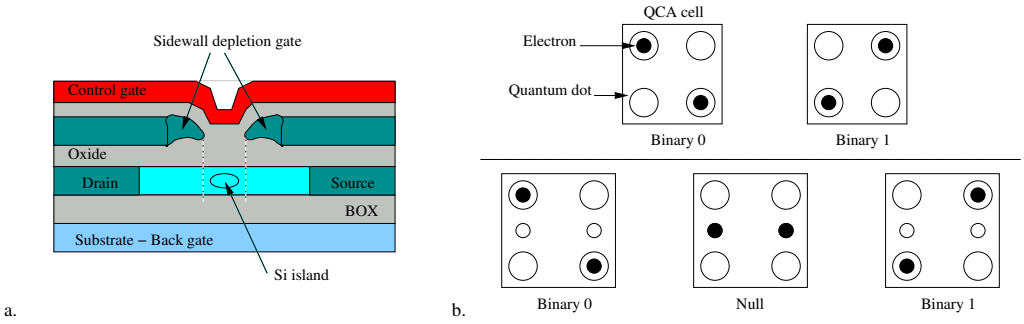


Figure A.6: SET and QCA; a. Gate-induced Si island SET; b. QCA cells with four and six quantum dots.

[32]. The limitations of the QCA structures are the need for very low temperature and the synchronization complexity (adiabatic clocking field). In the 2005 ITRS roadmap the electrostatic QCA devices where dropped from the tables because they are slow, need low temperatures and their applications are different from the ones of interest in the semiconductor industry. Magnetic QCA is treated as another category called ferromagnetic logic and molecular QCA appears in the molecular devices category. The properties researched in ferromagnetic logic are the non-volatility and reconfigurability [43]. Figure A.6 shows the representation of QCA cells with four and six quantum dots.

Molecular electronics concerns the devices where the switching or storage capacity is based on the operation of single molecules as basic building blocks. Organic and inorganic molecular circuits are being researched to produce two and three-terminal devices and the necessary interconnects. Considering the dimensions involved, molecular devices promise very high densities, increased switching speeds and reduced energy consumption [105]. Organic molecular electronics are suitable for chemical manufacturing (self-assembling), solving the increasing problem of nanoscale lithography. Among the demonstrated functions are two-terminal (resistive switches or diode-like) and three-terminal (transistor-like) structures, molecular memory and nano-electromechanical systems (NEMS) [106]. Along with conventional logic architectures, molecular electronics are suitable for integrating crossbar structures. Although some progress has been made in the research of molecular devices, many challenges related with the molecular operation and molecular manufacturing remain unsolved at this point, and it is still unclear whether and when this technology will be a viable replacement/complement for CMOS technology.

The exploration of non-volatility and radiation tolerance properties of ferromagnetic materials like **Fe**, **Ni** and **Co** is the base for the research of ferromagnetic logic devices [43]. Researched devices are moving domain wall (MDW) and magnetic QCA (M:QCA). Slow speed of operation is the main drawback for magnetic logic devices evolution and the candidate architectures are reconfigurable circuits and CNNs.

Formerly known as *spintronic* devices, the spin logic category in the 2005 roadmap has evolved to many research devices like the spin MOSFET, spin-torque transistor, spin-gain transistor, magneto resistive element (MRE) and hybrid Hall effect (HHE). One of the main objectives in the study of spin devices is to overcome the thermodynamic limit [32], that is present for all charge-based transport devices. Despite that, the devices studied so far are still dependent on charge transport and this problem must be solved to allow spin devices evolve beyond CMOS.

A direct comparison between all of these emerging research devices and CMOS shows that none of them have the characteristics needed to guarantee advantages over advanced CMOS structures [43, 109, 111], what can be seen in section A.4 of this appendix. The research results concerning these emerging devices is showing that their properties may be useful in specific applications and as a complement to CMOS devices, instead of a replacement. Despite this fact, the research of these emerging devices remains important since involves new materials, architectures and processing models that can be used in the CMOS technology and can inspire other devices.

To fully exploit the properties of these emerging devices, new architectures and computing models have been proposed by the research community. An overview of these proposals is presented in the next section of this appendix.

A.3 Defect-tolerant approaches

The research of new and emerging architectures follows the same objectives that drive the devices research. New computing models have been proposed to take advantage of the characteristics of emerging devices or to explore the use of CMOS devices in specific applications. The terms architecture or *nanoarchitecture* used here refer to how the devices and derived components are interconnected to each other to execute their processing function. The architectures cited in the 2005 roadmap are the Quantum Cellular Automata (QCA), the Cellular Nonlinear (or Neural) Networks (CNNs), the reconfigurable computers, the biologically inspired and the quantum computing. References for related research in these subjects can be seen in [43]. The work of Stan et al. [27] presents a good scenario of the challenges related with the electronic evolution, discussing all aspects from devices to architectures. Starting from the reasons for bottom-up assembly paradigms, the work focus on the alternatives to CMOS evolution, the natural choice for crossbar arrays and mesh structures, the problems associated with them, the need for defect tolerance and the integration of CMOS and nano(structures), called nano on CMOS (NoC), to guarantee scaling.

As referenced in almost all of the works in the area, one of the characteristics that must be taken into account when evaluating an architecture is its capacity to tolerate permanent or transient errors, that will be present in higher degrees in future technologies [43].

Considering advanced integrated devices, CMOS or not, the main objective of the semiconductor industry is to guarantee the scaling process and integrate an increasing number of devices in a single chip. Manufacturing processes, materials and equipments have limited precision and tolerance, that are dependent of technological and economical factors. As circuits dimension decrease to nanometers, the number of systematic, design-induced and parametric defects tends to increase, and the production of perfect circuits will become economically inviable [37, 105]. This way, in the near future, the ability to increase yield will be related with the ability to design architectures that can achieve correct operation in the presence of manufacturing defects. The ability to sustain correct operation, despite defective components, is called defect tolerance, and is related with the presence of permanent errors in the fabricated hardware. In contrast, fault tolerance is a property that's related with the ability of a circuit to recover from a intermittent or transient fault. Defect tolerance is a design approach intended to improve *yield* and fault tolerance is a design approach intended to improve *reliability*. Another approach, called error tolerance [28], is related with the ability of a circuit to produce acceptable results, instead of correct results, within certain limits, in the presence of manufacturing defects.

Error tolerance can be thought as a relaxation of the defect tolerance approach and this way, is also targeted to improve yield.

One traditional way to circumvent the problem of defective components is to introduce hardware redundancy in the form of spare parts, at some level(s) of the architecture, and to use reconfiguration to implement the necessary changes in the hardware. To deal with transient errors, modular redundancy, NAND multiplexing or data coding can be used as fault-tolerant techniques, to allow recovering from errors during system operation. From a simplified point of view, the level of defect and fault tolerance that can be achieved by a system is a function of the allowed redundancy, given that reliability and yield are traded by area, speed, power and cost overheads in the system.

Defect-tolerant techniques consider that defects may and will occur at any (and many) parts of the circuit, and based on the probability of multiple defects randomly distributed in the circuit, defect-tolerant schemes have better results introducing redundancy at the lower levels of the architecture (fine-grained redundancy), like switches, logic blocks, memory blocks and interconnections [32, 106, 37, 28]. The basic idea behind many proposed defect-tolerant nanoscale architectures is the manufacturing of a high density array of simple components and interconnections, with (re)configurable capabilities, and the post-fabrication connection of non-defective components, to synthesize the desired functions. Such design process is known as bottom-up assembly. Although the manufacturing of a high number of simple components represents no challenge, that's not the only step in producing operating structures. After fabrication, these arrays of defect-prone simple components must be tested, mapped and configured to implement the desired architecture. One problem with this approach is that testing and mapping tasks must be done separately, for each integrated circuit, and the complexity of this operation, thus cost and time, is an important constraint in high density nanoscale architectures. Recent researches show that area and power overheads, to overcome manufacturing defects with re-programmability, can be maintained at modest levels, and that crossbar type or crossbar-like are the preferred topologies [43].

Many of the defect-tolerant architectures researched today were inspired or influenced by the Teramac experiment, one of the first reprogrammable computer architecture implemented as a bottom-up assembly of basic components, presented in the work of Heath et al. [112]. Designed to be a custom configurable computer (CCC), the Teramac was built with field-programmable gate array (FPGA) chips that were responsible for logic operations and redundant interconnections in the form of crossbars or *fat tree* networks. The design hierarchy of the Teramac computer is shown in figure A.7. From the beginning of the project, the Teramac was designed to be a defect-tolerant architecture, and its implementation was based on unreliable components. The Teramac used 864 FPGAs in its structure, and 647 of them had some kind of defect. A total of 3% of all resources in the architecture were defective, but the associated problems were circumvented by the extremely high degree of interconnections implemented. Test procedures and defect mapping were driven by an independent workstation, but after determining a small reliable portion of the structure, the Teramac could be programmed to test itself. The Teramac computer showed that was possible to build defect-tolerant nanoarchitectures using only wires, switches (the crossbar) and memory (the look-up tables in the FPGAs). Its architecture allowed highly parallelized computing, and the Teramac could achieve high performance with a low operation frequency.

The work of Goldstein and Budiu [113] has introduced the concept of the *nanofabric*, an array of dense regular structures with reconfigurable capabilities. The idea presented in the

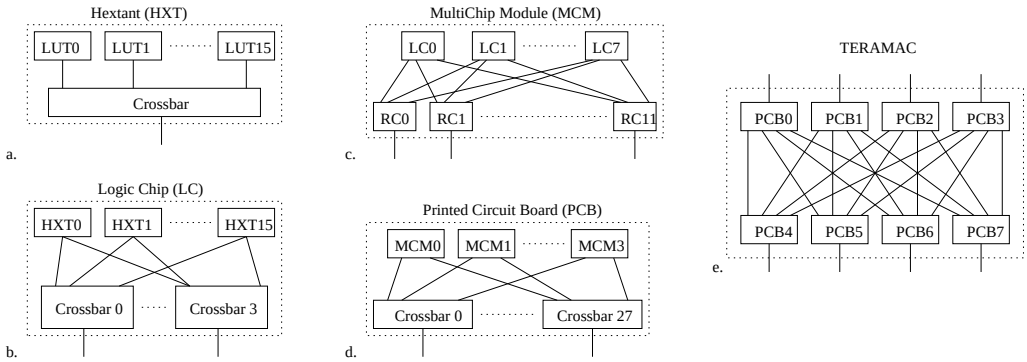


Figure A.7: Teramac design hierarchy; a. 16 LUTs in a Hextant; b. 16 Hextants in a Logic Chip; c. 8 LCs and 12 Routing Chips (RCs) in a MultiChip Module; d. 4 MCMs in a Printed Circuit Board; e. Teramac composed of 8 PCBs.

work is that regularity of layout is an important feature to future nanoscale circuits, where chemically assembled electronic nanotechnology (CAEN) can be explored to produce high-density programmable fabrics. Considering the number of devices, and the variability of the manufacturing process, the defect density in the nanofabrics will be much higher than that of classical CMOS. Thus, the reconfigurable capacity of the nanofabric is necessary to allow the circumvention of defective components, and the instantiation of the desired system. Following the general principle of bottom-up assembly, the design methodology is divided in test, defect mapping and architecture implementation using the non-defective resources. The proposed nanofabric can be configured to allow self-test and defect mapping. The structure of the proposed nanofabric is shown in figure A.8, being similar to that of island-style FPGAs. Another work of Goldstein with Mishra [114] focuses on the algorithm to test and map the defects in a generic nanofabric, that is considered to be a rectangular array of *components* with unlimited interconnection resources. The *components* granularity and type are not specified and they may be considered as logic gates, look-up tables or another configurable circuit.

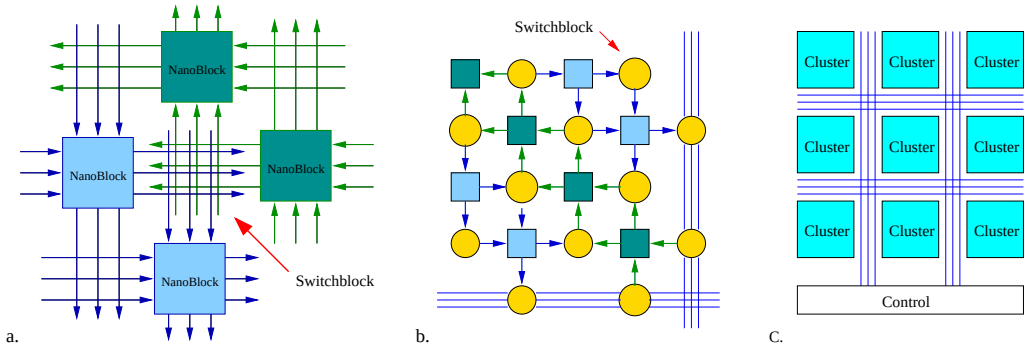


Figure A.8: Nanofabric organization; a. NanoBlocks and Switchblock; b. Cluster detailed; c. Nanofabric island-style layout.

A detailed work about challenges associated with diode-based crossbars is presented by Ziegler et al. [115]. The study discusses the use of crossbar structures as the base for bottom-up design assembly, considering the use of crossed nanowires with bistable junctions (*rotaxanes*), and the challenges associated with this kind of approach. Based on a

junction model, the work presents quantitative results of simulations from entire crossbar circuits and shows some tradeoffs related with its design. Combining the nanowires crossbar (sometimes referenced as simply *nano*) with CMOS structures to make useful circuits, the study explores the complementary characteristics of both technologies and shows the design of an adder using this approach. The impact of some problems in the overall circuit are detailed, like the resistivity of the nanowires that limit the size of the crossbar array, the degraded signal that is present in the output of the array since no gain is available in diode-based junctions, and the inability to generate inverted functions. Another contributions are the comparison of logic implementation in the crossbar using PLAs and LUTs design style, and the use of multiple arrays in the instantiation of more complex systems. The work focuses on the challenges and tradeoffs in crossbar implementation and fault and defect-tolerant approaches are not directly discussed. These considerations are done by Hogg and Snider [116], that also study the implementation of adders in diode-based crossbars. The work details the relation between defect levels, size of the crossbar, size of the circuit, mapping time and yield achievable, discussing many tradeoffs and limitations of the crossbar approach. The generic structure of diode-based crossbars is shown in figure A.9.

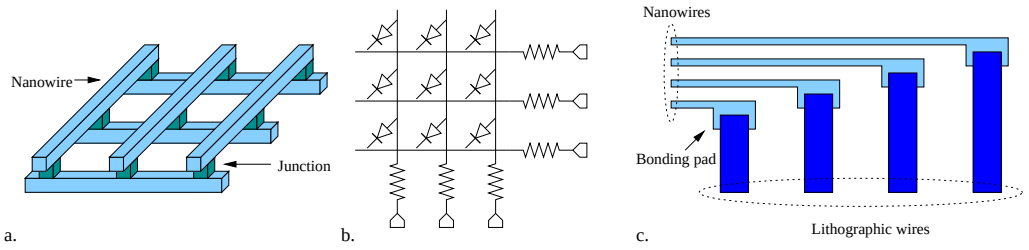


Figure A.9: Diode-based Crossbars; a. Generic structure; b. Simplified model; c. Possible interface CMOS/nano.

The work of Lee et al. [117] also considers diode-based crossbars of nanowires as the nanofabric to implement memory architectures. The work focuses on the mapping of storage elements and decoding logic circuit in the presence of defective interconnections and crossing points. The study shows the relation between the defect rate and successful instantiation probability. Defect tolerance is achieved with spare rows and columns, and the level of redundancy necessary to successfully implement the desired memory is determined, according to the defect rate. The work shows that a high amount of redundancy is necessary to successfully instantiate the desired circuit in the presence of defect rates of 5% in the nanofabric.

The work of DeHon and Naeimi [108] proposes a reconfigurable diode-based crossbar nanofabric and some specific strategies to deal with test, defect mapping and circuit instantiation. The proposed crossbar is modeled with nanowires of 3 to 10 nm in diameter with configurable cross-points, and regeneration and inversion structures. Combining such crossbars with CMOS infrastructure makes possible to implement a programmable block that is called nanoPLA. The overall structure is similar to an island-style FPGA where the grid of look-up tables is replaced by a grid of nanoPLAs. Defect tolerance is achieved by spare nanowires in a nanoPLA and spare nanoPLAs in a high level. Test and map are thought to be executed by an integrated microprocessor, designed with reliable CMOS. Considering the proposed configuration strategy, where logic functions are matched with the defect pattern, the area overhead due to spare nanowires is minimized. The relation

between defect rate and achievable yield is presented for the target nanofabric, and the mapping results of many benchmark circuits is shown. An interesting comparison is that even with the overhead predicted to cover 10% defect rates, the density of the circuit is still 100 times higher than a 22 nm equivalent CMOS FPGA.

A study that considers reconfiguration at a coarser grain level is presented by Chen et al. [51]. The work focuses on a nanofabric organized in small regions, to which simple function flows have to be instantiated. One region in the proposed nanofabric represents a 4×4 array, with eight memory-based processing elements (PEs) connected by eight switching elements (SEs). PEs can perform 8-bit arithmetic and logic operations. The next level in the design hierarchy is a mapping unit (MU) that implements the interconnection of different regions. The third and last level of the design is an abstract component that is composed of many mapping units and is responsible to implement the application kernel. The main idea is not to fully test and map all the resources in the nanofabric, but simply to test the resources necessary in a region considering the desired function flows that may be implemented. Test is achieved by configuring different elements in a region to operate as a triple-modular redundant test circuit. The mapping step is optimized by the use of pre-computed configuration possibilities. This overall approach tries to minimize the time-consuming task of testing and mapping all of the nanofabric resources. Figure A.10 shows the organization of the proposed nanofabric and an example of processing flow that could be mapped to it.

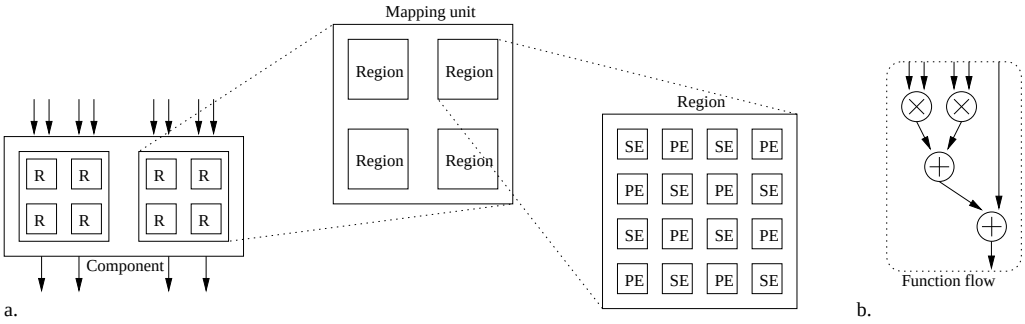


Figure A.10: Hierarchic nanofabric; a. Hierarchy of PEs, SEs, mapping units and components; b. Example of function flow to be instantiated in a region.

Another approach to bottom-up design based on self-assembly fabrics can be seen in the work of Tour et al. [118], where the basic building block is called a *Nanocell*. Composed by I/O leads lithographically distributed and reprogrammable self-assembled molecules (disordered topology), a nanocell can be programmed to execute logic operations. The work presents a training procedure based on genetic algorithms (GAs). Husband et al. [119] show the implementation of adders and registers based on the interconnection of nanocells. Figure A.11 shows the nanocell structure and its use as a logic building block.

Likharev [120] and Türel et al. [121] propose the implementation of neuromorphic architectures based on hybrid CMOS/nanowires/molecular (CMOL) circuits, to construct a nanofabric with high density, high connectivity and low power consumption. In CMOL circuits, configuration is achieved by a training process, and the researchers propose a class of distributed crosspoint networks (CrossNets) that can implement pattern recognition in Hopfield mode. The work explores the use of three-terminal and two-terminal components in the implementation of the proposed nanofabric. CrossNets can be seen as a kind of neural network, and this way, fault and defect tolerance are inherent characteristics of

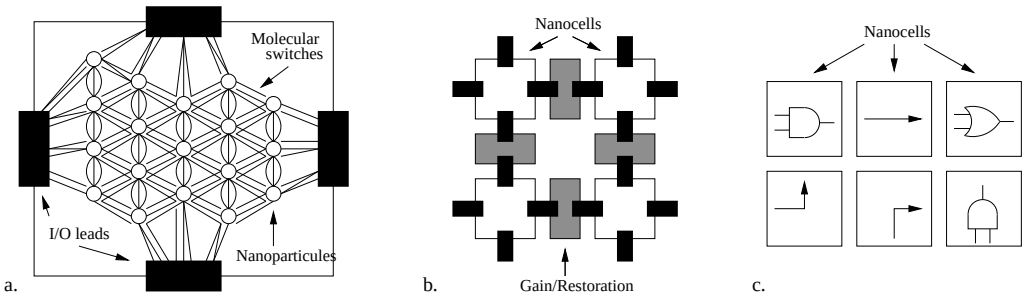


Figure A.11: Nanocell approach; a. Nanocell structures; b. Interconnections; c. Circuit implementation.

these systems.

In a design for yield defect-tolerant approach, Sirisantana et al. [37] present a design process that is focused to determine whose are the vulnerable transistors in a logic circuit. Defect-tolerant design is achieved by introducing a transistor in parallel with the vulnerable ones to enhance the yield of the manufacturing process. The study compares different logic styles like selective clock skewed-logic (SCSL), domino and static CMOS, and shows that SCSL circuits are less vulnerable than the others. The work also shows a considerable yield improvement achieved with the proposed parallel transistor technique, and presents the area and power overheads considering a set of benchmark circuits.

Another top-down approach is presented by Mitra et al. [122], where defect and fault tolerance are based on reconfigurable architectures in a coarser-grain. The target system uses two FPGAs, that are capable of autonomous detection, diagnosing and correction of errors during operation. This structure can be configured as a processor and a coprocessor running in different FPGAs, as concurrent circuits executing different operations, or concurrent circuits executing the same operation. The basic idea is that one FPGA must be capable of reconfiguring the other one. Besides that, each FPGA can apply test and repair algorithms for self-recover from temporary and permanent faults.

