



**HAL**  
open science

# ASIC Design Methodology for 3D NOC Based Heterogeneous Multi Processor on Chip

Abir M'Zah

► **To cite this version:**

Abir M'Zah. ASIC Design Methodology for 3D NOC Based Heterogeneous Multi Processor on Chip. Micro and nanotechnologies/Microelectronics. Ecole Polytechnique X, 2012. English. NNT: . pastel-00769455

**HAL Id: pastel-00769455**

**<https://pastel.hal.science/pastel-00769455v1>**

Submitted on 1 Jan 2013

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



## THESE DE DOCTORAT

En co-tutelle entre

Ecole Doctorale de l'Ecole Polytechnique

Université Tunis El Manar

**Spécialité : Physique Polytechnique**

**Spécialité : Génie Electrique**

Ecole : ENSTA ParisTech U2IS-Lab

Ecole : ENIT Tunis

Présentée par :

Mme Abir M'zah Ben Nejma

Sujet

# ASIC Design Methodology for 3D NOC Based Heterogeneous Multi Processor on Chip

Soutenue le 14/12/2012 devant les membres du Jury:

M. Bernard COURTOIS, Invité, Directeur de Recherches, CMP Grenoble, France

M. Marc DURANTON, Examineur, Docteur, CEA LIST Saclay, France

M. Omar HAMMAMI, Directeur, Professeur, ENSTA ParisTech, France

M. Jaouhar MOUINE, Co-directeur, Maître de conférences, ENIT, Tunisie

M. Smail NIAR, Rapporteur, Professeur, Université de Valenciennes, France

Mme. Laurence PIERRE, Examinatrice, Professeur, Laboratoire TIMA Grenoble, France

M. Hassan RABAH, Rapporteur, Professeur, Université Henri Poincaré Nancy, France

M. Frédéric ROUSSEAU, Examineur, Professeur, Laboratoire TIMA Grenoble, France



## ABSTRACT

ITRS Road Map predicts that the number of cores in the same chip will increase following an exponential curve. Insuring the interconnections between the different cores in the same chip is a real challenge when the number of components is high. The use of the NoC (Network On Chip) is a suitable solution overcoming the limitations of the classical interconnects methodologies. The regular NoC topology is costly in term of area and power consumption that is why designing an optimized architecture is a major problematic in MPSOC design. Moreover, with the semi-conductor CMOS shrinking, the interconnect delay has overcome the gate delay. In fact there is a real need to find other methodologies to continue the evolution of the chip design. 3D IC is one of the promising solutions which can reduce the interconnect delay, minimize the area of the chip and allow the use of mixed technologies. With the shortage of real 3D IC MPSOC implementation, we propose in this thesis to study the 3D design methodologies on ASIC for MPSOC architectures based on 3D NoC. Even though the NoC was proven to be an efficient solution to deal with the interconnect problems between the different cores, only few works have validated the architectures based NoC by a real implementation on FPGA/ASIC. We consider that the validation of 3D NoC by synthesis, place and route workflow is an essential step which guarantees the good functionality of the architecture before moving to 3D technology. That is why we have validated our MPSOC based 16 PEs architecture with a butterfly NoC on different FPGAs platforms. 3D IC design is facing new challenges like TSV assignment, heat dissipation and partitioning problems. That is why, in order to generate an optimized 3D NoC for a specific application and subject to the 3D Tezzaron technology, we propose in this work a new 3D NoC synthesis methodology based on MOEA. A real 3D IC design implementation of our tested and validated 3D MPSOC architecture was performed using the 3D IC Tezzaron technique. Our real case study represents a significant example proving that there is no actual 3D tool taking in consideration all the 3D IC challenges like mapping and partitioning.

Keywords: NOC, MPSOC, 3DIC, EDA tools, Validation/Verification

## RÉSUMÉ

---

La feuille de route d'ITRS prévoit que le nombre de processeurs dans la même puce va augmenter suivant une courbe exponentielle. Assurer la connexion entre les différents processeurs dans la même puce constitue un vrai défi quand le nombre des composants est important. L'utilisation d'un réseau sur puce est une solution efficace qui résout les problèmes des moyens classiques de connexion comme le bus et le point à point. Le réseau sur puce régulier coûte cher en termes de surface et d'énergie, c'est pourquoi la conception d'une architecture optimale représente une motivation majeure. En plus, avec la réduction de la taille des transistors, le temps de propagation dans les liens dépasse celui des portes logiques. En effet, il est indispensable de trouver de nouvelles techniques qui permettent de continuer le développement des circuits du semi conducteur. La conception 3D des circuits intégrés est une solution prometteuse qui peut réduire la longueur des liens, la surface de la puce et qui permet d'utiliser des technologies différentes dans la même architecture. Vu le manque d'implémentations réelles des architectures à base de multiprocesseurs avec la technique 3D, nous proposons dans cette thèse d'étudier les méthodologies de conception ASIC des architectures MPSOC à base du NoC 3D. Bien que les réseaux sur puce soient considérés comme une solution efficace pour le problème de connexions entre les processeurs, rares sont les travaux qui valident le NoC par une vraie implémentation sur FPGA/ASIC. Nous considérons que la validation d'un NoC par émulation nous permet de garantir la bonne fonctionnalité de notre architecture lors de l'implémentation en 3D. La technique de conception en 3D IC est confrontée à plusieurs problèmes comme le placement des connexions verticales, la dissipation de chaleur et le problème de partitionnement. Dans ce cadre, nous proposons dans cette thèse une nouvelle méthodologie de synthèse NoC 3D qui se base sur les algorithmes évolutionnaires. Nous avons implémenté une architecture MPSOC avec la technologie 3D de Tezzaron. Notre cas d'étude représente une architecture significative qui tient en considération les contraintes de la technologie 3D de Tezzaron.

Mots clés : NOC, MPSOC, 3DIC, EDA tools, Validation/Verification

## REMERCIEMENTS

---

Je tiens vraiment à remercier mes deux rapporteurs, Prof S. Niar et Prof H.Rabah, d'avoir accepté d'évaluer mon rapport de thèse. Je vous remercie pour le temps précieux que vous m'avez accordé. Vos remarques et votre évaluation ont très bien contribué à l'amélioration de mon manuscrit.

Je tiens à remercier tous les membres de jury d'avoir accepté de faire partie de mon comité de thèse. Je suis honorée par la participation de M.Courtois, M.Duranton, Mme Pierre et M.Rousseau dans l'évaluation de ma thèse.

Je voudrais maintenant remercier la personne que j'ai croisée dans l'accueil de l'ENSTA il y a maintenant trois ans et qui est devenu par un coup de destin inévitable mon directeur de thèse. Prof. Omar, merci de m'avoir appris des leçons pour toute ma vie. Je n'oublierai jamais mes moments de souffrance, de stress, de patience et de succès. Je vous remercie pour votre temps, pour votre énergie incroyable et pour votre confiance en moi. Je n'ai jamais croisé une personne comme vous, quand vous étiez parfois dur avec moi je regardais vos yeux très fatigués et je me disais que vous êtes plus dur avec vous-même et ça me donnait de la force pour continuer. C'était vraiment philosophique de faire une thèse avec vous.

Je tiens aussi à remercier mon co-directeur de thèse Prof. Jaouhar. Grâce à votre formation de base j'ai pu bien avancer dans mes études doctorales. Je vous remercie pour vos qualités humaines invraisemblables et pour votre support moral.

Enfin je me rappelle de tous les visages qui m'ont accompagnée pendant ma thèse. Je voudrais spécialement remercier Xinyu pour son encadrement au début de la thèse.

Khawla mon amie de thèse, je pense que tu étais mon meilleur support quand les choses allaient parfois mal tu étais là pour m'encourager et pour en discuter. Je suis très contente pour toi et pour la petite qui vient.

Thank you for Mohamed Hairol, we started the PHD together and it was a very good experience. I will always like Malaysian people because they are kind and frank like you.

Je remercie tous mes amis de l'ENSTA Isslem, Nessrine, Mazen, Ghassen, Souhir, Gangui d'avoir fait partie de ma vie à l'ENSTA.

## DEDICACES

---

Je dédie ce travail à mon cher mari Youssef. Je te remercie pour ton support moral merci d'avoir transformé mes larmes en rires. Je te remercie de m'avoir aidé à voir les choses simplement. Je t'ai demandé une fois de me cacher de toute cette vie mais je n'oublierai jamais ce que tu m'as dit : " tout le monde peut arrêter, c'est très facile, mais seule une personne forte puisse continuer ". J'ai voulu toujours te voir fier de moi et j'espère que nos sacrifices portent bien leurs fruits.

Je dédie ce travail à mes chers parents. Je vous aime fort, vous êtes ma raison d'être et mon grand amour dans cette vie. Je suis désolée maman par ce que je ne t'ai pas accompagné dans ta maladie j'aurais aimé être là pour te donner de ma vie si il le faut. Cher papa, tu étais parmi les gens à qui je pensais dans mes moments difficiles je savais que tu tenais vraiment à ce que je termine mes études alors je t'offre cette thèse comme une vraie reconnaissance pour ton amour pour moi.

Je dédie ce travail à ma chère sœur Imen et sa petite famille, tu me manques beaucoup, à mes deux frères Mokhtar et Aladine, je n'arrive pas à vous oublier je pense à vous chaque jour. Vous êtes mes deux anges diaboliques.

Je tiens à remercier mes beaux parents et toute ma belle famille. Je vous remercie pour votre amour et pour votre support.

Je dédie ce travail à toute ma grande famille.

Je dédie ce travail à ma chère amie Maryouma, tu es le meilleur cadeau que j'ai eu de l'ENSTA merci d'avoir été une sœur pour moi.

Je dédie cette thèse à tous mes amis et spécialement à Hana , l'amie de ma vie.

Enfin, je dédie ce travail à ma belle TUNISIE enfin libre, révolutionnaire et prometteuse !





## List of Abbreviations

- **MPSOC**: Multi Processor System On Chip
- **IC** : Integrated Circuit
- **NOC** : Network On Chip
- **GA** : Genetic Algorithm
- **MOEA** : Multi Objective Evolutionary Algorithm
- **LP** : Linear Programming
- **TSV** : Through Silicon Via
- **EDA** : Electronic Design Automation
- **IC** : Integrated Circuit
- **ILP** : Integer Linear Programming
- **MOGA** : Multi Objective Genetic Algorithm
- **ID** : Individual
- **FSL** : Fast Simplex Link
- **IP** : Intellectual Property
- **PE** : Processing Element
- **CMP** : Chemical-Mechanical Polishing
- **WNS** : Worst Negative Slack
- **WTW** : Wafer To Wafer
- **DTD** : Die To Die
- **DTW** : Die To Wafer

# Table of Contents

Introduction .....	16
1 MPSOC State of The Art.....	21
1.1 Trends.....	21
1.2 MPSOC State of the Art.....	23
1.3 MPSOC Actual implementation.....	26
1.4 MPSOC Design methodologies.....	28
1.5 Conclusion.....	32
2 2D MPSOC Design and implementation .....	34
2.1 Theoretical Complexity Problems in 2D Design and implementation.....	34
2.2 Regular NoC implementation on FPGA: case study Butterfly.....	35
2.2.1 Synthesis results .....	39
2.2.2 Parallel Programming: Filter Harris .....	40
2.3 NoC Design Space exploration on FPGA .....	42
2.3.1 ModeFRONTIER tool.....	42
2.3.2 Multi objective Genetic Algorithm NSGA-II Algorithm.....	45
2.3.3 ModeFRONTIER project: MOEA on FPGA.....	46
2.3.4 Machines Specifications.....	49
2.3.5 Sequential DSE.....	50
2.4 Parallel and multi-scale software implementation.....	54
2.5 Return on experience: Analyses and discussions .....	60
2.6 Conclusion.....	60
3 3D Semi conductor Technology.....	63
3.1 3D Semi conductor Technology: Motivation.....	63
3.2 3D Semi conductor Technology: State of the Art .....	64
3.3 3D Design Methodologies.....	68
3.3.1 Wire Bonded System-in-Package.....	68
3.3.2 Peripheral Vertical Interconnects .....	69
3.3.3 Micro Bumps.....	70
3.3.4 Through silicon via (TSV).....	72
3.3.5 Contactless.....	78
3.4 Benefits and challenges in 3D Design.....	78
3.4.1 Benefits of 3D Design .....	78

3.4.2	Challenges of 3D Design.....	81
3.5	3D Academic and industrial devices .....	84
3.5.1	3D Academic .....	85
3.5.2	3D industrial .....	85
3.6	Conclusion.....	85
4	NoC Synthesis methodologies.....	88
4.1	2D NoC synthesis methodologies .....	88
4.1.1	Deterministic methods.....	88
4.1.2	Mixed methods .....	89
4.1.3	Heuristic methods.....	89
4.2	FPGA based NoC synthesis Design Flow .....	91
4.3	Case study and performance evaluation results.....	95
4.3.1	Introduction to linear programming LP .....	95
4.3.2	OPL Modelling and CPLEX solver.....	96
4.3.3	Our LP Problem Definition .....	96
4.3.4	Experimental Results.....	102
5	NoC Synthesis Methodology for 3D ASIC Design.....	110
5.1	3D NoC synthesis state of the Art .....	110
5.2	3D NoC synthesis design Flow .....	112
5.3	Tezzaron Technology methodology .....	117
5.4	3D NoC Synthesis with GA .....	121
5.5	Performance Evaluation Results.....	127
5.5.1	Case Study .....	127
5.6	Conclusion.....	134
6	ASIC Design Methodology for 3D NoC based 3D Heterogenous Multiprocessor On Chip .....	136
6.1	3D Multiprocessor Architecture Homogenous.....	136
6.2	3D Heterogeneous Multiprocessor architecture .....	141
6.3	3 Hardware Accelerator synthesis in 3D Heterogeneous Multiprocessor architecture .....	143
6.4	Conclusion.....	143
7	Theoretical Complexity and Parallel EDA for 3D .....	146
7.1	Parallel EDA: Hierarchical MPSOC based 64 PEs on FPGA.....	146
7.2	3D Theoretical Complexity from return of experience .....	150
7.2.1	Core to layer mapping .....	150
7.2.2	Floorplanning .....	150

7.2.3	NoC topology .....	150
7.2.4	NoC floorplanning.....	150
7.3	Parallel EDA for 3D IC implementation.....	151
7.4	Parallel EDA for 3D : Case study.....	153
7.5	Conclusion.....	155
8	3IC Design and Modelling Case of Tezzaron .....	157
8.1	MPSOC basic components .....	157
8.1.1	Processor .....	157
8.1.2	Fast Simplex Link (FSL) Bus.....	157
8.1.3	3D Router .....	158
8.2	Architecture 1 : MPSOC1 based on mesh topology.....	159
8.3	Architecture 2 : MPSOC2 based on Butterfly topology.....	161
8.4	Implementation results and discussion.....	162
8.5	Complexity of 3D implementation.....	164
8.6	3D IC Fabrication.....	165
8.7	Conclusion.....	169
9	Conclusion.....	171
	References .....	176
	List of Publications.....	194

## Table of Figures

Figure 0.1. Global and local wire delay evolution [1].....	16
Figure 1.1 Design Complexity trend [2] .....	21
Figure 1.2. Point to Point Architecture .....	22
Figure 1.4. Industrial MPSOC number of cores evolution [4] .....	23
Figure 1.5. IEEE Xplorer hits for different “network-on-chip” researches [5] .....	24
Figure 1.6. Tile 64 Block Diagram Processor[17] .....	27
Figure 1.7. SpiNNaker MPSoC block diagram[13] .....	27
Figure 1.8. SpiNNaker MPSoC plot[13] .....	27
Figure 1.9. Kumar et al MPSOC Design flow[18].....	28
Figure 1.10. Application Specific MPSOC workflow [22] .....	28
Figure 1.11. Xpipes Synthesis Flow[23] .....	29
Figure 1.12. STARSOC Design flow overview .....	30
Figure 1.13. Automatic heterogeneous design flow [25] .....	31
Figure 1.14. MPSOC Methodology Workflow of SpiNNaker [13] .....	32
Figure 2.1. MPSOC based Butterfly NoC: 2Ary 4Fly Architecture .....	36
Figure 2.2. Our MPSOC implementation workflow .....	39
Figure 2.3. Harris Filter Execution Time and speed up .....	41
Figure 2.4. ModeFRONTIER project example .....	42
Figure 2.5. ModeFRONTIER initial population .....	43
Figure 2.6. NSGA-II algorithm illustration.....	46
Figure 2.7. ModeFRONTIER DSE Project.....	47
Figure 2.8. Pareto Front DSE of Harris Filter on MPSOC 16x16.....	48
Figure 2.9. SSM IP Architecture.....	50
Figure 2.10. MPSOC7: Execution time variation in function of the population size:.....	51
Figure 2.11. THALES2 : Comparison of FlexNoC (a) EDK (b) execution with different population sizes.....	52
Figure 2.12. MPSOC4: Comparison of FlexNoC (a), EDK (b), zCui (c) with different population sizes .....	53
Figure 2.13. Design Space Exploration for mathematical Model generation.....	55
Figure 2.14. MPSOC7 : Comparison of FlexNoC (a), EDK (b) execution with different numbers parallel Ids... ..	56
Figure 2.15. THALES2: Comparison of FlexNoC (a), EDK (b) execution with different numbers parallel Ids.. ..	57
Figure 2.16. SSM IP 48 processors 32 BRAMs.....	58
Figure 3.1. Gate and Interconnect Delay as a function of gate technology [39] .....	63
Figure 3.2. Cost CMOS scaling[40].....	64
Figure 3.3. Example of 3D Design [42] .....	64
Figure 3.4. Illustration of the evolution of the semi conductor technology with CMOS scaling with other ways of development offering new functionalities [2] .....	65
Figure 3.5. 3D IC industriel design .....	67
Figure 3.6. Illustration of vertical interconnect technologies: wire bonded (a); microbump—3D package (b) and face-to-face (c); contactless—capacitive with buried bumps (d) and inductive (e); through via—bulk (f) and silicon on insulator (g) [54].....	68
Figure 3.7. Wire bonding design [55] .....	69
Figure 3.8. Wire bonded System-in-Package[57] .....	69
Figure 3.9. SiP with peripheral connections: (a) solder balls (b) through-hole via and spacers .....	70
Figure 3.10. 3D Chip with Micro Bumps.....	70
Figure 3.11. Process flow fabrications of CuSn solder Microbump[60].....	71
Figure 3.12. SEM picture a die part of the interwoven daisy chain with 10µm diameter CuSn bumps formed by electrochemical plating. The pitch of the bumps is 20µm [61].....	71
Figure 3.13. Through silicon via (TSV)[62] .....	72
Figure 3.14. TSV examples [64] .....	73
Figure 3.15. General TSV flow fabrication [65] .....	73
Figure 3.16. Via first (left), Via last (right) 3D IC methodologies[63] .....	74

Figure 3.17. Stacking Methods[66].....	76
Figure 3.18. RC delay vsTSV diameter[67].....	77
Figure 3.19. TSV Area estimation[68].....	77
Figure 3.20. 3D IC Inductive coupling[57]      Figure 3.21. 3D IC capacitive coupling[70].....	78
Figure 3.22. Area reduction with 3D Stacking [71] .....	79
Figure 3.23. Average latency 3D IC and 2D IC [67] .....	80
Figure 3.24. Number of repeaters with different technologies.....	80
Figure 3.25. 3D chip with heterogenous technologies [72].....	81
Figure 3.27. Temperature distribution along the z Axis with different Silicon layers [69].....	82
Figure 3.28. Microfluidic cooling .....	82
Figure 3.29. Thermal vias for heat dissipation [74] .....	82
Figure 3.30. Examples of TSV defects: insufficiently filled TSV (right), TSV containing Micro voids (left).....	83
Figure 3.31. 3D IC testing model[75] .....	83
Figure 3.32. Geographic mapping of 3D IC players [76].....	84
Figure 3.33. 3D TSV applications and players [77].....	85
Figure 4.1. NoC synthesis on FPGA [87] .....	91
Figure 4.2. 2D NoC synthesis workflow .....	92
Figure 4.3. Prediction of the Frequency variation when the number and the average node degree of the benchmark change [88] .....	93
Figure 4.4. An FPGA design flow[19].....	94
Figure 4.5. The overall flow for analyzing multiple use-cases: the software part is performed for each application, the hardware is performed only one time [89].....	94
Figure 4.6. LP graphical solution .....	96
Figure 4.7. 263 enc MP3 Dec : coregraph (left), NoC topology (right).....	104
Figure 4.8. MPEG4 Decoder[78] .....	105
Figure 4.9. MPEG 4 Decoder NoC topology .....	105
Figure 4.10. H264 Decoder.....	106
Figure 4.11. H264 Decoder NoC topology .....	106
Figure 4.12. Coregraph partitioning.....	107
Figure 5.1. 3D Design flow.....	113
Figure 5.2. Algorithm Steps .....	113
Figure 5.3. Communication graph with bandwidth demands on the edges [97] .....	114
Figure 5.4. Partitioning Graph (PG) and the min-cut partitions[97] .....	114
Figure 5.5. Scaling Parameter Graph (SPG)[97].....	114
Figure 5.7. (left) D26_media communication, (middle) NoC architecture phase 1, (right) NoC Architecture phase 2 [97].....	115
Figure 5.8. 3D NoC synthesis Design flow with GA, .....	115
Figure 5.9. 3D NoC synthesis Design flow based on floorplanning[92].....	116
Figure 5.10. 3D NoC synthesis workflow[95] .....	117
Figure 5.11. 3D-IC Automatic P&R using DBI and TSV.....	119
Figure 5.13. Signal to Bumps assignment.....	120
Figure 5.12. Create Bumps Array .....	120
Figure 5.14. Create pins under Bumps Tezzaron technology.....	121
Figure 5.15. Our 3D NoC synthesis workflow.....	122
Figure 5.16. Coregraph 1 : 12 Masters 8 slaves .....	127
Figure 5.17. Case 1: Error Path constraint evolution .....	129
Figure 5.18. Case 1 : Area Constraint evolution .....	129
Figure 5.19. Case 2 Min area constraint evolution.....	130
Figure 5.20. Case 2 Error path constraint Evolution.....	130
Figure 5.21. Case 3 : Area constraint evolution .....	131
Figure 5.22. Case 3 : Overlapping Constraint .....	132
Figure 5.23. Case 3 : Path constraint.....	132
Figure 5.24. Our 3D NoC synthesis floorplan Solution .....	133

Figure 5.25. The optimized NoC topology.....	133
Figure 6.1. Various layout views of the 3D-MAPS processor[48] .....	136
Figure 6.2. The structure of the PE [49].....	138
Figure 6.3. The SAR FFT processor architecture[49].....	138
Figure 6.4. T. Thorolfsson 3D Design flow[49].....	139
Figure 6.5. The SAR FFT floorplan[49] .....	139
Figure 6.6.Schematic and layout view of 3D SoC H.264 Application [51] .....	140
Figure 6.7. 3D DRAM stacking [51].....	140
Figure 6.8. The estimated cost of OpenSPARC : the separate core and memory fabrication reduces the cost [101] .....	141
Figure 6.9. cross section of the final package .....	142
Figure 6.10. 3D processor architecture CMP-FPGA[103] .....	143
Figure 7.1. MPSOC architecture 64 PEs on Multi-FPGA platform Zebu-UF4 .....	146
Figure 7.2. Parallel workflow EDA MPSOC implementation on FPGA .....	148
Figure 7.3. 3D IC parallel and automatic workflow.....	152
Figure 7.4. The placed and routed Filter with 3D Tezzaron Technology.....	153
Figure 7.5.. WNS (ps), Density (%) and power (mw) for the different combinations of the DSE (Freq=100MHZ) .....	154
Figure 8.1. Openfire processor architecture .....	158
Figure 8.2. 3D Router architecture.....	159
Figure 8.3. Network interface architecture.....	159
Figure 8.4. Architecture 1: MPSOC based on Mesh topology .....	160
Figure 8.5. Tile Block Diagram .....	160
Figure 8.6. Architecture 2 : MPSOC Based on the butterfly architecture .....	161
Figure 8.7. . MPSOC1 Mesh: Bottom tier routed layout.....	162
Figure 8.8. MPSOC2 Butterfly: Bottom Tier routed layout.....	162
Figure 8.9. 3D MPSOC partitioning .....	164
Figure 8.10. Tezzaron 3D Techniques: Super-Via(left), Super Contact(right)[107].....	165
Figure 8.11. Illustration of Tezzaron's Stacking method with the " Super Contact" Interconnect[107].....	167
Figure 8.12. Tezzaron metal bonding[62] .....	167
Figure 8.13. Deposit D2D Vias[108] .....	167
Figure 8.14. CMP step for stack thinning [108].....	168
Figure 8.15. Etching for power/Gnd TSV [108] .....	168
Figure 8.16.The Heat Sink creation [108].....	168

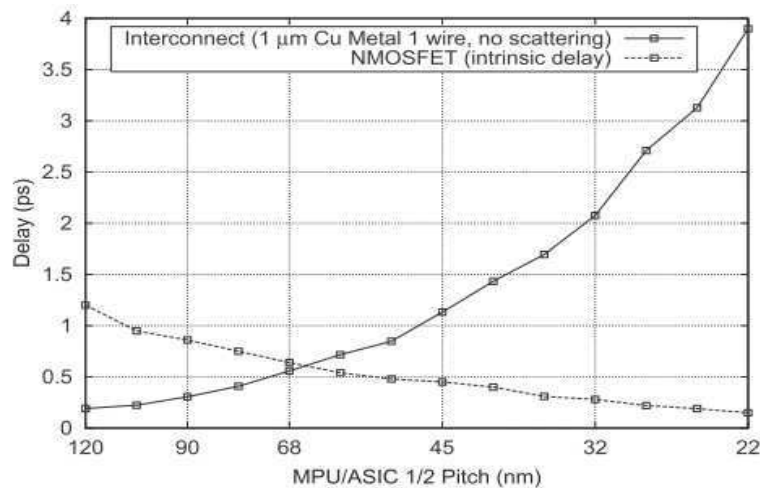
## List of Table

Table 1.1. ITRS 3D Interconnect TSV Roadmap.....	22
Table 1.2. MPSOC state of the Art.....	24
Table 1.3. MPSOC using busses communication .....	26
Table 1.4. Actual MPSOC implementation.....	26
Table 2.1. Comparison Mesh and Butterfly topology .....	35
Table 2.2 Address of the slaves .....	36
Table 2.3. Different switch routing tables.....	37
Table 2.4. Resource utilization of Zebu-UF4 .....	39
Table 2.5. Microblaze parameters .....	47
Table 2.6. Switch Options .....	48
Table 2.7. The properties of the machines .....	49
Table 2.8. ZEBU UF4 Emulator specifications.....	50
Table 2.9. SSM IP 48x32 on Zebu UF4 Resource utilizations.....	58
Table 2.10. SSM IP 48x32 Exploration Results.....	59
Table 3.1. 3D MPSOC design implementation [50] .....	66
Table 3.2. High-density through silicon via projections in 2008 ITRS update [63] .....	72
Table 3.3. TSV process flows [63].....	74
Table 3.4. Comparison between bonding methods (KGD: Known Good Die)[60] .....	75
Table 3.5. Comparison of Via Filling Materials [55].....	76
Table 3.6. Performance and power comparison between different 3D architectures[71] .....	79
Table 3.7. Reduction of integrated yield with stacking using wafer on wafer [70] .....	83
Table 4.1. Summary of 2D NOC synthesis Methods .....	90
Table 4.2. Semi conductor properties.....	103
Table 4.3. Properties and execution time for the different benchmarks.....	104
Table 4.4. Routers Description 263 Enc MP3 Dec.....	104
Table 4.5. Routers Description MPEG 4 Decoder .....	106
Table 4.6. Router configurations H264 Decoder.....	106
Table 5.1.3D NoC synthesis methodologies .....	111
Table 5.2. ARM used in the Design Kit Tezzaron library.....	119
Table 5.3. MOEA Project parameters .....	123
Table 5.4. Case study different configurations.....	128
Table 6.1. Physical Design Summary [48].....	137
Table 6.2. Architectural Performance Metrics[48].....	137
Table 7.1. Resources utilization .....	148
Table 7.2. 64PE MPSOC Used IPs .....	148
Table 7.3. Resource utilization of the MPSOC based 64 PEs NoC[104].....	149
Table 7.4. Options of FILTER DSE.....	151
Table 7.5.Different ID (L: Low, H: High, M: Medium) .....	155
Table 8.1. MPSOC Implementation results.....	164



## Introduction

ITRS road map has predicted that the number of cores in the same chip will increase following an exponential curve. This was the major motivation to create new techniques to solve the interconnection problem between high number of cores in the same MPSOC. The use of the Network On Chip represents an efficient solution to deal with classical interconnect methods limitations like the point to point and the shared bus. But with the evolution of the CMOS semi-conductor reaching the Nanometre scales, the interconnect delay is overcoming the gate delay which is illustrated in Figure 0.1. The Interconnect delay is the new performance limitation of the chip. The reduction of the interconnection length is the new challenge in the MPSOC design. 3D IC is emerging as a suitable solution to reduce the global interconnects delay thanks to the use of the vertical links.



**Figure 0.1. Global and local wire delay evolution [1]**

Even though 3D IC design is not a new methodology, the number of works making real 3D ASIC implementation is too limited. The major problem of the 3D IC design is the shortage of 3D IC dedicated EDA tools. In fact, there is no complete industrial software allowing the implementation of all the steps of 3D workflow.

The objective of this work is to evaluate the different MPSOC design methodologies used in 3D IC design. We focus on the implementation of the NoC synthesis with heterogeneous architectures.

We propose in the first part of this work, to explore different MPSOC implementation methodologies. A design space exploration of the different configurations of the hardware

architectures will be performed in order to find the optimal one in term of cost and performance. The objective of this work is to validate the used MPSOC architectures before their implementation on 3D IC.

The main objective of this thesis is to propose a new 3D NoC synthesis methodology taking in consideration the 3D IC used technology. A real implementation with 3D ASIC design will be performed in order to show up the advantages of the 3D design compared to the obtained results in 2D.

The organization of this report is directly related to the scientific approach and methodology adopted during this PHD studies. In fact, the first step of this work which is the MPSOC state of the art represents an important introduction to the general research field. We can then propose an MPSOC design implementation of an interesting architecture when compared to the actual industrial and research achievements. Chapter two and three are performed in parallel in order to have a good 2D experimental knowledge and a rich 3D theoretical background which are necessary before the NoC synthesis section. As we believe that the 3D NoC synthesis problem has a very high complexity, we propose to start by exploiting our Lab experience in 2D NoC synthesis by proposing a new NoC synthesis model based on the Microelectronics characteristics. After having a first experience in the 2D NoC synthesis we can then move to the 3D NoC synthesis chapter with an efficient strategy. The last three chapters concern the 3D IC MPSOC state of the art, complexity and real implementation.

We present in chapter 1 of this thesis, the state of the art of the existing MPSOC real implementations listed in the literature. We also detail the different design methodologies with real implementations.

We introduce in chapter 2 the 2D design and implementation of our MPSOC architecture based on software processors and regular NoC. The execution result of a parallel image processing on FPGA will be reported. We will use the GA in order to explore and evaluate the performance of the different industrial tools.

Chapter 3 will be the subject of 3D semi conductor technology state of the art. We will detail the different 3D ASIC design methodologies. We will then discuss the different issues and challenges in 3D design.

The NoC synthesis methodologies will be the subject of chapter 4. We will define the exact, the mixed and the heuristic methods already used in literature. Our NoC synthesis 2D solution will be presented as a case study of this chapter.

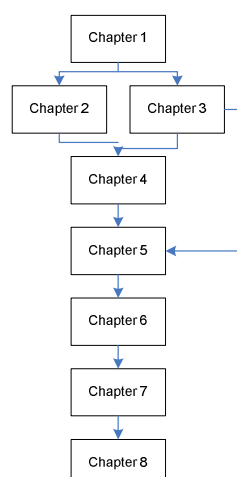
We will introduce in chapter 5 the different properties of the 3D Tezzaron technology. We will also detail the 3D ASIC methodologies presented in the literature. The state of the art of the 3D NoC synthesis problem will be summarized. We will detail in this chapter our 3D NoC synthesis methodology based on the MOEA.

In chapter 6, we will study different MPSOC architectures based on heterogeneous components. We will then discuss the different methodologies to create MPSOC architectures using 3D Hardware Accelerator.

We will evaluate in chapter 7, the theoretical complexity and the experimental results of our parallel EDA of the 3D NoC synthesis problem. We will perform a design space exploration on the cadence tool to study the effect of the different properties on the synthesis, place and route results.

Chapter 8, will present the 3D ASIC design implementation of our MPSOC using the 3D Tezzaron technology. The results of synthesis, place and route of our 3D architecture will be reported.

The last part of this report will include the conclusion and our future work.



*PHD Plan and Scientific Approach Organization*



## Chapter 1 : MPSOC State of The Art

# 1 MPSOC State of The Art

The use of a single processor is not any more possible in the industrial products like smart phones and medical devices needing high computational time and fast parallel programming. That is why; the use of Multi Processor System On Chip (MPSOC) is emerging. As a real example, we can notice that the MPSOC Cortex-A9 of ARM Company was included in many industrial chips like Nvidia's Tegra 2 and Samsung's Exynos 4210.

## 1.1 Trends

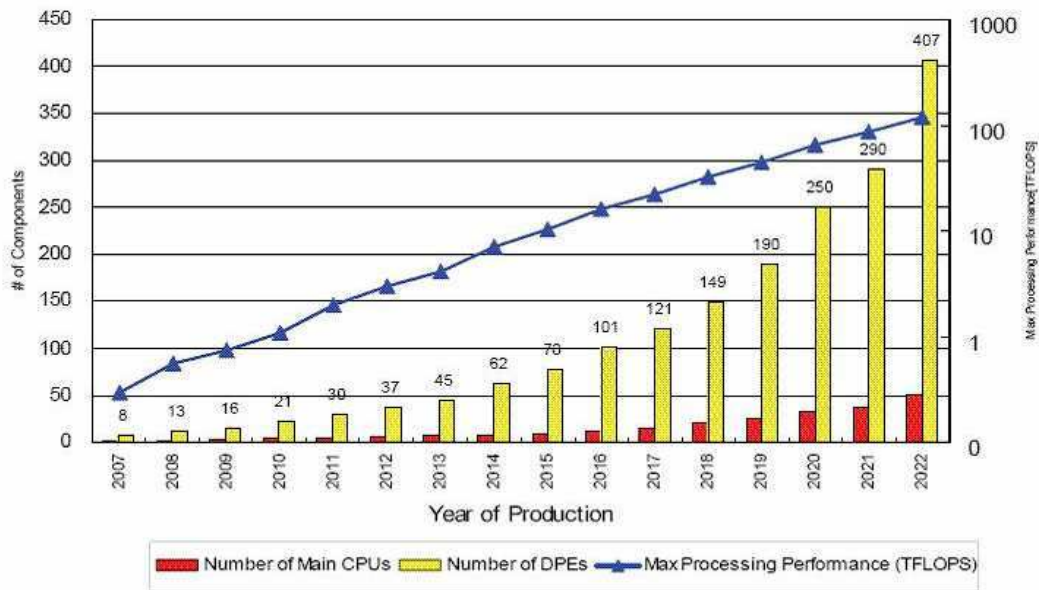
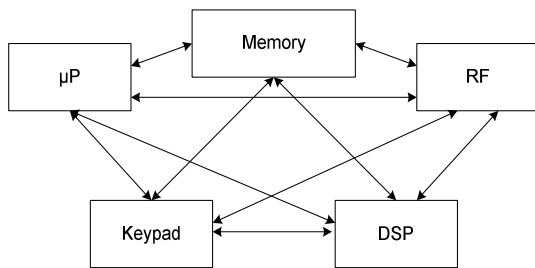


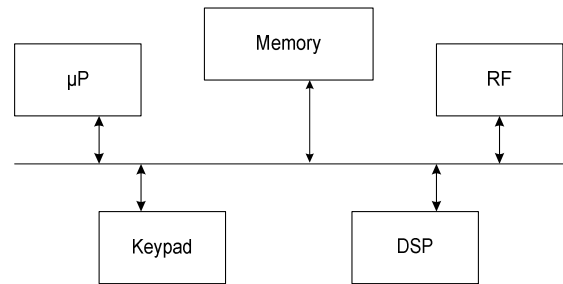
Figure 1.1 Design Complexity trend [2]

The prevision of the International Technology Roadmap in Semi conductor (ITRS) was considered as a sort of science fiction in the beginning of the last decades but researchers have respected and sometime they have even exceeded this prevision. In fact, the evolution of the number of IPs in the same chip was supposed to double each about 18 months as presented in Figure 1.1. For this, the first alternative was scaling the technology but this methodology has almost reached its limitation face to the physical problems of semi conductor. In Figure 0.1, we can see that the global wire delay is becoming more important than gates delay in Nanometre technologies. The global delay of the NoC is limited in this case not by the gate delay but by the global wires delay. The connectivity becomes a major problem when we increase the number of cores in the same chip. With the high complexity of the multiprocessors system on chip MPSOC, the need for scalable and efficient communication architectures to support inter-core data transfers has become paramount. Network-on-Chip (NoC) fabrics have been shown to provide superior communication

bandwidth, scalability, and modularity compared to traditional bus-based architectures (Figure 1.2 and Figure 1.3).



**Figure 1.2. Point to Point Architecture**



**Figure 1.3. Bus Architecture**

NoCs have gradually gained acceptance as the dominant interconnection paradigm for emerging CMP systems with tens to hundreds of cores. The interconnect challenge is one of the major problems in MPSOC architectures. Once solved with the NoC solution, a new limitation related to the interconnect length is now faced. In fact, the design of future MPSOC architectures especially with Nanometre technologies should minimize the interconnection length in order to increase the performance.

A new solution is emerging to deal with this limitation which is the 3D ASIC design. In fact, with a 3D MPSOC, the global routing length can be shortening thanks to the use of the vertical connection called TSV. ITRS roadmap presents in the report published in 2011[3], the prediction of the evolution in the world of 3DIC which is presented in Table 1.1. Referring to this table, the reduction of the TSV diameter and pitch will reach 50% between the years 2012-2015. This can be a major reason to increase the use of the 3DIC in the future chips.

**Table 1.1. ITRS 3D Interconnect TSV Roadmap**

<b>Global Level, WTW, DTW, or DTD 3D stacking</b>	<b>2009-2012</b>	<b>2012-2015</b>
Minimum TSV diameter	4-8 $\mu\text{m}$	2-4 $\mu\text{m}$
Minimum TSV pitch	8-16 $\mu\text{m}$	4-8 $\mu\text{m}$
Minimum TSV depth	20-50 $\mu\text{m}$	20-50 $\mu\text{m}$
Maximum TSV aspect ratio	5:1-10:1	10:1-20:1
Bonding overlay accuracy	1.0-1.5 $\mu\text{m}$	0.5-1.0 $\mu\text{m}$
Minimum contact pitch(thermo compression)	10 $\mu\text{m}$	5 $\mu\text{m}$
Minimum contact pitch ( solder or SLID)	20 $\mu\text{m}$	10 $\mu\text{m}$
Number of tiers	2-3	2-4
<b>INTERMEDIATE Level, WTW 3D stacking</b>	<b>2009-2012</b>	<b>2012-2015</b>
Minimum TSV diameter	1-2 $\mu\text{m}$	0.8-1.5 $\mu\text{m}$
Minimum TSV pitch	2-4 $\mu\text{m}$	1.6-3 $\mu\text{m}$
Minimum TSV depth	6-10 $\mu\text{m}$	6-10 $\mu\text{m}$
Maximum TSV aspect ratio	5:1-10:1	10:1-20:1
Bonding overlay accuracy	1.0-1.5 $\mu\text{m}$	0.5-1.0 $\mu\text{m}$
Minimum contact pitch	2-3 $\mu\text{m}$	2-3 $\mu\text{m}$
Number of tiers	2-3	8-16(DRAM)

## 1.2 MPSOC State of the Art

The first real MPSOC was created in the beginning of the last decade based on two processors. The communication between cores was ensured in the beginning via the classical communication technologies like shared busses and point to point methodologies. With reference to the Figure 1.4, it is clear that the number of cores is increasing respecting an exponential curve. Until the year 2000, only single processors were available like the well known one which is the Pentium. This domain knew a real revolution in the previous decade with the apparition of different MPSOCs architectures. The evolution of the MPSOC is directly related the evolution of the interconnection methodologies.

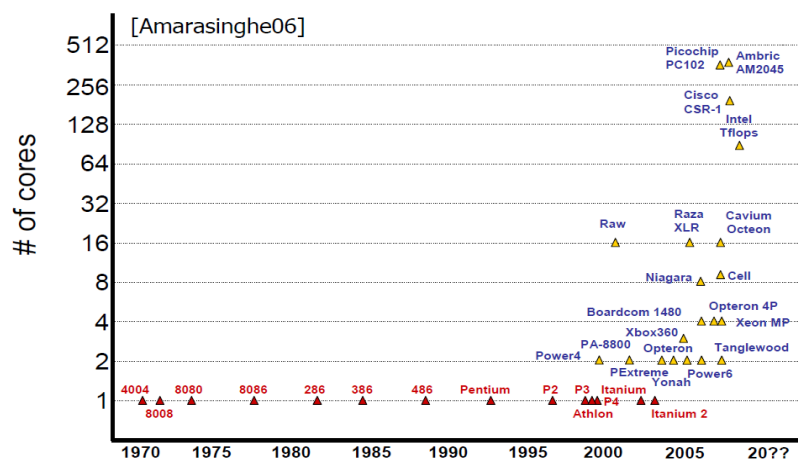


Figure 1.4. Industrial MPSOC number of cores evolution [4]

This last decade has known a real explosion in term of research and industrial products in these fields. The evolution of number of hits in the IEEE Xplorer for different NoC searches presented in Figure 1.5 can be a real witness of this trend. We can conclude from this curve that the Hardware field overcomes the Software filed in term of research papers. This can explain the decrease of the NoC and the MPSOC curves by the year of 2010. In fact, there is no meaningful need to increase the number of cores in the same chip if there is no suitable parallel application to use the hardware researches. Moreover, the competition in the MPSOC becomes costly with large Scale designs needing adequate emulation platforms instead of the simulation. Many big companies like INTEL, TILERA, Philips and ST Microelectronics are the leaders in this domain.



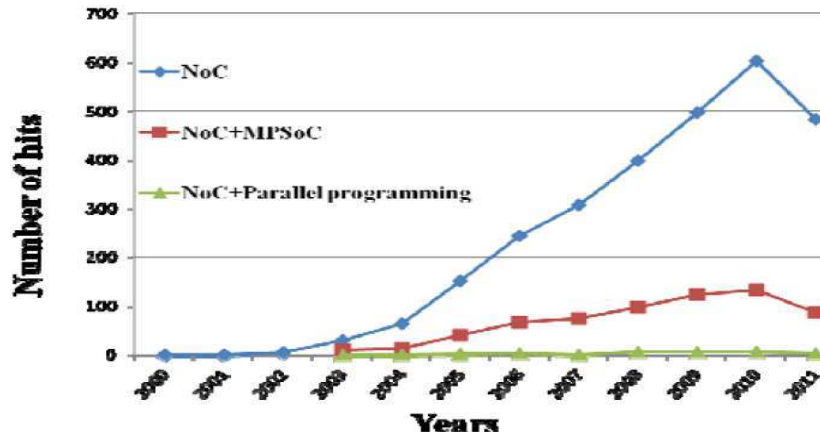


Figure 1.5. IEEE Xplorer hits for different “network-on-chip” researches [5]

The point-to-point architecture and the bus based communication schemes are considered as the origin of the first MPSOC topologies. We present in Table 1.2 some first industrial chips based on the first MPSOC architectures. Starting from a trivial idea to connect all the cores to each others, the P2P (point to point) method is an easy way to ensure the communication between all the components of a design. This solution can be suitable with few cores and reaches its limitation with larger designs. On the other hand, by using the shared Bus architecture we can connect tens of cores. These two methods suffer from the lack of scalability needed for big applications, but they are still used in the case of few cores in the same chip due to their simplicity in term of design.

Table 1.2. MPSOC state of the Art

MPSOC	Architecture	Application
Lucent Daytona [6]2000		Daytona was designed for wireless base Stations Processor : SPARC V8
Philips Viper Nexperia[7] 2001		Multimedia processing Processor : MIPS PR3940

<p>The C-5 Network Processor [8] 2002</p>		<p>Packet Processing in networks. Executive Processor: RISC CPU</p>
<p>The Intel IXP2855 [9]2002</p>		<p>Network processor</p>
<p>(TI) OMAP [10] 2003</p>	<p>TI OMAP 5912.</p>	<p>The cell phone processor. Processors : ARM9 and a TMS320C55x digital signal processor (DSP)</p>
<p>ARM MPCore. [11] 2005</p>		
<p>The ST Microelectronics Nomadik [12] 2006</p>	<p>ST Nomadik SA.</p>	<p>The cell phone processor. Processor : ARM926EJ</p>

**Table 1.3. MPSOC using busses communication**

Company	Name	Number of cores	Topology
INTEL	Intel IXP2850	2	Busses
Philips [7] 2001	Philips Nexperia <sup>TM</sup> PNX-8500.	2	Busses
TI 2002 [10]	TI OMAP <sup>TM</sup> 5910	2	Bridge
ST 2003 [12]	ST Nomadik <sup>TM</sup>	2	Bridge
Toshiba 2008	Venezia	8	Busses
TI 2008	TMS320VC544	4	Busses
ARM	ARM11	4	Busses

### 1.3 MPSOC Actual implementation

In the actual implementations, the use of the NoC in the MPSOC industry is taking more and more place. With the increasing number of cores, the classical interconnection technologies are not any more sufficient. We present in Table 1.4, a summary of actual MPSOC implementation. We notice that the most popular NoC topology is the Mesh due to its symmetry and to its simple routing algorithm. ARM has produced the Cortex family with the technology 40 nm and below and Tileria has reached the 64 cores with its MPSOC Tile 64 presented in Figure 1.6. We present in Figure 1.7 and Figure 1.8 respectively the architecture and the spot of the chip SpiNNaker including 18 processors.

**Table 1.4. Actual MPSOC implementation**

	NAME	Number of cores	Topology	Technology
2011[13]	SpiNNaker	18	Hierarchical	UMC 8-metal layer 130nm
2009 [14]	81.6 GOPS	10	Mesh	180nm
LETI 2009 [15]	The MAGALI	15	Mesh	ST 65nm
ARM 2009[16]	ARM Cortex-15 Cortex-5, Cortex-8	4 cores per cluster	Mesh	40nm & below
Tileria 2009 [17]	Tile 64 family	64	Mesh	90nm
MIPS 2009	MIPS32 1074K	Dual core	Bus	
MIPS 2009	MIPS32 1074K	Dual core	Bus	

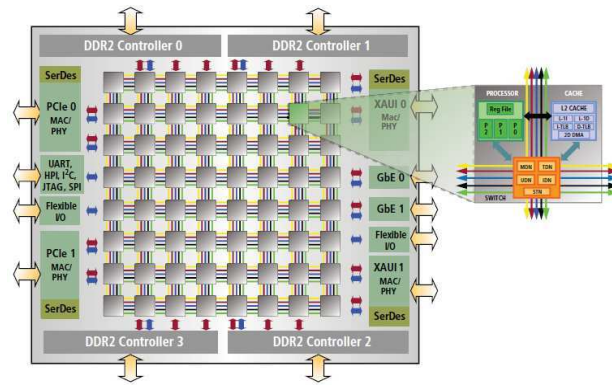


Figure 1.6. Tile 64 Block Diagram Processor[17]

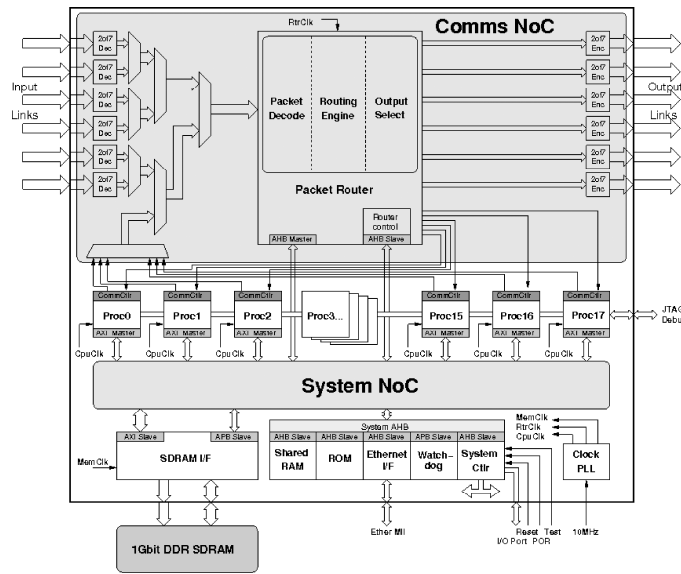


Figure 1.7. SpiNNaker MPSoC block diagram[13]

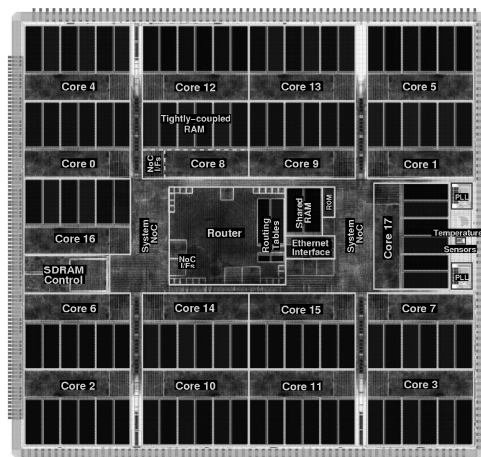


Figure 1.8. SpiNNaker MPSoC plot[13]

### 1.4 MPSOC Design methodologies

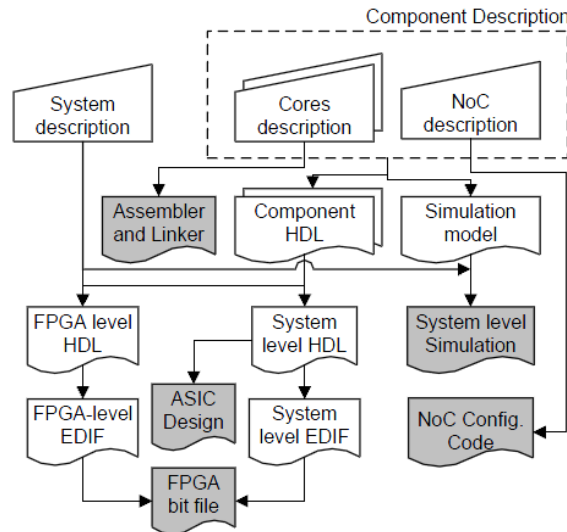


Figure 1.9. Kumar et al MPSOC Design flow[18]

Kumar et al proposed in [19], a full and complete MPSOC design workflow. They proposed to generate a customized NoC and cores in the same workflow design. The Silicon Hive [20] processing core was used. This one is an entire tool chain for rapid design of custom cores. For the NoC generation they proposed to use *Æthereal* [21] design flow. In Figure 1.9 , we present the proposed design flow. The system level description is considered as the input point of the design flow. In fact after the description of the NoC and the cores, the HDL entities are generated together with a simulation model. An .edif file is then generated automatically from Handel-C. These files and with the system level edif file are used during Place and Route (P&R) to obtain the bit file (for FPGA configuration). ASIC design can also be produced from the system-level HDL if desired.

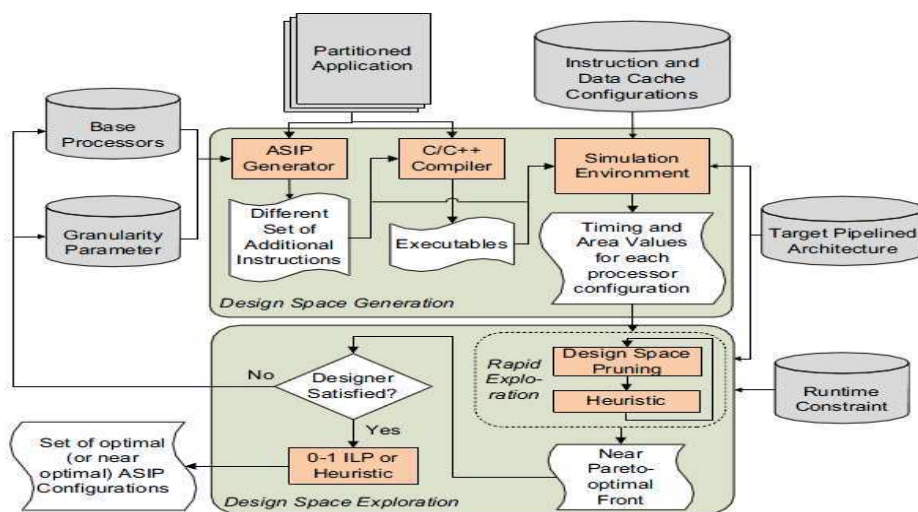


Figure 1.10. Application Specific MPSOC workflow [22]

In [22] the proposed design flow aiming to generate an application specific heterogeneous pipelined multiprocessor system, is composed of two separate stages which are: the Design Space Generation and the Design Space Exploration. The designer provides the partitioned application, the pipelined architecture and the runtime constraint as inputs. The different configurations and instructions are determined during the Design Space Generation phase. The user can choose the basic processors as an input to this design flow. In this methodology authors used ASIPs (Application Specific Instruction Set Processors) which is generated using a commercial tool Tensilicia. Thanks to the input parameters, a designer can control the amount of design space to be generated for a particular application. As shown in Figure 1.10 , the simulation results are used to record the timing and the area values for all the ASIP configurations which will be used in the exploration phase. In the second phase dedicated to the Design Space exploration, authors proposed to use heuristic approach to run a rapid exploration and to find a near Pareto front. This one will be the new design space to explore in the last step and to find the optimal configuration.

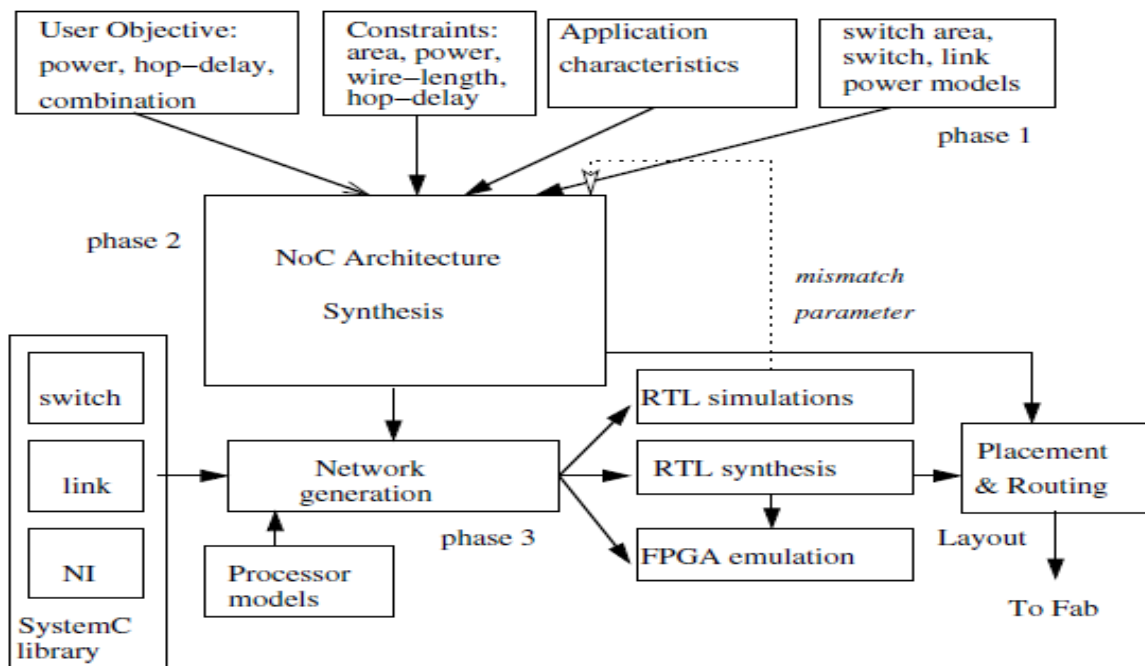


Figure 1.11. Xpipes Synthesis Flow[23]

In this paper [23] L. Benini proposed a new MPSOC Design flow based on the NoC architecture synthesis called Xpipes NoC synthesis Design flow. In the first phase, the user specifies the objectives and the constraints that should be satisfied by the designed NoC. The application traffic characteristics, the size of the cores, the area and the power models of the network components are also obtained. In the second phase of the flow, the NoC architecture that optimizes the user objectives and satisfies the design constraints is automatically

synthesized. In the last step, the XpipesLite is used to generate the RTL (SystemC) code of the switches, the network interfaces and the links for the designed topology. The RTL files can be then synthesized and implemented on FPGA. A place and route steps are performed using the industrial tool soc encounter from Cadence. The output of this phase is a total floorplan design specification which can be sent to the fabrication. A real implementation was realized in this work with a design consisting of 30 cores: 10 ARM7 processors with caches, 10 private memories (a separate memory for each processor), 5 custom traffic generators, 5 shared memories and devices to support inter processors communication.

