Modeling, design and characterization of delay-chains based true random number generator

Molka Ben Romdhane

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2014-ENST-0055

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « Electronique et Communication »

présentée et soutenue publiquement par

Molka BEN ROMDHANE

le 1er Octobre 2014

Modélisation, implémentation et caractérisation de circuits générateurs de nombres aléatoires vrais utilisant des chaînes à retards

Directeur de thèse : **Jean-Luc DANGER, Professeur, Télécom ParisTech**
Co-encadrement de la thèse : **Tarik GRABA, Maître de Conférences, Télécom ParisTech**

Jury

M. Viktor FISCHER	Rapporteur, Professeur, Université Jean Monnet Saint Etienne
M. Olivier SENTIEYS	Rapporteur, Professeur, Université de Rennes I
M. Yannick TEGLIA	Examinateur, Expert en sécurité, ST Microelectronics
M. Guy GOGNIAT	Examinateur, Professeur, Lab-STICC, Université de Bretagne Sud
M. Philippe NGUYEN	Examinateur, Secure-IC
M. Sylvain GUILLEY	Invité, Professeur associé, Télécom ParisTech

T H È S E

TELECOM ParisTech
école de l'Institut Mines-Télécom - membre de ParisTech
46 rue Barrault 75013 Paris - (+33) 1 45 81 77 77 - www.telecom-paristech.fr

"Coming together is a beginning. Keeping together is progress. Working together is success."
Henry Ford

"God does not play dice with the universe."
Albert Einstein

"He who hesitates is lost."
David J. Kinniment

Remerciements

Les travaux que nous présentons dans ce manuscrit de thèse ont été effectués dans le cadre de la préparation d'un diplôme de Doctorat en collaboration entre Télécom ParisTech au sein du laboratoire COMELEC (Communications et Électronique) et l'entreprise Secure-IC basée à Rennes qui développe des solutions de sécurisation des nouvelles générations de cartes à puce. Les travaux de recherche réalisés au cours de cette thèse ont été menés au sein du groupe de recherche SEN (Systèmes Électroniques Numériques).

Je remercie tout particulièrement Monsieur Jean-Luc Danger, Professeur à Télécom ParisTech et chef du groupe SEN, pour m'avoir accueillie dans son équipe, pour avoir accepté de diriger cette thèse et pour son soutien scientifique. Je tiens à témoigner ma gratitude également à Monsieur Tarik Graba, Maître de conférences à Télécom ParisTech, pour avoir accepté le co-encadrement de cette thèse, pour son soutien et sa participation considérable à la bonne évolution des travaux. Je le remercie spécialement pour son support scientifique ainsi que ses critiques constructives qui m'ont permis d'avancer. Qu'ils soient assurés de ma profonde gratitude et qu'ils acceptent mes remerciements pour avoir contribué à rendre cette expérience enrichissante aussi bien sur le plan professionnel que personnel. L'intérêt qu'ils ont apporté à ces travaux et leur confiance m'a beaucoup aidé à évoluer scientifiquement et techniquement. Je remercie aussi les directeurs de Secure-IC, Hassan Triqui, directeur général, et Philippe Nguyen, directeur technique, d'avoir accepté de financer cette thèse en 2011 surtout que l'entreprise en était à ses débuts.

Je tiens également à remercier Monsieur Viktor Fischer, Professeur à l'Université Jean Monnet de Saint Etienne, et Monsieur Olivier Sentieys, Professeur à l'université de Rennes I, pour avoir accepté de rapporter le présent travail. J'exprime aussi ma reconnaissance à Messieurs Yannick Teglia, expert en sécurité chez ST Microelectronics, et Guy Gogniat, Professeur à l'université Bretagne Sud, pour avoir accepté d'être examinateurs au sein de mon jury de thèse. Je remercie également Yves Mathieu, directeur d'études à Télécom ParisTech, pour son aide ainsi que le temps qu'il m'a accordé pour la supervision des travaux de conception des circuits ASIC qui ont permis de tester et valider ces travaux de thèse.

Mes remerciements s'adressent aussi à Monsieur Sylvain Guilley, Professeur associé à Télécom ParisTech, ainsi que Laurent Sauvage, enseignant chercheur à Télécom ParisTech et responsable du laboratoire sécurité du groupe SEN et Florent Lozac'h, ingénieur de recherche à COMELEC, pour leur soutien scientifique et leur support technique qui m'a permis de mener à bien ces travaux. Je tiens aussi à remercier tous les membres de l'équipe SEN et de l'équipe Threat Protection de Secure-IC plus particulièrement ma

chef d'équipe Karine Lorvellec et mes collègues Youssef Souissi, Thibault Porteboeuf, Cédric Murdica, Pierre Viland, Zouha Chérif Jouini, Pablo Rauzy, Annelie Heuser, Zakaria Najm et Shivam Bhasin. Je profite aussi pour remercier mon ancien encadrant industriel et chef de projet chez Secure-IC, Lionel Tchernatinsky ainsi que mon ancien collègue Sébastien Briais qui m'ont aidée pendant les tous débuts de cette thèse. Qu'ils trouvent tous dans ce travail le résultat de leurs conseils et encouragements.

Je tiens d'autre part à remercier spécialement Chadi Jabbour, enseignant chercheur et Hussein Fakhouri, ingénieur de recherche du groupe SIAM et Karim Benkalai pour leur disponibilité et le support matériel et scientifique qu'ils m'ont apporté. Ma reconnaissance s'adresse aussi au Professeur Patrick Loumeau, chef du groupe SIAM, et à Mesdames les Professeurs Lirida Naviner et Patricia Desgreys pour leur soutien, leur temps et leur présence au cours de la période passée à Télécom ParisTech. Je souhaite également faire part de mes remerciements envers l'équipe administrative du laboratoire COMELEC et de l'école doctorale EDITE pour leur efficacité et leur soutien. Je remercie particulièrement Bruno Thédrez, directeur du laboratoire COMELEC, Alain Sibille, directeur de la formation doctorale, Florence Besnard, Responsable de la scolarité de l'EDITE, ainsi que Zouina Sahnoune, Chantal Cadiat et Yvonne Bansimba, secrétaires gestionnaires comptabilité à COMELEC et finalement Bernard Cahen, directeur de la Maisel de Télécom ParisTech qui m'ont beaucoup facilité les démarches administratives tout au long de mon parcours doctoral.

Je profite aussi pour témoigner ma reconnaissance à mes amis au travail et dans la vie pour leurs conseils, leur soutien et leur présence très appréciés. Je cite particulièrement Emna Amouri, Arwa Ben Dhia, Imen Mansouri, Hassen Karray ainsi que Lobna Haouari. Enfin, et non le moins important, je remercie profondément mes parents Sadika et Noomane ainsi que ma sœur Manel pour m'avoir toujours soutenue et encouragée malgré la distance et les difficultés.

Résumé long en français

Les nombres aléatoires sont indispensables dans de nombreuses applications. Par exemple, ils sont requis pour la validation des calculs où des variables statistiques interviennent, comme la simulation de Monte-Carlo ou l'émulation de canaux de transmission numérique. Une autre application fondamentale est la cryptographie où l'aléa est utilisé dans certains protocoles de sécurité. Les générateurs de nombres aléatoires, plus connus sous le nom de RNG pour "Random Number Generator" se déclinent en deux familles, les PRNG (pseudo RNG) qui sont des générateurs de nombres aléatoires ayant des séquences déterministes et les TRNGs (True RNG) qui sont des générateurs d'aléa "vrai", donc non prévisibles.

Les applications cryptographiques utilisent à la fois les TRNGs et les PRNGs. Un PRNG nécessite une valeur initiale, ou graine, qui peut être la sortie d'un TRNG. Les TRNGs tirent profit de l'aléa de certain phénomènes physiques en technologies CMOS comme le bruit thermique causé par l'agitation des électrons dans un conducteur ou encore l'effet de charges piégées de manière aléatoire à la surface de l'oxyde de grille. L'aléa obtenu grâce à ce genre de phénomènes est ainsi plus facile à générer avec des technologies analogiques. Pour des raisons de portabilité et d'intégration, nous visons à étudier les TRNGs plutôt numériques.

De façon à évaluer la qualité entropique d'un TRNG, des standards basés sur des tests statistiques ont été élaborés par des organismes de certification comme le NIST ou la BSI. Cependant, il est recommandé de formaliser, par le biais d'un modèle, le caractère stochastique de la génération d'aléa. Dans la littérature, plusieurs architectures de TRNG ont déjà été proposées et certaines d'entre elles sont déjà utilisées dans des produits commerciaux. En outre, des méthodes ont été proposées pour évaluer la qualité des nombres aléatoires générés. Ces tests statistiques sont même utilisés dans la procédure de certification standard et y sont bien définis. Dans le chapitre 2, nous détaillons les méthodes d'évaluation de la qualité et de la robustesse d'un TRNG.

Les paramètres primordiaux aux quels un concepteur de TRNG doit prêter attention sont :

- Complexité en terme de ressources
- Débit
- Robustesse
- Auto-testabilité
- Possibilité d'évaluation au moment de conception

- Conformité aux tests standard statistiques

Un des principaux inconvénients des TRNGs existants est la non portabilité de leur architecture à travers différentes technologies. En effet, une fois conçu pour une technologie particulière, il est souvent impossible de les porter vers une nouvelle technologie sans une refonte complète. Parfois, l'extraction du phénomène physique est même impossible dans certaines technologies. Par exemple, les circuits FPGA (Field Programmable Gate Arrays) sont limités à des primitives logiques numériques contrairement aux circuits ASIC (Application Specific Integrated Circuit) où des primitives analogiques peuvent être trouvées lors de la conception. La plupart des TRNGs dans les technologies numériques, comme les FPGA, font appel à des oscillateurs en anneau en exploitant l'aléa induit par le phénomène de gigue d'horloge. Cependant, récemment, des attaques physiques contre ces TRNGs ont été menés. Ces attaques représentent une menace contre certains protocoles d'authentification, de chiffrement par blocs et de signature digitales dont la sécurité dépend de l'aléa généré par les TRNGs. Ces vulnérabilités concernent particulièrement les TRNGs à base d'oscillateurs en anneau. Ces derniers présentent en effet l'inconvénient de pouvoir être attaqués par couplage harmonique. Depuis lors, une nouvelle catégorie de TRNG, qui exploitent le phénomène de métastabilité se produisant dans les cellules CMOS bi-stables, a reçu une grande attention de la part des concepteurs de TRNG. Néanmoins, pour cette catégorie de TRNG un modèle mathématique, décrivant l'aléa généré et permettant d'évaluer sa qualité durant la conception, manque.

Dans cette thèse, nous adressons la problématique de génération d'aléa vrai dans des systèmes électroniques numériques. Dans un premier temps, nous étudions les différentes classes de TRNG ainsi que les méthodes qui permettent de les évaluer statistiquement. Puis, nous présentons les différentes techniques et phénomènes physiques étudiés dans la littérature pour générer de l'aléa vrai sur puce en technologie CMOS. Par la suite, nous présentons un état de l'art des menaces qui peuvent détériorer la qualité entropique d'un TRNG.

Nous étudions ensuite une architecture de TRNG, peu coûteuse et robuste face aux attaques par injection d'harmoniques car elle n'utilise pas d'oscillateurs. Ce TRNG extrait une variable aléatoire en exploitant à la fois les états métastables des bascules et les fluctuations temporelles (ou gigue) des signaux échantillonnés.

Nous proposons par la suite un modèle stochastique qui nous permet de décrire le comportement aléatoire du TRNG, et d'établir ainsi l'entropie théorique minimale qu'il peut fournir, indépendamment de la technologie ciblée. Les caractérisations et évaluations sur des circuits prototypes en technologies FPGA et ASIC montrent que l'architecture du TRNG proposée génère de l'aléa de qualité, est robuste face aux variations environnementales et utilise exclusivement des cellules numériques.

Les utilisations de nombres aléatoires en cryptographie sont diverses : génération de clés, génération de vecteurs d'initialisation en cryptographie symétrique, randomisation en cryptographie asymétrique. Contrairement à beaucoup d'applications qui sont moins exigeantes que les applications cryptographiques, l'aléa en cryptographie doit garantir plusieurs propriétés. En effet, pour éviter certain types d'attaques, les clés secrètes ou les graines doivent être non prédictibes, uniformément réparties et statistiquement indépendantes mais surtout et aussi le TRNG qui les génères ne doit pas être manipulables. Il est très important qu'un générateur de nombre aléatoire soit fiable et génère de l'aléa de bonne qualité c'est à dire avec une entropie élevée. L'échec de génération d'entropie ou la vulnérabilité du RNG à certaines attaques peut conduire à l'échec de tout le système cryptographique [NS02, NS03, LHA+12, MM09].

Pour la vérification des propriétés statistiques, il existe des batteries de tests statistiques qui permettent de tester la qualité de l'aléa. Il y a celles qui sont standardisées par des organismes comme le NIST ou la BSI et d'autres qui ne sont pas standardisées mais qui sont aussi utilisées par la communauté des concepteurs de TRNG comme DieHard. Pour valider les séquences générées pas notre TRNG, nous avons utilisé les batteries de tests statistiques AIS31 et NIST sp800-22.

Ces tests statistiques permettent de vérifier la bonne distribution statistique de l'aléa généré, mais aucun des tests existants ne garantit l'imprévisibilité d'une source de nombres aléatoires. Le caractère imprévisible de l'aléa cryptographique ne s'étudie pas sur la suite de bits générée mais sur le mécanisme de génération lui même. D'où le besoin de modéliser les TRNGs. En plus de ces propriétés statistiques, le circuit TRNG doit être non manipulable physiquement.

Dans le cadre de cette thèse, nous avons pour objectif d'étudier une nouvelle architecture qui vise à exploiter deux phénomènes physiques pour générer l'aléa qui sont d'une part la métastabilité et d'autre part la gigue d'horloge. Communément le phénomène de gigue d'horloge est exploité à travers l'utilisation d'oscillateurs en anneau. Les TRNGs utilisant des oscillateurs en anneau sont sensibles particulièrement aux attaques par injection de fréquence dont le principe consiste à verrouiller la fréquence de ces oscillateurs. Le couplage entre les oscillateurs en anneau et le signal oscillant injecté conduit à l'élimination de la variation de phase entre plusieurs oscillateurs tournant en parallèle. Donc le but de notre nouvelle structure de TRNG est d'exploiter la gigue d'horloge en évitant d'utiliser les oscillateurs en anneau.

Nous voulons plutôt exploiter des chaînes de bascules en boucle ouverte qui permettent de capturer en même temps un bruit de phase (gigue d'horloge) et un bruit d'amplitude. Dans ces travaux, nous adressons tout d'abord l'étude de la faisabilité de ce genre de structure en ASIC ensuite nous étudions sa faisabilité en technologie FPGA.

Aussi, d'une manière générale, il est recommandé de traiter l'aléa issu

d'une source aléatoire non maîtrisée pour en garantir les caractéristiques statistiques. Pour obtenir de l'aléa vrai sous forme numérique, il est communément indiqué d'utiliser les mécanismes suivants :

- une source physique d'aléa,
- un bloc de numérisation, des tests en ligne embarqués avec la puce du TRNG et
- finalement un retraitement algorithmique de l'aléa à la sortie du bloc de numérisation.

La figure 1 illustre la structure commune à tous les TRNGs.

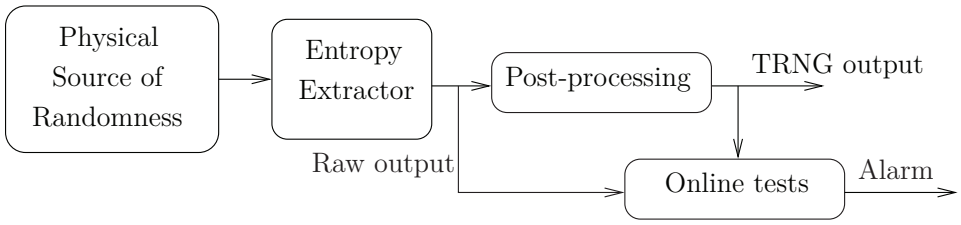


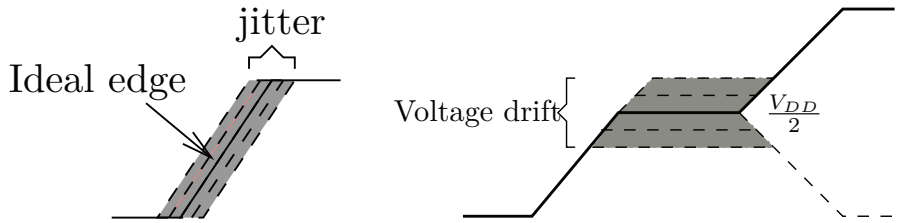
Figure 1 – Structure générique d'un TRNG

Dans les circuits numériques, l'extraction d'entropie se fait communément à travers l'échantillonnage d'un signal bruité. L'échantillonnage se fait à l'aide d'éléments mémorisant comme les bascules-D (D-flip-flops,DFF), des verrous (D-latches) ou les cellules de mémoire statique SRAM (Static Random Access Memory).

Pour générer de l'aléa dans les circuits numériques, une première méthode consiste à échantillonner un signal aléatoire à des instants réguliers. Ce cas est équivalent à échantillonner le signal aléatoire en utilisant une horloge idéale La deuxième méthode est d'échantillonner des signaux réguliers à des intervalles de temps aléatoires. La troisième voie consiste à échantillonner un signal bruyant avec une horloge bruitée.

Dans le circuit TRNG que nous proposons dans ce manuscrit, nous exploitons la dernière méthode qui tire profit de l'accumulation des deux types de bruit. Le bruit généré par les porteurs de charges dans les transistors constituant le circuit, influe d'une part le moment de l'échantillonnage et d'autre part l'amplitude des signaux [FAB+09]. La figure 2 illustre les deux types de bruit.

Le bruit sur le signal échantillonnant est communément appelé "gigue d'horloge", nous l'appellerons "bruit de phase" comme il arrive sur l'axe de temps autour du moment d'échantillonnage. Se rajoute à ce bruit, le "bruit d'amplitude" est quant à lui saisi autour d'un "état métastable" de l'élément



(a) Phase noise (b) Voltage noise

Figure 2 – Illustration des bruits de phase et d'amplitude sur le signal échantillonneur et le signal échantillonné

de mémoire au moment de l'échantillonnage. Dans la technologie CMOS, la tension de l'état métastable est autour de $\frac{V_{DD}}{2}$.

Le plan de ce manuscrit de thèse se répartit comme suit :

Une introduction générale traite du contexte des ces travaux, en d'autres termes des besoin des TRNGs surtout dans le domaine de la cryptographie. Ensuite, nous discutons les critères nécessaires en cryptographie pour concevoir un TRNG qui soit apte à fournir de l'aléa de bonne qualité et qui soit non manipulable physiquement. Dans cette introduction générale, nous présentons les motivations qui nous ont mené à proposer l'architecture ici traitée ainsi que les motivations de chacune des tâches entreprises durant cette thèse. Finalement, l'introduction générale résume nos contributions.

Par la suite, le manuscrit présente cinq chapitres. Les deux premiers présentent un état de l'art des circuits TRNGs existants dans la littérature scientifique ou encore des brevets industriels. Y sont expliqués leur fonctionnement général et l'origine de l'aléa.

Dans cette partie, nous proposons de classifier les circuits TRNGs existants suivant le critère de source de bruit. Une première catégorie est celle des TRNGs qui exploitent le bruit de phase qui est la plus répandue. La deuxième catégorie c'est celle qui exploite le bruit d'amplitude. Finalement, nous introduisons une nouvelle catégorie de TRNG qui exploitent la superposition de ces deux phénomènes.

Le deuxième chapitre est consacré aussi à la suite de l'état de l'art du point de vue de l'évaluation des TRNGs. Ce chapitre traite respectivement l'état de l'art de trois aspects importants vers la certification de TRNGs : l'évaluation statistique, la modélisation stochastique et l'évaluation physique.

Nous présentons les méthodes d'évaluation statistique et physique des TRNGs existants ainsi que les vulnérabilités identifiées dans la littérature. Afin d'évaluer la qualité de la l'aléa que génère les circuits TRNGs, des tests statistiques standards sont appliqués à la séquence générée. Par contre, évaluer un TRNG comprend non seulement l'évaluation statistique de la qualité de l'aléa, mais aussi sa capacité d'auto-test et sa robustesse face à la variation des conditions de fonctionnement qui peuvent être naturelles ou d'origine malveillante. Un TRNG devrait également être modélisé de point de vue stochastique pour permettre son évaluation au moment de la conception.

Dans la section 2.2, nous présentons les normes d'évaluation existantes en termes de qualité statistique. En outre, nous proposons une étude approfondie de la certification suivant la méthodologie d'évaluation AIS-31. Nous présentons dans la section 2.3 un bref état de l'art de la modélisation stochastique de TRNGs.

Troisièmement, la section 2.4 traite des vulnérabilités de TRNGs sur puce et présente l'état de l'art des attaques physiques sur les TRNGs décrites dans la littérature. Dans cette même section, nous présentons un état de l'art concis des méthodes de perturbations des générateurs de nombres aléatoires dans des conditions anormales d'utilisation permettant de biaiser l'aléa généré. C'est ce qui nous aidera par la suite à comparer en terme de robustesse, entre autres, les TRNGs existants.

Enfin, dans la section 2.5, nous présentons une comparaison entre les TRNGs existants sur puce en termes de complexité, de débit et vulnérabilités physiques. Cette étude comparative exhaustive des TRNGs existants prend comme paramètres de comparaison la robustesse physique face aux attaques par couplage, l'existence ou non d'un modèle stochastique et le débit de génération d'aléa. Cette étude comparative nous a amené à conclure qu'il n'existe, à ce jour, aucun TRNG qui soit en même temps robuste face aux attaques par couplage, qui présente une bonne qualité statistique d'aléa et qui ait un bon débit de fonctionnement tout en étant basé sur des cellules numériques. Ce manque dans l'état de l'art nous a donc amené à proposer une nouvelle architecture de TRNGs exploitant la superposition des deux phénomènes de bruit de phase et d'amplitude. Cette nouvelle architecture qu'on appelle TRNG à base de chaînes à retards fait l'objet des chapitres suivants.

Dans le chapitre 3, nous commençons par présenter les principes de fonctionnement de l'architecture proposés. Notamment la description de la source de bruit, l'explication des raisons pour lesquelles nous avons choisis d'utiliser des chaînes à retards. Dans ce même chapitre, nous présentons ensuite le modèle stochastique que nous introduisons dans cette thèse. Ce modèle nous permet en tant que concepteur de TRNGs d'évaluer et d'estimer l'aléa attendu avant sa mise en œuvre ou encore son envoi en fonderie s'il s'agit d'un circuit ASIC. Le concepteur d'un TRNG doit fournir un modèle sto-

chastique de la cible d'évaluation pour être en conformité avec le processus d'évaluation standard AIS-31.

Nous présentons le modèle d'un TRNG qui exploite le comportement aléatoire de la sortie d'une bascule D fonctionnant dans la région métastable. Cette incertitude est causée par le bruit omniprésent dans un circuit intégré CMOS. Un tel modèle doit être mis en place pour le processus d'évaluation de l'entropie minimale fournie par le TRNG. Dans ce travail, nous décrivons et analysons le caractère aléatoire provenant d'une chaîne de bascules D lorsque ces dernières opèrent près de l'état métastable. La principale valeur ajoutée de ce chapitre est le modèle permettant calcul de la probabilité en sortie du TRNG en fonction des paramètre du système.

L'idée de base des TRNGs existants est d'exploiter la superposition de différents types de bruit intrinsèques au silicium pour générer une sortie aléatoire. Nous proposons de forcer des D-latch à opérer dans la région métastable et de profiter des deux phénomènes exploitables qui sont la gigue d'horloge et la métastabilité. Supposons que les données changent dans la fenêtre interdite autour du front d'échantillonnage. Cela signifie que le bi-stable est forcé de fonctionner dans le mode métastable. Dans cette région métastable, le résultat de la cellule bi-stable est imprévisible et dépend de :

- L'incertitude autour de l'instant d'échantillonnage de la donnée. En fait, si le signal de données, D, passe avant le t_{setup} limite, la sortie du bi-stable, Q, permettrait de s'établir à V_{DD} sinon à 0V. Ce moment incertain de l'état de commutation des données est provoqué par la gigue d'horloge.

- Les fluctuations de la tension autour de $\frac{V_{DD}}{2}$. En fait, la résolution de l'état de la bi-stable dépend du décalage par rapport à $\frac{V_{DD}}{2}$. Si ce décalage est positive la sortie Q s'établira à V_{DD}. Si elle est négative Q s'établira à 0V.

On note l'incertitude de synchronisation caractérisé par la gigue δt et l'incertitude du décalage de tension par δv telles que représentées dans la figure 3.

Une zone de certitude, donnée par le constructeur, est garantie par le respect des temps de *setup* et de *hold*. Si on ne respecte pas cette zone de certitude la bascule entre dans la zone métastable. Dans cette zone, on ne peut plus garantir l'état final de la sortie de la bascule. Dans cette zone, l'état final du bi-stable va dépendre du bruit. Donc si on échantillonne la donnée au moment où elle change d'état, on ne peut pas prédire la sortie de la bascule.

Le défi est d'arriver à contrôler le délai entre donnée et horloge dans le but de faire opérer la bascule dans cette fenêtre de métastabilité. Une approche naïve serait de retarder le signal donnée de T_{setup0} (tel que défini

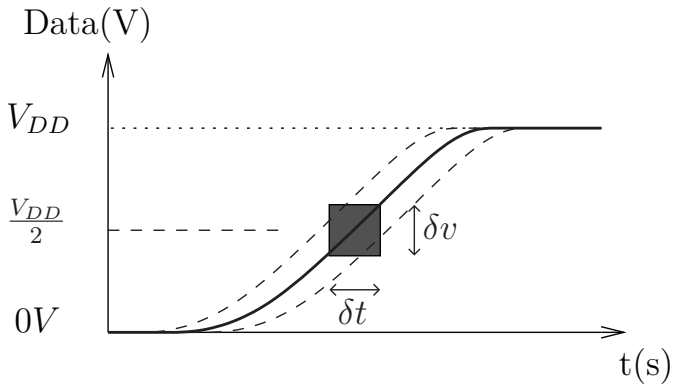


Figure 3 – Incertitude autour de $\frac{V_{DD}}{2}$

au paragraphe 3.2.2) à partir du front d'horloge avec un élément de retard contrôlable. La figure 4 illustre cette approche.

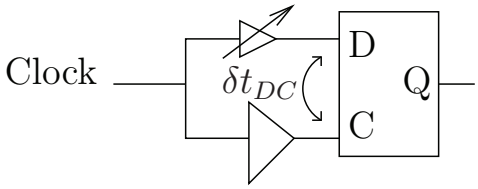


Figure 4 – Principe du TRNG à base de chaînes de retard

Ce type de contrôle et la précision nécessaire est impossible dans les FPGA, ni dans des cellules standard ASIC. Pour cela, nous proposons d'utiliser de multiples éléments de retard : une chaîne à retard sur le chemin de donnée et une autre chaîne à retard sur le chemin horloge avec une légère différence de retard et considérer le retard différentiel.

Cela permet d'atteindre une meilleure précision. En fait, plus la précision est élevée, plus l'imprédicitibilité de la sortie de la bascule est grande.

Dans les sous sections suivantes du chapitre, nous introduisons la structure des chaînes à retard. Par la suite, nous présentons le modèle stochastique du TRNG à base de chaînes à retard. La figure 5 illustre la structure générique du TRNG que nous introduisons.

Deux chaînes à délais fins sont connectées à une chaîne de bascules. Les temps de propagation de chacune de ces chaînes sont légèrement différents. Ceci nous permet d'avoir un délai différentiel très fin entre les entrées donnée et horloge de chacune des bascules constituant la chaîne. Cela nous permet de contrôler δt_{DC} avec une grande précision de sorte que le front du signal de donnée rattrape le front d'horloge tel que représenté dans la figure 6.

La longueur de la chaîne fine, c'est à dire le nombre d'éléments de délai,

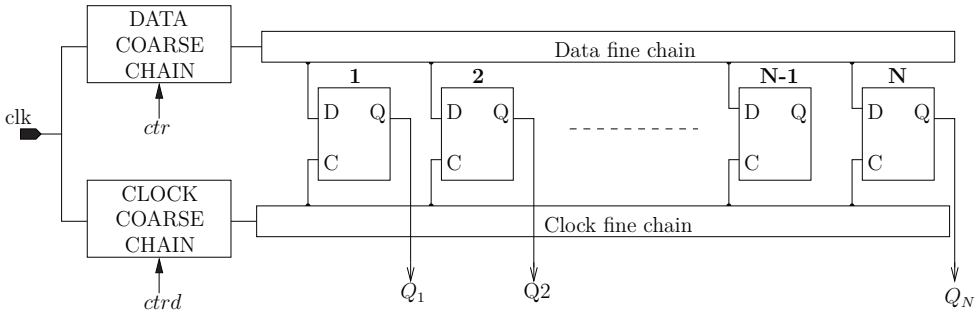


Figure 5 – Structure générique du TRNG à base de chaînes à retard

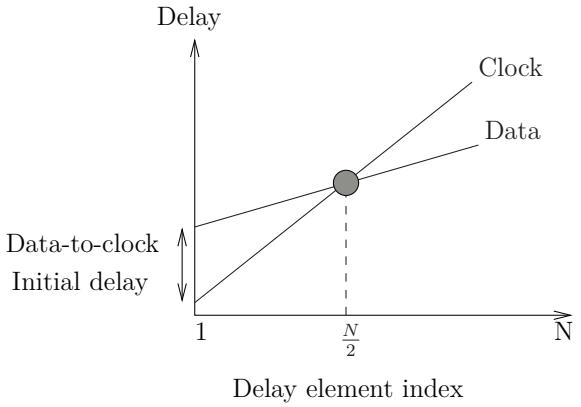


Figure 6 – Course entre donnée et horloge à travers la chaîne de latches

est égal au nombre de bascules. D'autre part, comme illustré dans la figure 5, nous utilisons également deux blocs de chaînes à retard grossier qui aident à calibrer le retard initial entre le signal d'horloge et de donnée pour compenser tout éventuel changement dans les conditions environnementales de fonctionnement.

Le retard dans la chaîne grossière est réglable en contrôlant les M multiplexeurs chaînés. Le nombre de multiplexeurs, M, en fonction du nombre de bascules, N, est donné dans l'équation (1). Cette équation est définie par l'hypothèse que les signaux donnée et horloge se rencontrent quelque part à l'entrée d'une des bascules du milieu de la chaîne.

$$\frac{M}{2}.dT_{mux} + \frac{N}{2}.\delta t_{DC} = T_{setup0} \tag{1}$$

La chaîne de retard grossier est utilisée pour ajuster les retards qui peuvent changer avec les conditions environnementales. La mise en œuvre de ce type de structure est difficile à réaliser en raison du :

- déséquilibre physique de routage,
- des asymétries entre les cellules.

Des retards entre les signaux donnée et horloge existent donc inévitablement. En fait, les variations technologiques entre les puces, et même entre les LUT et les matrices de commutation sur la même puce FPGA, peuvent conduire à une différence de retard en fonction de la région FPGA il est mis en œuvre.

La sortie du TRNG, *trngo*, est obtenue par addition modulo 2 de toutes les sorties des bascules comme indiqué dans 2.

$$trngo = \bigoplus_{i=1}^{N} Q_i \tag{2}$$

Ensuite, nous discutons des paramètres de conception qui influent sur la sortie du TRNG et établissons la formulation mathématique de la probabilité de la sortie brute, *trngo*, du TRNG. La sortie brute du circuit TRNG est représentée par un bit qui peut prendre la valeur logique 1 ou 0. L'entropie de la sortie brute du TRNG est exprimée dans l'équation (3) où p_1 et p_0 représente respectivement la probabilité d'obtenir 1 sur *trngo* et la probabilité d'obtenir 0.

$$H_{trngo} = -p_0 \cdot log_2(p_0) - p_1 \cdot log_2(p_1) \tag{3}$$

Dans ce qui suit, nous établirons une expression mathématique de la probabilité $p(trngo =' 1')$. Plus tard, nous essayons d'extraire les paramètres optimaux qui génèrent l'entropie maximale pour l'architecture du TRNG présenté plus haut. Pour cette question, nous devons identifier les paramètres qui influencent l'entropie du point de vue de la technologie et de la structure. Donc la suite de la section 3.4 du chapitre 3 nous présentons une analyse de l'état métastable de la bascule afin de modéliser le comportement aléatoire du TRNG.

L'expression de la probabilité que le TRNG génère un 1 est calculée en fonction de l'écart type du bruit, des caractéristiques du D-Latch utilisé et le retard différentiel δt des éléments des chaînes à retard utilisés. Les paramètres du modèle stochastique identifiés dans ce chapitre nous permettent de caractériser plus tard les paramètres technologiques qui influencent la qualité de l'aléa et la façon d'établir les spécifications de conception du TRNG.

Dans les chapitres 4 et 5 respectivement, le modèle présenté est validé sur une puce prototype en technologie CMOS 65nm et sur un FPGA. Le chapitre 4 est consacré à la mise en œuvre de l'ASIC TRNG auparavant présenté dans le chapitre 3. Il traite de la faisabilité de l'architecture générique du TRNG à chaines à retards avec la bibliothèque de cellules standards CMOS 65nm de ST Microelectronics.

Nous comparerons les résultats de la modélisation par rapport aux résultats de la simulation et des mesures sur le prototype TRNG ASIC. Habituellement, on commence par tester un circuit sur FPGA mais vu que nous avons voulu premièrement valider le modèle stochastique, il était plus intéressant de commencer par une conception en technologie ASIC puisque on a accès aux paramètres technologique, aux architectures des cellules standards, aux modèles des portes et nous avons la possibilités de faire des simulations en ajoutant du bruit transitoire intrinsèque aux éléments constitutifs.

Deux types de contrôles de retard pour la chaîne grossière ont été proposés. Le premier est d'utiliser une chaîne de multiplexeurs et la seconde méthode utilise des cellules amplificateurs trois états (tristate buffers).

La section 4.2 est dédié à la description de la structure ASIC et aux détails de conception du TRNG ASIC avec une chaîne grossière utilisant des multiplexeurs. Nous désignerons ce premier prototype ASIC_TRNG#1. Puis dans la section 4.3.1, nous présentons des tests statistiques standard NIST et AIS-31 effectuées sur ce premier prototype TRNG. En outre, nous présentons les résultats de mesure des conditions de travail des perturbations sur le TRNG à savoir alimentation des variations de tension et les variations de température. Enfin, dans la section 4.4, une amélioration des chaînes de retard grossières et fines est proposée. Nous nous référons à la mise en œuvre de ce second prototype ASIC ASIC_TRNG#2.

Pour mettre en œuvre la structure générique de la figure 5 en technologie ASIC, nous devons définir les spécifications des chaînes à retard grossier et chaînes à retard fin. La chaîne de retard fin devrait permettre un réglage très fin du délai entre les signaux horloge et donnée, de l'ordre de la picoseconde.

Dans ce chapitre nous traitons des contraintes de placement et routage des chaînes à retards ainsi que de l'ajustement des délais pour tirer profit de la gigue d'horloge et du bruit d'amplitude en zone métastable. Par la suite, nous présentons les résultats de simulations avec bruit et les comparaisons avec le modèle stochastique. Finalement nous présentons les résultats obtenus sur circuits et validons la qualité de l'aléa du TRNG avec les batteries de tests statistiques standards. Au sein du testchip ASIC_TRNG#1 lui même, nous proposons de concevoir différentes versions de circuits TRNG dans le but de :

- étudier l'impact de la précision des chaînes à délais sur l'entropie,
- comparer l'apport du bruit vertical dans deux cellules bi-stables différentes, la cellule standard verrou et un autre verrou personnalisé utilisant un multiplexeur rebouclé,
- étudier l'impact du placement et routage des chaînes à délais sur le retard entre donnée et horloge.

Notre méthodologie de conception du prototype ASIC TRNG se présente sous forme des étapes suivantes :

- D'abord, nous choisissons les éléments dans les cellules de la bibliothèque pour garantir l'apparence de métastabilité au milieu de la chaîne de délais fins.
- Ensuite, nous contraignons attentivement la mise en œuvre du Layout du TRNG et par la suite nous intégrons les différents versions du TRNG chacune comme un bloc pré-placé-routé dans un flot backend automatique.
- Nous réalisons des simulations post layout dans les conditions de fonctionnement typique, pire cas et meilleur cas, chacune en ajoutant le bruit transitoire.
- Nous confrontons le modèle stochastique proposé dans le chapitre 3 aux résultats de simulation.
- Nous comparons les résultats de simulation et du modèle stochastique aux mesure sur puce.
- Enfin, nous appliquons les tests standards statistiques pour évaluer le caractère aléatoire du TRNG ASIC.

Nous présentons la mesure de l'impact de la température, la tension d'alimentation et les variations de processus sur le premier circuit ASIC_TRNG#1. Les résultats des mesures sur le prototype TRNG sont conformes aux résultats théoriques et de simulation. Les résultats des tests statistiques et l'analyse et comparaison des 4 versions TRNG au sein du prototype ASIC_TRNG#1 se trouvent dans les sections 4.3.3 et 4.2.4.

Après la caractérisation et l'évaluation de l'ASIC_TRNG#1, nous proposons une amélioration des chaînes de retard du TRNG. La nouvelle architecture de TRNG proposée permet de régler beaucoup plus finement le délai différentiel incrémental entre donnée et horloge. C'est ce qui assure l'amélioration de l'entropie générée. Les détails d'implémentation et les résultats de simulation sur la deuxième version du TRNG ASIC_TRNG#2 sont présentés dans la deuxième partie du chapitre 4.

Nous présentons les résultats de mesures de l'aléa du TRNG dans des conditions de travail limite en faisant varier la tension d'alimentation et la température ambiante de l'environnement du circuit. La puce TRNG peut être exposée à certaines perturbations environnementales telles que la variation de la température de fonctionnement iet la tension d'alimentation.

Les variations technologiques, la température et V_{DD} affectent les délais d'interconnexion et les temps de propagation. C'est pour cela que nous ajoutons un bloc d'auto-calibration, qui cherche, durant le fonctionnement, la configuration de délai différentiel optimal dans les chaînes à retard grossier. Figure 7 illustre les paramètres potentiels qui peuvent affecter la sortie de TRNG.

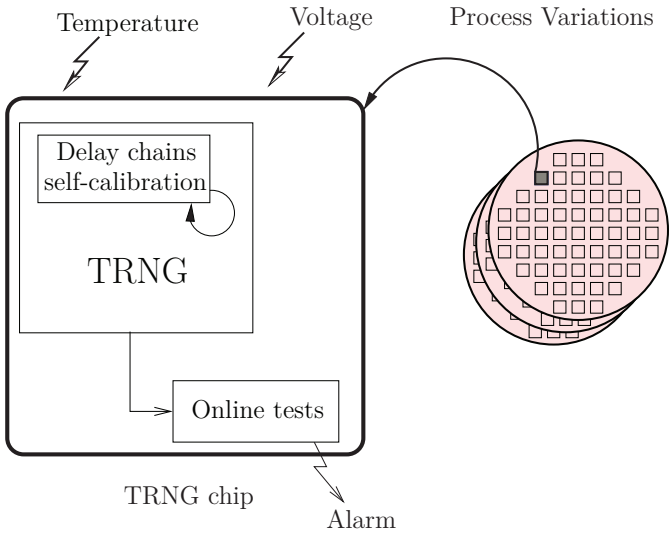


Figure 7 – Environnement du TRNG

Le bloc d'auto-calibration sélectionne la meilleure configuration de commande de retard et contrôle les chaînes de retard grossiers dans un mode de rétroaction. Ce bloc calcule la fréquence de la sortie brute TRNG pour chaque configuration de la chaîne de retard grossier, puis, configure les chaines de retard à la configuration qui correspond à la fréquence optimale. L'algorithme d'auto-calibration des chaînes du TRNG est présenté dans l'algorithme 4.

Le retard différentiel entre donnée et horloge, peut être modifié de manière automatique en cas de perturbation de l'environnement ou de malveillance. Ainsi, cette méthode s'avère être très efficace pour régler les chaînes grossières pour équilibrer la sortie du TRNG et palier à toute perturbation visant à modifier les temps de propagation. Le flux de bits qui a été utilisé lors de la calibration n'est pas conservé.

Le choix de la périodicité d'activation du mécanisme d'auto-calibration revient à l'utilisateur. Ce choix dépend évidemment du niveau de sécurité requis par le TRNG ainsi que de l'application utilisée. Ce test peut être utilisé comme test de démarrage du TRNG.

Dans la suite du chapitre 4, nous présentons les résultats de mesures sur la puce TRNG ASIC en faisant varier température et tension d'alimentation. Nous avons prélevé la fréquence d'apparition de bits égaux à 1 pour des températures variants entre -10 et 70 degrés Celsius et des tension d'alimentation entre 0.6 et 1.8V. Pour chacune des températures nous prélevons aussi la configuration de délai optimale qui nous permet d'avoir le maximum d'entropie. Ceci nous as permis de valider le bon fonctionnement du mécanisme d'auto-calibration.

Des mesures sur une vingtaine de puce ont été aussi entreprises pour étu-

dier l'effet des variations technologiques sur la qualité d'aléa. Comme prévu, les résultats de mesures démontrent bien que les retards de propagation dans les multiplexeurs des chaînes à retard augmentent lorsque la température augmente. En fait, dans le cas de températures typiques et élevés (50C et 70C), le signal de données atteint la fenêtre de métastabilité plus rapidement que à basse température (-10C et 0C). Ces mesures confirment les résultats de simulation pour la variation de la température et d'alimentation V_{DD}.

Le bloc d'auto-calibration s'est avéré être très important non seulement en cas de variations de l'environnement mais aussi pour les variations de processus technologiques. Par la suite, nous comparons les résultats de modélisation du chapitre 3 aux résultats de mesures sur puce.

Le dernier chapitre est consacré à l'implémentation en technologie FPGA. Grâce à leur flexibilité de reconfiguration et des délais de commercialisation courts, les FPGA sont largement utilisés pour des application de cryptographie. C'est pourquoi le besoin de TRNGs sur circuit FPGA a augmenté.

Premièrement nous étudions la faisabilité de chaînes à délais fins. Nous décrivons la méthodologie de conception du TRNG avec des contraintes au niveau placement des éléments de délai. Nous présentons ensuite les résultats des tests statistiques standards et nous comparons les performances de notre circuit TRNG sur un Virtex 5 avec d'autres TRNGs récents dans la littérature.