In the work of Breuer et al. [28], error tolerance is studied as an alternative to defect and fault tolerance, considering that some applications can operate reasonably well in the presence of defects, like multimedia, DSP, and neural network applications. With this assumption, yield enhancement can be achieved by categorizing defective circuits in different levels and associating them to different allowed applications. This approach poses many challenges to the test area, that will have to deal not only with go/no go tests but with fault diagnosing, allowing the identification of defects according to some thresholds. To support error tolerance, *intelligible* testing [123] will be application-driven, instead of being universal, and some error metrics will probably be introduced, like the proposed rate, accumulation and significance ones (RAS measures). The impact of errors in the capacity, performance, throughput, dB loss, attenuation and distortion of the circuits should also be considered. Error tolerance is still a proposal and there are many aspects to consider before it becomes reality.

A.4 Perspectives

Previous sections presented a review of many topics and works concerning the expected evolution of integrated circuits, yield and reliability issues and researched solutions. Con-

sidering current knowledge, it's difficult to preview when CMOS technology will stop evolving, and whether emerging logic devices will be integrated in future chips. At this point, none of the emerging devices have projected characteristics and attributes that meet the requirements to justify the investment in a new technology. Figure A.12 shows a comparison of technology densities that can be expected with each emerging logic device circuit and figure A.13 shows the speeds that are projected to be achieved with such circuits.

The values presented in figures A.12 and A.13 reflect circuit operation and not individual devices characteristics. Many of the studies on emerging devices emphasize the characteristics of the switching device only, as an isolated component, but to implement any useful logic cell or block, a circuit will need many such devices or some additional support structure. A carbon nanotube, for example, may have a diameter of 1-nm, but when used as the channel in a FET structure there's no gain in the area of the complete device. Another example is the extremely high switching speed of RTD devices, that do not reflect in high circuit frequency since a traditional transistor is necessary to control its operation.

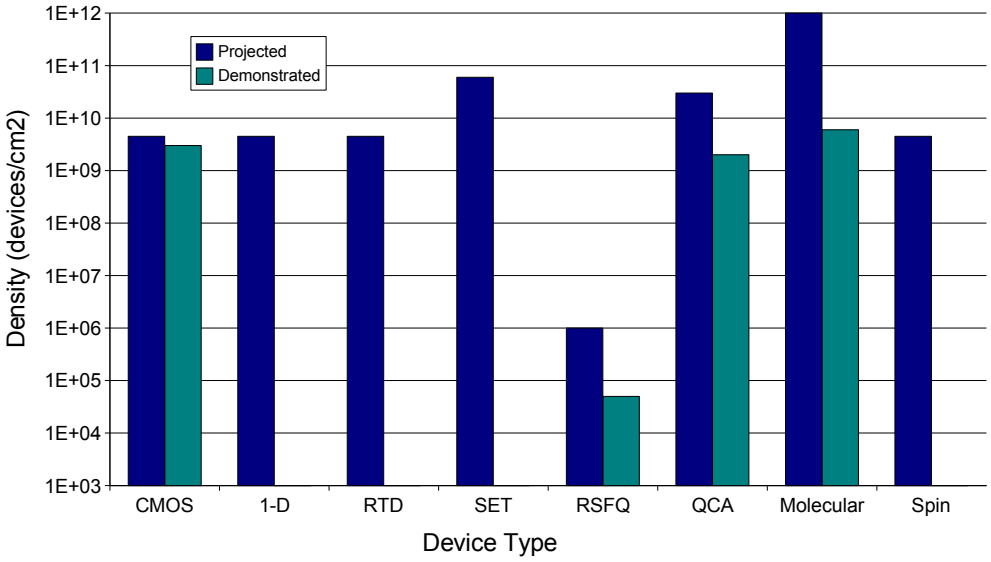


Figure A.12: Density (devices/cm^2) of CMOS and emerging logic devices [109].

Regarding only the density and speed prospects, it's possible to see that the density advantage of some technologies is not followed by speed advantages and vice versa. Additional problems make the adoption of the proposed technologies even more difficult, like the high error rate predicted in molecular and SET structures, or the extremely low temperature necessary to operation in RSFQ, QCA and SET circuits. All these issues point to future technologies based on advanced CMOS devices, with application-specific use of emerging logic devices to complement CMOS characteristics. Although advanced CMOS shows a promising future, yield and reliability issues may anticipate the end of Moore's law. The 90 nm manufacturing process, for example, implies 35 masks and 700 steps [124]

and advanced CMOS devices will represent more masks and steps, raising the challenges to maintain yield. Finally, thermal noise and heat removal limits will prevent further improvements in speed and density, not only for CMOS devices, but for any charge-based technology.

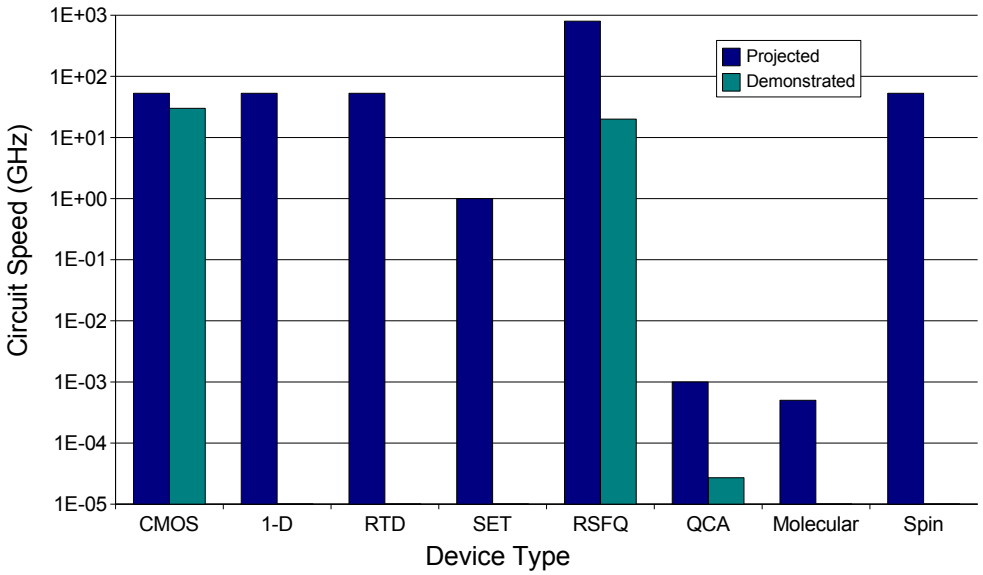


Figure A.13: Circuit speed (GHz) according to different devices [109].

The probable architecture for future nanosystems cannot be anticipated at this point because there are too many candidates, with different and desirable properties. The choices are complicated by the lack of comparative data on devices and architectures, whose concepts are relatively new. The research areas of devices and architectures face different challenges. In one side, new devices are experimentally demonstrated, and there's a need for models to explain their behavior, allowing the design of increasing complex systems. On the other side, new architectures are proposed, described and simulated, and there's a need for demonstration that their manufacturing is possible in large scale with the expected accuracy.

Researches on architectures and devices are based on assumptions that are waiting to be proved. Many published works focus only in some aspects of the design space, making direct comparisons difficult. Reconfigurable crossbar architectures, for example, are the most researched alternatives so far, due to their claimed advantages: regular self-assembled low cost fabrics, higher density, low power operation and defect-tolerant capabilities. Self-assembly methods have been already demonstrated, but they are in an early stage of development, and it is assumed that self-assembly will be capable of generating regular full-sized structures, creating components with the desired properties. It is assumed a low cost manufacturing but at this point this is speculative, because the methods are not mature. Even with a tuned manufacturing process, researches indicate that CMOS and nano structures will have to be integrated together, and manufacturing cost will at

least remain the same. Nanofabrics concept trades simpler manufacturing by an increased complexity in post-fabrication procedures, and thus, low cost manufacturing may not mean low cost chips. This will depend on the strategy implemented to test, map and configure the manufactured circuits (e.g., self-test, external test, defect mapping, etc). High density is assumed, but the number of low-level devices necessary to implement higher-level functions are dependent of some variables like defect rate, fault-tolerant method implemented and CMOS/nano ratio. Molecular circuits, for example, may achieve densities of 10^{12} devices but a redundancy of even 10^4 may be needed to guarantee acceptable reliability levels [111], resulting in less useful devices than current CMOS circuits. Low power is assumed but exact models of system behavior aren't available, and full scale circuits were not simulated to verify all the contributions on static and dynamic power consumption, in the presence of defects, for example. As reconfigurable arrays, nanofabrics are inherently defect-tolerant but the process of circumventing defects may expose other problems, like the synchronization (clocking) of circuits through paths with unknown delays.

Despite all of the problems, researches show a promising future to nanofabrics, since reconfiguration is considered the most adequate method for defect tolerance [124]. Acting as FPGAs, with much higher density, nanofabrics will be probably integrated with CMOS devices and circuits, and Moore's law may be guaranteed by a constant increase in the ratio between nano and CMOS [115] circuits. To take advantage of nanofabrics, scalable test, mapping methodologies and asynchronous circuit design must achieve maturity.

Concerns with reliability will bring fault-tolerant design techniques to mainstream circuits in future technologies. While fault-tolerant approaches have been researched and applied for a long time, most of them are not suited to the level of failures expected to occur in the nanostructures. New methods proposed to deal with multiple simultaneous faults rely on space and time redundancy, coded computation, sequential processing and memory-based logic, but a direct comparison of the methods is not possible, since the studied parameters are not the same in the related works. Benchmark circuits and fault coverage metrics should be used to categorize solutions, and allow a clear definition of the tradeoffs involved in fault-tolerant methods like area, performance and power consumption.

With CMOS arriving at its evolutionary limits, the research community is searching for alternatives and studying proposals that range from a simple extension of current technologies to completely new and exotic ones. Considering the history of the IC's industry, evolution has been guaranteed by small steps, from one generation to another, and that's how nanocircuits will probably evolve. Implementation of new devices and architectures will depend on their compatibility with current (or preceding) design, manufacture and test processes, and that's why exotic proposals will not be implemented anytime soon. In such scenario, IC's evolution will depend on clever application of known techniques, like reconfiguration and enhanced coding, to deal with the yield and reliability problems.

Appendix B

Fault-tolerant library

B.1 Introduction

This appendix presents a detailed view of the circuits and fault-tolerant approaches that have been implemented in the current arithmetic library. This detailed presentation is targeted to a clear understanding of circuit topologies, according to their respective description in the VHDL files. This should guide future modifications and extensions of the circuits in the library. The main idea behind the library is to group VHDL and Verilog descriptions of various types of arithmetic circuits and related ones, allowing their use and comparison in different works and projects. Despite the reduced number of circuits in the initial version of the library, this number is intended to evolve continuously.

The library is composed of standard adders, carry-free adders and a multiplier, and the respective fault-tolerant versions of these circuits. The idea of studying carry-free adders is to evaluate their capacity to restrict fault propagation across the circuit, since no carry propagation occurs. An extension of the current library concerning fault-tolerant finite impulse response (FIR) filters has also been implemented but is not detailed in this work.

B.2 Standard adders

B.2.1 Ripple-carry adder

The ripple-carry (RC) adder is the simplest type of adder topology. Is based on a full adder (FA), that implements the addition of 3 binary digits. A half adder (HA), that implements the addition of 2 binary digits, can be also used to add the least significant bits (LSBs). The truth-table of a FA can be seen in Fig. B.1a. Several different implementations of a full adder are possible and figure B.1b presents the implementation described in the library. For CMOS synthesis, other structures may be more adequate like a functional description. The FA cell is represented in figure B.1c, and may be considered the basic block of the ripple-carry adder, that will have as many FAs as the number of data bits (data width).

Figure B.2a shows the organization of the adder with its inputs, outputs and internal signals, following the parameterized description of the circuit in the library. Figure B.2b shows a traditional block representation for adders and another one can be seen in figure B.2c. The name *ripple-carry* is related to the need of the carry signal to propagate through all of the adder cells, making the circuit delay directly proportional to the width of the operands. Therefore, despite of its simplicity (and regularity of structure), the ripple-carry adder is the slowest type of adder, due to carry propagation through the chain length. Many

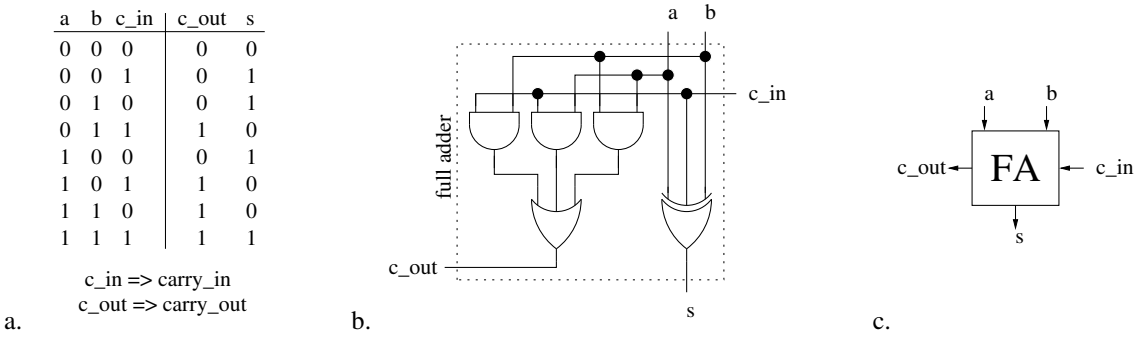


Figure B.1: Full adder (FA). a. FA truth-table; b. FA logic circuit; c. FA block.

other types of adders have been proposed in the literature to reduce and even eliminate this carry chain propagation time, at the cost of a higher hardware complexity.

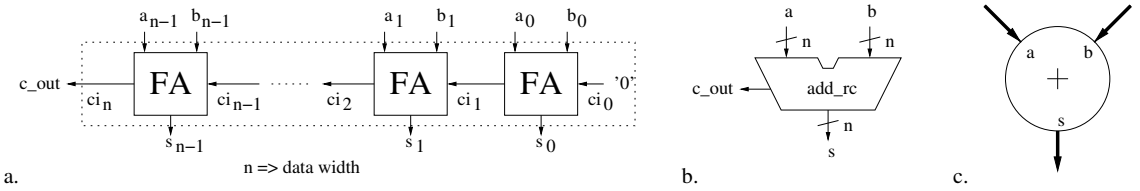


Figure B.2: Ripple-carry adder (RC). a. Architecture; b. Adder block; c. Alternative representation.

B.2.2 Carry-select adder

The carry-select (CS) adder is an architecture that explores parallelism to reduce the carry chain path. Instead of waiting for the carry signal to ripple through all of the full adders, the carry select adder concurrently calculates two result values considering the two possible values of the carry signal at a given point [125]. Considering a n-bit ripple-carry adder, the carry signal must propagate through n full adders to generate the addition result. For a carry-select adder the carry chain can be reduced by a factor f, resulting in a critical path of n/f full adders. An example of a carry-select adder that breaks the carry chain by a factor of 2 is shown in Fig. B.3. In this case, half of the adder structure is duplicated, generating two concurrent outputs, one considering the carry input signal with value 0 (c_m in the figure) and another considering the carry input signal with value 1. The correct output will be selected in a multiplexer (MUX) by the correct carry value, that must ripple only through half of the full adders.

The gain in propagation time achieved by a carry-select adder depends on the number of full adders that are connected in the same carry chain and that's a choice of the designer. A limiting factor is the delay associated with the multiplexers that are necessary in the structure, and the speed gain achieved with the reduction in the carry chain is degraded in part by the propagation delay of the multiplexers. In the present library, the carry select adder will be organized in 2-bit blocks, like the one presented in Fig. B.4. This choice is based on the structure necessary for the self-checking version of this adder, proposed in [125].

As shown in Fig. B.4, the basic block is composed of half adders (HA0 and HA1) and

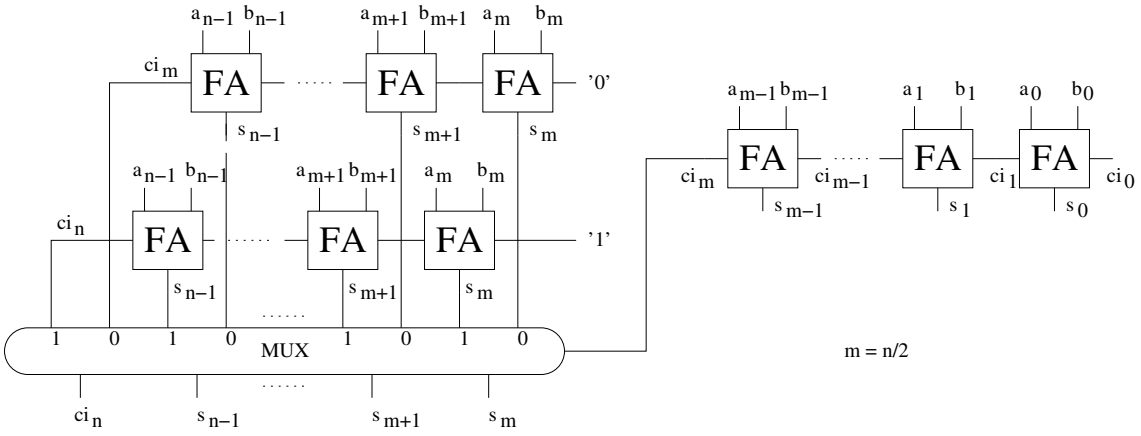


Figure B.3: Carry-select adder structure.

full adders (FA) to simplify the circuit. These 2-bit adders must be connected in a chain to implement the complete adder. The carry output of each block ($c\_out(1)$) controls the multiplexer of the subsequent block.

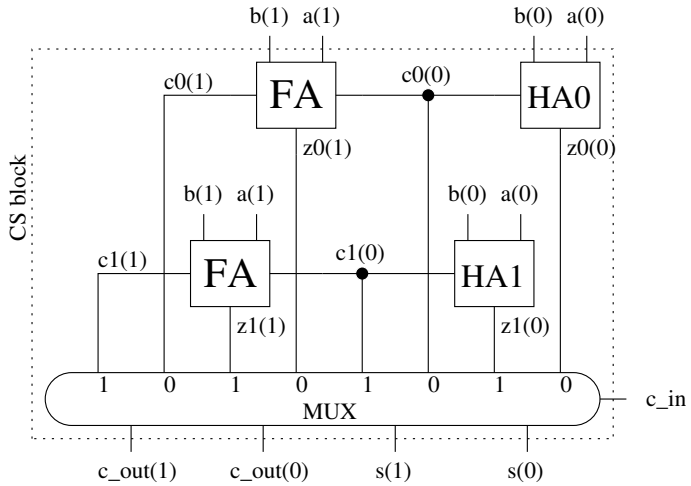


Figure B.4: 2-bit adder basic block of the CS adder in the library and its VHDL signal notation.

B.2.3 Carry-lookahead adder

The carry lookahead (CLA) adder is an optimization of the ripple-carry adder that explores some characteristics of the full adder cell. Instead of waiting the carry to propagate through all of the precedent cells, the lookahead adder determines the carry based on two-level logic. According to the values of the input bits (a_i and b_i), the carry-out signal of an adder cell may be considered as being generated by the cell or simply propagated by the cell. This interpretation of the truth-table of the full adder can be seen in the Fig. B.5a.

The signals *generate* (G_i) and *propagate* (P_i) can be obtained with AND and XOR gates, as shown in Fig. B.5b. These signals can be used to reduce the critical path of the carry signal in the adders, considering that they can drive the carry signal through

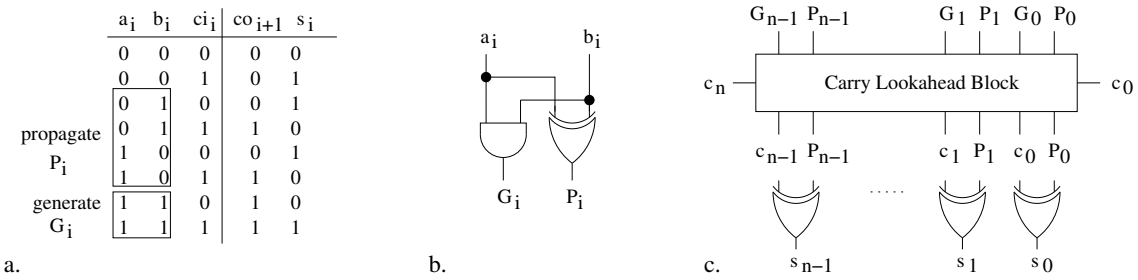


Figure B.5: Carry lookahead adder; a. Full adder truth-table; b. Generate and propagate signal generation; c. CLA block.

"shortcuts". The carry-skip adder is another circuit that takes advantage of the generate/propagate property of the addition operation. The carry lookahead adder uses the generate/propagate signals to drive a two-level logic that supplies the addition gates with fast carry signals, as can be seen in figure B.5c.

Expressions B.1 through B.5 show the possible developments of the carry generation logic. The expressions reflect the fact that the carry-out signal of a cell can be generated in that cell or propagated from a previous cell.

$$c_0 = carry - in \tag{B.1}$$

$$c_1 = G_0 + P_0.c_0 \tag{B.2}$$

$$c_2 = G_1 + P_1.c_1 \tag{B.3}$$

$$c_3 = G_2 + P_2.c_2 \tag{B.4}$$

$$c_4 = G_3 + P_3.c_3 \tag{B.5}$$

These considerations can be translated into a two-level logic as presented in expressions B.6 through B.9.

$$c_1 = G_0 + P_0.c_0 \tag{B.6}$$

$$c_2 = G_1 + P_1.G_0 + P_1.P_0.c_0 \tag{B.7}$$

$$c_3 = G_2 + P_2.G_1 + P_2.P_1.G_0 + P_2.P_1.P_0.c_0 \tag{B.8}$$

$$c_4 = G_3 + P_3.G_2 + P_3.P_2.G_1 + P_3.P_2.P_1.G_0 + P_3.P_2.P_1.P_0.c_0 \tag{B.9}$$

Considering that actual gates have a limited number of entries, the two-level logic approach has its own limitations, restricting the lookahead block to four carry bits long. This way, a chain of carry lookahead blocks is used to implement carry lookahead adders. The two-level logic of a carry lookahead block is shown in Fig. B.6.