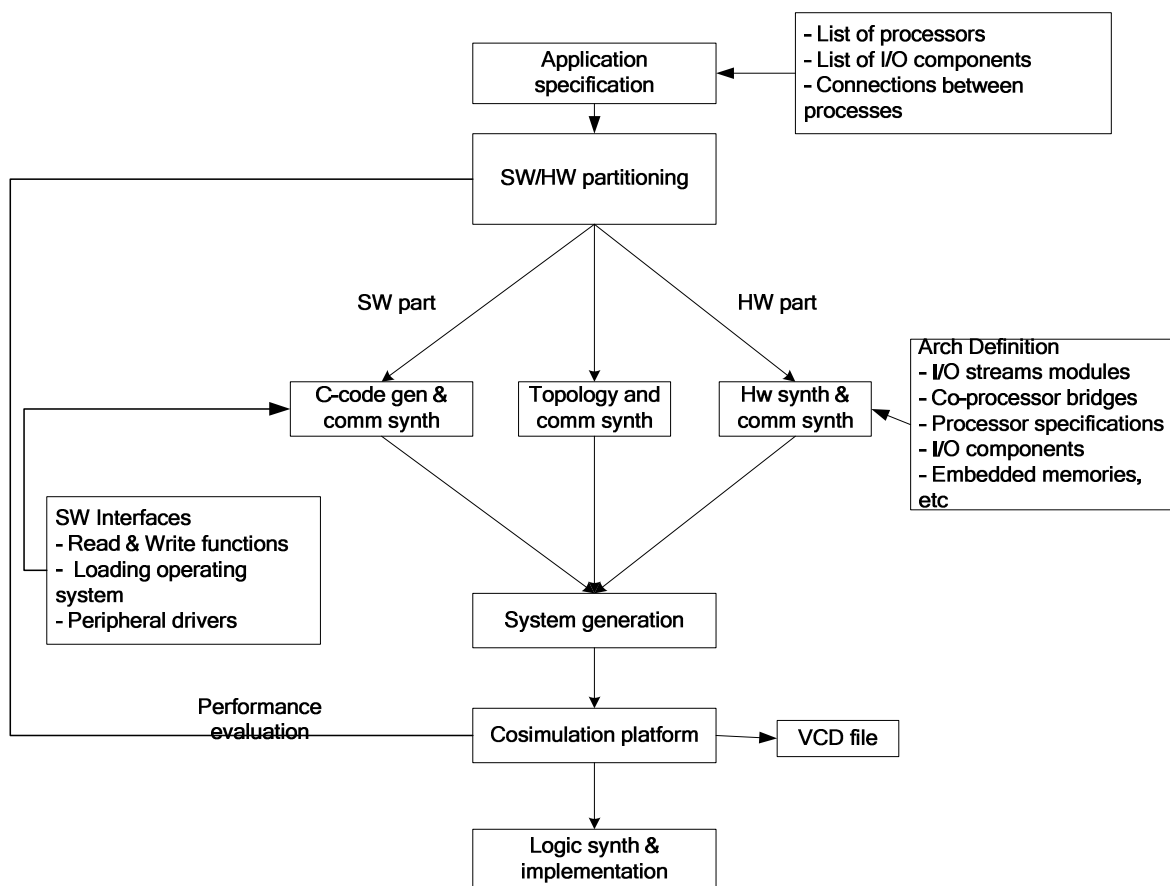
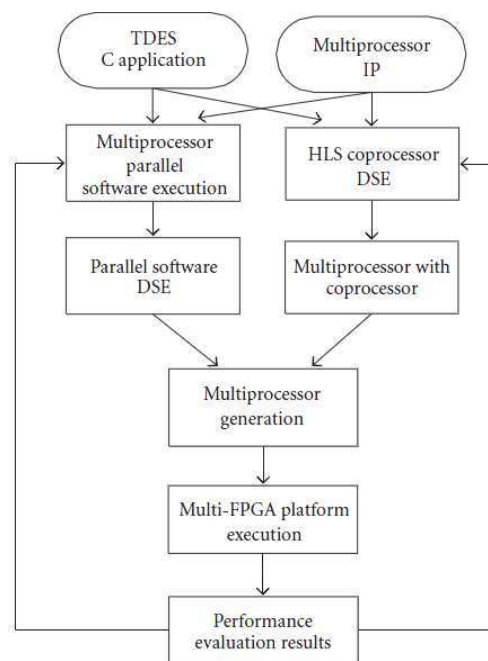


Figure 1.12. STARSOC Design flow overview

The input of STARSOC [24] design flow is a set of files in C code describing the whole design: the number of processors, their configurations and their interconnections. After the step of the hardware-software partitioning, the hardware part is synthesized into register transfer-level (RTL) architecture, and the software part is distributed on the whole of processors. The software part will be modified to ensure the operation of the hardware call. The RTL code generated by the high-level synthesis and the communication system synthesizer is then downloaded to the FPGA.

X.Li and O.Hammami presented in their work [25] an automatic Heterogeneous MPSOC design flow on Multi-FPGA. The input of the workflow is an ANSI C code of Triple Data Encryption Standard (TDES) and a small-scale multiprocessor (SSM) IP which will serve as the basic element for the parallelization process. At the first step, software parallelization is explored through direct execution on multi-FPGA platform to find out the best data parallel and the pipelined configuration. While the parallel programming ensures to achieve a maximum design space exploration, the second step is reserved to explore coprocessor – based TDES by incrementally adding TDES C-based synthesis generated coprocessor. The last step compares the two paths and chooses the appropriate one.



**Figure 1.13. Automatic heterogeneous design flow [25]**

We present in Figure 1.14, the workflow methodology to implement the SpiNNaker MPSOC. Plana et al used a hierarchical place and route method to implement this architecture considered complex. In fact the SpinNNaker includes different IPs devices ARM cores, SDRAM controller, timers, interrupt controller and watchdog as well as the devices developed specifically for SpiNNaker: multicast router, DMA controller and bridge, Ethernet interface, and system controller. In this method the place and route step of the IP blocks was applied separately. This was possible thanks to the fact that the MPSOC has a GALS nature which means that each IP can have its different clock signal.



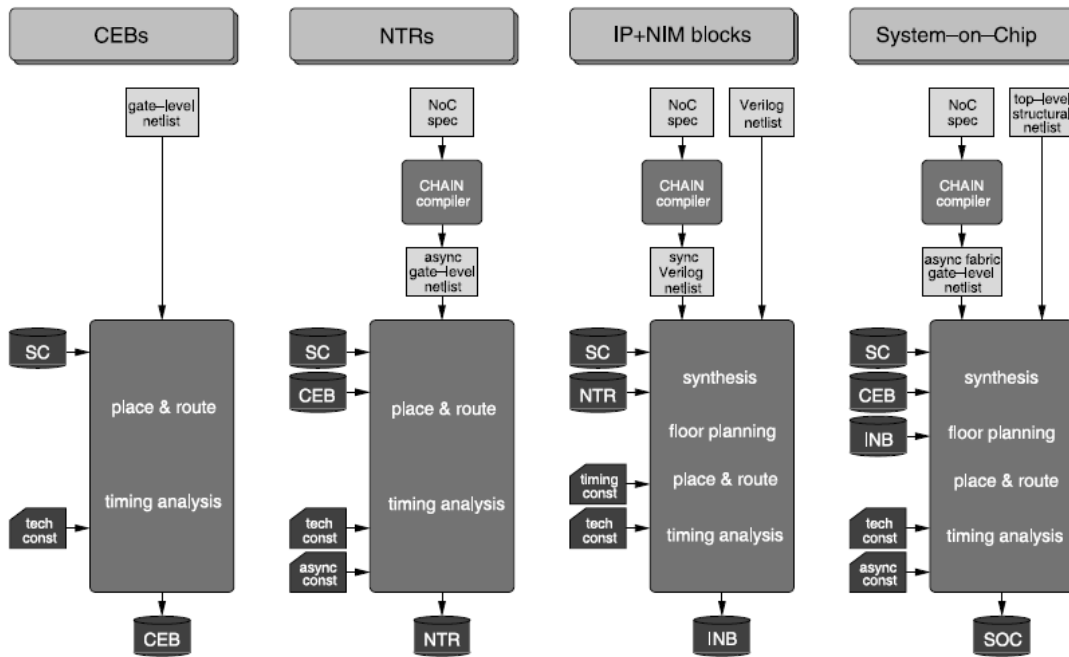


Figure 1.14. MPSOC Methodology Workflow of SpiNNaker [13]

## 1.5 Conclusion

With reference to the ITRS road map, the number of cores in the same chip will double each about 18 months to reach hundreds of cores by the end of the current decade (see Figure 1.1). Actually, many works have already reached few hundreds of cores in the same chip: for example the number of cores in [26] is equal to 762. With this evolution, classical interconnection methodologies like bus and point to point are not any more possible and other interconnect solutions like the Network On Chip have appeared to deal with those limitations.

We presented in this chapter a brief description of the evolution of the MPSOC during last decades. The evolution of the number of cores is exponential which means that the design complexity is also exponential. We presented the actual MPSOC implementations which have different network On Chip topologies starting from a simple bus connection to a free NoC topology. Designers used different methodologies to implement their own chips; we presented a set of workflows with real implementations.

## Chapter 2 : 2D MPSOC Design and implementation

## 2 . 2D MPSOC Design and implementation

Before moving to 3D IC implementation, we propose in this chapter to test and to verify our MPSOC designs with 2D implementation on FPGA. This approach will guarantee the good functionality of our chip which helps us to solve the MPSOC problems and to only focus on the specific 3D challenges. Thanks to the MPSOC state of the art performed in chapter 1, we choose to study the Butterfly architecture which is an interesting case study for EDA evaluation and of an eventual 3D IC design.

### 2.1 Theoretical Complexity Problems in 2D Design and implementation

To perform an efficient implementation of a 2D Design, we have to deal with the complexity of the workflow steps. We will present the complexity of the basic steps in 2D design implementation on FPGA which are the partitioning, the floorplanning and the place and route.

#### *The partitioning:*

When the Netlist of a component or a design can not fit on a single FPGA the step of partitioning on Multi-FPGA becomes necessary. This operation can be trivial with symmetric and homogenous MPSOC but it becomes a real challenge with asymmetric and heterogeneous architectures. The main goal of this operation is to find a methodology which minimizes the connections between the partitions. In the provided solution, each partition should meet all the design constraints (size, number of external connections...) and get a balanced distribution between the different groups. The problem of VLSI partitioning can be defined by a Hypergraph partitioning. Given a graph  $G=(V,E)$ , we can model the cells of the Netlist or the components by the set of the vertices  $V$  and the nets ( or interconnections) by the set of the edges  $E$ .  $K$ -way partitions can be defined by the division of the vertices into  $K$  groups. This problem is known to be NP-Hard that is why the proposed solutions are based on approximations and heuristic methods [27].

#### *Placement:*

The step of placement on FPGA is the determination of the cell's locations of the Netlist in order to optimize the area and the frequency. Given a Netlist of logic blocks, I/O pads and the set of interconnections between the cells, the output of the placement operation are the coordinates  $(x_i, y_i)$  for each block. This problem is known to be NP Hard which means that no polynomial algorithm is known to propose an exact solution to solve it [28].

**Routing:**

Routing is one of the major steps in 2D FPGA designs as it has an important impact on the performance of the circuit. The success of this operation is affected by the previous step which is the placement. The routing step has been also proved to be NP-hard [29].

The listed steps in the FPGA 2D design are classified in the family of the NP hard problems which means that the optimization in the EDA tools is still possible. In general the common tools used in the synthesis place and route for FPGA like ISE and EDK from Xilinx [30], perform each step sequentially which is not an efficient approach as there is a real dependency between the different operations. We propose in this chapter to study the implementation of an MPSOC with 16 masters and 16 slaves based on the Butterfly NoC topology. A real execution on FPGA will be performed.

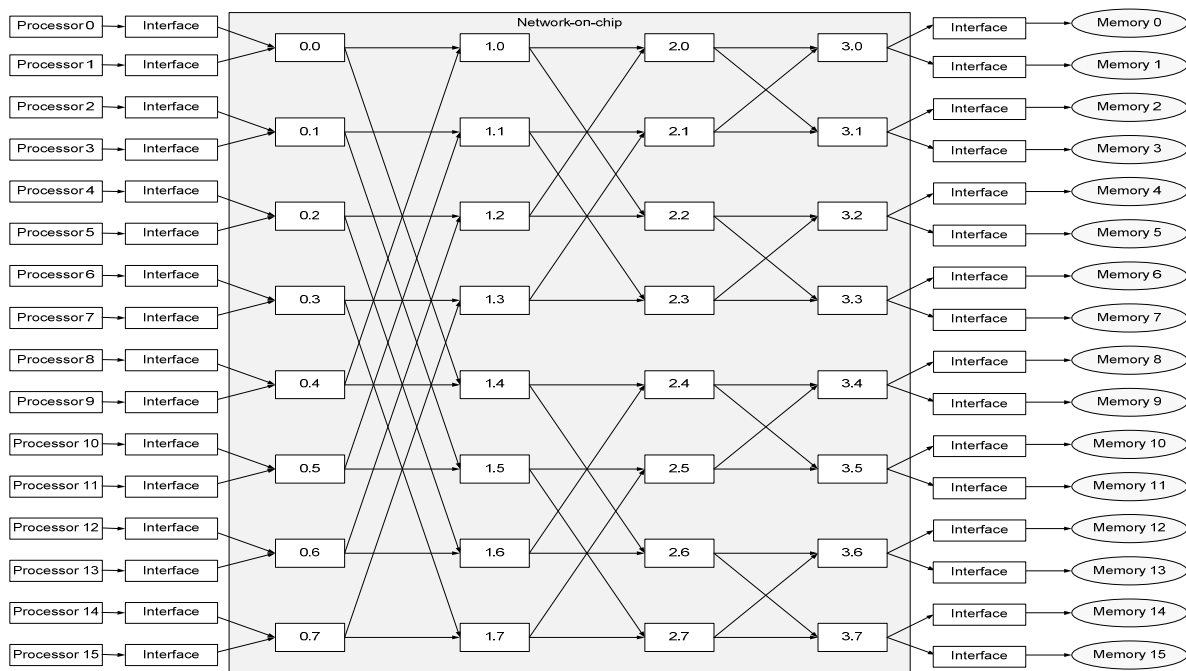
**2.2 Regular NoC implementation on FPGA: case study Butterfly**

The common NoC topology which is usually used in the literature and in the industrial products is the mesh topology. This one is famous thanks to its symmetric architecture and to the possibility to use a repetitive structure. A NoC with a mesh topology is relatively an easy case study for the EDA tools implementation. In fact, partitioning tools can find a symmetric structure even with a NoC with a very Large Scale MPSOC [26]. In this work [31], authors implemented an MPSOC with 2048 cores. We choose in this chapter, to study the butterfly topology to evaluate the limitation of EDA tools in order to solve the design step problematic. In Table 2.1, we compare the mesh and the butterfly topology. With its asymmetric architecture, the butterfly NoC represents an interesting case study to evaluate the portioning algorithm used in the EDA tools. This NoC is composed of long wires compared to the mesh topology with its equal short links. This characteristic can affect the maximum propagation delay time and the power consumption of the NoC. It can also represent difficulties for the automatic routing algorithms. In the mesh topology, links are bidirectional which is not the case for the butterfly case which needs a request and a response NoC.

**Table 2.1. Comparison Mesh and Butterfly topology**

NoC topology	Mesh	Butterfly
Symmetry	Symmetric	Asymmetric
Wires	Short	Long
Direction	Bidirectional	unidirectional

To study in deep the different characteristics of the butterfly architecture, we have designed a NoC with 16 masters and 16 slaves. A such NoC is called 2Ary-4Fly : the term 2Ary refers to the fact that all the used switches have a degree equal to 2, the term 4Fly refers to the number of stages needed which is equal to 4. The NoC's topology is presented in the Figure 2.1; Masters in the left are sharing 16 slave memories presented in the right part of the NoC.



**Figure 2.1. MPSOC based Butterfly NoC: 2Ary 4Fly Architecture**

The main component of the NoC is the switch, we use the switch library from the company Arteris [32]. This switch presents different options like arbitration, pipelines...In each router we have to modify the routing table according to the physical links and to the position of the switch in the NoC. We can verify from the NoC's architecture that all masters are connected to all the slaves through a single path. According to the routing table and to the message's address, the packets can be routed by the switch. For each switch, we have 2 input ports and 2 output ports. When the switch receives the packets, it can decide thanks to its routing table through which output port to send it. We present the routing tables of each switch in Table 2.2.

**Table 2.2 Address of the slaves**

Cores	Addresses	Cores	Addresses
<b>Slave0</b>	0x0	<b>Slave8</b>	0x800000
<b>Slave1</b>	0x100000	<b>Slave9</b>	0x900000
<b>Slave2</b>	0x200000	<b>Slave10</b>	0xA00000

<b>Slave3</b>	0x300000	<b>Slave11</b>	0xB00000
<b>Slave4</b>	0x400000	<b>Slave12</b>	0xC00000
<b>Slave5</b>	0x500000	<b>Slave13</b>	0xD00000
<b>Slave6</b>	0x600000	<b>Slave14</b>	0xE00000
<b>Slave7</b>	0x700000	<b>Slave15</b>	0xF00000

**Table 2.3. Different switch routing tables**

Switches	TX0	TX1
{0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7}	0x0	0x800000
	0x100000	0x900000
	0x200000	0xA00000
	0x300000	0xB00000
	0x400000	0xC00000
	0x500000	0xD00000
	0x600000	0xE00000
	0x700000	0xF00000
	{1.0,1.1,1.2,1.3}	0x0
0x100000		0x500000
0x200000		0x600000
0x300000		0x700000
{1.4,1.5,1.6,1.7}	0x800000	0xC00000
	0x900000	0xD00000
	0xA00000	0xE00000
	0xB00000	0xF00000
{2.0,2.1}	0x0	0x200000
	0x100000	0x300000
{2.2,2.3}	0x400000	0x600000
	0x500000	0x700000
{2.4,2.5}	0x800000	0xA00000
	0x900000	0xB00000
{2.6,2.7}	0xC00000	0xE00000
	0xD00000	0xF00000
3.0	0x0	0x100000
3.1	0x200000	0x300000
3.2	0x400000	0x500000
3.3	0x600000	0x700000
3.4	0x800000	0x900000

3.5	0xA00000	0xB00000
3.6	0xC00000	0xD00000
3.7	0xE00000	0xF00000

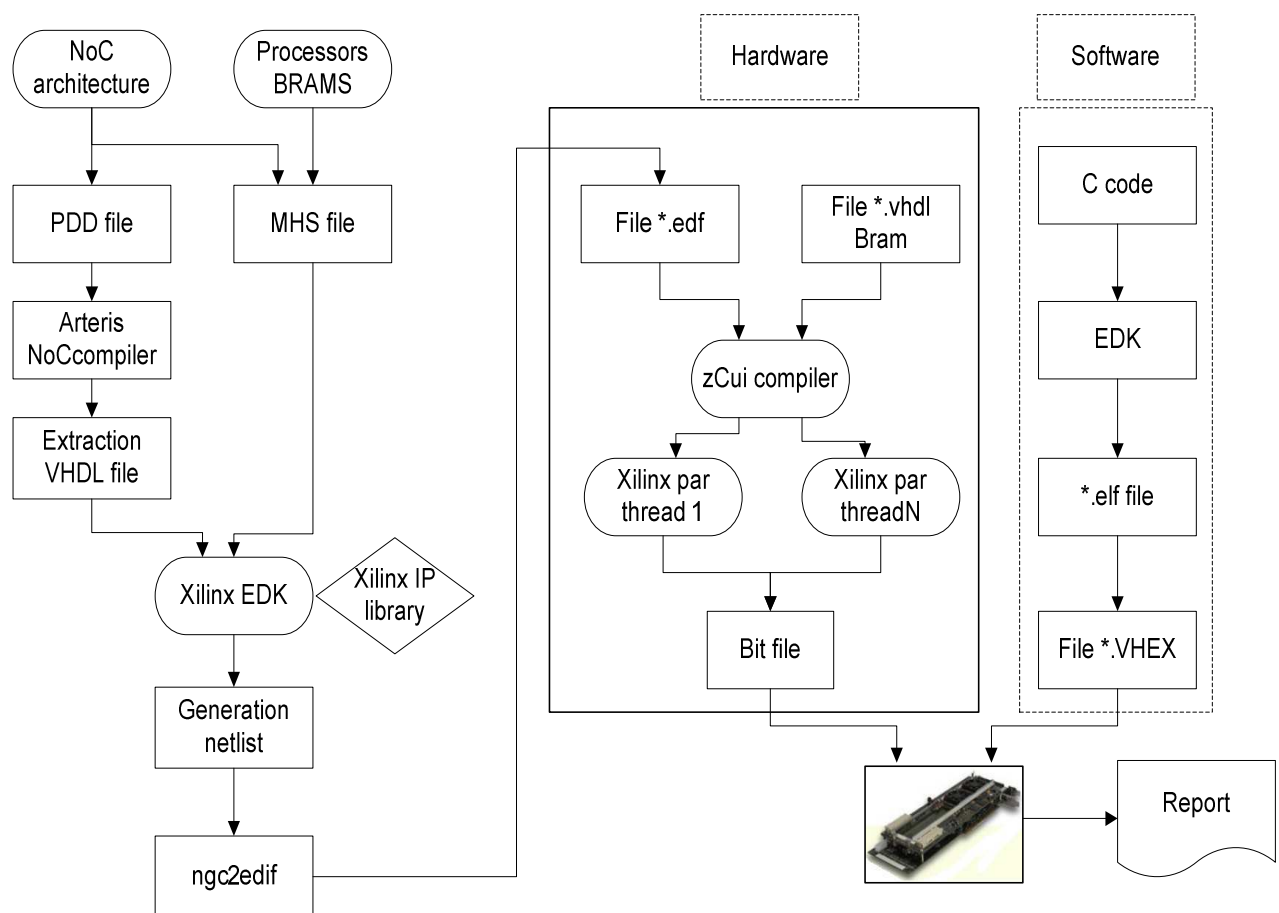
We add our NoC as an IP in the EDK project to connect 16 Microblaze processors which are the masters to sixteen block memories representing the 16 slaves. To evaluate the execution time of the design we add a timer which is connected to an OPB (On-Chip Peripheral Bus). We start by the hardware part of the design. After we generate the Netlist of the project, we use the command *ngc2edif* to transform the Netlist to edif files used by zCui. This one is a compiler tool designed by EVE company [33] to synthesis the DUT (Design Under Test) by calling Xilinx tools. At the end, it generates a bit file and several reports describing the FPGA resources used after the place and route on the board. For each Microblaze, we create a software application in the EDK project. In each application we add a C code and header files to describe the tasks to be performed by the processor. Once we define a software for each processor, EDK calls the C code compiler and generates \*.elf files. We transform these files to \*.vhex files which can be run on the board. At this level we have now hardware and software files ready to be run on the zebu board. We call the command zRun to make the execution on the board. Eve tool offers the possibility to use static and dynamic probes. In our case, we use a static probe to capture the end of the execution time. In fact, the run will stop when the condition of stop which is related to the probe value will be true. Then we can read the execution time needed for the program to be complete. A summary of this workflow is presented in Figure 2.2

### 2.2.1 Synthesis results

ZCui compiler makes the synthesis using Xilinx's tool. As Zebu-UF4 board has 4 FPGAs virtex 4 LX 200, zCui tool gives the possibility to make the mapping on one or more FPGAs. In this work we choose to make the synthesis on only one FPGA. Table 2.4 presents the resource utilization when we implement the project on the FPGA board. We note that this design presented in Figure 2.1 reaches 100% of DSP48 available on the board, 81% of BRAMs and 60% of slices. In this board, we have 96 slices which are totally used. Number of BRAMS in the FPGA board is equal to 336. Each block is an 18 kb memory. The need of this project in term of BRAM is almost equal to 5 Mbytes.

**Table 2.4. Resource utilization of Zebu-UF4**

	Resources utilization in %		
	BRAM	Slices	DSP48
<b>2Ary-4Fly</b>	81	60	100



**Figure 2.2. Our MPSOC implementation workflow**



### 2.2.2 Parallel Programming: Filter Harris

Thanks to this MPSOC architecture, we are now able to perform a parallel programming using 16 processors and 16 slaves. In our case, we use this hardware design to apply a parallel image application which is the Harris Filter in order to evaluate its speed up.

#### Principle

Harris filter is a corner detector algorithm [34]. Corner detection is an approach used in image analyses especially for object tracking. For an original image I we calculate the corner curvature K defined as:

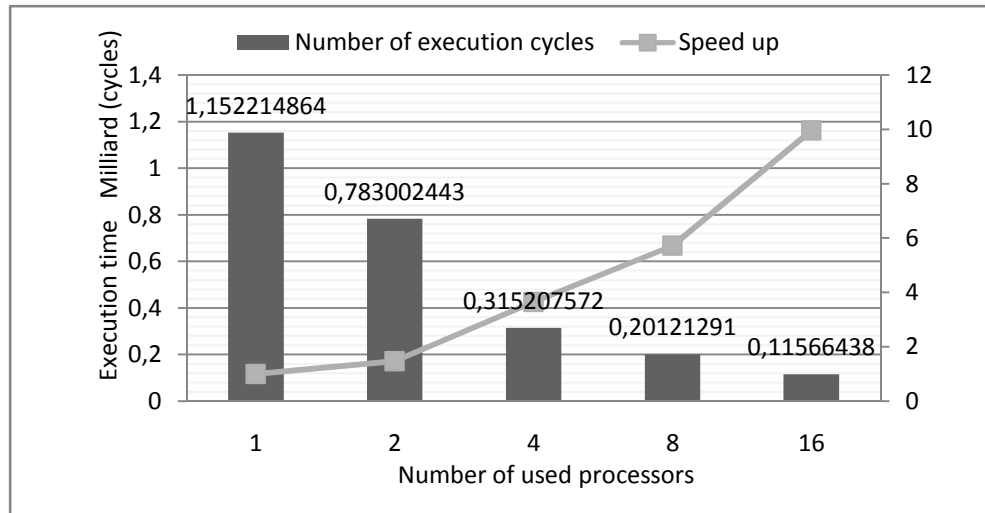
$$k = \det(A) - \lambda * \text{trac}(A)^2$$

A is the Harris matrix,  $I_x$  and  $I_y$  are the derivations of the image intensity I successively with the axes X and Y.  $\lambda$  is a parameter of corner detection tolerance which is usually used between 0.05 and 0.15. Local maxima of K determine the location of interest points.

$$A = \begin{pmatrix} \text{Mean}(I_x)^2 & \text{Mean}(I_x * I_y) \\ \text{Mean}(I_x * I_y) & \text{Mean}(I_y)^2 \end{pmatrix}$$

#### Parallel Programming

The Harris filter is written in C code associated to each Microblaze processor. In this application, we compute the Harris filter of a grey image of 256x256. Initial image will be stored in the shared memory BRAM0. At the beginning of the algorithm, Microblaze 0 stores the image I in the shared Memory then gives the order to all the Microblazes to start the task1. Each Microblaze i will access the memory and calculate the values  $I_x$  and  $I_y$  of the pixels composing a number of rows and then stores the result in BRAM1. When a Microblaze finishes task 1, it changes the value of the flag of end of task 1. The Microblaze 0 finishes its task and waits the other processors. When the task one is finished, Microblaze 0 gives the permission to start the task2. Each MBI will access  $d_x$  and  $d_y$  memories and compute the value of the Gaussian filter of each pixel as an estimation of  $\text{Mean}(I_x)$  and  $\text{Mean}(I_y)$ . The last step of task2 is to compute the curvature k. A comparison of the value k with a threshold value will decide if the point is a corner or not. Each processor i computes the value of the matrix A for each pixel of a specific number of rows. This number is equal to the total rows of the image divided by the number of processors.



**Figure 2.3. Harris Filter Execution Time and speed up**

### Implementation Results

To evaluate the speed up of our parallel program we change the number of active processors. We present in Figure 2.3 the execution time of the algorithm with different NoC sizes. Harris filter uses in several steps the matrix multiplication and convolution, which is an independent task suitable for parallelization. In the Figure 2.3, we notice the efficiency of our parallel program. In fact, the speed up is equal to 3,65 with 4 processors to reach 9,96 for 16MB. The use of 16 processors reduces the time to 10% of the time needed in the sequential program using one processor. Our results confirm theoretical results in [35], with a real implementation of a NoC with different sizes. In fact, the speed up increases when the size of the NoC increases (see Figure 2.3) but the curve of the speed up has a non linear form. This is due to the fact that, when the number of cores increases, the time needed for synchronization becomes considerable. In such case, the access to the shared memories is a difficult operation. To deal with this problematic we use a home developed function called *Read\_exclusive*. Thanks to it, only one processor can access a memory at a time. When a processor is writing or reading from a memory, the other masters must wait until the end of the operation that is why it is called Exclusive.

## 2.3 NoC Design Space exploration on FPGA

We propose in this section to perform a design Space exploration of our MPSOC design presented in section 2.2. The goal of this analysis is to study the effect of the different parameters on the industrial tools used in FPGA workflow presented in Figure 2.2. We propose to explore the different NoC and processor configurations in order to get the execution time of each used tool in this workflow. We use a multi-objective genetic algorithms looking for the set of combinations which minimize the number of cycles, the used Bram and the number of slices.

### 2.3.1 ModeFRONTIER tool

In this section we use ModeFRONTIER[36] to perform our optimization. This tool is commercial software provided to solve multi-objective problems and to allow easy coupling to almost the most used Computer Aided Engineering tools. ModeFRONTIER has a set of specific optimization algorithms. We can use deterministic optimizers specific for single objective problems like SIMPLEX. The designer can also use different multi-objective optimizers like MOGA-II and NSGA-II. This tool is suitable to solve real constrained and multi-objective problems like scheduling and resource optimization.

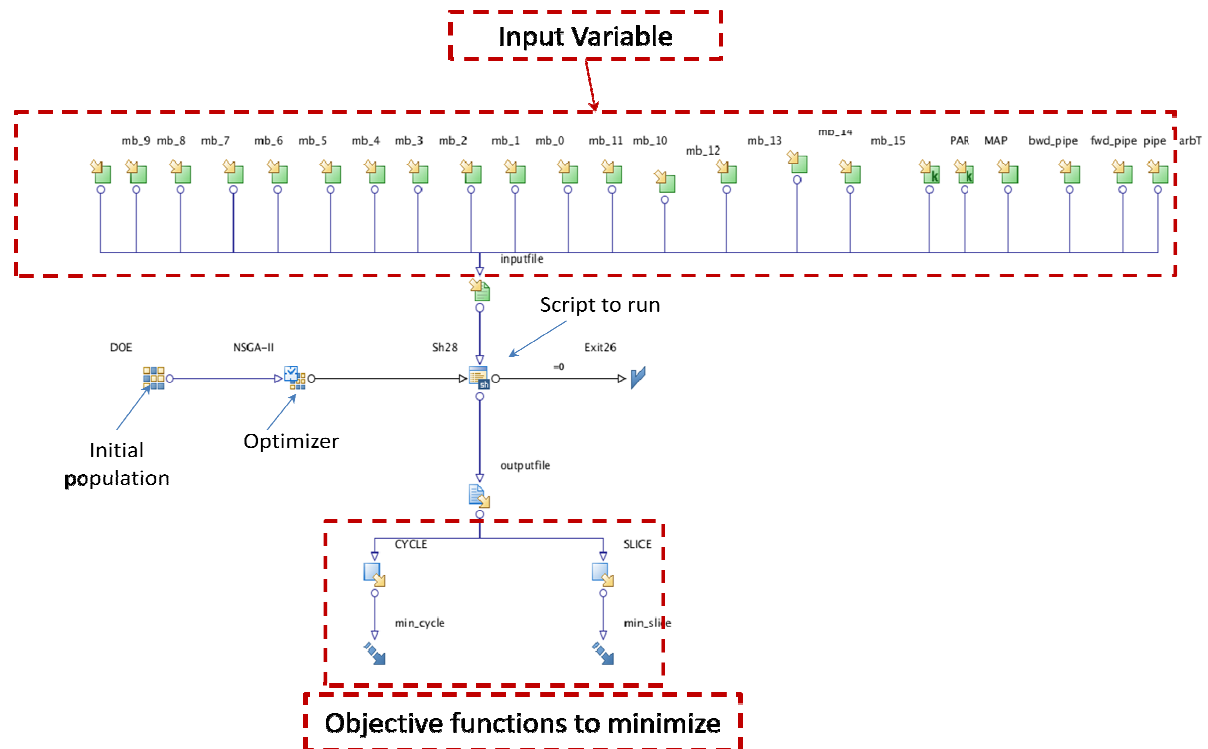


Figure 2.4. ModeFRONTIER project example

We present in Figure 2.4, a model of a ModeFRONTIER project with its basic components:

- **Input Variables:** The user should define at the beginning, the input variables and their properties. These variables define the design space exploration and will be affected to an input file. They can be real, integer and binary. A new input file is generated during each iteration taking in consideration the new values of the variables.
- **Initial population:** The user should define the initial population. This choice depends on the problem's properties and affects the final result. ModeFRONITER offers the possibility to choose between a set of initial population which are summarized in Figure 2.5. The initial population can be generated using one of the presented algorithms or predefined by the user.



Figure 2.5. ModeFRONTIER initial population

- **Optimizer:** The tool offers different families of optimizers: Basic like MOGA-II, advanced like NSGA-II and quadratic like NLPQLP. The choice of the adequate optimizer is essential to get an optimized solution. We present the different algorithms provided by ModeFRONTIER.

**Schedulers**

- **DOE** : Sequence
- **MACK** : Multivariate Adaptive Crossvalidating Kriging

**Basic Optimizers**

- **SIMPLEX** : Single-objective derivative-free optimizer
- **B-BFGS** : Single objective Bounded BFGS algorithm
- **Levenberg-Marquardt**
- **MOGAI** : Multi Objective Genetic Algorithm
- **ARMOGA** : Adaptive Range MOGA

**Advanced Schedulers**

- **MOSA** : Multi Objective Simulated Annealing Algorithm
- **NSGA-II**: Non-dominated Sorting Genetic algorithm
- **MOGT** : Game Theory coupled with Simplex algorithm
- **F-MOGAI** : Fast Multi Objective Genetic Algorithm
- **MOPSO** : Multi Objective Particle Swarm Optimizer
- **F-SIMPLEX** : Fast Single-objective derivative-free optimizer

**Evolution Strategy Schedulers**

- **1P1-ES**
- **DES** : Derandomised Evolution Strategy
- **MMES** : Multi-membered evolution strategy

**Sequential Quadratic programming**

- **NLPQLP** : Robust implementation of a sequential quadratic programming algorithm.
- **NLPQLP-NBI** : Multi-objective scheduler based on the
- **The program**: we should define the algorithm to evaluate during the exploration. ModeFRONTIER offers the possibility to include different type of languages like Bash script. We can also use an executable file or other industrial tools like Matlab. Taking the variable from the input files, this program provides the value of the output variables which can be the objective functions or the design constraints.

- **The Objective Functions:** We define the objective functions of the project using the objective nodes provided by ModeFRONTIER. We can choose to minimize or to maximize our objectives.

### 2.3.2 Multi objective Genetic Algorithm NSGA-II Algorithm

Multi-objective evolutionary algorithms (MOEA) have been considered as a successful solution to optimize the problems with multiple conflicting objectives. The Multi objective problems are different from the single objective function. In the first one, the solution of the single objective function is a single optimized point while it can be set of points in the case of the multi objective formulation. The dependency between the objectives can affect the final result. A classical formulation of the minimization of multi-objective problem with  $m$  decision variables and  $n$  objectives is:

$$\text{Minimize } y = f(x) = (f_1, \dots, f_n)$$

$$\text{subject to } Ci_x \leq 0, i = 1, 2, \dots, p$$

$$Ce_x = 0, e = 1, 2, \dots, q$$

The result of an optimization problem depends on different factors like the choice of the optimizer and its parameters. As we are interested to solve multi objective problems using MOEA, we choose to use the NSGA-II algorithm. This one proved its efficiency compared to other algorithms especially with Multi objective problems. The NSGA-II is a fast elitist Multi objective Non dominated Sorting Genetic algorithm suggested by K. Deb et al [37]. The NSGA-II algorithm is based on the sorting of the different populations with respect to all the functions. An individual  $p$  in the population  $P_t$  dominates an individual  $q$  if  $p$  dominates  $q$  in all the objective functions. If for only one function,  $q$  is not dominated by  $p$ , the domination of  $p$  to  $q$  is not any more valid. Starting from this idea the set of the individuals in the population which are not dominated by any ID form the first front  $F_1$  of the non dominated Ids. The set of individuals included in the first front  $F_1$  are deleted from the population and the computation of the new front  $F_2$  is performed using the same method. This operation is repeated until all the individuals of the population  $P$  are affected to a front. To create the new generation  $P_{t+1}$ , the algorithms inserts the front with the increasing order until the number of population is reached. When the insertion of the last front overcomes the number population size an

algorithm based on the crowding distance is used to choose the number of needed individuals  
Figure 2.6.

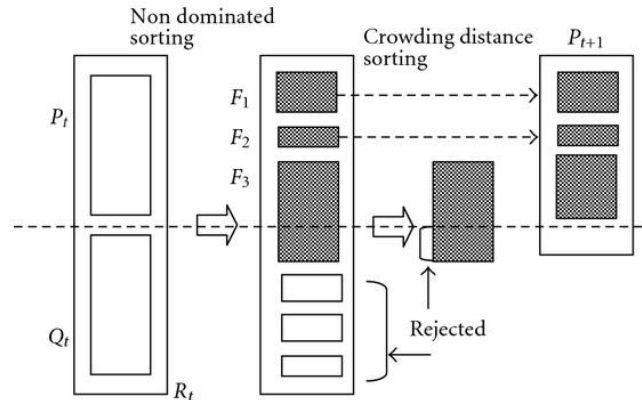


Figure 2.6. NSGA-II algorithm illustration

### 2.3.3 ModeFRONTIER project: MOEA on FPGA

We have already presented in section 2.2, the design of a 16x16 MPSOC and the execution of a parallel Harris filter on FPGA. During this step, different choices should be performed to define the different options of the processor and of the NoC. The Microblaze processor can be used with full, basic or medium options. The user should find the needed combination depending on his application. We propose in this section to study the effect of the variation of these options on the area and on the execution time of the algorithm. In order to get the optimized solution, we perform in this section a multi objective evolutionary algorithm.

We present in Figure 2.7 the project using the Tool ModeFRONTIER which is specific for the optimization and the exploration. We propose to explore the different processors and NoC options of our designed MPSOC already presented in section 2.2 . As we affect the same configuration to all the processors, the number of possible individuals is equal to  $2^6 \times 2^3 \times 3$ . We use the evolutionary Genetic Algorithm NSGA-II to find the optimized combination. We present in Table 2.5 and Table 2.6 the input variables of our project. We use the Microblaze processor from Xilinx presented in Figure 2.4. For each individual the execution of the Harris Filter Algorithm on FPGA is performed. The outputs of this exploration are the values of the number of cycles and the number of slices to be minimized.

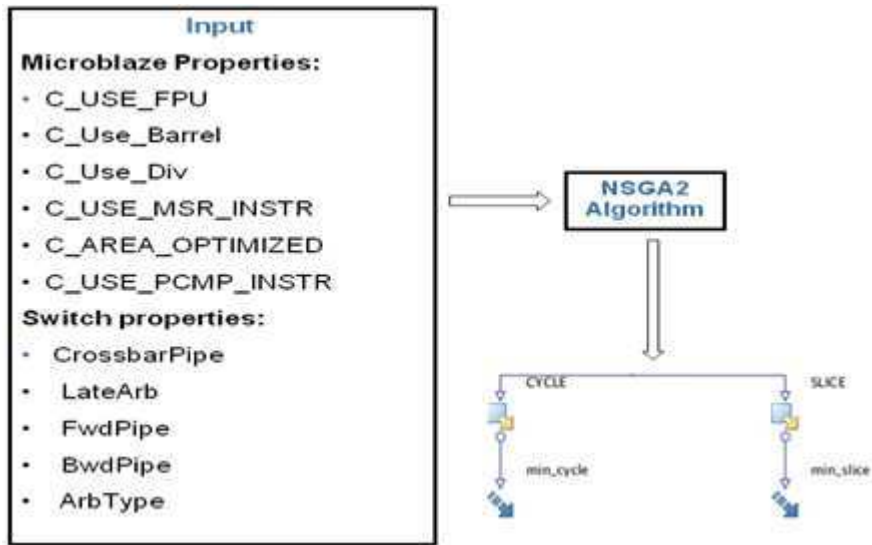


Figure 2.7. ModeFRONTIER DSE Project

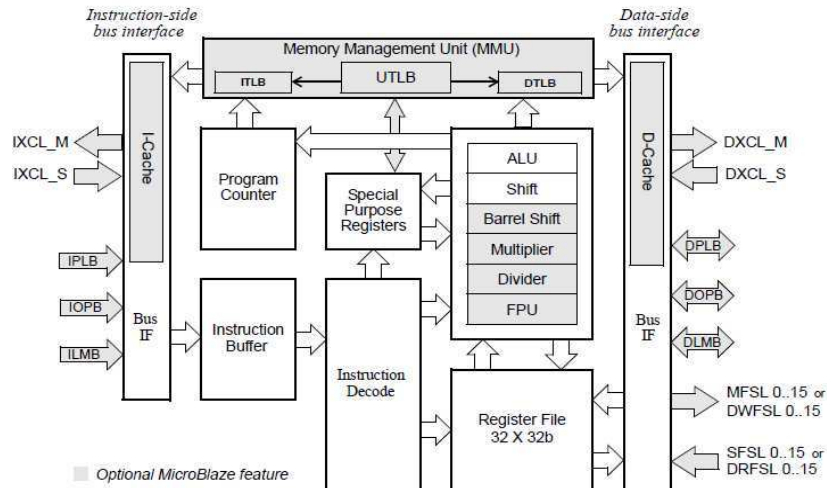


Figure 19. Microblaze Architecture [30]

Table 2.5. Microblaze parameters

Microblaze parameters	Definition	Values
<b>C_AREA_OPTIMIZED</b>	Optimize Area	False(0), True(1)
<b>C_USE_BARREL</b>	Use Barrel shifter	False(0), True(1)
<b>C_USE_MSR_INSTR</b>	Use Msrset and Msrclr instructions	False(0), True(1)
<b>C_USE_PCMP_INSTR</b>	Use Pattern Compare instruction	False(0), True(1)
<b>C_USE_DIV</b>	Use divider	False(0), True(1)
<b>C_USE_FPU</b>	Use FPU	False(0), True(1)



**Table 2.6. Switch Options**

Switch Options	Values
Arbitration Type	Round-Robin(0), Rotate(1), Fifo(2)
Input pipeline register	True(1), False(0)
Forwards pipeline register	True(1), False(0)
Backwards pipeline register	True(1), False(0)

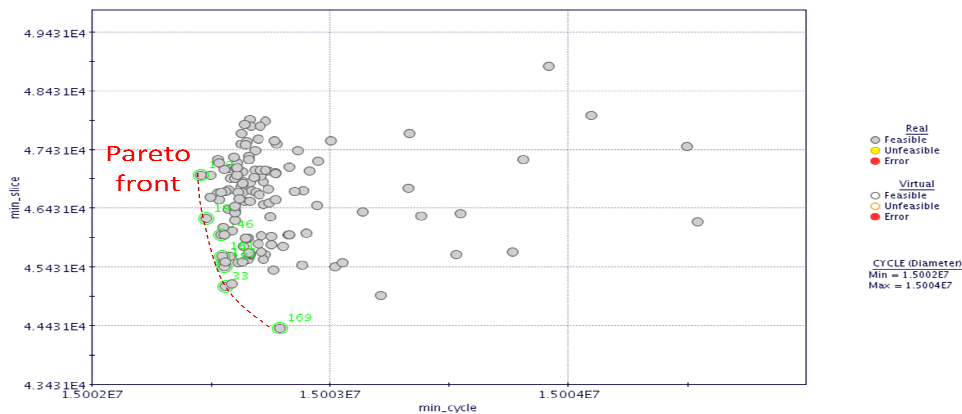
We present the different options of the NSGA-II algorithm:

#### NSGA II Parameters:

- Number of generations : 100
- Crossover Probability : 0.9
- Mutation Probability for real coded vectors :  $1/n$  ;  $n$  : number of decision variable in real code
- Mutation Probability for BinaryStrings:  $1/l$  ;  $l$  : is the string length for binary coded

#### Advanced Parameters:

- Distribution index for Real-Coded Crossover : 20
- Distribution index for Real-Coded Mutation : 20
- Crossover Type for binary-coded variables : simple
- Random Generator Seed : 1



**Figure 2.8. Pareto Front DSE of Harris Filter on MPSOC 16x16**

We present in Figure 2.8, the obtained result of the Evolutionary Genetic Algorithm applied on FPGA. The exploration of the different options of the processors and the NoC affects the execution time of the Harris Filter Algorithm. With powerful processors, the execution time is faster and the number of used slices is higher. Starting from a random population and after many generations, the GA converges into the set of solutions optimized

for both objective functions. The output result of our Genetic Algorithm is a set of points known by the name of the Pareto Front presented with green points in the previous figure. The user can choose the appropriate configuration depending on his needs in term of time and resources. The implementation of our design on FPGA, was a fast way to explore our hardware design. The needed time to perform each step of the workflow depends on the performance of the machines but it also depends on the design's options. We will present in the next section a design space exploration of our design on different machines.

### 2.3.4 Machines Specifications

We present in this section the different specifications of the three used machines during the sequential and the parallel exploration. We specify their software and their hardware specifications.


**Table 2.7. The properties of the machines**

MPSOC7 SPEC			
Hardware Summary		Software Ssummary	
<b>Type of System</b>	Homogeneous	<b>gcc</b>	4.1.2-48
<b>Compute Node</b>	MPSOC7	<b>Linux</b>	Red Hat 5
<b>Total Chips</b>		<b>ModeFRONTIER</b>	modeFRONTIER 4.2.0 b20091201
<b>Total Cores</b>	4	<b>FlexNoC</b>	VFC 2.2
<b>Total Threads</b>	8	<b>EDK</b>	Edk 9.2.02i
<b>Total Memory</b>	12 GB	<b>ISE</b>	Ise 9.2.04i
<b>Memory Type</b>	DDR3-800/1066/1333	<b>zCui</b>	Version 4.3.3B.00

MPSOC4 SPEC			
Hardware Summary		Software Ssummary	
<b>Type of System</b>	Homogeneous	<b>gcc</b>	4.1.2
<b>Compute Node</b>	MPSOC4	<b>Linux</b>	Red Hat 5
<b>Total Chips</b>	1	<b>ModeFRONTIER</b>	ModeFRONTIER 4.2.0 b20091201
<b>Total Cores</b>	1	<b>FlexNoC</b>	VFC 2.2
<b>Total Threads</b>	2	<b>EDK</b>	Edk 9.2.02i
<b>Total Memory</b>	8 GB	<b>ISE</b>	Ise 9.2.04i
<b>Memory Type</b>		<b>zCui</b>	Version 4.3.3B.00

THALES2 SPEC			
Hardware Summary		Software Summary	
Type of System	Homogeneous	gcc	3.4.6
Compute Node	Thales2	Linux	Red Hat 4
Total Chips	2	ModeFRONTIER	modeFRONTIER 4.2.0 b20091201
Total Cores	4	FlexNoC	VFC 2.2
Total Threads	4	EDK	edk 9.2.02i
Total Memory	8 GB	ISE	Ise 9.2.04i
Memory Type	GB DDR2 RAM ECC	zCui	Version 4.3.3B.00

Table 2.8. ZEBU UF4 Emulator specifications

Emulator platform : ZEBU UF4					
Modules	Descriptions	Design 1	SSM IP	12MBs ,8 Brams	
FPGA	4 Virtex-4 LX200	Resources utilizations			
DRAM	512 MBytes	Bram	Slices	DSP48	
SSRAM	64 MBytes	62	66	54	
ICE	Smart and Direct				

### 2.3.5 Sequential DSE

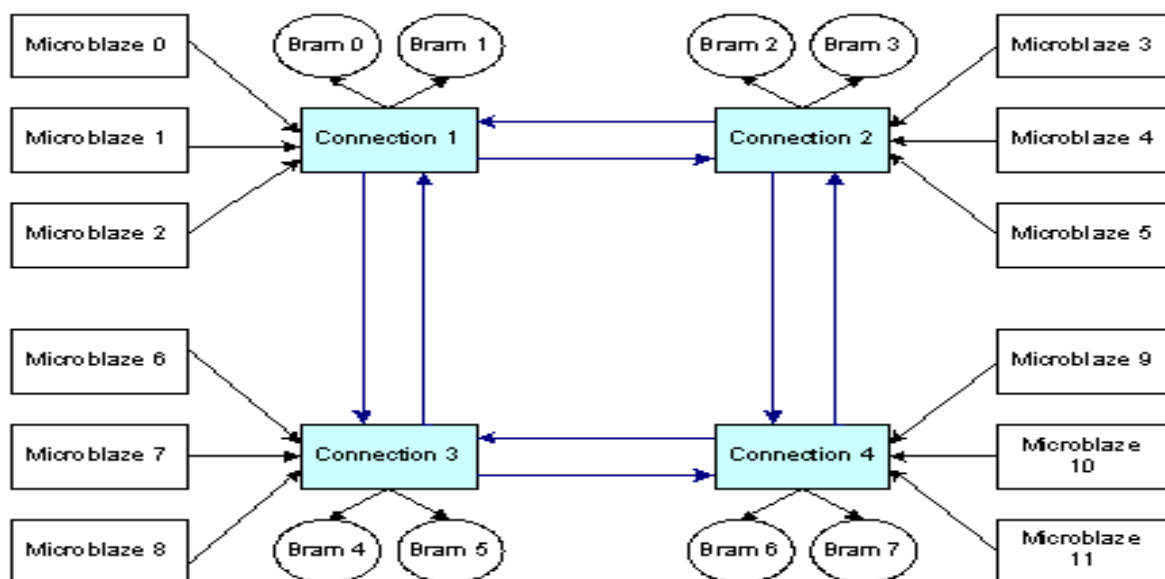
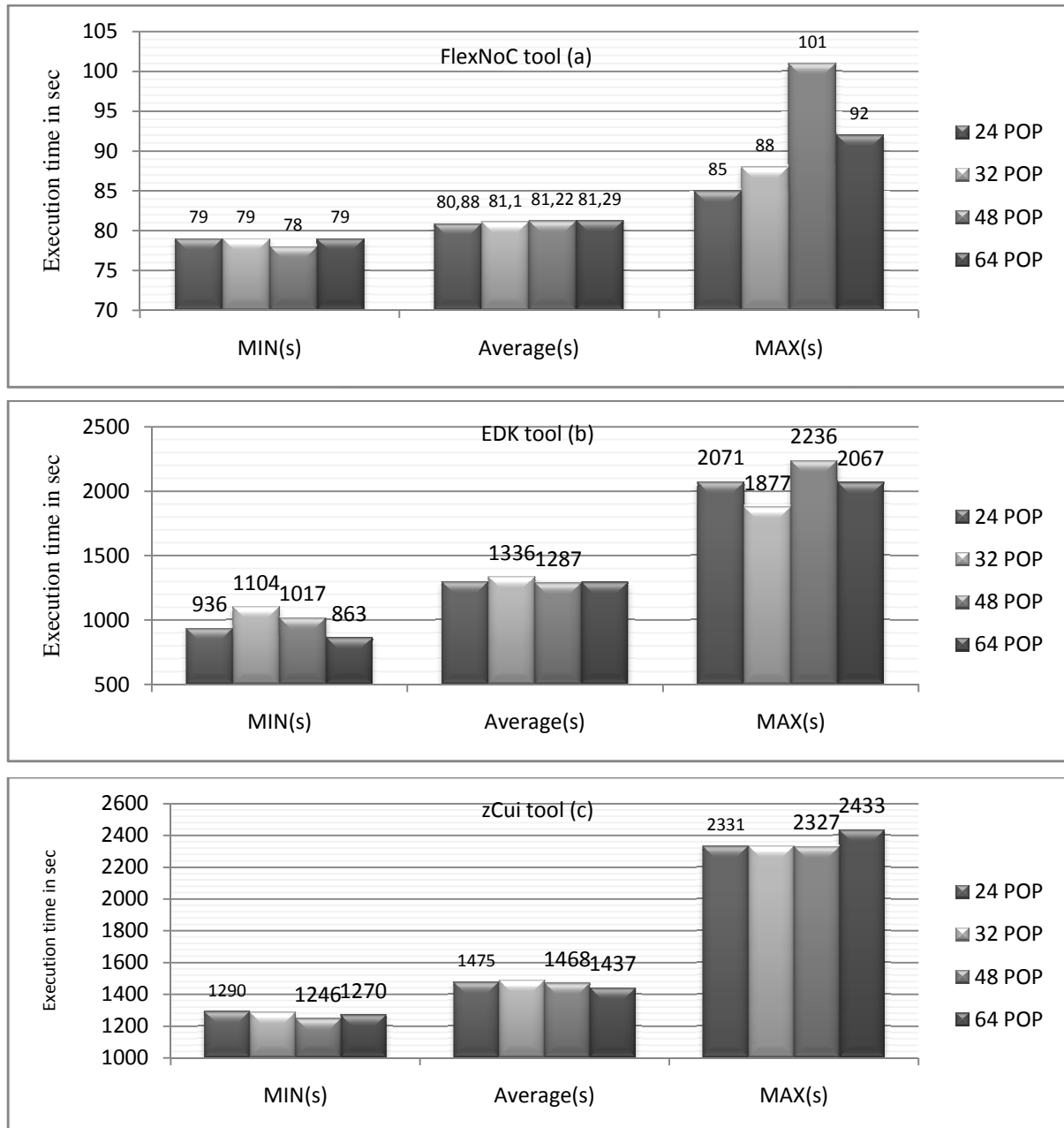


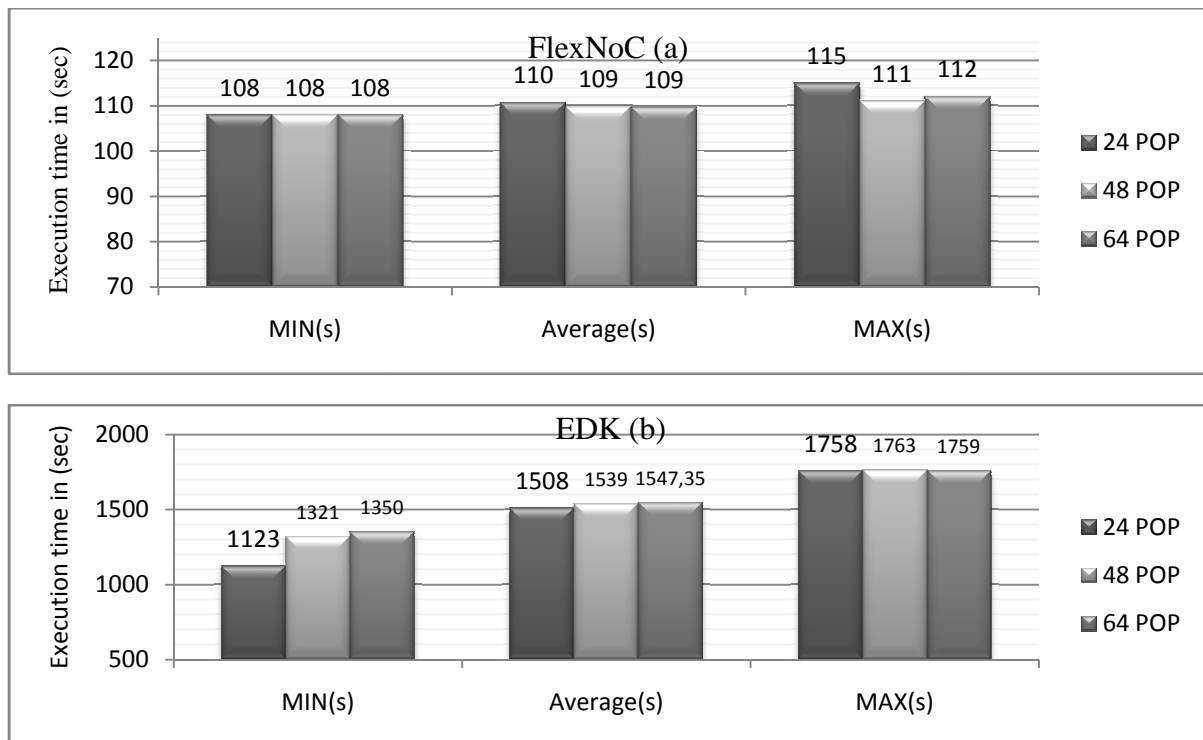
Figure 2.9. SSM IP Architecture

When we have used the GA during our work, we remarked that the choice of the different parameters like the number of individuals and the number of generations is usually not justified. That is why we choose to study the effect of the variation of some of those options using different machines. For this, we implement a Mesh MPSOC topology SSM –IP presented in Figure 2.9. We apply the same 2D FPGA workflow already presented in Figure 1.10 and we study the variation of the execution time needed by the 3 basic tools: FlexNoC, EDK and zCui. We present the results in Figure 2.10.