L'architecture du TRNG à chaînes à retard est fortement sensible au déséquilibre dans le routage. A la différence de la conception du TRNG ASIC, les choses sont moins évidentes côté FPGA. Ceci est principalement dû au fait que pour la technologie FPGA (Xilinx dans notre cas) nous n'avons pas d'informations sur la façon de contraindre le routage et sur le détail de conception des cellules logiques et d'interconnexion. De plus, tout le long du flot de conception FPGA, les outils Xilinx procèdent à des vérifications pour éviter les violations de timing et permettre des optimisations de la fréquence de fonctionnement. Dans ce travail, nous proposons de contourner le processus automatique de routage pour pouvoir concevoir des chaînes à retard fins.

Dans la section 5.2, nous présentons la méthodologie de la conception et la mise en œuvre du TRNG dans la technologie FPGA qui permet de garantir les retards pendant les étapes de placement et de routage. Ensuite, la section 5.3 est consacré aux premiers résultats de la mise en œuvre du TRNG proposé en FPGA. Enfin pour valider notre méthodologie, nous présentons les résultats expérimentaux et l'évaluation statistique de la séquence générée sur un circuit FPGA Virtex 5 de Xilinx.

Contrairement à la technologie ASIC, les modèles de bruit transitoires des primitives des circuits FPGA ne sont pas fournis par les vendeurs. Ainsi, contrairement aux technologies ASIC, nous ne pouvons pas simuler le comportement aléatoire du TRNG. Cependant, nous pouvons programmer le circuit FPGA autant de fois que nous voulons jusqu'à ce que nous trou-

vions la configuration de chaînes à retard optimale qui fournit les meilleurs résultats statistiques.

Dans la suite, nous présentons les spécifications de l'adaptation du TRNG exploitant des chaînes à retard à la technologie FPGA. Dans un premier temps, nous étudions la faisabilité de la conception générique et l'idée que nous proposons à fin d'obtenir de très fins délais différentiels. Ensuite, nous présentons la méthodologie de la conception et la mise en œuvre du TRNG à chaînes à retard dans un FPGA Xilinx.

Côté technologie FPGA, que ce soit pour implémenter des TRNGs ou autres circuits nécessitant un contrôle fin de délai, il existe plusieurs méthodes proposées dans la littérature. Citons notamment celles qui ont été proposées pour implémenter des TRNGs.

- Chaînes à propagation de retenue
- Contrôle des sortances
- Modification des fonctions logiques des LUT
- Changement de routage par reconfiguration dynamique

Nous proposons une nouvelle méthode qui consiste à mettre en œuvre les chaînes de retard sur les données et les chemins d'horloge en utilisant Look-Up-Tables (LUT) avec des entrées différentes. Selon l'entrée LUT utilisée, nous pouvons obtenir des retards différents selon le chemin de routage utilisé par la boîte de commutation. En effet, cette différence de retard n'est pas due au temps de propagation de la LUT lui-même. Elle est due à différents chemins de routage dédiées à chaque entrée de table de conversion. La figure 8 illustre l'idée.

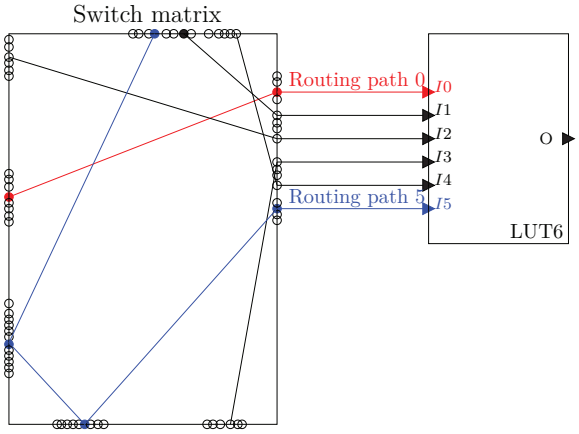


Figure 8 – Chemin de routage introduisant des délais différentiels différents

Donc, nous proposons d'utiliser des entrées de LUT différentes pour chaque chaîne de retard et d'examiner le retard différentiel introduit par les différences dans les chemins de routage. Ceci est illustré dans différentes couleurs dans la figure 9.

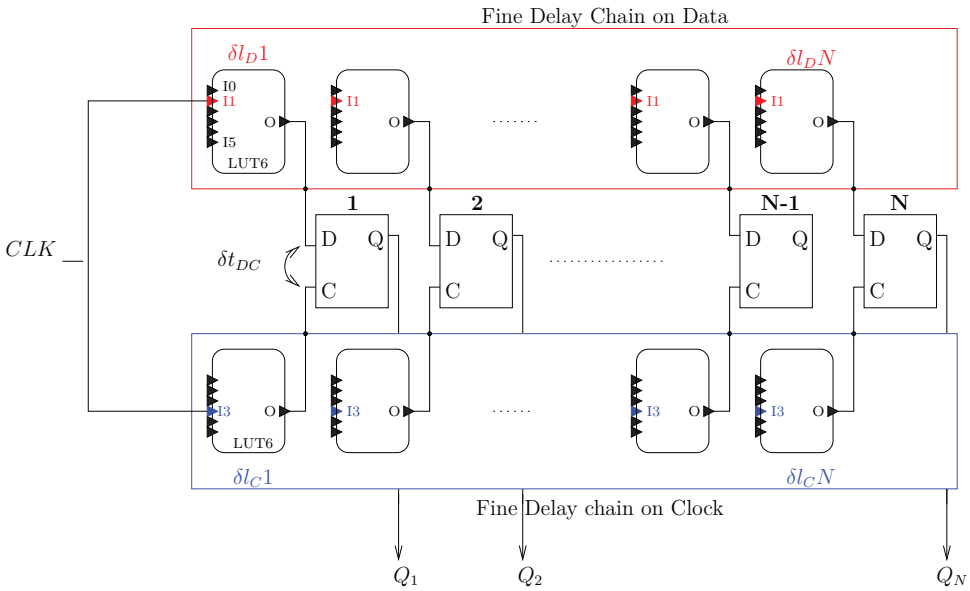


Figure 9 – Chaîne à délais fin utilisant des LUT

La chaîne de retard de données passe par I3 de la LUT et la chaîne de retard d'horloge passe par l'entrée I5 de la LUT. Le processus de placement et de routage doit être examiné très soigneusement et vérifié pour garantir aux chaînes de retard les spécifications voulues. Le problème ici est que les outils de placement et routage Xilinx procèdent à des optimisations et vérification automatique et de synchronisation tout au long du processus de conception. Nous devons donc contourner toute optimisation afin de garantir le même placement et le routage le long des chaînes à retard pour garantir un délai différentiel régulier.

En suite, comme la qualité de la séquence générée dépend étroitement de la finesse du retard sur les latches, nous devons vérifier la qualité de l'aléa à chaque fois que nous modifions le routage. En effet, avec une première acquisition sur FPGA de quelques centaines de bits, nous ne pouvons pas simplement dire si la sortie du TRNG est suffisament aléatoire, nous avons besoin à chaque fois de lancer les tests statistiques standard. Donc, nous devons procéder par essais-erreurs tel que représenté dans le diagramme de la figure 5.1. En cas d'échec des tests statistiques nous devons réajuster les contraintes de placement. Nous réitérons chacune des étapes jusqu'à ce que tous les tests statistiques soient satisfaisant.

Les deux prototypes du TRNG en technologie FPGA et ASIC ont été testés avec les batteries de tests NIST et AIS-31. Ces derniers sont appliquées sur des séquences aléatoires générées à partir de nombreuses configurations de délais des chaînes de réglage à délaiis grossiers. Les résultats des tests statistiques NIST sont donnés dans la section 5.4 et plus de détail dans le tableau 5.10 dans l'annexe. Les tableaux des résultats montrent que le TRNG passe des tests pour 83 % de toutes les configurations de retard. Cela garantit que le TRNG présente de bonnes propriétés statistiques indépendamment des variations des conditions environnementales et technologiques (d'un FPGA à un autre et sur différents emplacement de la puce FPGA). Des mesures en faisant varier température et alimentation devraient être menées sur le Virtex 5 pour étendre les résultats présentés dans le chapitre 5. En fonction de ces résultats, la complexité de conception peut être éventuellement allégée et son débit augmenté en réduisant le nombre de cellules qui composent chacune de ses chaînes de retard.

D'autre part, nous avons conclu, suite à des travaux comparatifs, que le niveau de bruit sur ASIC est beaucoup plus faible que sur FPGA et nécessite un contrôle précis des délais. Les résultats des tests statistiques NIST et AIS-31 prouvent que la qualité de l'aléa sur FPGA est meilleure. Ceci prouve bien que le niveau de bruit en technologie FPGA est plus élevé qu'en technologie ASIC.

Finalement, nous concluons ce manuscrit avec la synthèse des contributions et la proposition de quelques perspectives que nous estimons intéressantes pour la suite de ces travaux. Entre autres, comme travaux futurs, les tests statistiques seront menés sur le second prototype ASIC qui n'était pas livré lors de la rédaction de ce manuscrit de thèse. Un autre bloc devrait être ajouté dans le TRNG qui embarque des tests statistiques appliqués en ligne et lève une alarme en cas de fonctionnement anormale ou de perturbations d'entropie malveillantes. Cette fonction est donc nécessaire pour être conforme aux normes de certification. Le bloc de test en ligne devrait tenir compte de la contrainte de faible complexité, particulièrement importantes pour des applications à faibles ressources. Une analyse d'injection de fautes pourrait être effectuée pour évaluer l'impact du sur-échantillonnage, de l'injection de faute laser et des attaques par injection électromagnétiques sur la quantité d'entropie. Cela permettrait de vérifier l'efficacité des tests en ligne et donnerait quelques idées pour améliorer l'architecture contre ces attaques.

Abstract

Random numbers are required in numerous applications. For instance, they are needed in system validation by Monte Carlo simulations and emulation of channels for digital communications. Another important application is cryptography where randomness is used in security protocols. There are two main classes of Random Number Generators (RNG): The Pseudo RNG (PRNG) which have a deterministic sequence, and the True RNG (TRNG) which generates unpredictable random numbers. Cryptographic applications use both TRNG and PRNG. The PRNG needs an initial value, or seed, which can be the output of a TRNG. The TRNG exploits physical phenomenon, thus it is preferable to design them using analog technologies. In digital technologies, like FPGAs, TRNGs are commonly based on oscillators which have the drawback of being biased by harmonic coupling. In order to assess the entropic quality of TRNGs, standards based on statistical tests have been elaborated by certification organisms namely the NIST and the BSI. However, it is recommended to formalize the stochastic behaviour of the randomness generation process.

In this Ph.D, we address the design and quality evaluation of TRNGs in digital circuits. As a first step, we review the different TRNGs classes and the methods to evaluate the randomness quality. The techniques and physical phenomena exploited for on-chip randomness in CMOS technologies are presented. Then, the different threats which can deteriorate the randomness are reviewed. In the second step, a study of a low-cost digital TRNG without oscillators (open-loop TRNG), hence robust against harmonics attacks, is presented. The proposed TRNG exploits both the metastability phenomenon and the jitter noise in CMOS digital flip-flops to generate the random numbers. A stochastic model of this TRNG has been formalized. This model describes the random generation process that allows to obtain the minimum entropy level that can be generated, regardless of the targeted technology. The characterization and evaluation on a prototype circuit, in FPGA and ASIC technologies, has shown that the proposed TRNG architecture generates randomness of good quality and is robust against environmental variations.

Keywords: True Random Numbers Generator, metastability, ASIC, programmable delay, delay chains, FPGA, noise, standard statistical tests, NIST, AIS-31

Contents

List of Figures

List of Algorithms

List of Tables

List of Abbreviations and Acronyms

AES Advanced Encryption Standard

AIS Anwendungshinweise und Interpretationen zu Common Criteria

ANSSI Agence Nationale de la Sécurité des Systèmes d'Information

ASIC Application Specific Integrated Circuit

BRAM Block Random Access Memory

BSI Bundesamt für Sicherheit in der Informationstechnik

CC Common Criteria

CMOS Complementary Metal Oxide Semiconductor

CRI Cryptography Research, Inc

DFA Differential Frequency Analysis

DFF D-Flip-Flop

DFT Discrete Fourier Transform

DPA Differential Power Analysis

DNA Deoxyribo-Nucleic Acid

EMI Electro-Magnetic Interference

EMIA Electro-Magnetic Injection Attack

EMV European Mastercard Visa

FIA Fault Injection Attack

FIB Focused Ion Beam

FIPS Federal Information Processing Standard

FPGA Field Gate Programmable Array

HRNG Hybrid Random Number Generator

IC Integrated Circuit

IV Initialisation Vector

LASER Light Amplification by Stimulated Emission of Radiation

LED Light Emitting Diode

LUT Look-Up-Table

MDL Measurement Design Language

MOSFET Metal Oxyde Semiconductor Field Effect Transistor

MSS MetaStable State

MUX MUltipleXor

NIST National Institute of Standards and Technology

NMOS N-channel MOSFET

NONCE Not ONCE the same number

NPTRNG Non Physical True Random Number Generator

OSC OSCillator

Op.Amp. Operational Amplifier

PAR Place And Route

PCB Printed Circuit Board

PDL Programmable Delay Lines

PLL Phase-Locked Loop

PMOS P-channel MOSFET

PP Post-Processing

PRNG Pseudo Random Number Generator

PTRNG Physical True Random Number Generator

QPTRNG Quantum Physical True Random Number Generator

RAM Read Access Memory

RO Ring Oscillator

SCA Side Channel Analysis

SOA State Of the Art

SPA Simple Power Analysis

SRAM Static Read Access Memory

STM ST Microelectronics

STR Self Timed Ring

TOE Target Of Evaluation

TRNG True Random Numbers Generator

VCO Voltage Controlled Oscillator

VN Von Neumann

XOR eXclusive OR

General introduction

Context

Randomness is used in a large scope of applications covering test vectors for Integrated Circuits (ICs), Monte Carlo simulations and gambling. Randomness is also essential in some cryptographic protocols such as random session keys generation, Initialization Vectors (IVs), NONCE (Not Once the same number) and protections against physical attacks [MvOV96]. Dynamically generated randomness, which occurs during circuit operations, is due to ambient noise caused by voltage fluctuations and many other noise sources, as thermal noise. This dynamic randomness is exploited in True Random Number Generator (TRNG) because of its unpredictability. Thanks to their simplicity of design, Pseudo Random Number Generators (PRNG) are widely used although not cryptographically secure as they are generated based on a deterministic sequence [MvOV96]. This is why PRNGs are initialized with a *seed* that are generated from a True Random Number Generator. Such a combined structure is often called a Hybrid RNG (HRNG). The unpredictability of the seed of PRNGs or stream cipher is a sore point in cryptographic protocols.

Physically implemented RNGs in digital Integrated Circuits (IC) should generate "true" random bits. In fact, untrustworthy TRNGs may threaten the security of the whole cryptographic system. To be cryptographically secure and compliant with Common Criteria (CC) certification schemes, the TRNG should [KS11, Joi13]:

1. have good statistical properties *i.e.* first the generated sequence must be uniformly distributed and, second, knowing the history of the generated sequence, one can not predict the successors,

2. embed self tests that rises an alarm in case of source of noise failure and

3. be robust against environmental perturbations and physical attacks.

The second and third requirements are as much important as the first one for the cryptographic applications. Ideally on-line tests should be performed to detect environmental perturbations or malevolent attacks [KS11].

Motivations

Up till the late 90's, hardware random number generators was designed exclusively with analog components such as zener diodes and resistors which

exploit respectively thermal noise [JK99] and shot noise [KS08]. The noise is then amplified and converted to a digital random bit '0' or '1'. For digital technologies, like FPGAs or standard cell ASICs, the most common method to build TRNG is based on ring oscillators (RO) which takes advantage of the noise jitter on a clock signal. However, recent physical attacks on RO-based TRNGs have shown that a bias can be introduced by means of harmonic injection [MM09]. Apart from its vulnerability towards this attack, RO-based TRNGs can present a good quality of randomness.

Another method to extract the noise is to place a flip-flop in a metastable state and observe the convergence towards a stable state. This convergence is given by the current voltage noise. This technique is hardly possible in digital technologies as the metastability is not digitally controllable given that the metastable voltage is around $V_{dd}/2$, V_{dd} being the power supply volatge. In this Ph.D work, we propose to study an "open-loop" architecture which theoretically extracts noise coming from both metastability and jitter and is intrinsically robust against harmonics injection attack. The other motivations are to provide a portable TRNG in fully digital technology, like FPGA and standard cells ASIC, and compliant with standard statistical test suites. The robustness of a TRNG is strongly related to a permanent quality of the generated bit stream despite the variations in its environment, the potential external attacks and aging effects. To summarize, the main motivations are:

- Propose and study a TRNG structure which is portable, compliant with standard statistical tests.
- Prove the randomness quality by proposing a stochastic model.
- Address the security and reliability properties to make sure the quality of the TRNG is not impacted by environmental variations or attacks.

Contributions

In this Ph.D work, the main contributions are the study, design and characterization of a TRNG based on open-loop delay chains in both ASIC and FPGA technologies. This kind of TRNG structure offers the advantage of high throughput and resiliency against harmonic Electromagnetic Injection Attacks (EMIA). The characterisation stage includes environmental tests on real silicon to ensure that the randomness quality is preserved.

Another contribution is to introduce a stochastic model of the open-loop delay chains based TRNG. It has been proven that the metastability phenomenon can be efficiently mixed up with jitter noise for true randomness generation. A model of the delay chains based TRNG has been build to allow the designer to simulate the TRNG. Hence the statistical evaluation of the

randomness can be performed before the fabrication process. This model has been validated through simulation and evaluation of an ASIC prototype.

This manuscript is organized as follows:

The first chapter describes the state-of-the-art with the principles of the existing TRNGs which are presented according to a proposed classification.

Then, we present a second chapter addressing the existing techniques to evaluate a TRNG in terms of randomness quality and physical robustness. This chapter presents three essential parts towards the certification of a TRNG:

1. the method to formalize the randomness by stochastic modeling,
2. the methods to evaluate statistically the TRNG,
3. the robustness against physical attacks with the description of some significant cases.

At the end of this chapter, a synthetic comparison of existing TRNGs is given in terms of complexity, throughput, testability at design time and security.

In the third chapter, we present the principle of the proposed "open-loop" TRNG architecture based on delay chains. This chapter presents also a mathematical stochastic model that describes the design parameters that affect the entropy of the delay chains based TRNG. We give a probabilistic study of the TRNG output.

In chapter four, we study the ASIC architecture of the generic delay chains based TRNG. This technology allows us to study the impact of the placement and routing and build characterization methods based on electrical simulations. Along with the design of the prototype circuit, models issued from simulation and stochastic approach have been validated through measurements. Also the impact of constrained placement and routing on the randomness quality has been studied by applying standard statistical tests. In order to be sensitive to very small noise level, a second architecture with new delay chains has been proposed in a second ASIC prototype.

Chapter five is dedicated to the feasibility and adaptation of the generic architecture of the delay chains based TRNG on a Xilinx FPGA. It first presents the study the architecture of Xilinx FPGAs and propose an implementation based on LUT differential delays to get high delay precision. This chapter also presents the randomness evaluation for different environmental conditions of this TRNG FPGA.

Finally, we conclude this manuscript by summarizing the main contributions of this Ph.D work and some openings to future works.

Chapter 1

State-of-the-art of True Random Number Generators

1.1 Introduction

Over the past ten years, great efforts have been undertaken to implement on-chip true randomness generation. In this chapter, we present the existing solutions to generate on-chip true randomness. In the first section, we present the TRNGs standard structure by describing the role of each of its main blocks. First, we give an overview of the main sources of noise in CMOS technology. Then we describe the requirements of the entropy extraction, online tests and post-processing blocks. Meanwhile, we propose a novel criterion of classification of the existing TRNGs. Then, we give an overview of the existing on-chip TRNGs by classifying them in terms of source of noise and entropy extraction.

1.2 Principle and Classification Proposal of TRNGs

Basically, a random set of bits or set of states could be seen as the result of the flips of a coin with a head side labeled '0' and a tail side labeled '1' with each flip having a probability of exactly 50% of producing a '0' or a '1'. Furthermore, each flip is independent from the others such that the current coin flip result does not affect the future ones. A TRNG should operate exactly in this same way.

Several True Random Number Generators (TRNG) have been presented in the literature, spanning from Non-Physical TRNG (NPTRNG) to physical TRNG (PTRNG). NPTRNGs extract randomness from non physical processes. For example, from computer peripherals namely from the random access to RAM, mouse motion, hard disk access or USB port access on the computer [MvOV96]. Software entropy collector exist such as the Linux kernel RNG whose output can be accessed through special files such as */dev/random* and */dev/urandom*. Unlike NPTRNGs, PTRNGs extract randomness from physical phenomena. The exploited phenomena can be as complex as generating a random number from a solid phase synthesis of an oligonucleotide sequence of the DNA [GAR10] or the random events of the radioactive atom disintegration [Wal01]. Other optoelectronic TRNGs exist for

instance the super-luminescent LED (Light Emitting Diode) based PTRNG [LCMR11] or the chaotic semiconductor laser PTRNG [ZWW11]. Quantum based PTRNG (QPTRNG) have also been introduced recently. The need for quantum TRNGs has recently overgrown because of the extension of the quantum information and computation and quantum cryptography namely the Quantum Key Distribution protocol [SBpC$^+$09]. QPTRNGs generate a random state for example from the polarisation beam splitting of a random polarized photon which is generated by a LED. This QPTRNG is proposed in [JAW$^+$00] and in [Sti11]. Another QPTRNG which exploit the random transmission of a photon through a semi-transparent mirror was introduced in [IDQ10].

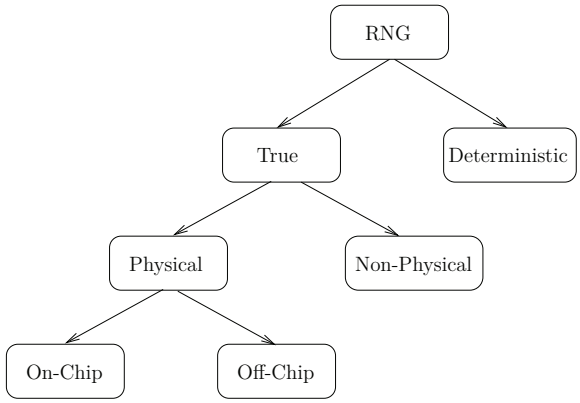


Figure 1.1: Random Number Generators Classification

In the diagram of Figure 1.1, we present the general classification of random number generators. A wide spectrum of on-chip or "integrated" TRNGs have been published over the last two decades. In this work, we focus on the integration of on-chip TRNGs in both Application Specific Integrated Circuits (ASIC) and Field Programmable Gate Arrays (FPGAs). Historically, the first on-chip TRNGs have been designed for ASICs. Then with the spread of FPGAs use, not only for research and prototyping, but also for industrial products, the need for robust TRNG designs on FPGA has grown especially for cryptographic applications. Whatever the source of randomness and the targeted technology are, all TRNGs have the common structure of Figure 1.2. This standard structure is proposed by AIS-31 (Anwendungshinweise und Interpretationen zu Common Criteria) [KS11].

Basically a TRNG is composed of four blocks such as illustrated in Figure 1.2:

1. A physical source of randomness block which is also called source of entropy

2. An entropy extraction and digitisation block

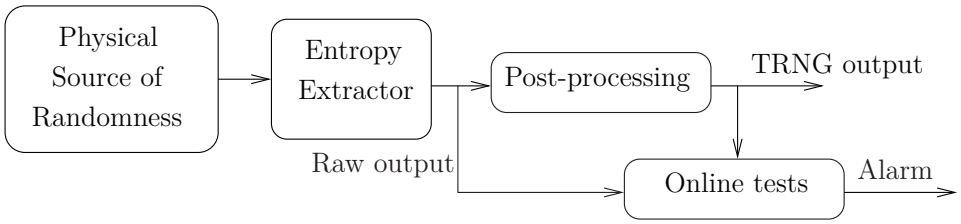


Figure 1.2: AIS-31 standard architecture of True Random Number Generators

3. An online tests block
4. A post-processing block

1.2.1 Source of entropy

This block constitutes the analog part of a TRNG. Each TRNG uses a specific randomness source which is based on an unpredictable physical phenomenon. In silicon technologies, physical sources of noise come from non-deterministic effects on electronic components. MOSFET transistors exhibit different sources of noise [Bak10]. The first one is the thermal noise in the MOS transistor channel and also in the resistive metal of the transistor gate. It is characterized by voltage fluctuations caused by the random motion of electrons in any resistive medium. Its amplitude increases with the temperature. The thermal noise is the dominant noise at higher frequencies. The magnitude, $V^2_{thermal}$, of the total thermal noise power is given by Equation (1.1).

$$V^2_{thermal} = \frac{k_B \cdot T}{C} \tag{1.1}$$

where $k_B = 1.3810^{-23} J/K$ is the Boltzmann constant, T the temperature in Kelvin and C the load capacitance in an RC equivalent circuit. Second, the flicker noise is characterized by the effect of randomly trapped charges at the surface of the gate oxide [Liu06]. The effect from the randomly trapped charges can be modeled as a random voltage source at the gate of the device. The magnitude of the flicker noise is given by Equation (1.2).

$$V^2_{flicker}(f) = \frac{K}{f \cdot L \cdot C_{ox}} \tag{1.2}$$

where K is an empirical constant found by measurement, f is the frequency of the noise, L is the effective gate length of the MOSFET transistor and C_{ox} is the capacitance of the gate oxide. Equation (1.2) shows the flicker noise voltage is inversely proportional to the frequency. It is especially pronounced in MOSFET transistors with small channels. Figure 1.3 depicts the spectral

density increase of the flicker noise for lower frequencies. The flicker noise is the principal source of the jitter [Abi06]. Its effect is more important in deep sub-micron CMOS technologies.

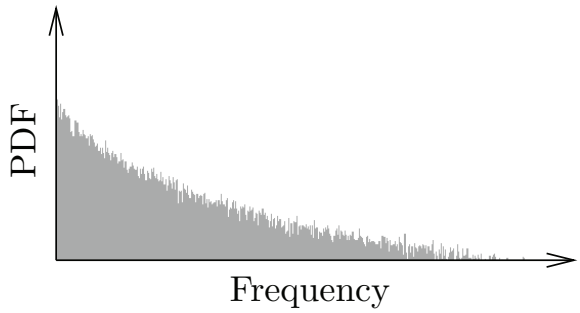


Figure 1.3: Noise power density function (PDF) of Flicker noise.

Third, the shot noise is characterized by the leakage current of the drain-to-source reverse diodes. The power spectral density of the shot noise is determined empirically and it is given by Equation (1.3) according to [Bak10]. The shot noise is not present in short channel CMOS technologies under 20nm.

$$I_{shot}^2 = 2 \cdot q \cdot I_D \tag{1.3}$$

where q is the electron charge of 1.610−19 coulombs and I_D is the current flowing in the MOSFET channel. At low frequencies, the primary dominant noise sources are the thermal noise and the flicker noise. Another source of noise has been identified in CMOS technology which is the popcorn noise, also known as random telegraph signal (RTS) noise. The popcorn noise characterises the noise caused by the electrons and holes combination-generation process in a PN junction. The baud diagram of the popcorn noise is equivalent to a low pass filter [Bak10]. The power spectral density of the pop-corn noise can be modeled using Equation (1.4).

$$I_{rts}^2(f) = \frac{K_{rts} \cdot I_D}{1 + (\frac{f}{f_c})} \tag{1.4}$$

where K_{rts} is an empirical evaluated constant which has a unit of Ampers per Hertz, f_c is the cut-off frequency of the equivalent low pass filter and I_D is the current flowing in the MOSFET channel. To evaluate the total popcorn noise power, Equation (1.4) is integrated for f from zero to infinity. At low frequencies, the PSD of the popcorn noise exhibits a white noise similarly to the thermal noise and shot noise. However, at higher frequencies, the popcorn noise decreases as $\frac{1}{f^2}$. In addition to thermal, shot, flicker and popcorn noise, phenomena such cross-talk effects between adjacent wires and electromagnetic interferences (EMI) from external environment intensify the

ambient noise effect [Fai99]. EMI is caused by the radiation outside the chip mainly by the I/O pads of the chip, power supplies and I/O cables on the printed circuit board (PCB).

1.2.2 Entropy extractor

In this block, the analog noise signal is extracted and digitised. This is done by means of sampling a noisy signal using "sample and hold" circuits. Figure 1.4 illustrates a sample and hold circuit with both the sampling and sampled signals.

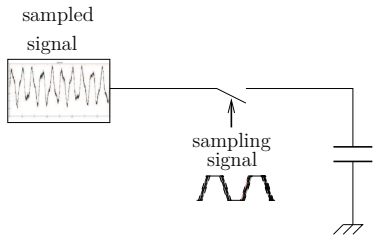


Figure 1.4: Principle of noise extractor.

For digital circuits, they are performed by memory elements (e.g. D-flip-flops(DFF), D-latches or SRAM (Static Random Access Memory) cells). The sampling is generally performed by a clock signal which allows the TRNG to output random bits periodically. Figure 1.5 illustrates all the CMOS memory elements that can be used in order to sample a noisy signal.

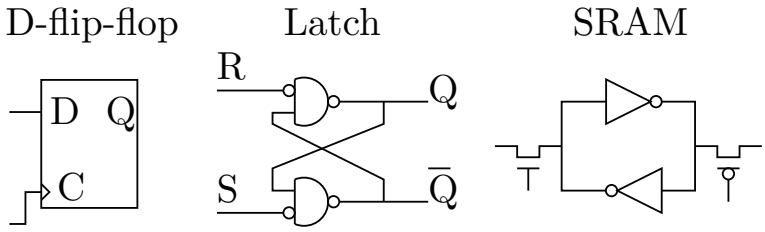


Figure 1.5: CMOS bi–stable elements for entropy extraction.

The noise impacts both the timing and the amplitude of signals [FAB+09]. Figure 1.6 illustrates two types of noise. The noise on timing is commonly called "jitter noise", we will call it "phase noise" as it happens on the time axis around the sampling edge of the sampling signal. The noise on the amplitude is grabbed at a "metastable state" of the memory element. we refer to this type of noise "voltage noise" as it comes from the amplitude of the sampled signal. In CMOS technology, the metastable state voltage is around $\frac{V_{DD}}{2}$. The time of sampling uncertainty can be considered as a

time uncertainty around the sampled signal switching. This extraction and

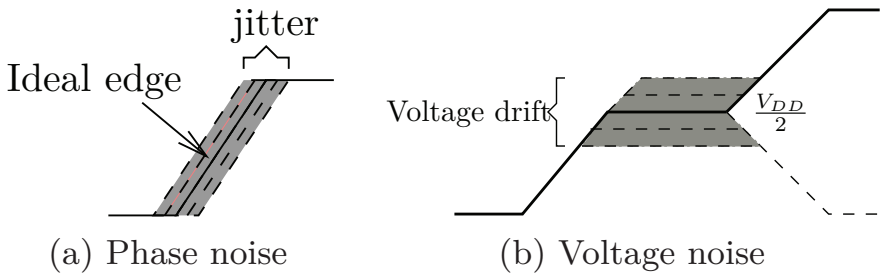


Figure 1.6: Illustration of phase and voltage noises respectively on the sampling and the sampled signals.

digitisation mechanism generates a sequence of digital bits called the raw output.

1.2.3 Online tests

Some hardware statistical tests should be embedded within the TRNG chip to test the TRNG output sequence during operation [Sch01]. Online tests should check the quality of the random numbers first at TRNG reset, then applied continuously or permanently [KS11]. The online tests block should rise an alarm in case of failure of the source of entropy. In fact, on-chip TRNGs can be influenced by ageing, deterministic noise of surrounding logic, temperature and supply variations and also physical attacks [Joi13].

The complexity of these online tests should be reasonable in comparison with the TRNG core since they are embedded directly within the TRNG chip. Power-up online tests should run fast to avoid big latency times. The latency time or power-up time is defined by the time needed from the TRNG online tests to release a safe random sequence. Moreover, when the online tests are applied periodically they decrease the throughput. The online check tests are not intended to measure the entropy per bit of the TRNG. They are actually used in order to check if the TRNG output is stuck outputting periodically the same patterns or long sequences of consecutive zeros or ones or alternating zeros and ones periodically. Hence online tests can be very simple and light. For example, we can count the number of occurrences of ones and check whether it is not highly offset from 50%. Online tests can be applied either on the TRNG raw output sequence or on the TRNG post-processed sequence. This choice depends on the level of security strength required from the certification standard AIS [KS11].

1.2.4 Post-processing

The raw output of the TRNG may present bad statistical properties such as long sequences of ones or zeros or a repetitive pattern of bits. This bias can actually originate from deterministic noise from external sources such as power supply variations or crosstalk effects coming from surrounding logic. In this case, post-processing algorithm can be used in order to increase the entropy per random bit. The post-processing block is not mandatory if the TRNG raw output presents good statistical properties. Post-processing techniques may be arithmetic or cryptographic [NIS12, KS11]. Typical arithmetic post-processing methods are using simple de-biasing function namely the Von Neumann correction technique [vN51, ECS05]. The von Neumann debiasing solution is the most used one since it requires a very lightweight hardware. Von Neumann correction algorithm considers the RNG sequence pair by pair and proceeds like the following. *"00"* and *"11"* pairs are discarded and the *"10"* and *"01"* pairs are replaced respectively by *'1'* and *'0'*. However, the bit rate of the post-processed output may drop of 25% compared to the bit rate of the raw output sequence. The drawback of this method is the

Algorithm 1 Algorithmic implementation of the Von Neumann post-processing technique.

N : raw sequence length
Valid : table of N bits
$j = 1$
for i in $(0, \frac{N}{2} - 1)$ **do**

  if $S_{raw}[2i+1] \oplus S_{raw}[2i+2] = 1$ **then**

    $S_{vn}[j] = S_{raw}[2i+1]$

    j=j+1

    Valid[i]=1
  else

    Valid[i]=0
  end if
end for
return S_{vn}

unknown waiting time until the required post-processed random bits are available [vN51, Dic07]. In fact, the throughput of the post-processed output depends on the randomness of the raw output. In fact the bit rate of the post-processed output varies depending on the entropy of the raw output.

Other post-processing methods are also widely used such as Linear Feedback Shift Registers (LFSR). The commonly cryptographic post-processing are hash functions or stream cyphering algorithm [NIS12, KS11]. Block cipher algorithms, such as AES (Advanced Encryption Standard), are also commonly used for post-processing. This can be found, for example, in the VIA Nano CPU [VT08] and Intel's new Ivy bridge CPU [HKM12]. In [KEC+11], authors studied and compared these post-processing techniques.

1.2.5 TRNG classification

In literature, several solutions for on-chip randomness generation in digital ICs have been introduced. We propose to classify the exiting TRNGs in terms of entropy extraction. We identify three families of TRNGs:

- TRNGs that exploit the phase noise. The jitter phenomenon characterizes the sampling time uncertainty. This family encloses the method of sampling a jittery oscillating elements, such as ring oscillators, by jittery clock.

- TRNGs that exploit the voltage drift phenomena around the $\frac{V_{DD}}{2}$ voltage level. This is commonly known as metastability in bi–stable elements. This family encloses the following randomness extraction solutions:

  - Writing data to the same memory location or sampling randomly initialized memories at power-up.
  - Placing a bi–stable cell in a metastable state, then observing the stable state which is the consequence of ambient noise.

- TRNGs that exploit both phase and voltage uncertainties. We refer to the family of TRNGs that exploit both phase and voltage uncertainty noises as phase-voltage noise based TRNGs.

In the next section, we present the TRNG state-of-the-art for each family.

1.3 Phase Noise based TRNGs

The common method to generate on-chip true randomness is to use a jittery clock which samples a jittery oscillating structures such as voltage controlled oscillator (VCO), phase locked loop (PLL) and free-running ring oscillators. In this state-of-the-art of TRNGs exploiting phase noise, we present the most known and used TRNG architectures.

1.3.1 Intel's dual-oscillator based TRNG

In 1999, Cryptography Research Inc. (CRI) presented a review of Intel's 1^{st} generation TRNG [JK99] where they introduce its principle. The dual-oscillator based TRNG extracts randomness from the amplification of the thermal noise of an integrated resistor. Figure 1.7 illustrates the architecture of the first Intel TRNG. A voltage controlled oscillator is fed by the noisy

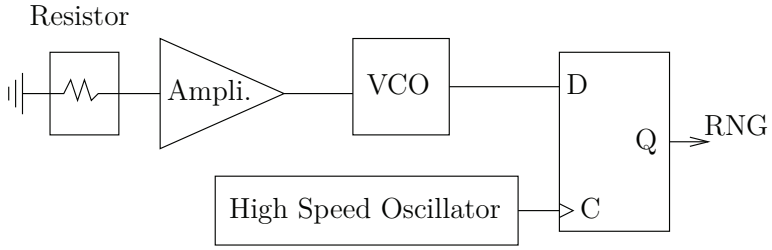


Figure 1.7: The dual oscillator based architecture of Intel's first generation of TRNGs.

amplified signal. The resulting signal is a slow jittery clock, which is then sampled by the high frequency system clock. The drift between the noisy low frequency oscillator and the clock produces random bits.

The TRNGs based on the extraction of resistors thermal noise are very much sensitive to the chip temperature. In [SCDC+11], authors studied the impact of temperature variations on such analog TRNG structures. For temperature values higher than the ambient temperature, the TRNG output is biased and the NIST statistical tests fail.

1.3.2 Two-ring-oscillator based TRNG

Sampling free running oscillators with a jittery output is the common method to generate true randomness since it requires simple ring oscillators structures which are easily implementable. A ring oscillator consists in a chain of delay elements which are logically inverted in a feedback loop. The IC community have devoted several studies to characterize jittery behaviour of free running oscillators. Ring oscillators are also embedded by circuit manufacturer into IC wafers to measure to characterize technology process variations and the maximum working frequency of the produced ICs [BGKD06].

The uncertainty around the clock edges is due to instabilities in the propagation delays of the inverting cells which compose the ring oscillator. These instabilities may come from temperature variations. Semiconductor process variations namely the non uniform doping density can also cause the clock jitter. The random properties of jitter combined with several characterisation studies for both ASIC and FPGA technologies, have made ring oscillators a

widely used structure to generate on-chip randomness.

To understand the principle of jittery ring oscillators based TRNGs, let us go further with a simple two ring oscillators based TRNG, the Kohlbrenner and Gaj's TRNG [KG04]. The latter has the architecture presented in Figure 1.8. Here S_j is the random output. If the rising edge of the *OSC0* output, $C0$, comes before the rising edge of $C1$, S_j is equal to '1' else S_j is '0'.

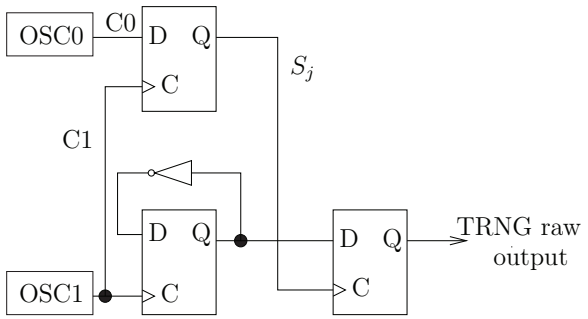


Figure 1.8: Architecture of the two ring oscillators based TRNG.

The TRNGs that extract noise from jittery oscillators use either two ring oscillators (RO) such as described above [KG04] or multiple RO such as proposed in [SMS07], [SPV06] and [AFR08] or several RO-based PRNGs [Gol06, DG07]. Another two ring oscillators (RO) TRNG structure is proposed by Tkacik in [Tka02]. It is a US patented by Motorola. It exploits two jittery ring oscillators. One feeds a Linear Feedback Shift Register (LFSR) and the second one feeds a Cellular Automata Shift Register (CASR). Figure 1.9 illustrates the Tkacik's TRNG structure.

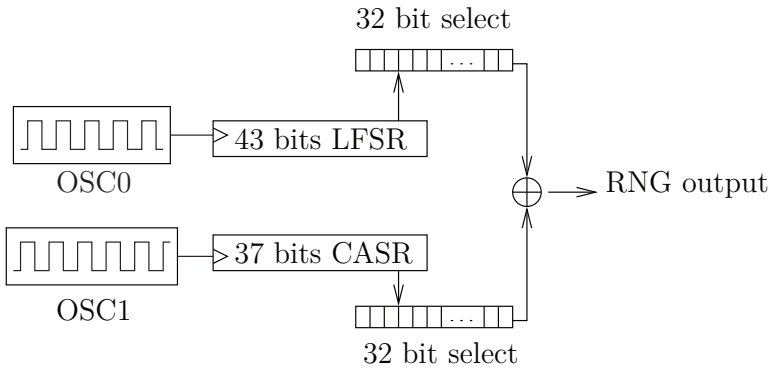


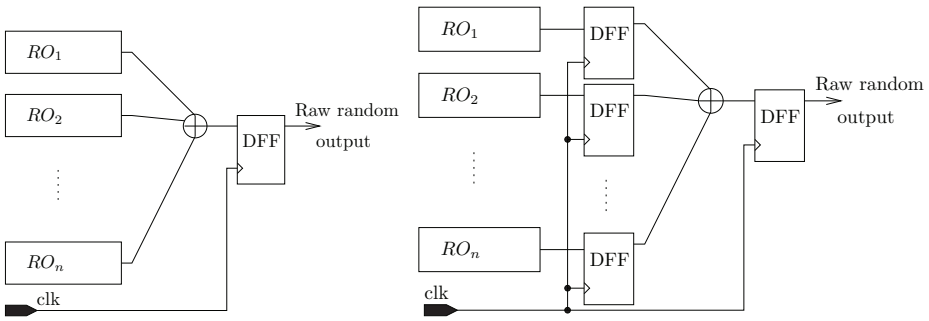
Figure 1.9: Architecture of the two ring oscillators based TRNG.

This structure is very simple however it does not embed any logic to test online the statistical properties of the TRNG raw output. In CHES 2003, an

attack has been experienced on the Tkacik's TRNG in [Dic03].

1.3.3 Multiple-ring-oscillator based TRNG

Figure 1.10a illustrates the multiple ring oscillators based TRNG. The outputs of n ring oscillators are XORed. The resulting signal is then sampled using a DFF. The DFF output represents the raw random bit output of the TRNG. Sunar et al. published a stochastic model of this TRNG in [SMS07, YSKB07].



