

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

**SOFT ERROR MITIGATION IN
ASYNCHRONOUS NETWORKS ON CHIP**

JULIAN JOSÉ HILGEMBERG PONTES

Thesis presented as a partial requirement to obtain the degree of Doctor in Computer Science at the Pontifícia Universidade Católica do Rio Grande do Sul.

Advisor: Prof. Dr. Ney Laert Vilar Calazans
Porto Alegre
August 2012

Dados Internacionais de Catalogação na Publicação (CIP)

P814s Pontes, Julian José Hilgemberg
Soft error mitigation in asynchronous networks on chip / Julian
José Hilgemberg Pontes. – Porto Alegre, 2012.

143 f.

Tese (Doutorado) – Fac. de Informática, PUCRS.

Orientador: Prof. Dr. Ney Laert Vilar Calazans.

1. Informática. 2. Circuitos Assíncronos. 3. Arquitetura de
Redes. 4. Confiabilidade de Sistemas. I. Calazans, Ney Laert Vi-
lar. II. Título

CDD 004.6

**Ficha Catalográfica elaborada pelo
Setor de Tratamento da Informação da BC-PUCRS**



Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE TESE DE DOUTORADO

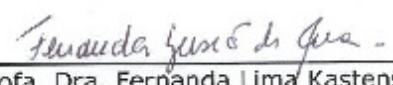
Tese intitulada "*Soft Error Mitigation in Asynchronous Networks on Chip*", apresentada por Julian José Hilgemberg Pontes, como parte dos requisitos para obtenção do grau de Doutor em Ciência da Computação, Sistemas Embarcados e Sistemas Digitais, aprovada em 28/08/2012 pela Comissão Examinadora:


Prof. Dr. Ney Laert Vilar Calazans -

PPGCC/PUCRS


Prof. Dr. Fernando Gehm Moraes -

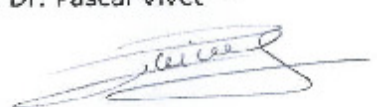
PPGCC/PUCRS

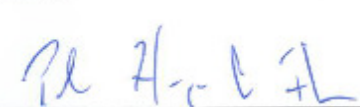

Profa. Dra. Fernanda Lima Kastensmidt -

UFRGS

Prof. Dr. Pascal Vivet -

CEA/LETI - França


Homologada em 13/11/2012, conforme Ata No. 24..... pela Comissão Coordenadora.


Prof. Dr. Paulo Henrique Lemelle Fernandes
Coordenador.

PUCRS

Campus Central

Av. Ipiranga, 6681 - P. 32 - sala 507 - CEP: 90619-900

Fone: (51) 3320-3611 - Fax (51) 3320-3621

E-mail: ppgcc@pucrs.br

www.pucrs.br/facIn/pos

Agradecimentos

Agradeço primeiramente a Deus: além de ser a fonte de todo o conhecimento, tenho certeza que foi ele quem colocou no meu caminho tantos amigos e companheiros tão importantes. Sem essas pessoas tão importantes, esse trabalho não seria possível.

Durante todo o caminho que percorri estudando computação/eletrônica (Eletricista Industrial (Senai) – Técnico em Eletrônica (Cefet) - Engenheiro de Computação (UEPG) – Mestre e agora Doutor em Ciência da Computação (PUCRS)) tive diversos amigos e diversos mestres que me guiaram e me ajudaram a chegar até esse ponto importante do meu caminho. Agradeço a todos, e em especial a minha Família que esteve sempre presente em todos os meus passos. Agradeço e dedico essa Tese aos meus pais Zaclis e Luiz José. Aos meus irmãos: Andrea, Emerson, Tais e Marcos. A minha mulher, Patricia Loren que sempre me apoiou e me ajudou. Amo muito todos vocês.

Agradeço ao Professor Ney Calazans que foi meu orientador durante seis anos e meio e que durante todo esse tempo sempre acreditou no meu trabalho e sempre me apontou o caminho e respeitou minhas decisões, mesmo quando eu escolhia os caminhos mais difíceis (como no final do Doutorado quando decidi mudar todos os caminhos que já havíamos planejado na proposta de tese). Obrigado Ney!

Durante minha estadia em Porto Alegre fiz grandes amigos. Gostaria de agradecer muito a esses amigos, especialmente aos meus grandes amigos Ewerson Carvalho, Edson Moreno, Ricardo Souza, Daniel Ferrão e Guilherme Rodhe. Ao meu grande amigo Odorico Mendizabal e a Cristina Meinhardt.

Agradeço aos meus amigos do laboratório GAPH da PUCRS pelo apoio e ajuda. De novo agradeço aos meus grandes amigos Edson Moreno e Ewerson Carvalho. Ao Rafael Soares e Matheus Trevisan (companheiros na pesquisa de assíncronos), ao Luciano Ost e Gabriel Marchesan que me ajudaram muito durante a minha estadia na França. Ao Eduardo Wachter e ao Bruno Oliveira. Aos Professores Alexandre Amory, Marcon e principalmente ao Professor Fernando Moraes pela grande contribuição e paciência ao avaliar os meus trabalhos de assíncronos.

Agradeço aos amigos do CEITEC, com quem aprendi muitas coisas que me auxiliaram no desenvolvimento dessa Tese. Gostaria de agradecer principalmente a equipe do DIGITAL: Daniel Ferrão, Guilherme Rodhe, Marcos Herve, Marcelo Moraes, Alam Menezes, Cyrille Lambert, Wagner Silvério e Rafael Garibotti.

Aos meus companheiros da Banda Rock Raccoon: Ossanai (vocal), Ricardo (baixo), Athos (te-

clado), nas guitarras: Odorico, Cesar e Scrico. Na verdade, a banda atrapalhou muito essa Tese, mas em contrapartida, me proporcionou sempre bons momentos que ajudaram a suportar todo esse período.

Je tiens à remercier mes amis du CEA/LETI: Jani, Tanuj, Santhosh, Alejandro, Guillaume et Bilel. Je suis sûr que mon séjour à Grenoble a été bien plus agréable parce que je vous ai rencontré. Finalement, je dois beaucoup à Pascal Vivet, qui m'a énormément aidé dans ces deux dernières années, depuis notre première rencontre à PATMOS 2010. Merci de croire en moi et d'être, dans la pratique, le directeur adjoint de cette thèse.

RESUMO

O aumento agressivo das frequências de operação de sinais de relógio em tecnologias submicrônicas profundas chegou ao seu limite. O uso de relógios globais não é mais viável em tais tecnologias, o que fomenta a popularização do paradigma Globalmente Assíncrono, Localmente Síncrono na construção de sistemas integrados complexos, onde se empregam ilhas síncronas de lógica interconectadas através de comunicação assíncrona. Redes intrachip assíncronas proveem um modelo de comunicação baseado em troca de pacotes e paralelismo de comunicação escalável quando comparado com arquiteturas de comunicação tradicionais, como as baseadas em barramentos compartilhados. Devido a estas características, tal tipo de redes vem revelando benefícios, quando comparadas com suas equivalentes síncronas, para construir as arquiteturas *many-cores* do futuro, e isto em termos de ambos, desempenho e dissipação de potência. Um dos próximos desafios para as arquiteturas de comunicação em questão é a confiabilidade, na forma de robustez a **efeitos de evento único** (em inglês, *single event effects* ou SEEs), quando o circuito sofre impactos de partículas geradas por radiação ionizante. Isto ocorre porque a diminuição contínua das geometrias de dispositivos semicondutores em tecnologias sucessivas aumenta cada vez mais a sensibilidade destes a tais efeitos. Ao contrário do que ocorre em circuitos síncronos, variações de atraso induzidas por radiação em geral não geram qualquer impacto, exceto por possíveis perdas de desempenho, em circuitos lógicos assíncronos construídos usando técnicas **quase insensíveis a atrasos** (em inglês *quasi-delay insensitive* ou QDI). Contudo, a inversão de valores de bits em dispositivos de armazenamento pode corromper o estado do circuito sem possível solução de recuperação, mesmo no caso de assíncronos. Este trabalho propõe um novo conjunto de técnicas aplicáveis a redes intrachip assíncronas, que visa o aumento de robustez contra efeitos de evento único. Apresentam-se estudos de caso práticos de tais técnicas e avaliam-se as mesmas em ambientes que simulam casos reais de uso. Os resultados obtidos mostram que o aumento de robustez alcançado sobre redes intrachip tem o potencial de tornar esta arquitetura de comunicação a principal candidata para integrar as novas gerações de dispositivos de silício complexos construídos com o emprego de nodos tecnológicos avançados tais como 32 nm, 28 nm, 20 nm e abaixo.

ABSTRACT

In advanced deep submicron technologies, the aggressive scaling of the clock to increasingly higher frequencies has now terminated. At the circuit top level, global clocking is not feasible anymore, which has led to the popularization of the Globally Asynchronous Locally Synchronous paradigm for constructing complex system on chip devices, with local islands of clocked logic interconnected by asynchronous communication. By providing packet-based communication and scalable communication parallelism compared to traditional bus-based communication, asynchronous network-on-chip have recently shown their benefits compared to their synchronous counterparts to build future many-core architectures, in terms of both performance and power. One of the next challenges for such asynchronous communication architectures is reliability, in the form of robustness to **single event effects**, when under the impact of particles generated by ionizing radiation. This occurs because technology downscaling continuously increases the logic sensitivity of silicon devices to such effects. Contrary to what happens in synchronous circuits, delay variations induced by radiation usually have no impact on asynchronous **quasi-delay insensitive** (QDI) combinational logic blocks, but in case of storage logic, bit flips may corrupt the circuit state with no recovery solution, even when using asynchronous circuits. This work proposes a new set of hardening techniques against single event effects applicable to asynchronous networks-on-chip. It presents practical case studies of use for these techniques and evaluates them in close to real life situations. The obtained results show that the achieved increase in asynchronous network-on-chip robustness has the potential to leverage this communication architecture solution as the main choice for the next generations of complex silicon devices on advanced nodes technologies such as 32 nm, 28 nm, 20 nm and below.

LIST OF FIGURES

Figure 1 – Interaction of an alpha particle with a semiconductor device.	30
Figure 2 – A PMOS transistor junction in reverse bias crossed by an energetic particle.	31
Figure 3 – Examples of NoC topologies: (a) 2D mesh; (b) 2D torus; (c) ring; (d) tree [80].	33
Figure 4 – Example of a deadlock situation in a unidirectional (clockwise) ring topology NoC with five routers and five IP Cores. A packet from IP0 to IP4 managed to allocate buffer space in routers 0, 1 and 2, while at the same time a packet from IP3 to IP1 managed to allocate buffer space in routers 3, 4 and 5. None of the packets can move further, characterizing a deadlock.	36
Figure 5 – Symbol of a 2-input C-element.	38
Figure 6 - Push channel-based communication using a handshake protocol.	39
Figure 7 – Pull channel-based communication using a handshake protocol.	39
Figure 8 – Two-phase handshake protocol.	39
Figure 9 – Four-phase handshake protocol examples.	40
Figure 10 – Bundle data generic circuit structure, displaying the local handshake circuit, the combinational part (Logic) and the storage element (L).	41
Figure 11 – Early data four-phase handshake protocol for the DI Code data transfer.	42
Figure 12 - Data transmission using dual rail encoding and a four-phase protocol.	43
Figure 13 – A Dual rail completion detector for a code that can represent values of a vector of two bits. .	43
Figure 14 - Two-bit weak conditioned half buffer (WCHB).	45
Figure 15 – Typical asynchronous pipeline stage example.	45
Figure 16 –A XOR DIMS gate for a one-bit dual rail code.	46
Figure 17 – The fork component data flow representation.	47
Figure 18 - Example of a possible fork circuit implementation.	47
Figure 19– Data flow representation of the Join component.	47
Figure 20 - Example of a possible Join component implementation.	48
Figure 21 – Merge component data flow representation.	48
Figure 22 - Example of a Merge implementation.	48
Figure 23 – Dual rail multiplexer symbol.	49
Figure 24 – Example of a dual rail multiplexer implementation.	49
Figure 25 – Demultiplexer symbol and implementation in Dual Rail.	49
Figure 26 – Example of a dual rail demultiplexer implementation.	50
Figure 27 – The block diagram of an if/else module.	50
Figure 28 – S-element symbol.	50
Figure 29 – State transition graph representation for the behavior of the S-element component.	51
Figure 30 – Symbol and interface of an arbiter.	51
Figure 31 – Arbiter implementation.	52
Figure 32 – Four input arbiter built as an arrangement of six two input arbiters.	52
Figure 33- A 4-input multiplexer controlled by an arbiter.	53
Figure 34 – A 3x3 NoC mesh topology. The numbers in the router denote the X and Y address of the router.	63
Figure 35 – Router architecture adopted in this work and the IP Core connection through the Network Interface.	64
Figure 36 – A packet with four flits example.	64
Figure 37 – A one-of-four FIFO composed by weak conditioned buffers.	65
Figure 38 - Input port block diagram.	67
Figure 39 – Path Definition architecture.	67
Figure 40 – Equivalent code for the Path Definition module.	69
Figure 41 – Block diagram of Section Close module.	69
Figure 42 - Block diagram of a 4→1 output port. The dashed lines represent a single wire while the filled lines are DI encoding.	70
Figure 43 – Output Port Control Circuit.	71
Figure 44 – Waveform example of a two flits packet transmission through an output port.	71
Figure 45 - State transition graph of a C-element under presence of SEE.	74
Figure 46 – Example of the double check technique applied to a three-input one-output logic. In the Figure, Logic A and Logic B are instances of a same circuit and inputs are equivalent.	75

Figure 47 – A possible double check arbiter implementation.	76
Figure 48 – The double check implementation equivalent to the four-input arbiter presented in Chapter 2 and used in the Hermes-A Output Port. Individual arbiters labeled with letter A are equivalent to the circuit presented in Figure 30.	76
Figure 49 – The structure of a double check S-element.....	77
Figure 50 – One of four FIFO with a C-element victim of a SEE.	78
Figure 51 - SEE and timing diagram for the 1-of-n pipeline 4-phase protocol, displaying all possible consequences of SEEs in each phase of the protocol.	79
Figure 52 – Timing and SEE analysis of the m-of-n 4-phase protocol for an asynchronous buffer, displaying all possible consequences of SEEs in each phase of the protocol.	81
Figure 53 – Normal open asynchronous pipeline register for a 1-of-4 pipeline.	83
Figure 54 – Timing and SEE analysis for the NOAP technique.....	84
Figure 55 – Parity check when using 1-of-4 DI code for NoC transmission.....	87
Figure 56 – Communication Architecture using the proposed parity scheme for m-of-n codes.....	87
Figure 57 - Trellis TRDIC encoding based on the 1-of-4 DI encoding.	89
Figure 58 - Overview of the proposed TRDIC communication system.	90
Figure 59 - Trellis TRDIC encoding based on the 1-of-4 DI encoding.	92
Figure 60 – Four-flit packet with double Flit Type signaling.....	94
Figure 61 – Double check BOP demux control.	94
Figure 62 - One of four FIFO with a C-element victim of a SEE.....	98
Figure 63 – Synthesis flow adopted for the asynchronous components that compose the NoC library.....	99
Figure 64 - Proposed flow for digital circuit design based on SEE constraints.	102
Figure 65 - Example of a modified NAND gate for SEE simulation.	103
Figure 66 - Method for SEE characterization of standard cells.	104
Figure 67 - Characterization values of a 2-input NAND gate.....	105
Figure 68 - Fault simulation environment.	106
Figure 69 - SEE adapted Verilog example for a 2-input NAND gate.	106
Figure 70 - Example of a timing library fragment for a 2-input NAND gate with timing arcs from SEE pin to output Z.	107
Figure 71 - Waveform of an SEE example simulation for a 2-input NAND.	108
Figure 72 - Communication between IP Cores described using a CWG.	110
Figure 73 – NoC containing four layers. Each layer is an independent 3x3 mesh NoC.	112
Figure 74 – PARANA NoC synthesis flow.....	113
Figure 75 – Model used to represent the Input Port inside the PARANA tool. In the illustration, one incoming traffic is split in two different outputs due to the dynamic algorithm property.	114
Figure 76 – Model used to represent the output port. The output traffic is the sum of the traffic at the four inputs.....	114
Figure 77 – Illustration of a router composed by heterogeneous ports adapted to a HGDG description..	115
Figure 78 – Relation between the injected charge and the Soft Error Rate for different Output Ports implementations.....	118
Figure 79 – Relationship between the injected charge and the Soft Error Rate for different Input Port implementations.....	119
Figure 80 – Relationship between the number of packet errors and the injected charge for different Output Port implementations.	119
Figure 81 - Relation between the number of packet errors and the injected charge for different Input Port implementations.....	120
Figure 82 – DI code corruption classification for different Output Port implementations. The results are for an injection charge of 100 fC.....	120
Figure 83 - DI code corruption classification for different Input Port implementations. The results are for an injection charge of 100 fC.	121
Figure 84 – Output Port flit corruption evaluation as a function of the injected charge.....	121
Figure 85 – Input Port flit error classification when a 100fC charge is injected.	122
Figure 86 – Evaluation of the stall errors in Output Port as a function of the injected charge.	122
Figure 87 - Evaluation of the stall errors in Input Port in function of the injected charge.....	123
Figure 88 – Packet corruption classification as a function of the injected charge for the Output Port.	123
Figure 89 – Data error evaluation as a function of the SEE injected charge, for the m-of-n Parity correction evaluation at the Output Port.....	124
Figure 90 – CWG for the MMS application adapted for the Hardening Mapping.	126
Figure 96 – Multimedia System after the mapping done with the adapted CWG for hardened mapping..	127
Figure 91 – Soft Error Evaluation Environment used to characterize the Asynchronous NoCs.	128
Figure 92 - Data error evaluation of the NoCs – (a) shows the results for the NoC without hardening technique, (b) shows the results for the NoC with parity correction, (c) shows the results of the hardened NoC and (d) shows the results of the hardened NoC and data correction	128

Figure 93 – VCD error evaluation of the hardened NoC using the TRDIC data correction mechanism..... 129
Figure 94 – Stall Errors evaluation of the NoC without hardening techniques (Hermes-A). 130
Figure 95 – Stall Errors evaluation of the Hardened NoC (H₂A)..... 130

LIST OF TABLES

Table 1 - Truth table for a basic 2-input C-element. 38

Table 2 - Dual rail code with L=2. The data spacer here is represented by the value "00". 43

Table 3 - 1-of-4 encoding used to represent two logical bits. 44

Table 4 - Routing table employed for the source routing decision in the Hermes-A NoC. 68

Table 5 - SEE critical charge to generate a fault according to the current C-element state when it is driving a charge of 8.1 fF. 74

Table 6 - Parity correction operations. 88

Table 7 - State transition table for the implemented TRDIC..... 90

Table 8 - Evaluation of the Output Port implementations. 125

Table 9 - Evaluation of the Input Port implementations. 125

Table 11 - Parity encoder and decoder area evaluation..... 129

Table 10 - Area results for the hardened multilayer NoC, for the adapted Multimedia System application. The Output Port and Input Port columns show the number of used components and the number of the components in a full mesh NoC. 131

LIST OF ACRONYMS

ARQ	Automatic Repeat Request
ASIC	Application Specific Integrated Circuit
BOP	Begin of Packet
CMOS	Complementary Metal-Oxide-Semiconductor
CWG	Communication Weighted Graph
CWM	Communication Weighted Model
DI	Delay Insensitive
DIMS	Delay Insensitive Minterm Synthesis
EMI	ElectroMagnetic Interference
EOP	End of Packet
FEC	Forward Error Correction
HGCG	Hardened GALS Communication Graph
HGCM	Hardened GALS Communication Model
HVT	High Voltage Threshold
IP	Intellectual Property
LET	Linear Energy Transfer
LVT	Low Voltage Threshold
Mbps	Megabits per second
MPSoC	Multiprocessing System on a Chip
NoC	Network on Chip
PCFB	Precharge Full Buffer
PCHB	Precharge Half Buffer
PSO	Power Shut Off
SEE	Single Event Effect

SET	Single Event Transient
SEU	Single Event Upset
QDI	Quasi Delay Insensitive
SEC	Single Error Correction
SECEDED	Single Error Correction Double Error Detection
SoC	System on a Chip
SVT	Standard Voltage Threshold
TMR	Triple Modular Redundancy
WCHB	Weak Conditioned Half Buffer

CONTENTS

RESUMO	9
ABSTRACT	11
LIST OF FIGURES	13
LIST OF TABLES	17
LIST OF ACRONYMS	19
CONTENTS	21
1. INTRODUCTION	23
1.1 Motivation.....	23
1.2 Problem Description	25
1.3 Objectives.....	26
1.4 Original Contributions.....	26
1.5 Document Structure	26
2. BASIC CONCEPTS	29
2.1 Radiation Effects in Digital Systems.....	29
2.1.1 Radiation Effects	29
2.1.2 Critical Charge and Single Event Effects	30
2.1.3 Hardening Techniques for Digital Circuits.....	31
2.2 Networks on Chip.....	33
2.2.1 Topology.....	33
2.2.2 Routing Algorithm	33
2.2.3 Switching Mode.....	34
2.2.4 Quality of Service	36
2.2.5 NoCs for GALS SoCs	36
2.3 Asynchronous Circuits	37
2.3.1 C-elements.....	37
2.3.2 Handshake Protocols	38
2.3.3 Asynchronous Design Styles.....	40
2.3.4 DI Data Encoding	41
2.3.5 Asynchronous Circuits Implementation.....	44
3. RELATED WORKS	55
3.1 Asynchronous NoCs.....	55
3.2 SEE Hardening Techniques for Asynchronous Circuits	56
3.3 Hardened NoCs.....	58
3.4 NoC Synthesis	59
4. THE HERMES-A NOC	63
4.1 Input Port	66
4.2 Output Port	70
5. HARDENED HERMES-A (H₂A)	73
5.1 Cell Level Hardening.....	73
5.1.1 SEE impact on C-elements	73
5.2 Logic Level Hardening	74

5.2.1	Double Check	75
5.2.2	Double Check Arbiter	75
5.2.3	The Double Check S-element.....	76
5.3	Handshake Protocol Hardening	77
5.3.1	SEE Impact on Four-phase, 1-of-n Pipelines.....	78
5.3.2	Normal Open Asynchronous Pipeline (NOAP)	82
5.4	Data Level Hardening	84
5.4.1	QDI Parity Check.....	84
5.4.2	Temporal Redundancy in 1-of-4 DI Codes (1-of-4 TRDIC).....	88
5.5	Packet Protocol Hardening	93
5.5.1	Flit Type Double Checking - Packet Signaling (BOP, EOP)	93
5.5.2	Double Check in Routing Decision.....	94
6.	NOC COMPONENT LIBRARY	97
6.1	NoC Components Synthesis.....	97
6.1.1	Noc Component Description	99
6.2	Robustness Characterization	100
6.2.1	SEE Digital Design Flow	101
6.2.2	SEE Flow Implementation	104
7.	SYNTHESIS OF HARDENED ASYNCHRONOUS NETWORKS ON CHIP.....	109
7.1	Communication Model for Hardened Applications	109
7.2	Layered NoC Architecture	111
7.3	Hardened NoC Synthesis.....	112
8.	RESULTS.....	117
8.1	NoC Components Evaluation.....	117
8.1.1	Injected Charge Evaluation	118
8.2	1-of-4 Parity Implementation	123
8.3	Area, Power and Performance Evaluation	124
8.4	PARANA Synthesis Tool Evaluation	125
8.4.1	Regular NoCs Evaluation.....	127
8.4.2	Layered NoC Evaluation.....	130
9.	CONCLUSIONS AND FUTURE WORK	133
	REFERENCES	135

1. INTRODUCTION

1.1 Motivation

Gordon Moore predicted that the number of transistors per chip would double every 18-24 months. This prediction was done in 1965, before Intel, founded by Moore and others release its first microprocessor, and it holds until nowadays [35]. The first Intel microprocessor, the I4004 was released in 1971 with 2200 transistors. In 2006, Intel announced its first processor with more than one billion transistors [56]. To keep this trend, foundries must keep improving transistor-manufacturing processes, in order to reduce transistor dimension and consequently increase transistor density.

Nowadays, transistor dimensions have reached the stage of deep submicron technology (technologies below 100nm). The consequence is that some physical effects that were neglected in the past now are imposing huge barriers to electronic design. The International Technology Roadmap for Semiconductors (ITRS) refers to this set of physical constraints as *Silicon Complexity* [38].

The effect of *local variations*, the increasing impact of *signal integrity* problems, and the phenomena related to *energetic particle impacts on silicon structures* are examples of these new types of constraints [38]. Signal integrity and process variations already have well-established solutions available in commercial CAD tools. Therefore, they can be treated at design time. Unfortunately, the same is not true for effects caused by energetic particle impacts on silicon.

A high-energy particle (alpha particles from packaging or neutrons from the atmosphere [56]) can cause different types of effects when crossing silicon structures. The resulting effect depends on the *linear energy transfer* (LET) of the particle and on the state of the crossed junction (if any) [56]. When the victim junction is in reverse bias, the generated electron-hole pairs eventually produced by the particle are collected by the electrical field of the reversed junction. The magnitude of the current induced by the particle strike determines if the resulting effect will be permanent or non-permanent. A non-permanent event, also known as Single Event Effect (SEE), is classified as a Single Event Transient (SET), if it only creates a glitch in combinational logic, or a Single Event Upset (SEU), when the glitch is stored by a memory element. The timing deviation, without bit inversion, can be classified also as a non-permanent error if it can generate a timing violation in the circuit. Predictions exist that on advanced nodes technologies, high Single Event Rates (SERs or the rate at which SEEs

occur) will become significant robustness problems for systems on chip (SoCs) [93].

The ITRS also calls attention for a second class of constraints called *System Complexity* [38]. To take full advantage of transistor density and meet time-to-market constraints, SoC design must employ *reuse* in all design steps. Reuse usually implies heterogeneous SoCs, since reusable modules may each have independently defined timing and power constraints.

At the architectural and system levels, there are also set of new design directions. The use of Networks on Chip (NoCs) as communication architectures is a trend to solve problems related to scalability of classical bus-based communication. NoCs may provide high throughput, parallel communication, modularity and scalability [28].

Another trend is the adoption of Globally Asynchronous Locally Synchronous (GALS) [23][42][88] architectures, instead of using global synchronization strategies. This is motivated by the limitations of high frequency global lines and clock distribution [36], and to reduce the high energy consumption originated by clock distribution structures. In a GALS design, the SoC is divided into several clock domains. All data processing inside a clock domain is performed synchronously, locally keeping a synchronous design and timing constraints, which allows using synchronous CAD tools. Synchronization is required though in data transfer between two distinct clock domains [23].

The combined use of the last mentioned trends, NoCs and GALS, is a natural way to help SoC designers building complex SoCs respecting timing-to-market and system constraints related to aspects such as power, timing and area. In this case, NoCs provide synchronous IP Cores communication and synchronization services. Two basic NoC implementation options are possible to enable this:

- NoCs with synchronous routers: in this strategy, all routers have a synchronous implementation. Clock domains can be built in several different ways. For example, the whole NoC may form just a single clock domain [41]. In this case, IP Cores may have a clock different from the NoC clock. This implies that synchronization interfaces need to be present between the NoC and the IP Cores. Another case occurs when the number of clock domains is equal to the number of routers. In this case, each router defines a clock domain and synchronization interfaces are needed between routers and between routers and IP Cores. Between these two extremes, there are several possible combinations of clustering and clock domain creation.
- Fully asynchronous NoC: The use of fully asynchronous NoCs is also a well-explored solu-

tion [18]. In this case, the NoC does not contain any clock domain. When the IP Core is a synchronous circuit, a synchronization interface must be employed between this and any point of the NoC.

Fully asynchronous implementations have some advantages when compared to synchronous or GALS NoC versions. The most important advantage for advanced technology nodes is the reliability under different process (global and local), voltage and temperature (PVT) variations [25]. Asynchronous circuits from the *delay insensitive* (DI) class are immune to delay variations in gates and wires. This property enables such circuits to operate even under extreme PVT variations situations [77]. Another advantage is the reduction in switching activity and in number of simultaneously switching points. Reductions in switching activity bring also reductions in dynamic power, voltage drop (also called IR drop) and electromagnetic interference (EMI). Fully asynchronous NoC implementations may also present high throughput, depending on the adopted asynchronous implementation strategies.

Immunity to delay variations is also a good characteristic for mitigating the effects of energetic particles that traverse silicon dies. The delay variations caused by these energetic particles are, in most situations, not capable of generating mal-functioning. However, when energetic particles do generate a bit flip, asynchronous circuits can be dangerously affected. Since asynchronous circuits operate based on local signaling, a bit flip may change the order of control events and even make the circuit stalling.

1.2 Problem Description

This work focuses on hardening techniques for asynchronous NoCs to support robust GALS systems design. Fully asynchronous NoCs can be an attractive choice due to their intrinsic delay robustness. However, the occurrence of bit flips inside the data path and in the asynchronous control can be disastrous for asynchronous implementations. This work presents a study of Single Event Effects in fully asynchronous NoCs, and proposes some hardening techniques to be applied at several design levels.

The hardening techniques proposed here apply to asynchronous NoCs at the:

1. Circuit Level: to reduce the SEE propagation probability inside the circuit logic gates;
2. Data Level, to ensure data correction;
3. Protocol level: to reduce the probability of stall in asynchronous handshake protocols;

4. Packet Level: to reduce the probability of stall at the NoC link level.

Since asynchronous circuit design is a complex task and few digital designers have in-depth knowledge of the involved techniques, this work proposes a set of methods and algorithms that are integrated in an automatic asynchronous NoC synthesis framework, guided by robustness constraints established by system and interconnect designers.

1.3 Objectives

Based on the motivation exposed before, this work proposes the implementation of a fully asynchronous NoC with hardening techniques to mitigate soft errors arising in an intrachip communication architecture. To achieve this, a set of objectives were defined. The main objectives of this thesis are:

1. To provide a fully asynchronous NoC implementation: to serve as base communication architecture;
2. To propose a set of techniques to increase the robustness of asynchronous NoCs in general and that of the proposed NoC in particular;
3. To propose generic methods to measure the robustness of digital circuits to SEE that are amenable to use in asynchronous circuits.

1.4 Original Contributions

The main contributions of this work are the proposition of hardening techniques for QDI NoCs at multiple levels of NoC design. At the asynchronous protocol level, a modification in the pipeline implementation proposed here brought good results for some NoC components. Another contribution is the data error detection and correction mechanisms for m-of-n DI Codes protection. A framework for SEE modeling at the digital level using commercial CAD tools is another relevant contribution. Finally, the thesis offers a new method to model and synthesize SEE-hardened NoCs, based on communication and robustness constraints.

1.5 Document Structure

The rest of this document comprises eight Chapters with the following contents: Chapter 2 describes some basic concepts about radiation effects, networks on chip and asynchronous circuits. Chapter 3 presents related works about asynchronous NoCs, hardening techniques for asynchronous

circuits, hardened NoCs and NoC synthesis. Chapter 4 presents the asynchronous NoC Hermes-A, which is the base architecture for the work. Chapter 5 presents the Hardened Hermes-A (H2A) NoC. Chapter 6 shows the NoC component library, including the adopted soft error characterization method. This library serves as a repository for the PARANA NoC synthesis tool. The PARANA synthesis tool itself is the subject of Chapter 7. Chapter 8 presents some evaluation of test cases, to validate the proposed hardening techniques, the NoC components and the PARANA tool. Chapter 9 ends the thesis discussing some conclusions and direction for future work.

2. BASIC CONCEPTS

This Chapter brings the definition of some basic concepts about Radiation Effects in digital systems, NoCs and asynchronous circuits. The Radiation Effects are first presented. Next, some concepts about NoCs and asynchronous circuits are also introduced. When appropriate, the impact of SEE is analyzed in NoCs and in asynchronous circuits as well.

2.1 Radiation Effects in Digital Systems

In this Section, the radiation effects in silicon is explained starting from the overview of the electrical phenomenon until the evaluation of the bit flip in digital circuits. Some traditional hardening techniques that are commonly applied to synchronous implementations are described also.

2.1.1 Radiation Effects

There are two main sources of radiation-induced faults in digital circuits – alpha particles from packaging and neutrons from the atmosphere [56]. In advanced technologies, say those below 45nm, the direct ionization caused by low energy protons can be an issue as well [82]. An alpha particle consists of a helium nucleus, with two protons and two neutrons. These particles are emitted in the alpha decay process and have energy of about 5MeV. Inside chips, alpha particles are produced by impurities in packaging. The bumps of the flip-chip packaging have been recently identified as containing significant alpha particle sources [1]. The neutrons that cause soft errors are a product of atom divisions. Neutrons are generated when primary cosmic rays (mainly protons from galactic particles or solar particles) hit the atmospheric atoms, creating a cascade of secondary particles.

Alpha particles cause perturbations in the electrical field of the semiconductor lattice. The result of this perturbation is the generation of electron-hole pairs, as showed in Figure 1. The number of generated electron-hole pairs depends on the energy of the particle and on the *stopping power*.

According to Verma [92], the energy loss per unit path length is commonly called stopping power of the target material for a penetrating ion. The stopping power is in fact a resistive force instead of power and given by the equation:

$$(1) \quad SP = -\frac{dE}{dx}$$

When an incident particle penetrates a material, it loses its energy interacting with atoms. The interactions are usually divided into two separate processes, elastic collisions with sample atom nuclei (nuclear stopping power) and inelastic collisions with electrons (electronic stopping power). The nuclear energy loss dominates in the low-velocity (energy) region but electronic energy loss is much larger at high velocities. In the energy range of 1–3 MeV/u, the energy loss is mainly due to interaction of the ions with the electrons in the material, causing excitation and ionization of the target atoms.

Neutrons do not generate electron-hole pairs, since they do not cause ionization [12]. However, the collision of a neutron with some atoms in the semiconductor generates nuclear fragments, like protons, alpha particles and several others, that can cause ionization tracks and produce electron-hole pairs.

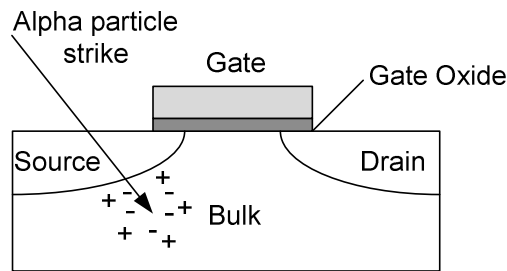


Figure 1 – Interaction of an alpha particle with a semiconductor device.

Ionizing radiation in silicon can cause long-term degradations and the shifting in MOS devices characteristics, due to the cumulative energy absorbed by the matter. Total Ionization Dose (TID) is the term usually used to refer to these effects. TID effects cause the degradation of the circuit due to threshold voltage shifts, increase in the leakage power and change in transistor switching timing. Consequently, TID effects may cause the violation of setup and hold times in synchronous designs. In some asynchronous circuits, this effect is less problematic, since the later are almost immune to timing deviations in gates.