**Figure 2.10. MPSOC7: Execution time variation in function of the population size: FlexNoC (a), EDK (b), zCui (c)**

The number of individuals in the same population is an important parameter for the NSGA-II algorithm used in ModeFRONTIER tool to perform the design space exploration. We present the variation of the execution time of the different tools on the machine MPSOC7 presented in Table 2.7. We notice that the average time values presented in the Figure 2.10, are almost the same for different population sizes, we can conclude that the average time does not depend on the population size property. We notice that for the three tools the difference between the Min average value and the Max Average value is considerable: about 20% for FlexNoC and 50% for EDK and zCui. This variation is due to the difference between the individuals in term of options. EDK and zCui tools are sensitive to the variation of the hardware options. In fact they perform the steps of synthesis, place and route. The complexity of those steps is directly related to the complexity of the hardware components. For example with a hardware design using a full processor, the number of logic cells is high and the synthesis, the place and route operations are harder to perform. The variation of the execution time when the population size changes is also due to the diversity of the population which has the probability to increase when the population size increases.



**Figure 2.11. THALES2 : Comparison of FlexNoC (a) EDK (b) execution with different population sizes**

We present the results of the same exploration on the machine THALES 2 (Table 2.7). We notice that the average execution time for all the tools is almost the same when the population size changes. The value of the distance (difference between the Min and the Max

values) is most important for the smallest population size. It is clear that the difference between the maximum and the minimum values is meaningful for EDK tool. This result is similar to the one already presented for the machine THALES 2. The variation of the design properties has an important effect on the steps of place and route. In fact the complexity of those operations depends directly on the complexity of the design.

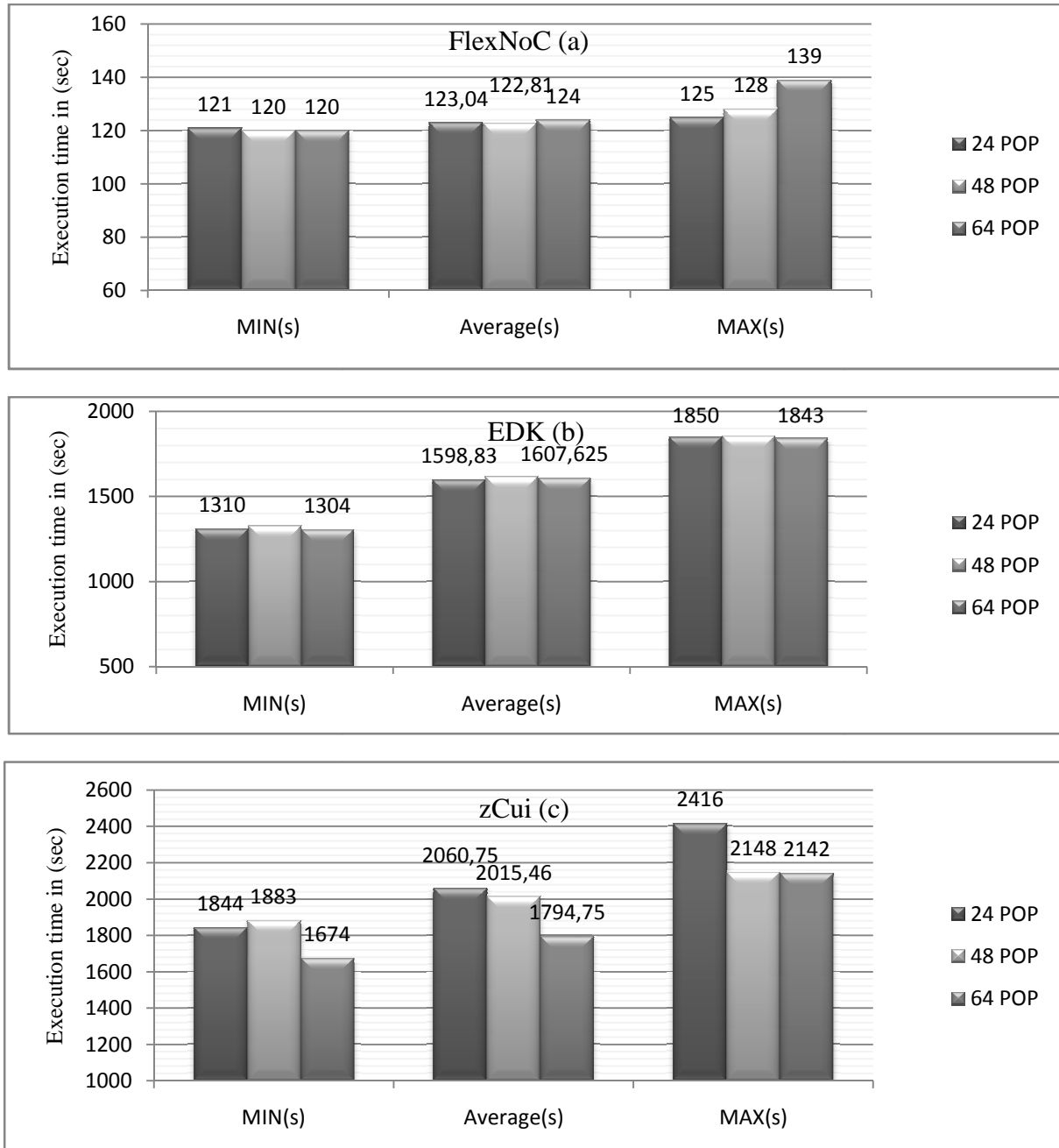


Figure 2.12. MPSOC4: Comparison of FlexNoC (a), EDK (b), zCui (c) with different population sizes

We present in Figure 2.12 the DSE results on the machine MPSOC 4 presented in Table 2.7. The average execution time is almost invariant when the population size changes for the tools FlexNoC and EDK. This average time is low when the population size is high. The design space exploration of our MPSOC on different machines gives a better idea about the effect of the population size on the MOEA. In fact, the average time is almost the same for all the tools on the different machines. The value of the distance is meaningful with the tools (EDK, zCui). In fact, the place and route steps take more time to be performed when the design is complex. With full version hardware (processor full options, NoC full options) the number of used slices and DSP are multiplied.

If we compare now the average execution time to extract the NoC by the tool FlexNoC using the 3 machines (Figure 2.10, Figure 2.11 Figure 2.12), the value is about 120 sec for THALES 2 and MPSOC 4 while it is about 80 sec for the machine MPSOC7. With 8 threads and 12G memory, this machine is the most powerful one. The number of threads is a major factor affecting the performance of the machine. This result is also the same for EDK and zCui tools. The machine MPSOC7 can perform all the workflow faster than the other machines. The industrial used tools offer the possibility to use all the processors of the machine which is a way to perform a parallel execution.

## 2.4 Parallel and multi-scale software implementation

With the shortage of information about the different parameters during the step of design, we propose in this section to study the effect of different variables of the design. The real goal of this work is to create a data base of designs which can be the input of a mathematical predictive model. We present in Figure 2.13 the main idea of this work. The different input variables of our work are:

- Parameter 0: NSGA-II options
- Parameter 1 : FPGA platform
- Parameter 2 : parallel flow
- Parameter 3 : the properties of the machines

These different parameters can be the input to create a mathematical model to predict the behavior of the workflow for a specific design. This step is a learning step in the predictive model.

At the end of this work and thanks to the obtained model, we can advice the user referring to his personnel constraints (number of licenses, maximum time...) about the way to schedule his DSE. This work can be used by the EDA companies as a consulting service to get an optimized utilization of their tools.

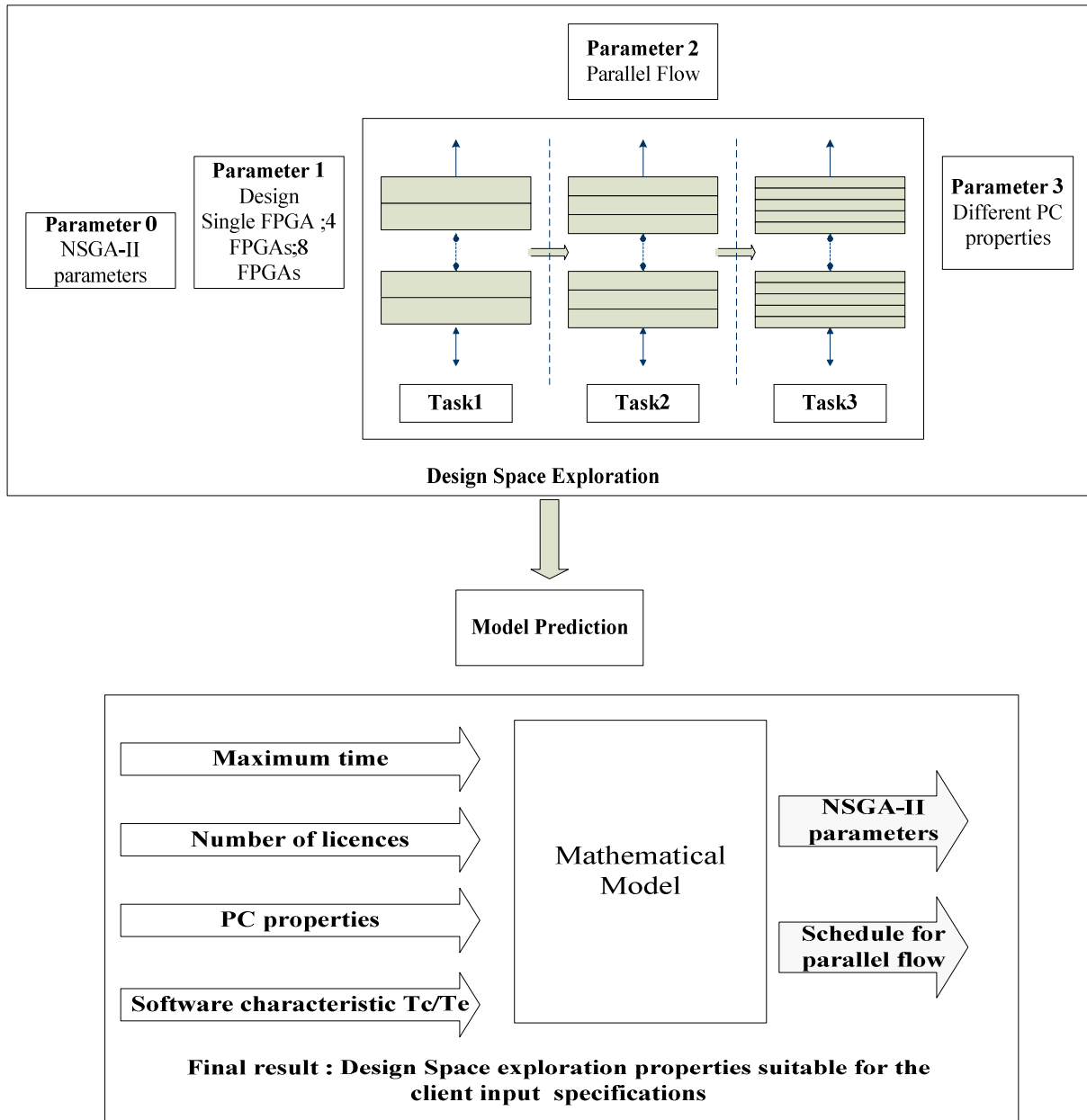
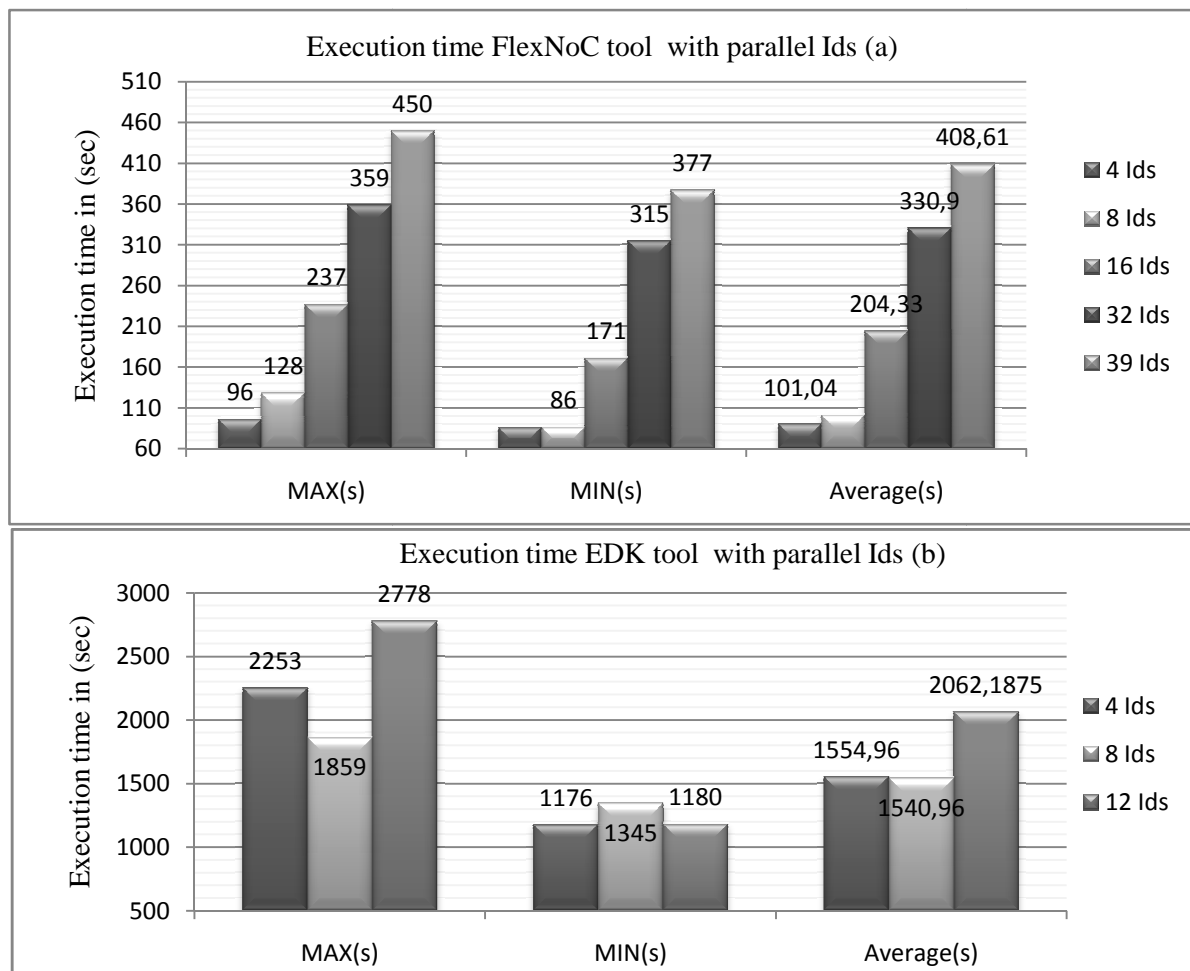


Figure 2.13. Design Space Exploration for mathematical Model generation

We performed at the previous section a study of the population size which is a basic parameter in the NSGA-II algorithm. We have also presented the effect of the different machines on the design workflow. We propose now to change the number of parallel used threads. In fact, we can run the workflow at the same time on the same machine thanks to the



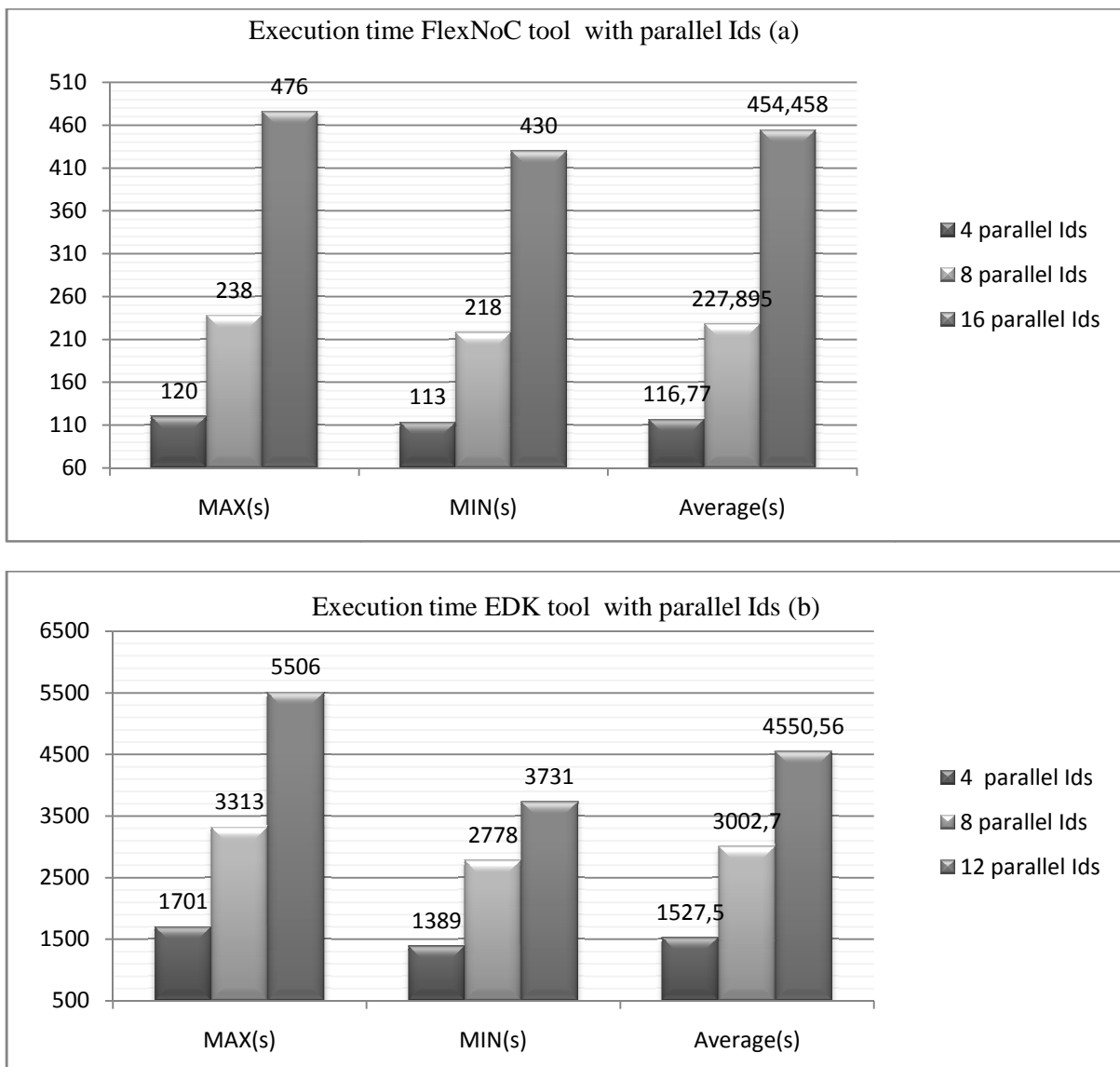
option of parallel IDs run offered by ModeFRONTIER tool. We present the results of this Design Space Exploration in Figure 2.14 and Figure 2.15. Thanks to the high performance of the Machine MPSOC 7, we can run until 39 IDs at the same time, a problem of out of memory is faced with higher number. The parallel execution increases the needed time to extract the NoC files by individual. Until 8 parallel IDs the execution time is not really affected compared to the sequential treatment (see Figure 2.10 ), but it is almost the double when the number of parallel IDs reaches 16. There is a challenge between the number of parallel threads and the whole exploration time. As the EDK tool is hungry in term of memory, we can only reach 12 parallel IDs at the same time. There is no real difference between running 4 or 8 parallel threads, but the real delay is reached when the number of IDs is equal to 16.



**Figure 2.14. MPSOC7 : Comparison of FlexNoC (a), EDK (b) execution with different numbers parallel Ids**

We perform the same parallel Design space exploration on the machine THALES 2 using the tools FlexNoC and EDK. The execution time is more than the double when the number of parallel IDs changes from 4 to 16. The execution time is also multiplied when the number of parallel IDs increases with EDK tools.

The user should find the good combination between the number of parallel IDs and the execution time by IDs. Increasing the number of parallel designs make the achievement of the single workflow slower. A predictive model can give a better approach to make the optimized scheduling.



**Figure 2.15. THALES2: Comparison of FlexNoC (a), EDK (b) execution with different numbers parallel Ids**

Until now all the presented work was performed on a single FPGA, we propose now to use a multi FPGA platform. That is why we implemented the next architecture with a mesh topology using 48 processors and 32 memories. This architecture needs 5 FPGAs platforms. The implementation of this design was performed on ZEBU-UF4 board. The resource utilization is presented in Table 2.9.

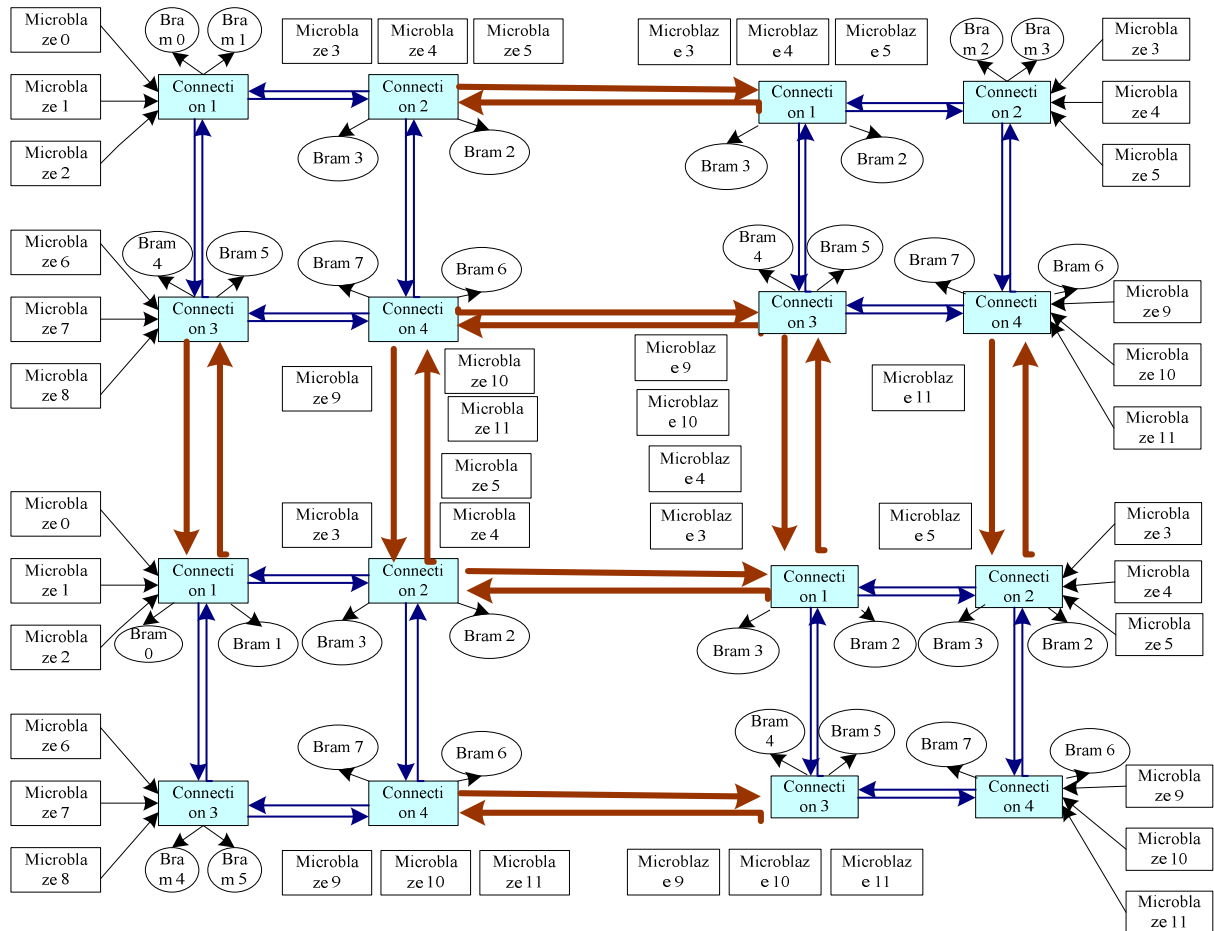


Figure 2.16. SSM IP 48 processors 32 BRAMs  
 Table 2.9. SSM IP 48x32 on Zebu UF4 Resource utilizations

Modules	Descriptions	Design 2	SSM IP	48MB,32 BRAMs
FPGA	4 Virtex-4 LX200	<b>Ressources utilizations</b>		
DRAM	512 MBytes			
SSRAM	64 MBytes			
ICE	Smart and Direct	<b>Bram</b>	<b>Slices</b>	<b>DSP48</b>
FPGA1	F_0_0_0	76%	44%	31%
FPGA2	F_0_0_1	57%	14%	9%
FPGA3	F_0_1_0	1%	78%	66%
FPGA4	F_0_1_1	95%	27%	18%
FPGA5	F_1_0_0	2%	32%	17%

We perform the same design space exploration already presented in 2.3.3. In fact, we change the population size of the generation and the number of parallel IDs. The obtained results are summarized in Table 2.10. By using MPSOC 7, it was only possible to run 8 parallel IDs with FlexNoC tool, 4 parallel IDs with EDK tool and only 2 parallel IDs with zCui while no parallel execution was possible for zCui tool using the MPSOC 4 machine. With this design the memory size of the used machines was the real limitation of the parallel DSE. To perform the place and route operations, the machines use all the available memory. The variation of the average execution time of the FlexNoC tools is not really meaningful with the machine MPSOC7, but this one doubles when the population size doubles on the machine MPSOC4. The parallel run of the FlexNoC tool is affected by the difference between the available memories in both machines.

From those results we can conclude that with the EDA tools, using deterministic mathematical algorithms to perform the synthesis the place and route design, the choice of the machine is a major operation to optimize the execution time of the exploration. For example to perform the synthesis, the place and route operations with EDK (2 parallel Ids) the machine MPSOC7 needs about 2 hours while MPSOC4 takes about 4 hours.

**Table 2.10. SSM IP 48x32 Exploration Results**

<b>Exploration Results : MPSOC7</b>					
<b>software</b>	<b>Population size</b>	<b>Parallel Ids</b>	<b>Max(s)</b>	<b>Min(s)</b>	<b>Average(s)</b>
<b>FlexNoC</b>	24	2	593	539	554,96
<b>FlexNoC</b>	24	4	614	568	591,71
<b>FlexNoC</b>	24	8	629	580	611,17
<b>EDK</b>	24	2	8297	6796	7600,58
<b>EDK</b>	24	4	16309	8944	12678,79
<b>zCui</b>	24	2	8829	6759	7778,71

<b>Exploration Results : MPSOC 4</b>					
<b>software</b>	<b>Population size</b>	<b>Parallel Ids</b>	<b>Max(s)</b>	<b>Min(s)</b>	<b>Average(s)</b>
<b>FlexNoC</b>	24	2	816	799	807,666667
<b>FlexNoC</b>	24	4	1683	1583	1633,20833
<b>FlexNoC</b>	24	8	4051	2193	3267,625
<b>EDK</b>	24	2	14181	11850	13105,6667
<b>EDK</b>	24	4	BLOCKED	BLOCKED	BLOCKED
<b>zCui</b>	24	2	BLOCKED	BLOCKED	BLOCKED

## 2.5 Return on experience: Analyses and discussions

We presented in this chapter an experimental study of FPGA implementation using different MPSOC designs. During the first step of the implementation, we needed to make different choices on the machine, the tools and the scheduling of the exploration. It was clear that there is no detailed work giving such information. That is why we performed a design space exploration to get the optimized configuration. The number of input variables in the step of design is important. That is why we performed a design space exploration to test the effect of the different parameters.

Thanks to our result we have now a data base of experimental results presenting the relationship between different parameters. The real implementation and execution on FPGA, offers a real freedom to realize this exploration compared to the simulation methodology. This work can be a good database to perform a theoretical study and to create a predictive model serving as an “Adviser” to the new users of the explored tools.

## 2.6 Conclusion

We presented in this chapter the design of an MPSOC design having 16 masters and 16 slaves based on the Butterfly NoC topology. This project was implemented on the Multi FPGA platform ZEBU–UF4. We applied on this hardware design a parallel programming of the Harris Filter algorithm. Thanks to this parallel programming, the speed up has reached the value 10 when the sixteen processors were fully used.

During this step of design, the shortage of information about the choice of the suitable machines, tools, platforms, number of licences was a real handicap. That is why we performed a Design Space Exploration by changing the different options of the processors and the NoC. We applied a Multi Objective Evolutionary Algorithm in order to find the optimized sets of individuals which are minimizing the area and the execution time of the design. We got a pareto front of the optimized IDs.

During our workflow implementation, three basic tools were used which are FlexNoC (NoC extraction), EDK (place and route), zCui (portioning and execution). We performed a Design Space Exploration in order to study the effect of the different parameters on the performance of those tools. Thanks to the variation of the machines, the population size, the number of FPGA and the architecture of the MPSOC, we provide a wide data base which can be the input of a predictive mathematical model which is created later during an internship in

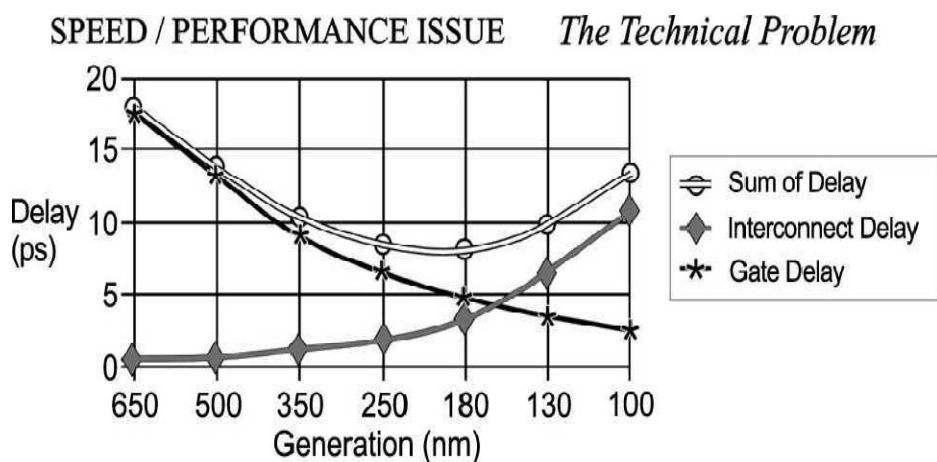
the lab [38]. Thanks to this model the user can have a better idea to make different choices during the implementation of his design.

## Chapter 3 : 3D Semi conductor Technology

### 3 3D Semi conductor Technology

#### 3.1 3D Semi conductor Technology: Motivation

The development of semi conductor technology has targeted at the same time the increase of the performance and the reduction of the power consumption and the cost. Transistor shrinking was the principle key of this evolution but in these last years this method has reached its limits. One of the major faced problems with high technology is presented in Figure 3.1. In fact, between the 180 nm and the 130 nm the interconnect delay overcomes the gate delay. In other words, the links are becoming the new limitation of the design's performance.



**Figure 3.1. Gate and Interconnect Delay as a function of gate technology [39]**

Three-dimensional (3D) integrated circuits (ICs), is a design containing multiple active silicon layers. This new approach of design can deal with the problem of the interconnection delay. In fact, the key of the 3D IC design is the use of the vertical connection called TSV (Through Silicon Via) which is an efficient technique to shorten the global interconnection of the design.

The other major limitation of the 2D shrinking is the increase of the cost changing from a technology to another. CMOS scaling requires much higher cost, For example a node mask set for 90 nm is about \$1 million while it is about \$2 million for 45 nm technology (see Figure 3.2). We present in the same figure, the comparison between the CMOS scaling and two different methodologies of 3D IC. It is clear that the cost of a single die overcomes the cost of 3D design with high technology.



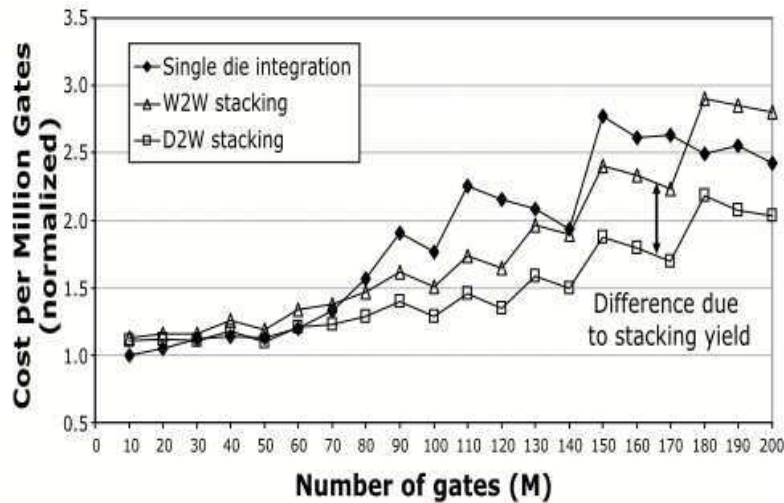


Figure 3.2. Cost CMOS scaling[40]

### 3.2 3D Semi conductor Technology: State of the Art

Three-dimensional integrated circuit (3D-IC) is not really a new field, in fact it has been studied since the 1980s [41][42]. However, CMOS scaling following Moore’s Law has been the most explored in order to increase the density and the performance of ICs. The 3D IC is emerging in the last decade as CMOS scaling has already reached its limitation.

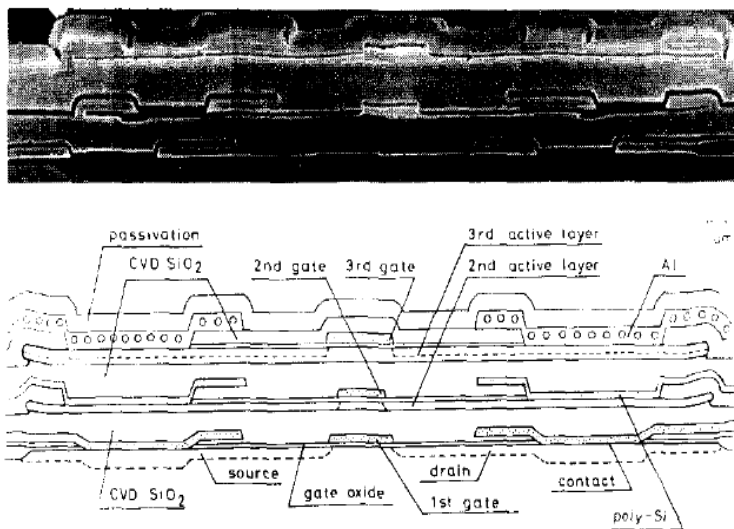


Figure 3.3. Example of 3D Design [42]

The 2005 edition of the ITRS was the first one which is projecting the need for focusing not only on device integration that relies on the improvement of the form factor (More Moore) but also on applications leveraging silicon technology to provide added Options (More than Moore) (see Figure 3.4).

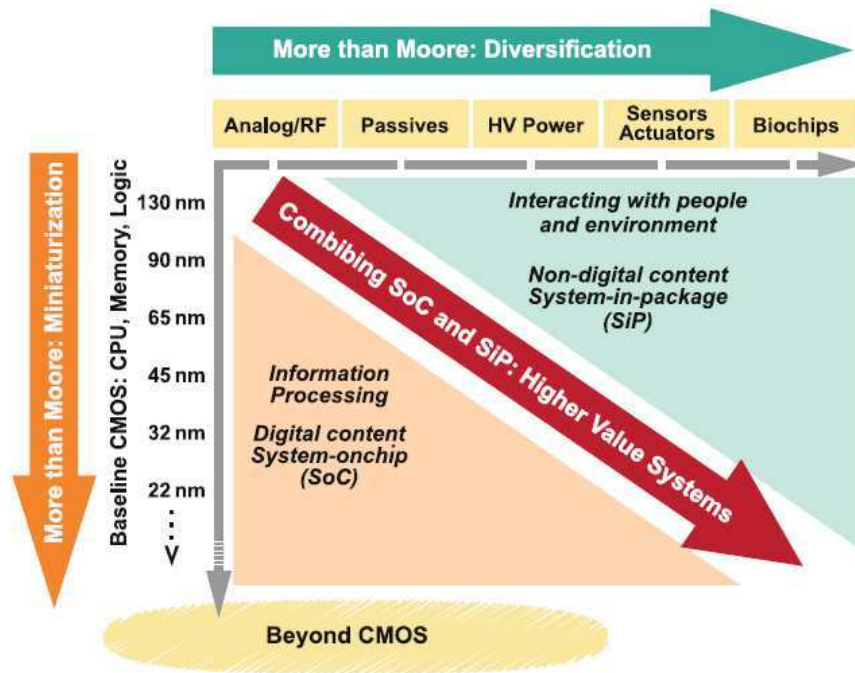


Figure 3.4. Illustration of the evolution of the semi conductor technology with CMOS scaling with other ways of development offering new functionalities [2]

We present in the Table 3.1, a summary of the implemented 3D MPSOC for these last years. Even though the number of the papers studying 3D design is considerable, only few works present a real implementation of a 3D MPSOC design.

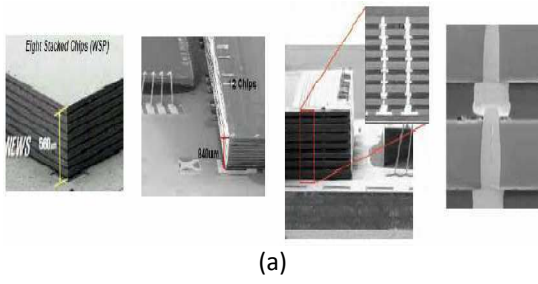
L.Zhou et al [43] implemented a 3D LDPC (Low Density Parity Check) design using three tiers with the technology 180 nm. The frequency of this design reached the value of 128 MHz. The comparison of this 3D design with a 2D equivalent one has shown an improvement of the 3D technology in term of power consumption, global interconnection, clock skew and area. In [44] authors have designed a 3D NoC with Mesh topology. This design fitted on 1mm<sup>2</sup> area with the technology 130nm. To perform this operation, 100 vertical connections were used to ensure the connections between the different layers. The frequency of the design can only reach 25 MHZ. An implementation of 3D adder and multiplier was performed in [45]. Authors have used the technology 180 nm and TSV to ensure the communication between the three different wafers. This work has proved the efficiency of 3D design with an improvement up to 34% for speed and up to 46% reduction for power consumption. Memory stacking is one of the major motivations in 3D IC design. In both works presented in [46][47], authors have designed a 3D stacked SRAM memories. In fact, Chen et al have designed a 3D SRAM using MITLL 180 nm FDSOI process, which improves the access time by 32%. and speed up the access time to the world line of the memory. A 3D implementation of a complete MPSOC design containing 64 cores and having a mesh topology was the subject of the paper

[48]. This work was performed using 3D Tezzaron technology with 130 nm technology. This design has a frequency of 277 MHz . The vertical connections were ensured using TSV and Micro bumps from 3D Tezzaron technology. The same technology was also used by Thorolfsson et al to implement the 3D SoC for H.264 design[49].

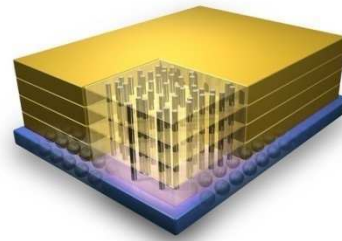
**Table 3.1. 3D MPSOC design implementation [50]**

Teams	Architecture	Technology/Number of tier
<b>L.Zhou et al [43] 2006</b>	3D LDPC decoder	180 nm / 3tiers
<b>C.Mineao et al 2008 [44]</b>	3D mesh NoC	130 nm / 2 tiers
<b>J. Ouyang et al [45] 2009</b>	3D adder and 3D multiplier	180 nm / 3 tiers
<b>X.Jing et al [47] 2010</b>	3D SRAM	180 nm / 3tiers
<b>M. B. Healy 2010 [48]</b>	3D multicore (64 cores)	130 nm / 2 tiers
<b>T. Thorolfsson et al [51]2010</b>	3D SoC for H.264	130 nm / 5 tiers (2 tiers for logic, 3 tiers DRAM)
<b>X.Chen et al [46], 2011</b>	3D SRAM	180 nm / 3tiers

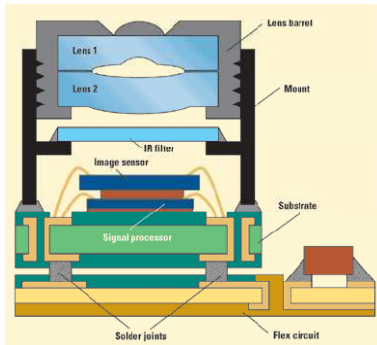
3D IC is relatively a new research field that is why there is a shortage of 3D industrial chips. The leader semi conductor companies like Intel and IBM are in a real course to be the first provider of the first 3D industrial chip. We present in the Figure 3.5, a set of 3D industrial designs. In 2006 and for the first time, the three dimensionally stacked NAND Flash memory was produced by the company Samsung. Thanks to the technology S3 (single-crystal Si layer stacking), which was used to develop S3 SRAM previously, it was possible to double the memory capacity without increasing the chip size [52]. On October 2011, Samsung and Micron announced the new project of a new Hybrid Cube Memory (HMC) presented in Figure 3.5 (b). This memory is designed to ensure high performance computing which can send information from memory chips to the CPU. IBM has announced in November 2011 the production of the first commercial chip based on Micron's Hybrid Memory Cube using TSV technology. The actual high volume production is noticed for the image sensors of Toshiba.



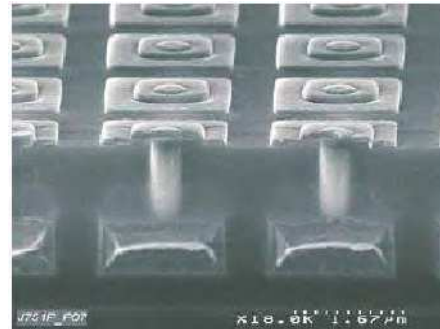
(a) Samsung's Stacked Flash Memories 2006



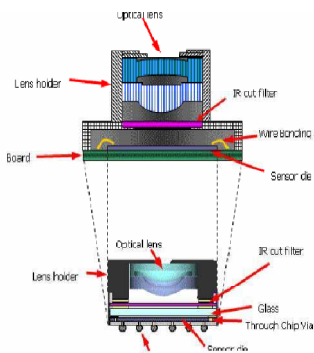
(b) Micron's Hybrid Memory Cube (HMC) produced by IBM[53] (2011)



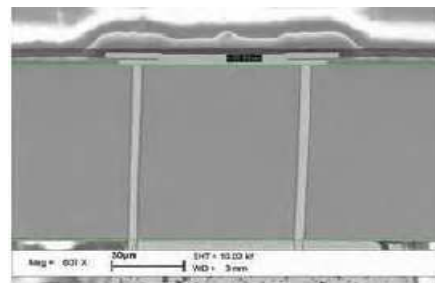
(c) CMOS Imaging Sensor  
CIS 1Q'07 Market Share



(d) 2MPixel (2.6x2.6um pixel) CIS: Leti & ST (Jun. 2007)



(e) Toshiba Image Sensor with TSV 2008

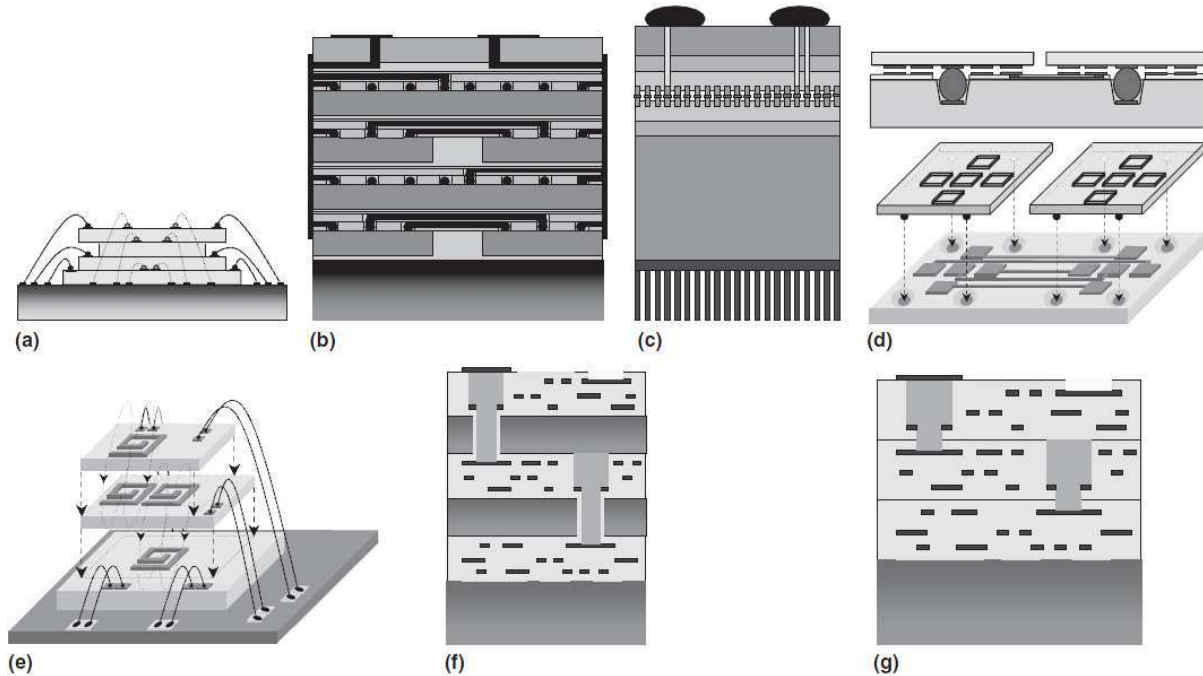


(d) Cross-section image of IBM's "through-silicon-via" technology in a stacked chip. (Source: IBM)

**Figure 3.5. 3D IC industrial design**

### 3.3 3D Design Methodologies

The fabrication of a 3D IC design can be performed using different approaches. We present in Figure 3.6, the basic 3D IC methodologies.

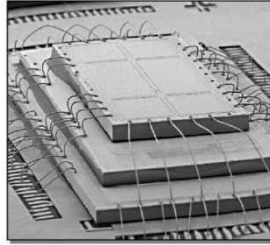


**Figure 3.6. Illustration of vertical interconnect technologies: wire bonded (a); microbump—3D package (b) and face-to-face (c); contactless—capacitive with buried bumps (d) and inductive (e); through via—bulk (f) and silicon on insulator (g) [54]**

#### 3.3.1 Wire Bonded System-in-Package

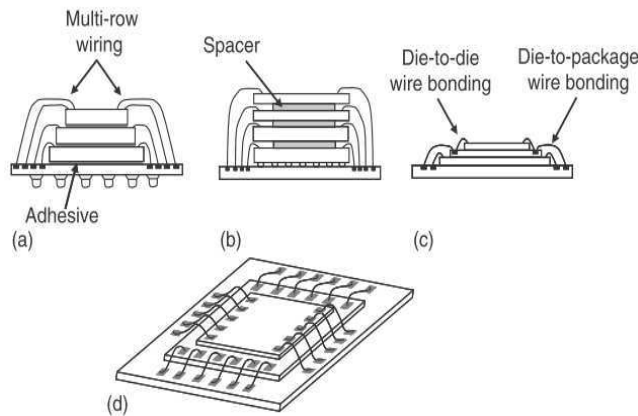
The Wire Bonding approach is the most common 3D methodology. After processing and testing the independent components, they are stacked to create a System In Package (SiP). The connection between the stacked chips is performed using external wires. The major limitation of this solution is the resolution of wire bonders and especially the increase of the number of inter chip connections. Unlike the other 3D IC approaches the vertical connections in the wire bonded solution can be only performed on the chip's periphery. This can seriously limit the vertical interconnection density which is estimated to 102-103/cm<sup>2</sup> [54]. The reduction of size is the only meaningful benefit offered by the chip stacking method. In fact the connecting wires can be shorter but the size of the components is almost the same compared to 2D Design.

Chip stacking is a technology offered by companies like Sharp and STATSChipPAC. Chipstacked. The produced SiPs are employed in cell phones thanks to their tight size. We present in Figure 3.7, an example of a 3D design based on the wire bonding methodology.



**Figure 3.7. Wire bonding design [55]**

We present in Figure 3.8, different structures of wire-bonded. The separation between the different stacks can be performed using spacer or adhesive. The connection using wire bonding can be applied between Die to Die or die-to-package. The performance of the 3D stacked design is determined from the length of the bonding wires and the resulting parasitic impedance. Stacking up to four or five dies have been already implemented [56]. But the parasitic impedance and the shortage of the number of available bonding wires are the principle limitations of this approach leading designers to find other SiP methods to deal with those problems.

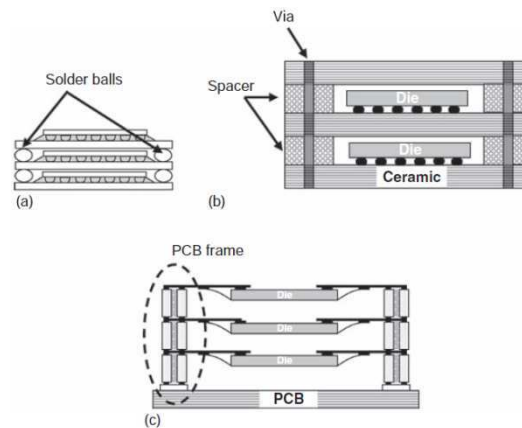


**Figure 3.8. Wire bonded System-in-Package[57]**

### 3.3.2 Peripheral Vertical Interconnects

In order to overcome the limitation of the Wire bonding method, a new SiP using vertical peripheral connections is used. In fact the designer can replace the classical wires by solder balls or Through Hole vias illustrated in Figure 3.9. Thanks to this solution, the number of stacked dies can increase as the constraints of parasitic, the impedance and the number of

linking wires are relaxed. Those techniques were used by several companies. In fact, Micron Corporation company uses the Via hole method to produce fast 3D memory chips while Hitachi attached the vertical pillars to the Printed Circuit Boards (PCB) to perform the vertical connections of the 3D chip.



**Figure 3.9. SiP with peripheral connections: (a) solder balls (b) through-hole via and spacers , (c) through-hole via in a PCB[57]**

### 3.3.3 Micro Bumps

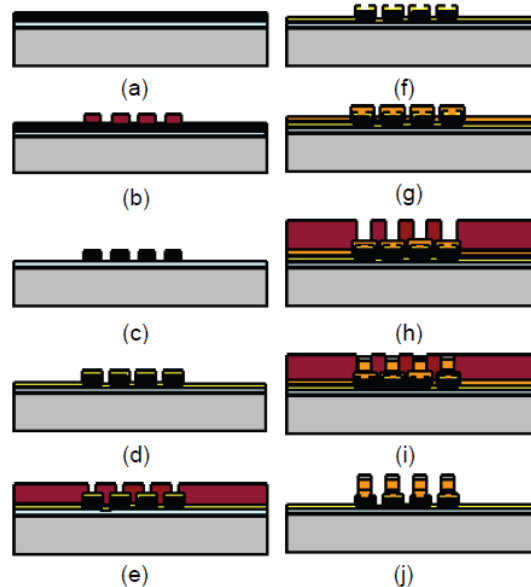
The Micro Bumps can be defined as Small solder ball used to connect one die to another; they are normally connected to a micro bump pad, or micro ball. Micro bump technology is the use of solder or gold bumps on the surface of the die in order to establish the vertical connections between the different dies. This 3D methodology was used in different packaging technologies, like wire bonded chip stacking, chip stacking, system in package and 3D IC integration. In fact, the use of Micro bumps increases meaningfully the interconnections density with a low cost. In order to decrease their resistance, the Micro bumps are usually made up of Cu and Sn. Different bonding processes have been developed for example the solid-liquid-inter diffusion method [58] and the thermal compression method [59].

We present in Figure 3.11, an example of a fabrication process of CuSn Solder Micro bumps [60]. The Micro bumps are created on both sides of the Si chips to allow the assembly of the 3D Chip with a Face to Face stacking method (see Figure 3.10).



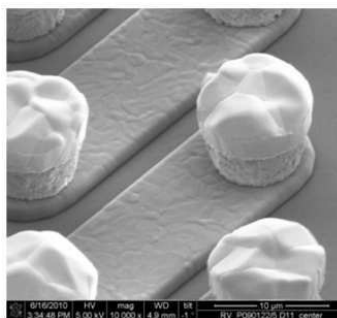
**Figure 3.10. 3D Chip with Micro Bumps**

At the first step, two layers of SiO<sub>2</sub> and Al films with a thick of 1 $\mu$ m are deposited on the wafer (Figure 3.11.a). Then a patterned layer of 2 $\mu$ m photoresist is applied (Figure 3.11.b). In order to form the metal pad the Al is etched and the photoresist is removed (Figure 3.11.c). Another layer which has a passivation function is applied (Figure 3.11.(d,e,f)). After this, a Ta/Cu of adhesion layer and seed layer are sputtered (Figure 3.11.g). After depositing a layer of a thick photoresist (Figure 3.11.h), the CuSn is applied. In the last step, the photo resist is stripped; the Cu and the Ta are etched sequentially.



**Figure 3.11. Process flow fabrications of CuSn solder Microbump[60]**

This 3D approach is used by many companies. For example the company IMEC [61] is providing CMOS image Sensors with high interconnect density using the Bump with a pitch of 20 $\mu$ m in CuSn. These Micro bumps are illustrated in Figure 3.12.



**Figure 3.12. SEM picture a die part of the interwoven daisy chain with 10 $\mu$ m diameter CuSn bumps formed by electrochemical plating. The pitch of the bumps is 20 $\mu$ m [61]**



### 3.3.4 Through silicon via (TSV)

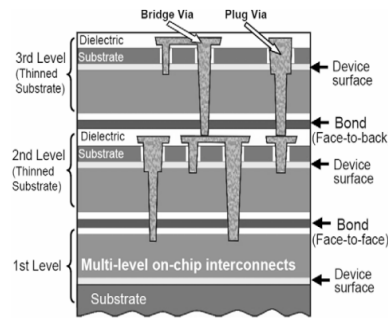


Figure 3.13. Through silicon via (TSV)[62]

The TSV (Through Silicon Via) is the most used 3D technology because it can be scaled to achieve a high vertical connection density. It presents also physical advantages like low electrical resistance, low parasitic capacity and impedance which represent real motivations in the Microelectronics design. The 3 main steps in the fabrication of a 3D IC are:

- Wafer Thinning
- TSV etching and filling
- Tier Bonding

When we change the order of those steps we can define different processes like “TSV first”, “TSV last” and ‘ TSV middle’.

#### 3.3.4.1 Structure

The TSV is basically characterized by its diameter and its pitch. These two values are shrinking when the technology is improved. We present in Table 3.2, the prediction of ITRS roadmap for the 3D TSV evolution. In fact by 2015, the TSV diameter will reach  $1\mu\text{m}$  while the TSV pitch will be equal to  $2.5\mu\text{m}$ . This evolution will increase the density of the 3D interconnections in a 3D chip and improve the global interconnect delay.

Table 3.2. High-density through silicon via projections in 2008 ITRS update [63]

Principle parameters	2008	2009	2010	2011	2012	2013	2014	2015
TSV diameter, $D(\mu\text{m})$	1.6	1.5	1.4	1.3	1.3	1.2	1.2	1
TSV pitch, $P(\mu\text{m})$	5.6	5.5	4.4	3.8	3.8	2.7	2.6	2.5
Pad spacing, $S(\mu\text{m})$	1	1	1	0.5	0.5	0.5	0.5	0.5
Pad diameter, $PD(\mu\text{m})$	4.6	4.5	3.4	3.3	3.3	2.2	2.1	2
Bonding accuracy, $\Delta(\mu\text{m}), 3\text{ sigma}$	1.5	1.5	1	1	1	0.5	0.5	0.5

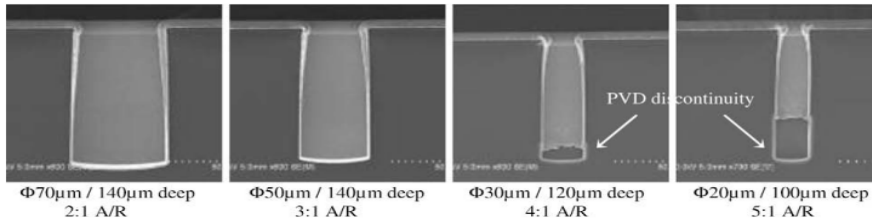


Figure 3.14. TSV examples [64]

Figure 3.14 is an illustration of different TSV examples designed in [64]. The diameter is varying from 20µm to 70µm while the deep is in the margin of 100-140µm. Thanks to this configuration the TSV density can allow for a pitch of 10µm a number of 10 000 vertical TSV connections on the area of 1mm<sup>2</sup>.