(a) Sunar's structure (b) Wold's structure

Figure 1.10: Multiple ring oscillator based TRNG.

Wold et al. propose an enhancement of Sunar's TRNG in [WT08]. A DFF is added at the output of each ring oscillator such as illustrated in Figure 1.10b. Thanks to the synchronization of ring oscillators outputs, the probability of interaction between the ring oscillator jittery outputs is reduced. The TRNG output sequence of Wold's enhancement on the multiple ring oscillator based TRNG allows to pass the standard statistical tests without any post-processing. In fact, sampling the output of each RO independently allows to exploit the phase uncertainty at each sample at the outputs of the added DFFs.

The advantages of the RO-based TRNG cited above are the ease of design. In fact, they require only inverter gates and therefore are highly portable. They also feature several Mbps throughputs. RO-based TRNG structure have been experienced through several FPGA chips from different vendors and also on ASIC technologies. According to their authors, the above presented RO-based TRNGs present good quality of randomness since they pass NIST standard statistical tests. However, this class of TRNGs has proven to present a serious issue regarding security applications. Its drawback is declined in design oriented drawbacks and security oriented drawbacks:

- Design oriented disadvantage is the need for placement constraints to reduce the effect of process variation on the RO. This is needed to

ensure that RO frequencies are relatively primes [SMS07].

- The security oriented drawbacks are related to its vulnerability towards physical attacks. In fact, the ring oscillators can be mutually locked by a harmonic injection attack [BBA+12]. If the rings are not independent, their mutual phases are not uniformly distributed. This causes a bias at the TRNG output.

Physical attacks are out of the scope of this section. We present the harmonic injection attack of [BBA+12] in the Section 2.4 which is dedicated to the issue of TRNG physical evaluation. Other FPGA TRNG designs exploiting jitter as source of randomness exist namely the *PLL* (Phase Locked Loop) based TRNGs proposed in [VF02, FDcC04, LM05].

1.3.4 PLL-based TRNG

In [FDcB04], authors propose an FPGA implementation of a TRNG. The TRNG exploits the jitter produced in a PLL. The random output is obtained by sampling the CLJ signal by the CLI jittery clock signal. The timing diagram of Figure 1.11 depicts the random samples extraction. Figure 1.12

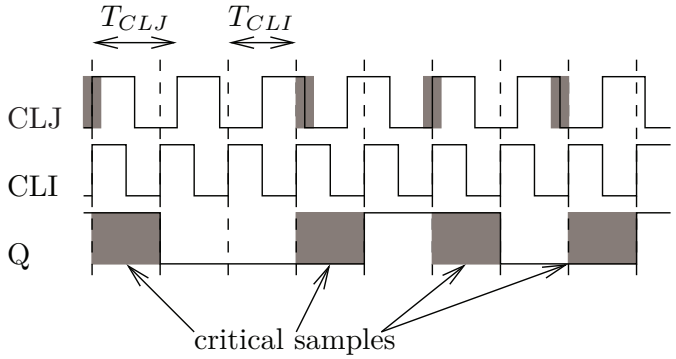


Figure 1.11: Digital timing diagram of randomness extraction in the PLL-based TRNG.

illustrates the structure of the PLL-based TRNG.

The PLL-based TRNG has been experienced and its output statistically evaluated through several Altera and Actel FPGAs. Authors also validated the quality of randomness through a stochastic model [cDFF06]. In this model authors identify the PLL-based TRNG design parameters and propose a mathematical model to test and enhance the TRNG quality of randomness at design time. Stochastic modeling is out of the scope of this chapter. The PLL-based TRNG stochastic model is described in Section 2.3. The main drawback of the PLL-based TRNGs is their need for a PLL block. However,

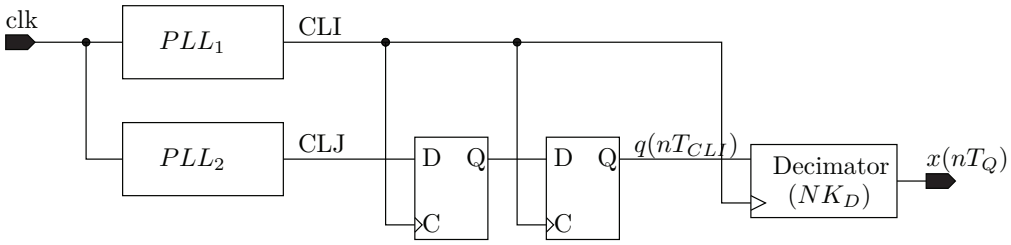


Figure 1.12: PLL-based TRNG architecture.

most FPGAs feature a limited number of PLL which are generally used to generate different clock domains. All the above cited RO-based TRNGs present a good quality of randomness as they pass the NIST statistical tests. However these structures are technology dependent and potentially vulnerable against physical attacks. In the same family of jitter-based TRNGs, a novel RO-based TRNG has been proposed. It exploits an asynchronous free running oscillator.

1.3.5 Self-timed ring based TRNG

Recently, a single ring-oscillator based TRNG has been proposed in [CFAF13]. The Self-Timed Ring (STR) oscillator is designed by an asynchronous FIFO (First In First Out memory) that has been closed to form a ring of L stages. The Self-Timed Ring block provides L jittery signals. A clock samples each inverting stage output, C_i where $1 \leq i \leq L$, using DFFs. The L outputs of the DFFs are combined using a XOR function which result in the TRNG raw random bit output. The entropy extraction block of this STR-based TRNG is depicted in Figure 1.13.

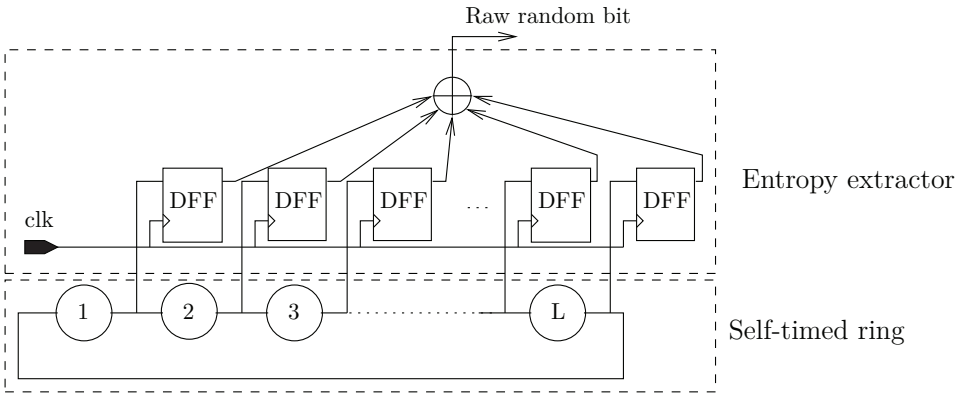


Figure 1.13: Self-timed ring based TRNG architecture.

The advantage of the STR-based TRNG is that it allows to control precisely the relative phase differences between the asynchronous ring stages. Since it exploits only a single ring oscillator, this architecture is inherently robust against mutual locking phenomena. In the following subsection, we present another family of TRNGs which is the memory cells based TRNGs. This category of TRNGs extract randomness either from memory writing conflict or from non initialized memories.

1.4 Voltage Noise based TRNGs

This section includes TRNGs which exploit voltage drift uncertainty around a metastable state. Given our current knowledge, three different methods exist in the literature:

1. forcing bi–stable cells to operate in the metastable region by customizing the transistors of the bi–stable cells.
2. forcing memory cells to metastable states by means of memory writing conflicts.
3. extracting memory cells initial state which is random at power-up.

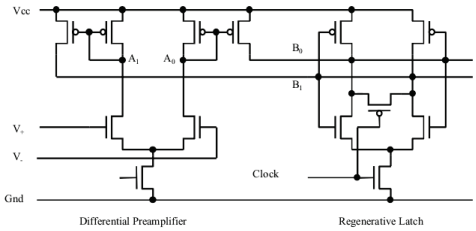
TRNGs designs which exploit voltage noise are rarer in the literature than those that exploit solely the jitter. In the following, we present two designs which exploit the first method [KC02, HKM12] and two other designs that exploit respectively the second method [GCS09] and the third method [HBF09].

1.4.1 Metastability based TRNG

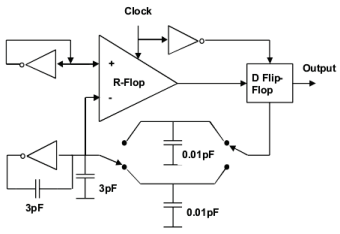
1.4.1.1 Kinniment's TRNG

In 2002, Kinniment et al. presented the first TRNG based on the principle of metastability in [KC02]. This TRNG uses a custom analog cell which is controlled in order to force the metastable state. The principle of such metastability based TRNG is to drive a bi–stable cell in a metastable state which will resume eventually in a random stable state. The metastability control circuitry is called R-flop. Figure 1.14a illustrates the R-flop full custom circuit [KC02].

The R-flop circuit is composed of a differential amplifier and a latch. The analog input stage of the differential amplifier drives a bi–stable cell with a small current difference. Thus, the output resolves to a high logic value or low, when the latch is clocked with very low voltage levels. If $\Delta V = V^+ - V^-$ at the amplifier input is lower than than 0.1 mV then the ΔV at the bi–stable cell input is lower than the thermal noise voltage uncertainty. The R-flop present a bias caused by temperature variations which need to be adjusted using a



(a) R-flop circuit



(b) Kinniment's metastability based TRNG

Figure 1.14: Entropy extraction in Kinniment's metastability based TRNG.

circuit-switched capacitor followed by a DFF. The entropy extractor block is illustrated in Figure 1.14b [KC02]. Then a post-processing circuit composed of an LFSR generates the final output sequence of random bits. Authors of [VSB10] and [TBM07] proposed also an analog metastability based TRNG. Such analog designs of TRNGs are not portable on reconfigurable devices.

1.4.1.2 Intel's Ivy bridge metastability based TRNG

Recently, Intel introduced its 2^{nd} generation of TRNG which is based on metastability. It is published under the US patent 8489660 [HCG+13]. In 2012, the CRI published an evaluation of the new Intel's Ivy bridge TRNG in [HKM12]. Figure 1.15 illustrates the entropy extraction block of this TRNG. The core of the entropy source is the metastable RS latch. The final state to which the RS latch resolves depends on the thermal noise. The RS latch settling state is controlled by the feedback trimming circuit which adjust the amount of charge at the PMOS inputs of the RS latch circuit. Based on how the latch resolves, a fixed amount of charge is drained from one capacitance to the other to maintain the metastable state. The buffering circuit detects when the RS latch settles down to a stable state and stores the random bit. Finally *data_out* is sampled by the *clock_out* to generate a random bit. The final DFF output represents the TRNG raw random output.

The entropy source output feeds an AES conditioner which is used as post-processing. However, the raw entropy source output is not accessible to the user. The post-processed output delivers random number at 800 Mbps and passed the NIST SP900-22 statistical test suite. The user can only access to the post-processed output.

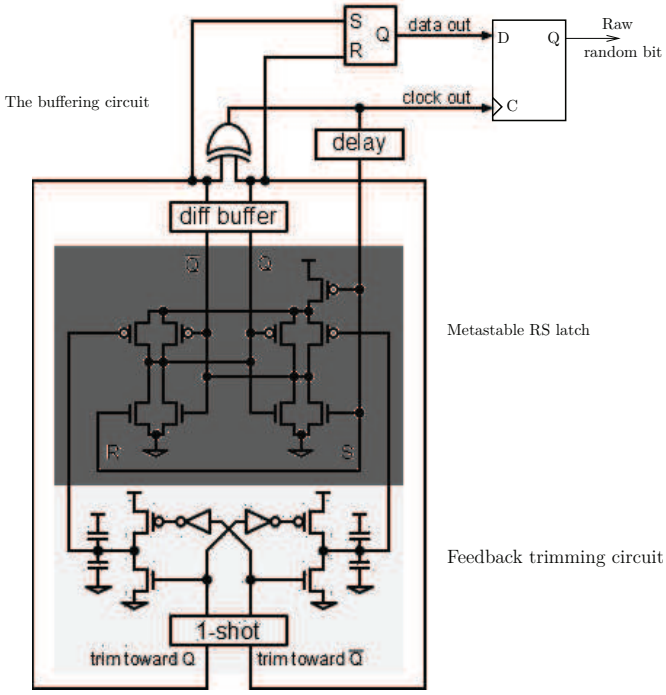


Figure 1.15: Entropy source block of Intel's metastability based TRNG.

1.4.2 BRAM based TRNG

The principle of the Block RAM (BRAM) based TRNG is to write simultaneously a logic one and a logic zero on the same memory cell. The BRAM based TRNG exploits writing conflict on a dual-port BRAM cells embedded in Xilinx FPGAs to generate random bits [GCS09]. Figure 1.16 illustrates the principle of the writing conflict at the data inputs of a BRAM.

The concurrent write operation makes the resulting value at the BRAM output a non-deterministic non-logic value. This is the case for some BRAM cells, however some others generate a biased result. According to [GCS09], only a few BRAM cells over the total available BRAMs are exploitable for true random bit generation. In fact, because of the clock skew caused by placement and routing asymmetries and the board heating-up, the optimal BRAM cells cannot generate as high quality as when started-up. Hence, authors propose to sweep all the BRAM cells by testing the frequency of ones constantly at run time. This test is performed over 2^{16} clock cycles for each of the potential optimal BRAMs which are of the number of 8. The design has been experienced and its randomness tested and validated on several Xilinx FPGA chips, the Virtex 2 PRO, the Spartan 3 and the Virtex 4. It needs 686 Slices and 16 BRAMs. It passes the NIST statistical tests and operates at 25 Mbps. The main disadvantages of the BRAM-based

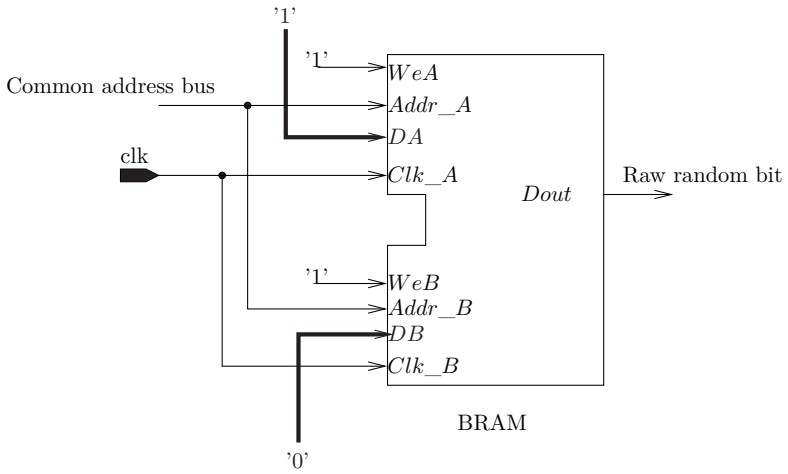


Figure 1.16: Principle of the BRAM concurrent data writing operation.

TRNG are:

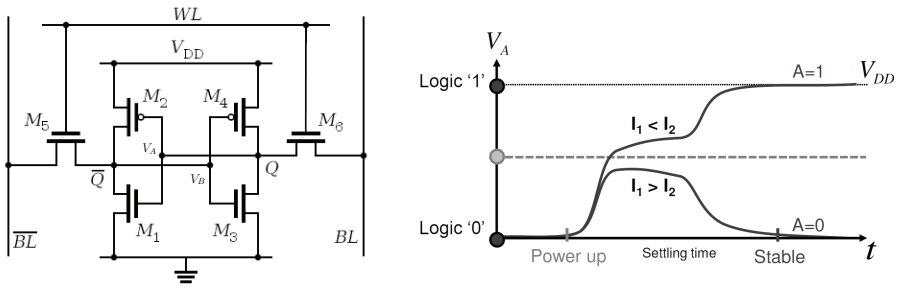
- Its dependency to BRAM blocks. In fact, BRAM are not available in all the FPGAs on the market.
- Its long start-up time 2^{16} clock cycles needed before the first random bit is released.
- Its design complexity.

According to authors analysis, this structure exploits only the phenomena of non-deterministic voltage value at the output of the BRAM since the written conflicted inputs, A and B, are constant. This is why we classify the BRAM-based TRNG in the voltage uncertainty category. However, we note that one can also force a timing violation one the switching time of A and B data signals. This would force the BRAM cell to operate in the metastable region.

1.4.3 SRAM-based TRNG

In [HBF09], Holcomb et al. propose to exploit the non-initialized state of an SRAM as source of randomness. The SRAM cell is composed of 6 MOSFET transistors: two cross-coupled inverters and two access transistors which control the read and write operations. Figure 1.17a illustrates the SRAM cell structure. At power-up the initial SRAM states are unpredictable. The process variation at SRAM manufacturing induces a difference in the transistors threshold voltage. The transistors mismatch fixes the SRAM value at power-up. In this case, at each power-up the SRAM has the same value. However, if the inverters are perfectly balanced, the SRAM cell starts in a

metastable state. Figure 1.17b illustrate the metastable behaviour of the back-to-back inverters node, V_A. The amplitude noise around the metastable state $\frac{V_{DD}}{2}$ of the back-to-back inverters makes the cell resume to either a logic one or a logic zero.



(a) SRAM 6 transistor cell (b) Random initial state of SRAM at power-up.

Figure 1.17: Principle of SRAM-based TRNG.

The true randomness generated by the SRAM cells results from the voltage drift uncertainty impacting some SRAM cells that do not present fabrication process mismatch. Authors of [HBF09] exploit 256 bytes of non-initialized SRAM to produce 128 random bits. The sequence generated by the SRAM-based TRNG passes only 3 over 15 of the NIST statistical test suite. In comparison with RO-based TRNGs, these RAM-based TRNGs present lower bit rate and are more demanding in terms of resources. However, such TRNG designs can exploit the existent RAM circuitry. In the next section, we present the state-of-the-art of TRNGs which exploit both phase and voltage noise.

1.5 Phase-Voltage Noises based TRNGs

In this section, we present four TRNG designs that exploit both time and voltage uncertainties [KS08, DGH09, VHK12, VD10].

1.5.1 Diodes based TRNG

Killmann and Schindler presented a noisy diodes based TRNG in [KS08]. Figure 1.18 illustrates its architecture. The diode based TRNG exploits both shot noise induced by the Zener diodes and jitter at the output signals of the T-Flip-Flop and the sampling clock. A Zener diode generate 1 mV of voltage uncertainty. The output of the operational amplifier feeds a Schmitt trigger. The mean voltage of the Op.Amp. output corresponds approximately to the middle of the two threshold voltage values of the Schmitt trigger. The

Schmitt trigger output consists of consecutive low and high voltage levels. The time length of these signals is random.

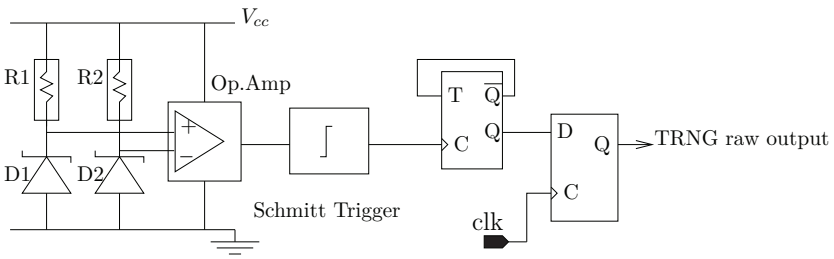


Figure 1.18: The generic design of noisy diodes based PTRNG of Killmann and Schindler

Each 0 to 1 transition at the output of the Schmitt Trigger activate the T-flip-flop (TFF). The TFF output is sampled by an external clock. The final DFF output represents the TRNG raw random bit.

1.5.2 Open-Loop delay chains based TRNG

Metastability based TRNGs take advantage from metastable states in bi–stable cells. Several analog and digital TRNG designs that extract randomness from metastability have been proposed in [KC02, DGH09, VSB10]. A way to create metastability in any bi–stable cell is to toggle its data input simultaneously with the sampling edge of the clock. In this case, the data input toggles in the interval $[t_{setup}, t_{hold}]$ leading possibly to a metastable state. A very simple implementation of TRNGs could then be done by sampling the clock itself. This approach has been introduced in [KC02] with a controlled custom analog cell to force the metastable state. Later, an FPGA implementation of the metastability based TRNG principle was introduced in [DGH07]. In this design, the delay between data and clock signals is controlled so that the data edge catches the clock edge. The higher is the definition of the data-to-clock delay, greater the chances are for the data to switch in the metastable region. In order to do so, a chain of multiplexors is used in order to control the data-to-clock delay at the latches input. Figure 1.19 illustrates the principle of delay chains tuning of the Open-Loop TRNG [DGH09].

1.5.3 Transient effect ring oscillator based TRNG

The transient effect ring oscillator (TERO) based TRNG exploits phase and voltage noises. It extracts randomness from temporary oscillations when a bi–stable cell resolves from a temporary oscillating phase to a stable state. The TERO loop cell, as illustrated in Figure 1.20a, is composed of two XORs and

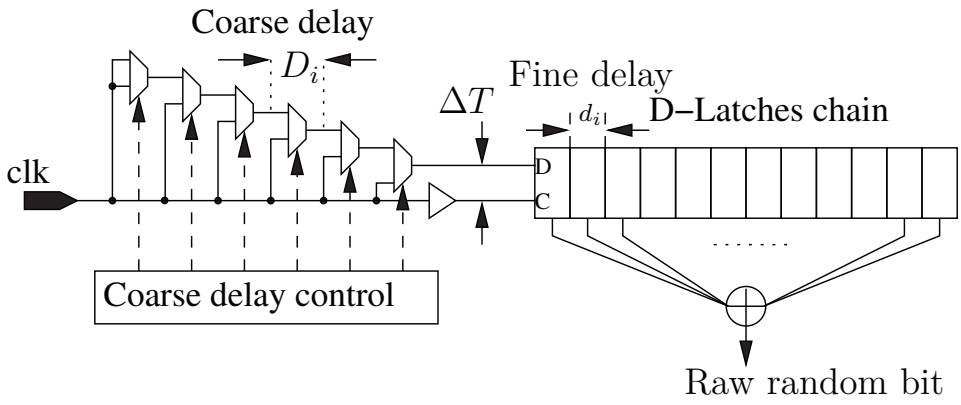
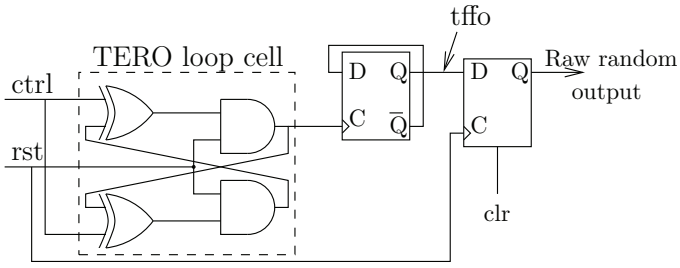


Figure 1.19: Delay tuning of open-loop delay chains based TRNG.

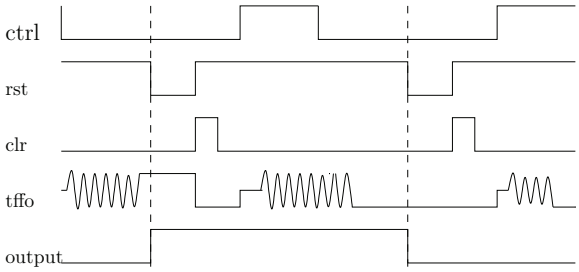
two ANDs. This block behaves as a flip-flop. When ctrl equals to 1 or 0 the TERO structure enters a temporarily oscillatory state. It begins to oscillate at each falling or rising edge of the ctrl signal. This is the transient mode. Then the output settles down to a stable logic state. Figure 1.20b illustrates the timing diagram of the TERO-based TRNG. The number of oscillation in the oscillatory phase depends on the level of noise. The random behavior of the TERO-based TRNG lies in the random number of oscillations in the transient mode. The oscillatory phase is introduced periodically in accordance with the timing diagram of Figure 1.20b. The number of oscillations are counted in T-Flip-Flop (TFF) which stops counting when the oscillations stop. On each rst falling edge, the final DFF samples the random value at the TFF output such as illustrated in the timing diagram of Figure 1.20b. The TERO-based TRNG was experienced in Spartan 3. A TERO cell occupies only one CLB. Authors tested two TERO cells where the output is the XOR of both outputs. This combination improved the statistical tests results.

1.5.4 Metastable ring oscillator based TRNG

Vasyltsov et al. propose a TRNG which extract randomness from both jittery ring oscillators and metastable states called the Meta-RO TRNG [VHKK08]. The structure of the Meta-RO TRNG is illustrated by Figure 1.21. The multiplexers are used to choose between two different phases. In the first phase, the fed back multiplexor behaves like a bi–stable cell. In this phase, the inverter loop output oscillates around a non-logic value that is neither a zero nor a one. This instable outputs set the local signals in the global ring oscillator. In the second phase, called the generation phase, the inverters are connected in a loop to form a free running ring oscillator. The ring oscillator begins to oscillate from this instable state. The ring oscillator output is sampled with the delayed clock system to generate a random bit.



(a) TERO structure



(b) TERO timing diagram

Figure 1.20: Transient effect ring oscillator based TRNG.

Hence, in the Meta-RO structure, the randomness comes from:

- the voltage value uncertainty at the output of the inverters.
- the phase uncertainty in the ring oscillator

The Meta-RO TRNG is US patented by Samsung [VHK12](US 8341201). Simulated random sequences of Meta-RO TRNG targeting CMOS 65nm ASIC technology pass only the statistical tests of procedure A of the AIS-31 with a throughput of 140 Mbps. Authors present also an investigation of the Meta-RO TRNG on Xilinx FPGA which pass the FIPS 140-1 tests and only the statistical tests of procedure A of the AIS-31. However, authors do not present results of NIST battery of statistical tests. The FPGA design presents a throughput of 35 Mbps after post-processing. The advantages of the Meta-RO TRNG are its very low design complexity and its portability.

Even though several TRNGs have been introduced in each category, given our current knowledge, no digital versions of TRNGs exploiting both phase and voltage noises and robust against frequency injection attacks have been prototyped in ASIC technology. In this Ph.D work, we propose to exploit both voltage and phase noises to extract randomness while targeting both FPGA and ASIC technologies.

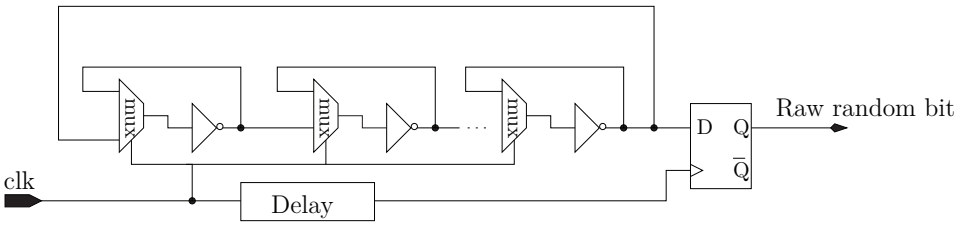


Figure 1.21: Structure of the Meta-RO-TRNG.

1.6 Conclusion

In this chapter, we give the design guidelines of the TRNGs four main blocks according to the AIS-31 standard for TRNGs certification. First, we presented the main sources of noise in silicon technologies then we discuss the methods of noise extraction and the digital elements used in order to sample the noise and generate a random bit. Then, we present the specifications and guidelines of TRNG online testing and post-processing. Besides, we introduce a new TRNG classification criteria and present the state-of-the-art of on-chip true randomness generation methods. The classification we propose in this chapter helps us study and identify the source of noise exploited in each method proposed in the literature. To evaluate a TRNG quality of randomness and robustness, some common and standard methods exist. In the next chapter, we present the existing methods namely stochastic modeling, statistical standard tests and physical evaluation.

Chapter 2

Evaluation of True Random Numbers Generators

2.1 Introduction

In order to evaluate the quality of TRNG randomness, standard statistical tests are applied to the generated sequence. Evaluating a TRNG includes not only the statistical evaluation of the quality of randomness but also its self-testing ability and its robustness against working conditions and malevolent perturbations. A TRNG should also be stochastically modeled to allow its evaluation at design time. This chapter addresses respectively the state-of-the-art of three important aspects towards TRNG certification: the statistical evaluation, the stochastic modeling and the physical evaluation. First, in Section 2.2, we present existing TRNGs evaluation standards in terms of quality of randomness. Besides, we propose a thorough study of a specific standard certification scheme for TRNGs which is the AIS-31 evaluation methodology. Second, we present in Section 2.3 the state-of-the-art of stochastic modeling of TRNGs. Third, Section 2.4 deals with the vulnerabilities of on-chip TRNGs and presents the state-of-the-art of TRNGs physical attacks. Finally, in Section 2.5, we draw a comparison between the existing on-chip TRNGs in terms of complexity, throughput and physical vulnerabilities.

2.2 Statistical Evaluation

This section is undertaken in the regard of studying the AIS-31 standard. We begin with presenting the statistical tools and the existing standards of evaluation that allow to test the quality of randomness of a TRNG. Then we propose to study the design and statistical evaluation requirements of the AIS-31 standard. Through the thorough description of the AIS-31 evaluation methodology, we propose to study the feasibility and complexity of embedding some of the AIS-31 statistical tests in hardware.

2.2.1 Overview of statistical evaluation standards

There are three practical and open source standard batteries of tests to evaluate the quality of randomness namely the FIPS 140-1 [Sta94] tests,

the NIST (National Institute of Standards and Technology) battery of tests SP800-22 [RSN+10] and finally the AIS-31 procedure A and B tests [KS11] issued from the BSI, the German IT security certification authority. Two other statistical tests batteries exist but are not standardized such as the TEST01 [LS07] and the DieHard [EB07] batteries of tests. However, the AIS-31 [KS11] and the NIST SP800-90B draft [NIS12] standards propose an evaluation methodology dedicated to test Physical True Random number Generators a part from the batteries of tests.

Truly generated random numbers have to fulfill these two properties: First, they have to be uniformly distributed. It means that all the numbers are equally likely to be observed. Second, the generated numbers must be inpredictible in the way that we cannot predict the number to come from the current generated number. Third, they have to be independent. Standard statistical tests check if the random number generator under test respects the above discussed requirements. A TRNG should fulfill the forward secrecy *i.e.* if you know the generated sequence $(S_0, S_1, .., S_n)$ you cannot predict the sequence S_{n+1}. The length of the sequence to be tested varies depending on the battery of tests performed. Each of the standard tests batteries cited above defines its own permitted intervals of test parameters for each test.

In the next section, we give a detailed overview of the AIS-31 evaluation methodology. We propose to study the AIS-31 standard in particular for two reasons. The first one is to study the statistical tests procedure to help propose a lightweight hardware online test. Understanding the AIS-31 statistical requirements of a true random sequence would help us to propose the appropriate post-processing to correct the TRNG eventual bias. The second reason is related to the fact that the AIS-31 standard is a complete evaluation methodology that helps setting the specifications of a robust TRNG that fulfill the common criteria requirements. Then, we give a brief state-of-the-art of TRNG hardware online tests.

2.2.2 AIS-31 standard certification

2.2.2.1 Statistical tools and deifinitions

- **Uniform distribution**
  A random variable X is uniformly distributed if all the possible outcomes of X, P, are equally likely to be observed. It intends that all the outcomes have equal probabilities to happen such that $p_x[i]$ verifies Equation (2.1).

$$p_x[i] = \frac{1}{P} \ \forall i \in [1, P] \tag{2.1}$$

  In the following, the TRNG output is our random variable X.

- **Chi square test χ^2**
  The χ^2 test is a statistical tool that does not directly answer this question, however it will give a decision whether the distribution of the random variable X is approximately the same as an ideal RNG with a risk of mistaking α %. The χ^2 test is commonly used to compare observed data with data we would expect for an ideal system.

$$\chi^2 = \sum_{i=0}^{df} \frac{\left(\nu_{obs}[i] - \nu_{exp}[i]\right)^2}{\nu_{exp}[i]} \tag{2.2}$$

  where $\nu_{obs}[i]$ denotes the frequency of i observed at the output of the TRNG. While $\nu_{exp}[i]$ represents the expected frequency of i for a uniform distribution as given by Equation (2.3).

$$\nu_{exp}[i] = \frac{1}{P} \tag{2.3}$$

  The χ^2 *test* defines two parameters:

  - First, the degree of freedom denoted *df*. *df* is equal to the number P of the possible states of a random variable X that are free to vary minus one such as given in Equation (2.4).

$$df = P - 1 \tag{2.4}$$

  - Secondly, the rejection level, α. Depending on α and *df*, a table, called the contingency table, is given by the χ^2 *test*. The rejection probability defines the level of rejection of the comparison between the expected distribution and the observed one. In the AIS-31, we confront a TRNG sequence with a sequence generated by an ideal TRNG which generates equally distributed numbers. The rejection probability, α, for an ideal RNG is 9.6 10^{-7} which is given by the Central Limit Theorem. In the AIS-31 standard, this value is approximated to 10^{-6} according to §192 of [KS11]. The contingency table of the χ^2 *test* is given in Table 5.13 in the Appendix.

Figure 2.1 summarizes the TRNG standard structure towards AIS-31 certification scheme as it is described in the AIS-31 Evaluation Methodology for True (Physical) Random Number Generators in §263 [KS11].

The TRNG system of Figure 2.1 represents the Target Of Evaluation (TOE) of the AIS-31 standard. The AIS-31 standardisation scheme defines two levels of certification of the TRNG TOE: PTG.1 and PTG.2. Table 2.1 gives a summary of the requirements of the AIS-31 levels of certifications.

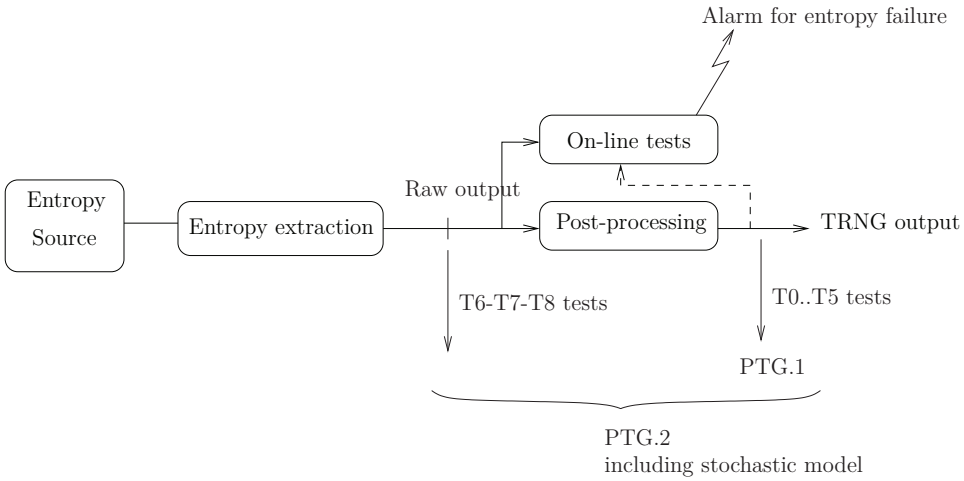


Figure 2.1: Standard structure of TRNGs as described in the AIS-31 standard certification scheme.

Using AIS-31 statistical tests helps us find out how much the tested sequence is close to a uniform distribution. Here, we present the statistical tools and the background of AIS-31 statistical tests. This section present a thorough study of the AIS-31 standard. The aim of describing each test is:

- to explain the error margins given in the standard,
- to study the complexity of each test and
- to propose a relevant lightweight online tests.

2.2.2.2 PTG.1 certification class

This class requires from the post-processed output of a TRNG to pass T0 to T5 statistical tests such as defined by AIS-31 standard according to §273 of [KS11]. It also requires from a TRNG to embed within the core a health online tests. These hardware embedded tests should rise an alarm in case of total failure of the source of entropy. The TRNG certified as PTG.1 class can be used for challenge response protocols and as seeds for PRNGs [Sch99]. In the PTG.1 certification class, the BSI allows to apply the statistical tests on a post-processed sequence. In the following, we give a detailed description of each statistical test. Lower and upper bound margins of errors allowed in the AIS-31 standard statistical tests are given by AIS-31 documentation for each statistical test however no explanations are given. In the following, we provide mathematical evidence and explanations of the margins imposed by the AIS-31 standard. Through this study, we propose a lightweight version of the AIS-31 test suite to embed within the TRNG chip.

Certification class	Statistical test procedure	AIS-31 Requirements
PTG.1	Test procedure A (T0-T5 tests)	- The TRNG shall embed online tests that detect a total failure of the entropy source - Online tests shall be applied at the TRNG post-processed output. - The TRNG post-processed output shall pass the test procedure A.
PTG.2	Test procedure B (T6-T8 tests)	- The TRNG shall be compliant with the PTG.1 requirements - Online tests shall be applied at the TRNG raw output. - The TRNG raw output shall pass the test procedure B - A stochastic model of the entropy source shall be provided.
PTG.3	No additional tests	- PTG.2, additionally with cryptographic post-processing - If failure of entropy source, generates the internal random numbers with a post-processing algorithm

Table 2.1: AIS-31 certification classes of Physical True Random Number Generators according to §262 of [KS11].

+ **T0 Disjointness test:**
  This test consist basically in comparing $N = 2^{16}$ words of $n = 48$ bits pair by pair. To pass this test, the words $w_1, \cdots, w_{2^{16}} \in \{0,1\}^{48}$ have to be pairwise different. To perform this test on-chip we need a memory of 600 kB.

+ **T1 Monobit test:** T1 is performed on N=20000 bits. This test is equivalent to flipping a coin 20000 times and recording the frequency of occurrences of heads and tails. If the coin is perfect, we should expect 10000 heads and 10000 tails. Let $\nu[1] = \sum_{j=1}^{N} b_j$ which corresponds to the number of bits equal to '1' in the sequence. We consider that this test is successful if almost half of the generated bits are ones and the other half are zeros. The sequence $b_1, \cdots, b_N$ passes T1 if n_1 lies in $[\frac{N}{2} - \epsilon, \frac{N}{2} + \epsilon]$.

  Let us determine ϵ. Basically, T1 test consists in applying the χ^2 test for:

  - $df = 1$, the degree of freedom. As we consider the sequence bit by bit, the possible outcomes, P, are '0' or '1',
  - $\alpha = 10^{-6}$, is the rejection level.

  The test variable used to validate the T1 test is given in §192 of [KS11] such that, T_1 follows a χ^2 distribution with $df = 1$. The χ^2 test variable is given in Equation (2.5) where $\nu[0]$ is the observed number of occurrences of '0' and $\nu[1]$ is the observed number of occurrences of

'1'.

$$\begin{aligned}
\chi^2_{T1} &= \sum_{i=0}^{df=1} \frac{\left(\nu_{obs}[i] - \nu_{exp}[i]\right)^2}{\nu_{exp}[i]} \\
&= \frac{\left(\nu[0] - \frac{N}{2}\right)^2}{\frac{N}{2}} + \frac{\left(\nu[1] - \frac{N}{2}\right)^2}{\frac{N}{2}} \\
&= \frac{\left(\nu[0] - \frac{\nu[0]+\nu[1]}{2}\right)^2 + \left(\nu[1] - \frac{\nu[0]+\nu[1]}{2}\right)^2}{\frac{N}{2}} \\
&= \frac{\left(\frac{\nu[0]}{2} - \frac{\nu[1]}{2}\right)^2 + \left(\frac{\nu[1]}{2} - \frac{\nu[0]}{2}\right)^2}{\frac{N}{2}} \\
\chi^2_{T1} &= \frac{\left(\nu[1] - \nu[0]\right)^2}{N}
\end{aligned} \tag{2.5}$$

And the condition to pass the T1 test is:

$$\chi^2_{T1} \leq \chi^2_\alpha \tag{2.6}$$

So ϵ should verify the Inequality (2.7)

$$\begin{aligned}
\frac{\left(\nu[0] - \frac{N}{2}\right)^2}{\frac{N}{2}} + \frac{\left(\nu[1] - \frac{N}{2}\right)^2}{\frac{N}{2}} &\leq \chi^2_\alpha \\
\frac{\epsilon^2 + \epsilon^2}{\frac{N}{2}} &\leq \chi^2_\alpha \\
4.\frac{\epsilon^2}{N} &\leq \chi^2_\alpha \\
|\epsilon| &\leq \frac{\sqrt{N}}{2}.|\chi_\alpha|
\end{aligned} \tag{2.7}$$

χ^2_α is equal to 23.92 for a rejection probability $\alpha = 10^{-6}$ according to the χ^2 contingency table of Table 5.13. Then from the Inequality (2.7) and for N=20000, we obtain $|\epsilon| \leq 346$ that specifies the permitted intervals in T1 test $[\frac{N}{2} - \epsilon, \frac{N}{2} + \epsilon] = [9654, 10346]$.

+ **T2 Poker test:**
This test is called the poker test because it treats numbers grouped together as a poker hand. The hands obtained are compared to what is expected using the χ^2 test. The testing method proceeds as the following: We divide N bit stream into $\frac{N}{m}$ consecutive m-bit words denoted as w_j for $j \in [1, \frac{N}{m}]$. Then we count and store the number of occurrences of the 2^m possible values i as represented in Figure 2.2. T2 is performed on N=20000 bits for m=4. Note that T2 yields to T1 for m=1.

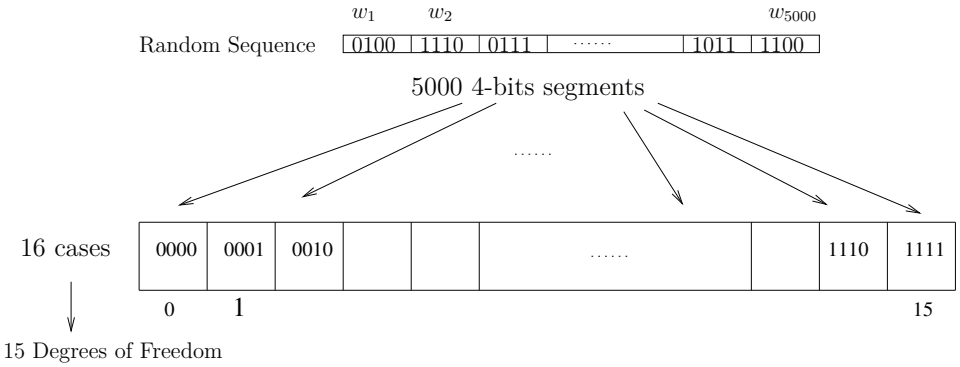


Figure 2.2: T2 Poker test representation

Let then denote $\nu[i]$ as the frequency of occurrences of the value i, where $0 \leq i \leq 15$. The function ν is defined as:

$$\begin{array}{rcccl} \nu & : & [0, 2^m - 1] & \rightarrow & [1, \frac{N}{m}] \\ & & i & \mapsto & \nu[i] = \text{card}\{j : w_j = i\} \\ & & & & where\ w_j = 8 \cdot b_{4j-3} + 4 \cdot b_{4j-2} + 2 \cdot b_{4j-1} + b_{4j} \\ & & & & for\ j = 1, .., \frac{N}{m} \end{array}$$

To compute the test variable, χ^2_{T2}, we need the following parameters:

- df is the degree of freedom and corresponds to $2^m - 1$.
- The frequency expected, $\nu_{exp}[i]$, of each 4-uplet is given by $\frac{N}{m.2^m}\ \forall\ i\ \in\ [0, 15]$.
- The sum of all the observed frequencies, $\sum_{i=0}^{2^m-1} \nu[i]$, is equal to $\frac{N}{m}$.