The CLA block chain implemented in the library is shown in Fig. B.7. The critical path can be further reduced with the use of CLA blocks in a higher level to generate/propagate carry signals of the lower level CLA chain.

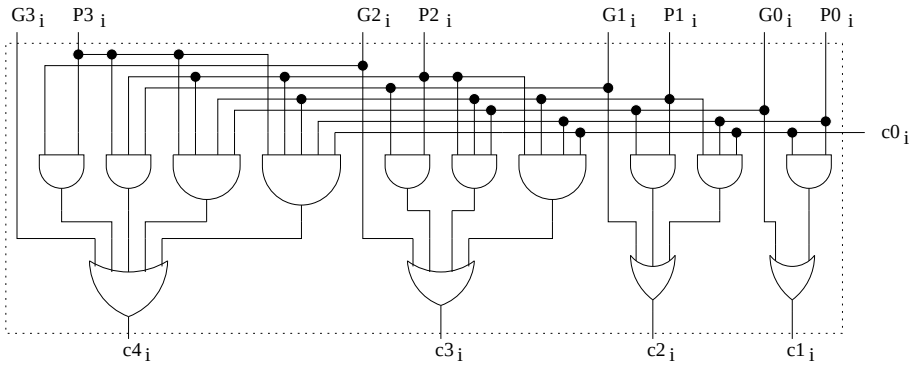


Figure B.6: 4-bit carry lookahead block.

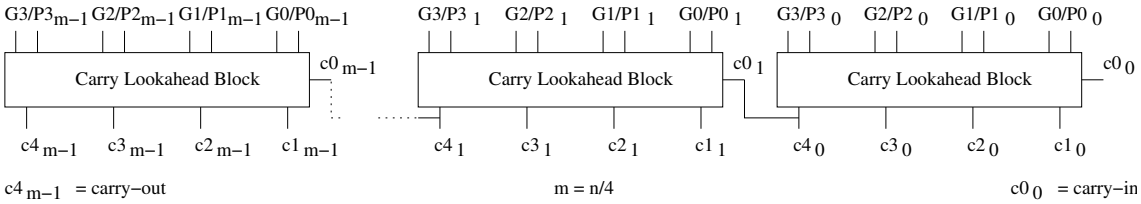


Figure B.7: CLA blocks chain.

B.3 Carry-free adders

B.3.1 Signed digit (radix-2)

Signed digit (SD) adders use the properties of redundant number representation to eliminate the need for carry signal propagation. These type of adders are also known as carry-free adders, since carry propagation is restricted to only one digit of the data, independently of the data width. Signed digit adders are the fastest ones but at the price of a great hardware overhead to process redundant numbers, and the need of conversion from/to binary representations at the beginning/ending of the operation. The implementation described in the library is based on the work of Cardarilli et al. [126]. The signed digit adder implemented in the library is a radix-2 version, where digits may assume values of -1 ($\bar{1}$), 0 and 1. At the logic level, each signed digit is coded by two bits and the chosen codification for the present adder is shown in Fig. B.8a. The value "11" is not allowed in the inputs, despite the corresponding value '0' on the figure. Such code was chosen due to a self-checking version of the adder, but other codifications are possible. Fig. B.8b shows examples of signed digit to decimal conversion.

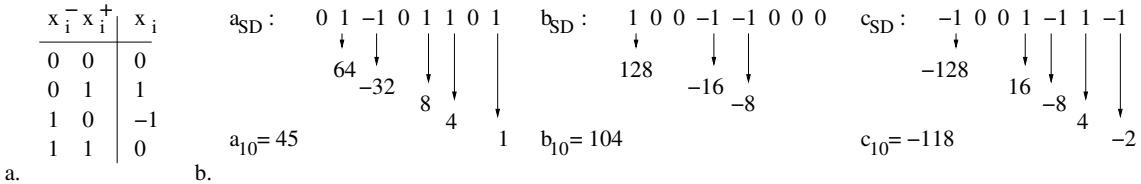


Figure B.8: SD codification; a. Implemented coding; b. SD to decimal conversion.

Considering input and output operands of the type $x \in [-(2^n - 1), (2^n - 1)]$, where x_i are signed digits, the carry-free addition is performed by the structure seen in Fig. B.9a.

The interface of the adder is shown in Fig. B.9b, where the width n corresponds to the number of signed digits to be added.

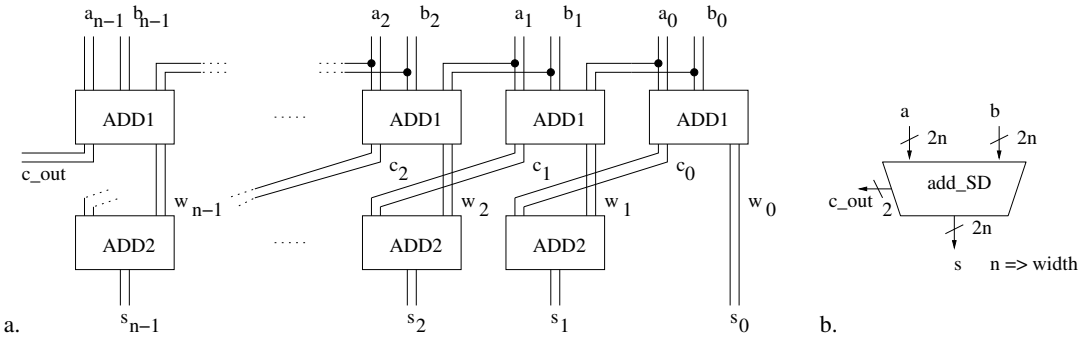


Figure B.9: SD adder; a. Internal structure; b. Circuit interface.

In the proposed architecture [126], the block ADD1 generates intermediate signals of sum and carry, according to the rules presented in Fig. B.10a. The same figure shows the interface of the block. Block ADD2 is responsable for generating the result according to the intermediate sum and carry (from the previous digit). Fig. B.10b shows the table of the function implemented by block ADD2 ($s_i = w_i + c_{i-1}$) and its interface.

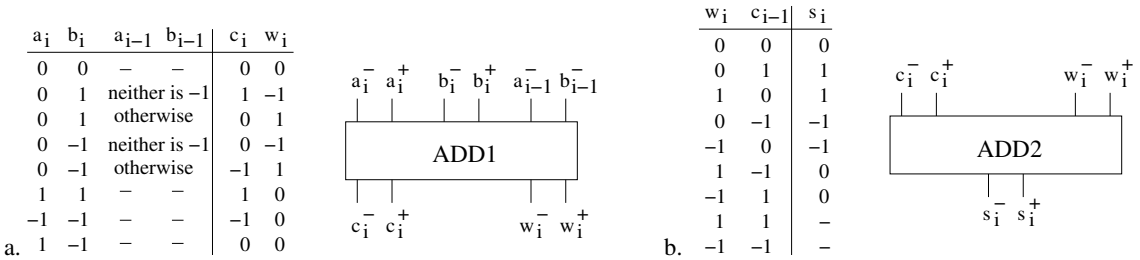


Figure B.10: SD adder; a. Rules and interface of ADD1; b. Rules and interface of ADD2.

According to the specified rules for signed digit addition, examples of the procedure to add signed digit numbers are shown in Fig. B.11.

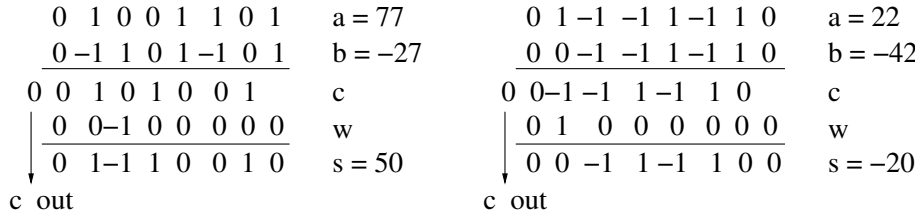


Figure B.11: Examples of signed digit addition.

The logic structure of blocks ADD1 and ADD2 will not be detailed since they are rather complex (specifically the block ADD2).

As referred before, the main drawback of carry-free adders is the conversion between redundant and binary representations. The conversion from a redundant data representation to a binary one is implemented by an ordinary adder, and so, the speed gain is lost in the conversion step. These type of adders are then targeted to circuits where redundant

data can be used for all types of functions. The conversion from binary representations to redundant ones is much more simple and allows a reduction of the size of the concerned carry-free adder, due to logic simplifications.

A version of the SD adder considering two's complement inputs is present in the library, and its interface is shown in Fig. B.12a. With the coding scheme presented in Fig. B.8a, the conversion from two's complement binary numbers to SD is achieved by simple wiring of the signals, as can be seen in Fig. B.12b, where grounded signals are for reference only.

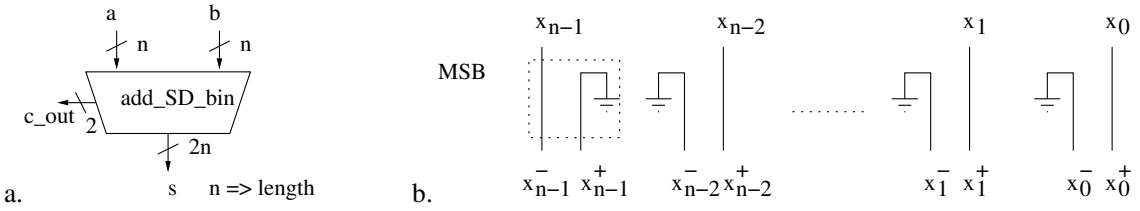


Figure B.12: SD adder with binary inputs; a. Adder interface; b. Binary to SD conversion wiring.

The reduced set of possible signed digit inputs allows a considerable simplification of the blocks responsible for the SD addition. The internal structure is shown in Fig. B.13 and the simplified circuits for blocks ADD1 and ADD2 are shown in figure B.14.

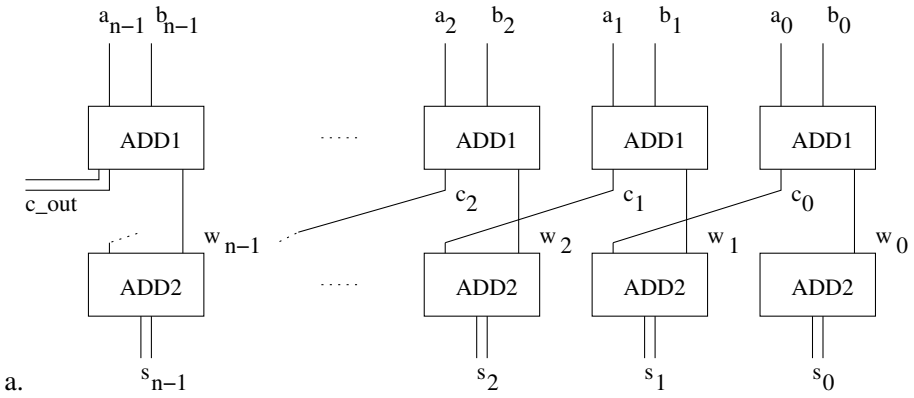


Figure B.13: Internal structure of the SD adder with binary inputs.

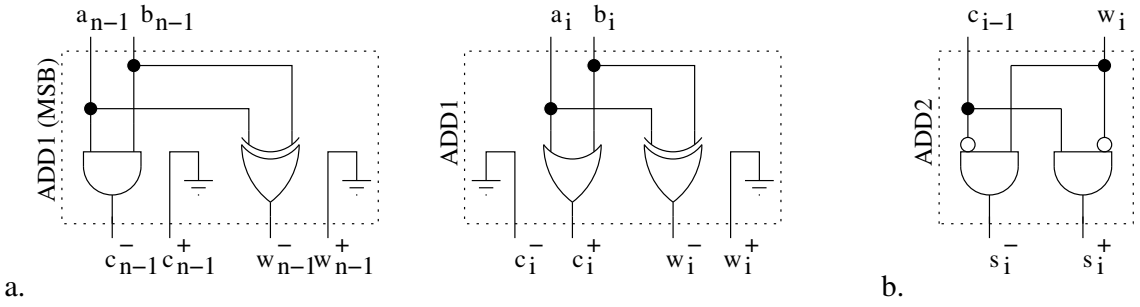


Figure B.14: SD adder with binary inputs; a. ADD1 logic circuits; b. ADD2 logic circuit.

According to the codification of the redundant data, different block structures can be designed. Another version of the radix-2 signed digit adder is available in the library,

based on the work of Lindstrom [127, 128]. The chosen codification, adequate to residue operation, leads to a more compact implementation, reducing the overhead of the signed digit adder. The structure of the basic block of this version of the adder is shown in Fig. B.15.

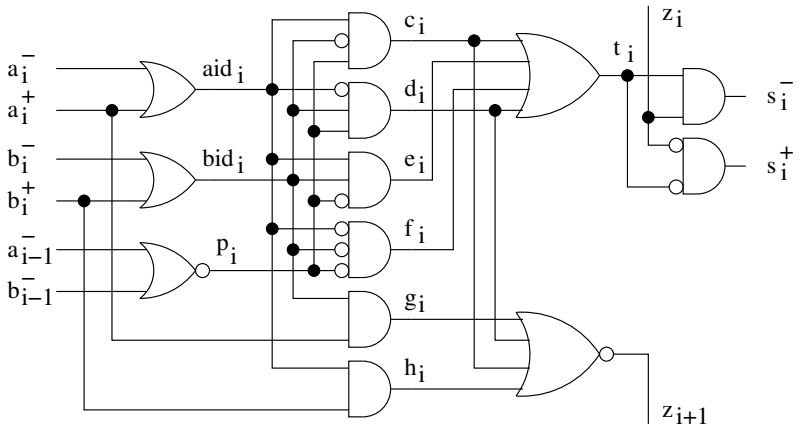


Figure B.15: Basic block logic circuit for the optimized SD adder.

Fault-tolerant versions of these last two signed digit adders were not implemented, since the self-checking properties are related with the specific redundant codification defined by Cardarilli et al. The presence of these circuits in the library is targeted for their characterization according to the traditional design objectives, and validation of some auxiliary circuits.

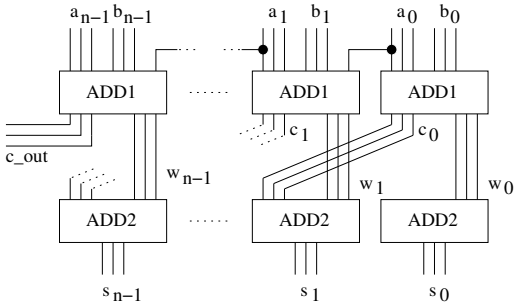
B.3.2 Signed digit (1-out-of-3)

A second type of carry-free adder is present in the library, with a 1-out-of-3 data encoding, and is based on the work of Townsend et al. [129]. The use of 1-out-of-3 data encoding is the way proposed to allow detection of faults in the operation of the circuit. Self-checking capabilities are allowed by the use of m-out-of-n checkers, that generate a two-rail code [10] indicating the presence of non codewords in the output. The implementation of this carry-free adder uses the encoding presented in Fig. B.16a, which was chosen to simplify the conversion of inputs from binary and signed magnitude representations to 1-out-of-3 ones. The general structure of the adder, shown in Fig. B.16b, is similar to the radix-2 signed digit adder, and is based on similar blocks for intermediate sum and carry and final sum. The interface of the circuit can be seen in Fig. B.16c.

a.

x_i	x_i^H	x_i^M	x_i^L
0	0	1	0
1	0	0	1
−1	1	0	0

b.



c.

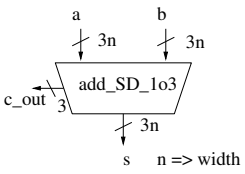


Figure B.16: SD adder 1-out-of-3 encoding, internal structure and circuit interface.

The general truth-tables for blocks ADD1 and ADD2, with the corresponding logic circuits, are presented in Fig. B.17 and Fig. B.18. To represent 1-out-of-3 digits in VHDL, the signal type ONE_OUT_3 was defined and added to the library of types. The correspondence between ONE_OUT_3 vhdl signals and 1-out-of-3 notation is presented in figure B.17.

a_i	b_i	a_{i-1}	b_{i-1}	c_i	w_i
010	010	—		010	010
010	001	010	010, 001	001	100
010	001	010	100	010	001
010	001	001		001	100
010	001	100		010	001
010	100	010	010, 001	010	100
010	100	010	100	100	001
010	100	001		010	100
010	100	100		100	001
001	010	010	010, 001	001	100
001	010	010	100	010	001
001	010	001		001	100
001	010	100		010	001
001	001	—		001	010
001	100	—		010	010
100	010	010	010, 001	010	100
100	010	010	100	100	001
100	010	001		010	100
100	010	100		100	001
100	001	—		010	010
100	100	—		100	010

$x_i^L = x(i)(0)$
$x_i^M = x(i)(1)$
$x_i^H = x(i)(2)$

1−out−of−3 notation	ONE_OUT_3 representation

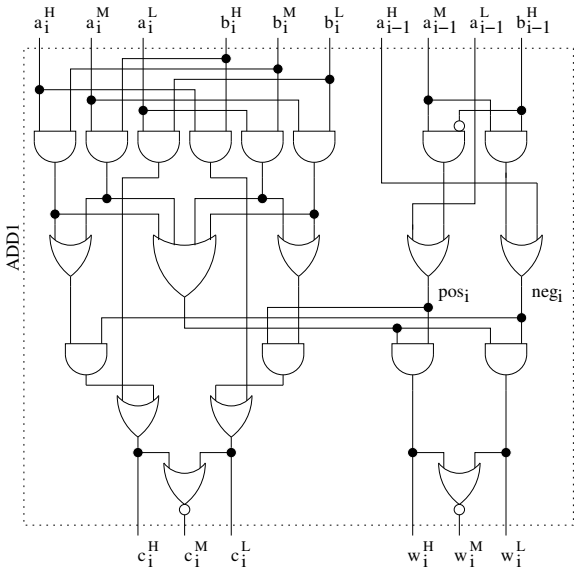


Figure B.17: ADD1 block function table and its logic circuit.

Examples of addition in 1-out-of-3 representation (and signed digit rules) can be seen in Fig. B.19. The conversion from 1-out-of-3 representation to two's complement or signed magnitude is also exemplified. The decoding of the 1-out-of-3 result into a two's complement form or signed magnitude is based on the subtraction of the negative part (value of the negative digits) from the positive part (value of the positive digits). This is accomplished by adding the positive part to the two's complement of the negative part. To add the values in the decoding process is necessary the use of a binary adder, meaning that the 1-out-of-3 addition, regardless its carry-free structure, will depend on a traditional adder with speed limitations to generate its (binary) output. If total self-checking is needed, the binary adder must also be a self-checking version.

w_i	c_i	s_i
010	100	100
001	100	010
100	010	100
010	010	010
001	010	001
001	001	010
010	001	001

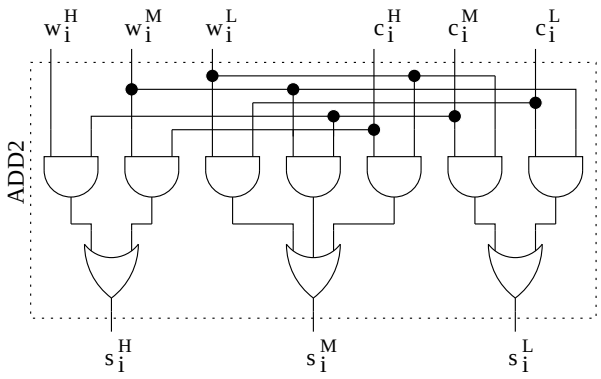


Figure B.18: ADD2 block function table and its logic circuit.

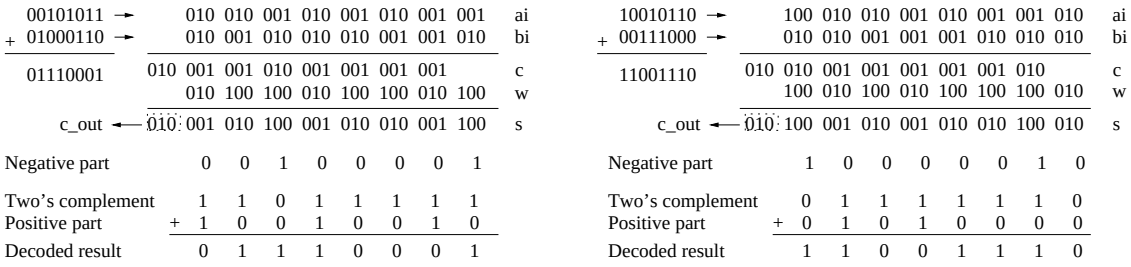


Figure B.19: Examples of 1-out-of-3 addition and decoding.

As in the case of the radix-2 carry-free adder, a version of the present adder targeted for binary inputs representation has been implemented. Considering that input are in two's complement (or signed magnitude) representation, the input set is reduced and simplification is possible. The conversion structure is shown in detail in Fig. B.20a, and the interface of the circuit can be seen in Fig. B.20b. The conversion of the most significant bit (MSB) is similar but not equal to the other digits in the number. The grounded signals presented in the figures are for reference only.

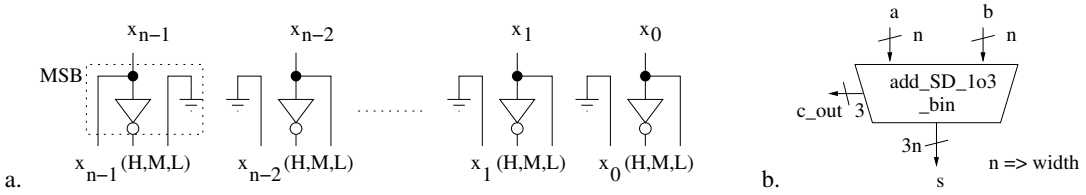


Figure B.20: Adder with binary inputs; a. Conversion circuits; b. Circuit interface.

The rules for generation of the intermediate sum and carry values (ADD1 block) are presented in the tables in Fig. B.21a and Fig. B.21b, for MSB digits and the remaining (n-1) digits. These tables and the circuits are simplified versions, where possible values of inputs are restricted due to their representation (two's complement or signed magnitude).