3.3.4.2 Process of fabrication

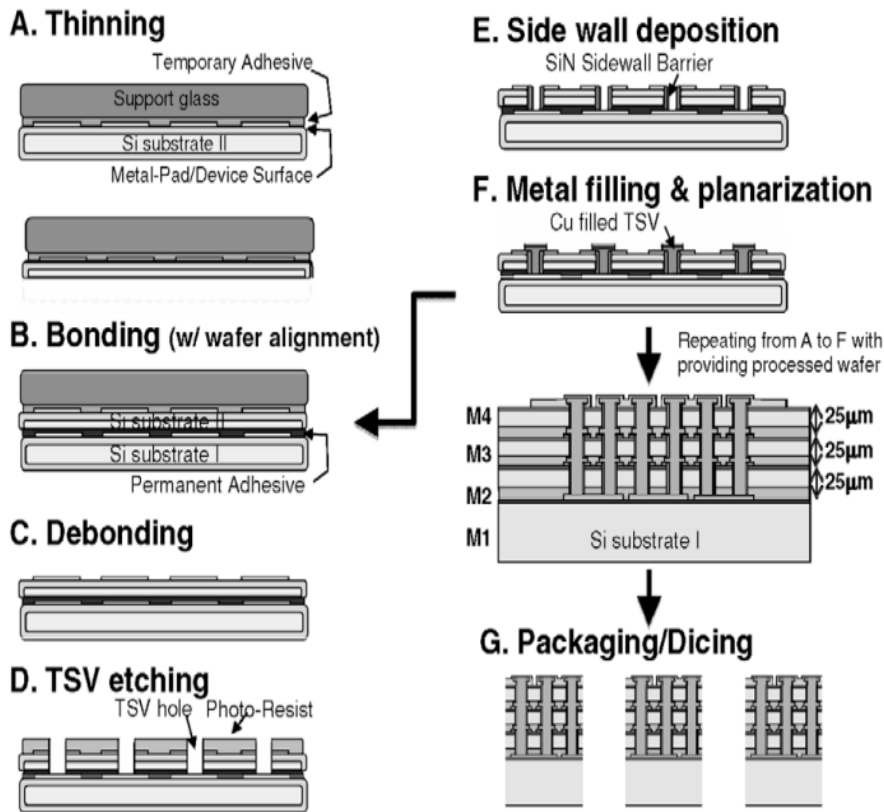


Figure 3.15. General TSV flow fabrication [65]

We present in Figure 3.15, a WOW (Wafer On Wafer) flow process. At the first step illustrated by (A) the Si wafer called Substrate I is thinned then bonded to another Si Wafer (Si Substrate II) (B). These two steps are the step of thinning and bonding. During the third operation called “Debonding”, the support glass is removed. The TSV etching is performed during the fourth step of this process. Playing the role of an isolator, the SiN is chemically deposited inside the TSV holes (E) then the TSV is filled with the metal Cu which is

illustrated in Figure 3.15 (F). After the Wafer stacking the process is repeated to create all the TSVs at the different levels.

The presented process is related to the Via first process fabrication. In fact, The TSV creation is performed before the stacking operation. When we invert the order of these steps, the process is called Via last. We can see the difference between the two methodologies in Figure 3.16. The summary of the different TSV process flows is presented in Table 3.3.

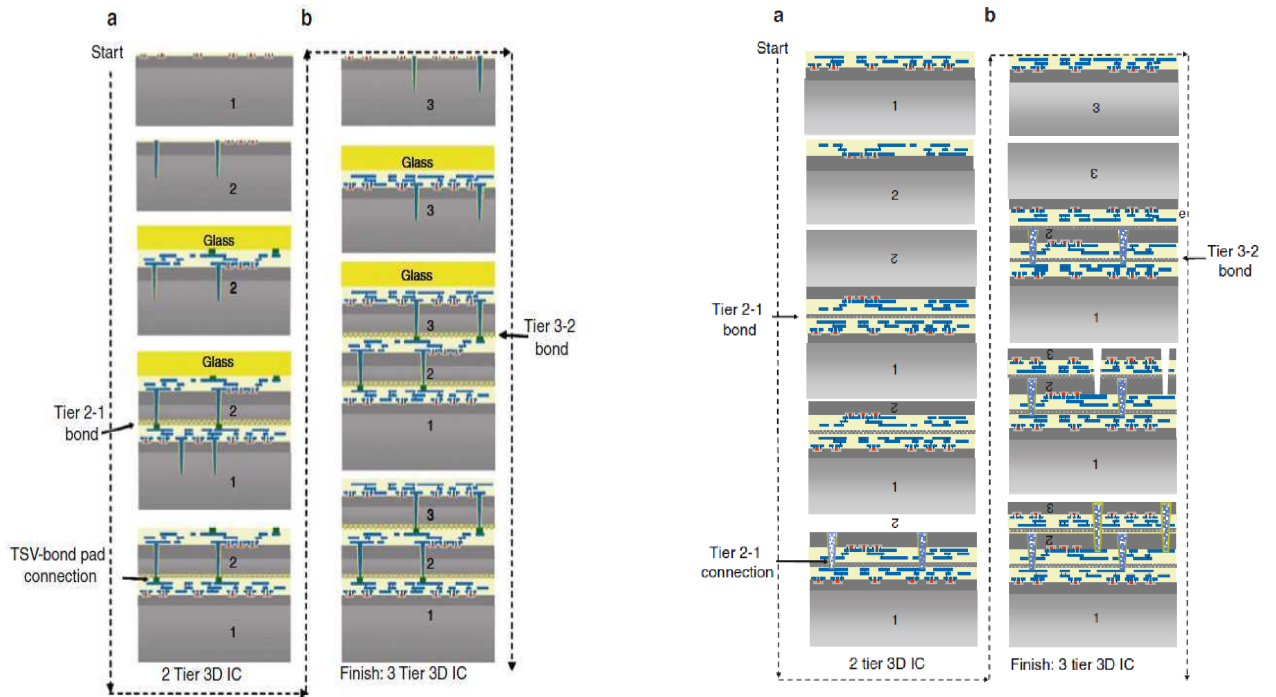


Figure 3.16. Via first (left), Via last (right) 3D IC methodologies[63]

Table 3.3. TSV process flows [63]


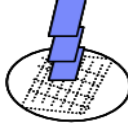
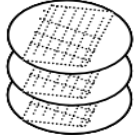
TSV first	TSV middle	TSV last
Etch deep silicon cavities	Etch deep silicon cavities	Fabricate transistors
Insulate cavities	Insulate cavities	Fabricate BEOL interconnect
Fill cavities with a conductor	Fabricate transistors	Bond Wafer pair
Fabricate BEOL interconnect	Fill cavities with a conductor	Thin backside of upper wafer
Bond wafer pair	Fabricate BEOL interconnect	Backside etch deep silicon cavities
Thin Backside of upper wafer	Bond wafer pair	Insulate cavities
Fabricate BEOL interconnect on upper wafer	Thin backside of upper wafer	Fill cavities with conductor

### 3.3.4.3 Stacking Methods

The step of stacking or bonding can be realized with different methods: Chip to chip, chip to wafer or Wafer to Wafer. The choice of the appropriate technology depends on the user's application and especially on the size of the die, the density of the interconnections, the

alignment and the bonding yield. The main advantage of using a Chip to Chip, or a Chip to Wafer methodologies is the possibility to test the used dies before the stacking operation which reduces the probability of the design failure. In fact only Known Good Die (KGD) which are already tested are used. This is not the case when we use a wafer to wafer technology where the testing step is performed after the fabrication of the whole chip. For this last approach the low cost is the main argument to choose it. Adding other steps in the 3D IC process fabrication represents the principle inconvenient in the Chip stacking techniques.

**Table 3.4. Comparison between bonding methods (KGD: Known Good Die)[60]**

	<b>Chip to Chip</b> 	<b>Chip to Wafer</b> 	<b>Wafer to wafer</b> 
Pro	Flexible, use of KGD	Flexible, use of KGD	Low cost
Con	Handling and bonding	Handling and bonding	Overall yield, chip size
Wafer thickness	< 4 $\mu\text{m}$ to > 150 $\mu\text{m}$	< 4 $\mu\text{m}$ to > 150 $\mu\text{m}$	< 4 $\mu\text{m}$ to > 150 $\mu\text{m}$
Bonding technology	Solder Metal to Metal Adhesive	Solder Metal to Metal Adhesive	Solder or Metal Oxide bonding Adhesive

Stacking methods can be also classified with reference to the direction of the active Silicon area into 3 families: Face to Face, Back to Back and Face to Back (see Figure 3.17). The Face to Face stacking is also suitable for 3D IC using Micro Bumps. For both solutions (Face to Face and Back to Back) the number of dies is limited to 2 which is the major limitation of these technologies. The Face to back methodology is more scalable and gives the possibility to use a non limited number of stacked dies.

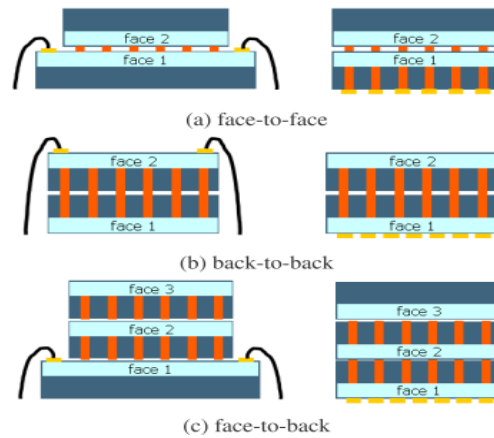


Figure 3.17. Stacking Methods[66]

3.3.4.4 TSV properties

We present in Table 3.5, the comparison between the used materials to fill the TSV. The copper presents the main advantage which is the high conductivity and the compatibility with the FEOL process. That is why; it is the most used in TSV technology. The Tungsten (W) is also a common material in the 3D integration but it has lower conductivity compared to the Copper. For the other materials, the low conductivity and the high price are the major problems limiting their use in 3D IC.

Table 3.5. Comparison of Via Filling Materials [55]

Method	Advantages	Disadvantages
W	<ul style="list-style-type: none"> <li>• W CVD is compatible with FEOL process</li> <li>• Perfect to fill small via with high aspect ratio</li> <li>• Can be applied at a relatively low temperature of 200°C</li> </ul>	<ul style="list-style-type: none"> <li>• Lower conductivity compared to Cu</li> <li>• Cannot fill big vias</li> </ul>
Poly-Si	<ul style="list-style-type: none"> <li>• Poly-Si CVD is compatible with FEOL process</li> <li>• Well established</li> </ul>	<ul style="list-style-type: none"> <li>• Low conductivity compared to Cu</li> <li>• Cannot fill big vias</li> <li>• Very high deposition temperature</li> </ul>
Cu	<ul style="list-style-type: none"> <li>• High conductivity</li> <li>• Well established ECD process</li> <li>• Compatible with metal BEOL process</li> </ul>	<ul style="list-style-type: none"> <li>• ECD depends on PVD seed layer</li> <li>• PVD seed layer is limited in achieving continuous film in high aspect ratio vias</li> </ul>
Au wire	<ul style="list-style-type: none"> <li>• Compatible with packaging standard wire bonding process</li> </ul>	<ul style="list-style-type: none"> <li>• Au price is increasing</li> <li>• The via dimension is limited by the wire diameter</li> </ul>
Conductive polymer	<ul style="list-style-type: none"> <li>• A standard polymer via filling process</li> <li>• Lower temperature process</li> <li>• High throughput and low cost</li> </ul>	<ul style="list-style-type: none"> <li>• Lower conductivity compared to metal</li> <li>• High moisture absorption compared to metal</li> <li>• Silver diffusion problem</li> </ul>

The use of the TSV can increase meaningfully the frequency of the design. In fact the delay is reduced to less than 1fs while it is more than 10 ps in a conventional CMOS wire [67]. The RC delay value in TSV depends on its material and on its diameter. We present in Figure 3.18, the variation of this value function of the TSV diameter for Copper and Tungsten. We can see that the copper has a better conductivity than Tungsten with lower RC delay values. For both materials the RC delay is proportional to the TSV diameter.

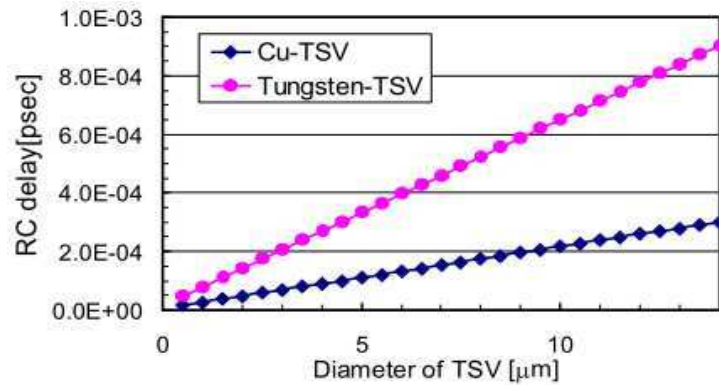


Figure 3.18. RC delay vs TSV diameter [67]

#### 3.3.4.5 TSV challenges

Even though TSV technology is a promising technique by reducing the problem of interconnect delay and form factor, it is facing a number of challenges. Area consumption is one of the main problems. In fact, one TSV fits in a significant silicon area: For example a TSV size is about 5 to 10 times the size of a standard cell in 32 nm (see Figure 3.19), which is also about 15 to 30 times of the minimum width of M1 [68]. The use of TSV has reached 66% of the total area consumption in the 3D design presented in [69]. The shortage of TSV tools, the high cost and the difficulty to test the chip are also challenges in the TSV methodology.

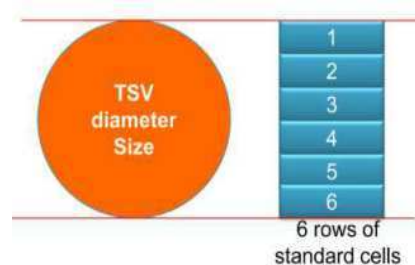


Figure 3.19. TSV Area estimation [68]

### 3.3.5 Contactless

Contactless coupling is another methodology of 3D design which does not require the processing steps to create the connections between the different layers. That is why this method is known to be cheaper than TSV or wire bonding methods. We can have two ways of contactless coupling: the inductive and the capacitive. When the vertical connections are based on the inductive coupling (see Figure 3.20), each layer has spiral conductor which is responsible of the vertical communications. In the capacitive coupling the small plate capacitors on chip are playing the role of the vertical interconnects.

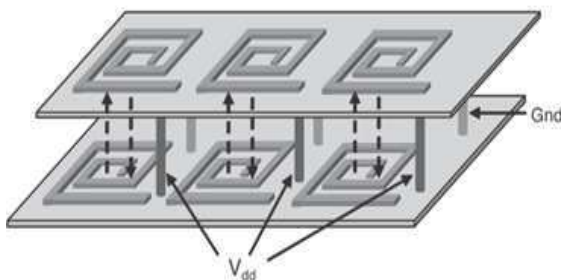


Figure 3.20. 3D IC Inductive coupling[57]

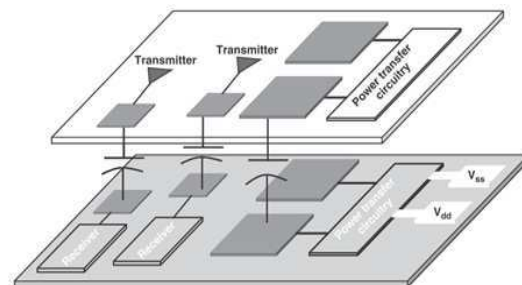


Figure 3.21. 3D IC capacitive coupling[70]

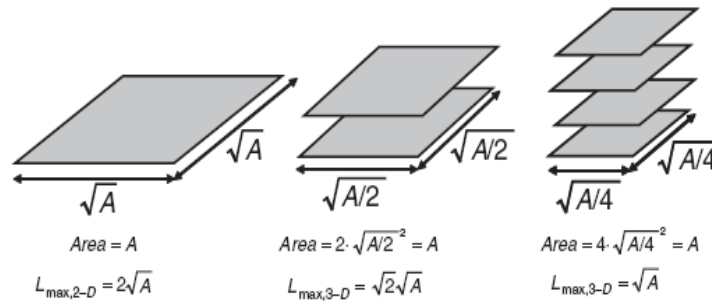
## 3.4 Benefits and challenges in 3D Design

3D IC stacking and use high number of vertical connections presents a lot of advantages for the final 3D chip. However there is a shortage of adequate tools to perform all the steps of the 3D design. We will detail in this sub section the benefits and the challenges of the 3D design approach.

### 3.4.1 Benefits of 3D Design

#### 3.4.1.1 Area Reduction:

3D design is considered as the new way to continue the evolution of semi conductor technology. A new rule know by “More than Moore “predicts to find new solutions for this domain (see Figure 3.4). 3D Stacking represents a promising solution to reduce the chip area. In fact, when the number of stacks increases the total chip area is reduced; an illustration of this reduction is presented in Figure 3.22. For a square Chip format, the area can be reduced to the half when the number of layers doubles: a 2D chip with an area A can be realized by a 3D chip with 2 layers but with an area equal to  $A/2$ . This is a suitable technique for big sizes of memory stacking.



**Figure 3.22. Area reduction with 3D Stacking [71]**

### 3.4.1.2 Performance improvement

The use of the 3D design reduces the area of the chip which directly affects the interconnection length. In fact, we can see that the longest 2D length in the 2D chip with an area  $A$  is equal to  $2\sqrt{A}$  as already illustrated in Figure 3.22. When the number of dies increases to 4, the maximum length of a wire can be equal to  $\sqrt{A}$ . If we increase the number of layers to  $n$  stacked dies, the maximum length will be reduced to  $\sqrt{n} \sqrt{A}$ . The reduction of the global interconnection delay improves the performance of the chip by reducing the propagation delay time of the wires. We present in Table 3.6, a comparison between the interconnect delay values when the number of levels increases. The reduction of the interconnect delay can reach 32% compared to a classical 2D design (ex: kogge-Stone Adder).

For tight technology, the interconnect delay reduction in 3D IC design, improves the chip performance better than CMOS scaling. This result is illustrated in Figure 3.23 : scaling the technology from 90nm to 65 nm reduces 7% of the average latency of the design, while with 3D IC and using the same technology the delay reduction is equal to 14% of the initial value. This comment is also valid with the technology 65 nm and 45 nm. The difference between the two techniques is more important when the technology is tighter. In fact for the 45 nm technology, the gain in term of delay reduction with 3D IC is about 22% while it is only equal to 7% with a 2D design using 32nm CMOS.

**Table 3.6. Performance and power comparison between different 3D architectures[71]**

# of input bits	Kogge–Stone Adder			Brent–Kung Adder
	16-bits		32-bits	32-bits
	Delay	Power	Delay	Delay
2 planes	20.23%	8%	9.6%	13.3%
3 planes	23.60%	15%	20.0%	18.1%
4 planes	32.70%	22%	20.0%	21.7%



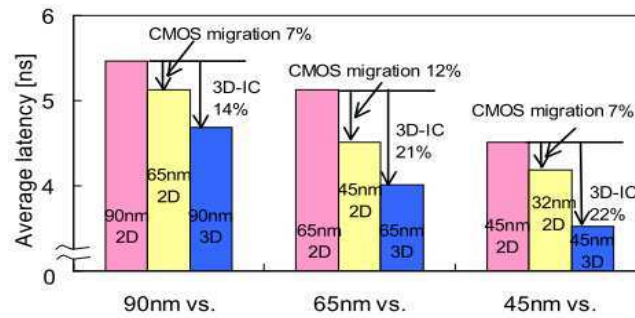


Figure 3.23. Average latency 3D IC and 2D IC [67]

### 3.4.1.3 Improve power consumption

Power consumption is considered as the second most important design optimization objective after cost nowadays. The chips are almost used in portable devices like cell phones that is why power should be minimized in order to maximize the time between battery recharges. The reduction of the interconnection's length, thanks to the use of the vertical links in 3D IC design, reduces the number of repeaters per gate Figure 3.24. This reduction decreases the power consumption of the whole design.

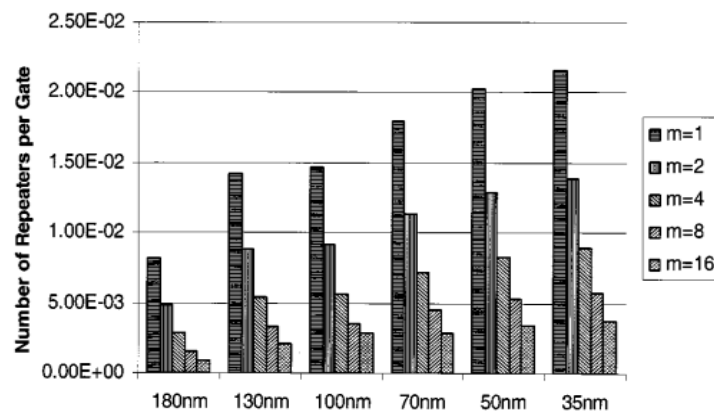
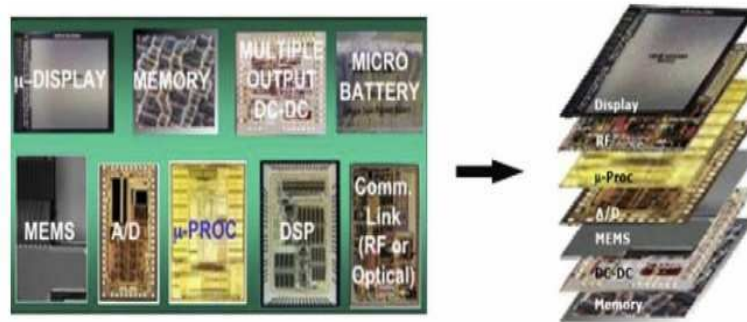


Figure 3.24. Number of repeaters with different technologies

### 3.4.1.4 Heterogeneous Technology

A typical Chip design includes different modules with different functionalities: Processing, memory, networking... In order to optimize the functionality of each IP, it is better to choose the suitable process of fabrication. This possibility is offered by the 3D stacking, allowing the connection between different dies of the chip even having different CMOS technologies. Dies can be produced by different vendors and then packaged in the same 3D chip.



**Figure 3.25. 3D chip with heterogeneous technologies [72]**

#### 3.4.1.5 Cost reduction

The cost is the real motivation behind the evolution of semi conductor technology. The classical CMOS scaling technique is becoming more difficult and more expensive to develop with tight technologies. The price of developing a new process for a new node technology is becoming so expensive that only big companies can choose it. We present in Figure 3.2, the cost of 3D approaches compared to 2D design integration. From these curves we can conclude that, when the number of gates increases the cost of 3D techniques (W2W and D2W) is usually lower than the cost of 2D integration.

### 3.4.2 Challenges of 3D Design

#### 3.4.2.1 Thermal dissipation

Thermal dissipation is a critical challenge in 3D design. In fact a 3D chip with stacked dies has higher temperature than a classical 2D design. A case study of a wireless sensor node presented in [69], shows that when the number of levels increases in 3D design the temperature increases. The obtained results are illustrated in Figure 3.26. There is a meaningful difference between the temperature of the first layer and the second one ( more than 200% of difference). It is clear that farther layers have higher Temperature. Thermal dissipation is a serious problem in chip design, in fact an increase of 15°C increases up to 15% of the interconnect delay and reduces the chip life time by 4 times[73].

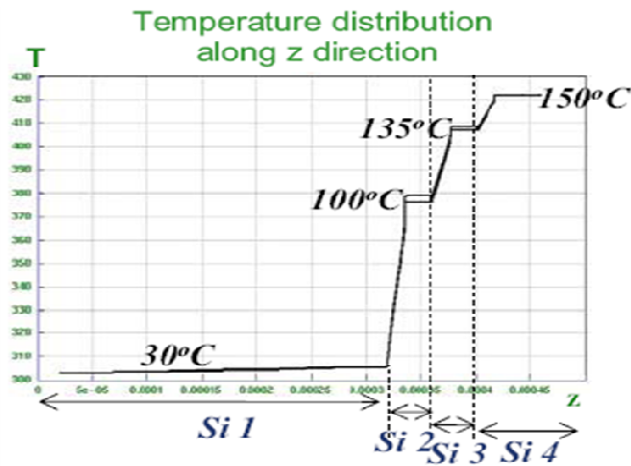


Figure 3.26. Temperature distribution along the z Axis with different Silicon layers [69]

To deal with this limitation, some methods are proposed to reduce the chip temperature. In fact, Thermal vias can be added to ensure the heat transfer from the different levels of the 3D chip. The Cu TSV, known by its high conductivity, is used to perform this operation (see Figure 3.28). These TSVs are passing through all the dies of the 3D chip. Microfluidic cooling is also another proposed solution to evacuate the heat from the 3D chip, this method is illustrated in Figure 3.27.

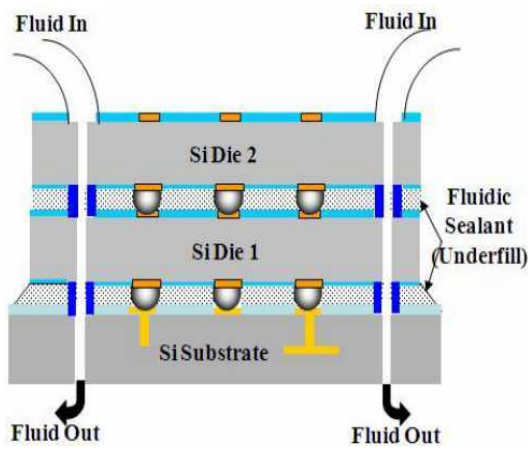


Figure 3.27. Microfluidic cooling

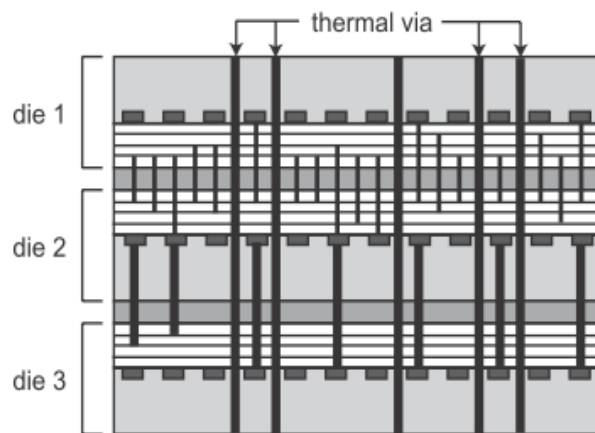


Figure 3.28. Thermal vias for heat dissipation [74]

3.4.2.2 Difficult Testing

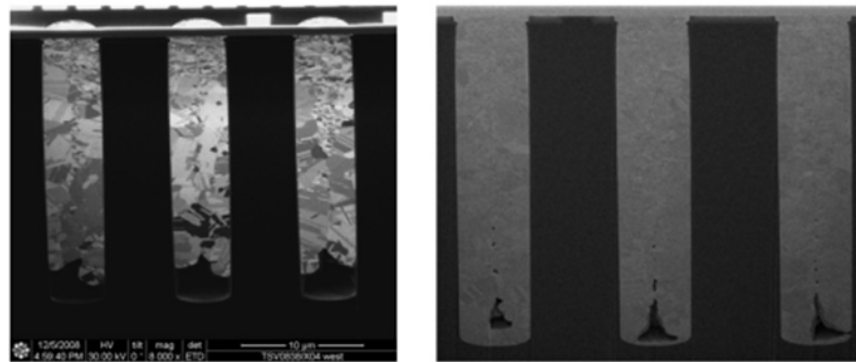


Figure 3.29. Examples of TSV defects: insufficiently filled TSV (right), TSV containing Micro voids (left)

Table 3.7. Reduction of integrated yield with stacking using wafer on wafer [70]

Number of tiers	1	2	3	4
Yield	95%	91%	85%	81%

During the step of TSV fabrication, many defaults can cause the damage of the whole system on chip. We present in Figure 3.29, two examples of TSV defects. When the TSV is not fully filled or containing micro voids, the vertical interconnect is not ensured. That is why when the number of stacks increases the yield of the chip decreases which is presented in Table 3.7 , this value can move from 95% for one layer to 81% with a 3D chip having 4 layers. The need of developing suitable testing methodologies is a real need to prevent such situations. Testing 3D becomes costly when the number of stacks increases especially when the designer wants to perform a complete test, an illustration of an example of 3D testing chip is presented in Figure 3.30.

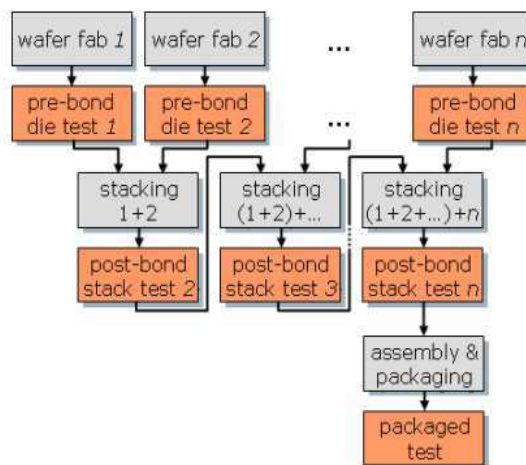


Figure 3.30. 3D IC testing model[75]

### 3.4.2.3 Bonding strategy

There is three different ways to perform 3D bonding: wafer to wafer, die to wafer and die to die. We have already presented the advantages and the issues of each methodology Table 3.4. Wafer to wafer bonding is the fastest one as all the wafers are created at the same time with the constraint to have dies with the same size and shape at the same wafer which reduces the possibility to have heterogeneous architectural layers. In the die to wafer stacking, we can have different dies on the same wafer. Single dies will be stacked on the fixed wafer. Even though this method presents a better freedom in term of design, the problem of alignment is more difficult. The die to die Bonding offers the freest way to use mixed technologies on 3D design however it increases production time and cost meaningfully[63].

## 3.5 3D Academic and industrial devices



Figure 3.31. Geographic mapping of 3D IC players [76]

Figure 3.31 represents the actual geographic mapping of the players in the 3D IC field. It is clear that 3D design is attracting the big semi conductor leaders. 3D stacking is commonly used in CMOS image Sensors by different companies like IBM, Samsung, Sharp ( see Figure 3.32). 3D research field is attracting industrial and academic labs. Even though researches are progressing, there is a real course between concurrent industrials to develop the first industrial 3D chip.



of stacking like wafer to wafer, die to wafer and die to die. The vertical interconnections are usually performed using TSV (Through Silicon Via).

3D IC design offers various advantages like the chip area reduction, the decrease of the interconnect delay and the decrease of power consumption, but challenges are also considerable. In fact, the heat dissipation and the shortage of specific 3D tools represent the major actual 3D IC difficulties.

The main objective of this chapter was to detail the different characteristics of the 3D technology in order to take them in consideration during the NoC synthesis step.

## Chapter 4 : NoC Synthesis methodologies



## 4 NoC Synthesis methodologies

In 3D IC design, new challenges should be taken in consideration like TSV assignment, heat dissipation and partitioning. With the high cost of the 3D IC design, the designer should consider the additional constraints in order to generate an optimized 3D NoC for a set of objectives; a such problematic is called the 3D NoC synthesis problem which will be the subject of this chapter.

### 4.1 2D NoC synthesis methodologies

Network-on-Chip (NoC) architectures have been gaining widespread acceptance as the new communication technology for multi-core systems, thanks to their high scalability, their predictability, and their performance. However, regular NoCs are resources hungry components; this fact represents the main reason to create NoC synthesis methodologies satisfying with optimal resources the needs of a particular application.

The definition of the NoC synthesis problem is the generation of a NoC topology optimized for a specific objective function with respect to various constraints. The NoC synthesis methodology should consider a multitude of non-trivial design problems which enlarge the design space of the possible configurations that is why some people choose to use heuristic methods to deal with this complexity. In spite of the complexity known to be NP-hard, some other use deterministic methods which are a real guarantee to find the optimized solution. Trying to find a deal between time and accuracy, we can find methods using a mixture of heuristic and deterministic techniques.

#### 4.1.1 Deterministic methods

The Linear Programming (LP) and the Genetic Algorithms (GA) are the main deterministic used methods in the literature to solve the NoC synthesis problem. The resolution of this problem with Linear Programming, when solved, will generate the optimized solution but when the problem is complex the resolution time can be too important that we can face memory or solver limitations. When the Design Space of possible NoC configurations is huge making impossible a Linear Programming resolution, a GA can be the solution. The use of GA coupled with the exploration of the Design Space does not guarantee reaching the optimized solution but will approach it in a faster time compared to exact methods. The main advantage of GA is the possibility to solve accurately multi objective problems, which is not the case of LP, in this case the resolution of the problem will not be a

single solution but a set of possible solutions representing a Pareto Front curve. Srinivasan et al [78] have presented an integer linear programming (ILP) to generate a low power custom NoC topology. In this work, authors have divided the NoC synthesis problem into 2 sub problems: a floorplanning problem and the interconnection problem. M.Jun et al have used in [79][80] a Mixed Integer Linear Programming (MILP) as a step of an iterative method to solve the NoC synthesis problem. The GA has been used in [81] to generate two NoC architectures: the first one is optimized for area while the other one is optimized for delay. The generated solution is a combination of both separate topologies. This technology is also used in other papers like [82][83]. In the majority of these previous works, an operation of core partitioning is applied at some level to reduce the mathematical complexity of the problem. This step is in the majority of cases done with heuristic algorithms.

#### 4.1.2 Mixed methods

In the previous section we presented works using deterministic methods for the main NoC synthesis step. In reality the problem can be divided into sub problems and in this case we can find that in some works authors use a deterministic method in one step while they use a heuristic methodology in another. We present in Table 4.1, the summary of 2D NoC synthesis methodologies. The work of K.Srinivasan et al presented in [78], is in reality a mixture of deterministic and heuristics methodologies. Even though the principle steps of the NoC synthesis problem were performed using LP algorithm, some steps like the floorplanning of the routers or the clustering were heuristics. In fact the authors have considered that routers can only be placed on the corners of the cores. The NoC synthesis solution proposed by B.Yu et al in [84], is based on the Min-Cut algorithm for partitioning and NoC synthesis generation. In this work authors suppose that the routers can only be only inserted in the white places of the floorplan.

#### 4.1.3 Heuristic methods

In some works, authors choose heuristic methods to solve the problem of NoC synthesis. In the quasi totality of the cases, this is used to reduce the complexity of the problem when deterministic methods fail to generate the optimized solution. Heuristic is usually used in clustering and solution post processing. V. Dumitriu et al [85], have used a fully heuristic method to deal with the complexity of the NoC synthesis problem. Using the principle of merging and dividing routers, the authors can explore different topologies. With this method there is no prove to attend the optimized solution, but this algorithm can find

feasible solutions for complex designs. It is interesting to study the reduction of the complexity when using heuristic methods which is the real motivation to choose between the different proposed solutions. In the majority of the presented solutions, there is a need to study the theoretical complexity of the proposed methodology and the complexity of the tested benchmarks.

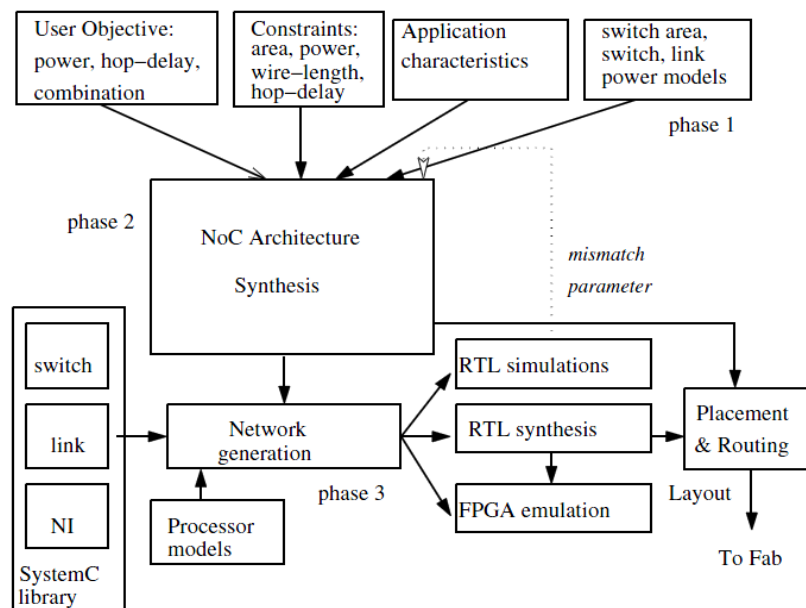
**Table 4.1. Summary of 2D NOC synthesis Methods**

<b>Works</b>	<b>Clustering</b>	<b>Core Floorplanning algorithm</b>	<b>Switches Placement algorithm</b>	<b>NoC Topology</b>	<b>Assumptions</b>	<b>Nature</b>
<b>K.Srinivasan et al [78]</b>	* Heuristic	* MILP(Mixed Integer Linear Programming)	*Heuristic	*MILP(Mixed Integer Linear Programming)	* Routers are smaller than cores. * Place routers in the corners.	Mixed( deterministic + Heuristic)
<b>M.Jun et al MIRO [79]</b>	*Heuristic	* Parquet tool: system level Floorplanner	*Heuristic	*MILP	* Single frequency of the crossbar switch network. * Single path between two communicating terminals.	Mixed( deterministic + Heuristic)
<b>M.Jun et al [80]</b>	*No	*No	*No	*MILP	* Consider only the router hops.	deterministic
<b>A.A.Morgan et al [81]</b>	*No	* No	*No	*MOGA	* The length of all the links is the same and allows for a single clock cycle data transfer *Fixed routers number	deterministic
<b>G.Leary et al [82]</b>	*No	*Parquet floorplanner	*MOGA	*MOGA	* Latency constraints are represented by the number of	deterministic

					hops.	
<b>X.Li, O.Hammami [86]</b>	*No	*No	*No	*MOGA	* Consider a NoC with only 2 stages.	deterministic
<b>B.Yu et al [84]</b>	*yes , Min-cut	*yes	*Heuristic	*Min-cost	*consider partitioning after floorplanning	Mixed( deterministic + Heuristic)
<b>V. Dumitriu [85]</b>	*No	*No	*No	* point-to- point oriented algorithm * Partitioned Crossbar Topologies	* use the principle of merging and dividing routers.	Heuristic

## 4.2 FPGA based NoC synthesis Design Flow

Even though the problem of NoC synthesis has been treated since many years, only limited number of works have performed real implementation. In this section, we present the main NoC synthesis design flow in the literature based on FPGA execution.



**Figure 4.1. NoC synthesis on FPGA [87]**

The objectives and the constraints specification represents the first step of the NoC synthesis flow presented in Figure 4.1. The library of the routers and the characteristics of the application are also inputs to the workflow. During the step of Network generation, the user changes the values of the aimed frequency, the maximum number of routers and the link width. For each configuration, the algorithm should find the optimized NoC topology satisfying all the user constraints. The generated topology is then simulated before the step of

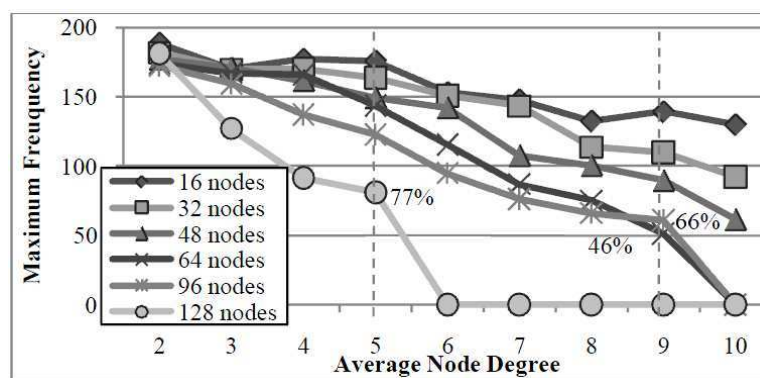
implementation. In this methodology, authors have chosen to vary the maximum number of used routers in order to deal with the complexity of the general problem. This exploration is limited by the number of the cores in the design; in fact the authors supposed that the number of routers should not overcome the number of cores.

X.Li and O.Hammami have proposed a new NoC synthesis methodology with FPGA emulation. The used workflow is summarized in Figure 4.2. The user objectives, the core graph properties and the constraints are the input of this methodology. Depending on the used core graph, the space of exploration is defined in function of the number of nodes and the number of switches in each stage. After the generation of the initial population, different NoC architectures are constructed. For each proposed design the total area of the NoC is determined referring to the used switch library. At this level, a TLM simulation is performed to measure the performance of the design. If the solution is not satisfying the user constraints, a penalty is automatically added to the fitness function. The MOEA performs the different steps of genetic algorithm to select the best solutions at each generation. At the end of this exploration, a number of solutions are defining the Pareto solutions depending on the objectives of the user.



**Figure 4.2. 2D NoC synthesis workflow**

J.Lee et al have proposed in [88], an analytical model based on simple equations defining the dependence between the operating frequency of the design and the different parameters of the architecture like the number of nodes, the links properties and the used board. This model can predict the implementation result of a specific design on a chosen FPGA platform. Authors have used different families of FPGA (Virtex 2,4,5,6) and different architectures to create the predictive model. Figure 4.3 illustrates the variation of the frequency of the design when the number of nodes and the average degree change. When the number of nodes increases the operation frequency decreases, the average degree is also a very important parameter affecting the performance of the design.



**Figure 4.3. Prediction of the Frequency variation when the number and the average node degree of the benchmark change [88]**

A.Kumar et al have proposed an integrated flow in order to generate a highly configurable NoC suitable for FPGA implementation, this methodology is presented in Figure 4.4. The description of the complete architecture is performed at the high level of abstraction. In fact the VHDL of the processing cores and the NoC components are generated at the same level with their simulation models. In this workflow, a hardware description which is FPGA level HDL is also provided at the top level of this methodology. Handel-C will then generate the EDIF files from the VHDL files which are used together with the system level EDIF files during the step of place and route on FPGA. The P&R tool generates at the end of this workflow the bit file to be embedded on FPGA. The user can also create an ASIC design from the VHDL files.

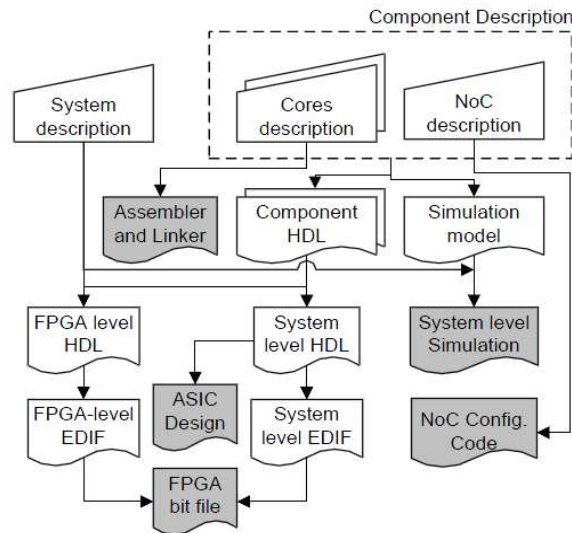


Figure 4.4. An FPGA design flow[19]

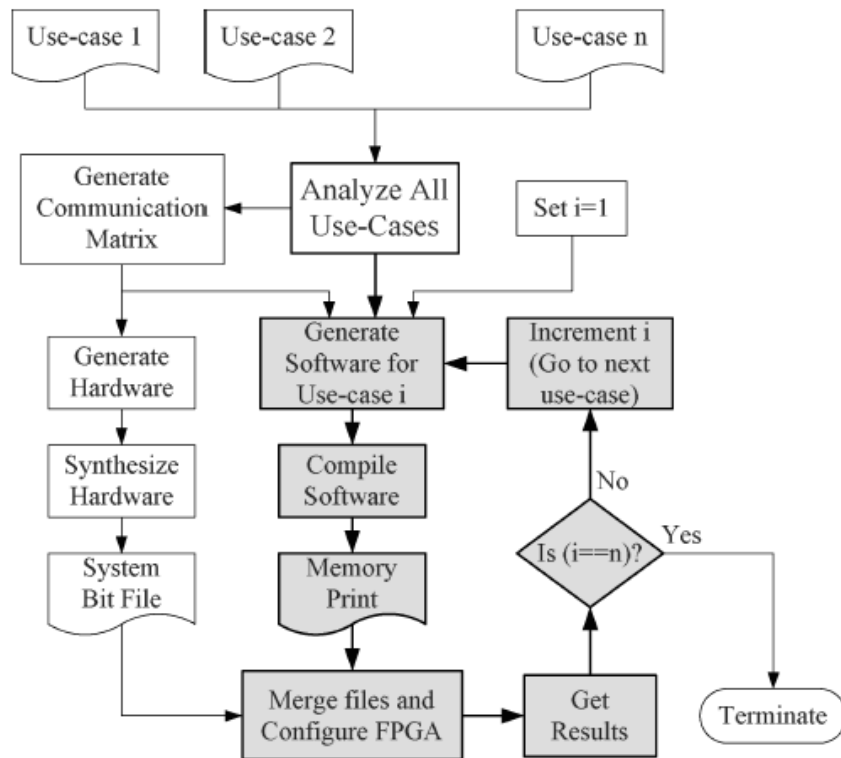


Figure 4.5. The overall flow for analyzing multiple use-cases: the software part is performed for each application, the hardware is performed only one time [89]

A.Kumar et al have developed a new methodology for 2D application specific NoC on FPGA presented in [89]. In this solution, authors proposed to generate a common hardware design suitable for different specific applications. Different Use-cases defining multiple coregraphs are the first input of this workflow. For each one, there is a specific hardware and software properties which should be respected. The study of all the use-cases allows the generation of a unique communication matrix satisfying all the input requirements. Thanks to

this matrix, the user can define the hardware description of the full design or the software of only one application. The grey boxes are determining the steps to be applied with all the Use-cases separately. The software part is studied for each input application in order to specify the hardware needs and properties. The configuration of the FPGA is updated via the bit file during each iteration. We notice that the hardware flow part is performed only one time. The bit file is updated to take in account the software specifications of all the applications. When the exploration of all the benchmarks is performed, the final multiple use-cases is generated. Such methodology is very interesting to optimize the MPSOC architecture for real multimedia devices using different functional modules.

### 4.3 Case study and performance evaluation results

We propose in this section our 2D NoC synthesis solution with Linear Programming. We create a mathematical model describing our application specific NoC. We present in this section our LP problem definition and the obtained results.

#### 4.3.1 Introduction to linear programming LP

The Linear Programming can be defined as the generation of a solution which maximizes or minimizes a linear objective function subject to linear constraints. This algorithm can be used in different real life applications like scheduling, minimizing the cost of a production and maximizing the profit. We present a simple example of a LP :

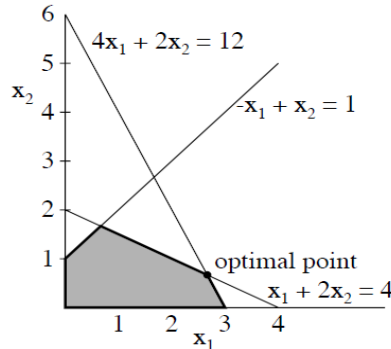
We suppose that we want to find the maximum of the sum  $F(X_1, X_2) = X_1 + X_2$  subject to the following constraints:

$$\begin{aligned} X_1 + 2X_2 &\leq 4 \\ 4X_1 + 2X_2 &\leq 12 \\ -X_1 + X_2 &\leq 1 \\ X_1 &\geq 0, X_2 \geq 0 \end{aligned}$$

In this problem there is only two unknown variables called the *decision variables* which are  $X_1$  and  $X_2$  and five constraints. The objective function  $F$  and all the constraints are linear. All the constraints are in form of inequalities. The two constraints  $X_1 \geq 0, X_2 \geq 0$  are called *nonnegativity constraints*, the other constraints are defined as *main constraints*. The function  $F$  to be maximized is called the *objective function*. As we have only two variables to



find, this problem can be graphically solved as illustrated in Figure 4.6. A graphical presentation of the objective function and the constraints, allows the definition of the feasible points presented by the grey region. It is easy to find the optimal point giving the maximum value of the objective function.



**Figure 4.6. LP graphical solution**

When the number of decision variables is more than two, there are different algorithms to solve the problem like the Simplex method.

#### 4.3.2 OPL Modelling and CPLEX solver

We use in this section the OPL Modeling language to describe our linear Programming problem. This modeling language is provided by IMB ILOG OPL. The user should create a model file \*.mod, describing the LP problem with the objective function and the different constraints. The initialization of the input data is performed in the input data file. These two files will be the input of the CPLEX solver to give the optimized solution. Applying the Simplex algorithm, the IBM ILOG CPLEX optimizer is a LP solver known to be efficient.

#### 4.3.3 Our LP Problem Definition

We propose in this section to model the NoC synthesis problem using the OPL language. A presentation of this problem can be defined as:

**Given:**

- A directed communication trace graph  $G(V,E)$ , where each  $v_i \in V$  denotes either a processing element or a memory unit and the directed edge  $e_k = \{v_i, v_j\} \in E$  denotes a communication trace from  $v_i$  to  $v_j$ .
- $N$  is the cardinal of  $V$ , representing the total number of cores,  $NE$  is the cardinal of  $E$  which is the number of edges in the graph.

- For each  $e_k = \{v_i, v_j\} \in E$ ,  $\omega(e_k)$  denotes the bandwidth requirements in bits/cycle.
- A library of routers  $Rout$ , for each  $r_i \in Rout$ ,  $In(r_i)$  denotes the number of the input router ports,  $Out(r_i)$  denotes the number of output router ports,  $Area(r_i)$  the area of the router,  $\Omega(r_i)$  is the peak bandwidth that one port can support bits/cycle ( we suppose that each router has the same peak bandwidth for all its input output ports).
- A set of the used routers  $R$  where  $R \subset Rout$ .
- For each core  $v_i \in V$ ,  $Req(v_i) \in R$  denotes a router request associated to  $v_i$  and  $Resp(v_i)$  denotes a router response associated to  $v_i$ .
- $E_r$  is the set of the used links between the routers.
- A set “long” defining a family of possible links with different lengths, we suppose that we have a fixed width for all the links. The length is to be defined in unit length.
- *Freq* : is the desired frequency for the NoC.
- *Timehops*: is the delay time needed for data to be routed through one router (ps).
- *Linkdelay* : is a delay time for each unit length specified in ps.

***The objective of the NoC synthesis problem is to :***

- Generate a NoC topology  $T(R, V, E_r)$
- Define the area and number of input output ports of each used router.
- Define the length of each link.

***Such that:***

- For each  $e_k = \{v_i, v_j\} \in E$ , there exists a route  $p = \{(v_i, r_m), (r_m, r_n), \dots, (r_c, v_j)\}$  that satisfies  $\omega(e_k)$
- The bandwidth constraints on the ports of the routers are satisfied.
- The total area of the NoC is minimized.
- The aimed NoC frequency is respected.

### **Linear problem formulation**

In this section, we present the linear problem formulation of the NoC synthesis problem. We present some used assumptions to simplify the problem formulation:

- We affect to each couple of routers related to a core  $v_i$  these value  $Req(v_i)=i$  and  $Resp(v_i)=i+N$ .

- Each edge  $ek=\{v_i,v_j\}$  must pass through the switch  $Req(v_i)=i$  and through the switch  $Resp(v_j)=v_j+N$ .
- $Req = \{0..N-1\}$  is representing the set of the request routers of all the cores.
- $Resp=\{N..2*N-1\}$  is representing the set of the response routers of all the cores.
- For each request router  $Req(v_i)=i$  we associate a set of routers destination in the set  $Resp$  defined as :  $r \in Dest(i)$  if  $\exists ek=\{v_i,v_j\} \in E / Resp(v_j)=r$ .
- For each response router  $Resp(v_j)=j+N$  we associate a set of routers origin in the set  $Req$  defined as :  $r \in Orig(j+N)$  if  $\exists ek=\{v_i,v_j\} \in E / Req(v_i)=r$ .
- $Lr,w$  is representing a link between the router  $r$  in  $R$  and the router  $w \in R_{\{r\}}$ .

### Decision Variables

#### *Independent variables*

- For each edge  $ek \in E$ ,  $r \in R$ ,  $w \in R_{\{r\}}$  and for each link  $l_{r,w}$ , we define the Boolean variable  $XRR[ek][l_{r,w}]$ . This variable is equal to 1 if the edge  $ek$  pass through the link  $l_{r,w}$ , 0 else.
- For each used router  $r \in R$  and for each family of router  $rout \in Rout$  we define the Boolean variable  $XRF[r][rout]$ . This variable is equal to 1 if the router  $r$  has the type family of  $rout$ , 0 else.
- For each used link  $l_{r,w}$  with  $r \in R$ ,  $w \in R_{\{r\}}$  we define the Boolean variable  $Long[l_{r,w}][long]$ , which is equal to 1 when the link  $l_{r,w}$  has a length equal to  $long$ , 0 else.

#### *Dependent variables*

- For each  $r \in Req$ ,  $ek \in E$  and  $w \in R_{\{r\}}$ , we define a boolean variable  $Xrrm[r][w]$  which is equal to 1 if there is a link between  $r$  and  $w$ , 0 otherwise. This variable is equal to 1 if there exists at least for any edge  $ek$  a variable  $XRR[ek][l_{w,r}]$  equal to 1, else  $XRR[ek][l_{w,r}]$  will be null. We can express this variable as:

$$Xrrm [ r ][ w ] = \max \{ XRR [ ek ][ l_{w , r } ] \}, \text{ for each } ek \in E$$

- For each router  $r=Req(v_i) \in Req$ , we define the integer variable  $Xrm[r]$  which is equal to the number of input ports of the router  $r$ . We suppose that each router  $r$  has an input port linked to the corresponding core  $v_i$  that is why we add the value 1 to the variable.

$$Xrm [ r ] = 1 + \sum_{w=0}^{2*N-1} Xrrm [ r ][ w ]$$

- For each  $r \in \text{Req}$ ,  $ek \in E$  and  $w \in R$ , we define a Boolean variable  $Yrrm[r][w]$ . This variable is equal to 1 if there exists at least, for any edge  $ek$  a variable, the value  $XRR[ek][lr,w]$  which is equal to 1, else  $XRR[ek][lr,w]$  will be null. We can express this variable as :

$$Yrrm [ r ][ w ] = \max \{ XRR [ ek ][ lr , w ] \}, \text{ for each } ek \in E$$

- For each router  $r = \text{Req}(vi) \in \text{Req}$ , we define the integer variable  $Yrm[r]$  which is equal to the number of output ports of the router  $r$ .

$$Yrm [ r ] = \sum_{w=0}^{2*N-1} Yrrm [ r ][ w ]$$

- For each  $r \in \text{Resp}$ ,  $ek \in E$  and  $w \in R$ , we define a Boolean variable  $Xrrs[r][w]$ . This variable is equal to 1 if there exists at least for any edge  $ek$  a variable  $XRR[ek][lw,r]$  equal to 1, else  $XRR[ek][lw,r]$  will be null. We can express this variable as:

$$Xrrs [ r ][ w ] = \max \{ XRR [ ek ][ lw , r ] \}, \text{ for each } ek \in E$$

- For each router  $r = \text{Resp}(vi) \in \text{Resp}$ , we define the integer variable  $Xrs[r]$  which is equal to the number of input ports of the router  $r$ .

$$Xrs [ r ] = \sum_{w=0}^{2*N-1} Xrrs [ r ][ w ]$$

- For each  $r = \text{Resp}(vi) \in \text{Resp}$ ,  $ek \in E$  and  $w \in R$ , we define a Boolean variable  $Yrrs[r][w]$ . This variable is equal to 1 if there exists at least for any edge  $ek$  a variable  $XRR[ek][lr,w]$  equal to 1, else  $XRR[ek][lr,w]$  will be null. Each router  $r$  has one link to its associated core  $vi$ . We can express this variable as:

$$Yrrs [ r ][ w ] = \max \{ XRR [ ek ][ lr , w ] \}, \text{ for each } ek \in E$$

- For each router  $r = \text{Resp}(vi) \in \text{Resp}$ , we define the integer variable  $Yrs[r]$  which is equal to the number of output ports of the router  $r$ .

$$Yrs [r] = 1 + \sum_{w=0}^{2*N-1} Yrrs [r][w]$$

- For each router  $r \in R$ , we define an integer variable  $Area[r]$  which is equal to the area of the used switch. Each used router is affected to a family router type  $Rout \in L$ . We define this variable as:

$$Area [r] = \sum_{w=0}^{cardinal(L)} XRF [r][w] * Area (w)$$

- For each link  $l_{r,w}$  defined in the NoC topology we associate the variable  $linklong[l]$  which is equal to its length. We define this variable as :

$$Linklong [l] = \sum_{k=0}^{cardinal(long)} Long [l][k] * k$$

- For each edge  $ek \in E$ , we define the total length of all the links defined for this edge as:

$$Linkedge [ek] = \sum_{l \in LRR} XRR [ek][l] * linklong [l]$$

### **Objective Function**

The objective of this NoC synthesis problem is to generate a NoC topology which minimizes the area of the NoC taking in consideration the area of routers and links subject to a given timing delay constraint. We define the total area of the NoC as:

$$Minimize(A = \sum_{r=0}^{2N-1} Area(r) + \sum_{l \in LRR} Linklong(l))$$

### **Constraints**

- C1 : For each router  $r \in R$  and for each edge  $ek=(vi,vj) \in E : r=Req(vi)$  or  $Resp(vi)$ , there is exactly one link between  $vi$  and  $wi \in \{Dest(vi) \cup Orig(vj)\}$

$$\sum_{w \in \{Dest(r) \cup Orig(vj)\}} XRR [e][lr, w] = 1, \text{ for each } r \in R, \text{ for each } ek=(v_i, v_j) \in E$$

- C2 : There is one link at most between 2 routers :

$$\begin{aligned} Xrrm [r][w] + Xrrm [w][r] &\leq 1, r \in \text{Re } q, w \in R_{-}\{r\} \\ Xrrs [r][w] + Xrrs [w][r] &\leq 1, r \in \text{Re } sp, w \in R_{-}\{r\} \end{aligned}$$

- C3: for each router if there is an input link related to an edge  $ek$  there is an output link for this edge.

$$XRR [e][lv_i, r] \leq XRR [e][lr, v_j]: e = (v_i, v_j) \in E, r \in R_{-}\{\text{Req}(v_i), \text{Resp}(v_j)\}$$

- C4: A link passes through a router if it is the origin or the destination of this link.

$$XRR [e][lr, w] = 0, \text{ for all } e = (v_i, v_j) \in E, r \in R \setminus \{\text{Req}(v_i)\} \text{ and } w \in R \setminus \{\text{Resp}(v_j)\}$$

- C5: the bandwidth of the edge must be less than the peak bandwidth of the chosen family router.

$$\begin{aligned} XRR [ek][lr, w] * \omega(ek) &\leq \sum_{f \in \text{Rout}} XRF [r][f] * \Omega(f) \\ e = (v_i, v_j) \in E, r \in R, w \in R_{-}\{r\} \end{aligned}$$

- C6: For each router  $r \in R$ , we should have exactly one family from the library Rout.

$$\sum_{f=0}^{\text{cardinal}(\text{Rout})} XRF [r][f] = 1, \text{ for each router } r \in R$$

- C7: The number of the input and output ports of each router  $r \in R$  should not exceed the corresponding number of the router family chosen.

$$Xrm [r] \leq \sum_{f=0}^{\text{cardinal}(\text{Rout})} XRF [r][f] * In(f), r \in \text{Re } q$$

$$Xrs [r] \leq \sum_{f=0}^{\text{cardinal}(\text{Rout})} XRF [r][f] * In(f), r \in \text{Re } sp$$

$$Yrm [r] \leq \sum_{f=0}^{\text{cardinal}(\text{Rout})} XRF [r][f] * Out(f), r \in \text{Re } q$$

$$Yrs [r] \leq \sum_{f=0}^{\text{cardinal}(\text{Rout})} XRF [r][f] * Out(f), r \in \text{Re } sp$$

- C8: for each router  $r \in R$ , the input flow should not exceed the output one.

$$\sum_{e \in E} \sum_{w \in R_{-}\{r\}} XRR [e][lw, r] * \omega(e) \leq$$

$$\sum_{f=0}^{\text{cardinal}(\text{Rout})} XRF [r][f] * Out(f) * \Omega(f), e \in E$$

- C9 : for each link  $lw, r$  in LRR we should have one only one length:

$$\sum_{long \in long} XRR [l][long] = 1, l \in LRR$$

- C10: When a link does not exist in the generated topology, its length will be equal to 0.

$$\max \{XRR[ek][lw, r]\} \leq \sum_{k \in long} Long [lw, r][k] * k, \text{ for each } lw, r \in LRR$$

- C11: The length of one link should not exceed a maximum value.

$$\sum_{long \in long} Long [l][long] * long \leq \max link, \\ l \in LRR$$

- C12: The delay of time depending on the frequency of the NoC, should not be exceeded for each path. We take in consideration routers and link delay.

$$\left( \sum_{l \in LRR} XRR [ek][l] - 1 \right) * timehops + \\ linkedge [ek] * linkdelay \leq 1 / lat \max,$$

#### 4.3.4 Experimental Results

Even though it is possible to present and before the experimental results the complete resolution trace of the LP solver, this trace is based on an automatic execution of the Linear Programming Solver. This technique is similar to all others used in Linear Programming resolution. Observing the intermediary resolution stages does not have any effect on the original model.

We test our program using the modelling language OPL to create the mathematical model and the tool CPLEX (version 12.2.0) to solve the generated problem. The used machine has a dual processor Pentium(R) Dual-Core CPU E6500 @ 2.93GHz with a 2G memory. We need as an input file a \*.data file where we define the properties of the core graph: the number of nodes, the different edges and the bandwidth of each edge. We define also the library of routers. In our case we have 64 possible configurations including the switch 0 having the number of input output ports and area equal to 0. To avoid the case of giving solutions with routers having one input port and one output port or a router with only one input port or one output port we add to the library 3 fictive router configurations. These 3 routers have an area equal to 0. In the output result, we will only keep routers with an area superior to 0. All the routers with an area null will be replaced by a simple link.

We use various core graph of different applications like 263 dec M3 dec, 263 enc mp3 dec, mp3 enc mp3dec, mpeg4, MPW, H.264 ( see Figure 4.7, Figure 4.8) . In all these graphs, the bandwidth values which are mentioned on the edges are in Kbits/s.

We choose to affect to each core  $v_i$  a couple of routers one for request  $Req(v_i)=i$  and one for  $Resp(v_i)=i+N$ . At the end, the generated solution is a combination of the remaining routers having a non null area. As we have already explained a router with an area equal to 0 is replaced by a simple link. We can see in the generated solution of the application 263 enc MP3 dec (see Figure 4.7) that only 4 switches are needed to ensure the different communications between cores. Switch 12 is the response router of the core 0 and the switch 0 is the request one. We propose to use properties of the technology 45 nm [90][87]. In Table 4.2, we present the summary of the used parameters. We suppose that all the links have the same width which is the minimum specified by ITRS2004-2007. We choose to affect to all the used switches the same maximum value of delay time, represented by the variable Timehops.

**Table 4.2. Semi conductor properties**

<b>Timehops</b>	253ps
<b>Linkdelay/1nm</b>	100ps
<b>Freq</b>	50Mhz
<b>Maxlink</b>	200nm

With reference to Table 4.3, we can compare the needed time to get the optimal solution of the presented multimedia applications. The execution time can give us a good idea about the complexity of core graph from the point of view of our solution methodology. If we compare the execution time of the 2 first applications, we can see that application 2 has less nodes and edges than application 1, but the solver takes longer time to find the optimal solution. In fact App (2), has a node with a degree equal to 5 whereas the maximum node degree in the App (1) is equal to 4. We can conclude that the complexity of the graph has a direct relation with the degrees of the nodes. App (5) takes 5058 seconds to find the optimal solution. This application has exactly the same number of nodes than App (4), the number of edges has only 4 more edges than application 4. If we analyse the core graph of H.264 we can see that the average degree is equal to 4.54. This core graph has many adjacent nodes with high degrees. This can be the reason of the complexity of the graph.



Table 4.3. Properties and execution time for the different benchmarks

Application	Number of nodes	Number of edges	Max(degree)	Number of router	Execution Time(s)
App(1) 263 dec MP3 dec	14	15	4	4	24.3
App(2) 263 enc MP3 dec	12	13	5	4	31.01
App(3) MP3 enc MP3 dec	13	12	3	4	11.89
App(4) MPEG 4	12	25	13	5	1512.88
App(5) H.264	12	29	10	10	5058.83
App(6) MWD	12	12	3	3	29

Even though it is difficult to compare our work to another one as we don't have the same mathematical problem modelling, we will try to compare the NoC topology size. For example, we propose to compare the NoC topology generated for the application 263 enc MP3 Dec in our work and in the work of Srinivasan et al [87]. The solution provided by our work needs 4 routers with a total number of ports is equal to 15 while the same application needs 5 routers with a total number of ports equals to 25. If we consider the area of routers from the Tezzaron library the area of our NoC is equal to 3897 (Nand 2x2) while it is equal to 8340 (Nand 2x2) for the other work. Our problem modeling reduces the NoC area to 47% compared to the one generated in [87].

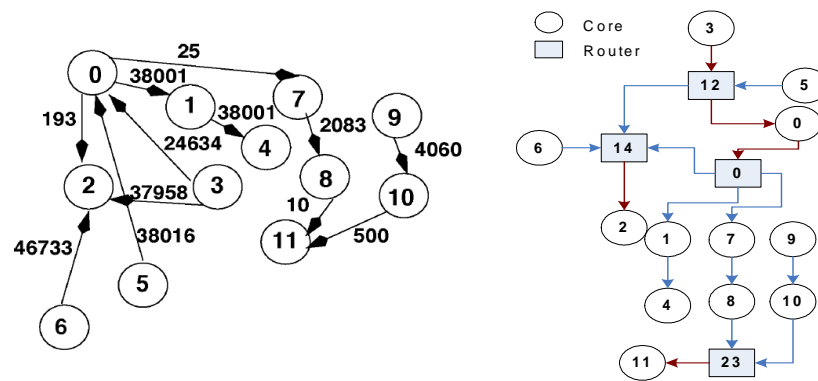


Figure 4.7. 263 enc MP3 Dec : coregraph (left), NoC topology (right)

Table 4.4. Routers Description 263 Enc MP3 Dec

$r \in R$	In(r)	Out(r)	$\Omega(r)$	Area( r)
0	1	3	$52 \cdot 10^{-6}$	1334
12	2	2	$44 \cdot 10^{-6}$	1448
14	3	1	$30 \cdot 10^{-6}$	611
23	2	1	$48 \cdot 10^{-6}$	504

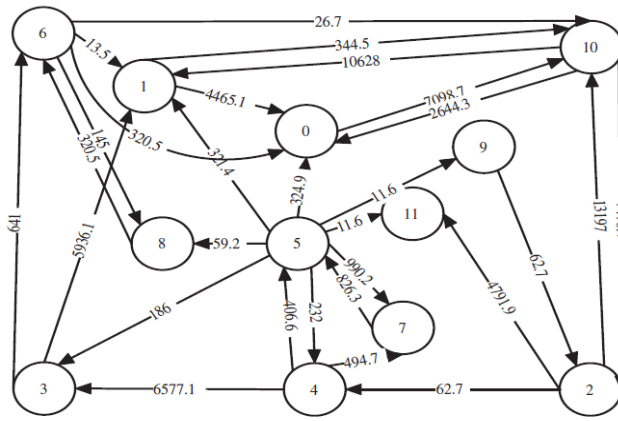


Figure 4.8. MPEG4 Decoder[78]

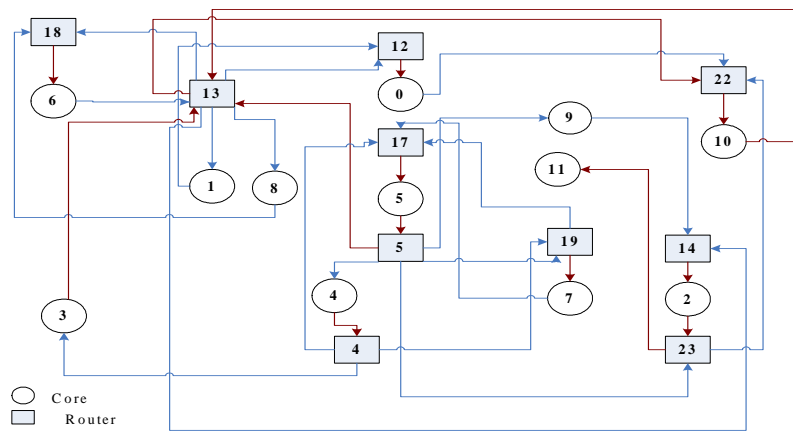


Figure 4.9. MPEG 4 Decoder NoC topology

Table 4.5. Routers Description MPEG 4 Decoder

$r \in R$	In(r)	Out(r)	$\Omega(r)(b/s)$	Area(r)(Nand2x2)
4	1	3	$52 \cdot 10^{-6}$	1334
5	1	5	$42 \cdot 10^{-6}$	2003
12	2	1	$48 \cdot 10^{-6}$	504
13	4	6	$48 \cdot 10^{-6}$	4464
14	2	1	$48 \cdot 10^{-6}$	504
17	3	1	$48 \cdot 10^{-6}$	504
18	2	1	$48 \cdot 10^{-6}$	504
19	3	1	$48 \cdot 10^{-6}$	504
22	3	1	$30 \cdot 10^{-6}$	611
23	2	2	$10^{-6}$	611

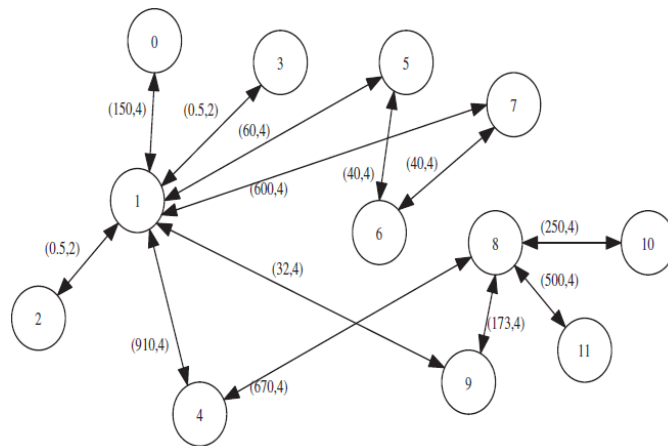


Figure 4.10. H264 Decoder

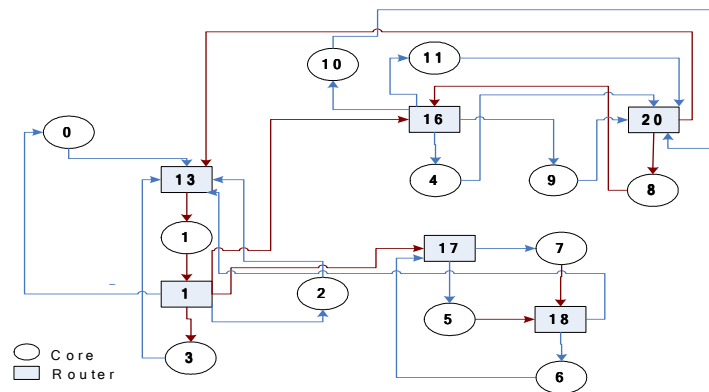


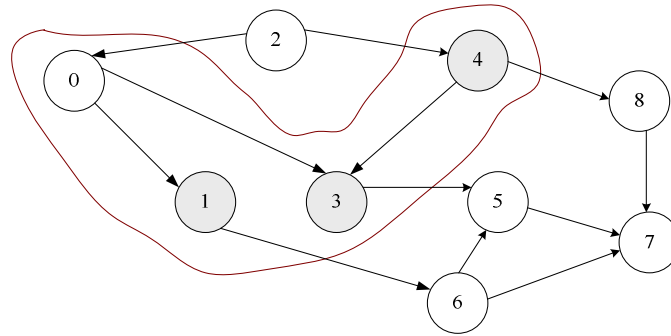
Figure 4.11. H264 Decoder NoC topology

Table 4.6. Router configurations H264 Decoder

$r \in R$	In(r)	Out(r)	$\Omega(r)(b/s)$	Area(r)(Nand2x2)
1	1	5	$42 \cdot 10^{-6}$	2003
13	5	1	$46 \cdot 10^{-6}$	828
17	2	2	$44 \cdot 10^{-6}$	1448
18	2	2	$44 \cdot 10^{-6}$	1448
20	4	2	$49 \cdot 10^{-6}$	2338
16	2	4	$46 \cdot 10^{-6}$	2252

#### 4.1 Theoretical Complexity issues and 2D challenges

We evaluate in this section the mathematical complexity of our algorithm. The NoC synthesis problem is known to be NP-Hard. The complexity of this problematic increases when the number of nodes increases. To deal with this complexity, we choose to perform a coregraph partitioning related to the spatial adjacency between nodes. In the step of pre-treatment, we define for each node a group of routers defining its space of exploration. In fact we reduce the design space exploration of a router to the set of the router destination of the node and their antecedent nodes. We present in Figure 4.12, the partition of the node 0: the request router of the node 0 can be connected to the response routers of its destinations (node 1, node 3) and the request routers of their antecedent, in this case the node 4.



**Figure 4.12. Coregraph partitioning**

We remind that  $N$  is the number of nodes in the coregraph and  $NE$  is the number of its edges. The complexity of the constraint  $C1$  is equal to  $\theta(NE * 2 * N * \text{card}(\text{max\_group}(r)))$ , where  $NE$  is the number of edges,  $2*N$  is the number of all routers in  $R$ , we define  $\text{max\_group}(r)$  as the maximum cardinal of the groups related to each router. We present the complexity of each constraint:

- $C2: \theta(2N * (2N - 1))$ ,  $N$  is the cardinal of  $V$
- $C3: \theta(NE * (2N - 2))$
- $C4: \theta(NE * (2N - 2))$
- $C5: \theta(NE * 2N * (2N - 1) * \text{card}(Rout))$
- $C6: \theta(2N * \text{card}(Rout))$
- $C7: \theta(N * \text{card}(Rout))$
- $C8: \theta(2N * (2N - 1) * NE * \text{card}(Rout))$

From this study, we can see that the complexity of C1 is reduced as the router is now able to be connected to switches from his group. For other constraints like C5 and C8 the complexity is still important. This model can be improved to reduce the constraint of all other constraints.

## 4.2 Conclusion

The NoC synthesis problem is the generation of NoC architecture optimized for a specific application and subject to a set of constraints. This problem is proven to be classified in the set of NP-Hard problems. For this reason, solving this problem with exact methods is not mathematically possible.

We presented in this chapter, the used methodologies in the literature to solve this problem. We can classify these methods into three families: the exact, the mixed and the heuristic methods. A solution is called exact, when only deterministic methods are used during all the steps of the workflow like the use of LP, Min-Cut, Djikstra... The use of such methodology can not solve the problems with a high degree of complexity. In the Mixed methodology, the user reduces the complexity of the problem by mixing the use of exact and heuristic algorithms. Fully heuristic methodology is the use of heuristic methods during all the steps of the resolution.

We propose a new NoC synthesis solution using the LP methods. We have modelled the NoC synthesis problem with OPL Modelling language in order to minimize the NoC area and the interconnect delay. In order to deal with the complexity of big coregraphs, we performed a partitioning based on the degree of adjacency between the different nodes. Thanks to this solution, we got application specific NoC free NoC topologies representing an optimized solution which respects our problem considerations.

Chapter 5 : NoC Synthesis Methodology for 3D ASIC  
Design

## 5 NoC Synthesis Methodology for 3D ASIC Design

The NoC synthesis problem is known to be NP-Hard. We have already discussed this problematic in chapter 4. The complexity of this problem increases when we propose to solve it with 3D IC design. In fact additional problematic specific to 3D design are added in 3D NoC synthesis problem like core to layer mapping, TSV area, symmetry... We propose in this chapter to study the existing 3D NoC methodologies and to propose our new solution to solve this problem.

### 5.1 3D NoC synthesis state of the Art

We present in Table 5.1, a summary of the existing 3D NoC synthesis methodologies. L.Benini team has proposed a 3D NoC synthesis workflow in order to generate a design power-performance efficient 3D NoC. The core to layer mapping and the floorplanning are taken as inputs. The 3D NoC synthesis problem is divided into sub problems: core to switch connectivity, switch to switch connectivity and switch floorplanning. The resolution of this problem is performed sequentially. The authors propose mixed algorithms based on the Min-cut partitioning to solve the two first problems while LP is used to find the positions of the routers. The 3D NoC synthesis problem is also an NP-Hard problem that is why, the partitioning of the coregraph was used in order to reduce the mathematical complexity. X.Jiang[91] et al have proposed another 3D NoC synthesis methodology. In fact they have used the Tarjan Algorithm to perform the core to switch connectivity step and the Min-Cut algorithm in order to partition the coregraph. The GA was used to solve the switch to switch connectivity and routers floorplanning. In this work the core to layer mapping and the initial core floorplanning are taken as an input to the workflow. In [92] W.Zhong et al have proposed a power performance 3D NoC synthesis methodology. In this work the clustering of the cores is performed after the floorplanning and routers can be only inserted in the white places. A new workflow based on stochastic algorithms was proposed by Zhou et al in [93][94]. In fact, authors have used the Simulation Allocation Algorithm SAL to find near optimal solutions for the traffic flow. The use of this methodology avoids the need to choose an order of treatment of the coregraph's paths. In [95] S.Yan et al have proposed a 3D NoC synthesis methodology based on the rip up and reroute procedure to generate a 3D NoC topology. The step of core to layer mapping was taken as the input of the initial problem. In order to optimize the generated NoC topology, a step of router merging is performed.

Table 5.1.3D NoC synthesis methodologies

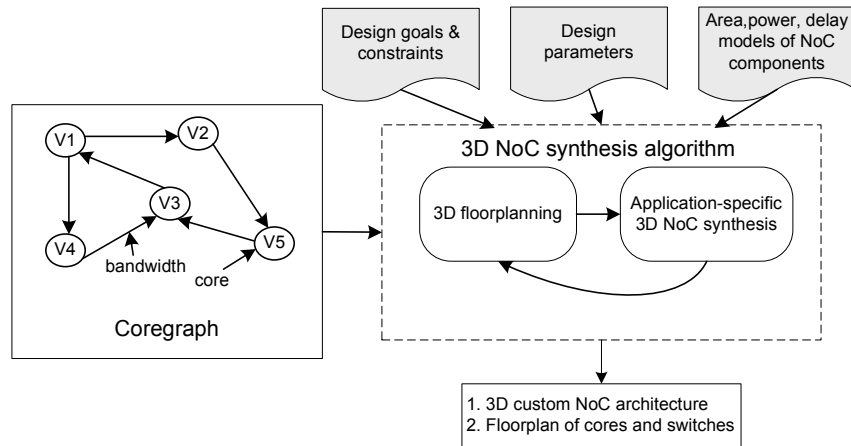
Team	Objective	Methodology	Comments
L.Benini Team[96][97] 2009, 2010	<ul style="list-style-type: none"> <li>• design power-performance efficient 3D NoCs</li> <li>• Main objective: Design NoC topology and determine switches positions.</li> </ul>	<ul style="list-style-type: none"> <li>• power and delay of both switches and links are taken in consideration</li> <li>• Heuristic core graph Partitioning</li> <li>• Core to switch connectivity(same layer) (Heuristic+Min-Cut)</li> <li>• Switch to switch connectivity with path computation (Heuristic+Min-cut)</li> <li>• Switch positions computation (Linear Programming)</li> </ul>	<ul style="list-style-type: none"> <li>• Core to layer mapping and 2D layer floorplan are taken as input</li> <li>• 65 nm technology</li> <li>• 3 layers : Processors in top an high layer, memories in the middle layer</li> <li>• Real implementation</li> </ul>
X. Jiang and T. Watanabe[91],2010	<ul style="list-style-type: none"> <li>• 3D NoC synthesis with Genetic Algorithms. Minimize power consumption in the NoC</li> </ul>	<ul style="list-style-type: none"> <li>• Tarjan Algorithm[98] : core to switch connectivity</li> <li>• CoreGraph Partitioning : Min Cut Partitioning</li> <li>• Switch to Switch Connection GA, path computation and flow control.</li> <li>• GA switch position</li> </ul>	<ul style="list-style-type: none"> <li>• Core to layer mapping and 2D layer floorplan are taken as input</li> <li>• Core can only be connected to a switch from the same layer</li> <li>• 65 nm low power technology and 3 layers</li> <li>• No real implementation.</li> <li>• No information about method of power estimation.</li> </ul>
W.Zhong et al[92] 2011	<ul style="list-style-type: none"> <li>• the power-performance efficient 3-D NoC topology for the application</li> </ul>	<ul style="list-style-type: none"> <li>• Cluster cores during 3D floorplanning</li> <li>• Use ILP to place switches and NIU in the 3D floorplanning</li> <li>• determine the connectivity across different switches using a power and timing aware path allocation algorithm</li> <li>• a min-cost max-flow based algorithm is proposed for Through-Silicon Via (TSV) assignment to minimize the link power consumption</li> </ul>	<ul style="list-style-type: none"> <li>• The algorithm is sequential</li> <li>• Use of the tool IARFP for the multi layer floorplanning with a weighted function</li> <li>• Insert switches and NIU in white spaces using the ILP</li> <li>• applying Dijkstra's shortest path Algorithm for path allocation</li> <li>• TSV assignment was using min-cost maxflow algorithms layer by layer</li> <li>• No real implementation</li> </ul>



			<ul style="list-style-type: none"> <li>• 3D layers Floorplanning and core to layer mapping included.</li> </ul>
<ul style="list-style-type: none"> <li>• P.Zhou et al[93],[94] 2010,2012</li> </ul>	<ul style="list-style-type: none"> <li>• find the best topology for the application, under different optimization objectives such as power and network latency, and determine the paths for traffic flows.</li> </ul>	<ul style="list-style-type: none"> <li>• Thermal aware floorplan based on B*-tree floorplan model</li> <li>• Use the Simulation Allocation Algorithm SAL to find near optimal solutions for the traffic flow.</li> <li>• Return information to the floorplanner to refine the result.</li> </ul>	<ul style="list-style-type: none"> <li>• Use SAL stochastic framework.</li> <li>• No real implementation</li> </ul>
<ul style="list-style-type: none"> <li>• S.Yan et B.Lin [95] 2008</li> </ul>	<ul style="list-style-type: none"> <li>• Use Rip-up and Reroute procedure for routing flows and Router Merging (RRRM) to optimize the network topology.</li> <li>• Minimize Power Consumption under performance constraints</li> </ul>	<ul style="list-style-type: none"> <li>• 3D core to layer mapping</li> <li>• 3D Floorplanning</li> <li>• Use the flow Ripup and Rerouting to generate the topology of the NoC</li> <li>• Use the Router Merging procedure to optimize the generated NoC.</li> </ul>	<ul style="list-style-type: none"> <li>• Floorplanning included.</li> <li>• Core to layer mapping heuristic.</li> <li>• Use power Modeling for links and routers using Orion</li> <li>• Use 70 nm technology.</li> <li>• No real implementation</li> </ul>

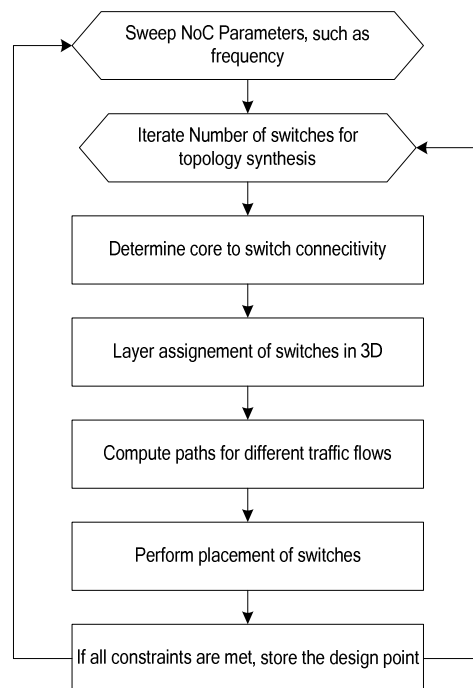
## 5.2 3D NoC synthesis design Flow

We propose in this section to detail the different 3D NoC synthesis workflow cited in the literature. In the work [96], the authors have presented a 3D NoC synthesis methodology which is summarized in the Table 5.1. In this methodology, we have 3 main input files. The first one is the communication specification file which is describing the coregraph application (connection, bandwidth, latency ...). The second one is called a Core specification file. In fact authors choose to treat manually the problem of core to layer assignment. The floorplan of each layer is also taken as the input of the NoC synthesis problem. To take in consideration the 3D technology specification, they used a third input file including the maximum number of allowed TSV across adjacent layers. In this work, the NoC synthesis problem is restricted to the NoC topology synthesis and the switches placement.



**Figure 5.1. 3D Design flow**

The obtained results were compared to mesh NoC topology and presented a large interconnect power reduction with an average of 38% and a latency reduction with an average equal to 25%. To have a complete 3D NoC synthesis problem, it would be better to include the problem of core to layer assignment and the problem of floorplanning in the NoC synthesis problem. With these initial manual consumptions this methodology can be classified as a mixed one, including heuristic and deterministic algorithms. This work is a generation of a specific NoC with a coregraph transformation. In fact, we can summarize the basic steps of the algorithm as presented in the next table.



**Figure 5.2. Algorithm Steps**

<p>Figure 5.3. Communication graph with bandwidth demands on the edges [97]</p>	<p>We present in the initial communication coregraph. Each vertex is representing a core and each link is representing the communication between the cores. Values on edges are the bandwidth values on each flow.</p>
<p>Figure 5.4. Partitioning Graph (PG) and the min-cut partitions[97]</p>	<p><b>Core to switch connectivity : Phase 1</b></p> <ul style="list-style-type: none"> <li>• Create a (PG) similar to the initial communication graph, but the weight of the edges defined as <math>h_{i,j}</math> (connection between the core <math>i</math> and <math>j</math>) <math>h_{i,j} = \alpha \times bw_{i,j} / max-bw + (1 - \alpha) \times min-lat / lat_{i,j}</math> <math>max-bw</math> is the maximum bandwidth value over all flows, <math>min-lat</math> is the tightest latency constraint over all flows and <math>\alpha</math> is a weight parameter</li> <li>• Partition the coregraph PG into the number of switches: cores in the same partition are connected to the same switch.</li> <li>• If for a particular core to switch assignment there is no possible solution meeting the constraints the coregraph will be scaled (SPG). We denote <math>max-wt</math> by the maximum edge weight in PG by</li> </ul> $l_{i,j} = \begin{cases} h_{i,j} & \text{if } (u_i, u_j) \in PG \& layer_i = layer_j \\ \frac{h_{i,j}}{\theta \times  layer_i - layer_j } & \text{if } (u_i, u_j) \in PG \& layer_i \neq layer_j \\ \frac{\theta \times max\_wt}{10 \times \theta_{max}} & \text{if } (u_i, u_j) \notin PG \& layer_i = layer_j \\ 0 & \text{otherwise} \end{cases}$ <ul style="list-style-type: none"> <li>• Partitioning and switch to layer assignment is applied on the SPG.</li> </ul>
<p>Figure 5.5. Scaling Parameter Graph (SPG)[97]</p>	<p><b>Core to switch connectivity : Phase 2 :</b></p> <ul style="list-style-type: none"> <li>• In this step cores can only be connected to switches in the same layer.</li> <li>• This phase can be used when a tight inter-layer link restriction is in place or when the technology restricts connection between adjacent layers.</li> </ul>
<p>Figure 5.6. LPG for two layers[97]</p>	

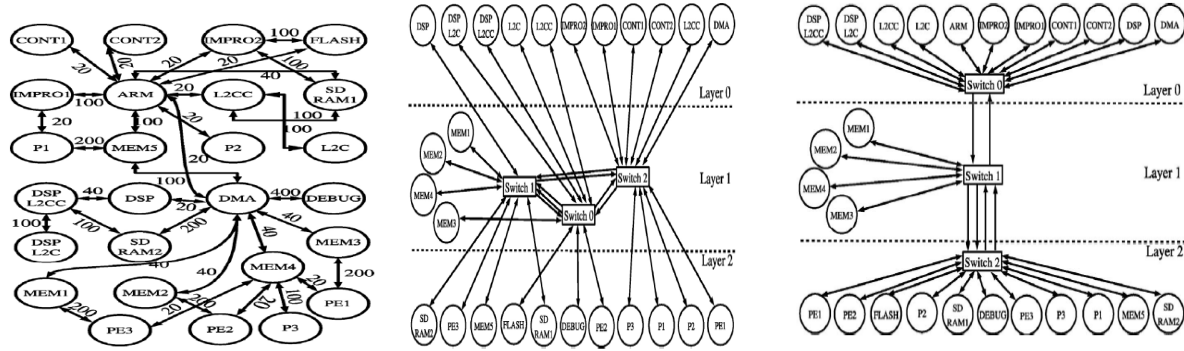


Figure 5.7. (left) D26\_media communication, (middle) NoC architecture phase 1, (right) NoC Architecture phase 2 [97]

We present in Figure 5.7, the obtained results for the presented methodology of the work of S.Murali et al. The proposed workflow is applied on the media benchmarks. The result of the first phase of the 3D NoC synthesis problem is presented in the figure of the middle where switches can be only connected to the switches from the same layer. A second phase of this work can be performed when the connections between switches from different layers are allowed.

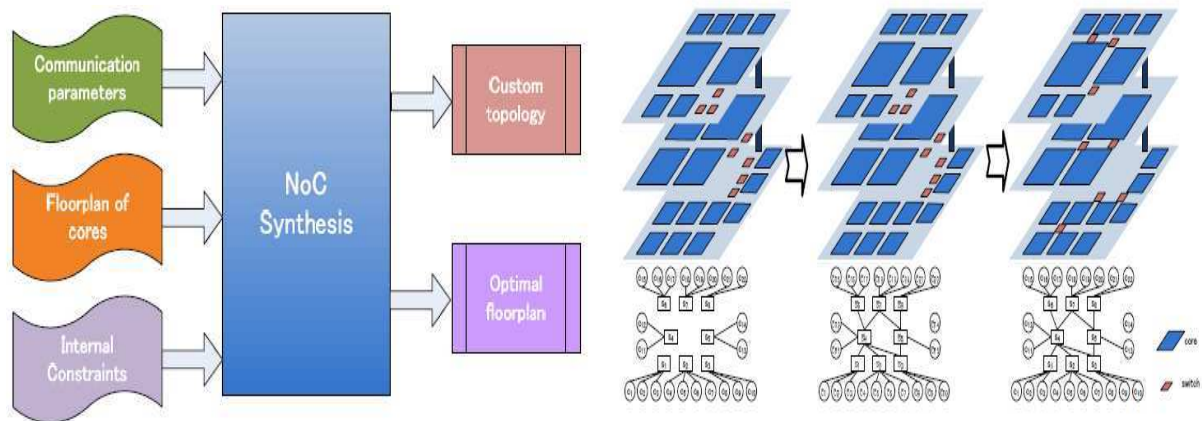
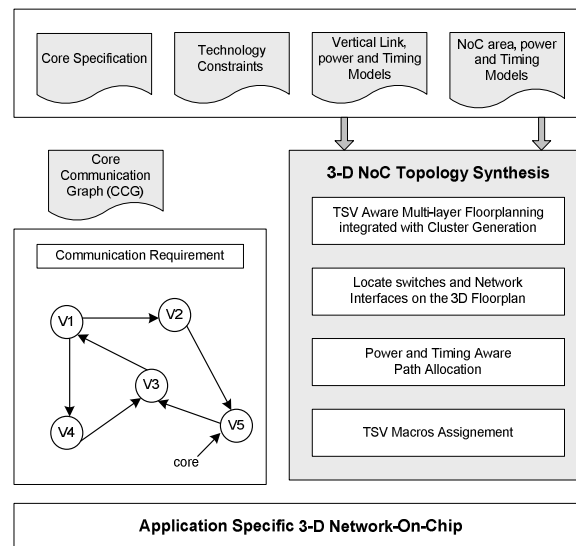


Figure 5.8. 3D NoC synthesis Design flow with GA, (a) left , (b) right [88]

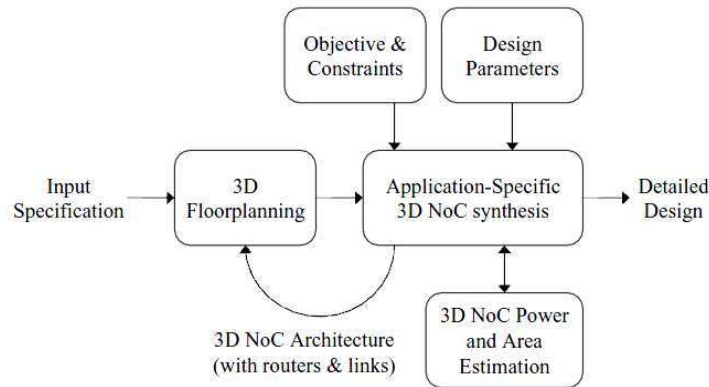
In [91] Jiang et al, proposed a workflow based on the Genetic Algorithms for the problem of the NoC synthesis. This solution is illustrated in Figure 5.8.(a). This workflow takes as input the communication parameters file which describes the Coregraph characteristics. The Floorplan and the core to layer mapping is also taken as inputs. The system analyzes the input data and automatically implements the synthesis process in three phases as we can see in Figure 5.8.(b). In fact, as cores and switches are already mapped to the different layers, the only aim of the first step of this methodology is the core to switch

connectivity. The core to switch connectivity step is applied for each layer separately. A core can only be connected to a switch from the same layer. The number of sub graphs will decide about the number of switches, as cores in the same partition are connected to the same switch [99]. In this work, authors applied Tarjan Algorithm [98] to find the strong connectivity sub-graphs. They have then used the methodologies presented in [99][96] to obtain the number of switches and core to switch connectivity. The second step of the algorithms which is the switch to switch connectivity is based on Genetic Algorithms to find the NoC architecture which is optimized for power consumption. Authors also used the GA methods in the last operation to define the optimized switch positions which minimizes the power length consumption.



**Figure 5.9. 3D NoC synthesis Design flow based on floorplanning[92]**

In [92] Zhong et al presented a sequential Design flow to solve the NoC synthesis problem which is illustrated in Figure 5.9. As this problem is known to be NP hard, authors divided it into 4 stages. In the first step which is the initial partitioning they applied a recursive min-cut bi-partitioning algorithm on Core Communication Graph taking in consideration the input communication file and the physical locations of the cores. They then used a multi-layer floorplanning tool IARFP to ensure a TSV Aware Multi-layer Floorplanning and Clustering. In the second step, an ILP based algorithm is proposed to place switches and network interfaces on the 3-D floorplan in white places. Authors used the min-cut max-flow algorithm to assign the TSV in order to minimize the link power consumption.



**Figure 5.10. 3D NoC synthesis workflow[95]**

We present in Figure 5.10, a 3D NoC synthesis methodology proposed by S.Yan et al. The core to layer mapping and the 3D floorplanning are taken as an input of this solution. The NoC topology is generated by using the Rip-up and reroute procedure. This work proposes to satisfy the coregraph edges following an increasing order. The floorplan of the chip is updated after the generation of the NoC topology.

### 5.3 Tezzaron Technology methodology

With the actual shortage of industrial tools for 3D design, Tezzaron Company provides a custom script based on the classical Place and Route encounter version 8.1. This flow is presented in Figure 5.11 . The basic functions of the 3DIC flow are:

- **Pre-synthesis logic simulation**

The user should verify the good functionality of his design at the HDL level. This step is performed using Test bench models. The designer can use the Modelsim tool to check the good behavior of the design without taking in consideration any timing constraint. This verification is the first one of the set of verifications during the 3D design. It is applied before the RTL synthesis operation.

- **RTL/logic synthesis**

The RTL (Register Transaction Level) synthesis is the translation of the input RTL description using the gate-level description. The output of this step is a generated Netlist which does not only respect the functionality of the input design but it also satisfies the user constraints (frequency, area...). The used cells in the generated Netlist are provided from the user input library. The RTL synthesis step can be performed using different tools like Design

Compiler from Synopsys and RTL compiler from Cadence. These tools generate several output files: a Verilog gate-level Netlist, timing constraint files and reports.

- **Post-synthesis logic simulation**

In this second step of verification, the same Test bench models already developed can be used to test the functionality of the generated Verilog Netlist. To perform this step, the models of standard cells are generated by the RTL synthesis tool.

- **Standard cell placement and routing**

The place and route step is the geometrical realization of the generated Gate Netlist which is also called Layout. The logic gates are placed following rows of equal high by the standard cell design style. That is why, all the standard cells from the same library have the same heights but with different widths. The connection between the cells called also routing is performed over the design since current processes allow several metal layers. Placement and routing can take in consideration the timing constraints already defined during the RTL synthesis step. At the end of these two steps several output files are generated by cadence encounter place and route tool like the geometric description (Layout) of the design with GDS format. The generated SDF (Standard Delay Format) description is including the gate and the interconnect delays.

Based on the presented operation, Tezzaon company provides a 3D IC flow by changing some steps in the classical 2D IC design methodology (see Figure 5.11). The first step of this workflow is to load the design. We use the libraries from ARM provided with Tezzaron Design kit Table 5.2. The pre-synthesis simulation represents the next step of the design which is performed using Modelsim. Once the design is verified, the step of RTL synthesis is applied using RTL Compiler from Cadence. We use Tezzaron technology with 130 nm Global Foundries low power standard Library. The RTL synthesis needs basically 3 main input files which are the Hdl files, the library files and the user constraints file. Timing constraints can be applied on the design to meet a specific frequency; these constraints should be defined in the user constraint file. The output of this step is a generated Verilog Netlist which will be the input of the Place and Rout step. This methodology proposes to use the encounter tool from Cadence to make the Place and Route of the design. After the placement

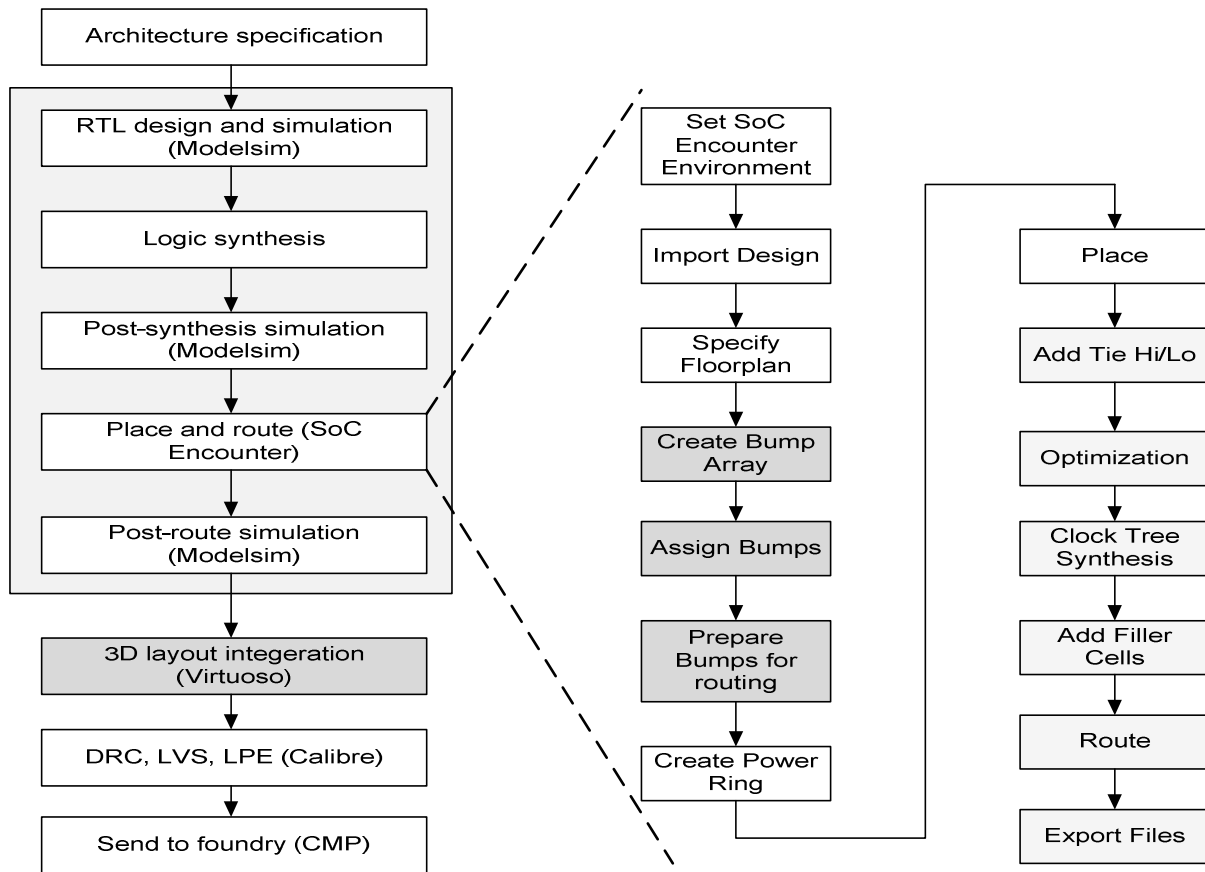


Figure 5.11. 3D-IC Automatic P&R using DBI and TSV

Table 5.2. ARM used in the Design Kit Tezzaron library

<b>ARM Standard-cells CORELIB LP-LVT : CSM013LP_LVT_SC_2007q2v1</b>	scx2_csm13lp_lvt_ff_1p65v_m40c	P/V/T = FF/1.65V/-40C
	scx2_csm13lp_lvt_ss_1p35v_125c	P/V/T = SS/1.35V/125C
	scx2_csm13lp_lvt_tt_1p5v_25c	P/V/T = TT/1.50V/25C
	scx2_csm13lp_lvt_ff_1p32v_m40c	P/V/T = FF/1.32V/-40C
	scx2_csm13lp_lvt_ss_1p08v_125c	P/V/T = SS/1.08V/125C
	scx2_csm13lp_lvt_tt_1p2v_25c	P/V/T = TT/1.20V/25C
<b>ARM Standard-cells CORELIB LP : CSM013LP_SC_2005q1v1</b>	ff_1v65_cm40	P/V/T = FF/1.65V/-40C
	ss_1v35_c125	P/V/T = SS/1.35V/125C
	tt_1v50_c25	P/V/T = TT/1.50V/25C
	ff_1v32_cm40	P/V/T = FF/1.32V/-40C
	ss_1v08_c125	P/V/T = SS/1.08V/125C
	tt_1v20_c25	P/V/T = TT/1.20V/25C



operation and before the routing step, Tezzaron proposes additional steps which are : creating Bumps Array, assigning signals to Bumps and preparing Bumps for routing. In fact, the user should create an Array of Bumps by defining the number of rows and columns, the pitch between the different Bumps and their format Figure 5.12.

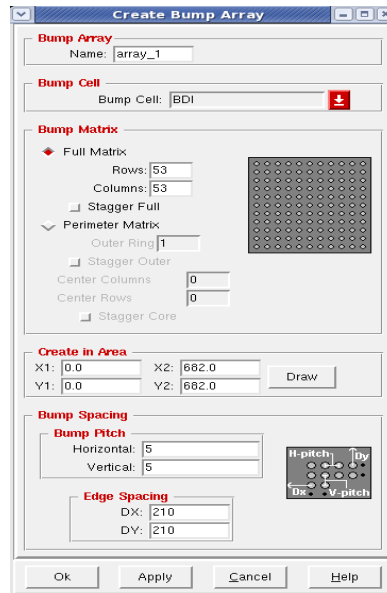


Figure 5.12. Create Bumps Array

Once the Array of Bumps is created, the user should assign the signals to the different Bumps. A vertical signal is affected to one Bump. The same signal should be affected to symmetric Bumps from the different layers. When a signal is assigned to a Bump, its color changes to blue, this step is illustrated in Figure 5.13.

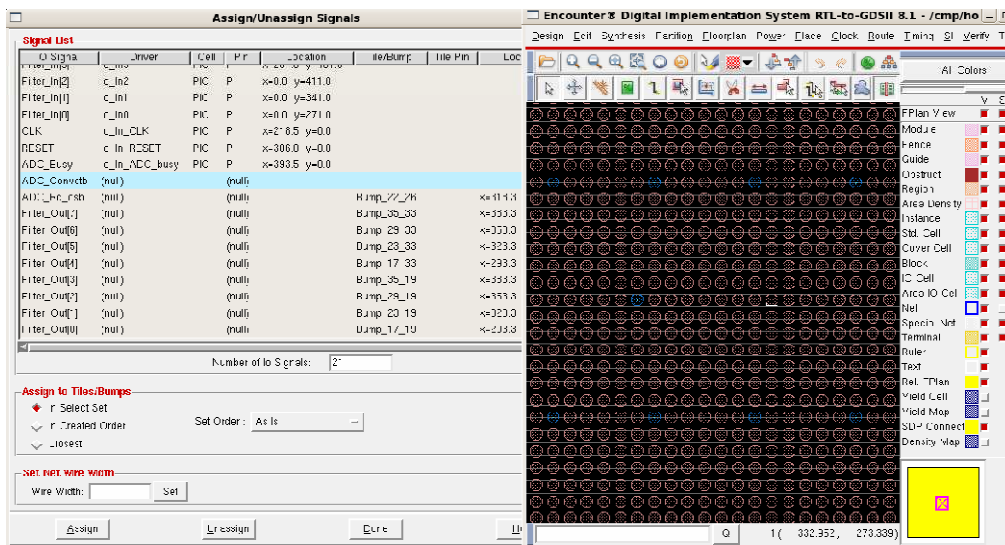
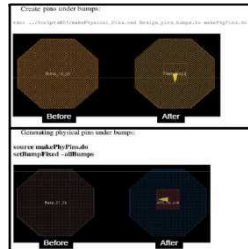


Figure 5.13. Signal to Bumps assignment

The main contribution of 3D Tezzaron methodology is the modification of the Bumps to enable their routing. In fact this company provides automatic scripts which add pins under the Bumps. We can see the added pins in red in the Figure 5.14. Thanks to this modification, Cadence tool can perform the routing of all the signals taking in consideration the vertical interconnections.



**Figure 5.14. Create pins under Bumps Tezzaron technology**

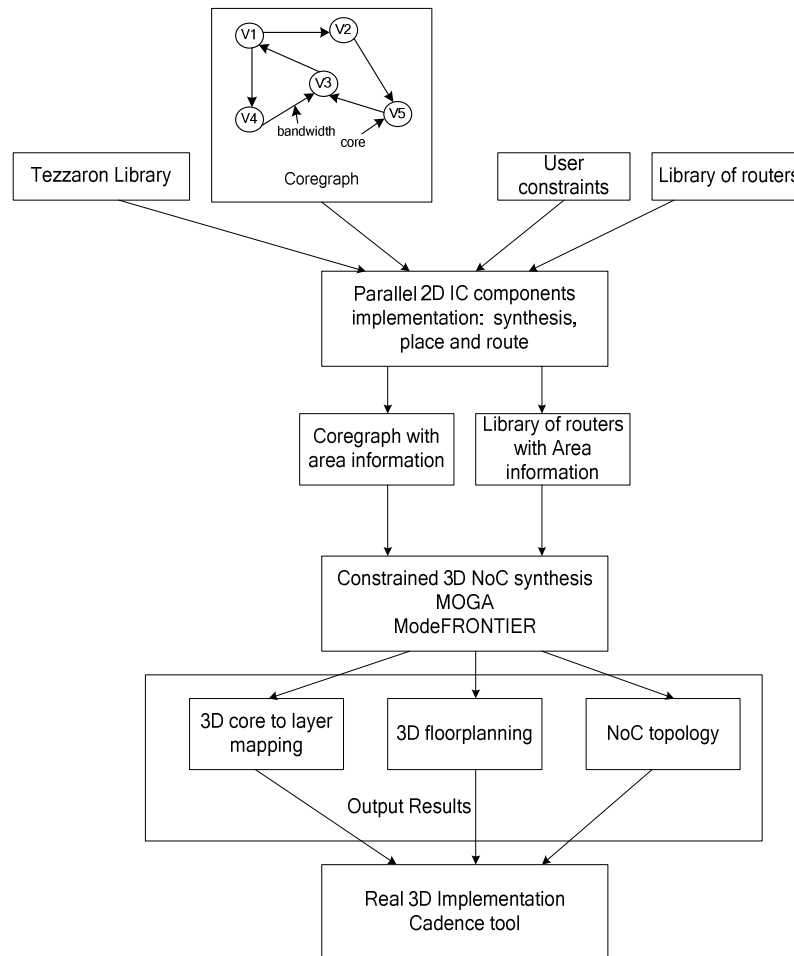
The presented 3D Tezzaron workflow should be applied on the two layers of the design. The whole 3D design is created during the packaging step. This solution is relatively easy to integrate especially with people who are familiar with the classical 2D IC design. The main challenge of this solution is the cost of design and verification time.

#### **5.4 3D NoC Synthesis with GA**

We propose in this work to solve the problem of 3D NoC synthesis with a whole parallel algorithm using the Evolutionary Genetic Algorithm. We believe that dividing the problems and especially treat them separately and sequentially can affect the final result. We propose in this work to solve the problem of the core to layer mapping, floorplanning and NoC topology at the same time.

- Our 3D NoC synthesis workflow

We present in Figure 5.15, our proposed 3D NoC synthesis methodology. In fact, the coregraph, the Tezzaron technology, the router library and the user constraints represent the input of our workflow. We apply at the first time the synthesis, the place and route using the 2D Tezzaron technology. Thanks to this step we can have an accurate idea about the area of each core, memory and the library routers. This information represents the input of our 3D NoC synthesis problem. In fact we propose in this work to solve the complete 3D NoC synthesis problem without dividing it into sub-problems. We propose to solve the problem of core to layer mapping, the NoC synthesis and the floorplanning using a MOEA. We choose to describe our 3D NoC synthesis problem using ModeFRONTIER tool.



**Figure 5.15. Our 3D NoC synthesis workflow**

- Our GA problem modeling

We propose to describe the 3D NoC synthesis problem using the GA methodology. We use the modeFRONTIER tool to describe the genome and to solve it. Our 3D NoC synthesis problem is a multi objective project with multiple constraints.

- Input File Constraints

For each individual, a NoC topology is generated thanks to the presented parameters. In fact, we connect each master and each memory to one random router, which is described by the variable Router. The different connections between the routers are represented by the vector RouterVector relative to each router. We choose to affect input constraints to generate NoC topologies with routers from the library. A feasible individual or ID must respect the input and the output constraints. To avoid topologies with circular paths we use the input constraint C1. With this constraint a router can have an output port to a router with a higher

index. We accept NoC topologies using routers from the library; in our case we have 64 possible router configurations from 1x2 router to 8x8 router. This constraint is described by the constraint C2.

- C1 : RouterVector[i][j] is equal to 0 if  $j \leq i$  , a router can have an output link only to a router with higher index
- C2 : The number of the Input ports and the Output ports should be less than the maximum value defined in the switch library.

- Output File Constraints

- In order to help the GA algorithm to find the feasible solutions, we define a set of output constraints C3-C6.:

- C3: The ERROR\_path error must be equal to 0 otherwise the generated NoC topology does not have a path for some demands in the coregraph.
- C4: The ERROR\_BP error must be equal to 0 otherwise in the generated NoC topology there is a switch which is not respecting the bandwidth constraint.
- C5: The ERROR\_overlapping error must be equal to 0 otherwise in the actual floorplan there is some overlaps between cores.
- C6: ERROR\_Ratio, We define the value of the aspect ratio of our chip which is a ratio between the width and the high of the chip.

We present in Table 5.3, the decision variables of our ModeFRONTIER project. For each core and memory, we define the variables X,Y and Z. X and Y are the coordinates of the lower left corner of the core while Z is the number of the layer which can be in our case 0 or 1. The constants Width and High are respectively the width and the high of the core when it is placed and routed in 2D. Each core must be connected to only one router; the index of this router is affected to the variable Router. We define the maximum value of used routers in our NoC synthesis problem. For each router we define the same X,Y and Z variables. The NoC topology is defined using the RouterVector which is a binary variable ; if there is a connection between router i and router i+1 RouterVector[i][i+1] is equal to 1, 0 otherwise.

**Table 5.3. MOEA Project parameters**

Core		
X (μm)	The abscise of the lower left corner of the core	Variable : [0 .. MaxX ] with a step of 50
Y (μm)	The coordinate of the lower left corner of the core	Variable : [0 .. MaxY ] with a step of 50
Z	The choice of the layer	Variable : [0..1] Tezzaron tech we have 2 layers

Width ( $\mu\text{m}$ )	The width of the core after a place and route in 2D	Constant
High( $\mu\text{m}$ )	The high of the core after a place and a route in 2D	Constant
Router	Each core is connected to only one router	Variable : [0...MaxRout]
<b>Router</b>		
X ( $\mu\text{m}$ )	The abscisse of the lower left corner of the router	Variable : [0 .. MaxX ] with a step of 50
Y ( $\mu\text{m}$ )	The coordinate of the lower left corner of the router	Variable : [0 .. MaxY ] with a step of 50
Z	The choice of the layer	Variable : [0..1] Tezzaron tech we have 2 layers
Width ( $\mu\text{m}$ )	The width of the core after a place and route in 2D (Tezzaron library)	Constant
High( $\mu\text{m}$ )	The high of the core after a place and a route in 2D (Tezzaron library)	Constant
RouterVector	This variable is a binary vector describing the connection between the routers. RouterVector[i][j] is 1 if there is a link from the router i to the router j , 0 otherwise. <b>Rq:</b> to simplify the explication we will call j a descendant of i if RouterVector[i][j] is equal to 1	Variable : [0..2 <sup>Maxrout</sup> ]

- Objective functions

- We propose in this methodology to solve a multi objective function. In fact, we propose to generate a NoC topology which is optimized for chip area and NoC diameter.
- Chip Area : The Chip area is one objective of our NoC synthesis to be Minimized
- Diameter: The Diameter of the NoC is the second objective to be minimized.

- Objectives and Constraints Computation

ERROR\_path:

---

**Algorithm 1 : ERROR\_path computation**

---

```

1.   ERROR_path=0
2.   For each e(orig,dest) in Edges
3.   do
4.       E={ }
5.       Routorig=e.orig
6.       Routdest=e.dest
7.       For i in Routorig+1 ..Maxrout
8.       do
9.           if RouterVector[Routorig][i]==1
10.            Then
11.                E=E∪{i}
12.            end if
13.        end for
14.        Level=0
15.        While (Routdest ∉ E & E!=F)
16.        do
17.            F=E

```

```

18.      For i in E
19.      do
20.          For j in i+1..Maxrout
21.          do
22.              if ( RouterVector[i][j]==1 & j ∉ E)
23.              Then
24.                  E=E∪{j}
25.                  Level=level+1
26.              end if
27.          end for
28.      end for
29.  end while
30.
31.  if Routdest ∉ E
32.      ERROR_path= ERROR_path+1
33.  else
34.      path[e]=1
35.  end if
36.  end for

```

---

The goal of the *Error\_path* constraint is to satisfy the required NoC topology. We present in algorithm 1 the different steps of this algorithm. With this constraint we can only verify the geometrical aspect of the NoC topology without taking in consideration the bandwidth constraints. The first step of the algorithms is to make an initialization to the *Error\_path* variable to 0. For each edge  $e \in \text{Edges}$  which is the set of the demands in the coregraph, we define the Routerig and the Routdest which are respectively the routers of the master and the slave of the edge  $e$  (see lines 5, 6). We define the set of routers  $E$  containing the routers which are connected to the Routerig (see lines 7-13). The set of routers  $E$  includes all the descendants of the router Routerig and their own descendants ( see lines 15-28) . The loop while will be stopped when the set  $E$  does not change any more or the router Routdest  $\in E$ . In the last step of the algorithm we verify if the Routdest appears in the descendants otherwise the *Error\_path* constraint will be incremented. When there is a path for all the edges of the coregraph, this constraint has the value equal to 0 if there is no path for any edge the value of the constraint will be equal to the total number of edges.

#### ERROR\_BP :

To guide the MOEA, we separate the geometrical aspect from the bandwidth limitation. In fact, we can have a path ensuring the communication between the origin and the destination of a specific demand in the coregraph, but this path cannot respect this demand's bandwidth. We need to verify when there is a correct path for a demand if this path can respect the bandwidth constraint. We can then conclude that the ERROR\_BP is directly

affected by the ERROR\_path. If the ERROR\_path is equal to 100%, which means that 0 path is found in the actual NoC topology, the ERROR\_BP is equal to 0. The ERROR\_BP is a quotient between the existing path in the NoC respecting the Bandwidth and those not respecting it.

---

**Algorithm 2 : ERROR\_BP computation**

---

```

1.   ERROR_BP=0
2.   For each e(orig,dest) in Edges
3.   do
4.     if path[e]=1
5.     Then
6.       Descendant[level]= set of the descendant routers at the last level defined in Algorithm 1 line
25
7.       Descendant[level-1]= set of the descendant routers at the level-1
8.
9.     While (level ≠ 0 & Affect[level-1]=1 )
10.    do
11.    For ri in Descendant[level-1]
12.    Do
13.    For rj in Descendant[level]
14.    Do
15.      If ( capacity Link_ri_rj ≥ e.BP && Affect[level-1]≠1 )
16.      then
17.        Affect[level-1]=1
18.        capacity Link_ri_rj = capacity Link_ri_rj-E.BP
19.      end if
20.    End for
21.  End for
22.  Level=level-1
23. End while
24.
25. If ( level ]≠ 0 or Affect[l]≠1 ( l≤level) )
26. Then
27.   ERROR_BP= ERROR_BP+1
28. End if
29. End if
30. End for

```

---

*ERROR overlapping :*

We propose to solve the floorplanning step simultaneously. In fact the generation of the coordinates of all the cores are done at the same time thanks to the variables (X,Y,Z). That is why we can have an overlapping situation. A feasible solution is the one with a value of Error\_overlapping equal to zero. The Error\_overlapping constraint is equal to the ratio of the overlapped area dividing the total area of the cores.

$$Erro\_overlapping = \frac{Overlapped\ area * 100}{total\ cores\ area}$$

### ERROR ratio :

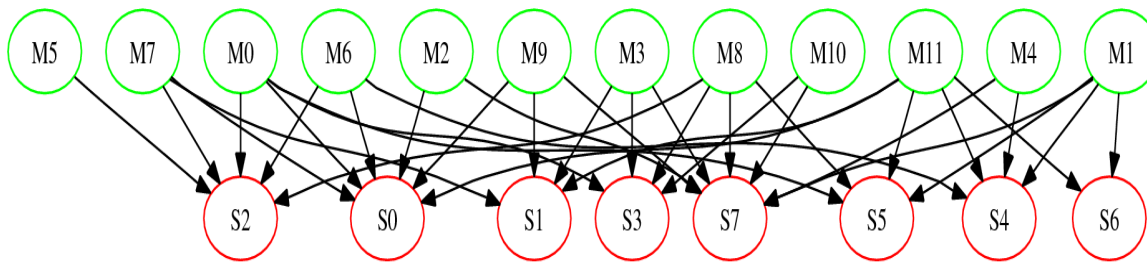
In order to determine the aspect ratio of our chip, we propose to use the Error\_ratio constraint. Thanks to this constraint we avoid to have a big difference between the width and the high of our chip.

$$\text{Minvalue} \leq \text{ERROR - ratio} = \frac{\text{Min}(\text{chipwidth}, \text{chiphigh})}{\text{Max}(\text{chipwidth}, \text{chiphigh})} \leq \text{Maxvalue}$$

## 5.5 Performance Evaluation Results

### 5.5.1 Case Study

With the shortage of information about the different choices when using the Genetic Algorithms, we propose as a first step of our experimental work, to perform a study of the different properties of the whole design. We use in this work the GA algorithms proposed by ModeFRONTIER. We choose to test our 3D NoC synthesis methodology on the coregraph presented in Figure 5.16. This coregraph is including 12 masters and 8 slaves with 36 demands.



**Figure 5.16. Coregraph 1 : 12 Masters 8 slaves**

When we use the GA to solve an optimization problem, we should choose different options like: the initial population, the GA solver, the size of the population, the number of generations... In this work we propose to solve the 3D NoC synthesis problem based on 3D Tezzaron technology using the MOEA methodology. For this, we present the different case studies, detailed in Table 5.4 , applied on our core graph.

- **Initial Population:** We explore different configurations from the initial population. In our work we are using a Multi objective Evolutionary algorithm with input and output constraints. We choose to test two initial populations: Constraint Satisfactory Problem CSP and Sobol. CSP is an initial population where only individuals which are respecting the initial



constraints are accepted. The use of Sobol initial population can guarantee an uniform distribution for each variable, individuals with initial violated constraints are not accepted.

- **Population size:** We use different population sizes to study their effects on the convergence of the algorithm.
- **Constraints:** We propose to change the limits of the accepted constraints, especially for the area constraint.
- **Constraints Priority:** The different constraints of our problem are interdependent. We suppose that the GA solver should satisfy all the constraints at the same order of priority, we will then multiply the constraints by weighted values to define a specific order of resolution.
- **GA:** We will choose two different Genetic Algorithm solvers which are NSGA-II and MOGA-II. These two algorithms treat the constrained problems differently, in fact the NSGA-II uses the non dominance concept to satisfy the constraints while MOGA-II treats that by adding a penalty to the fitness function when the ID is not respecting at least one constraint.
- **Number of Generations:** We explore the number of generations for the different case studies.

**Table 5.4. Case study different configurations**

	<b>Initial Population</b>	<b>Population Size</b>	<b>Constraints</b>	<b>Constraints Priority</b>	<b>GA</b>	<b>Number of Generations</b>
<b>Case 1</b>	CSP Constraint Satisfactory Problem	100	Area<100 Bandwidth<0 Overlapping<0 Ports<8	No	NSGA-II	500
<b>Case 2</b>	Sobol	100	Area<50 Bandwidth<0 Overlapping<0 Ports<8	No	NSGA-II	500
<b>Case 3</b>	Sobol	250	Area<60 Bandwidth<0 Overlapping<0 Ports<8 Ratio<0,75	Yes	NSGA-II	500

### **Case 1 results**

We use in case 1 the CSP as initial population. All the Ids of this population satisfy the input constraints which are the number of ports of the routers and the floorplan of the chip is included in the defined area. The population size is equal to 100 Ids and the used GA is NSGA II. We don't define any order of priority between the constraints of the problem.

The evolution of the error path and area constraints is represented in Figure 5.17 and Figure 5.18. Referring to these two curves, it is clear that the error path constraint was solved at almost the 2800 Ids, which means during the generation number 28, while the area constraint is not respected. We consider that a constraint is really respected when the GA provides a feasible solution for a time while. The path constraint is maintained almost stable around the value of 0, which means that there is no error path in the provided NoC. The evolution of the area constraint seems to be stable around the value of 140% without reaching the objective value which is 100%. This result is due to the first population choice. In fact, the input constraints are basically related to the number of the input and output ports, which has a direct effect on the NoC topology. We can conclude from this result that the initial population was not diverse enough to provide individuals with area constraint favour.

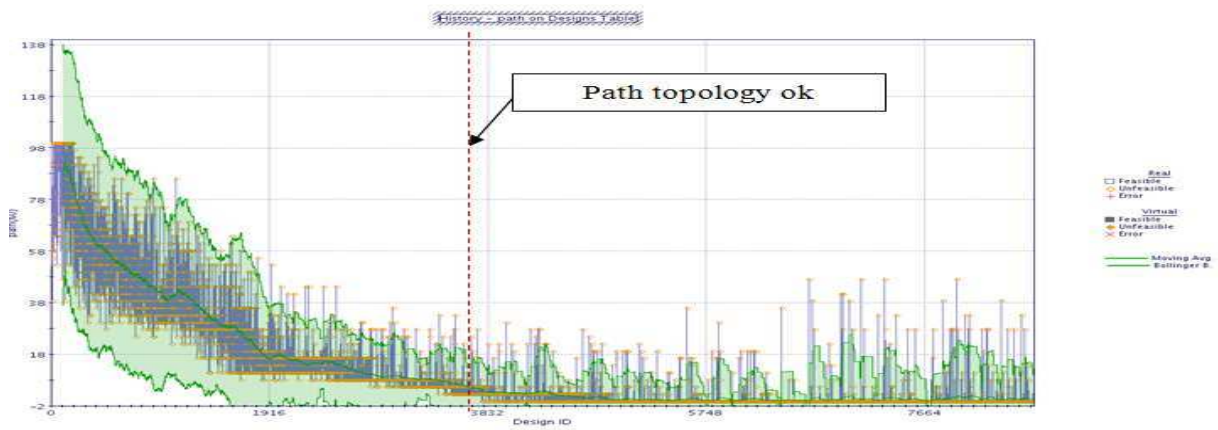


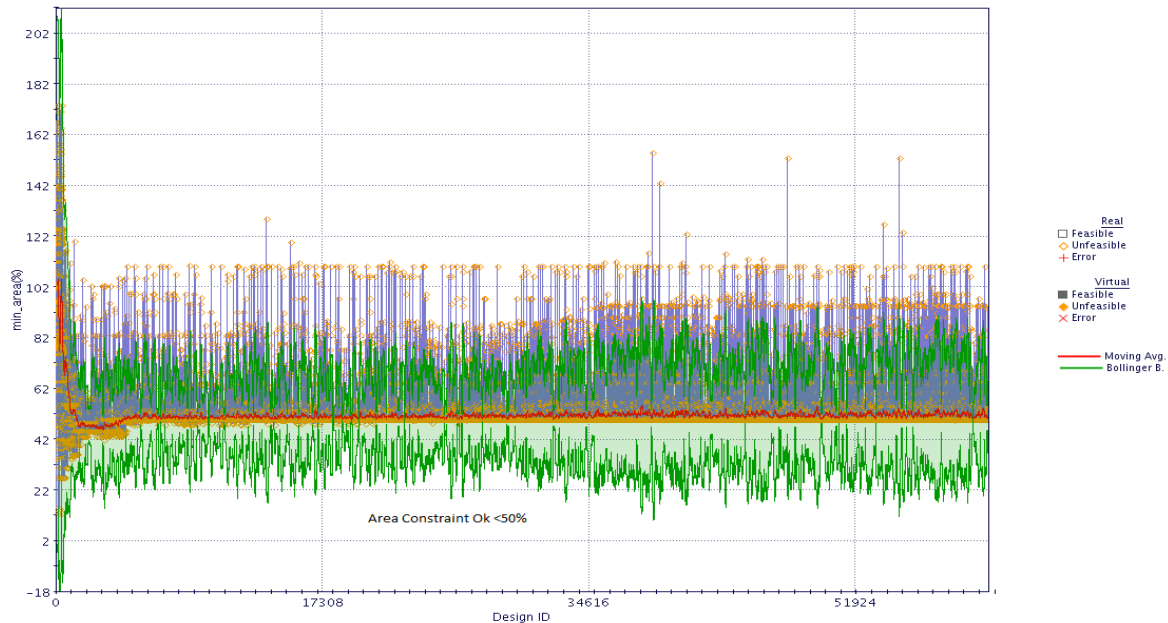
Figure 5.17. Case 1: Error Path constraint evolution



Figure 5.18. Case 1 : Area Constraint evolution

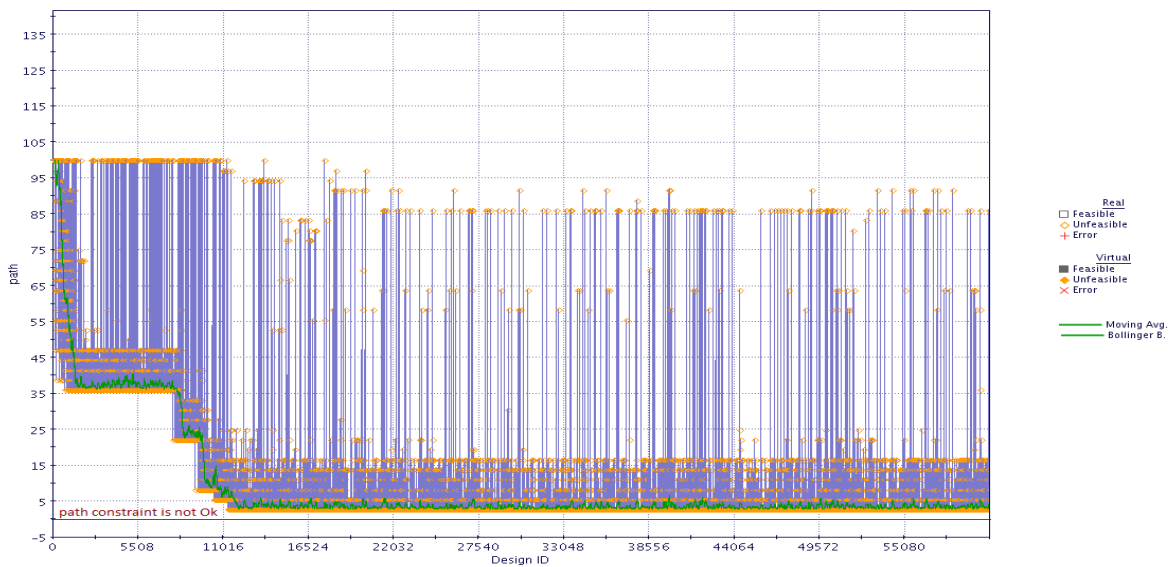
**Case 2 results**

In case 2, we change the initial population by using the Sobol algorithm proposed by ModeFRONTIER. This one ensures a better space distribution than a random generation. We keep the other parameters unchanged like the population size and the number of generations. The obtained results are presented in Figure 5.19 and Figure 5.20.



**Figure 5.19. Case 2 Min area constraint evolution**

History - path on Designs Table



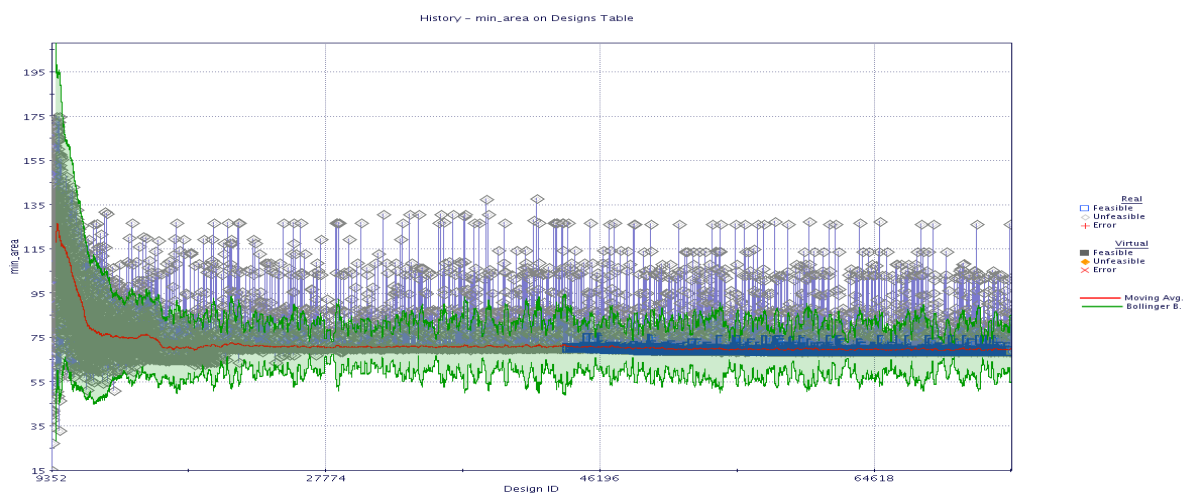
**Figure 5.20. Case 2 Error path constraint Evolution**

When we have changed the first population from CSP to Sobol the area constraint is satisfied but the path topology and the overlapping are not satisfied (see Figure 5.19 and

Figure 5.20). Even though we are using NSGA-II algorithm, we conclude referring to this experience that the tool ModeFRONTIER is using a weighted function with a sequential treatment of the constraints. In this case, ModeFRONTIER tried to solve the overlapping constraint before the path constraint but after more than 55 thousand iterations both constraints are not solved. The normalization of the constraints has a big effect on the evolution and the selection of the Ids. This configuration was not suitable to solve our 3D NoC synthesis problem. There is a shortage in term of information about the algorithms behind the ModeFRONTIER tool to manage a constrained problem. By our different experiences, we can conclude that this tool tries to satisfy each constraint separately and sequentially.

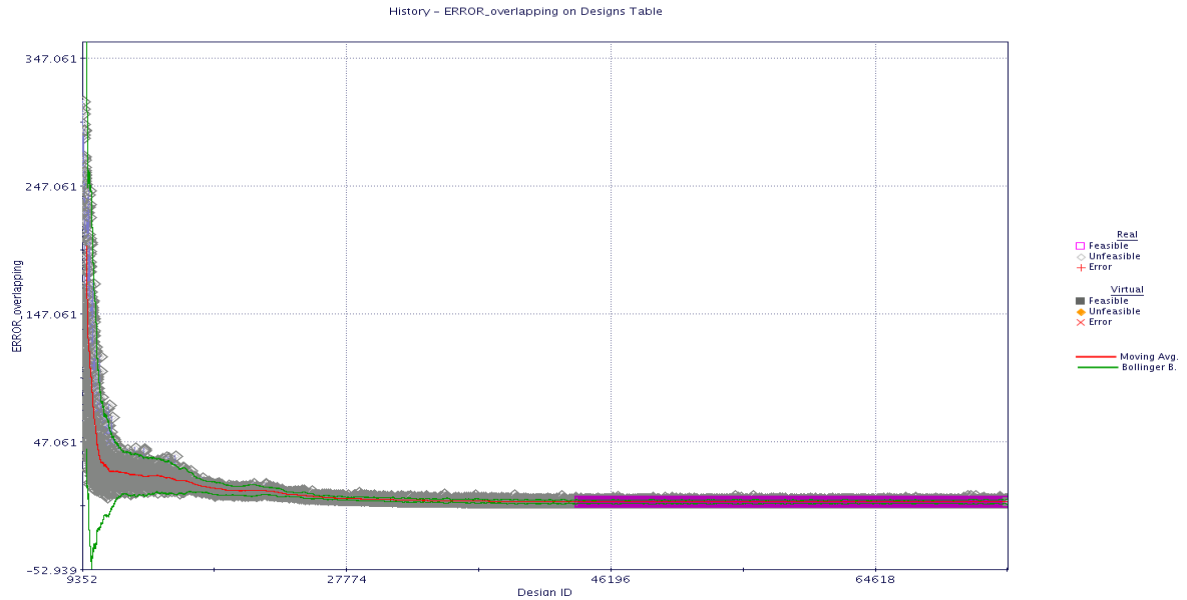
### Case 3 results

After testing different configurations, we choose to test the configuration presented in case 3. In this one, the Sobol algorithm is used to generate the initial population with a size of 250 IDs. We propose to guide the optimizer to solve the different constraints with a specific order. That is why we multiply them by different weights. We remark that ModeFRONTIER starts by solving the first constraint having a far value from the objective. That is why we multiply the path constraint by 1, the overlapping constraint by 0.3 and the min area constraint by 0.5. Thanks to this choice the solver should start by generating the NoC topology then the Min area constraint to finish by solving the overlapping of the floorplan. We present the obtained results in the Figure 5.21, Figure 5.22 and Figure 5.23.

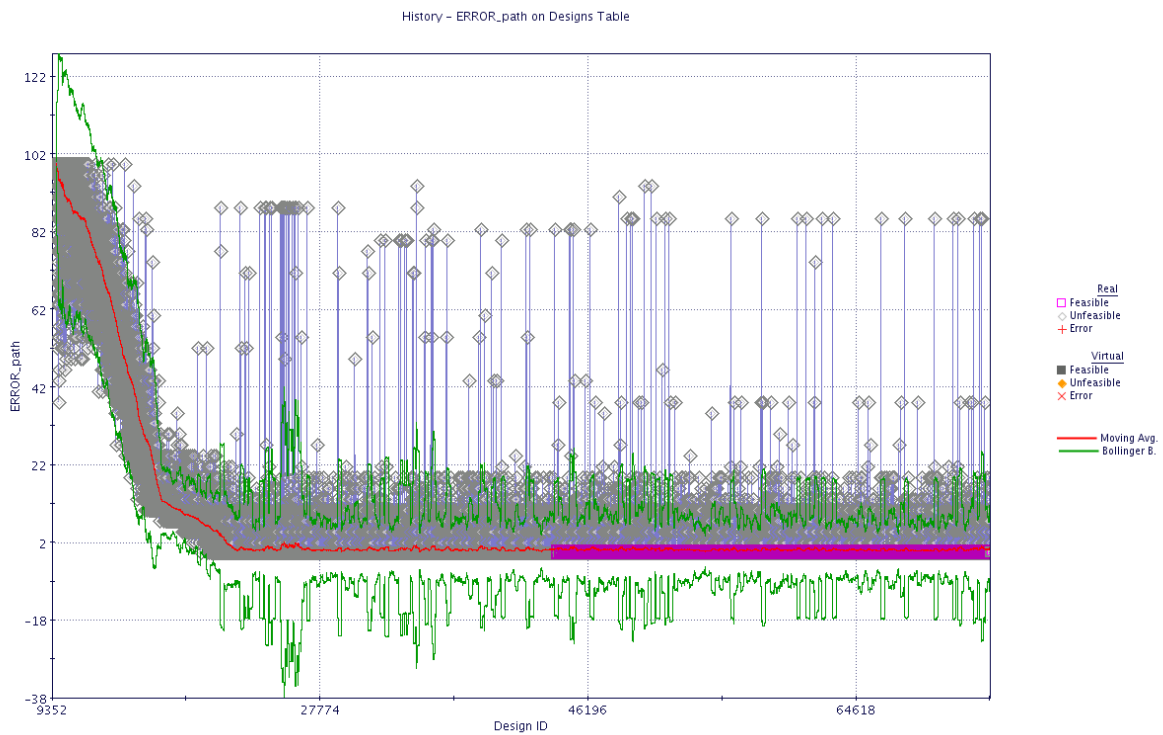


**Figure 5.21. Case 3 : Area constraint evolution**

Thanks to this configuration, all the constraints are satisfied. The feasible IDs are illustrated by blue squares in Figure 5.21. They are respecting all the input and output constraints. Thanks to the use of the weights, the Path constraints is respected at an early stage before the other constraints while the overlapping constraints takes the most important part time to be satisfied.



**Figure 5.22. Case 3 : Overlapping Constraint**



**Figure 5.23. Case 3 : Path constraint**

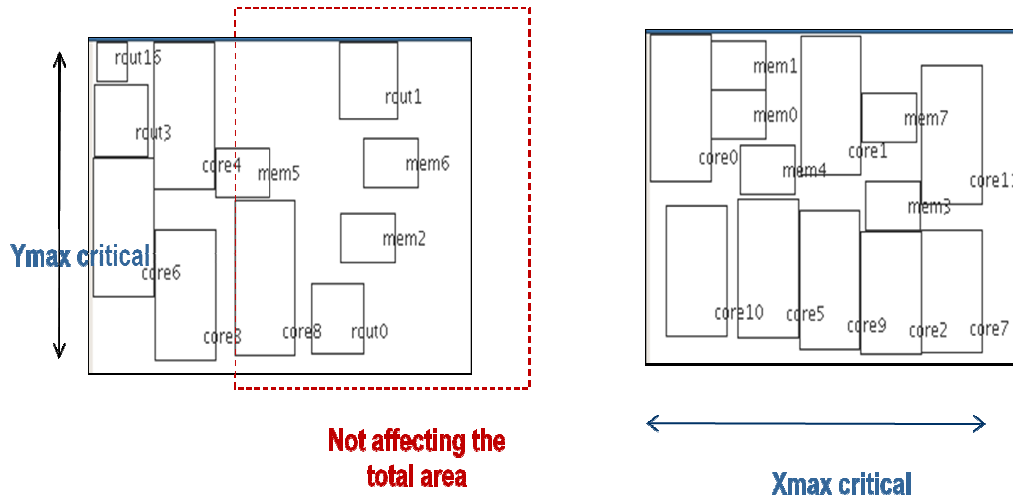


Figure 5.24. Our 3D NoC synthesis floorplan Solution

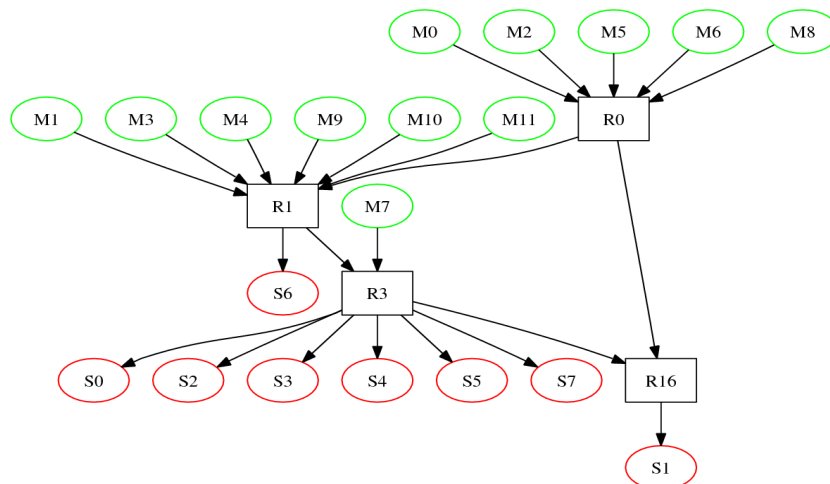


Figure 5.25. The optimized NoC topology

Thanks to our 3D NoC synthesis methodology, we have solved the core to layer mapping, the NoC topology and the floorplanning problems simultaneously. We present in Figure 5.25 and Figure 5.24 the illustration of these results. This example is a good validation of our 3D NoC synthesis methodology. In fact, starting from a full random combination, the solver has converged to an acceptable solution which is treating at the same time and with interaction the different constraints of the problem in order to minimize the area of the chip and the diameter of the NoC. The generated NoC topology which is describing the different connections between the masters and the slaves is respecting the initial core demands (see Figure 5.16) in term of connections and bandwidth. Even though the actual floorplan is not optimized, we can see that the chip width and high are almost optimal.

## 5.6 Conclusion

The 3D NoC synthesis problem is the generation of a NoC topology optimized for a specific application in order to optimize one or more objective functions with respect to the different constraints including the 3D IC design characteristics. We presented in this chapter the state of the art of the 3D NoC synthesis methodologies using heuristic, mixed and deterministic algorithms. In all the previous works and in order to reduce the mathematical complexity, authors tried to solve this problem by dividing it in sub problems then solving them sequentially and separately. The different steps like the core to layer mapping, floorplanning, NoC topology and routing are known to be NP-Hard that is why it is impossible to find a deterministic methodology to solve any 3D NoC synthesis problem.

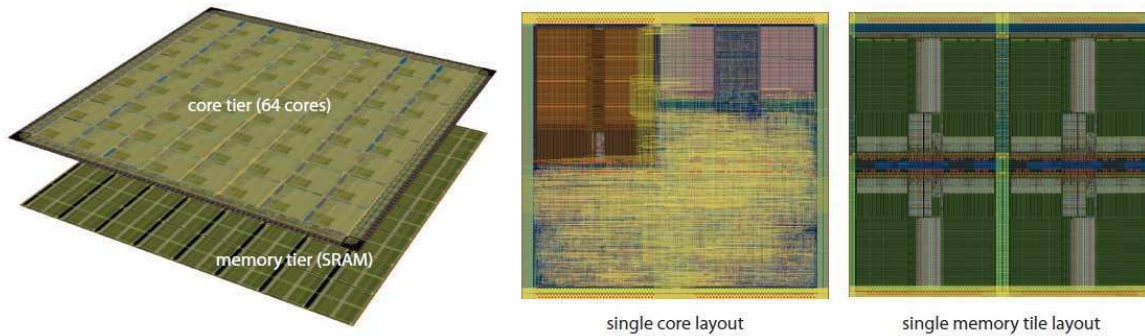
We proposed in this chapter a new 3D NoC synthesis methodology based on the 3D Tezzaron technique. The new idea of our work is to solve all the 3D NoC synthesis sub-problems at the same time. That is why we used the MOEA in order to generate the NoC topology and the floorplan of the different dies specific for each coregraph applications. We performed a design space exploration to experiment the different parameters of our genetic algorithm project in order to determine the suitable choices: initial population, number of generations... We tested our workflow methodology with different coregraphs. In our knowledge, this work is the first proposition to solve all the 3D NoC synthesis sub problems at the same time.

**Chapter 6 : ASIC Design Methodology for 3D NoC  
based 3D Heterogeneous Multiprocessor On Chip**



## 6 ASIC Design Methodology for 3D NoC based 3D Heterogenous Multiprocessor On Chip

### 6.1 3D Multiprocessor Architecture Homogenous



**Figure 6.1. Various layout views of the 3D-MAPS processor[48]**

We present in this section a 3D MPSOC real implementation realized by Heal et al [48]. The objective of this work is to implement a 3D-MAP (3D Massively Parallel processor with Stacked memory). The advantage of this design is to demonstrate the extremely large memory bandwidth available when using vertical 3D interconnects. In order to fabricate the 3D-MAPS, the authors used the 3D Tezzaron Technology which is based on 130nm process provided with global Foundries. In this work, the used TSV are manufactured in Via-first process and the chip is using two stacked dies with a face to face disposition. The thick of the thinned die is equal to  $12\mu\text{m}$  while for the thick one this value is equal to  $765\mu\text{m}$ . The physical implementation results of this 3D MPSOC is presented in Table 6.1. The global Foundries 130nm represent the used process technology. The size of the die architecture is equal  $5\text{mm}^2$ . The total vertical connections are equal to 47940 which mean that about 90% of the TSV are used to ensure the Power Ground connections. Each core needs 116 F2F vertical connections dedicated for the clock and the various signals which is a total of 7424 over the entire die. Figure 6.1 is the illustration of the 3D-Maps processor. The footprint of the core is equal to  $560 \times 560\mu\text{m}$ . A layout of a single memory tile is also shown: this one is composed of 4 memory banks of 1KB. The total memory capacity of this processor is equal to  $4\text{KB} \times 64$  which is equal to 256KB. In this work, the authors choose to create a layer for the cores and another one for the memories. We have 64 cores tiles in the upper die and 64 memory tiles in the other one. This design is run at the frequency of 277MHz.

**Table 6.1. Physical Design Summary [48]**

Process technology	Global Foundries 130nm
Die size	5x5 mm
Core footprint	560x560 $\mu$ m
Core-to-core pitch	570 $\mu$ m
PG 3D connections/core	668
Total PG 3D connections	42,752
Data 3D connections/core	116
Total data 3D connections	7,424
TSVs/IO pad	204
Total IO TSVs	47,940
Dummy TSVs	6,540
Total maximum IR-drop	78mV
Maximum operating frequency	277MHz

The simulation results of the different optimized Muticore benchmarks which were applied on the hardware 3D-MAPS architecture are presented in Table 6.2. This table presents the memory bandwidth in gigabytes per second (GB/s). Depending on the behavior of the application, the memory bandwidth can reach up to 63.8 GB/s which is more important than that of a modem Intel Core i7 processor[48].

This work represents an interesting case study of a real 3D MPSOC design implementation. In fact thanks to this work, the high bandwidth of the memory is proven to be a principle reason to use the 3D IC design. This architecture is based on the mesh topology which is a symmetric architecture linking the cores thanks to its short links. For this design, the authors have used homogenous cores and tile memories. This choice can avoid showing other faced problems when we implement a general MPSOC architecture like the core to layer mapping and floorplanning.

**Table 6.2. Architectural Performance Metrics[48]**

<b>Benchmark</b>	<b>Memory Bandwidth (GB/s)</b>
String_search	8.9
Matrix_multiply	13.8
Median	63.8
Aes_encrypt	49.5
Motion estimation	24.1
Histogram	30.3
Edge detection	15.6
K-means	40.6

T. Thorolfsson et al have implemented in [49] a 1024-point, memory-on-logic 3DIC FFT processor for a synthetic aperture radar (SAR). This work was based on the MIT Lincoln Labs' manufacturing process which is using 3 tiers called A, B and C. The MPSOC architecture is including 8PEs, one controller, thirty two SRAMs and 8 ROMs. The processing element is illustrated in Figure 6.2. This core is implementing the FFT Butterfly

with four floating point multiplies and six addition/subtraction units. We can see that this processor is based on the Butterfly architecture which is interesting to test with 3D IC design due to its long links. The whole FFT SAR MPSOC architecture is presented in Figure 6.4.

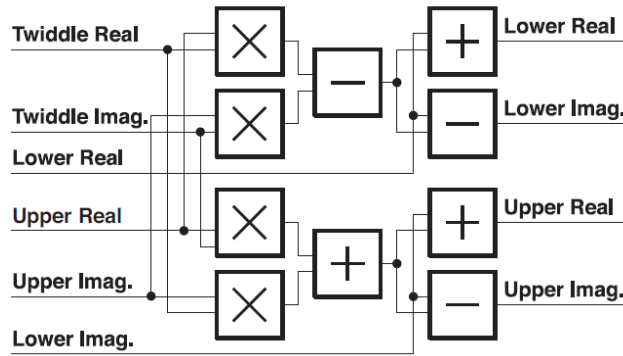


Figure 6.2. The structure of the PE [49]

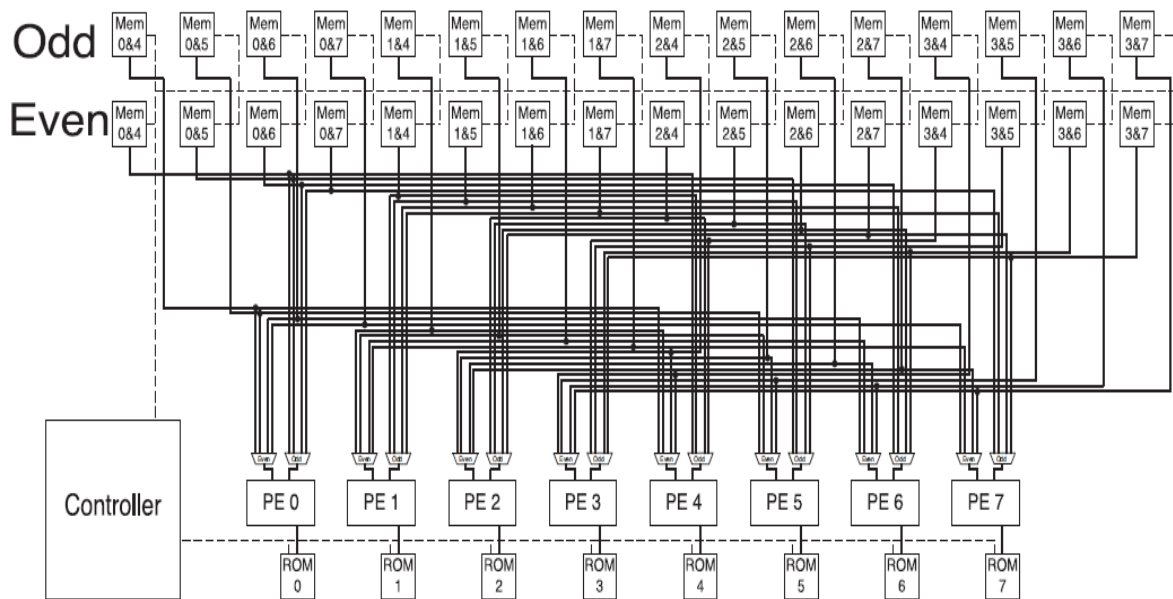


Figure 6.3. The SAR FFT processor architecture[49]

Figure 6.4 represents the complete 3D workflow used in this work. The 3D floorplanning and partitioning are the first steps of this flow. To perform these operations, the authors had the objective to get the memories as close as possible to the processing elements. That is why; they placed the memories with their interfaces in the middle tier, the obtained result is illustrated in Figure 6.5.

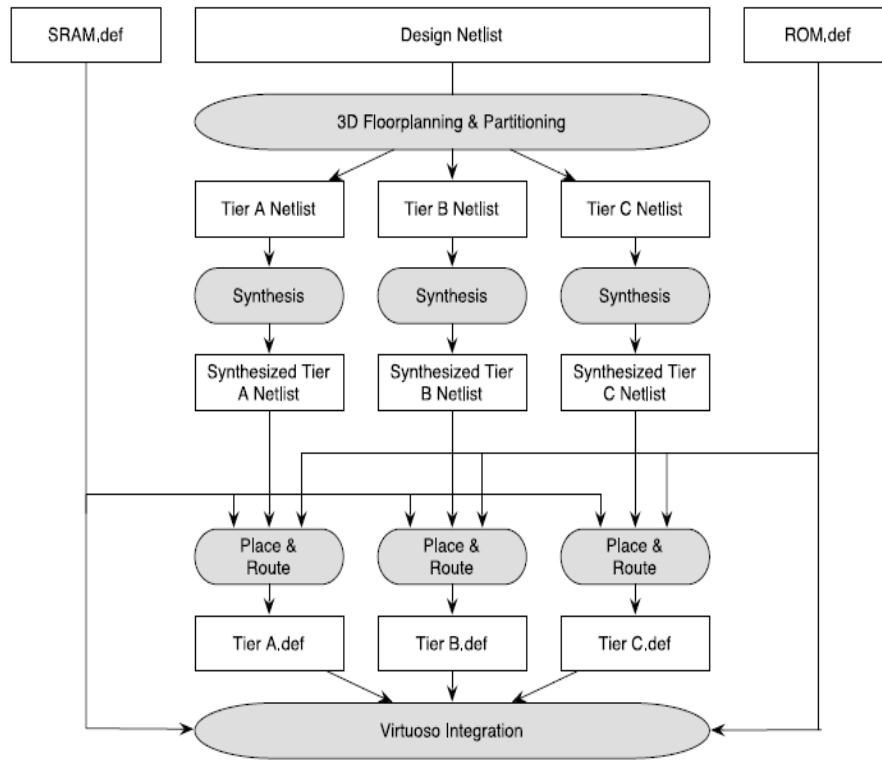


Figure 6.4. T. Thorolfsson 3D Design flow[49]

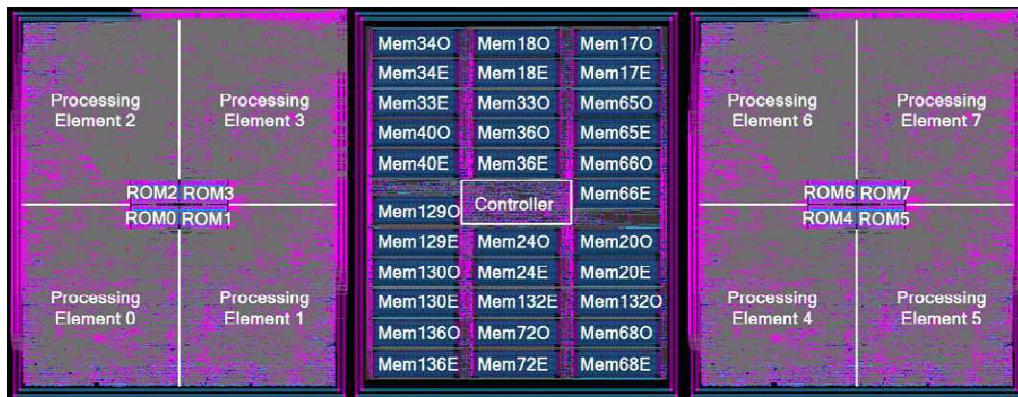


Figure 6.5. The SAR FFT floorplan[49]

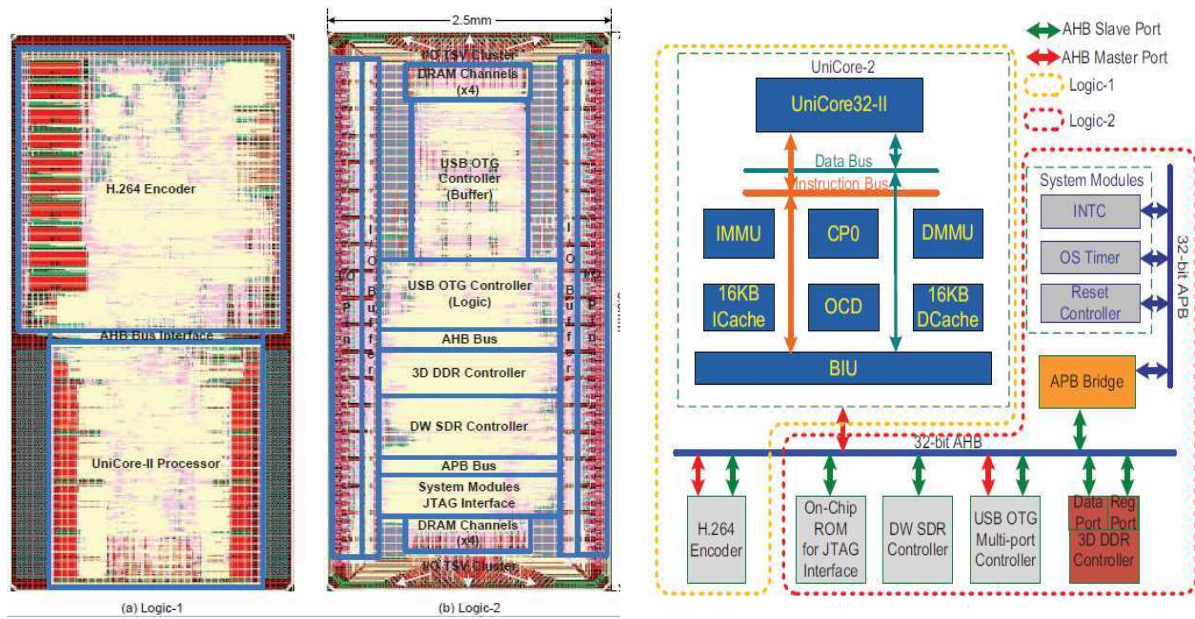


Figure 6.6. Schematic and layout view of 3D SoC H.264 Application [51]

T. Zhang et al have presented in [51] the 3D IC implementation of H.264 Application. The illustration of this design is presented in Figure 6.6. This design is including different components with various properties respecting an irregular topology. Authors choose to divide this architecture into two logic tiers and three DRAM tiers as presented in Figure 6.7. The size of the two logic tiers is equal to  $2.5 \times 5.0 \text{mm}^2$  while the DRAM tiers are  $12.3 \times 1.8 \text{mm}^2$ . All the I/O pads are placed on the back surface of DRAM tier. The partitioning of this SoC is based on the power and on the area of the two tiers. The authors choose to place the UniCore-II and H.264 encoder on Logic-1 as they consume high power and fit on a larger area. All the remaining components including the DRAM controller are placed on the other tier. Thanks to this partitioning the hotter tier is placed on the top of the chip which is suitable to deal with the 3D thermal issues. Each layer will be synthesized using Synopsys tool. The implementation of this architecture is presented in Figure 6.6.

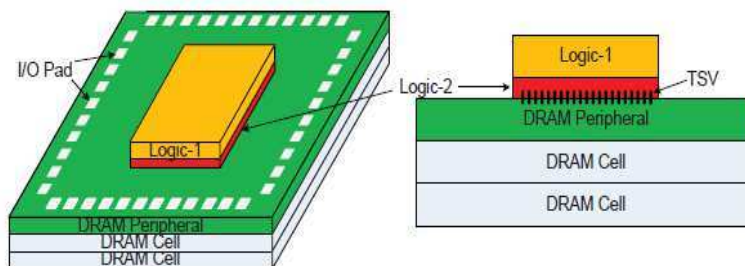


Figure 6.7. 3D DRAM stacking [51]

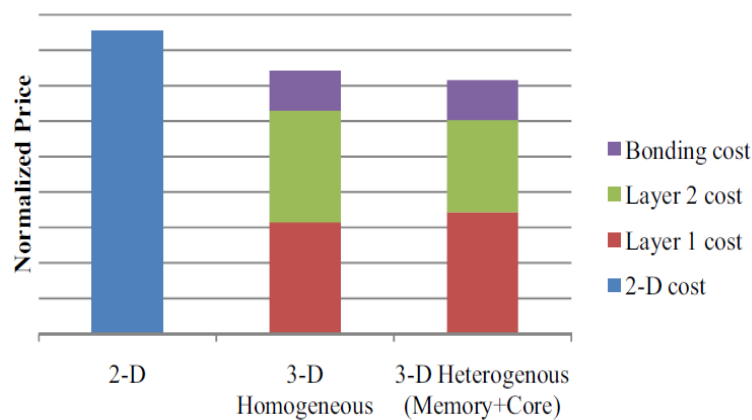
The 3D IC chip has been fabricated using the Global-Foundries 130nm low-power process together with Tezzaron's TSV bonding technology. The total area of tier-1 is equal to  $6.2\text{mm}^2$  while it is about  $7.3\text{mm}^2$  for tier-2. The frequency of the chip is equal to 60 MHz .

## 6.2 3D Heterogeneous Multiprocessor architecture

Supporting heterogeneous stacking is considered as a major advantage to use 3D integration. In fact, the different components of the architecture can be fabricated separately. The use of heterogeneous technologies for large 2D design can reduce the cost of the chip by three times, this result was proven by Intel [100]. The two principle cost reduction methodologies are:

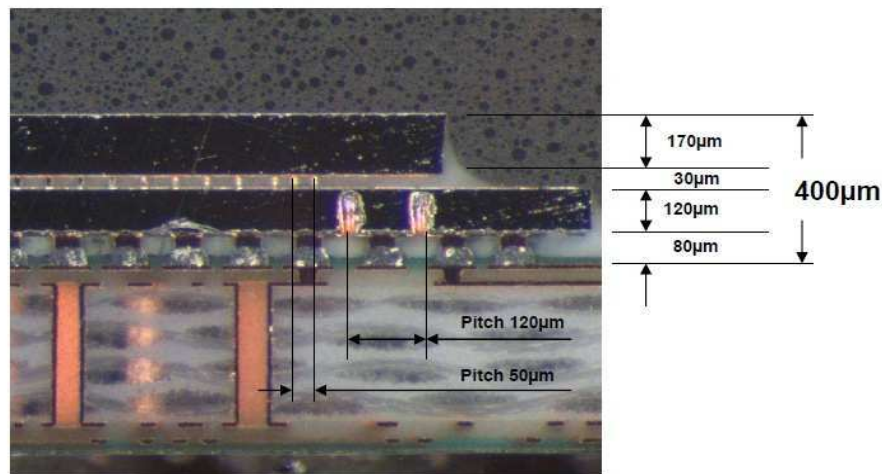
- Metal Layer Reduction : the use of the vertical interconnection can reduce the number of metal layers during the fabrication of the chip
- Heterogeneous Technology stacking: thanks to this technique the noncritical components can be mapped to a similar die which is manufactured using older and chipper process node.

In [101] Dong et al, have taken the OpenSPARC processor as a case study to test the cost of the 3D heterogeneous integration. The 2D equivalent chip has an area size of  $342\text{mm}^2$  and fabricated with TI 65-nm process using 11 metal layers. The SRAM cache is fitting on about 50% of the chip area. That is why, authors have mapped the memory to one layer while all the remaining components are affected to another one. The estimation of the cost design is presented in Figure 6.8. We can see that the 3D integration is cheaper than 2D with homogenous and heterogeneous techniques.



**Figure 6.8. The estimated cost of OpenSPARC : the separate core and memory fabrication reduces the cost [101]**

In the [101] authors have presented two partitioning methods in order to reduce the cost when using 3D stacking which are the coarse-granularity partitioning and the Fine-granularity. The first one, which is based on the separation of the memory and the other components into different dies, is already presented. In the Fine-granularity method, the components are divided at the unit level. The 8-core OpenSPARC T1 processor was divided using this technique into two-layers. The authors have tested two different methods. In the first one called 90nm-90nm stacking, the two layers are implemented using the 90nm technology. Based on their cost model, the estimation of this implementation is equal to 125\$ compared to the original 2D cost which is equal to 146\$. In the second methodology called 90nm-130nm heterogeneous process technology, the authors have used the timing analysis results to perform the partitioning step. In fact, thanks to this information, it is possible to define the sets of components which are not situated on the critical path to move them into the slower layer. With reference to the cost model the use of this technique can reduce the cost of the chip to 121\$ which is 82% of the classical 2D Chip.



**Figure 6.9. cross section of the final package**  
(Courtesy of ST Microelectronics)[102]

We present in Figure 6.9, the cross section of a real 3D heterogeneous chip presented in [102]. This design is a set top box demonstrator developed the complete workflow of 3D implementation. The top layer is implemented using a 45 nm technology while the bottom one is based on the 130nm. We can see clearly the difference between the TSV properties; in fact the pitch is equal to 50µm for the 45nm technology while it is equal to 120µm for the 130nm library.

### 6.3 3 Hardware Accelerator synthesis in 3D Heterogeneous Multiprocessor architecture

Stacked Multiprocessor architecture is a promising application for 3D IC integration. The high bandwidth and the low latency which are characterizing such design are also behind its high performance. The medical image processing is one of the basic domains where we need to transfer a huge amount of data with a very powerful computation and in real time. Some works have treated this application. In [103] Cong et al have designed a 3D specific processor based on FPGA accelerator and applied on the medical image processing. The architecture of this 3D processor is illustrated in Figure 6.10.

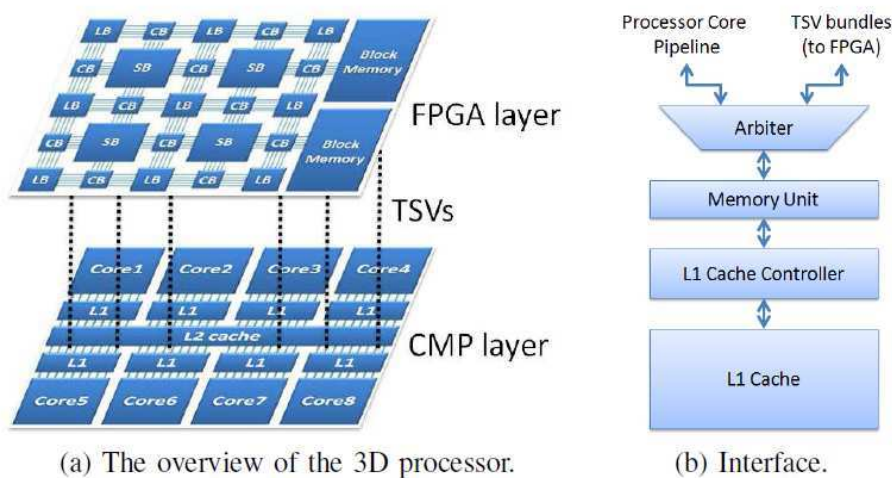


Figure 6.10. 3D processor architecture CMP-FPGA[103]

This proposed architecture is designed by stacking a programmable layer on a CMP layer. The connection between them is ensured using the TSVs. Authors have used the medical imaging to apply the idea of domain-specific acceleration, where many accelerators are sharing the same set of applications. The medical imaging needing high performance computational techniques are the basic tool to perform the treatment of many medical problems. This architecture based on FPGA acceleration can improve the performance of these computations.

### 6.4 Conclusion

One major advantage of 3D IC design is giving the possibility to integrate heterogeneous technologies. In fact, the different tiers of a stacked chip can be fabricated separately. Depending on the cost and on the performance constraints, the designer defines the appropriate technology for each layer.



We presented in this chapter the state of the art of real 3D stacked processors. The direct application of the heterogeneous technology is to map the processor and its cache memory into different cores which can improve the access time and increase the bandwidth of the design. A 3D MPSOC architecture can be used in the medical image processing where we need to transfer a big amount of data in real time.

**Chapter 7 : Theoretical Complexity and Parallel EDA**  
**for 3D**

## 7 Theoretical Complexity and Parallel EDA for 3D

### 7.1 Parallel EDA: Hierarchical MPSOC based 64 PEs on FPGA

We propose in this section to implement a hierarchical MPSOC design based on 64 PEs. This work is a multiplication of the elementary design based on 16 masters and 16 slaves already presented in section 2.2. We propose to perform a parallel implementation of this architecture on multi FPGA platform.

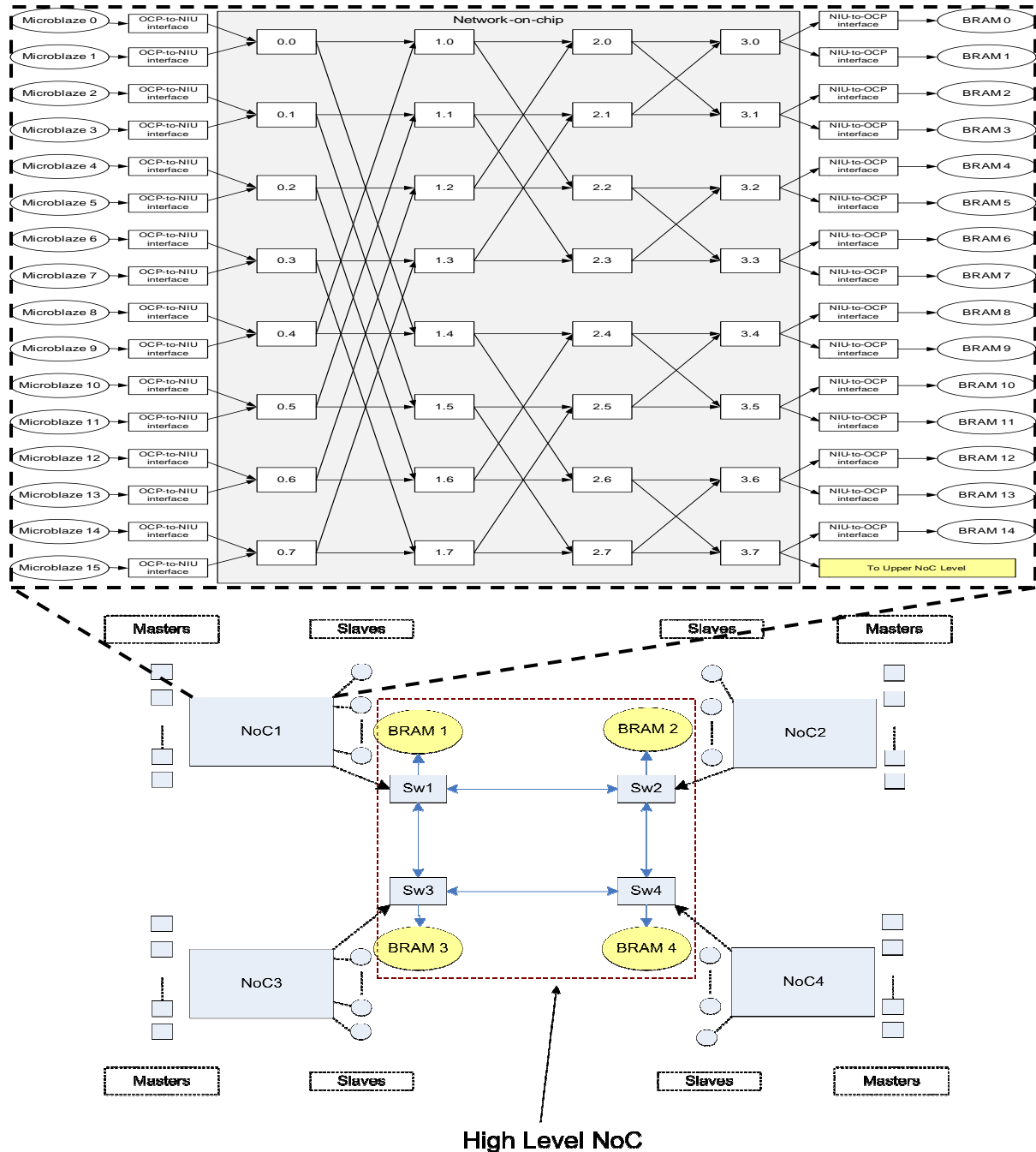


Figure 7.1. MPSOC architecture 64 PEs on Multi-FPGA platform Zebu-UF4

We propose to design the MPSOC architecture which is presented in Figure 7.1. This topology is basically composed of 4 elementary NoCs and a hierarchical central one. The elementary MPSOC is based on a NoC with 16 masters and 16 slaves interconnected with a Butterfly NoC (2Ary-4Fly). During the design of this architecture, we multiply this MPSOC four times and we connect the separate NoCs thanks to a central mesh NoC with four routers. We add common shared memories to the central NoC. To ensure the connections between the elementary NoCs and the central one, we should scarify one slave memory. In fact, instead of the memory 15, we connect the NoC directly to the central one. Thanks to this architecture, all the processors can access their local memories and the common memories situated at the high level. The central NoC has a major role to ensure the synchronization between the 64 processors.