2.1.2 Critical Charge and Single Event Effects

When a particle crosses a semiconductor junction in reverse bias, the electrical field present in the junction can collect the ions generated by the particle strike. If the quantity of the collected ions is bigger than the *Critical Charge* (minimum charge required to create a bit-flip in the transistor node) for the transistor node, the particle generates a Single Event Effect (SEE). Figure 2 shows a

PMOS with the drain node in reverse bias victim of a particle strike. In this case, if the particle has enough LET and the charge collection is efficient to gather enough ions to charge the node capacitance, an event may occur at the PMOS drain node.

SEEs can be divided in two main categories: hard errors and soft errors. Hard errors may cause permanent damage to the MOS device. A soft error can be a pulse transient at some combinational logic device (a Single Event Transient or SET) or a bit-flip in some storage element (a Single Event Upset or SEU).

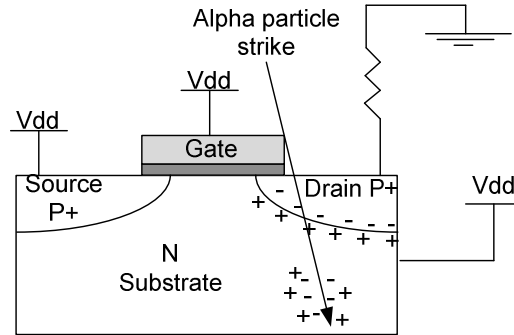


Figure 2 – A PMOS transistor junction in reverse bias crossed by an energetic particle.

In a digital circuit, the Soft Error Rate (or SER) is measured in failures-in-time FIT (1 FIT = 1 failure in one billion device hours) [4]. Several techniques and methods are available in the literature to increase the reliability of digital circuits by reducing the SER [3].

Digital circuits have three fault masking properties that can reduce the SER: electrical filtering, logical filtering and latch window masking [56]. Electrical masking results from the victim node property at the time that it is hit. If the node is not in a reversed bias state, the masking probability increases, due to the absence of an electrical field to collect the charges. If the node is in the reversed bias state, the LET must be high enough to charge/discharge the capacitance at the victim node. If the charge collected by the node it is not enough to create a bit flip or create a timing error (violation in setup/hold times, for example), then the charge was electrically filtered. Logic filtering is a property of the combinational logic. If a SET is generated and it arrives at the input of an AND gate, then the SET can be filtered if the other AND gate input is equal to '0'. Latch masking occurs in synchronous circuits when an SEE is propagated to the input of a memory element but this memory element does not sample the faulty value.

2.1.3 Hardening Techniques for Digital Circuits

Hardening methods usually employ redundancy to achieve higher levels of robustness. The classical *Triple Modular Redundancy* (TMR) scheme is an example of redundant hardening tech-

nique. In the spatial redundancy TMR, three equivalent modules are employed to perform the same task. A voter is employed to decide the correct data output, by comparing if there are at least two equal answers from the three modules. The temporal TMR consists in using the same module performing the same processing three times. Voters are used again to decide the appropriate answer. In data communication, coding techniques employ redundant information to enable error detection and/or correction.

Data detection needs less redundancy than data correction. In this way, in systems where the receiver can ask for retransmission (half-duplex or full-duplex systems), this can be a good choice. Communications systems that use retransmission are referred to as Automatic Repeat ReQuest systems [55]. In the ARQ system, the Receiver must send a positive acknowledge message (ACK) to the Sender if no error is detected. In case of error, the Receiver must send a negative Acknowledge (NACK). In the case of an NACK, the Sender must retransmit the message. This means that the message must stay in a buffer until an ACK message arrives. There are several different ways to implement ARQ.

Data correction is the only option for communication systems where it is not possible to ask for retransmission, like simplex mode communication systems, or where the message latency is a hard constraint, as in Real Time Systems. A system that can correct errors and stay in an error free state is named a *Forward Error Correction* (FEC) system [55]. There are two mechanisms for adding redundancy to communication, block coding and convolutional coding. In block codes, a message composed of k bits is usually encoded by a bijective function in a new message with n bits, where $n > k$. This means that $(n - k)$ redundant bits are added to the message. The code rate of the encoded message is the ratio between the number of information bits and the number of coded bits:

$$(2) \quad R_c = \frac{k}{n}$$

Since additional information is inserted, the communication rate must be increased by a factor of $1/R_c$ to satisfy the transmission rate of the original message.

The use of redundancy incurs in area and power overheads, and reduction in system performance. To satisfy robustness requirements without incurring in wasted resources and/or design time delays, it is necessary to develop methods and tools to evaluate, in an accurate and fast way, the robustness of digital system implementations.

2.2 Networks on Chip

A *network on chip* (NoC), also called *intrachip network* is a communication architecture typically composed by routers that are interconnected to create configurable communication channels. NoCs can support the scalability requirement necessary for the development of complex SoCs [43]. In addition to scalability, a structure based on point-to-point links between routers is well suited for the GALS paradigm, another trend in the development of SoCs. This Section introduces some basic NoC concepts.

2.2.1 Topology

The topology of an intrachip network is the manner how routers are interconnected. A NoC can present a regular or irregular topology. Figure 7 shows some regular NoC topologies: (a) 2D mesh, (b) 2D torus (c) ring, (d) tree.

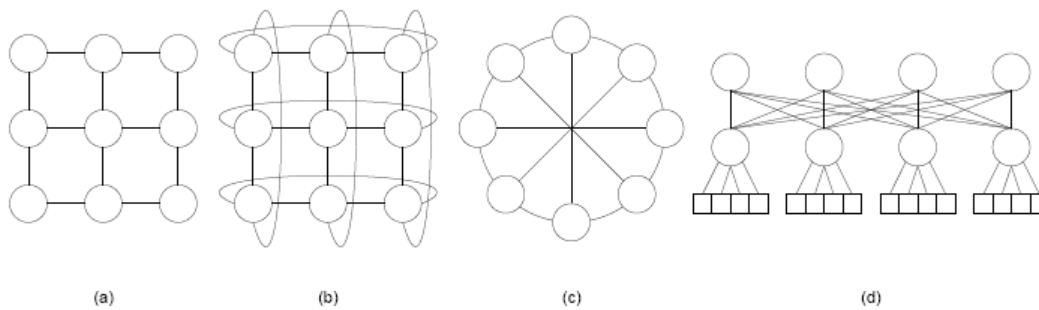


Figure 3 – Examples of NoC topologies: (a) 2D mesh; (b) 2D torus; (c) ring; (d) tree [80].

2.2.2 Routing Algorithm

Routing algorithms are responsible to define the packet path between source and target IP Cores. They can be classified as *source* or *distributed* routing according to where path is defined:

- **Source Routing:** The packet source has enough knowledge about the NoC topology and IP Core localization so that it can define through which routers the packet will pass until it reaches the target IP Core. The NoC path is encoded and inserted into the packet.
- **Distributed Routing:** Each router is responsible to calculate, based on the target address and the current router address, which is the appropriate next router. When the packet reaches the target address, the router must forward the packet to the local IP Core.

Routing algorithms can also be classified according to its capacity to adapt the chosen path in *deterministic* or *adaptive*. Deterministic algorithms compute one, and just one, path between any

source and target. The XY algorithm is one example of deterministic algorithm for 2D mesh NoC topologies. In this routing algorithm, packets proceed first along the X-axis until reaching the target IP column. After this, the packet follows along the Y-axis until it reaches the IP Core target address router. In adaptive routing, a communication between a source IP Core and a target IP Core may follow one of several possible paths. In this case, the current state of the routers are usually taken into account in the routing decision. For example, if in the current router there are two routers options to forward the packet to and one of these is already in use by some another packet, the routing decision may select the free path to be followed.

Adaptive routing algorithms may still be classified as *minimum* or *non-minimum*. In fact, even a deterministic algorithm can be non-minimum, but this usually makes little sense. In a minimum adaptive algorithm, the distance (measured in number of different links traversed) between a source IP Core and target IP Core is always the smallest one. This distance is usually measured in hops, a synonym of NoC links here. In an adaptive algorithm, each router can choose one router among a set of possible next routers. In the case of minimum adaptive algorithms, the set is formed by the neighbors of the current router that are at a distance from the destination router shorter than that of the current router. In non-minimum adaptive routing algorithms, packets may pass by a router more distant from the target IP Core than the current router. This decision is usually taken only when all routers that are part of some minimum path cannot be used.

2.2.3 Switching Mode

A NoC switching mode defines how network resources are allocated to the transmission of a packet. A first classification of switching modes distinguishes circuit-switched transmission from packet-switched transmission [27][31][28]. Circuit switching is always connection-oriented, i.e. a path is allocated before starting data transmission and remains allocated until a special procedure is taken to release the allocated NoC resources. In packet switching, a link between two routers is allocated just during the packet transmission between routers. Packet switching allows better sharing of NoC resources.

Packet-switching communication requires storage buffers, since it is possible that not all the links are available during packet transmission. The allocation of NoC resources for packet switching can be done using usually one of three different switching modes:

1. *Store and forward*: Here, a packet is received and stored completely in the router buffer before its transfer to the next router. This approach necessarily requires storage capaci-

ty for at least the largest possible packet that may traverse the NoC. The fact that the package is stored before starting transmission to another router can also insert relevant latency values in communication.

2. *Virtual cut-through*: Here, a router can send a packet to the next router when the last one gives a guarantee that the package can be fully accepted and stored. This means that the packet transmission can start before the entire packet be received. However, router buffers must again be at least as big as the biggest packet allowed to traverse the NoC.
3. *Wormhole*: In this approach, a packet is always split into small units, called *flits*. A packet transmission does not require storage of a full packet in any router, because flits of a packet may be distributed in buffers of some or even all routers in the path to destination. This reduces latency in transmission because each router can be seen by a flit as a stage of a pipeline. At the same time, this approach decouples the maximum packet size from the storage requirements in the NoC. The main disadvantage of this method is that the NoC congestion is facilitated because at any moment every packet may be occupying several routers. In fact, wormhole implies also a greater care to avoid deadlock and livelock situations. Figure 4 shows an example of possible deadlock created by careless use of the wormhole switching mode. In this example, assume the NoC is a unidirectional ring topology. Assume also that two packets need to be sent in the NoC and both these packets cannot fit in the buffer space available in three consecutive routers. Assume finally that IP Core 0 sent a packet to IP Core 4 through routers 0, 1, 2, 3, 4, while, at the same time IP Core 3 sends a packet to IP Core 1 through routers 3, 4, 5, 0, 1. Assume that the first packet managed to allocate buffers in routers 0, 1 and 2, while the second packet allocated buffers in router 3, 4 and 5. The deadlock is clear because the packet sent by IP Core 0 needs buffers in router 3 and 4, but these are allocated by the packet sent from IP Core 4. At the same time, the packet sent by IP Core 3 allocated buffers in routers 3, 4 and 5 and needs buffers from routers 0 and 1 that are in use by the packet sent by IP Core 0. Deadlock avoidance can be guaranteed by applying specific rules while designing routing algorithms. In the previous example, the token ring algorithm can avoid deadlock.

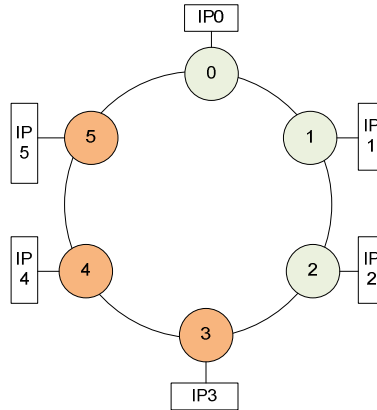


Figure 4 – Example of a deadlock situation in a unidirectional (clockwise) ring topology NoC with five routers and five IP Cores. A packet from IP0 to IP4 managed to allocate buffer space in routers 0, 1 and 2, while at the same time a packet from IP3 to IP1 managed to allocate buffer space in routers 3, 4 and 5. None of the packets can move further, characterizing a deadlock.

2.2.4 Quality of Service

In a NoC, several packets can be transmitted at the same time through data links. However, in some cases it is possible to share the use of some data links among two or more IP Core communications. When this happens, a packet can be blocked by another packet, resulting in decrease in the flit throughput and the increase in packet latency. Consequently, violation of application communication requirements may occur. In the wormhole-switching mode, the blocking of a packet can cause blocking of several other packets inside the network. One way to mitigate this effect is through the use of *virtual channels*. For some applications, like real time applications, communication requirements can be critical and the violation of requirements may lead to system failures. In some NoCs, specific techniques are implemented to guarantee application quality-of-service (QoS) in hardly constrained communications. On the other hand, when a NoC is not capable of providing service guarantees it is said to offer a *best-effort service*. To achieve QoS, there are two basic methods: the use of circuit switching, or the use of virtual/physical channels [22] [50] with associated packet priority levels. In this way, a packet with high priority may preempt another packet with low priority.

2.2.5 NoCs for GALS SoCs

There are several implementations of NoCs to support the design of GALS SoCs [8][13][17][69][70][67]. A coarse classification between the NoCs for GALS systems is possible by separating such NoCs in two classes: GALS NoCs and Asynchronous NoCs.

GALS NoCs are implemented with synchronous routers [41][67]. However, the clock of each router can be different from that of other routers and from the IP Core connected to the router. Asyn-

chronous networks have routers developed as fully asynchronous circuits [68]. To support GALS SoCs, GALS NoCs use asynchronous interfaces [27][66]. Asynchronous interfaces are responsible for the clock synchronization of data arriving from an asynchronous environment or from another clock domain. In a GALS NoC, asynchronous interfaces are required between routers of different clock domains and between router and IP Core if they belong to distinct clock domains. In asynchronous NoCs, the asynchronous interfaces are required just between router and IP Cores if the IP Core is a synchronous module. Since asynchronous interfaces incur in additional latency, asynchronous NoCs are better candidates for high performance GALS SoCs, because they typically drastically reduce the number of asynchronous interfaces crossed by packets [83].

2.3 Asynchronous Circuits

Asynchronous circuits are a class of sequential circuits that do not use clocks to implement sequencing [48][48][49]. The absence of a global or some local clock signals is in fact the only characteristic shared by all asynchronous circuits. This class of circuits is indeed a vast territory containing a large number of design choices. In the past, several different approaches to design asynchronous circuits were developed and some are still being proposed nowadays. According to Sparsø and Furber [85], the multitude of approaches arises from the combination of: (a) different specification formalisms, (b) different assumptions about delay models for gates and wires, and (c) different assumptions about the interaction between circuit and its environment. Other criteria may be employed as well [21]. The consequence is that, even if many efforts exist to define asynchronous design styles formally, there are always difficulties to classify any and every circuit inside some available style. This happens because most asynchronous circuits of medium or high complexity display a mixture of different design styles.

This Chapter presents some criteria to classify asynchronous circuits. It is not the goal here to cover the whole set of available theories about asynchronous circuits. The focus is in presenting some definitions that will help the evaluation of the proposed implementations and of those on related works, all discussed in the next Chapters.

2.3.1 C-elements

Before start the discussions of asynchronous circuits design styles and other related issues, we discuss a fundamental component for asynchronous circuits, the Muller C-element also called simply the C-element. The C-element is one of the most important components for most, if not all asyn-

chronous design styles. It is used in practically all asynchronous design styles as well as in some hardened synchronous designs [29].

The C-element can act as an event synchronizer, producing an event at its output only when input events have all occurred [85]. Table 1 presents the behavior of a C-element with two inputs, where *A* and *B* are the component inputs and *Z* is the component output signal. The C-element copies its inputs to the output when all inputs are either 0 or 1. Otherwise, it keeps its prior output value. It can be implemented in several ways [11][56]. Figure 5 shows the C-element symbol adopted here.

Table 1 - Truth table for a basic 2-input C-element.

A	B	Z_i
0	0	0
0	1	Z _{i-1}
1	0	Z _{i-1}
1	1	1

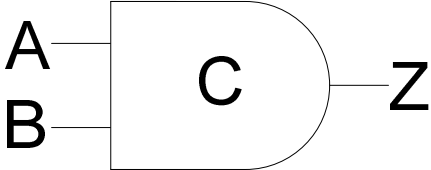


Figure 5 – Symbol of a 2-input C-element.

2.3.2 Handshake Protocols

One important aspect of asynchronous circuits is how they interact with their environment. There is a relation between the assumptions made on the circuit-environment interaction and the timing robustness of the circuit. In this way, when one delegates responsibilities to the environment, the complexity of circuit design reduces. In asynchronous circuits, the handshake protocol is a common way to implement the interaction between a circuit and its environment.

The handshake protocol is implemented using two wires. One of these wires, usually referred as *Request*, is used to initiate the communication. The second wire, usually called *Acknowledge*, informs that the Request signal was received and the required operation was, or will be, performed.

The data signals, when present, are enclosed by the control signals so they are free to switch before the control signals of the protocol are put to an active state. To guarantee correct operation, the events in the control signals must be signaled just when the circuit data interface is in a stable state.

Handshake is the widest adopted protocol in asynchronous circuits. In fact, this protocol is not

exclusive for asynchronous circuits. Synchronous and Globally Asynchronous and Locally Synchronous circuits employ this communication protocol [66].

The handshake protocol can be implemented in several ways. The protocol may be used to transfer data or just to synchronize two circuits. When transferring data through a channel, the handshake protocols can be classified as *push channels* or *pull channels*. In a push channel, the circuit that initiates the handshake is responsible to send the data. In the pull channel, data is transferred by the circuit that receives the Request signal. Figure 6 shows an example of a push channel while Figure 7 depicts an example pull channel.

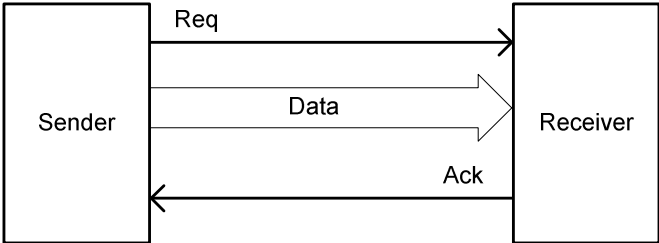


Figure 6 - Push channel-based communication using a handshake protocol.

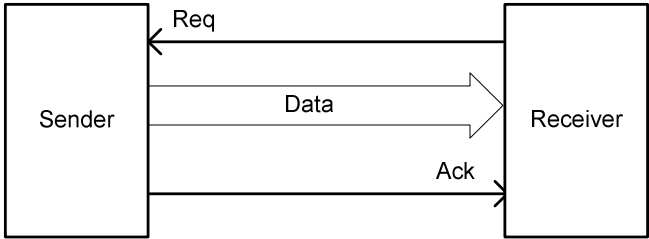


Figure 7 – Pull channel-based communication using a handshake protocol.

Handshake protocols can also be classified depending on the number of transitions in the protocol. The *two-phase* handshake protocol is started with a transition in the request and finalized with one transition in the acknowledge signal. This protocol is also known as signal-based protocol or Non Return to Zero (NRZ) protocol. Figure 8 shows one example of a two-phase protocol used for data transmission.

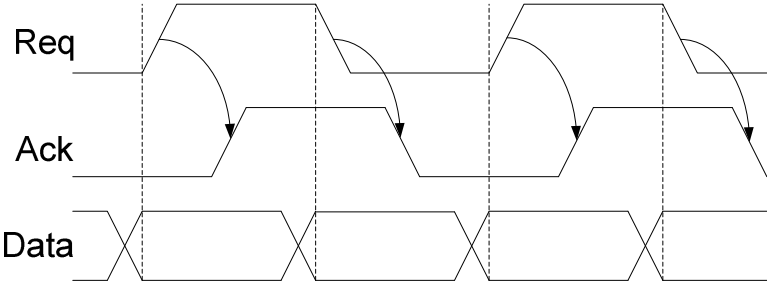


Figure 8 – Two-phase handshake protocol.

The *four-phase* protocol comprises two extra events. This protocol is known as level encoding

protocol or Return to Zero (RTZ) protocol. Based on the implementation, the data transfer inside the protocol can be classified as *broad*, *early* or *late*. Early and broad protocols can be implemented in the push channel. Late protocols are more common in pull channels, where the data transfer can start just after the Sender receives the request. Figure 9 shows some examples of four-phase protocol.

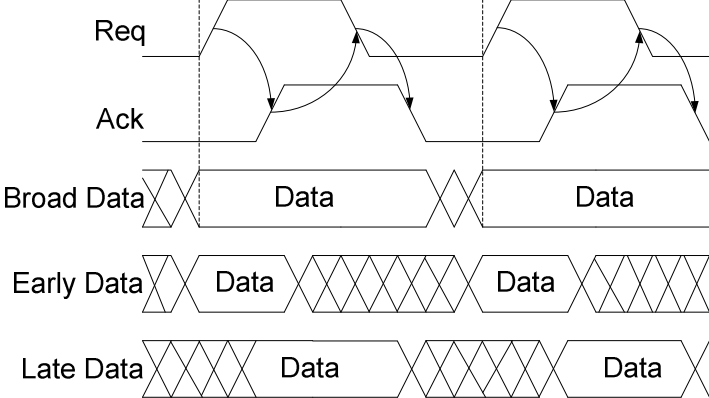


Figure 9 – Four-phase handshake protocol examples.

2.3.3 Asynchronous Design Styles

The most useful classification for asynchronous circuits regards the timing assumption criteria of each circuit class. We follow here the definition of Sparsø & Furber [85]. This classification divides asynchronous circuits in the following groups: Delay Insensitive (DI), Quasi Delay Insensitive (QDI), Speed Independent and Self Timed (ST) circuits.

A digital circuit is DI when its correct operation is independent of the delays in operators and in wires connecting the operators, except that delays are finite and positive [48]. Delay insensitivity is achieved when the circuit respects the non-interference, stability and acknowledging properties [48].

The non-interference and stability are essential to guarantee the hazard free gate operation whereas the acknowledge property is a special property of DI circuits. The acknowledge property states that any event at the input must be acknowledged. An AND gate for example is not part of the delay insensitive class since when one input is with ‘0’ and the other input change from ‘0’ to ‘1’ the output of the AND gate does not acknowledge this input transition.

In practice, just a small class of circuits exists inside the DI class. The weakest timing constraint possible for a circuit is the *isochronic fork* constraint. An isochronic fork is a wire fork where if a transition is detected in one branch of the fork, this implies that the transition arrived also at the other branch of the fork [91]. If the isochronic fork is the only assumption made in a circuit, the circuit is classified as a quasi-delay insensitive (QDI) [48].

The *Speed Independent* circuits in this class must operate correctly with any positive bounded gate delay. However, the wire delay must be guaranteed to be smaller than the gate delay. This is not always true for advanced technology nodes. Another issue is that this constraint is not easily guaranteed for long wires, in data communication for example.

Speed independence and delay insensitivity are formally defined concepts. Any circuit where correct operation relies on any additional constraints, following the definition in reference [85], is called *self timed*. This includes the circuits that employ delay elements to match the propagation delay of paths. A sub-class of self-timed circuits that is commonly used are the bundled data circuits.

Bundled data circuits have the general structure illustrated in Figure 10. The circuits in this class work inside a delay window, just like synchronous circuits. The difference is that instead of using a global clock signal, bundled data circuits employ local delay elements (the Match Delay element in Figure 10). Any computation must have delay propagation smaller than the Match Delay element.

Providing guarantees that the delay considers all possible path delays is a complex task. This must include any global and local process variations. For advanced node SoCs, this requires statistical simulation of the whole circuit, which is prohibitive for complex SoCs. Additionally, their design process must take into account temperature, voltage and problems related to signal integrity as well.

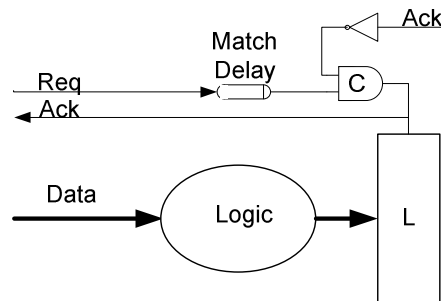


Figure 10 – Bundle data generic circuit structure, displaying the local handshake circuit, the combinational part (Logic) and the storage element (L).

2.3.4 DI Data Encoding

The most common way to encode data in digital domains is to use the regular binary encoding where each logical bit value is associated to a voltage level range. However, data encoding can be manipulated in order meet data transmission constraints. Transmission may require increasing the number of bits (e.g. 8b/10b code to permit: the clock recovering [94]), or in some other cases reduce this number to reduce the energy consumption [65]. Other goals are increasing data throughput (e.g. differential data transmission) enabling data correction (e.g. TRDIC [72]). In asynchronous circuits,

Table 2 – Dual rail code with L=2. The data spacer here is represented by the value “00”.

Logic Value	At	Af
Spacer	0	0
'0'	0	1
'1'	1	0

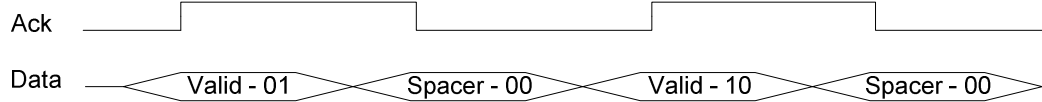


Figure 12 - Data transmission using dual rail encoding and a four-phase protocol.

Circuits using DI codes must be capable to detect validity or absence of data (through Spacers). To do this a circuit, usually called *completion detector*, is necessary to implement the control part of the handshake. One possible implementation of a completion detector for dual rail code is composed of OR gates, which determine the validity of each individual dual rail encoding and a tree of C-elements, to synchronize the individual bit detections and signal valid codeword detection. An example of a completion detector circuit for four bits encoded using dual rail appears in Figure 13.

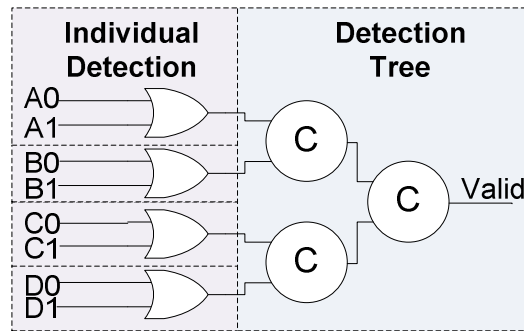


Figure 13 – A Dual rail completion detector for a code that can represent values of a vector of two bits.

The dual-rail codes are special cases of m-of-n codes. Another common m-of-n code used in QDI circuits is the 1-of-4 [13][89]. The completion detector for the 1-of-4 encoding has a close implementation with the dual rail. The main difference is in the NOR gates. In the 1-of-4 encoding, 4-inputs OR gates are used. In fact, the Completion Detector of all 1-of-n encodings consists of individual encoding detection, that is done through a n-input OR gate, that can be followed by a tree of C-elements. One way to evaluate codes is to calculate their code density. *Code density* is the relation between the number of available binary words for a given code length and the number of codewords actually existing in some DI code, obviously without counting the spacer. The dual rail and the 1-of-4 have the same code density. The main advantage of the 1-of-4 is that it requires half of the switching in a data communication.

Table 3 - 1-of-4 encoding used to represent two logical bits.

Value	A3	A2	A1	A0
Spacer	0	0	0	0
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	1	0	0	0

Weak and Strong Indication

A DI circuit can be classified according to the way it reacts to the transitions of input signals as *strongly indicating* or *weakly indicating*. A *strongly indicating* circuit waits all that inputs are valid to start the computation over the data and the output generation. Likewise, a spacer in a strongly indicating block is generated just when all inputs are spacers. *Weakly Indication* circuits react to changes in the input signals before all reach the valid or the spacer. However, in order to keep the timing properties, circuits must generate a valid or spacer output just after all inputs are with a valid or spacer codeword. If this property is not always true, the circuit is said to be *early output* circuits [19]. This kind of circuit implementations generates transitions that are called orphans. *Orphan transitions* are transitions that reach a module after the module generates the output event related to this transition [19].

2.3.5 Asynchronous Circuits Implementation

In this Section, some circuits that are used in typical asynchronous NoC implementations are described and analyzed.

Asynchronous Buffers

Asynchronous buffers serve for storing temporary values and support the control tokens to communicate with neighboring registers through handshake protocols. A register is classified according to, among other criteria, the amount of tokens that it can keep [96]. A token can be a data or a spacer. A buffer that is able to store only one token is called *half buffer*. A buffer capable of storing valid data *and* a spacer is called a *full buffer* [96].

There are different buffer implementations for asynchronous circuits. WCHB buffers may be used with all DI Codes and is employed in all circuit implementations in this work. Figure 14 shows a two input dual rail register implementation with WCHB.

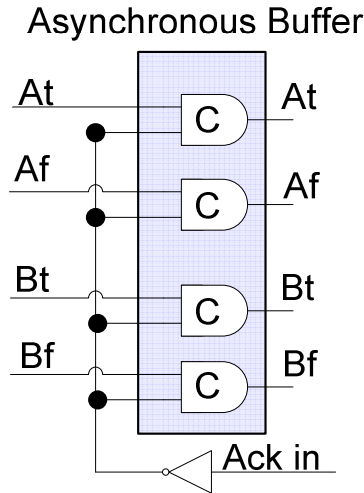


Figure 14 - Two-bit weak conditioned half buffer (WCHB).

Asynchronous Pipelines and Rings

Asynchronous circuit implementation can be performed using either single track code, like used in synchronous logic, or using DI codes.

Asynchronous circuits that perform computations over data vectors, such as arithmetic circuits, are implemented in asynchronous logic as pipelines and/or rings. In this type of circuit, the handshake protocol is responsible for the control flow of data within the pipeline or ring.

Logic blocks are inserted among storage (or buffer) stages. These blocks are equivalent to the synchronous combinational logic circuits. Logic blocks are transparent to the handshake signals, which means that an observed sequence of data in the input of a logic has a correspondent output that is also observed at the output with a delay [85]. This delay is due to the propagation time of the function block. Figure 15 shows an asynchronous pipeline stage, comprising the series connection of asynchronous registers with logic between buffers.

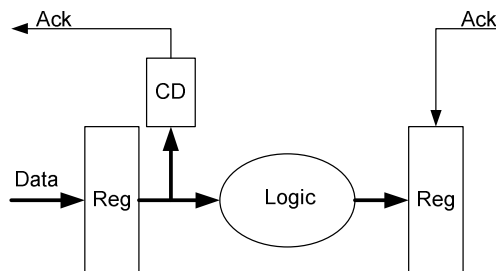


Figure 15 – Typical asynchronous pipeline stage example.

Delay Insensitive Minterm Synthesis (DIMS)

There are various implementation techniques of functional blocks for dual rail codes [85]. An implementation example of logic block is the use of the Delay Insensitive Minterm Synthesis (DIMS) technique [85]. In this technique, all possible minterms are generated using C-elements. An OR gate is used to combine the minterms that lead to the state of the output to logic '1' (the *set* state) and another for adding the output minterms that lead to the *reset* state (the one that makes the output go to logic '0'). Figure 16 shows an example of XOR using the DIMS technique with one logic inputs. The DIMS technique can be easily employed with different DI codes.

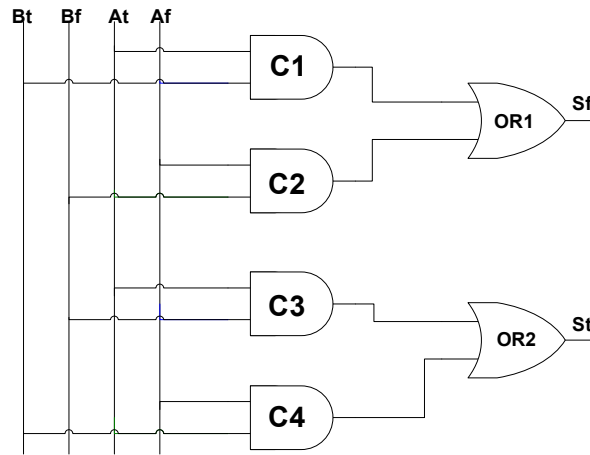


Figure 16 –A XOR DIMS gate for a one-bit dual rail code.

Control Flow Circuits

Apart from buffers and logic blocks, asynchronous circuits also comprise control circuits responsible for command the flow of tokens in the pipeline or ring. These circuits can be unconditional flow control circuits like join, fork and merge, or conditional flow control circuits, like multiplexers (muxes) and demultiplexers (demuxes). Besides the above-mentioned components, arbiters and mutual exclusion elements can also be considered as flow control circuits. The next sub-Sections cover the components used in the implementation of the NoCs proposed in this work.

Fork

The fork component can split a wire in two or more wires. It can be viewed as a gate with a single input and multiple outputs. In a QDI system, forks must obey the isochronic fork rule. Their *Ack* signals must be synchronized, or if any of the branches do not generate such a signal, it is necessary to ensure that when the *Ack* is generated by some branch the other branch(es) must be guaranteed to finish by generating acknowledgements, too.

Figure 17 shows the fork data flow symbol while Figure 18 shows the circuit implementation of a two-output fork.

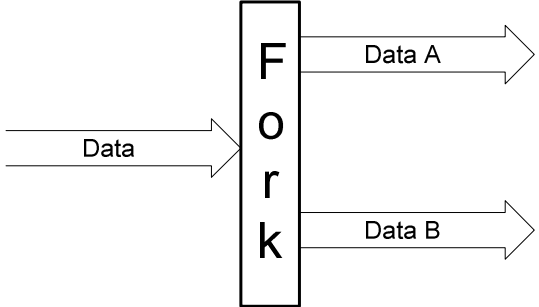


Figure 17 – The fork component data flow representation.

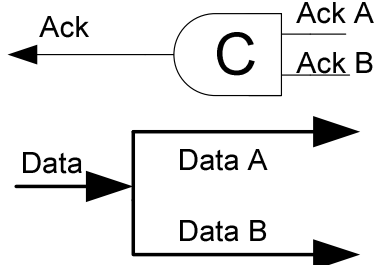


Figure 18 - Example of a possible fork circuit implementation.

Join

The Join component is responsible for the convergence of two data streams into one. The implementation used for this circuit in Figure 19 is weakly indicated. Figure 19 shows the symbol and Figure 20 shows an implementation of the Join adopted in the routers proposed here.

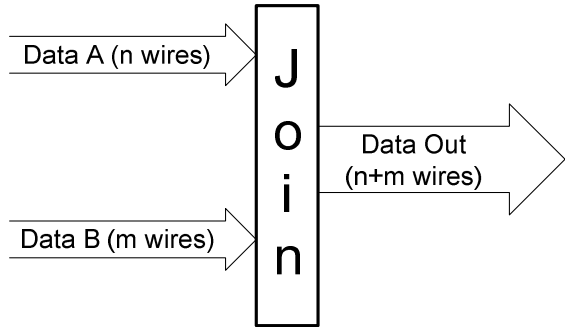


Figure 19– Data flow representation of the Join component.

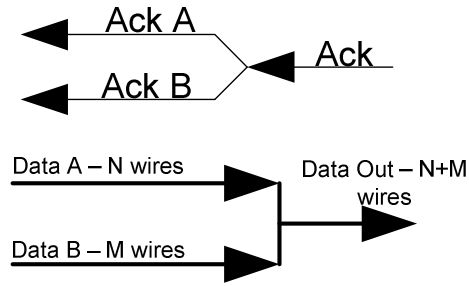


Figure 20 - Example of a possible Join component implementation.