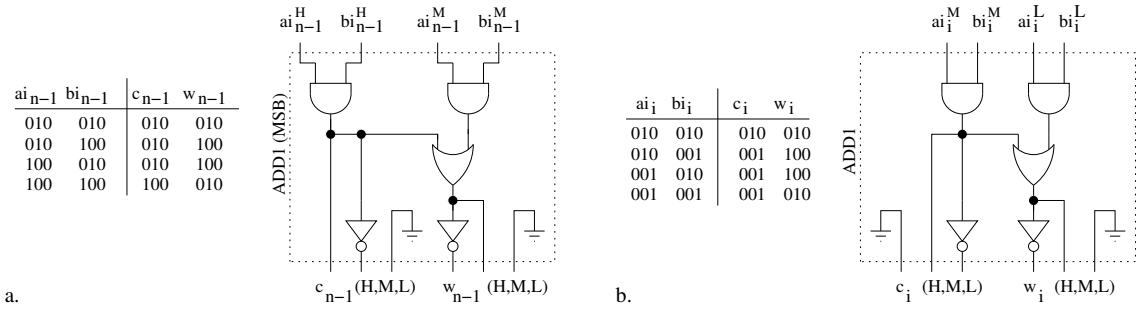
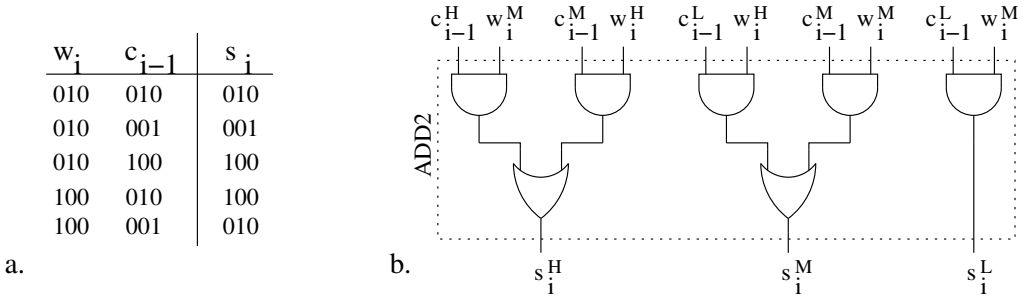


Figure B.21: Rules and circuits for intermediate sum and carry generation for MSB and remaining digits.

The final sum is generated by block ADD2, following the rules on the table shown in Fig. B.22a, and the corresponding circuit is presented in Fig. B.22b.

These two versions of the 1-out-of-3 carry-free adder are compatible with the fault-tolerant approach of m-out-of-n data checking.



w_i	c_{i-1}	s_i
010	010	010
010	001	001
010	100	100
100	010	100
100	001	010

Figure B.22: Rules and circuit for the final sum generation.

B.4 Booth multiplier

The traditional multiplication algorithm is similar to the paper and pencil algorithm taught in school. In a first step, each digit of the multiplier operand multiplies the multiplicand operand, generating a partial product. In the second step, all the partial products are added, respecting the weight of the digits. In the binary version of the paper and pencil algorithm, each bit of the multiplier operand generates a partial product (zero or the multiplicand itself), as can be seen in the example in Fig. B.23a.

The generation of partial products for the paper-and-pencil algorithm can be achieved by an AND array [130], and the addition of the partial products can be achieved by many types of adder structures, like the ones presented in the previous sections. But to optimize the area and the speed of the multiplier, specific structures targeted for multiple operands addition are recommended, like the carry-save arrays and Wallace trees [131].

The most simple multiplier can be seen as an array of $n \times n$ full adders (some can be half adders), n being the width of the operands. The main problems with this simple approach is the length of the critical path, the area of the circuit and the need for specific solutions to multiply signed numbers. The Booth multiplication algorithm is one of the most used implementations, since it allows a reduction in the number of additions necessary and, this way, reduces the hardware size and the delay of the circuit. Another benefit is that the algorithm deals with signed numbers in two's complement representation. The algorithm is based on recoding one operand (multiplicand) according to the value of the other one (multiplier). A more detailed description of the recoding algorithm used in Booth multipliers can be seen in the work of Nicolaidis and Duarte [7] and Marienfeld et al. [8].

In the Booth algorithm, a group of bits in the multiplier operand is responsible for the generation of a partial product. The size of this group may vary, leading to different implementations of the multiplier circuit. In the present library a 2-bit Booth recoding (or Booth-2) implementation is described. In this case, groups of 3 bits are responsible of generating the partial products, with 1-bit overlapping between groups. Fig. B.23b shows the relation among the bits on a group and the partial product that must be generated by the Booth-2 multiplier. Fig. B.23c shows the multiplication of the same operands present on Fig. B.23a, but according to the 2-bit Booth recoding method.

As presented in the figure, the least significant group uses an implicit 0 as the rightmost digit. The general structure of the implemented Booth multiplier can be seen in Fig. B.24.

The partial product generation circuit is presented in Fig. B.25, where the input of the recoding control is a 3-bit data set from the multiplier operand. In a n-bit multiplier, $n/2$ copies of this circuit are necessary for partial products generation. The recoding control for the first partial product can be simplified due to the implicit 0 in the rightmost digit.

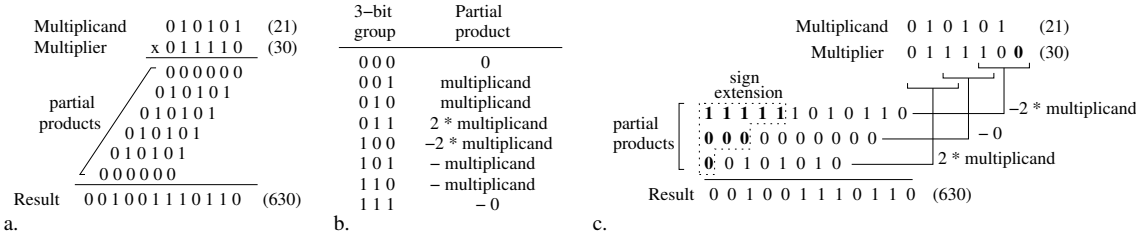


Figure B.23: Multiplication examples; a. Traditional algorithm; b. Partial products generation in Booth-2 algorithm; c. Booth-2 multiplication.

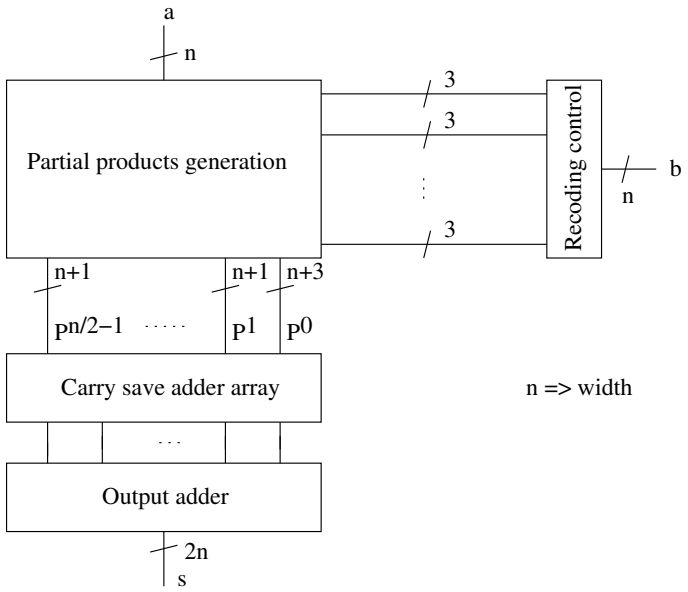


Figure B.24: General structure of the Booth-2 multiplier.

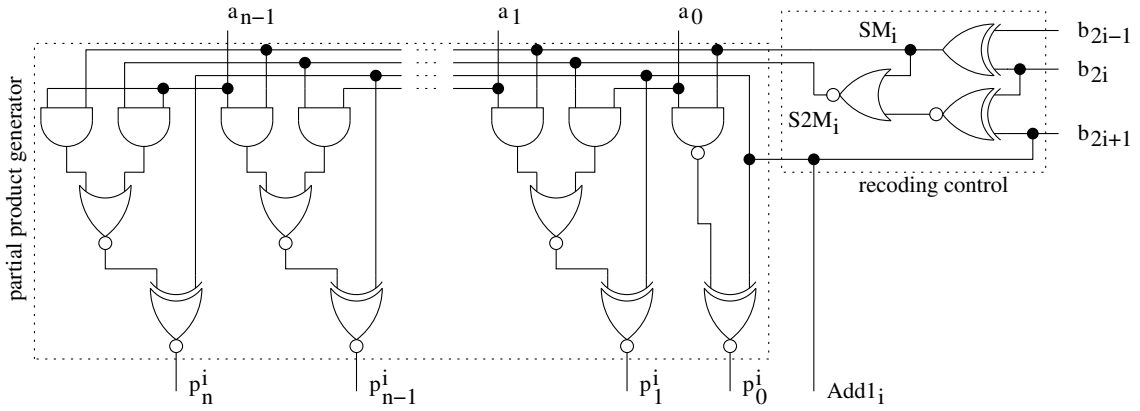


Figure B.25: Partial product generation circuit.

The Booth-2 description in the current library considers $n \times n$ multipliers. The addition of the partial products is done by a carry-save adder, implemented as an array of half adders and full adders. The final sum is computed by a ripple-carry adder. Figure B.26 shows the carry-save adder array structure implemented in the current library, based on the circuit described in the work of Nicolaidis [7]. The proposed structure optimizes the use of hardware resources and reduces the number of sign extended digits.

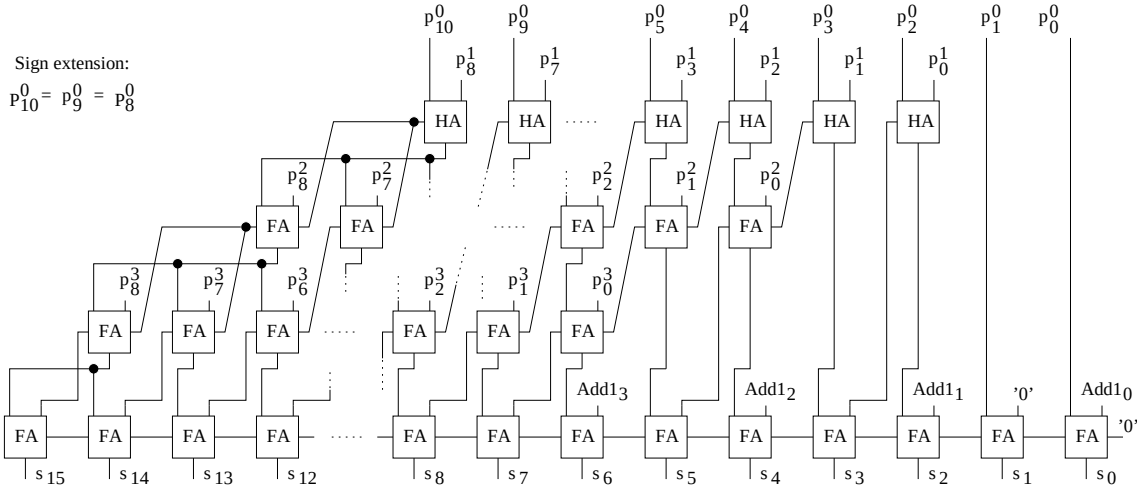


Figure B.26: Carry-save adder array for an 8×8 multiplier.

B.5 Concurrent error detection

Concurrent error detection (CED) and self-checking schemes are based on the concurrent computation of a circuit function or a characteristic of this function, and the checking of the concurrent data against the data generated by the main circuit. Self-checking designs prevent the systems to operate over wrong values and to achieve self-checking capabilities, some kind of redundancy must be introduced in the design, spatial or temporal. Among the existent schemes, duplication is one of the most simple in application, at the price of an overhead of over 100%. Fig. B.27 shows the structure of CED scheme based on duplication.

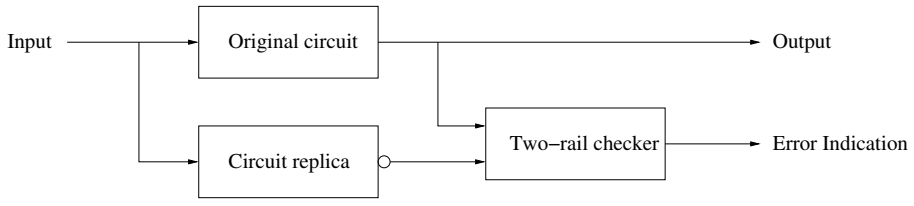


Figure B.27: General structure of a CED scheme based on duplication.

The main advantage of duplication is its straightforward application, where the circuit to be checked is replicated and the outputs of the two circuits are directly compared. In the current library, a self-checking CED scheme based on duplication is implemented, where a circuit replica with complemented outputs is checked against the main circuit by

self-checking two-rail checkers [10]. This scheme can be used to check all types of circuits, independent of their function or data representation.

A two-rail checker works with 1-out-of-2 coded data (code words "01" and "10"), an encoding that prevents the checker to supply erroneous checking values in the presence of stuck-at faults on its own outputs, and allows the indication of errors in the checker itself. Fig. B.28 shows some topologies for a two-rail checker block [10, 132, 133].

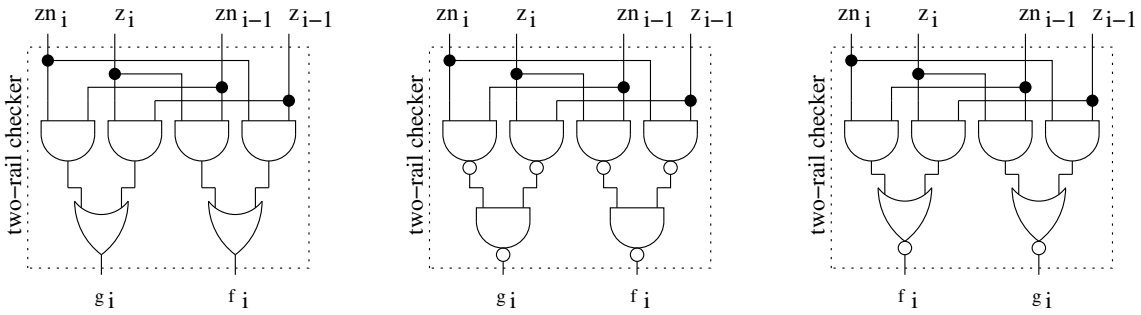


Figure B.28: Two-rail checker implementations.

The checker tree is implemented as a cascaded structure of two-rail checkers, and two examples of 8-bit checker trees are presented in Fig. B.29, a linear tree in Fig. B.29a and a balanced tree in Fig. B.29b. Despite the fact that balanced trees have delay advantages, linear trees are described in the library code, since balanced trees need recursive coding approaches that are not so simple to describe in the VHDL language, but synthesis tools may infer balanced trees from the linear trees description. The checker block indicates the occurrence of an error in the addition or an error on its own structure by supplying the values "00" or "11" (non-code words) at its output, represented by the *ei* signal.

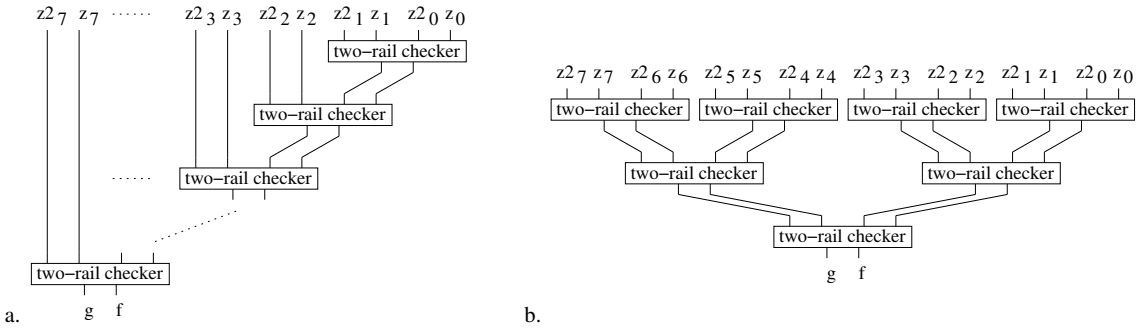


Figure B.29: Checker tree implementations; a. Linear tree; b. Balanced tree.

Fig. B.30a shows the scheme of the described self-checking duplicated adder, as defined in the library. Fig. B.30b presents the interface of the circuit.

Only the standard adders have been implemented with this fault-tolerant approach but the implementation of this CED scheme to all the circuits on the library is a straightforward task.

B.6 Parity prediction

Despite the simplicity of duplication-based CED schemes, the related overhead is far from negligible and alternative schemes, based on coded data have been proposed. Among these

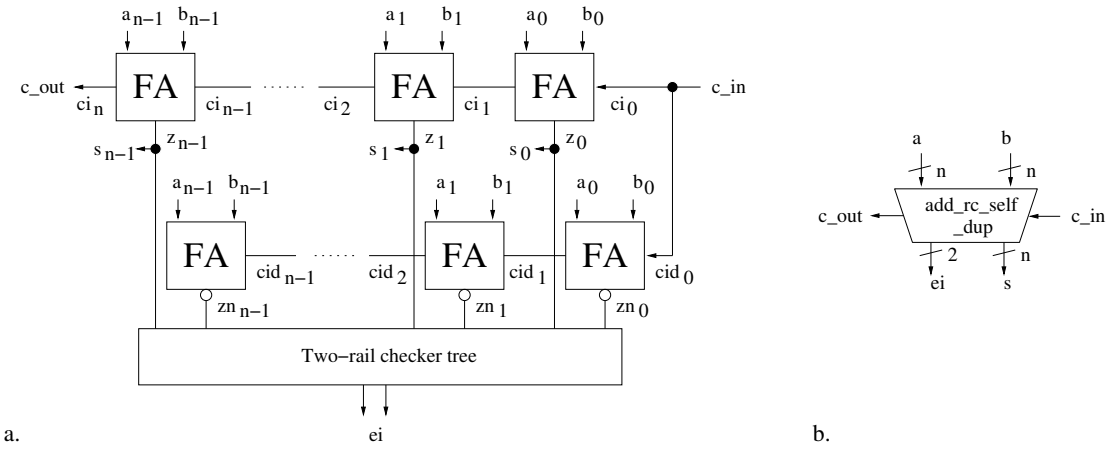


Figure B.30: Duplicated RC adder; a. General scheme; b. Adder interface.

alternatives, parity prediction schemes were found to be cost-effective ones [9, 133]. In the present library, parity prediction self-checking versions of the arithmetic operators have been implemented and are presented in the current section.

The self-checking ripple-carry adder architecture follows the propositions in the work of Nicolaidis [6]. The basic idea is that the parity of the output word (Ps) can be predicted according to the parity of the operands (Pa and Pb) and the parity of the internal carry signals (Pc). The relation between these parities is presented in expression (B.10).

$$Ps = Pa \oplus Pb \oplus Pc \tag{B.10}$$

The parity of the input data can be obtained by XOR trees, as well as the parity of the output, but considering that the adder is a block of a more complex system, the input parities may be considered to be generated outside the adder block as well as the output parity for checking the result. The parity of the carry signal could be obtained by *XORing* the normal carry signals, but common mode faults/errors in the carry circuits would propagate to the outputs as well as to the checking circuit, remaining undetected. To avoid this problem, a duplicated carry generator circuit is implemented. Instead of using an XOR tree to generate the parity of the carry signal, two-rail checkers are used to generate the parity (one property of two-rail checkers is that their outputs reflect the parity of their inputs). Fig. B.31a shows the general structure of the self-checking adder and Fig. B.31b presents the circuit interface.

The second carry generator is similar to the one present in the full adder cell. Fig. B.32a shows the basic adder cell with the complemented carry signal generator. The checker tree for an 8-bit adder is presented in Fig. B.32b. The output of the two-rail tree is a signal (ei, error indication) that indicates the occurrence of an error in the carry signals when its two bits have the same value. The errors in the output parity can be checked by further circuitry according to the parity predicted signal Ps.

The work of Vasudevan and Lala [125] proposes a self-checking version of a carry-select adder that is based on self-checking multiplexers and a configuration logic block that generates complemented outputs that can be verified by two-rail checkers. The self-checking multiplexer circuit is described on MOS transistor level and for the purpose of the present library the description of low-level circuits is not desired. This way, the self-checking version of the carry select adder is based on the work of Nicolaidis [6]. Fig. B.33a

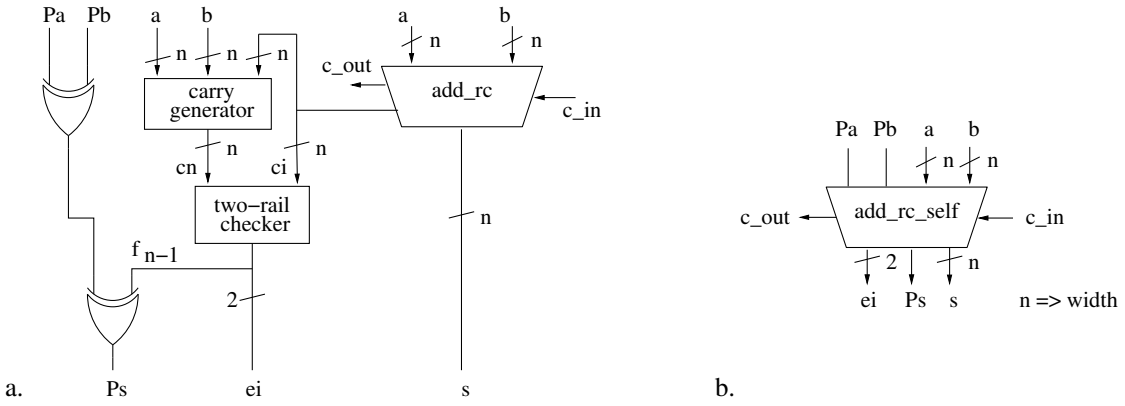


Figure B.31: Self-checking RC adder; a. General structure; b. Circuit interface.