Equation (2.8) comes from the generic expression of the χ^2 test given in Equation (2.2).

$$
\begin{aligned}
\chi^2_{T2} &= \sum_{i=0}^{df} \frac{\left(\nu_{obs}[i] - \nu_{exp}[i]\right)^2}{\nu_{exp}[i]} \\
&= \sum_{i=0}^{2^m-1} \frac{\left(\nu[i] - \frac{\frac{N}{m}}{2^m}\right)^2}{\frac{\frac{N}{m}}{2^m}} \\
&= \frac{m.2^m}{N}\left(\sum_{i=0}^{2^m-1}\nu[i]^2 - 2.(\sum_{i=0}^{2^m-1}\nu[i]).\frac{N}{m.2^m} + \sum_{i=0}^{2^m-1}.(\frac{N}{m.2^m})^2\right) \\
&= \frac{m.2^m}{N}\left(\sum_{i=0}^{2^m-1}\nu[i]^2 - 2.\frac{N}{m}.\frac{N}{m.2^m} + 2^m.(\frac{N}{m.2^m})^2\right) \\
&= \frac{m.2^m}{N}.\sum_{i=0}^{2^m-1}\nu[i]^2 - \frac{N}{2} + \frac{N}{m} \\
\chi^2_{T2} &= \frac{m.2^m}{N}.\sum_{i=0}^{2^m-1}\nu[i]^2 - \frac{N}{m} \qquad (2.8)
\end{aligned}
$$

For $N = 20000$, $m = 4$ and $df = 15$, such as set in the AIS-31 standard, we obtain Equation (2.9).

$$\chi^2_{T2} = \frac{16}{5000}.\sum_{i=0}^{15}\nu[i]^2 - 5000 \qquad (2.9)$$

The bit sequence $b_1, \cdots, b_{20000}$ passes the poker test if the test variable, χ^2_{T2} verifies the Inequality (2.10).

$$\chi^2_{T2} \leq \chi^2_\alpha \qquad (2.10)$$

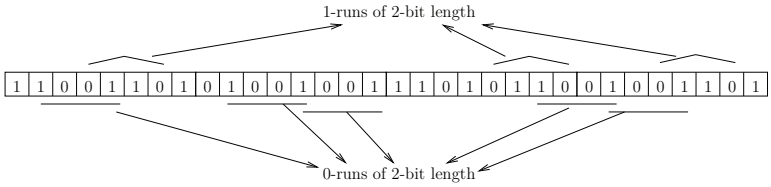
So ϵ should verify the Inequality (2.11)

$$
\begin{aligned}
\sum_{i=0}^{2^m-1} \frac{\left(\nu[i] - \frac{\frac{N}{m}}{2^m}\right)^2}{\frac{\frac{N}{m}}{2^m}} &\leq \chi^2_\alpha \\
2^m \sum_{i=0}^{2^m-1} \left(\frac{\epsilon}{\frac{N}{m}}\right)^2 & \\
\frac{m.2^{2m}}{N}\epsilon^2 &\leq \chi^2_\alpha \\
|\epsilon| &\leq \frac{N}{m.2^{2m}}.|\chi_\alpha| \qquad (2.11)
\end{aligned}
$$

χ^2_α for $df = 2^4 - 1 = 15$ is equal to 56.49 for a rejection probability $\alpha = 10^{-6}$ according to the χ^2 contingency table of Table 5.13 in the Appendix. Hence, $\chi_\alpha = \sqrt{56.49} = 7.51$. Then from the Inequality (2.11) and for N=20000 and m=4, we obtain $|\epsilon| \leq 36$ that specifies the permitted intervals in T2 test $[\frac{\frac{N}{m}}{2^m} - \epsilon, \frac{\frac{N}{m}}{2^m} + \epsilon] = [276, 348]$.

We propose to embed the T2 test as an online test. For example, we can apply the test only on $N = 1024$ and $m = 4$, then we need a 7-bit adder.

+ **T3 Runs test:**
A *run* is a sequence of identical bits (zeros or ones). T3 is performed on a sequence of $N = 20000$ bits. It counts the frequencies $\nu(N, r)$ of runs of ones of various lengths $r \in [1, 6]$ and checks for deviations from expected values as represented in the table 2.2. For example, for r=1, we count the number of occurrences of the word 010 and the number of occurrences of 101. For r= 2, we count the number of occurrencses of the word 0110 and the number of occurrences of 1001. Figure 2.3 illustrates an example of sequence of N=32 bits.



N=32, r=2	Word	Occurrences
0-run	1001	$\nu_0(32, 2) = 5$
1-run	0110	$\nu_1(32, 2) = 3$

Figure 2.3: T3 runs test example for N=32 and r=2.

The test value $\chi_{T3}(r)$ is computed $\forall r \in [1, 6]$ (2.12) where $\nu(N, r)$ expected equals to $\frac{N}{2^{r+2}}$ $\forall r \in [1, 6]$ for an ideal RNG. The observed frequency is denoted $\nu(N, r)_{obs}$.

$$\begin{aligned} \chi^2_{T3}(r) &= \frac{(\nu(N,r)_{obs} - \nu(N,r)_{exp})^2}{\nu(N,r)_{exp}} \\ \chi_{T3}(r) &= \frac{\nu(N,r)_{obs} - \nu(N,r)}{\sqrt{\nu(N,r)}} \\ \chi_{T3}(r) &= \frac{\nu(N,r)_{obs} - \frac{N}{2^{r+2}}}{\sqrt{\frac{N}{2^{r+2}}}} \end{aligned} \tag{2.12}$$

The test condition to pass T3 $\forall r \in [1, 6]$ is given in 2.13.

$$\begin{aligned} \chi^2_{T3}(r) &\leq \chi^2_\alpha \\ |\epsilon(r)| &\leq |\chi_\alpha| . \sqrt{\frac{N}{2^{r+2}}} \end{aligned} \tag{2.13}$$

By applying the Inequality (2.13) for N=20000 bits and for $\alpha = 10^{-6}$ and $r = 1$, $|\epsilon(1)| \leq \sqrt{2500 * 23.92} = 244$. Then, T3 is considered successful if for each run length r if the frequency of occurrences $\nu(20000, r)$ lies in the corresponding permitted intervals of Table 2.2.

Run length	**Boundary values for χ^2 test**	**Permitted interval**
r	χ^2_α	$[\frac{N}{2^{r+2}} - \epsilon(r), \frac{N}{2^{r+2}} + \epsilon(r)]$
1	23.92	[2256,2744]
2	30.66	[1055,1445]
3	35.88	[502,748]
4	40.52	[233,402]
5	44.81	[90,223]
6	48.86	[17,139]

Table 2.2: Table of the permitted intervals in T3 Runs test for $\alpha = 10^{-6}$ and $N = 20000$ bits.

If for any $r \in [1, 6]$ the observed frequency of occurrences $\nu(20000, r)$ lies outside the permitted intervals of Table 2.2, it means that the number of runs occurred in the tested sequence is far from what is accepted.

+ **T4 Long run test:**
The long run test is an extension of the runs test T3. T4 checks whether there is an uninterrupted sequence of identical bits (*i.e.*, either all zeroes or all ones). The test is successful if there is no long run of length $LR = 34$ or longer in a sequence of 20000 bits.

The 4 cited above tests, T1, T2, T3 and T4, are nearly the same tests of FIPS 140-1 standard [Sta94] but with narrower permitted intervals as reported in Table 2.3.

+ **T5 auto-correlation test:**
T5 is performed on a sequence of N=10000 bits. This test looks for

Tests \ Permitted intervals	FIPS 140-1	AIS-31
Frequency test (T1)	[9726,10274]	[9655,10345]
Poker test (T2)	[2.16,46,17]	[1.03,57.4]
Run test (T3)	[2343,2657]	[2326,2674]
Long run test (T4)	26 bits	34 bits

Table 2.3: Table of comparison between FIPS 140-1 tests and AIS-31 tests

correlation between s and a τ-shifted version of it.

$$for\ \tau \in \{1, ..., 5000\}, Z_\tau = \sum_{i=1}^{5000} (b_i \oplus b_{i+\tau}) \tag{2.14}$$

Similarly to the test T1 and T2, we use the χ^2 test by computing the following test variable of (2.15) for each $\tau \in [1, \frac{N}{2}]$:

$$\chi_{T5}(\tau) = \frac{Z_\tau - \frac{(N-\tau)}{2}}{\sqrt{\frac{N-\tau}{2}}} \tag{2.15}$$

The generic condition applied in this test is the Inequality (2.16).

$$\begin{aligned} \chi_{T5}(\tau) &\leq \chi_\alpha \\ \frac{Z_\tau - \frac{(N-\tau)}{2}}{\sqrt{\frac{N-\tau}{2}}} &\leq \chi_\alpha \\ |\epsilon| &\leq \sqrt{\frac{N-\tau}{2}} \cdot \chi_\alpha \end{aligned} \tag{2.16}$$

By applying the Inequality (2.16) for N=20000 bits and for $\alpha = 10^{-6}$, $|\epsilon| \leq \sqrt{2500 * 23.92} = 174$. Then, the tested sequence passes the test T5 if the test variable Z_τ fulfills 2.17.

$$\forall \tau \in \{1, ..., 5000\}, 2326 < Z_\tau < 2674 \tag{2.17}$$

In the nex sub-section, we present the second AIS-31 class of certification, PTG.2.

2.2.2.3 PTG.2 certification class

The PTG.2 class of certification requires from the TOE TRNG, first, to be compliant with the class PTG.1 then to pass also Tests T6, T7 and T8 not only, but require from the TRNG designer to provide a stochastic model of the TRNG. PTG.2 certified TRNGs can be used in more sensitive cryptographic protocol:

- Session keys generation for symmetric encryption algorithms
- Bits padding
- Seed generation which can be used for higher classes of certification of PRNGs as specified in [Sch99].

The PTG.2 class includes the first class and adds more complex tests and requirements about online tests and entropy source modeling. Unlike PTG.1 tests, PTG.2 tests must be applied on the raw output of the TRNG without any post-processing.

+ **Test T6: Multinomial Distributions Test**
  T6 test is divided in two sub-tests, T6a and T6b. Both tests are performed on a sequence of N=100000 bits.

  – **T6a**
  This test performs the same test procedure as in T1 test (*i.e.* records the frequency of occurrences of '1') although on 100000 bits this time. T6a is successful if $S = \sum_{i=1}^{100000} b_i$ respects the condition (2.18).

$$|\frac{S}{N} - 0.5| < 0.025 \tag{2.18}$$

  – **T6b**
  The tested sequence is split by pairs $\{w_{2i+1}, w_{2i+2}\}$. Then we get $\frac{N}{2} = 50000$ pairs which are sub-classified in 2 groups according to the value of the first bit w_{2i+1}. If $w_{2i+1} = 0$, the i^{th} pair belongs to the group G_0 else w_{2i+2} belongs to G_1. Let us note S_0 as the cardinal of the group G_0 and S_1 as the cardinal of the group G_1. The TRNG passes T6b if condition (2.19) is fulfilled.

$$|\frac{S_0 - S_1}{N}| < 0.02 \tag{2.19}$$

+ **Test T7: Test for homogeneity**

  – **T7a**
  In T7a, the tested sequence of $N = 100000$ bits is split into

3-bit words designated as $\{w_{3i+1}, w_{3i+2}, w_{3i+3}\}$. Then, they are sub-classified in 4 groups according to the value of the first pair $\{w_{3i+1}, w_{3i+2}\}$. If $\{w_{3i+1}, w_{3i+2}\} = 00$, the i^{th} word belongs to the group G_{00}. If $\{w_{3i+1}, w_{3i+2}\} = 01$, the i^{th} word belongs to the group G_{01}. If $\{w_{3i+1}, w_{3i+2}\} = 10$, the i^{th} word belongs to the group G_{10}. If $\{w_{3i+1}, w_{3i+2}\} = 11$, the i^{th} word belongs to the group G_{11}. The frequency of occurrences of the bit '1' respectively '0' in the sub pairs of groups $G_{00}, G_{01}, G_{10}, G_{11}$ is listed producing $\nu^0_{00}, \nu^0_{01}, \nu^0_{10} and \nu^0_{11}$ respectively ν^1_{00}, ν^1_{01}, ν^1_{10} and ν^1_{11}. The TRNG passes T7a if conditions (2.20) and (2.21) are fulfilled.

$$\frac{(\nu^0_{00} - \nu^0_{01})^2}{\nu^0_{00} + \nu^0_{01}} + \frac{(\nu^1_{00} - \nu^1_{01})^2}{\nu^1_{00} + \nu^1_{01}} < 15.13 \tag{2.20}$$

$$\frac{(\nu^0_{10} - \nu^0_{11})^2}{\nu^0_{10} + \nu^0_{11}} + \frac{(\nu^1_{10} - \nu^1_{11})^2}{\nu^1_{10} + \nu^1_{11}} < 15.13 \tag{2.21}$$

– **T7b**

Unlike T7a test T7b splits the tested sequence of $N = 100000$ bits in words of 4 bits $\{w_{4i+1}, w_{4i+2}, w_{4i+3}, w_{4i+4}\}$. Words are then classified along 8 $(= 2^3)$ groups : $G_{000}, G_{001}, .., G_{111}$ according to the value of the 3 first bits $\{w_{4i+1}, w_{4i+2}, w_{4i+3}\}$. Conditions of validation of T7b are the same as in T7a. The TRNG passes T7b if condition (2.22) is fulfilled for all the pairs of Groups G_w, G_{w+1} where $w \in \{000, 010, 100, 110\}$.

$$\frac{(\nu^0_w - \nu^0_{w+1})^2}{\nu^0_w + \nu^0_{w+1}} + \frac{(\nu^1_w - \nu^1_{w+1})^2}{\nu^1_w + \nu^1_{w+1}} < 15.13 \tag{2.22}$$

+ **Test T8 : Entropy test**

T8 applies basically the test procedure of the enhanced version of the Maurer's Universal Test in accordance with [Cor99]. Considering a sequence of length N, test T8 splits the sequence in Q+K disjoint L-bit words $\{w_1, .., w_{Q+K}\}$. A_N specifies the minimal distance of a bit from its predecessor with the same value. A_N is defined in Equation (2.23). For example, for the sequence "1000100001", A_{10} is equal to 5.

$$A_N = \begin{cases} N & \text{if} \quad card\{i : w_N = w_{N-i}\} = 0 \\ \min\{\, i : w_N = w_{N-i}\} & \text{else} \end{cases} \tag{2.23}$$

This test allows to estimate the global entropy of the generated sequence. In fact, [Cor99] presents a variant of the Maurer's Universal Test. In [Cor99], authors demonstrated that the test function, T_8 of Equation (2.25), is asymptotically equal to the Shanon's entropy of the

tested sequence. In AIS-31 standard the entropy test is applied for L = 8, Q = 2560 and K = 256000. The maximum entropy H_{max} for equally distributed probabilities $p_i = \frac{1}{2^L}$ for $i \in [1, L]$ is equal to 8 according to Equation (2.24).

$$H = -\sum_{i=0}^{2^L-1} p_i.log_2(p_i) \tag{2.24}$$

A source of entropy is ideal if the random sequence it generates provides full entropy *i.e.* at least $(1-\epsilon).L$ bits of entropy, where $0 < \epsilon < 2^{-64}$ [NIS12]. The test variable, t_8, of Equation (2.25) has to be approximately equal to $H_{max} = L$.

$$T_8 = \frac{1}{K} \cdot \sum_{n=Q+1}^{Q+K} g(A_n) \text{ where } g(i) = \frac{1}{ln(2)} \sum_{k=1}^{i-1} \frac{1}{k} \tag{2.25}$$

The test T8 is considered successful if the test variable t_8 is greater than $7,976$. When a TRNG passes T8, this would tell that the entropy source is good enough that the generated output can be safely used to seed a block cipher of an 8 bits key length.

Table 2.4 summarizes the minimum length required of the sequence to be tested for each AIS-31 test.

AIS-31 tests	Length in bits
T0	$2^{16} * 48 = 3145728$
T1-T5	$257 * 20000 = 5140000$
T6a-T6b-T7a-T7b	100000
T8	7200000

Table 2.4: Minimum length required for each AIS-31 test

Aside from the online tests, PTG.2 class includes also the mathematical model of the physical source which must be provided to the evaluator for two main objectives: First, identify the physical parameters which impact the entropy of the source of noise. The lower bound of the entropy would then help verifying the expected behavior of the TRNG. The stochastic model of the TRNG helps set the specifications of the online tests. Second, the stochastic model would help evaluating the quality of the TRNG output at design time. The evaluator would be able then to test the robustness of the TOE against malevolent or environmental perturbations. It is also important to note that the applicant should perform statistical tests of both PTG.1 (T0

to T5) and PTG.2 (T6 to T8) classes respectively on the post-processed and the raw output of the TRNG for all relevant environmental conditions. These added requirements have been recently published by the BSI in the form of a note for the TRNG designer application towards AIS-31 evaluation [HG13].

2.2.3 Online tests

After the specifications of the AIS-31 tests, we present the state-of-the-art of the TRNG online tests. We propose to describe the hardware online tests proposed by Intel in [HKM12]. Intel proposes a simple lightweight HW online tests for the novel Ivy bridge TRNG. It simply counts the number of occurrences of certain bit pattern in a sequence of 256 bits. The test is validated if the number of occurrences lies within the allowed bounds presented in Table 2.5.

Bit pattern	Bounds
1	[109,165]
01	[46,84]
010	[8,58]
101	[8,58]
0110	[2,35]
1001	[2,35]

Table 2.5: Online tests bounds for 256-bit sequences.

According to [Ham12], the CRI report of Intel's evaluation, the bounds were determined empirically by Intel. However, they are close to what an ideal random generator would generate. In fact, the probability that a random sample from a uniform distribution fails these online tests is about 1%.

Santoro et al. recently proposed an online embedded version of the FIPS 140-1 tests (which are the same as T1-T4 of AIS-31) in [SSR09] using only 482 LUTs of a Xilinx Virtex 5 FPGA. Since the FIPS statistical tests are not very much demanding in terms of resources compared to the AIS-31 tests, they propose to embed a hardware implementation of the FIPS tests. We propose to reduce the length of the tested sequence. As a consequence the TRNG start-up time is considerably reduced. Hence, as a simple TRNG start-up test, we propose to apply a light hardware T1 and T3 tests at the output of the TRNG where N equals to 1024. So, a 10 bit adder is needed. According to Equation (2.7), for N=1024 and $\chi_\alpha = 23.92$, ϵ_{T1} would be equal to 77. According to Equation (2.13), for N=1024 and $\chi_\alpha = 23.92$ and r=1, $\epsilon_{T3}(1)$ would be equal to 55. In this HW T3 test, we need to add on-the-

Tests	Resources
T1	log_2N bits adder 2 * log_2N bits comparators
T2	$log_2(\frac{N}{m})$ bits adder 2 * $log_2(\frac{N}{m})$ bits comparators $log_2(\frac{N}{m}) \times log_2(\frac{N}{m})$ bits multiplier
T3	12 *log_2N bits adder 24 * log_2N bits comparators

Table 2.6: Complexity of the hardware online AIS-31 tests for a generic sequence length.

fly the number of ones and the number of 1-run and 0-run of six different lengths. We note that we keep the same rejection probability $\alpha = 10^{-6}$ as in the AIS-31 standard. Hence, if no alarm is triggered by the HW T1 and T3 tests, we conclude that the generated sequence is considered approximately close to an ideal RNG for a significance level of 10^{-6}.

Apart from statistical tests evaluation, the TRNG designer should develop a stochastic model to evaluate the TRNG at design time. In the next section, we present examples of stochastic models of TRNGs.

2.3 TRNGs Stochastic Modeling

We need a model to estimate the minimum entropy provided by the output of the TRNG. A stochastic model of the TRNG aimed at its evaluation for Common Criteria certification should be provided according to §62 of [KS11]. In such a model, the designer has to identify the design parameters that impact the quality of randomness at the output of the TRNG. The TRNG designer can then tune those model parameters to enhance the quality of the TRNG. Stochastic models of a PLL-based TRNG, a noisy diodes physical TRNG and a floating-gate-based TRNG were introduced respectively in [cDFF06], [KS08] and [XHA08]. Killmann and Schindler published in [KS08] a mathematical formulation of the entropy lower bound of the diodes based TRNG. The stochastic model was then confronted to experimental results.

Wold and Petrovic [WP11] presented also a behavioral model of a RO-based TRNG. They studied the probability of hitting the jittery region while varying the TRNG design parameters namely the RO number, the sampling frequency and the number of inverters used in the ROs. The proposed model was simulated in Matlab. Modeling results has shown that:

- the higher the standard deviation of the ring oscillator's jitter σ_j is,

higher the entropy of the TRNG output is.

- the lower the sampling frequency is, higher the entropy of the TRNG output is.

However, this behavioral model has never been confronted to any experimental result of the RO-based TRNG design implemented in an Altera Cyclone II FPGA [WT08]. In addition, the results of this behavioral model lacks a mathematical formulation of the entropy lower bound of the TRNG output.

Simka et al. also presented a stochastic model of the PLL-based TRNG [cDFF06] which we described in Section 1.3.4. The latter is an enhancement of the PLL-based TRNG design introduced by Fischer et al. in [FDcB04]. In [cDFF06], authors established the probability of the TRNG output as a function of the division factor of the PLL and the jitter standard deviation. The relevance of the mathematical formulation of the TRNG entropy has been experimentally verified using random sequences generated from an Altera Stratix FPGA device.

Several other stochastic models of TRNGs have been studies namely in [SMS07, KS08, XHA08, HTBF14]. Despite the fact that several digital TRNGs exploiting metastability have been proposed in the recent years [DGH07, DGH09, MKD11, HI12], a stochastic model of such designs lacks in the literature. In fact, the issue of stochastic modeling is, to our best knowledge, not yet treated by the scientific community with regard to randomness generation using metastability. Once the stochastic modeling process and statistical tests evaluation is done, we need to evaluate a TRNG in terms of physical robustness. In the next section, we present the state-of-the-art of physical vulnerabilities and attacks on TRNGs.

2.4 TRNGs Attacks and Vulnerabilities

2.4.1 TRNGs vulnerabilities identification

Crypto-systems are vulnerable if the key generation process is threatened. The above cited statistical tests assure the unpredictability of the sequence making the TRNG robust against a mathematical attack. However, the implementation of the TRNG itself can make it vulnerable against environmental or malevolent perturbations such as cryptanalysis or invasive attacks. Figure 2.4 summarizes the vulnerable points of a TRNG. We categorize TRNGs potential vulnerabilities in two main attacks types:

- Passive attacks where the electromagnetic (EM) emanation or the power consumption can provide the attacker information about the TRNG such that operating frequency [BBAF13]. This type of threat is represented by an external-oriented arrow. In Figure 2.4, attacks (3) and (8) illustrates this category of attacks.

- Vulnerability to active attacks which can disturb the TRNG behavior towards fault creation or total inhibition of the source of entropy. This type of attacks are represented by bottom-oriented arrows. This category of attacks can be sub-categorized in two applicable attack method:
  - Non-invasive using a permanent or transient influence of the operating conditions such as:
    * Under-power the circuit core or signal superimposing on power supply voltage [KW10, Sou12, SCDEV13, Joi13].
    * Over-heat the circuit [SCDC$^+$11, KW10].
    * Over-clock the TRNG.
    * Apply clock glitches.

    In Figure 2.4, label (1) and (2) illustrates this category of attacks.
  - Invasive by physical manipulation of the TRNG circuitry such as forcing the TRNG output or the alarm hardware online test output to a zero by means of laser injection. In Figure 2.4, labels (4), (5), (6) and (7) illustrates this category of attacks. To our best knowledge, these kind of attacks have never been experienced on a TRNG. Another physical manipulation is the frequency injection on the power pad of the TRNG. In Figure 2.4, attack label (2) illustrates this category of attack. There have been two publication of this kind of attacks on RO-based TRNGs.

Table 2.7 summarizes the potential vulnerabilities of a TRNG.

Vulnerability label	Attack type	Threat	Reference
(1),(2)	Environmental perturbations	Bias TRNG output	[SCDC$^+$11, KW10, SSR09]
(3)	DFA	Operating frequency detection	[BBAF13, Sou12]
(4),(7)	FIA	Force or bias TRNG output	[SCDEV13]
(5)	EMIA	Bypass online tests	[MM09, BBA$^+$12]
(6)	FIA	Online tests alarm stuck to 0	-
(8)	SPA	Detect TRNG output by EM or power analysis	-

Table 2.7: Summary of types of attacks on TRNGs.

2.4.2 Side channel threat analysis on TRNGs

Before citing and describing example of existing attacks on TRNGs, let us first give an overview of physical threats towards any crypto-system in general. In cryptography, an attack is by definition the recovery of a secret data, which is very often the key or parts of the key used for message encryption, by means of physical or mathematical tools or the combination of both. Recent physical attacks have been introduced, namely the Kocher's cryptanalysis known as

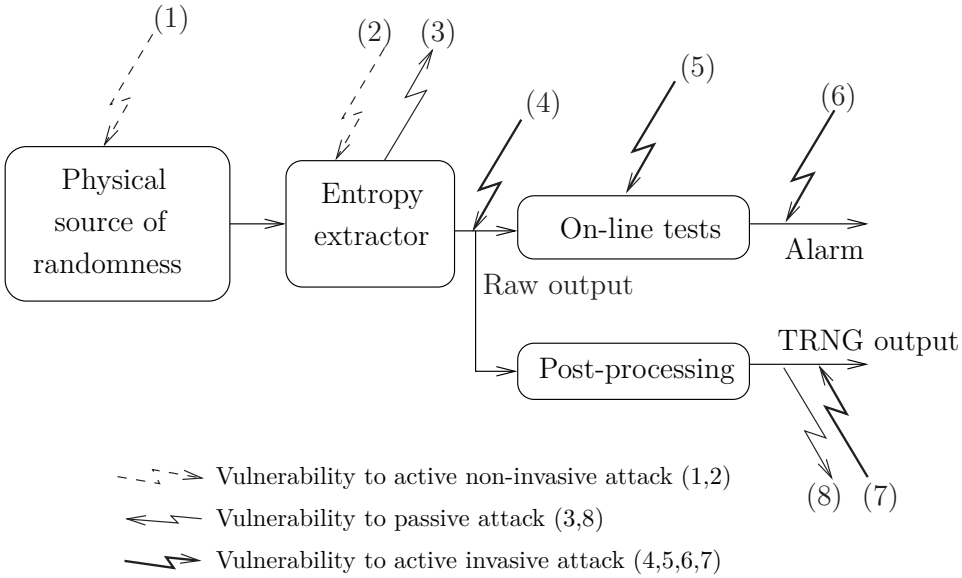


Figure 2.4: TRNGs vulnerabilities at different TRNG blocks

Simple Power Analysis (SPA) [KJJ99]. SPA exploits the power consumption of the system to reveal secret information as the key encryption or the time of execution. Other well known measurement techniques and statistical methods exists such as Differential Power Analysis (DPA), Electromagnetic emanation attack (EMA) [GMO01], Timing attacks [KJJ96] and Differential Frequency Analysis (DFA) [BBAF13]. The book [MOP07] gives a detailed overview of all the existing methods that allow to reveal sensitive data from a smart cards. These techniques allow to retrieve the secret key during its storage operations or encryption and decryption. All these attacks are known as Side Channel Attacks (SCA). By definition, a crypto-system is secure against SCA if all its sensitive data is not correlated to its time of execution or to its power consumption or its electromagnetic emanation. So, to design a secure TRNG, we should pay attention to its power consumption. The less the TRNG consumes, the less it leaks information, the more it is likely to be robust against SPA.

A different vulnerability may exist in case of secured implementation. A crypto-system may embed a counter-measure that uses a random number to mask the secret key [SPQ05] or introduce random delays to randomize the position of sensitive operation against SCA [CK10]. SCA protection by masking aims to decorrelate the power consumption from the secret information. In this kind of counter-measure, the random number is called the mask. In this case, the system security is threatened if the mask is biased [OMHT06, CGPR08].

Up till now and given our current knowledge, the only SCA attack experienced on TRNGs is the DFA presented in [BBAF13]. The DFA attack aims to extract the operating frequencies of the ring oscillators. Besides the electromagnetic (EM) analysis can identify the ring oscillators location on the FPGA. To extract these side channel information, authors perform what we call an EM cartography of the FPGA chip. To do so, the EM emanations of the device under operation is acquired point by point. This means that the EM emanations is acquired for different X, Y and Z positions of the magnetic probe. Hence, authors place the FPGA board which embed the RO-TRNG on a XYZ table with a shift precision of 1 µm. Figure 2.5 illustrates the setup environment used in order to perform the DFA attack on an FPGA RO-based TRNG.

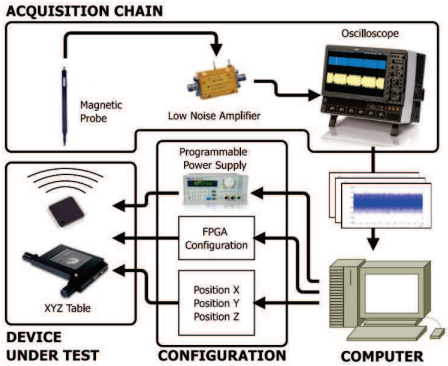


Figure 2.5: Attack setup of differential frequency analysis on multiple ring oscillator based TRNG.

There have been more publications on TRNG active attacks. Some studies have been done concerning active attacks on TRNG which consist in deliberate perturbations to bias the TRNG output. The most known are the Fault Injection Attack (FIA) [BBKN12] and Electromagnetic Injection Attack (EMIA) [PTL$^+$11]. As opposed to the SCA attacks, these attacks aim to inhibit the TRNG source of entropy. This kind of attack is more threatening than an SCA since it can disable the random number generation process. In the following, we present in Sub-section 2.4.3.1 and Sub-section 2.4.3.2 respectively the harmonic injection attack and the laser injection attack on TRNGs.

2.4.3 Active attacks on RO-based TRNGs

2.4.3.1 Harmonic injection attack on RO-based TRNGs

The so-called frequency injection attack was publised in [MM09]. The principle of this attack is to cause the mutual coupling of the ring oscillators aiming at reducing the ring oscillators jitter. To understand the principle of this attack, let us first present the theory behind this attack then a simple circuit to test the effectiveness of the attack [MM09]. Figure 2.6 illustrates the principle of harmonic injection attack on a simple two-ring-oscillator circuit.

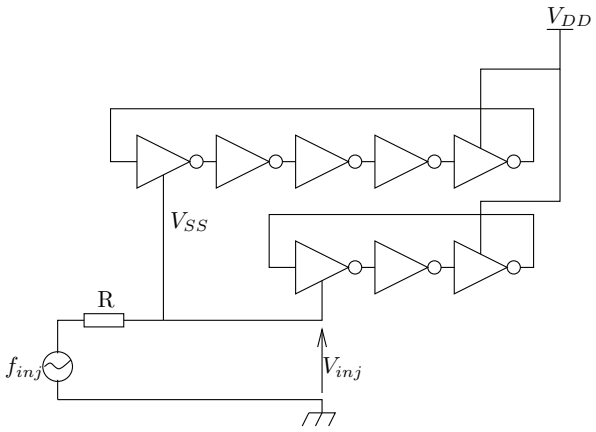


Figure 2.6: Harmonic injection attack on a simple two-ring-oscillator circuit.

This attack was performed on an EMV (European Mastercard Visa) embedding a Ro-based TRNG with relatively prime RO lengths. The frequency injection attack on multiple RO-based TRNG was performed in two steps [MM09]: First, the attacker need to deduce the random generator operating frequency. Authors activate the TRNG by sending the ISO 7816 GET CHALLENGE command to the smart card. Then they apply analyze the spectrum of the EM emanation of the smart card under operation first when the TRNG is off then when the TRNG is running by means of spectrum analyser such as illustrated in Figure 2.7. This frequency analysis is described in [BBAF13].

They identify a frequency corresponding to the TRNG operation mode. After that, they inject a sine signal, whose frequency is the one previously identified, on the power supply pad feeding the ring oscillators. Figure 2.8 presents a description of the setup used to perform the frequency injection attack.

NIST statistical tests were applied on sequences acquired from the EMV card under attack. This attack caused the failure of the statistical tests. According to [MM09], Table 2.8 gives the NIST statistical tests results of

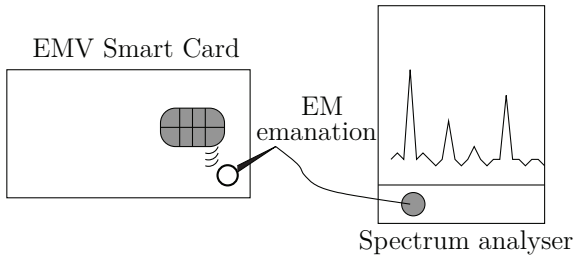


Figure 2.7: Electric field characterisation of the EMV smart card embedding a multiple ring oscillator based TRNG.

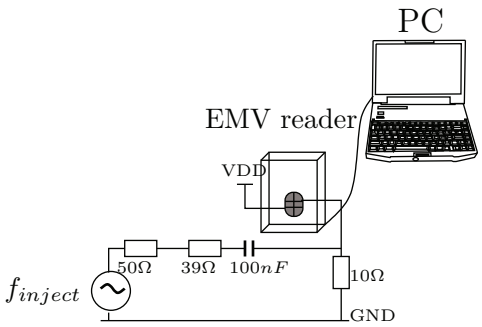


Figure 2.8: Attack setup of multiple ring oscillator based TRNG by means of frequency injection.

a sequence generated under attack versus a sequence generated before the frequency injection attack.

NIST	% of success
Without frequency injection	99%
With frequency injection	15%

Table 2.8: NIST statistical tests result of a multiple ring oscillator based TRNG under attack

To counter-act the frequency injection attack, authors of [MM09] propose some solutions. They suggest for example filtering power pad injected frequencies or smoothing the power supply to prevent from signal injection. However, the same attack principle can be performed by contactless EM injection called EMIA attack. It has been introduced in [BBA$^+$12]. Similarly to [MM09], Bayon et al.[BBA$^+$12] succeeded in controlling the bias of the RO-based TRNG implemented in FPGA by electromagnetic injection. The

latter cited attack can thwart any chip filtering on the power pad since it is contactless. This type of attack can even bypass the TRNG online tests when they are performed periodically. We classify the above cited attacks as attack label (2) of Figure 2.4. EMIA attacks may also be classified as attack labels (4),(5),(6) and (7) depending on the strength of the EM field injected. This type of attacks reduced the scope of use of RO-based TRNGs as they represent a threat on the security of smart cards embedding such TRNGs [MM09]. Since then, new TRNG architectures namely using metastable DFFs and latches [DGH07, MKD11, HI12] have risen a great interest thanks to their resiliency against frequency injection attack.

2.4.3.2 Laser injection attacks on TRNGs

The laser FIA attack has been experienced in [SCDEV13]. It is harmful to the TRNG raw, the post-processed output and the alarm signal. It characterizes respectively the threats (4), (6) and (7) such as illustrated in Figure 2.4. By means of a laser spot injection, an attacker may set the online tests output alarm to a logic 1. This would be very harmful to the TRNG since it would not be aware anymore in case of source of entropy failure. To our best knowledge, no such attacks have been experienced on TRNGs. However in [SCDEV13], Soucarros et al. experienced a laser injection on a VIA processor embedding a TRNG. Since the vendor does not provide any information about the TRNG design nor the TRNG position inside the CPU chip, Soucarros et al. propose to perform a cartography of the chip by applying different laser shots all over the chip using a XY table. Figure 2.9 illustrates the attack setup of a fault injection perturbation of a TRNG using a laser source.

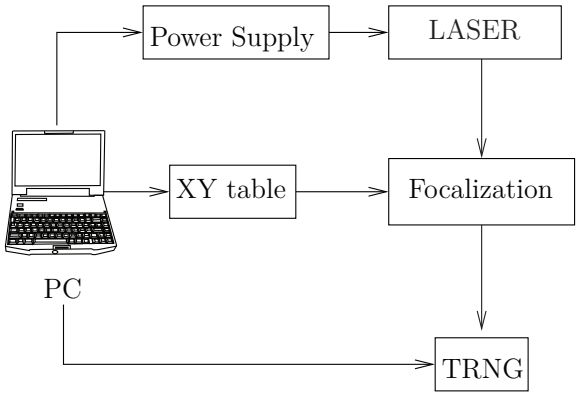


Figure 2.9: Attack setup of the TRNG's entropy perturbation by means of laser injection.

A computer drives the power supply voltage of the laser diode to perform several laser shots. It also controls an XY table to move the laser beam

over the component step by step. This allows to impact the chip to different positions and generate a cartography from the obtained results. Finally, the PC communicates with the component to start the generation of random numbers, acquire the TRNG sequences and reset the component before each new laser shot. Fortunately, detection of such invasive attacks is possible by embedding on-chip sensors [SBGD11, All08] to detect any malevolent perturbation by triggering an alarm. Another protection would be shielding the TRNG circuit to make it immune against EMA. The shield is a top metal wired pattern that is drawn on top of the circuit layout to make complex invasive attacks on the circuit front size. It is a protection against probing that attempts to modify the value of a signal. In case of external probing, the shield alarm is triggered.

2.5 Comparison between Existing TRNGs

To synthesize and conclude the state-of-the-art in terms of design conducted in Chapter 1 and the state-of-the-art in terms of TRNG evaluation of this chapter, we propose a summary in Table 2.9. It illustrates the classification of the existing full digital TRNGs in terms of throughput, design complexity and resiliency against physical attacks.

TRNG name[References]	Noise source	Technology	Complexity in LUT	Vulnerabilities to EMIA	Throughput (Mbps)	Stochastic model
BRAM-TRNG [GCS09]	V	Virtex 4	1294 LUT 16 BRAM	-	25	-
PLL-TRNG [FDcC04]	H	Stratix Cyclone 3	-	x	2	x -
Two-RO-TRNG [KG04]	H		x	x	0.5	-
FIGARO-TRNG [DG07]	H	Spartan 3	-	x	12.5	-
Multiple-RO-TRNG [SMS07]	H	Virtex 2	1000 LUT	**x**	80	x
Open-Loop-TRNG [DGH07]	H-V	Cyclone 2	-	-	100	-
Multiple-RO-TRNG [WT08]	H		83 LUT	x	100	x
TERO-TRNG [VD10]	H-V	Spartan 3 Actel Fusion	- 8 LUT	- x	0.25	x
Meta-TRNG [MKD11, HI12]	V	Virtex 5	256 LUT	-	12.5	-
Meta-RO [VHK12]	H-V	Virtex 2	x	x	140	-
STR-TRNG [CFAF13]	H	Virtex 5	-	x	16	x

Table 2.9: Comparison between existing TRNGs implemented in FPGAs

In Table 2.9, in the vulnerabilities column, a - means that the TRNG is inherently robust against the EMIA attack. The bold **x** means that an EMIA attack has been experienced and published. However the x means that this TRNG is potentially vulnerable to the EMIA attack since it does exploit more than two oscillating signals. In the stochastic model column, a - means that the TRNG is not provided with a stochastic model. However the x means that authors provide a stochastic model and validated it through experiments. In this work, we target the design of a TRNG that exploits both jitter and voltage noises and would be:

- resilient against harmonic injection and EMIA attacks,
- testable at design time through a stochastic model,
- robust against environmental perturbations,
- feasible in both FPGA and ASIC technologies,
- not very demanding in terms of resources and
- features throughputs in the order of the state-of-the-art.

2.6 Conclusion

This chapter deals with three important steps towards the certification of a TRNG: the statistical evaluation, the stochastic modeling and finally the physical vulnerabilities issue. We present the state-of-the-art of TRNGs evaluation methods and propose a thorough study of the AIS-31 standard. An overview of the existing stochastic models of TRNGs is also given. Then, we studied the potential vulnerabilities and discussed the state-of-the-art of physical attacks on TRNGs. It helped us to identify the evaluator guidelines towards the test and validation of our TRNG. The conducted study in this chapter allows us to set the TRNG design specifications. Therefore, in Chapter 3, we propose a novel digital TRNG design and its stochastic model. In fact, our goal is to design a TRNG which is feasible in FPGA and ASIC, testable at design time through a stochastic model and robust against environmental perturbations and harmonic injection attacks.

Chapter 3

Principle and Model of the Delay Chains based True Random Number Generator

3.1 Introduction

As the quality of randomness is a crucial matter for cryptographic applications, the source of entropy used to generate random bits should be studied. In fact, to make the TRNG testable at design time the designer should identify the design parameters that impacts the quality of randomness and quantify the entropy at the TRNG output. This chapter is undertaken with regards to providing a stochastic model of the delay chains based TRNG. We describe how to take advantage of the combination of phase noise and voltage noise in CMOS technology to extract randomness. The primary objectives of this chapter is first to discuss the source of noise used and how to sample this noise to generate a random bit. The second objective is to model this behavior and establish an estimation of the probability that the TRNG output is equal to one depending on the TRNG design parameters and the noise source. This chapter is divided into three sections. Section 3.2 presents the principle of entropy extraction in the delay chains based TRNG. Then, in Section 3.3 we present the generic architecture of the delay chains based TRNG. Finally, Section 3.4 addresses the stochastic modeling of a delay chains based TRNG.