Merge

This component combines two mutually exclusive data inputs in one output. The use of completion detectors is required to propagate the Ack signal only to the active input branch. Figure 21 shows the symbol and Figure 22 displays an example circuit implementation of the merge component. The implementation is weakly indicating, since the input Oring will generate the data output after any input switching. Since the OR between the inputs assumes that just one input is active at each time, the merge OR gates are DI. This means that any input switching has one corresponding output switching, following the indication principle.

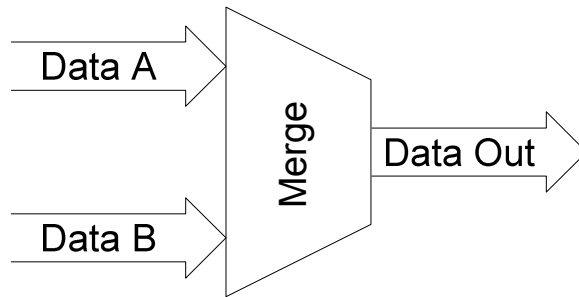


Figure 21 – Merge component data flow representation.

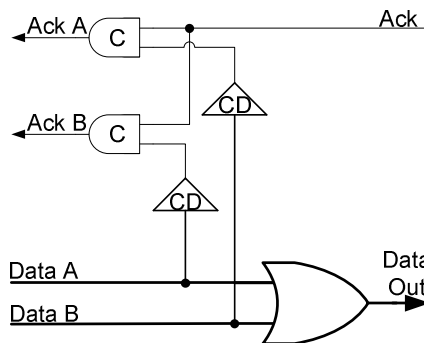


Figure 22 - Example of a Merge implementation.

Multiplexers and Demultiplexers

Multiplexers and demultiplexers are well-known digital components. M-of-n demultiplexers and multiplexers show the same behavior as the equivalent single track implementation. Figure 23 and Figure 24 show the symbol and structure of multiplexers, while Figure 25 and Figure 26 respectively show symbol and structure for the demultiplexer.

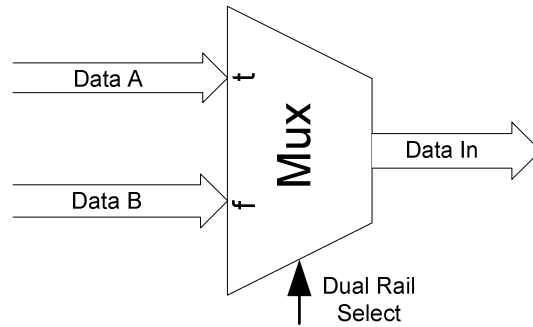


Figure 23 – Dual rail multiplexer symbol.

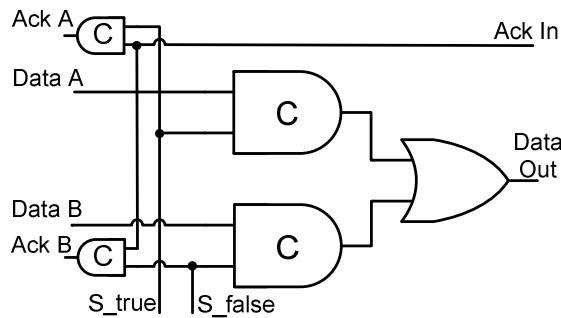


Figure 24 – Example of a dual rail multiplexer implementation

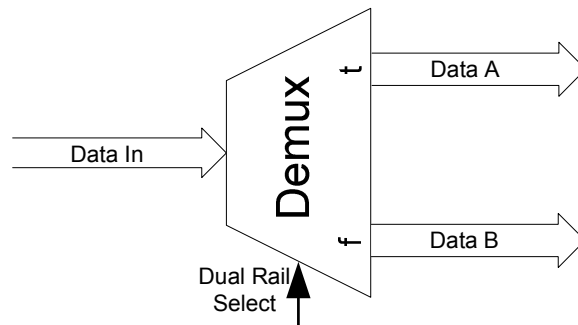


Figure 25 – Demultiplexer symbol and implementation in Dual Rail.

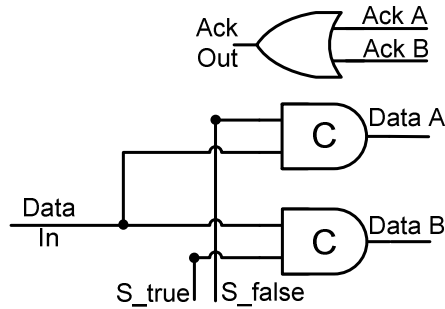


Figure 26 – Example of a dual rail demultiplexer implementation.

IF/ELSE

The Demux and the Merge components permit the implementation of more complex control circuits. Figure 27 shows the implementation of a block IF/ELSE. In this implementation, the Select signal is the control of the IF statement. When the Select is true Logic A is executed, when the Select is false, the Logic B is executed. This circuit arrangement is widely employed in the asynchronous implementation of the NoC.

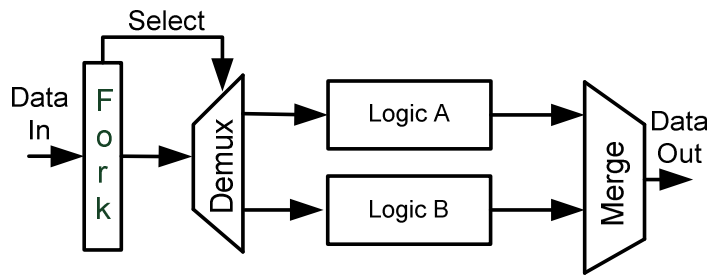


Figure 27 – The block diagram of an if/else module.

S-Element

The Element S-element is a control circuit designed to sequence two or more actions from an event in its input. Figure 28 shows the general structure of this component. Note that this component has an input handshake channel and two output channels. When a request arrives at the input of the S-Element, it executes a four-phase protocol with channel A and the output followed by a four-phase protocol with channel B. Only after completing the protocol in the two outputs, the S-element ends the handshake protocol with the input port.



Figure 28 – S-element symbol.

The behavior of an S-Element is described by the state transition graph (STG) showed in Figure 29. The STG was used as input to the public domain tool for synthesis of speed independent circuits Petrify [24]. The implementation of the circuit was made using complex gates. The circuit was verified at layout level to ensure that the constraints related to the feedback wires are obeyed.

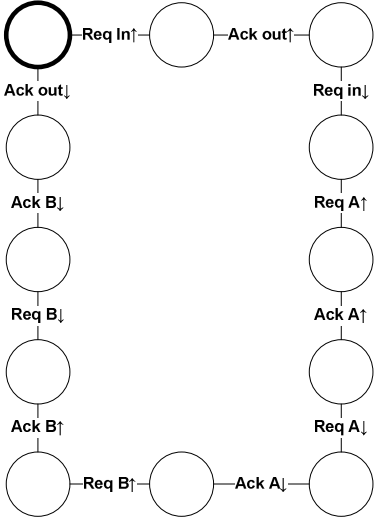


Figure 29 – State transition graph representation for the behavior of the S-element component.

Arbiters

The use of mutual exclusion elements is another way to implement sequencing of events. In this case, however, an arbiter component is responsible for coordinating access to a resource shared by two concurrent requests. Figure 30 shows the symbol of a mutual exclusion element. The circuit has two request inputs and two acknowledge outputs. When a requisition is signaled, the circuit responds with the related acknowledge. If the other request is signaled at the same time, it is blocked until the first request is released.

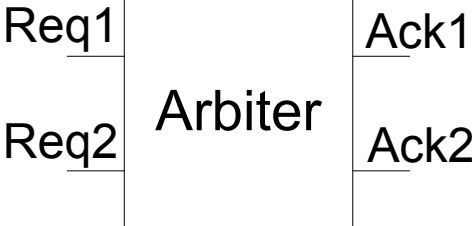


Figure 30 – Symbol and interface of an arbiter.

Figure 31 shows the implementation of a mutual exclusion element. The implementation includes a filter connected to the NAND outputs, to solve the metastability problem. This element is required because in cases where the two requests arrive simultaneously, the output of NAND gates can become metastable. In this case, the filter removes both Acknowledge signals until the metastability solved.

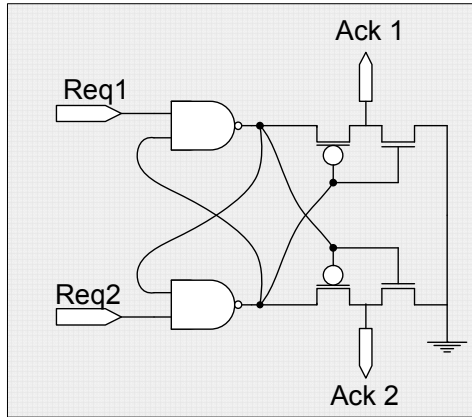


Figure 31 – Arbiter implementation.

In order to control the access to a resource that is shared by more than two requesters, a network of arbiters can be used. Figure 32 shows a four requester arbiter built as arrangement network of two input arbiters. The fairness of this arbiter is proved on [8].

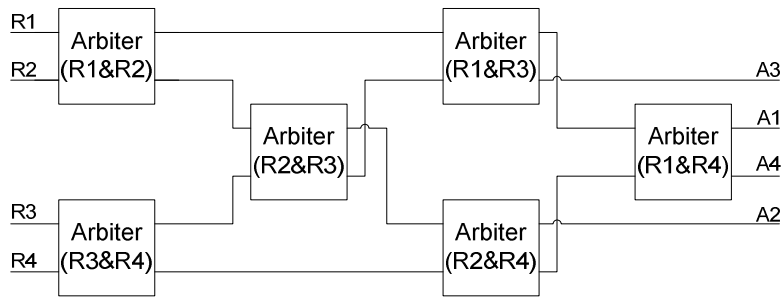


Figure 32 – Four input arbiter built as an arrangement of six two input arbiters.

In NoC implementations, arbiters are used to control the access to shared links. This is done by a multiplexer that is controlled by the arbiter decision. Figure 33 shows the implementation of a multiplexer controlled by an arbiter.

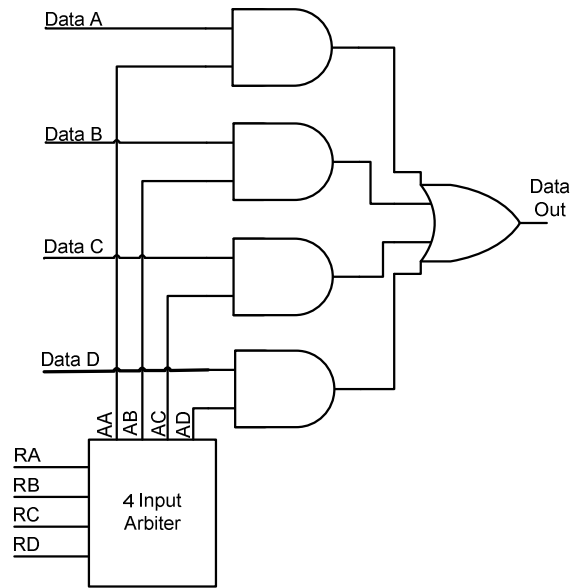


Figure 33- A 4-input multiplexer controlled by an arbiter.

3. RELATED WORKS

This Chapter brings the related works associated to the asynchronous NoCs, soft error hardening techniques for asynchronous circuits, hardened NoCs and about the NoC synthesis tools.

3.1 Asynchronous NoCs

Due to the migration from fully synchronous to GALS circuits, the asynchronous NoCs are becoming a common communication architecture choice. The asynchronous implementation is able to deliver high throughput with less dynamic power consumption. Moreover, the immunity to delay variations presented in the QDI NoCs made then a good option for advanced nodes technologies.

The first fully asynchronous NoC was implemented by Bainbridge and Furber [8]. The NoC employs the 4-phase handshake protocol and the 1-of-4 encoding. A special signal is used to delimit the end of packet. In this way, packets with arbitrary size can be transmitted through the NoC. The NoC uses the source routing scheme. The NoC is composed by several building blocks in such a way that different topologies can be generated. Since the source routing scheme is employed, the IP Cores must know the NoC topology in order to generate the path definition.

The works of Beigné [13] presents the ANOC. The NoC has a mesh topology, wormhole packet switching and source routing. The ANOC is composed by routers with two virtual channels in order to provide QoS. The four-phase 1-of-4 data encoding is used in the NoC. The ANOC was adapted in [89] to support dynamic power controls by applying activity monitoring to control DVS and Power Shut-Off (PSO).

Bjerregaard [17] and Sparsø present the MANGO (Message-passing Asynchronous Network-on-chip Providing Guaranteed services through OCP interfaces). The MANGO NoC has virtual channels that provide Guaranteed Services, connection-oriented and connectionless routing algorithm with best-effort. The MANGO NoC router is implemented through bundled data four-phase asynchronous circuits.

Quartana et al. [74] present an asynchronous NoC implementation target to FPGAs. The development of asynchronous modules that compose the NoC was done using the TAST tool [30], which generates QDI circuits based on a CHP description of the circuit behavior. The synchronous-

asynchronous interface was developed using two flip-flops in series. This arises as the main disadvantage of this approach, since the synchronization through flip-flops in series increases the latency significantly.

In the proposed Rostislav et al. [79] is shown the structure of a fully asynchronous QNoC. The authors claim they can guarantee QoS. The network is based on QNoC a router architecture that supports multiple service levels. It is also proposed the use of preemptive according to priority of packets. The packets are partitioned into flits, which are sent over the network using wormhole switching mode. The source routing is employed in the QNoC.

The work of Sheibanyrad [83] proposes a fully asynchronous network called ASPIN. The ASPIN NoC has a 2D mesh topology with 32-bit flits. The routing algorithm is the XY algorithm. The encoding used is dual rail associated with the four-phase handshake protocol. Dual Rail and one-of-four encode employ the same number of wires. However, the main disadvantage of the dual rail is that it requires two times more switching than the one-of-four.

The work of Lines [45] presents a 36-bit asynchronous crossbar with 16 ports. The proposed structure falls into the category QDI circuit and uses 1-of-4 data encoding. The results at 130 nm technology show a 780Gbits data flow when all the 16 ports of the crossbar are transmitting.

This work proposed and implemented a set of fully asynchronous NoCs. The Hermes-A [69] was the first fully asynchronous NoC implementation generated for this Thesis. The Hermes-A was initially designed with the dual rail encoding and four phase handshake protocol. It is a mesh based and the main contribution was the distributed routing algorithm. The Hermes-AA [70] took the full advantage of the distributed routing by implementing adaptive routing algorithms. The Hermes-A was also adapted to permits the use of source routing algorithm and one-of-four encoding. The main contribution related to asynchronous NoCs however was the proposition of the Hardened Hermes-A (H₂A) NoC. The H₂A NoC, differently of the NoCs presented in this Chapter, is designed for Soft Error mitigation. Just some few works, which are presented later in this Chapter, perform some analysis and implement some hardening technique for asynchronous NoCs. In general, there are just few proposition and applications of hardened asynchronous circuits. Some of these techniques are presented bellow.

3.2 SEE Hardening Techniques for Asynchronous Circuits

Bastos et al. [10] evaluate the implementation of asynchronous and synchronous versions of a DES crypto-processor. The results show that the asynchronous version has a better response when

increasing SEE pulse widths. Rahbaran and Steininger [75] compare the synchronous and asynchronous versions of a 16-bit processor using an FPGA as a target for SEE emulation. The asynchronous implementation shows the best results under several different SEE scenarios. However, none of the two works proposes hardening techniques for asynchronous implementations. Even if they display better robustness, QDI circuits are not free from soft errors, and in order to enable their use in hostile environments or even for new technologies, it is necessary to implement additional techniques to ensure low Single Effect Rates (SERs).

Agyekum and Nowick [2] propose an unordered DI code that enables two bit error detection and one bit error correction capabilities. However, this code is difficult to implement in a fully QDI way and becomes complex for several parallel bits due to the completion detection complexity. Bainbridge and Salisbury [9] present a set of techniques to apply to QDI Networks on Chip links, particularly for links based on m-of-n encoding, to reduce glitch sensitivity due to crosstalk. The presented techniques are mostly sampling filtering techniques to reduce glitches in data signals and in some of them in acknowledge signals. However, none of these is adequate for SEE and none offers data error correction. Note also that data error correction and/or detection is required since the techniques are not able to eliminate the consequences of glitches or SEEs in QDI links.

The work of Bastos et al. [11] shows a comparison between different implementations of C-elements operating under SEEs. They also present a hardening technique based on electrical filtering by resizing these components. Resizing consists in increasing the capacitance of the nodes to filter small size pulses. Since C-elements are the most important building blocks of QDI circuits and the main component of memory elements in these circuits, the increase of their robustness may bring enhanced SEE tolerance to the whole circuit. However, increasing the size of transistors expands the die area susceptible to particle strikes and the area of the junctions, responsible for charge collection.

Monnet et al. [52] show three different techniques to mitigate SEEs in asynchronous logic. The first consists in the duplication of the asynchronous logic and a check at memory elements. This technique is able to remove errors in case of single events. It clearly involves a big area overhead, since the whole circuit is duplicated. The second technique consists in synchronizing different DI bits of the computation part. For example, a less significant bit of an AND applied bit per bit to a 16-bit word will be synchronized with the second less significant bit. This is done successively to the other bits. The synchronization is performed by a four input C-element in such a way that the output of the cell is set just when the AND computation in both bits has finished. This technique does not ensure the elimination of errors but increases the logic filtering property of the cell. The last technique adds a single rail signal for data validity in the forward direction between memory blocks,

which is used to validate the data at the input of each memory block. This technique, as the previous one, only increases the logic filtering property.

Jang and Martin [39] propose an SEE hardening technique using spatial redundancy for asynchronous circuits called double check. The technique consists in duplicating the logic and adding a verification point after the computation, to ensure that the results are equivalent in both branches. Double check is capable of providing multi-bit error correction, and is accomplishable with simple, C-element-based circuits.

3.3 Hardened NoCs

The most part of the error control in NoCs are limited to the application of error correction or detection schemes in the data links and/or packets. Some of these present retransmission schemes that can be applied between the routers (switch-to-switch), at the Sender/Receiver (end-to-end) [59]. Another way to prevent data errors in NoC links is using *Forward Error Correction* (FEC) as proposed in . The work of Rossi [78] et al. presents different levels of error protection, enabling correction and or detection. Depending on the requirements, the errors can be detected, corrected by Single Error Correction (SEC), Single Error Correction Double-Error Detection (SEDED) or symbol-error correction can be applied. The work of [69] presents data correction at switch-to-switch and at end-to-end levels. None of these implementations proposes error control for the control circuits in the routers.

The work of Frantz et al. [33] presents the NoC evaluation under the presence of faults generated by SEU and/or crosstalk. Hardware and software mitigation techniques are evaluated for a NoC with 8 bit data flits. The fault injection method inserts faults generated by crosstalk and SEU. The response of the NoC to SET is not analyzed. The evaluated NoC has synchronous implementation. The hardening techniques were applied at data and control logic. In the data level, Hamming Codes or Cyclic Redundancy Check is applied. In the control level, the TMR is used combined with a triple sampling scheme. The TMR utilizes a combinational voter inside the circuit. Since the SET was not evaluated, the voter is considered fault free. This leads to optimistic soft error rates.

The works above mentioned are synchronous NoC implementations. Just a few works propose hardening techniques for asynchronous NoCs implementations.

The work of [97] presents some hardening techniques to the inter-chip communication using the SpiNNaker NoC. The results show the reduction in the NoC deadlock due to glitches in the inter-chip link. However, none technique is proposed in this work to problems likes SEE and glitches ar-

riving inside the chip.

Ogg et al. [60] proposes a resilient link based on a dual-rail transmitted with a reference code. The difference in the phase between the reference code and the data code determines the transmitted data. When any transition that violates the phase permitted phase transitions occur, the receiver do not consider as valid transitions. Again, the proposed solution applies only for data link protection. No hardening technique is proposed to mitigate the SEE in the control logic.

The work of Bainbridge and Salisbury [9], already mentioned in Section 3.2 about the asynchronous hardening techniques, also proposed several techniques to increase the reliability in asynchronous NoC links. The techniques are intent for glitch filtering generated by crosstalk in the NoC data links. However, some of the techniques can be used as well to filter the glitch generated by Single Event Effects.

The work of [95] presents the investigation of transient fault effects in an asynchronous router. This work presents just the evaluation of an asynchronous router under different fault scenarios using fault injection. No hardening technique is proposed.

3.4 NoC Synthesis

This work proposes a set of different NoC implementations. This means that different NoC components – Input/output Ports – are implemented. The components can have different flit size, routing algorithm, hardening techniques and configurations. In addition, in order to remove the complexity a library of components is proposed in Chapter 6. In order to offer support to the synthesis of hardening NoCs, a synthesis method is proposed and implemented as shown in the Chapter 7. The rest of this Section presents some related works about NoC synthesis that were used as base to the synthesis method proposed in this work.

Atienza [6] uses the Xpipes NoC [15] library as the base for the NoC synthesis tool. The Xpipes library provides features like: handshake or credit based control flow, different clocks (but synchronized) between the NoC and IP Core, network interface (NI) which has LUTs containing the paths between the source and various destinations (source routing), and Open Core Protocol 2.0 (OCP) interface between NoC IP Core.

The main purpose of the tool is the generation of a NoC (SystemC and RTL code) with a topology suited for a specific application. The communication between IP Cores is defined by communication graphs. Moreover, the tool generates the routing tables and the NoC floorplan. The tool can generate routers with different number of port. According to the author, the tool includes the follow-

ing steps: the generation of the parameters of the NoC, the RTL code generation, NoC synthesis and simulation. The last step is performed using commercial simulation and synthesis tools. However, the tool supports the NoC floorplan generation to be used in place and route steps.

Results show that the custom topologies presents better timing, power results while the area are not significantly different when compared to a NoC with mesh topology. The results also show that the choice of standard cell library (LVT, SVT, HVT) have a great impact on the performance of the communication structure, but the tool does not support this choice which is left to the developer.

Seiculescu [81] proposes a method for generating NoCs with multiple voltage domains with the ability to apply power shut-off. The tool takes a communication graph and a partitioning of the voltage and frequency domains for the IP Cores as its starting point. All the IP Cores from the same domain operate at the same voltage but not necessarily at the same frequency. The first step in the synthesis is a router insertion for each IP Core. The router will always belong to the same IP Core island. In the next steps, the tool can also generate new voltage domains when it is advantageous in terms of energy and performance. The generated voltage domains are composed only by intermediate routers and are never switched off. The purpose of these islands is to serve as alternate route for communication between islands without connecting islands that are off. All routers on the same voltage domains operate on the same frequency, which is determined by the highest request throughput. Asynchronous interfaces are employed for clock synchronization and voltage conversion (using level shifters). The power shut off is enabled when all elements of the same island become idle. The proposed tool is used to implement a SoC with 26 modules using the Xpipes library in 65nm technology.

Bertozzi [16] shows a flow of synthesis known as NetChip. This flow is also based on Xpipes NoC library. The tool uses a specification of the communication constraints between the IP Cores. The synthesis of NoC is divided into three steps: IP Cores mapping in different NoC topologies, topology selection and NoC generation. In the first step, the application is mapped to different topologies in a defined topology library. For each mapping, an evaluation of performance, area and power consumption is performed. Based on these evaluations and on the optimization target passed as input to the tool (area, performance or power) the topology selection is performed and the routing tables are generated. The routing tables are generated from the evaluation of different routing. The third and final step uses the description generated by the tool to feed the XpipeCompiler and generate the SystemC and the RTL code of the selected NoC.

The work of Leung [44] presents a tool for partitioning intrachip networks in voltage domains.

The tool synthesizes energy efficient NoCs starting from the communication dependency and computing graph (CDCG) and the deadline of the tasks. The tool generates the IP Core mapping, the routing definition, the power domains and the voltage definition for each power domain. The ultimate goal is to reduce energy consumption while maintaining the satisfaction of hard deadlines for the tasks. This is done by a framework based on genetic algorithms. The adopted NoC uses worm-hole switching mode, source routing and mesh topology. To determine the operating voltage of the connecting links between the IPs, the author follows the following rule: the links between IPs follow the slightest tension between Transmitter and Receiver.

The work of Ogras [61], also presents a method to generate voltage domains for the NoC based on traffic constraints. The method starts in a configuration where each router has an individual voltage domain. At each iteration, the tool selects the voltage domains and attempts to combine them with the neighbor voltage domains. The pair of neighbors that produces the greater energy reduction is selected to be merged. The results obtained by simulation show a reduction of up to 40% in power consumption when compared to a system not partitioned into voltage domains.

Jang's work [40] is an extension of work previously presented and developed by Ogras. In the Jang's approach, methods for mapping and routing are proposed in addition to the voltage partitioning. According to Jang, the IP Core mapping allows better use of the multiple voltage technique. The routing definition has as target the reduction in the number of the points to cross the voltage domain and therefore the number of voltage converters and frequency required. The results suggest an 82% reduction in the number of voltage converters and 9% in power consumption when compared with the approach of Ogras.

Srinivasan's work [86] employs an optimization method known as linear programming to synthesize intrachip networks that offer reduced power and achieve the performance constraints. The proposed tool receives the communication constraints in the form of a graph and the dimensions of the IP core. The tool uses a library of routers characterized in terms of performance and power. The output of the proposed tool is the description of the NoC topology, number of routers and interconnections between them, the floorplan of the system and the definition of routing. The results presented are related to the implementation of a NoC with 32 bits flit, two virtual channels and eight flit packets. The author compares the results to a NoC with regular mesh topology. The results show a reduction of 3.5 times the average number of routers and 2.3 times in the energy consumption.

The NoC synthesis methods and tools presented in the literature are driven by performance and/or low power constraints. Just a few of them are suitable for GALS systems and none uses asyn-

chronous NoCs as building blocks. This work is distinguished from the propositions presented in this Chapter because it proposes a NoC Synthesis Environment driven by robustness and performance constraints using Asynchronous NoCs to support the design of Hardened GALs systems.

4. THE HERMES-A NoC

This Section presents the Hermes-A NoC, a fully asynchronous communication architecture designed to support GALS SoCs. Hermes-A comprises two distinct DI Codes-based implementations: dual-rail and one-of-four. Both implementations employ four-phase handshake protocols. The NoC has a mesh topology and uses wormhole switching. Figure 34 shows a 3x3 Hermes-A NoC example where each router has a XY-coordinate address associated to it.

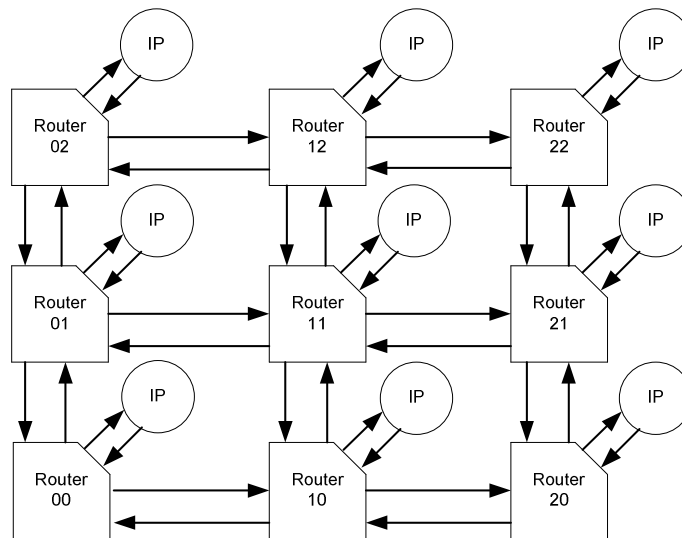


Figure 34 – A 3x3 NoC mesh topology. The numbers in the router denote the X and Y address of the router.

Hermes-A routers comprise up to five Input and Output ports. Ports are named North, South, East, West and Local, as Figure 35 shows. The Local port is used to perform communication between the router and its IP Core. The remaining ports are used for communication between neighbor routers.

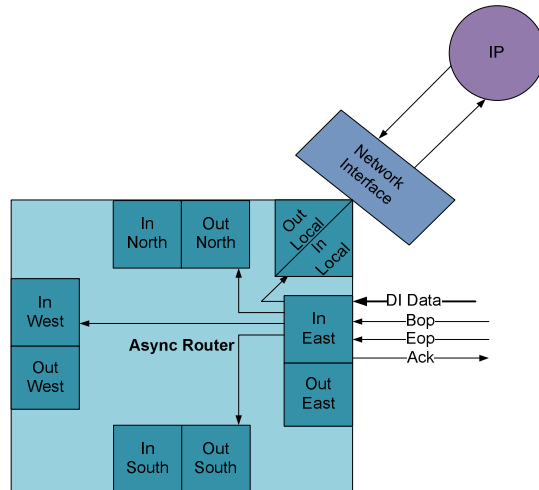


Figure 35 – Router architecture adopted in this work and the IP Core connection through the Network Interface.

Packets in Hermes-A are delimited by two special side band signals, the *Begin of Packet* signal (BOP) and the *End of Packet* signal (EOP) as Figure 35 details. In the one-of-Four implementation, these two signals are encoded in a single one-of-four encoding called *Flit Type*. These signals are responsible for indicating the first and last flits of a packet. The first flit, containing routing information, carries with it a Begin of Packet (BOP) signal. The last flit is delivered together with the End of Packet (EOP) signal and is important to release the allocated resources in the NoC. Figure 36 shows an example of a four-flit packet with the BOP and EOP signaling. Using these side band signals, it is possible to have packet sizes with any positive size (>1). It is not possible to transmit a packet with size equal to 1 since the BOP and EOP cannot be both equal to 1 in the same flit.

First Flit - Routing Information	Flit Type = "0010"
Data Flit	Flit Type = "0001"
Data Flit	Flit Type = "0001"
Last Flit - Data Flit	Flit Type = "0100"

Figure 36 – A packet with four flits example.

The Network Interface (NI) is not part of the NoC but it implements some tasks that are important for the proper use of the NoC and for the correct NoC operation. In this work, the NI is responsible for the following tasks:

- Single Rail \leftrightarrow DI Code conversion: The NI receives the single rail encoding and produces the appropriated DI Code for the NoC. Conversely, it transforms data received from the NoC to single rail representations usually required by synchronous IP Cores.
- Packet assembly: The NI generates the BOP and EOP signals when transmitting packets to

the NoC.

- **Data Synchronization:** Since the NI is a synchronous module, it is necessary to synchronize the data that comes from the NoC to the NI/IP Core clock domain. The SCAFFI asynchronous interface [66] accomplishes this task.
- **Routing information management:** In the case of source routing use, the NI is responsible to insert the routing information in the first flit of the packet. The NI receives a message from the IP Core and then sends one or more packets containing the XY address of the IP Core target of the message and the message contents.
- **Data error detection and correction:** Hermes-A may contain mechanisms for data error detection and correction to deal with soft errors from radiation sources and with glitches from crosstalk, for example.

Since target SoCs in this work are GALS systems, the IP Core that generates data to send through the NoC may have an operating frequency distinct from the receiver's frequency. To decouple sender and receiver data rates, FIFO are usually inserted in the NoC. Figure 37 shows an example of a one-of-four asynchronous FIFO. The asynchronous buffer of a FIFO may also serve to avoid long wires and replace buffers. Ideally, a three-stage FIFO can break a wire with length = L in four wires with length $L/4$. This property of FIFOs helps the place and route phase of asynchronous NoCs. A FIFO can be inserted in several different positions of the NoC, such as between the NI and its router, between two routers (i.e. between an output port of a router and an input port of another) and even inside the router, between an input and an output port of this.

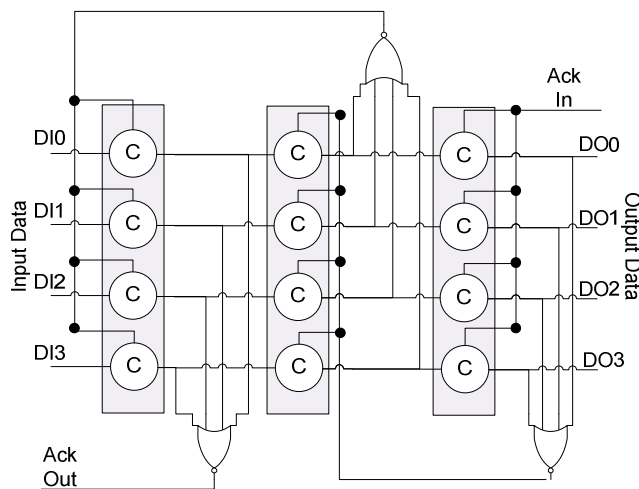


Figure 37 – A one-of-four FIFO composed by weak conditioned buffers.

Ports in the Hermes-A NoC are independent, in the sense that there is no shared logic between

ports. This occurs, for example in the synchronous Hermes NoC [54], where ports share a centralized arbitration and routing control circuit. In Hermes-A a router input port is responsible for the routing of the packet while the corresponding output port is responsible for the arbitration among concurrent requests to use it. Port independence has several advantages. At the physical layout level, it permits a more flexible floorplan, where ports may be the atomic building blocks instead of routers. This, combined with the FIFO property of breaking wires offers an enhanced capability to reduce wire sizing. Port independence also allows that the power saving techniques like Power Shut-Off and Dynamic Voltage Scaling be implemented at the port level instead of using it at the router level as proposed e.g. by Thonnart et al. [89]. In this way, just the ports involved in some specific traffic need to be active at a given moment, and this need not provoke impacts on the state of the other ports in some router. Is it also possible also to insulate an individual port, instead of a complete router, in case of permanent damage, possibly caused by hard errors. The use of physical channels to obtain QoS can also be more efficient, since it is possible to duplicate just the paths with higher traffic constraints instead of resorting to full physical channel duplication in the whole communication architecture as proposed by Carara et al. in [22].

4.1 Input Port

Figure 38 depicts the block diagram of a Hermes-A router Input Port. This Input Port consists in two modules: *Path Definition* and *Packet Dispatcher and Section Closing*. The Input Port in this work has one data input and four possible data outputs. A North Input Port for example, can send the incoming packet to the South, East, West or Local ports. The routing of a packet to the opposite direction usually does not make sense, and is thus forbidden. This means, for example that the North Input Port cannot send packets to the North Output Port. The Input Port is responsible to determine the appropriate Output Port for the incoming packet. The Path Definition Sub-Module does this when the Input Port receives the first flit. The first flit creates a *Communication Section*, i.e. it allocates some NoC resources (Input/output ports). The use of wormhole requires resource allocation that creates a reserved path similar to what happens when using circuit switching. The main difference is that here small pieces of information (the so-called *flits*) are assumed. Thus, the virtual circuit created by the first flit usually do not hold the resources for very long times. The last flit must release the components for other packets inside the NoC. The Packet Dispatcher and Section Closing module performs the releasing resource task.

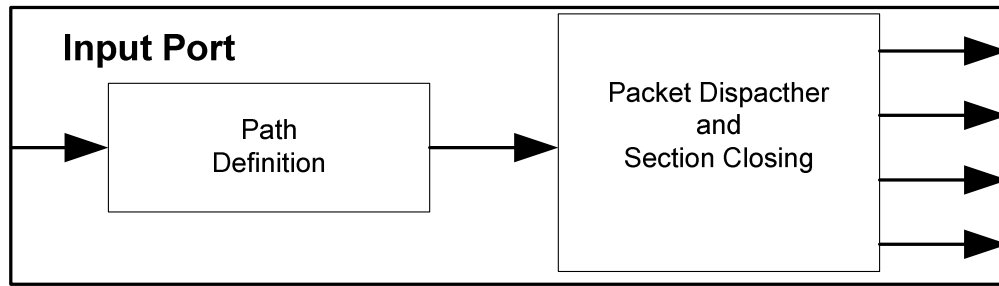


Figure 38 - Input port block diagram.

Figure 39 shows the architecture of the Path Definition module. This module is controlled by the BOP signal, e.g. by the first flit. When the first flit arrives, this module forwards it to the Path Calculation circuit. This circuit may apply a distributed routing algorithm, when the first flit contains the target IP Core X and Y address. In this case, path calculation bases its decision on the address of the current Input Port and that of the target IP Core. The routing algorithm employs DIMS logic to calculate the target Output Port. It may consider the state of the Output Port in case of adaptive routing use. The Hermes-AA NoC [70] performs such an adaptive routing. This work does not explore the techniques proposed in the Hermes-AA NoC, since this NoC and the advantages of adaptive routing are not targets here for Soft Error mitigation.

When employing source routing, the first flit already carries the routing decision to take at each Input Port. Since each of the later has four outputs, two bits are necessary to store the target Output Port. For example, a packet composed by flits with 32 bits can be used for direct communication between IP Cores positioned at most 16 hops apart.

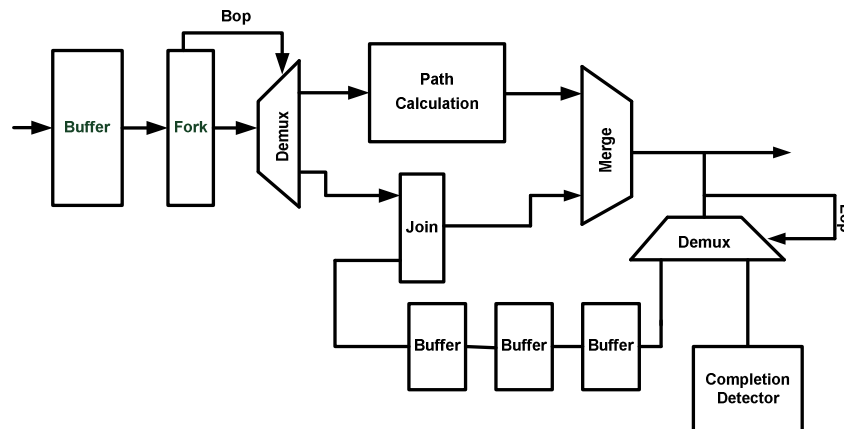


Figure 39 – Path Definition architecture.

The source routing Path Calculation module uses always the two more significant bits as the routing decision. It rotates the first flit in order to prepare the flit for the next router and keeps the routing information that the receiver can then verify. Since just two bits represent the target port, and

the router may have up to five different ports, the routing decision encoding assumes different meanings depending on what is the current port: North, East, South, West or Local. Table 4 shows the adopted routing decision encoding for each Input port. Rows correspond to the Input Port where the packet currently is, while columns indicate target Output Ports. In this way, if a flit arrives at the North Input Port and the routing decision is equal to two, the West Output Port is selected.

The main advantage of using source routing is the area reduction at the Input Ports. In fact, the path calculation module for source routing consists just in wires. Since there is no logic inside, it is a better choice for Soft Error mitigation. The advantages of distributed routing are gains in performance when adaptive routing is employed [70] and fault tolerance. A packet can reach the target even when a possible path is blocked. This blocking in wormhole switching mode can be a starvation generated by an SEE, for example.

Table 4 – Routing table employed for the source routing decision in the Hermes-A NoC.

	North	East	South	West	Local
North	-	0	1	2	3
East	0	-	1	2	3
South	0	1	-	2	3
West	0	1	2	-	3
Local	0	1	2	3	-

Comparing the Path Definition module architecture to Figure 27 it is easy note that the Path Calculation is equivalent to an IF/ELSE module. Figure 40 shows an equivalent pseudo-code for the Path Calculation module behavior. The first flit defines the target address for either source routing or distributed routing. In the case of source routing, data bits of the flit are rotated while the BOP and EOP bits stay in the less significant bit positions. A ring composed by three asynchronous buffers register the routing decision. A minimum of three buffers is necessary in a QDI ring to enable the Data/Spacer flow inside the ring [85]. In the initial state, the ring is empty (each buffer output contains a spacer). When BOP arrives, Path Calculation is performed and the result enters the ring. All subsequent flits join the routing decision. The join implementation is strongly indicating. This means that the join blocks the routing decision until another data arrives. When the last flit, marked with the EOP, arrives at the Input Port, it consumes the routing and the later does not enter again in the ring. A Demux controlled by the EOP signal sends the routing decision to a Completion Detector. This acknowledges the data and then the latter is removed from the ring. In this step, the ring is clear again, that is, it contains just spacers.

```

PathCalculation(flit)
  If first flit
    target port = Calculate_path(flit)
    rotate(flit)
    write_mem(target port)
    output data = cat(flit,target port)
  else
    target port = read_mem()
    output data = cat(flit,target port)
  end

```

Figure 40 – Equivalent code for the Path Definition module.

Figure 41 shows the Packet Dispatcher/Section Closing module in more detail. The EOP signal controls this module. If EOP is active, the path is forwarded to the S-Control circuit. This circuit is controlled by the Speed Independent S-element discussed in Chapter 2. The S-element circuit has one input channel and two output channels. When data arrive at the S-Control input it performs the sequencing of the two output channels. First, the last flit of the packet is sent to output A. After that, the *Kill Section Flit* is sent to output B. The Kill Section Flit is a special flit that the S-Control generates after it completes sending of the last flit. The Kill Section Flit is indicated by the EOP = BOP = '1'. The Output Port uses this special signaling to close the Section with the Input Port, releasing allocated resources.

The Packet Dispatcher/Section Closing module forwards the packet to the selected Output Port. It performs this action using a four-output Demux controlled by the routing decision.

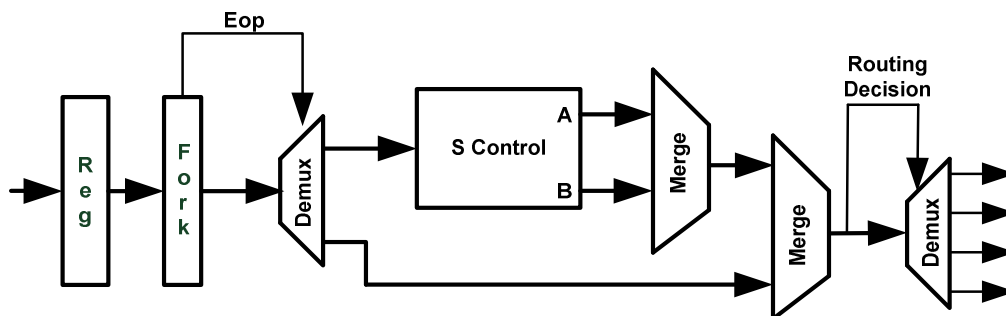


Figure 41 – Block diagram of Section Close module.

4.2 Output Port

The Hermes-A Output Port is responsible for controlling the access to shared resources. In this case, the resource is the next router Input Port or the access to the NI. To perform this control action, it is necessary the implementation of arbitration mechanisms to decide which incoming traffic to serve at each specific moment.

Figure 42 shows an example of a four-input Output Port. The first flit generates a request to the arbiter. The arbiter chooses one among the incoming requests to grant. The Output State of the Output Port can be used to sniff and evaluate the occupation of the NoC Ports at the run time. The sampling of the port state can be used to implement the adaptive routing [70], trigger a procedure to update the routing tables based on the congestion information, control a timer that checks if the Ports are blocked due a stall in the path. Since the Output Port feeds a single Input Port, the occupation rate of the Output Port is the same for the following Input Port.

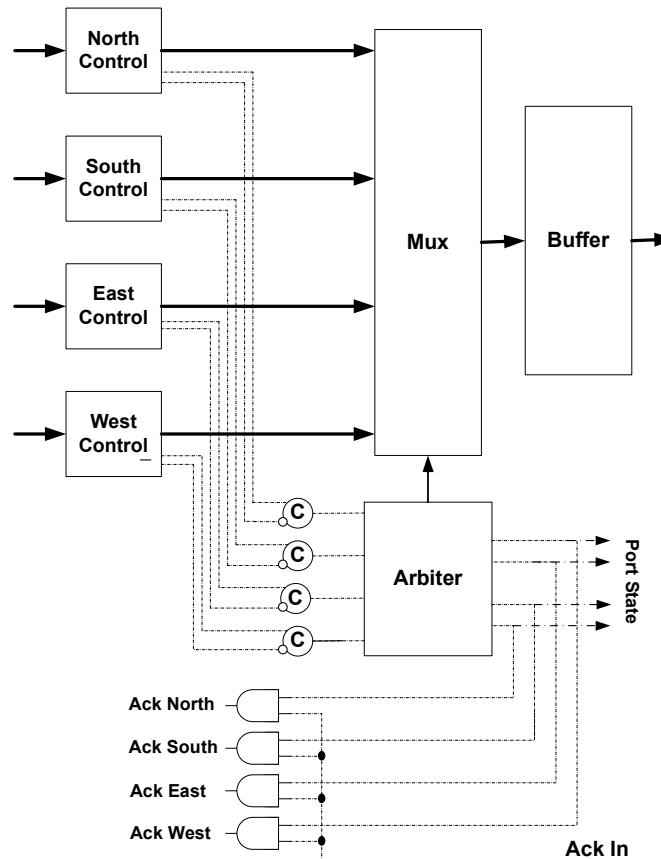


Figure 42 - Block diagram of a 4→1 output port. The dashed lines represent a single wire while the filled lines are DI encoding.

The arbiter must guarantee fairness among all incoming requests. The arbiter adopted in Hermes-A follows the same structure as the arbiter employed in [8]. In that work, fairness of such an

arbiter circuit is explained and proved.

The arbiter request must be persistent until the end of the whole packet transmission. This occurs by applying a dual-rail to single rail transformation using a C-element. The C-element guarantees that the request will keep the request even when the spacer is being transmitted. The output of the arbiter is used to select the port that receives the input acknowledge and to control the output multiplexer.

Figure 43 details the Output Port control circuit in detail. The arbiter request is generated when the BOP is received. The request is released just when the BOP = EOP = '1'. This is the value corresponding to the Section Killer Flit. This flit is not transmitted to the output Multiplexer. Instead of this, it is sent to a Detection Circuit that generates the necessary acknowledge signal to perform the four-phase handshake protocol.

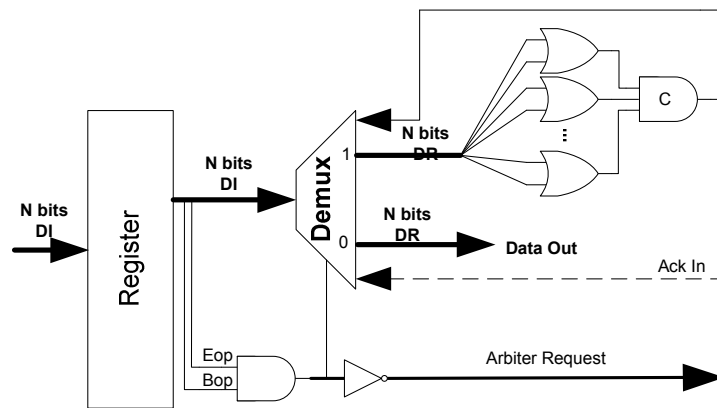


Figure 43 – Output Port Control Circuit.

Figure 44 shows an example of a two-flit packet transmission through the Output Port. During the first flit, the control circuit performs the arbitration request. When the last flit arrives, it is transmitted as a regular flit. The Section Killer Flit is then responsible for releasing the arbiter. The four-phase protocol is completed by the detection circuit and communication ends.

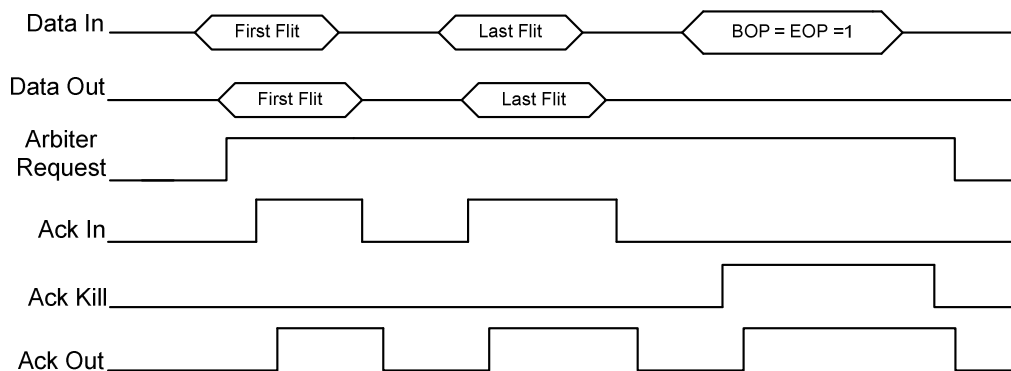


Figure 44 – Waveform example of a two flits packet transmission through an output port.

5. HARDENED HERMES-A (H₂A)

The construction of circuit modules robust to SEEs is a complex task, due to the number of considerations needed to ensure that the module behavior can resist to the many distinct effects possibly caused by high-energy particles traversing the silicon where the module resides. A coarse classification of such effects split them into effects on data and control parts of the module, because each of these are amenable to treatment by distinct robustness enhancement techniques. However, a more detailed view of SEE effects may benefit the achieved module robustness. Accordingly, this work addresses robustness concerns at five distinct levels:

- Cell level – Different logic components implemented e.g. as standard cells react differently to SEEs, depending on the transistor interconnection pattern, number of pins and individual cell layout;
- Logic level – Logic Gates and flip-flops interconnected in a netlist may undergo specific techniques that render the overall netlist more robust to SEEs;
- Handshake protocol level – Most asynchronous handshake protocols employ sequential components that subject to SEEs may corrupt the protocol working. Specific techniques may be designed to guarantee reliability to the protocol under the effect of SEEs;
- Data level – It is unnecessary to stress how relevant is to guarantee data corruption does not occur under the effect of SEE. Specific techniques to address this issue are thus very relevant;
- Link level or data protocol level – Specifically in NoCs, the reliability of data and control links (e.g. between routers) is crucial. Devising techniques to enhance link reliability is thus important.

Sections 5.1 to 5.5 of this Chapter respectively discuss each of the levels above. They propose a set of techniques to enhance robustness at each abstraction level. The starting point is the Hermes-A design. The result is a new NoC architecture, called Hardened Hermes-A or H₂A for short.

5.1 Cell Level Hardening

5.1.1 SEE impact on C-elements

In order to analyze precisely QDI pipelines behavior under effect of SEEs, it is first necessary to examine the behavior of the C-element under radiation. This behavior is presented in Figure 45 by a state graph that concerns a 2-input C-element in the presence of radiation that causes SEEs. In this directed graph, each vertex corresponds to a state of the C-element, represented by a binary vector

where the two leftmost bits represent the instantaneous input values and the rightmost bit is the instantaneous output value. Edges represent transitions

The C-element can present SETs or Single Effect Upsets (SEU). The resulting effect depends on the state of the victim C-element when it hit by a particle. In states 000 and 111, the C-element has a direct electrical path from the inputs to the output. In this case, a particle hit can generate an SET. In states 010, 100, 011 and 101, the C-element acts as a memory element. In these cases, radiation may cause SEUs.

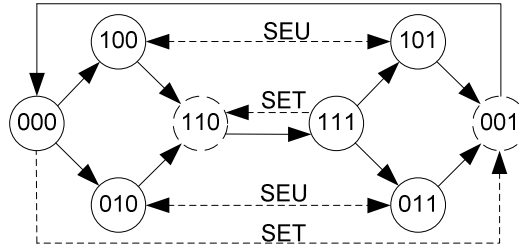


Figure 45 - State transition graph of a C-element under presence of SEE.

Table 5 provides the normalized SEE critical charge that is required in order to generate an SEU or SET on the analyzed C-element, according to its state when hit by a particle. The normalization was done in function of the biggest charge (State = “111”). This result has been extracted by precise Spice level simulation. It is interesting to note that the SEE amplitude required to have the C-element flip is smaller when the C-element is holding its output, compared to when the C-element is driving its output (000 and 111 state).

Table 5 - SEE critical charge to generate a fault according to the current C-element state when it is driving a charge of 8.1 fF.

State	C-element States					
	000	010	011	100	101	111
Normalized Critical Charge	0.720	0.088	0.120	0.097	0.100	1.000

5.2 Logic Level Hardening

Chapter 2 presented an implementation of a set of asynchronous components employed to build the NoCs discussed here. Some of these components have a major impact on the NoC behavior. An example is the arbiter circuit, responsible for keeping the routing decision and the speed independent control circuits like the S-element. In order to achieve more robust NoCs, these basic components were chosen to be hardened. The hardened implementation of these components significantly affects the NoC robustness, as Chapter 8 demonstrates.

5.2.1 Double Check

Double check is a spatial redundancy technique that Jang and Martin proposed [39] to filter SEE mainly in combinational logic. This technique consists in duplicating the logic and performing a logic check using the event synchronization property of C-elements. Figure 46 shows an example of the Double Check technique applied to a three input logic. In the Figure, *Logic A* and *Logic B* are two instances of a same circuit. The inputs of the circuits are also equivalent, i.e. ($X_A = X_B$, $Y_A = Y_B$ and $Z_A = Z_B$). The C-elements with outputs Q_A and Q_B are responsible for synchronizing the results of the logic. Updates in outputs Q_A and Q_B obviously only occur when the C-element inputs are equal. In this way, the C-elements filter any SET in Logic A or in Logic B. Since during normal operation each C-element input pair have always the same value, C-elements may only present SETs.

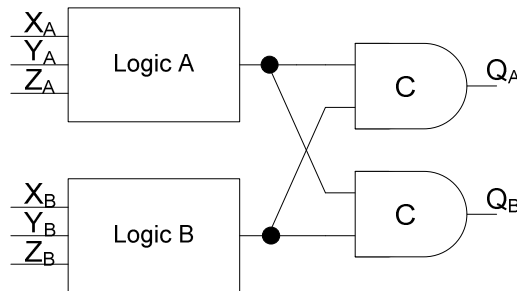


Figure 46 – Example of the double check technique applied to a three-input one-output logic. In the Figure, Logic A and Logic B are instances of a same circuit and inputs are equivalent.

A hardening option for a NoC design is to duplicate all the circuitry and use the double check technique in the asynchronous buffers. This however incurs in large area overheads since it requires duplication of all gates and wires. Besides, the technique implies the insertion of the output double check C-element gates. Double check reduces the performance of the circuit also. In this work, some weak points of the routers were identified and the double check technique was the applied at these points. This reduces area, timing and power overheads and allows the achievement of a high level of robustness.

5.2.2 Double Check Arbiter

Arbiters perform sequencing of packets in NoC Output Ports. In each Output Port, a SET in the arbiter decision may break the arbitration sequence and mix two distinct packets or create a new flit. Double check at the logic level can mitigate these effects with low area overhead, since the arbiter occupies just a small portion of the Output Port. However, duplicating the arbiter is impossible. The arbiter functionality consists in the sequencing of input events. In this way, is not easy to guarantee that two duplicated arbiters will produce the same output. Figure 47 shows the application of double

checking to the arbitration logic. For example: Consider that in Arbiter A request $R1_A$ arrives before request $R2_A$ with a small time difference but in Arbiter B $R2_B$ arrives before $R1_B$ due to timing differences in wire/gate propagation delays. This may result in a deadlock in the arbitration mechanism. The deadlock arrives due the fact that the arbiters take different decisions and the C-elements at the output filter result of both arbiters since they are not equal.

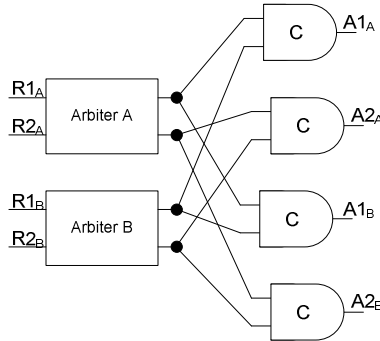


Figure 47 – A possible double check arbiter implementation.

To solve this problem, the arbiter double check design uses a distinct structure displayed in Figure 48. This Figure shows the double check implementation equivalent to the four input arbiter presented in Chapter 2 and used in the Hermes-A Output Port. To support this double check implementation, the arbiter request is checked with the output of the arbiter. The arbiter output has the same four acknowledge signals than the implementation showed in Chapters 2 and 4. The acknowledge signal, in the Output Port implementation, is used to control a multiplexer.

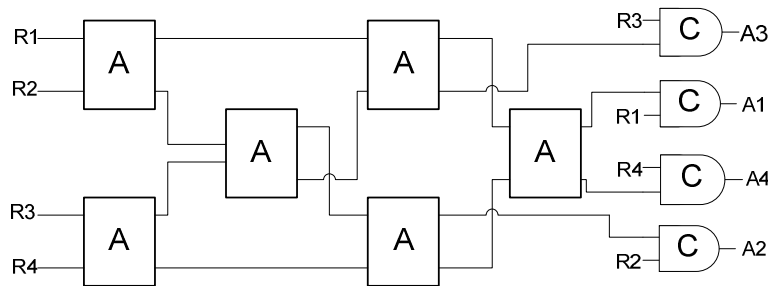


Figure 48 – The double check implementation equivalent to the four-input arbiter presented in Chapter 2 and used in the Hermes-A Output Port. Individual arbiters labeled with letter A are equivalent to the circuit presented in Figure 30.

5.2.3 The Double Check S-element

The S-element is another critical circuit that is responsible for performing packet sequencing. However, the S-element does not present any race between the inputs as the arbiter. In the Input Port, the S-element is responsible to send the last flit, generate the kill flit and execute the four-phase protocol for both flits. The S-element is a speed independent control circuit. This means that it is robust

to delay variations in its gates. However, the glitches generated in the signals from the circuit environment or the SEE inside the circuit can lead the circuit to an unspecified state in the transition graph that describes the circuit (see Figure 29). This situation may stall the circuit operation and as a consequence the Input Port. Because of the use of wormhole switching, this stall may propagate to several routers inside the NoC. Another consequence is the generation of the kill flit out of position. If the kill flit generation occurs outside the scope of a packet transmission, it will be removed from the NoC in the output ports. However, if it happens in the middle of a packet, the packet may lose its grant in the arbitration at the next Output Port. This may result in incomplete packets and in the stall of some components, since the packet protocol does not end properly.

The double check technique was used for the S-element also. Here the S-element is duplicated and the circuit outputs and the feedback signal are checked by C-elements. Figure 49 shows the double check implementation of the S-element. Again, the area overhead is not an issue, since the S-element is a small control circuit present just at the Input Ports. Wires RA, RB and AA, AB are equivalent to the Req A, Req B and Ack A, Ack B in Figure 29.

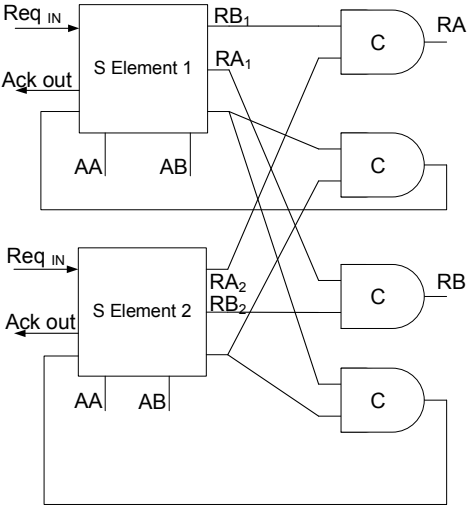


Figure 49 – The structure of a double check S-element.

5.3 Handshake Protocol Hardening

Asynchronous data transmission is always controlled by some handshake protocol. This work assumes the use of a 4-phase protocol. This protocol has four events that must occur in a well-defined sequence. If the four-phase protocol events sequence is changed, this may result in data transmission stall. This Section first analyzes the impact of SEEs in a four-phase protocol. Next, it proposes a technique to increase the protocol robustness.

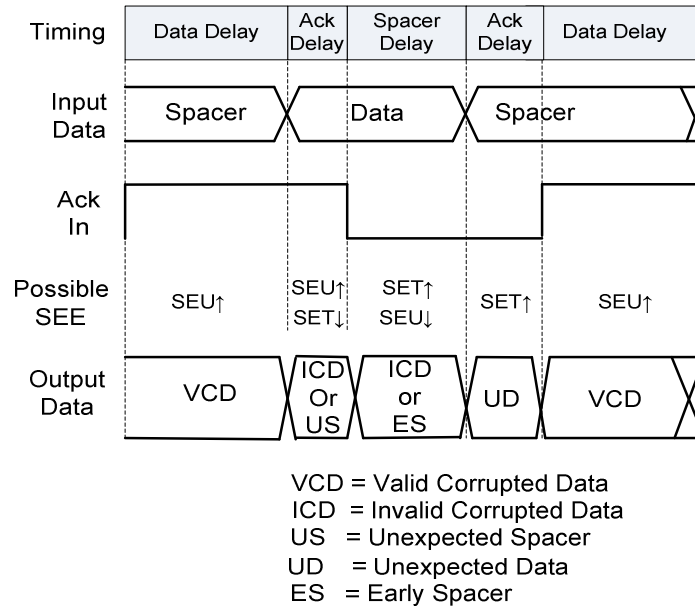


Figure 51 - SEE and timing diagram for the 1-of-n pipeline 4-phase protocol, displaying all possible consequences of SEEs in each phase of the protocol.

The *Data Delay* in Figure 51 is the delay between the *Ack Out* Rising Transition (Figure 50) and the generation of a new data transition by the buffer environment. The *Ack Delay* is the delay between data propagation through the following stage of the pipeline and the consecutive data detection. The *Spacer Delay* is similar to the *Data Delay*. It starts with the Falling Transition of the *Ack Out* signal and stops when the circuit environment removes the data from the input. The possible SEE within each timing window relates to the C-element states in the timing window. In the *Data Delay* timing window, C-elements of the asynchronous register are in the retention state, since one of its inputs is '0' (Input Data Signal) while the Ack Signal is '1'. In this state, all cells in the register are susceptible to a SEU \uparrow (considering that all the outputs of the register are '0'), that may generate a Valid Corrupted Data (VCD). A VCD is a data token generated by the SEE that has a valid 1-of-n encoding. Completion detectors may notice the VCD, and propagate it from one end to the opposite end of a pipeline. Depending on the sequence of the events, this new token can be:

1. An early data indication: If the real data arrives in the same step of the 4-phase protocol (before the propagation delay of the next register stage delay, plus delay to obtain completion detection at the next stage) and the data wire is the same victim wire.

2. Incomplete Corrupted Data (ICD): real data arrives in the same step of the 4-phase protocol but in a different wire. The occurrence of an ICD means that the bit that carries the data is still present but one new bit flipped, resulting in an invalid 2-of-n symbol. The receiver may easily detect this erroneous data, but it is not possible to determine which wire carries the correct information.

3. If the data arrives after the next register stage delay plus the delay to obtain completion detection at the next stage, then the register will be closed, preserving the VCD.

An SEE that occurs inside the *Ack Delay* timing window has two possible effects: an Invalid Corrupted Data (ICD) in the case of an SEU \uparrow and an Unexpected Spacer (US) in the case of a SET \downarrow . US means that some datum disappears, generating a spacer before the currently 4-phase protocol finishes. As Table 5 presents, this occurs in the state where the C-element presents the highest critical charge value. Besides, this is a transient event. This means that after some amount of time, corresponding to the amount of charge injected, the correct data will reappear at the pipeline stage. The Unexpected Spacer (US) and the Unexpected Data (UD) presented in the *Ack Delay* timing windows may change the sequence of the protocol and the result can be a stall in the pipeline.

The completion detection of 1-of-n circuits is done by an n-input NOR gate. In this way, the only possible SEE is a SET. However, one way to increase the robustness of 1-of-n codes is to add detection based on several 1-of-n encodings, by using a C-element tree as Figure 13 shows. In this case, data become valid just after all 1-of-n encodings involved on the detection become valid. This simple modification strongly changes the timing behavior of the pipeline. Applying this model of completion detection in a dual rail pipeline with 8 bits, the completion detector must detect four changes before it changes its state. This changes the timing properties of the pipeline to a behavior closer to the 4-of-8 encoding rather than the 1-of-n. Another impact is that the generation of an SEE in any cell of the completion detection except in the last C-element of the tree will simply be propagated to the output, if completion detection is already switching. This means that only early events may be generated. The only susceptible cell is the last C-element of the tree. This type of detection increases the robustness of individual 1-of-n but adds delays to the detection circuit and consequently increases the cycle time of the circuit. In the NoC under consideration, the C-element tree is used in the completion detectors inside the Input Port and Output Port, while in the FIFOs detection occurs at each single data encoding.

Next, Figure 52 shows the timing and SEE analysis for a m-of-n asynchronous buffer. M-of-n DI codes are more robust to SEEs if $m > 1$, since every codeword is always m bits distant from the spacer encoding. This means that an SEE is not able to create new valid codewords or clear some data from the data link if the data is not already switching.

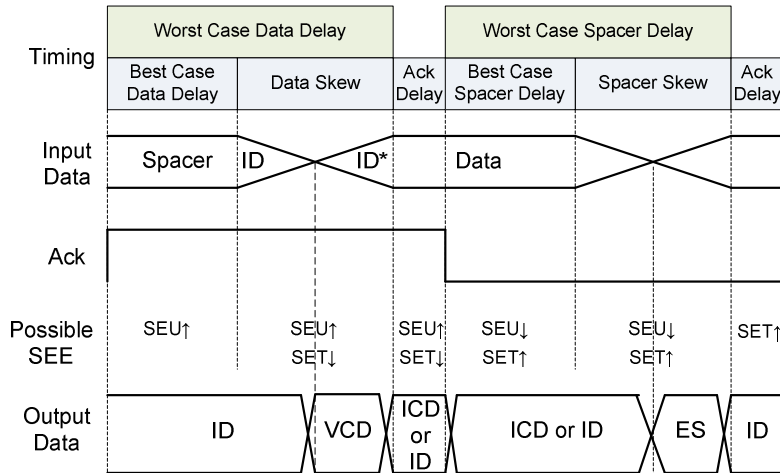


Figure 52 – Timing and SEE analysis of the m-of-n 4-phase protocol for an asynchronous buffer, displaying all possible consequences of SEEs in each phase of the protocol.

For each phase of the protocol, the timing description appears on the top of the Figure. The Best Case Data Delay indicates the fastest path between the previous stage and the register input. Data Skew is the timing difference between the fastest and the slowest bit in the data-path. The Acknowledge Delay (*Ack Delay*) is the time needed from data propagation to the completion detection plus the detection delay. SEEs observed in each timing window result from the C-element behavior.

In the Best Case Delay timing window, the only possible SEE is the occurrence of an SEU \uparrow . This event generates Incomplete Data (ID). This is the main difference between this and the timing behavior of 1-of-n codes (Figure 52) where a VCD may be generated and a new token may be erroneously created.

The Data Skew timing window divides itself into two other windows. The Input Data State delimits the frontier between these new windows. If the m-1 bits of the Input Data already switched (ID*), then the encoding reaches an excited state, otherwise the inertial property of the code will keep filtering any event at the Input Data. The Data Skew ID* timing window is the only window where a VCD may happen. Since the data is already switching during this window, it is expected that the remaining data wire switch before the Ack-In signal arrives (this is the event that closes the register for the data propagation). If the data wire arrives before the acknowledge signal, the result will be an ICD instead of a VCD. In this way, the VCD occurrence is avoidable at design time, by guaranteeing that the data skew is smaller than the Ack-In propagation. Data skews are usually rather small, since all data signals are generated in the rising edge of the acknowledge signal. The environment must always detect and sample the data before acknowledging it. Looking inside the pipeline, it is possible to note that the acknowledge propagation will correspond to the time to cross the next stage register plus the completion detection time. This gives enough time to generate the missing data wire

value.

The timing property of m-of-n codes causes a conceptual enlargement the ICD window inside the protocol. An ICD presents the correct data information plus an extra bit that turns the DI code invalid (m+1-of-n). The invalid code presented by a corrupted data is able to cross the pipeline stages and arrive to the final data receiver since it is detected by a regular completion detector implementation.

By comparing Figure 51 and Figure 52, one can finally note that the unexpected data and spacer (UD and UE in Figure 51) do not appear in the protocol when using m-of-n codes. Even if m-of-n codes display better filtering properties than 1-of-n codes, the ICD fault is still present. The main advantage here is that this class of errors may be detected easily at the receiver side. However, no correction can be done at the pipeline level.

5.3.2 Normal Open Asynchronous Pipeline (NOAP)

The timing windows presented in Figure 51 and Figure 52 are strongly dependent of the Sender/Receiver frequency operation. However, there are modifications in buffer control implementation that may change the timing window as well. Bainbridge and Salisbury [9] present some asynchronous pipeline modifications capable of reducing circuit glitches generated by crosstalk. This work proposes a modification of pipeline to change the timing windows, to increase four-phase protocol robustness. The name of this technique is Normal Open Asynchronous Pipeline (NOAP). The main idea is to synchronize data and acknowledge at each buffer stage.

Figure 53 shows a pipeline implementation using the NOAP technique. The only difference of this implementation with regard to the pipeline of Figure 50 is the insertion of a C-element to synchronize data and the acknowledge signal. In the initial state, Input Data = “0000”; in all stages the additional C-element (referred here as NOAP C-element) has inputs at “01” (‘0’ from the pre-buffer detection and ‘1’ from the acknowledge signal). Since the NOAP C-element was initialized (the reset signal is omitted from the Figure), the output is equal to ‘0’. In this way, the C-elements of the buffer have all inputs equal to “00”. This means that the buffer is transparent to the spacer. No data can cross the buffer before the data/acknowledge synchronization. In this initial state, just SET \uparrow can happen in the buffer C-elements but this SET is filtered by the next stages of the pipeline that are in the same state.

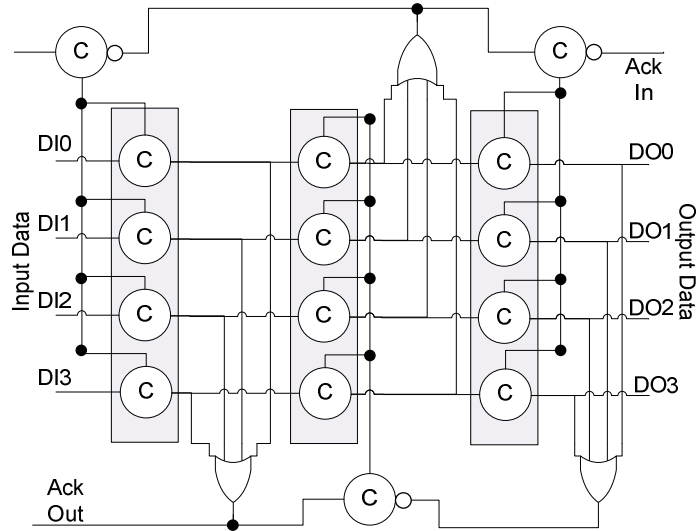


Figure 53 – Normal open asynchronous pipeline register for a 1-of-4 pipeline.