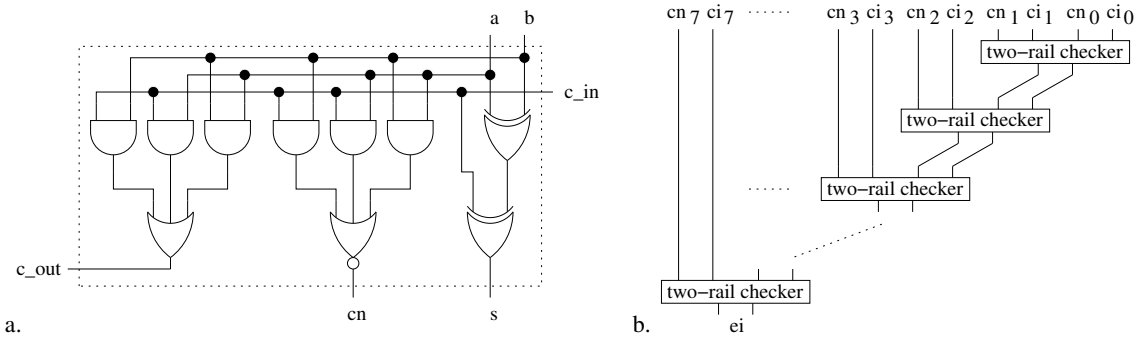


Figure B.32: Self-checking RC adder; a. Full adder with duplicated carry; b. 8-bit two-rail checker tree.

shows the general structure of the implemented self-checking carry-select adder and the interface of the circuit is shown in Fig. B.33b. As in the self-checking ripple-carry adder, the output *ei* supplies a two-rail code concerning the carry signal, where "00" and "11" values indicate the occurrence of an error. The *Ps* signal is the predicted parity for the result value and must be checked by an external checker.

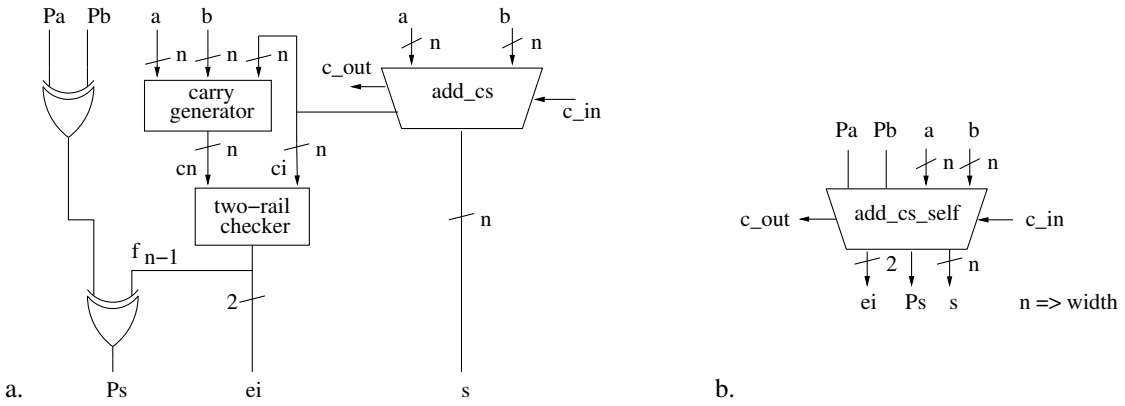


Figure B.33: Self-checking CS adder; a. General structure; b. Circuit interface.

Fig. B.34a shows the circuits that generate the complemented carry signals that are used as a two-rail vector along with the normal carry signal. Fig. B.34b shows the two-rail checker tree that generates the signal *ei*. The area overhead for this adder is due to the checking tree, the duplicated carry circuits and the need for a fourth multiplexer in the carry select block.

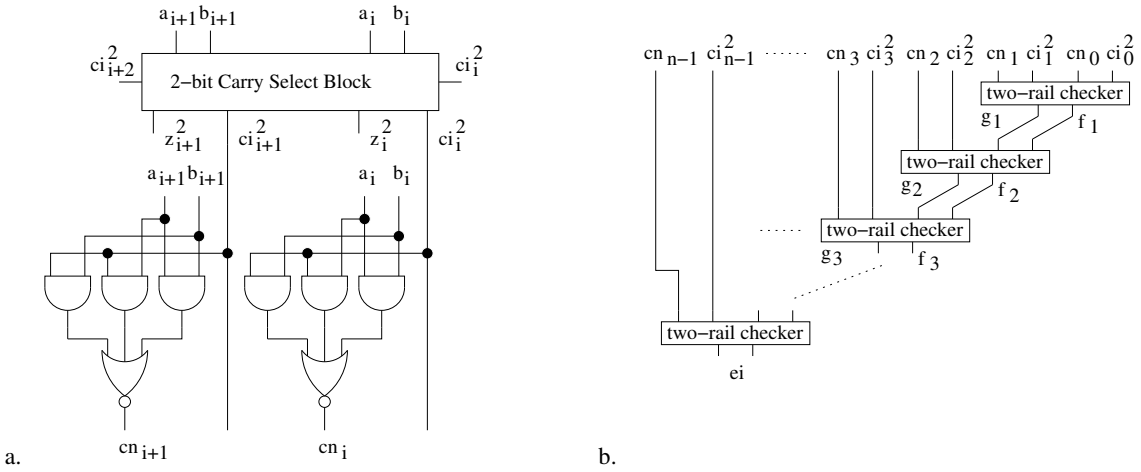


Figure B.34: Self-checking CS adder; a. 2-bit CS block with complemented carry generation; b. Two-rail checker tree.

The carry-lookahead adder uses the same addition "algorithm" found in the ripple-carry adder, and the techniques applied to check the operation of the RC adder can be used to check the CLA adder [6]. The general structure of this self-checking CLA adder is shown in Fig. B.35a and its interface is presented in Fig. B.35b. As is the case for the self-checking RC adder, the output *ei* supplies a two-rail checking code concerning the carry signal, where "00" and "11" values indicate the occurrence of an error. The *Ps* signal

represents the predicted parity of the result value and must be checked by a further circuit in the system.

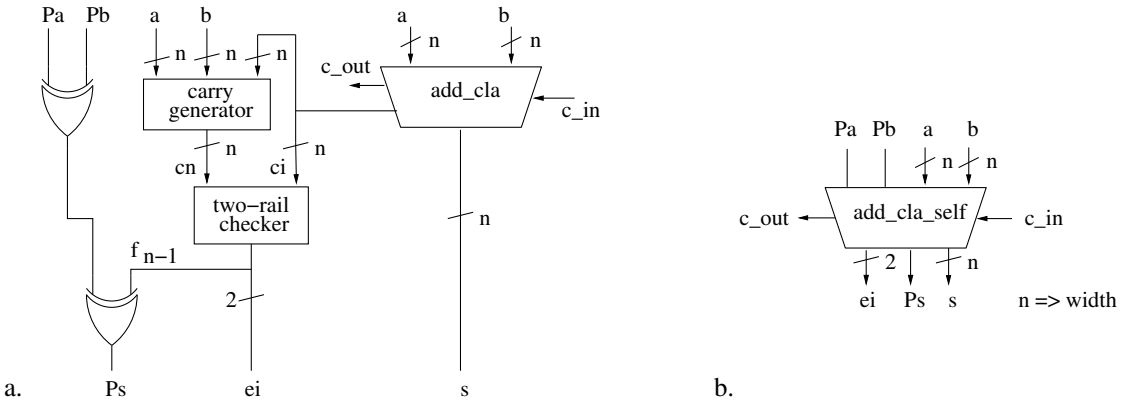


Figure B.35: Self-checking CLA adder. a. General structure; b. Circuit interface.

Considering that the carry generation structure of the CLA adder is too complex to be duplicated, an alternative proposed is the use of simple ripple-carry-like circuits to generate the carry check signals. There are no speed reduction since the carry duplicated cells use as its inputs the carry signals generated by the CLA blocks. The circuit that generates the complemented versions of the carry signal is presented in Fig. B.36a. The checking circuits for the carry signals of each CLA block are presented in Fig. B.36b and the linear tree that checks the two-rail code of each CLA block is exemplified in Fig. B.36c.

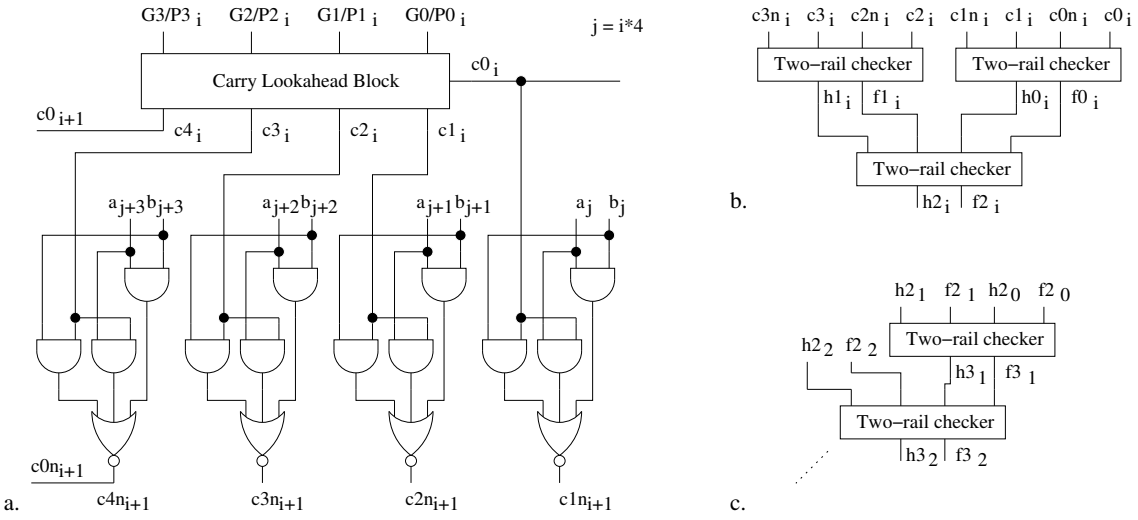


Figure B.36: Self-checking CLA adder; a. Duplicated carry generation; b. CLA block carry checking; c. Two-rail tree.

As referred before, the self-checking version of the radix-2 signed digit adder is based on the work of Cardarilli et al. [126]. Considering the coding defined in figure B.9a and the tables presented in figure B.10, some parity relations can be explored to implement self-checking capabilities in the related signed digit adder structure. The parity of the intermediate sum w can be predicted by the relation in expression (B.11) and the parity of the result s can be predicted by the relation in expression (B.12).

$$Pw = Pa \oplus Pb \tag{B.11}$$

$$Ps = Pc \oplus Pw \tag{B.12}$$

The parities of w and s are generated by XOR trees like the one in Fig. B.37a. The parity of c ($Pc(i)$) is predicted by a custom circuit (Fig. B.37b) based on the table in Fig. B.10a, plus an XOR tree. The parities of a and b must be generated by previous circuits. The organization of the self-checking adder is shown in Fig. B.38a and its interface is presented in Fig. B.38b. In the original proposal in [126], the parity checking circuits that generate $ei1$ and $ei2$ are two-rail parity checkers but in the present library parity checking is done by simple XOR gates. A value "1" in output $ei1$ (error indicator 1) or $ei2$ indicates that an error has occurred (stuck-at faults) in the addition or in the checking circuits. This version is not totally self-checking (TSC), since the two-rail checkers were not used.

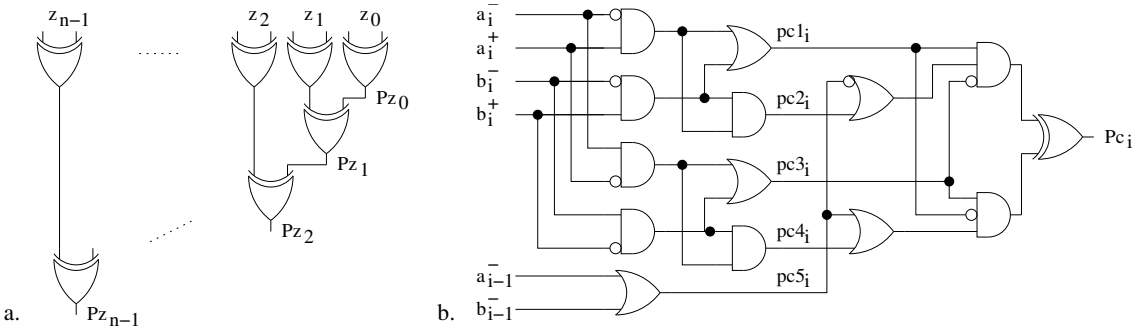


Figure B.37: Self-checking circuits; a. XOR tree; b. Intermediate carry parity prediction circuit.

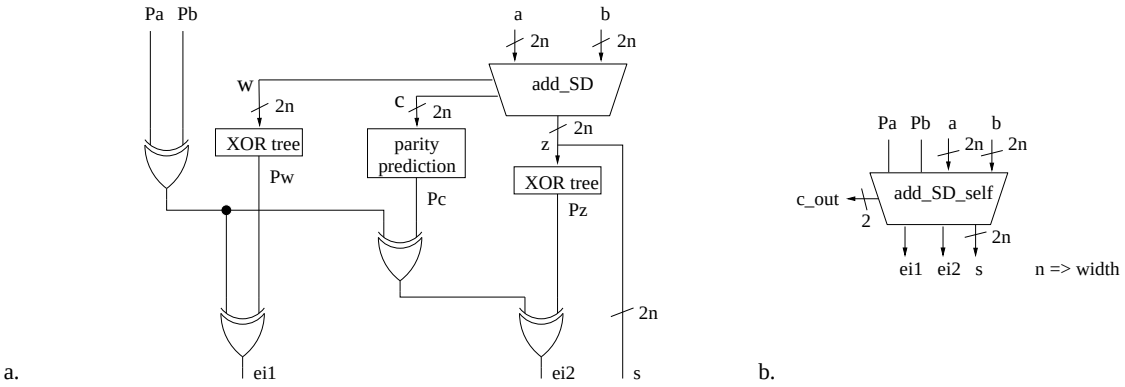


Figure B.38: Self-checking adder; a. General structure; b. Circuit interface.

The general structure of the self-checking Booth multiplier is presented in Fig. B.39. The circuit follows the principles of the work of Nicolaidis and Duarte [7], and is based on parity prediction.

Considering that multiplication is achieved by the addition of multiple partial products, the parity checking property applied to adders can be extended to multipliers. In this case, the parity prediction in a multiplier is achieved by comparing the parity of the result (Ps) with the modulo 2 sum of the parity of the partial products (Ppp) and the parity of the carries (Pc) of the adder cells in the circuit, as can be seen in expression (B.13).

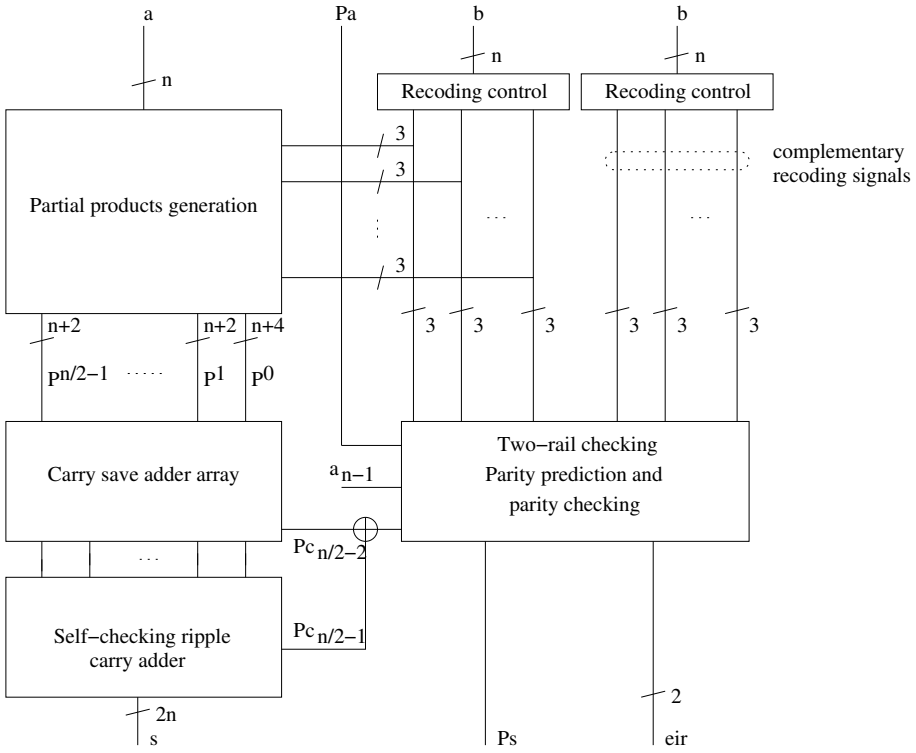


Figure B.39: General structure of the self-checking Booth multiplier.

$$Ps = Ppp \oplus Pc \tag{B.13}$$

In a traditional multiplier (paper-and-pencil algorithm), the parity of the partial products can be calculated from the parity of the operands, Pa and Pb, according to the expression in (B.14). In a Booth multiplier, with the operand recoding method, the determination of the parity of the partial products is more complex and takes the parity of the recoding signals into account.

$$Ppp = Pa \wedge Pb \tag{B.14}$$

To add fault secureness to the circuit, other components are modified to be checked. To detect faults in the recoding control circuit, that could propagate to multiple digits in the related partial product, a two-rail checking is implemented, with the recoding control circuit being duplicated and generating complemented recoding signals. Fig. B.39 shows the recoding control checking scheme and Fig. B.40a details the recoding control signals. The recoding control checking and the parity of the recoding signals is determined by two-rail checking trees, as shown in Fig. B.40b.

To guarantee fault secureness in the carry-save array, the carry generation circuits of the adder cells have been duplicated, and additional sum cells were introduced in the carry-save array topology [7]. Despite the fact that the additional digits of the partial products needed by the sum cells are copies of the sign digits, they must be generated by a duplicated circuit instead of being hardwired signed extended. The modifications in the carry-save array for an 8×8 multiplier can be seen in Fig. B.41.

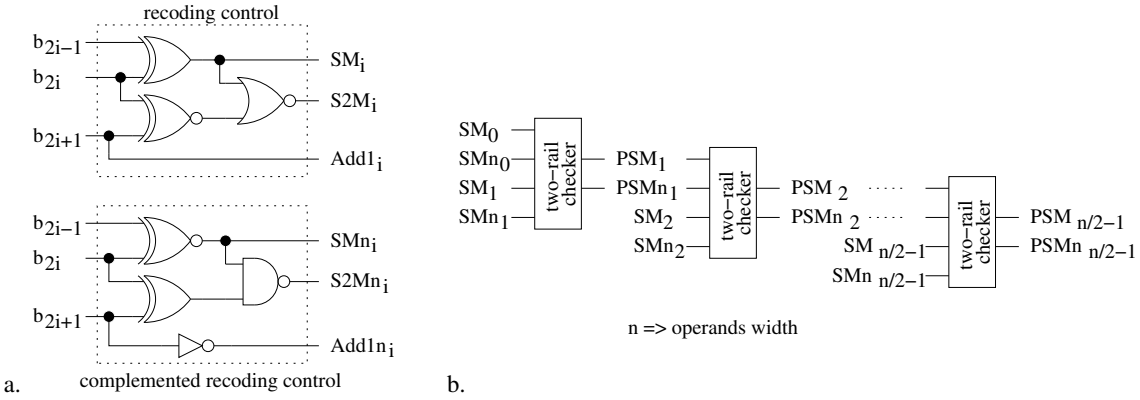


Figure B.40: Recoding control; a. Complementary logic; b. Two-rail checking tree.

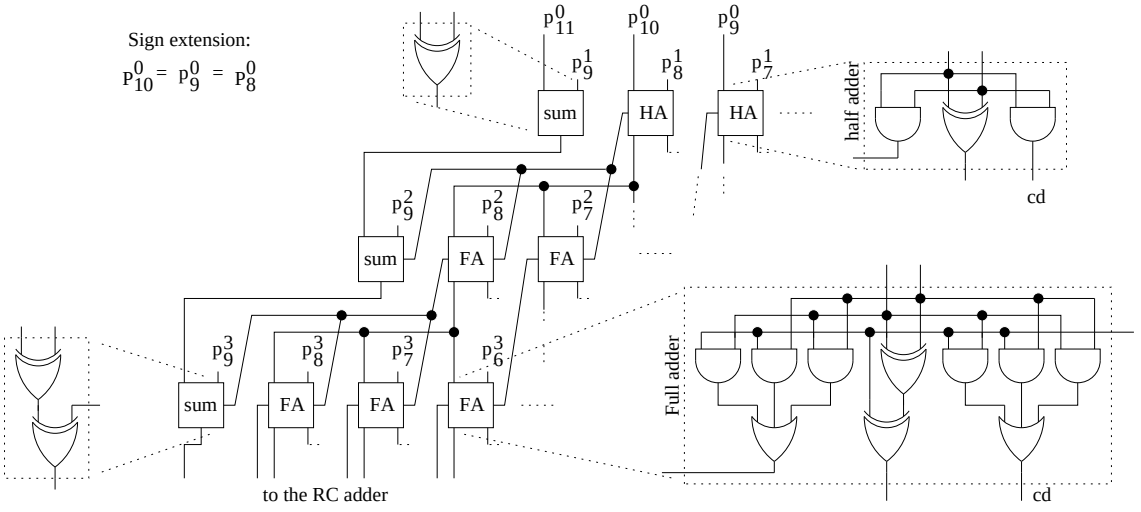


Figure B.41: Modifications in the carry-save adder array.

The parity of the carry signals (Pc) in the carry-save array is determined by an XOR tree, which can be seen in Fig. B.42. The inputs for this tree are the duplicated carry signals. A modulo 2 sum of the parity of the carries in the carry-save array, and the parity computation of the carries in the final RC adder, are implemented to generate the parity Pc.