3.2 Principle and Source of Randomness of the Delay Chains based TRNG

3.2.1 Source of randomness

The principle behind the delay chains based TRNG is to take advantage of the uncertainty when sampling a signal around a metastable region. This delay uncertainty is basically caused by the omnipresent noise in a CMOS integrated circuit. The basic idea of the existing TRNGs is to exploit the superimposition of different types of noise intrinsic to the silicon to generate random output. We propose to force D-latches to operate in the metastable

region to take advantages of two exploitable phenomena in the delay chains based TRNG. Assume that the data switches in the metastable forbidden window. This means that the bi-stable is forced to operate in the metastable mode. In this metastable region, the bi-stable cell's outcome is unpredictable. This may be provoked either by:

- The delay uncertainty around the sampling time of data in bi–stables cells. In fact, if the data signal, D, switches before the t_{setup} limit, the bi–stable output, Q, would resolve to V_{DD} else to 0V. This uncertain moment of data switching state is caused by the clock jitter.
- Or, it can be caused by the voltage fluctuations around $\frac{V_{DD}}{2}$. In fact, the resolving state of the bi-stable depends on the offset from $\frac{V_{DD}}{2}$. If this offset is positive Q output would resolve to V_{DD}. If it is negative Q output would resolve to 0V.

We denote the timing uncertainty characterized by the jitter as δt and the voltage offset uncertainty by δv such as represented in Figure 3.1.

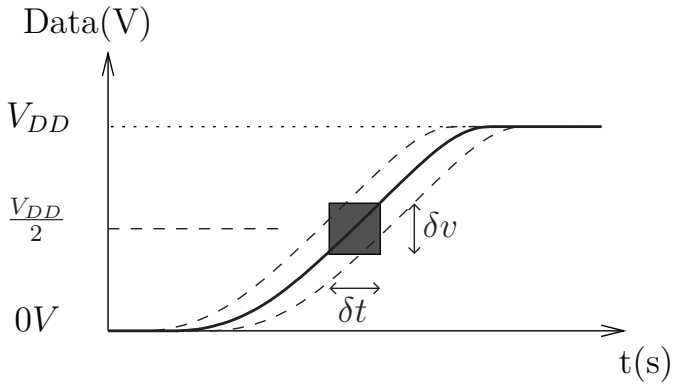


Figure 3.1: Sampling uncertainty around $\frac{V_{DD}}{2}$

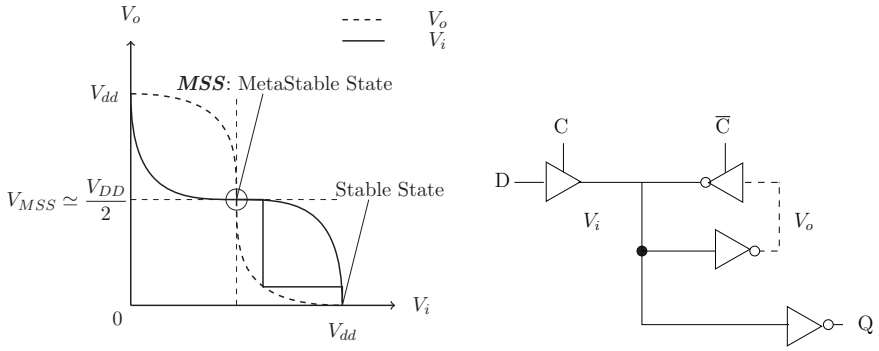
3.2.2 Metastability principle and characterization

In nowadays sub-micron silicon technologies, IC designers work on how to avoid the hazardous phenomenon of metastability by improving synchronizers and predicting the failure of bi–stable cells [Gin11, Fol96, CSC+10]. In the case of our TRNG design, we want to exploit this phenomenon to generate true random bits. More precisely, we study the behavior of latches in case of timing violation and exploit the ambient noise to generate unpredictable bits. The output of such bi–stables produces an unpredictable state if the data is forced to change near the metastability window. The very first implementation idea is to sample the clock by the clock itself, this way the

data switches in the forbidden interval $[t_{setup}, t_{hold}]$where t_{setup} and t_{hold} are defined by the following conditions:

- D must be stable for a duration of at least t_{setup} before the active edge of clock.
- D must remain stable for at least t_{hold} after the clock edge.

Failure to meet these timing conditions may cause an awkward operation mode of the latch and make the latch fall in a metastable state. The metastable state, *MSS*, is a state where the output voltage is neither a valid low nor a high logic state such as depicted in the transfer characteristic of Figure 3.2a. In this state, the voltage values of both the input and the output of the static storage element, have the same value $V_{MSS} \simeq \frac{V_{DD}}{2}$.



(a) Transfer Characteristic (b) CMOS D-Latch cell

Figure 3.2: D-Latch characterization around metastability.

Depending on data and clock signal arrival times, three situations are possible:

(i) The delay between the clock C and the data D, δt_{DC}, is greater than t_{setup}. This implies the Q output goes rapidly to V_{DD}.

(ii) $\delta t_{DC} \simeq t_{setup}$. This means that there is a t_{setup} violation and Q may remain stuck around an intermediate voltage level V_{MSS} which is neither a $0V$ nor V_{DD}.

(iii) δt_{DC}, is lower than t_{setup} the output Q never leaves $0V$.

Figure 3.3 illustrates these 3 possible scenarios.

As we want to force the data signal to switch in the metastable window, we need to use a very precise limit value of δt_{DC} for which the Q output needs a high propagation time to resolve to a stable state. Later we refer to this

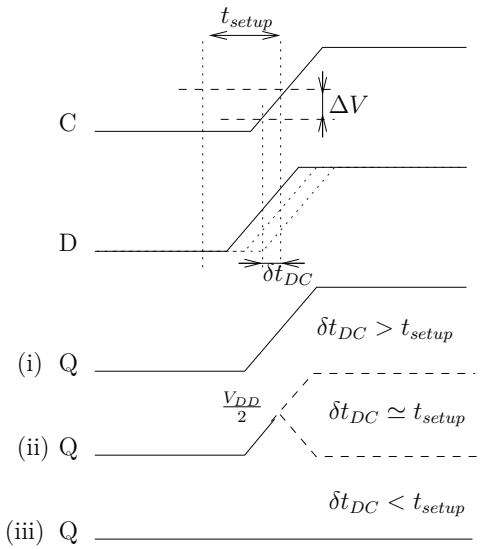


Figure 3.3: Q output value depending on data arrival time uncertainty

asymptotic value as T_{setup0}. To sort out T_{setup0}, we perform several simulations on bi–stable cells with a tunable delay to set different δt_{DC} then observe the behavior of the V_i node of Figure 3.2b. Transient simulations of bi–stable standard cells are done to characterize T_{setup0}, t_{setup} limit of metastability appearance. The results presented here are obtained with the LDLQ latch standard cell of STM 65nm CMOS technology. Simulation conditions are a temperature of 25° C, typical fabrication process of the transistors and a supply voltage VDD equal to 1.2 V. In this simulation, we perform a simple dichotomy on the δt_{DC} to extract a very precise T_{setup0} in the order of 1 fs. δt_{DC} is a tunable parameter in the Spectre netlist of the circuit under test of Figure 3.4. The circuit used to measure the T_{setup0} and the time needed for the D-Latch to resolve,T_{CQ}, from MSS is presented in the Figure 3.4. From this measurements, we visualize the internal node, V_i, of the back to back inverters of the D-Latch as illustrated in Figure 3.2b. In fact, we cannot visualize metastable state on the Q output since it is amplified.

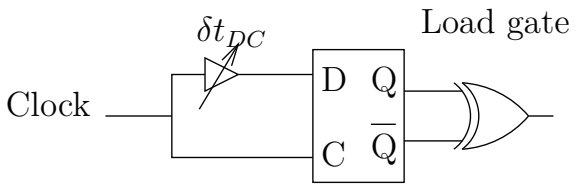


Figure 3.4: Measurement circuit.

This simulation is performed using the Cadence simulation tool Spectre MDL (Measurement Description Language). The principle is to tune the data-to-clock delay, δt_{DC}, with a very high precision to capture the exact moment of transition of the Q output and thus report what we call T_{setup0}. When the value of V_Q is higher than $0.9V_{DD}$, the δt_{DC} is too short to cause the transition. When the value of V_Q is equal to or less than $0.9V_{DD}$, δt_{DC} time is long enough to cause Q transition. The δt_{DC} searching process continues running until the condition $\frac{V_{DD}-V_Q}{V_{DD}} < 0.1$ is verified. The source code of Algorithm 2 represents the methodology of T_{setup0} characterization.

Algorithm 2 Algorithmic implementation of the T_{setup0} measurement process.

alias measurement *trans1*
{
  export real t_c, t_d
  run tran(stop=4ns, errpreset='conservative)
  t_c = **cross**(sig=C, thresh=0.95*VDD, dir='fall, n=1)
  t_d = **cross**(sig=V_i, thresh=0.95*VDD, dir='rise, n=-1)
  export real $T_{CQ} = t_d - t_c$
  export real $V_Q = V(Q)@4n$
}
search δt_{DC} from **swp**(start=-100ps, stop=0ps, tol=0.001ps)
{
  run *trans1*
}
until ($\frac{V_{DD}-V_Q)}{V_{DD}} < .01$)

Figure 3.5 shows the clock-to-output propagation delay, T_{CQ}, versus the data-to-clock delay δt_{DC}.

We notice from Figure 3.5 that, the more we get close to the metastable state, longer time is needed to go to a stable state. When the timing requirements are respected ($\delta t_{DC} < t_{setup}$), the propagation delay converges to a constant value which corresponds to the propagation delay of a transparent D-Latch $T_{CQ_{max}}$ given by the manufacturer. When δt_{DC} decreases, the propagation time T_{CQ} increases with a logarithmic shape. This increase is due to the recovery time from metastability. The asymptotic limit defines a minimum setup time for which the propagation delay T_{CQ} becomes infinite. In the following, we refer to this asymptotic value as T_{setup0}.

When the delay between clock and data inputs is around, T_{setup0}, the final state is sensitive to any noise in the circuit or external perturbation on the supply voltage [KD90]. To exhibit this behavior, we perform several Monte Carlo simulations using an additional transient noise. The simulator adds realistic estimation of the internal noise of the circuit based on prior

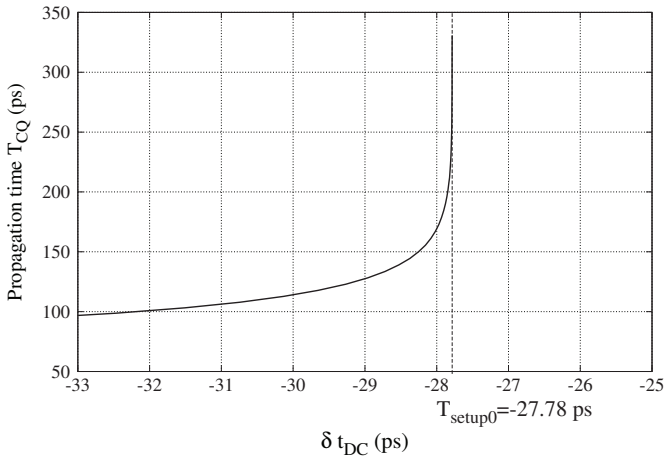


Figure 3.5: Propagation delay time T_{CQ} vs. δt_{DC} of a CMOS D-Latch.

library characterisation. The final state is not deterministic and depends on the noise value. Figure 3.6 represents different runs of this simulation showing the behaviour of the internal net, V_o, of LDLQ latch. The transient noise simulation is performed with a data-to-clock delay equal to T_{setup0}.

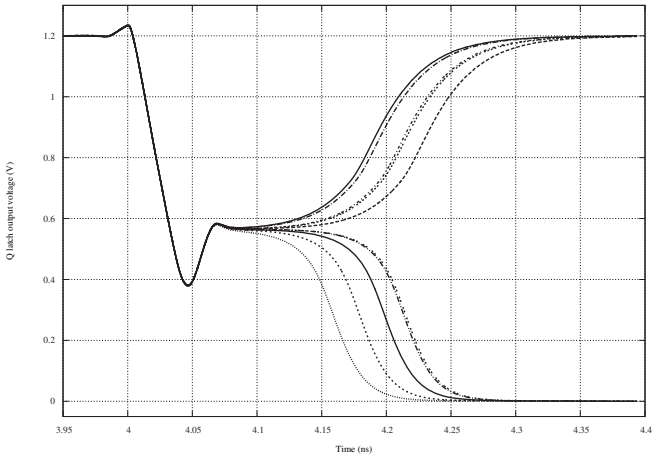


Figure 3.6: Transient noise simulation: The internal net behaviour for 10 noise iterations for LDLQ latch where $\delta t_{DC} = T_{setup0}$.

3.3 Proposed Delay Chains based TRNG

3.3.1 Delay chains structure specifications

As previously discussed, we propose to exploit the metastable state in Latches. This assure that the probability for Q to settle down to 1 is equal to 50% such as represented in Figure 3.7. If we sample the data when it switches we cannot predict its output.

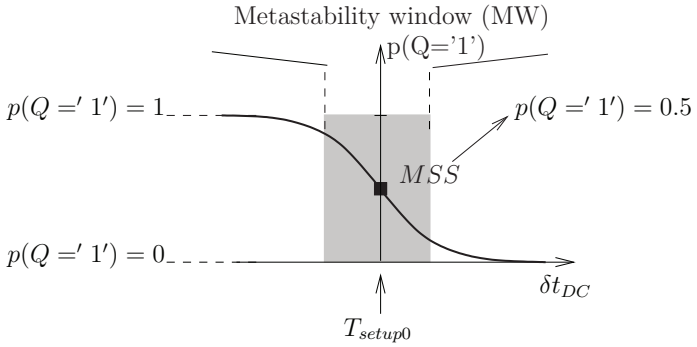


Figure 3.7: p(Q='1') vs. clock-to-data delay around the metastable state

The challenge is to manage to set the data-to-clock delay on purpose in the metastability window. A naive approach would be to delay the data of T_{setup0} (as defined in subsection 3.2.2) from the clock edge with a controllable delay element. Figure 3.8 illustrates this approach. This kind of control and precision is impossible in FPGAs nor in ASIC standard cells. So, we propose to use multiple delay elements: A delay chain on the data path and another delay chain on the clock path with a slight delay difference and consider the differential delay.

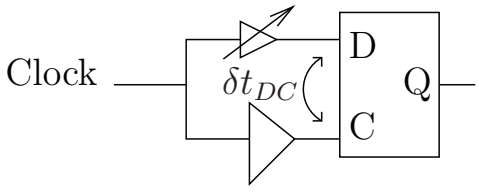


Figure 3.8: Principle of the delay chains based TRNG.

This allows to reach better precision. In fact, the higher is the precision, the greater are the chances to obtain metastability. In the next sub-section we introduce the delay chains based structure.

3.3.2 Delay chains based TRNG generic structure

So we propose the generic structure of Figure 3.10 as the delay chains based TRNG. Two fine chains feed a latch chain with slightly different incremental delays at the data and clock inputs. This allows us to control δt_{DC} with a fine delay control precision so that the data edge catches up the clock edge such as represented in Figure 3.9.

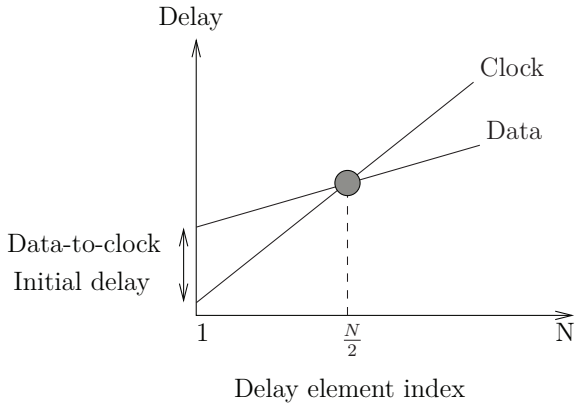


Figure 3.9: Data and clock race through the latches to catch metastability.

As represented in the Figure 3.10, we also use 2 blocks of coarse delay chains which help setting the initial delay between clock and data signals to compensate eventual change in the working conditions.

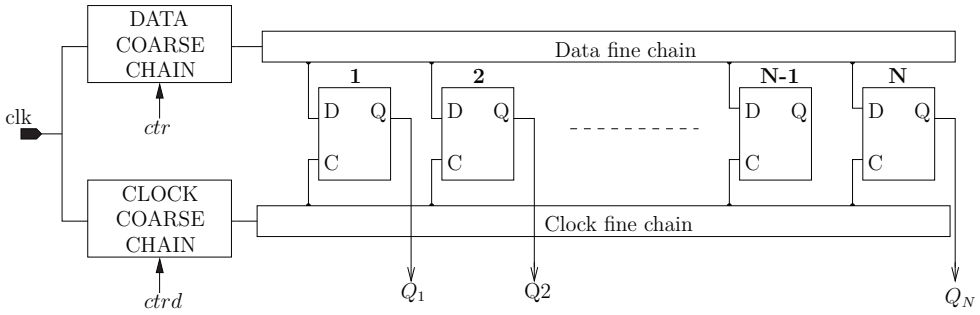


Figure 3.10: Generic structure of the delay chains based TRNG

To shrewdly control delays to make the D-latch operate in the metastable region, two adjustable delay blocks are used on both data path and clock path. The first chain should introduce a thin differential delay between data and clock. We name it a fine delay chain. The second chain set the initial data-to-clock delay which is higher in resolution than the first chain. We name it a coarse delay chain.

Fine delay chain The fine delay chain is used to tune the delay between the clock and data inputs of the latch with high precision. We propose to use two fine delay chains, one on the data path and the second one on the clock path and consider the differential delay. The fine delay chain length is equal to the number of latch used, N.

Coarse delay chain Similarly, we use two coarse delay chains, one on the data path and the second on the clock path. The delay in the coarse chain is adjustable by controlling M chained multiplexers. The number of MUXes, M, as a function of the number of latches is given in Equation (3.1). Equation (3.1) is set by the assumption that the data and clock signals meet somewhere at the input of one of the middle latches in the chain.

$$\frac{M}{2}.dT_{mux} + \frac{N}{2}.\delta t_{DC} = T_{setup0} \tag{3.1}$$

The coarse delay chain is used to adjust delays that may change with the operating conditions variations. The implementation of this kind of structure is hard to carry out due to (physical routing imbalance, asymmetries between cells, etc), delays between the data and the clock inevitably exist. In fact, process variations between chips, and even between LUTs and switch matrices on the same FPGA chip, can lead to a difference in delays depending on the FPGA region it is implemented in.

The raw output of the TRNG, *trngo*, is obtained by XORing all the latch outputs as given in 3.2.

$$trngo = \bigoplus_{i=1}^{N} Q_i \tag{3.2}$$

In the next section, we discuss the design parameters that impact the TRNG output and establish the mathematical formulation of the probability at *trngo* raw random bit.

3.4 Stochastic Model of Delay Chains based TRNG

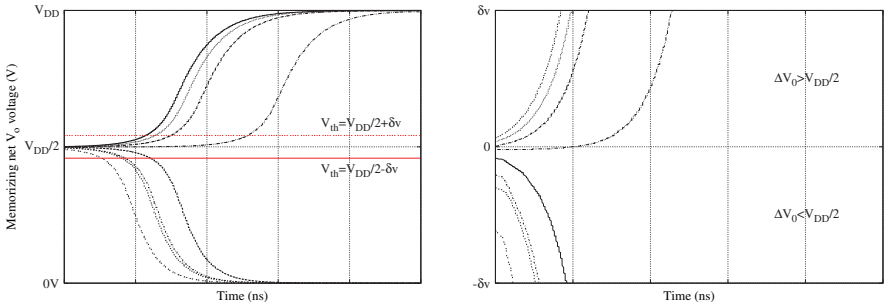
TRNG designers should provide a stochastic model, first to estimate the entropy of the raw and post-processing output, then to identify the parameters that may affect the entropy and quantify their impact on it. The TRNG raw output is represented by one bit which can take the logic value '1' or '0'. The TRNG entropy of the raw output, *trngo*, is expressed in Equation (3.3) where p_1 and p_0 represents respectively the probability to get '1' at *trngo* and the probability to get '0'.

$$H_{trngo} = -p_0.log_2(p_0) - p_1.log_2(p_1) \tag{3.3}$$
$$\tag{3.4}$$

where p_1 represents the TRNG output to be equal to '1'. In the following, we try to establish a mathematical expression of the probability $p(trngo =' 1')$ to identify the TRNG parameters that affect it. Later we try to extract optimal parameters that generate the maximum possible entropy of the TRNG architecture presented above. For that matter, we need to identify the technology and design parameters that are involved in the TRNG entropy.

3.4.1 Stochastic model parameters identification

To study the probability for Q to settle down to either a logic '1' or a '0', we need to go deeper in the analysis of the metastable behavior. So, let us zoom around the *MSS* state of Figure 3.6. The metastable state, *MSS*, is a state where the output voltage is neither a valid low nor a high logic state such as depicted in Figure 3.2a. In this state the voltage values of both the input and the output of the static storage element, have the same value $V_{MSS} \simeq \frac{V_{DD}}{2}$.



(a) Exit from metastability (b) Voltage offset around MSS

Figure 3.11: D-Latch characterization around metastability.

Around this point, the inverters of the static storage element can be modeled as two amplifiers with a negative gain $(-A)$ where $A \gg 1$ [CSC+10, Gin11]. We note that we consider equal gains for both inverters for the simplicity of expression. Each inverter drives a resistance R and a capacitive load C which models the gates and connections at each output. To simplify computations the capacitance and the resistance of both inverter are considered to be equal to each other. In the absence of noise, the voltage of the internal node V_o remains stuck at this intermediate voltage, around $\frac{V_{DD}}{2}$. The probability to enter a *MSS* whose duration is longer than t_m is expressed as in Equation (3.5) [Vee80].

$$p(t > t_m) = e^{-\frac{(A-1)}{\tau} \cdot t_m} \tag{3.5}$$

Practically, when the clock C switches to '1' the node voltage is never exactly $\frac{V_{DD}}{2}$, and even then, ambient noise can shift this position. This bias

will determine the final logical value and the time to reach it as shown in Figure 3.11a. In fact, ΔV_{DG_0} impacts the final state of the D-Latch. Figure 3.11b shows the behavior of the internal storage net at *MSS* state for different data-to-clock delays. The sub-figure (b) is a zoom of (a) around $\frac{V_{DD}}{2}$.

The expression of the voltage difference $V(t) = V_o(t) - V_i(t)$ around *MSS* is given in Equation (3.6) [Gin11]:

$$V(t) = \Delta V_{DC_0} \cdot e^{\frac{A-1}{\tau}t} \tag{3.6}$$

where $\Delta V_{DC_0} = (V_o - V_i)(T_{setup0})$ is the voltage difference at the moment where the D-Latch switches to memorizing mode. $\tau = R \cdot C$ is the time constant.

We introduce a threshold voltage ΔV_{th} around *MSS*. This threshold corresponds to the voltage over which the state goes from *MSS* to a valid logic value at the moment of sammpling.

$$T_r = \frac{\tau}{A-1} \ln(\frac{\Delta V_{th}}{\Delta V_{DC_0}}) \tag{3.7}$$

T_r represents the time needed to leave the metastable state or the increase in the propagation delay. We consider a linear relation between the voltage differences ΔV and the delay in arrival times of the D and C signals such as:

$$\Delta V_{DC_0} = \alpha\ A \cdot (\delta t_{DC} - T_{setup0}) \tag{3.8}$$

where α is the slope of the clock and data input and A the gain of inverter.

Thus, by replacing the expression of V_{th} in Equation (3.7), we can express the resolving time as a function of the time delays as in Equation (3.9).

$$T_r = \frac{\tau}{A-1} \ln(\frac{\Delta V_{th}}{\alpha\ A \cdot (\delta t_{DC} - T_{setup0})}) \tag{3.9}$$

Which can be rewritten as in Equation (3.10).

$$T_r = \gamma(\beta - \ln(\delta t_{DC} - T_{setup0})) \tag{3.10}$$

$$T_{CQ} = T_r + T_{CQ_{max}} \tag{3.11}$$

where $\gamma = \frac{\tau}{A-1}$ and $\beta = \ln \frac{\Delta V_{th}}{\alpha\ A}$.

Equations (3.10) and (3.11) show that the D-Latch propagation delay, T_{CQ}, increases as δt_{DC} decreases. And this is what we obtained at electrical simulation as shown previously in Figure 3.5. Hence, we HenceHence,clude that the analytical expression models well the simulation waveform of T_{CQ} *vs.* δt_{DC} in the absence of noise.

3.4.2 Proposed stochastic model

Now we need to establish the probability expression of the TRNG output. To do so we consider the generic structure of Figure 3.10. For the i^{th} D-Latch, δt_{DC_i} represents the delay between D and C signals (C being the clock input of the latch). This delay is incremented between two consecutive latches by a differential delay δt, as expressed in Equation (3.12). δt comes from the difference between the two fine delay chains D and CLK.

$$\delta t_{DC_{i+1}} = \delta t + \delta t_{DC_i} \tag{3.12}$$

Figure 3.12 shows the clock-to-data delay at consecutive latches, superposed

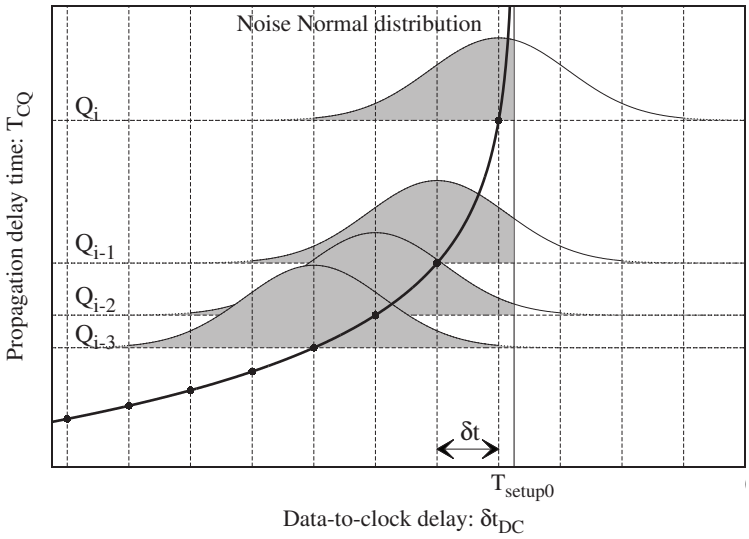


Figure 3.12: The probability to correctly sample the input for consecutive latches.

with the propagation characteristic of a D-Latch. This delay can be expressed, as in (3.13), as the sum of deterministic delays, which correspond to the signals race and a random delay, which models the noise impact.

$$\delta t_{DC_i} = \Delta D_0 - i \cdot \delta t + \mathcal{N}(\delta t) \tag{3.13}$$

where $\mathcal{N}(\sigma)$ is the Normal distribution and ΔD_0 is the initial delay between C and D introduced by the data and clock coarse chains. The delay incertitude is distributed approximately normally as it characterizes the influence of the superimposition of multiple noise sources according to the Central Limit Theorem [Tro59].

In the case of rising edge sensitive D-latch, the output Q resumes to a high logic value '1' if this delay is smaller than T_{setup0}. We denote $p_{Q_i} = p(Q_i = 1)$ this probability:

$$p_{Q_i} = p(\delta t_{DC_i} < T_{setup0}) \tag{3.14}$$

This corresponds to the gray colored area of the Normal bell in Figure 3.12. This probability can thus be analytically expressed as:

$$p_{Q_i} = \frac{1}{2}\left[1 - \mathrm{erf}(\frac{\delta t_{DC_i} - T_{setup0}}{\sigma\sqrt{2}})\right] \tag{3.15}$$

where σ is the noise standard deviation, T_{setup0} is the experimental asymptotic limit such as represented in Figure 3.12 and $\mathrm{erf}(x)$ is the error function.

In the following, we use the notation p_X, representing $p(X =' 1')$, where X is a Normal random variable. Since the TRNG output is the XOR of the N D-Latch outputs, as illustrated in Figure 3.10, the probability of the TRNG output to be equal to 1 is the probability parity of having an odd number of the N D-Latch outputs settling down to a logic '1'. Let $p_{trngo} = p(trngo =' 1')$ be the probability to have 1 on the TRNG output. Here we distinguish two cases:

(i) Influence of noise on each of the N D-Latches is independent.

(ii) The value of Q_i of the i_{th} D-Latch impacts the output value Q_{i+1} of the $(i+1)^{th}$ D-Latch.

Basically, computing the TRNG output probability, p_{trngo}, is equivalent to compute the probability of a N-input XOR to be equal to '1'. Let us consider the first two latches Q_1 and Q_2. Equation (3.16) represents the probability to obtain '1' at the output of the first stage 2-input XOR.

$$\begin{aligned} p_{Q_1 \oplus Q_2} &= p_{Q_1} \cdot p_{\overline{Q_2}} + p_{\overline{Q_1}} \cdot p_{Q_2} \\ &= p_{Q_1} \cdot (1 - p_{Q_2}) + (1 - p_{Q_1}) \cdot p_{Q_2} \\ p_{Q_1 \oplus Q_2} &= p_{Q_1} + p_{Q_2} - 2p_{Q_1}p_{Q_2} \end{aligned} \tag{3.16}$$

Equation (3.17) is the factorized expression of (3.16).

$$\begin{aligned} 1 - 2p_{Q_1 \oplus Q_2} &= 1 - 2p_{Q_1} - 2p_{Q_2} + 4p_{Q_1}p_{Q_2} \\ &= (1 - 2p_{Q_1}) \cdot (1 - 2p_{Q_2}) \end{aligned} \tag{3.17}$$

Then, by mathematical induction, we can generalize the expression for N-input XOR as shown in Equation (3.18).

$$1 - 2p(\bigoplus_{i=1}^{N} Q_i = 1) = \prod_{i=1}^{N}(1 - 2p_{Q_i}) \tag{3.18}$$

Thus, from Equation (3.18), the final expression of $p(trngo = 1)$ in terms of p_{Q_i} is given in Equation (3.19).

$$p_{trngo} = \frac{1}{2}[1 - \prod_{i=1}^{N}(1 - 2p_{Q_i})] \tag{3.19}$$

In the case (i) where the influence of noise on each D-Latch is independent, then, we end up with Equation (3.20) written as a function of design parameters, N and δt_{DC}, and the technology related parameters, σ and T_{setup0}.

$$p_{trngo} = \frac{1}{2}\left[1 - \prod_{i=1}^{N}\left(\text{erf}(\frac{\delta t_{DC_i} - T_{setup0}}{\sigma\sqrt{2}})\right)\right] \tag{3.20}$$

In the second case, (ii), where the value Q_i of the i^{th} D-Latch impacts the output value Q_{i+1} of the $(i+1)^{th}$ D-Latch, then Q_{i+1} cannot be equal to '0' if Q_i is equal to '1'. Thus, we eliminate some terms from the truth table of $\bigoplus_{i=1}^{N} Q_i$.

For example, for a 3-input XOR, only the following input triplets (1,0,0) and (1,1,1) are left. Hence, the probability of the output of XOR $p_{Q_0 \oplus Q_1 \oplus Q_2} = p(Q_0 \oplus Q_1 \oplus Q_2 = 1)$ would be expressed as follows:

$$p_{Q_1 \oplus Q_2 \oplus Q_3} = p_1 \cdot (1 - p_2) \cdot (1 - p_3) + p_1 \cdot p_2 \cdot p_3$$

For a 4-input XOR, the product of all p_i does not appear in the final probability, as the XOR of an even number of ones is 0.

$$p_{Q_1 \oplus Q_2 \oplus Q_3 \oplus Q_4} = p_1 \cdot (1 - p_2) \cdot (1 - p_3) \cdot (1 - p_4) + p_1 \cdot p_2 \cdot p_3 \cdot (1 - p_4)$$

By mathematical induction, we establish a general expression of $p_{trngo} = p(trngo = 1)$ for an N-input XOR (here N is even). Equation (3.21) represents thus the probability p_{trngo} versus the probability at the latches output for the case (ii).

$$p_{trngo} = \sum_{i=1}^{\frac{N}{2}} \prod_{j=1}^{2i-1} p_{Q_j} \cdot \prod_{j=2i}^{N} (1 - p_{Q_j}) \tag{3.21}$$

Finally, we get the probability expression of the TRNG as a function of design parameters: N and δt_{DC}, and the technology related parameters: the noise standard deviation, σ, and the T_{setup0} (Equation 3.22).

$$p_{trngo} = \frac{1}{2^N} \sum_{i=1}^{\frac{N}{2}} \prod_{j=1}^{2i-1} \left[1 - \text{erf}(\frac{\delta t_{DC_j} - T_{setup0}}{\sigma\sqrt{2}})\right] \cdot \prod_{j=2i}^{N} \left[1 + \text{erf}(\frac{\delta t_{DC_j} - T_{setup0}}{\sigma\sqrt{2}})\right] \tag{3.22}$$

Now it is an open question whether the ambient noise of the targeted technology present a correlated influence all along the chain of latches or

an independent impact on each latch. In the next chapter, we will present experimental results on the ASIC implementation of the TRNG and confront the analytical expressions of Equations (3.22) and (3.21) to the empirical frequencies of '1' at the TRNG output.

3.5 Conclusion

In this chapter, we have presented the principle of exploiting metastability and jitter to generate randomness. The principle is to place a D-Latch in a metastable state, then sample the stable state which is the consequence of the chip ambient noise impact. We discussed and presented the method that allows to compute the parameters of the TRNG output probability modeling equation through electrical simulation. The probability expression of the TRNG is computed in terms of the noise standard deviation, the characteristics of the D-Latch T_{setup0}, and the differential delay δt of the delay chains elements used in the TRNG core. The stochastic model parameters identified in this chapter allow us later to characterize the technology parameters that influences the TRNG quality of randomness and how to establish the TRNG design specifications.

After having introduced the principle behind the proposed TRNG and described the model parameters to design a true random bit generator, we present in next chapters how to design such an architecture in FPGA and ASIC technologies. The TRNG architecture introduced in this chapter is generic. In fact, it is based on an open loop structure composed only of latches and delay elements. This architecture assures full portability to both FPGA and ASIC technologies. In the next chapter, we will compare the modeling results versus the simulation results and measurements on the TRNG ASIC prototype.

Chapter 4

Study of the Delay Chains based True Random Number Generator for ASIC Technology

4.1 Introduction

This chapter is dedicated to the ASIC implementation of the TRNG previously presented in Chapter 3. It deals with the feasibility of the generic architecture with the CMOS 65nm standard cells library from STMicroelectronics. Section 4.2 is dedicated to the ASIC implementation TRNG design with multiplexors based coarse chain. We later refer to this ASIC implementation as ASIC_TRNG#1. Then in Section 4.3.1, we present NIST and AIS-31 standard statistical tests performed on the MUX based delay chains TRNG. Besides,we present measurement results of working conditions perturbations on the TRNG namely power supply voltage variations and temperature variations. Finally, in Section 4.4 an enhancement of the coarse and fine delay chains is proposed. We refer to the implementation of this ASIC prototype as ASIC_TRNG#2.

4.2 ASIC Design of the TRNG with MUX-based Delay Chains

4.2.1 Specifications

To implement the generic structure presented in Section 3.3 in ASIC technology, we need to set the specifications of the coarse and fine delay chains. The fine delay chain should allow very fine tuning of the clock-to-data delay, in the order of the *ps*. We propose to design different versions of implementation of the delay chains based TRNG in order to study:

- the impact of the fine delay chain precision on the entropy,
- two different bi–stable elements, the LDLQ standard cell latch and a custom latch,

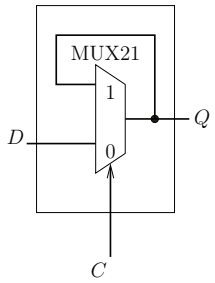


Figure 4.1: Custom latch structure.

- the impact of placement and routing of the delay chains on the data-to-clock delay.

Hence, we propose to implement 4 versions of the TRNG:

- TRNG0: Latches are LDLQ standard cells and the fine delay chains are designed with buffers where the data-to-clock differential delay $\delta t = 5ps$ (Figure 4.2),
- TRNG1: Custom latch with buffers for the fine delay chains where $\delta t = 5ps$,
- TRNG2: LDLQ with routing wires for the fine delay chains where $\delta t = 1ps$ and
- TRNG3: Custom latch with routing wires for the fine delay chains where $\delta t = 1ps$.

The custom latch is composed of a 2 to 1 multiplexor (MUX21) with a feedback connection such as illustrated in Figure 4.1.

The coarse chain delay is designed with M chained MUX21. The output of each MUX21 feeds the high active input. All the low active MUX21 inputs are fed with the same signal which is the clock input. This makes the global delay across the data coarse delay chain, ΔT_{muxD} varies as a function of the number of active bit on the *ctrd* input selection as in Equation (4.1). Similarly for the clock coarse delay chain. ΔT_{muxC} varies as a function of the number of active bit on the *ctr* input selection as in Equation (4.2).

$$\Delta T_{muxD} = ctrd.T_{muxD} \tag{4.1}$$
$$\Delta T_{muxC} = ctr.T_{muxC} \tag{4.2}$$

Figure 4.3a and Table 4.3b give respectively the architecture and table of possible delay at the output of the coarse chain as a function of the multiplexors control bits.

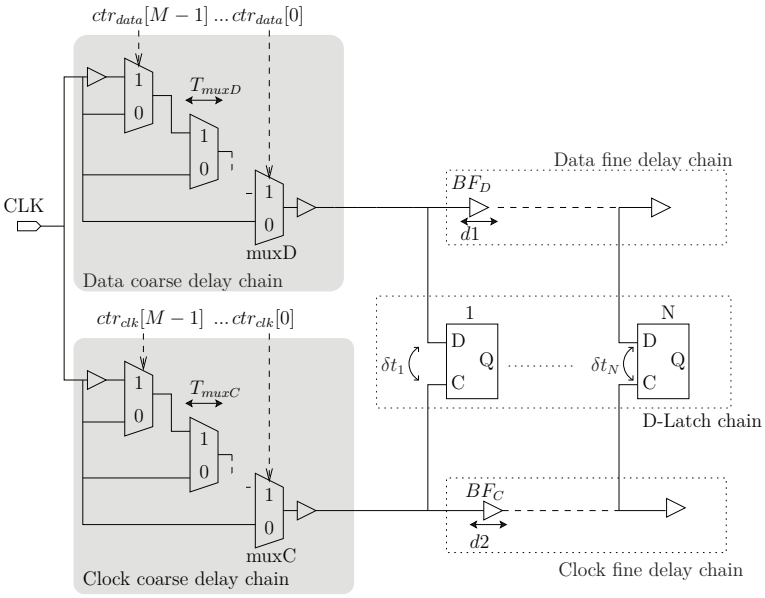


Figure 4.2: Detailed view of the TRNG0.

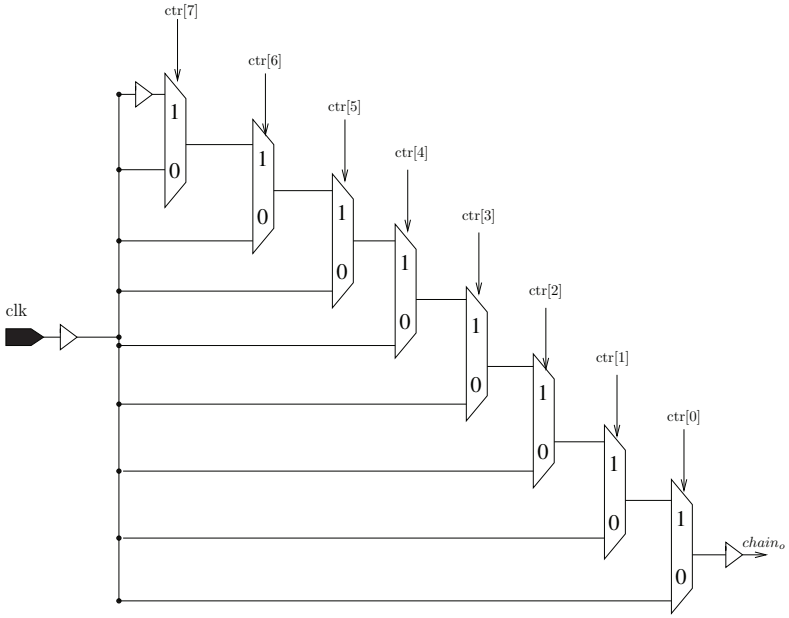
Table 4.1 gives all the notations. Hereinafter, the position of the metastable Latch will be referred to as i_m. To force the middle latch operate in the metastability region, Equation (4.3) has to be verified.

N	Number of latches
M	Number of multiplexors in each coarse delay chain bloc
i_m	Index of potentially metastable D-latch
$d1$	Propagation delay of the buffer BF_D on the data fine delay chain
$d2$	Propagation delay of the buffer BF_C on the clock fine delay chain
δt_i	Added delay on the i^{th} bi–stable element
T_{muxC}	Propagation delay in clock coarse delay chain
T_{muxD}	Propagation delay in data coarse delay chain
dT_{mux}	Differential delay between the two coarse delay chains
$ctr[M-1:0]$	Control bits of the M multiplexors of the clock coarse delay chain
$ctrd[M-1:0]$	Control bits of the M multiplexors of the data coarse delay chain

Table 4.1: Notations of the TRNG design parameters

$$\frac{N}{2} \cdot \delta t = \frac{M}{2} \cdot dT_{mux} \tag{4.3}$$

Equation (4.3) helps to choose the number and the capacitance of the MUX21 standard cells to use to design the coarse delay chains. The differential delay $\delta t = d1 - d2$ and dT_{mux} are obtained from the standard cells library data-sheet provided by the manufacturer STM. For N equal to 64, with δt



(a) Coarse chain architecture

ctr[7]	ctr[6]	ctr[5]	ctr[4]	ctr[3]	ctr[2]	ctr[1]	ctr[0]	**Global delay at coarse chaine output**
-	-	-	-	-	-	-	0	T_{muxC}
-	-	-	-	-	-	0	1	$2.T_{muxC}$
-	-	-	-	-	0	1	1	$3.T_{muxC}$
-	-	-	-	0	1	1	1	$4.T_{muxC}$
-	-	-	0	1	1	1	1	$5.T_{muxC}$
-	-	0	1	1	1	1	1	$6.T_{muxC}$
-	0	1	1	1	1	1	1	$7.T_{muxC}$
0	1	1	1	1	1	1	1	$8.T_{muxC}$

(b) Coarse chain possible delays

Figure 4.3: Mux-based coarse chain architecture and delay configurations

equals to 5 ps and dT_{mux} equals to 40 ps, we can compute the number M of MUXs needed as in Equation (4.4).

$$M = \frac{N \cdot (d1 - d2)}{dT_{mux}} = 8 \tag{4.4}$$

To make sure that the i^{th} Latch operates in the metastable region, i_m has to verify Equation (4.5). The general expression of the position i_m is given by Equation(4.5). If ctr and $ctrd$ are equal, Equation (4.6) is obtained. As N is equal to 64, the condition of appearance of metastability is given in (4.7).