Once some new data arrive, these must be detected by the Completion Detector and the result of the detection will enable the NOAP C-element. The NOAP C-element then changes the buffer state. The buffer becomes transparent to Data like the normal buffer implementation. The main difference is that when the acknowledge arrives, the NOAP C-element do not close the asynchronous buffer. This buffer stays open until the synchronization between data and acknowledge takes place. Thus, under normal operation the C-element is open. The main advantage of the NOAP implementation is the filtering property, mainly in the empty state. In NoC implementations, several NoC links with a low rate of usage stay for long periods in this state. In this way, this technique is a good option for this kind of links.

Figure 54 shows the timing and SEE analysis of the NOAP four-phase handshake protocol. The modification in the protocol brings two main advantages related to pipeline robustness:

- **Circuit electrical filtering increase:** In the NOAP handshake protocol C-elements that are part of the WCHB stay in states “111” and “000”. In a conventional asynchronous pipeline implementation, C-elements inputs remain mainly in “01” and “10” states. Considering Table 5, it is possible to note that the new implementation increases the overall critical charge.
- **Circuit logic filtering increase:** The main advantage of NOAP is the change in the timing windows of the four-phase handshake protocol. The NOAP timing windows reduce the VCD and stall probability. It reduces the probability of an SEU in the asynchronous buffer as well. Since C-elements of the asynchronous buffer are enabled just when both data and acknowledge arrive, NOAP has an ability to logically filter SET transients in the data sig-

nals as well as in the acknowledge wire.

The main drawback of NOAP is the additional propagation delay in the forward and backward path. The consequence is the increase in the pipeline cycle time and a consequent reduction in communication throughput and in latency as well.

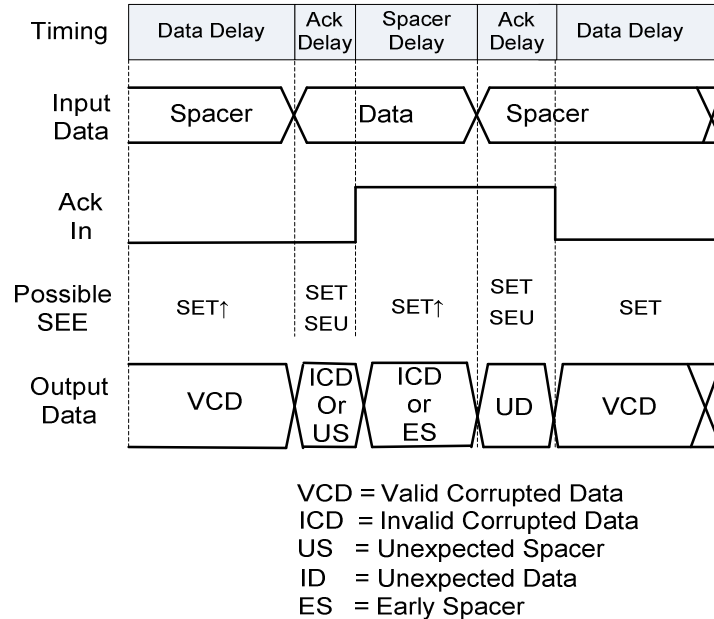


Figure 54 – Timing and SEE analysis for the NOAP technique.

5.4 Data Level Hardening

This Section proposes two data level hardening mechanisms. The first is the adaptation of widely used parity schemes for data transmission of information using unordered, non-systematic codes. The second technique is the insertion of convolutional codes combined to the parity and unordered schemes.

5.4.1 QDI Parity Check

Parity check, also known as Hamming Code, is the most simple data error detection/correction mechanism for digital data transmission. Parity check is a block code where redundant bits are added to a message to enable error detection and or correction.

The use of unordered codes also enables error detection in data transmission. Here parity check is used associated to unordered encodings to detect and correct data errors in DI data transmission.

This work assumes that the communication system is a GALS SoC where the IP Cores are synchronous, while the communication medium is an asynchronous channel employing some m-on-f

code. In the synchronous side, binary single rail encoding is used. This means that each datum is represented as a binary data vector $d = [a_0, a_2, \dots, a_{k-1}]$, where for all a_i , $a_i \in \{0, 1\}$. To perform some data communication through the NoC, data is translated to an m-of-n encoding by a bijective function denoted here by $DI(d)$. The binary data vector has length $L(d)$ equal to k , where, as usual, the length is defined as the number of bits in the codeword. Of course, in this case, the number of possible different codewords that can represent some data is 2^k . The number of possible codewords encoded by a single m-of-n vector is defined by the combination:

$$(3) \quad C_{n,m} = \frac{n!}{m!(n-m)!}$$

Therefore, the number of m-of-n codewords $S(d)$ employed to cover the binary data vector d is:

$$(4) \quad S = \text{ceiling}(\log_{C_{n,m}} 2^k)$$

Any data with $L(d)=k$ is transformed into a m-of-n data matrix (flit) $F = [f_{i,j}]$ $i=1, \dots, s; j=1, \dots, n$, where $s = S(d)$ and n is the number of wires of the m-of-n encoding.

For example, the binary vector $d = [0111100110111100]$ with Length $L(d) = 16$ when converted to the 1-of-4 encoding by $DI(d)$ generates a 1-of-4 data flit matrix:

$$(5) \quad F = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

In (5) the number of columns is equal to the number of wires in the m-of-n encoding (four in this example) while the number of lines is $S(d)$ and for all f_{ij} , $f_{ij} \in \{0,1\}$.

Starting from (5), a parity vector $P = (p_0, p_2, \dots, p_{n-1})$ with length $L(P) = n$ is generated as follows:

$$(6) \quad \forall_i, P_i = f_{i,0} \oplus f_{i,1} \oplus \dots \oplus f_{i,s-1}$$

Equation (6) is equivalent to the XOR operation applied to all elements of the same column.

In the previous example, for Matrix (5), the parity vector P would be computed as:

$$(7) \quad P(F) = (1001)$$

After generation, the parity vector is translated to the appropriate m-of-n encoding to transmit in the asynchronous NoC. Translation is done by the bijective function $DI(P)$. The results are $S(P)$ a new set of m-of-n codewords that are concatenated to Matrix F to generate the Extended Flit EF . In (8), the last two rows are the equivalent 1-of-4 encoding $DI(P)$.

$$(8) \quad EF = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Figure 55 shows an example of the parity scheme over the 1-of-4 encoding. In the example there are four 1-of-4 codewords $[[A_0, A_1, A_2, A_3][B_0, B_1, B_2, B_3][C_0, C_1, C_2, C_3][D_0, D_1, D_2, D_3]]$. Each wire of the 1-of-4 encoding has an associated parity bit, as explained before. P_0 is the parity bit of vector $[A_0, B_0, C_0, D_0]$; P_1 is the parity bit of vector $[A_1, B_1, C_1, D_1]$; P_2 is parity bit for vector $[A_2, B_2, C_2, D_2]$ and P_3 is the parity bit of vector $[A_3, B_3, C_3, D_3]$. Parity bits are translated to the 1-of-4 code in order to be transmitted with the data.

In the running example, the result of the translation are two new 1-of-4 codewords $S_A = "0010"$ and $S_B = "0100"$.

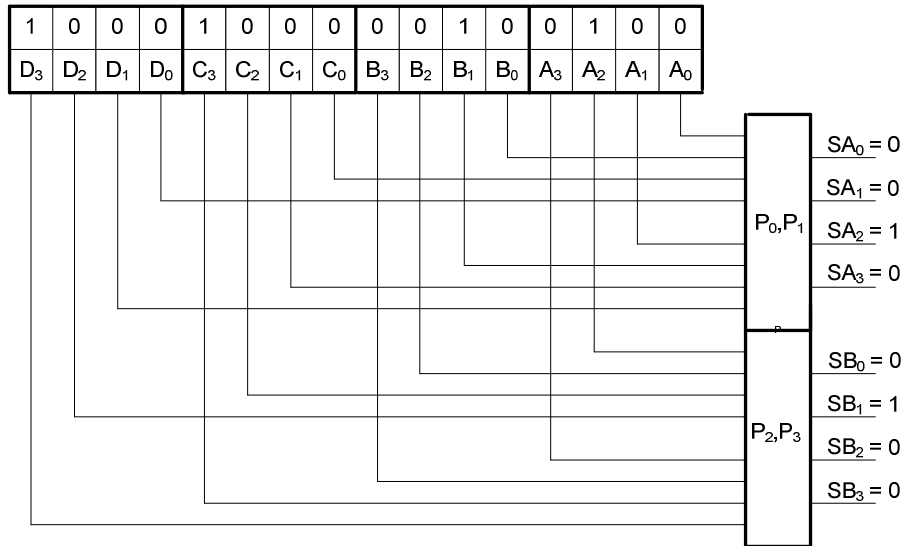


Figure 55 – Parity check when using 1-of-4 DI code for NoC transmission.

Figure 56 shows the communication architecture of a GALS SoC based on the H2A NoC employing the proposed parity scheme for m-of-n codes. The Sender IP Core is a clocked module synchronized with the Parity Encoder module. The Receiver IP Core is another clocked module synchronized with the Parity Decoder.

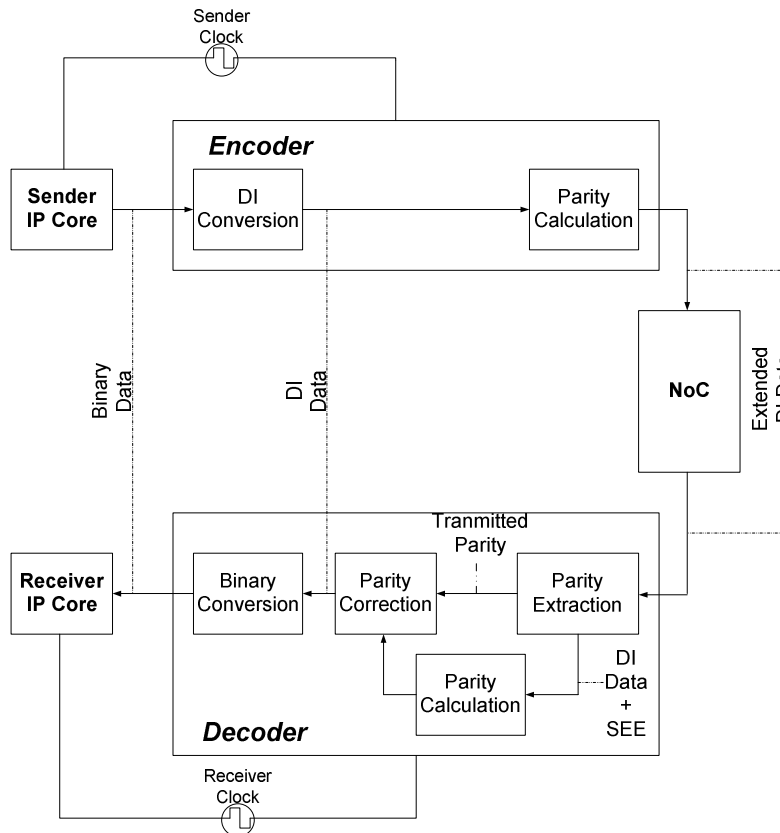


Figure 56 – Communication Architecture using the proposed parity scheme for m-of-n codes.

The Parity Decoder is divided into three modules. The Parity Extraction removes the parity bits from the Extended Flit (matrix EF) and forwards these to the parity check unity. The Parity Calculation module receives the Flit (matrix F), with possible bit flips due to SEEs, and applies the same parity generation method used in the Encoder. The Parity Correction module receives the flit matrix F and the two Parity vectors. If the transmitted vector contains an ICD, the correction it is not performed. In this case, any ICD detected at the Flit is indicated as an error to the IP Core. If the transmitted parity vector does not contain an ICD, then the correction algorithm is performed. In the correction algorithm, an XOR operation is performed between the two parity vectors. If the result is non-zero, there are errors in the Flit. The position of the '1' in the resulting XOR indicates the position of the error. To identify the m-of-n encoding that contains the error, the Rows of the matrix are verified to identify an ICD. When an ICD is detected, the operation defined in the truth table depicted in Table 6 is performed wire by wire.

Table 6 – Parity correction operations.

Data	WireParity	Corrected Data
0	0	0
0	1	0
1	0	1
1	1	0

5.4.2 Temporal Redundancy in 1-of-4 DI Codes (1-of-4 TRDIC)

In order to increase the robustness of QDI data links, this work proposes the insertion of temporal redundancy in the DI encoding by using a convolutional encoding combined with the unordered 1-of-4 encoding. This is the second version of the Temporal Redundancy Delay Insensitive Code (TRDIC) Code. The first version of the TRDIC was proposed by the author in [72]. It consisted in the conversion of 1-of-n codes to 2-of-n+1 codes by using the extra valid codewords of the new code to add temporal redundancy. The TRDIC as initially proposed has the ability to filter, detect and correct errors caused by glitches in asynchronous data transmission. However, the TRDIC as proposed in [72] can only guarantee data error detection and correction of all single data errors.

This work proposes a new TRDIC implementation that explores the 1-of-4 code. The new encoding proposition is capable to detect any error in 1-of-4 codes and correct most such errors. Here, data encoding and decoding is done at each single 1-of-4 codeword. In the new proposition, each 1-of-4 codeword transmits just one data bit. Since the 1-of-4 encoding allows four different codewords, the remaining codewords serve to insert the temporal redundancy. Since encoding/decoding applies to 1-of-4 unit codewords, this new TRDIC can detect/correct multibit errors during data transmis-

sion. This is useful in NoCs since SEEs can generate faults that remain undetectable inside the NoC Input/output ports until a packet crosses these components and carry these errors with it.

Figure 57 shows the trellis representation of the TRDIC encoding. Each vertex of the trellis is a state and the edges are equivalent to the data transmission of some data bit. Starting from the “0001” state for example, the TRDIC uses the “0001” data encoding to transmit a bit in ‘0’ and the “0100” to transmit a bit in ‘1’. For each state, the set of 1-of-4 codewords ({"0001", "0010", "0100", "1000"}) is divided into a subset containing just two valid codes. For the state “0001” for example, this subset is {"0001", "0100"}. If the data bit ‘1’ needs to be transmitted, the Data Encoder generates the “0100” 1-of-4 encoding that is transmitted. This changes the state of the Trellis from “0001” to “0100”.

The error detection circuit verifies if the trellis transition is valid and if the received data is a valid 1-of-4 codeword. If an ICD arises, it is easily detected, since it is not a valid 1-of-4 codeword. To correct the data it is necessary to look at the trellis present state and to its previous state, and the next data to correct in the path of the trellis. In this way, the TRDIC always add a latency of three receiver clock cycles to each flit.

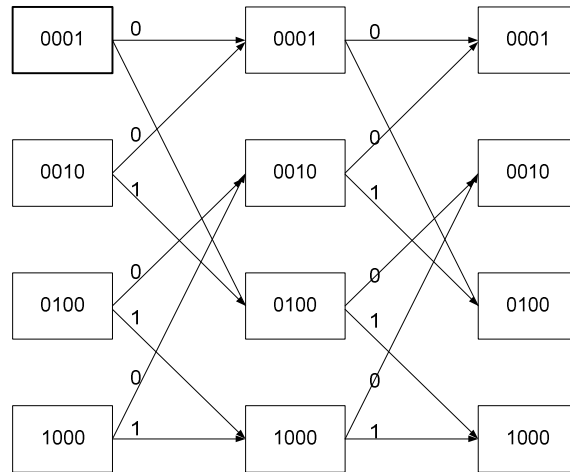


Figure 57 - Trellis TRDIC encoding based on the 1-of-4 DI encoding.

Figure 58 shows the general structure of the TRDIC communication system. Communication starts with some data encoded as a regular binary number. Next, the data conversion by the TRDIC encoder takes place, producing a temporally redundant 1-of-4 codeword. The new codeword follows to a regular 1-of-4 QDI link until it reaches the receiver. Before data reception of the data may occur. The TRDIC Decoder processes the data, which perform the necessary error corrections using a trellis decoder. In this way, temporal redundancy can be used jointly with the spatial property of unordered codes to detect single errors.

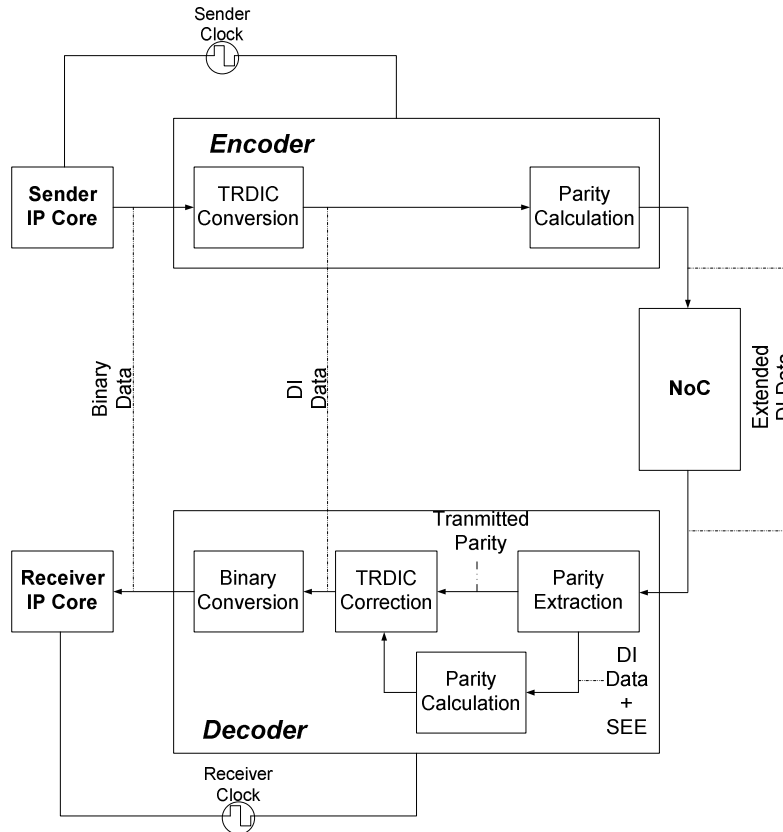


Figure 58 - Overview of the proposed TRDIC communication system.

Temporal Redundancy Encoder

The 1-of-4 TRDIC Encoder consists of a circuit responsible for combining the present state of the trellis with the current data, using the conversion table in Table 7. The NoC then transmits the resulting state. The conversion process based on Table 7 can be represented $TRDIC(s_{i,k-1}, a_{i,k})$, where $s_{i,k-1}$ is the i -th state at iteration $k-1$ while the $a_{i,k}$ is the i -th bit of the data vector d to transmit at iteration k .

Table 7 – State transition table for the implemented TRDIC.

Current State	Data bit	Next State
"0001"	'0'	"0001"
"0001"	'1'	"0100"
"0010"	'0'	"0001"
"0010"	'1'	"0100"
"0100"	'0'	"0010"
"0100"	'1'	"0010"
"1000"	'0'	"1000"
"1000"	'1'	"1000"

In the transmission of a k -flit packet a data flit is represented by a binary vector $d=[a_0, a_2, \dots,$

$a_{n-1}]$, where for all a_i , $a_i \in \{0, 1\}$ and the length of the data is $L(d) = n$. This data requires a state vector $S = [s_1, s_2, \dots, s_{k-1}]$ containing n state elements where for all s_i , $s_i \in \{“0001”, “0010”, “0100”, “1000”\}$. In the initial state the vector is defined by expression (9).

$$(9) \quad \forall_i, S_{i,k} = 0001$$

Data flit conversion follows Equation (10):

$$(10) \quad \forall_i, S_{i,k} = TRDIC(s_{i,k-1}, a_i).$$

In this last Equation $S_{i,k}$ denotes the i -th row of the State matrix S in iteration k .

Expression (11) shows an example of the matrix of states S_{k-1} and S_k , where S_k is generated from S_{k-1} and forms the data vector $d = [01010000]$.

$$(11) \quad S_{k-1} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \rightarrow S_k = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

There are some special cases in the implemented TRDIC encoding scheme. The first transmitted flit does not employ temporal redundancy. In the first flit, the State Matrix returns to the initial state. In this way, the second flit utilizes the initial state during the encoding phase. The bits used to encode the Flit Type information are not encoded with the TRDIC either. The next Sections describe special techniques for hardening these special data fields.

The Encoder also adds parity bits to all flits to help in Data Error Correction. The parity bits generation follows the binary to TRDIC conversion and employs the same method explained in the previous Section.

Temporal Redundancy Decoder

The TRDIC decoder uses the trellis structure to detect and correct the errors. To implement the trellis, TRDIC keeps the two last received data in the buffer. It keeps also the last two parity indications. Figure 59 illustrates this. In this way, the TRDIC decoder performs the following decoding procedure:

$$(12) \quad TRDIC^{-1}(S_k, S_{k-1}, S_{k-2}, Pe_k, Pe_{k-1}, Pe_{k-2})$$

The TRDIC function can be implemented by a table where for each input combination of the parameters in (12) there is an associated entry. The problem of this implementation is the number of different combinations for the table. However, several combinations, that have the Hamming Distance bigger than the trellis correction capability, can be neglected in the decoder implementation without incurring in loss of correction capabilities. For example, if the S_k , S_{k-1} and S_{k-2} are equal to “0001”, “0010” and “0100”, the combination “1000”, “1000” and “1000” can be excluded from the list of possible correct paths, since the trellis could not correct it anyway.

The bits Pe_k , Pe_{k-1} , Pe_{k-2} are used in case of VCD errors only. It can be shown that all VCD errors are detectable in the trellis. Nevertheless, with only three states it cannot decide among the three flits which one it must correct. If the Parity error bits give this information, the TRDIC Decoder can correct VCDs that do not arrive in bursts (for example three VCDs in three consecutive data flits).

For a three-stage trellis, two extra flits are required at the end of the packet to permit the decoding of the last flit. In this work, just one of these extra flits is sent, containing the size of the packet. This helps the decoder to check if the packet has the correct size and reduces the flit overhead.

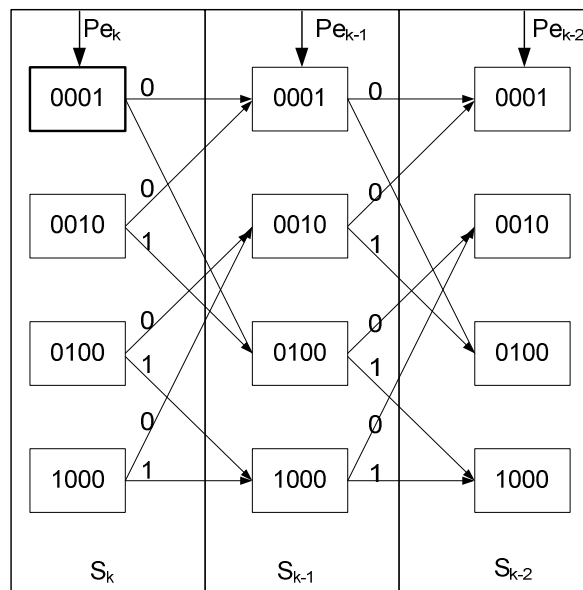


Figure 59 - Trellis TRDIC encoding based on the 1-of-4 DI encoding.

Since the trellis uses the parity scheme in the encoding, the data correction is configurable. This means that the error correction mechanism to employ in the packet encoding/decoding can be defined at run time. During packet transmission, information about which encoding is employed in

the packet is sent in the first flit (using the double check technique explained in the next Section). The payload is then decoded with the corresponding error detection/correction. The robustness evaluation of this dynamic encoder/decoder is an ongoing work. The results about this technique will be presented at the Thesis defense. In the Parity Mode, the Data Rate can be sent at maximum rate using each 1-of-4 codeword to transfer two bits. In the TRDIC mode, the data error correction/detection capability is increased. However, the data rate is half the data rate in the parity mode. This happens because in this mode each 1-of-4 codeword contains a single bit.

5.5 Packet Protocol Hardening

In the same way as for the handshake protocol, the packet protocol sequence can also suffer with SEEs. Two sideband signals (BOP and EOP) that designate the flit type control the packet protocol in the Hermes-A NoC. In addition, the packet protocol depends on the routing information contained in the first flit. To increase the packet protocol robustness, this work suggests applying the double check technique to the BOP/EOP as well as to the routing decision.

5.5.1 Flit Type Double Checking - Packet Signaling (BOP, EOP)

Inside the Input and Output ports, the BOP and EOP signals are responsible for the packet control and by the flit flow, as Chapter 4 discussed. An SEE in these signals may alter the correct data path inside the NoC. An SEE in the BOP signal may, for example, propagate a regular flit to the Path Definition module and thus forward part of a packet to the wrong target. In the Output Port, a wrong indication on the EOP signal may release the arbitration mechanism. Therefore, the packet can be interrupted and not reach the correct target complete.

The suggestion is to use double check in the flit type in each demux component controlled by the BOP/EOP. The flit type is duplicated as depicted in Figure 60. The demux decision is extracted from the double check and the two 1-of-4 codewords related to the flit decision are then restored. In this way, if an SEE occurs, the double check circuit filters it.

First Flit - Routing Information	Flit Type = "0010"	Flit Type = "0010"
Data Flit	Flit Type = "0001"	Flit Type = "0001"
Data Flit	Flit Type = "0001"	Flit Type = "0001"
Last Flit - Data Flit	Flit Type = "0100"	Flit Type = "0100"

Figure 60 – Four-flit packet with double Flit Type signaling.

Figure 61 shows an example of demux controlled by a Double Checked Flit BOP Decision. In this Figure, the second wire indicates the BOP. The circuit is equivalent to an IF (BOP == true).

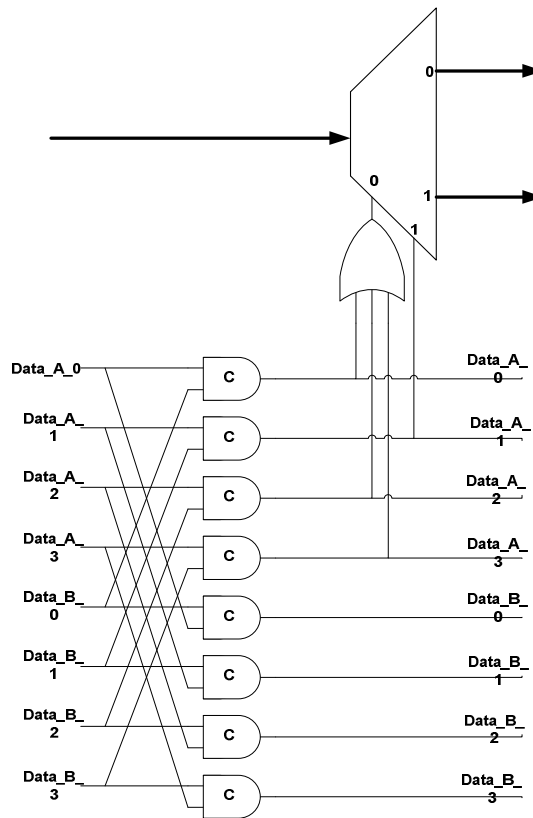


Figure 61 – Double check BOP demux control.

5.5.2 Double Check in Routing Decision

The routing decision is part of the first flit in each packet. It controls the path that the packet must cross inside the NoC. In this way, an SEE that changes the routing decision may forward the packet to the wrong target. In addition, the first flit of each packet is a critical flit, since dormant faults that accumulate during the idle state of the NoC resources may alter mainly the first flit. The hardening of the routing decision was done with the double check technique also. Double check applies to the source routing mechanism. In source routing, each 1-of-4 encoding carries one routing

decision. The duplication of each routing decision may reduce the number of lost packets inside the NoC. The Hermes-A Input Port takes the most significant 1-of-4 encoding from the flit as the routing decision and rotates the rest of the flit information. Only the Flit Type, at the less significant 1-of-4 encoding, is left untouched by the Path Calculation module.

In double check routing, the routing decision is duplicated in the first flit. In this way, in the H2A NoC, each Input Port takes the two most significant 1-of-4 codewords and performs a double check over them. The result is used to control the Output Multiplexer (Figure 41) of the Input Port and is kept in the routing register, composed by a ring of buffers. The routing register is also duplicated to increase packet robustness.

6. NoC COMPONENT LIBRARY

The commercial CAD tools and standards for logical synthesis, timing analysis and place and route are designed to work with synchronous circuits. There are a few asynchronous synthesis tools available and unfortunately, these tools are not mature enough to support the whole design flow for asynchronous circuits. In addition, since asynchronous circuits are not common among digital design teams, the use of asynchronous logic is not commonly employed in digital circuit design even when it can generate significant advantages such as reduction in the switching activity. In order to make the use of the proposed NoCs less complex, a NoC component library was created. The NoC component library enable the hierarchical design and reduces the design timing and design complexity.

6.1 NoC Components Synthesis

The Hermes-A and H₂A NoCs are composed by basic components: Input Ports, Output Ports and FIFOs. Each component can differ in function of the DI encoding, robustness technique applied, flit size and routing algorithm. In the NoC Component Library, the same component can also have different performance and power implementations in function of the constraints applied during the individual component synthesis.

The individual components are synthesized using the *pseudo-synchronous* synthesis method proposed by [90]. In this method, the QDI asynchronous circuits are synthesized using commercial synthesis tools. To do this, the C-elements that are part of asynchronous buffers in the design, are modeled as edged triggered flip-flops. The reset plays a role of clock signal in this method in order to make the synthesis tool treat the circuit as synchronous.

To implement the pseudo synchronous method, the function description of the C-element in the *liberty* files [63] was changed to a complex flip-flop. This enables the tool to swap the C-elements based on timing constraints and in the design rules such as maximum output load/fanout for the output pins and the maximum input transition for the input pins. Another advantage of this method is that it avoids the feedback loop breakers that are inserted by the synthesis tools.

Synthesis tools and static timing analysis tools are not capable to calculate the timing inside a combinational logic if there are loops present in the logic. To solve this, the synthesis tools insert a

buffer in the loop and remove the timing arc from the input to the output of the buffer. Doing this, the feedback is removed in the timing analysis point of view. The asynchronous pipeline in Figure 62 has two loops between the buffers and the completion detectors. If a buffer is inserted in this loop, the circuit will continuous to operate correctly. However, the buffer insert delay reducing the circuit performance and increasing area and power.

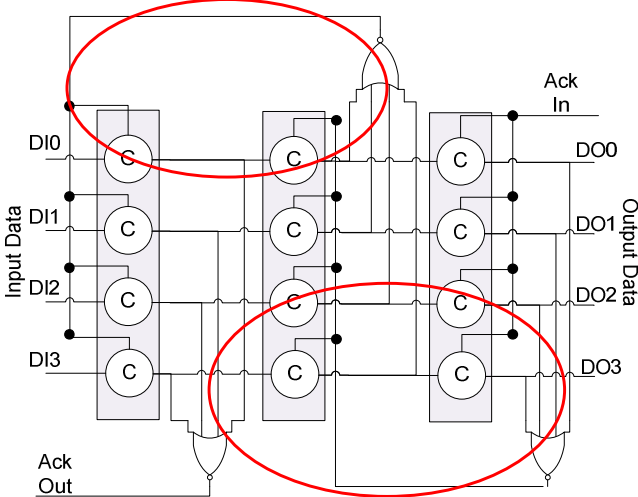


Figure 62 - One of four FIFO with a C-element victim of a SEE.

In [90] an efficient way to synthesize asynchronous QDI circuits using commercial synthesis tool is presented. This method take the full advantage of the commercial synthesis tools but it is not trivial to be implemented. Thus, the proposed NoC component library helps the designers to take the full advantages of this synthesis method by providing a broad range of components for NoC design.

Figure 63 shows the adopted synthesis flow. The Logical Synthesis was performed using the RTL Compiler tool from Cadence. The synthesis inputs are: the pseudo synchronous constraint file, the pseudo synchronous liberty file, the .lef file [20], the Capacitance File and the RTL description of the cells. The outputs of the place and route step enable the construction of hierarchical designs where the asynchronous components can be instantiated as small tiles. The use of the router ports as basic building blocks, enable a more optimized floorplan i.e. the NoC ports can be manipulated to reduce the wire lengths. Asynchronous FIFOs, as presented in Figure 62 can be used between NoC ports to break the wires without reduce the throughput.

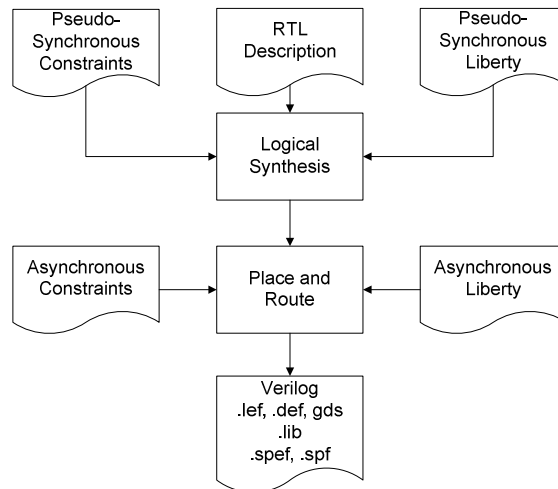


Figure 63 – Synthesis flow adopted for the asynchronous components that compose the NoC library.

6.1.1 Noc Component Description

After the place and route of the NoC components, is performed the characterization of the NoC components. The characterization extracts the timing, power and robustness levels of each NoC component. In this work, the robustness characterization is focused. After the characterization, each component has a set of attributes that describes the component behavior. These attributes can be divided as not operating condition dependent and operating condition dependent. The non operating condition dependent are:

- Flit size;
- Area;
- Routing type (source routing, distributed XY [69], distributed adaptive routing WF [70]).

The operating condition dependent attributes are:

- Throughput;
- Latency;
- Energy per flit;
- Static power;
- Robustness level.

The timing and power attributes are obtained by simulation and the characterization process is not described in this work. The robustness characterization is a contribution of this work and it is described in the next Section.

6.2 Robustness Characterization

There are several SEE models available in the literature. These models cover different abstraction levels [51]. A first abstraction level consists in measuring the injection of charge in silicon structures using laser sources, protons or ion radiation [34][32]. At this level, accurate values for collected charge and current shape are extracted. The method is well suited to characterize fabrication technology processes, memory elements or small test circuits. The characterization of a complex digital circuit at this level is unfeasible, since it must include the measurement of every node at different circuit states and PVT conditions. At the same level of abstraction are the models obtained using physical simulators. These simulators model the (physical) interactions between radiation and materials. Clearly, the same restrictions found at the physical measurement level apply to physical simulators to obtain system level validation for SEEs. Nonetheless, measurements obtained at these two levels are extremely useful as input parameters for the next abstraction level: electrical simulations.