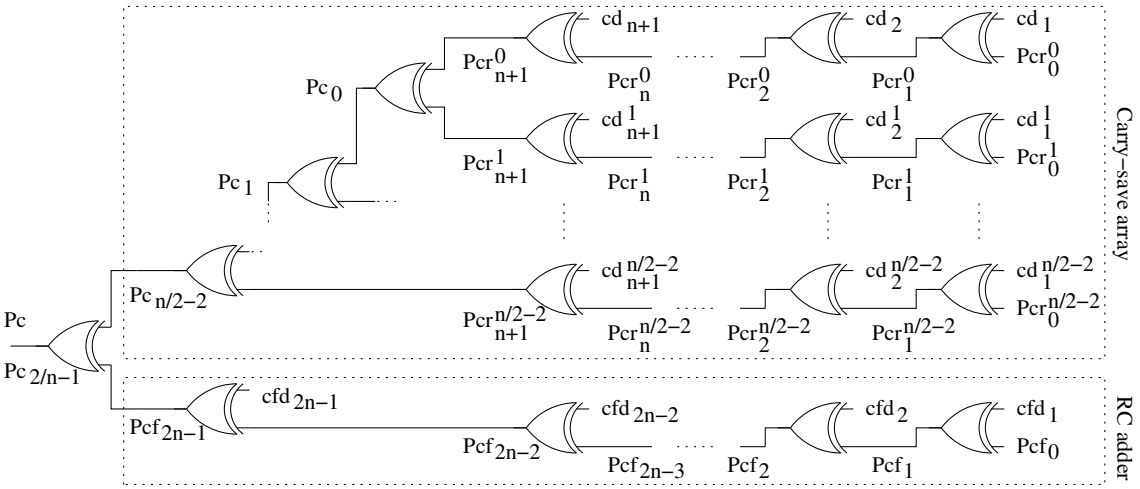


Figure B.42: Parity determination for the carries in the carry-save array and the final RC adder.

Fig. B.43 shows the circuit that generates the output parity prediction and checks the recoding signals. The parity of the recoding signals is used in conjunction with the operand a and its externally produced parity Pa, to generate the parity of the partial products Ppp. The predicted parity of the output (Ps) must be checked by an external circuit. The recoding checking is achieved by two-rail checkers, that generate the recoding error indication signal eir. Values '00' or '11' in the eir output indicate an error in the recoding circuits.

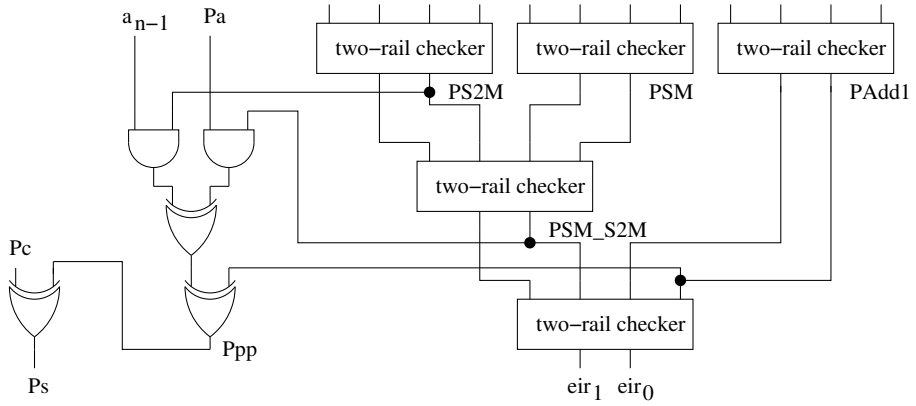


Figure B.43: Parity prediction and recoding logic checking circuit.

B.7 1-out-of-3 checking

The 1-out-of-3 data representation in the considered signed digit adder allows the detection of all unidirectional errors [10]. Since the output of the adder is in 1-out-of-3 representation, self-checking is achieved by introducing 1-out-of-3 checkers in each of the output digits. Fig. B.44a shows the interface of the self-checking adder, where the signal **ei** means *error indication* and is a 2-bit signal that indicates an error when its 2 bits present the same value ("00" or "11"). Fig. B.44b shows the general structure of the self-checking adder.

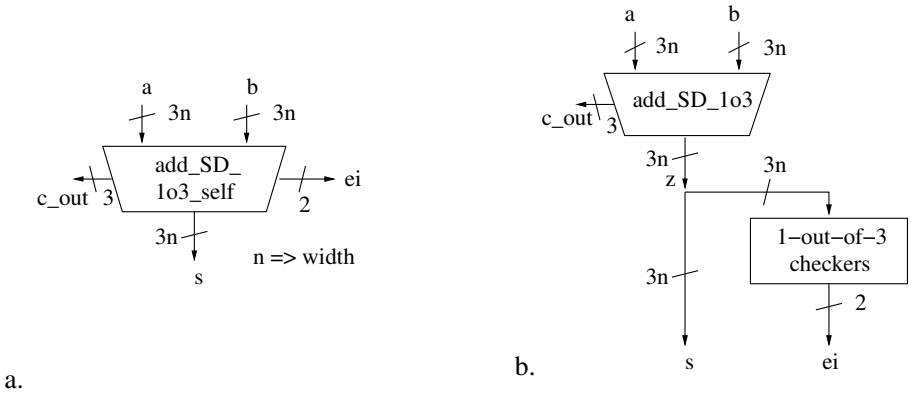


Figure B.44: Self-checking 1-out-of-3 adder; a. Circuit interface; b. Internal structure.

The type of checker implemented in the circuit is presented in figure B.45 and is based on the works of Lala, Wang and Paschalis et al. [10, 134, 135].

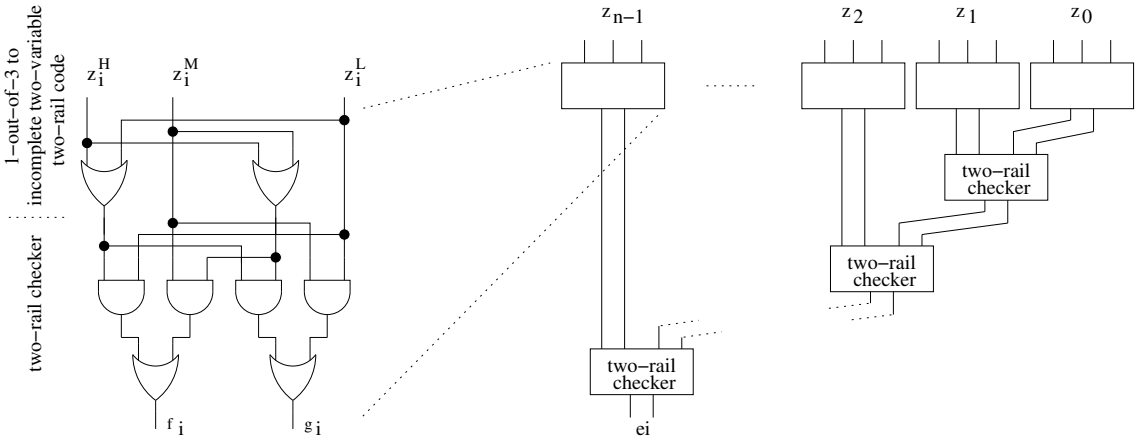


Figure B.45: 1-out-of-3 and two-rail checkers tree.

Since self-checking is achieved by verifying the code on the outputs of the referred adder, the same structure of the circuit with 1-out-of-3 inputs can be used to verify the presence of errors in the adder with two's complement inputs. The interface of the circuit and its internal structure are presented in figure B.46.

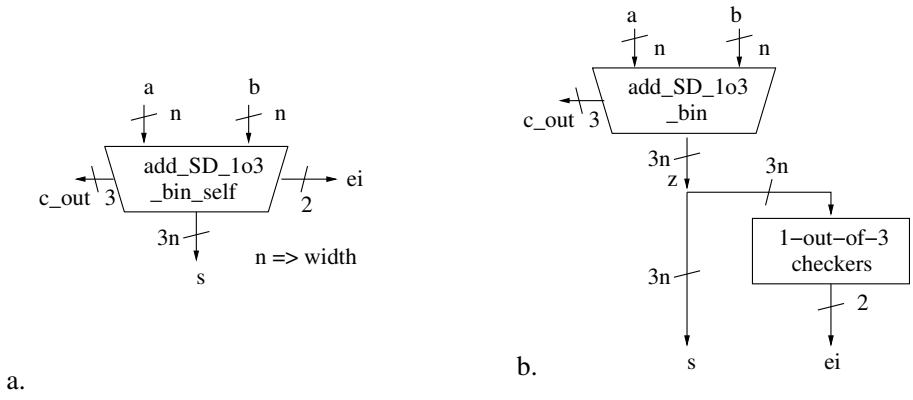


Figure B.46: Self-checking 1-out-of-3 adder; a. Circuit interface; b. Internal structure.

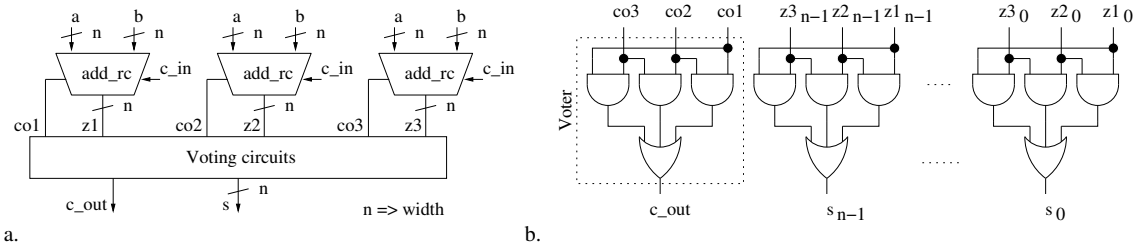


Figure B.47: TMR RC adder; a. General structure; b. Output and carry voting logic.

B.8 Triple modular redundancy

The last fault-tolerant approach considered in the present library is the triple modular redundancy (TMR). In this method, three replicas of the same circuit operate concurrently and the output is determined by a majority function or voting [10]. In the case of a fault/error in one circuit, the output is automatically masked by the other two values. The simplest voting circuit is a two-level AND-OR logic that implements a 2-out-of-3 (majority) function. The voting circuit is the main weakness of this method, since a single-fault in the logic of a voter may lead to an incorrect output. An increase in the reliability of the circuit may be achieved by replicating the voting circuit, as presented in the work of Wirthlin et al. [136].

Fig. B.47a shows the general structure of the TMR version of the ripple-carry adder and Fig. B.47b shows the voting logic for generating the output and carry signals.

The remaining circuits of the library can be hardened in the same way, since the general structure of the TMR approach is universal, and that is one of its advantages.

In the particular case of the ripple-carry adder, a second topology based on a finer grain TMR implementation has been described. This adder uses the concept of triple modular redundancy (TMR) in a interlaced structure. The traditional TMR implementation replicates the complete system and the voting operation determines the output value. This way, the fault-tolerant structure can sustain many faults in one system replica but cannot sustain simultaneous single-faults in different replicas. By applying the voting scheme to internal signals of the adder (the carry signals in this case), the implemented fault-tolerant scheme can achieve correct operation in the presence of single/multiple faults in different system replicas. In the case of the RC adder, the internal redundant voting scheme is done

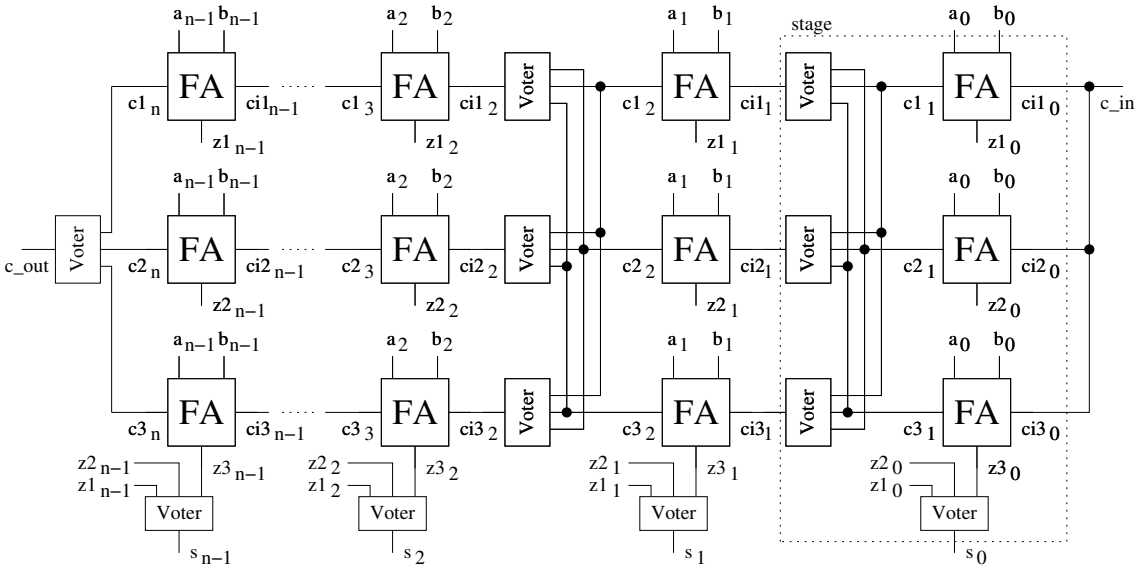


Figure B.48: General structure of the fine-grain TMR RC adder.

after each full adder stage, as shown in Fig. B.48. With such structure, each stage can sustain faults in one replica and correct operation is achieved even with multiple faults in different stages. In the case of a fault/defect in one replica of a stage, the output is masked by the other two replicas. The internal voting circuits are TMR versions of the single voter of the output signals, as presented in [136].

B.9 Digital filters

The arithmetic fault-tolerant components implemented in the current library were originally defined to be used in the development of signal processing applications. As an important representative of signal processing architectures, finite impulse response (FIR) filters are a natural choice for the exploration of fault-tolerant and self-checking methods. In the present work different organizations of this type of filter were described and compared to evaluate the impact of the concerning methods in such systems. As presented in the previous sections, some of the fault-tolerant blocks depend on signals generated by the system where they are embedded, and so, the overhead of a fault-tolerant system may vary from the overhead of its fault-tolerant components.

Finite impulse response filters are systems that implement a convolution, the mathematical operation that generates an output signal by combining two other ones, the impulse response ($h(n)$) of the filter itself and the input signal to be filtered ($x(n)$). Fig. B.49 shows a filter block and the convolution operation that it implements along with another usual representation of convolution [137, 138].

The number of taps in a filter represents the number of multiplicative coefficients in the impulse response and is a characterizing parameter of the filter. Considering an M-tap filter, the convolution to be executed can be modeled by the expression in (B.15).

$$tp_{y(n)} = x(n)h_0 + x(n-1)h_1 + ... + x(n-M-1)h_{M-1} \tag{B.15}$$

Considering different constraints for a filtering application, like operation frequency,

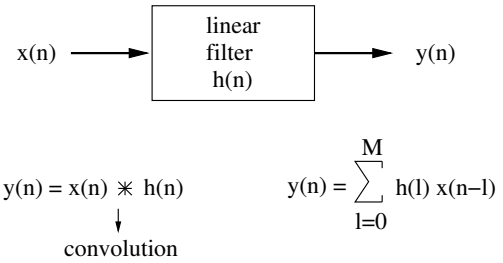


Figure B.49: Filter block and convolution representation.

circuit area, data throughput or power consumption, different architectures can be explored to execute the operation defined in expression (B.15). Two architectures are considered in the present work, a sequential filter and a parallel filter. In the sequential version of the filter, each convolution is calculated through many sequential steps, where a step is considered to be executed in one clock cycle. The simplest implementation is based on an architecture with one multiplier and one adder that execute the multiply-and-accumulate (MAC) operation necessary for sequential processing. In this case, the MAC operation must be repeated as many times as the number of the taps in the filter.

To avoid confusion concerning the graphical representation of the arithmetic operators, they will be shown as the blocks in Fig. B.50. In the case of the parity prediction operators, the parity of the output (Ps) is a predicted value for the result (s). The signal ei is the two-rail signal that signalizes an error in the circuit.

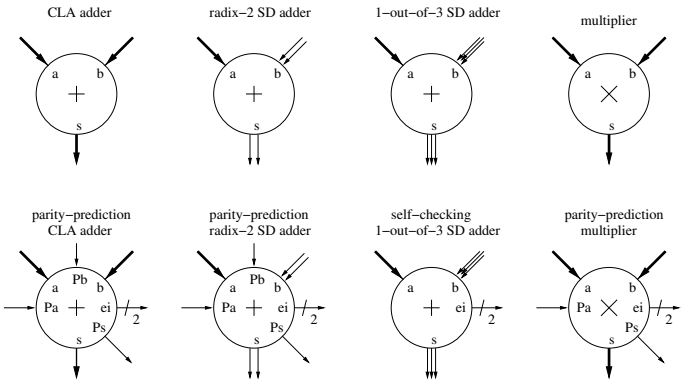


Figure B.50: Representation of arithmetic blocks.

Fig. B.51 shows a sequential FIR filter architecture, with the delay line that receives the input data, the registers for the coefficients storage and the MAC unit. Memory-based or register-based storage may be used, but in the present work only registers are considered. The circuit necessary to update the coefficients is not presented in the figure.

The performance of this type of filter is constrained by its operation frequency and its size (number of taps), which determines the number of storage locations and the number of clock cycles needed to implement a single convolution. Most of digital signal processors (DSP processors) are based on a sequential filtering algorithm, since the convolution operation uses the fixed MAC architecture in conjunction with the embedded memory available on such processors to implement applications demanding different filter sizes. Another part of the filter that's not presented in the figure is the control circuit, based on an ordinary finite state machine (FSM).

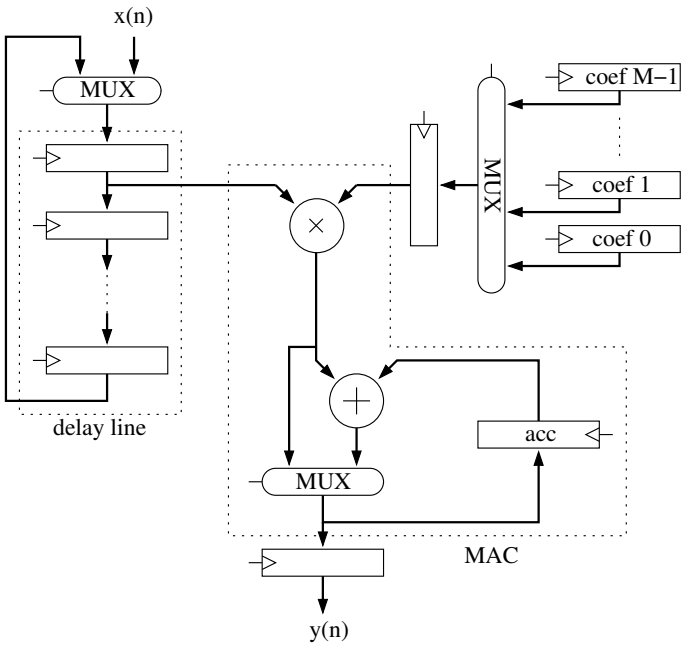


Figure B.51: Sequential FIR filter architecture.

Fig. B.52 shows a parallel FIR filter architecture. This version of the filter is capable of calculating one convolution at each clock cycle. To achieve such throughput this version of the filter uses as many multipliers and adders as the number of taps. The circuit that updates the coefficients is not presented in the referred figure.

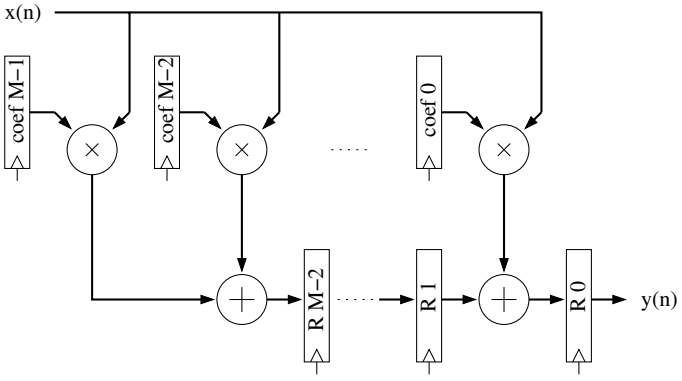


Figure B.52: Parallel FIR filter.

Specific optimization procedures are not considered, like pipelining or parallel filter reduction based on symmetric coefficients.

The introduction of self-checking capabilities in the filters, concerning the duplication method, will not be detailed here since its implementation does not seems to pose any problem. Examples of parity prediction schemes can also be seen in the works of Nicolaidis [6] and [139], but some details regarding the proposed application of parity checking in the target filters will be presented.

As shown in the previous sections, the design of self-checking circuits is based on two-rail error indication signals and the use of two-rail checker trees to generate a single two-rail

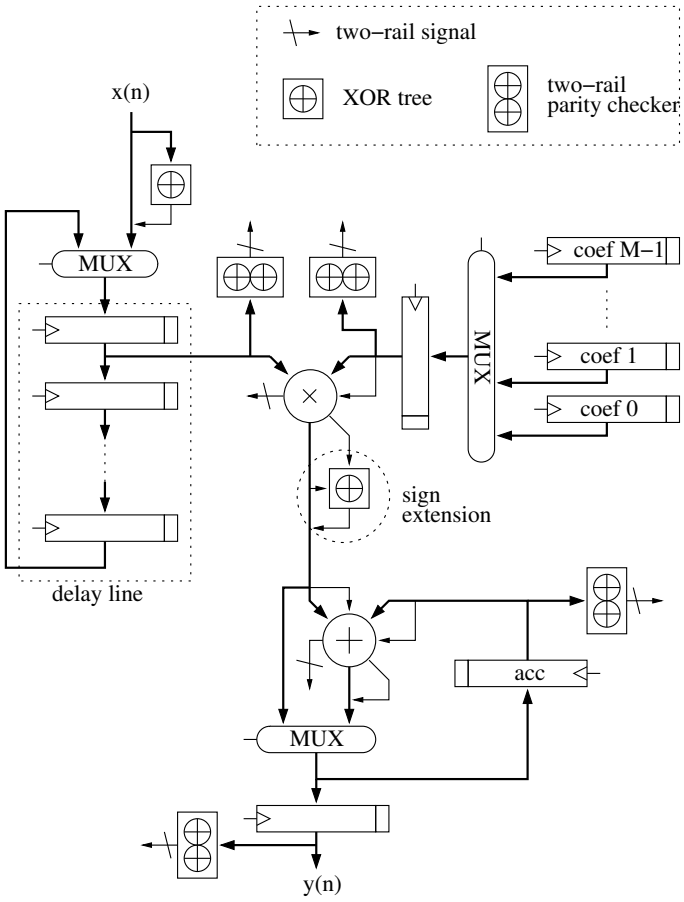


Figure B.53: Self-checking sequential filter.

output signal to sign any error along the architecture. That's the method applied in the filters of the present work. Fig. B.53 shows some of the modifications implemented in the sequential filter to introduce self-checking capabilities.