Equation (4.7) is verified for best and worst cases of T_{muxD}, T_{muxC}, $d1$ and $d2$ from the CMOS 65nm data-sheet library.

$$(ctrd \cdot T_{muxD} + i_m \cdot d1) - (ctr \cdot T_{muxC} + i_m \cdot d2) = \delta T_{setup0} \tag{4.5}$$

$$i_m = \frac{ctr \cdot dT_{mux} + T_{setup0}}{\delta t} \tag{4.6}$$

$$1 \leq i_m \leq 64,\ 1 \leq ctr \leq 8,\ \forall \mathrm{T}^{\circ}C,\ \forall V_{dd} \tag{4.7}$$

In the case of wired fine chain TRNGs, the TRNG2 and TRNG3 versions, the fine delay δt is introduced by the routing differences on the data and clock paths. For the TRNG0 and TRNG1 versions, the fine delay δt is introduced by the differential propagation delay between the buffer on the data and the one on the clock.

4.2.2 Place and route process

Fine and coarse delay chains are very sensitive to the delays introduced by the wires. Coupling effects between routing wires may introduce additional delays at the clock to data differential delay. Added delays are proportional to the coupling capacitance between the routing metals along the fine and coarse chains and neighboring routing wires of the same metal level and upper metal levels. Hence, the place and route process (PAR) is a very important step and it has to be done manually and with special care to:

- balance the data and clock routing paths in both fine and coarse chains,
- balance routing paths at every stage of the XOR tree,
- minimize coupling effects between data and clock routing paths along the fine and coarse chains by outdistancing routing paths in the same metal and different levels of metals.

Figure 4.4 shows a zoom of the TRNG layout on the fine delay chains. The wire connecting the delay element BF_D to the input `D` and the wire connecting BF_C to the clock input `C` must be of equal length. The same thing has to be done for the coarse delay chains. The custom physical layout has been performed manually. We developed a placement and routing constraints script. Cadence Encounter parses the script and executes it to generate the layout. We can see a balanced routing of data and clock paths along the latches.

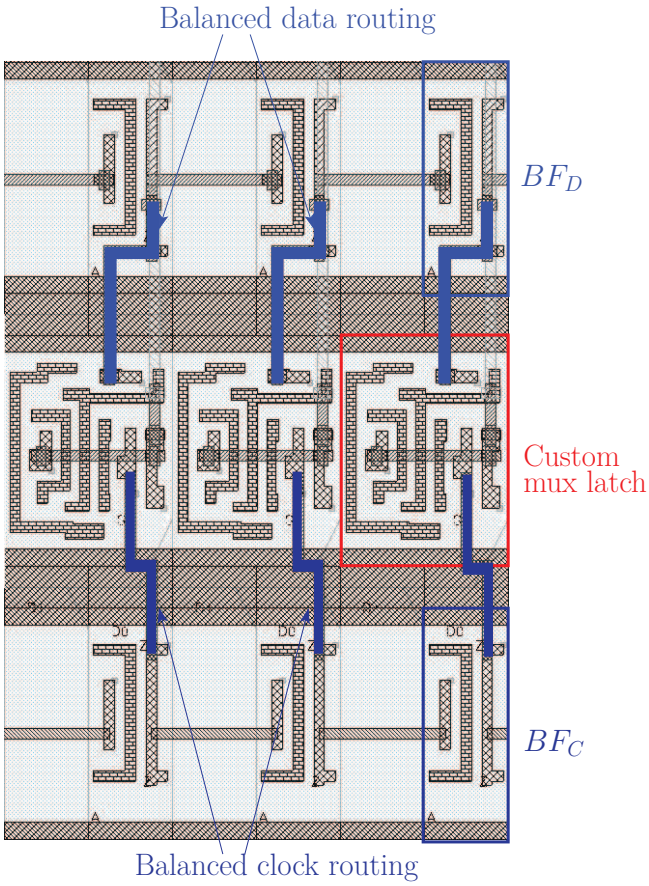


Figure 4.4: Zoom of the layout of the TRNG with custom latches: view of the balanced routing of the fine delay chains on data and clock paths.

4.2.3 Post place and route simulation results

Now that we have a balanced placed and routed design, we perform the resistances and capacitances coupling network extraction. This step is very important in the design of the delay based chains TRNG. In fact, coupling effects of routing lines inside the delay chains influence very much the data to clock differential delay. Hence, it is important to verify the data to clock delays all along the chain. After the parasitics extraction, post layout simulations are performed. Post layout verifications are done on 4 steps:

1. The first post layout simulation is intended to extract the data to clock delays after parasitics extraction and check whether at least one latch is metastable.

2. If it is not the case, the re-adjustment of the initial data-to-clock delay is needed. Then, we re-launch the previous step.

3. After that, we extract the delay configuration for which a metastable behavior is perceived.

4. Finally, we perform a transient noise simulation to verify the randomness behavior.

The transient noise simulation is an important step. It represents the essence of TRNG verifications before final integration of the TRNG hard macro and the GDSII delivery to foundry. In fact, when the data signal switches in the metastable region, the final state Q is sensitive to any noise in the circuit. This sensitivity has been described in [KD90]. To exhibit this behavior, we have performed several Monte Carlo simulations using an additional transient noise. The simulator adds realistic estimation of the internal noise of the circuit based on prior library characterisation. The final state is not deterministic and depends on the noise. This simulation allows us to check:

1. if the design generates random bits by running electrical simulations using transient noise. Figure 4.5 shows a random state in the middle of the latches chain depending on the noise simulation iteration.

2. if i_m is somewhere in the middle of the latches chain considering different working conditions (Typical, best and worst cases).

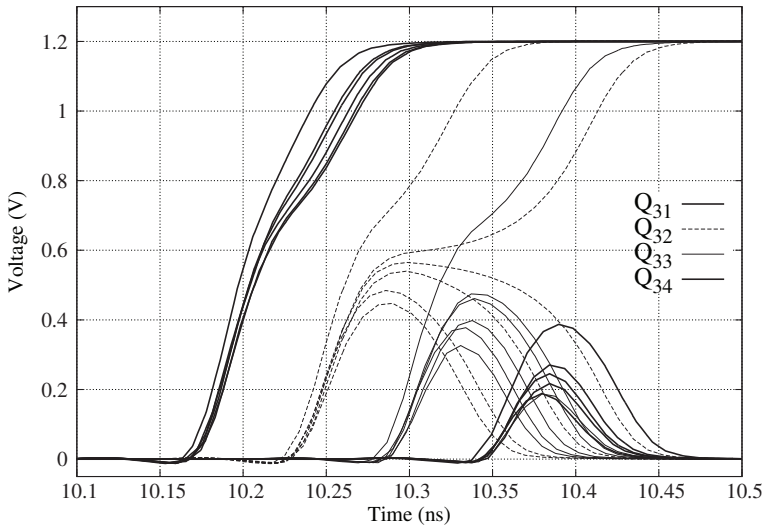


Figure 4.5: Post place and route simulation for 6 transient noise iterations of the TRNG.

Such a simulation was not possible to be performed for the TRNG2 and TRNG3 with wire-based fine delay chains because there is no noise model

for the passive element. In fact, the Monte Carlo transient noise simulation models exist only for the standard cells. Hence, for wired fine delay chains of the TRNG versions 2 and 3, we cannot perform such a simulation. Instead, we performed a simulation with a jittery data signal. We simulate the behavior of the TRNG1, with transient noise, for the 64 possible values of control inputs *ctr* and *ctrd* of the coarse delay chains. Table 4.2 shows the number of ones at the output of the TRNG2 for 100 iterations of the Monte Carlo simulation. The best results are those highlighted in gray in Table 4.2 .

ctr \ *ctrd*	0x00	0x01	0x03	0x07	0x0f	0x1f	0x3f	0x7f
0x00	73	90	82	91	68	78	100	100
0x01	98	98	90	95	86	67	79	97
0x03	59	56	84	97	77	91	67	74
0x07	64	64	84	94	95	57	89	74
0x0F	0	0	86	98	98	94	90	89
0x1f	0	0	0	98	98	93	59	91
0x3f	0	0	0	0	64	86	96	70
0x7f	0	0	0	0	0	88	77	97

Table 4.2: Post-layout simulation results considering transient noise: frequency of occurrences of bit '1' over 100 iterations.

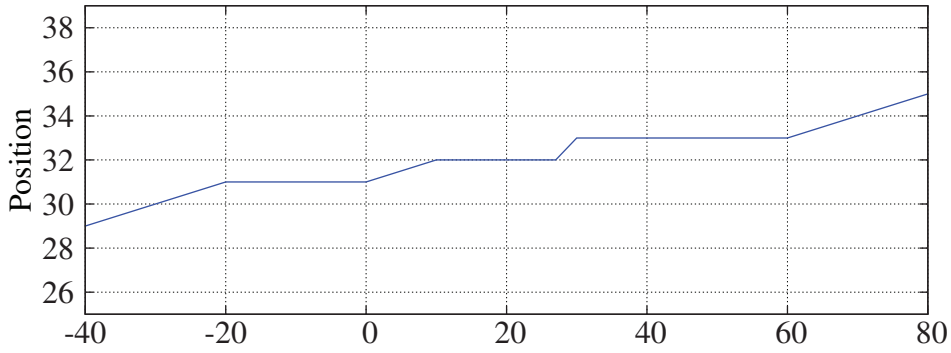
The index i, where the data and clock signal meet around the metastable region is likely to shift with temperature, voltage and process variations. The variation of i depending on the transistors corners for a chosen control value is reported in the Table 4.3. Results of Table 4.3 are presented for *ctr*=0x1f and *ctrd*=0x3f. In fact, for delay control value, (*ctr*=0x1f, *ctrd*=0x3f), i shifts from 32 (for typical and best transistors corners) to 27 (for slow transistors, $T = 27°C$ and $V_{DD} = 1.1V$).

Process variations	Typical	Best case	Worst case
Index of the metastable latch	Q_{32}	Q_{32}	Q_{27}

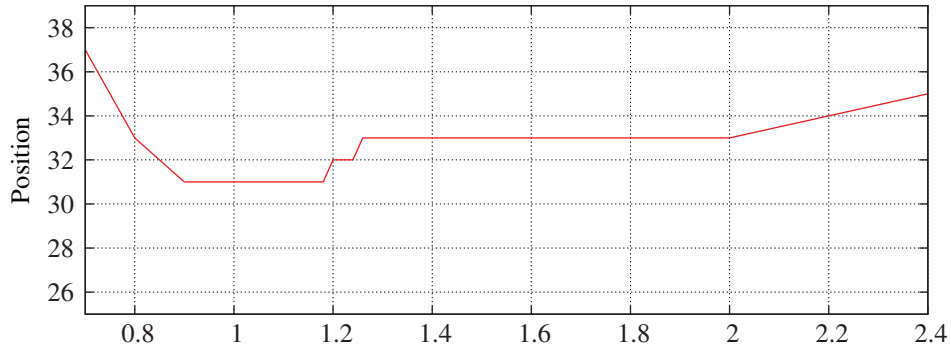
Table 4.3: Metastability position in the chain of latches depending on process variations for T=25 and Vdd=1.2.

The variation of the metastability position, i, toward temperature and supply voltage variations, is reported respectively in Figure 4.6a and Fig-

ure 4.6b. The position i is always in the range $[1 : 64]$ for a wide range of supply voltage and temperature. The proposed TRNG structure is robust against environmental perturbations. In fact, even in environmental perturbations, at least one latch is metastable.



(a) *vs.* Temperature(°C) @1.2V



(b) *vs.* Supply voltage(V) @25C°

Figure 4.6: Metastability position for ctr = 0x3 and $ctrd$ = 0x1.

4.2.4 Stochastic model versus simulation results

After presenting the simulation results, we propose to compare the post PAR simulation results to the stochastic model presented in Chapter 3. We propose to plot the probability p_{trngo} versus the noise standard deviation for all the delay configurations while considering Equation (3.20). To do so, we use Algorithm 3. We end up with 4096 delay values. In fact, N=64 latches multiplied by 8 data coarse delay configurations and 8 clock coarse delay configurations.

Algorithm 3 Algorithmic implementation of the probability computation at the delay chains based TRNG output.

NBLatches=64
CONFIG=8*8
DELAYS=[CONFIG][NBLatches]
$T_{setup0} = 27.87\ ps$
for ctr in (1,CONFIG) **do**
  for σ in (1,10) **do**
    for i in (1,NBLatches) **do**
      $\delta t_{DC}[i] = DELAYS[ctr][i]$
      $p_Q[i] = \frac{1}{2}.(1 - \frac{\mathrm{erf}(-\delta tDC[i]+T_{setup0}}{\sqrt{2}.\sigma})$
      $p = p.(1 - 2p_Q[i]))$
    end for
  end for$p_{trngo}[config] = \frac{1}{2}(1 - p)$
end for

Then we plot the p_{trngo} result for each of the 64 delay configurations in Figure 4.7. We notice from Figure 4.7 that:

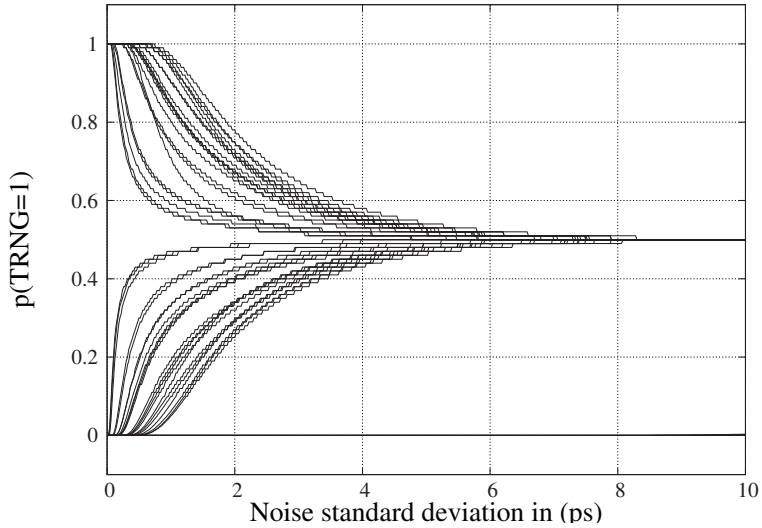


Figure 4.7: p(trngo='1') for all the possible delays configuration of (ctr,ctrd) for case (i)

- the proposed architecture can extract entropy almost equal to 1 for σ higher than 4 ps.
- according to the comparison between simulation results from Table 4.2 and waveforms in Figure 4.7, the standard deviation of the noise is

lower than 2 ps.

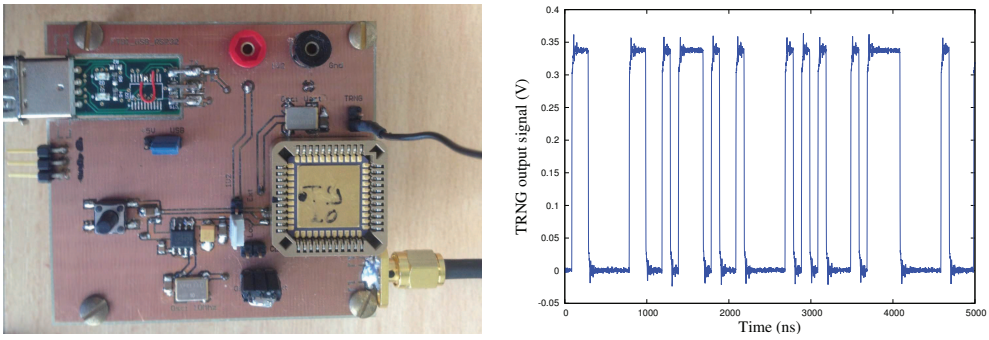
In the next section, we will present the measurement and statistical tests results that allow to validate the TRNG_ASIC#1 prototype.

4.3 ASIC Prototype Test and Validation of the TRNG with MUX-based Delay Chains

4.3.1 Experimental results and statistical evaluation on the ASIC TRNG

For the sake of relevant comparison, we use both AIS-31 and NIST to evaluate the TRNGs designed in this Ph.D work. In fact, most of the previously published TRNGs were evaluated using the NIST SP800-22 [RSN+10] battery of statistical tests. NIST statistical tests battery as well as the AIS-31 battery require a big amount of samples to perform the tests. Several millions of samples is needed depending on the test to be performed such as reported in the Table 2.4.

Twenty identical test-chips of the ASIC_TRNG#1 design were fabricated in the 65nm CMOS technology process by STMicroelectronics. This section presents experimental results and statistical evaluation of the ASIC_TRNG#1 presented in Section 4.2. Figure 4.8a shows the test board used for the acquisition of ASIC_TRNG#1 output sequences. Figure 4.8b shows the TRNG output signal on oscilloscope with a throughput of 10 Mbps.



(a) (b)

Figure 4.8: Test board and measurements of the ASIC_TRNG#1 prototype

From Table 4.4, we notice that all the latches output are random as they have different logic states for each of the 100000 samples. In fact, v_i represents the frequency of occurrences of the XOR output of the i^{th} latches packet at the 3^{rd} stage of XOR tree such as given in Equation (4.8). v_{trngo} represents the final frequency of occurrences of '1' at the raw output of the TRNG.

$$v_i = v\left(\bigoplus_{j=8*i}^{8*i+7} Q[j] = 1\right) \tag{4.8}$$

v_{trngo}	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
50.31	51.6	51.4	82.52	99.81	21.04	77.44	80.6	74.64

Table 4.4: Measurement results of the TRNG3 version for the delay configuration (ctr,ctrd)=(0x00,0x00) on 100000 samples.

First, we begin with applying the monobit test for AIS-31 standard. This test evaluates the frequencies of ones and zeros in the sequence. If these frequencies are not balanced, we should not engage the standard statistical tests because the chances to pass all of them is small. Table 4.5 reports the statistical tests results of ASIC samples for AIS-31 standards. Test procedure A (T0 to T5) are performed on von Neumann post-processed samples of 10 M bits generated at a bit rate of 100 Mbps. The test procedure B (T6, T7 and T8) are performed on raw samples as specified by [KS11]

TRNG version / **AIS-31 tests**	**TRNG0**	**TRNG1**	**TRNG2**	**TRNG3**
Disjointness test T0	pass	pass	pass	pass
Monobit test T1	pass	pass	pass	pass
Poker test T2	fail	pass	fail	pass
Run test T3	fail	pass	fail	pass
Long run test T4	pass	pass	pass	pass
Auto-correlation test T5	fail	pass	fail	pass
Uniform distribution test T6a	fail	fail	pass	pass
Uniform distribution test T6b	fail	pass	fail	pass
Test for homogeneity T7a	fail	fail	pass	pass
Test for homogeneity T7b	fail	fail	pass	pass
Entropy estimation test T8	fail	fail	fail	pass

Table 4.5: AIS-31 statistical tests for the ASIC TRNG (PTG.1 tests with Von Neumann PP and PTG.2 without PP).

For the TRNG3, we report that only the Entropy estimation test T8 results are not regular from chip to chip and from a sequence to another. In fact, depending on the chip the observed test variable varies from 6.86 and goes up to 7.9962. To be compliant with T8 test, AIS-31 standard claim an $H_{expected}$ of at least 7.976. Details of the AIS-31 statistical tests results are

given in Table 5.12 in Appendix. NIST statistical tests [RSN+10] are also performed on a raw sample of 10M bits acquired on the four versions of the ASIC_TRNG#1. The sequences are generated at a throughput of 100 Mbps. The acquisition was performed at 100MHz, the test board local oscillator. However the TRNG2 and TRNG3 version can operate at frequencies up to the GHz. This is not the case for TRNG0 and TRNG1 versions which are limited by a 100 MHz because of their longer delay chains. The statistical tests results are poor for the TRNG0 and TRNG1 versions. So, a von Neumann post-processing is applied [vN51]. The best statistical results concern the TRNG3 version sample. In fact, it fails only the Universal test out of the 15 NIST tests. AIS-31 and NIST statistical tests results are very much consistent with each other. In fact, the Maurer's Universal test of NIST is the same test as T8 entropy estimation test of AIS-31.

TRNG version / **NIST tests**	**TRNG0**	**TRNG1**	**TRNG2**	**TRNG3**
Frequency	pass	pass	pass	pass
Block Frequency	pass	pass	pass	pass
Cumulative Sums	pass	pass	pass	pass
Runs	pass	pass	pass	pass
Longest Runs	pass	pass	pass	pass
Rank	pass	pass	pass	pass
FFT	fail	fail	pass	pass
Non Overlapping Template Matching	pass	146/148	147/148	pass
Overlapping Template Matching	fail	pass	pass	pass
Universal	pass	pass	pass	fail
Approximate Entropy	pass	pass	pass	pass
Random Excursions	pass	pass	17/18	pass
Random Excursions Variant	pass	pass	pass	pass
Serial	pass	pass	pass	pass
Linear Complexity	pass	pass	pass	pass

Table 4.6: NIST statistical tests results of the ASIC TRNG

Details of the NIST statistical tests results are given in Table 5.9 in Appendix. Statistical tests results on the ASIC_TRNG#1 from Tables 4.5 and 4.6 show that the best version is the TRNG3. We identify 3 reasons that make the TRNG version present higher entropy:

- small differential δt_{DC} all along the latches
- the use of custom latches designed with multiplexors
- very high slope to capture the vertical noise

In the following, we present environmental variations tests of the TRNG3 version.

4.3.2 Environmental variations tests of delay chains based TRNG

Process variations, temperature and V_{DD} variations affect both interconnect and propagations delays. For this matter, a self-calibration block, which sets the optimal delay configuration of the TRNG coarse delay chains, is added to the TRNG. The TRNG chip may be exposed to certain environmental perturbations such as temperature and power supply voltage variations. Figure 4.9 illustrates the potential parameters that may affect the TRNG output.

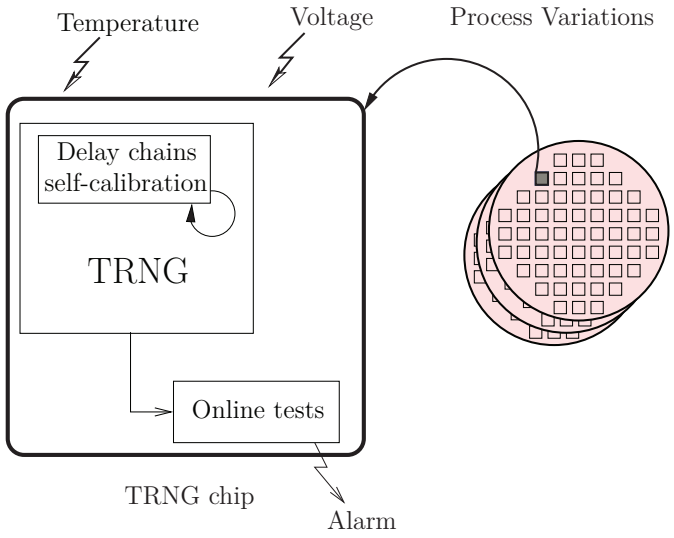


Figure 4.9: TRNG system

All CMOS on-chip TRNG structures are sensitive to environment perturbations. As physical attacks are an important issue in the cryptography field, the coarse chains are implemented to adjust the delays and assure that the TRNG works within a wide temperature and supply voltage ranges. A self-calibration block selects the best delay control configuration and controls the coarse delay chains in a feedback mode. The self-calibration block is embedded within the TRNG as a feedback control of the coarse delay chains as represented in Figure 4.9. This TRNG auto-calibration block has been proposed in [DGH09]. This block computes the frequency of ones at the TRNG raw output for each configuration of the coarse delay chain at the TRNG start-up. Then, it sets the input of the coarse delay chains to the configuration that corresponds to the optimal frequency. The algorithm of the self-calibration of the TRNG coarse delay chains is presented in Algorithm 4

The data-to-clock differential delay may change in case of malevolent or environmental perturbations so as the index i. So self-calibration method

Algorithm 4 Algorithmic implementation of the self-calibration technique.

c=1
NCONFIG=2.M
NSAMPLES=256
$\nu_{opt} = \frac{NSAMPLES}{2}$
$\epsilon_{opt} = 16$
$\nu_{trngo} = 0$
for c in (2,NCONFIG) **do**
  for i in (1,NSAMPLES) **do**
    return $trngo[i]$
    if $trngo[i] = 1$ **then**
      $\nu_{trngo} = \nu_{trngo} + 1$
    end if
  end for
  $E = |\nu_{trngo} - \nu_{opt}|$
  if $E > \epsilon_{opt}$ **then**
    $ctr_{opt} = c$
  end if
end for
$ctr = ctr_{opt}$

turns out to be very effective to set the coarse chains on the effective delay configurations to balance out the perturbations. However, it has a drawback. We have to throw the bit stream which was used when applying the calibration. The more configurations we have, more clock cycles are needed to calibrate, hence more bits to throw. The length of non-used sequence, L, is given in Equation (4.9).

$$L = NSAMPLES \times NCONFIG \tag{4.9}$$

Table 4.7 reports the frequency of occurrences of '1' at the output of the TRNG for 7 different dies of the ASIC prototype. The optimal coarse delay chains configuration (ctr, ctrd) differs from die to die.

Die #	1	2	3	4	5	6	7
Optimal (CTR,CTRD)	(x1f,x3f)	(x03,x01)	(x03,x03)	(x01,x01)	(x0f,x0f)	(x1f,x3f)	(x0f,x1f)
$\nu(trngo =' 1')$	51.71	50.58	46.77	50.9	53.93	55.72	51.64

Table 4.7: Impact of process variations on the TRNG.

In Table 4.8, we report the NIST statistical tests results on a sequence generated at $-10°C$ and $70°C$.

NIST tests \ Temperature	$-10°C$	$70°C$
Frequency	pass	pass
Block Frequency	pass	pass
Cumulative Sums	pass	pass
Runs	pass	pass
Longest Runs	pass	pass
Rank	pass	pass
FFT	pass	pass
Non Overlapping Template Matching	146/148	146/148
Overlapping Template Matching	pass	pass
Universal	fail	fail
Approximate Entropy	pass	pass
Random Excursions	pass	pass
Random Excursions Variant	pass	pass
Serial	pass	pass
Linear Complexity	pass	pass

Table 4.8: NIST statistical tests results of the ASIC TRNG3 version at $-10°C$ and $70°C$

CMOS circuits are tolerant to work correctly only for $\pm 10\%$ of V_{DD} as specified by the manufacturer. A common ways of disturbing a cryptographic CMOS IC device are FIA attacks as discussed in Section 2.4. In fact, an attacker can over or under supply the TRNG to bias its output. The lower V_{DD} is, the higher I_{DS} is [KK06], and the higher the propagation delays are. This implies that the MUXes propagation delays increase with the increase of V_{DD}. For the delay chains based TRNG, if propagation delays change, the optimal configuration of the coarse delay chains change. For that matter, power supply voltage variations have been applied to the TRNG ASIC prototype to assure that at least one delay configuration is good. Figure 4.10 reports the TRNG output frequency of occurrences of '1' versus the power supply voltage V_{DD}. In Table 4.9, we report the optimal delay configuration of the coarse chain *vs.* the power supply voltage variation on the test-chip for 10K samples of TRNG output.

The optimal delay configuration is generated by the self-calibration block. Temperature variations have also been performed on the ASIC test-chip of the delay chains based TRNG. In CMOS technology, the higher the temperature

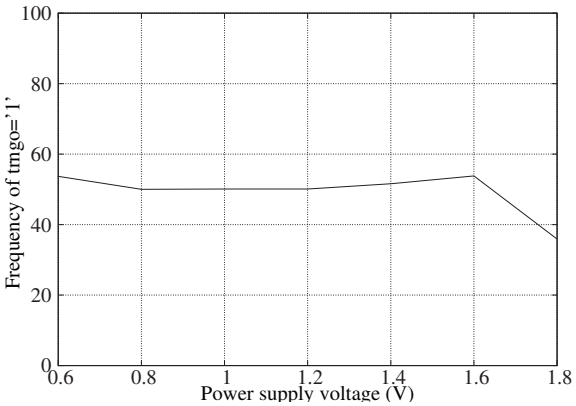


Figure 4.10: Frequency of occurrences of '1' at TRNG3 output *vs.* power supply voltage variation for 10K samples.

V_{DD}(V)	1.8	1.6	1.4	1.2	1	0.8	0.6
Optimal (ctr,ctrd)	(x0f,x3f)	(x0f,x3f)	(x0f,x3f)	(x0,x0)	(x0f,x1f)	(x1f,x3f)	(x01,x00)

Table 4.9: Optimal coarse delay chain configuration *vs.* power supply voltage.

is, lower are the propagation delays. [KK06] and [Aas12] present a study of the impact of temperature fluctuations on CMOS technologies. In the coarse delay chains, the optimal delay configuration (ctr,ctrd) of the TRNG differs with temperature variation. Tests and measurements while varying the temperature of the test-chip have been carried out in the laboratory using a cooling and heating oven. Figure 4.11 reports the observed TRNG output frequency of occurrences of '1' versus temperature variations. Table 4.10 reports the optimal delay configuration of the coarse delay chains *vs.* the temperature variations for which we obtain around 50% of '1' at the TRNG output sequence.

T°C	-10	0	27	50	70
Optimal (ctr,ctrd)	(x07,x07)	(x01,x01)	(x00,x00)	(x00,x00)	(x00,x00)

Table 4.10: Optimal coarse delay chain configuration *vs.* temperature.

As expected, Table 4.10 demonstrates that the MUXes propagation delays increase when the temperature increases. In fact, in the case of typical and

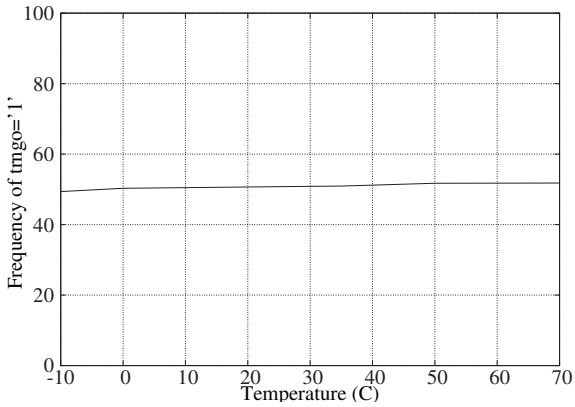


Figure 4.11: Frequency of occurrences of '1' at TRNG3 output *vs.* temperature variations for 10K samples.

high temperatures (50°C and 70°C), the data signal reaches the metastability window more rapidly than at low temperatures (−10°C and 0°C). We notice that the measurements on ASIC prototype for temperature and power supply voltage variations results of Figures 4.10 and 4.11 confirm the simulation results of temperature and V_{DD} variations of Figure 4.6a and Figure 4.6b respectively. The self-calibration block introduced in this chapter has proven to be very important not only in case of environmental variations but also for process variations as the propagation delays of the TRNG varies from die to die.

4.3.3 Model versus ASIC prototype results

Modeling results of Figure 4.12 are confronted with real samples from the prototype ASIC test-chip ASIC_TRNG#1. Table 4.11 reports the number of occurrences of '1' at the output of the ASIC TRNG for different values of *ctr* and *ctrd*. When we compare the values reported in Table 4.11 with the waveforms of Figure 4.12, we can estimate the noise standard deviation. We plot a vertical line varying from $1ps$ to $10ps$ and investigate on which σ matches best the values from Table 4.11. For example, for the coarse chain configuration (ctr=0x00, ctrd=0x00), the probability p_{TRNG} measured on the test-chip over 100000 samples as in Table 4.11 is 84.25%. This shows that the proposed model matches well the measurement results extracted from the ASIC prototype. We deduce that the noise standard deviation of the test-chip is around 2 *ps*.

The previous results help us to enhance the ASIC_TRNG#1 architecture by designing more precise delay chains. The next section describes the new ASIC architecture. From the comparison study between post PAR simulation results and measurements on the ASIC_TRNG#1 test-chip, we deduce that

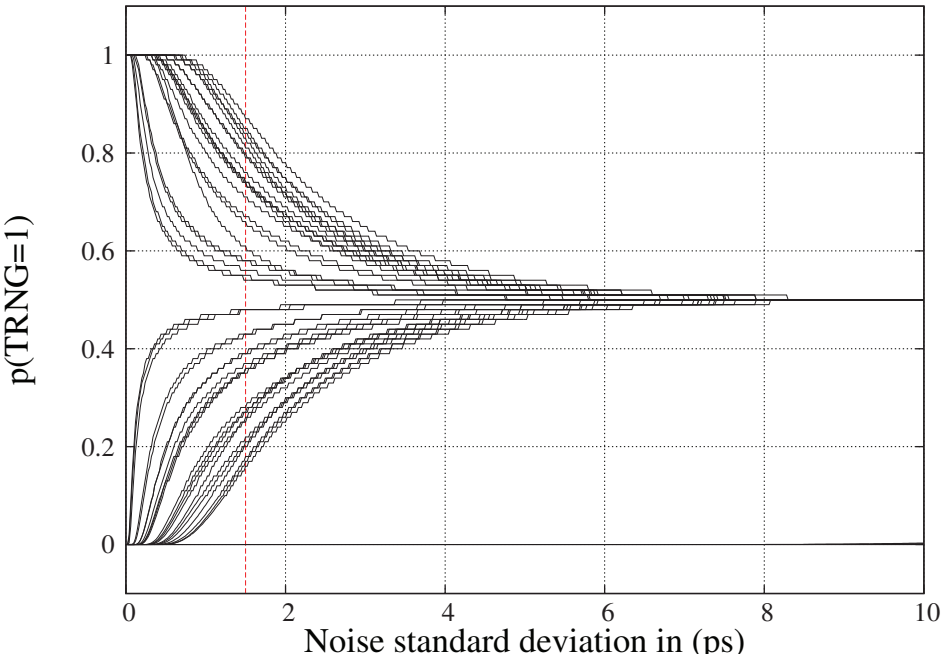


Figure 4.12: The probability p_{TRNG} at the TRNG raw output in terms of the noise standard deviation for all delay configurations from analytic expression

the standard deviation of the noise is around 2 ps. 2 ps represents the uncertainty around the delay between data and clock. Although the fine tuning step of the Fine delay chains of the ASIC_TRNG#1 is equal to 5 ps. We need to refine this value to reach a resolution lower than 2 ps. Hence, we need to enhance the precision of the fine delay chains The coarse tuning step of the MUX-based coarse chains is not optimal neither. It is equal to 35 ps. This makes a lot of configurations non-usable as we can see from Figure 4.13. In fact the configurations that works the best corresponds to a data-to-clock delay in the range of $[0\ ps, 100\ ps]$ as we can see in the gray range in Figure 4.14. Only the gray zone configurations generate about 50% of *trngo* at '1' with a 3.5% error rate around 50% such as required in the *monobit* test of AIS-31 [KS11]. Only 16 configurations, at most, allow good measurement results. This represents only a $\frac{1}{4}$ of the total number of possible configurations. So we should design a coarse chain with a thinner tuning step, such as we can sweep into the metastable region with a fine precision. Previous transient simulations of bi–stable standard cells are done to characterize t_{setup} limit of metastability appearance. For the mux_latch it has been shown that the T_{setup0} equals to 27.8742 ps. This value should be integrated in the sizing process to assure the metastability appearance.

Comparison of measurement and statistical tests results between the

ctrd \ *ctr*	0x00	0x01	0x03	0x07	0x0F	0x1f	0x3f	0x7f
0x00	84.25	95.97	77.91	94.13	88.45	4.98	8.11	91.79
0x01	0.14	0.68	21.43	49.65	96.93	100	94.88	55.91
0x03	100	100	74.94	53.8	98.95	99.99	2.16	99.87
0x07	0.11	7.23	98.49	99.73	74.98	0.28	27.95	100
0x0f	0	0	100	97.39	99.7	26.31	49.83	63.27
0x1f	0	0	93.9	44.96	28.17	76.4	94.09	9.37
0x3f	0	0	0	58.84	2.16	40.51	32.82	99.97
0x7f	0	0	0	1.6	38.2	99.98	5.47	66.77

Table 4.11: The probability p_{TRNG} at the TRNG raw output for all delay configurations from ASIC prototype measurements.

different versions of the ASIC_TRNG#1 shows also that TRNG2 and 3 versions present better results that the TRNG0 and 1 versions thanks not only to thiner fine delay chains but also to a very high slope of the data signal. In fact, in the TRNG2 and TRNG3 versions, the fine delay chains are maid only of routing wires. Hence the BFX2 output of the coarse chains feeds 64 stages of latches which makes the data slope extremely high. The comparison between the four different versions of the ASIC_TRNG#1 makes us also deduce that using a custom latch and place and route the cells in a balanced way assure better results.

Besides, the TRNG2 and TRNG3 versions present better results first because the fine delay chain is very thin. Second, unlike TRNG0 and TRNG1, TRNG2 and TRNG3 do not use buffering elements on the fine chain hence the slope is very high. This make the structure of TRNG2 and TRNG3 sensitive to the voltage uncertainty around the $MSS = \frac{V_{DD}}{2}$ position. In fact, higher is the slope, higher is the sensitivity of latches to the vertical noise. Consequently, we propose to study a new architecture of delay chains based TRNG by enhancing the fine chains and coarse chain precision. In the next sub-section, we describe the specifications we set for the new ASIC design and present the principle of our novel architecture.

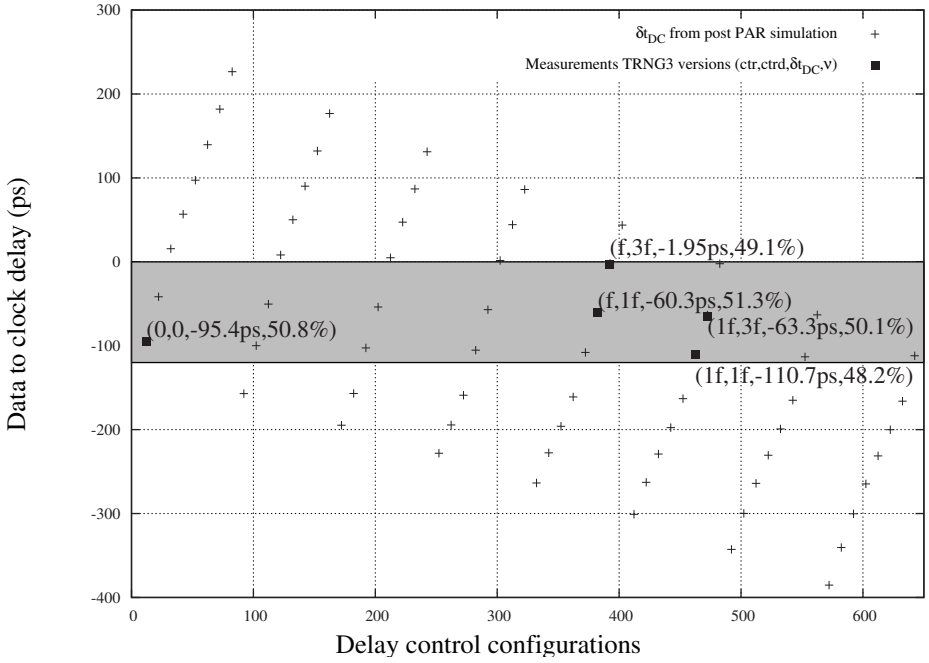


Figure 4.13: ASIC_TRNG#1: measurement results vs. post PAR extracted delays

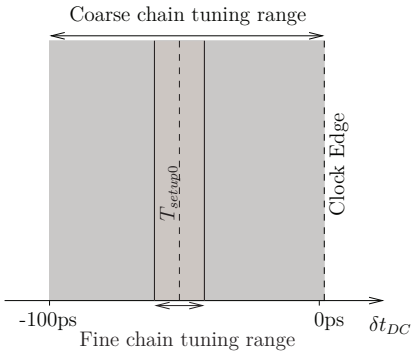


Figure 4.14: Optimal region for randomness generation from comparison between test-chip measurement results and post PAR extracted delays.

4.4 TRNG Design with Tri-states Buffers based Delay Chains

4.4.1 Principle of the tri-state buffers based delay chains

Based on the previous conclusions on the ASIC_TRNG#1 characterisations in the Section 4.3.3, we should:

- control with more precision the delays between the clock and data along the fine chain,
- control the coarse delay with higher precision,
- decrease the latches chain,
- make the clock signal slope sharp and
- make the data signal slope very high.

We propose to use a delay chain designed with controllable tri-state buffers as represented in the Figure 4.15.

Tri-state buffers drive strength of the upper block 1 and the lower block 2 are identical. However we use different delay elements (a buffer) on the input of each block and consider the differential delay between them. The global delay between the input delay block and the output delay block depends on:

- $d2 - d1$: the differential delay between the upper delay chain and the lower one.
- N_u: the number of active tri-state buffers in the upper delay chain
- N_l: the number of active tri-state buffers in the lower delay chain.

We set the same number of active tri-state buffers on the block 1, N_u as on the block 2, N_l, to get the same slope for each configuration and an almost constant δt according to Equation (4.10) such as the waveforms of Figure 4.16b.

$$\delta t = \frac{d2 - d1}{N} \tag{4.10}$$

where $N = N_u = N_l$. In Figure 4.16a, the tri-state buffers outputs have different slopes.