We present in Figure 7.2, our EDA workflow to implement this architecture. In fact the design of the different NoCs can be performed in parallel. Even though the NoC topology is the same, we can choose different properties and options for each NoC. This step is performed simultaneously. Using Xilinx tools (ISE, EDK), we can design the elementary MPSOCs using the IPs presented in Table 7.2. During the hierarchical MPSOC design step, the complete architecture is created.

We apply the same implementation workflow already presented in Figure 2.2. The Eve company tool zCui offers the possibility to run a multithread synthesis place and route. Thanks to this option, we perform a parallel synthesis, place and route design. The implementation of this architecture is realized on the platform emulation Zebu-UF4 which is including five FPGAs board Virtex-4 LX-200 (see Table 2.8). Our MPSOC architecture fits on five FPGAs, this result is illustrated in Table 7.1. We use zCui compiler which is a software tool of EVE Company, to make the synthesis, the placement and the routing on the different FPGAs. We can choose in the zCui compiler the clustering options: manual or automatic. In our case, we use the full automatic clustering, that is why this tool will share the Netlist between FPGAs with equal rates which is illustrated by the synthesis results in Table 7.1 . We use more than 66% of Slices in all the FPGAs. Resources in term of memories are also used with a percentage of 66% for 4 FPGAs and 55% for the fifth one. Partitioning is a critical step when the tool is faced to cluster the Netlist of an asymmetric component like the Butterfly topology.

The Butterfly based NoC is a suitable case study to test the efficiency of the partitioning algorithms used in the industrial tools. In fact it is difficult to find a method to cluster this architecture. We have experimented this in our lab during the implementation of a 64PEs NoC with a Butterfly topology.

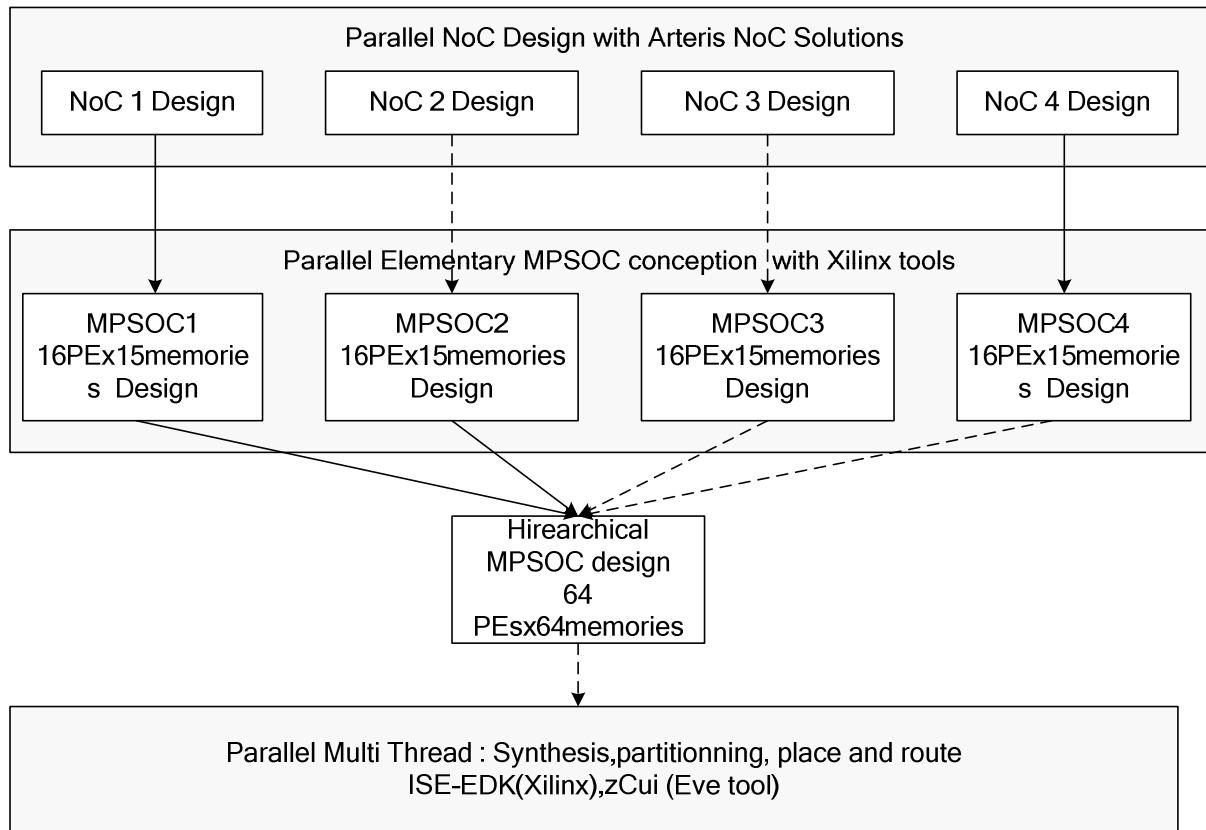


Figure 7.2. Parallel workflow EDA MPSOC implementation on FPGA

Table 7.1. Resources utilization

FPGAs	Slices	RAM
FPGA1	66%	66%
FPGA2	67%	66%
FPGA3	65%	66%
FPGA4	62%	66%
FPGA5	77%	55%

Table 7.2. 64PE MPSOC Used IPs

IP Name	Version	From
Microblaze	7.00.b	Xilinx
lmb_v10	1.00a	Xilinx
lmb_bram_if_cntlr	2.10.a	Xilinx
bram_block	1.00.a	Xilinx
opb_v20	1.10.c	Xilinx

<b>opb_timer</b>	1.00.b	Xilinx
<b>fsl_v20</b>	2.11.a	Xilinx
<b>fsl2ocp_data</b>	1.00.a	ENSTA
<b>ocp_bram</b>	2.00.a	ENSTA

We propose to compare our hierarchical architecture with an MPSOC based on 64 PEs with a single Butterfly NoC. In fact, Hamwi and Hammami have designed an MPSOC architecture with 64 processors, 64 memories and a central Butterfly NoC with five stages of routers [104]. This work is exactly using the same basic IPs that we use in our design and which are presented in Table 7.2. The authors have designed a network on chip with 3 stages, in each one there are sixteen routers with a degree equal to four. This Netlist is a big challenge to test the efficiency of the EDA tools; in fact it can not fit on a single FPGA. This work is using the same workflow that we use to implement the design on FPGA. The implementation of this 64 MPSOC architecture is performed on the board Zebu-UF4.

We can compare the implementation results of both designs, the MPSOC design presented in [104] needs 323% of the available BRAM of an FPGA Virtex-4 LX200 which is almost the same value needed for our 64 PEs design. The number of slices used in our design is equal to 337% of the slices available in a single FPGA while the compared work needs 207%. This difference in term of number of slices is due to the fact that we are using 4 basic NoCs and a hierarchical one which includes 132 routers for each request and response part while in the single big NoC used in the other work we find 48 routers in both NoC sides. The degree of all the used routers in our work is equal to two while this value is equal to four in the work presented in [104] which means that the size of our routers is smaller than the ones used in their design. Our work is based on the elementary 16x16 MPSOC architecture that is why the partition of this design was relatively easy compared to the MPSOC with a single NoC with 64 nodes. That is why we have almost a balanced distribution of the design on all the FPGAs which is not the case of the other work where we can have an FPGA with 91% of used slices and another one with only 31% Table 7.3. The used EDA tools take an important time to perform the partitioning of the designs but it was clear that partitioning the design with one big asymmetric Netlist was a real challenge.

**Table 7.3. Resource utilization of the MPSOC based 64 PEs NoC[104]**

<b>FPGAs</b>	<b>Slices</b>	<b>RAM</b>
<b>FPGA1</b>	32%	95%
<b>FPGA2</b>	91%	38%
<b>FPGA3</b>	44%	95%
<b>FPGA4</b>	40%	95%
<b>FPGA5</b>	0	0

## 7.2 3D Theoretical Complexity from return of experience

### 7.2.1 Core to layer mapping

We propose in this section to evaluate the core to layer mapping complexity. This step is usually performed manually by the user. In our 3D NoC synthesis methodology already presented in the paragraph 5.4, we propose to solve this sub problem using the MOEA. We propose to have a number of cores equal to  $N$  to be mapped into  $k$  layers, the complexity of this step is equal to  $k^N$ . In our case, we use the 3D Tezzaron methodology, which is a Face to Face technology using 2 layers. The complexity of a core to layer mapping operation of a coregraph including  $N$  cores is equal to  $2^N$ .

### 7.2.2 Floorplanning

We can define the floorplanning step by fixing the different positions of the cores in the chip area. We propose that we define a maximum values  $X_{\max}$  and  $Y_{\max}$  which are respectively the maximum values of the horizontal and vertical coordinates  $(X,Y)$  of the core.  $X$  and  $Y$  are the coordinates of the upper left corner of the core which are two integers in the margins  $[0,X]$  and  $[0,Y]$ . The complexity of the floorplanning of a coregraph with  $N$  cores is equal to  $(X_{\max} \cdot Y_{\max})^N$ . In our case, the core can take any position in the chip area which means that it is possible to have an overlap between different cores. A floorplanning of the cores which is taking in consideration the already placed cores is not any more simultaneous. In the case of a sequential floorplanning, new problems appear like the order of the core treatment which can affect the final result. We use in our 3D NoC synthesis problem the Error\_overlapping constraint to guide the Evolutionary Algorithm toward feasible solutions avoiding the superposition between the cores.

### 7.2.3 NoC topology

In order to generate the NoC topology which is respecting all the demands of the coregraph, we propose to define a set of routers indexed from 0 to  $N-1$  where  $N$  is the number of cores in the coregraph. A core can be connected to any router but a router  $i$  can only be connected to router with a higher index. The complexity of the NoC topology generation is equal to  $N!$ , the complexity of all the NoC topology generation is equal  $(N!)^N$ .

### 7.2.4 NoC floorplanning

We propose in our 3D NoC topology to generate the NoC floorplan at the same time of the topology generation. The positions of the routers are generated for all the set of used or

not used switches. Which means that the complexity of this step is equal  $(X_{\max} \times Y_{\max})^N$  where  $X_{\max}$  and  $Y_{\max}$  are the coordinates of the upper left corner of the router and  $N$  is the maximum of the used routers which is equal to the number of cores in the coregraph. In our methodology, even though the floorplan is generated for all the routers only the used ones in the NoC topology are taken in consideration. The output result is only representing the used routers.

We choose in our methodology to solve all the 3D NoC synthesis sub problems simultaneously. That is why the complexity of our workflow is equal to

$$2^N \cdot (X_{\max} \times Y_{\max})^N \cdot (N!)^N \cdot (X_{\max} \times Y_{\max})^N = 2^N \cdot (X_{\max} \times Y_{\max})^{2N} \cdot (N!)^N$$

We can see that the mathematical complexity of our 3D NoC synthesis is exponential. It is known that the 3D NoC synthesis problem is NP-Hard. That is why; it is not possible to solve it with deterministic algorithms. The complexity of the 3D NoC synthesis problem represents our major motivation to use the MOEA( section 5.4) in order to solve it. In this work, we choose to solve the 3D NoC synthesis problem as a complete system without dividing it into sub problems. Dividing the 3D NoC synthesis problem is an efficient method to reduce its complexity, but it can have a meaningful effect on the final results.

### 7.3 Parallel EDA for 3D IC implementation

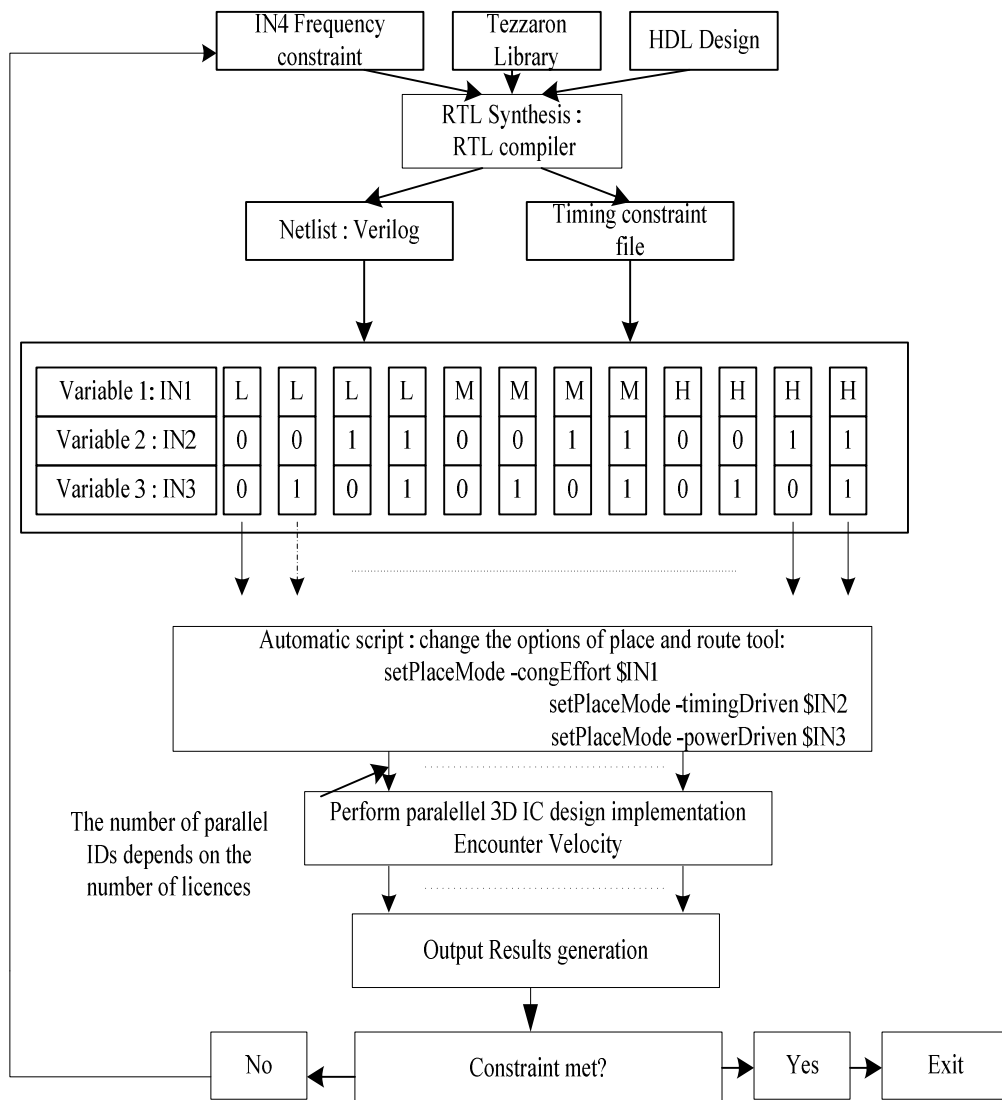
We propose in this section to perform a parallel DSE to the 3DIC Tezzaron workflow. In fact the 3D Tezzaron technology is based on the use of automatic scripts to perform the 3D IC implementation. The basic steps of this workflow are: the synthesis, the floorplanning, the placement, the Bumps creation and the routing of the signals. We basically use the Velocity tool from cadence to perform these steps. In order to evaluate the compatibility of this tool with the 3D Tezzaron technology, we propose to perform an exploration to the different options of the tool during the placement and the routings steps. We propose to explore all the combinations related to congestion, the timing and the power driven placement, the different options of these parameters are presented in Table 7.4.

**Table 7.4. Options of FILTER DSE**

IN1	CongEffort (Medium, High, Low)
IN2	TimingDriven(0,1)
IN3	PowerDriven(0,1)
IN4	Frequency



The variables presented in the previous table represent the different options for the placement operation. The variable IN1 can have 3 possible values: Medium, High and low. This option decides if the tool takes in consideration the congestion effort during the placement step. The variables IN2 can have the value 0 or 1. When this value is true the velocity tool performs a timing driven placement while the value zero means that the placement operation is independent from the timing constraint. If we want to perform a power driving placement, the value IN3 should be equal 1, 0 otherwise. The last value which is IN4 defines the frequency constraint value.



**Figure 7.3. 3D IC parallel and automatic workflow**

We propose to create an automatic workflow using Bash scripts, to modify automatically the input files of the 3D IC implementation. We present in Figure 7.3, our

parallel EDA workflow. In fact, we automatically modify the input constraint file before the RTL-synthesis step. This modification is ensured thanks to the variable IN4. For each fixed frequency we generate the corresponding Netlist which will be the input of the place and route step. We define all the possible combinations of the input variables in an input file and we develop automatic scripts to read the value of the variables then to modify the 3DIC Tezzaron script. For a fixed frequency value we have 12 possible configurations. For each configuration also called ID we implement the whole 3D IC script to perform the place and route operations. These different implementations are run in parallel, providing different GDS output results. The fact that the RTL synthesis provides a Netlist which is satisfying the timing constraint does not guarantee that the place and route steps can be performed with the same constraints. That is why, a step of constraints verification should be performed after each 3D IC place and route. When there is a constraint violation, the input constraints should be relaxed. The illustration of this parallel 3D IC flow is represented in Figure 7.3.

#### 7.4 Parallel EDA for 3D : Case study

We propose in this section to perform the design space exploration of the options used by Encounter tool to place the Filter design provided with Tezzaron Design kit. We perform all the possible combinations of the 3 first inputs with fixed frequencies. These options are affecting the step of the placement of the design. After each combination we take the value of the density, the power and the WNS (Worst Negative Slack). The WNS is the difference between the critical path of the design and the period which is the inverse of the frequency. The result of the 3D implementation of the Filter is presented in Figure 7.4.

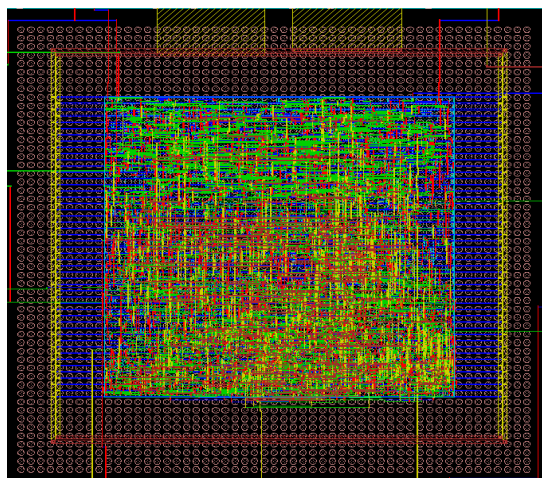


Figure 7.4. The placed and routed Filter with 3D Tezzaron Technology

We illustrate in Figure 7.5, the obtained results of our exploration. We propose to evaluate the WNS value, the density of the chip and its steady power estimation. We explore the twelve configurations presented in Table 7.5. We define in the user constraint file, the objective frequency of the design but after the place and route the velocity tool gives the value of the WNS which gives a better idea about the real reached frequency. That is why when the WNS is positive it means that the frequency constraint is met which is not the case when this value is negative. From the curve of the WNS we can conclude the optimized frequency of the design.



**Figure 7.5.. WNS (ps), Density (%) and power (mw) for the different combinations of the DSE (Freq=100MHZ)**

We can see that the smallest worst negative slack is for the ID 1 which means that we got the worst frequency with this configuration. If we look to the configuration of ID 1, we have a low congestion effort, the value of the timing driven option is 0 while the power driven option is activated. The activation of the power driven option ensures the power optimization of the design, but there is a small difference between these values even though we change the other parameters. We can conclude that the worst case for power consumption is the ID number 0: this configuration is almost the best one in term of frequency but it is also the worst one in term of power consumption. When the 3 options are activated, which means that we use the High congestion effort with timing and power aware placement (configuration 11), the output result is almost the average in term of frequency, density and power.

**Table 7.5. Different ID (L: Low, H: High, M: Medium)**

ID	IN1	IN2	IN3
0	L	0	0
1	L	0	1
2	L	1	0
3	L	1	1
4	M	0	0
5	M	0	1
6	M	1	0
7	M	1	1
8	H	0	0
9	H	0	1
10	H	1	0
11	H	1	1

## 7.5 Conclusion

With the increase of the number of cores in the same chip which is following an exponential curve, the graphical use of the EDA tools can not be any more possible. That is why we proposed in this chapter to present our automatic and parallel used methodologies. In fact, starting from a first MPSOC designed with 16 cores and 16 memories; we have designed an hierarchical architecture with 64PEs and 64 memories. The basic design was multiplied four times and a central high level NoC was designed to ensure the different interconnections. The design of the sub-architectures was performed separately but the steps of synthesis, place and route were performed at the same time using the multithread option.

We use in this work, the 3D Tezzaron methodology based on cadence tools. We presented in this chapter an automatic exploration of the different options of these tools in order to define the optimized design configurations which improves the performance and minimizes the cost.

## Chapter 8 : 3IC Design and Modelling Case of Tezzaron

## 8 3IC Design and Modelling Case of Tezzaron

With the shortage of information and examples of real 3D IC implementation, we propose in this chapter to implement an MPSOC architecture using the 3D IC Tezzaron technology. The goal of this work is to experiment the different steps of the 3D workflow methodology proposed by Tezzaron in order to choose the appropriate NoC synthesis technique.

### 8.1 MPSOC basic components

We present in this chapter the 3D IC implementation of two MPSOC architectures based on Mesh and Butterfly NoCs. We present in this section the main components needed to design these architectures.

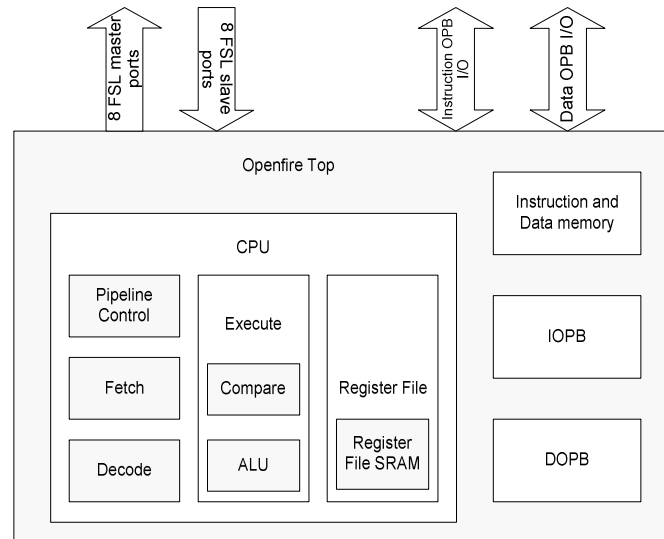
#### 8.1.1 Processor

We choose in this work to use the OpenFire processor as a software core for our designs. This one has almost the same architecture as the Microblaze processor from EDK (Xilinx)[105] but the OpenFire processor is provided as an open source and can be downloaded from opencores.org website with the Verilog Language. The architecture of this processor is presented in Figure 8.1. In fact, the top level entity of this processor can be connected to other components thanks to 8 FSL master ports (output) and 8 FSL slave ports (input). The FSL (Fast Simplex Link) interface can be connected to an FSL bus which allows a simple point to point connection. The communication with the processor can be also performed thanks to the OPB (On chip Peripheral Bus) ports. The first port, called IOPB, is dedicated to perform the read operation from the instruction memory while the DOPB port allows the read/write from the OpenFire's data memory. The CPU core, the local data memory and instruction memory and the two OPB ports controller represent the main units inside the top level entity. The CPU module uses a three stage pipeline based on the fetch, decode and execute blocks. The pipeline control is the responsible to stall the pipeline when multi-cycle instructions are executed. The implementation of all the internal programs is performed thanks to the register files. This component interfaces with the other units in the CPU in order to perform the data routing operation.

#### 8.1.2 Fast Simplex Link (FSL) Bus

The Fast Simplex Link (FSL) bus is a basic mono directional bus ensuring a point to point based FIFO communication. FSL can perform a fast communication between any two

design elements having the suitable FSL interface which is the case of the Openfire Processor. Thanks to its 8 FSL ports, this processor can be connected to 8 different components which increases its bandwidth. The depth of the FIFO can reach 8K. This bus can support both FIFO modes: synchronous and asynchronous which give the designer the freedom to affect different clock domains to the different sides (master and slave).



**Figure 8.1. Openfire processor architecture**

### 8.1.3 3D Router

3D router architecture which is illustrated in *Figure 8.2 and Figure 8.3* comprises four neighbouring ports, one vertical port for the connection to another tier and one local port to the processor through network interface unit. Each input/output port has 35 bits data flits and 2 bits control signals for packet transfer between routers. Handshake protocol is used for router to router communication and router to network interface communication. Each input port has one buffer built using 16 words FIFO based dual port RAM architecture to support a maximum of 16 data blocks transfer. As XY routing is deadlock free and we do not implement priority packets transfer, virtual channel implementation is not necessary. We use round robin arbitration for output port selection when there is more than one input requesting the same output route. Wormhole switching is used for packet transfer in the NoC because it does not require large buffer and has lower latency. For the routing, deterministic coordinate based routing is implemented using XYZ coordinate where each packet will travel first in the X direction followed by Y direction and finally through Z direction (vertical) to the other die.

The network interface architecture as shown in *Figure 8.3*, connects the router to the processor through two FIFO ports. Based on data address and number of words sent by the processor

through one of the FIFO port, the network interface will access the processors data memory to process data blocks through DMA. Each network interface unit connects a router to a processor through 2 FSL ports (FIFO) of the Openfire processor; the first one is a master FSL for writing data to be transferred through the NoC and the other one is a slave FSL for reading synchronization flags sent by other processors. The synchronization FIFO has 16 words (one word per processor) with 5 bits data width each. There is one 11 bits counter in the network interface unit for measuring packets travel timing. The timing information is included in the head flit attached to the packets when entering the network and is processed when the packets arrive at the destination network interface.

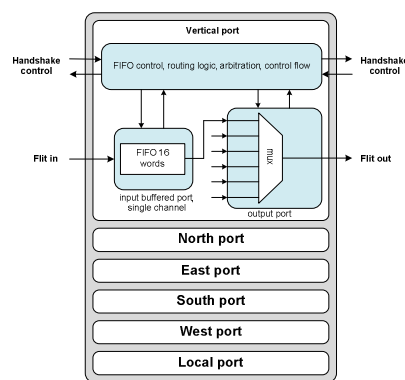


Figure 8.2. 3D Router architecture

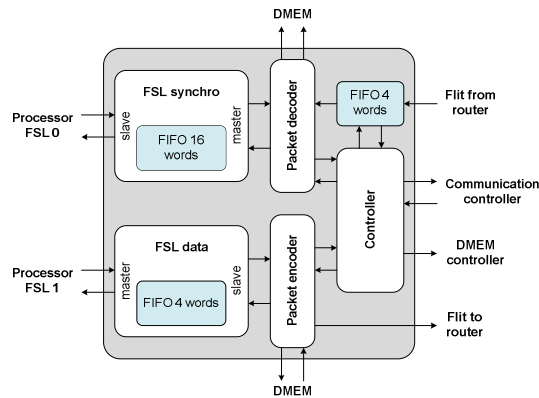


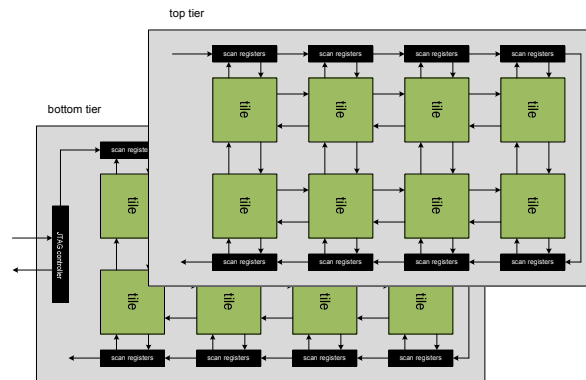
Figure 8.3. Network interface architecture

## 8.2 Architecture 1 : MPSOC1 based on mesh topology

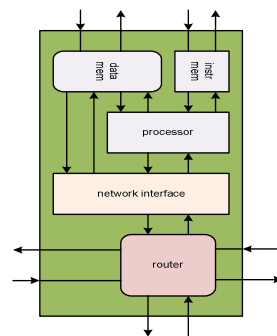
In order to explore the complete 3D IC workflow, we propose to design a basic 3D architecture based on the Mesh topology. This design is a 16 PEs MPSOC architecture fitting on two face to face layers which is illustrated in Figure 8.4. Thanks to the symmetry of the design the partitioning of the different cores is a trivial task. In fact, we group each core with its local memories, a Network Interface Unit (NIU) and a router into an independent tile. We present in Figure 8.5 a detailed architecture of a tile. In each layer, we place 8 tiles following



mesh topology architecture. A processor can communicate with the different processors from the same layer or from the opposite one thanks to the 3D router. The different routers are connected creating a 3D Mesh NoC ensuring the data transfer between all the PEs included in the 3D design. The routing of the different packets is performed in the router who decides to send it to the horizontal PE neighbours via the X and Y ports or to the opposite processor in the other tier thanks to the port Z.



**Figure 8.4. Architecture 1: MPSOC based on Mesh topology**

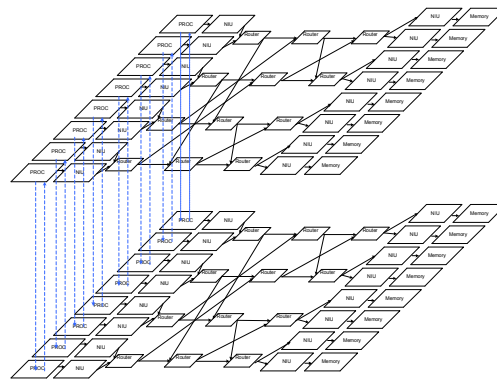


**Figure 8.5. Tile Block Diagram**

Synchronization between processors is ensured using FSL linked to the NoC. Processors communicate together through their data memories. A processor will synchronize before accessing its data memory by waiting for a tag word in its FSL sent by the writer processor. This is a simple synchronization hardware implementation in order to reduce die area. If we compare this 3D mesh MPSOC architecture with a 2D equivalent one (4PEs, 4PEs), the diameter of the 3D NoC is equal to 5 where this value is equal to 6 in a 2D architecture. The reduction of the NoC diameter is a theoretical proof that the 3D IC conception should increase the performance of the design.

### 8.3 Architecture 2 : MPSOC2 based on Butterfly topology

We present in this section the second MPSOC architecture based on the Butterfly NoC. We choose to implement this topology as we believe that it represents an interesting case study to show up the 3D IC advantages and limitations. In fact the transformation of the long links into vertical connections is a real motivation to move from 2D to 3D design. Moreover, the Butterfly NoC has an asymmetric architecture which represents a new problem when performing the core to layer mapping step. This 3D architecture is mapped into 2 tiers: Top and Bottom. We create a design based on an 8x8 Butterfly NoC linking 8 master processors to 8 slave memories. We use the FSL (Fast Simplex Link) Bus to connect the Openfire processor to the NoC via Network Interfaces. This processor gives the possibility to connect up to 8 FSL links. That is why; we connect each processor to the NoC in the same layer with an FSL port 1 and keep the FSL port 2 to make a vertical link with the processor in the opposite tier. This architecture is presented in Figure 8.6. With these vertical connections, processors from the Top tier and the bottom Tier can communicate and synchronize together.



**Figure 8.6. Architecture 2 : MPSOC Based on the butterfly architecture**

This 3D MPSOC architecture is based on two Butterfly NoCs; each one is linking 8 cores to 8 memories. The number of routers in each NoC is equal to twelve forming three different cascaded stages. The processors are connected to the first stage through the network interface units called FSL2OCP. These elements transform the FSL bus signals to fit the OCP (On Chip Protocol) interface. Another interface called OCP-to-NTTP transforms those signals to fit the internal protocol of the NoC called NTTP. The routing of the packets is performed thanks to the different routing tables included in all the routers. Depending on its address and on the router's routing table, the packet is routed to the suitable output port to finally reach its last destination which is a slave master. In this architecture, processors from the same layer can only communicate by reading or writing from the shared slave memories while the

communication between the two different layers can be performed directly thanks the FSL links.

#### 8.4 Implementation results and discussion

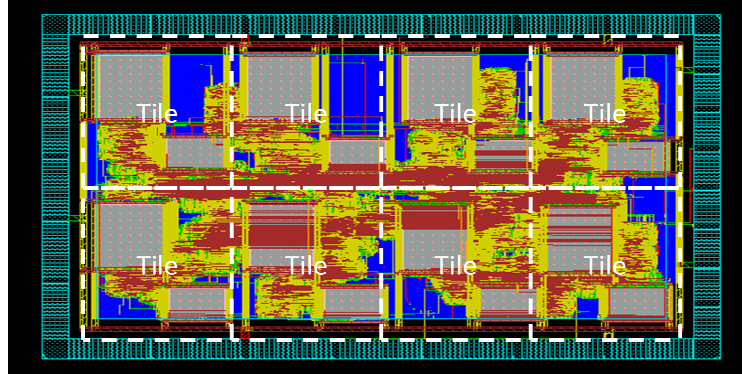


Figure 8.7. . MPSOC1 Mesh: Bottom tier routed layout

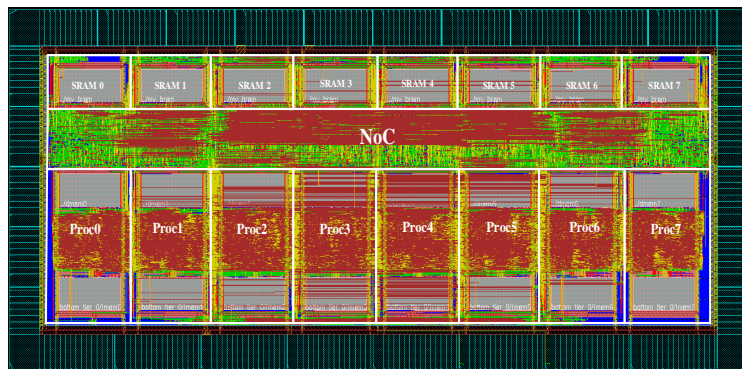


Figure 8.8. MPSOC2 Butterfly: Bottom Tier routed layout

We apply our 3D Tezzaron methodology on the two MPSOC architectures: MPSOC 1 which is based on the Mesh topology and MPSOC 2 which is based on a Butterfly NoC. We present in Figure 8.7 and in Figure 8.8 the implementation results of the two chips. The different properties of the two chips are summarized in Table 8.1. MPSOC 1 needs about 15.7 mm<sup>2</sup> area while MPSOC 2 requires about 10mm<sup>2</sup>. In the MPSOC 1, we use larger data memories than those used in MPSOC2. In fact, the processors in MPSOC 2 can access the different shared slave memory which is not the case of the first architectures where the processors can only store data in their local memories. The MPSOC based on the mesh topology needs about 1.3 Million Gate per tier while the other one needs 4 times more. The NoC used in MPSOC 2 is created with an industrial tool which is offering more options that is why the need of one router in term of logic gate is more important than the home made 3D

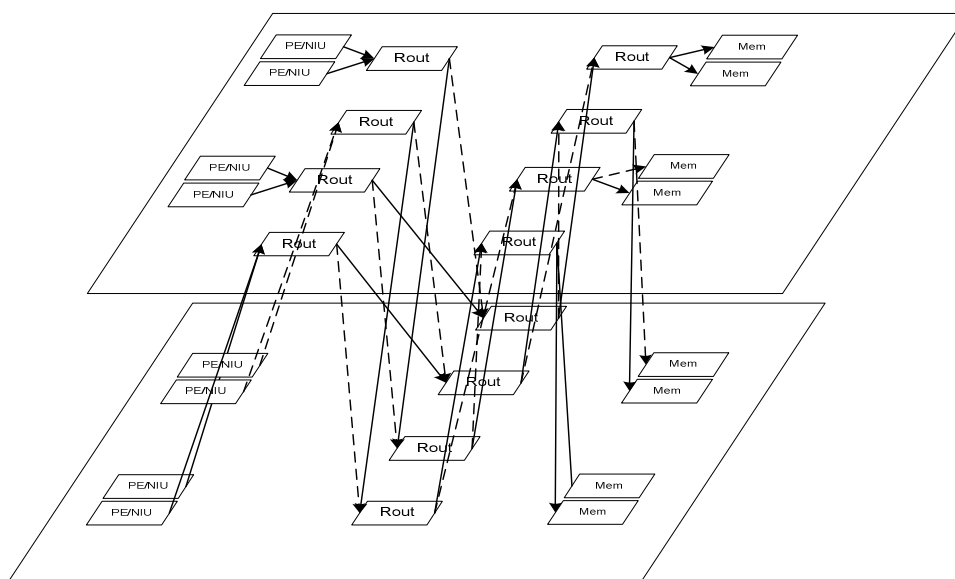
router used in MPSOC 1. Both architectures have more than five hundred vertical signals which are ensuring the communication between the different cores. In both cases the frequency of one tier is equal to 100 MHz but we can not consider this value as the total frequency of the 3D chip, such result can be only obtained at the end of the 3D chip creation. After this 3D design experience, we get a clear idea about the 3D IC Tezzaron methodology: issues and limitations. The floorplanning step is trivial with the Mesh topology of MPSOC 1. In fact, the processors are homogenous in term of configuration and local memory sizes. In this case, each tile is including a processor, a data memory, an instruction memory, a router and a Network Interface. The grey boxes presented in each tile are reserved for the memories. The implementation of the architecture MPSOC 1 was an easy way to validate and to experiment the complete workflow. In fact, the complete workflow takes 2 hours and a half to generate the GDS file describing the layout of the design. The step of routing where the cadence tool takes in consideration the vertical 3D signals assigned to the different Bumps represents about 50% of the whole design time. The implementation of the second architecture presents more serious problems. The floorplanning of a heterogeneous MPSOC architecture is known to be NP hard. The actual 3D IC Tezzaron workflow does not take in consideration the automation of this step. With our chip, which is considered as a small design, we perform this operation manually taking in consideration the architectural properties of our design. We place the NoC in the middle between the processors and the memories. The routing step takes about 3 hours which represents more than 75% of the complete workflow implementation. The router takes a double time to perform the routing step of MPSOC 2 compared to MPSOC1. This result is due to the complexity of the Butterfly architecture which has an asymmetric topology but also to the limitation of the SoC encounter tool. In fact, the 3D Tezzaron workflow methodology is a set of sequential and independent steps. For example, the routing is performed after the placement step which means that the placement does not take in consideration the cell connections. It is clear that the floorplanning, the placement and the routing steps are interdependent that is why a sequential and a static methodology can never guarantee an optimized result.

**Table 8.1. MPSOC Implementation results**

Parameters	MPSOC 1	MPSOC 2
Die size per tier	3.2 mm x 4.895mm	1.99 mm x 4.95 mm
Number of ASIC gate per tier	1.3 Million	5.934 Millions
Inter die signal connections	594	560
Frequency	100Mhz	100 MHz

### 8.5 Complexity of 3D implementation

3D conception is facing a big limitation which is the lack of the industrial EDA tools. In fact until now, there is no complete tool for the real implementation of 3D ASIC design. Tezzaron is providing home made scripts to modify the 2D EDA tools by adding pins under the Bumps. The first difficulty in the design is the partitioning of the project conception. The perfect partitioning of the 3D NoC Butterfly is the one replacing its long interconnect links by vertical connections, a possible architecture is illustrated in Figure 8.9. As we are using a face to face 3D Technology, we propose to place the different stages of the NoC alternatively on the different layers. The implementation of the different dies separately represents a major problem in 3D IC conception. In fact, we should perform the complete place and route workflow for each tier separately. When we have asymmetric designs this will be time costly. In addition, the verification of the complete chip can be only performed at the end of the workflow. The lack of EDA tools dedicated for 3D designs is the major faced difficulty. As we are from the first users of Tezzaron technology in 3D , there is only few reference designs provided with the design kit which are in the almost cases simple and not representing facing the EDA faced problems.

**Figure 8.9. 3D MPSOC partitioning**

From return of experience, we can report the advantages and the issues of the 3D IC Tezzaron workflow. In fact using this technique was a good experience to get familiar with the 3D IC properties. It was a genius idea to modify the classical 2D workflow to suit the 3D conception. This technique is principally dedicated for a chip with two face to face layers and can be extended to at most 4 layers. The use of 2D tools in some critical steps like the placement and the routing does not really prove the real motivation behind the 3D IC technique. In fact the implementation of each tier is done separately without taking in consideration the information from the opposite one which does not guarantee the optimal chip result.

## 8.6 3D IC Fabrication

We propose in this work to use the 3D Tezzaron technology provided by Tezzaron[106] company. This 3D technique is a wafer level, Via-first and metal-to-metal thermal bonding. This technique has produced two generations of 3D vertical connections: The first one is “Super Via“ and the second one is “Super Contact”, both are illustrated in Figure 8.10.

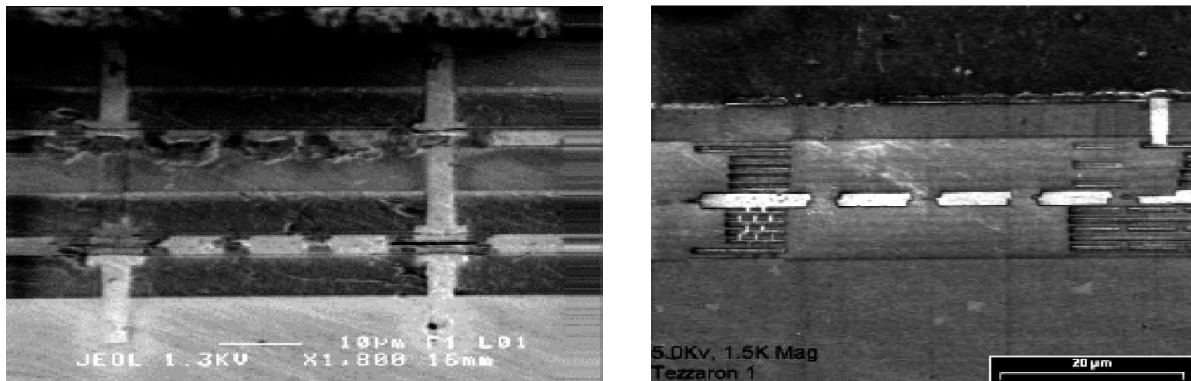
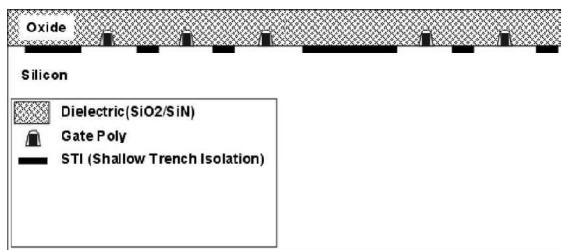


Figure 8.10. Tezzaron 3D Techniques: Super-Via(left), Super Contact(right)[107]

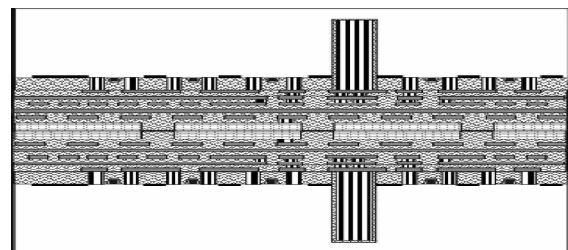
The Advantage of the first generation of Tezzaron TSV is the fact that the fabrication of the TSV is applied on the wafers after their complete process at a vendor fab. The main issue of this method is the high cost of TSV insertion in term of area. The “super-Contact” process needs to add a new process module at the vendor fab which is an easier task compared to the “Super Via” method.

We detail in Figure 8.11, the complete process of the “Super Contact “ fabrication. We present in the first step a cross section of a wafer after the transistors process creation and before the contact metal. In the next step, the “Super Contact” is etched passing through the

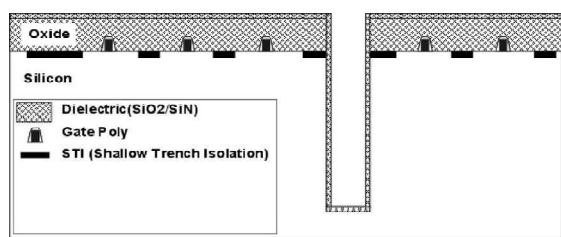
oxide and the Silicon Substrate to be then lined with  $S_iO_2/S_iN$ . In the third step the “Super Contact” is filled with Tungsten and finalized with chemical-mechanical polishing (CMP). These are the only steps which are performed at the wafer level. During step 4, the wafer is finished normally by adding the wiring layers. After recessing the oxide surface of two wafers, those one area then aligned and bonded using a copper thermal diffusion process which needs approximately about  $400^\circ C$ , this step is illustrated in step 5. During the sixth step and after the bonding operation, the wafer situated on the top is thinned until reaching the bottom of the “Super Contact”. The thickness of the substrate is about  $4\mu m$ . After this, the backside of the thinned wafer will be covered with an oxide. An additional process is then performed to create bonding pads for an eventual stacking. The stack is then inverted which is illustrated in step 7, the first wafer is now situated on the top level. A final process will be applied on the first level. In the last step of this process, the first wafer is thinned in the same way of the previous steps and stops at the level of the bottom of the Tungsten super-contact. This wafer is then covered by an aluminum layer to perform a normal bonding.



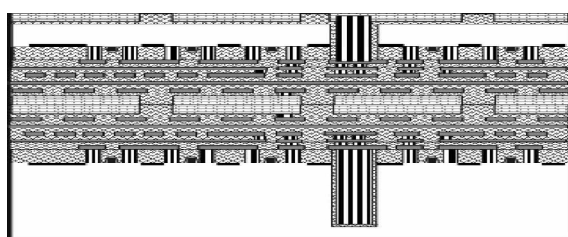
Step 1



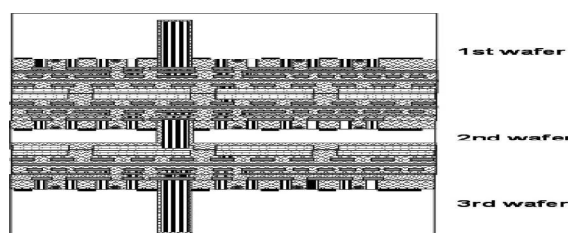
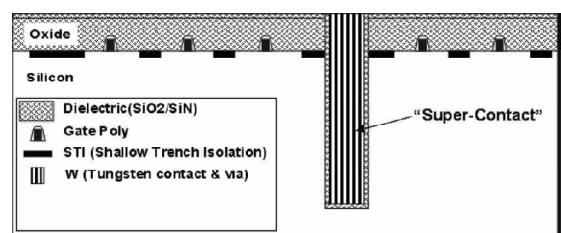
Step 5



Step 2



Step 6



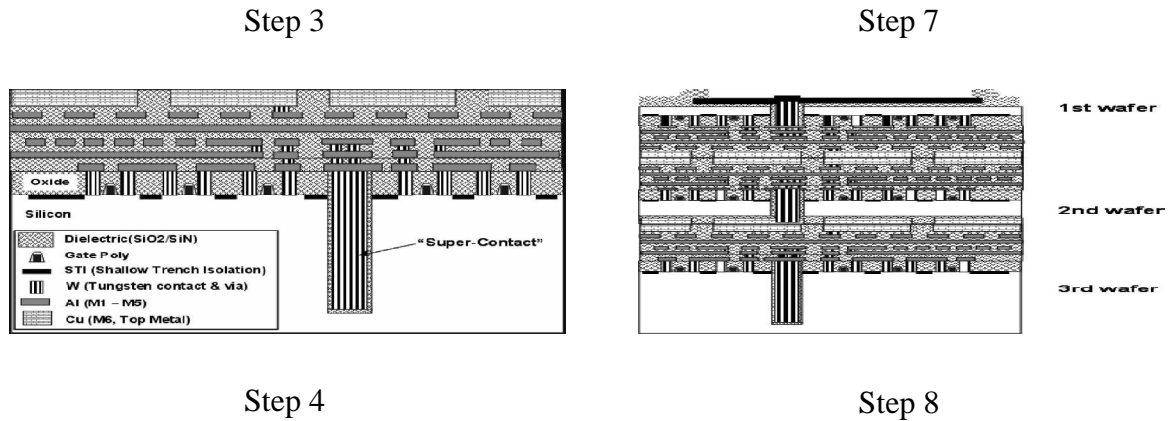


Figure 8.11. Illustration of Tezzaron's Stacking method with the " Super Contact" Interconnect[107]

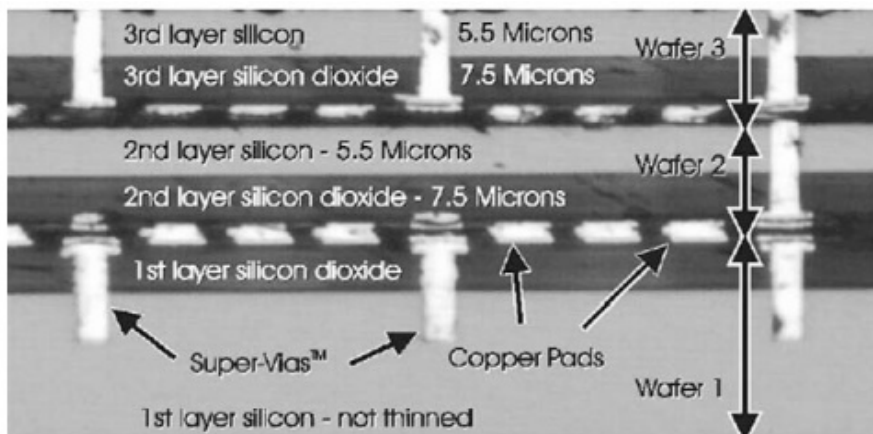


Figure 8.12. Tezzaron metal bonding[62]

After the creation of each die separately, the process of the 3D chip manufacturing can be performed. In fact, the vias can be deposit thanks to the stubs which are connected to the last metal layer, in the case of Tezzaron Technology this one is metal 6. We can see the illustration of this step in Figure 8.13.

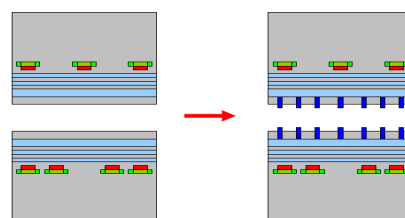
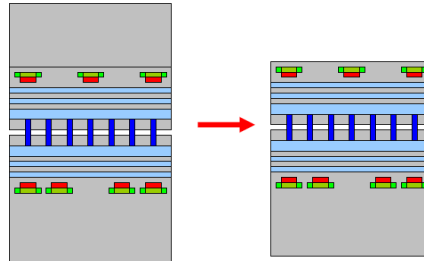


Figure 8.13. Deposit D2D Vias[108]

After the step of Via creation, the Thermo compression bonding operation will be performed on the separate dies. The applied pressure and temperature during this step cause the fusion of the copper stubs, at the end of this step opposite Via coppers from both dies are

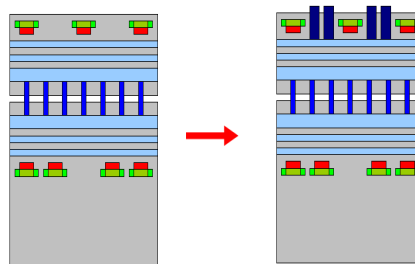


fused together. The next step is the CMP (chemical mechanical polishing) thinning. In fact, during this process the upper stack is thinned to 10 or 20mm [108]. We present in Figure 8.14, the illustration of the CMP thinning step.



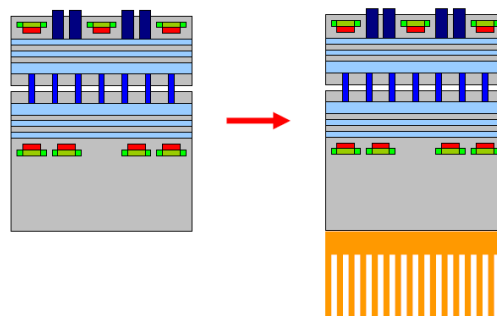
**Figure 8.14. CMP step for stack thinning [108]**

The next step of the 3D chip creation is the Backside etching dedicated for power, ground and I/O. Thanks to the 3D stacking, the connections of the supply signals and the I/O to be shorter. New TSVs are added on the thinned die (see Figure 8.15 ) to ensure this functionality.



**Figure 8.15. Etching for power/Gnd TSV [108]**

The last step of the 3D Chip manufacturing is the packaging. But with the thermal dissipation problem which is considered as a serious challenge in 3D IC integration, a Heat Sink must be added to the stacked chip. It is recommended to place the active component near to the Heat sink in order to easily evacuate the thermal dissipation.



**Figure 8.16. The Heat Sink creation [108]**

## 8.7 Conclusion

3D conception is an emerging and an attracting research field, but only few works have performed real implementation. In order to have a practical experience with 3D IC conception, we designed two MPSOC architectures with 16 processors based on the Mesh and the Butterfly architectures. The first design called MPSOC1 based on the Mesh topology was an easy example to experiment the whole 3D workflow provided by the Tezzaron Company. With its short interconnection, its homogenous tiles and its small size, this architecture does not reveal critical problems during the implementation. The gain in term of chip area is reduced to 50% of a 2D design but the frequency does not notice a meaningful increase. The second design, based on the Butterfly NoC, which represents an asymmetric topology, was an interesting case study for the 3D methodology. In fact, the partitioning of the architecture and the core to layer mapping represent serious problems to perform the implementation of the chip. Thanks to this experience, it was clear that the 3D Tezzaron methodology should be modified to fit with different architectures. In fact, the core to layer mapping and the partitioning should be taken in consideration during the conception. The perfect 3D conception should be performed using specific 3D tools solving the different sub problems with interaction between the different tasks.

## Conclusion

## 9 Conclusion

Referring to the ITRS roadmap, the number of cores increases each 18 months following an exponential curve. The NoC has been considered as an emerging solution to deal with the problem of the chip interconnects. But the use of the Nanometer technology has presented a new major limitation where the interconnect delay overcomes the gate delay. 3D IC was one of the proposed solutions to deal with this problematic. Even though there exist some industrial and academic 3D tools, the shortage of a complete 3D dedicated workflow represents the major challenge in this field. We presented in this work a state of the art of the existing 2D and 3D workflow methodologies. New 2D and 3D NoC synthesis workflow were also proposed.

In the first chapter, we presented the state of the art detailing the evolution of the MPSOC design. Different 2D real implementation workflows used in literature have been discussed. Even though the basic steps are the same for all the methodologies, which are the design, the synthesis the place and the route and the execution, there is a big difference between the different workflows. With the increase of the number of cores in the SoC, the simulation is not any more possible. Only methodologies based on the emulation can deal with large scale designs.

In order to evaluate a set of industrial EDA tools, we presented in Chapter 2 the implementation of MPSOC architecture with 16 processors, 16 memories and a Butterfly NoC on FPGA. The used methodology is based on the industrial tools from the companies Arteris, Xilinx and Eze. A real execution of our MPSOC architecture has been performed on different FPGA emulators like Zebu-UF4 and Zebu-Server. To find the optimized MPSOC configuration, we performed a MOEA on the different Hardware options. The results of our DSE provided a set of Pareto front with a compromise between the area and the frequency. Our design space exploration of the complete architecture represents a database which can be used as a reference design to prevent the needs of the user in term of Hardware and Software options.

Chapter 3 was the subject of the 3D technology state of the art. We detailed the different techniques of 3D IC stacking and the basic 3D interconnects notions like TSV and Microbumps. 3D IC presents various advantages like reducing the interconnect length which decreases the power consumption of the chip. The main difficulty which is discouraging the designers to move from 2D to 3D design is the shortage of specific 3D tools.

A major motivation of this work is to study the optimization methods of MPSOC design. In this thesis, we focus on the NoC optimization based on the user constraints, which represents, the subject of chapter 4. In fact, the basic 2D NoC synthesis methodologies present in literature were discussed. We proposed 2D NoC synthesis solution based on LP and spatial coregraph partitioning. We generated free NoC topologies optimized for area and delay tested with different benchmarks.

We presented in chapter 5, the 3D NoC synthesis methodologies already proposed in literature. We then detailed the 3D Tezzaron technology properties to perform a suitable 3D NoC synthesis methodology. We presented in this part our new 3D NoC synthesis methodology with 3D Tezzaron technology. Our proposed solution is full parallel 3D NoC synthesis solution taking in consideration all the 3D NoC synthesis sub-problems simultaneously.

We presented in chapter 6, a state of the art of real 3D MPSOC architecture. A set of 3D Hardware Accelerator were summarized. These 3D architectures are suitable in the image processing field where we need to transfer a huge amount of data with a high frequency.

Chapter 7 was the subject of parallel EDA methodology. We presented a parallel implementation of an MPSOC with 64PEs on a multi FPGA board. A basic MPSOC design with 16 processors and 16 slaves has been duplicated four times then connected with a hierarchical level. In order to evaluate the different options of the 3D Tezzaron methodology we performed a DSE on the used EDA tools. The place and route algorithms behind the cadence tools are not dedicated for 3D IC. The proposed 3D Tezzaron technology depends basically from the efficiency of the synthesis, place and route performances.

We presented in chapter 8, a real 3D IC design implementation of our MPSOC architecture with 16 processors and 16 memories. We performed a comparison between two different MPSOC with different NoC topologies (Mesh, Butterfly). Thanks to its symmetry, the mesh based NoC architecture is easier to implement compared to the other one based on the Butterfly NoC. In fact with this one, we have to deal with additional problems like mapping and partitioning. With its long links, the Butterfly architecture is a better example than the mesh topology to prove the efficiency of 3D design.

The main parts of this work are basically classified into two different families: technical and research. In fact the experimental or the practical operation is the first step to

define the properties and the challenges of the used methodology. The research step is directly affected by the obtained results. That is why we can present the main contribution of this work as follow:

- A design of MPSOC architecture (16PEs, 16 memories) based on a Butterfly NoC. This design was implemented on FPGA emulator Plateform (Zebu -UF4, Zebu-Server).
- A design of a hierarchical MPSOC design (64PEs, 64 memories) by the multiplication of an MPSOC elementary design.
- A proposition of a 2D NoC synthesis solution based on linear Programming.
- A comparison between the different 2D/3D NoC synthesis methodologies.
- A proposition of a new 3D NoC synthesis methodology based on MOGA. The parallelism and the simultaneous of the sub-problems resolution represent the main originality of this work.
- A real 3D ASIC design implementation of our 3D MPSOC architecture using 3D Tezzaron technology. A comparison between different 3D MPSOC architectures based on Mesh and Butterfly NoCs.

We believe that this work is a common platform to address other important issues such as reconfigurability, models of programming and convergence of disciplines.

3D IC design technology was the main motivation of this PHD studies. In this work we have performed a real 3D ASIC real implementation using the face-to-face 3D Tezzaron technology. In the future works, we are targeting the development of multi layer chips with more developed techniques (Ex: 4 layers, use of the TSV, other stacking techniques). The use of more than 2 layers in the 3D IC design can reveal new challenges like the use of the TSV and the choice of the stacking technique.

Another perspective of this work is to implement the extension of our hierarchical MPSOC architecture based 64 PEs to create a large scale design with 256 PEs. Such architecture will need a powerful machine to ensure the EDA tools functionality but it will be a real challenge to test the efficiency and the limitations of the 3D Tezzaron workflow. With a large scale design there will be a real need to increase the parallel tasks in the 3D IC methodology.

With the high theoretical complexity of the 3D NoC synthesis problem, we propose in the future works to perform a Design Space Exploration on the different 3D MPSOC designs. The goal of this work is to define the critical variables affecting the 3D chip performance. When we define those parameters, we can reduce the Design Space Exploration by choosing the most important options.

The use of heterogeneous technologies represents one important motivation behind the 3D IC technique. In the future works we propose to implement heterogeneous 3D chips taking in consideration the advantages of this methodology in term of area, cost and power consumption. The implementation of the processors and the memories in separate layers and with different technologies can be an interesting experimental case study.

After this first 3D IC implementation experience we believe more and more that it is not possible to study or to propose a theoretical 3D IC solution without having a deep and a real knowledge of the 3D physical techniques. It is meaningless to propose a 3D ASIC methodology which is only based on theoretical studies far from the fabrication reality.

During this thesis we were faced to a major problem which is the limitations of the actual EDA tools. Thanks to the real 3D ASIC design implementations we have proved that it is not any more possible to perform a manual Place and Route with the evolution of the MPSOC designs. The EDA tools perform these steps without taking in consideration the architectural design information lost after the logical synthesis operation. It is necessary to create a new EDA tool applying a design aware 3D ASIC workflow.





## References

- [1] R.Fischbach, J.Lienig, and T.Meister, "From 3D circuit technologies and data structures to interconnect prediction," *Proceedings of the 11th international workshop on System level interconnect prediction ACM*, pp. 77-84, 2009.
- [2] [www.itrs.com](http://www.itrs.com).
- [3] ITRS, "The International Technology Roadmap Semiconductors ," 2011.
- [4] L.Benini, "MPSoCs – Hardware platforms," ARTIST2 / UNU IIST , 2007.
- [5] E.F.Alonso, D.C.Rufas, J.Joven, and J.Carrabina, "Survey of NoC and Programming Models Proposals for MPSoC," *IJCSI International Journal of Computer Science Issues*, vol. 9, no. 3, pp. 1694-0814, March 2012.
- [6] B.Ackland and al, "A single-chip, 1.6-billion, 16-b MAC/s multiprocessor DSP," *IEEE J. Solid-State Circuits* , no. 35, pp. 412–424, 2000.
- [7] S Dutta, R Jenson, and A Rieckmann, "'Viper: A multiprocessor SOC for advanced set-top box and digital TV systems," *IEEE Des. Test. Comput*, vol. 18, no. 5, pp. 21-31, October 2001.
- [8] *C-5TM Network Processor Data Sheet Supporting C-5 Network Processor Version D0*, 4004th ed.: North Andover, 2002.
- [9] M.Adilett and al, "The Next Generation of Intel IXP Network Processors," *Intel Technology Journal*, vol. 6, no. 3, pp. 6-18, August 2002.
- [10] TI, "OMAP5912 : OMAP5912 Applications Processor," Texas Instruments , Literature Number: SPRS231B, 2003.
- [11] J.Goodacre and A.N. Sloss, "Parallelism and the ARM instruction set architecture," *Computer* 38, vol. 38, no. 7, pp. 42-50, July 2005.
- [12] ST NOMADIK, "'Power management for multimedia processor," 2006, technical article STw4810, 2006.
- [13] L.A.Plana and al, "SpiNNaker: Design and Implementation of a GALS Multicore System-on-Chip," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 7, no. 4, p. 18, December 2011.
- [14] K.Donghyun and K.Kawanho, "81.6 GOPS Object Recognition Processor Based on a Memory-Centric NoC," *Congres International Symposium on Networks-on-Chip (NOCS)*, vol. 17, no. 3, pp. 370-383, 2009.
- [15] F.Clermidy, R.Lemaire, X.Popon, D. Kténas, and Y.Thonnart, "An Open and Reconfigurable Platform for 4G Telecommunication: Concepts and Application," *Euromicro Conference on*

*Digital System Design (DSD)*, pp. 449-456, August 2009.

- [16] Cortex-a 15. [Online]. [Cortex-a 15: http://www.arm.com/products/processors/cortex-a/cortex-a15.php](http://www.arm.com/products/processors/cortex-a/cortex-a15.php)
- [17] "Tile 64: TILE64™ Processor – Product Brief," 2008. [Online]. [Tile 64: TILE64™ Processor – Product Brief, 2008](#)
- [18] A.Kumar, S.Fernando, Yajun Ha, B. Mesman, and H.Corporaal, "Multi-Processor System-Level Synthesis for Multiple Applications on Platform FPGA," *Field Programmable Logic and Applications, 2007. FPL 2007. International Conference on*, pp. 92-97.
- [19] A.Kumar, A.Hansson, J. Huisken, and H.Corporaal, "An FPGA Design Flow for Reconfigurable Network-Based Multi-Processor Systems on Chip," *Proc. Design, Automation & Test in Europe Conference & Exhibition. (DATE)*, pp. 1-6, 2007.
- [20] [Online]. [Silicon hive: http://www.silicon-hive.com](http://www.silicon-hive.com)
- [21] A.Hansson, K.Goossens, and M.Bekooij, "CoMPSoC: A Template for Composible and Predictable Multi-Processor System on Chips," *ACM Transactions on Design Automation of Electronic Systems*, vol. 14, no. 1, 2009.
- [22] H.Javaid and S. Parameswaran, "A Design Flow for Application Specific Heterogeneous Pipelined Multiprocessor Systems," *DAC '09 Proceedings of the 46th Annual Design Automation Conference*, pp. 250-253, 2009.
- [23] L.Benini, "Application Specific NoC Design," *DATE '06 Proceedings of the conference on Design, automation and test in Europe: Proceedings*, vol. 1, no. 1-5, pp. 491-495, March 2006.
- [24] A.Samahi and E.B.Bourennane, "Automated Integration and Communication Synthesis of Reconfigurable MPSoC Platform," *Proceeding AHS '07 Proceedings of the Second NASA/ESA Conference on Adaptive Hardware and Systems*, pp. 379-385, Aug 2007.
- [25] X.Li and O.Hammami, "An automatic design flow for data parallel and pipelined signal processing applications on embedded multiprocessor with NoC: application to cryptography," *Int. J. Reconfig. Comput*, vol. 2009, no. 5, p. 1, January 2009.
- [26] X.Li and O. Hammami, "BB-762: Design and Implementation of 762 Processor Multiprocessor and OCP-IP Benchmarking," *Date*, 2009.
- [27] C. J. Alpert, J.H. Huang, and A. B. Kahng, "Multilevel Circuit Partitioning," *In Proceedings of the 1998 IEEE/ACM international conference on Computer-aided design (ICCAD '98)*, pp. 505-511, 1997.
- [28] K.Shahookar and P.Mazumder, "VLSI cell placement techniques," *ACM Comput. Surv.* 23, vol. 23, pp. 143-220, June 1991.

- [29] W.Yu-Liang, T.Shuji, M.Dowska, and M.Marek-Sa, "On Computational Complexity of a Detailed Routing Problem in Two-Dimensional FPGAs," *VLSI, 1994. Design Automation of High Performance VLSI Systems. GLSV '94, Proceedings., Fourth Great Lakes Symposium*, pp. 70-75, March 1994.
- [30] xilinx. [www.xilinx.com](http://www.xilinx.com).
- [31] O.Hammami, X.Li, L.Burgun, and S.Delerse, "Automatic Embedded Multicore Generation and Evaluation Methodology: a Case Study of a NOC Based 2400-cores on Very Large Scale Emulator," *WARP - 5th Annual Workshop on Architectural Research Prototyping*, 2010.
- [32] arteris. [www.arteris.com](http://www.arteris.com).
- [33] eve. [Online]. <http://www.eve-team.com/>
- [34] C.Harris and M.Stephens, "A Combined Corner And Edge Detector," *Proceedings of the 4th Alvey Vision Conference*, pp. 147-151, 1988.
- [35] X.Chen, Z.Lu, A.Jantsch, and S.Chen, "Speedup Analysis of Data-parallel Applications on Multi-core NoCs," *ASIC, ASICON '09. IEEE 8th International Conference*, pp. 105 - 108, 20-23 Oct. 2009 2009.
- [36] modefrontier. Modefrontier. [Online]. <http://www.modefrontier.com>
- [37] K.Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *Evolutionary Computation, IEEE Transactions on*, vol. 6, pp. 182 - 197 , Apr 2002.
- [38] R.Trapani Possignolo and O.Hammami, "Optimized joint NARX ANN - embedded processor design methodology," *ICECS*, pp. 499-502, 13-16 Dec 2009.
- [39] G. Metze, M. Khbels, N. Goldsman, and B.Jacob, "Heterogeneous integration," *Tech Trend Notes*, vol. 12, no. 3, p. 3, 2003.
- [40] D. Velenis, M. Stucchi, E. J. Marinissen, B. Swinnen, and E. Beyne, "Impact of 3D design choices on manufacturing cost, 3D System Integration," *3DIC IEEE International Conference*, pp. 1-5, 2009.
- [41] M. Yasumoto, H. Hayama, and T. Enomoto, "Promising new fabrication process developed for stacked LSIs," *Int Electron Devices Meeting*, p. 816-819, Dec 1984.
- [42] Y.Akasaka and T.Nishimura, "Concept and Basic Technologies for 3-D IC Structure," *IEEE Proceedings of International Electron Devices Meetings*, vol. 32, pp. 488-491, 1986.
- [43] L.Zhou, C.Wakayama, N.Jangkrajarn, H.Bo, and C. J. R.Shi, "A high-throughput low power fully parallel 1024 bit 1/2 -rate low density parity check code decoder in 3D integrated circuits,"

- Design Automation Asia and South Pacific Conference*, p. 2, 2006.
- [44] R. Jenkal, S. Melamed and W. R. Davis C. Mineo, "Inter-die signaling in three dimensional integrated circuits," *Custom Integrated Circuits Conference CICC IEEE*, pp. 655-658, 2008.
- [45] J.Ouyang and al, "Arithmetic unit design using 180nm TSV-based 3D stacking technology," *3D System Integration 2009. 3DIC 2009. IEEE International*, pp. 1-4, Sept 2009.
- [46] T. Zhu and W. R. Davis X. Chen, "Three dimensional SRAM design with on-chip access time measurement ," *Electronics Letters* , no. 47, pp. 485-486, 2011.
- [47] X.Jing, D.Xiangyu, and X. Yuan, "3D memory stacking for fast checkpointing/restore applications," *3D Systems Integration Conference (3DIC) 2010 IEEE International*, pp. 1-6, 2010.
- [48] M.B.Healy and al, "Design and analysis of 3D-MAPS: A many-core 3D processor with stacked memory," *Custom Integrated Circuits Conference (CICC),2010 IEEE*, pp. 1-4, 2010.
- [49] T.Thorolfsson, K.Gonsalves, and P. D.Franzon, "Design automation for a 3DIC FFT processor for synthetic aperture radar: A case study," *Design Automation Conference DAC ACM/IEEE*, pp. 51-56, 2009.
- [50] M.H.Jabbar and D.Houzet, "3D Architecture Implementation: A Survey," *IP-SOC, IP EMBEDDED SYSTEM Conference*, Dec 2011.
- [51] T.Zhang and al, "A 3D SoC design for H.264 application with on-chip DRAM stacking," *3D Systems Integration Conference (3DIC), 2010 IEEE International*, pp. 1-6, 2010.
- [52] J.Soon-Moon and al, "Three Dimensionally Stacked NAND Flash Memory Technology Using Stacking Single Crystal Si Layers on ILD and TANOS Structure for Beyond 30nm Node," *Electron Devices Meeting, 2006. IEDM '06. International* , pp. 1-4, Dec 2006.
- [53] IBM. <http://www-03.ibm.com>.
- [54] W.R.Davis and al, "Demystifying 3D ICs: The Pros and Cons of Going Vertica," *Design & Test of Computers, IEEE* , vol. 22, no. 6, pp. 498-510, Nov 2005.
- [55] T.Jiang and L.Shijian, "3D Integration-Present and Future," *Electronics Packaging Technology Conference, 2008. EPTC 2008. 10th*, pp. 373-378, 2008.
- [56] M.Karnezos, "3-D Packaging: Where All Technologies Come Together," *Proceedings of the IEEE/SEMI International Electronics Manufacturing Technology Symposium*, pp. 64-67, July 2004.
- [57] F.Pavlidis.Vasilis and G.Friedman. Eby, *Three-Dimensional Integrated Circuit Design.*: Morgan Kaufman Publishers, 2009.

- [58] H.Hübner and al, "Micro Contacts with Sub-30 $\mu$ m Pitch for 3D Chip-on-Chip Integration," *MAM*, March 2006.
- [59] K.Takahashi and al, "Ultra-high-density interconnection technology of three-dimensional packaging," *Microelectronics Reliability*, vol. 43, pp. 1267-1279, 2003.
- [60] A.Yu and al, "Study of 15 $\mu$ m pitch solder microbumps for 3D IC integration," *Electronic Components and Technology Conference, 2009. ECTC 2009. 59th*, pp. 6-10, May 2009.
- [61] IMEC. [Online]. <http://www.imec.be/ScientificReport/SR2010>
- [62] C.T.Ko and K.N.Chen, "Wafer-level bonding/stacking technology for 3D integration," *Microelectronics Reliability*, vol. 50, no. 4, pp. 481-488, April 2010.
- [63] A.Papanikolaou, D.Soudris, and R.Radojic, *Three Dimensional System Integration IC Stacking Process and Design.*: Springer, 2011.
- [64] X.Gagnard and T. Mourier, "Through silicon via: From the CMOS imager sensor wafer level package to the 3D integration," *Microelectronic Engineering*, vol. 87, no. 3, pp. 470-476, March 2010.
- [65] T.Ohba and al, "Thinned wafer multi-stack 3DI technology," *Microelectronic Engineering*, vol. 87, pp. 485-490, 2010.
- [66] E.J.Marinissen and Y. Zorian, "Testing 3D chips containing through-silicon vias," *Test Conference, 2009. ITC 2009. International*, pp. 1-11, 2009.
- [67] K. Abe, S. Fujita, Y. Kurosawa and A.Kageshima K. Nomura, "Performance analysis of 3D-IC for multi-core processors in sub-65nm CMOS technologies," *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium*, pp. 2876-2879, 2010.
- [68] V.Gerousis, "Physical Design Implementation for 3D IC: Methodology and Tools," *In Proceedings of the 19th international symposium on Physical design (ISPD '10)*, pp. 57-57, 2010.
- [69] R.Weerasekera, D.Pamunuwa, Z. Li-Rong, and H. Tenhunen, "Two-Dimensional and Three-Dimensional Integration of Heterogeneous Electronic Systems Under Cost, Performance, and Technological Constraints," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, no. 28, pp. 1237-1250, 2009.
- [70] E.Culurciello and A.G.Andreou, "Capacitive Inter-Chip Data and Power Transfer for 3-D VLSI," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 53, pp. 1348-1352, 2006.
- [71] V.F.Pavlidis and E.G.Friedman, "3-D Topologies for Networks-on-Chip," *SOC Conference, 2006 IEEE International*, pp. 285-288, Sept 2006.

- [72] A.Braun, "3D Integration in Design and Test Support," Semiconductor International (<http://www.semiconductor.net>), CA6615469, 2009.
- [73] M.Santarini, "Thermal integrity: A must for low-power IC digital design," *EDN*, pp. 37–42, Sept 2005.
- [74] E.Wong. and L. Sung Kyu, "3D Floorplanning with Thermal Vias," *Design, Automation and Test in Europe, 2006. DATE '06. Proceedings*, pp. 1-6, 2006.
- [75] E.J.Marinissen and Y.Zorian, "Testing 3D chips containing through-silicon vias," *International Test Conference, 2009. ITC*, pp. 1-11, 2009.
- [76] GSA, "3D-IC Design tools and services Tour Guide," DAC, 2011.
- [77] Yole developpement, "3DIC & TSV ProfilesDatabase,Database & Company Profiles Report on the "Top 50" players developing 3D TSV Technologies," Yole developpement, 2008.
- [78] K.Srinivasan, K.S. Chatha, and G. Konjevod, "Linear-Programming-Based Techniques for Synthesis of Network-on-Chip Architectures," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* , vol. 14, no. 4, pp. 407-420, April 2006.
- [79] M.Jun, S.Yoo, and E.Y Chung, "Topology Synthesis of Cascaded Crossbar Switches," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 28, no. 6, pp. 926-930, June 2009.
- [80] M.Jun, S.Yoo, and E.Y Chung, "Mixed integer linear programming-based optimal topology synthesis of cascaded crossbar switches," *Design Automation Conference, 2008. ASPDAC 2008. Asia and South Pacific*, pp. 583-588, March 2008.
- [81] A.A.Morgan, H.Elmiligi, K.M.El-Kharashi, and F.Gebali, "Area and delay optimization for Networks-on-Chip architectures using Genetic Algorithms," *Design and Test Workshop (IDT), 2009 4th International* , pp. 1-6, Nov 2009.
- [82] G.Leary, K.Srinivasan, K.Mehta, and K.S.Chatha, "Design of Network-on-Chip Architectures With a Genetic Algorithm-Based Technique," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 17, no. 5, pp. 674-687, May 2009.
- [83] X.Li and O.Hammami, "Multi-objective topology synthesis and FPGA prototyping framework of application specific network-on-chip," *In Proceedings of the 21st edition of the great lakes symposium on Great lakes symposium on VLSI (GLSVLSI '11)*, pp. 55-60, 2011.
- [84] Y.Bei, D.Sheqin, C.Song, and S.Goto, "Floorplanning and Topology Generation for Application-Specific Network-on-Chip," *Design Automation Conference (ASP-DAC), 2010 15th Asia and South Pacific*, pp. 535 – 540, Jan 2010.
- [85] V.Dumitriu and G.N.Khan, "Throughput-Oriented NoC Topology Generation and Analysis for High Performance SoCs," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* , vol.

- 17, no. 10, pp. 1433-1446, Oct 2009.
- [86] X.Li and O.Hammami, "Multi-objective Network-on-Chip synthesis with transaction level simulation," *Microelectronics (ICM), 2010 International Conference on Microelectronics*, pp. 487 – 490, 2010.
- [87] S.Murali and al, "Designing Application-Specific Networks on Chips with Floorplan Information," *Computer-Aided Design, 2006. ICCAD '06. IEEE/ACM International Conference*, pp. 355-362, Nov. 2006.
- [88] J.Lee and L.Shannon, "Predicting the performance of application-specific NoCs implemented on FPGAs," *In Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays (FPGA '10). ACM*, pp. 23-32, 2010.
- [89] A.Kumar, S.Fernando, Y.Ha, B.Mesman, and H.Corporaal, "Multiprocessor systems synthesis for multiple use-cases of multiple applications on FPGA," *ACM Trans. Des. Autom. Electron*, vol. 3, no. 13, p. 27, July 2008.
- [90] J.Xu, W.Wolf, J.Henkel, and S.Chakradhar, "A design methodology for application-specific networks-on-chip," *ACM Trans. Embed. Comput. Syst.*, pp. 263-280, May 2006.
- [91] X.Jiang and T.Watanabe, "An efficient 3D NoC synthesis by using genetic algorithm," *TENCON 2010 - 2010 IEEE Region 10 Conference*, pp. 1207-1212, November 2010.
- [92] W.Zhong, S.Chen, F.Ma, T.Yoshimura, and S. Goto, "Floorplanning driven Network-on-Chip synthesis for 3-D SoCs," *Circuits and Systems (ISCAS), 2011 IEEE International Symposium*, pp. 1203-1206, May 2011.
- [93] P.Zhou, P.H.Yuh, and S.S.Sapatnekar, "Application-specific 3D Network-on-Chip design using simulated allocation," *Design Automation Conference (ASP-DAC), 2010 15th Asia and South Pacific*, pp. 517-522, Jan 2010.
- [94] P.Zhou, P.H.Yuh, and S.S.Sapatnekar, "Optimized 3D Network-on-Chip Design Using Simulated Allocation," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 17, no. 2, p. 19 pages, April 2012.
- [95] S.Yan and B.Lin, "Design of application-specific 3D Networks-on-Chip architectures," *Computer Design, 2008. ICCD 2008. IEEE International Conference on*, pp. 142-149, Oct 2008.
- [96] S.Murali, C.Seiculescu, L.Benini., and G.De Micheli, "Synthesis of networks on chips for 3D systems on chips," *Design Automation Conference ASP-DAC 2009. Asia and South Pacific*, pp. 242-247, 19-22 Jan 2009.
- [97] S.Murali, L.Benini,G. De Micheli C.Seiculescu, "SunFloor 3D: A Tool for Networks on Chip Topology Synthesis for 3-D Systems on Chips," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions*, vol. 29, no. 12, pp. 1987-2000, December 2010.

- [98] R.E Tarjan, "Depth-first search and linear graph algorithms," *Switching and Automata Theory, 1971., 12th Annual Symposium on*, pp. 114-121, 13-15 October 1971.
- [99] I.Loi, F.Angiolini, and L.Benini, "Supporting vertical links for 3D networks-on-chip: toward an automated design and analysis flow," *Proceedings of the 2nd international conference on Nano-Networks (Nano-Net '07). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium*, p. 5, 2007.
- [100] S.Borkar, "3D-Technology: A System Perspective," in *International 3D System Integration Conference*, 2008.
- [101] X.Dong and Y. Xie, "System-level cost analysis and design exploration for three-dimensional integrated circuits (3D ICs)," .. In *Proceedings of the 2009 Asia and South Pacific Design Automation Conference (ASP-DAC '09). IEEE Press, Piscataway, NJ,USA* , pp. 234-241, 2009.
- [102] S.Cheramly and al, "3D integration process flow for set-top box application: Description of technology and electrical results," *Microelectronics and Packaging Conference, 2009. EMPC 2009. European* , pp. 1-6, 15-18 June 2009.
- [103] J.Cong and al, "Domain-specific processor with 3D integration for medical image processing," *Application-Specific Systems, Architectures and Processors (ASAP), 2011 IEEE International Conference*, pp. 247-250, 11-14 Sept 2011.
- [104] K.Hamwi and O.Hammami, "64-Node Butterfly Network: Design and Implementation on Multi-FPGA," *5th International Design & Test Workshop IDT'10, Abu-Dhabi*, 14-16 Dec 2010.
- [105] xilinx, "Microblaze processor reference guide ," UG081 (v13.2), 2011.
- [106] <http://www.tezzaron.com/>.
- [107] R.S.Patti, "Three-Dimensional Integrated Circuits and the Future of System-on-Chip Designs," *Proceedings of the IEEE*, vol. 94, no. 6, pp. 1214 - 1224 , june 2006.
- [108] K.Puttaswamy and G.H.Loh, "Thermal Herding: Microarchitecture Techniques for Controlling Hotspots in High-Performance 3D-Integrated Processors," *High Performance Computer Architecture, 2007. HPCA IEEE 13th International Symposium*, pp. 193-204, 10-14 Feb 2007.
- [109] [Online]. [www.sonics.com](http://www.sonics.com)
- [110] G. Qun, X. Zhiwei, K. Jenwei, and C. Mau-Chung Frank, "Two 10Gb/s/pin Low-Power Interconnect Methods for 3D ICs," *Solid State Circuits Conference, 2007. ISSCC Digest of Technical Papers. IEEE International*, pp. 448-614, 2007.
- [111] Dae Hyun Kim, S.Mukhopadhyay, and Sung Kyu Lim, "TSV-aware interconnect length and power prediction for 3D stacked ICs," *Interconnect Technology Conference, 2009. IITC 2009. IEEE International*, pp. 26-28, 2009.



- [112] J.Yang, K.Athikulwongse, Y.Lee, S.K.Lim, and D.Z.Pan, "TSV stress aware timing analysis with applications to 3D-IC layout optimization," *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, pp. 803-806, June 2010.
- [113] J.H. Lau, "TSV manufacturing yield and hidden costs for 3D IC integration," *Electronic Components and Technology Conference (ECTC), 2010 Proceedings 60th*, pp. 1031-1042, June 2010.
- [114] Y.Jang, J. Kim, and C.M Kyung, "Topology Synthesis for Low Power Cascaded Crossbar Switches," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* , vol. 29, no. 12, pp. 2041-2045, 2010.
- [115] Y.Ar, S.Tosun, and H.Kaplan, "TopGen: A new algorithm for automatic topology generation for Network on Chip architectures to reduce power consumption," *Application of Information and Communication Technologies, 2009. AICT 2009. International Conference*, pp. 1-5, Oct 2009.
- [116] E.Beyne and al, "Through-silicon via and die stacking technologies for microsystems-integration," *Electron Devices Meeting, 2008. IEDM 2008. IEEE International*, pp. 1-4, Dec 2008.
- [117] D.Cuesta and al, "Thermal-aware floorplanning exploration for 3D multi-core architectures," *In Proceedings of the 20th symposium on Great lakes symposium on VLSI (GLSVLSI '10).*, pp. 99-102, 2010.
- [118] G.M.Link and N.Vijaykrishnan, "Thermal trends in emerging technologies,Quality Electronic Design," *ISQED '06. 7th International Symposium on*, p. 632, March 2006.
- [119] G.M.Link and N.Vijaykrishnan, "Thermal trends in emerging technologies," *Quality Electronic Design, 2006. ISQED '06. 7th International Symposium on* , p. 8 , March 2006.
- [120] C. Chiang and S. Sinha, "The road to 3D EDA tool readiness," *Design Automation Conference, 2009. ASP-DAC 2009. Asia and South Pacific*, pp. 429-436, 2009.
- [121] A.M'zah, O.Hammami, and J.Mouine, "The Impact of EDA Tools in 3D IC Design Space Exploration: A Case Study," *DATE 2012 Workshop on 3D Integration Applications, Technology, Architecture, Design, Automation, and Test*, 2012.
- [122] R.Tligue, Y.Aydi, M. Baklouti, M.Abid, and J.L. Dekeyser, "The design methodology and the implementation of MPSOC based on Delta MINs on FPGA," *Microelectronics International Conference (ICM)*, pp. 221-224, 2009.
- [123] Y. Chen , K.Chakrabarty , Y.Xie X.Wua, "Test-access mechanism optimization for core-based," *Microelectronics Journal*, vol. 41, no. 10, pp. 601-615, July 2010.
- [124] A.K.Coskun, A.B. Kahng, and T.S.Rosing, "Temperature- and Cost-Aware Design of 3D Multiprocessor Architectures," *Digital System Design, Architectures, Methods and Tools, 2009. DSD '09. 12th Euromicro Conference on*, pp. 183-190, Aug 2009.

- [125] S.Schliecker and al, "System Level Performance Analysis for Real-Time Automotive Multi-Core and Network Architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 7, pp. 979-992, 2009.
- [126] S.Yan, "Synthesis of Application-Specific On-Chip Networks," University of California, San Diego, Phd 2009.
- [127] K.Soo Hyun, S.Pasricha, and C.Jeonghun, "POSEIDON: A framework for application-specific Network-on-Chip synthesis for heterogeneous chip multiprocessors," *Quality Electronic Design (ISQED), 2011 12th International Symposium on*, pp. 1-7, 2011.
- [128] Y.J.Chen, C.L. Yang, and P.H.Wang, "PM-COSYN: PE and Memory Co-Synthesis for MPSoCs," *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1590 – 1595, 2010.
- [129] K.Nomura, K.Abe, S.Fujita, Y.Kurosawa, and A.Kageshima, "Performance analysis of 3D-IC for multi-core processors in sub-65nm CMOS technologies," *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pp. 2876-2879, May-June 2010.
- [130] M.Ebrahimi, M.Daneshtalab, P.Liljeberg, and H.Tenhunen, "Performance Analysis of 3D NoCs Partitioning Methods," *VLSI (ISVLSI), 2010 IEEE Computer Society Annual Symposium on*, pp. 479-480, July 2010.
- [131] D.Milojevic, R.Radojicic, R.Carpenter, and P.Marchal, "Pathfinding: A design methodology for fast exploration and optimisation of 3D-stacked integrated circuits," *System-on-Chip, 2009. SOC 2009. International Symposium on*, pp. 118-123, 2009.
- [132] T. Dorta, J. Jiménez, J.L. Martín, U. Bidarte, and A. Astarloa, "Overview of FPGA-Based Multiprocessor Systems," *International Conference on Reconfigurable Computing and FPGAs*, pp. 273-278, 2009.
- [133] O.Ozturk, F.Wang, M.Kandemir, and Y.Xie, "Optimal topology exploration for application-specific 3D architectures," *Design Automation, 2006. Asia and South Pacific Conference on*, p. 6, Jan 2006.
- [134] P.Emma and E.Kursun, "Opportunities and Challenges for 3D Systems and Their Design," *Design & Test of Computers, IEEE*, vol. 26, no. 5, pp. 6-14, Sept.-Oct 2009.
- [135] J.Chan and S. Parameswaran, "NoCOUT : NoC topology generation with mixed packet-switched and point-to-point networks," *Design Automation Conference, 2008. ASPDAC 2008. Asia and South Pacific*, pp. 265-270, March 2008.
- [136] D. Bertozzi and al, "NoC synthesis flow for customized domain specific multiprocessor systems-on-chip," *IEEE Trans. Parallel and Distributed Systems*, vol. 16, no. 2, pp. 113–129, 2005.
- [137] B.S.Feero and P.P.Pande, "Networks-on-Chip in a Three-Dimensional Environment: A Performance Evaluation," *Computers, IEEE Transactions on*, vol. 58, no. 1, pp. 32-45, 2009.

- [138] T.Huang, Y.Chen, J.Hu, and X.Ling, "Networks-on-Chip Emulator Design with FPGA Array," *Communications, Circuits and Systems (ICCCAS)*, pp. 886 – 890, 2010.
- [139] D.Atienza and al, "Network-On-Chip Design and Synthesis Outlook," *Integration-The VLSI journal*, vol. 41, no. 3, pp. 340-359, 2008.
- [140] W. Wolf, A.A. Jerraya, and G. Martin, "Multiprocessor System-on-Chip (MPSoC) Technology," *IEEE Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, no. 10, pp. 1701-1713, 2008.
- [141] A.Fourmigue, G.Beltrame, G.Nicolescu, E.M.Aboulhamid, and I.O'Connor, "Multi-granularity thermal evaluation of 3D MPSoC architectures," *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011* , pp. 1-4, March 2011.
- [142] M.Abouelatta-Ebrahima, R.Dahmani, O.Valorgea, F.Calmon, and C.Gontran, "Modelling of through silicon via and devices electromagnetic coupling," *Microelectronics Journal*, vol. 42, no. 2, pp. 316-324, Feb 2011.
- [143] L.Ost et al., "Model-based design flow for NoC-based MPSoCs," *ICECS 2010, The 17th IEEE International Conference on Electronics, Circuits and Systems*, pp. 750-753, Dec 2010.
- [144] A.Janarthanan and K.A.Tomko, "MoCSYS: A Multi-Clock Hybrid Two-Layer Router Architecture and Integrated Topology Synthesis Framework for System-Level Design of FPGA Based On-Chip Networks," *VLSI Design, 2008. VLSID 2008. 21st International Conference on*, pp. 397-402, 2008.
- [145] A.B.Abderazek, M.Akanda, T.Yoshinaga, and M. Sowa, "Mathematical Model for Multiobjective Synthesis of NoC Architectures," *Parallel Processing Workshops, 2007. ICPPW 2007. International Conference on* , p. 36, Sept 2007.
- [146] G.Chen and E.G.Friedman, "Low-power repeaters driving RC and RLC interconnects with delay and bandwidth constraints," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 14, no. 2, pp. 161-172, Feb 2006.
- [147] X.Li and O.Hammami, "Linear Programming Based Design of Reconfigurable Network on Chip on eFPGA," *Electronics, Circuits and Systems, 2008. ICECS 2008. 15th IEEE International Conference* , pp. 782 – 785, 2008.
- [148] K.Srinivasan and K.S.Chatha, "ISIS: a genetic algorithm based technique for custom on-chip interconnection network synthesis," *VLSI Design, 2005. 18th International Conference on*, pp. 623- 628, Jan 2005.
- [149] W.L.Hung, G.M.Link, Yuan Xie, N.Vijaykrishnan, and M.J.Irwin, "Interconnect and thermal-aware floorplanning for 3D microprocessors," *Quality Electronic Design, 2006. ISQED '06. 7th International Symposium on*, pp. 6-14, March 2006.

- [150] Chia-Jen Chang, Pao-Jen Huang, Tai-Chen Chen, and C.-N.J. Liu, "ILP-based inter-die routing for 3D ICs," *Design Automation Conference (ASP-DAC), 2011 16th Asia and South Pacific*, pp. 330-335, Jan 2011.
- [151] <http://www.zy-cube.com/e/index.html>.
- [152] <http://www.ziptronix.com/>.
- [153] <http://www.statschippac.com/>.
- [154] <http://www.sharppusa.com>.
- [155] Openfire Processor Core. [http://opencores.org/project,openfire\\_core](http://opencores.org/project,openfire_core).
- [156] M.Koyanagi, T.Fukushima, and T.Tanaka, "High-Density Through Silicon Vias for 3-D LSIs," *Proceedings of the IEEE*, vol. 97, no. 1, pp. 49-59, Jan 2009.
- [157] S.Lin, L.Su, H.Su, D.Jin, and L.Zeng, "Hierarchical Cluster-Based Irregular Topology Customization for Networks-on-Chip," *Embedded and Ubiquitous Computing, 2008. EUC '08. IEEE/IFIP International Conference on*, vol. 1, pp. 373-377, Dec 2008.
- [158] N.Choudhary, M.S.Gaur, V. Laxmi, and V.Singh, "Genetic algorithm based topology generation for application specific Network-on-Chip," *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pp. 3156-3159, June 2010.
- [159] G.Lai, X.Lin, and S.Lai, "GA-based floorplan-aware topology synthesis of application-specific network-on-chip," *Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on*, vol. 2, pp. 554-558, Oct 2010.
- [160] N.Choudhary, M.S.Gaur, V. Laxmi, and V.Singh, "GA based Congestion Aware Topology Generation for Application Specific NoC," *Electronic Design, Test and Application (DELTA), 2011 Sixth IEEE International Symposium on*, pp. 93 – 98, 2011.
- [161] A.Kumar and P.R. Panda, "Front-End Design Flows for Systems on Chip: An Embedded Tutorial," *VLSI Design VLSID '10. 23rd International Conference*, pp. 417 – 422, 2010.
- [162] H. Oprins and al, "Fine grain thermal modeling and experimental validation of 3D-ICs," *Microelectronics journal*, vol. 42, no. 4, pp. 572-578, April 2011.
- [163] S.Secchi, P. Meloni, and L. Raffo, "Exploiting FPGAs for technology-aware system-level evaluation of multi-core architectures," *IEEE International Symposium on Performance Analysis of Systems and Software - ISPASS*, pp. 194 - 202, 2010.
- [164] W.Chen and X.Han and R.Domer, "ESL Design and Multi-Core Validation using the System-on-Chip Environment," *High Level Design Validation and Test Workshop (HLDVT), 2010 IEEE International*, pp. 142 – 147, 2010.

- [165] P.G.DelValle and D.Atiienza, "Emulation-based transient thermal modeling of 2D/3D systems-on-chip," *Microelectronics journal*, vol. 42, no. 2, pp. 564-571, April 2011.
- [166] I.Savidis and al, "Electrical modeling and characterization of through-siliconvias(TSVs) for 3-D integratedcircuits," *Microelectronics Journal*, vol. 41, pp. 9-16, Jan 2010.
- [167] IBM, "Efficient Modeling with CPLEX Studio," IBM Advanced Analytics Summit , April 2011.
- [168] B.Black and al, "Die Stacking (3D) Microarchitecture," *In Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 39)*. IEEE Computer Society, Washington, DC, USA, , pp. 469-479, 2006.
- [169] S.Murali, L.Benini, and G.De Micheli, "Design of Networks on Chips for 3D ICs," *Design Automation Conference (ASP-DAC), 2010 15th Asia and South Pacific* , pp. 167-168, Jan 2010.
- [170] G. Van der Plas and al, "Design issues and considerations for low-cost 3D TSV IC technology," *Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pp. 148-149, Feb 2010.
- [171] J.Wu, J.Williams, N.Bergmann, and P.Sutton, "Design Exploration for FPGA-based Multiprocessor Architecture: JPEG Encoding Case Study," *Field Programmable Custom Computing Machines FCCM '09*, pp. 299 – 302, April 2009.
- [172] S.Yan and B.Lin, "Custom Networks-on-Chip Architectures With Multicast Routing," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* , vol. 17, no. 3, pp. 342-355, March 2009.
- [173] S.K.Kim, C.C.Liu, L.Xue, and S.Tiwari, "Crosstalk reduction in mixed-signal 3-D integrated circuits with interdevice layer ground plane," *Electron Devices, IEEE Transactions on (2005)*, vol. 52, no. 7, pp. 1459-1467, July 2005.
- [174] P.D.Franzon, W.R.Davis, and T.Thorolffson, "Creating 3D specific systems: Architecture, design and CAD," *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1684-1688, 2010.
- [175] X.Wu and al, "Cost-driven 3D integration with interconnect layers," *Design Automation Conference (DAC), 2010 47th ACM/IEEE* , pp. 150-155, June 2010.
- [176] C. Patterson and P. Athanas S. Craven, "Configurable Soft Processor Arrays Using the OpenFire Processor," *Proceedings of the 8th Annual Conference on Military and Aerospace Programmable Logic Devices MAPLD*, 2005.
- [177] M.Lukasiewicz, M.Streubuhr, M M.Glass, C. Haubelt, and J.Teich, "Combined system synthesis and communication architecture exploration for MPSoCs," *Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE '09*, pp. 472-477, April 2009.
- [178] F.Escolano, E.R.Hancock, and M.A. Lozano, "Birkhoff polytopes, heat kernels and graph complexity," *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pp. 1-5,

Dec 2008.

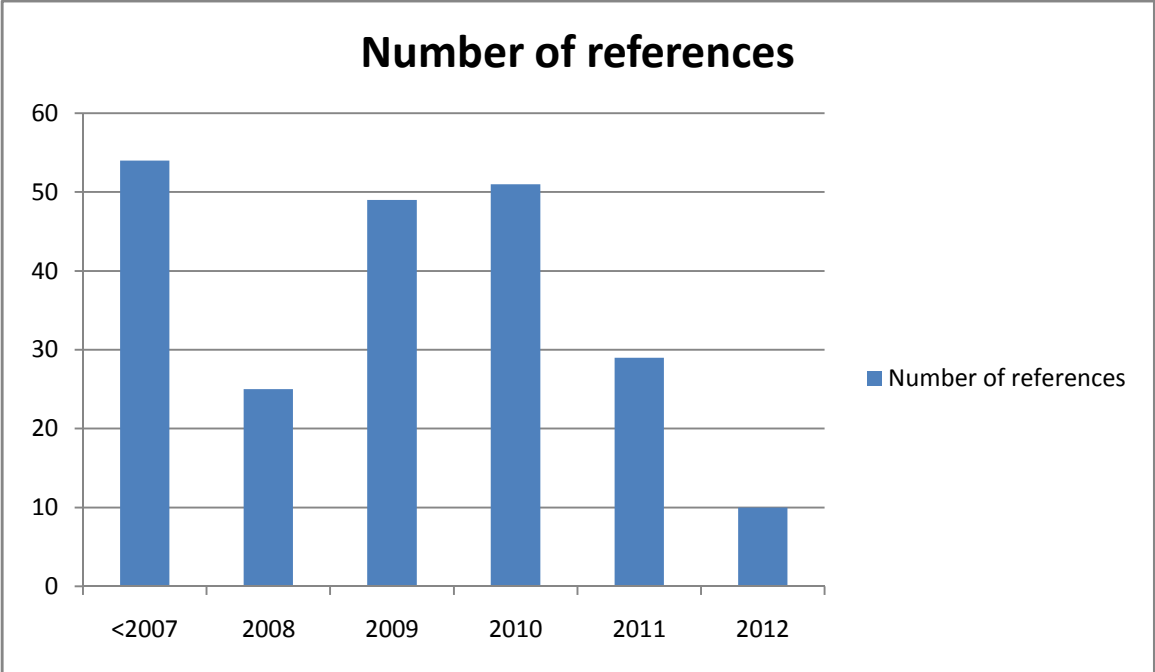
- [179] O.Hammami, X.Li, L.Larzul, and L.Burgun, "Automatic design methodologies for MPSoC and prototyping on multi-FPGA Platforms," *SoC Design Conference (ISOCC)*, pp. 141 – 146, 2009.
- [180] K.S.Chathak, K.Srinivasan, and G. Konjevod, "Automated Techniques for Synthesis of Application-Specific Network-on-Chip Architectures," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, no. 8, pp. 1425-1438, Aug 2008.
- [181] B.Vaidyanathan and al, "Architecting Microprocessor Components in 3D Design Space," *In Proceedings of the 20th International Conference on VLSI Design held jointly with 6th International Conference: Embedded Systems (VLSID '07). IEEE Computer Society, Washington, DC, USA, 1*, pp. 103-108, 2007.
- [182] H.Ishebabi and al, "Answer Set versus Integer Linear Programming for Automatic Synthesis of Multiprocessor Systems from Real-Time Parallel Programs," *Hindawi Publishing Corporation International Journal of Reconfigurable Computing* , vol. 2009, p. 11.
- [183] I.Loi and L.Benini, "An efficient distributed memory interface for many-core platform with 3D stacked DRAM," *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 99-104, March 2010.
- [184] G.Palermo, C.Silvano, and V.Zaccaria, "An Efficient Design Space Exploration Methodology for On-Chip Multiprocessors Subject to Application-Specific Constraints," *Application Specific Processors, 2008. SASP 2008. Symposium on* , pp. 75-82, 2008.
- [185] L. Fiorin S. Lukovic, "An Automated Design Flow for NoC-based MPSoCs on FPGA," *Rapid System Prototyping, RSP '08 The 19th IEEE/IFIP International Symposium*, pp. 58 – 64, 2008.
- [186] J.Cong, Jie Wei, and Yan Zhang, "A thermal-driven floorplanning algorithm for 3D ICs," *Computer Aided Design, 2004. ICCAD-2004. IEEE/ACM International Conference on*, pp. 306-313, Nov 2004.
- [187] Dae Hyun Kim, K.Athikulwongse, and Sung Kyu Lim, "A study of Through-Silicon-Via impact on the 3D stacked IC layout," *Computer-Aided Design - Digest of Technical Papers, 2009. ICCAD 2009. IEEE/ACM International Conference on*, pp. 674-680, 2009.
- [188] J.Rabindra Ku. and S. Gopal Ku, "A Multi-Objective Evolutionary Algorithm Based Optimization Model for Network-on-Chip Synthesis," *Information Technology, 2007. ITNG '07. Fourth International Conference on*, pp. 977-982, 2007.
- [189] A.Pinto, L.P.Carloni, and A.LSangiovanni-Vincentelli, "A Methodology for Constraint-Driven Synthesis of On-Chip Communications," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* , vol. 28, no. 3, pp. 364-377, March 2009.
- [190] K.Srinivasan and K.S.Chatha, "A Low Complexity Heuristic for Design of Custom Network-on-Chip Architectures," *Design, Automation and Test in Europe, 2006. DATE '06. Proceedings* , vol.

- 1, pp. 1-6, March 2006.
- [191] X.Qin P. Yang, "A hybrid optimization approach for chip placement of multi-chip," *Microelectronics Journal*, vol. 40, pp. 1235-1243, 2009.
- [192] G.Leary and K.S.Chatha, "A holistic approach to Network-on-Chip synthesis," *Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2010 IEEE/ACM/IFIP International Conference on*, pp. 213-222, Oct 2010.
- [193] A.K.Singh and al, "A Design Space Exploration Methodology for Application Specific MPSoC Design," *VLSI (ISVLSI), 2011 IEEE Computer Society Annual Symposium*, pp. 339 – 340, 2011.
- [194] K.Goossens and al, "A design flow for application-specific networks on chip with guaranteed performance to accelerate SOC design and verification," *In Proceedings of the conference on Design, Automation and Test in Europe, IEEE Computer Society*, vol. 2, pp. 1182 - 1187, 2005.
- [195] V.Tyree, "3DIC multi-project-wafer program: A collaboration to provide fabrication access," *3D Systems Integration Conference (3DIC), 2010 IEEE International (2010)*, pp. 1 – 17, Nov 2010.
- [196] J.U.Knickerbocker and al, "3D silicon integration," *Knickerbocker, J.U.; Andry, P.S.; Dang, B.; Horton, R.R.; Patel, C.S.; Polastre, R.J.; Sakuma, K.; Sprogis, E.S.; Tsa Electronic Components and Technology Conference, 2008. ECTC 2008. 58th*, pp. 538-543, May 2008.
- [197] P. Marchal, A. Pullini, L. Benini I. Loi, "3D NoCs Unifying inter and intra chip communication," *Circuits and Systems (ISCAS)*, pp. 3337-3340, 2010.
- [198] A.Zia, S. Kannan, H.J.Chao, and G.S.Rose, "3D NOC for many-core processors," *Microelectronics Journal*, vol. 42, no. 12, pp. 1380-1390, December 2011.
- [199] S. Borkar, "3D integration for energy efficient system design," *In Proceedings of the 48th Design Automation Conference (DAC '11). ACM*, pp. 214-219, 2011.
- [200] L.Xue, F.Shi, W.Ji, and H.Khan, "3D floorplanning of low-power and area-efficient Network-on-Chip architecture," *Microprocessors and Microsystems*, vol. 35, no. 5, pp. 484–495, 2011.
- [201] T.Dorta, J. Jiménez, J.L. Martín, and U. Bidarte, "Reconfigurable Multiprocessor Systems: A Review," *Hindawi Publishing Corporation International Journal of Reconfigurable Computing Volume*, vol. 2010, no. 7, p. 10, 2010.
- [202] J.Dally W and Towles B, "Principles and Practices of Interconnection Network," *Morgan Kaufmann publishers*, 2004.
- [203] H.Javaid, X.He, A.Ignjatovic, and S. Parameswaran, "Optimal synthesis of latency and throughput constrained pipelined MPSoCs targeting streaming applications," *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hard/software codesign and system synthesis (CODES/ISSS '10)*, pp. 75-84.

- [204] A.A Jerraya, G. Martin W. Wolf, "Multiprocessor System-on-Chip (MPSoC) Technology," *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems - TCAD* , vol. 27, no. 10, pp. 1701-1713, 2008.
- [205] A.Samahi and M.Boukadoum, "Improved MPSoC Co-Design Methodology for Stream Oriented Processing Applications," *Electronics, Circuits, and Systems (ICECS), 2010 17th IEEE International Conference*, pp. 754 – 757, 2010.
- [206] N.Genko, D.Atiienza, G. De Micheli, and L.Benini, "Feature - NoC emulation: a tool and design flow for MPSoC," *Circuits and Systems Magazine, IEEE Issue Date : Fourth Quarter 2007*, no. 7, pp. 42 – 51, 2007.
- [207] <http://www.sharputa.com/>.
- [208] [Online]. [www.arteris.com](http://www.arteris.com)
- [209] O.He, J.S.Yang, D.Z. Pan W.Jang, "Chemical-mechanical polishing aware application-specific 3D NoC design," *In Proceedings of the International Conference on Computer-Aided Design (ICCAD '11). IEEE Press, Piscataway, NJ*, pp. 207-212, 2010.
- [210] S.Pasricha, "A Framework for TSV Serialization-aware Synthesis of Application Specific 3D Networks-on-Chip," *In Proceedings of the 2012 25th International Conference on VLSI Design (VLSID '12). IEEE Computer Society, Washington, DC, USA* , pp. 268-273, 2012.
- [211] D.Matos and al, "Floorplanning-aware design space exploration for application-specific hierarchical networks on-chip," *In Proceedings of the 4th International Workshop on Network on Chip Architectures (NoCArc '11). ACM, New York, NY, USA* , pp. 31-36, 2011.
- [212] O. Hammami, A. M'zah, and K. Hamwi, "Design of 3D-IC for Butterfly NOC Based 64 PE-Multicore: Analysis and Design Space Exploration," *IEEE International 3D System Integration Conference (3DIC)*, January 31-February 2012.
- [213] N. Kapadia and S. Pasricha, "A Power Delivery Network Aware Framework for Synthesis of 3D Networks-on-Chip with Multiple Voltage Islands," *In Proceedings of the 2012 25th International Conference on VLSI Design (VLSID '12). IEEE Computer Society, Washington, DC, USA* , pp. 262-267, 2012.
- [214] Y.Xie, J.Cong, and S.Sapatnekar, "Three-Dimensional Integrated Circuit Design: EDA, Design and Microarchitecture," *Springer*, 2010.
- [215] M.H.Jabbar, O.Hammami D.Houzet, "3D IC 2-tier 16PE Multiprocessor with 3D NoC Architecture Based on Tezzaron Technology," *IP-SOC* , 2011.
- [216] A.Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency.*: Springer, 2002.
- [217] A.Marschner, S. D. Craven, and P. M. Athanas, "A Sandbox for Exploring the OpenFire



- Processor," *ERSA* , pp. 248-251, 2007.
- [218] LogiCORE IP, "LogiCORE IP Fast Simplex Link (FSL) ," DS449 V20 (v2.11c), April 19, 2010.
- [219] V.Fresse , F.Rousseau J.Tan, "Generation of emulation platforms for NoC exploration on FPGA," *Rapid System Prototyping (RSP), 2011 22nd IEEE International Symposium on* , pp. 186-192 , 24-27 May 2011.
- [220] F.Rousseau,F.Petrot,al A.ELMrabti, "Design environment for the support of configurable Network Interfaces in NoC-based platforms," *Embedded Computer Systems (SAMOS), 2010 International Conference on*, pp. 63 - 70, 19-22 July 2010.
- [221] G.Tsiligiannis and L.Pierre, "A Mixed Verification Strategy Tailored for Networks on Chip," *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on*, pp. 161 - 168, 9-11 May 2012.
- [222] V.Lefftz and al, "A design flow for critical embedded systems," *Industrial Embedded Systems (SIES), 2010 International Symposium on*, pp. 229- 233, 7-9 July 2010.
- [223] M. Tawk, K.Z. Ibrahim, and S. Niar, "Parallel application sampling for accelerating MPSoC simulation," *presented at Design Autom. for Emb. Sys.*, pp. 367-387, 2010.
- [224] R.BenAtitallah, É.Piel, S.Niar, P.Marquet, and J.L.Dekeyser, "A fast MPSoC virtual prototyping for intensive signal processing applications," *Microprocessors and Microsystems - Embedded Hardware Design*, vol. 36, no. 3, pp. 176-189, 2012.
- [225] M. Bakhouya, S. Suboh, J. Gaber, T. El Ghazawi, and S. Niar, "Performance evaluation and design tradeoffs of on-chip interconnect architectures," *Simulation Modelling Practice and Theory*, vol. 19, no. 6, pp. 1496-1505, June 2011.
- [226] H.Liu, S.Niar, Y.El-Hillali, and A.Rivenq, "Embedded architecture with hardware accelerator for target recognition in driver assistance system," *SIGARCH Comput. Archit*, pp. 56-59, December 2011.
- [227] A.Ahmad, A.Amira, M.Guarisco, H.Rabah, and Y.Berviller, "Efficient implementation of a 3-D medical imaging compression system using CAVLC," *ICIP* , pp. 3773-3776, 2010.
- [228] B.Courtois, "SOC/SIP for energy management," *International SoC Design Conference (ISOC), 2009*.
- [229] A.Ahmad, A.Amira, M.Guarisco, H.Rabah, and Y.Berviller, "Efficient implementation of a 3-D medical imaging compression system using CAVLC," *ICIP*, vol. , pp. 3773-3776, 2010.
- [230] <http://www.modefrontier.com>. Modefrontier. [Online]. <http://www.modefrontier.com>



## List of Publications

### International conferences

1. A.M'zah ; O.Hammami; , "Parallel programming and speed up evaluation of a NoC 2-ary 4-fly," IEEE International Microelectronics (ICM), 2010 International Conference on , vol., no., pp.156-159, 19-22 Dec. 2010
2. A.M'zah; O.Hammami , "Area/delay driven NoC synthesis," IEEE International Microelectronics (ICM), 2011 International Conference on , vol., no., pp.1-6, 19-22 Dec. 2011
3. A.M'zah, O. Hammami and J. Mouine, "The Impact of EDA Tools in 3D IC Design Space Exploration: A Case Study", DATE 2012 Workshop on 3D Integration Applications, Technology, Architecture, Design, Automation, and Test, Dresden, Germany, March, 2012
4. O.Hammami, A. M'zah and K.Hamwi, " Design of 3D-IC for Butterfly NOC Based 64 PE-Multicore: Analysis and Design Space Exploration", IEEE International 3D System Integration Conference (3DIC) January 31-February 2, 2012, Osaka, Japan
5. O.Hammami, A.M'zah, M.H.Jabbar and D.Houzet, ' ' 3D IC Implementation for MPSOC Architectures: Mesh and Butterfly Based NoC'', ASQED 2012, Kuala Lumpur, Malaysia

### National conferences

6. A.M'zah,O.Hammami,' ' La Technologie Tezzaron pour la Conception d'une Architecture Multiprocesseurs 16PEs en 3D IC ' ', GDR SOC-SIP 2012, Paris
7. O.Hammami, K.Hamwi, M.H.Jabbar M.Khaddour,A.M'zah, "Design and Implementation of NOC Based 16 PE", GDR SOC-SIP 2011, Lyon

### Industrial conferences

8. A.M'zah and O.Hammami, "Multi-FPGA Design Based 64 PE with NoC Extension'', IP-SOC 2010 Conference & Exhibition - IP Based Electronic System, Grenoble 30 November – 1 December 2010
9. O.Hammami, A.M'zah, K.Hamwi, " Design of 3D-IC for Butterfly NOC Based 64 PE-Multicore: Analysis and Design Space Exploration'', IP-SOC 2011 Conference & Exhibition - IP Based Electronic System, Grenoble 2011

### Journal

10. A.M'zah and O.Hammami, "NOSYTE : Multi Objective 3D NoC Synthesis with Tezzaron Technology'', submitted to Journal of Microelectronics