The two-rail parity checking circuit follows the design proposed in the works of Noufal and Anghel [133, 29] for self-checking parity checkers. The circuit responsible for the update of the coefficients is not presented in the figure but this circuit must also generate the parity bit to be stored in the registers. Other hidden blocks in the figure are the control FSM that is implemented using duplication and the checking tree that receives all of the two-rail signals generated in the system.

The versions of the filter with signed digit (SD) adders have some differences regarding Fig. B.53, as the use of 1-out-of-3 checkers and improved bus widths (and widths of the concerned multiplexers, registers and checkers). The SD adders can be optimized considering that one of their inputs, coming from the output of the multiplier, is in binary representation. As referred before, one major drawback of SD adders is the time and area overhead necessary for output data conversion circuits, based on a binary adder, and these circuits are not yet considered in the present work for simplicity. Fig. B.54 shows the self-checking addition part of the filter with the radix-2 SD adder.

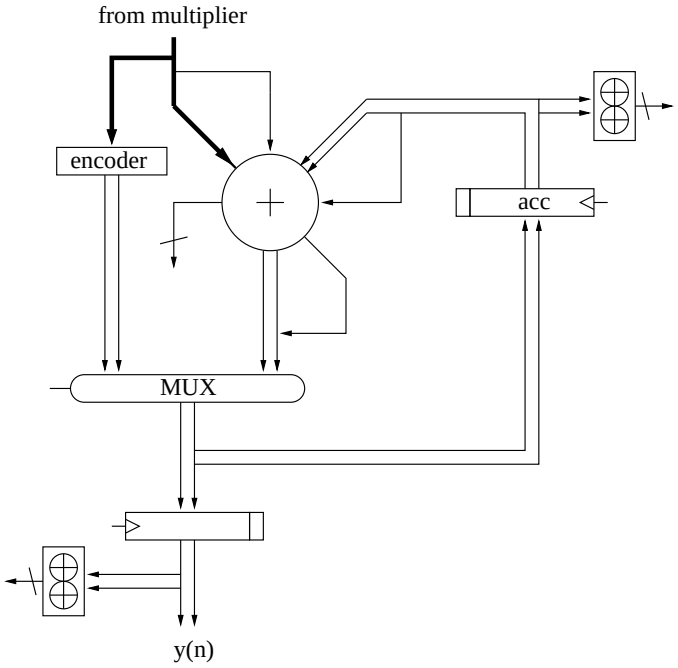


Figure B.54: Self-checking sequential filter with SD adder.

As presented in Fig. B.52, the parallel filter is a repetitive structure based on a replication of the circuits that integrate a tap block. The parity-based self-checking version of a filter tap is presented in Fig. B.55. The writing circuit for the coefficient registers is not presented in the figure and neither is the two-rail checker tree that checks the two-rail signals concerning the tap block.

Fig. B.56 shows the addition part of a filter tap based on the signed digit adder with 1-out-of-3 encoding. The parity checker in the binary input of the adder is necessary since the parity predicted in the self-checking multiplier is not used in the self-checking addition.

The filter taps discussed in the current section have been described based on the arith-

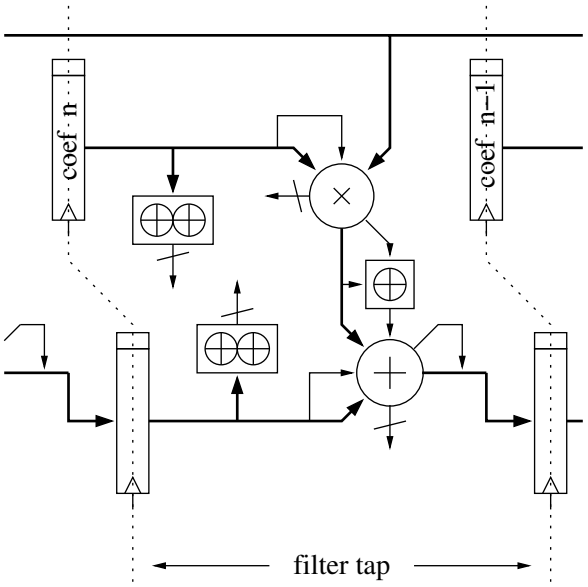


Figure B.55: Self-checking parallel filter tap.

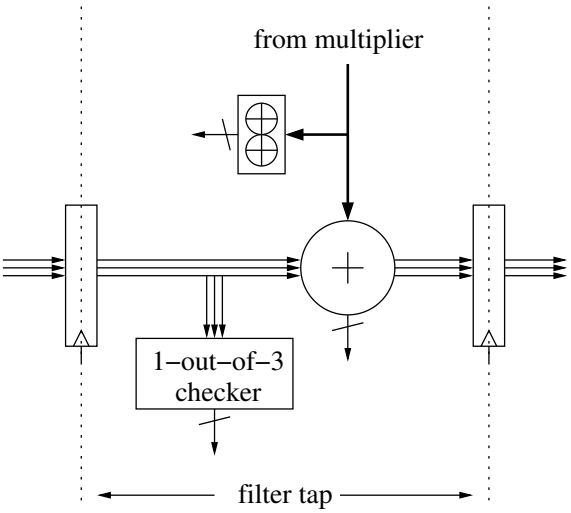


Figure B.56: Self-checking parallel filter tap based on a 1-out-of-3 SD adder.

metic operator blocks present in the fault-tolerant library. Besides the filter taps, parallel and sequential versions of complete filters have also been described and synthesized. These circuits help to give a more realistic idea of the overheads involved in fault tolerance of complex processing blocks. The synthesis results for these circuits can be seen at the next section, along with the synthesis results of all the block on the fault-tolerant library.

B.10 Synthesis results

The parameterized circuits described in the present fault-tolerant library have been synthesized by the tool Cadence BuildGates, targeted to 130-nm standard cells from STMicroelectronics. The complete results of the synthesis of the arithmetic operators can be seen on Tables B.1 and B.2. Table B.1 presents the area of implementation and number of logic cells in the synthesized circuits according to different data widths.

The nomenclature used corresponds to the following definitions : **rca**, ripple-carry adder; **csa**, carry-select adder; **cla**, carry-lookahead adder; **fca**, functional adder; **sd2**, radix-2 signed digit adder; **sd3**, 1-out-of-3 signed digit adder; **booth**, Booth multiplier; **fcm**, functional multiplier; **dup**, duplication-based fault tolerance; **par**, parity-based fault tolerance; **tmr2**, interlaced triple modular redundance; **1o3**, 1-out-of-3 checking. Functional adders and multipliers corresponds to proprietary circuits generated according to datapath functions available from the synthesis tool or standard cell library.

Table B.2 presents the propagation delay and the estimated power consumption of the synthesized circuits according to different data widths.

The results presented here are the raw data obtained from the synthesis reports and several analysis can be made based on the presented results. Since the objective of this section is indeed the presentation of this raw data, no analysis on the data will be presented here and comparisons targeting a definition of the best circuits is not treated here, since the choice of the best circuit is dependent on the design objectives. A brief discussion about these synthesis results is presented in section 2.8, where some critical analysis is made, concerning a pre-selection of the most adequate circuits to evaluate considering a reliability analysis.

Table B.3 presents the synthesis results for the filter taps discussed in the previous section. Consider 8-bit data filter taps, with different combinations of adders and multipliers. The nomenclature used for taps denomination are : **cla**, carry-lookahead adder as the tap adder; **fca**, functional block as the tap adder; **sd2**, radix-2 signed digit adder; **sd3**, 1-out-of-3 signed digit adder; **fcm**, functional multiplier; **dup**, duplication-based fault tolerance; **par**, parity-based fault tolerance; **full**, radix-2 signed-digit-based multiplier. Booth multipliers are considered in the cases where no other references are made.

Table B.4 presents the synthesis results for the parallel filters discussed in the previous section. Consider 8-bit data and 16-tap filters, with different combinations of adders and multipliers.

Table B.5 presents the synthesis results for the sequential filters discussed in the previous section. Consider 8-bit data and 16-tap filters, with different combinations of adders and multipliers.

Table B.1: Synthesis results for the circuits in the fault-tolerant library

circuit	area(μm^2)				logic cells			
	data width				data width			
	8	12	16	32	8	12	16	32
rca	337	514	692	1402	18	26	34	66
rca dup	948	1440	1932	3901	69	101	133	261
rca par	674	1053	1432	2949	51	79	107	219
rca tmr	1138	1727	2316	4672	63	91	119	231
rca tmr2	1327	2062	2796	5733	72	112	152	312
csa	484	758	1033	2130	28	44	60	124
csa dup	1243	1928	2614	5358	89	137	185	377
csa par	821	1297	1773	3677	58	92	126	262
csa tmr	1579	2459	3338	6856	93	145	197	405
cla	492	773	2655	5446	44	70	96	200
cla dup	1259	1957	1727	3583	121	189	257	529
cla par	799	1263	3399	6990	73	117	161	337
cla tmr	1604	2501	1052	2175	141	223	305	633
fca	520	922	1309	3022	72	126	174	420
fca dup	1351	2255	3167	7141	177	301	413	969
fca tmr	1688	2949	4168	9531	225	391	539	1293
sd2	2513	3869	5225	10647	193	297	401	817
sd2 par	3958	5313	6669	12091	312	416	520	936
sd3	1694	2622	3550	7262	177	273	369	753
sd3 1o3	2201	3419	4638	9511	233	361	489	1001
booth	2868	6195	10711	40703	182	372	626	2282
booth par	4726	9777	16497	60068	296	585	962	3350
fcm	2860	5920	9888	35481	273	564	875	2642
fcm dup	6161	13443	22536	76654	604	1184	1970	5728

Table B.2: Synthesis results for the circuits in the fault-tolerant library

circuit	delay(ns)				power(μw)			
	data width				data width			
	8	12	16	32	8	12	16	32
rca	2.64	3.92	5.2	10.32	2.25	3.4	4.57	9.27
rca dup	3.21	4.5	5.78	10.92	6.39	9.72	13.1	26.3
rca par	3.41	4.95	6.48	12.63	4.44	6.95	9.46	19.5
rca tmr	3.15	4.46	5.78	11.05	7.44	11.3	15.2	30.6
rca tmr2	4.22	6.28	8.33	16.63	8.1	12.6	17.1	35
csa	1.76	2.51	3.26	6.25	3.48	5.47	7.46	15.4
csa dup	3.06	4.23	5.42	10.21	8.9	13.9	18.9	38.8
csa par	3.1	4.47	5.84	11.38	5.75	9.12	12.5	11.83
csa tmr	2.22	2.97	3.72	6.79	11.2	17.5	23.8	49.2
cla	1.71	2.29	2.9	5.33	3.17	4.95	6.72	13.8
cla dup	3.03	4.2	5.39	10.18	8.28	12.8	17.4	35.4
cla par	2.92	4.11	5.3	10.12	5.19	8.18	11.2	23
cla tmr	2.11	2.7	3.31	5.81	10.3	15.9	21.6	44.3
fca	1.07	1.21	1.33	1.52	3.73	6.56	9.36	21.3
fca dup	3.08	4.26	5.44	10.28	9.39	16	22.6	50.3
fca tmr	1.48	1.61	1.74	1.96	11.9	20.8	29.5	66.7
sd2	1.49	1.49	1.49	1.51	16.6	25.5	34.5	70.3
sd2 par	2.52	2.53	2.55	2.56	26	34.9	43.9	79.7
sd3	1.83	1.83	1.84	1.86	9.77	15.1	20.4	41.8
sd3 1o3	3.81	4.99	6.17	10.98	13.7	21.4	29	59.8
booth	6.33	9.47	12.56	25.46	19.5	42.8	76.3	397
booth par	7.79	11.26	14.73	29.18	32.1	68.8	123	846
fcm	3.32	4.09	4.47	5.33	18.7	42.4	70.5	256
fcm dup	8.84	8.5	11.6	21.4	39.9	85.5	160	554

Table B.3: Synthesis results for filter taps

	area (μm^2)	delay (ns)	power (μw)
cla	4740	7.15	32.9
cla dup	9929	7.57	68.2
cla par	7373	9.01	51.3
sd2	6118	7.18	41.1
sd2 par	10094	9.41	69.5
sd2 full	15399	6.83	172
sd2 fcm	6074	4.32	39.3
sd3	6895	7.49	47.2
sd3 1o3	11091	10.52	76.5
sd3 fcm	6850	4.66	45
fca	4999	7.05	35.6
fca fcm	4276	3.43	27.6
fca fcm dup	8940	3.43	57.5

Table B.4: Synthesis results for parallel filters

	area (μm^2)	delay (ns)	power (μw)
cla	81285	10.09	574
cla dup	163142	10.11	1185
cla par	130555	12.38	924
sd2	104882	9.42	727
sd2 par	169289	11.84	1191
sd3	120278	9.62	863
sd3 1o3	191852	11.81	1379

Table B.5: Synthesis results for sequential filters

	area (μm^2)	delay (ns)	power (μw)
cla	18173	8.26	133
cla dup	36919	8.27	268
cla par	24019	9.15	174
sd2	21919	8	171
sd2 par	29120	8.85	223
sd3	23666	7.55	181
sd3 1o3	31821	8.42	240

References

[1] E. Mollick. Establishing moore's law. *Annals of the History of Computing, IEEE*, 28(3):62–75, July-Sept. 2006.

[2] Denis Teixeira Franco, Jean-François Naviner, and Lirida Naviner. Yield and reliability issues in nanoelectronic technologies. *Annales des télécommunications*, 61(11-12):1422–1457, Nov.-Dec. 2006.

[3] Premkishore Shivakumar, Michael Kistler, Stephen W. Keckler, Doug Burger, and Lorenzo Alvisi. Modeling the effect of technology trends on the soft error rate of combinational logic. In *DSN '02: Proceedings of the 2002 International Conference on Dependable Systems and Networks*, pages 389–398, Washington, DC, USA, 2002. IEEE Computer Society.

[4] N. Seifert, P. Slankard, M. Kirsch, B. Narasimham, V. Zia, C. Brookreson, A. Vo, S. Mitra, B. Gill, and J. Maiz. Radiation-induced soft error rates of advanced cmos bulk devices. *Reliability Physics Symposium Proceedings, 2006. 44th Annual., IEEE International*, pages 217–225, March 2006.

[5] Laszlo B. Kish. End of moore's law: Thermal (noise) death of integration in micro and nano electronics. *Physics Letters A*, 305(3-4):144–149, Dec. 2002.

[6] M. Nicolaidis. Carry checking/parity prediction adders and alus. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 11(1):121–128, Feb 2003.

[7] Michael Nicolaidis and Ricardo O. Duarte. Fault-secure parity prediction booth multipliers. *IEEE Design and Test of Computers*, 16(3):90–101, 1999.

[8] D. Marienfeld, E.S. Sogomonyan, V. Ocheretnij, and M. Gossel. New self-checking output-duplicated booth multiplier with high fault coverage for soft errors. *Test Symposium, 2005. Proceedings. 14th Asian*, pages 76–81, Dec. 2005.

[9] S. Mitra and E.J. McCluskey. Which concurrent error detection scheme to choose ? *Test Conference, 2000. Proceedings. International*, pages 985–994, 2000.

[10] Parag K. Lala, editor. *Self-checking and fault-tolerant digital design.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

[11] K. Mohanram and N.A. Touba. Cost-effective approach for reducing soft error failure rate in logic circuits. *Test Conference, 2003. Proceedings. ITC 2003. International*, 1:893–901, Sept. 30-Oct. 2, 2003.

[12] Quming Zhou and K. Mohanram. Gate sizing to radiation harden combinational logic. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 25(1):155–166, Jan. 2006.

[13] A.K. Nieuwland, S. Jasarevic, and G. Jerin. Combinational logic soft error analysis and protection. *On-Line Testing Symposium, 2006. IOLTS 2006. 12th IEEE International*, pages 6 pp.–, 10-12 July 2006.

[14] B. D. Sierawski, B. L. Bhuva, and L. W. Massengill. Reducing soft error rate in logic circuits through approximate logic functions. *Nuclear Science, IEEE Transactions on*, 53(6):3417–3421, Dec. 2006.

[15] Alessandro Birolini. *Quality and Reliability of Technical Systems: Theory-Practice-Management*. Springer-Verlag, Germany, 1994.

[16] Ketan N. Patel, Igor L. Markov, and John P. Hayes. Evaluating circuit reliability under probabilistic gate-level fault models. *Proceeding of the Twelfth International Workshop on Logic and Synthesis (IWLS 2003)*, 1:59–64, May 2003.

[17] S. Krishnaswamy, G.F. Viamontes, I.L. Markov, and J.P. Hayes. Accurate reliability evaluation and enhancement via probabilistic transfer matrices. *Design, Automation and Test in Europe, 2005. Proceedings*, pages 282–287 Vol. 1, March 2005.

[18] S. Krishnaswamy, I.L. Markov, and J.P. Hayes. Logic circuit testing for transient faults. *Test Symposium, 2005. European*, pages 102–107, May 2005.

[19] Maí Correia Vasconcelos, Denis Teixeira Franco, Lirida Naviner, and Jean-François Naviner. Reliability analysis of combinational circuits based on a probabilistic binomial model. *IEEE Northeast Workshop on Circuits and Systems, 2008. NEWCAS-TAISA 2008*, pages 310–313, June 2008.

[20] Denis Teixeira Franco, Maí Correia, Lirida Naviner, and Jean-François Naviner. Reliability of logic circuits under multiple simultaneous faults. *Accepted in the IEEE International MWSCAS 2008*, Aug. 2008.

[21] Denis Teixeira Franco, Maí Correia, Lirida Naviner, and Jean-François Naviner. Signal probability for reliability evaluation of logic circuits. *Accepted in ESREF 2008*, Sep.-Oct. 2008.

[22] K.P. Parker and E.J. McCluskey. Analysis of logic circuits with faults using input signal probabilities. *Computers, IEEE Transactions on*, C-24(5):573–578, May 1975.

[23] S. Ercolani, M. Favalli, M. Damiani, P. Olivo, and B. Ricco. Estimate of signal probability in combinational logic networks. *European Test Conference, 1989., Proceedings of the 1st*, pages 132–138, 12-14 Apr 1989.

[24] International Technology Roadmap for Semiconductors. Itrs report 2007, executive summary. ITRS Website, Dec. 2007. http://www.itrs.net/Links/2007ITRS/ExecSum2007.pdf.

[25] Ministère délégué à la Recherche et aux Nouvelles Technologies. A la découverte du nanomonde. http://www.nanomicro.recherche.gouv.fr/, 2003.

[26] J. Gea-Banacloche and L.B. Kish. Future directions in electronic computing and information processing. *Proceedings of the IEEE*, 93(10):1858–1863, Oct. 2005.

[27] M.R. Stan, P.D. Franzon, S.C. Goldstein, J.C. Lach, and M.M. Ziegler. Molecular electronics: from devices and interconnect to circuits and architecture. *Proceedings of the IEEE*, 91(11):1940–1957, Nov 2003.

[28] M.A. Breuer, S.K. Gupta, and T.M. Mak. Defect and error tolerance in the presence of massive numbers of defects. *Design & Test of Computers, IEEE*, 21(3):216–227, May-June 2004.

[29] Lorena Anghel. *Les Limites technologiques du Silicium et Tolerance aux Fautes*. Phd thesis, Laboratoire Techniques de l'Informatique et de la Microélectronique pour l'Architecture de l'Ordinateur (TIMA), Dec. 2000.

[30] R. Baumann. The impact of technology scaling on soft error rate performance and limits to the efficacy of error correction. *Electron Devices Meeting, 2002. IEDM '02. Digest. International*, pages 329–332, 2002.

[31] T. Skotnicki, J.A. Hutchby, Tsu-Jae King, H.-S.P. Wong, and F. Boeuf. The end of cmos scaling: toward the introduction of new materials and structural changes to improve mosfet performance. *Circuits and Devices Magazine, IEEE*, 21(1):16–26, Jan.-Feb. 2005.

[32] G. Bourianoff. The future of nanocomputing. *Computer*, 36(8):44–53, Aug. 2003.

[33] D. Edenfeld, A.B. Kahng, M. Rodgers, and Y. Zorian. 2003 technology roadmap for semiconductors. *Computer*, 37(1):47–56, Jan. 2004.

[34] Algirdas Avizienis, Jean-Claude Laprie, and Brian Randell. Fundamental concepts of dependability, April 2001.

[35] Xiaojun Li. *Deep Submicron CMOS VLSI Circuit Reliability Modeling, Simulation and Design*. Phd thesis, Faculty of the Graduate School of the University of Maryland, College Park, Nov. 2005.

[36] M.H. Woods. Mos vlsi reliability and yield trends. *Proceedings of the IEEE*, 74(12):1715–1729, Dec. 1986.

[37] Naran Sirisantana, B.C. Paul, and Kaushik Roy. Enhancing yield at the end of the technology roadmap. *Design & Test of Computers, IEEE*, 21(6):563–571, Nov.-Dec. 2004.

[38] Y. Zorian, D. Gizopoulos, C. Vandenberg, and P. Magarshack. Guest editors' introduction: Design for yield and reliability. *Design & Test of Computers, IEEE*, 21(3):177–182, May-June 2004.

[39] Michael Sydow. Compare logic-array to asic-chip cost per good die. Chip Design Magazine. http://www.chipdesignmag.com/.

[40] Marc Levitt. The role of design in enhancing nanometer process yield. International Engineering Consortium Newsletter, Jan. 2006. http://www.iec.org/newsletter/jan06_1/design_1.html.

[41] Mark Miller. Manufacturing-aware design helps boost ic yield. EE Times website. http://www.eetimes.com/.

[42] Jack HORGAN. Design for manufacturability (dfm). EDA Café website. http://www.edacafe.com/.

[43] International Technology Roadmap for Semiconductors. Itrs report 2005, executive summary. ITRS Website, Dec. 2005. http://www.itrs.net/Links/2005ITRS/ExecSum2005.pdf.

[44] Y. Zorian. Nanoscale design & test challenges. *Computer*, 38(2):36–39, Feb. 2005.

[45] C. Constantinescu. Trends and challenges in vlsi circuit reliability. *Micro, IEEE*, 23(4):14–19, July-Aug. 2003.

[46] R. Baumann. Soft errors in advanced computer systems. *Design & Test of Computers, IEEE*, 22(3):258–266, May-June 2005.

[47] P. Hazucha and C. Svensson. Impact of cmos technology scaling on the atmospheric neutron soft error rate. *Nuclear Science, IEEE Transactions on*, 47(6):2586–2594, Dec 2000.

[48] Carlos A. Lang Lisbôa, Luigi Carro, and Erika Cota. Robops - arithmetic operators for future technologies. *Informal Proceedings of 10th IEEE European Test Symposium*, 1, May 2005.

[49] M. Spica and T.M. Mak. Do we need anything more than single bit error correction (ecc)? *Memory Technology, Design and Testing, 2004. Records of the 2004 International Workshop on*, pages 111–116, Aug. 2004.

[50] M. Nicolaidis. Design for soft error mitigation. *Device and Materials Reliability, IEEE Transactions on*, 5(3):405–418, Sept. 2005.

[51] Chen He, M.F. Jacome, and G. de Veciana. A reconfiguration-based defect-tolerant design paradigm for nanotechnologies. *Design & Test of Computers, IEEE*, 22(4):316–326, July-Aug. 2005.

[52] K. Nikolic, A. Sadek, and M. Forshaw. Architectures for reliable computing with unreliable nanodevices. *Nanotechnology, 2001. IEEE-NANO 2001. Proceedings of the 2001 1st IEEE Conference on*, pages 254–259, 2001.

[53] P.K. Lala and A. Walker. On-line error detectable carry-free adder design. *Defect and Fault Tolerance in VLSI Systems, 2001. Proceedings. 2001 IEEE International Symposium on*, pages 66–71, 2001.

[54] B. Parhami. Approach to the design of parity-checked arithmetic circuits. *Signals, Systems and Computers, 2002. Conference Record of the Thirty-Sixth Asilomar Conference on*, 2:1084–1088 vol.2, Nov. 2002.

[55] G.C. Cardarilli, M. Ottavi, S. Pontarelli, M. Re, and A. Salsano. A signed digit adder with error correction and graceful degradation capabilities. *On-Line Testing Symposium, 2004. IOLTS 2004. Proceedings. 10th IEEE International*, pages 141–146, July 2004.

[56] Jie Han and P. Jonker. A system architecture solution for unreliable nanoelectronic devices. *Nanotechnology, IEEE Transactions on*, 1(4):201–208, Dec 2002.

[57] A.J. KleinOsowski, K. KleinOsowski, V. Rangarajan, P. Ranganath, and D.J. Lilja. The recursive nanobox processor grid: a reliable system architecture for unreliable nanotechnology devices. *Dependable Systems and Networks, 2004 International Conference on*, pages 167–176, June-1 July 2004.

[58] F.G. de Lima Kastensmidt, G. Neuberger, R.F. Hentschke, L. Carro, and R. Reis. Designing fault-tolerant techniques for sram-based fpgas. *Design & Test of Computers, IEEE*, 21(6):552–562, Nov.-Dec. 2004.

[59] S. Almukhaizim and Y. Makris. Fault tolerant design of combinational and sequential logic based on a parity check code. *Defect and Fault Tolerance in VLSI Systems, 2003. Proceedings. 18th IEEE International Symposium on*, pages 563–570, Nov. 2003.

[60] C.A.L. Lisbôa and L. Carro. Arithmetic operators robust to multiple simultaneous upsets. *Defect and Fault Tolerance in VLSI Systems, 2004. DFT 2004. Proceedings. 19th IEEE International Symposium on*, pages 289–297, Oct. 2004.

[61] E. Schüler and L. Carro. Increasing fault tolerance to multiple upsets using digital sigma-delta modulators. *On-Line Testing Symposium, 2005. IOLTS 2005. 11th IEEE International*, pages 255–259, July 2005.

[62] C. A. L. Lisbôa, E. Schüler, and Luigi Carro. Going beyond tmr for protection against multiple faults. *Integrated Circuits and Systems Design, 18th Symposium on*, pages 80–85, Sept. 2005.

[63] A. Schmid and Y. Leblebici. Robust circuit and system design methodologies for nanometer-scale devices and single-electron transistors. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 12(11):1156–1166, Nov. 2004.

[64] F. Peper, Jia Lee, F. Abo, T. Isokawa, S. Adachi, N. Matsui, and S. Mashiko. Fault-tolerance in nanocomputers: a cellular array approach. *Nanotechnology, IEEE Transactions on*, 3(1):187–201, March 2004.

[65] C. Lazzari, L. Anghel, and R.A.L. Reis. On implementing a soft error hardening technique by using an automatic layout generator: case study. *On-Line Testing Symposium, 2005. IOLTS 2005. 11th IEEE International*, pages 29–34, July 2005.

[66] Ketan N. Patel and Igor L. Markov. Error-correction and crosstalk avoidance in dsm busses. *IEEE Trans. Very Large Scale Integr. Syst.*, 12(10):1076–1080, 2004.

[67] P. Chan, G.A. Jullien, L. Imbert, V.S. Dimitrov, and G.H. McGibney. Fault-tolerant computation within complex fir filters. *Signal Processing Systems, 2004. SIPS 2004. IEEE Workshop on*, pages 316–320, Oct. 2004.

[68] D. Marculescu. Energy bounds for fault-tolerant nanoscale designs. *Design, Automation and Test in Europe, 2005. Proceedings*, pages 74–79 Vol. 1, March 2005.

[69] Y.S. Dhillon, A.U. Diril, A. Chatterjee, and A.D. Singh. Analysis and optimization of nanometer cmos circuits for soft-error tolerance. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 14(5):514–524, May 2006.

[70] S. Tosun, N. Mansouri, E. Arvas, M. Kandemir, and Yuan Xie. Reliability-centric high-level synthesis. *Design, Automation and Test in Europe, 2005. Proceedings*, pages 1258–1263 Vol. 2, March 2005.

[71] Manoj Sachdev and José Pineda de Gyvez. *Defect-Oriented Testing for Nano-Metric CMOS VLSI Circuits*. Springer, 2007.

[72] R.C. Ogus. The probability of a correct output from a combinational circuit. *Computers, IEEE Transactions on*, C-24(5):534–544, May 1975.

[73] S.P. Dokouzgiannis and J.M. Kontoleon. Exact reliability analysis of combinational logic circuits. *Reliability, IEEE Transactions on*, 37(5):493–500, Dec 1988.

[74] A. Bogliolo, M. Damiani, P. Olivo, and B. Ricco. Reliability evaluation of combinational logic circuits by symbolic simulation. *VLSI Test Symposium, 1995. Proceedings., 13th IEEE*, pages 235–242, Apr-3 May 1995.

[75] Martin Omana, Daniele Rossi, and Cecilia Metra. Model for transient fault susceptibility of combinational circuits: On-line testing (guest editors: Cecilia metra and matteo sonza reorda). *Journal of Electronic Testing*, 20:501–509(9), October 2004.

[76] Bin Zhang, Wei-Shen Wang, and M. Orshansky. Faser: fast analysis of soft error susceptibility for cell-based designs. *Quality Electronic Design, 2006. ISQED '06. 7th International Symposium on*, pages 6 pp.–, March 2006.

[77] Natasa Miskov-Zivanov and Diana Marculescu. Circuit reliability analysis using symbolic techniques. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 25(12):2638–2649, Dec. 2006.

[78] Rajeev R. Rao, Kaviraj Chopra, David T. Blaauw, and Dennis M. Sylvester. Computing the soft error rate of a combinational logic circuit using parameterized descriptors. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 26(3):468–479, March 2007.

[79] Smita Krishnaswamy, George F. Viamontes, Igor L. Markov, and John P. Hayes. Probabilistic transfer matrices in symbolic reliability analysis of logic circuits. *ACM Trans. Des. Autom. Electron. Syst.*, 13(1):1–35, 2008.

[80] T. Rejimon and S. Bhanja. An accurate probabilistic model for error detection. *VLSI Design, 2005. 18th International Conference on*, pages 717–722, Jan. 2005.

[81] T. Rejimon and S. Bhanja. Scalable probabilistic computing models using bayesian networks. *Circuits and Systems, 2005. 48th Midwest Symposium on*, pages 712–715 Vol. 1, Aug. 2005.

[82] T. Rejimon and S. Bhanja. Time and space efficient method for accurate computation of error detection probabilities in vlsi circuits. *Computers and Digital Techniques, IEE Proceedings* -, 152(5):679–685, Sept. 2005.

[83] Jie Han, E. Taylor, Jianbo Gao, and J. Fortes. Faults, error bounds and reliability of nanoelectronic circuits. *Application-Specific Systems, Architecture Processors, 2005. ASAP 2005. 16th IEEE International Conference on*, pages 247–253, July 2005.

[84] G. Asadi and M.B. Tahoori. An analytical approach for soft error rate estimation in digital circuits. *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, pages 2991–2994 Vol. 3, May 2005.

[85] Debayan Bhaduri, Sandeep Shukla, Paul Graham, and Maya Gokhale. Scalable techniques and tools for reliability analysis of large circuits. *VLSI Design, 2007. Held jointly with 6th International Conference on Embedded Systems., 20th International Conference on*, pages 705–710, Jan. 2007.

[86] Mihir R. Choudhury and Kartik Mohanram. Accurate and scalable reliability analysis of logic circuits. In *DATE '07: Proceedings of the conference on Design, automation and test in Europe*, pages 1454–1459, San Jose, CA, USA, 2007. EDA Consortium.

[87] M.P. Baze and S.P. Buchner. Attenuation of single event induced pulses in cmos combinational logic. *Nuclear Science, IEEE Transactions on*, 44(6):2217–2223, Dec 1997.

[88] V. L. Levin. Probability analysis of combination systems and their reliability. *Engin. Cybernetics*, (6):78–84, Nov.-Dec. 1964.

[89] B. Krishnamurthy and I.G. Tollis. Improved techniques for estimating signal probabilities. *Computers, IEEE Transactions on*, 38(7):1041–1045, Jul 1989.

[90] M.A. Al-Kharji and S.A. Al-Arian. A new heuristic algorithm for estimating signal and detection probabilities. *VLSI, 1997. Proceedings. Seventh Great Lakes Symposium on*, pages 26–31, Mar 1997.

[91] K.P. Parker and E.J. McCluskey. Probabilistic treatment of general combinational networks. *Computers, IEEE Transactions on*, C-24(6):668–670, June 1975.

[92] F. Brglez. On testability analysis of combinational networks. *Circuits and systems, IEEE International symposium on*, pages 221–225, May 1984.

[93] S.C. Seth, L. Pan, and V.D. Agrawal. Predict-probabilistic estimation of digital circuit testability. *Fault-Tolerant Computation Symposium (FTCS-15), Dig. Papers of*, pages 220–225, June 1985.

[94] J. Savir, G.S. Ditlow, and P.H. Bardell. On random pattern test length. *Computers, IEEE Transactions on*, C-33(1):79–90, January 1984.

[95] S.K. Jain and V.D. Agrawal. Statistical fault analysis. *Design & Test of Computers, IEEE*, 2(1):38–44, Feb. 1985.

[96] Quming Zhou and K. Mohanram. Transistor sizing for radiation hardening. *Reliability Physics Symposium Proceedings, 2004. 42nd Annual. 2004 IEEE International*, pages 310–315, April 2004.

[97] D.T. Franco, J.-F. Naviner, and L. Naviner. Convolution blocks based on self-checking operators. *Mixed Design of Integrated Circuits and Systems, 2007. MIXDES '07. 14th International Conference on*, pages 487–491, June 2007.

[98] L.A.B. Naviner, M.C.R. de Vasconcelos, D.T. Franco, and J.-F. Naviner. Efficient computation of logic circuits reliability based on probabilistic transfer matrix. *Design and Technology of Integrated Systems in Nanoscale Era, 2008. DTIS 2008. 3rd International Conference on*, pages 1–4, March 2008.

[99] Denis Teixeira Franco, Maí Correia, Lirida Naviner, and Jean-François Naviner. Reliability analysis of logic circuits based on signal probability. *Accepted in ICECS 2008*, Aug.-Sep. 2008.

[100] Maí Correia Vasconcelos, Denis Teixeira Franco, Lirida Naviner, and Jean-François Naviner. On the output events in concurrent error detection schemes. *Accepted in ICECS 2008*, Aug.-Sep. 2008.

[101] Maí Correia Vasconcelos, Denis Teixeira Franco, Lirida Naviner, and Jean-François Naviner. Relevant metrics for evaluation of concurrent error detection schemes. *Accepted in ESREF 2008*, Sep.-Oct. 2008.

[102] Yuan Taur, D.A. Buchanan, Wei Chen, D.J. Frank, K.E. Ismail, Shih-Hsien Lo, G.A. Sai-Halasz, R.G. Viswanathan, H.-J.C. Wann, S.J. Wind, and Hon-Sum Wong. Cmos scaling into the nanometer regime. *Proceedings of the IEEE*, 85(4):486–504, Apr 1997.

[103] P.A. Gargini. The global route to future semiconductor technology. *Circuits and Devices Magazine, IEEE*, 18(2):13–17, Mar 2002.

[104] A. Allan, D. Edenfeld, Jr. Joyner, W.H., A.B. Kahng, M. Rodgers, and Y. Zorian. 2001 technology roadmap for semiconductors. *Computer*, 35(1):42–53, Jan 2002.

[105] Microelectronics Advanced Research Initiative MELARI NANO. Technology roadmap for nanoelectronics - edition 1999. Microelectronics Advanced Research Initiative - MELARI NANO, 1999. ftp://ftp.cordis.lu/pub/esprit/docs/melnarm.pdf.

[106] J.A. Hutchby, G.I. Bourianoff, V.V. Zhirnov, and J.E. Brewer. Extending the road beyond cmos. *Circuits and Devices Magazine, IEEE*, 18(2):28–41, Mar 2002.

[107] K.B.K. Teo, R.G. Lacerda, M.H. Yang, A.S. Teh, L.A.W. Robinson, S.H. Dalal, N.L. Rupesinghe, M. Chhowalla, S.B. Lee, D.A. Jefferson, D.G. Hasko, G.A.J. Amaratunga, W.L. Milne, P. Legagneux, L. Gangloff, E. Minoux, J.P. Schnell, and D. Pribat. Carbon nanotube technology for solid state and vacuum electronics. *Circuits, Devices and Systems, IEE Proceedings* -, 151(5):443–451, Oct. 2004.

[108] A. DeHon and H. Naeimi. Seven strategies for tolerating highly defective fabrication. *Design & Test of Computers, IEEE*, 22(4):306–315, July-Aug. 2005.

[109] V.V. Zhirnov, J.A. Hutchby, G.I. Bourianoffls, and J.E. Brewer. Emerging research logic devices. *Circuits and Devices Magazine, IEEE*, 21(3):37–46, May-June 2005.

[110] M.T. Bohr. Nanotechnology goals and challenges for electronic applications. *Nanotechnology, IEEE Transactions on*, 1(1):56–62, Mar 2002.

[111] V.V. Zhirnov, J.A. Hutchby, G.I. Bourianoff, and J.E. Brewer. Emerging research memory and logic technologies. *Circuits and Devices Magazine, IEEE*, 21(3):47–51, May-June 2005.

[112] James R. Heath, Philip J. Kuekes, Gregory S. Snider, and R. Stanley Williams. A defect-tolerant computer architecture: Opportunities for nanotechnology. *Science*, 280(5370):1716–1721, 1998.

[113] S. Copen Goldstein and M. Budiu. Nanofabrics: spatial computing using molecular electronics. *Computer Architecture, 2001. Proceedings. 28th Annual International Symposium on*, pages 178–189, 2001.

[114] M. Mishra and S.C. Goldstein. Defect tolerance at the end of the roadmap. *Test Conference, 2003. Proceedings. ITC 2003. International*, 1:1201–1210, 30-Oct. 2, 2003.

[115] M.M. Ziegler and M.R. Stan. Cmos/nano co-design for crossbar-based molecular electronic systems. *Nanotechnology, IEEE Transactions on*, 2(4):217–230, Dec. 2003.

[116] T. Hogg and G.S. Snider. Defect-tolerant adder circuits with nanoscale crossbars. *Nanotechnology, IEEE Transactions on*, 5(2):97–100, March 2006.

[117] Myung-Hyun Lee, Young Kwan Kim, and Yoon-Hwa Choi. A defect-tolerant memory architecture for molecular electronics. *Nanotechnology, IEEE Transactions on*, 3(1):152–157, March 2004.

[118] J.M. Tour, W.L. Van Zandt, C.P. Husband, S.M. Husband, L.S. Wilson, P.D. Franzon, and D.P. Nackashi. Nanocell logic gates for molecular computing. *Nanotechnology, IEEE Transactions on*, 1(2):100–109, Jun 2002.

[119] C.P. Husband, S.M. Husband, J.S. Daniels, and J.M. Tour. Logic and memory with nanocell circuits. *Electron Devices, IEEE Transactions on*, 50(9):1865–1875, Sept. 2003.

[120] K.K. Likharev. Neuromorphic cmol circuits. *Nanotechnology, 2003. IEEE-NANO 2003. 2003 Third IEEE Conference on*, 1:339–342 vol.2, Aug. 2003.

[121] O. Turel, Jung Hoon Lee, Xiaolong Ma, and K.K. Likharev. Nanoelectronic neuromorphic networks (crossnets): new results. *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, 1:–394, July 2004.

[122] Subhasish Mitra, W.-J. Huang, N.R. Saxena, S.-Y. Yu, and E.J. McCluskey. Reconfigurable architecture for autonomous self-repair. *Design & Test of Computers, IEEE*, 21(3):228–240, May-June 2004.

[123] M.A. Breuer. Intelligible test techniques to support error-tolerance. *Test Symposium, 2004. 13th Asian*, pages 386–393, Nov. 2004.

[124] S.C. Goldstein. The impact of the nanoscale on computing systems. *Computer-Aided Design, 2005. ICCAD-2005. IEEE/ACM International Conference on*, pages 655–661, Nov. 2005.

[125] D.P. Vasudevan and P.K. Lala. A technique for modular design of self-checking carry-select adder. *Defect and Fault Tolerance in VLSI Systems, 2005. DFT 2005. 20th IEEE International Symposium on*, pages 325–333, Oct. 2005.

[126] G.C. Cardarilli, M. Ottavi, S. Pontarelli, M. Re, and A. Salsano. Error detection in signed digit arithmetic circuit with parity checker [adder example]. *Defect and Fault Tolerance in VLSI Systems, 2003. Proceedings. 18th IEEE International Symposium on*, pages 401–408, Nov. 2003.

[127] Anders Lindstrom, Michael Nordseth, and Lars Bengtsson. Vhdl library of nonstandard arithmetic units. *Technical Report 03-01, Departament of Computer Engineering, Chalmers University of Technology*, Aug. 2003.

[128] Anders Lindstrom, Michael Nordseth, and Lars Bengtsson. Contributions to residue-number-system/signed-digit arithmetic. http://www.ce.chalmers.se/research/proj/arithdb/files/doc/letter.pdf, 2003.

[129] W.J. Townsend, J.A. Abraham, and P.K. Lala. On-line error detecting constant delay adder. *On-Line Testing Symposium, 2003. IOLTS 2003. 9th IEEE*, pages 17–22, July 2003.

[130] D. Marienfeld, E.S. Sogomonyan, V. Ocheretnij, and M. Gossel. A new self-checking multiplier by use of a code-disjoint sum-bit duplicated adder. *Test Symposium, 2004. ETS 2004. Proceedings. Ninth IEEE European*, pages 30–35, May 2004.

[131] *Application-Specific Integrated Circuits*. Addison-Wesley, 1997.

[132] K. S. Papadomanolakis, T. Kakarountas, A. Tsoukalis, S. Nikolaidis, and C.E. Goutis. Cosafe, low power hardware-software co-design for safety-critical applications. Project Report, University of Patras, Sep. 1999.

[133] Issam AlZaher Noufal. *Outils de CAO pour la Génération d'Opérateurs Arithmétiques Auto-Contrôlables*. Phd thesis, Laboratoire Techniques de l'Informatique et de la Microélectronique pour l'Architecture de l'Ordinateur (TIMA), May 2001.

[134] J.Q. Wang and P.K. Lala. Partially strongly fault secure and partially strongly code disjoint 1-out-of-3 code checker. *Computers, IEEE Transactions on*, 43(10):1238–1240, Oct 1994.

[135] Antonis Paschalis, Nikos Gaitanis, Dimitris Gizopoulos, and Panagiotis Kostarakis. A totally self-checking 1-out-of-3 code error indicator. In *EDTC '97: Proceedings of the 1997 European conference on Design and Test*, page 450, Washington, DC, USA, 1997. IEEE Computer Society.

[136] Michael Wirthlin, Nathan Rollins, Michael Caffrey, and Paul Graham. Hardness by design techniques for field programmable gate arrays. *Proceedings of the 11th Annual NASA Symposium on VLSI Design*, 1.

[137] Paulo Sergio Ramirez Diniz, Eduardo Antonio Barros da Silva, and Sergio Lima Netto. *Processamento Digital de Sinais*. Bookman, Brazil, 2004.

[138] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. Analog Devices, USA, 1999.

[139] X. Wendling, H. Chauvet, L. Reveret, R. Rochet, and R. Leveugle. Automatic and optimized synthesis of dataparts with fault detection or tolerance capabilities. *Defect and Fault Tolerance in VLSI Systems, 1997. Proceedings., 1997 IEEE International Symposium on*, pages 195–203, Oct 1997.