Electrical simulations are faster if compared to their physical counterparts. At this level, however, the silicon irradiation process is not directly modeled. The process is usually captured using an exponential current source as the Equation (13):

$$(13) \quad I(t) = I_0(e^{-t/\tau\alpha} - e^{-t/\tau\beta})$$

Parameters $\tau\alpha$ and $\tau\beta$ of the exponential source are provided at the physical modeling level, either through measurements or simulations. They represent the behavior of the silicon charge collection and are highly dependent on the technology. Through electrical simulation it is possible to extract the behavior of a circuit when submitted to a specific exponential current curve. Two measurements are used to characterize a digital circuit when facing a charge impact: the critical charge and the output pulse width. Critical charge is the minimum amount of charge necessary to create an event at the output of a digital circuit. The pulse width can be just the size of the pulse or in some works its shape as well [76][93]. Both, critical charge and pulse width are dependent on the circuit state and on the node in which the exponential source is applied. Even if it is faster when compared to physical abstraction levels, electrical simulations are not adequate for analyzing the overall SEE behavior in complex digital systems. To be able to tackle the SEE in complex digital systems it is fundamental to treat such events at higher abstraction levels. This next abstraction is the digital design level, which models SEEs as digital pulses [8]. Measurements obtained here relate to the Single Event Rate. The SER in digital circuits can be statistically modeled by Equation (14)[5][51].

$$(14) \quad SER(n_i) = P_{SEE}(n_i) \times P_{sensitized}(n_i) \times P_{latched}(n_i)$$

Here:

- P_{SEE} is the rate at which SEEs occur in a circuit node n_i with sufficient strength to propagate to a memory element;
- $P_{sensitized}$ is the probability that a functionally sensitized path from node n_i to a memory element exist;
- $P_{latched}$ is the probability that the SEE be captured by a memory element.

Equation (14) represents the relationship among the cell electrical filtering property (P_{SEE}), the circuit logical filtering property ($P_{sensitized}$) and the memory elements' sampling filtering property ($P_{latched}$). The logical filtering and sampling window are easily verified using a digital simulator. Nevertheless, the electrical filtering property of SEEs in standard cells is not present in a digital simulation flow. In this way, the simple insertion of digital pulses is not enough to determine circuit robustness, since a pulse may have different meanings, depending on the node where the pulse appears and on the circuit state. Another way to estimate the SER is through analytical methods [5][62][53] [99]. In an analytical method it is possible to determine if a digital pulse produced in a specific node can reach the input of a register and consequently create an SEU. Analytical models are adequate for small size combinational circuits and in this way enable estimating the $P_{sensitized}$ part of Equation (14), but the time required to evaluate a complex circuit increases exponentially with circuit complexity [5]. The sequential behavior of a circuit is difficult to predict using this method, which makes it not suitable for predicting the SER in complex digital systems. In this sense, analytical models fail to determine the real failure rate at the boundary of a System on a Chip or of an IP Core. However, the precise definition of the system failure rate is a necessary measure to a complete SEE modeling.

6.2.1 SEE Digital Design Flow

The previous Section showed that to address SEE-resistant SoC design it is mandatory to provide an accurate modeling of SEE faults at the digital level. This modeling has to be as accurate as possible compared to the electrical simulation level. With this goal, this work proposes the design flow of Figure 64. The flow is mostly based on existing tools and standard formats, for providing wide usability. Figure 64 comprises five columns, where the first, second and fifth columns present some new design steps to provide an SEE-aware design. The second and fourth columns represent a simplified conventional SoC design flow.

The SEE-aware flow was conceived to fit in a conventional digital design flow. The first step is the SEE Characterization, represented in the first column of Figure 64. This step performs SEE susceptibility and SEE timing measurements in the (standard) cell library. Characterization employs exponential current sources with parameters that extractable from measurement or simulations at the physical level. These parameters are represented as the SEE Technology Model in Figure 64. From the SEE Characterization, an encompassing set of electrical characteristics are extracted from electrical simulations, to form the SEE Database, including:

- Voltage level before SEE current insertion;
- Maximum Voltage reached at the victim node;
- Minimum Voltage reached at the victim node;
- SEE critical charge;
- Output pulse width.

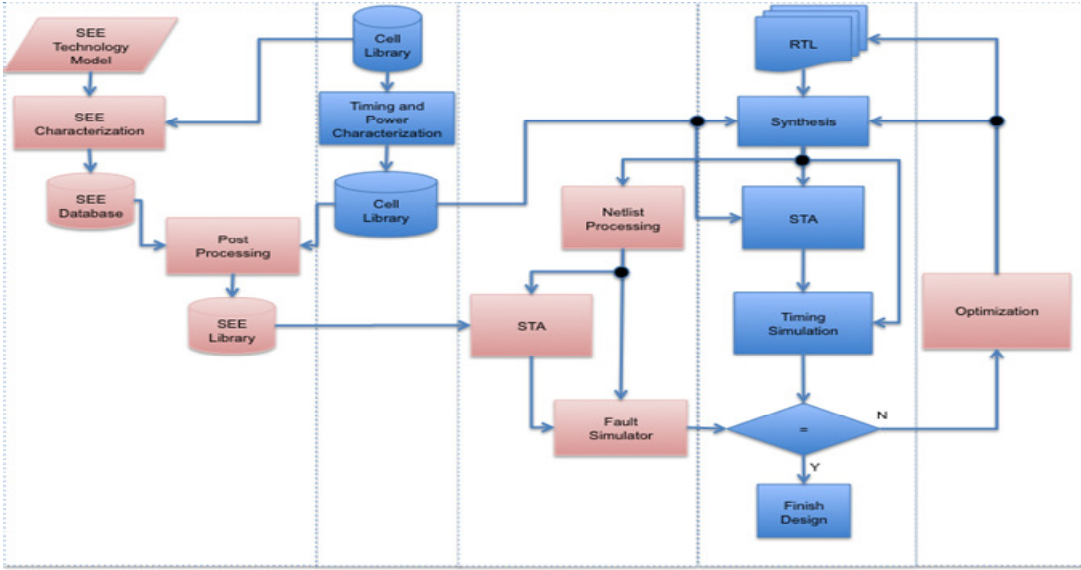


Figure 64 - Proposed flow for digital circuit design based on SEE constraints.

Based on the result of the SEE characterization, the Liberty standard [63], used for timing and power description of the cells is adapted to hold SEE information about critical charge and pulse width in the Post Processing step. Adaptation of Liberty is achieved applying a transformation from critical charge to timing. Transformations permit a digital simulator to perform the electrical and logic filtering processes. In this way, the proposed may cover all the components of Equation (14) and thus provide accurate results related to the SER of a circuit.

To model the cells' SEE behavior, a new input pin is created. This pin is necessary to capture the effects of the SEE on the output pin of the cells and is added to each library cell. The new pin

allows modeling both, the SEE behavior in cells and their timing properties. Figure 65 shows an example modeling scheme (for a 2-input NAND gate). The cell keeps the original behavior when its SEE pin is not asserted. In the presence of an SEE (SEE = '1'), the cell has its behavior changed, due to the bit flip caused by a radiation particle impact, for example. This simple change in the cell description enables the implementation of a flow capable to support SEE modeling in digital standard cells.

Using this modification is the first step to create a verification environment (third column of Figure 64) for SEE where events can be created with controlled rates and duration in a Fault Simulator.

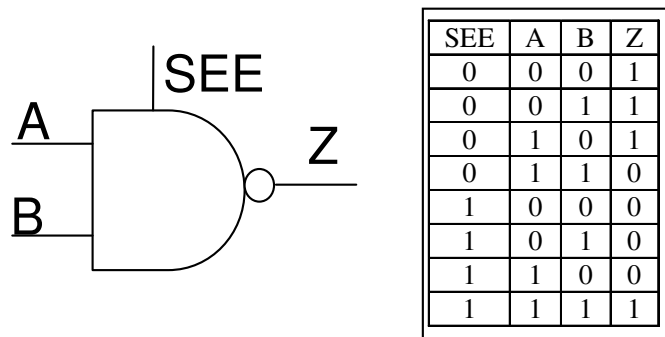


Figure 65 - Example of a modified NAND gate for SEE simulation.

Digital simulators implement timing filtering processes for glitches using a pulse control mechanism. In the default mode, a digital simulator exhibits the inertial delay mode for simulating the cell behavior in a digital circuit. In the inertial delay mode, any glitch in a cell is filtered if the glitch size is smaller than the propagation delay of the cell. In order to take advantage of this simulator property, a conversion from critical charge into delays is performed in the Post Processing step of the flow of Figure 1. In this way, the simulator is capable to filter SEEs indications with less charge than required for a cell in a given state. The charge to timing conversion uses Equations (15)(16) and (11):

$$(15) \quad \text{SEE_Charge} = \int_{t_1}^{t_2} I_0 (e^{-t/\tau\alpha} - e^{-t/\tau\beta})$$

$$(16) \quad \text{SEE_Modeling_Charge} = \int_0^{t_x} I(t)$$

In Equation (11), I(t) is an equivalent constant current necessary to create the critical charge in a time interval defined as tx. The equivalent time (tx) is extracted from the relationship between charge and timing. Thus, tx is the value of the pulse width (measured in seconds) generated by a charge of one Coulomb. This charge to time transformation is applied as a characterization post

processing phase, before entering the SEE effect within the timing library.

Finally, to analyze a complete system level design using this fault modeling at cell level, a static timing analysis tool (STA) generates all values of SEE timing. The timing of the cell is kept and a new timing arc is added from the SEE pin to the cell output. This process can be applied to either combinational and sequential cells with single or multiple outputs.

Characterization values are applied to instances in the netlist, including the timing arcs of the new input pin SEE to the cell output. Based on these values, it is possible to perform an SEE timing simulation on the design. This step is performed using a fault simulator. The fault simulator is responsible for generating the SEE indication in the design instances. SEE injection can be fully controlled by the fault simulator. It can control both, SEE injection rate and SEE injection charge. The fault simulator can verify if the desired levels of robustness are reached in a specific design implementation. Estimations generated by the fault simulator can be used to implement design optimizations.

6.2.2 SEE Flow Implementation

All current standard cell digital design flows work based on some estimation of cell electrical properties (timing, power). The SEE characterization is based on the same principle: extraction of the SEE susceptibility from electrical simulations under distinct PVT conditions. Cell characterization is achieved through electrical simulation. Since every cell state has a different level of susceptibility, characterization is conducted for all steady states of a cell. For each steady state, just the nodes that have a reversed bias junction are considered. The transient states, those where one output of the cell is switching, are ignored, since the voltage in switching junctions is smaller than the supply voltage. Thus, they do not represent an SEE-worst case scenario. Figure 66 depicts the characterization method. The process referred as FindCriticalCharge() basically performs a binary search for the minimum charge of each susceptible node in a state.

```
SEE Characterization()  
  for each cell in the library  
    for each cell state  
      Nodes = FindSusceptibleNodes(cell)  
      Charge(cell,state) = FindCriticalCharge(Nodes)  
    end
```

Figure 66 - Method for SEE characterization of standard cells.

Figure 67 shows an example SEE characterization. In this example, it is possible to compare the critical charge and pulse width for a 2-input NAND gate with two different drive strengths when driving a 5pF output load. For State=101, the critical node is the internal node IN1, while for the other states the critical node is the output pin. The values for critical charge were normalized based on the most robust combinational cell in the library.

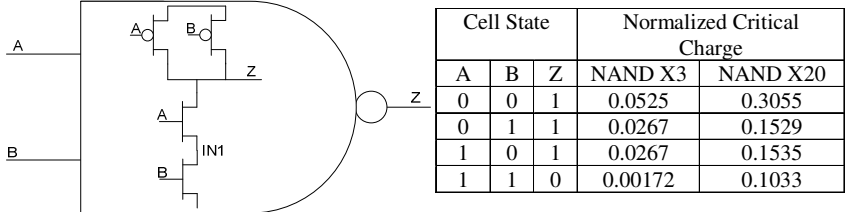


Figure 67 - Characterization values of a 2-input NAND gate.

Fault Simulator

Fault simulation is performed using digital simulation with timing annotation. The principle of the SEE simulation is based on inertial delay properties. Figure 68 shows the adapted verification environment for SEE evaluation. SEE verification reuses the verification environment and test cases generated for the system behavior and timing verification. A fault generator module is introduced to produce the SEE indication in each instance of the design. The environment was designed using SystemC code. The process to select the victim instance is based on a random number generator, which picks a number in the interval between 0 and n, where n is the number of instances in the design. The fault generator is able to create single and multiple events. The event rate insertion can be easily controlled and allows generating events anywhere within a clock cycle.

To permit SEE generation and propagation, the SEE to Output pin behavioral and timing arc is created. To permit simulation of the new behavior as well as SEE and timing annotation, the changes are performed in the cell Verilog model.

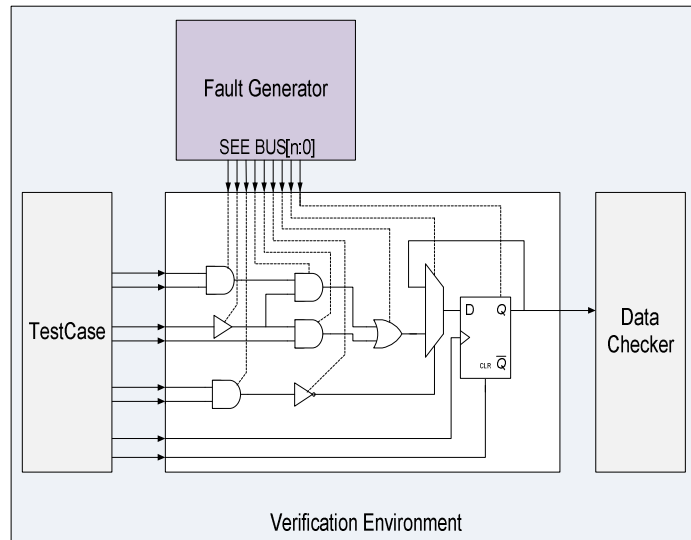


Figure 68 - Fault simulation environment.

The left side of Figure 69 shows the *Verilog udp* model generated to implement the bit flip at the output of a digital cell. In this *udp* model, when the SEE input pin is equal to logic level '1', then the cell has its behavior altered. At the right side of Figure 69 the *SEE udp* is instantiated in a 2-input NAND gate Verilog model. In the cell Verilog model the timing arc is added between the input pin SEE and the output cell pin. This timing description is state dependent, to allow a better pulse description.

<pre>primitive u_seu (Z, A, SEE); output Z; input A, SEE; table // A SEE : Z 0 0 : 0; 1 0 : 1; 0 1 : 1; 1 1 : 0; endtable endprimitive</pre>	<pre>module NAND2 (Z, A, B, SEE); output Z; input A; input B; input SEE; and U1 (INTERNAL1, A, B); not U2 (INTERNAL2, INTERNAL1); u_seu (Z, INTERNAL2, SEE); specify if (B) (A ==> Z) = (0.01,0.01); if (A) (B ==> Z) = (0.01,0.01); if (!A && !B) (SEE ==> Z) = (0.01,0.01); if (!A && B) (SEE ==> Z) = (0.01,0.01); if (A && !B) (SEE ==> Z) = (0.01,0.01); if (A && B) (SEE ==> Z) = (0.01,0.01); endspecify endmodule</pre>
--	---

Figure 69 - SEE adapted Verilog example for a 2-input NAND gate.

The most accepted timing description standard for digital cells is the Liberty standard [63]. SEE susceptibility, after transformed in timing values, is modeled inside the Liberty format using the Non Linear Delay Model (NLDM). Using NLDM tables it is possible to represent the state dependence of the cell susceptibility as a function of the output load capacitance of the cell. Figure 70

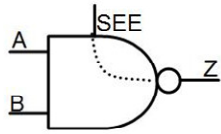
shows a 2-input AND Liberty file on the left side and a corresponding Standard Delay Format (sdf) on the right side. The values in the Liberty files are used to guide the synthesis tools when searching the best cells to implement a given function, as well as to annotate the cell timing properties. In this way, values inside the Liberty file are used to overwrite the default values of the VHDL or Verilog models for timing simulation. This is usually done in two steps: generation of the descriptor of the cell delay file and the behavioral and timing model association.

<pre> cell(NAND2){ pin(SEE){ capacitance : 0; direction : input; } pin(Z){ timing(){ related_pin : "SEE"; when : "A' * B"; sdf_cond : "!A && B"; timing_sense : negative_unate; cell_fall(SEU_table_9){ values(" 1.10503 1.10565 ... "); } cell_rise(SEU_table_9){ values(" 0.00775537 0.0003982 ..."); } fall_transition(table_10){ values("0"); } rise_transition(table_10){ values("0"); } } } } </pre>	<pre> (CELL (CELLS "NAND2") (INSTANCE inst_multtruegte_70_26trueg12644) (DELAY (ABSOLUTE (COND A&&!B (IOPATH SEE Z (0.005::0.010) (1.112::1.113))) (COND !A&&B (IOPATH SEE Z (0.005::0.004) (1.107::1.107))) (COND !A&&!B (IOPATH SEE Z (0.005::0.007) (2.214::2.215))) (COND A&&B (IOPATH SEE Z (0.751::0.751) (0.014::0.010))) (COND B (IOPATH A Z (0.014::0.016) (0.019::0.022))) (COND A (IOPATH B Z (0.016::0.020) (0.020::0.025)))))) </pre>
--	---

Figure 70 - Example of a timing library fragment for a 2-input NAND gate with timing arcs from SEE pin to output Z.

Figure 71 shows the waveform of an SEE plus timing simulation of a 2-input NAND gate. It displays the generation of three different SEEs, each one with a different value of pulse width and thus with a different charge value. The first SEE has the same width as the critical charge, and an equivalent pulse is generated at the output of the cell. The second SEE width does not convey enough charge to create an event on the output of the cell. Thus, it is filtered. The third SEE has a pulse width capable to generate an event at the cell output.

Cell Critical Charge → Timing Modeling
 SEE↑ → Z↑ = 80ps (positive unate)
 SEE↑ → Z↓ = 60ps (negative unate)



Cell SEE Pulse Width Modeling
 SEE↓ → Z↑ = 6ps (negative unate)
 SEE↓ → Z↓ = 8ps (positive unate)

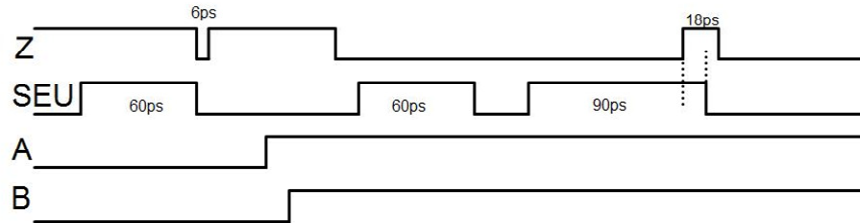


Figure 71 - Waveform of an SEE example simulation for a 2-input NAND.

7. SYNTHESIS OF HARDENED ASYNCHRONOUS NETWORKS ON CHIP

Synthesis of asynchronous circuits is a difficult task, even more when the robustness becomes the critical constraint of the design. In addition, for asynchronous NoC synthesis one important point is leveraging the performance constraints from the traditional timing specification to the communication level in order to help the match of these constraints in earlier design steps. In order to automate the NoC generation process taking in account robustness and communication constraints, this work proposes the PARANA NoC synthesis tool. The PARANA tool uses the components of library proposed in the previous Chapter as building blocks to build NoCs tailored for specific applications.

7.1 Communication Model for Hardened Applications

In the traditional design of digital circuits, the circuit must obey a set of physical and electrical rules. The circuit synthesis, for example, is driven by timing constraints such as maximum frequency and other area and power constraints. The design of complex SoCs requires the system modeling at higher level of abstraction in order to evaluate the architectural aspects in early stages of the SoC design [64]. To enable this, new methods for system modeling and new types of constraints are required. For SoCs based on several applications, the communication can play an important role in the overall system performance. In this way, the communication modeling between the applications is essential to improve the application mapping inside the SoC [46], the floorplan, and the NoC synthesis and customization [81] [57].

Hu and Marculescu [37] proposed the Application Characterization Graph (APCG) to describe applications and perform the task mapping. A more generic model, called Communication Weighted Model (CWM), was proposed by Marcon [46] to model the interaction between applications in terms of weights. This model is part of the CAFES mapping tool. The CAFES tool performs task mapping inside a SoC based on a NoC.

The CWM model employs Communication Weighted Graph (CWG) for each communication between IP Cores. A CWG is a directed graph represented by $CWG = \langle N, W \rangle$, where $N = \{n_1, n_2, \dots, n_c\}$ is the set of vertices representing the IP Cores and $W = \{W_{12}, W_{13}, \dots\}$ is the set edges that

represent the communication weight between two IP Cores. Since the weight is a unit-less value, it can be used to model different types of communication constraints. Figure 72 shows an example of a generic communication between four different IP Cores described by a CWG.

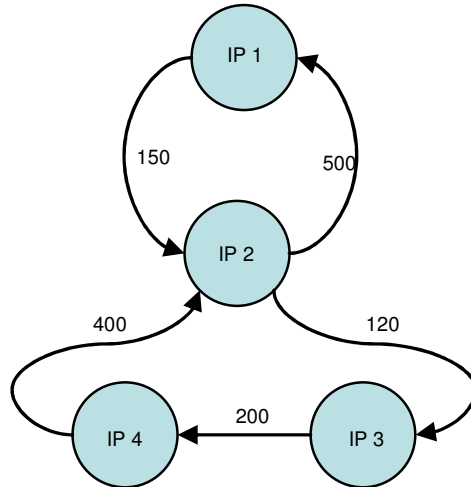


Figure 72 - Communication between IP Cores described using a CWG.

A SoC aimed to PARANA tool is a GALS SoC composed by n IP Cores interconnected by n routers. The IP Cores can operate under different clock domains. In addition, it is assumed that these IP Cores can employ techniques for the Soft Error mitigation. For example, two IP Cores inside the SoC that shared messages through the NoC can employ the TMR technique. In order to guarantee that the whole application is hardened, the communication between hardened IP Cores must be hardened also. In this way, a new model capable to model hardened and non-hardened GALS applications communication is required. In this work, we propose the use of a Hardened GALS Communication Model (HGCM).

The HGCM employs the Hardened GALS Communication Graph (HGCG) to describe the applications. The HGCG is a directed graph represented by $HGCG = \langle N, W \rangle$, where $N = \{n_1, n_2, \dots, n_c\}$ is the set of vertices representing the IP Cores and $W = \{W_{12}, W_{13}, \dots\}$ is the set of edges. Each edge in the HGCG represents a Hardened GALS Communication (HGC). The Hardened GALS Communication is represented by a 5-tuple $HGC = \langle pr, sfr, rfr, ps, hl \rangle$ containing:

- Packet Rate (pr): Average Number of transmitted packets that are transmitted between the source and target IP Cores inside the time unity. The time unity can be defined by the user. The default value is 1ms.
- Sender Flit Rate (sfr): Flow rate in millions of flits that the transmitter inserts the NoC in one second. In an IP Core that inject one flit per clock, the flow rate is equal to the clock frequency divided by 10^6 .

- Receiver Flit Rate (*rfr*): Number of flits (in millions) that the receiver can sample in one second.
- Average Packet Size (*ps*): Average number of flits in the packets.
- Hardening Level (*hl*): Required hardening level for the specific communication. The hardening level in this work can assume four different values: 1 = No hardening technique is required; 2 = Data error correction/detection is required; 3 = Hardened Handshake/Packet protocol is required; 4 = Data error correction/detection and Hardened Handshake/Packet protocol are required.

Based on the parameters of each edge of the HGCG, the NoC synthesis is performed inside the PARANA tool. The hardening/ performance requirements guide the tool to generate a NoC where each component (Input/output Port) satisfies all constraints of all the communications that cross the components.

7.2 Layered NoC Architecture

The intrachip communication architecture generated by the PARANA tool is a set of NoCs separated in different layers, that is, each layer is an independent NoC. All Input and Output Ports in the same layer implement a set of hardening techniques that are suitable to that layer. Figure 73 shows a NoC composed by all the four possible layers where each layer is an independent 3x3 mesh NoC.

In this work, four different hardened layers are allowed.

1. The first layer has no hardening technique and no data correction applied to it.
2. The second layer has data correction scheme applied to flits. This means that a Data Encoder/Decoder is used at the IP Cores. The NoC however, has no hardening technique.
3. The third layer has hardening applied to NoC. This means that the control signals of the NoC are protected in order to reduce errors in the packet transmission and the stall probability. This layer has no data correction and it is recommended for data communication where the data correction is already applied in another level other than the flit transmission.
4. The fourth layer has data correction and hardening techniques applied to the NoC. In this way, it can increase the availability of the NoC and provide data correction.

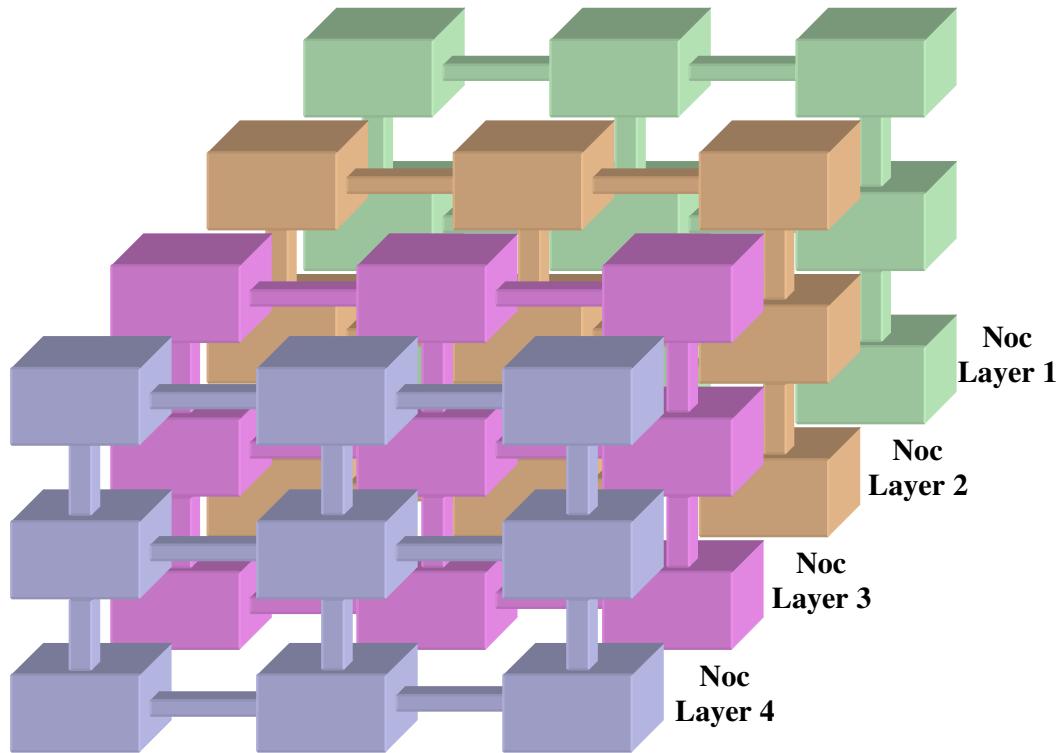


Figure 73 – NoC containing four layers. Each layer is an independent 3x3 mesh NoC.

In the wormhole, a single packet can be distributed through several Input/output Ports inside the NoC. The consequence is that a stall in a component is propagated to other routers in the NoC. In this way, a stall in a non-hardening component of a wormhole NoC can propagate through the NoC causing starvation or the complete NoC stall after some time. In these cases, the only solution is the reset of the full NoC.

The main goal of the layered communication architecture is to isolate the communication of hardened applications from non-hardened applications and enable separated reset mechanisms for each layer. The layering isolation ensures that a reset in the non-hardened layer do not disturbs the hardened components. The consequence is the increase in the hardened components availability.

7.3 Hardened NoC Synthesis

The PARANA generates heterogeneous layered NoCs based on traffic description. The traffic description states the characteristics and constraints of communication between IP Cores. Based on traffic description, the synthesis process build the paths between IPs in such a way that the components that belong to this path are capable to fulfill the communication requirements. Figure 74 shows the PARANA synthesis process.

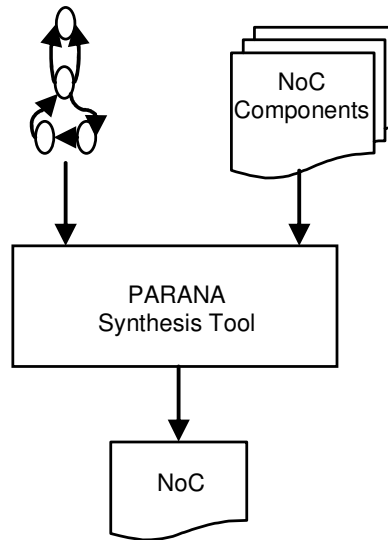


Figure 74 – PARANA NoC synthesis flow.

The NoC synthesis is divided into the following steps:

- Traffic Distribution and Component Constraint Definition;
- Idle Channel Cut-off
- Component Selection;
- NoC Generation.

Traffic Distribution and Component Constraint Definition

The first step during the synthesis is the traffic distribution over Input and Output ports. In this step, each packet is routed through the NoC model inside the tool. Every Port reached by the packet updates the information about the maximum required throughput and about the resource occupation. The resource occupation is used to define when the NOAP technique is employed. This is done because the NOAP technique presents better results in components with long idle times.

The packets are distributed differently to the Input Ports and Output Ports. The Input Port is modeled as a single data source with four outputs sinks. Each output sink correspond to an *Internal Router Link* between the Input Port and successive Output Ports. When a deterministic algorithm is evaluated, the Data Rate that arrives the Input Port is assigned to the sink selected by the routing algorithm. In this way, the relation between the Input Traffic and the Output Traffic in the Input Port is modeled as showed in Figure 75, where λ, μ, θ and ρ are the probabilities related that a packet at the input of the Input Port be forwarded to the associated output sink. The sum of the probabilities $s = \lambda + \mu + \theta + \rho = 1$.

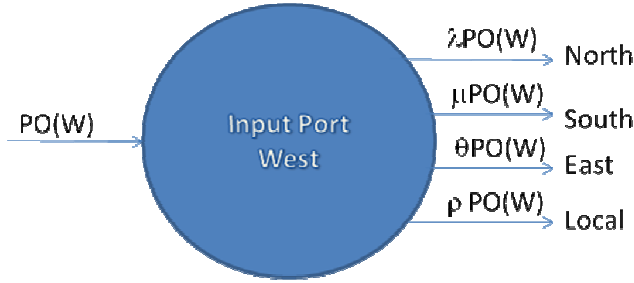


Figure 75 – Model used to represent the Input Port inside the PARANA tool. In the illustration, one incoming traffic is split in two different outputs due to the dynamic algorithm property.

The Output Port is modeled by a component with four different Data Sources and a single Data Sink. The Output Port occupation here is the sum of the source port occupations. Since the output port feeds exactly one Input Port/NI in the next router/IP Core the calculated Output Port occupation is always equal to the connected Input Port. In this way, Port Occupation measurements done at the Output Port (using the Output Port State signal from Figure 42) can be used to the Input Port as well.

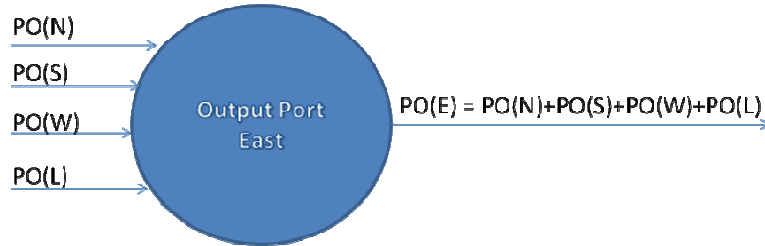


Figure 76 – Model used to represent the output port. The output traffic is the sum of the traffic at the four inputs.

The Port Occupation of a component is calculated inside the tool by the following equation:

$$(17) \quad PO(x) = \phi * \sum_{i=1}^j (\min(sfr, rfr) * 10^6 * ps)$$

Here, ϕ is an occupation adjustment variable. This variable can be determined during the NoC simulation by measuring the occupation rate at the Port State signal in the Output Ports. Based on this value, the tool can readjust the Port Occupation calculation and performs an incremental synthesis. The Port Occupation is compared to a threshold value, defined by the user, to verify that traffic is within the design constraints. Since the NOAP technique has better results in components with long idle time, the use of the NOAP technique depends on the port occupation.

An interesting point to be noted is that the Port Occupation (PO) at internal router links is always smaller or equal to the port occupation of the Input Ports link. This can be a guide to perform the NoC floorplan. By positioning the components in such a way that the external links are smaller than the internal ones can reduce the switching power in the NoC since the more active links are re-

duced.

Channel Cut-off

To avoid the area overhead due to NoC layering, the synthesis method proposed in this work uses a method to cut off the resources that are not used by the traffic description. For example, Figure 77 shows an illustrative example of a router generated by the PARANA tool. In this Figure, the Input/output Ports of the router are selected to ensure the throughput constraint. The North Output Port and the South Input Port are not presented in the router because these ports are not used by the HGCG.

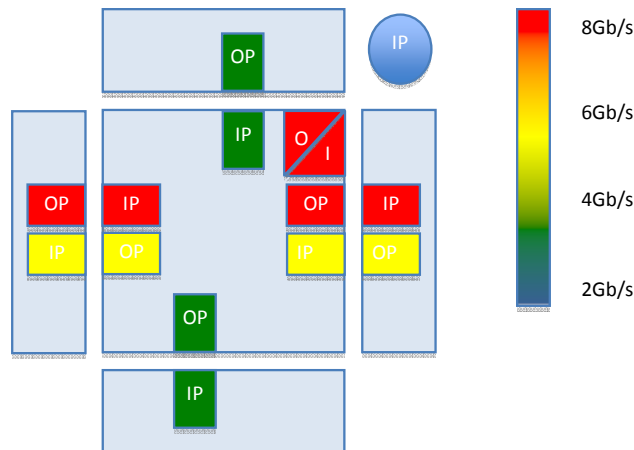


Figure 77 – Illustration of a router composed by heterogeneous ports adapted to a HGDG description.

Component Selection

Once the traffic distribution is completed, each component model inside the tool can evaluate the throughput required, the Port Occupation and the hardening level required. Based on this information, each individual Input/output Port can select from the NoC Component Library the most appropriated component to satisfy the constraints. The resource selection is done by filtering the components that do not fulfill the requirements and applying a second optimization target to perform the Sorting of the components. For example, if after removing from the list all components that not reach the desired throughput and hardening level, the component list can be organized in function of the Area, Latency or Power. By selecting the first element of the list after the sorting, the most adapted component to the input constraints is selected.

NoC Generation

Once all the resources of the NoC have been select from the NoC Component Library, the NoC

generation is performed. This process involves the generation of the netlist files and the evaluation environment. The PARANA tool generates three different netlists, (a) a netlist for design at NoC Component Macro level; (b) a netlist for the design at standard cell level and a netlist for SEE evaluation at NoC level using the method presented in [71]. Besides the netlists, the tool also generates the environments required for timing simulation (using standard Delay Format *.sdf files* [87]) and SEE evaluation. In the next versions of the tool, a SystemC modeling the power, timing and Soft Error behavior of the NoCs will be generated as well to reduce the NoC evaluation time.

In the physical design of the NoC based SoC, the designer can choose between use the standard cell level netlist to have more control during the place and route or the Macro Level netlist where the necessary Macro views (*.lef, .gds and .def* [20]) of the NoC Components are accessible as well.

8. RESULTS

This Chapter shows the experimental results achieved using the proposed hardening techniques discussed in previous Chapters. First, it presents the hardening evaluation of the NoC Components under several different scenarios. The scenarios represent different SEE conditions and distinct packet traffic conditions. This is essential to decide what techniques are more suitable to each traffic condition and to each NoC component. Results presented here show the error rate and the classification of each error in terms of DI encoding data corruption (VCD, ICD), flit corruption (BOP, EOP or Data Flit), packet corruption and data transmission corruption.

All results refer to the 65nm bulk technology. This Chapter employs only NoC components synthesized with Standard Voltage Threshold Gates. In addition, synthesis of all components employs the pseudo-synchronous method, using a 1GHz pseudo clock constraint. The SEE evaluation assumes conditions of 1 Volt, 25°C, and typical process parameters. These operating conditions apply to the timing cell description as well as to the SEE standard cell characterization.

Results are average values for multiple 10-millisecond simulations. The number of errors in the graphs corresponds to the number of errors for each 10 ms.

8.1 NoC Components Evaluation

The next Sections present the NoC components evaluation. Different implementations are evaluated: *(i)* the regular Hermes-A Output Port; *(ii)* the Normal Open Asynchronous Protocol implementation, *(iii)* the double check; *(iv)* the combined use of Normal Open Asynchronous Pipeline and Flit Type Double Check; and finally *(v)* the m-of-n error correction based on Parity. The new TRDIC is currently under evaluation. Results on the use of this technique will be available at the thesis defense.

The presented evaluations here comprises the response of the NoC components: *(i)* under SEE with different charges; *(ii)* under SEE with fixed charge and with different SEE rates, *(iii)* with fixed SEE charge and rate and *(iv)* changing the Packet Interval and Packet Size.

8.1.1 Injected Charge Evaluation

To perform the SEE injected charge evaluation in the Output Port, the traffic was inserted at one of the input channels of the Output Port while the other three channels are not used. The inserted packet rate is 10×10^6 packets/second, with all packets being 16-flit ones. The SEE injection rate used in this evaluation is equal to 10×10^6 events/second. Figure 78 shows the relation between the Number of Data Errors and the Injected Charge for four different Output Port implementations. In the graph, it is possible to note that the use of the NOAP combined with the flit type double check technique displays the best results for all levels of injected charge. All implementations present similar curves, where it is possible to distinguish a point where the increase in the injected charge does not significantly increase the error rate. This point is close to the C-element critical charge level. This means that the injected charge level has less impact in the soft error after exceeding the critical charge of the C-elements.

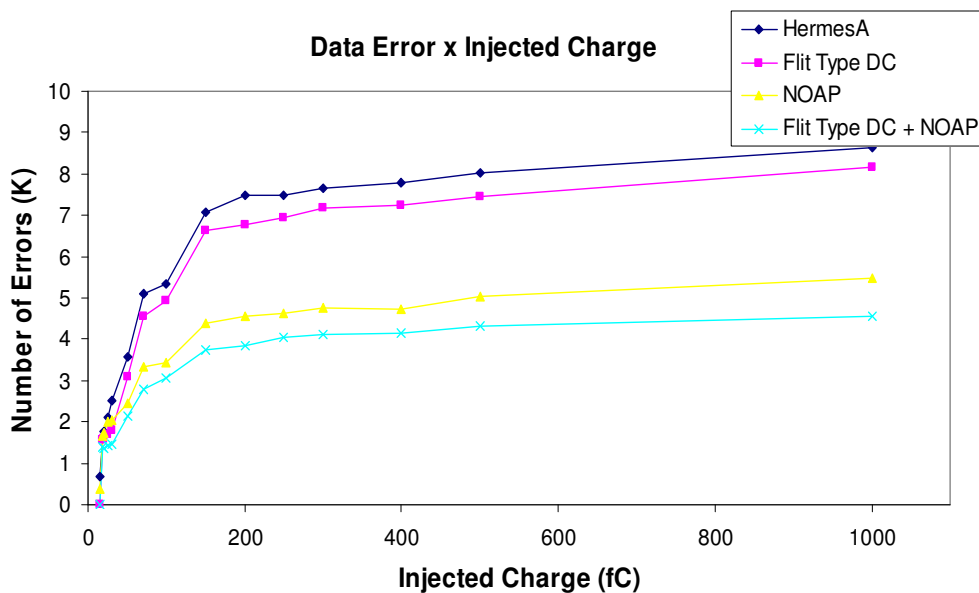


Figure 78 – Relation between the injected charge and the Soft Error Rate for different Output Ports implementations.

Figure 79 shows the evaluation of Input Port implementations. Parameters for the Input Port evaluation are the same presented before for the Output Port: packet rate= 10×10^6 packets/second; packets size=16 flits; SEE injection rate= 10×10^6 events/second. The difference here is that transmission of packets take place so that packets follow always to the same output. This means that the routing information is the same for the results presented here. This facilitates evaluation. In Figure 79 it is possible to see that the use of the Double Check at the control signals of the Input Port (data flit and routing information) combined with the NOAP implementation achieve the best results.

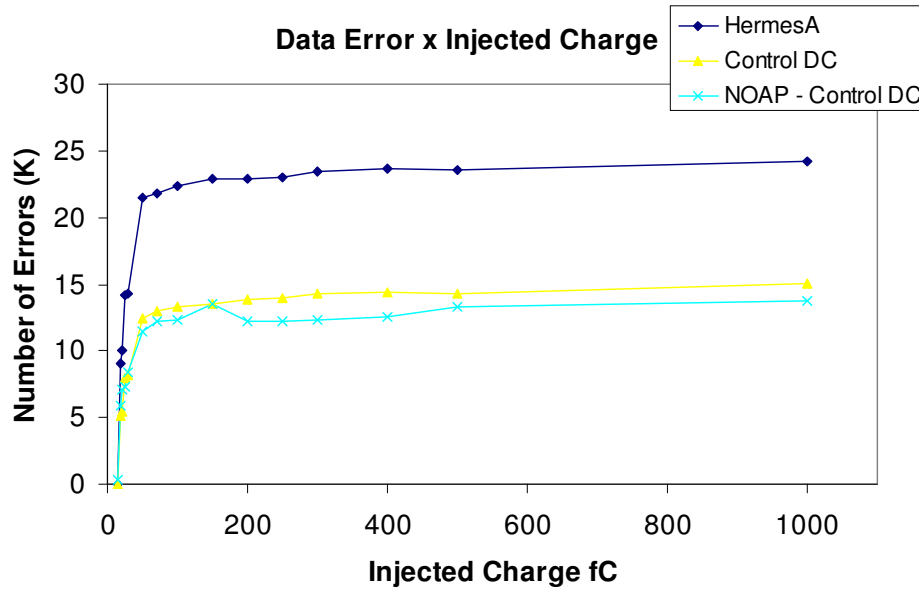


Figure 79 – Relationship between the injected charge and the Soft Error Rate for different Input Port implementations.

The benefits of the proposed NoC hardening solutions are clearer when looking to the number of errors during packet transmission. Figure 80 shows results for the Output Port while Figure 81 shows the same evaluation for the Input Port. Again, the combined use of Double Check in control and NOAP technique brought to components an increase of about 45 times of robustness to the Output Port and about 5 times to the Input Port.

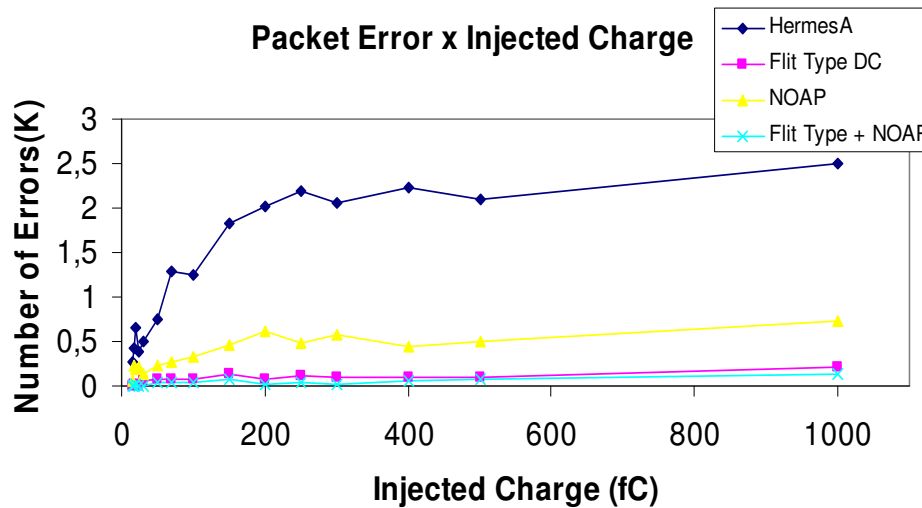


Figure 80 – Relationship between the number of packet errors and the injected charge for different Output Port implementations.

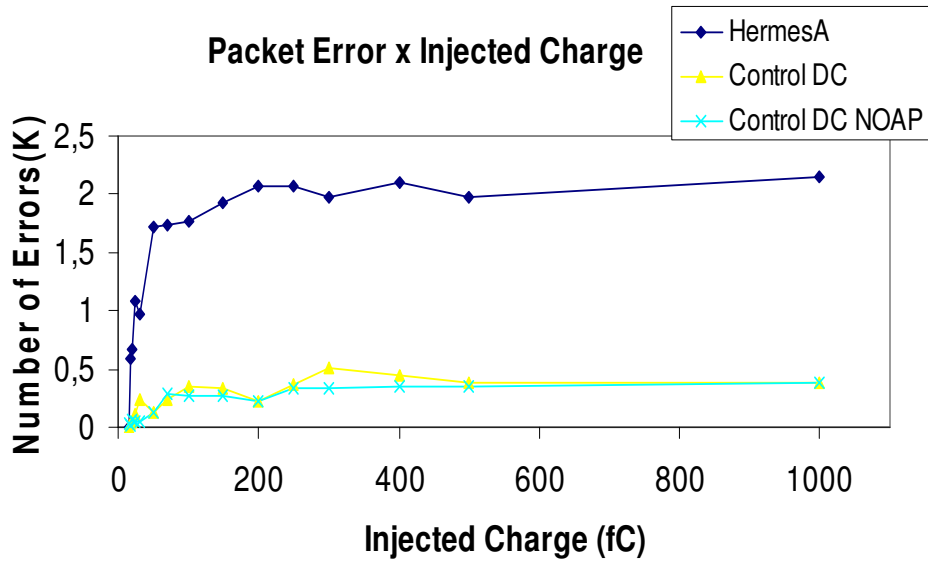


Figure 81 - Relation between the number of packet errors and the injected charge for different Input Port implementations.

The next experiments relate to the average result of the NoC Components under the incidence of SEEs with a 100 fC charge. Other variables, like packet rate and SEE rate, remain unchanged with regard to the previous Injected Charge evaluation. Figure 82 and Figure 83 show the error classification as a function of the resulting DI code corruption for the Output and Input Ports, respectively. The possible errors presented here are the ICD and the VCD. As predicted before by the timing window analysis, the most common type of error is the ICD. The VCDs result from glitches and the consequent duplication of some flits. The consequence is the misalignment between the golden and the victim designs under test. In these cases, the Data Checker restarts the design after a burst of errors is detected (five errors in sequence were used to obtain the results presented in this Chapter).

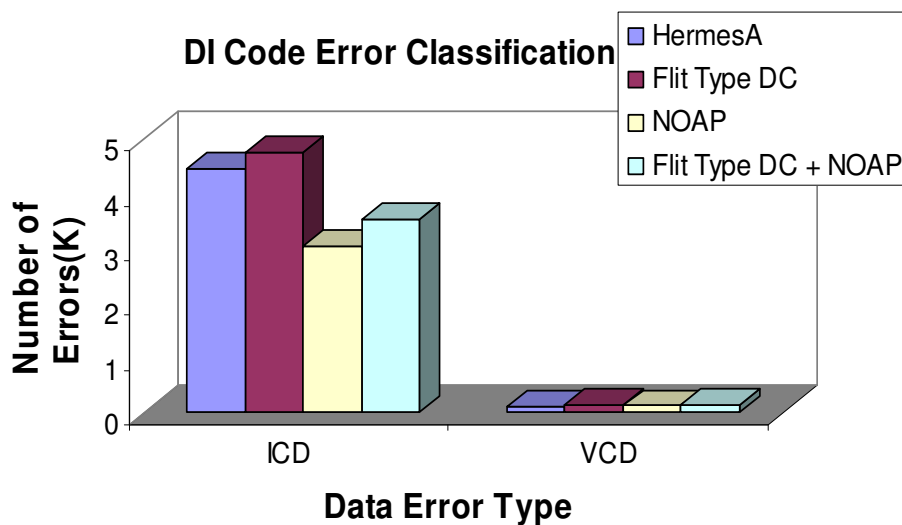


Figure 82 – DI code corruption classification for different Output Port implementations. The results are for an injection charge of 100 fC.

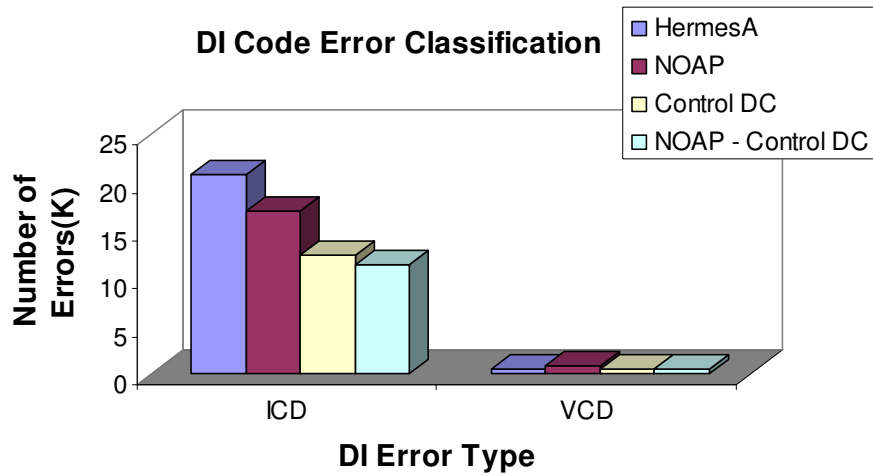


Figure 83 - DI code corruption classification for different Input Port implementations. The results are for an injection charge of 100 fC.

Figure 84 and Figure 85 show the classification of the errors as a function of the victim flit. In these Figures, it is possible to note the capacity of the NOAP technique to filter errors when the Output Port is in the idle state. This observation relies on the number of errors in the first flit. The first flit carries the faults that are dormant in the circuit during the idle state of the NoC component. The NOAP technique filters this type of errors by enabling the SET rather than SEU.

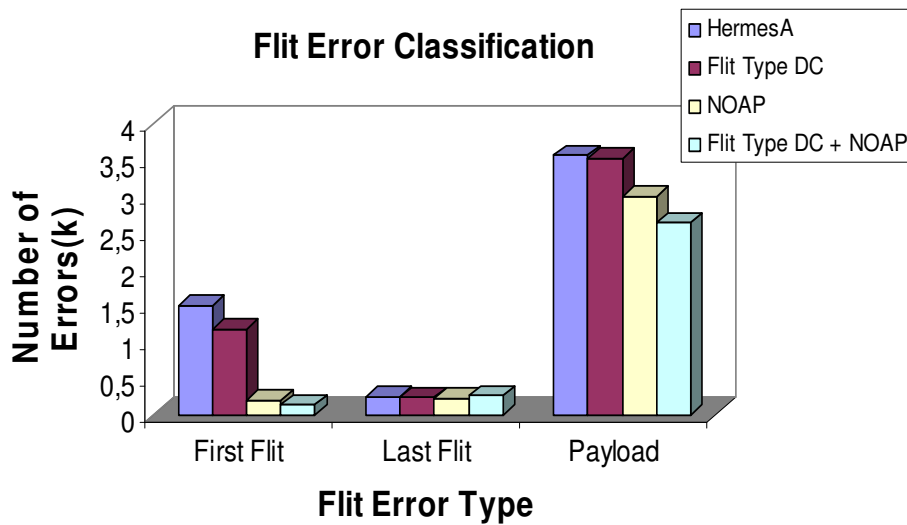


Figure 84 – Output Port flit corruption evaluation as a function of the injected charge.

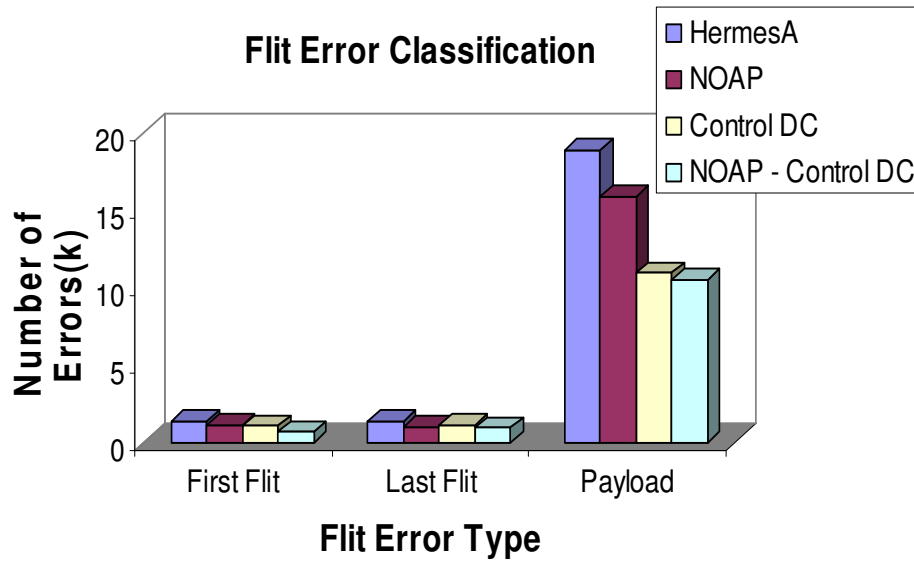


Figure 85 – Input Port flit error classification when a 100fC charge is injected.

Figure 86 and Figure 87 show the stall occurrences during the Output and Input Port evaluations. The best technique to reduce stall as predicted is the double check in the flit type signals and in the routing information presented in the first flit. However, it is interesting to note that even without redundancy, the NOAP technique displays good results. The stall reduction is a function of the filtering property during the idle state. This reduces the number of faults in the flit type signals also and therefore, reduces the number of errors in the packet control flow. The combined use of NOAP and double check do not improve the results obtained by the Double Check technique. Since double check already covers control signals, NOAP does not bring any further improvement.

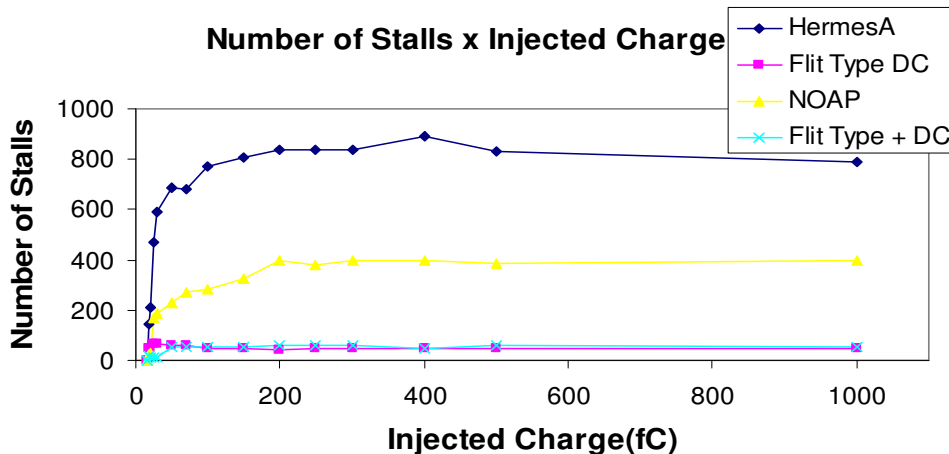


Figure 86 – Evaluation of the stall errors in Output Port as a function of the injected charge.

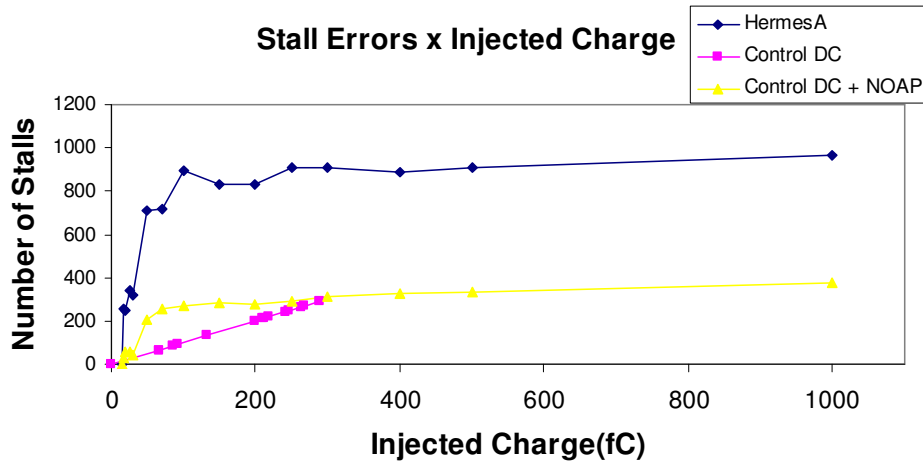


Figure 87 - Evaluation of the stall errors in Input Port in function of the injected charge.

The last classification presented in this Section is the impact of the Injected Charge in packet error classification. In Figure 87, UF refers to the presence of an *Unexpected Flit* in the Data Checker. IP means that the Data Checker detected an *Incomplete Packet*. The CFI abbreviation stands for a *Corrupted Flit Indication*, i.e. the Data Checker received a flit with wrong flit type indication. The TPE acronym means *Total Packet Error Number* registered by the Data Checker.

As expected, double check applied to the flit type brings the best results for packet hardening. The high number of Unexpected Flits occurs because of the multiplexer controlled by the arbiter. The glitch in the arbitration may duplicate a flit in the Multiplexer. Since the Begin of Packet Indication and the Kill Flit control the input of the arbiter, double check may reduce errors caused by glitches at the input of the arbitration. However, it is not possible to filter the SEE generated inside the arbiter.

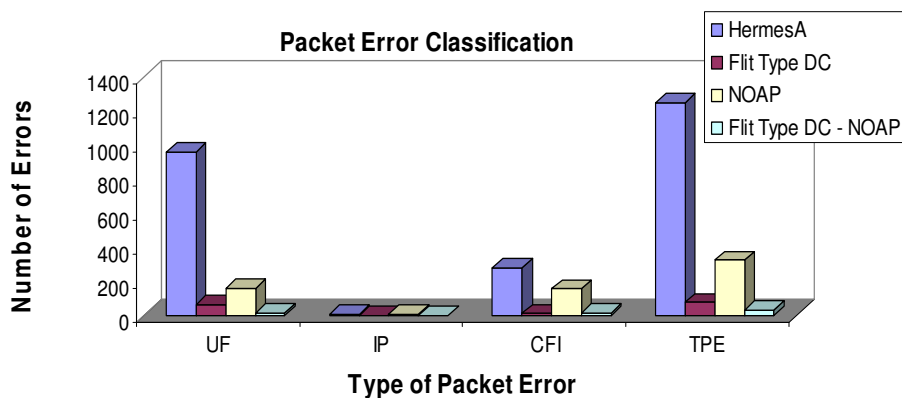


Figure 88 – Packet corruption classification as a function of the injected charge for the Output Port.

8.2 1-of-4 Parity Implementation

In this Section, the use of parity check associated to Unordered Codes helps detecting and cor-

recting errors specifically for the 1-of-4 code. Results presented here compare the Output Port Data Errors results for the flit type associated to NOAP. Two implementations were tested: one using the proposed 1-of-4 parity correction and another, without data correction. Implementations are compared under the scenarios exploited in the previous Sections: varying the injected charge. No ICD violations arose during any simulation. Errors reported in the 1-of-4 parity results are VCDs that arise mainly from duplicated flits.

Figure 89 shows a comparison when changing the injected charge. The number of errors observed in the 1-of-4 parity implementation has a slow increasing rate. For higher levels of charge, the number of errors is more than 20 times smaller than the other implementation.

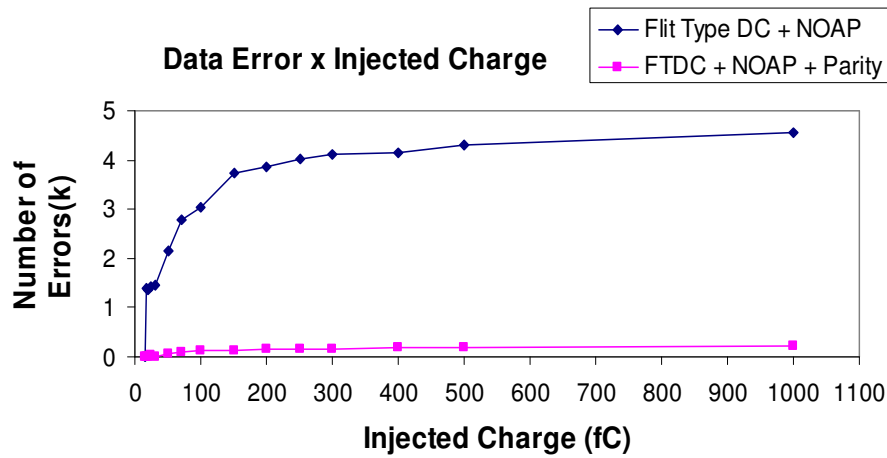


Figure 89 – Data error evaluation as a function of the SEE injected charge, for the m-of-n Parity correction evaluation at the Output Port.

8.3 Area, Power and Performance Evaluation

This Section presents area, static power and performance results for the Output Port implementation. The evaluated implementations are the same of previous robustness results. These components were synthesized using the method proposed by Thonnart et al. in [90]. The same timing constraints apply to all designs.

The results show best performance in some more complex implementations, like for the flit type double check (DC) and for the design with parity. This is due the fact that the RTL Compiler synthesis tool needs to optimize these designs more deeply to meet constraints than it does to the simpler designs. As a result, the component has smaller propagation paths between asynchronous buffers. In the NoC library, the fastest Output Port implementation (using standard voltage threshold as the components evaluated in this work) is the regular Hermes-A implementation, which reaches 17.5Gbits/sec. Another interesting result is for the NOAP with and without the flit type DC. In the

implementation with the flit type DC and the NOAP technique, the NOAP the result of the double check in the flit type controls the C-element. This is faster than performing completion detection, as the regular NOAP implementation does.

The results show that the area overhead of the proposed techniques are minor – 9% for the flit type DC, 2% for the NOAP technique, 12% for the flit type+NOAP and 11.6% for the parity implementation. The same is true for the static power where the maximum overhead is equal to 11% for the Parity implementation.

Table 8 - Evaluation of the Output Port implementations.

Implementation	Area (μm^2)	Throughput (Mbits/sec)	Latency (ns)	Static Power (μW)
Hermes-A	10940	11.1	1.823	81.5
FT-DC	11938	11.6	1.72	84.7
NOAP	11203	8.85	2.68	85.7
FT-DC+NOAP	12350	9.44	2.38	86
Parity	12208	12	1.61	91
FT-DC+NOAP+Parity	13600	10	2.24	101

Table 9 shows the evaluation of the Input Port. As in the Output Port, the results shows that the overhead in area, power and performance are small also if compared with traditional techniques.

Table 9 – Evaluation of the Input Port implementations.

Implementation	Area (μm^2)	Throughput (Mbits/sec)	Latency (ns)	Static Power (μW)
Hermes-A	9480	12	2.63	60
CDC	11507	8.2	3.42	72
NOAP	10205	8.5	3.77	61
CDC+NOAP	12522	9.8	3.25	74.4
Parity	10853	12	1.61	67.9
CDC+NOAP+Parity	13847	10.1	311	80.5

8.4 PARANA Synthesis Tool Evaluation

Two different evaluations were done. In the first evaluation, the PARANA tool was used to generate four different NoCs without Channel Cut-Off, that is, all generated NoCs have all components. These NoCs were evaluated using the SEE Evaluation Environment. In the second scenario, the PARANA synthesis tool generates the NoCs with the Channel Cut-off. For this scenario, estimations about the NoC area are generated.

The PARANA tool evaluation was done based on MultiMedia System (MMS) application described initially in [37]. Marcon suggests in [47] to cluster application tasks and perform the mapping using the CWM model, which employs the communication volume as the weight in graph edges. In order to enable the use of the MMS application description in the PARANA tool, the application CWG was adapted using: (a) packet size (ps) = 16; (b) packet rate pr = the original CWG

communication volume of the original graph divided by the packet size (ps); (c) all IP Cores working at 100MHz ($sfr = rfr = 100$).

Mapping was done using the CAFES tool [47]. This tool performs application mapping based on the traffic description of applications. CAFES offers different models to permit the representation of several different SoCs behaviors. In order to generate a mapping based on the hardening level of the communications, the following equation was employed to convert from HGCG to CWM:

$$(18) \quad W_i = \alpha * pr_i + \beta * \text{maximum}(sfr_i, rdr_i) + \chi * ps + \delta * hl$$

Where α , β , χ , δ are individual weights associated for each element of the HGCG edge ($=\alpha = 0$, $\beta = 1$, $\chi=0$ and $\delta = \text{maximum}(sfr)$). Figure 90 shows the adapted CWG for the MMS application. The original CWG for the application is described in [37].

In order to enable the the NoC synthesis in the PARANA tool, the following considerations about the hardening level (hl) were done for the MMS application: (a) All communication from and to $CPU1$ node assumes the hardening level equal to maximum ($\theta = 4$ – meaning that Data and packets are protected). (b) All communication with memories (MEM1, MEM2 and MEM3), are assumed to have the hardening applied just to the Data ($\theta = 2$). (c) All other communications are assumed not to be hardened ($\theta = 1$).

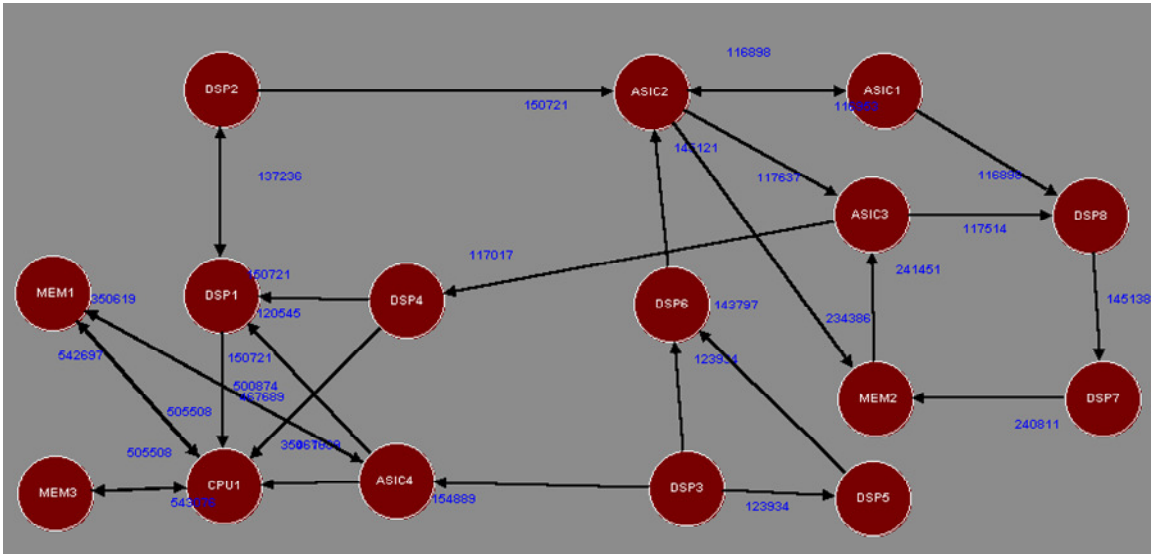


Figure 90 – CWG for the MMS application adapted for the Hardening Mapping.

The adapted graph was used in the CAFES tool to generate a mapping for the application. The mapping result of the adapted CWG generated by the Cafes tool appears in Figure 91.

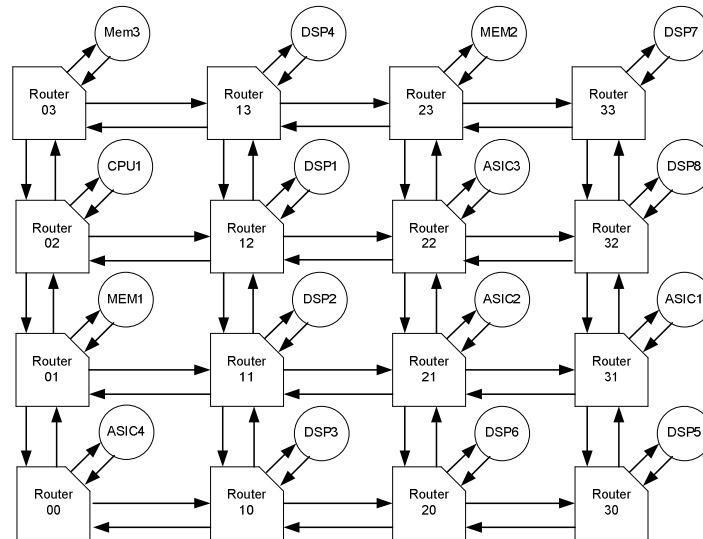


Figure 91 – Multimedia System after the mapping done with the adapted CWG for hardened mapping.

8.4.1 Regular NoCs Evaluation

In order to evaluate the NoCs behavior under SEEs, the PARANA tool was used to generate four NoCs. The mapping of Figure 91 served as input to the PARANA tool. The NoCs were generated without any channel cut-off, that is, in this first evaluation all the NoCs have all the Input/output Ports. Bellow is the description of generated NoCs.

1. No hardening technique on NoC and no data correction
2. Data correction and no hardening technique on NoC
3. Hardened NoC without data correction
4. Hardened NoC and Data Correction

The SEE evaluation environment was modified to enable the fault injection in the NoC Components. In the NoC SEE evaluation the Input/output Ports that are part of the same router are clustered. A hierarchical fault generator is then used. The hierarchical fault generator has a single fault generator master and one fault generator slave for each router. The master randomly chooses a slave to inject fault. The slave is responsible for randomly selecting a victim cell to inject the error. In order to enable the timing/fault annotation of the NoC, each NoC component has a specific timing/fault file (.sdf). Figure 92 shows the NoC SEE evaluation environment. Each slave inside the Fault Generator controls one SEE bus.

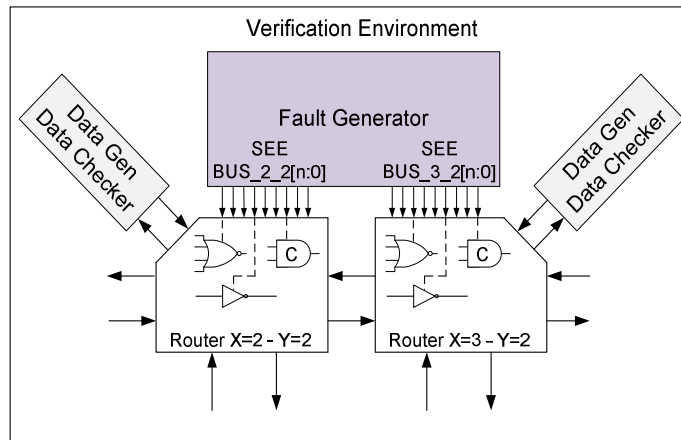


Figure 92 – Soft Error Evaluation Environment used to characterize the Asynchronous NoCs.

Figure 93 shows the data error evaluation of the NoCs – (a) shows the results for the NoC without hardening technique, (b) shows the results for the NoC with parity correction, (c) shows the results of the hardened NoC and (d) shows the results of the hardened NoC and data correction. The graphs show that the combined use of Parity Data correction and hardened NoC can reduce drastically the Data Errors: reducing the number of VCDs and eliminating the ICDs.

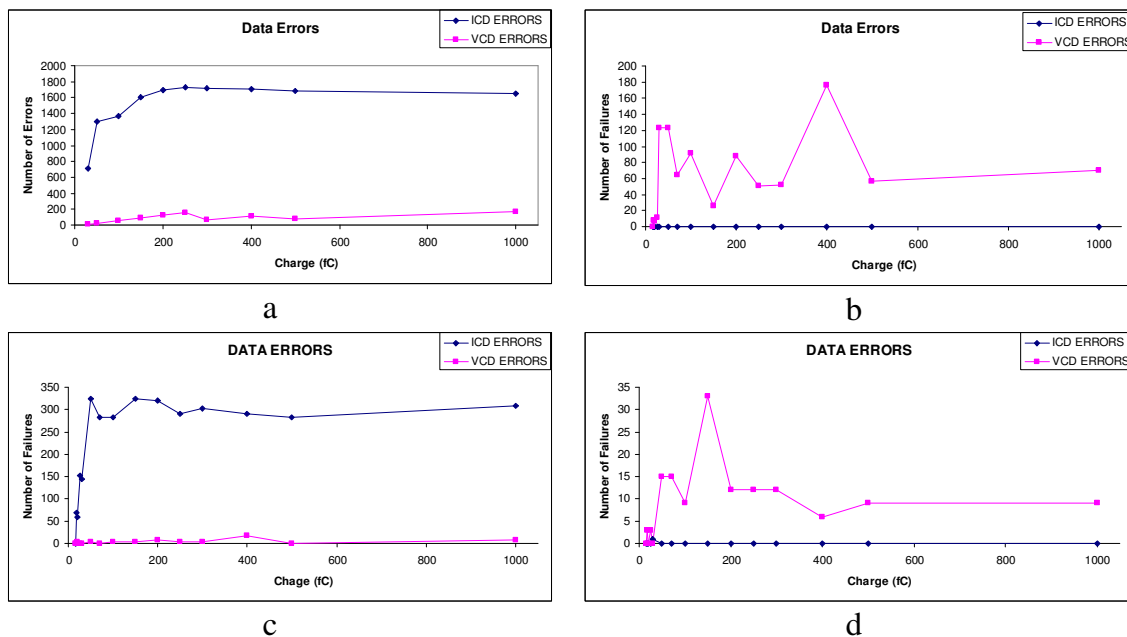


Figure 93 - Data error evaluation of the NoCs – (a) shows the results for the NoC without hardening technique, (b) shows the results for the NoC with parity correction, (c) shows the results of the hardened NoC and (d) shows the results of the hardened NoC and data correction

Table 10 shows the area results of the Parity Encoder/Decoder. The average area overhead when compared to a NoC router for Parity Decoder/Encoder is also evaluated. The results show that the Parity scheme can correct the most part of the data errors without incur in significant area overhead.

Table 10 – Parity encoder and decoder area evaluation.

Implementation	Single Area (μm^2)	Area Overhead (%)
Encoder	1121	1.37
Decoder	4299	5.2

Figure 94 shows the VCD errors detected in the NoC simulation of the Hardened NoC but using the TRDIC Decoder in place of the Parity Decoder. In this scenario, all ICDs were corrected. The few VCDs are cases of burst of errors inside the Trellis. If more than one error is presented at the Trellis, the TRDIC can perform the wrong decision during the correction. This results in an Data Error. This error bursts are only possible in environments with high SEE rate. The results in Figure 93 and in Figure 94 prove the increasing in the NoC reliability. The data errors were almost eliminated from the packet transmission. The errors presented in the results are for high SEE injection rates. The evaluation environment is more pessimist than the real cases in order to speed up the evaluation. In real case situations, the proposed mechanisms are probably free from Data Errors.

The TRDIC Encoder/Decoder is under evaluation and the results about area were not generated so far. The evaluation of area and power of the NoCs and Data Correction mechanisms is a future work.

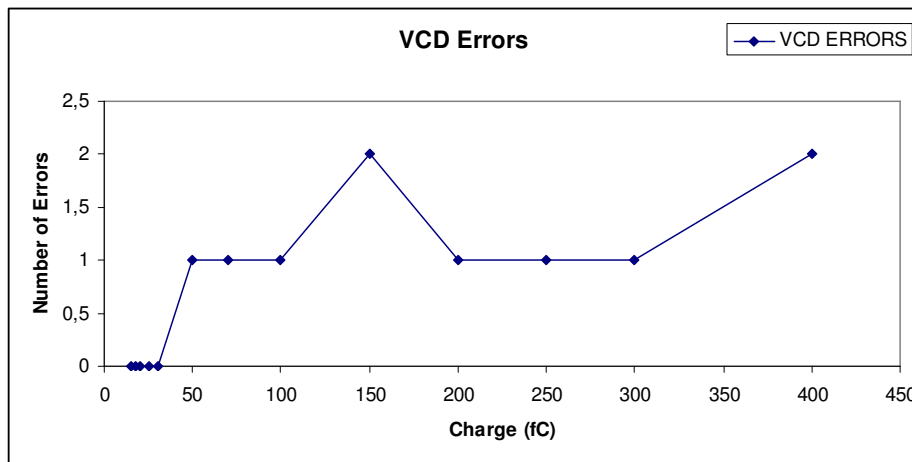


Figure 94 – VCD error evaluation of the hardened NoC using the TRDIC data correction mechanism.

Figure 95 and Figure 96 show the stall error evaluation of the NoCs. Figure 95 shows the results for the NoC without hardening technique, while Figure 96 shows the results of the hardened NoC and data correction. The results show that the proposed hardening techniques for the NoC can reduce the number of stalls in the NoC. To solve the stall a reset must be generated. The reset puts the NoC in the initial state and all packets are vanished. The hardened version of the NOC shows better results for stall errors also. In this way, a layered NoC can increase the availability of the critical applications since a reset in one layer do not affect the other NoC layers.

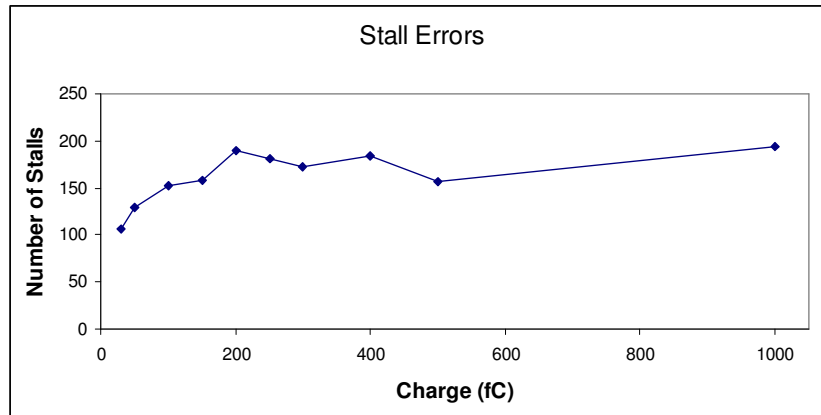


Figure 95 – Stall Errors evaluation of the NoC without hardening techniques (Hermes-A).

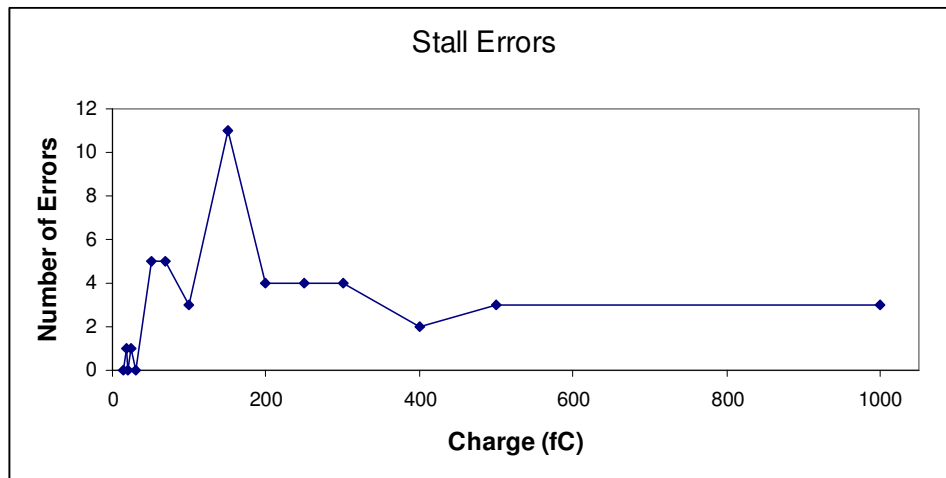


Figure 96 – Stall Errors evaluation of the Hardened NoC (H2A).

8.4.2 Layered NoC Evaluation

In the second evaluation, the PARANA tool was used to generate a NoC composed of three different layers for the application, corresponding to the MMS application described in Figure 91. Each communication of the graph has a hardening level associated, as described before in this Chapter. The required robustness levels define the layer where the application is mapped.

Table 11 shows the area results for each NoC layer and for the whole NoC. In addition, results show the comparison with a NoC fully implemented with a hardening level equal to four (the maximum level). The results show that the PARANA tool can generate hardened NoCs, increasing the reliability and the availability without incurring in excessive area overhead.

Table 11 - Area results for the hardened multilayer NoC, for the adapted Multimedia System application. The Output Port and Input Port columns show the number of used components and the number of the components in a full mesh NoC.

NoC Layer	Output Ports	Input Ports	Cell Area (mm²)
Layer 1	31	30/64	0.69
Layer 2	10/64	11/64	0.23
Layer 3	0/64	0/64	0
Layer 4	10/64	13/64	0.25
Total	51	54	1.18
Full NoC	64	64	1.76

9. CONCLUSIONS AND FUTURE WORK

The timing robustness of QDI circuits give asynchronous NoCs the robustness needed to help SoC designers to tackle problems that arise in modern deep submicron technologies. By studying the radiation effects over asynchronous NoCs, this work helps to increase the robustness of these communication architectures against Soft Errors also. Results of this work show that asynchronous NoCs can be easily hardened with small area and static power overhead.

The presented data correction methods are a significant contribution of the thesis. The proposed use of parity associated to the inherited properties of unordered codes to detect errors generates a data correction scheme with low complexity and capable of correcting most errors in data transmission. The TRDIC code adds even more power to data correction and can be used in combination with the parity scheme. In a static combination, parity decoding is applied first to remove burst errors in Trellis. Conversely, the parity-TRDIC combination can be dynamic, switching the error correction mechanism between parity and TRDIC based e.g. on the data error rate. Another property of parity and TRDIC is that they can be adapted to several different m-of-n codes [8][13][17][79]. In this way, different NoC implementations can use these two methods to provide data correction inside the NoC.

As far as the author could verify, the H₂A NoC proposed here is the first Hardened Asynchronous NoC. Hardening is treated at several different levels during NoC design. The proposed NOAP technique is efficient to increase the handshake protocol robustness. In addition, this work proposed the use of double check to increase the robustness in specific NoC control signals, increasing robustness without incurring in the intrinsic significant area overhead of double check.

Hermes-A is the communication architecture that served as base of this work. This NoC architecture was extended and now corresponds to a library of NoCs with different DI codes, different routing algorithms and implementations. This brings new possibilities for the design exploration at the architecture level, allowing the construction of NoCs more adapted to each application. The PARANA NoC Synthesis Tool helps in this task, by providing methods to automate the NoC generation process. The proposed NoC Component Library, together with the PARANA Tool enable design space exploration of NoC architectures using high-level constraints including robustness level specification.

The infrastructure generated in this work enables further investigation of new techniques for mitigating the radiation effects over NoC components. Further investigation over the NoC components behavior in presence of SEEs and under different traffic conditions may help identify the most suitable solutions for each application or/and enable the NoC to adapt at runtime to robustness requirements.

This work also enables the design exploration of GALS SoCs using higher-level specifications. This includes the research of appropriated mapping techniques for GALS SoCs and for Hardened SoCs. The integration of mapping methods in the NoC Synthesis Flow may help providing better NoC solutions.

REFERENCES

- [1] Actel Inc. "Understanding soft and firm errors in semiconductor devices". Captured in "<http://www.actel.com/documents>", Jul. 2012. 6 p.
- [2] Agyekum, M. Y.; Nowick, S. M. "An error-correcting unordered code and hardware support for robust asynchronous global communication". In: Design, Automation & Test in Europe Conference, 2011, pp. 765-770.
- [3] Amde, M.; Felicijan, T.; Efthymiou, A.; Edwards, D.; Lavagno, L. "Asynchronous on-chip networks". IEE Proceedings - Computers and Digital Techniques, vol. 152-2, Mar. 2005, pp. 273-283.
- [4] Angiolini, F.; Ceng, J.; Leupers, R.; Ferrari, F.; Ferri, C.; Benini, L. "An integrated open framework for heterogeneous MPSoC design space exploration". In: Design, Automation & Test in Europe Conference, 2006, pp. 1145-1150.
- [5] Asadi, H.; Tahoori, M. "Soft error modeling and remediation techniques in ASIC designs". Microelectronics Journal, vol. 41, Jul. 2010, pp. 506-522.
- [6] Atienza, D.; Angiolini, F.; Murali, S.; Pullini, A.; Benini, L.; De Micheli, G. "Network-on-Chip Design and Synthesis Outlook". Integration the VLSI Journal, vol. 41-3, 2008, pp. 340-359.
- [7] Bailey, A.; Zahrani, A.; Guoyuan, F.; Di, J.; Smith, S. "Multi-Threshold Asynchronous Circuit Design for Ultra-Low Power". Journal of Low Power Electronics, vol. 4-3, Dec. 2008, pp. 337-348.
- [8] Bainbridge, J.; Furber, S. "Chain: A Delay-Insensitive Chip Area Interconnect". IEEE Micro, vol. 22-5, Sep.-Oct., 2002, pp. 16-23.
- [9] Bainbridge, W. J.; Salisbury, S. J. "Glitch sensitivity and defense of quasi delay insensitive network-on-chip links". In: IEEE International Symposium on Asynchronous Circuits and Systems, 2009, pp. 35-44.
- [10] Bastos, R. P.; Sicard, G.; Kastensmidt, F.; Renaudin, M.; Reis, R. "Asynchronous circuits as alternative for mitigation of long-duration transient faults in deep-submicron technologies". Microelectronics Reliability, vol. 50, Nov. 2010, pp. 1241-1246.

- [11] Bastos, R. P.; Sicard, G.; Kastensmidt, F.; Renaudin, M.; Reis, R. "Evaluating transient-fault effects on traditional C-element's implementation". In: International On-Line Testing Symposium, 2010, pp. 35-40.
- [12] Baumann, R. "Radiation induced soft errors in advanced semiconductor technologies", IEEE Transactions on Device and Materials Reliability, vol. 5-3, Sep. 2005, pp.305-316.
- [13] Beigné, E.; Clermidy, F.; Vivet, P.; Clouard, A.; Renaudin, M. "An Asynchronous NoC Architecture Providing Low Latency Service and its Multi-level Design Framework". In: IEEE International Symposium on Asynchronous Circuits and Systems, 2005, pp. 54-63.
- [14] Beigné, E.; Clermidy, F.; Miermont, S.; Vivet, P. "Dynamic Voltage and Frequency Scaling Architecture for Units Integration within a GALS NoC". In: 2nd ACM/IEEE International Symposium on Networks-on-Chip, 2008, pp.129-138.
- [15] Bertozzi, D.; Benini, L. "Xpipes: a network-on-chip architecture for gigascale systems-on-chip". IEEE Circuits and Systems Magazine, vol. 4-2, 2004, pp. 18-31.
- [16] Bertozzi, D.; Jalabert, A.; Tamhankar, R.; Stergiou, S.; Benini, L.; De Micheli, G. "NoC Synthesis Flow for Customized Domain Specific Multiprocessor System-on-Chip". IEEE Transactions on Parallel and Distributed Systems, vol. 16-2, Feb. 2005, pp. 113-129.
- [17] Bjerregaard, T.; Sparsø, J. "A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip". In: Design, Automation & Test in Europe Conference, 2005, pp. 1226-1231.
- [18] Bjerregaard, T.; Mahadevan, S. "A Survey of Research and Practices of Network-on-Chip". ACM Computing Surveys, vol. 38-1, 2006, pp. 1-51.
- [19] Brej, C. "Early Output Logic and Anti-Tokens". PhD Thesis, Department of Computer Science, University of Manchester, 2005, 137p.
- [20] Cadence Inc. "Encounter Digital Implementation System". User Guide, Cadence, 2010.
- [21] Calazans, N. L. V. "Automated Logic Design of Sequential Digital Circuits". Imprinta, 1998. 342p. (In Portuguese).
- [22] Carara, E.; Calazans, N.; Moraes, F. G. "New Router Architecture for High-Performance Intrachip Networks". Journal of Integrated Circuits and Systems, vol. 3-1, 2008, pp. 23-31.
- [23] Chapiro, D. "Globally-Asynchronous Locally Synchronous Systems". PhD Thesis, Stanford University, 1984, 134p.

- [24] Cortadella, J.; Kishinevsky, M.; Kondratyev, A.; Lavagno, L. "Introduction to asynchronous circuit design: specification and synthesis". Tutorial. In: 6th International Symposium on Advanced Research in Asynchronous Circuits and Systems, 2000.
- [25] Cortadella, J.; Kondratyev, A.; Lavagno, L.; Sotiriou, C. "Coping with the variability of combinational logic delays" In: IEEE International Conference on Computer Design, 2004, pp. 505- 508.
- [26] Cadence Inc., "Common Power Format". User Guide, Jan. 2010.
- [27] Cummings, C. E. "Simulation and Synthesis Techniques for Asynchronous FIFO Design". In: Synopsys User Group, 2002, 18p.
- [28] Dally, W. J.; Towles, B. "Principles and Practices of Interconnection Networks". San Francisco: Morgan Kaufmann, 2004, 550 p.
- [29] Devarapalli, S.V.; Zarkesh-Ha, P.; Suddarth, S.C. "SEU-Hardened Dual Data Rate Flip-Flop Using C-Elements". In: IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2010, pp.167-171.
- [30] Dinh-Duc, A. V.; Fesquet, L.; Renaudin, M. "Synthesis of QDI Asynchronous Circuits from DTL-style Petri-net". In: International Workshop on Logic and Synthesis, 2002, pp.191-196.
- [31] Duato, J.; Yalamanchili, S.; Ni, L. "Interconnection Networks". Elsevier Science, 2002, 600p.
- [32] Ferlet-Cavrois, V.; Paillet, P.; McMorrow, D.; Melinger, J.; Campbell, A.; Gaillardin, M.; Faynot, O.; Thomas, O. "Analysis of the transient response of high performance 50-nm partially depleted SOI transistors using a laser probing technique". IEEE Transactions on Nuclear Science, vol. 53-4, Aug. 2006, pp. 1825-1833.
- [33] Frantz, A. P.; Cassel, M.; Kastensmidt, F. L.; Cota, E.; Carro, L. "Crosstalk and SEU-Aware Networks on Chips". IEEE Design & Test of Computers, vol. 24-4, Jul.-Aug. 2007, pp. 340-350.
- [34] Gaillardin, M.; Paillet, P.; Ferlet-Cravois, V.; Baggio, J.; McMorrow, D.; Faynot, O.; Jahan, C.; Tosti, L.; Cristoloveanu, S. "Transient radiation response of single and multiple gate FD SOI transistors". IEEE Transactions on Nuclear Science, vol. 54-6, Dec. 2007, pp. 2355-2362.
- [35] Hennessy, J. L.; Patterson, D. A. "Arquitetura de Computadores: Uma abordagem quantitativa". Campus, Rio de Janeiro, 2003 827p.

- [36] Ho, R.; Mai, K.; Horowitz, M. "The future of wires". Proceedings of the IEEE, vol. 89-4, Apr. 2001, pp. 490-504.
- [37] Hu, J.; Marculescu, R. "Energy- and performance-aware mapping for regular NoC architectures". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 24-4, Apr. 2005, pp. 551-562.
- [38] International Technology Roadmap for Semiconductors, Captured on: <http://www.itrs.net/>, January 2012.
- [39] Jang, W.; Martin, A. "SEU-tolerant QDI circuits". In: IEEE International Symposium on Asynchronous Circuits and Systems, 2005, pp. 156-165.
- [40] Jang, W.; Ding, D.; Pan, D. "A Voltage-Frequency Island Aware Energy Optimization Framework for Networks-on-Chip". In: IEEE/ACM International Conference on Computer-Aided Design, 2008, pp. 264-269.
- [41] Kim, D.; Kim, M.; Sobelman, G. E. "Asynchronous FIFO Interfaces for GALS On-Chip Switched Networks". In: International System on Chip Design Conference, 2005, pp. 186-189.
- [42] Krstic, M.; Grass, E.; Gurkaynak, F.; Vivet, P. "Globally Asynchronous, Locally Synchronous Circuits: Overview and Outlook". IEEE Design & Test of Computers, vol. 24-5, Sep.-Oct. 2007, pp. 430-441.
- [43] Kumar, S.; Jantsch, A.; Soininen, J.-P.; Forsell, M.; Millberg, M.; Oberg, J.; Tiensyrja, K.; Hemani, A. "A network on chip architecture and design methodology". In: IEEE Computer Society Annual Symposium on VLSI, 2002, pp. 105-112.
- [44] Leung, L-F.; Tsui, C-Y. "Energy Aware Synthesis of Network-on-Chip Implemented with Voltage Islands". In: 44th ACM/IEEE Design Automation Conference, 2007, pp. 128-131.
- [45] Lines, A. "Asynchronous Interconnect for Synchronous SoC Design". IEEE Micro, vol. 24-1, Jan.-Feb. 2004, pp. 32-41.
- [46] Marcon, C. A. M.; Borin, A.; Susin, A.; Carro, L.; Wagner, F. "Time and Energy Efficient Mapping of Embedded Applications on NoCs". In: 10th Asia and South Pacific Design Automation Conference, 2005, pp. 33-38.
- [47] Marcon, C. A. M. "Models for Applications Mapping in Intrachip Communication Infrastructures". PhD Thesis, PPGC-II-UFRGS, Porto Alegre, 2005, 192 p.
- [48] Martin, A. "The limitations to delay-insensitivity in asynchronous circuits". In: MIT Conference on Advanced Research in VLSI, 1990, pp. 263-278.

- [49] Martin, A.; Nystrom, M. "Asynchronous techniques for system-on-chip design". Proceedings of the IEEE, vol. 94-6, Jun. 2006, pp. 1089-1120.
- [50] Mello, A.; Tedesco, L.; Calazans, N.; Moraes, F. "Virtual Channels in Networks on Chip: Implementation and Evaluation on Hermes NoC". In: 18th Symposium on Integrated Circuits and Systems Design, 2005, pp. 178-183.
- [51] Mohanram, K. "Closed-form simulation and robustness models for SEU-tolerant design". In: 23rd VLSI Test Symposium, 2005, pp. 327-333.
- [52] Monnet, Y.; Renaudin, M.; Leveugle, R. "Hardening techniques against transient faults for asynchronous circuits". In: International On-Line Testing Symposium, 2005, pp. 129-134.
- [53] Monet, Y.; Renaudin, M.; Leveugle, R. "Formal analysis of quasi delay insensitive circuits behavior in the presence of SEUs". In: International On-Line Testing Symposium, 2007, pp. 113-120.
- [54] Moraes, F.; Calazans, N.; Mello, A.; Möller, L.; Ost, L. "Hermes: an Infrastructure for Low Area Overhead Packet-switching Networks on Chip". Integration the VLSI Journal, vol. 38-1, Oct. 2004, pp. 69-93.
- [55] Moreira, J. C.; Farrel, P. G. "Essentials of error-control coding". Wiley, 2006, 388p.
- [56] Moreira, M. T.; Oliveira, B. S.; Pontes, J. J. H.; Moraes, F. G.; Calazans, N. L. V. Impact of C-Elements in Asynchronous Circuits. In: International Symposium on Quality Electronic Design, 2012, pp. 438-444.
- [57] Moreno, E. I. "Mapeamento e Adaptação de Rotas de Comunicação em Redes em Chip". PhD Thesis, PPGCC - PUCRS, Jan. 2010, 175p. (In Portuguese).
- [58] Mukherjee, S. "Architecture Design for Software Errors". Morgan Kaufmann Publishers, Burlington, 2008. 337p.
- [59] Murali, S.; Theocharides, T.; Vijaykrishnan, N.; Irwin, M.J.; Benini, L.; De Micheli, G. "Analysis of error recovery schemes for networks on chips". IEEE Design & Test of Computers, vol. 22-5, Sept.-Oct. 2005, pp. 434- 442.
- [60] Ogg, S.; Al-Hashimi, B.; Yakovlev, A. "Asynchronous transient resilient links for NoC". In: International Conference on Hardware/Software Codesign and System Synthesis, 2008, pp. 209-214.
- [61] Ogras, U.; Marculescu, R.; Choudhary, P.; Marculescu, D. "Voltage-frequency island partitioning for GALS-based networks-on-chip". In: 44th ACM/IEEE Design Automation Conference, 2007, pp. 110-115.

- [62] Omaña, M.; Rossi, D.; Metra, C. “Model for transient fault susceptibility of combinational circuits”. *Journal of Electronic Testing: Theory and Applications*, vol. 20-5, Oct. 2004, pp. 501-509.
- [63] Open Source Liberty. Captured on <http://www.opensourceliberty.org>, Jan 2011.
- [64] Ost, L. C. “Abstract Models of NoC-Based MPSoCs for Design Space Exploration”. PhD Thesis, PPGCC-PUCRS, Porto Alegre, 2010, 99p.
- [65] Palma, J. C. S. “Reduzindo o Consumo de Potência em Networks-on-Chip através de Esquemas de Codificação de Dados”. PhD Thesis, PPGC-UFRGS, Porto Alegre, 2007, 144p. (In Portuguese).
- [66] Pontes, J.; Soares, R.; Carvalho, E.; Moraes, F.; Calazans, N. “SCAFFI: An intrachip FPGA asynchronous interface based on hard macros”. In: *IEEE International Conference on Computer Design*, 2007, pp. 541-546.
- [67] Pontes, J.; Moreira, M.; Soares, R.; Calazans, N. “Hermes-GLP: A GALS Network on Chip Router with Power Control Techniques”. In: *IEEE Computer Society Annual Symposium on VLSI*, 2008, pp. 347-352.
- [68] Pontes, J. “Projeto e Prototipação de Interfaces e Redes Intrachip Não-Síncronas em FPGAs”. MSc. Dissertation, PPGCC-FACIN-PUCRS. February 2008. 122p. (In Portuguese).
- [69] Pontes, J.; Moreira, M.; Moraes, F.; Calazans, N. “HERMES-A - An Asynchronous NoC Router with Distributed Routing”. In: *International Workshop on Power and Timing Modeling, Optimization and Simulation*, 2010, pp. 150-159.
- [70] Pontes, J. J. H.; Moreira, M. T.; Moraes, F. G.; Calazans, N. L. V.: “Hermes-AA: A 65nm Asynchronous NoC Router with Adaptive Routing”. In: *IEEE International SoC Conference*, 2010, pp. 493-498.
- [71] Pontes, J.; Vivet, P.; Calazans, N. “An Accurate Single Event Upset Digital Design Flow for Reliable System Level Design”. In: *Design, Automation & Test in Europe Conference*, 2012, pp. 224-229.
- [72] Pontes, J.; Calazans, N.; Vivet, P. “Adding Temporal Redundancy to Delay Insensitive Codes to Mitigate Single Event Effects”. In: *IEEE International Symposium on Asynchronous Circuits and Systems*, 2012, pp. 142-149.
- [73] Qiaoyan Yu; Ampadu, P. “Dual-Layer Adaptive Error Control for Network-on-Chip Links”. *IEEE Transactions on Very Large Scale Integration Systems*, vol. 20-7, July 2012, pp.1304-1317.

- [74] Quartana, J.; Renane, S.; Baixas, A.; Fesquet, L.; Renaudin, M. “GALS systems prototyping using multiclock FPGAs and asynchronous network-on-chips”. In: International Conference on Field Programmable Logic and Applications, 2005, pp. 299-304.
- [75] Rahbaran, B.; Steininger, A. “Is asynchronous logic more robust than synchronous logic?” IEEE Transactions on Dependable and Secure Computing, vol. 6-4, Dec. 2009, pp. 282-294.
- [76] Rao, R.; Chopra, K.; Blaauw, D.; Sylvester, D. “Computing the soft error rate of a combinational logic circuit using parameterized descriptors”. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 26-3, Mar. 2007, pp. 468-479.
- [77] Renaudin, M. “Asynchronous circuits and systems : a promising design alternative”. Microelectronics for Telecommunications: Managing High Complexity and Mobility, vol. 54-1, Dec. 2000, pp. 133–149.
- [78] Rossi, D.; Angelini, P.; Metra, C. “Configurable Error Control Scheme for NoC Signal Integrity”. In: IEEE International On-Line Testing Symposium, 2007, pp.43-48.
- [79] Rostislav, D.; Vishnyakov, V.; Friedman, E.; Ginosar, R. “An asynchronous router for multiple service levels networks on chip”. In: IEEE International Symposium on Asynchronous Circuits and Systems, 2005, pp. 44-53.
- [80] Scherer, C. A. “Redes Intrachip com Topologia Toro e Modo de Chaveamento Wormhole: Projeto, Geração e Avaliação”. Master Dissertation, PPGCC-PUCRS, Porto Alegre, 2007, 101p. (In Portuguese).
- [81] Seiculescu, C.; Murali, S.; Benini, L.; De Micheli G. “NoC Topology Synthesis for Supporting Shutdown of Voltage Islands in SoCs”. In: 46th ACM/IEEE Design Automation Conference, 2009, pp. 822-825.
- [82] Seifer, N.; Gill, B.; Pellish, J. A.; Marshall, P. W.; LaBel, K. A. “The Susceptibility of 45 and 32 nm Bulk CMOS Latches to Low-Energy Protons”. IEEE Transactions on Nuclear Science, vol. 58-6, Dec. 2011, pp. 2711-2718.
- [83] Sheibanyrad, A.; Greiner, A.; Miro-Panades, I. “Multisynchronous and Fully Asynchronous NoCs for GALS Architectures”. IEEE Design & Test of Computers, vol. 25-6, Nov.-Dec. 2008, pp. 572-580.
- [84] Singh, M.; Nowick, S. M. “Synthesis for Logical Initializability of Synchronous Finite-State Machines”. IEEE Transactions on VLSI Systems, vol. 8-5, Oct. 2000, pp. 542-557.
- [85] Sparsø, J.; Furber, S. “Principles of Asynchronous Circuit Design – A Systems Perspective”. Kluwer Academic Publishers, Boston, 2001. 354p.

- [86] Srinivasan, K.; Chatha, K.; Konjevod, G. "Linear-Programming-Based Techniques for Synthesis of Network-on-Chip Architectures". In: IEEE International Conference on Computer Design, 2006, pp. 422-429.
- [87] IEEE. "IEEE Standard for Standard Delay Format (SDF) for the Electronic Design Process". Captured on: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=972829, January 2001.
- [88] Teehan, P.; Greenstreet, M.; Lemieux, G. "A Survey and Taxonomy of GALS Design Styles". IEEE Design & Test of Computers, vol. 24-5, Sep.-Oct. 2007, pp.418-428.
- [89] Thonnart, Y.; Beigné, E.; Valentian, A.; Vivet, P. "Power Reduction of Asynchronous Logic Circuits Using Activity Detection". IEEE Transactions on Very Large Scale Integration Systems, vol. 17-7, Jul. 2009, pp.893-906.
- [90] Thonnart, Y.; Beigné, E.; Vivet, P. "A Pseudo-Synchronous Implementation Flow for WCHB QDI Asynchronous Circuits". In: IEEE International Symposium on Asynchronous Circuits and Systems, 2012, pp.73-80.
- [91] Van Berkel, K. "Beware the Isochronic Fork". Integration, the VLSI journal, vol. 13-2, Jun. 1992, pp. 103-128.
- [92] Verma, H. R. "Atomic and Nuclear Analytical Methods", Springer, Berlin, 2007, 376p.
- [93] Wang, F.; Xie, Y. "Soft error rate analysis for combinational logic using an accurate electrical masking model". IEEE Transactions on Dependable and Secure Computing, vol. 8-1, Feb. 2011, pp. 137-146.
- [94] Widmer, A. X.; Franaszek, P. A. "A DC-balanced, partitioned-block, 8B/10B transmission code". IBM Journal of Research and Development, vol. 27-5, September 1983, pp. 440-451.
- [95] Yaghini, P.M.; Eghbal, A.; Pedram, H.; Zarandi, H.R. "Investigation of Transient Fault Effects in an Asynchronous NoC Router". In: Euromicro International Conference on Parallel, Distributed and Network-Based Processing, 2010, pp.540-545.
- [96] Yahya, E.; Renaudin, M. "Asynchronous Buffers Characteristics: Modeling and Design". Research Report, TIMA, 2006, 11p.
- [97] Yebin Shi; Furber, S.B.; Garside, J.; Plana, L.A. "Fault Tolerant Delay Insensitive Inter-chip Communication". In: IEEE International Symposium on Asynchronous Circuits and Systems, 2009, pp.77-84.
- [98] Zahrani A.; Bailey A.; Fu G.; Di J. "Glitch-free design for multi-threshold CMOS NCL circuits". In: 19th ACM Great Lakes symposium on VLSI, 2009, pp 215-220.

- [99] Zhang, B.; Wang, A.; Orshansky, M. "FASER: Fast analysis of soft error susceptibility for cell-based designs". In: International Symposium on Quality Electronic Designs, 2006, pp. 755-760.
- [100] Zimmer, H.; Jantsch, A. "A fault model notation and error-control scheme for switch-to-switch buses in a network-on-chip". In: International Conference on Hardware/Software Co-design and System Synthesis, 2003, pp. 188-193.