4.4.2 Design description of the TRNG with tri-state buffers based delay chains

Three delay blocks are used to control the data-to-clock delay around the metastable region (i.e. around T_{setup0}). The architecture of the TRNG using

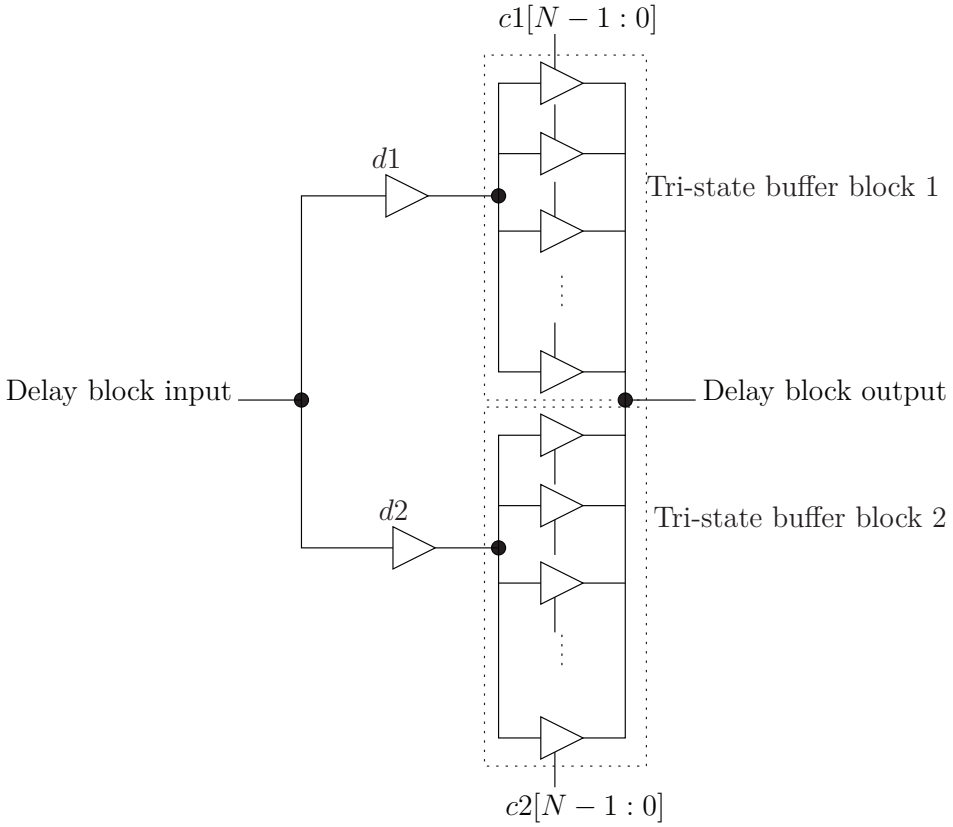


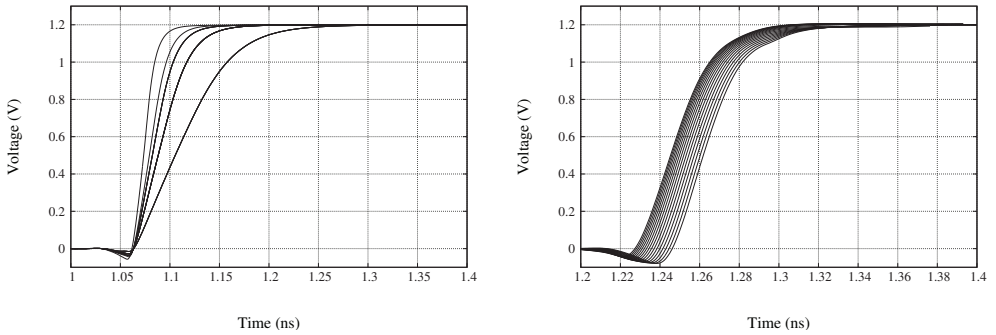
Figure 4.15: Differential delay principle using tri-state buffers.

the tri-state-buffers-based delay chains is presented in Figure 4.17 The tri-state buffers block on data is called the small delay chain. It is used to control finely the delay on data. The second tri-state buffers block on clock is called the average delay chain which is used to control finely the delay on clock. A third block is used to control coarsely the delay on clock. It is called the large delay block.

We need to specify:

- The number NT of tri-state buffers constituting each of the 3 delay blocks that introduce a small, medium and large tuning step delay. The total delay introduced by the large block, ΔTl, is arbitrarily chosen at 30 ps with a coarse tuning delay step of $\delta tl = \frac{\Delta Tl}{NT}$. Hence, the ΔTa have to be greater than δtl which means that the medium delay tuning step, δta, should verify Condition (4.11).

$$\delta ta > \frac{\delta tl}{NT} \tag{4.11}$$



(a) For all configurations (b) For configurations assuring the same slope

Figure 4.16: 2^N possible delays through tri-state buffers based delay blocks.

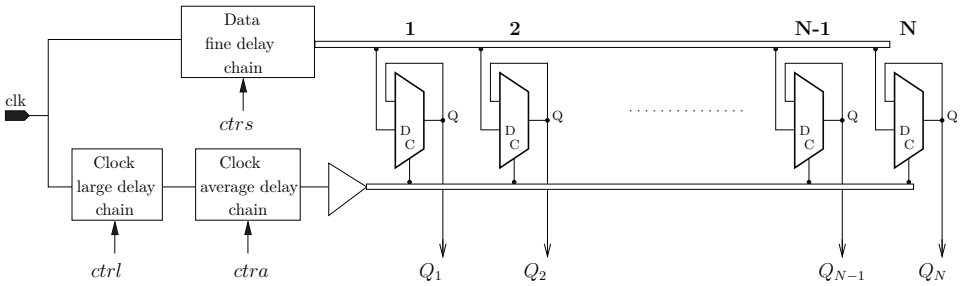


Figure 4.17: Architecture of the delay chains based TRNG using tri-state buffers.

And finally the ΔTs have to be greater than δta which means that the small delay tuning step,δts, should verify Condition (4.12).

$$\delta ts > \frac{\delta ta}{NT} \tag{4.12}$$

So, we summarize these conditions into Equation (4.14).

- Buffers which introduce a global initial delay on the clock equal to T_{setup0} such as given in Equation (4.13).
- Buffers on data and clock inputs of the latches assuring a high clock slope and a low data slope.
- The number of latches: As the latches represent the load, the number of latches N influences the clock and data slopes. The larger N is, the greater is the slope.

The following conditions are used to specify $d1$ and $d2$ buffers drive strength for each of the three delay tri-state blocks (Figure 4.15).

$$clk\_del = T_{setup0} \tag{4.13}$$

$$\delta ts = \frac{\frac{\frac{\Delta Tl}{NT} \times (1+\frac{m}{100})}{NT} \times (1+\frac{m}{100})}{NT} = \frac{\Delta Tl}{NT^3} \times (1+\frac{m}{100})^2 \tag{4.14}$$

where:

- m represents the overlapping margin we set arbitrarily for the 3 blocks.
- NT the number of tri-state buffers used which corresponds to the number of possible delay intervals.
- δts the tuning delay step of the small delay block.

Later we will be referring to the 3-uplet $(ctr\_large, ctr\_average, ctr\_small)$ respectively the decimal values of the control bits of the large, average and small delay blocks. Next, we move to the place and route process and the simulations of the tri-state buffers delay chains based TRNG.

4.4.3 Post place and route simulations

The TRNG layout was carried out with special attention to the placement of the delay chains standard cells. The tri-states buffers constituting each delay chain are tightly placed in columns to assure small wire lengths and thus small delays. Figure 4.18 presents the layout of the tri-state buffers delay chains based TRNG. Latches are tightly placed in a single row to guarantee also small wire lengths between them.

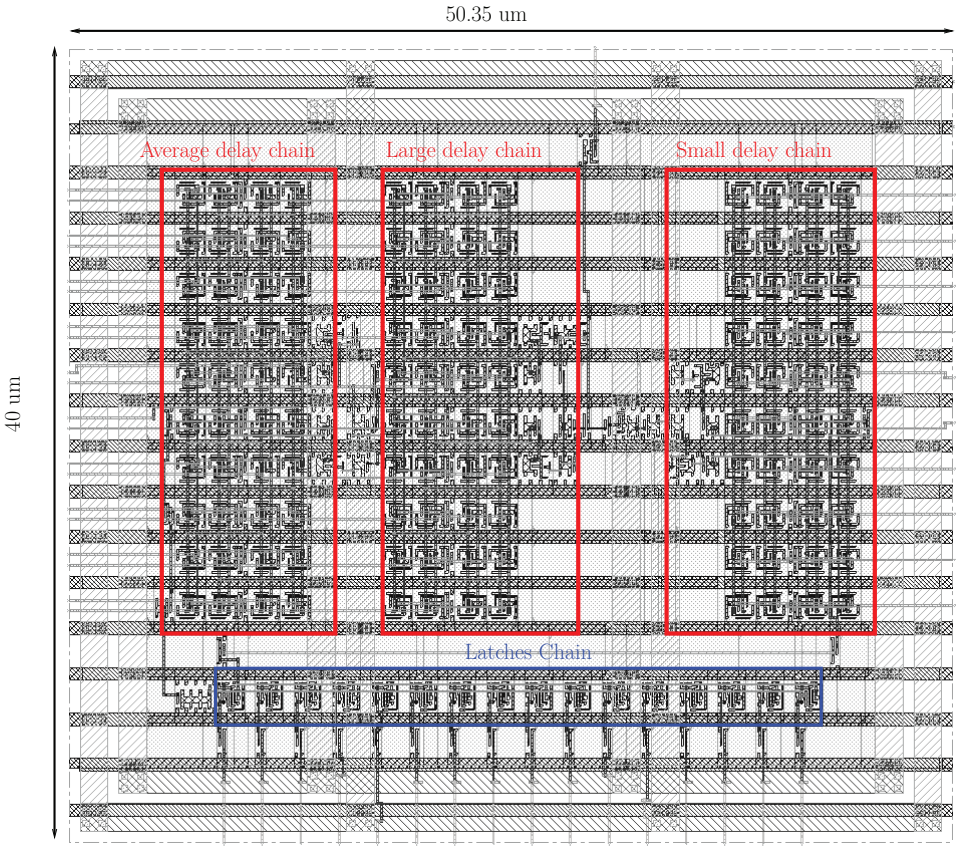


Figure 4.18: Layout capture of the tri-state buffers delay chains based TRNG.

After the place and route process, we perform post layout parasitics extraction. Then, post layout simulations are carried out for all delay configurations. Figure 4.19 shows all the possible delays for the $1331 = 11^3$ possibles cases. If we zoom this figure we can see that we managed to sweep the whole metastable region with a step of the order of tens of femto seconds where:

- $\Delta Ts = 0.5\ ps \rightarrow \delta ts = \frac{\Delta Ts}{10} \simeq 0.05\ ps$ the tuning delay step of the small delay block.

- $\Delta Ta = 3.7\ ps \rightarrow \delta ta = \frac{\Delta Ta}{10} \simeq 0.37\ ps$ the tuning delay step of the average delay block.
- $\Delta Tl = 30\ ps \rightarrow \delta tl = \frac{\Delta Ta}{10} \simeq 3\ ps$ the tuning delay step of the average delay block.

The fine tuning delay, δt introduced by the routing wires along the latches chains is equal to $\frac{0.44}{16} = 0.02\ ps$ where $N = 16$ latches. This small tuning delay allows to sample the data signal at multiple latches in the metastable region. Figure 4.20 shows that $Q[0]$ state flips for the position $(15, 7, 31)$ for

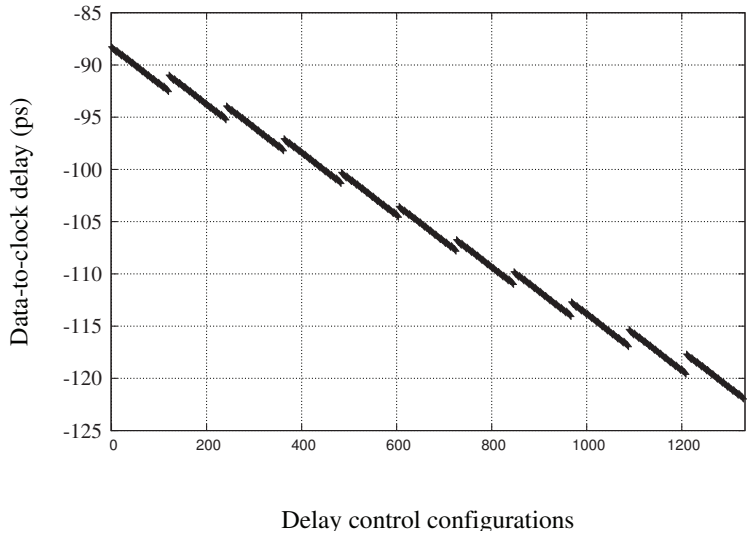


Figure 4.19: Feasible delays for different *ctr_large*, *ctr_average* and *ctr_small* from post PAR simulation for typical corners.

typical corners. Figure 4.21 presents the waveforms the 16 latches outputs in typical case without any noise. It shows that all the latches outputs are metastable for the delay configurations $(63, 63, 31)$. Besides, some outputs settle down to 0V and others settle down to V_{DD}. The high precision of the delay tuning allows to make all the latches operate in the metastable region for several delay configurations. We also performed the same simulations for worst and best cases of working conditions. Table 4.12 also reports the optimal delay configuration for which some latches present a metastable behavior.

Post PAR simulation results shows that we managed to design a very high precision delay chains. Post PAR simulation results of the tri-state buffers based TRNG are better than the TRNG designed with multiplexors based delay chains. This very fine tuning capacity allows to extract a high entropy at the TRNG output. Besides, as all the 16 latches are metastable, we do

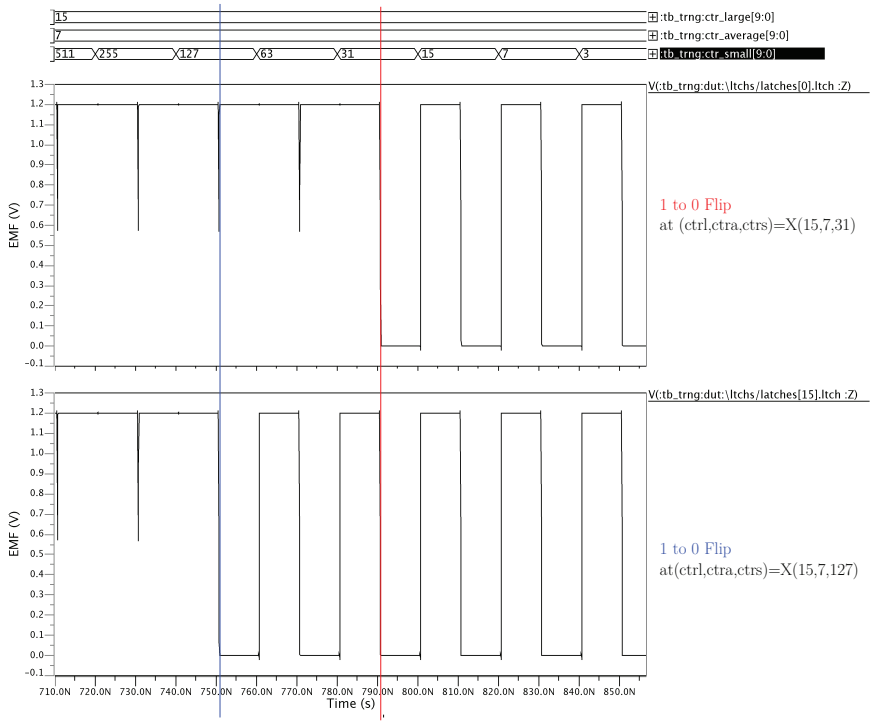


Figure 4.20: Q[0] and Q[15] waveforms shows the state flipping respectively for (ctr_large, $ctr\_average$,ctr_small)=$(15, 7, 31)$ and $(15, 7, 127)$ from post PAR simulation

not apply the XOR at the latches output. This multiplies the TRNG bit rate by 16. The post PAR simulation results of the ASIC_TRNG#2 introduced in this chapter assure a very promising novel TRNG design. At the time of writing this manuscript, test-chips of ASIC_TRNG#2 prototype are not yet delivered. Standard statistical tests will be performed in our laboratory as soon as they are delivered.

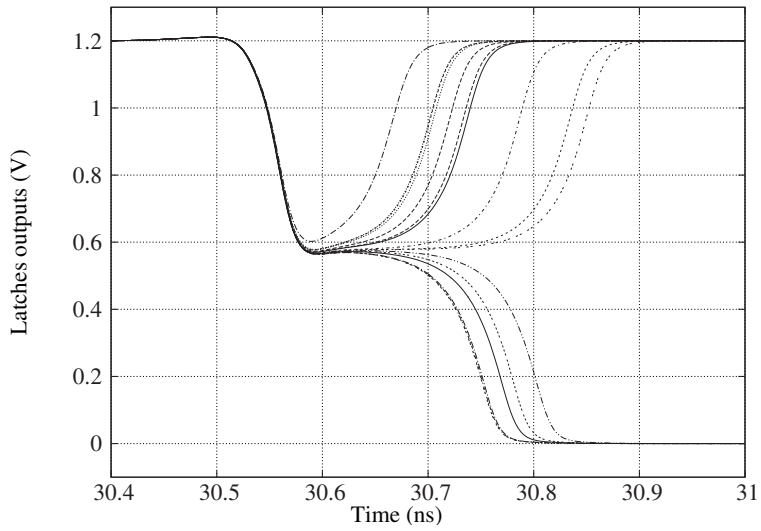


Figure 4.21: Metastable states on latches chain outputs at (*ctr_large*, *ctr_average*, *ctr_small*)=(63,31,31) from post PAR simulation.

Process	**Temperature(°C)**	**VDD(V)**	**Optimal delay configuration** (*ctr_large*, *ctr_average*, *ctr_small*)
Fast-Fast	-40	1.3	(3,63,15)
Typical-Typical	25	1.2	(15,7,31)
Slow-Slow	105	1.1	(63,31,7)

Table 4.12: Optimal delay configuration depending on process variation and environmental conditions.

4.5 Conclusion

In this chapter we presented the ASIC implementation of the delay chains based TRNG, the following methodology have been used:

- First, choose narrowly TRNG elements in the cells library to assure the metastability appearance, then
- carefully constrain the TRNG layout and integrate the TRNG as a black box block into an automatically PAR test-chip design,
- perform simulations in typical, best and worst case conditions with the addition of noise
- confront the stochastic model proposed in the previous chapter to the test-chip measurement
- finally, apply statistical standard tests to evaluate the randomness of the delay-chains based TRNG implemented in ASIC.

We presented the measurement of the impact of temperature, power supply voltage and process variations on the ASIC_TRNG#1. The measurement results on the TRNG prototype are consistent with theoretical and simulation results. After characterisation and evaluation of the ASIC_TRNG#1, we proposed an enhancement of the delay chains based TRNG. The new proposed TRNG architecture allows:

- very fine delay tuning
- high entropy extraction
- to multiply the bit rate by 16

Measurements on the second version of the delay chains based TRNG, ASIC_TRNG#2 proposed in the second section of this chapter will be carried out in our laboratory as soon as the test-chips are delivered.

Chapter 5

Study of the Delay Chains based True Random Number Generator for FPGA Technology

5.1 Introduction

The TRNG architecture of the delay chains based TRNG is highly sensitive to any routing imbalance in the design. The FPGA implementation of such an architecture is more challenging than its implementation in ASIC technology. This is mainly due to the fact that Xilinx FPGA vendor do not provide any information about how to constrain the routing of a design. In this work, we propose to counteract the automatic place and route process to apply the delay chains specifications of the TRNG. In the section 5.2, we present the design methodology to implement the TRNG in FPGA technology while taking care on the delays during the placement and routing steps. Section 5.3 is dedicated to the primary results of the FPGA implementation of the delay chains based TRNG. Finally to validate the TRNG implementation, we present the experimental results and statistical evaluation of the sequence generated on the Virtex 5 FPGA.

5.2 FPGA Implementation of the Delay Chains based TRNG

Thanks to their flexibility, reconfigurability and short time-to-market, FPGAs are extensively used in cryptography research. This is why the need for FPGA-based TRNGs has grown. The design of delay chains based TRNG is more challenging on FPGA technology than in ASIC technology. First, because of the difficulty to manipulate the routing in FPGAs. Second, unlike ASIC, transient noise models of the FPGA primitives are not provided by the vendor. Hence, unlike ASIC technologies, we cannot simulate the TRNG random behavior. However, we can program the FPGA as many times as we want until we find the optimal delay chains design which provide the best results. In the following subsections, we present specifications and problem

statement of the adaptation of the delay chains based TRNG to FPGA technology. In fact, in the first sub-section, we study the feasibility of the generic design and the idea that we propose to get small differential delays. Then, we present the design methodology to implement the delay chains based TRNG in Xilinx FPGAs.

5.2.1 Specifications and problem statement

As the randomness quality of the generated sequence depends closely on the shrewdness of the delay along the latches, we need to check the quality of the output each time we modify the routing. In fact, with a first acquisition on FPGA of some hundreds of bits, we can not simply tell if the TRNG output is random enough, we need each time to run the standard statistical tests. So basically we have to proceed with the trial-and-error method such as represented in the diagram of Figure 5.1. In case of the statistical tests failure we need to readjust the placement constraints. We re-iterate each of the steps of Figure 5.1 until it passes all the AIS-31 statistical tests.

Next, we describe the design methodology that we propose in order to balance the routing inside the TRNG design on the Virtex 5 FPGA.

5.2.2 Design methodology

5.2.2.1 Delay chains specifications

We propose to implement the delay chains on data and clock paths using Look-Up-Tables (LUT) with different inputs. Depending on the LUT input used, we can get different delays according to the routing path used through the switch box. In fact, this difference of delays is not due to the propagation time of the LUT itself. It is due to the different routing paths dedicated to each LUT input. This is illustrated in Figure 5.2.

So we propose to use different LUT input for each delay chain and consider the differential delay introduced by the differences in the routing paths. This is illustrated in different colors in Figure 5.3. The data delay chain is routed through the I3 LUT input and the clock delay chain is routed through the I5 LUT input. The challenge of designing the delay chains based TRNG structure of Figure 5.3 lies in the placement and routing process. The PAR process needs to be very carefully considered and checked to meet the delay chains specifications.

The fine chain on the clock path has to be the clone of the fine chain on the data path(i.e. the reproduction of its placement and routing) in a way that both data and clock fine delay chains verify respectively Conditions (C1) and (C2).

$$\delta l_D i = \delta l_D \, \forall i \in [1, N] \tag{C1}$$

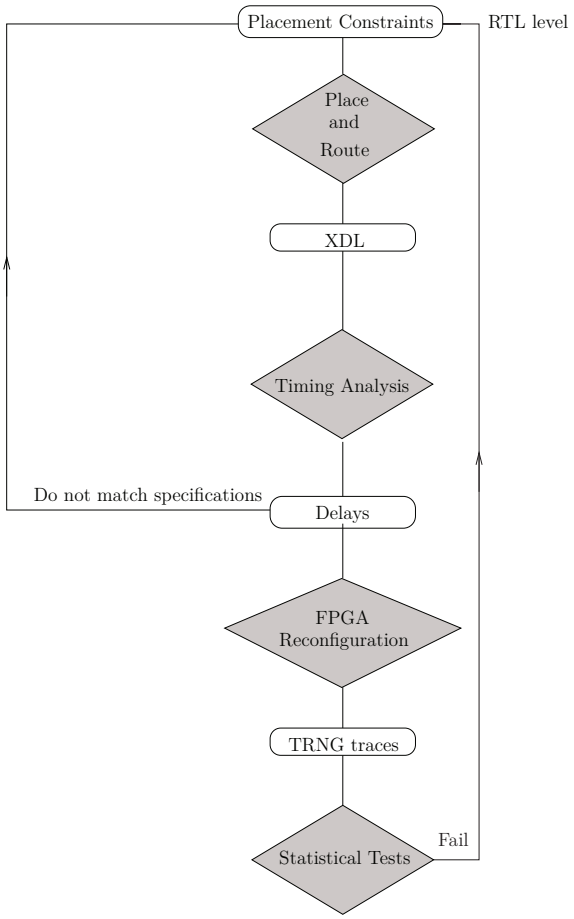


Figure 5.1: Trial-and-error procedure for the TRNG design on FPGA

$$\delta l_C i = \delta l_C \, \forall i \in [1, N] \tag{C2}$$

If the differential delay, $\delta t_{DC} = \delta l_D - \delta l_C$ is not zero, we can perceive a race between the data and the clock signals along the fine delay chains. At each latch, one signal catches up the other with a δt_{DC} increment.

Xilinx CLB contains two registers that can be configured as flip-flops or latches (level sensitive mode). However, these primitives are hardened to avoid metastability [Alf05, Xil97]. In fact, their Mean Time Between Failure (MTBF) is very high as presented by Xilinx in the reliability application note of Virtex 5 device [Cha]. We decided thus to implement a custom latch using an LUT. The proposed structure is designed with an LUT configured as a MUX21 where the MUX output is forwarded back to an input and the selection input acts as the enable input of a latch. This way, if S='1', the custom latch is in data copying mode else if S='0', the custom latch is in the storage mode.

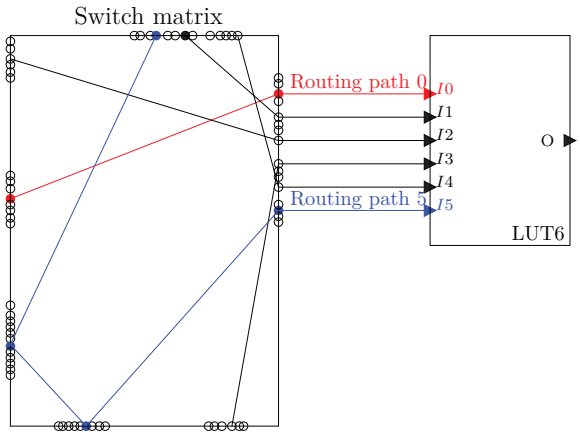


Figure 5.2: Different routing paths inducing different delays.

5.2.2.2 Place and route process

The problem here is that the Xilinx place and route tools perform many verification and timing optimisations all along the design flow. In order to avoid these optimisations, we need to hard-protect the delay chains in order to guarantee the same placement and routing along the chains.

Xilinx provides some specific features that permit to add placement constraints options on the RTL level and disable any undesired optimisation. Xilinx user guide document [Xil09] gives details on the syntax of the placement constraints to use in the RTL. The constraints we use for the TRNG design are cited below:

- `SAVE NET FLAG (S)`: When used on a net, this placement constraint prevents from the net removal.
- `LOCK_PINS`: The LOCK_PINS constraint is used on a primitive to avoid the LUT inputs swapping. Meaning that when adding this constraint on a LUT instantiation, we assure that the placement tool does not make any modification to the LUT mapping specified by the user on the RTL code.
- `KEEP` Prevent from undesired optimisation. Since we instantiate 6-input LUT with a true boolean function in the truth table, Xilinx tool would manage to entirely remove the LUT used and replace it by a simple wire.
- Explicitly specify the truth table. When instantiating a primitive in the RTL code we can specify the truth table. Through the truth table we can force the use of specific inputs of a slice or a LUT.

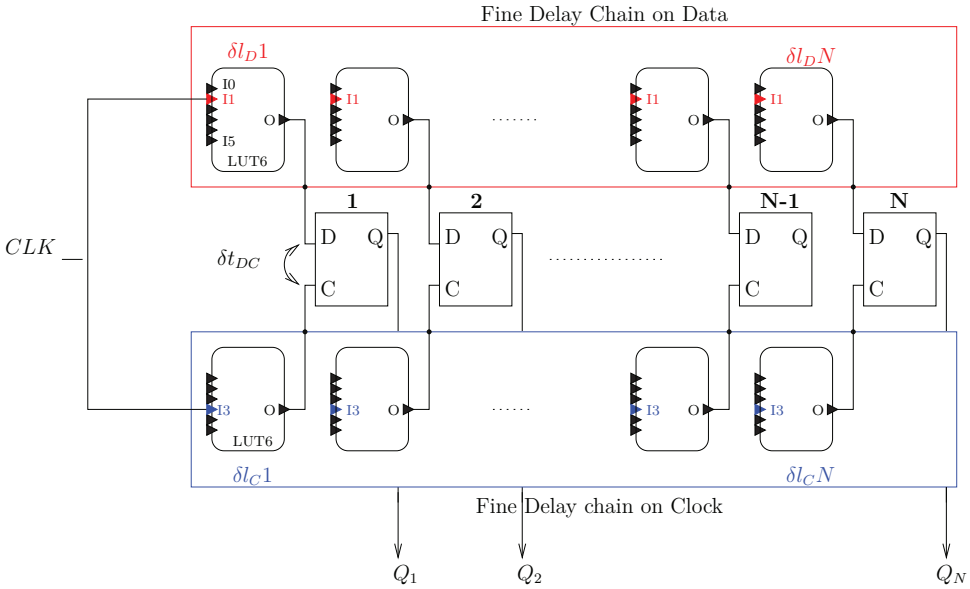


Figure 5.3: LUT based implementation of the fine delay chains of the TRNG.

- `LOC, RLOC, RLOC_ORIGIN`: These constraints are very convenient in our case since they allow to specify the location of the SLICE used. `LOC` would force the tool to use a primitive of an exactly specified SLICE X and Y coordinates. `RLOC` different from the LOC constraints as it makes the placement tool use a relative position to another block instead of specifying an absolute position. `RLOC_ORIGIN` is used to specify the X and Y coordinates of a hard macro.

We set the truth table of the delay element as a bit wise AND function. Then we add the following constraints to the RTL source code of the delay element module:

- Put the `SAVE NET FLAG (S)` constrain on the input nets of the 6-input LUT
- Set the option `LOCK_PINS` constrain on each LUT used
- Use the `KEEP` constrain to make sure all the nets of the bit wise AND is kept

To make the tool route through all the routing resources of each switch box of each delay element, we configurate the LUT as a bit wise AND. Later we would extract all the corresponding delays to compare and choose the optimal LUT input considering the fine delay chains specifications. Then to make sure that fine chains are identical we use hard macros. A hard macro is

a part of a placed and routed design. It allows to preserve the placement and routing inside an instantiated black box if it is used in the same resources.

For each hard macro design (coarse chain hard macro and fine chain hard macro), we go through the same flow. Figure 5.4 illustrates the hard macro generation flow of each of the delay blocks. Hard macros are stored in NMC files, a Xilinx specific file format that is quite similar to *NCD* files, already used to describe classic netlists. A custom script has been developed to automatically generate the hard macros.

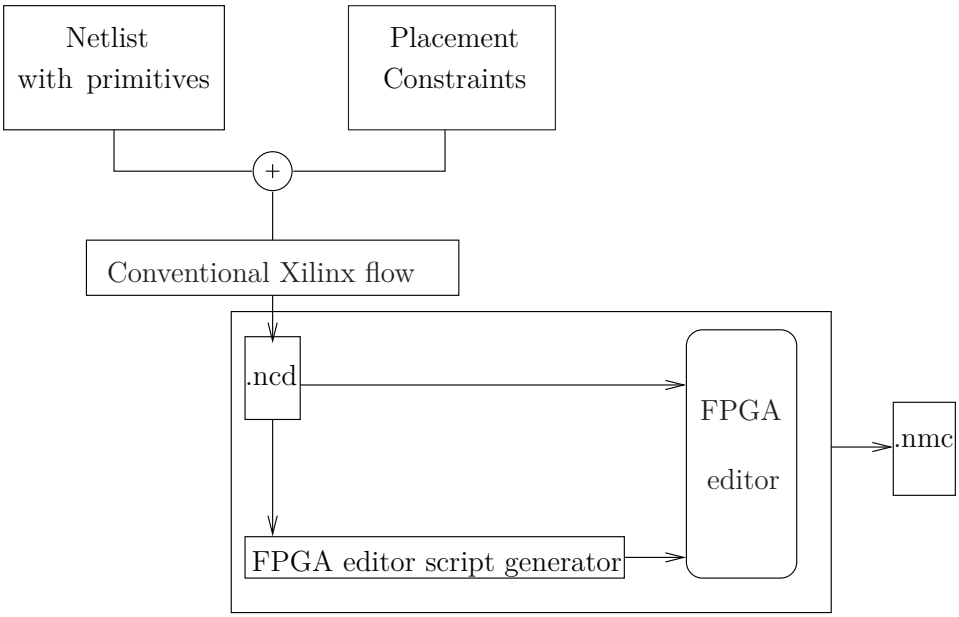


Figure 5.4: Hard Macro generation flow.

Each hard macro is a placed and routed design which will be instantiated later as a black box. The Hard macros do not contain any input or output buffers IOBs. Figure 5.5 illustrates the floor plan of the chains constituting the TRNG implementation on the Virtex 5 FPGA.

In an intermediate design we instantiated all the hard macros (Figure 5.5). In the next step, we use the *ReportGen* Xilinx tool to generate the timing report. In fact, at this point of our design flow, latches and delay chains are placed on the FPGA matrix, but the data-to-clock race is not yet guaranteed and the coarse chain is not yet controllable. With the reported timing delay, we have obtained the following possible differential delays for the fine chains. The results are reported in Table 5.1.

Depending on the delay results, we get (Table 5.1), we need to choose the optimal LUT input to use. $I1$ and $I2$ are dismissed as they introduce a differential delay equal to zero. $I3$ and $I4$ induce the same differential

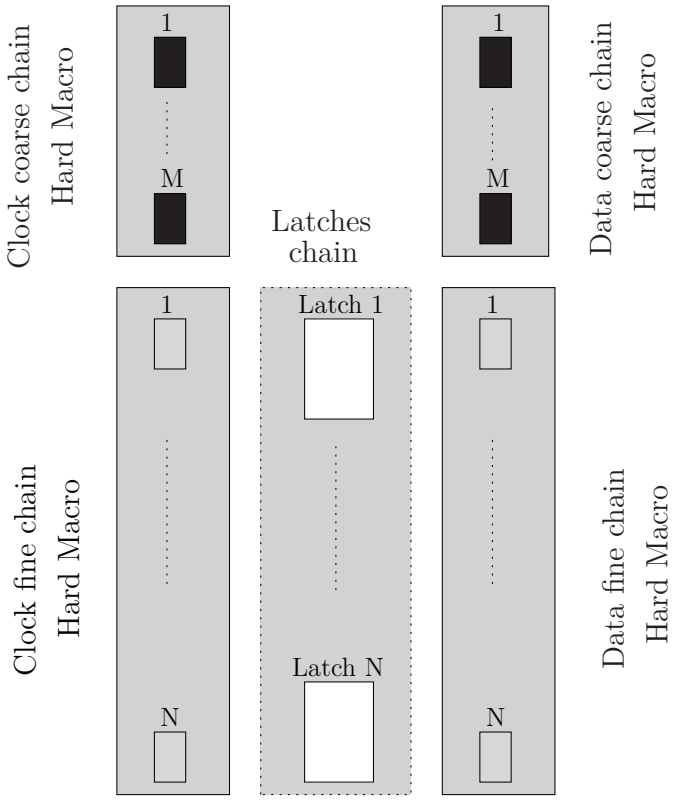


Figure 5.5: Integration of the delay chains hard macros.

delay $\delta t_{DC} = 13\ ps$. Hence, there are only two possibilities to consider: $I4$ or $I5$ LUT input for both fine chains. We need to test both designs, one with the $I4$ selected input and another design with the $I5$ input. To do so, we need to modify the truth tables in the already placed and routed design. LUTs truth tables have to be modified to consider only the chosen routing resources through the $I5$ LUT input. In this step of the design, we have the choice to use simply the FPGA Editor tool or the *RapidSmith* tool. *RapidSmith* [LPL+12, Uni] is an open source library specifically designed in order to manipulate *XDL* netlists. First, we need to export the names of all the instances we want to modify. For this purpose, the intermediate design *NCD* netlist is converted into an *XDL* file. Then, we use the java class, *design.Instance* from *RapidSmith* to list the names of all instances corresponding to the fine chains elements and modify them. As we used the

LUT input	clock chain	data chain	δt_{DC}
$I0$	1040	1040	0
$I1$	1091	1091	0
$I2$	937	937	0
$I3$	544	557	13
$I4$	570	583	13
$I5$	461	471	10

Table 5.1: Table of the realisable fine chains differential delays depending on LUT inputs selection (in ps).

RapidSmith to do this, we use it also to modify the truth tables. Finally we generate the final processed netlist *XDL*. This processed *XDL* netlist is converted back into a final *NCD* binary file.

5.3 Primary Tests

The last step of the flow from Figure 5.1 is to generate the final bitstream from the post processed NCD netlist and load it in the Virtex 5 XC5VLX50T on the Genesys board. Then we perform 100 acquisitions of 1M bits of samples on the FPGA board to see if the frequencies to get '1' at the TRNG output is around 50%. The frequency of occurrences of 1 at the output of the TRNG, v_{trngo} is reported on the Figure 5.6 for each acquisition.

To compare the traces from the TRNG designed with custom latches and DFF primitives, we report the zoomed out of 20K bits sequence generated by each design respectively in Figures 5.7a and 5.7b. We notice from the Figure 5.7 that in the bitstream of DFF primitives based TRNG, a pattern is regularly generated whereas it is not the case for the pattern generated by the custom latch LUT based TRNG.

What is left to do now that we have the final design is to verify the quality of randomness of the sequence output. We propose to perform both NIST and AIS-31 battery of tests to validate the design. This part is presented in the chapter 5.

5.3.1 TRNG throughput

The advantage of this TRNG structure is that it generates a random bit per clock cycle. This allow to reach better throughputs than the one proposed until now. As the throughput of the TRNG is limited by the clock frequency,

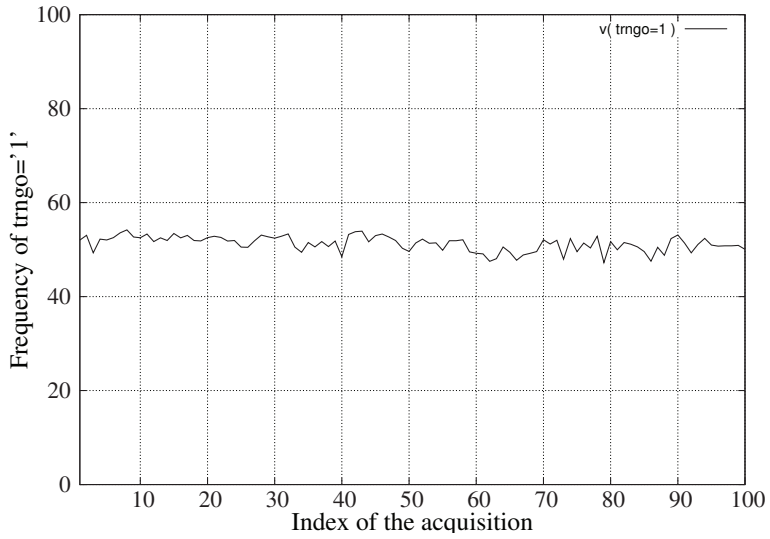


Figure 5.6: TRNG output for 100 different acquisitions on Virtex 5 FPGA

we only need to report f_{clk}. Equation (5.1) gives the frequency limit of the TRNG clock f_{clk} denoted Fc_{max} .

$$Fc_{max} = \frac{1}{((M-1)*(DT_{cc}+dT_{lut})+N*(DT_{fc}+dT_{lut}))} \tag{5.1}$$

where:

- dT_{lut} represents the propagation delay of an LUT
- DT_{cc} = the maximum delay between propagation delay on data coarse delay chain and the propagation delay on clock coarse delay chain)
- DT_{fc} = the maximum delay between propagation delay on data fine delay chain and the propagation delay on clock fine delay chain)
- M the number of MUX used in the coarse delay chain
- N the length of custom latches chain

According to the timing analysis report, the maximum frequency the TRNG can achieve is about 22 MHz. This value can be enhanced using only 16 latches and 4 MUXes on the delay coarse chains making the TRNG reach a throughput of 88 $Mbps$.

5.3.2 TRNG design complexity

In case of our design on Virtex 5, we use :

- 2 coarse delay chains of 12 LUT,
- 2 fine chains of 64 LUTs,
- 64 custom latches design with 64 LUTs, and
- 4 LUT for the XOR tree to generate the final TRNG output.

However, to balance the routing through the delay chains, we place fine and coarse delay chains in columns of SLICES. Moreover, it is important to note that each chain is used as a black box Hard Macro and avoid using resources allocated for the TRNG design when it is embedded with other blocks. Consequently, only one LUT is used per slice. Hence, the TRNG core occupies 224 slices. So, this implementation occupies about 4% of the FPGA Virtex 5 slice resources. It is also interesting to mention that the design is reusable at different locations of the FPGA. Measurement tests on the FPGA board were done for different design locations and it was verified that the TRNG keeps the same performances regardless its location. Table 5.2 reports the comparison of our TRNG FPGA design *vs.* two other recent implementation of metastability based FPGA TRNGs [MKD11, HI12] in terms of complexity and throughput.

Reference	Device	Number of Slices	Throughput
Our design [LBRGD13]	Virtex 5 XC5VLX50T	224	20 Mbps
Hata et al. [HI12]	Virtex 4 XC4VFX20	580	12.5 Mbps
Majzoobi et al. [MKD11]	Virtex 5 XC5VLX110T	32	16 Mbps

Table 5.2: Comparison of existing metastability based TRNGs implementations in FPGA Xilinx devices in terms of complexity and throughput

The next section is dedicated to the TRNG tests, measurements and statistical evaluation of the FPGA implementation of the delay chains based TRNG

5.4 Experimental Results and Statistical Evaluation

5.4.1 Statistical tests evaluation in typical use mode

Standard statistical tests were performed on the implementation described in Section 5.2 on the Virtex 5 Xilinx FPGA. Tables 5.3 and 5.4 give respectively the results of the AIS-31 and NIST statistical tests.

AIS-31 tests	Final result
Disjointness test T0	pass
Monobit test T1	pass
Poker test T2	pass
Run test T3	pass
Long run test T4	pass
Auto-correlation test T5	pass
Uniform distribution test T6a	pass
Uniform distribution test T6b	pass
Test for homogeneity T7a	pass
Test for homogeneity T7b	pass
Entropy estimation test T8	pass

Table 5.3: AIS-31 statistical tests for the FPGA TRNG (test procedure A with von Neumann PP and test procedure B without PP).

5.4.2 Evaluation study in environmental variation

Table 5.5 reports the results of the test procedure A of AIS-31 statistical tests for different coarse chain configurations Tests are applied on raw sequences. In Table 5.5, P refers to successful results and - refers to the failure of 2 tests over 15.

Results of Table 5.5 are satisfactory since 80% of the delay configurations meet the statistical requirements of test procedure A without post-processing. Nevertheless, as AIS-31 standard does allow post-processing on the sequence tested, we perform also the tests for the same sequences with post-processing. The corresponding results are reported in Table 5.6. The table reports that over the 81 possible configuration only 15% fails the PTG.1 class statistical tests. Table 5.7 reports the results of test procedure B without post-processing for all the delay configurations. Results report of the AIS-31 statistical tests are given in Table 5.11 of the Appendix. NIST statistical tests are also applied on the TRNG FPGA generated sequences. The NIST test suite for TRNGs consists of 15 statistical tests such as specified in [RSN$^+$10]. NIST tests are applied on sequences of 20 Mbits for different configurations of coarse chains. To compensate the encvironmental variations, more delay configurations should pass the statistical tests. Hence, we apply a Von-Neumann post-processing on the TRNG output. The corresponding results are given in Table 5.8. P means all the NIST statistical tests were successful while - means that two tests failed and $P_{\overline{u}}$ means that all tests

NIST statistical tests	Final result
Frequency	pass
Block Frequency	pass
Cumulative Sums	pass
Runs	pass
Longest Runs	pass
Rank	pass
FFT	pass
Non Overlapping Template Matching	pass
Overlapping Template Matching	pass
Universal	pass
Approximate Entropy	pass
Random Excursions	pass
Random Excursions Variant	pass
Serial	pass
Linear Complexity	pass

Table 5.4: NIST statistical tests results of the FPGA TRNG with Von Neumann post-processing

passed except the universal test.

Details on the NIST statistical tests results are given in Table 5.10 of the Appendix. The statistical tests results tables, presented above, show that the TRNG pass the tests for 83% of all the delay configurations. This guarantees that the TRNG presents good statistical proporties regardless its working conditions variations. Tests on working conditions variations on the Virtex 5 TRNG design should be conducted to extend the FPGA results presented in this chapter. After the analysis of temperature and power supply voltage variations impact on the TRNG, the design complexity can be lighten and its throughput increased by reducing the number of cells composing each of its delay chains.

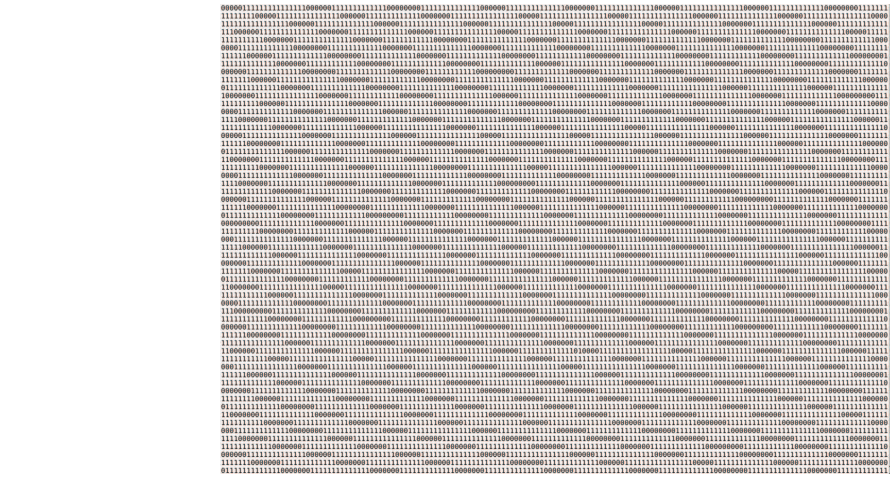
ctrd \ ctr	0x00	0x01	0x03	0x07	0x0F	0x1F	0x3F	0x7F	0xFF
0x00	*P*	*P*	*P*	-	*P*	-	-	*P*	-
0x01	*P*	-	-	-	*P*	*P*	*P*	*P*	*P*
0x03	-	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*
0x07	*P*	*P*	-	*P*	*P*	*P*	*P*	*P*	*P*
0x0F	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*
0x1F	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*	-
0x3F	*P*	-	*P*	*P*	*P*	*P*	*P*	-	*P*
0x7F	*P*	-	-	*P*	*P*	*P*	*P*	*P*	-
0xFF	-	-	-	-	-	-	-	-	-

Table 5.5: Results of AIS-31 tests of PTG.1 class for different coarse chain configurations on raw sequences (-: Fail, P: Pass).

ctrd \ ctr	0x00	0x01	0x03	0x07	0x0F	0x1F	0x3F	0x7F	0xFF
0x00	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*
0x01	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*
0x03	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*
0x07	*P*	*P*	-	*P*	*P*	*P*	*P*	*P*	*P*
0x0F	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*
0x1F	*P*	*P*	*P*	*P*	*P*	-	*P*	*P*	*P*
0x3F	*P*	*P*	*P*	*P*	*P*	*P*	*P*	-	*P*
0x7F	*P*	*P*	-	*P*	*P*	*P*	*P*	*P*	-
0xFF	*P*	-	-	-	-	-	-	*P*	-

Table 5.6: Results of AIS-31 tests of test procedure A for different coarse chain configurations with post-processing (-: Fail, P: Pass).

(a) LUT based custom latch



(b) DFF primitives

Figure 5.7: Random pattern of the output of the delay chains based TRNG designed with:

ctrd \ ctr	0x00	0x01	0x03	0x07	0x0F	0x1F	0x3F	0x7F	0xFF
0x00	-	*P*	-	-	-	-	-	-	-
0x01	*P*	-	-	-	*P*	-	*P*	*P*	-
0x03	-	-	-	*P*	*P*	*P*	*P*	-	*P*
0x07	*P*	*P*	-	*P*	*P*	*P*	*P*	*P*	*P*
0x0F	-	*P*	-	-	*P*	*P*	*P*	*P*	-
0x1F	-	-	-	*P*	*P*	*P*	-	-	-
0x3F	-	-	-	*P*	-	*P*	*P*	-	*P*
0x7F	-	-	-	*P*	-	-	-	-	-
0xFF	-	-	-	-	-	-	-	-	-

Table 5.7: Results of AIS-31 test procedure B for different coarse chain configurations without any post-processing (-: Fail, P: Pass).

ctrd \ ctr	0x00	0x01	0x03	0x07	0x0F	0x1F	0x3F	0x7F	0xFF
0x00	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*	-
0x01	*P*	*P*	*P*	*P*	*P*	-	*P*	*P*	-
0x03	-	*P*	*P*	*P*	*P*	*P*	*P*	*P*	-
0x07	*P*	*P*	$P_{\overline{u}}$	*P*	*P*	*P*	*P*	*P*	-
0x0F	*P*	*P*	*P*	*P*	$P_{\overline{u}}$	*P*	-	*P*	*P*
0x1F	*P*	*P*	*P*	*P*	*P*	*P*	*P*	-	-
0x3F	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*
0x7F	*P*	*P*	-	*P*	*P*	*P*	*P*	*P*	*P*
0xFF	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*	-

Table 5.8: NIST results for different coarse chain configurations with post-processing.

5.5 Conclusion

In this chapter, we study the feasibility of the delay chains based TRNG in FPGA technology. A design methodology is proposed to bypass the automatic PAR in FPGA. We managed to obtain a fine differential delay to force the appearance of metastability. The delay chains based TRNG proposed in this Ph.D work has proven to be feasible on both FPGA and ASIC. Even though the FPGA design requires special care on the placement and route process, the TRNG FPGA version presented better results. In fact, the test-chip characterisation results have shown that the level of noise on FPGA is five times higher than what is perceived on ASIC test-chip.

Conclusion and perspectives

General Conclusion

This manuscript presents research works concerning modeling, characterisation and design of a delay chains based TRNG which exploits both voltage noise and phase noise. A study of the feasibility of delay chains with high precision in FPGA and ASIC technologies has been conducted. The study has been validated by designing prototypes in FPGA and ASIC technologies. A stochastic model has been proposed for the delay chains based TRNG. It allows the designer to express the random behavior of the TRNG, and establish the theoretical probability of getting one at the TRNG output, regardless the targeted technology. The FPGA and ASIC implementation and validation represent major contributions of this work. Three target prototypes, two ASIC in CMOS 65nm technology and one Xilinx FPGA have been studied. The design method has been adapted at every stage: architecture with cell choice, place and route, simulation and validation. Thus it offered a good base to characterize and assess the quality of the randomness issued from the TRNG. Simulation results have been intensively used to calibrate the architecture and assess the quality of randomness. The two first prototypes have been tested with NIST and AIS-31 statistical tests which are applied for many configurations of the tuning coarse chains. It has been noticed that the level of noise on ASIC is much lower than on FPGA and needs accurate delay control and precision. Two types of delay controls for the coarse chain have been proposed. The first one with a chain of multiplexers and the second one with open-drain buffers. Associated with a dynamic control of the best configuration, the TRNG architecture could be automatically adjusted to deliver the best randomness regardless environmental and technological variations.

Perspectives

As future works, statistical tests will be conducted on the second ASIC prototype which was not delivered when writing this manuscript. An other feature should be added within the TRNG to provide online statistical testing and rise an alarm in case of entropy abnormal situation or malevolent perturbations. This feature is also required to be compliant with certification standards. The online test block should take into account the low complexity constraint, specifically important for lightweight applications. The fault injection analysis will be carried out to evaluate the impact of over-clocking, laser fault injection and electromagnetic injection attacks on the quality of

randomness. This will check the efficiency of the online testing and give some insights to enhance the architecture against attacks.

Publications

[1] Molka Ben-Romdhane, Tarik Graba, and Jean-Luc Danger. Stochastic model of a metastability-based true random number generator. In Michael Huth, N. Asokan, Srdjan apkun, Ivan Flechais, and Lizzie Coles-Kemp, editors, Trust and Trustworthy Computing, volume 7904 of Lecture Notes in Computer Science, pages 92–105. Springer Berlin Heidelberg, 2013.

[2] M. Ben-Romdhane, T. Graba, J.-L. Danger, and Y. Mathieu. Design methodology of an ASIC TRNG based on an open-loop delay chain. In New Circuits and Systems Conference (NEW-CAS), 2013 IEEE 11th International, pages 1–4, 2013.

[3] Florent Lozach, Molka Ben-Romdhane, Tarik Graba, and Jean-Luc Danger. FPGA design of an open-loop true random number generator. In Digital System Design (DSD), 2013 Euromicro Conference on, pages 615–622, 2013.

[4] Molka Ben-Romdhane, Tarik Graba, and Jean-Luc Danger. ASIC enhancement of a delay chains based true random number generator. Under preparation.

Appendix

NIST standard statistical tests results

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	Result	STATISTICAL TEST
8	10	6	7	12	10	13	13	13	8	0.699313	100/100	PASS	Frequency
12	13	15	9	11	10	11	5	8	6	0.474986	99/100	PASS	BlockFrequency
11	6	8	6	8	5	14	18	13	11	0.075719	100/100	PASS	CumulativeSums
8	10	9	11	10	7	12	9	13	11	0.964295	99/100	PASS	CumulativeSums
9	11	13	10	10	8	14	8	13	4	0.534146	100/100	PASS	Runs
9	6	9	9	10	15	7	17	12	6	0.202268	99/100	PASS	LongestRun
8	14	11	9	8	9	10	10	14	7	0.816537	100/100	PASS	Rank
15	15	7	13	9	10	9	8	8	6	0.401199	96/100	PASS	FFT
13	14	10	11	10	6	8	7	15	6	0.383827	98/100	PASS	NonOverlappingTemplate
100	0	0	0	0	0	0	0	0	0	0.000000	0/100	FAIL	Universal
12	13	8	11	9	8	10	11	6	12	0.883171	100/100	PASS	ApproximateEntropy
0	4	5	1	2	1	4	3	1	1	0.162606	22/22	PASS	RandomExcursions
3	4	2	1	2	5	1	0	2	2	0.350485	22/22	PASS	RandomExcursionsVariant
14	8	10	8	7	7	13	10	12	11	0.779188	100/100	PASS	Serial
11	9	11	14	9	8	12	9	7	10	0.924076	100/100	PASS	Serial
9	8	11	11	10	12	11	9	7	12	0.978072	98/100	PASS	LinearComplexity

Table 5.9: NIST results for the uniformity of p-values and the proportion of passing sequences of the PP output of ASIC TRNG3

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	Result	STATISTICAL TEST
10	7	6	9	7	15	16	11	9	10	0.366918	99/100	PASS	Frequency
15	13	6	12	10	9	6	9	15	5	0.202268	97/100	PASS	BlockFrequency
10	8	4	8	8	12	11	11	11	17	0.319084	99/100	PASS	CumulativeSums
10	9	5	11	8	10	8	5	17	17	0.071177	99/100	PASS	CumulativeSums
7	9	6	12	10	10	10	13	11	12	0.883171	98/100	PASS	Runs
10	6	7	9	10	9	13	12	12	12	0.851383	97/100	PASS	LongestRun
10	16	6	11	10	7	5	14	16	5	0.058984	99/100	PASS	Rank
13	9	9	8	9	8	18	8	8	10	0.419021	99/100	PASS	FFT
13	9	10	10	6	12	7	11	10	12	0.883171	99/100	PASS	NonOverlappingTemplate
10	13	14	9	15	6	6	10	9	8	0.455937	99/100	PASS	OverlappingTemplate
10	3	15	11	13	5	7	12	14	10	0.129620	99/100	PASS	Universal
13	12	9	9	12	6	8	8	9	14	0.739918	100/100	PASS	ApproximateEntropy
6	8	4	8	10	8	12	7	12	6	0.572333	81/81	PASS	RandomExcursions
9	9	7	8	6	7	14	9	8	4	0.572333	78/81	PASS	RandomExcursionsVariant
12	11	11	12	10	8	9	6	11	10	0.955835	99/100	PASS	Serial
10	10	10	15	7	14	6	6	16	6	0.145326	100/100	PASS	LinearComplexity

Table 5.10: NIST results for the uniformity of p-values and the proportion of passing sequences of raw output of the FPGA TRNG

AIS-31 standard statistical tests results

Test Name	Lower and the upper bound	Test variable value	Result
T0	-	-	Pass
T1	[9655 ; 10345]	10058	Pass
T2	[1.03 ; 57.4]	13.75	Pass
T3	[2267; 2733] [2267; 2733] [1079; 1421] [1079; 1421] [502; 748] [502; 748] [223; 402] [223; 402] [90; 223] [90; 223] [90; 223] [90; 223]	0-Runs[1] = 2470 1-Runs[1] = 2405 0-Runs[2] = 1225 1-Runs[2] = 1231 0-Runs[3] = 598 1-Runs[3] = 651 0-Runs[4] = 309 1-Runs[4] = 317 0-Runs[5] = 154 1-Runs[5] = 181 0-Runs[6] = 179 1-Runs[6] = 150	Pass
T4	34 long run of 0's and 1's	None	Pass
T5	[2326 ; 2674]	2459	Pass
T6a		|P(1) - 0.5| = 0.00365	Pass
T6b		p(01) = 0.49621 p(11) = 0.49885 $\|p_(01) - p_(11)\| = 0.00264$	Pass
T7a		T[0] = 0.259 T[1] = 3.378	Pass
T7b		T[0] = 0.0387 T[1] = 6.4297 T[2] = 2.5063 T[3] = 0.0000	Pass
T8	8	T= 8.00	Pass

Table 5.11: AIS-31 results for the FPGA TRNG without post-processing

Test Name	Lower and the upper bound	Test variable value	Result
T0	-	-	Pass
T1	[9655 ; 10345]	10246	Pass
T2	[1.03 ; 57.4]	9.03	Pass
T3	[2267; 2733] [2267; 2733] [1079; 1421] [1079; 1421] [502; 748] [502; 748] [223; 402] [223; 402] [90; 223] [90; 223] [90; 223] [90; 223]	0-Runs[1] =2501 1-Runs[1] =2466 0-Runs[2] =1232 1-Runs[2] =1260 0-Runs[3] =602 1-Runs[3] =645 0-Runs[4] =319 1-Runs[4] =312 0-Runs[5] =179 1-Runs[5] =149 0-Runs[6] =154 1-Runs[6] =156	Pass
T4	34 long run of 0's and 1's	None	Pass
T5	[2326 ; 2674]		Pass
T6a		|P(1) - 0.5| = 0.0192	Pass
T6b		p(01) = 0.47832 p(11) = 0.48234 $\|p_(01) - p_(11)\| = 0.004$	Pass
T7a		T[0] = 0.2164 T[1] = 1.4916	Pass
T7b		T[0] = 0.4744 T[1] = 0.0845 T[2] = 0.6699 T[3] = 0.4263	Pass
T8	8	T= 7.9962	Pass

Table 5.12: AIS-31 results for the ASIC_TRNG#1 (TRNG3 version) with post-processing for T0-T5 and without for T6-T7-T8

χ^2 contingency table

df \ α	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
1	2.71	6.63	10.8	15.13	19.51	23.92
2	4.61	9.21	13.8	18.42	23.02	27.63
3	6.25	11.3	16.3	21.1	25.90	30.66
4	7.78	13.3	18.5	23.51	28.47	33.37
5	9.24	15.1	20.5	25.74	30.85	35.88
6	10.6	16.8	22.5	27.85	33.10	38.25
7	12	18.5	24.3	29.87	35.25	40.52
8	13.4	20.1	26.1	31.82	37.33	42.69
9	14.7	21.7	27.9	33.71	39.34	44.81
10	16	23.2	29.6	35.56	41.29	46.85
11	17.3	24.7	31.3	37.36	43.20	48.86
12	18.5	26.2	32.9	39.13	45.07	50.82
13	19.8	27.7	34.5	40.87	46.91	52.76
14	21.1	29.1	36.1	42.57	48.71	54.63
15	22.3	30.6	37.7	44.26	50.49	56.49

Table 5.13: χ^2 contingency table where α is the error probability and df is the degree of freedom.

References

[Aas12] Paul. Ampadu, author., and SpringerLink (Online service). *Managing Temperature Effects in Nanoscale Adaptive Systems*. Springer New York, 2012.

[Abi06] A.A. Abidi. Phase noise and jitter in cmos ring oscillators. *Solid-State Circuits, IEEE Journal of*, 41(8):1803–1816, Aug 2006.

[AFR08] Massimo Alioto, Luca Fondelli, and Santina Rocchi. Analysis and performance evaluation of area-efficient true random bit generators on fpgas. In *ISCAS*, pages 1572–1575, 2008.

[Alf05] Peter Alfke. Metastable recovery in virtex-ii pro fpgas, February 2005. `http://www.xilinx.com/support/documentation/application_notes/xapp094.pdf`.

[All08] Smart Card Alliance. What makes a smart card secure? a smart card alliance contactless and mobile payments council white paper. Technical report, Smart Card Alliance, October 2008.

[Bak10] R. Jacob Baker. *CMOS Circuit Design, Layout, and Simulation*. Wiley-IEEE Press, 3rd edition, 2010.

[BBA+12] Pierre Bayon, Lilian Bossuet, Alain Aubert, Viktor Fischer, François Poucheret, Bruno Robisson, and Philippe Maurine. Contactless electromagnetic active attack on ring oscillator based true random number generator. In *Proceedings of the Third international conference on Constructive Side-Channel Analysis and Secure Design*, COSADE'12, pages 151–166, Berlin, Heidelberg, 2012. Springer-Verlag.

[BBAF13] P. Bayon, L. Bossuet, A. Aubert, and V. Fischer. Electromagnetic analysis on ring oscillator-based true random number generators. In *Circuits and Systems (ISCAS), 2013 IEEE International Symposium on*, pages 1954–1957, 2013.

[BBKN12] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache. Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures. *Proceedings of the IEEE*, 100(11):3056–3076, 2012.

[BGKD06] M. Bhushan, A. Gattiker, M.B. Ketchen, and K.K. Das. Ring oscillators for cmos process tuning and variability control. *Semiconductor Manufacturing, IEEE Transactions on*, 19(1):10–18, Feb 2006.

[cDFF06] M. Šimka, M. Drutarovský, V. Fischer, and J. Fayolle. Model of a true random number generator aimed at cryptographic applications. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, page 4 pp., may 2006.

[CFAF13] Abdelkarim Cherkaoui, Viktor Fischer, Alain Aubert, and Laurent Fesquet. A self-timed ring based true random number generator. *2012 IEEE 18th International Symposium on Asynchronous Circuits and Systems*, 0:99–106, 2013.

[CGPR08] Jean-Sébastien Coron, Christophe Giraud, Emmanuel Prouff, and Matthieu Rivain. Attack and improvement of a secure s-box calculation based on the fourier transform. In Elisabeth Oswald and Pankaj Rohatgi, editors, *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2008.

[Cha] Ken Chapman. Seu strategies for virtex-5 devices. application note: Virtex-5 family. `http://www.xilinx.com/support/documentation/application_notes/xapp864.pdf`.

[CK10] Jean-Sébastien Coron and Ilya Kizhvatov. Analysis and improvement of the random delay countermeasure of ches 2009. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 95–109. Springer Berlin Heidelberg, 2010.

[Cor99] Jean-Sebastien Coron. On the security of random sources. In *Public Key Cryptography*, volume 1560 of *Lecture Notes in Computer Science*, pages 29–42. Springer Berlin Heidelberg, 1999.

[CSC+10] Doris Chen, Deshanand Singh, Jeffrey Chromczak, David Lewis, Ryan Fung, David Neto, and Vaughn Betz. A comprehensive approach to modeling, characterizing and optimizing for metastability in fpgas. In *Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays*, FPGA '10, pages 167–176, New York, NY, USA, 2010. ACM.

[DG07] Markus Dichtl and Jovan Dj. Golic. High-speed true random number generation with logic gates only. In *CHES*, pages 45–62, 2007.

[DGH07] Jean-Luc Danger, Sylvain Guilley, and Philippe Hoogvorst. Fast True Random Generator in FPGAs. In *IEEE MWSCAS/NEWCAS*, pages 506–509, Montréal, Canada, aug 2007. DOI: 10.1109/NEWCAS.2007.4487970.

[DGH09] Jean-Luc Danger, Sylvain Guilley, and Philippe Hoogvorst. High Speed True Random Number Generator based on Open Loop Structures in FPGAs. *Microelectronics Journal*, 40(11):1650–1656, November 2009. DOI: 10.1016/j.mejo.2009.02.004.

[Dic03] Markus Dichtl. How to predict the output of a hardware random number generator. In ColinD. Walter, ÇetinK. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 181–188. Springer Berlin Heidelberg, 2003.

[Dic07] Markus Dichtl. Bad and good ways of post-processing biased physical random numbers. In Alex Biryukov, editor, *Fast Software Encryption*, volume 4593 of *Lecture Notes in Computer Science*, pages 137–152. Springer Berlin Heidelberg, 2007.

[EB07] Dirk Eddelbuettel and Robert G. Brown. Rdieharder: An r interface to the dieharder suite of random number generator tests, 2007.

[ECS05] D. Eastlake, S. Crocker, and J. Schiller. Randomness recommandations for security, 2005. RFC 4086.

[FAB+09] Viktor Fischer, Alain Aubert, Florent Bernard, Boyan Valtchanov, Jean-Luc Danger, and Nathalie Bochard. True Random Number Generators in Configurable Logic Devices, 2009. `http://www.lirmm.fr/~w3mic/ANR/PDF/D2.pdf`.

[Fai99] Application Note Semiconductor Fairchild. Design innovations address advanced cmos logic noise considerations: Understanding the fundamentals of noise, 1999.

[FDcB04] Viktor Fischer, Miloš Drutarovský, Martin Šimka, and Nathalie Bochard. High performance true random number generator in altera stratix fplds. In Jurgen Becker, Marco Platzner, and Serge Vernalde, editors, *Field Programmable Logic and Application*, volume 3203 of *Lecture Notes in Computer Science*, pages 555–564. Springer Berlin Heidelberg, 2004.

[FDcC04] Viktor Fischer, Miloš Drutarovský, Martin Šimka, and Frédéric Celle. A simple pll-based true random number generator for embedded digital systems. *Computing and Informatics*, pages 5–6, 2004.

[Fol96] Clark Foley. Characterizing metastability practical measurement techniques to accurately determine "device dependent coefficients" used to predict synchronizer mtbf. *Asynchronous Circuits and Systems, International Symposium on*, 0:175, 1996.

[GAR10] Christy M. Gearheart, Benjamin Arazi, and Eric C. Rouchka. Dna-based random number generation in security circuitry. *Biosystems*, 100(3):208–214, 2010.

[GCS09] Tamas Gyorfi, Octavian Cret, and Alin Suciu. High performance true random number generator based on fpga block rams. *Parallel and Distributed Processing Symposium, International*, 0:1–8, 2009.

[Gin11] R. Ginosar. Metastability and synchronizers: A tutorial. *Design Test of Computers, IEEE*, 28(5):23 –35, sept.-oct. 2011.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In *Proceedings of the Third International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '01, pages 251–261, London, UK, UK, 2001. Springer-Verlag.

[Gol06] Jovan Dj. Golic. New methods for digital generation and postprocessing of random data. *IEEE Trans. Computers*, 55(10):1217–1229, 2006.

[Ham12] Michael Hamburg. Understanding intel's ivy bridge random number generator, December, 11 2012.

[HBF09] D.E. Holcomb, W.P. Burleson, and K. Fu. Power-up sram state as an identifying fingerprint and source of true random numbers. *Computers, IEEE Transactions on*, 58(9):1198–1210, Sept 2009.

[HCG+13] H.C. Herbert, G.W. Cox, S. Gueron, J. Walker, C.E. Dike, S.A. Fischer, E. Brickell, M.G. Dixon, D. Johnston, G. Thuraisingham, et al. Digital random number generator using partially entropic data, July 16 2013. US Patent 8,489,660.

[HG13] Hasselmann and Gabriel. Developer evidence for the evaluation of a physical true random number generator, February 28 2013. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_31_PTRNG–Developer_Evidence.pdf.

[HI12] Hisashi Hata and Shuichi Ichikawa. Fpga implementation of metastability-based true random number generator. *IEICE Transactions*, 95-D(2):426–436, 2012.

[HKM12] Mike Hamburg, Paul Kocher, and Mark E. Marson. Analysis of intel's ivy bridge digital random number generator, March, 12 2012.

[HTBF14] P. Haddad, Y. Teglia, F. Bernard, and V. Fischer. On the assumption of mutual independence of jitter realizations in p-trng stochastic models. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, pages 1–6, March 2014.

[IDQ10] IDQuantique. Quantis white paper random number generation using quantum physics, April 2010.

[JAW+00] Thomas Jennewein, Ulrich Achleitner, Gregor Weihs, Harald Weinfurter, and Anton Zeilinger. A fast and compact quantum random number generator. *Review of Scientific Instruments*, 71(4):1675–1680, 2000.

[JK99] Benjamin Jun and Paul Kocher. The Intel Random Number Generator, 1999. `http://www.cryptography.com/intelRNG.pdf`.

[Joi13] Joint Interpretation Library. Application of Attack Potential to Smartcards, Version 2.9, January 2013. `http://www.ssi.gouv.fr/site_documents/JIL/JIL-The_application_of_attack_potential_to_smartcards_V2-1.pdf`.

[KC02] D.J. Kinniment and E.G. Chester. Design of an on-chip random number generator using metastability. In *Proceedings of the 28th European Solid-State Circuit Conference.*, 2002.

[KD90] L.-S. Kim and R.W. Dutton. Metastability of cmos latch/flip-flop. *Solid-State Circuits, IEEE Journal of*, 25(4):942 –951, aug 1990.

[KEC+11] Siew-Hwee Kwok, Yen-Ling Ee, Guanhan Chew, Kanghong Zheng, Khoongming Khoo, and Chik-How Tan. A comparison of post-processing techniques for biased random number generators. In *Proceedings of the 5th IFIP WG 11.2 International Conference on Information Security Theory and Practice: Security and Privacy of Mobile Devices in Wireless Communication*, WISTP'11, pages 175–190, Berlin, Heidelberg, 2011. Springer-Verlag.

[KG04] Paul Kohlbrenner and Kris Gaj. An embedded true random number generator for FPGAs. In *Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays*, 2004.

[KJJ96] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Proceedings of CRYPTO'96*, volume 1109 of *LNCS*, pages 104–113. Springer-Verlag, 1996.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in Cryptology - CRYPTO'99*, pages 388–397. Springer-Verlag, 1999.

[KK06] Ranjith Kumar and Volkan Kursun. Impact of temperature fluctuations on circuit characteristics in 180nm and 65nm cmos technologies. In *ISCAS*. IEEE, 2006.

[KS08] Wolfgang Killmann and Werner Schindler. A design for a physical rng with robust entropy estimators. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems CHES 2008*, volume 5154 of *Lecture Notes in Computer Science*, pages 146–163. Springer Berlin Heidelberg, 2008.

[KS11] Wolfgang Killmann and Werner Schindler. A proposal for: Functionality classes for random number generators1, September 2011.

[KW10] S. Petrovic K. Wold. Robustness of trng against attacks that employ superimposing signal on fpga supply voltage. In *Proceedings of the Norwegian Information Security Conference*, pages 81–92. Tapir Akademisk Forlag, 2010.

[LBRGD13] Florent Lozac'h, Molka Ben-Romdhane, Tarik Graba, and Jean-Luc Danger. Fpga design of an open-loop true random number generator. In *Digital System Design (DSD), 2013 Euromicro Conference on*, pages 615–622, 2013.

[LCMR11] Xiaowen Li, Adam B. Cohen, Thomas E. Murphy, and Rajarshi Roy. Scalable parallel physical random number generator based on a superluminescent led. *Opt. Lett.*, 36(6):1020–1022, Mar 2011.

[LHA+12] Arjen K. Lenstra, James P. Hughes, Maxime Augier, Joppe W. Bos, Thorsten Kleinjung, and Christophe Wachter. Ron was wrong, whit is right. Cryptology ePrint Archive, Report 2012/064, 2012. `http://eprint.iacr.org/`.

[Liu06] Chengxin Liu. *Jitter in oscillators with 1/f noise sources and Application to True RNG for Cryptography*. PhD thesis, Worcester Polytechnic Institute, January 2006.

[LM05] Chengxin Liu and J. McNeill. A digital-pll-based true random number generator. In *Research in Microelectronics and Electronics, 2005 PhD*, volume 1, pages 113–116 vol.1, July 2005.

[LPL+12] Christopher Lavin, Marc Padilla, Jaren Lamprecht, Philip Lundrigan, Brent Nelson, Brad Hutchings, and Michael Wirthlin. A library for low-level manipulation of partially placed-and-routed fpga designs. technical report and documentation. Technical report, NSF Center for High Performance Reconfigurable Computing (CHREC), Department of Electrical and Computer Engineering, Brigham Young University,Provo, UT 84602, 2010-2012.

[LS07] Pierre L'Ecuyer and Richard Simard. Testu01: A c library for empirical testing of random number generators. *ACM Trans. Math. Softw.*, 33(4), August 2007.

[MKD11] Mehrdad Majzoobi, Farinaz Koushanfar, and Srinivas Devadas. Fpga-based true random number generation using circuit metastability with adaptive feedback control. In *Proceedings of the 13th international conference on Cryptographic hardware and embedded systems*, CHES'11, pages 17–32, Berlin, Heidelberg, 2011. Springer-Verlag.

[MM09] A. Theodore Markettos and Simon W. Moore. The frequency injection attack on ring-oscillator-based true random number generators. In *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, volume 5747 of *Lecture Notes in Computer Science*, pages 317–331. Springer, 2009.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[MvOV96] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, October 1996.

[NIS12] NIST. Recommendation for the entropy sources used for random bit generation, 2012. `http://csrc.nist.gov/publications/drafts/800-90/draft-sp800-90b.pdf`.

[NS02] Nguyen and Shparlinski. The insecurity of the digital signature algorithm with partially known nonces. *Journal of Cryptology*, 2002.

[NS03] Phong Q. Nguyen and Igor E. Shparlinski. The insecurity of the elliptic curve digital signature algorithm with partially known nonces. *Des. Codes Cryptography*, 2003.

[OMHT06] Elisabeth Oswald, Stefan Mangard, Christoph Herbst, and Stefan Tillich. Practical Second-Order DPA Attacks for Masked Smart Card Implementations of Block Ciphers. In David Pointcheval, editor, *CT-RSA*, volume 3860 of *LNCS*, pages 192–207. Springer, 2006.

[PTL+11] François Poucheret, Karim Tobich, Mathieu Lisart, Laurent Chusseau, Bruno Robisson, and Philippe Maurine. Local and Direct EM Injection of Power Into CMOS Integrated Circuits. In Luca Breveglieri, Sylvain Guilley, Israel Koren, David Naccache, and Junko Takahashi, editors, *FDTC*, pages 100–104. IEEE, 2011.

[RSN+10] Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Elaine Barker, Stefan Leigh, Mark Levenson, Mark Vangel, David Banks, Alan Heckert, James Dray, and San Vo. A statistical test suite for the validation of random number generators and pseudo random number generators for cryptographic applications„ april 2010.

[SBGD11] Nidhal Selmane, Shivam Bhasin, Sylvain Guilley, and Jean-Luc Danger. Security evaluation of application-specific integrated circuits and field programmable gate arrays against setup time violation attacks. *IET Information Security*, 5(4):181–190, December 2011. DOI: 10.1049/iet-ifs.2010.0238.

[SBpC+09] Valerio Scarani, Helle Bechmann-pasquinucci, Nicolas J. Cerf, Miloslav Dušek, Norbert Lütkenhaus, and Momtchil Peev. The security of practical quantum key distribution, 2009.

[SCDC+11] M. Soucarros, C. Canovas-Dumas, J. Clediere, P. Elbaz-Vincent, and D. Real. Influence of the temperature on true random number generators. In *Hardware-Oriented Security and Trust (HOST), 2011 IEEE International Symposium on*, pages 24–27, June 2011.

[SCDEV13] Mathilde Soucarros, Jessy Clédière, Cécile Dumas, and Philippe Elbaz-Vincent. Fault analysis and evaluation of a true random number generator embedded in a processor. *Journal of Electronic Testing*, 29(3):367–381, 2013.

[Sch99] Werner Schindler. Ais 20: Functionality classes and evaluation methodology for deterministic random number generators,

December 1999. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/ais20e_pdf.pdf.

[Sch01] Werner Schindler. Efficient online tests for true random number generators. In ÇetinK. Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 103–117. Springer Berlin Heidelberg, 2001.

[SMS07] Berk Sunar, William J. Martin, and Douglas R. Stinson. A provably secure true random number generator with built-in tolerance to active attacks. *IEEE Trans. Computers*, 56(1):109–119, 2007.

[Sou12] Mathilde Soucarros. *Analysis of random number generators in abnormal usage conditions.* PhD thesis, Institut Fourier, Université de Grenoble, 2012.

[SPQ05] F. X Standaert, E. Peeters, and J-J Quisquater. On the masking countermeasure and higher-order power analysis attacks. In *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*, volume 1, pages 562–567 Vol. 1, 2005.

[SPV06] Dries Schellekens, Bart Preneel, and Ingrid Verbauwhede. Fpga vendor agnostic true random number generator. In *FPL*, pages 1–6, 2006.

[SSR09] R. Santoro, O. Sentieys, and S. Roy. On-line monitoring of random number generators for embedded security. In *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*, pages 3050 –3053, may 2009.

[Sta94] NIST FIPS (Federal Information Processing Standards). Security Requirements for Cryptographic Modules publication 140-1, January 11 1994. http://csrc.nist.gov/publications/fips/fips140-1/fips1401.pdf.

[Sti11] M. Stipcevic. Quantum random number generators and their use in cryptography. In *MIPRO, 2011 Proceedings of the 34th International Convention*, pages 1474–1479, 2011.

[TBM07] Carlos Tokunaga, David T. Blaauw, and Trevor N. Mudge. True Random Number Generator with a Metastability-Based Quality Control. In *IEEE International Solid-State Circuits Conference (ISSCC)*, pages 404–611. IEEE, February 2007.

[Tka02] Thomas Tkacik. A hardware random number generator. In BurtonS. Kaliski, ÇetinK. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 450–453. Springer Berlin Heidelberg, 2002.

[Tro59] H.F. Trotter. An elementary proof of the central limit theorem. *Archiv der Mathematik*, 10:226–234, 1959.

[Uni] Brigham Young University. Rapidsmith 0.5.1. `http://sourceforge.net/projects/rapidsmith/files/`.

[VD10] Michal Varchola and Miloš Drutarovský. New high entropy element for fpga based true random number generators. In *Proceedings of the 12th International Conference on Cryptographic Hardware and Embedded Systems*, CHES'10, pages 351–365, Berlin, Heidelberg, 2010. Springer-Verlag.

[Vee80] H.J.M. Veendrick. The behaviour of flip-flops used as synchronizers and prediction of their failure rate. *Solid-State Circuits, IEEE Journal of*, 15(2):169 – 176, apr 1980.

[VF02] M. Drutarovsky V. Fischer. True Random Number Generator Embedded in Reconfigurable Hardware. In *Proc. the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, 2002.

[VHK12] I. Vasyltsov, E. Hambardzumyan, and B. Karpinskyy. Random number generator, December 25 2012. US Patent 8,341,201.

[VHKK08] Ihor Vasyltsov, Eduard Hambardzumyan, Young-Sik Kim, and Bohdan Karpinskyy. Fast digital trng based on metastable ring oscillator. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems CHES 2008*, volume 5154 of *Lecture Notes in Computer Science*, pages 164–180. Springer Berlin Heidelberg, 2008.

[vN51] John von Neumann. Various Techniques Used in Connection with Random Digits. *J. Res. Nat. Bur. Stand.*, 12:36–38, 1951.

[VSB10] V.B V.B. Suresh and W.P. Burleson. Entropy extraction in metastability-based TRNG. In *HOST*, pages 135–140, 2010.

[VT08] Inc. VIA Technologies. PADLOCK QUICK REFERENCE FOR VIA NANO PROCESSORS VERSION 0.95, 25th July 2008. ftp://ftp.vt-bridge.org/Docs/CPU/Nano/padlock_quick_reference_V095.pdf.

[Wal01] John Walker. Hotbits: Genuine random numbers generated by radioactive decay, 2001.

[WP11] K. Wold and S. Petrovic. Behavioral model of trng based on oscillator rings implemented in fpga. In *Design and Diagnostics of Electronic Circuits Systems (DDECS), 2011 IEEE 14th International Symposium on*, pages 163–166, 2011.

[WT08] Knut Wold and Chik How Tan. Analysis and enhancement of random number generator in fpga based on oscillator rings. In *ReConFig*, pages 385–390, 2008.

[XHA08] Peng Xu, T. Horiuchi, and P. Abshire. Stochastic model and simulation of a random number generator circuit. In *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, pages 2977 –2980, may 2008.

[Xil97] Xilinx. Metastability considerations, 1997. `http://www.coe.montana.edu/ee/courses/ee/ee367/pdffiles/xapp077.pdf`.

[Xil09] Xilinx. Constraints guide, December 2009. `www.xilinx.com/support/documentation/sw_manuals/xilinx11/cgd.pdf`.

[YSKB07] Sang-Kyung Yoo, Berk Sunar, Deniz Karakoyunlu, and Berk Bir. A robust and practical random number generator, 2007.

[ZWW11] J. Z. Zhang, J. F. Wang, and Y. C. Wang. High-speed physical random number generation using a chaotic semiconductor laser. In *Proc. SPIE, International Conference on Optical Instruments and Technology: Optoelectronic Devices and Integration*, volume 8198, pages 81980I–81980I–6, 2011.

Modélisation, implémentation et caractérisation de circuits générateurs de nombres aléatoires vrais utilisant des chaînes à retards

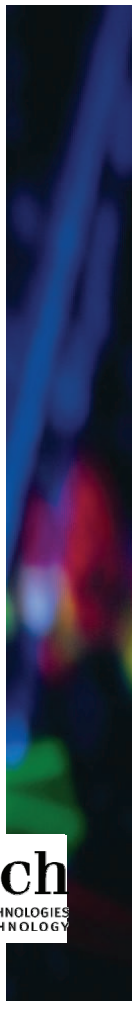
Molka BEN ROMDHANE

RESUME : Les nombres aléatoires sont indispensables dans de nombreuses applications notamment en cryptographie où l'aléa est utilisé dans les protocoles de sécurité. Les générateurs de nombres aléatoires, plus connus sous le nom de RNG comme "Random Number Generator" se déclinent en deux familles, les PRNG (Pseudo RNG) qui sont des générateurs de nombres aléatoires ayant des séquences déterministes et les TRNG (True RNG) qui sont des générateurs d'aléa "vrai", donc non prédictibles. Les applications cryptographiques utilisent à la fois les TRNG et les PRNG. Un PRNG nécessite une valeur initiale, ou graine, qui peut être la sortie d'un TRNG. Les TRNG tirent profit de l'aléa des phénomènes physiques. Les TRNGs dans les technologies numériques comme les FPGAs font appel à des oscillateurs qui présentent l'inconvénient de pouvoir être attaqués par couplage harmonique. De façon à évaluer la qualité entropique d'un TRNG, des standards basés sur des tests statistiques ont été élaborés par des organismes de certification comme le NIST ou la BSI. Cependant, il est recommandé de formaliser, par le biais d'un modèle, le caractère stochastique de la génération d'aléa. Dans cette thèse, nous étudions une architecture de TRNG, peu coûteuse et robuste face aux attaques harmoniques car elle n'utilise pas d'oscillateurs. Ce TRNG extrait une variable aléatoire en exploitant à la fois les états métastables des bascules et les fluctuations temporelles (ou gigue) des signaux échantillonnés. Nous proposons par la suite un modèle stochastique qui nous permet de décrire le comportement aléatoire du TRNG indépendamment de la technologie ciblée. Les caractérisations et évaluations sur des circuits prototypes en technologies FPGA et ASIC montrent que l'architecture TRNG proposée génère de l'aléa de qualité et est robuste face aux variations environnementales.

MOTS-CLEFS : TRNG, métastabilité, modélisation, ASIC, FPGA, retards programmables, tests statistiques, NIST, AIS-31.

ABSTRACT : Random numbers are required in numerous applications namely in cryptography where randomness is used in security protocols. There are two main classes of Random Number Generators (RNG) : The Pseudo RNG (PRNG) which have a deterministic sequence, and the True RNG (TRNG) which generates unpredictable random numbers. Cryptographic applications use both TRNG and PRNG. The PRNG needs an initial value, or seed, which can be the output of a TRNG. In digital technologies, like FPGAs, TRNG are commonly based on oscillators which have the drawback of being biased by harmonic coupling. In order to assess the entropic quality of TRNGs, standards based on statistical tests have been elaborated by certification organisms namely the NIST and the BSI. However, it is recommended to formalize the stochastic behaviour of the randomness generation process. In this Ph.D, we address the design and quality evaluation of TRNGs in digital circuits. We study of a low-cost digital TRNG without oscillators, hence robust against harmonics attacks. The proposed TRNG exploits both the metastability phenomenon and the jitter noise in CMOS digital flip-flops to generate the random numbers. A stochastic model of this TRNG has been formalized. This model describes the random generation process regardless of the targeted technology. The characterization and evaluation on a prototype circuit, in FPGA and ASIC technologies, has shown that the proposed TRNG architecture generates randomness of good quality and is robust against environmental variations.

KEY-WORDS : True Random Numbers Generator, metastability, ASIC, programmable delay, delay chains, FPGA, noise, standard statistical tests, NIST, AIS-31



TELECOM ParisTech

EDiTE

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY