T.C. BAHÇEŞEHİR ÜNİVERSİTESİ

NEW LOGIC ARCHITECTURES FOR ROUND ROBIN ARBITRATION AND THEIR AUTOMATIC RTL GENERATION

M.S. Thesis

ONUR BAŞKİRT

İSTANBUL, 2008

T.C. BAHÇEŞEHİR ÜNİVERSİTESİ

INSTITUTE OF SCIENCE

ELECTRICAL & ELECTRONICS ENGINEERING

NEW LOGIC ARCHITECTURES FOR ROUND ROBIN ARBITRATION AND THEIR AUTOMATIC RTL GENERATION

M.S. Thesis

ONUR BAŞKİRT

Advisor: Dr. H. FATİH UĞURDAĞ

İSTANBUL, 2008

BAHÇEŞEHİR ÜNİVERSİTESİ institute of science electrical & electronics engineering

Name of the thesis: New Logic Architectures for Round Robin Arbitration and Their Automatic RTL Generation Name/Last Name of the Student: Onur BAŞKİRT Date of Thesis Defense: June 6, 2008

The thesis has been approved by the Institute of Science.

Prof. Erol SEZER Director

I certify that this thesis meets all the requirements as a thesis for the degree of Master of Science.

Asst. Prof. Bülent BİLİR Program Coordinator

This is to certify that we have read this thesis and that we find it fully adequate in scope, quality, and content as a thesis for the degree of Master of Science.

Examining Committee Members	Signature
Asst. Prof. H. Fatih UĞURDAĞ	
Asst. Prof. Sezer GÖREN UĞURDAĞ	
Prof. Nizamettin AYDIN	
Asst. Prof. Bülent BİLİR	

ACKNOWLEDGMENTS

It has been an adventurous journey. This journey is called "MS Thesis". When I was a senior at Bahçeşehir University, I saw a course name in the elective course list in Spring semester. That course was "EENG4050 VLSI Design". Then, I picked that course and my journey started at point. Thanks to EENG4050, I met Dr. H. Fatih Uğurdağ and I entered the world of digital design.

I would like to express enormous appreciation to my advisor, Dr. H. Fatih Uğurdağ, for his invaluable guidance throughout my long and adventurous MS journey. His many years of digital circuit design experience in Silicon Valley has allowed him to focus on critical and interesting issues of digital design problems. It has been a matchless privilege to be his assistant, and without doubt the credibility and prestige of his name will continue to open doors for me in the future. ACTreS project with Intersil in Silicon Valley, FPGA based GPF card Project with Boğaziçi University and MIT, and my STMicroelectronics employment is an example of his prestige, credibility, and network. I also want to express my gratitude for Dr. Sezer Gören Uğurdağ. It was her connections that allow us to go to University of California at Santa Cruz as a group (BUVLSI) for a three month research appointment in the summer of 2006 for HDLtk project.

I would like to thank the members of my thesis committee for their valuable feedback. I would like to thank Dr. Bülent Bilir for his insightful comments. Also, I would like to thank Dr. Nizamettin Aydın, a member of BUVLSI.

For their help and invaluable friendship, my special thanks go to Aylin Saydağ and Sait Can Saydağ. I will never forget my Orange County visit, their kindness and warmth. Additionally, I am extremely grateful to Aylin and Sait for their help with proofreading of my thesis.

I would like to thank Erinç Topdemir, who was always there for me as a fellow graduate student, as a colleague (teaching – research assistant), and more importantly as one of my best friends. By the way I would like to thank Bahçeşehir University supporting me as a graduate assistant from 2005 to 2007. I also thank all of my ex-colleagues at Bahçeşehir University.

I sincerely appreciate everybody's kindness and warmth at ST. They have truly created a wonderful environment, and I am very pleased to be a member of this team. First, I would like to thank my manager Fatma Özdemir for her understanding and endless support. I want to thank Levent Çetrez for his technical guidance and patience. I have learnt many things from him. I thank my team leader Görkem Canverdi and my teammate Levent Ergün. Whenever I got in trouble, they always helped me. I would like to express my graditute to Ömer Yetik for proofreading and his amazing impersonate talent. I am very thankful to Burak Akoğuz for technical suggestions and our enjoyable table tennis matches. I thank Erdal Öztürk for discussions on technical and more importantly non-technical issues. I also thank Bariş Güven, Cağdaş Sengel, Sinan Topçu, Orkun Sağlamdemir, and the funniest colleague of all Sırma Altay. There are so many people that deserve my appreciation for their friendship, support, and help. If I had to explain why I have to thank each person, these acknowledgments would have to be a chapter in this thesis.

Finally, I would like to thank my wonderful parents. Words cannot convey my appreciation of their love. This thesis would not have been possible without their help. For their dedication to my education, I dedicate this thesis to my parents.

ABSTRACT

NEW LOGIC ARCHITECTURES FOR ROUND ROBIN ARBITRATION AND THEIR AUTOMATIC RTL GENERATION

BAŞKİRT, Onur

Electrical & Electronics Engineering Advisor: Asst. Prof. H. Fatih UĞURDAĞ June 2008, 72 pages

Resource arbitration is a major problem in communications and computer systems. One of the most prevalent usage areas of arbitration is in computer networks. In gigabit and terabit routers, the challenge is to design ultra high speed, cost effective, and fair arbitration hardware to speed up packet forwarding. This issue is highly important for supporting high quality multimedia services in next generation networks.

This thesis is focused on architectures for fast and area efficient round robin arbiters (RRA) and their Register Transfer Level (RTL) design generation. One of the most notable works in this area is the work of Pankaj Gupta and Nick McKeown at Stanford University — which we call Stanford Round Robin Arbiter (STA_RRA). Although there have been further enhancements on top of STA_RRA, we have seen that there is still room for improvement in both speed and area departments.

This thesis work proposes two new RRA logic architectures with better speed or area metrics than STA_RRA and its variants. One of the proposed RRA designs is focused on achieving minimum area results, and the other one is designed for speed. The novelty of these designs is in their use of parallel prefix tree (PPT) algorithms for thermometer encoding and priority encoding operations. Synthesis of proposed arbiters and their rivals were carried out from 8 bits to 256 bits. Benchmarks of 256 bits arbiters show that our proposed architectures perform better than their rivals by a factor of 42% in speed and 22% in area.

Keywords: Round Robin Arbiters, RTL Generation, Parallel Prefix Tree Algorithms

ÖZET

DEĞİŞMEZ ZAMAN PAYLAŞIMLI İŞ-DÜZENLEME İÇİN YENİ MİMARİLER VE BU MİMARİLERİN OTOMATİK YTS ÜRETEÇLERİ

BAŞKİRT, Onur

Elektrik - Elektronik Mühendisliği

Tez Danışmanı: Yrd. Doç. Dr. H. Fatih UĞURDAĞ

Haziran 2008, 72 sayfa

İşlem isteklerinin sıraya konulması (iş-düzenleme), bilgisayar ve iletişim sistemlerinin önemli problemlerinden birisidir. İş-düzenleme işleminin en çok kullanıldığı alanlardan birisi bilgisayar ağlarıdır. Gigabit ve terabit yönlendirici tasarımının önemli uğraşlarından biri, hızlı, maliyeti düşük ve adil iş-düzenleyici donanımları tasarlayarak paket yönlendirme işlemini hızlandırmaktır. Bu konu, yüksek kaliteli, gelecek nesil, çoklu-ortam servislerinin desteklenmesi için son derece kritiktir.

Bu tezde, hızlı ve alan açısından verimli iş-düzenleyici mimarilerine ve bunların Yazmaç Transfer Seviyesi (YTS) tasarım üreteçlerine odaklanılmıştır. Bu alanda en çok dikkate değer çalışma Stanford Üniversitesi'nden Pankaj Gupta ve Nick McKeown'un çalışmasıdır. Biz bu çalışmaya STA_RRA adını verdik. Daha sonraları STA_RRA üzerinde iyileştirme çalışmaları yapılmasına rağmen, hala hız ve alan açısından ilerleme kaydedilebileceğini gördük.

Bu tezde, STA_RRA ve değişik türevlerinden hız ve alan açışından daha iyi iki yeni işdüzenleyici mimarisi önerilmektedir. Önerilen iş-düzenleyici tasarımlarından birisi minimum alan sonuçlarına odaklanırken, diğeri hız için tasarlanmıştır. Bu tasarımlardaki yenilik, termometre kodlamasında ve öncelik kodlamasında Paralel Prefiks Ağaç yordamlarının kullanılmasıdır. Önerilen ve rakip iş-düzenleyiciler, 8 bit'ten 256 bit'e kadar sentezlenmiştir. Yapılan karşılaştırma çalışmalarında, bizim iş-düzenleyicilerimizin rakip iş-düzenleyicilere göre hız açısından 42% ve alan açısından 22% oranla daha iyi sonuç verdiği görülmüştür.

Anahtar Kelimeler: Değişmez Zaman Paylaşımlı İş-düzenleyiciler, YTS Üretimi, Paralel Prefiks Ağaç Yordamları

TABLE OF CONTENTS

LI	ST OF TABLESix
LI	ST OFFIGURES x
LI	ST OF ABBREVIATIONS xii
1.	INTRODUCTION1
2.	PREVIOUS WORK
	2.1. STA_RRA ARCHITECTURE7
	2.1.1. Simple Priority Encoder10
	2.1.2. Thermometer Encoder12
	2.1.3. N to LOGN Encoder14
	2.2. CHN_RRA_PPE_CONFLICT
	2.3. CHN_RRA_PPE_NONCONFLICT
	2.4. LITERATURE SUMMARY
3.	PROPOSED ARCHITECTURES
	3.1. PPT_RRA_RS ARCHITECTURE
	3.1.1. OR Binary Tree
	3.1.2. Parallel Prefix Tree Pre Thermo Encoder Overview
	3.1.3. Brent Kung PPT_Pre_Thermo Architecture
	3.1.4. Ladner Fisher PPT_Pre_Thermo Architecture
	3.1.5. Kogge Stone PPT_Pre_Thermo Architecture
	3.1.6. Han Carlson PPT_Pre_Thermo Architecture
	3.1.7. Edge Detector
	3.2. PPT_RRA_BT ARCHITECTURE
	3.3. BOW-TIE ARCHITECTURE
	3.4. ENHANCEMENTS ON PREVIOUS WORK
	3.4.1. STA_RRA_N2LOGN Architecture
	3.4.2. STA_PPT_RRA Architecture44
	3.4.3. CHN_PPT_RRA_PPE_Conflict Architecture
	3.4.4. CHN_PPT_RRA_PPE_NonConflict Architecture

4.	AUTOMATIC RTL GENERATION	. 47
	4.1. STA_RRA GENERATION	. 47
	4.2. CHN_RRA GENERATION	. 49
	4.3. PPT_RRA GENERATION	. 50
	4.4. BOW-TIE_RRA GENERATION	. 51
5.	VERIFICATION AND SYNTHESIS METHODOLOGY	. 53
	5.1. VERIFICATION	. 53
	5.2. SYNTHESIS METHODOLOGY	. 56
6.	SYNTHESIS RESULTS	. 59
7.	CONCLUSION AND FUTURE WORK	. 66
RE	FERENCES	. 68
Vľ	ΓΑ	.72

LIST OF TABLES

Table 2.1	:	STA_RRA_N2N data flow	9
Table 2.2	:	8-bit Smple_PE truth table	. 10
Table 2.3	:	Truth table for tothermo block	. 13
Table 2.4	:	Truth table for N2LOGN Encoder	. 15
Table 3.1	:	RR block's truth table	. 40

LIST OF FIGURES

Figure 1.1	:	Block diagram of a conventional crossbar switch	2
Figure 1.2	:	High level block diagram of the scheduler	3
Figure 1.3	:	HOL blocking problem without VOQs and with VOQs	4
Figure 1.4	:	Block diagram of a scheduler/arbiter based on RGA/RG maximal	size
		matching algorithm	4
Figure 1.5	:	32X32 Network switch architecture	5
Figure 2.1	:	STA_RRA_N2N architecture	9
Figure 2.2	:	Thermometer encoding	12
Figure 2.3	:	N2LOGN encoder operation	14
Figure 2.4	:	CHN_RRA_PPE_CONFLICT architecture	16
Figure 2.5	:	Iterative scheduling	17
Figure 2.6	:	CHN_RRA_PPE_CONFLICT architecture's timing paths	17
Figure 2.7	:	CHN_RRA_PPE_NONCONFLICT architecture	19
Figure 2.8	:	Proposed CHN_RRA_PPE_NONCONFLICT architecture	20
Figure 2.9	:	Optimized CHN_RRA_PPE_NONCONFLICT architecture	20
Figure 3.1	:	PPT_RRA_RS architecture	25
Figure 3.2	:	OR_BT with OR gates and OR_BT with NOR-NAND gates	27
Figure 3.3	:	8-bit and 16-bit NOR-NAND OR_BT	27
Figure 3.4	:	CMOS OR gate and CMOS NOR gate	28
Figure 3.5	:	CMOS AND gate and CMOS NAND gate	28
Figure 3.6	:	Taxonomy of PPT topologies	29
Figure 3.7	:	PPT_Pre_Thermo example	30
Figure 3.8	:	BK_PPT_Pre_Thermo structure	31
Figure 3.9	:	LF_PPT_Pre_Thermo structure	32
Figure 3.10	:	Equivalence of NOR—NAND—INV PPT and OR PPT	32
Figure 3.11	:	KS_PPT_Pre_Thermo structure	33
Figure 3.12	:	HC_PPT_Pre_Thermo structure	34
Figure 3.13	:	Edge detector's top level block diagram	34
Figure 3.14	:	Edge detection examples	35
Figure 3.15	:	Edge detection architecture is constructed by AND—INV gates	35
Figure 3.16	:	Edge detector optimization	36
Figure 3.17	:	Optimized edge detector	36

Figure 3.18 :	PPT_RRA_BT architecture	37
Figure 3.19 :	PPT_RRA_BT architecture with simplified multiplexer	38
Figure 3.20 :	BOW-TIE_RRA high level architecture	39
Figure 3.21 :	BOW-TIE_RRA's building blocks	40
Figure 3.22 :	RR block's all possible states	41
Figure 3.23 :	GUNIT block	42
Figure 3.24 :	STA_RRA_N2LOGN architecture	43
Figure 3.25 :	Zero request example	44
Figure 3.26 :	STA_PPT_RRA architecture	44
Figure 3.27 :	CHN_PPT_RRA_PPE_Conflict architecture	45
Figure 3.28 :	CHN_PPT_RRA_PPE_NonConflict architecture	46
Figure 4.1 :	STA_RRA generation	48
Figure 4.2 :	CHN_RRA_PPE_Conflict generation	49
Figure 4.3 :	CHN_RRA_PPE_NonConflict (*OPTIMIZED) generation	50
Figure 4.4 :	PPT_RRA_RS generation	51
Figure 4.5 :	PPT_RRA_BT generation	51
Figure 4.6 :	BOW-TIE generation	52
Figure 5.1 :	Verification strategy	54
Figure 5.2 :	Synthesis methodology	57
Figure 6.1 :	8 Bit RRAs synthesis results	59
Figure 6.2 :	16 Bit RRAs synthesis results	60
Figure 6.3 :	32 Bit RRAs synthesis results	61
Figure 6.4 :	64 Bit RRAs synthesis results	62
Figure 6.5 :	128 Bit RRAs synthesis results	53
Figure 6.6 :	256 Bit RRAs synthesis results	54
Figure 6.7 :	Synthesis results at first positive or zero slack	65

LIST OF ABBREVIATIONS

Round Robin Arbiter	:	RRA
Stanford Round Robin Arbiter	:	STA_RRA
Chinese Round Robin Arbiter	:	CHN_RRA
Programmable Priority Encoder	:	PPE
Binary Tree Search	:	BTS
Parallel Round Robin Arbiter	:	PRRA
Improved Parallel Round Robin Arbiter	:	IPRRA
Switch Arbiter	:	SA
Ping Pong Arbiter	:	PPA
Parallel Prefix Tree Round Robin Arbiter	:	PPT_RRA
Parallel Prefix Tree Round Robin Arbiter Resource Sharing	:	PPT_RRA_RS
Parallel Prefix Tree Round Robin Arbiter Best Timing	:	PPT_RRA_BT
Resource Sharing	:	RS
Round Robin	:	RR
Grant Unit	:	GUNIT
Virtual Output Queue	:	VOQ
OR Binary Tree	:	OR_BT
Simple Priority Encoder	:	Smple_PE
Thermometer Encoder	:	tothermo
Han Carlson	:	НС
Brent Kung	:	ВК
Kogge Stone	:	KS
Ladner Fisher	:	LF
Complementary Metal Oxide Semiconductor	:	CMOS
Set Max Area	:	SMA
Compile Incremental	:	CI
Map Effort High	:	MEH
No Constraint	:	NC

1. INTRODUCTION

As chip manufacturing technology shrinks toward sub-nanometers, a chip die will comprise more and more of processing blocks. Interconnection, communication, and utilization of shared resources of these blocks are getting more complicated for System-on-Chip (SoC). This complex structure introduces an important challenge to the designer: fast and fault-free on-chip communication. When processing blocks access a shared resource simultaneously, arbitration of these clients has to be ensured. Priority encoders (PE) and arbiters are widely used to allow only one block to access a shared resource. Priority encoding scheme always selects the highest precedence as defined by a priority sequence. As a result of this static scheme, unfairness is revealed. This unfairness is also called starvation. Conversely, programmable priority encoder (PPE) provides a non-static scheme to alter the priority sequence during an operation. These are the core functional blocks of round robin arbiters (RRA). If an arbiter is designed in round-robin fashion, the priority of the system is altered in every cycle, and round-robin arbiter selects the highest priority starting from the last selected request. This technique almost always guarantees fairness.

RRAs are widely used in network switches and routers. Network switches and routers consist of crossbar switches as the internal switching fabric, as shown in Figure 1.1. A crossbar switch comprises of three macro blocks: Input FIFO buffer, arbiter/scheduler, and a crossbar fabric core.

Switch scheduling algorithms are important aspects to implement high speed network switches. These algorithms are implemented in schedulers/arbiters. A scheduling algorithm selects input packets and generates proper control signals for crossbar fabric to set up conflict free paths between input ports and output ports. Then, the crossbar core transfers the requests or packets according to granted control signals. Hence, in order to ensure high speed and fairness, a crossbar switch requires an intelligent, centralized, and conflict free scheduler/arbiter algorithm.

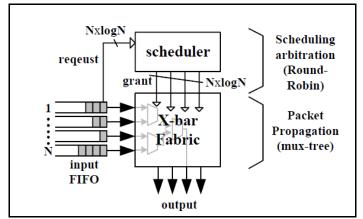


Figure 1.1: Block diagram of a conventional crossbar switch

Most crossbar schedulers are implemented in round robin fashion to prevent starvation of input ports. For example, a well-known design is applied in Stanford University's Tiny Tera prototype. The scheduler combines *i*SLIP unicast scheduling algorithm and mRRM multicast scheduling algorithm. The resulting algorithm is almost identical to the ESLIP algorithm. A brief overview of this algorithm is explained in the following paragraph and a high level block diagram of the scheduler is shown in Figure 1.2. This scheduling algorithm consists of three steps; Request, Grant, and Accept (RGA).

- Request: Each unmatched input sends a request to the destination output as pointed out by the queued cell. In this step requests or packets are just transferred to grant arbiters.
- Grant: Each unmatched output acknowledges one of the requests is received.
 Round-robin schedule starting from the highest priority element.
- *3. Accept:* Each input accepts one of the received grants to establish the connection. Round-robin schedule starting from the highest priority element.

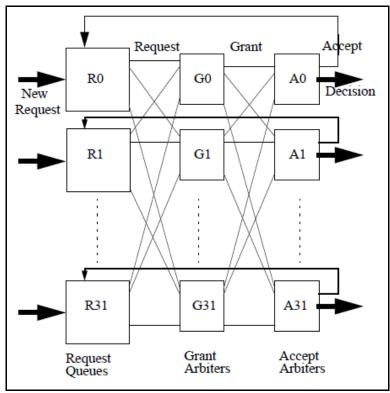


Figure 1.2: High level block diagram of the scheduler

In order to assuage the Head-of-Line (HOL) blocking problem Virtual Output Queues (VOQs) are employed as request queues. When a single FIFO input queue is used for each input port, HOL blocking problem occurs. A request/packet at the end of the queue is blocked from being destined to its corresponding output port because of port connection, and thus the entire FIFO is blocked. HOL problem is eliminated by using separate input queues for each input-output port pair.

HOL blocking problem and its solution is shown in Figure 1.3. "For this example, assume that input port 1 is granted when output port contentions occur. Each numbered rectangle in Figure 1.3 corresponds to a packet with the destination specified by the number. Thus, the packet numbered '1' indicates that this packet is destined to output port 1. Without VOQs case, packet 1 in the queue at input port 0 is blocked by packet 0 located at the head of the queue, even though output port 1 is available at this point. Therefore, only packet 0 is sent to output port 1 in the current cycle. To remove HOL blocking, multiple VOQs are placed at input ports. In the VOQs case, packet 1 at VOQ (0, 1) is forwarded to output port 1 simultaneously as packet 0 at VOQ (1, 0) is delivered to output port 0. Consequently, multiple packets

can be delivered to the appropriate unique destinations by employing VOQs." (Shin 2003, pp.23)

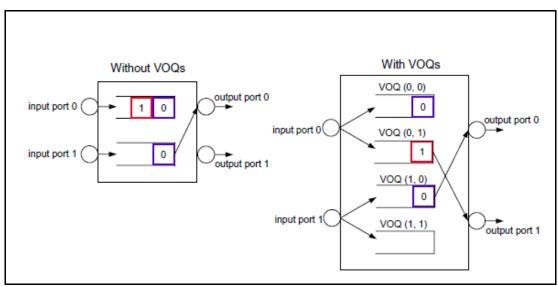


Figure 1.3: HOL blocking problem without VOQs and with VOQs

Detailed block diagram of scheduler based on RGA maximal size matching algorithm is shown in Figure 1.4 and 32X32 network switch example is shown in Figure 1.5. These figures show the scheduler blocks and how it works in a network switch/crossbar switch.

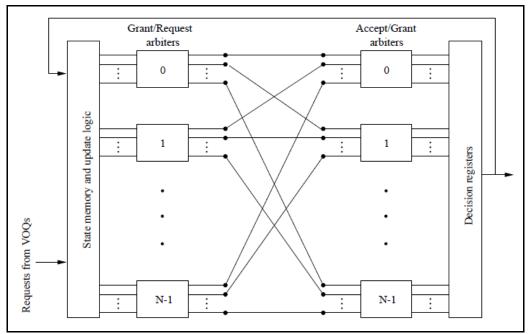


Figure 1.4: Block diagram of a scheduler/arbiter based on RGA/RG maximal size matching algorithm

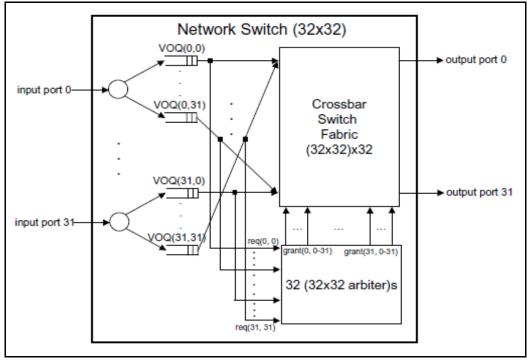


Figure 1.5: 32X32 Network switch architecture

Gupta's and McKeown's work combines the request and grant steps of the scheduler. The decision feedback information is provided by accept arbiters. This feedback information performs as a pointer. The scheduler uses this information to mask off requests from already matched inputs and outputs in successive iterations, and starts round robin schedule from the highest priority element.

A comprehensive explanation of pipelined implementation of the scheduler is explained in Gupta's and McKeown's paper. This thesis is not focused on scheduling algorithms so details of algorithms are omitted.

In this thesis, two new round-robin arbiter logic architectures are proposed by taking parallel prefix tree algorithms into consideration. Register Transfer Level (RTL) generators, Application Specific Integrated Circuit (ASIC) synthesis, and benchmarks of all RRA architectures are represented.

A brief overview of the previous work is provided in Chapter 2. Conventional RRA design and its macro blocks, most recent architectures based on conventional design, and other round robin arbitration schemes are represented. Chapter 3 introduces

hardware implementation and breakthrough of related work. In this chapter, a new macro block is embedded into the rival architectures to strengthen them. Verilog hardware description language (HDL) register transfer level (RTL) code generators of all architectures are explained in chapter 4. In Chapter 5, verification and synthesis techniques are represented in detail. Synthesis results are illustrated in Chapter 6. Chapter 7 presents some concluding remarks.

2. PREVIOUS WORK

A well-known fast crossbar scheduler design is implemented by Pankaj Gupta and Nick McKeown from Stanford University. They published their arbiter architecture in 1999. This design is called conventional round robin arbiter or conventional round robin scheduler in many research papers. In this thesis, this architecture is called Stanford Round Robin Arbiter (STA_RRA). Seven years later, Gao Xiaopeng, Zhang Zhe, and Long Xiang tried to enhance the conventional round robin scheduler different design. They proposed two architectures: PPE Conflict and PPE NonConflict. These architecture modified two names are as CHN_RRA_PPE_Conflict and CHN_RRA_PPE_NonConflict. This work is explained in Sections 2.2 and 2.3 in detail. A brief overview of other RRA architectures in literature is represented in Section 2.4.

2.1. STA_RRA ARCHITECTURE

Top level block diagram of STA_RRA architecture is shown in Figure 2.1. It is comprised of the following macro blocks: Simple Priority Encoder (Smpl_PE), Simple Priority Thermo Encoder (Smpl_PE_thermo), thermometer encoder (tothermo), and n to log₂N (N2LOGN) encoder for update path. A simplified multiplexer is also used to select the appropriate Priority Encoder's (PE) grant.

Data flow of the architecture is explained as follows. When a new request is asserted from request queues, and the feedback information of priority pointer is forwarded from thermometer encoder, these two signals go into an AND gate to mask off new requests with respect to priority pointer or accepted request in the previous iteration. This new/masked request is connected to Smpl_PE_thermo block. This block is identical to Smpl_PE. Its function is to perform fixed priority encoding.

In PEs, each of the requester has a fixed priority and PE gives its grant to the active requester which has the highest priority. This is a static and unfair scheme. It always gives the shared resource's usage authority to the most powerful requester or in other words, the highest priority requester. The highest priority requester depends on

priority direction and your preference. You can change priority direction by flipping your architecture scheme horizontally. In some research work, Least Significant Bit (LSB) is selected as highest priority requester. For example, Gupta and McKeown selected LSB as the highest priority requester. However, in this thesis, Most Significant Bit (MSB) is selected as the highest priority requester for all architectures. For instance, if the request vector is declared as req[N-1:0], req[N-1] is defined as the highest priority. If req[N-1] is active, grant[N-1] gets the grant. Else, if req[N-2] is active, grant[N-2] is asserted, and so on.

Smple_PE_thermo block has two outputs which are grant (Gnt_smpl_PE_thermo) and any (anyGnt_smpl_PE_thermo). Gnt_smpl_PE_thermo is the output of priority encoding operation. anyGnt_smpl_PE_thermo bit is the OR of Smple_PE_thermo block's all inputs. This bit gives us information on whether there is any unmatched request in at least one bit position or not. On the other hand, unmasked request moves to Smpl_PE. This block performs priority encoding operation and outputs its grant (Gnt_smpl_PE) as well. anyGnt_smpl_PE_thermo is inverted and ANDed with Gnt smpl PE. Therefore, if an unmatched request exists from the previous iteration, anyGnt_smpl_PE_thermo disables Gnt_smpl_PE. As а matter of fact anyGnt_smpl_PE_thermo is used as a multiplexer select; Gnt_smpl_PE and Gnt_smpl_PE_thermo are this multiplexer's inputs. In some papers this operation is shown as carried on by a normal multiplexer rather than a simplified multiplexer. Pankaj and McKeown simplified this block in order to reduce the loading on the select signal by half.

Final output is obtained by OR operation of Gnt_smpl_PE_thermo and Gnt_smpl_PE_masked. Final output of this iteration should be forwarded to tothermo block for next iteration. The next iteration uses the previous iteration's final output as a priority pointer. This pointer is generated by tothermo block. This block executes thermometer encoding operation. Thermometer encoding masks all accepted requesters of previous iterations. Then, tothermo's output and a new request is ANDed to mask off the new request's accepted bit positions. All iterations are performed in this way.

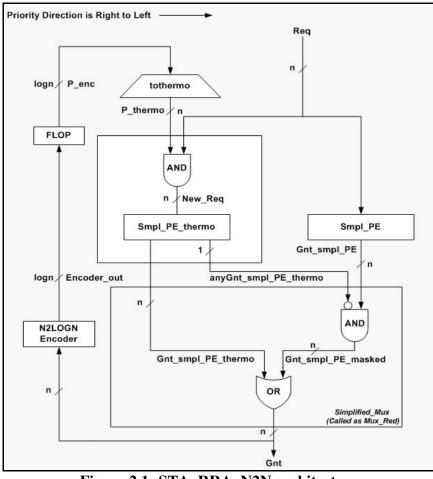


Figure 2.1: STA_RRA_N2N architecture (Stanford Round Robin Arbiter N to N Smpl_PEs)

According to top level block diagram and signal flow description, a sample test case is shown for 8 bits STA_RRA in Table 2.1.

Table 2.1: STA_RRA_N2N data flow									
Initial Iteration									
P_thermo	\rightarrow	00000000	Gnt_smpl_PE	\rightarrow	00100000				
Request	\rightarrow	00101001	Gnt_smpl_PE_masked	\rightarrow	00100000				
New_Req	\rightarrow	00000000	Gnt	\rightarrow	00100000				
Gnt_smpl_PE_thermo	\rightarrow	00000000	Encoder_out	\rightarrow	101				
anyGnt_smpl_PE_thermo	\rightarrow	0	P_enc	\rightarrow	101				
		Second I	teration						
P_thermo	\rightarrow	00011111	Gnt_smpl_PE	\rightarrow	10000000				
Request	\rightarrow	10100101	Gnt_smpl_PE_masked	\rightarrow	00000000				
New_Req	\rightarrow	00000101	Gnt	\rightarrow	00000100				
Gnt_smpl_PE_thermo	\rightarrow	00000100	Encoder_out	\rightarrow	010				
anyGnt_smpl_PE_thermo	\rightarrow	1	P_enc	\rightarrow	010				

9

2.1.1. Simple Priority Encoder (Smpl_PE and Smpl_PE_thermo)

Detailed definition of simple priority encoding is described in the previous section. This encoding is a fixed and static encoding that always grants the highest priority requester. Its truth table is shown in Table 2.2.

									_		-		-	-	
in[7]	in[6]	in[5]	in[4]	in[3]	in[2]	in[1]	in[0]	out[7]	out[6]	out[5]	out[4]	out[3]	out[2]	out[1]	out[0]
1	х	х	х	х	х	х	х	1	0	0	0	0	0	0	0
0	1	х	х	х	х	х	х	0	1	0	0	0	0	0	0
0	0	1	х	х	х	х	х	0	0	1	0	0	0	0	0
0	0	0	1	х	х	х	х	0	0	0	1	0	0	0	0
0	0	0	0	1	х	х	х	0	0	0	0	1	0	0	0
0	0	0	0	0	1	х	х	0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	х	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2.2: 8-bit Smple_PE truth table

From table 2.2, following equation list for 8-bit Smpl_PE can be derived.

```
out[0] = ~in[7]&~in[6]&~in[5]&~in[4]&~in[3]&~in[2]&~in[1]&in[0];
out[1] = ~in[7]&~in[6]&~in[5]&~in[4]&~in[3]&~in[2]&in[1];
out[2] = ~in[7]&~in[6]&~in[5]&~in[4]&~in[3]&in[2];
out[3] = ~in[7]&~in[6]&~in[5]&~in[4]&in[3];
out[4] = ~in[7]&~in[6]&~in[5]&in[4];
out[5] = ~in[7]&~in[6]&in[5];
out[6] = ~in[7]&in[6];
out[7] = in[7];
```

This implementation has some drawbacks. When we feed this code into a synthesis tool, a ripple carry chain is inferred. This scheme produces significant timing problems and it is not area efficient. Hence, Smpl_PE should be implemented in a better way.

In order to eliminate the aforementioned drawbacks, we applied two techniques and optimized Smpl_PE. This optimization also strengthened our rivals' ASIC synthesis results. In these optimizations, binary tree algorithm/technique is used to reduce the logic level of the architecture. Also, a smart pre-computation/factoring method reduced the area dramatically. These two modifications can be explained as follows.

Firstly, all HDL files are coded with respect to binary tree fashion. This implementation model lowers the logic depth from n to log_2N . The sample HDL code for this implementation is shown below.

```
wire temp_0_7_6 = ~req[7] & ~req[6];
wire temp_0_5_4 = ~req[5] & ~req[4];
wire temp_0_7_4 = temp_0_7_6 & temp_0_5_4;
wire temp_0_3_2 = ~req[3] & ~req[2];
wire temp_0_1_0 = ~req[1] & req[0];
wire temp_0_3_0 = temp_0_3_2 & temp_0_1_0;
wire temp_0_7_0 = temp_0_7_4 & temp_0_3_0;
assign out[0] = temp_0_7_0;
```

The code snippet above is written for first grant bit's computation. This bit's computed values such as temp_0_7_6, temp_0_5_4, temp_0_7_4, etc... are used for computation of other grant bit positions. These pre-computed values are used with binary tree methodology. Eventually, design's area cost is alleviated and timing is improved. This implementation technique's HDL code snippet is shown below.

```
//FIND PRECOMPUTED VALUES and USE THEM FOR INDEX 1
wire temp_1_1_0 = temp_0_7_4 & temp_0_3_2;
wire temp_1_1 = req[1];
wire temp_1_2_0 = temp_1_1_0 & temp_1_1_1;
assign out[1] = temp_1_2_0;
//FIND PRECOMPUTED VALUES and USE THEM FOR INDEX 2
wire pre_temp_2_3_2 = ~req[3] & req[2];
wire temp_2_1_0 = temp_0_7_4 & pre_temp_2_3_2;
assign out [2] = temp_2_1_0;
//FIND PRECOMPUTED VALUES and USE THEM FOR INDEX 3
wire temp_3_1_0 = temp_0_7_4 \& req[3];
assign out [3] = temp_3_1_0;
//FIND PRECOMPUTED VALUES and USE THEM FOR INDEX 4
wire pre_temp_4_5_4 = ~req[5] & req[4];
wire temp_4_1_0 = temp_0_7_6 & pre_temp_4_5_4;
assign out [4] = temp_4_1_0;
```

```
//FIND PRECOMPUTED VALUES and USE THEM FOR INDEX 5
wire temp_5_1_0 = temp_0_7_6 & req[5];
assign out[5] = temp_5_1_0;
//FIND PRECOMPUTED VALUES and USE THEM FOR INDEX 6
wire pre_temp_6_7_6 = ~req[7] & req[6];
wire temp_6_1_0 = pre_temp_6_7_6;
assign out[6] = temp_6_1_0;

→ FIND PRECOMPUTED VALUES and USE THEM FOR INDEX 7
wire temp_7_1_0 = req[7];
assign out[7] = temp_7_1_0;
```

2.1.2. Thermometer Encoder (tothermo)

Thermometer encoding performs log_2N -bit to n-bit transformation. Its equation can be defined in this way:

$$out |index| = 1$$
 if and only if $index < value(in)$ for all $0 \le i < n$ (2.1)

This transformation works similarly to a normal thermometer operation. It takes log_2N -bit input and increases thermometer level according to this input value. For example, 8-bit thermometer encoder takes 3-bit wide input vector and outputs an 8-bit wide vector. In a way, this output vector designates the level number of a thermometer. Its indicator level starts from 0 and ends up at 7. If thermometer encoder gets 101 as an input vector, it produces five piece of logic 1. In short, it transforms input 101 to 00011111 as an output. This operation is clearly shown in Figure 2.2.

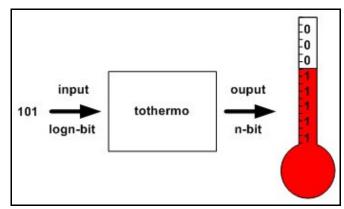


Figure 2.2: Thermometer encoding

Truth table for 3-bit to 8-bit thermometer encoding block is shown in Table 2.3.

in [2:0]	out[7:0]
000	0000000
001	0000001
010	00000011
011	00000111
100	00001111
101	00011111
110	00111111
111	0111111

Table 2.3: Truth table for tothermo block

This transformation's equations are shown below.

```
out7 = 0
out6 = in2.in1.in0
out5 = in2.in1
out4 = in2.(in1+in0)
out3 = in2
out2 = in2+in1.in0
out1 = in2+in1
out0 = in2+in1+in0
```

This algorithm is used to generate thermometer encoder code for any number of n. We designed STA_RRA arbiter's thermometer encoder with respect to this algorithm. Algorithm's code snippet shows that each output bit is either ORed or ANDed not greater than log_2N bit width. This is also stated in Gupta's and McKeown's work. This algorithm's parameterized Verilog HDL code snippet is as follows.

```
parameter log_n = 3;
parameter n=(1<<log_n);</pre>
always @(thermo_in) begin
  pow2 = \{ (log_n) \{ 1'b0 \} \};
  thermo_out = \{(n) \{1'b0\}\};
  tmp = 0;
  pow2[0] = 1'b1;
  thermo_out[0] = 1'b0;
  for(i = 0; i < log_n; i = i+1) begin
    for (j = 0; j < pow2; j = j+1) begin
      tmp = thermo_out[j];
      thermo_out[j] = tmp | thermo_in[i];
      thermo_out[j+pow2] = tmp & thermo_in[i];
    end
    pow2 = pow2 + pow2;
  end
end
```

The purpose of thermometer encoding is to mask off new request bit positions which are previously accepted. This is also explained is Section 2.1 in detail.

2.1.3. N to LOGN Encoder (N2LOGN Encoder)

STA_RRA architecture outputs n-bit grant output. This grant output is also used to generate priority pointer for the next iteration. Typically, STA_RRA's grant vector is one-hot –only one bit position is high for each arbitration iteration. The active bit position shows us the accepted requester. Therefore, in the next iteration, priority precedence must start from this bit position.

In previous sections we mentioned that priority pointer generation is accomplished by thermometer encoder macro block. However, this block takes log₂N-bit input, then executes thermometer encoding algorithm in order to output n-bit priority pointer. For this reason, there must be a logic block between the thermometer encoder block and the grant output of STA_RAA. This block's core function is to convert n bit grant vector to log₂N bit vector.

Simply put, the N2LOGN encoder block outputs the active bit's index value of its nbit one-hot input vector. In other words, the N2LOGN encoder states the accepted bit-position – or accepted requester's index to thermometer encoder. Its log_2N bit output is taken by thermometer encoder. N2LOGN block's operation is shown in Figure 2.3.



Figure 2.3: N2LOGN encoder operation

Gupta's and McKeown's work did not consist of priority pointer feedback information part. Furthermore, we did not find any explanation about this macro block's implementation from other research work. In some papers this block is stated as "encode" or "binary_enc". Therefore, we implemented this block by taking into consideration the binary tree structure. This is the best way to reduce logic depth and to speed up timing.

Table 2.4: Truth table for N2LOGN Elicouer									
out[2]	out[1]	out[0]	input bit positions						
0	0	0	0						
0	0	1	1						
0	1	0	2						
0	1	1	3						
1	0	0	4						
1	0	1	5						
1	1	0	6						
1	1	1	7						

Truth table of N2LOGN encoder is shown in Table 2.4.

 Table 2.4: Truth table for N2LOGN Encoder

According to Table 2.4 following equations are derived:

out[0] = in[1] | in[3] | in[5] | in[7] out[1] = in[2] | in[3] | in[6] | in[7] out[2] = in[4] | in[5] | in[6] | in[7]

In order to reduce the logic depth and timing bottleneck, these equations are coded in binary tree style. Verilog HDL code snippet below is an example of N2LOGN encoder implementation.

```
wire out_0_0 = in[1] | in[3];
wire out_0_1 = in[5] | in[7];
wire out_1_0 = in[2] | in[3];
wire out_1_1 = in[6] | in[7];
wire out_2_0 = in[4] | in[5];
wire out_2_1 = in[6] | in[7];
//output[0] assignments
wire out_0_0to1 = out_0_0 | out_0_1;
//output[1] assignments
wire out_1_0to1 = out_1_0 | out_1_1;
//output[2] assignments
wire out_2_0to1 = out_2_0 | out_2_1;
assign out[0] = out_0_0to1;
assign out[1] = out_1_0to1;
assign out[2] = out_2_0to1;
```

2.2. CHN_RRA_PPE_CONFLICT

This round robin arbiter architecture is generated by three Chinese researchers: Gao Xiaopeng, Zhang Zhe, and Long Xiang. Therefore, we used CHN prefix for this architecture. They proposed two different arbitration schemes. These architectures are similar to STA_RRA architecture. One of the proposed round robin arbitration architectures is PPE_CONFLICT. Consequently, this design is called CHN_RRA_PPE_CONFLICT, as shown in Figure 2.4. In this design, the round robin arbitra architecture is divided into two paths. These are grant path and update path. Grant path is shown from request to grant. On the other hand, update path is a feedback path that forwards the priority pointer for next iteration.

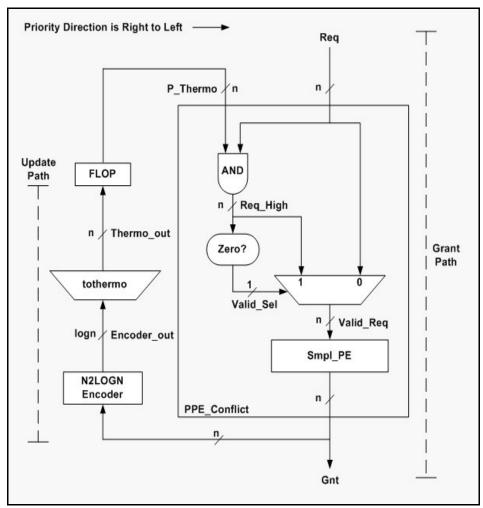


Figure 2.4: CHN_RRA_PPE_CONFLICT architecture (Chinese Round Robin Programmable Priority Encoder Conflict)

In conflicting virtual channel routers, arbiters work in an iterative way. Arbiters located at inputs and outputs perform Grant and Accept operations in a single clock cycle alternatively (in introduction section Request-Grant-Accept (RGA) pipeline is explained). Results of these two operations are dependent on each other, as shown in Figure 2.5. Grant/Update paths should be executed in two clock cycles. Therefore, the critical path in this scheme is the grant path; as the update path can be executed in parallel with respect to grant path. For this reason tothermo block is moved from grant path to update path.

Grant	Accept	Grant	 Accept
Grant Path(O)	Update Path(O) Grant Path (I)	Grant Path(O) Update Path (I)	 Update Path(O)
	Grant Patri (I)		

Figure 2.5: Iterative scheduling

CHN_RRA_PPE_CONFLICT architecture is divided into two timing paths as grant path and update path. Timing Path 1 is equals grant path; which is critical path for Conflicting VCR. Combo Logic 2's delay is equal to update path's delay. Timing Path 2 is comprised of both grant and update paths. In order to minimize the clock cycle, timing balance of grant and update paths is very crucial for Conflicting VCRs.

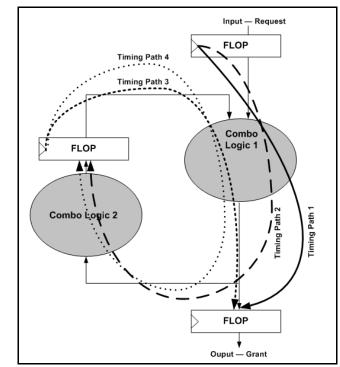


Figure 2.6: CHN_RRA_PPE_CONFLICT architecture's timing paths

One of the distinct points is the location of flip-flops. Flops are located after thermometer encoder. If their bit width is broadened, this may cause a negative effect on area. Also, flops can be located between N2LOGN encoder and thermometer encoder. Thus, sequential logic area can be lowered. On the other hand, this modification destabilizes timing balance between grant path and update path, and also maximizes the iteration cycle of arbitration.

In CHN_RRA_PPE_CONFLICT architecture Resource Sharing (RS) technique is implemented to reduce the combinational area cost. Only one Smpl_PE is used rather than two Smple_PEs. When masking operation is done from AND gate, the resulting output (Req_High) and request go to multiplexer's inputs. Then, the valid request is chosen by multiplexer select (Valid_Req). Valid_Req is just the bitwise OR of Req_High's all bits. The selected output is forwarded to Smpl_PE for final grant operation.

2.3. CHN_RRA_PPE_NONCONFLICT

This architecture is designed for non-conflicting VCRs. We called this design: CHN_RRA_PPE_NONCONFLICT, as shown in Figure 2.7. In non-conflicting VCRs, scheduling operation takes one cycle, and it is carried out by arbiters which are located at outputs. In a pipelined router, PPEs at outputs execute the arbitration cycle by cycle. Hence, priority pointer has to be ensured before next iteration start-up. PPE must succeed grant and update operations in a single clock cycle. Thus, critical path of PPE comprises both the grant and update paths.

In this architecture, grant and update operations are accomplished in parallel. Grant/Update path is also a timing path for synthesis tool. When we implement this architecture's ASIC synthesis, design compiler (DC) accepts grant/update path as a timing path. This design architecture is more realistic with respect to digital design timing concepts. Therewithal, its timing results should be better than CHN_RRA_PPE_CONFLICT architecture. On the other hand, RS technique is applied to increase the design's area efficiency. This design consists of only one Smple_PE rather than two Smpl_PEs. RS methodology is described in the previous

section in detail. The drawback of RS technique is to increase the critical path of the grant/update path. "Zero?" block is just a binary OR tree and it adds extra log₂N stages and decelerates timing of PPE.

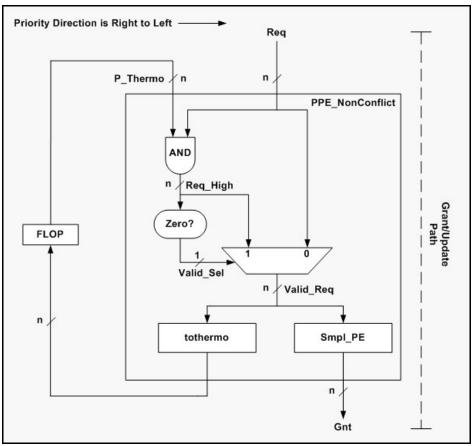


Figure 2.7: CHN_RRA_PPE_NONCONFLICT architecture (Chinese Round Robin Programmable Priority Encoder NonConflict)

Maybe the most conspicuous point in this architecture is the tothermo block because this thermometer encoder performs n-to-n conversion. In their implementation, this tothermo block combines Smple_PE, N2LONG encoder, and the tothermo block of PPE design. Actually, this implementation decreases the speed of the architecture. Hence, we implemented this block in a different way. We used one of the Parallel Prefix Tree (PPT) algorithms — Han Carlson (HC) tree. We used OR gates in HC tree's nodes, then finally we shifted tree's output to the right by one. One bit right shifting is just a wiring modification; it does not contain any logic element. In essence, this method is one of the core novelties for our proposed designs.

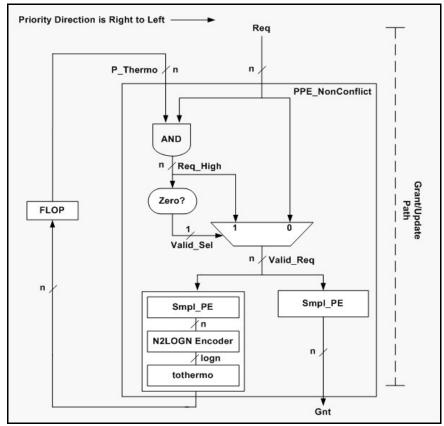


Figure 2.8: Proposed CHN_RRA_PPE_NONCONFLICT architecture

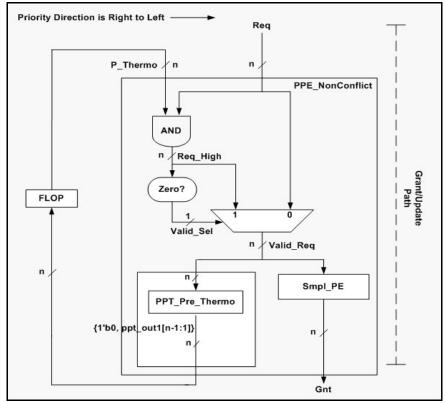


Figure 2.9: Optimized CHN_RRA_PPE_NONCONFLICT architecture

We applied this technique in this architecture to make our rival stronger. This technique is area efficient and a fast technique, which is much better than the combination of Smpl_PE — N2LOGN Encoder — tothermo blocks. This obvious improvement can be understood when we compare Figure 2.8 and Figure 2.9. You can easily see that the combination of Smpl_PE — N2LOGN Encoder — tothermo blocks significantly increases the area and slows down the speed of the RRA. This methodology is not proposing a novelty. Moreover, combinational path from request to flops' input is longer than CHN_RRA_PPE_CONFLICT architecture. However, if our PPT technique is applied rather than the Smpl_PE — N2LOGN Encoder — tothermo combination, area and timing results can be improved dramatically. Hence, in this thesis we applied our technique for this architecture to make our rival stronger. This ensures a fair benchmark. In Figure 2.9 PPT_Pre_Thermo block contains a HC tree and its nodes are comprise OR gates. Details of our novel approach and architecture will be described in later chapters.

2.4. LITERATURE SUMMARY

In literature there are many round robin arbitration algorithms and several design work are proposed. On the algorithm side, PIM, *i*SLIP, DDRM, FIRM, SSR, and PPA are the examples of the most practical scheduling algorithms. They are all iterative algorithms that approximate a maximum matching by finding a maximal size matching. They comprise of three or two steps/iterations, which were mentioned in previous sections. These steps are Request—Grant—Accept (RGA) or Request—Grant (RG). This algorithms' main goal is to ensure fairness. Also, round robin arbiter designs are carried out with respect to this criterion.

The most notable round robin architecture — STA_RAA is proposed by Gupta and McKeown which we explained in previous sections.

Some researchers have proposed modified version of STA_RRA. One of them is published by Savin C.E., McSmythurs T., and Czilli J. in 2004. They used Binary Tree Search (BTS) technique to minimize the area and maximize the speed. Their

proposed architecture has $(\log_2 N + 4)$ logic level, $(n\log_2 N + 7n - 6)$ combinational gate count and n-bit flop.

Other STA_RRA modifications are proposed by Gao X., Zhang Z., and Long X. in 2006 which we called CHN_RRA. Their proposed RRA architectures are depicted in former sections. Detailed synthesis results of their architecture's are given in Chapter 6.

On top of STA_RRA and its variants, different design architectures are proposed. The most recognized ones are Ping Pong Arbiter (PPA), Switch Arbiter (SA), Parallel Round Robin Arbiter (PRRA) and Improved Parallel Round Robin Arbiter (IPRRA).

PPA architecture is proposed by Chao H.J., Lam C.H., and Guo X. in 1999. PPA has O(log₂N) level tree structure and O(log₂N) gate delay. This round robin arbiter architecture performs the round robin arbitration rule if and only if all requests are available in each cell slot. If there are less than N request available, at that time unfairness occurs. We can examine this situation according the following example. This example is also mentioned in PPRA—IPPRA paper. Assume that N/2 + 1 input ports repeatedly serve requests in a pattern. Under this condition, one input port's request is captured by one-half of the tree. At the same time, the other half-of the tree captures the remaining input ports' requests. As a result of this situation, this round robin arbiter grants the one input port more than N/2 times more than each of the remaining N/2 input ports. This example demonstrates the unfairness of PPA design. Also, we can claim that PPA's scheduling algorithm performance is worse than iSLIP and mRRM, which are the algorithms of STA_RAA.

Other round robin arbiter design — Switch Arbiter (SA) is proposed by Shin E.S., Mooney V.J. III, and Riley G.F. in 2002. This architecture is designed with the same concept of PPA. This architecture is formed by a tree structure composed of 4x4 SA nodes. These nodes comprise of a flop, 4 PE, a 4-bit ring counter, five 4-input OR gates, and four 2-input AND gates. Some of the research work such as PPRA— IPPRA paper states and benchmarks that SA architecture is faster than the other architectures. On the other side, architectural complexity and unfairness for nonuniformly distributed requests are its drawbacks. For example, in a 64X64 SA: if request signals (req[0] to req[31]) are asserted from the top half of the tree and only one request — req [32] is asserted from the bottom half of the tree, then req [32] is granted thirty-two times while each of 32 request signals (req[0]-req[31]) are granted only once in sixty-four consecutive cycles. This unfairness is as the same as PPA architecture's unfairness.

The most recent and remarkable round robin arbiter architectures are PRRA and IPRRA. These architectures are presented by Zheng S. Q. and Yang M. in January 2007. The proposed architectures are constructed by a recursive binary tree structure. The hardware implementation of these architectures is based on simple binary tree search algorithm. They claimed that IPRRA achieved 30.8% timing improvement and 66.9% area improvement over PPE design.

3. PROPOSED ARCHITECTURES

In this chapter, three different RRA architectures are represented. One of the RRA architectures has a similar appearance as a BOW-TIE, and it is entirely different than STA_RRA architecture. It was inspired by a Silicon Valley engineer who decided to stay anonymous. This architecture is still in its development phase; it should be enhanced in the future. The other two proposed RRA architectures are based on STA_RRA architecture. However, in these architectures we constructed a new block which executes pre-thermometer encoding and pre-priority encoding operations. Hence, critical path is shortened by using this new logic block. In the following sections details of proposed RRA architectures are explained.

3.1. PPT_RRA_RS ARCHITECTURE

The full name of this architecture is Parallel Prefix Tree Round Robin Arbiter Resource Sharing and we simply called it PPT_RRA_RS. This architecture is applied for eliminating the area cost of the RRA. In order to diminish RRA's area, we used Resource Sharing (RS) method. This architecture has one PPT_Pre_Thermo block rather than two Smple_PE's, as shown Figure 3.1. This block is constructed by Parallel Prefix Tree (PPT) topologies. In this work, we used four different PPT topologies. These are Ladner Fisher (LF), Kogge Stone (KS), Han Carlson (HC), and Brent Kung (BK) topologies. These tree structures have internal nodes for each stage and any operations which have associative property could be used in these nodes. In order to accomplish pre-thermometer encoding operation, OR gate is placed in these nodes. The resulting structure performs pre-thermometer encoding operation. If this block's output is shifted right by one unit, actual thermometer encoder output can be obtained. 1-bit right shifting is just a wiring operation and it does not contain any logic elements. At this level, we have obtained "shifted pre_thermo_out" and "pre_thermo_out" signals. These two signals are routed to Edge Detector (ED) block to generate grant output of the RRA architecture. ED block has one level logic depth; this decreases the critical path of the architecture. ED's function is to output the final grant output by using "pre_thermo_out" and "shifted pre_thermo_out" signals. In fact the shifted signal is used as a priority

pointer for next iterations. This architecture is used as the same scheduling algorithm as STA_RAA. Therefore, fairness criterion is better with respect to that of PPA and SA architectures which are proposed in literature formerly. This structure is better than our rivals with respect to the area criterion. Especially, BK topology for PPT_Pre_Thermo block provides significant area reduction. Also, this architecture has a competent speed performance. Its ASIC synthesis results and comparison against its rivals are represented in next chapters. Building blocks of PPT_RRA_RS such as OR Binary Tree (OR_BT), PPT_Pre_Thermo, and ED are explained in next sections.

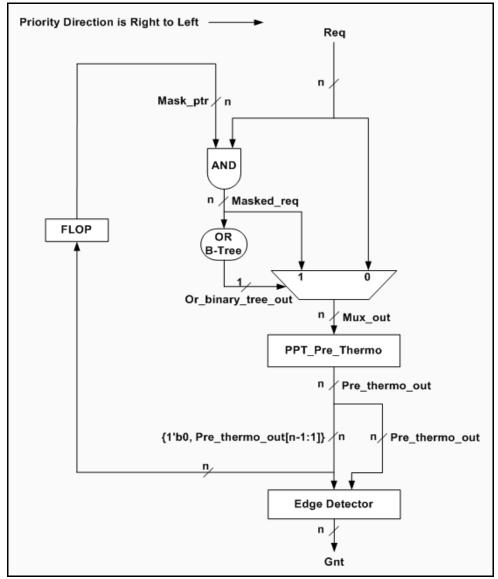


Figure 3.1: PPT_RRA_RS architecture

3.1.1. OR Binary Tree (OR_BT)

OR_BT block performs OR operation. It gets masked request (Masked_req) as an input and it outputs a select signal for multiplexer to select appropriate signal for PPT_Pre_Thermo block.

One bit right shifted version of PPT_Pre_Thermo signal is just a priority pointer (Mask_ptr) and it masks the matched requester in an ongoing iteration. In the next iteration, priority pointer is ANDed with a new request to mask matched bit positions for new request signal. This masking operation's output is called as Masked_req. It contains information on whether or not there is any unmatched requester available with respect to previous iterations or not. If all bit positions of Masked_req are zero, OR_BT block's output will be '0'. This comprises of two different meanings:

- 1- New request signal does not want to grant an unmatched requester. This means that unmatched bit positions with respect to previous iterations are zero. However some matched bit positions are active/one.
- 2- New request signal does not want to grant any of the requester. This means that all bit positions of new request are zero.

On the other hand, if a Masked_req signal has one or more active bits, at this time we can conclude that the new request signal wants to grant unmatched requesters, and OR_BT block outputs '1'.

OR_BT block is playing a key role to simplify RRA architecture's area. It controls the multiplexer and routes Masked_req or new request signals to PPT_Pre_Thermo block. For this reason, this architecture uses only one block rather that two Smple_PEs. This technique is called Resource Sharing (RS). Thus, we put RS prefix for this architecture's name.

OR_BT block is implemented by using NOR, NAND, and OR gates. This transformation is represented in Figure 3.2. For N bit input, stage number is calculated by log_2N . If stage number is odd, last stage must contain OR gate. On the other hand, if stage number is even last stage must contain NAND gate. It is shown in Figure 3.3.

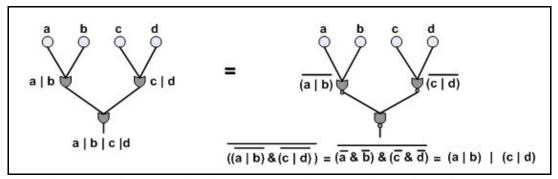


Figure 3.2: OR_BT with OR gates and OR_BT with NOR-NAND gates

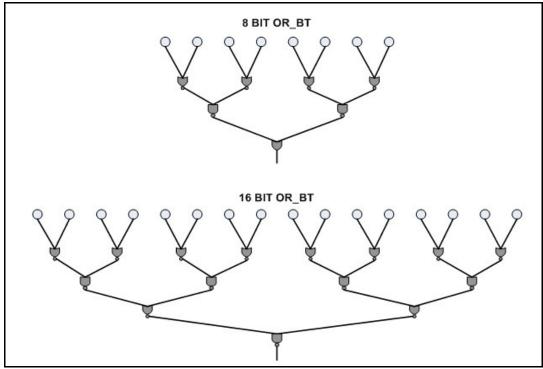


Figure 3.3: 8-bit and 16-bit NOR-NAND OR_BT

The reason for using these gates is to minimize the area cost. In Complementary Metal Oxide Semiconductor (CMOS) technology NOR and NAND gates are more area efficient than OR and AND gates. OR and AND gates have extra inverter part

that effects area efficiency negatively. This negative effect is represented in Figure 3.4 and Figure 3.5.

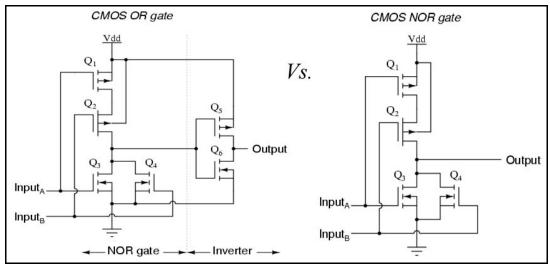


Figure 3.4: CMOS OR gate and CMOS NOR gate

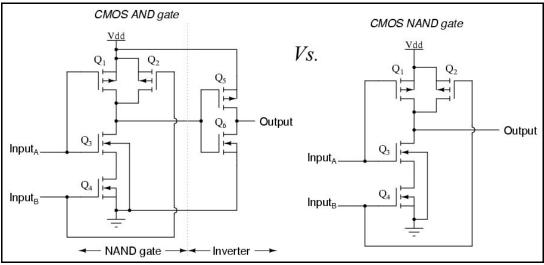


Figure 3.5: CMOS AND gate and CMOS NAND gate

3.1.2. Parallel Prefix Tree Pre Thermo Encoder (PPT_Pre_Thermo) Overview

Our proposed architecture's novelty is based on this logic block. PPT_Pre_Thermo block is constructed by PPT topologies. There are four well-known PPT topologies in literature. These are LF, KS, HC, and BK topologies. All of these topologies are implemented via our RTL generator in this work. Also, any of the PPT topology could be applied in this block. PPT topologies' taxonomy and their drawbacks among each other are stated by David L. Harris in 2003, as shown in Figure 3.6. In

this work, highest priority requester is selected as MSB bit rather than LSB bit. Hence, all topologies are flipped horizontally. Priority direction depends on our preference.

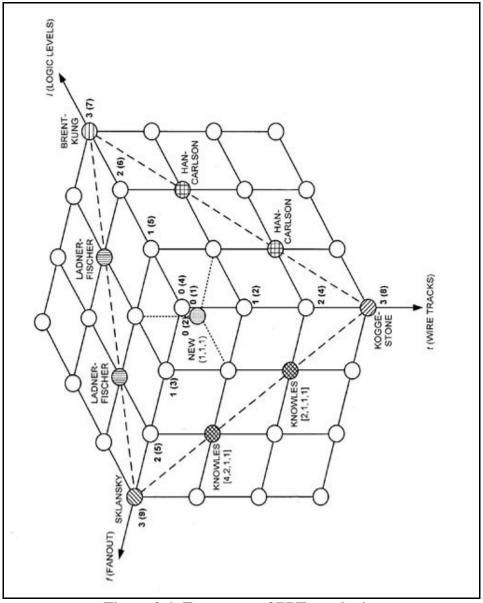


Figure 3.6: Taxonomy of PPT topologies

The function of this block is to prepare an output for thermometer encoding and priority encoding. If its output is 1-bit shifted to the right, this operation gives us thermometer encoder output (priority pointer). This shifting operation does not consist of any logic elements. It only modifies wiring connections. Moreover, if PPT_Pre_Thermo output is forwarded to ED block, ED block performs an edge detection operation to output RRA's grant. This edge detection operation has only

one level logic depth. Essentially, combination of PPT_Pre_Thermo block and ED block forms the Smpl_PE block. Also, one of the superior advantages of PPT_Pre_Thermo is any-bit computation. PPT_Pre_Thermo block's output LSB bit is equal to the OR of all its inputs. Thus, any-bit equals to this block's output's LSB. Operations of PPT_Pre_Thermo are divided into three parts: *pre-tothermo* computation, *pre-Smpl_PE* computation, and *any-bit* computation. So, it can be seen that this new macro block performs multiple operation at the same time. This makes our architecture exceptional with respect to our rivals.

In order to show this blocks functionality, a test scenario is shown in Figure 3.7. In that figure inputs and outputs of the 16-bit PPT_Pre_Thermo are illustrated.

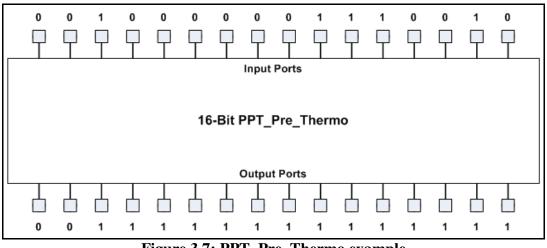


Figure 3.7: PPT_Pre_Thermo example

All PPT topologies which are implemented in this new block are explained in the following sections.

3.1.3. Brent Kung PPT_Pre_Thermo (BK_PPT_Pre_Thermo) Architecture

BK_PPT_Pre_Thermo topology is shown in Figure 3.8. In this topology's nodes, mostly NOR—NAND—INV gates are used rather than OR gate, as explained in OR_BT section. OR gates are used only in the last stages. For N-bit input, BK_PPT_Pre_Thermo architecture's logic level is equal to (2* log₂N -1). On the other hand its gate count is fewer than other PPT topologies. BK_PPT_Pre_Thermo topology is a more useful tree for implementing area efficient designs.

Implementation of NOR—NAND—INV—OR method for different N-bit input brings a variation for inverter placements. Therefore, the designer has to locate inverters attentively. This topology is coded via our RTL generator, and that generator considers these kinds of variations.

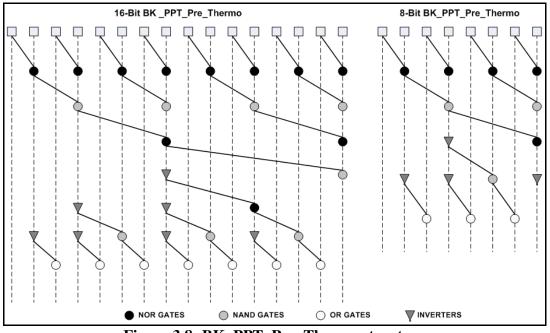


Figure 3.8: BK_PPT_Pre_Thermo structure

3.1.4. Ladner Fisher PPT_Pre_Thermo (LF_PPT_Pre_Thermo) Architecture

The number of stages is reduced by modifying the structure of the PPT graph. The minimum number of stages for PPT is log_2N . For n=32, the stage number is equal to $log_2N=5$. On the other hand, higher fan-out is the major drawback for this tree, as shown in Figure 3.9. For each stage, fan-out is equal to $2^{\text{stage_number}}$, and this can cause a negative effect on timing and area. In order to drive multiple cells, bigger driving cell are used. Unfortunately, those bigger cells can have increased area.

LF_PPT_Pre_Thermo is implemented with NOR—NAND—INV—OR technique too. RTL code for this tree is generated by our RTL generator. Inverter placement varies with respect to the total stage number. For example, when stage number is odd, at last stage no inverter is used. On the other hand, when stage number is even, we have to put inverters at the last stage to construct a functional PPT_Pre_Thermo block. Also, in Figure 3.10, equivalence of LF_PPT_Pre_Thermo is constructed by

OR gates, and LF_PPT_Pre_Thermo is constructed by NOR—NAND—INV—OR gates is shown.

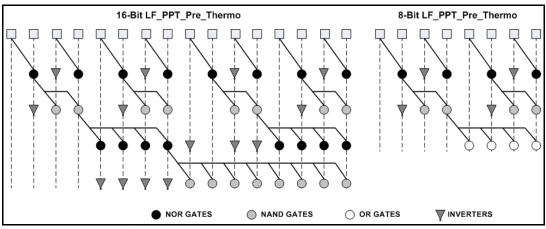


Figure 3.9: LF_PPT_Pre_Thermo structure

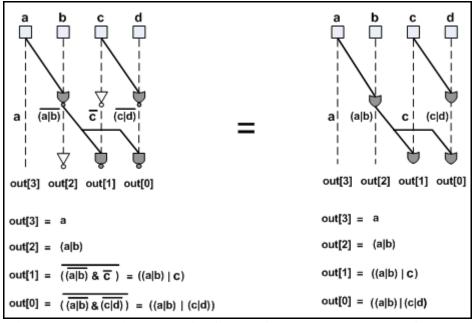
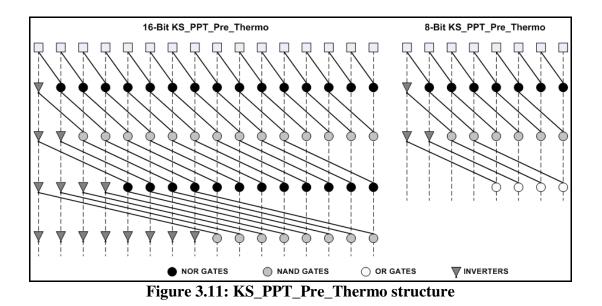


Figure 3.10: Equivalence of NOR—NAND—INV PPT and OR PPT

In our architectures, all of the topologies are constructed using NOR—NAND— INV—OR gates rather than only OR gates. This transformation ensures smaller area results. For example, in Figure 3.10 OR PPT structure has 4 OR gates and each OR gate has extra inverter part in CMOS technology. On the other hand, NOR— NAND—INV tree has 2 NOR, 2 NAND gates, and these gates do not contain any inverters. However, extra 2 inverters are added to ensure the functionality of PPT_Pre_Thermo. Even then, with this technique we save 2 inverters for 4-bit PPT structure. For large input bit width, this cost reduction effects the RRA's area significantly.

3.1.5. Kogge Stone PPT_Pre_Thermo (KS_PPT_Pre_Thermo) Architecture

KS_PPT_Pre_Thermo uses log₂N as similar as LF_PPT_Pre_Thermo. Also, it has low fan-in and fan-out requirement. However, this structure's main drawback is wiring tracks. It has a higher number of lateral wires with longer span which may need extra buffering, thus bringing extra delay. In order to accomplish further improvements on timing and area, it is constructed by NOR—NAND—INV—OR gates too. Its structure is shown in Figure 3.11. Its HDL code is generated via our RTL generator.



3.1.6. Han Carlson PPT_Pre_Thermo (HC_PPT_Pre_Thermo) Architecture

 $HC_PPT_Pre_Thermo$ structure is a hybrid structure which combines stages from KS and BK structures, as shown in Figure 3.12. For n-bit input, its stage number is equal to (log_2N+1) . Its wires have shorter span than KS, this is an advantage against KS structure. This structure is implemented by NOR—NAND—INV—OR gates and its HDL code is generated by our RTL generator. Inverter locations vary with respect to odd or even stage numbers too.

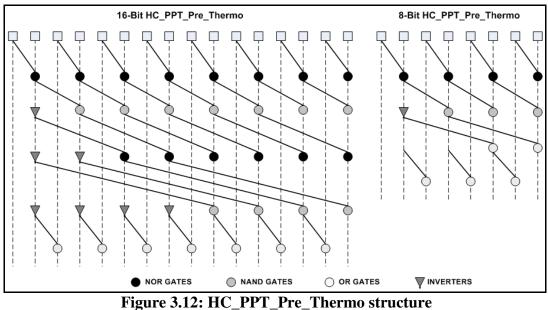


Figure 5.12. IIC_FFT_FTe_Thermo su ucu

3.1.7. Edge Detector (ED)

Edge detector is used to compute RRA's grant, as shown in Figure 3.13. It has two inputs; PPT_Pre_thermo's output and its 1-bit right shifted version. It has one output which is grant of RRA. ED block is the last block before RRA's output and it is bounded with PPT_Pre_Thermo block. It has one level logic depth and this extremely decreases the critical path of the RRA. Combination of PPT_Pre_Thermo and ED performs simple priority encoding operation. Smple_PE operation's logic level almost depends on PPT_Pre_Thermo block. ED has a negligible effect on this operation's logic level.

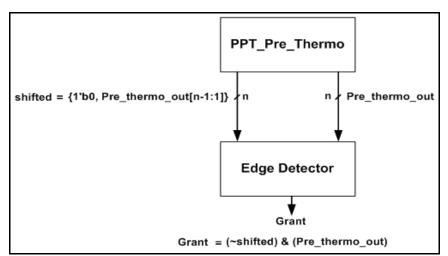


Figure 3.13: Edge detector's top level block diagram

Its main function is capturing $0 \rightarrow 1$ transition. Examples of this functionality are shown in Figure 3.14.

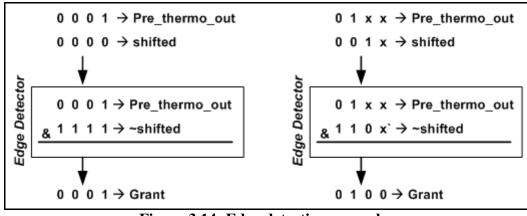


Figure 3.14: Edge detection examples

First, this architecture is implemented with AND—INV gates, as shown in Figure 3.15.

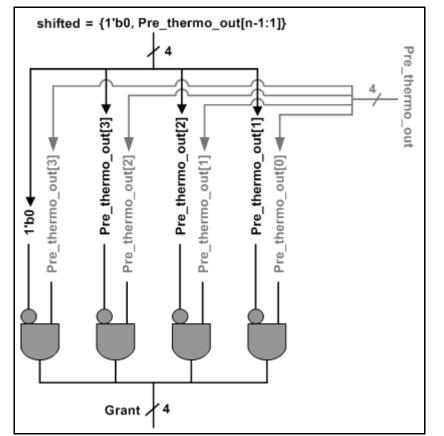


Figure 3.15: Edge detection architecture is constructed by AND—INV gates

Then, we optimized this structure with NOR—INV gates. Thus, this block's area efficiency is improved. This optimization and the new architecture are shown in Figures 3.16 and 3.17, respectively.

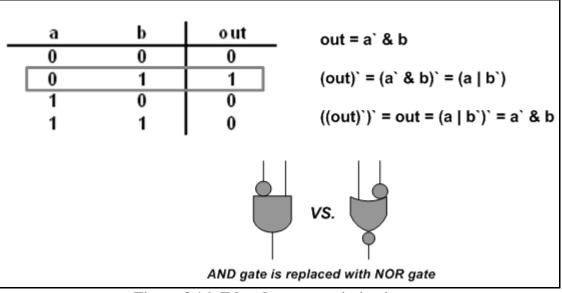


Figure 3.16: Edge detector optimization

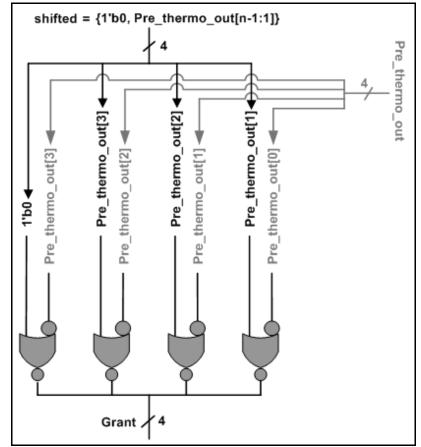


Figure 3.17: Optimized edge detector

3.2. PPT_RRA_BT ARCHITECTURE

This architecture is proposed to achieve maximum speed. Thus, we called this architecture Parallel Prefix Tree Round Robin Arbiter Best Timing (PPT_RRA_BT), as shown in Figure 3.18. When we compare PPT_RRA_BT architecture with PPT_RRA_RS architecture, we see that PPT_RRA_BT architecture uses two PPT_Pre_Thermo blocks. However, it does not consist of OR_BT block. In PPT_RRA_RS architecture, OR_BT is in critical path so this block adds extra delay to input—output path. This problem is eliminated by using two PPT_Pre_Thermo blocks. Therefore, logic level is decreased by factor in log2N. On the other hand, OR_BT gate count is smaller than most of the PPT_Pre_Thermo architecture, so this effects area negatively in some cases. This architecture is very similar to STA_RRA but its PPT_Pre_Thermo block performs pre-thermometer encoding and priority encoding operations to remove negative effects of Smpl_PE, tothermo, and N2LOGN encoder blocks.

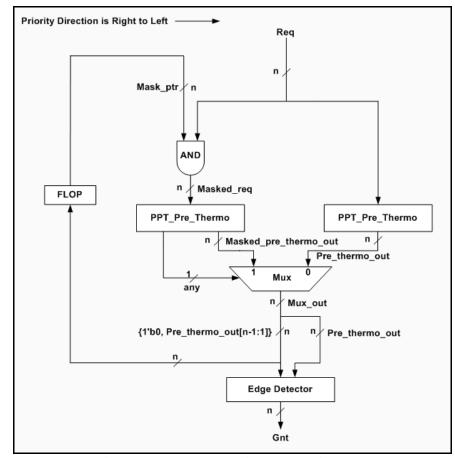


Figure 3.18: PPT_RRA_BT architecture

In Figure 3.19, we replaced the multiplexer with a simplified multiplexer which is used in STA_RRA too. This modification lessens the area cost. This architecture's logic level is shown as follows.

AND Gate \rightarrow 1 level PPT_Pre_Thermo \rightarrow BK:(2* log₂N -1) — KS:log₂N — LF:log₂N — HC:(log₂N +1) Simplified Mux \rightarrow 2 levels ED \rightarrow 1 level Total Logic Level \rightarrow 4 + Logic level of PPT_Pre_Thermo Block

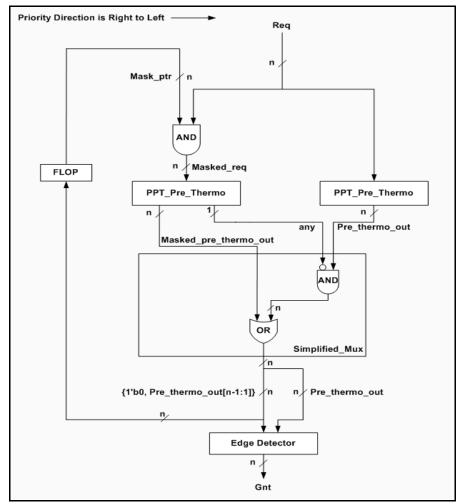


Figure 3.19: PPT_RRA_BT architecture with simplified multiplexer

Details of the sub blocks such as PPT_PRE_Thermo and ED are described in Section 3.1.

3.3. BOW-TIE ARCHITECTURE

This architecture is generated with respect to a Silicon Valley engineer's idea who wants to stay anonymous. As you can see in Figure 3.20, this is comprised of two macro blocks. These are Round Robin Arbiter Macro Block (RR BLOCK) and Grant Unit Macro Block (GUNIT BLOCK).

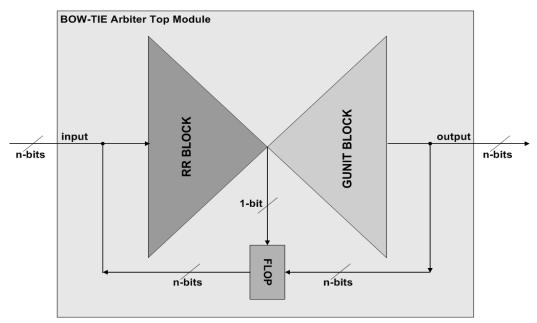


Figure 3.20: BOW-TIE_RRA high level architecture

RR blocks perform two bit round robin arbitration operation. It has six inputs and five outputs. Its inputs are request (req1, req0), pointer (ptr1, ptr0), and high-low (HL1, HL0). New incoming packets/requests are represented as request. Previous iteration's grant is called a pointer, and high-low represents the location of the pointer. If HL is one, it means that the pointer is located at the top of the request. Otherwise, its value is equal to zero and the pointer is located under the request.

RR blocks' outputs are divided into two categories; first category is previous grants (pgnt1, pgnt 0) and second category is request—pointer—high-low (req-ptr-HL). Previous grants are forwarded directly to GUNIT blocks for final output/grant generation. On the other side, req-ptr-HL outputs are routed to next level RR blocks and these RR blocks are used them as input. These connections are shown in Figure 3.21.

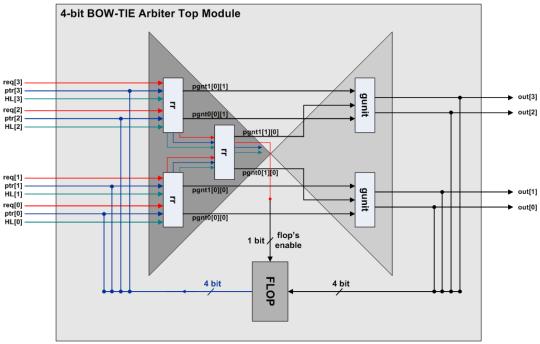


Figure 3.21: BOW-TIE_RRA's building blocks

Last level's RR block's "output req" signal is as the same as any of the STA_RRA. It enables or disables the flop with respect to new incoming request. If there is no request asserted for a new iteration, then last level's RR block's "output req" signal is equal to zero. Thus, the flop will be disabled and next iteration priority pointer will be equal to zero. This prevents the conflicts when no request is asserted.

	Table 5.1. KK block S ti util table											
	in1			in0				out_int			ptr/req	
	left input			right input				pgnts		next HL/ptr/req		
	req1	ptr1	HL1	reqO	ptrO	HLO		pgnt1	pgntO	HL	ptr	req
1	0	х	Х	0	х	х		0	0	х	х	0
2	0	0	х	1	0	х		0	1	х	0	1
3	0	0	х	1	1	0/1		0	1	0/1	1	1
4	0	1	х	1	х	х		0	1	1	1	1
5	1	0	х	0	0	х		1	0	х	0	1
6	1	0	х	0	1	х		1	0	0	1	1
7	1	0	х	1	0	х		1	0	х	0	1
8	1	0	х	1	1	0		1	0	0	1	1
9	1	0	х	1	1	1		0	1	1	1	1
10	1	1	0	0	х	х		1	0	0	1	1
11	1	1	0	1	х	х		0	1	1	1	1
12	1	1	1	х	х	х		1	0	1	1	1

Table 3.1: RR block's truth table

req[1] 0		pgnt1 → 0 out_int[2]	req[1] ()		pgnt1 → 0 out_int[2]
ptr[1] X		• 0 out_init2]	ptr[1] 0		U out_initz]
HL[1] X		← 0 req[0] req[1]	HL[1] X		→ 1 req[0] req[1]
	_			_	
	r	→ X ptr[0] ptr [1]		F	• 0 ptr[0] ptr [1]
req[0] 0		→ X out_int[0]	req[0] 1		→ X out_int[0]
ptr[0] X			ptr[0] 0		
HL[0] X		pgnt0 → 0 out_int[1]	HL[0] X		pgnt0 ► 1 out_int[1]
		1.3			
		panti	req[1] 0		pant1
req[1] 0		pgnt1 → 0 out_int[2]	ptr[1] 1		pgnt1 → 0 out_int[2]
ptr[1] 0		▶ 1 req[0] req[1]			
HL[1] X			HL[1] X		
	7	→ 1 ptr[0] ptr [1]		F	→ 1 ptr[0] ptr [1]
req[0] 1		0/1	req[0] 1		
ptr[0] 1		► 0/1 out_int[0]	ptr[0] X		→ 1 out_int[0]
нь[0] 0/1		pgnt0 ► 1 out_int[1]	HL[0] X		→1 out_int[1]
		pgino			pgnt0
			[]		
req[1] 1		pgnt1 → 1 out_int[2]	req[1] 1		pgnt1 → 1 out_int[2]
ptr[1] 0		▶ 1 req[0] req[1]	ptr[1] 0		▶ 1 req[0] req[1]
HL[1] X		l led[o]lied[i]	HL[1] X		
	R	• 0 ptr[0] ptr [1]		7	→ 1 ptr[0] ptr [1]
rog[0] 0	•		req[0] 0 ————	•	
req[0] 0		→ X out_int[0]			→ X out_int[0]
ptr[0] 0		► 0 out int[1]	ptr[0] 1		D out int[1]
HL[0] X		pgnt0 → 0 out_int[1]	HL[0] X		pgnt0 ► 0 out_int[1]
req[1] 1		pgnt1 → 1 out_int[2]	req[1] 1		pgnt1 → 1 out_int[2]
req[1] 1 ptr[1] 0		pgnt1 → 1 out_int[2]	req[1] 1		pgnt1 → 1 out_int[2]
		pgnt1 → 1 out_int[2] → 1 req[0] req[1]			pgnt1 → 1 out_int[2] → 1 req[0] req[1]
ptr[1] 0	2	→ 1 req[0] req[1]	ptr[1] 0	2	→ 1 req[0] req[1]
ptr[1] 0	гг		ptr[1] 0	r	
ptr[1] 0 HL[1] X req[0] 1	гг	→ 1 req[0] req[1]	ptr[1] 0 HL[1] X req[0] 1	Ţ	→ 1 req[0] req[1]
ptr[1] 0 HL[1] X req[0] 1 ptr[0] 0	r	treq[0] req[1] 0 ptr[0] ptr [1] x out_int[0]	ptr[1] 0 HL[1] X req[0] 1 ptr[0] 1	п	 1 req[0] req[1] 1 ptr[0] ptr [1] 0 out_int[0]
ptr[1] 0 HL[1] X req[0] 1	П	→ 1 req[0] req[1] → 0 ptr[0] ptr [1]	ptr[1] 0 HL[1] X req[0] 1 ptr[0] 1	Ţ	→ 1 req[0] req[1] → 1 ptr[0] ptr [1]
ptr[1] 0 HL[1] X req[0] 1 ptr[0] 0	7	1 req[0] req[1] 0 ptr[0] ptr [1] X out_int[0] pgnt0 0 out_int[1]	ptr[1] 0 HL[1] X req[0] 1 ptr[0] 1 HL[0] 0	г	1 req[0] req[1] 1 ptr[0] ptr [1] 0 out_int[0] 0 out_int[0]
ptr[1] 0 HL[1] X req[0] 1 ptr[0] 0	٦	1 req[0] req[1] 0 ptr[0] ptr [1] X out_int[0] pgnt0 0 out_int[1]	ptr[1] 0 HL[1] X req[0] 1 ptr[0] 1 HL[0] 0	T	1 req[0] req[1] 1 ptr[0] ptr [1] 0 out_int[0] 0 out_int[0]
ptr[1] 0 HL[1] X req[0] 1 ptr[0] 0 HL[0] X	ſſ	→ 1 req[0] req[1] → 0 ptr[0] ptr [1] → x out_int[0] → 0 out_int[1] pgnt0 → 0 out_int[2]	ptr[1] 0 HL[1] X req[0] 1 ptr[0] 1 HL[0] 0	ſſ	→ 1 req[0] req[1] → 1 ptr[0] ptr [1] → 0 out_int[0] → 0 out_int[1] pgnt0 → 0 out_int[1] pgnt1 → 1 out_int[2]
ptr[1] 0 HL[1] X req[0] 1 ptr[0] 0 HL[0] X req[1] 1	T	1 req[0] req[1] 0 ptr[0] ptr [1] X out_int[0] pgnt0 0 out_int[1]	ptr[1] 0	ſſ	→ 1 req[0] req[1] → 1 ptr[0] ptr [1] → 0 out_int[0] pgnt0 → 0 out_int[1]
ptr[1] 0 HL[1] X req[0] 1 ptr[0] 0 HL[0] X		→ 1 req[0] req[1] → 0 ptr[0] ptr [1] → x out_int[0] → 0 out_int[1] pgnt0 → 0 out_int[2] → 1 req[0] req[1]	ptr[1] 0 HL[1] X req[0] 1 ptr[0] 1 HL[0] 0		→ 1 req[0] req[1] → 1 ptr[0] ptr [1] → 0 out_int[0] → 0 out_int[1] pgnt0 → 0 out_int[1] pgnt1 → 1 out_int[2] → 1 req[0] req[1]
ptr[1] 0 HL[1] X req[0] 1 ptr[0] 0 HL[0] X	гг гг	→ 1 req[0] req[1] → 0 ptr[0] ptr [1] → x out_int[0] → 0 out_int[1] pgnt0 → 0 out_int[2]	ptr[1] 0 HL[1] X req[0] 1 ptr[0] 1 HL[0] 0	п	→ 1 req[0] req[1] → 1 ptr[0] ptr [1] → 0 out_int[0] → 0 out_int[1] pgnt0 → 0 out_int[1] pgnt1 → 1 out_int[2]
ptr[1] 0 HL[1] X req[0] 1 ptr[0] 0 HL[0] X		→ 1 req[0] req[1] → 0 ptr[0] ptr [1] → x out_int[0] → 0 out_int[1] pgnt0 → 0 out_int[2] → 1 req[0] req[1]	ptr[1] 0 HL[1] X req[0] 1 ptr[0] 1 HL[0] 0 req[1] 1 ptr[1] 1 HL[1] 0 req[0] 0		→ 1 req[0] req[1] → 1 ptr[0] ptr [1] → 0 out_int[0] → 0 out_int[1] pgnt0 → 0 out_int[1] pgnt1 → 1 out_int[2] → 1 req[0] req[1]
ptr[1] 0 HL[1] X req[0] 1 ptr[0] 0 HL[0] X		→ 1 req[0] req[1] → 0 ptr[0] ptr [1] → x out_int[0] → 0 out_int[1] pgnt0 → 0 out_int[1] → 1 req[0] req[1] → 1 ptr[0] ptr [1] → 1 out_int[0]	ptr[1] 0 HL[1] X req[0] 1 ptr[0] 1 HL[0] 0 req[1] 1 ptr[1] 1 HL[1] 0 req[0] 0 req[0] 0 req[0] 0		→ 1 req[0] req[1] → 1 ptr[0] ptr [1] → 0 out_int[0] pgnt0 → 0 out_int[1] pgnt1 → 1 out_int[2] → 1 req[0] req[1] → 1 ptr[0] ptr [1] → 0 out_int[0]
ptr[1] 0 HL[1] X req[0] 1 ptr[0] 0 HL[0] X		→ 1 req[0] req[1] → 0 ptr[0] ptr [1] → x out_int[0] → 0 out_int[1] pgnt0 → 0 out_int[1] pgnt1 → 0 out_int[2] → 1 req[0] req[1] → 1 ptr[0] ptr [1]	ptr[1] 0 HL[1] X req[0] 1 ptr[0] 1 HL[0] 0 req[1] 1 ptr[1] 1 HL[1] 0 req[0] 0		→ 1 req[0] req[1] → 1 ptr[0] ptr [1] → 0 out_int[0] pgnt0 → 0 out_int[1] pgnt1 → 1 out_int[2] → 1 req[0] req[1] → 1 ptr[0] ptr [1]
ptr[1] 0 HL[1] X req[0] 1 ptr[0] 0 HL[0] X		→ 1 req[0] req[1] → 0 ptr[0] ptr [1] → x out_int[0] → 0 out_int[1] pgnt0 → 0 out_int[1] → 1 req[0] req[1] → 1 ptr[0] ptr [1] → 1 out_int[0] → 1 out_int[0]	ptr[1] 0 HL[1] X req[0] 1 ptr[0] 1 HL[0] 0 req[1] 1 ptr[1] 1 HL[1] 0 req[0] 0 req[0] 0 req[0] 0		→ 1 req[0] req[1] → 1 ptr[0] ptr [1] → 0 out_int[0] → 0 out_int[1] pgnt0 → 0 out_int[1] → 1 out_int[2] → 1 req[0] req[1] → 1 ptr[0] ptr [1] → 0 out_int[0] → 0 out_int[0]
ptr[1] 0 HL[1] X req[0] 1 ptr[0] 0 HL[0] X		→ 1 req[0] req[1] → 0 ptr[0] ptr [1] → x out_int[0] pgnt0 → 0 out_int[1] pgnt1 → 0 out_int[2] → 1 req[0] req[1] → 1 ptr[0] ptr [1] → 1 out_int[0] pgnt0 → 1 out_int[1]	ptr[1] 0 HL[1] X req[0] 1 ptr[0] 1 HL[0] 0 req[1] 1 ptr[1] 1 HL[1] 0 req[0] 0 req[0] 0 req[0] 0		→ 1 req[0] req[1] → 1 ptr[0] ptr [1] → 0 out_int[0] pgnt0 → 0 out_int[1] pgnt1 → 1 out_int[2] → 1 req[0] req[1] → 1 ptr[0] ptr [1] → 0 out_int[0] pgnt0 → 0 out_int[1]
ptr[1] 0 HL[1] X req[0] 1 ptr[0] 0 HL[0] X		→ 1 req[0] req[1] → 0 ptr[0] ptr [1] → x out_int[0] → 0 out_int[1] pgnt0 → 0 out_int[1] → 1 req[0] req[1] → 1 ptr[0] ptr [1] → 1 out_int[0] pgnt0 → 1 out_int[1]	ptr[1] 0 HL[1] X req[0] 1 ptr[0] 1 HL[0] 0 req[1] 1 ptr[1] 1 HL[1] 0 req[0] 0 req[0] 0 req[0] 0 ptr[0] X HL[0] X		1 req[0] req[1] 1 ptr[0] ptr [1] 0 out_int[0] pgnt0 0 out_int[1] pgnt1 1 out_int[2] 1 req[0] req[1] 1 ptr[0] ptr [1] 0 out_int[0] pgnt0 0 out_int[1]
ptr[1] 0 HL[1] X req[0] 1 ptr[0] 0 HL[0] X req[1] 1 ptr[0] 1 req[0] 1 ptr[0] 1 req[0] 1 ptr[0] 1 ptr[1] 1		→ 1 req[0] req[1] → 0 ptr[0] ptr [1] → x out_int[0] pgnt0 → 0 out_int[1] pgnt1 → 0 out_int[2] → 1 req[0] req[1] → 1 ptr[0] ptr [1] → 1 out_int[0] pgnt0 → 1 out_int[1]	ptr[1] 0 HL[1] X req[0] 1 ptr[0] 1 HL[0] 0 req[1] 1 ptr[1] 1 HL[1] 0 req[0] 0 req[0] 0 ptr[0] X HL[0] X req[1] 1 ptr[1] 1		→ 1 req[0] req[1] → 1 ptr[0] ptr [1] → 0 out_int[0] pgnt0 → 0 out_int[1] pgnt1 → 1 out_int[2] → 1 req[0] req[1] → 1 ptr[0] ptr [1] → 0 out_int[0] pgnt0 → 0 out_int[1]
ptr[1] 0 HL[1] X req[0] 1 ptr[0] 0 HL[0] X	7	→ 1 req[0] req[1] → 0 ptr[0] ptr [1] → x out_int[0] → 0 out_int[1] pgnt0 → 0 out_int[1] → 1 req[0] req[1] → 1 ptr[0] ptr [1] → 1 out_int[0] pgnt0 → 1 out_int[1] pgnt1 → 0 out_int[2] → 1 req[0] req[1]	ptr[1] 0 HL[1] X req[0] 1 ptr[0] 1 HL[0] 0 req[0] 0 req[0] 0 req[0] 0 req[0] 0 req[0] 0 req[0] 0 ptr[0] X HL[0] X	7	→ 1 req[0] req[1] → 1 ptr[0] ptr [1] → 0 out_int[0] → 0 out_int[1] pgnt0 → 0 out_int[1] → 1 req[0] req[1] → 1 ptr[0] ptr [1] → 0 out_int[0] pgnt0 → 0 out_int[1] pgnt1 → 1 out_int[2] → 1 req[0] req[1]
ptr[1] 0 HL[1] X req[0] 1 ptr[0] 0 HL[0] X req[1] 1 ptr[0] 1 req[0] 1 ptr[0] 1 req[0] 1 ptr[0] 1 ptr[1] 1		→ 1 req[0] req[1] → 0 ptr[0] ptr [1] → x out_int[0] → 0 out_int[1] pgnt0 → 0 out_int[1] → 1 req[0] req[1] → 1 ptr[0] ptr [1] → 1 out_int[0] pgnt1 → 0 out_int[2]	ptr[1] 0 HL[1] X req[0] 1 ptr[0] 1 HL[0] 0 req[1] 1 ptr[1] 1 HL[1] 0 req[0] 0 req[0] 0 ptr[0] X HL[0] X req[1] 1 ptr[1] 1		→ 1 req[0] req[1] → 1 ptr[0] ptr [1] → 0 out_int[0] → 0 out_int[1] pgnt0 → 0 out_int[1] → 1 out_int[2] → 1 ptr[0] ptr [1] → 0 out_int[0] pgnt1 → 1 out_int[2]
ptr[1] 0 HL[1] X req[0] 1 ptr[0] 0 HL[0] X req[1] 1 ptr[0] 1 req[0] 1 ptr[0] 1 req[0] 1 ptr[0] 1 ptr[1] 1	7		ptr[1] 0 HL[1] X req[0] 1 ptr[0] 1 HL[0] 0 req[1] 1 ptr[1] 1 HL[1] 0 req[0] 0 req[0] 0 ptr[0] X HL[0] X req[1] 1 ptr[1] 1	7	
ptr[1] 0 HL[1] X req[0] 1 ptr[0] 0 HL[0] X req[1] 1 ptr[0] 1 HL[1] X req[0] 1 ptr[0] 1 ptr[0] 1 ptr[1] 1 ptr[1] 1 ptr[1] 1 ptr[1] 1	7	→ 1 req[0] req[1] → 0 ptr[0] ptr [1] → x out_int[0] → 0 out_int[1] pgnt0 → 0 out_int[1] → 1 req[0] req[1] → 1 ptr[0] ptr [1] → 1 out_int[0] pgnt0 → 1 out_int[1] pgnt1 → 0 out_int[2] → 1 req[0] req[1]	ptr[1] 0 HL[1] X req[0] 1 ptr[0] 1 HL[0] 0 req[1] 1 HL[1] 0 req[0] 0 req[0] 0 req[0] 0 req[0] 0 req[1] 1 HL[0] X HL[0] X req[1] 1 HL[1] 1	7	→ 1 req[0] req[1] → 1 ptr[0] ptr [1] → 0 out_int[0] → 0 out_int[1] pgnt0 → 0 out_int[1] → 1 req[0] req[1] → 1 ptr[0] ptr [1] → 0 out_int[0] pgnt0 → 0 out_int[1] pgnt1 → 1 out_int[2] → 1 req[0] req[1]
ptr[1] 0 HL[1] X req[0] 1 ptr[0] 0 HL[0] X req[1] 1 ptr[1] 0 HL[1] X req[0] 1 ptr[0] 1 ptr[1] 1 ptr[1] 1 ptr[1] 1 ptr[1] 1 ptr[1] 1 ptr[1] 1 ptr[1] 1 ptr[1] 1 ptr[1] 1 ptr[1] 1 ptr[1] 1 ptr[1] 1 ptr[1] 1 ptr[1] 1 ptr[1] 1 ptr[1] 1 ptr[1] 1	7		ptr[1] 0 HL[1] X req[0] 1 ptr[0] 1 HL[0] 0 req[1] 1 ptr[1] 1 HL[1] 0 req[0] 0 req[0] 0 req[1] 1 HL[0] X req[1] 1 ptr[1] 1 ptr[1] 1 req[0] X	7	

Figure 3.22: RR block's all possible states

The core blocks of this architecture are the RR blocks. Actually, they are working as small RRAs because they perform two-bit round robin arbitration and they produce any, pointer, and high-low outputs for next level RRA blocks. Their truth table is shown in Table 3.1. Also, all possible input—output combinations of RR blocks are shown in Figure 3.22. Essentially, RR blocks are slightly complex blocks, this effects this architecture's timing and especially area performance in a negative manner. If these blocks are designed in a simple way, this architecture's performance could be increased immensely.

GUNIT blocks are very simple blocks, as shown in Figure 3.23. It gets three inputs and produces the final outputs of the BOW-TIE_RRA. All RR blocks' pgnt signals are forwarded to GUNIT blocks and they are ANDed to output the final grant output. This routing is clearly seen in Figure 3.21.

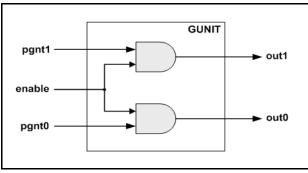


Figure 3.23: GUNIT block

3.4. ENHANCEMENTS ON PREVIOUS WORK

In order to enhance STA_RRA and CHN_RRA architectures we replaced Optimized Smple_PE blocks with Han Carlson PPT_Pre_Thermo (HC_PPT_Pre_Thermo)–ED combination. In this manner, we can see the effects of HC_PPT_Pre_thermo–ED on RRAs performance.

Furthermore, we tried to use N2LOGN Smple_PE encoder to eliminate the N2LOGN encoder block for STA_RRA, as shown Figure 3.24. This new architecture is called STA_RAA_N2LOGN. All of these architectures are described in next sections.

3.4.1. STA_RRA_N2LOGN Architecture

This architecture's Smple_PE and Smpl_PE_Thermo blocks take n-bit inputs and output log₂N bit outputs as shown in Figure 3.24. Therefore, there is no need to use N2LOGN Encoder block before tothermo block. Thus, area efficiency and timing performance can be increased with respect to well-known STA_RRA_N2N architecture.

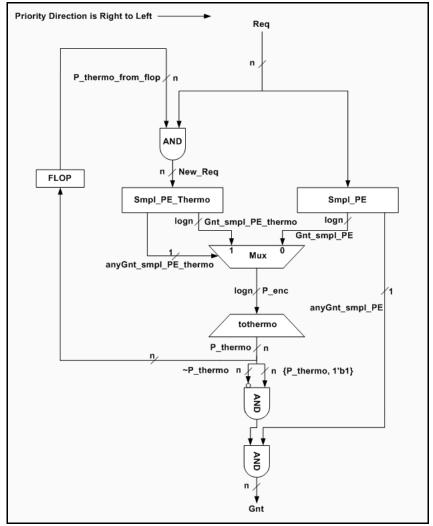


Figure 3.24: STA_RRA_N2LOGN architecture

After the tothermo block an edge detection operation is accomplished by AND gate. Also, final output is ANDed with Smpl_PE's any bit to eliminate the conflicts when zero requests are asserted. For example, a request which has 1'b0 value for each bit position is asserted for new iteration. In this iteration, RRA's output's all bit positions to be 1'b0. If we do not use final AND gate, RRA's output's LSB will be 1'b1. This example is illustrated in Figure 3.25. In order to eliminate this problem, that AND gate is placed at the end of the critical path.

Without using final AN	ID gate	With using final AND gate			
Request	ightarrow 0 0 0 0 0 0 0 0 0	Request	\rightarrow 0 0 0 0 0 0 0 0 0		
~P_thermo	→11111111	~P_thermo	→11111111		
{P_thermo,1'b1}	ightarrow 0 0 0 0 0 0 0 1	{P_thermo,1'b1}	ightarrow 0 0 0 0 0 0 1		
~P_thermo & {P_thermo,1'b1}	ightarrow 0 0 0 0 0 0 0 1	~P_thermo & {P_thermo,1'b1}	ightarrow 0 0 0 0 0 0 1		
Gnt = P_thermo & {P_thermo,1'b1}	ightarrow 0 0 0 0 0 0 0 1	Gnt = (8'b0 & (P_thermo & {P_thermo,1'b1})) → 0 0 0 0 0 0 0 0 0		

Figure 3.25: Zero request example

N2LOGN Smpl_PE's are implemented recursively with binary tree technique. These blocks have order of log₂N logic level and this is better than ripple carry Smpl_PE.

3.4.2. STA_PPT_RRA Architecture

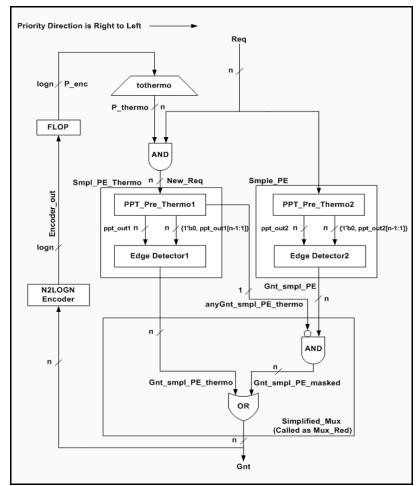


Figure 3.26: STA_PPT_RRA architecture

This architecture is almost as the same as STA_RRA_N2N architecture. Only difference is in Smpl_PE blocks. In this architecture HC_PPT_Pre_Thermo— ED blocks combination is used rather than Optimized Smpl_PE blocks. This architecture is shown in Figure 3.26.

3.4.3. CHN_PPT_RRA_PPE_Conflict Architecture

In this architecture HC_PPT_Pre_Thermo— ED blocks combination is used rather than Optimized Smpl_PE block. This architecture is shown in Figure 3.27. This is the only difference from the original CHN_RRA_PPE_Conflict architecture.

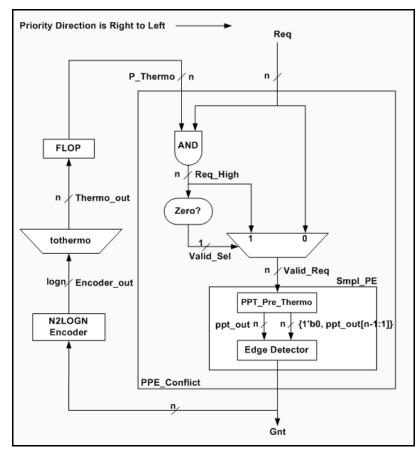


Figure 3.27: CHN_PPT_RRA_PPE_Conflict architecture

3.4.4. CHN_PPT_RRA_PPE_NonConflict Architecture

In this architecture HC_PPT_Pre_Thermo— ED blocks combination is used rather than Optimized Smpl_PE block too. This architecture is shown in Figure 3.28. This is the only difference from the original CHN_RRA_PPE_NonConflict architecture.

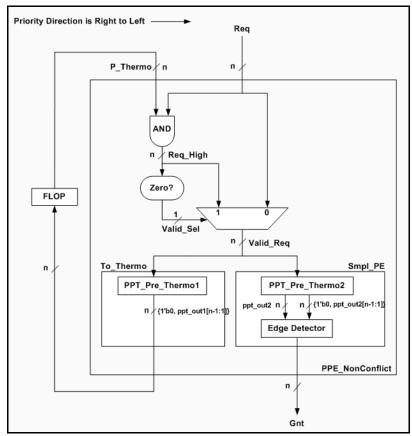


Figure 3.28: CHN_PPT_RRA_PPE_NonConflict architecture

4. AUTOMATIC RTL GENERATION

Register Transfer Level (RTL) coding is an HDL coding technique in which the behavior of a design is defined in terms of transfer of data between hardware registers, and logical operations performed on those signals.

Automatic RTL code generation has a very important role in this work. All RRA architectures have to be compared against each other from 8-bit input to 256-bit input variations. It is easy to write 8 Bit RRA's HDL code. On the other hand, if we try to write 256-bit RRA code, we would get into huge trouble. We probably lose consistency and make a lot of syntax and instantiation errors during writing up 128-bit or 256-bit RRA's Verilog HDL code. It is very cumbersome and tough to write 256-bit RRA code. In addition, it takes too much time. In order to get rid of these problems we tried to automate this process. We used PHP language and wrote scripts that automatically generate Verilog HDL code of RRA architectures with respect to their input bit widths. Those scripts take only one argument, which is the input bit width of the RRA. Then, scripts automatically generate all necessary Verilog HDL files for a specific RRA. Automatic RTL code generation task is the one of the most coercive tasks for this thesis and details of all generators are described in the next sections.

4.1. STA_RRA GENERATION

There are three different types of STA_RRA generators are coded. These are STA_RRA_N2N generator which is shown in figure 4.1., STA_RRA_N2LOGN generator, and STA_RRA_PPT generator. All generators perform the same task. They take an argument which is the input bit width of architecture, then generate Verilog HDL files, and finally put all generated files to a specific folder.

Generated Verilog HDL files of STA_RRA_N2N generator are shown in Figure 4.1.

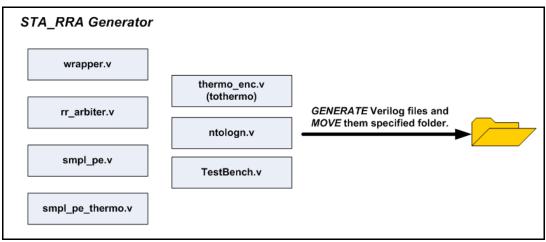


Figure 4.1: STA_RRA generation

wrapper.v is used for iterative synthesis. It is just a wrapper for top level design. It has two n-bit flops at its input and output ports. These flops are necessary to create timing path from request to grant of RRA.

rr_arbiter.v is top level block of RRA. It instantiates other blocks and performs RRA operation. Simplified multiplexer code is in this Verilog HDL file.

smpl_pe.v and *smpl_pe_thermo.v* is optimized simple priority encoder code. This priority encoder's Verilog code is generated with regard to binary tree and pre-computation/factoring techniques which are explained in Section 2.1.1. "Any bit" of smpl_pe_themro.v is generated with binary tree technique.

thermo_enc.v executes thermometer encoding and its algorithm is represented in Pankaj's and McKeown's work. Hence, we did not try to optimize this logic. Also, its Verilog HDL code is written in a parameterized fashion, so generation of this block became easier than the other blocks.

ntologn.v is generated with respect to binary tree technique as described in Section 2.1.3

testbecnh.v is the verification code of RRA which is used for RTL and gate level verifications.

In STA_RRA_N2LOGN architecture normal multiplexer is used rather than simplified multiplexer and N2LOGN encoder block generation is removed. Its architecture, which is shown in Figure 3.24 is taken into consideration when writing its RTL generator script.

Also, STA_RRA_PPT architecture generated with respect to Figure 3.26.

4.2. CHN_RRA GENERATION

There are four different generators written for CHN_RRA architectures. One of the generators is written for CHN_RAA_PPE_Conflict which is shown in Figure 4.2. It generates necessary Verilog HDL files and moves them into a specific folder. Smpl_PE, N2LOGN, tothermo, wrapper, and testbench modules are the same for STA_RRA and this architecture. The other generator is written for CHN_PPT_RAA_PPE_Conflict architecture with respect to Figure 3.27. In that architecture the difference is the HC_PPT_Pre_Thermo block plus ED block generation.

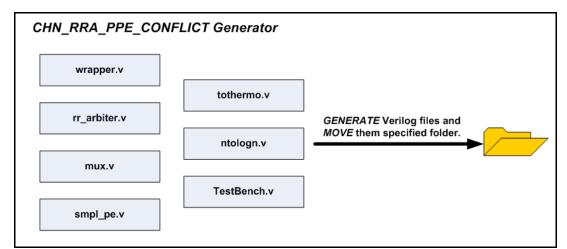


Figure 4.2: CHN_RRA_PPE_Conflict generation

CHN_RRA_PPE_NonConflict generation is shown is Figure 4.3 and its variant CHN_PPT_RRA_PPE_NonConflict architecture is generated according to Figure 3.28. In the original version of these architectures, combinations of "Smpl_PE + N2LOGN Encoder + tothermo" blocks are used to construct the N2N thermometer encoder block. This is not an efficient way to construct N2N thermometer encoder.

Thus, we used HC_PPT_Pre_thermo block to optimize N2N thermometer encoder block.

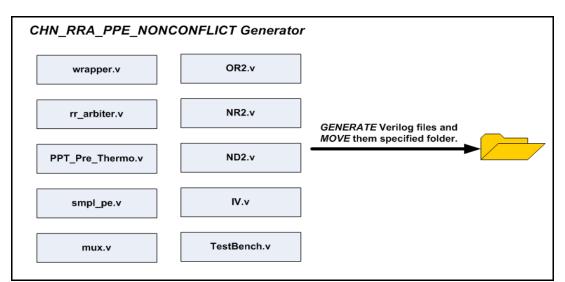


Figure 4.3: CHN_RRA_PPE_NonConflict (*OPTIMIZED) generation

4.3. PPT_RRA GENERATION

Nine different generators are written for PPT_RRA_RS and PPT_RRA_BT architectures, which are shown in Figures 4.4 and 4.5, respectively. All PPT algorithms' generators are written for PPT_Pre_Thermo Block. This is one of the most difficult tasks of this thesis. LF, BK, HC, and KS PPT_Pre_Thermo blocks are generated with respect to figures which are shown in chapter 3. In top-level module all of the sub-modules are instantiated and connected. Testbench code is same for all generators because they are performing the same RRA algorithm. Also, wrapper is the same for all architectures as well.

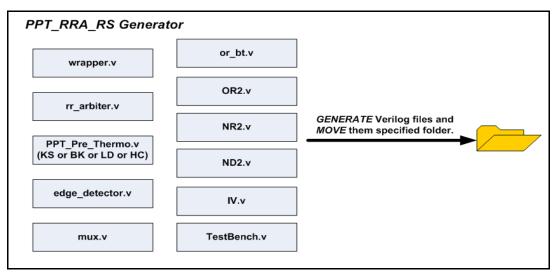


Figure 4.4: PPT_RRA_RS generation

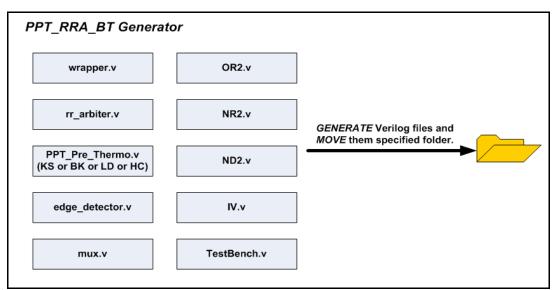


Figure 4.5: PPT_RRA_BT generation

4.4. BOW-TIE_RRA GENERATION

BOW-TIE_RRA architecture generation is accomplished with respect to all figures in Section 3.3. RR blocks' functionality is captured from Table 3.1 and its generator is written by using those equations. Also, GUNIT module is a very small module that it is generated easily. GUNIT modules are instantiated in gunit_n.v and RR modules are instantiated in rr_n.v. Eventually, GUNIT macro block and RR macro block are instantiated in bow-tie.v top level module. This architecture's generator outputs the same wrapper.v and testbecnh.v files as the other generators. This generator is shown in Figure 4.6.

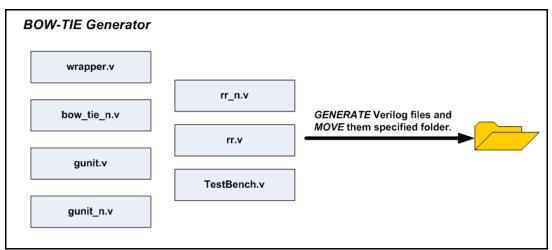


Figure 4.6: BOW-TIE generation

5. VERIFICATION AND SYNTHESIS METHODOLOGY

All RRA architectures' Verilog HDL codes are generated via their RTL generators. After this generation operation, all RTL codes are verified by RTL simulation. Then, verified RTL code is synthesized to get gate level netlist. There are many options in synthesis work. Different constraints could be applied to our design to get best synthesis results. All of these issues are covered in the following sections.

5.1. VERIFICATION

In verification task, we have to verify our RTL code's functionality first. Our design's RTL code is tested with behavioral test model. Behavioral test model performs our design functionality so RTL simulation is also called functional verification. The test model is written with a very high level of abstraction to fulfill the functionality of the RRA algorithm. This behavioral model, or in other words functional test model is in a module which is called testbench. We instantiate our RTL design into that testbench and apply the same test vectors to both our RTL design and behavioral test model. Eventually, outputs of these two blocks are compared for verification. In this way we check our RRA RTL design's functionality. If RTL verification phase is passed without any problem, then that RTL code could be synthesized by DC. After synthesis, we get gate level netlist regarding to our technology library. We have to verify this gate level netlist' functionality too, so gate level verification process is started. Verilog HDL code of technology library, gate level netlist, and test model are used to accomplish the gate level functional verification. Testbench instantiates gate level netlist to compare its functionality with that of the behavioral test model. After these phases our RTL design and gate level design are verified. Verification process is shown in Figure 5.1. This figure is valid for both RTL and gate level verification tasks.

In this thesis, the test model is written with respect to STA_RRA arbitration algorithm, which is the combination of *i*SLIP and mRRR algorithms, and it is almost identical to ESLIP algorithm, which is described in the introduction section. This

behavioral test model is coded with a high level of abstraction and it is used for verification of all RRA architectures.

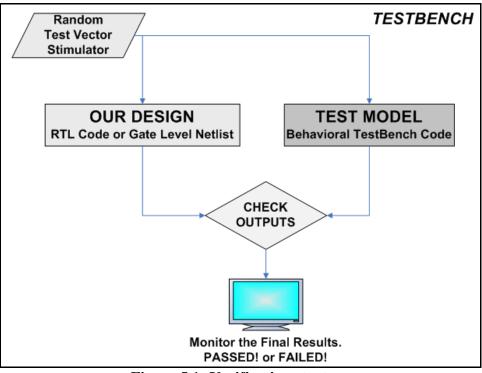


Figure 5.1: Verification strategy

Testbench code for 8 bit RRA is shown below.

```
module rr_arbiter_tb ();
    //Datatype Declerations
    reg clk, rst;
    reg arbitrate;
    reg [7:0] req;
    reg [7:0] tb_out;
    reg [3:0] ptr;
    reg flag;
    wire [7:0] design_out;
    integer i, k, m, down, up;
    //Instatiations
    rr_arbiter rr_arbiter_ins (.clk(clk), .rst(rst), .req(req),
                                .gnt(design_out));
    //Decleration of Initial Values
    initial begin
        clk = 0;
        #5 rst = 1;
        #15 rst = 0;
    end
```

```
//Toggle clock every 10 nanoseconds
   always begin
        #10 clk = \sim clk;
    end
//Test Vectors (Inputs) Declerations
   always @(posedge clk) begin
        if (rst) begin
           ptr <= 4'd8;
            req <= 8'b0;
            tb_out <= 8'b0;
            arbitrate <= 1'b0;
        end
        else begin
           req <= $random;</pre>
           arbitrate <= ~arbitrate;</pre>
        end
   end
//Test Model
    always @(arbitrate) begin:RR_ARBITER
       if (req == 8'b0) begin
         tb_out = 8'b0;
          ptr = 4'd8;
       end
       else begin
          flag = 0;
          if (ptr != 3'b0) begin
             for (i=(ptr-1); i>=0; i=i-1) begin
                 if (req[i]==1'b1) begin
                     tb_out[i] = 1'b1;
                     ptr = i;
                     for (down=(i-1); down>=0; down=down-1) begin
                          tb_out[down] = 1'b0;
                      end
                     for (up=(i+1); up<8; up=up+1) begin
                          tb_out[up]=1'b0;
                      end
                     flag = 1;
                     disable RR_ARBITER;
                 end
             end
          end
          if (flag == 1'b0) begin
              for (k=7; k>=ptr; k=k-1) begin
                  if(req[k]==1'b1) begin
                      tb_out[k] = 1'b1;
                      ptr = k;
                       for (m=(k-1); m>=0; m=m-1) begin
                           tb_out[m] = 1'b0;
                      end
                      disable RR_ARBITER;
                  end
              end
          end
       end
   end
```

55

```
//Control and Monitoring Part
    always @(posedge clk) begin
        #2 //Wait 2ns for correct comparison
        if (design_out == tb_out) begin
            $display ("Time = %d\t req = %b\t design_output = %b\t
                      testmodel_output = %b\t --> CORRECT", $time, req,
                     design_out, tb_out);
        end
        else begin
            $display ("Time = %d\t req = %b\t design_output = %b\t
                      testmodel_output = %b\t --> ERROR", $time, req,
                      design_out, tb_out);
            $finish;
        end
    end
endmodule
```

5.2. SYNTHESIS METHODOLOGY

The synthesis task is done by DC. It calculates timing results according to timing paths. Timing path is a path that starts from a flop's clock trigger and ends at another flop's input. The timing paths are taken into account when writing timing constraints. In order to write timing constraints for RRA design, we have to put that design into a wrapper. The wrapper has flops at its input and output ports. RRA design also comprises of a flop to keep priority pointer. Therefore, we can create two timing paths from flop1's clk input to flop2's input and flop1's clk input to flop3's input. These two timing paths and other two timing paths are shown in Figure 5.2. Critical path of the design depends on combinational logic blocks which are located in these timing paths. If a timing paths. This path is called the critical path. Any design's speed is calculated with its critical path's timing.

A wrapper structure is ideal to create timing constraints for all RRA designs. We created a clock period in our synthesis scripts iteratively and tried to find in which clock period a positive slack occurs. We started with an over-constrained clock period value, but for each synthesis iteration we increased the clock period by one nanosecond to reach positive slack results. Positive slack means that our design meets the timing constraint. There are no timing violations when we see positive slack at the end of synthesis.

Area result is reported for RRA design. If we report area for wrapper, area result contains input and output flops in wrapper block. Wrapper block is not our actual design so area result must be reported with respect to RRA design.

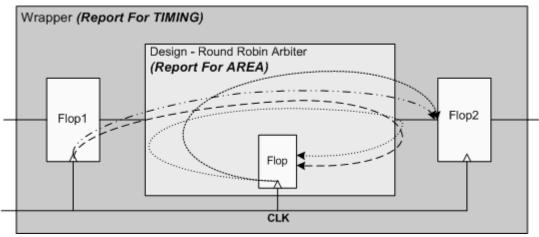


Figure 5.2: Synthesis methodology

We run several synthesis tasks to find the best timing and area constraint. Synthesis tasks are done with no constraint (NC), set_max_Area 0 (SMA0), and SMA0 + compile incremental –map effort high (CI_MEH) constraints. After all synthesis results we compare synthesis constraints to use the best one. Eventually we use the constraints below in our synthesis scripts.

```
# Timing constraint
create_clock -name clk -period 6 clk
# Area constraint
set_max_area 0
# Compile the design incrementally with high effort
compile -incremental_mapping -map_effort high
# Report area results
current_design rr_arbiter
report_area
# Report timing results
current_design wrapper
report_timing
```

In this work many RRA designs are synthesized. This task is notably time consuming. In order to shorten the synthesis task time, we automated synthesis with PERL scripts. Two PERL scripts are written for synthesis. One of them does the

synthesis task and the other one uses this script and accomplishes all RRA designs' synthesis tasks. These scripts are called core synthesis script and regression script. They are explained as follows.

Core Synthesis Script does the following tasks:

- Creates folders for synthesis scripts and results.
- Reads sample synthesis scripts and create new ones for iterative synthesis. In this part clock period is increased by one nanosecond from start point to end point.
 These points could be defined in the script. If you want to do synthesis from 6 to 20 you can easily set these points in the script.
- Invokes DC and runs all scripts and keeps their logs.
- Moves synthesis scripts and synthesis results into related folders.
- Modifies synthesis results, wraps and writes them into a related folder.
- Captures slack and total cell area values from modified result file.
- Writes total cell area and slack values to excel file.

Regression Script synthesizes all RRA designs by using specific *synthesis script* for each RRA designs. This script automatically synthesizes all synthesis tasks. Sample code snippet for this script is shown below.

```
$path1="C:/workspace/CHN_RRA/PPE_Conflict/8Bits";
$path2="C:/workspace/PPT_RRA/LF_RRA_BT/8Bits";
$path3="C:/workspace/STA_RRA/STA_RRA_N2N/8Bits";
print ("\n\n$path1 --> This path's synthesis is started!!!\n\n");
chdir($path1) || die "Can't chdir: $!";
system ("perl RRA_Syn_CHN.pl");
print ("\n\n$path1 --> This path's synthesis is finished!!!\n\n");
chdir($path2) || die "Can't chdir: $!";
system ("perl RRA_Syn_LF_PPT.pl");
print ("\n\n$path2 --> This path's synthesis is finished!!!\n\n");
chdir($path2) || die "Can't chdir: $!";
system ("perl RRA_Syn_LF_PPT.pl");
print ("\n\n$path3 --> This path's synthesis is started!!!\n\n");
chdir($path3) || die "Can't chdir: $!";
system ("perl RRA_Syn_STA_N2N.pl");
print ("\n\n$path3 --> This path's synthesis is finished!!!\n\n");
```

6. SYNTHESIS RESULTS

All of the synthesis is performed with Synopsys DC with lsi_10k technology library. Detailed iterative synthesis results from 8-bits to 256-bits and synthesis results at first positive or zero slack are shown in this chaper.

					Rou	nd Robi	n Arbite	ers TIM	ING C	omparis	son for a	8Bit In	put				
Time (ns)	STA N2N	STA PPT	STA N2LOGN	CHN PPE Conflict	CHN PPE Conflict PPT	CHN PPE Non Conflict	CHN PPE Non Conflict PPT	ВК_ВТ	BK_RS	НС_ВТ	HC_BT Simple Mux	HC_RS	KS_BT	KS_RS	LF_BT	LF_RS	BOWTIE
6	-2,51	-2,88	-2,65	-4,3	-5,12	-2,08	-2,58	-1,7	-2,83	-1,38	-1,24	-3,17	-1,35	-3,16	-1,27	-2,95	-2,76
7	-1,51	-1,88	-1,65	-3,3	-4,22	-1,08	-1,58	-0,45	-1,85	-0,39	-0,26	-2,17	-0,62	-2,06	-0,27	-1,95	-1,87
8	-0,49	-0,91	-0,65	-2,22	-3,14	-0,2	-0,61	0,01	-0,89	0		-1,17	0,02	-1,22	0	-0,96	-0,73
9	0	-0,15	0,02	-1,32	-1,99	0,01	0,01	0,39	-0,01	0,39	0,08	-0,11	0,02	0	0,48	-0,01	0,02
10	0,11	0,02	0,03	-0,5	-1,28	0,16	0,04	0,46	0	0,2	1,23	0,08	0,13	0	0,46	0,01	0,1
11	0,95	0,18	0,11	0,02	-0,42	0,15	0,13	1,46	0,17	1,2	2,23	0,02	1,13	0,1	1,46	0,03	0,06
12	1,75	0,06	0,8	0,04	0,01	1,15	0,49	2,46	0,11	2,2	3,23	0,38	2,13	0,61	2,46	0,41	0,08
13	2,75	0,74	1,82	0,24	0,01	2,15	1,49	3,46	1,11	3,2	4,23	1,38	3,13	1,61	3,46	0,86	0,86
14	3,75	1,74	2,82	0,21	0,01	3,15	2,49	4,46	1,34	4,2	5,23	2,38	4,13	2,19	4,46	1,86	1,86
15	4,75	2,74	3,82	1,61	0,13	4,15	3,49	5,46	2,34	5,2	6,23	3,38	5,13	3,19	5,46	2,86	2,86
<u>16</u> 17	5,75	3,74	4,82	2,55	0,74	5,15	4,49	6,46	3,34	6,2	7,23	4,38	6,13	4,19	6,46	3,86	3,86
18	6,75	4,74	5,82	3,55	1,69	6,15	5,49	7,46	4,34	7,2	8,23	5,38	7,13	5,19	7,46	4,86	4,86
10	7,75	5,74	6,82	4,55	2,69	7,15	6,49	8,46	5,34	8,2	9,23	6,38	8,13	6,19	8,46	5,86	5,86
20	8,75 9,75	6,74 7,74	7,82 8,82	5,55 6,55	3,69 4,69	8,15 9,15	7,49 8,49	9,46 10,46	6,34 7,34	9,2 10,2	10,23 11,23	7,38 8,38	9,13 10,13	7,19 8,19	9,46 10,46	6,86 7,86	6,86 7,86
	en RRA's				· · ·						· · · ·	0,30	10,13	0,19	10,46	7,00	/,00
					Ro	und Ro	bin Arbi	iter AR	EA Co	mpariso	on for 8	Bit Inp	ut				
Time (ns)	STA N2N	STA PPT	STA H2LOGH	CHN PPE Conflict	Ro CHN PPE Conflict PPT	und Rol CHN PPE Non Conflict	Din Arbi CHN PPE Non Conflict PPT	iter <u>AR</u> вк_вт		mpariso HC_BT	НС_ВТ	<u>Bit Inp</u> нс_rs	ut Ks_bt	KS_RS	LF_BT	LF_RS	BOWTIE
				PPE	CHN PPE Conflict	CHN PPE Non	CHN PPE Non Conflict				HC_BT Simple			KS_RS 209	LF_BT	LF_RS 212	BOWTIE
(ns) 6 7	N2N	РРТ	N2LOGN	PPE Conflict	CHN PPE Conflict PPT	CHN PPE Non Conflict	CHN PPE Non Conflict PPT	BK_BT	BK_RS	НС_ВТ	HC_BT Simple Mux	HC_RS	KS_BT				
(ns) 6 7 8	N2N 180	РРТ 188	N2LOGN 225	PPE Conflict 246	CHN PPE Conflict PPT 262	CHN PPE Non Conflict 236	CHN PPE Non Conflict PPT 244	ВК_ВТ 186	BK_RS	HC_BT	HC_BT Simple Mux 190	HC_RS	KS_BT 230 206 177	209	222	212	437
(ns) 6 7	N2H 180 179	PPT 188 188	N2LOGN 225 231	PPE Conflict 246 246	CHN PPE Conflict PPT 262 249	CHN PPE Non Conflict 236 237	CHN PPE Non Conflict PPT 244 244	ВК_ВТ 186 204	BK_RS 213 213	HC_BT	HC_BT Simple Mux 190 183	HC_RS 207 207	KS_BT 230 206	209	222 223	212	437 440
(ns) 6 7 8 9 10	H2H 180 179 170 167 138	PPT 188 188 188 187 189 153	H2LOGH 225 231 232 196 182	PPE Conflict 246 246 241 241 225	CHIN PPE Conflict PPT 262 249 251 256 254	CHII PPE Non Conflict 236 237 238 173 161	CHN PPE Non Conflict PPT 244 244 247 195 182	BK_BT 186 204 165 154 153	BK_RS 213 213 215 211 155	HC_BT 199 200 174 154 153	HC_BT Simple Mux 190 183 159 148 145	HC_RS 207 207 206 196 160	KS_BT 230 206 177 161 160	209 211 179 187 156	222 223 169 156 155	212 212 202 206 157	437 440 466 404 387
(ns) 6 7 8 9 10 11	H2H 180 179 170 167 138 132	PPT 188 188 187 189 153 147	H2LOGH 225 231 232 196 182 178	PPE Conflict 246 246 241 241	CHII PPE Conflict PPT 262 249 251 256 254 254 237	CHN PPE Non Conflict 236 237 238 173 161 152	CHIN PPE Non Conflict PPT 244 244 247 195	BK_BT 186 204 165 154 153 153	BK_RS 213 213 215 211 155 149	HC_BT 199 200 174 154 153 153	HC_BT Simple Mux 190 183 159 148	HC_RS 207 207 206 196	KS_BT 230 206 177 161 160 160	209 211 179 187 156 150	222 223 169 156 155 155	212 212 202 206 157 147	437 440 466 404 387 378
(ns) 6 7 8 9 10 11 12	H2H 180 179 170 167 138 132 133	PPT 188 188 187 189 153 147 146	H2LOGH 225 231 232 196 182 178 173	PPE Conflict 246 246 241 241 225 186 166	CHII PPE Conflict PPT 262 249 251 256 254 237 211	CHII PPE Hon Conflict 236 237 238 173 161 152 152	CHII PPE Non Conflict PPT 244 244 244 247 195 182 166 161	BK_BT 186 204 165 154 153 153 153	BK_RS 213 213 215 211 155 149 144	HC_BT 199 200 174 154 153 153 153	HC_BT Simple Mux 190 183 159 148 145 145 145	HC_RS 207 207 206 196 160 148 144	KS_BT 230 206 177 161 160 160 160	209 211 179 187 156 150 148	222 223 169 156 155 155 155	212 212 202 206 157 147 145	437 440 466 404 387 378 364
(ns) 6 7 8 9 10 11 12 13	H2H 180 179 170 167 138 132 133 133	PPT 188 188 187 189 153 147 146 146	H2LOGH 225 231 232 196 182 178 173 173	PPE Conflict 246 246 241 241 225 186 166 160	CHII PPE Conflict PPT 262 249 251 256 254 237 211 177	CHII PPE Hon Conflict 236 237 238 173 161 152 152 152	CHII PPE Hon Conflict PPT 244 244 244 247 195 182 166 161 161	BK_BT 186 204 165 154 153 153 153 153	BK_RS 213 213 215 211 155 149 144 144	HC_BT 199 200 174 153 153 153 153	HC_BT Simple Mux 190 183 159 148 145 145 145 145	HC_RS 207 207 206 196 160 148 144 144	KS_BT 230 206 177 161 160 160 160 160	209 211 179 187 156 150 148 148	222 223 169 156 155 155 155 155	212 212 202 206 157 147 145 145	437 440 466 404 387 378 364 364 362
(ns) 6 7 8 9 10 11 12 13 14	H2H 180 179 170 167 138 132 133 133 133	PPT 188 188 187 189 153 147 146 146 146 146	N2LOGN 225 231 232 196 182 178 173 173 173	PPE Conflict 246 246 241 241 241 225 186 166 166 160	CHII PPE Conflict PPT 262 249 251 256 254 237 211 177 171	CHII PPE Ion Conflict 236 237 238 173 161 152 152 152	CHII PPE Hon Conflict PPT 244 244 244 247 195 182 166 161 161	BK_BT 186 204 165 154 153 153 153 153	BK_RS 213 213 215 211 155 149 144 144	HC_BT 199 200 174 154 153 153 153 153	HC_BT Simple Mux 190 183 159 148 145 145 145 145 145	HC_RS 207 207 206 196 160 148 144 144	KS_BT 230 206 177 161 160 160 160 160	209 211 179 187 156 150 148 148 148	222 223 169 155 155 155 155 155 155	212 212 202 206 157 147 145 145 145	437 440 466 404 387 378 364 362 362 362
(ns) 6 7 8 9 10 11 12 13 14 15	II2II 180 179 170 167 138 132 133 133 133	PPT 188 188 187 189 153 147 146 146 146 146 146	N2LOGN 225 231 232 196 182 178 173 173 173 173	PPE Conflict 246 246 241 241 225 186 166 166 160 158 156	CHII PPE Conflict PPT 262 249 251 256 254 237 211 177 171	CHII PPE IIon Conflict 236 237 238 173 161 152 152 152 152 152	CHII PPE IIon Conflict PPT 244 244 247 195 182 166 161 161 161	BK_BT 186 204 165 154 153 153 153 153 153 153	BK_RS 213 213 215 211 155 149 144 144 144	HC_BT 199 200 174 154 153 153 153 153 153 153	HC_BT Simple Mux 190 183 159 148 145 145 145 145 145 145	HC_RS 207 207 206 196 160 148 144 144 144	KS_BT 230 206 177 161 160 160 160 160 160	209 211 179 187 156 150 148 148 148 148	222 223 169 155 155 155 155 155 155 155	212 212 202 206 157 147 145 145 145 145	437 440 466 404 387 378 364 362 362 362 362
(ns) 6 7 8 9 10 11 12 13 14 15 16	II2II 180 179 170 167 138 132 133 133 133 133 133 133	PPT 188 188 187 189 153 147 146 146 146 146 146	N2LOGN 225 231 232 196 182 178 173 173 173 173 173 173	PPE Conflict 246 246 241 241 225 186 166 160 158 156	CHII PPE Conflict PPT 262 249 251 256 254 254 237 211 1177 171 166 166	CHII PPE Ilon Conflict 236 237 238 173 161 152 152 152 152 152 152 152 152	CHII PPE Hon Conflict PPT 244 244 247 195 182 166 161 161 161 161 161	BK_BT 186 204 165 154 153 153 153 153 153 153 153 153	BK_RS 213 213 215 211 155 149 144 144 144 144	HC_BT 199 200 174 153 153 153 153 153 153 153 153	HC_BT Simple Mux 190 183 159 148 145 145 145 145 145 145 145 145 145	HC_RS 207 207 206 196 160 148 144 144 144 144	KS_BT 230 206 1777 161 160 160 160 160 160 160 160	209 211 179 187 156 150 148 148 148 148 148	222 223 169 155 155 155 155 155 155 155 155	212 212 202 206 157 147 145 145 145 145 145 145	437 440 466 404 387 378 364 362 362 362 362 362
(ns) 6 7 8 9 10 11 12 13 14 15 16 17	H2N 180 179 170 138 133 133 133 133 133 133 133 133 133 133	PPT 188 188 187 189 153 147 146 146 146 146 146 146	N2LOGN 225 231 232 196 182 173 173 173 173 173 173 173 173	PPE Conflict 246 246 241 241 225 186 166 160 158 156 156	CHII PPE Conflict PPT 262 249 251 256 254 254 237 211 177 171 166 166	CHII PPE Non Conflict 236 237 238 173 161 152 152 152 152 152 152 152	CHII PPE Hon Conflict PPT 244 244 247 195 182 166 161 161 161 161 161	BK_BT 186 204 165 154 153 153 153 153 153 153 153 153	BK_RS 213 213 215 211 155 149 144 144 144 144 144	HC_BT 199 200 174 153 153 153 153 153 153 153 153	HC_BT Simple Mux 190 183 159 148 145 145 145 145 145 145 145 145 145 145	HC_RS 207 207 206 196 160 148 144 144 144 144 144	KS_BT 230 206 1777 161 160 160 160 160 160 160 160 160	209 211 179 187 156 150 148 148 148 148 148	222 223 169 155 155 155 155 155 155 155 155	212 212 202 206 157 147 145 145 145 145 145 145	437 440 466 404 387 378 364 362 362 362 362 362 362 362
(ns) 6 7 8 9 10 11 12 13 14 15 16 17 18	H2N 180 179 170 167 138 133 133 133 133 133 133 133 133 133 133 133 133 133 133 133	PPT 188 188 189 153 153 145 146 146 146 146 146 146 146 146	N2LOGN 225 231 232 196 182 173 173 173 173 173 173 173 173 173 173	PPE Conflict 246 246 241 241 225 186 166 160 160 156 156 156	CHII PPE Conflict PPT 262 249 251 256 254 237 211 177 1166 166 166 166	CHII PPE Ilon Conflict 236 237 238 173 161 152 152 152 152 152 152 152 152 152	CHII PPE Non Conflict PPT 244 244 244 244 245 185 185 185 185 185 185 185 185 185 18	BK_BT 186 204 165 154 153 153 153 153 153 153 153 153	BK_RS 213 213 211 155 149 144 144 144 144 144 144	HC_BT 199 200 174 153 153 153 153 153 153 153 153	HC_BT Simple Mux 190 183 159 148 145 145 145 145 145 145 145 145 145	HC_RS 207 206 196 160 148 144 144 144 144 144 144	KS_BT 230 206 107 161 160 160 160 160 160 160 160	209 211 179 187 156 150 148 148 148 148 148 148 148	222 223 169 155 155 155 155 155 155 155 155 155	212 212 202 206 157 147 145 145 145 145 145 145 145	437 440 466 404 387 378 364 362 362 362 362 362 362 362 362
(ns) 6 7 8 9 10 11 12 13 14 15 16 17 18 19	H2N 180 179 170 167 138 132 133 133 133 133 133 133 133 133 133 133 133 133	PPT 188 188 187 189 153 145 146 146 146 146 146 146 146 146	N2LOGIN 225 231 232 196 182 178 173 173 173 173 173 173 173 173 173 173	PPE Conflict 246 246 241 225 186 166 156 156 156 156 156 156	CHII PPE Conflict PPT 262 249 251 256 254 237 211 1177 171 166 166 166 166	CHII PPE Ilon Conflict 236 237 238 173 161 152 152 152 152 152 152 152 152 2152 152	CHII PPE Hon Conflict PPT 244 244 244 245 182 166 161 161 161 161 161 161 161	BK_BT 186 204 165 154 153 153 153 153 153 153 153 153	BK_RS 213 213 215 211 155 149 144 144 144 144 144 144 144	HC_BT 199 200 174 154 153 153 153 153 153 153 153 153	HC_BT Simple Mux 1900 1833 159 148 145 145 145 145 145 145 145 145 145 145	HC_RS 207 206 196 160 148 144 144 144 144 144 144 144	KS_BT 230 206 177 161 160 160 160 160 160 160 160	209 211 179 187 156 150 148 148 148 148 148 148 148 148 148	222 223 169 155 155 155 155 155 155 155 155 155 15	212 212 202 206 157 147 145 145 145 145 145 145 145 145 145 145	437 440 466 404 387 378 364 362 362 362 362 362 362 362 362 362 362
(ns) 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	H2N 180 179 170 167 138 133 133 133 133 133 133 133 133 133 133 133 133 133 133 133	PPT 188 188 187 153 147 146 146 146 146 146 146 146 146	N2LOGIN 225 231 232 196 182 178 173 173 173 173 173 173 173 173 173 173	PPE Conflict 246 246 241 241 225 186 166 156 156 156 156 156 156 156	CHII PPE Conflict PPT 262 249 251 256 254 237 211 177 171 166 166 166 166 166 166	CHII PPE Ilon Conflict 236 237 238 173 161 152 152 152 152 152 152 152 152 2152 152	CHII PPE Non Conflict PPT 244 244 244 247 195 182 166 161 161 161 161 161 161 161 161 16	BK_BT 186 204 165 154 153 153 153 153 153 153 153 153	BK_RS 213 213 215 211 155 149 144 144 144 144 144 144 144 144	HC_BT 199 200 174 153 153 153 153 153 153 153 153	HC_BT Simple Mux 1900 1833 159 148 145 145 145 145 145 145 145 145 145 145	HC_RS 207 206 196 160 148 144 144 144 144 144 144	KS_BT 230 206 107 161 160 160 160 160 160 160 160	209 211 179 187 156 150 148 148 148 148 148 148 148	222 223 169 155 155 155 155 155 155 155 155 155	212 212 202 206 157 147 145 145 145 145 145 145 145	437 440 466 404 387 378 364 362 362 362 362 362 362 362 362

Figure 6.1: 8 Bit RRAs synthesis results

					Rour	nd Robi	n Arbite	rs TIM	ING Co	mparis	on for 1	6Bit Ir	nbut				ĺ
Time (ns)	STA N2N	STA PPT	STA N2LOG N	CHN PPE Conflict	CHN PPE Conflict	CHN PPE Non Conflict	CHN PPE Non Conflict PPT	ВК_ВТ		НС_ВТ	HC_BT Simple Mux	HC_RS	KS_BT	KS_RS	LF_BT	LF_RS	BOWTIE
6	-4,63	-5,11	-3,88	-7,43	-8,3	-4,02	-4,47	-2,39	-4,89	-2,39	-2,1	-4,8	-2,27	-4,48	-2,33	-4,39	-4,91
7	-3,63	-4,11	-2,76	-6,43	-7,3	-3,02	-3,47	-1,32	-3,77	-1,25	-1,1	-3,8	-1,17	-3,48	-1,27	-3,39	-4,2
8	-2,63	-3,08	-1,76	-5,46	-6,3	-2,04	-3,13	-0,38	-2,77	-0,24	-0,12	-2,8	-0,27	-2,48	-0,27	-2,39	-3,32
9	-1,79	-2,08	-0,87	-4,46	-5,3	-1,05	-1,98	0,01	-1,77	0	0	-1,9	0,02	-1,48	0,01	-1,41	-2,03
10	-0,62	-1,05	0	-3,22	-4,34	-0,18	-0,99	0,2	-0,93	0,11	0,02	-0,92	0,04	-0,46	0,02	-0,47	-1,34
11	0,02	0,01	0,04	-2,4	-3,01	0,01	0	0,11	0	0,36	0,11	0	0,36	0	0,51	0	-0,5
12	0,01	0,06	0,03	-1,59	-2,51	0	0,02	0,03	0,06	0,03	0,24	0	0,07	0	0,55	0,02	0
13	0,94	0,17	0,13	-0,62	-1,02	0,03	0,01	0,37	0,02	0,97	2,63	0,02	0,75	0,04	0,6	0,07	0,06
14	0,95	0,1	0,05	0,06	-0,02	0,32	0,13	0,06	0,01	0,66	3,63	0,3	0,59	0,04	1,6	0,23	0,08
15	0,89	0,06	0,62	0,12	0		1,13	1,06	0,16	1,66	4,63	0,93	1,59	0,47	0,54	0,19	0,03
16	2,14	0,33	0,25	0,15	0,03	0,27	0,46	2,06	0,13	2,66	5,63	0,66	2,59	1,37	1,54	1,04	0,49
17	3,14	1,33	0,12	0,13	0,02	1,27	1,46	3,06	0,49	3,66	6,63	1,26	3,59	0,76	2,54	0,98	1,49
18	4,14	2,33	1,12	0,24	0,13	0,65	0,85	4,06	0,46	4,66	7,63	0,65	4,59	1,76	3,39	1,37	2,49
19	5,14	3,33	2,12	0,84	0,27	1,65	1,85	5,06	1,46	5,66	8,63	1,65	5,59	2,76	4,39	0,75	3,49
20	6,14	4,33	3,12	0,79	0,87	4,27	2,85	6,06	0,85	6,66	9,63	2,65	6,59	3,76	5,39	1,75	4,49
Note: Gre	en RRA's	are our a	rchitectur	es, orange	e and blue	ones are	rival arch	illectures a	and their n	nodificatio	ns.						
			STA	es, orange	Rou CHN	und Rob CHN		ter ARE				Bit Inp	out				
Time (ns)	STA N2N	STA PPT			Roi	ind Rob	in Arbit CHN				n for 16	Bit Inp hc_rs	out ks_вт	KS_RS	LF_BT	LF_RS	BOWTIE
Time (ns)	STA N2H 412	STA PPT 409	STA N2LOG H	CHII PPE Conflict 516	Rou CHN PPE Conflict PPT 519	Ind Rob CHN PPE Non Conflict 510	in Arbit CHII PPE Non Conflict PPT 544	ter ARE вк_вт 393	EA Con BK_RS 405	npariso HC_BT 403	n for 16 HC_BT Simple Mux 425	HC_RS 411	KS_BT 496	409	415	425	920
Time (ns) 6 7	STA H2H 412 411	STA PPT 409 409	STA H2LOG H 515 498	CHII PPE Conflict 516 516	Rou CHN PPE Conflict PPT 519 570	Ind Rob CHN PPE Non Conflict 510 508	in Arbit CHN PPE Non Conflict PPT 544 544	вк_вт 393 414	EA Con BK_RS 405 428	npariso HC_BT 403 421	n for 16 HC_BT Simple Mux 425 429	HC_RS 411 411	KS_BT 496 523	409	415	- 425 425	920 906
Time (ns) 6 7 8	STA H2H 412 411 410	STA PPT 409 409 390	STA H2LOG H 515 498 501	CHN PPE Conflict 516 516 516 510	Rou CHII PPE Conflict PPT 519 570 570	Ind Rob CHN PPE Hon Conflict 510 508 493	in Arbit CHN PPE Non Conflict PPT 544 544 481	вк_вт 393 414 400	EA Con BK_RS 405 428 428	npariso HC_BT 403 421 421	n for 16 HC_BT Simple Mux 425 429 405	HC_RS 411 411 411	KS_BT 496 523 501	409 409 403	415 440 435	- 425 425 430	920 906 868
Time (ns) 6 7 8 9	STA H2H 412 411 410 393	STA PPT 409 409 390 390	STA HI2LOG II 515 498 501 507	CHII PPE Conflict 516 516 516 510 512	Rou CHII PPE Conflict PPT 519 570 570 570 569	Ind Rob CHII PPE Hon Conflict 510 508 493 448	in Arbit CHN PPE Non Conflict PPT 544 544 481 482	ter ARE BK_BT 393 414 400 361	EA Con BK_RS 405 428 428 418	HC_BT 403 421 421 349	n for 16 HC_BT Simple Mux 425 429 405 322	HC_RS 411 411 411 391	KS_BT 496 523 501 374	409 409 403 403	415 440 435 352	425 425 430 393	920 906 868 874
<i>Time</i> (<i>ns</i>)	STA H2H 412 411 410 393 372	STA PPT 409 409 390 390 411	STA H2LOG H 515 498 501 507 441	CHII PPE Conflict 516 516 516 510 512 542	Rou CHII PPE Conflict PPT 519 570 570 570 569 535	Ind Rob CHII PPE Hon Conflict 510 508 493 448 425	in Arbit CHII PPE Non Conflict PPT 544 544 481 482 483	BK_BT 393 414 400 361 320	EA Con BK_RS 405 428 428 418 379	HC_BT 403 421 421 349 326	n for 16 HC_BT Simple Mux 425 429 405 322 311	HC_RS 411 411 411 391 382	KS_BT 496 523 501 374 352	409 409 403 403 364	415 440 435 352 327	425 425 430 393 385	920 906 868 874 877
<i>Time</i> (<i>ns</i>)	STA 11211 4112 4111 4110 3933 3722 378	STA PPT 409 409 390 390 411 409	STA H2LOG H 515 498 501 507 441 422	CHII PPE Conflict 516 516 510 512 542 487	Rot PPE Conflict PPT 519 570 570 570 589 535 547	Ind Rob CHII PPE Hon Conflict 510 508 493 448 425 346	in Arbit CHII PPE Non Conflict PPT 544 544 481 482 483 417	вк_вт Вк_вт 393 414 400 361 320 316	EA Con BK_RS 405 428 428 418 379 377	HC_BT 403 421 421 349 326 326	n for 16 HC_BT Simple Mux 425 429 405 322 311 311	HC_RS 411 411 391 382 363	KS_BT 496 523 501 374 352 352	409 409 403 403 364 329	415 440 435 352 327 326	425 425 430 393 385 332	920 906 868 874 877 879
Time (ns) 6 7 8 9 10 11 12	STA 11211 4112 4111 4110 3933 3722 378 307	STA PPT 409 390 390 411 409 329	STA H2LOG II 515 498 501 507 441 422 414	CHII PPE Conflict 516 516 510 512 542 487 533	Rot PPE Conflict PPT 519 570 570 570 570 570 570 570 570 570 570	CHII PPE Hon Conflict 510 508 493 448 425 346 327	in Arbit CHII PPE Non Conflict PPT 544 544 481 482 483 417 357	BK_BT 393 414 400 361 320 316 315	EA Con BK_RS 405 428 428 418 379 377 321	HC_BT 403 421 421 326 326 325	n for 16 HC_BT Simple Mux 425 429 405 322 311 311 310	HC_RS 411 411 391 382 363 319	KS_BT 496 523 501 374 352 352 351	409 409 403 403 364 329 315	415 440 435 352 327 326 325	425 425 430 393 385 332 304	920 906 868 874 877 879 800
Time 6 7 8 9 10 11 12 13	STA H2H 412 411 410 393 372 378 307 289	STA PPT 409 409 390 390 411 409 329 318	STA H2LOG H 515 498 501 507 441 422 414 390	CHII PPE Conflict 516 516 510 512 542 487 533 512	Rot CHII PPE Conflict PPT 519 570 569 535 547 479 600	Ind Rob CHII PPE Hon Conflict 510 508 493 448 425 346 327 317	in Arbit CHII PPE Hon Conflict PPT 544 544 481 482 483 417 357 342	BK_BT 393 414 400 361 320 316 315 315	EA Con BK_RS 405 428 418 379 377 321 307	HC_BT 403 403 421 421 326 326 326 325	n for 16 HC_BT Simple Mux 425 429 405 322 311 311 310 309	HC_RS 411 411 391 382 363 319 300	KS_BT 496 523 501 374 352 352 351 351	409 409 403 403 364 329 315 310	415 440 435 352 327 326 325 324	425 425 430 393 385 332 304 299	920 906 868 874 877 879 800 791
<i>Time</i> (<i>ns</i>) 6 7 8 9 10 11 12 13 14	STA H2N 412 411 410 393 372 378 307 289 290	STA PPT 409 409 390 390 411 409 329 318 313	STA H2LOG N 515 498 501 507 441 422 414 422 414 390 383	CHII PPE Conflict 516 516 510 512 542 487 533 512 400	Rot PPE Conflict PPT 519 570 570 559 535 547 479 600 564	Ind Rob PPE Hon Conflict 510 508 493 448 425 346 327 3346 327 317 321	in Arbit CHII PPE Hon Conflict PPT 544 544 481 482 483 417 357 342 339	BK_BT 393 414 400 361 315 315 315 315	EA Con BK_RS 405 428 428 428 428 428 379 377 321 307 295	HC_BT 403 403 421 421 349 326 326 325 325 325	n for 16 HC_BT Simple Mux 425 429 405 322 311 311 311 310 309 309	HC_RS 411 411 391 382 363 319 300 297	KS_BT 496 523 501 374 352 352 351 351 351	409 409 403 403 364 329 315 310 309	415 440 435 352 327 326 325 324 324 324	425 425 430 393 385 332 304 299 297	920 906 868 874 877 879 800 791 756
Fime 6 7 8 9 10 11 12 13 14 15	STA H2H 412 411 410 393 372 378 307 289 290 285	STA PPT 409 390 390 411 409 329 318 313 309	STA H2LOG H 515 498 501 507 441 412 414 390 383 379	Chli PPE Conflict 516 516 510 512 542 487 533 512 400 379	Rot PPE Conflict PPT 519 570 570 570 569 535 547 479 600 564 420	Ind Rob CHII PPE Hon Conflict 510 508 493 448 425 346 327 317 321 321	in Arbit CHN PPE Ion Conflict PPT 544 544 481 482 483 417 357 342 339 339	BK_BT 393 414 400 361 320 316 315 315 315 315 315	EA Con BK_RS 405 428 418 379 377 321 307 295 292	HC_BT 403 421 421 349 326 325 325 325 325 325	n for 16 HC_BT Simple Mux 425 429 405 322 311 311 310 309 309	HC_RS 411 411 391 382 363 319 300 297 297	KS_BT 496 523 501 374 352 352 351 351 351 351	409 409 403 403 364 329 315 310 309 309	415 440 435 352 327 326 325 324 324 324 323	425 425 430 393 385 332 304 299 297 296	920 906 868 874 877 879 800 791 756 751
<i>Time</i> (<i>ns</i>) 6 7 8 9 10 11 12 13 14 15 16	STA H2H 412 411 410 393 372 378 307 289 290 285 293	STA PPT 409 390 390 411 409 329 318 313 309 315	STA H2LOG H 515 498 501 507 441 422 414 390 383 3379 378	CHII PPE Conflict 516 510 512 542 487 533 512 400 379 353	Rot PPE Conflict PPT 519 570 569 535 547 479 600 564 420 385	Ind Rob CHII PPE Hon Conflict 510 508 448 425 346 327 317 321 321 321 320	in Arbit CHI PPE Hon Conflict PPT 544 544 481 482 483 417 357 342 339 339 339	BK_BT 393 414 400 361 320 316 315 315 315 315 315	EA Con BK_RS 405 428 418 379 377 321 307 295 292 292	HC_BT 403 421 421 326 326 325 325 325 325 325	n for 16 HC_BT Simple Mux 425 429 405 322 311 311 310 309 309 309 309	HC_RS 411 411 391 382 363 319 300 297 297 296	KS_BT 496 523 501 374 352 351 351 351 351 351	409 409 403 403 364 329 315 310 309 309 309	415 440 435 352 327 326 325 324 324 324 323 323	425 425 430 393 385 332 304 299 297 296 295	920 906 886 874 877 879 800 791 756 751 746
Time 6 7 8 9 10 11 12 13 14 15 16 17	STA H2H 412 411 411 410 393 372 378 307 289 290 285 293 293	STA PPT 409 409 390 390 390 390 311 309 315 315	STA HI2LOG H 515 498 501 507 441 442 414 390 383 379 378 378 377	CHII PPE Conflict 516 510 542 487 533 512 487 533 512 400 379 353 342	Rot CHII PPE Conflict PPT 519 535 549 535 547 479 600 564 420 385 375	Ind Rob CHII PPE Hon Conflict 510 508 493 4425 346 327 317 321 321 320 320	in Arbit PPE Hon Conflict PPT 544 544 481 482 483 417 357 342 339 338 338 338	BK_BT 393 414 400 361 320 316 315 315 315 315 315 315	EA Con BK_RS 405 428 428 428 428 428 379 377 321 307 295 292 292 292 291	HC_BT 403 421 421 326 326 325 325 325 325 325 325 325	n for 16 HC_BT Simple Mux 425 429 405 322 311 311 311 310 309 309 309 309 309	HC_RS 411 411 391 382 363 319 300 297 297 296 296	KS_BT 496 523 501 374 352 352 351 351 351 351 351 351	409 409 403 364 329 315 310 309 309 309 309 308	415 440 435 352 327 326 325 324 324 323 323 323	425 425 430 393 385 332 304 299 297 296 295 294	920 906 868 874 877 879 800 791 756 751 746 746
<i>Time</i> 6 7 8 9 10 11 12 13 14 15 16 17 18	STA H2H 412 411 410 393 372 378 307 289 290 285 293 293 293 293	STA PPT 409 390 390 411 409 329 318 313 309 315 315 315 315	STA HI2LOG N 515 498 501 507 441 422 414 390 383 379 378 377 377	CHII PPE Conflict 516 516 510 512 542 487 533 512 400 379 353 342 339	Rot CHII PPE Conflict PPT 519 570 550 535 547 479 600 564 420 385 335	Ind Rob CHII PPE Hon Conflict 510 508 493 448 425 346 327 317 321 321 321 320 320 320 319	in Arbit PPE Hon Conflict PPT 544 544 481 482 483 417 357 342 339 338 338 338 338	BK_BT 393 414 400 361 315 315 315 315 315 315 315 31	EA Con BK_RS 405 428 428 418 379 321 307 295 292 292 292 291 291	HC_BT 403 421 421 326 326 325 325 325 325 325 325 325 325 325	n for 16 HC_BT Simple Mux 425 429 405 322 311 311 311 310 309 309 309 309 309 309 309	HC_RS 4111 4111 3911 3822 3633 319 3000 2977 2976 2976 2976 2926 2925	KS_BT 496 523 501 374 352 351 351 351 351 351 351 351 351	409 409 403 364 329 315 310 309 309 309 309 308 308	415 440 435 352 327 326 325 324 324 323 323 323 323 323	425 425 430 393 385 332 304 299 297 296 295 294 294	920 906 868 874 877 879 800 791 756 751 746 746 746 746
Fime 6 7 8 9 10 11 12 13 14 15 16 17 18 19	STA 1211 4112 4110 4110 3933 372 378 307 289 2900 285 2933 2933 2933 2933 2933	STA PPT 409 409 390 390 411 409 329 318 313 309 315 315 315 315 315	STA H2LOG H 515 507 441 422 414 390 383 379 378 377 377 377	CHII PPE Conflict 516 510 510 512 542 487 533 512 400 379 353 342 339 339	Rot CHII PPE Conflict PPT 519 570 570 570 570 5570 5570 568 547 479 600 564 420 385 375 359 355	Ind Rob CHII PPE Hon Conflict 510 508 493 448 425 3448 425 3448 425 3448 425 344 327 317 321 321 321 320 320 319 319	in Arbit CHI PPT 544 544 544 481 482 483 417 357 342 339 339 339 338 338 337	BK_BT 393 414 400 361 315 315 315 315 315 315 315 31	EA Con BK_RS 405 428 428 428 428 428 307 321 307 235 292 292 292 291 291 291 291	HC_BT 403 421 421 421 326 325 325 325 325 325 325 325 325 325 325	n for 16 HC_BT Simple Mux 425 429 405 322 311 311 310 309 309 309 309 309 309 309 309	HC_RS 411 411 391 382 363 319 300 297 296 296 295 295 295	KS_BT 496 523 501 374 352 351 351 351 351 351 351 351 351	409 409 403 364 329 315 310 309 309 309 309 309 308 308 308	415 440 435 352 327 326 325 324 324 324 323 323 323 323 323 323	425 425 430 393 385 332 304 299 297 296 295 294 294 294 293	920 906 888 874 877 879 800 791 756 751 746 746 746 746 746
6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	STA H2H 412 411 410 393 372 378 307 289 290 285 293 293 293 293	STA PPT 409 409 390 411 409 329 318 313 309 315 315 315 315 315 315	STA H2LOG H 515 498 501 507 441 422 414 422 414 390 383 379 378 377 377 377	CHII PPE Conflict 516 516 510 512 542 487 533 512 400 379 353 342 339 339 339 339	Rot CHII PPE Conflict PPT 519 570 570 570 570 589 535 547 479 600 564 420 385 355 359 356 356	Ind Rob CHII PPE IIon Conflict 510 510 508 493 448 425 346 327 317 321 320 320 320 319 319 319 320	in Arbit PPE Non Conflict PPT 544 544 481 482 483 417 357 342 339 338 338 338 338 337 337 337	BK_BT 393 414 400 361 320 316 315 315 315 315 315 315 315 315	EA Con BK_RS 405 428 418 379 377 321 307 295 292 292 292 291 291 291 291 291 291	HC_BT 403 421 421 349 326 325 325 325 325 325 325 325 325 325 325	n for 16 HC_BT Simple Mux 425 429 405 322 311 311 311 310 309 309 309 309 309 309 309 309 309 30	HC_RS 4111 4111 3911 3822 3633 319 3000 2977 2976 2976 2976 2926 2925	KS_BT 496 523 501 374 352 351 351 351 351 351 351 351 351	409 409 403 364 329 315 310 309 309 309 309 308 308	415 440 435 352 327 326 325 324 324 323 323 323 323 323	425 425 430 393 385 332 304 299 297 296 295 294 294	920 906 868 874 877 879 800 791 756 751 746 746 746 746

Figure 6.2: 16 Bit RRAs synthesis results

					Rour	nd Robii	n Arbite	rs TIMI	NG Co	mparis	on for 3	2Bit Ir	nput				
Time (ns)	STA N2N	STA PPT	STA N2LOGN	CHN PPE Conflict	CHN PPE Conflict PPT	CHN PPE Non Conflict	CHN PPE Non Conflict PPT	ВК_ВТ	BK_RS	НС_ВТ	HC_BT Simple Mux	HC_RS	KS_BT	KS_RS	LF_BT	LF_RS	BOWTIE
6	-6,55	-6,67	-5,26	-9,75	-10,33	-5,59	-5,79	-3,76	-6,14	-3,12	-2,92	-6,03	-3,21	-6,18	-2,97	-6,2	-8,03
7	-5,81	-5,82	-4,26	-8,81	-9,33	-4,64	-4,79	-2,76	-5,14	-2,13	-1,84	-5,03	-2,14	-5,18	-2	-5,2	-6,8
8	-4,73	-4,81	-3,26	-7,84	-8,33	-3,59	-3,82	-1,76	-4,05	-1	-0,84	-3,87	-1,2	-4,18	-1	-4,2	-5,46
9	-3,58	-3,81	-2	-6,88	-7,34	-2,62	-2,78	-0,63	-3,3	-0,24	0	-3,01	-0,05	-2,83	0	-2,86	-4,52
10	-2,8	-2,74	-0,9	-5,89	-6,34	-1,29	-1,83	0,02	-2,25	0,02	0,01	-2,22	0,01	-1,85	0,01	-2,24	-3,71
11	-1,87	-1,86	-0,13	-4,88	-5,3	-0,34	-0,83	0,02	-0,89	0,19	0,06	-1,08	0,19	-0,85	0,01	-1,1	-2,58
12	-0,55	-0,8	0,05	-3,8	-5,01	0,01	0	0,1	-0,59	0,43	0,35	0	0,43	0,01	0,1	0	-1,93
13	0	0	0,04	-3,13	-3,61	0,01	0	0,22	0	0,5	0,35	0	0,32	0,03	0,37	0	-1,3
14	0,03	0,01	0,01	-2,06	-2,54	0,17	0,04	0,52	0,01	1,15	1,29	0,02	1,22	0,31	0,37	0	0
15	0,02	0,04	0,01	-1,32	-1,41	0,11	0,29	1,35	0	2,15	1,02	0,14	2,22	0,6	0,53	0,18	0
16 17	0	0,08	0	0	-0,96	0,07	0,38	2,35	0,03	3,15	1,22	0,69	3,22	0,64	0,67	0,01	0,02
17	0,17	0,15	0,12	0,11	0	-1	0,33	3,35	0,14	4,15	1,96	0,35	4,22	0,94	2,16	0,1	0,05
10	0,24	0,02	0,2	0,02	0,01	0,38	0,82	0,17	0,49	3,08	5,29	0,83	3,14	1,6	3,05	0,77	0,01
	1,14	0,72	0,48	0,05	0,01	0,51	1,82	0,3 1,3	0,43	4,08 0,85	6,29 7,29	1,62 2,62	4,14 0,83	2,6 3,6	3,67 4,67	0,29	0
		0.61	0.35	0.16													
20	0,65	0,61 are our a	0,35 rchitecture	0,16 es, orange	0,04 and blue	0,52 ones are	2,82 rival archi					2,02	0,00				
20	0,65			es, orange	and blue	ones are	rival archi in Arbit CHN	tectures a	and their m	nodification	ns. n for 32	Bit Inp					
20	0,65				and blue	ones are Ind Rob	rival archi in Arbit	tectures a	A Con	nodification	ns.			KS_RS	LF_BT		BOWTIE
20 Note: Gree	0,65 en RRA's STA	are our a	sta	CHIN PPE	and blue Rou CHN PPE Conflict	ones are Ind Rob CHN PPE Non	rival archi In Arbit CHN PPE Non Conflict	tectures a	A Con	npariso	n for 32 HC_BT Simple	Bit Inp	out				BOWTIE 1743
20 Note: Gree Time (ns)	0,65 en RRA's STA N2N	are our a STA PPT	STA N2LOGN	CHN PPE Conflict	And blue Rou CHN PPE Conflict PPT	ones are Ind Rob CHN PPE Non Conflict	rival archi CHN PPE Non Conflict PPT	er ARE BK_BT	A Con BK_RS	nparison HC_BT	ns. n for 32 HC_BT Simple Mux	Bit Inp нс_rs	DUT KS_BT	KS_RS	LF_BT	LF_RS	
20 Note: Gree Time (ns) 6	0,65 en RRA's STA N2H 952	are our a STA PPT 1004	STA N2LOGH	CHIN PPE Conflict 1169	eand blue Rou CHN PPE Conflict PPT	ones are CHN PPE Non Conflict 1077	in Arbit CHN PPE Non Conflict PPT 1054	er ARE BK_BT 756	A Con BK_RS 937	nparison HC_BT 866	n for 32 HC_BT Simple Mux 878	Bit Inp HC_RS 924	о <i>ut</i> кs_вт 980	KS_RS 826	LF_BT 966	LF_RS 835	1743
20 Note: Gree <i>Time</i> (ns) 6 7 8 9	0,65 en RRA's STA II211 952 841	are our a STA PPT 1004 981	STA H2LOGH	CHIN PPE Conflict 1169 1185	and blue Rou CHII PPE Conflict PPT 1249 1249	Ind Rob CHII PPE Hon Conflict 1077 1052	in Arbit CHN PPE Non Conflict PPT 1054 1054	er ARE BK_BT 756 756	A Con BK_RS 937 937	nparison HC_BT 866 862	n for 32 HC_BT Simple Mux 878 951	Bit Inp HC_RS 924 924	KS_BT 980 992	KS_RS 826 828	LF_BT 966 980	LF_RS 835 835	1743 1794
20 Note: Gree Time (ns) 6 7 8 9 10	0,65 en RRA's STA 11211 952 841 890 902 884	are our a STA PPT 1004 981 964	STA H2LOGH 1024 1024 1024	CHII PPE Conflict 1169 1185 1139	Rou CHII PPE Conflict PPT 1249 1249 1248	Ind Rob CHII PPE Hon Conflict 1077 1052 1086	in Arbit CHN PPE Non Conflict PPT 1054 1054	er ARE BK_BT 756 756 751	EA Con BK_RS 937 937 993	HC_BT 8866 862 973	ns. n for 32 HC_BT Simple Mux 878 951 951	Bit Inp HC_RS 924 924 940	KS_BT 980 992 988	KS_RS 826 828 828	LF_BT 966 980 952	LF_RS 835 835 842	1743 1794 1801
20 Note: Gree <i>Time</i> (<i>ns</i>) 6 7 8 9 10 11	0,65 en RRA's STA II2II 952 841 890 902 884 870	are our a STA PPT 1004 981 964 923 945 880	STA H2LOGH 1024 1024 1024 1025 1058 1019	CHII PPE Conflict 1169 1185 1139 1190 1116 1135	and blue Rou CHII PPE Conflict PPT 1249 1249 1248 1251 1251 1245	ones are ind Rob CHII PPE Hon Conflict 1077 1052 1086 1046 1056 1027	in Arbit CHII PPE Non Conflict PPT 1054 1054 1055 1079 1039 1046	er ARE BK_BT 756 756 751 758 721 653	EA Con BK_RS 937 933 971 993 971 990 989	HC_BT 866 862 973 882 701 686	ns. n for 32 HC_BT Simple Mux 878 951 951 862 699 653	Bit Inp HC_RS 924 924 940 909 916 850	KS_BT 980 992 988 1038 785 769	KS_RS 826 828 828 964 880 868	LF_BT 966 980 952 875 776 697	LF_RS 835 835 842 920 860 814	1743 1794 1801 1774 1780 1801
20 Note: Gree (ns) 6 7 8 9 10 11 12	0,65 en RRA's STA II2II 952 841 890 902 884 870 832	are our a STA PPT 1004 981 964 923 945 880 930	STA H2LOGH 1024 1024 1029 1058 1019 899	CHII PPE Conflict 1169 1185 1139 1190 1116 1135	Rou CHII PPE Conflict PPT 1249 1249 1248 1251 1251 1251 1251 1251 1251 1251	ones are ind Rob CHII PPE Hon Conflict 1077 1052 1086 1046 1056 1027 893	rival archi in Arbit CHII PPE Non Conflict PPT 1054 1054 1054 1059 1039 1039 1039 1046 942	er ARE BK_BT 756 756 751 758 721 653 644	BK_RS 937 937 993 971 990 989 868	HC_BT 866 862 973 882 701 686 682	ns. n for 32 HC_BT Simple Mux 878 951 951 862 699 653 655	Bit Inp HC_RS 924 924 940 909 916 850 920	980 980 992 988 1038 785 769 764	KS_RS 826 828 828 964 880 868 808	LF_BT 966 980 952 875 776 697 681	LF_RS 835 835 842 920 860 814 878	1743 1794 1801 1774 1780 1801 1751
20 Note: Gree Time (ns) 6 7 8 9 10 11 11 12 13	0,65 en RRA's STA H2H 952 841 890 902 884 870 884 870 832 785	are our a STA PPT 1004 981 964 923 945 880 930 825	STA N2LOGH 1024 1024 1024 1029 1059 1019 899 853	CHII PPE Conflict 1169 1185 1139 1190 1116 1135 1135	Rou CHII PPE Conflict PPT 1249 1249 1249 1248 1251 1251 1251 1251 1251 1251 1272 1172	ones are md Rob CHII PPE Hon Conflict 1077 1052 1086 1046 1056 1027 893 756	rival archi in Arbit CHII PPE Hon Conflict PPT 1054 1054 1059 1039 1039 1039 1046 942 795	er ARE BK_BT 756 751 758 721 653 644 643	BK_RS 937 937 933 971 993 971 993 971 993 971 993 971 993 971 993 971	HC_BT 866 862 973 882 701 686 682 680	ns. hfor 32 HC_BT Simple Mux 878 951 951 862 699 653 655 652	Bit Inp HC_RS 924 924 929 909 916 850 920 686	KS_BT 980 992 988 1036 785 769 764 762	KS_RS 826 828 828 964 880 868 808 868 808 687	LF_BT 966 980 952 875 776 697 681 677	LF_RS 835 835 842 920 860 814 878 687	1743 1794 1801 1774 1780 1801 1751 1762
20 Note: Gree Time (ns) 6 7 8 9 10 11 12 13 14	0,65 en RRA's STA H2H 952 841 890 902 884 832 785 680	are our a STA PPT 1004 981 964 923 945 880 930 825 727	STA H2LOGH 1024 1024 1024 1059 1058 1019 899 853 810	CHII PPE Conflict 1169 1185 1139 1190 1116 1135 1135 1064 1072	Rou CHII PPE Conflict PPT 1249 1249 1249 1248 1251 1251 1245 1245 1245 1245 1245 1245	ones are md Rob CHII PPE Hon Conflict 1077 1052 1086 1046 1056 1027 893 756 723	rival archi in Arbit CHII PPE Hon Conflict PPT 1054 1054 1055 1079 1039 1039 1039 1039 1046 942 795 741	er ARE BK_BT 756 755 751 758 721 653 644 643 643	EA Con BK_RS 937 937 993 971 990 989 868 762 628	HC_BT 866 862 973 882 701 686 680 680	n for 32 HC_BT Simple Mux 878 951 951 862 699 653 655 652 647	Bit Inp HC_RS 924 924 940 909 916 850 920 686 636	KS_BT 980 992 988 1038 785 769 764 762 763	KS_RS 826 828 828 964 880 868 808 808 808 808	LF_BT 966 980 952 875 776 697 681 677 682	LF_RS 835 835 842 920 860 814 878 8687 651	1743 1794 1801 1774 1780 1801 1751 1762 1663
20 Note: Gree (ns) 6 7 8 9 10 11 12 13 14 15	0,65 en RRA's STA II2II 952 841 890 902 884 870 832 785 680 639	are our a STA PPT 1004 981 964 923 945 880 930 825 727 679	STA 1024 1024 1024 1029 1058 1019 899 853 810 810	CHII PPE Conflict 1169 1185 1139 1190 1116 1135 11064 1072 1112	Rou CHII PPE Conflict PPT 1249 1249 1249 1248 1251 1251 1251 1245 1172 1172 1172 1172	ones are Ind Rob CHII PPE Hon Conflict 1077 1052 1086 1046 1056 1046 1056 1027 893 756 723 687	rival archi in Arbit CHII PPE Hon Conflict PPT 1054 1054 1054 1055 1079 1039 1046 942 7955 741 715	er ARE BK_BT 756 756 751 758 721 653 644 643 643 643	EA Con BK_RS 937 933 971 993 971 990 989 868 762 628 609	HC_BT HC_BT 8666 8622 973 8822 701 6865 6800 6800 6800	ns. hs. hc_st Simple Mux 878 951 951 862 699 653 655 652 647 651	Bit Inp HC_RS 924 924 940 909 916 850 920 686 636 617	KS_BT 980 992 988 1038 785 769 764 762 763 763	KS_RS 826 828 964 880 868 808 868 808 687 650 651	LF_BT 966 980 952 875 776 697 681 677 682 676	LF_RS 835 835 842 920 860 814 878 687 651 630	1743 1794 1801 1774 1780 1801 1751 1762 1663 1624
20 Note: Gree (ns) 6 7 8 9 10 11 12 13 14 15 16	0,65 en RRA's STA 11211 952 884 884 890 902 884 884 880 882 785 680 639 639	are our a STA PPT 1004 981 964 923 945 880 930 825 727 679 666	STA H2LOGH 1024 1024 1059 1058 1019 899 853 810 810 810	CHII PPE Conflict 1169 1185 1139 1190 1116 1135 1064 1072 1112 1096	Rou CHII PPE Conflict PPT 1249 1249 1249 1249 1251 1251 1251 1251 1245 1172 1179 1112 11157 1140	ones are md Rob CHII PPE Hon Conflict 1077 1086 1046 1046 1056 1027 893 756 723 687 673	rival archi in Arbit CHII PPE Hon Conflict PPT 1054 1054 1059 1039 1039 1046 942 795 741 715 710	er ARE BK_BT 756 756 751 758 721 653 644 643 643 643 643	EA Con BK_RS 937 937 933 971 990 989 868 762 628 609 595	HC_BT 866 862 973 882 701 686 682 680 680 680 680 680	ns. h for 32 HC_BT Simple Mux 878 951 951 962 699 653 655 652 647 651 648	Bit Inp HC_RS 924 924 940 909 916 850 920 686 636 617 610	KS_BT 980 980 988 1038 769 764 762 763 763 763 763	KS_RS 826 828 964 880 868 808 687 650 651 649	LF_BT 966 980 952 875 776 697 681 677 682 676 670	LF_RS 835 835 842 920 860 814 878 651 651 630 627	1743 1794 1801 1774 1780 1801 1751 1762 1663 1624 1573
20 Note: Gree (ns) 6 7 8 9 10 11 12 13 14 15 16 17	0,65 en RRA's STA H211 952 844 890 902 884 870 832 785 680 639 639 639 639	are our a STA PPT 1004 981 964 923 945 880 930 825 727 679 666 664	STA H2LOGH 1024 1024 1029 1029 1029 1029 1029 1029 1019 899 853 810 810 804 773	CHII PPE Conflict 1169 1185 1139 1190 1116 1135 1135 1135 1135 1064 1072 1112 1096 887	Rou CHII PPE Conflict PPT 1249 1249 1249 1248 1251 1251 1245 1172 1179 1112 1157 1140 1018	ones are ind Rob CHII PPE Ilon Conflict 1077 1052 1086 1046 1026 1027 893 756 723 687 673 670	rival archi in Arbit CHI PPE Hon Conflict PPT 1054 1054 1055 1079 1039 1046 942 795 741 715 7110 7111	er ARE BK_BT 756 756 756 758 758 758 758 758 758 758 758 758 758	BK_RS 937 933 971 993 971 993 989 989 989 989 989 989 989 989 989	HC_BT 866 862 973 882 701 686 682 680 680 680 680 680 680	ns. h for 32 HC_BT Simple Mux 878 951 951 951 951 951 951 955 652 647 651 648 647	Bit Inp HC_RS 924 924 940 909 916 850 920 686 636 617 610 612	KS_BT 980 992 988 1038 769 764 762 763 763 763 763 763 763	KS_RS 826 828 8628 964 8688 868 868 868 868 650 651 649 669	LF_BT 966 980 952 875 776 697 681 677 682 676 670 670 674	LF_RS 835 835 842 920 860 814 878 687 651 637 651 637 626	1743 1794 1801 1774 1780 1801 1751 1762 1663 1624 1573 1553
20 Note: Gree (ns) 6 7 8 9 10 11 12 13 14 15 16 16 17 18	0,65 en RRA's STA H2H 952 841 890 902 884 890 902 884 870 832 785 680 633 639 633 626	are our a STA PPT 1004 981 964 923 945 880 9300 825 727 679 666 664 660	STA H2LOGH 1024 1024 1024 1024 1059 1058 1019 899 853 810 810 810 810 810 873 810 810 810 810 810 810 810 810 810 810	CHII PPE Conflict 1169 1185 1139 1190 1116 1135 1064 1072 1112 1096 887 808	and blue Rou CHII PPE Conflict PPT 1249 1249 1248 1251 1245 1172 1179 1112 1112 1179 1112 1178 3018	ones are md Rob CHII PPE Hon Conflict 1077 1052 1086 1046 1056 1046 1056 1046 1056 1046 1056 1046 1057 1052 1086 1046 1056 1046 1057 1056 1057 1057 1057 1057 1057 1057 1057 10566 1056 1056 1056 105	rival archi in Arbit CHII PPE Hon Conflict PPT 1054 1054 1054 1054 1054 1059 1039 1046 9422 795 741 715 710 711 707	er ARE BK_BT 756 756 751 758 721 653 644 643 643 643 643 643 643	BK_RS 937 933 971 990 989 868 762 628 609 595 590 589	HC_BT HC_BT 866 862 973 882 701 686 682 682 680 680 680 680 680 680 680 680	ns. n for 32 HC_BT Simple Mux 878 9511 9511 862 699 653 655 652 647 651 647 647	Bit Inp 924 924 924 940 909 916 850 920 686 636 617 610 612 608	KS_BT 980 992 988 1038 785 769 764 763 763 763 763 763 763	KS_RS 826 828 964 880 808 808 808 808 687 650 651 649 650 649	LF_BT 966 980 952 875 776 681 687 681 677 682 676 674 672	LF_RS 835 842 920 860 814 878 8687 651 630 651 630 626 624	1743 1794 1801 1774 1780 1801 1751 1762 1663 1624 1573 1553 1553
20 Note: Gree (ns) 6 7 8 9 10 11 12 13 14 15 16 17	0,65 en RRA's STA H211 952 844 890 902 884 870 832 785 680 639 639 639 639	are our a STA PPT 1004 981 964 923 945 880 930 825 727 679 666 664	STA H2LOGH 1024 1024 1029 1029 1029 1029 1029 1029 1019 899 853 810 810 804 773	CHII PPE Conflict 1169 1185 1139 1190 1116 1135 1135 1135 1135 1064 1072 1112 1096 887	Rou CHII PPE Conflict PPT 1249 1249 1249 1248 1251 1251 1245 1172 1179 1112 1157 1140 1018	0nes are Ind Rob CHII PPE Hon Conflict 1077 1052 1086 1046 1056 1027 1027 1052 1086 1046 1056 1027 1056 1027 1056 1027 1052 1086 1056 1027 1056 1027 1052 1086 1057 105	rival archi in Arbit CHI PPE Hon Conflict PPT 1054 1054 1055 1079 1039 1046 942 795 741 715 7110 7111	er ARE BK_BT 756 756 756 758 758 758 758 758 758 758 758 758 758	BK_RS 937 933 971 993 971 993 989 989 989 989 989 989 989 989 989	HC_BT 866 862 973 882 701 686 682 680 680 680 680 680 680	ns. h for 32 HC_BT Simple Mux 878 951 951 951 951 951 951 955 652 647 651 648 647	Bit Inp HC_RS 924 924 940 909 916 850 920 686 636 617 610 612	KS_BT 980 992 988 1038 769 764 762 763 763 763 763 763 763	KS_RS 826 828 8628 964 8688 868 868 868 868 650 651 649 669	LF_BT 966 980 952 875 776 697 681 677 682 676 670 670 674	LF_RS 835 835 842 920 860 814 878 687 651 637 651 637 626	1743 1794 1801 1774 1780 1801 1751 1762 1663 1624 1573 1553

Figure 6.3: 32 Bit RRAs synthesis results

					Rour	nd Robi	n Arbite	rs TIM	ING Co	mnaris	on for 6	4Bit Ir	nout				
Time (ns)	STA N2N	STA PPT	STA N2LOGN	CHN PPE Conflict	CHN PPE Conflict	CHN PPE Non Conflict	CHN PPE Non Conflict PPT	ВК_ВТ		нс_вт	HC_BT Simple Mux	HC_RS	кѕ_вт	KS_RS	LF_BT	LF_RS	BOWTIE
6	-8,87	-8,91	-6,69	-13,37	-13,33	-7,22	-7,47	-4,39	-8,32	-3,85	-3,66	-7,67	-3,76	-7,1	-3,81	-7,64	-10,21
7	-7,79	-7,88	-6,1	-12,37	-12,68	-6,18	-6,47	-3,37	-7,9	-2,97	-2,62	-6,65	-2,76	-6,1	-2,77	-6,6	-9,99
8	-6,46	-6,88	-4,96	-11,07	-11,62	-5,38	-5,62	-2,36	-6,52	-1,97	-1,76	-5,65	-1,75	-5,1	-1,79	-5,54	-9,11
9	-5,64	-6,03	-3,96	-10,64	-10,7	-4,44	-4,62	-1,29	-5,6	-0,97	-0,59	-4,67	-0,79	-4,12		-4,64	-7,91
10	-5,07	-4,81	-2,81	-8,83	-9,66	-3,25	-3,61	-0,37	-4,43	0	0	-3,67	0	-3,18	0	-3,67	-6,05
11	-3,53	-4,06	-1,39	-8,2	-8,51	-2,09	-2,59	0	-3,55	0,02	0,01	-2,7	0,09	-2,18	0,01	-2,52	-5,58
12	-2,59	-2,82	-0,67	-7,09	-7,69	-1,33	-1,52	0,01	-1,86	0,04	0,1	-1,7	0	-1,15	0,01	-1,9	-4,19
13	-1,39	-1,89	0	-5,45	-6,61	-0,44	-0,72	0	-1,25	0,37	0	-0,7	0,21	-0,18	0,01	-0,97	-3,07
14	-0,63	-0,84	0	-5,13	-5,56	0	0,01	0,01	-0,45	0,01	0,02	0	0,01	0,01	0,03	0	-2,27
15	0,01	0	0,01	-4,34	-4,66	0,01	0	0,01	0	0,46	0,04	0,01	0,03	0,03	0,09	0	-1,31
16	0,01	0,01	0,01	-3,23	-3,68	0	0,04	0,02	0	0,29	0,3	0	0,07	0,07	0,25	0,03	-0,31
17	0,02	0,04	0,06	-2,04	-2,73	0,01	0,07	0,03	0,01	0,02	1,53	0,06	0,08	0,08	0,6	0,02	0
18 19	0,38	0,01	0,26	-1,06	-1,58	0,03	0,11	0,09	0,07	0,61	2,75	0,08	0,45	0,36	0,66	0,03	0,02
20	0,06	0.01	0,01	-0,17 0	-0,67 0	0,02	0,05	0,05	0,04	0,91	3,75 0,36	0,41	0,89 1,89	0,13	0,11	0,2	0,02
	en RRA's											0,03	1,03	0,03	1,02	0,10	0,03
Time (ns)	STA N2N	STA PPT	STA	СНИ	Rou CHN	ind Rob CHN	in Arbit CHN	er ARE	A Con	npariso	n for 64	Bit Inp	out				
			N2LOGN	PPE Conflict	PPE Conflict	PPE Non	PPE Non Conflict	вк_вт	BK_RS	НС_ВТ	HC_BT Simple Mux	HC_RS	кѕ_вт	KS_RS	LF_BT	LF_RS	BOWTIE
0	1077	4070		Conflict	Conflict PPT	PPE Non Conflict	Non Conflict PPT		-	_	Simple Mux		_	_	_		
7	1977	1872	2128	Conflict 2283	Conflict PPT 2198	PPE Non Conflict 2019	Non Conflict PPT 1951	- 1603	1608	- 1814	Simple Mux	1676	2067	1764	- 1821	1712	3484
7	1971	1903	2128 2099	2283	2198 2157	PPE Non Conflict 2019 2130	Non Conflict PPT 1951 1949	- 1603 1607	- 1608 1571	- 1814 1713	Simple Mux 1722 1751	- 1676 1670	2067	- 1764 1763	- 1821 1840	- 1712 1708	3484 3450
8	1971 1946	1903 1886	2128 2099 2100	Conflict 2283 2282 2203	2198 2157 2152	PPE Hon Conflict 2019 2130 1986	Non Conflict PPT 1951 1949 1912	- 1603 1607 1598	- 1608 1571 1579	- 1814 1713 1712	Simple Mux 1722 1751 1821	- 1676 1670 1670	2067 2068 2048	1764 1763 1752	- 1821 1840 1784	- 1712 1708 1723	3484 3450 3440
8	1971 1946 1848	1903 1886 1886	2128 2099 2100 2100	2283 2282 2203 2209	2198 2157 2152 2141	PPE Non Conflict 2019 2130 1986 1973	Hon Conflict PPT 1951 1949 1912 1906	1603 1607 1598 1635	1608 1571 1579 1580	- 1814 1713 1712 1716	Simple Mux 1722 1751 1821 1870	1676 1670 1670 1691	2067 2068 2048 2071	1764 1763 1752 1752	- 1821 1840 1784 1738	1712 1708 1723 1689	3484 3450 3440 3481
8	1971 1946 1848 1874	1903 1886 1886 1978	2128 2099 2100 2100 2154	Conflict 2283 2282 2203 2209 2217	2198 2157 2152 2141 2141	PPE Hon Conflict 2019 2130 1986 1973 1983	Hon Conflict PPT 1951 1949 1912 1906 1840	1603 1607 1598 1635 1628	1608 1571 1579 1580 1636	- 1814 1713 1712 1716 1680	Simple Mux 1722 1751 1821 1870 1591	1676 1670 1670 1691 1605	2067 2068 2048 2071 1835	1764 1763 1752 1752 1621	1821 1840 1784 1738 1631	1712 1708 1723 1689 1568	3484 3450 3440 3481 3510
8 9 10	1971 1946 1848 1874 1896	1903 1886 1886 1978 1912	2128 2099 2100 2100 2154 2125	Conflict 2283 2282 2203 2209 2217 2214	2198 2157 2152 2141 2141 2239	PPE Hon Conflict 2019 2130 1986 1973 1983 2017	Hon Conflict PPT 1951 1949 1912 1906 1840 1816	1603 1607 1598 1635 1628 1500	1608 1571 1579 1580 1636 1609	- 1814 1713 1712 1716 1680 1446	Simple Mux 1722 1751 1821 1870 1591 1414	1676 1670 1670 1691 1605 1552	2067 2068 2048 2071 1835 1746	1764 1763 1752 1752 1621 1621	1821 1840 1784 1738 1631 1536	1712 1708 1723 1689 1568 1590	3484 3450 3440 3481 3510 3474
8 9 10 11	1971 1946 1848 1874	1903 1886 1886 1978	2128 2099 2100 2100 2154	Conflict 2283 2282 2203 2209 2217 2214 2262	2198 2157 2152 2141 2141 2239 2224	PPE Hon 2019 2130 1986 1973 1983 2017 1957	Non Conflict PPT 1951 1949 1912 1906 1840 1816 1816 1792	1603 1607 1598 1635 1628 1500 1384	1608 1571 1579 1580 1636 1609 1579	- 1814 1713 1712 1716 1680 1446 1436	Simple Mux 1722 1751 1821 1870 1591 1414 1375	1676 1670 1670 1691 1605 1552 1544	2067 2068 2048 2071 1835 1746 1656	1764 1763 1752 1752 1621 1621 1637	1821 1840 1784 1738 1631 1536 1464	1712 1708 1723 1689 1568	3484 3450 3440 3481 3510
8 9 10 11 12	1971 1946 1848 1874 1896 1956	1903 1886 1886 1978 1912 1881	2128 2099 2100 2100 2154 2125 2206	Conflict 2283 2282 2203 2209 2217 2214	2198 2157 2152 2141 2141 2239	PPE Ilon Conflict 2019 2130 1986 1973 1983 2017 1957 1917	Hon Conflict PPT 1951 1949 1912 1906 1840 1816	1603 1607 1598 1635 1628 1500	1608 1571 1579 1580 1636 1609 1579 1576	- 1814 1713 1712 1716 1680 1446	Simple Mux 1722 1751 1821 1870 1591 1414	1676 1670 1670 1691 1605 1552	2067 2068 2048 2071 1835 1746	1764 1763 1752 1752 1621 1621 1637 1616	1821 1840 1784 1738 1631 1536	1712 1708 1723 1689 1568 1590 1608	3484 3450 3440 3481 3510 3474 3505 3524
8 9 10 11 12 13	1971 1946 1848 1874 1896 1956 1867	1903 1886 1886 1978 1972 1881 1874	2128 2099 2100 2100 2154 2125 2206 2072	Conflict 2283 2282 2203 2209 2217 2214 2262 2289	Conflict PPT 2198 2157 2152 2141 2141 2239 2224 2236	PPE Hon 2019 2130 1986 1973 1983 2017 1957	Hon Conflict PPT 1951 1949 1912 1906 1840 1816 1792 1850	1603 1607 1598 1635 1628 1500 1384 1300	1608 1571 1579 1580 1636 1609 1579	- 1814 1713 1712 1716 1680 1446 1436 1435	Simple Mux 1722 1751 1821 1870 1591 1414 1375 1401	1676 1670 1670 1691 1605 1552 1544 1561	2067 2068 2048 2071 1835 1746 1656 1655	1764 1763 1752 1752 1621 1621 1637	1821 1840 1784 1738 1631 1536 1464 1443	1712 1708 1723 1689 1568 1590 1608 1556	3484 3450 3440 3481 3510 3474 3505
8 9 10 11 12 13 14	1971 1946 1848 1874 1896 1956 1867 1871	1903 1886 1886 1978 1912 1881 1874 1890	2128 2099 2100 2154 2155 2206 2072 1899	Conflict 2283 2282 2203 2209 2217 2214 2262 2289 2227	Conflict PPT 2198 2157 2152 2141 2141 2239 2224 2236 2204	PPE Ilon Conflict 2019 2130 1986 1973 1983 2017 1957 1917 1917	Non Conflict PPT 1951 1949 1912 1906 1840 1816 1792 1850 1660	1603 1607 1598 1635 1628 1500 1384 1300 1298	- 1608 1571 1579 1580 1636 1609 1579 1576 1519	- 1814 1713 1712 1716 1680 1446 1436 1435 1435	Simple Mux 1722 1751 1821 1870 1591 1414 1375 1401 1372	1676 1670 1691 1605 1552 1544 1561 1443	2067 2068 2048 2071 1835 1746 1656 1655 1654	1764 1763 1752 1752 1621 1621 1637 1616 1416	1821 1840 1784 1738 1631 1536 1464 1443 1482	1712 1708 1723 1689 1568 1590 1608 1556 1485	3484 3450 3440 3481 3510 3474 3505 3524 3513
8 9 10 11 12 13 14 15	1971 1946 1848 1874 1896 1956 1867 1871 1631	1903 1886 1978 1978 1912 1881 1874 1890 1801	2128 2099 2100 2154 2125 2206 2072 1899 1748	Conflict 2283 2282 2203 2209 2217 2214 2262 2289 2227 2164	Conflict PPT 2198 2157 2152 2141 2141 2239 2224 2236 2204 2161	PPE Hon Conflict 2019 2130 1986 1973 1983 2017 1983 2017 1957 1917 1917 1719 1619	Non Conflict PPT 1951 1949 1912 1906 1840 1816 1792 1850 1660 1551	1603 1607 1598 1635 1628 1500 1384 1300 1298 1298	- 1608 1571 1579 1580 1636 1609 1579 1576 1519 1395	- 1814 1713 1712 1716 1680 1446 1436 1435 1435 1435	Simple Mux 1722 1751 1821 1870 1591 1414 1375 1401 1372 1371	1676 1670 1691 1695 1552 1544 1561 1443 1317	2067 2068 2048 2071 1835 1746 1656 1655 1654 1717	1764 1763 1752 1752 1621 1621 1621 1637 1616 1416 1417	1821 1840 1784 1738 1631 1536 1464 1443 1482 1408	1712 1708 1723 1689 1568 1590 1608 1556 1485 1334	3484 3450 3440 3481 3510 3474 3505 3524 3513 3476
8 9 10 11 12 13 14 15 16	1971 1946 1848 1874 1896 1956 1867 1867 1871 1631 1479	1903 1886 1978 1912 1881 1874 1890 1801 1532	2128 2099 2100 2154 2125 2206 2072 1899 1748 1618	Conflict 2283 2282 2203 2209 2217 2214 2262 2289 2227 2164 2311	Conflict PPT 2198 2157 2152 2141 2239 2224 2236 2204 2161 2203	PPE Hon Conflict 2019 2130 1986 1973 1983 2017 1957 1917 1917 1719 1619 1460	Non Conflict PPT 1951 1949 1912 1906 1840 1816 1792 1850 1660 1551	1603 1607 1598 1635 1628 1500 1384 1300 1298 1298 1297	- 1608 1571 1579 1580 1636 1609 1579 1576 1519 1395 1281	- 1814 1713 1712 1716 1680 1446 1436 1435 1435 1435 1434 1433	Simple Mux 1722 1751 1821 1870 1591 1414 1375 1401 1372 1371 1371	1676 1670 1691 1605 1552 1544 1561 1443 1317 1274	2067 2068 2048 2071 1835 1746 1656 1655 1654 1717 1653	1764 1763 1752 1752 1621 1621 1637 1616 1416 1417 1369	1821 1840 1784 1738 1631 1536 1464 1443 1482 1408 1408	1712 1708 1723 1689 1568 1590 1608 1556 1485 1334 1284	3484 3450 3440 3481 3510 3474 3505 3524 3513 3476 3401
8 9 10 11 12 13 14 15 16 17	1971 1946 1848 1874 1896 1956 1867 1871 1631 1479 1401	1903 1886 1978 1912 1881 1874 1890 1801 1532 1420	2128 2099 2100 2154 2125 2206 2072 1899 1748 1618 1578	Conflict 2283 2202 2203 2209 2217 2214 2262 2289 2227 2164 2311 2241	Conflict PPT 2198 2157 2152 2141 2239 2224 2236 2204 2161 2203 2124	PPE Hon Conflict 2019 2130 1986 1973 1983 2017 1957 1917 1917 1917 1719 1619 1460 1438	Non Conflict PPT 1951 1949 1912 1906 1840 1816 1792 1850 1660 1551 1515	1603 1607 1598 1635 1628 1500 1384 1300 1298 1298 1297 1296	- 1608 1571 1579 1580 1636 1609 1579 1576 1519 1395 1281 1219	- 1814 1713 1712 1716 1680 1446 1435 1435 1435 1434 1433 1432	Simple Mux 1722 1751 1821 1870 1591 1414 1375 1401 1372 1371 1371 1368	1676 1670 1670 1691 1605 1552 1544 1561 1443 1317 1274 1262	2067 2068 2048 2071 1835 1746 1655 1654 1717 1653 1652	1764 1763 1752 1752 1621 1621 1637 1616 1416 1417 1369 1400	1821 1840 1784 1738 1631 1536 1464 1443 1482 1408 1408 1407	1712 1708 1723 1689 1568 1590 1608 1556 1485 1334 1284 1284	3484 3450 3440 3510 3474 3505 3524 3513 3476 3401 3276
8 9 10 11 12 13 14 15 16 17 18	1971 1946 1848 1874 1896 1956 1867 1871 1631 1479 1401 1368	1903 1886 1978 1912 1881 1874 1890 1801 1532 1420 1396	2128 2099 2100 2154 2125 2206 2072 1899 1748 1618 1578 1588	Conflict 2283 2282 2203 2209 2217 2214 2262 2289 2227 2164 2311 2241 2218	Conflict PPT 2198 2157 2152 2141 2141 2239 2224 2236 2204 2161 2203 2124 2098	PPE Hon Conflict 2019 2130 1986 1973 1983 2017 1957 1917 1917 1917 1619 1619 1460 1438 1427	Hon Conflict PPT 1951 1949 1912 1906 1840 1816 1792 1850 1660 1551 1515 1491	1603 1607 1598 1635 1628 1500 1384 1300 1298 1298 1297 1296 1296	- 1608 1571 1579 1580 1636 1609 1579 1576 1519 1395 1281 1219 1203	- 1814 1713 1712 1716 1680 1446 1435 1435 1435 1434 1433 1432 1432	Simple Mux 1722 1751 1821 1870 1591 1414 1375 1401 1372 1371 1371 1368 1367	1676 1670 1670 1691 1552 1554 1552 1544 1561 1443 1317 1274 1262 1256	2007 2068 2048 2071 1835 1746 1655 1654 1717 1653 1652 1652	1764 1763 1752 1752 1621 1621 1637 1616 1416 1417 1369 1400 1365	1821 1840 1784 1738 1631 1536 1464 1443 1482 1408 1408 1407 1403	1712 1708 1723 1689 1568 1590 1608 1556 1485 1334 1284 1284 1265 1269	3484 3450 3440 3481 3510 3474 3505 3524 3524 3513 3476 3401 3276 3221
8 9 10 11 12 13 14 15 16 17 18 19 20	1971 1946 1848 1874 1896 1956 1867 1871 1631 1479 1401 1368 1364	1903 1886 1978 1912 1881 1874 1890 1801 1532 1420 1396 1394 1449	2128 2099 2100 2100 2154 2206 2072 1899 1748 1618 1578 1578 1588	Conflict 2283 2282 2203 2209 2217 2214 2262 2289 2227 2164 2311 2241 2241 2218 2125 1842	Conflict PPT 2198 2157 2152 2141 2239 2224 2236 2204 2161 203 2123 203 2124 2161 203 2124 2161 203 2124 2155 2155 224 224 226 2204 2155 225 224 226 226 226 226 226 226 226 226 226	PPE Hon Conflict 2019 2130 1986 1973 2017 1983 2017 1957 1917 1719 1619 1460 1438 1427 1435 1425	Non Conflict PPT 1951 1949 1912 1906 1810 1810 1810 1850 1660 1551 1551 1551 1491 1491 1492 1494	1603 1607 1598 1635 1628 1500 1384 1300 1298 1297 1296 1296 1296 1296	- 1608 1571 1579 1580 1636 1609 1579 1576 1579 1395 1281 1219 1203 1201 1189	- 1814 1713 1712 1716 1680 1446 1436 1435 1434 1433 1432 1432 1432	Simple Mux 1722 1751 1821 1821 1870 1591 1414 1375 1401 1372 1371 1368 1367 1369	1676 1670 1691 1605 1552 1544 1561 1443 1317 1274 1262 1256	2067 2068 2048 2071 1835 1746 1655 1654 1717 1653 1652 1652 1652	1764 1763 1752 1621 1621 1621 1637 1616 1416 1417 1369 1400 1365 1370	1821 1840 1784 1738 1631 1536 1464 1443 1482 1408 1408 1408 1407 1403	1712 1708 1723 1689 1568 1590 1608 1556 1485 1334 1284 1284 1285 1269 1249	3484 3450 3440 3481 3510 3474 3505 3524 3513 3476 3401 3276 3241 3188

Figure 6.4: 64 Bit RRAs synthesis results

										mpariso							
Time (ns)	STA N2N	STA PPT	STA N2LOGN	CHN PPE Conflict	CHN PPE Conflict PPT	CHN PPE Non Conflict	CHN PPE Non Conflict PPT	вк_вт	BK_RS	НС_ВТ	HC_BT Simple Mux	HC_RS	KS_BT	KS_RS	LF_BT	LF_RS	BOWTIE
6	-10,48	-10,9	-8,12	-16,06	-15,95	-8,78	-8,92	-5,61	-10,73	-4,71	-4,71	-8,85	-4,98	-9,02	-4,67	-9,19	-13,2
7	-9,75	-9,69	-7,15	-15,53	-14,71	-7,59	-7,95	-4,55	-9,63	-3,74	-3,8	-7,92	-3,98	-8,02	-3,77	-8,3	-11,87
8	-8,46	-8,24	-6,29	-13,89	-14,15	-6,49	-6,9	-3,43	-8,36	-2,89	-2,71	-6,93	-2,94	-6,97	-2,75	-7,36	-10,9
9	-7,83	-7,69	-5,27	-12,73	-13,06	-5,75	-5,89	-2,58	-7,17	-1,89	-1,68	-5,96	-1,94	-5,79	-1,74	-6,1	-9,9
10	-6,23	-6,32	-4,04	-12,31	-12,03	-4,59	-5,12	-1,64	-6,4	-0,8	-0,77	-4,86	-0,84	-4,79	-0,59	-5,37	-8,9
11	-5,41	-5,91	-3,19	-10,45	-10,73	-3,44	-3,91	-0,39	-5,5	0	0	-4,01	0	-3,81	0	-4,32	-7,9
12	-4,46	-4,37	-2,16	-9,49	-10,25	-2,51	-3	0	-4,44	0,01	0,01	-2,93	0,01	-2,81	0	-3,29	-8,3
13	-3,68	-3,69	-1,28	-8,73	-9,34	-1,53	-1,91	0,01	-3,74	0,06	0,02	-1,96	0,02	-1,81	0,05	-2,19	-5,7
14	-2,35	-2,52		-7,37	-8,35	-0,67	-0,83	0	-2,14	0,08	0,17	-1	0,1	-0,81	0,02	-1,1	-5,3
15	-1,57	-1,73	0	-6,82	-6,89	0	-0,02	0,02	-1,31	0,05	0,07	0	0,14	0	0,01	-0,31	-3,9
16	-0,66	-0,89		-5,94	-5,69	0,05	0	0,01	-0,94	0,1	0,22	0	0,1	0,02	0,05	0	-3,2
17	0	0		-4,44	-5,12	0,01	0,03	0,05	0	0,15	0	0,01	0,1	0,03	0	0	-1,6
18	0	0	-	-3,39	-4,32	0	0,02	0,02	0	0,37	0,11	0,09	0,41	0,01	0,13	0,06	-1,1
		0.01	0.07	-2,47	-2,85	0,07	0,04	0,02	0	0,55	0,04	0,06	0,37	0	0,15	0,03	-0,1
19	0																
20	0,01	0,01	0,04 rchitecture	-1,74	-2,37 and blue							0,06	0,66	0,05	0,13	0,02	(
20	0,01	0,01	0,04	-1,74	-2,37 and blue Rou	ones are	rival archi n Arbite CHN		and their m	odification	ns.			0,05	0,13	0,02	
20	0,01	0,01	0,04	-1,74	-2,37 and blue	ones are	rivalarchi nArbite	itectures a	and their m	odification	ns.			0,05 KS_RS	0,13		BOWTI
20 Note: Gre	0,01 een RRA's	0,01 are our a STA	0,04 rchitecture STA	-1,74 es, orange CHN PPE	-2,37 and blue Rou CHII PPE Conflict	ones are nd Robi CHN PPE Non	n Arbite CHN PPE Non Conflict	itectures a	A Com	parison	for 12 HC_BT Simple	8Bit In	put				
20 lote: Gre Time (ns) 6 7	0,01 een RRA's STA N2N	0,01 are our a STA PPT	0,04 rchitecture STA N2LOGN	-1,74 es, orange CHII PPE Conflict	-2,37 and blue Rou CHN PPE Conflict PPT	ones are CHN PPE Non Conflict	n Arbite CHN PPE Non Conflict PPT	er ARE BK_BT	A Com BK_RS 3193 3192	parison HC_BT	HC_BT Simple Mux	8Bit In нс_rs	р <i>ut</i> ks_вт	KS_RS	LF_BT	LF_RS	BOWTI
20 lote: Gre <i>Time</i> (ns) 6 7 8	0,01 een RRA's STA H2H 3883 3960 3872	0,01 are our a STA PPT 3780	0,04 rchitecture STA H2LOGH 4089 4357	-1,74 ss, orange CHIN PPE Conflict 4225 4088 4398	-2,37 and blue Rou CHII PPE Conflict PPT 4673	ones are CHN PPE Non Conflict 4305	n Arbite CHII PPE Non Conflict PPT 4469	BK_BT	A Com BK_RS	parison HC_BT 3875	HC_BT Simple Mux 3837	8Bit In HC_RS 3626	but KS_BT 4220 4221 4273	KS_RS 3626	LF_BT 3741	LF_RS 3805	BOWTI 694 687
20 Note: Gree <i>Time</i> (ns) 6 7 8 9	0,01 sen RRA's STA H2H 3883 3960 3872 3717	0,01 are our a STA PPT 3780 3871 3974 3860	0,04 rchitecture STA H2LOGH 4089 4357	-1,74 es, orange CHN PPE Conflict 4225 4088 4398 4525	-2,37 e and blue Rou CHII PPE Conflict PPT 4673 5063 4774 4600	nd Robi CHII PPE Hon Conflict 4305 4690 4630 4379	n Arbit CHII PPE Hon Conflict PPT 4469 4427 4457	er ARE BK_BT 3128 3096 3236 3110	A Com BK_RS 3193 3192 3174 3318	parison HC_BT 3875 3914	HC_BT Simple Mux 3837 3896 3813 3834	8Bit In HC_RS 3626 3624 3722 3728	but KS_BT 4220 4221 4273 4287	KS_RS 3626 3625	LF_BT 3741 3657 3785 3724	LF_RS 3805 3539	BOWTII 694 687 686
20 Note: Gree Time (ns) 6 7 8 9 10	0,01 sen RRA's STA H2H 3883 3960 3872 3717 4033	0,01 are our a STA PPT 3780 3871 3974 3860 3980	0,04 rchitecture STA H2LOGH 4089 4357 4053 4130 4183	-1,74 es, orange CHN PPE Conflict 4225 4088 4398 4525 4348	-2,37 e and blue Rou CHII PPE Conflict PPT 4673 5063 4774 4600 4525	nd Robi CHII PPE Hon Conflict 4305 4690 4630 4379 4438	n Arbite CHII PPE Hon Conflict PPT 4469 4427 4455 4257	ег ARE вк_вт 3128 3096 3236 3110 3060	A Com BK_RS 3193 3192 3174 3318 3287	HC_BT	ns. for 12 HC_BT Simple Mux 3837 3896 3813 3834 3710	8Bit In HC_RS 3626 3624 3722 3728 3907	KS_BT 4220 4221 4273 4287 4307	KS_RS 3626 3625 3708 3883 3854	LF_BT 3741 3657 3785 3724 3751	LF_RS 3805 3539 3486 3690 3506	BOWTII 6941 687: 686: 684-
20 Note: Gree Time (ns) 6 7 8 9 10 11	0,01 een RRA's STA H2H 3883 3960 3872 3717 4033 3907	0,01 are our a STA PPT 3780 3871 3974 3860 3980 3810	0,04 rchitecture STA H2LOGH 4089 4357 4053 4130 4183 4240	-1,74 es, orange Chli PPE Conflict 4225 4088 4398 4525 4348 4525	-2,37 e and blue Rou CHII PPE Conflict PPT 4673 5063 4774 4600 4525 5184	nd Robi CHII PPE Hon Conflict 4305 4690 4630 4379 4438 4660	n Arbite CHII PPE Hon Conflict PPT 4469 4427 4455 4425 4455 4441	er ARE BK_BT 3128 3096 3236 3110 3060 3192	And their m A Com BK_RS 3193 3192 3174 3318 3287 3182	HC_BT 3875 3914 3632 3627 3818 3679	ns. for 12 HC_BT Simple Mux 3837 3896 3813 3834 3710 3182	8Bit In HC_RS 3626 3624 3722 3728 3907 3505	KS_BT 4220 4221 4273 4287 4307 3960	KS_RS 3626 3625 3708 3883 3854 3610	LF_BT 3741 3657 3785 3724 3751 3420	LF_RS 3805 3539 3486 3690 3506 3605	BOWTII 6944 687: 6884 6884 6884 6884
20 Note: Gree Time (ns) 6 7 8 9 10 11 12	0,01 een RRA's STA 11211 3883 3960 3872 3717 4033 3907 3928	0,01 are our a STA PPT 3780 3871 3974 3860 3980 3980 3980 3980 3980	0,04 rchitecture STA H2LOGH 4089 4357 4053 4130 4183 4240 4209	-1,74 es, orange Chili PPE Conflict 4225 4088 4398 4398 4525 4348 4546 4985	-2,37 e and blue Rou CHII PPE Conflict PPT 4673 5063 4774 4600 4525 5184 4802	nd Robi CHII PPE Hon Conflict 4305 4690 4630 4630 4379 4438 4660 4689	n Arbite CHII PPE Hon Conflict PPT 4469 4427 4455 4455 4257 4441 4393	BK_BT 3128 3096 3236 3110 3060 3192 2862	A Com BK_RS 3193 3194 3174 3318 3287 3182 3204	HC_BT 3875 3914 3632 3627 3818 3679 3113	HC_BT Simple Mux 3837 3896 3813 3834 3710 3182 2939	8Bit In HC_RS 3626 3624 3722 3728 3907 3505 3355	KS_BT 4220 4221 4273 4287 4307 3960 3734	KS_RS 3626 3625 3708 3883 3854 3610 3557	LF_BT 3741 3657 3785 3724 3751 3420 3343	LF_RS 3805 3539 3486 3690 3506 3605 3490	BOWTII 694 687 686 684 688 684 688
20 Note: Gre (ns) 6 7 8 9 10 11 12 13	0,01 een RRA's STA H2H 3883 3960 3872 3717 4033 3907 3928 3675	0,01 are our a STA PPT 3780 3871 3974 3860 3980 3980 3980 3810 3929 3894	0,04 rchitecture H2LOGH 4089 4357 4053 4130 4183 4183 4240 4209 4201	-1,74 es, orange CHII PPE Conflict 4225 4088 4398 4525 4348 4546 4985 4442	-2,37 e and blue CHII PPE Conflict PPT 4673 5063 4774 4600 4525 5184 4802 4530	nd Robi CHII PPE Hon Conflict 4305 4690 4630 4379 4438 4660 4689 4438	n Arbite CHII PPE Hon Conflict PPT 4469 4427 4455 4457 4455 4257 4441 4393 4458	BK_BT 3128 3096 3236 3110 3060 3192 2862 2656	A Com BK_RS 3193 3192 3174 3318 3287 3182 3204 3024	HC_BT 3875 3914 3627 3627 3818 3629 3627 3627 3818 3629 3627 3627 3627	HC_BT Simple Mux 3837 3896 3813 3834 3710 3182 2939 2891	8Bit In HC_RS 3626 3624 3722 3728 3907 3505 3355 3366	KS_BT 4220 4221 4273 4287 4307 3960 3734 3557	KS_RS 3626 3625 3708 3883 38854 3610 3557 3552	LF_BT 3741 3657 3785 37751 3420 3343 3224	LF_RS 3805 3539 3486 3690 3506 3605 3490 3500	BOWTII 694 687 686 684 688 688
20 lote: Gre (ns) 6 7 8 9 10 11 12 13 14	0,01 een RRA's STA II2II 3883 3960 3872 3717 4033 3907 3928 3675 3852	0,01 are our a STA PPT 3780 3871 3974 3860 3980 3810 3829 3894 3829	0,04 rchitecture H2LOGH 4089 4357 4053 4130 4183 4130 4183 4240 4209 4201 4002	-1,74 es, orange CHII PPE Conflict 4225 4088 4398 4525 4348 4525 4348 4546 4985 4442 5061	-2,37 and blue Rou CHII PPT 4673 5063 4774 4600 4525 5184 4802 4530 4492	nd Robi CHII PPE Hon Conflict 4305 4690 4630 4680 4379 4438 4660 4689 4327 4356	n Arbite CHII PPE Hon Conflict PPT 4469 4427 4455 4257 4455 4257 4441 4393 4458 4572	BK_BT 3128 3096 3236 3110 3060 3192 2862 2656 2616	A Com BK_RS 3193 3192 3174 3318 3287 3120 3204 3024 3118	parison HC_BT 3875 3914 3632 3627 3818 3679 3113 3009 3000	HC_BT Simple Mux 3837 3896 3813 3834 3710 3182 2939 2891 2876	8Bit In HC_RS 3626 3624 3722 3728 3907 3505 33555 33565 33565 33566 33419	Contemporation (Contemporation) (Contemp	KS_RS 3626 3625 3708 3883 3854 3610 3557 3552 3572	LF_BT 3741 3657 3785 3724 3751 3420 3343 3224 3166	LF_RS 3805 3539 3486 3690 3506 3605 3490 3500 3500 3500	BOWTII 694 687 686 684 688 684 687 690 689
20 lote: Gre <i>Time</i> (<i>ns</i>) 6 7 8 9 10 11 12 13 14 15	0,01 een RRA's STA H2H 3883 3960 3872 3717 4033 3907 3926 3675 3852 3675	0,01 are our a STA PPT 3780 3871 3974 3860 3980 3810 3980 3810 3929 3894 3820	0,04 rchitecture 12L0GH 4089 4053 4130 4183 4240 4209 4201 4002 3641	-1,74 es, orange Conflict 4225 4088 4525 4348 4526 4348 4546 4985 4442 5061 4384	-2,37 and blue Rou CHII PPE Conflict PPT 4673 5063 4774 4600 4525 5184 4800 4525 5184 4800 4492 4933	nd Robi CHII PPE IIOn Conflict 4305 4630 4630 4630 4639 4438 4660 4689 4327 4356 3786	n Arbit CHII PPE Hon Conflict PPT 4469 4427 4455 4455 4257 4445 4455 4257 4441 4393 4458 4572 4361	BK_BT 3128 3096 3236 3110 3060 3192 2656 2616 2612	A Com BK_RS 3193 3192 3174 3174 3318 3287 3182 3204 3024 3024 3118 3003	HC_BT 3875 3914 3632 3627 3818 3679 3113 3009 3000 2993	HC_BT Simple Mux 3837 3896 3813 3834 3710 3182 2939 2891 2895 2891	8Bit In HC_RS 3626 3624 3722 3728 3907 3505 3356 3356 3356 3356 3366 3419 3285	Contemporation (Contemporation) (Contemp	KS_RS 3626 3625 3708 3883 3883 3854 3610 3557 3552 3572 3364	LF_BT 3741 3657 3785 3724 3751 3420 33244 3166 3027	LF_RS 3805 3539 3486 3690 3506 3605 3490 3500 3500 33920 3428	BOWTII 694 687 686 684 688 684 687 690 689 690
20 lote: Gre (ns) 6 7 8 9 10 11 12 13 13 14 15 16	0,01 een RRA's STA 1211 3883 3960 3872 3717 4033 3907 3928 3675 3852 3612 3912	0,01 are our a STA PPT 3780 3870 3974 3860 3980 3980 3810 3929 3894 3929 3894 3922 3894 3922 3894 3922 3741	0,04 rchitecture STA H2LOGH 4089 4357 4053 4130 4183 4240 4209 4201 4209 4201 4002 3641 3587	-1,74 es, orange Chill PPE Conflict 4225 4088 4398 4398 4398 4398 4348 4348 4348 43	-2,37 and blue CHII PPE Conflict PPT 4673 5063 4774 4600 4525 5184 4802 4530 44923 5082	nd Robi CHII PPE Hon Conflict 4305 4690 4630 4379 4438 4660 4689 4327 4356 3786 3356 3526	n Arbite PPE Hon Conflict PPT 4469 4427 4457 4455 4257 4441 4393 4458 4572 4361 3326	BK_BT 3128 3096 3236 3110 3060 3192 2862 2656 2616 2612 2609	A Com BK_RS 3193 3192 3174 3318 3287 3182 3204 3024 3013 3003 2947	HC_BT 3875 3914 3627 3818 3679 3113 3009 3000 2993 2989	HC_BT Simple Mux 3837 3896 3813 3834 3710 3182 2939 2839 2839 2876 2876 2871 2865	8Bit In HC_RS 3626 3624 3722 3728 3907 3505 3355 3355 3355 3366 3419 3285 2876	KS_BT 4220 4221 427 4307 3960 3734 3556 3553 3558	KS_RS 3626 3625 3708 3883 3854 3610 3557 3572 3572 3572 3364 3005	LF_BT 3741 3657 3724 3751 3420 3343 3224 3166 3027 3001	LF_RS 3805 3639 3486 3690 3506 3690 3500 3500 3920 3920 3428 2915	BOWTI 694 687 686 684 688 684 687 689 689 689 689 689 690 682
20 lote: Gre <i>fime</i> (<i>ns</i>) 6 7 8 9 10 11 12 13 14 15 15 16 17	0,01 een RRA's STA H2H 3883 3960 3872 3717 4033 3907 3926 3675 3852 3675	0,01 are our a PPT 3780 3871 3974 3860 3980 3810 3929 3894 3820 3741 3820 3741 3494	0,04 rchitecture STA H2LOGH 4089 4357 4053 4130 4183 4240 4209 4201 4209 4201 4002 3641 3587	-1,74 es, orange Conflict 4225 4088 4525 4348 4526 4348 4546 4985 4442 5061 4384	-2,37 and blue Rou CHII PPE Conflict PPT 4673 5063 4774 4600 4525 5184 4800 4525 5184 4800 4492 4933	nd Robi CHII PPE Hon Conflict 4305 4690 4630 4630 4630 4689 4327 4356 3726 3726 3326 3321	n Arbit CHII PPE Hon Conflict PPT 4469 4427 4455 4455 4257 4445 4455 4257 4441 4393 4458 4572 4361	BK_BT 3128 3096 3236 3110 3060 3192 2656 2616 2612	A Com BK_RS 3193 3192 3174 3318 3287 3182 3204 3024 3024 3018 3003 2947 22916	HC_BT 3875 3914 3632 3627 3818 3679 3113 3009 3000 2993	HC_BT Simple Mux 3837 3896 3813 3834 3710 3182 2939 2891 2895 2891	8Bit In HC_RS 3626 3624 3722 3728 3907 3505 3356 3356 3356 3356 3366 3419 3285	Contemporation (Contemporation) (Contemp	KS_RS 3626 3625 3708 3883 3883 3854 3610 3557 3552 3572 3364	LF_BT 3741 3657 3785 3724 3751 3420 33244 3166 3027	LF_RS 3805 3539 3486 3690 3506 3605 3490 3500 3500 33920 3428	BOWTI 694 687 686 684 687 685 685 685 685 685 685 685 690 685 690 692
20 lote: Gree (ns) 6 7 8 9 10 11 12 13 14 15 16 17 18	0,01 een RRA's STA H2H 3883 3960 3872 3717 4033 3907 3928 3675 3852 3612 3912 3480 3215	0,01 are our a STA PPT 3780 3870 3974 3860 3980 3980 3810 3929 3894 3929 3894 3922 3894 3922 3894 3922 3741	0,04 rchitecture STA H2LOGH 4089 4357 4053 4130 4183 4240 4209 4201 4209 4201 4002 3641 3587	-1,74 es, orange CHII PPE Conflict 4225 4088 4398 4525 4348 4525 4348 4525 4348 4555 4442 5061 4384 4353 44559 4851	-2,37 and blue Rou PPE Conflict PPT 4673 5063 4774 4600 4525 5184 4800 4525 5184 4802 4530 4492 4530 4492 4530	nd Robi CHII PPE Hon Conflict 4305 4690 4630 4630 4630 4680 4680 4379 4438 4680 4327 4356 3786 3526 33261 3050	n Arbite CHI PPE Hon Conflict PPT 4469 4427 4455 44257 4445 4457 4455 4257 4445 4257 4458 4257 4458 4257 4458 4393 4458 4572 4361 33206 33205 3136	BK_BT 3128 3096 3236 3110 3060 3192 2862 2656 2616 2612 2609 2609 2609	A Com BK_RS 3193 3192 3174 3287 3204 3204 3204 3204 3204 3204 3204 3204	HC_BT 3875 3914 3632 3627 3818 3679 3113 3009 3000 2993 2989 2990 2989	HC_BT Simple Mux 3837 3896 3813 3834 3710 3182 2939 2839 2839 2876 2876 2871 2865	8Bit In HC_RS 3626 3624 3722 3728 3907 3505 3355 3366 3419 3285 2876 2697 2581	4220 4221 4273 4287 4307 3960 3734 3556 35556 35556 35556 35548 3548 3548	KS_RS 3626 3625 3708 3883 3854 3610 3557 3572 3572 3572 3364 3005	LF_BT 3741 3657 3785 3724 3751 3420 3343 3224 3166 3027 3001 2988 2959	LF_RS 3805 3639 3486 3690 3506 3690 3500 3500 3920 3920 3428 2915	BOWTI 634 687 686 684 684 687 690 692 692 692 692
20 lote: Gre (ns) 6 7 8 9 10 11 12 13 14 15 15 16 17	0,01 een RRA's STA H2H 3883 3960 3872 3717 4033 3907 3928 3675 3852 3612 3912 3912 3912	0,01 are our a PPT 3780 3871 3974 3860 3980 3810 3929 3894 3820 3741 3820 3741 3494	0,04 rchitecture H2LOGH 4089 4357 4053 4130 4183 4240 4201 4201 4201 4202 3641 3587 3285 3229	-1,74 es, orange Chll PPE Conflict 4225 4088 4398 4525 4348 4546 4985 4442 5061 4384 4353 4459	-2,37 and blue Rou PPE Conflict PPT 4673 5063 4774 4600 4525 5184 4802 4530 4492 4530 4492 4933 5082 4840	nd Robi CHII PPE Hon Conflict 4305 4690 4630 4630 4630 4689 4327 4356 3726 3726 3326 3321	n Arbite CHII PPE Hon Conflict PPT 4469 4427 4457 4457 4457 4457 4457 4453 4257 4441 3333 4458 4572 4361 3326 3205	BK_BT 3128 3096 3236 3110 3060 3192 2862 2656 2616 2612 2609 2609	A Com BK_RS 3193 3192 3174 3318 3287 3182 3204 3024 3024 3018 3003 2947 22916	HC_BT HC_BT 3875 3914 3632 3627 3818 3679 3113 3009 3000 2983 2989 2990	for 12. HC_BT Simple Mux 3837 3896 3813 3834 3710 3182 2891 2876 2871 2868	8Bit In HC_RS 3626 3624 3722 3728 3907 3505 3355 3355 3355 3355 3355 3356 3419 3285 2876 2697	4220 4221 4273 4287 43960 3734 3566 3553 3548 3549	KS_RS 3626 3625 3708 3883 3854 3610 3557 3552 3572 3364 3005 2961	LF_BT 3741 3657 3785 3724 3751 3420 3343 3224 3166 3027 3001 2988	LF_RS 3805 3539 3486 3690 3500 3500 3500 3500 3500 3500 3520 3428 2915 2753	BOWTI 694 687 686 684 688 684 689 689 689 689

Figure 6.5: 128 Bit RRAs synthesis results

					Dou	nd Dobir	n Arbita		NG Cor	nnarico	n for 2	SEDit I	anut				
Time (ns)	STA N2N	STA PPT	STA N2LOG N	CHN PPE Conflict	CHN PPE Conflict PPT	CHN PPE Non Conflict	CHN PPE Non Conflict PPT	вк_вт	BK_RS	нс_вт	HC_BT Simple Mux	HC_RS		KS_RS	LF_BT	LF_RS	BOWTIE
10	-8,66	-8,12	-5,97	-14,76	-14,57	-6,98	-6,45	-2,61	-8,53	-0,88	-1,35	-6,76	-1,62	-6,07	-1,78	-5,88	-12,26
11	-7,38	-7,14	-5,15	-13,92	-13,68	-5,88	-5,53	-1,32	-7,25	0	-0,42	-5,76	-0,7	-5,14	-0,69	-4,99	-11,15
12	-6,64	-6,21	-3,97	-12,84	-12,68	-4,61	-4,49	-0,27	-6,24	0,13	0	-4,8	0	-4,14	0	-3,98	-9,85
13	-5,61	-5,09	-2,74	-11,94	-11,62	-3,48	-3,55	0	-5,72	0,05	0	-3,81	0	-3,14	0	-3,39	-9,11
14	-4,45	-4,13	-1,96	-10,66	-10,55	-2,7	-2,64	0	-4,41	0,01	0,01	-2,77	0,04	-2,11	0	-2,49	-8,25
15	-3,51	-3,5	-1,7	-10,03	-9,55	-1,65	-1,64	0,01	-3,77	0,01	0,08	-1,7	0,01	-1,11	0,02	-1,33	-8,26
16 17	-2,76	-2,39	0	-8,86	-8,57	-0,4	-0,59	0	-2,79	0,15	0,14	-0,8	0,2	-0,1	0	-0,62	-7,3
17	-1,4	-1,33	0	-8,48	-7,49	0	0	0,02	-1,37	0,89	0,03	0	0,07	0.01	0	0	-4,71
19	-0,74	-0,44	0	-0,94	-6,6 -5,42		0,01	0,01	-0,48	1,89 2,89	0,02	0,01	0,22	0,01	0,04	0	-3,68 -2,6
20	0	0	0	-5,01	-0,42	0	0,01	0,01	0	2,89	0,08	0,01	0,02	0,04	0,01	0,01	-2,6
20	0	0	0	-3,98	-4,7	0	0,08	0.02	0	1,61	0,35	0,01	0,07	0,01	0,01	0,01	-1,38
22	0	0,01	0,01	-3,98	-3,37	0,03	0,01	0,02	0	2,61	0,57	0,03	0,05	0,02	0,01	0,01	-0,00
23	0,03	0,01	0,01	-1,75	-1,79	0,03	0,00	0,02	0	3,61	0,55	0,03	0,00	0,02	0,31	0,13	0,01
24	0,00	0,04	0,02	-0,67	-0,81	0,03	0,02	0,16	0,1	4,61	0,65	0,00	0,56	0,07	0,04	0,05	0,01
25	0.01	0,04	0,00	0,07	0,01	0,00	0.02	1,16	0,01	5,61	1,91	0,21	1,56	0.07	0,04	0,03	0,1
26	0,02	0,04	0,18	0	0	0,01	0.02	2,16	0,04	6,61	2,71	0,08	2,56	0,02	0,29	0,02	0,08
27	0.08	0.25	0.01	0.02	0	0.04	0.03	3,16	0,01	7,61	0,39	0,11	3,56	0.05	0.2	0,06	0.08
28	0,01	0,02	0,2	0,01	0	0,03	0,04	0,08	0	8,61	0,02	0,13	0	0,12	0,32	0,01	0,07
29	0	0,02	0,14	0	0,02	0,01	0,07	0,37	0,05	9,61	0,03	0,12	0,04	0,21	0,16	0	0,02
30	0,08	0,23	0	0,08	0	0,08	0	0,49	0,04	10,61	0,04	0,12	0,34	0,13	0,07	0,03	0,05
Note: Gr	een RRA'	s are our	architect	ures, orar	nge and l	blue ones	s are riva	larchitec	tures and	their mo	dification	s.					
					Rot	und Rob	in Arbit	er ARE	A Com	parison	for 25	6Bit In	put				
Time (ns)	STA N2N	STA PPT	STA N2LOG N	CHN PPE Conflict	Rou CHN PPE Conflict PPT	CHN PPE Non Conflict	in Arbit CHN PPE Non Conflict PPT		A Com BK_RS		HC_BT Simple Mux	5Bit In нс_Rs		KS_RS	LF_BT	LF_RS	BOWTIE
			N2LOG	PPE	CHN PPE Conflict	CHN PPE Non	CHN PPE Non Conflict				HC_BT Simple			KS_RS 7234	LF_BT 7669	LF_RS 6675	BOWTIE 13693
(ns) 10 11	N2N 8129 8396	PPT 8145 8285	N2LOG N 8289 8172	PPE Conflict 8877 8938	CHN PPE Conflict PPT 8999 9067	CHN PPE Non Conflict 8412 8382	CHN PPE Non Conflict PPT 8140 8348	BK_BT 6203 6369	BK_RS 6235 6506	HC_BT 6002 5689	HC_BT Simple Mux 7293 7155	HC_RS 6829 6644	KS_BT 9322 9092	7234	7669	6675 6505	13693 13676
(ns) 10 11 12	N2N 8129 8396 8238	PPT 8145 8285 8289	N2LOG N 8289 8172 8421	PPE Conflict 8877 8938 8823	CHN PPE Conflict PPT 8999 9067 8915	CHN PPE Non Conflict 8412 8382 8510	CHN PPE Non Conflict PPT 8140 8348 8136	BK_BT 6203 6369 6406	BK_RS 6235 6506 6595	HC_BT 6002 5689 5196	HC_BT Simple Mux 7293 7155 6477	HC_RS 6829 6644 6167	KS_BT 9322 9092 8402	7234 6689 6680	7669 7575 6923	6675 6505 6381	13693 13676 13673
(ns) 10 11 12 13	N2N 8129 8396 8238 8282	PPT 8145 8285 8289 8199	N2LOG N 8289 8172 8421 8657	PPE Conflict 8877 8938 8823 8791	CHN PPE Conflict PPT 8999 9067 8915 9093	CHN PPE Non Conflict 8412 8382 8510 8791	CHN PPE Non Conflict PPT 8140 8348 8136 8013	BK_BT 6203 6369 6406 5703	BK_RS 6235 6506 6595 6237	HC_BT 6002 5689 5196 5046	HC_BT Simple Mux 7293 7155 6477 6086	HC_RS 6829 6644 6167 6158	KS_BT 9322 9092 8402 7913	7234 6689 6680 6986	7669 7575 6923 6853	6675 6505 6381 6889	13693 13676 13673 13729
(ns) 10 11 12 13 14	N2N 8129 8396 8238 8282 8077	PPT 8145 8285 8289 8199 8130	N2LOG N 8289 8172 8421 8657 8364	PPE Conflict 8877 8938 8823 8823 8791 8767	CHN PPE Conflict PPT 8999 9067 8915 9093 9072	CHN PPE Non Conflict 8412 8382 8510 8791 8375	CHN PPE Non Conflict PPT 8140 8348 8136 8013 7937	BK_BT 6203 6369 6406 5703 5350	BK_RS 6235 6506 6595 6237 6107	HC_BT 6002 5689 5196 5046 5041	HC_BT Simple Mux 7293 7155 6477 6086 6003	HC_RS 6829 6644 6167 6158 6299	KS_BT 9322 9092 8402 7913 7642	7234 6689 6680 6986 6784	7669 7575 6923 6853 6801	6675 6505 6381 6889 6504	13693 13676 13673 13729 13724
(ns) 10 11 12 13 14 15	N2N 8129 8396 8238 8282 8077 7919	PPT 8145 8285 8289 8199 8130 8074	N2LOG N 8289 8172 8421 8657 8364 7901	PPE Conflict 8877 8938 8823 8791 8767 8738	CHN PPE Conflict PPT 8999 9067 8915 9093 9072 9066	CHN PPE Non Conflict 8382 8510 8791 8375 8523	CHN PPE Non Conflict PPT 8140 8348 8136 8013 7937 7936	BK_BT 6203 6369 6406 5703 5350 5259	BK_RS 6235 6506 6595 6237 6107 5959	HC_BT 6002 5689 5196 5046 5041 5041	HC_BT Simple Mux 7293 7155 6477 6086 6003 5992	HC_RS 6829 6644 6167 6158 6299 6738	KS_BT 9322 9092 8402 7913 7642 7635	7234 6689 6680 6986 6784 6785	7669 7575 6923 6853 6801 6668	6675 6505 6381 6889 6504 6386	13693 13676 13673 13729 13724 13686
(ns) 10 11 12 13 14 15 16	N2N 8129 8396 8238 8282 8077 7919 7897	PPT 8145 8285 8289 8199 8130 8074 8127	N2LOG N 8289 8172 8421 8421 8457 8364 7901 8180	PPE Conflict 8877 8938 8823 8791 8767 8738 8795	CHN PPE Conflict PPT 8999 9067 8915 9093 9072 9066 9338	CHN PPE Non Conflict 8382 8510 8791 8375 8523 8403	CHN PPE Non Conflict PPT 8140 8348 8136 8013 7937 7936 7872	BK_BT 6203 6369 6406 5703 5350 5259 5244	BK_RS 6235 6506 6595 6237 6107 5959 6144	HC_BT 6002 5689 5196 5046 5041 5041 5041	HC_BT Simple Mux 7293 7155 6477 6086 6003 5992 6039	HC_RS 6829 6644 6167 6158 6299 6738 6156	KS_BT 9322 9092 8402 7913 7642 7635 7632	7234 6689 6680 6986 6784 6785 6733	7669 7575 6923 6853 6801 6668 6402	6675 6505 6381 6889 6504 6386 6386 6486	13693 13676 13673 13729 13724 13686 13665
(ns) 10 11 12 13 14 15 16 17	N2N 8129 8396 8238 8282 8077 7919 7897 8192	PPT 8145 8285 8289 8199 8130 8074 8127 8111	N2LOG N 8289 8172 8421 8657 8364 7901 8180 7398	PPE Conflict 8877 8938 8823 8791 8767 8738 8595 8595	CHN PPE Conflict PPT 8999 9067 8915 9093 9072 9066 9338 9317	CHN PPE Non Conflict 8382 8510 8791 8375 8523 8403 7353	CHN PPE Non Conflict PPT 8140 8348 8136 8013 7937 7936 7872 7096	BK_BT 6203 6369 6406 5703 5350 5259 5244 5239	BK_RS 6235 6506 6695 6237 6107 5959 6144 6108	HC_BT 6002 5689 5196 5046 5041 5041 5041 5041	HC_BT Simple Mux 7293 7155 6477 6086 6003 5992 6039 5989	HC_RS 6829 6644 6167 6158 6299 6738 6156 6031	KS_BT 9322 9092 8402 7913 7642 7635 7632 7629	7234 6689 6680 6986 6784 6785 6733 6215	7669 7575 6923 6853 6801 6668 6402 6467	6675 6505 6381 6889 6504 6386 6486 6486 5967	13693 13676 13673 13729 13724 13686 13665 13769
(ns) 10 11 12 13 14 15 16 17 18	N2N 8129 8396 8238 8282 8077 7919 7897 8192 7783	PPT 8145 8285 8289 8199 8130 8074 8127 8111 8069	N2LOG N 8289 8172 8421 8421 8657 8364 7901 8180 7398 7236	PPE Conflict 8877 8938 8823 8791 8767 8738 8595 8544 9120	CHN PPE Conflict PPT 8999 9067 8915 9093 9072 9066 9338 9317 9123	CHN PPE Non Conflict 8382 8510 8791 8375 8523 8403 7353 7315	CHN PPE Non Conflict PPT 8140 8348 8136 8013 7937 7936 7872 7096 6794	BK_BT 6203 6369 6406 5703 5350 5259 5244 5239 5236	BK_RS 6235 6506 6595 6237 6107 5959 6144 6108 5899	HC_BT 6002 5689 5196 5046 5041 5041 5041 5041	HC_BT Simple Mux 7293 7155 6477 6086 6003 5992 6039 5989 6203	HC_RS 6829 6644 6167 6158 6299 6738 6156 6031 5538	KS_BT 9322 9092 8402 7913 7642 7635 7632 7629 7628	7234 6689 6880 6986 6784 6785 6733 6215 6178	7669 7575 6923 6853 6801 6668 6402 6467 6397	6675 6505 6381 6889 6504 6386 6486 6486 5967 6592	13693 13676 13673 13729 13724 13686 13665 13769 13862
(ns) 10 11 12 13 14 15 16 17 18 19	N2N 8129 8396 8238 8282 8077 7919 7897 8192 7783 7648	PPT 8145 8285 8289 8199 8130 8074 8127 8111 8069 7003	N2LOG N 82289 8172 8421 8657 8364 7901 8180 7398 7236 6668	PPE Conflict 8877 8938 8823 8791 8767 8738 8595 8544 9120 8899	CHN PPE Conflict PPT 8999 9067 8915 9093 9072 9066 9338 9317 9123 9077	CHN PPE Non Conflict 8342 8510 8791 8375 8523 8403 7353 7315 6574	CHN PPE Non Conflict PPT 8140 8348 8136 8013 7937 7936 7872 7096 6794 6567	BK_BT 6203 6369 6406 5703 5350 5259 5244 5239 5236 5233	BK_RS 6235 6506 6695 6237 6107 5959 6144 6108 5899 5597	HC_BT 6002 5689 5196 5046 5041 5041 5041 5041 5041	HC_BT Simple Mux 7293 7155 6477 6086 6003 5992 6039 5989 6203 5995	HC_RS 6829 6644 6167 6158 6299 6738 6156 6031 5538 5356	KS_BT 9322 9092 8402 7913 7642 7635 7632 7629 7628 7628	7234 6689 6680 6986 6784 6785 6733 6215 6178 6976	7669 7575 6923 6853 6801 6668 6402 6467 6397 6199	6675 6505 6381 6889 6504 6386 6486 6486 5967 5592 5495	13693 13676 13673 13729 13724 13686 13665 13769 13862 13842
(ns) 10 11 12 13 14 15 16 17 18 19 20	N2N 8129 8396 8238 8282 8077 7919 7897 8192 7783 7648 6877	PPT 8146 8285 8289 8199 8130 8074 8127 8111 8069 7003 6528	N2LOG N 82289 8172 8421 8657 8364 7901 8180 7398 7236 6668 6668	PPE Conflict 8877 8938 8823 8791 8767 8738 8595 8544 9120 8899 8748	CHN PPE Conflict PPT 8999 9067 8915 9093 9093 9093 9093 9093 9093 9093 909	CHN PPE Non Conflict 8412 8382 8510 8791 8375 8523 8403 7355 8543 7315 6574 6448	CHN PPE Non Conflict PPT 8140 8348 8136 8013 7937 7936 7937 7936 6794 6567 6485	BK_BT 6203 6369 6406 5703 5350 5254 5239 5236 5233 5487	BK_RS 6236 6606 6695 6237 6107 5959 6144 6108 5899 5697 5054	HC_BT 6002 5689 5196 5046 5041 5041 5041 5041 5041 5041 5041	HC_BT Simple Mux 7293 7155 6477 6086 6003 5992 6039 5989 6203 5995 5988	HC_RS 6829 6644 6167 6158 6299 6738 6156 6031 5538 5356 5302	KS_BT 9322 9092 8402 7913 7642 7636 7632 7629 7628 7628 7626 7880	7234 6689 6680 6986 6784 6785 6733 6215 6178 5976 6050	7669 7575 6923 6853 6801 6668 6402 6467 6397 6199 6130	6675 6505 6381 6889 6504 6386 6486 6486 6486 5967 5592 5495 5437	13693 13676 13673 13729 13724 13686 13666 13769 13862 13842 13842 13820
(ns) 10 11 12 13 14 15 16 17 18 19 20 21	N2N 8129 8396 8238 8282 8077 7919 7897 8192 7783 7648 6877 6522	PPT 8145 8285 8289 8199 8130 8074 8127 8111 8069 7003 6528 6358	N2LOG N 8289 8172 8421 8657 8364 7901 8180 7398 7236 6668 6460 6510	PPE Conflict 8877 8938 8823 8791 8767 8738 8595 8544 9120 8899 8748 8840	CHN PPE Conflict PPT 8999 9067 8916 9093 9072 9066 9338 9317 9123 9077 9077 9146	CHN PPE Non Conflict 83412 83412 83403 85413 8375 8523 8403 7353 7315 6574 6474 6448 6205	CHN PPE Non Conflict PPT 8140 8348 8136 8013 7937 7936 7937 7936 7872 7096 6794 6667 6485 6533	BK_BT 6203 6369 6406 5703 5350 5259 5236 5233 5487 5232	BK_RS 6235 6506 6237 6107 5959 6144 6108 5899 6597 5054 4914	HC_BT 6002 5689 5196 5046 5041 5041 5041 5041 5041 5041 5041 5040 5040	HC_BT Simple Mux 7293 7156 6477 6086 6003 5992 6039 5992 6039 5998 6203 5995 5988 5990	HC_RS 6829 6644 6167 6158 6299 6738 6165 6031 5538 5356 5302 5328	KS_BT 9322 9092 8402 7913 7642 7635 7632 7629 7629 7628 7626 7880 7626	7234 6689 6880 6986 6784 6785 6733 6215 6178 5976 6050 5970	7669 7575 6923 6853 6801 6668 6402 6467 6397 6199 6130 6132	6675 6505 6381 6889 6504 6386 6486 6486 5967 5592 5495 5437 5419	13693 13676 13673 13729 13724 13686 13686 13686 13769 13862 13842 13842 13842 13842
(ns) 10 11 12 13 14 15 16 17 18 19 20 21 22	N2N 8129 8396 8238 8282 8077 7919 7897 8192 7783 7648 6877 6522 6216	PPT 8145 8285 8289 8199 8130 8074 8127 8111 8069 7003 6528 6358 6290	N2LOG N 8289 8172 8421 8667 8364 7901 8180 7398 7236 6668 6668 6668 6450 6510 6357	PPE Conflict 8877 8938 8823 8791 8767 8738 8595 8544 9120 8899 8748 8840 88840	CHN PPE Conflict PPT 8999 9067 8916 9093 9072 9066 9338 9317 9123 9077 9077 9146 9771	CHN PPE Non Conflict 8412 83610 8791 8375 8523 8403 7353 7315 6574 6548 6548 6205 6179	CHN PPE Non Conflict PPT 8140 8348 8136 8013 7937 7936 7937 7936 7872 7096 6794 6567 6485 66533 6520	BK_BT 6203 6369 6406 6703 5350 5259 5244 5239 5236 5233 5487 5232 5231	BK_RS 6235 6506 6596 6237 6107 5959 6144 6108 5899 5697 5054 4914 4822	HC_BT 6002 5689 5196 5046 5041 5041 5041 5041 5041 5041 5040 5040	HC_BT Simple Mux 7293 7155 6477 6086 6003 5992 6039 5989 6039 6039 6039 6039 6039 6039 6039 603	HC_RS 6829 6844 6167 6158 6299 6738 6156 6031 6638 6356 6302 6328 5367	KS_BT 9322 9092 8402 7913 7642 7635 7632 7629 7628 7626 7626 7820 7826 7820	7234 6689 6680 6986 6784 6785 6733 6215 6178 6976 6050 5970 5966	7669 7575 6923 6853 6801 6668 6402 6467 6397 6199 6130 6132 6141	6675 6505 6381 6889 6504 6386 6486 6486 5967 5592 5495 5437 5419 5382	13693 13676 13673 13729 13729 13729 13729 13686 13686 13686 13686 13686 13882 13842 13820 13820 13820 13578
(ns) 10 11 12 13 14 15 16 17 18 19 20 21	N2N 8129 8396 8238 8282 8077 7919 7897 8192 7783 7648 6877 6622 6216 6196	PPT 8145 8285 8289 8199 8130 8074 8127 8111 8069 7003 6528 6358 6290 6221	N2LOG N 8289 8172 8657 8364 7901 8180 7398 7236 6668 6450 6450 6510 6357 6281	PPE Conflict 8877 8938 8791 8767 8738 8595 8544 9120 8544 9120 8549 8544 9120 8544 9120 8545 8540 8540 8546 8577 8546 8540 8546 8546 8546 8546 8546 8546 8547 8546 8546 8546 8546 8546 8546 8546 8546	CHN PPE Conflict PPT 8999 9067 8915 9093 9072 9066 9338 9317 9123 9077 9146 9771 9146 9771 9039	CHN PPE Non Conflict 83412 8382 8510 8791 8375 8523 8403 7353 7315 6574 6488 6205 6179 6149	CHN PPE Non Conflict PPT 8140 8348 8136 8013 7937 7936 7872 7096 6794 6567 6485 66533 66520 6476	BK_BT 6203 6369 6406 5703 5350 5244 5239 5244 5239 5233 5487 5231 5231	BK_RS 6236 6506 6595 6237 6107 5959 6144 6108 5899 6597 5054 4914 4882 4831	HC_BT 6002 6689 5196 5046 5041 5041 5041 5041 5041 5041 5040 5040 5040 5040 5040	HC_8T Simple Mux 7293 7155 6477 6086 6003 5992 6039 5989 6203 5989 5989 5989 5989	HC_RS 6829 6644 6167 6158 6299 6738 6156 6031 5538 5356 5326 5328 5327 5283	KS_BT 9322 9092 8402 7913 7642 7635 7632 7629 7628 7626 7880 7826 7820 7825 7624	7234 6689 6680 6986 6784 6785 6733 6215 6178 6976 6050 5970 5966 5964	7669 7575 6923 6853 6801 6668 6402 6467 6397 6199 6130 6132 6141 6126	6675 6505 6381 6889 6504 6386 6486 6486 5967 5592 5495 5495 5437 5419 5382 5356	13693 13676 13673 13729 13724 13686 13685 13685 13842 13842 13842 13842 13842 13878 12908 12871
(115) 10 11 12 13 14 15 16 17 18 19 20 21 22 23	N2N 8129 8396 8238 8282 8077 7919 7897 8192 7783 7648 6877 6522 6216 6196 6196	PPT 8145 8285 8289 8199 8130 8074 8127 8111 8069 7003 6528 6358 6290 6221 6188	N2L06 N 8289 8172 8421 8467 8364 7901 8364 7901 8364 7398 7236 6668 6450 6510 6510 6357 6281 6264	PPE Conflict 8877 8938 8823 8791 8767 8738 8595 8544 9120 8899 8748 8899 8748 8840 8885 8704	CHN PPE Conflict PPT 3999 9067 8915 9093 9072 9086 9338 9317 9123 9077 9173 9077 9146 9771 9077	CHN PPE Non Conflict 8382 8510 8791 8375 8523 8403 7365 8574 6574 6448 6205 6179 6149 6136	CHN PPE Non Conflict PPT 8140 8348 8136 8013 7937 7936 7937 7936 6794 6567 6485 6533 6520 6476	BK_BT 6203 6369 6406 5703 5350 5259 5244 5239 5236 5233 5487 5232 5231 5231	BK_RS 6236 6506 6695 6237 6107 5959 6144 6108 5899 5697 5054 4914 4882 4831 4774	HC_BT 6002 5689 5196 5046 5041 5041 5041 5041 5041 5040 5040 5040	HC_BT Simple Mux 7293 7155 6477 6086 6003 5992 6039 6039 6039 6989 6203 5985 5988 5990 5989 5984	HC_RS 6829 6644 6167 6158 6299 6738 6156 6031 6538 6356 6302 6328 5365 5328 5327	KS_BT 9322 9092 8402 7913 7642 7632 7632 7629 7628 7626 7880 7626 7623 7623 7623	7234 6689 6680 6784 6785 6733 6215 6178 6976 6050 5970 5970 5966 5964 6029	7669 7675 6923 6801 6863 6801 6668 6402 6467 6397 6199 6130 6132 6141 6126 6141	6675 6505 6381 6504 6386 6486 6486 6486 6486 5967 5592 5495 5437 5419 5382 5356 5345	13693 13676 13673 13724 13724 13686 13686 13769 13882 13842 13820 13878 12908 12871 12820
(ns) 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	N2N 8129 8396 8238 8282 8077 7919 7897 8192 7783 7648 6877 6522 6216 6196 6196 6196 6196 6195	PPT 8145 8285 8289 8199 8130 8074 8127 8111 8069 7003 6528 6358 6290 6221 6188 6194	N2L06 N 8289 8121 8421 8657 8364 7901 8180 7901 8180 7901 8180 6668 6668 6668 6460 6510 6551 6281 6284 6288	PPE Conflict 8877 8938 8823 8791 8767 8738 8566 8544 9120 8899 8748 8840 88840 88840 88840 88840 88845 8704 8722 7977	CHN PPE Conflict PPT 8999 9067 8915 9093 9072 9086 9338 9317 9123 9077 9146 9771 9077 9146 9771 9039 8569 8563	CHN PPE Non Conflict 8382 8510 8791 8376 8573 8403 7363 7315 6574 6448 6205 6179 6136 6179 6136	CHN PPE Non Conflict PPT 8140 8348 8013 7937 7936 7936 7872 7096 6794 6567 6485 6533 6520 6476 6471 6445	BK_BT 6203 6369 6406 5703 5259 5244 5239 5236 5233 5487 5232 5231 5231 5231	BK_RS 6235 6506 6595 6237 6107 5959 6144 6108 5899 5597 5054 4914 4822 4831 4774 4795	HC_BT 6002 5689 5196 5041 5041 5041 5041 5041 5041 5040 5040	HC_BT Simple Mux 7293 7155 6477 6086 6003 5992 6039 6203 5995 5988 5990 5987 5989 5984 5984	HC_RS 6829 6644 6167 6168 6299 6738 6156 6031 65386 5356 5356 5356 5356 5357 5283 5271 5320	KS_BT 9322 9092 8402 7913 7642 7632 7629 7628 7629 7628 7626 7880 7623 7623 7623 7623 7623	7234 6689 6680 6986 6784 6785 6733 6215 6178 6976 6976 6976 5966 5964 6029 5962	7669 7675 6923 6853 6801 6668 6402 6467 6397 6199 6130 6141 6141 6142 6141 6141 6141	6675 6505 6381 6389 6586 6486 6486 6486 6486 6486 5967 5592 5495 5495 5495 5419 5386 5386 5345	13693 13676 13673 13729 13724 13686 13789 13862 13842 13842 13820 13678 12908 12871 12820 12818
(ns) 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25	N2N 8129 8396 8238 8282 8077 7919 7897 8192 7783 7648 6877 6522 6216 6196 6069 6162 6899	PPT 8146 8285 8289 8199 8130 8074 8127 8111 8069 7003 6528 6358 6290 6221 6188 6194 6179	N2L06 N 8289 8172 8421 8667 8364 7901 8180 7398 7236 6668 6450 6450 6510 6357 6281 6264 6268 6259	PPE Conflict 8877 8938 8823 8791 8767 8738 8595 8544 9120 8896 8748 8840 8885 8748 8885 8704 8885 8704 8727 7771	CHN PPE Conflict PPT 8999 9067 8916 9093 9072 9086 9338 9317 9123 9077 9146 9771 9077 9146 9771 9077 9146	CHN PPE Non Conflict 8382 8510 8791 8376 8523 8403 7363 7315 6574 6448 6205 6179 6149 6136 6095 6105	CHN PPE Non Conflict PPT 8140 8348 8013 7937 7936 6794 6794 6794 6567 6485 6533 6520 6476 6471 6495 6477	BK_BT 6203 6369 6406 5703 5350 5259 5244 5239 5236 5487 5232 5231 5231 5231 5231 5231	BK_RS 6235 6506 6595 6144 6108 5899 6144 6108 5899 5054 4914 4882 4831 4774 4795 4789	HC_BT 6002 5689 5196 5046 5041 5041 5041 5041 5040 5040 5040 5040	HC_BT Simple Mux 7293 7155 6477 6086 6003 6992 6039 6989 6203 5986 5988 5980 5984 5984 5984	HC_RS 6829 6644 6167 6158 6299 6738 6156 6031 65362 6302 6328 6367 6283 6271	KS_BT 9322 9092 8402 7913 7642 7632 7629 7628 7629 7628 7620 7626 7820 7623 7623 7623 7623 7623 7623	7234 6689 6986 6986 6784 6733 6215 6178 6976 6050 5970 5966 6050 5970 5966 5974 6029 5962	7669 7575 6853 6853 6868 6402 6467 6397 6199 6130 6132 6141 6126 6110 6080 6063	6675 6505 6381 6889 6504 6386 6486 6486 6486 5495 5495 5495 5495 5495 5495 5495 5437 5519 5345 5345 5342 5336	13693 13676 13673 13729 13724 13665 13665 13769 13862 13842 13820 13878 12908 12871 12820 12811 12820 12618 12376
(ns) 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26	N2N 8129 8396 8238 8282 8077 7919 7897 8192 7783 7648 6877 6522 6216 6196 6196 6196 6196 6195	PPT 8145 8285 8289 8199 8177 8171 8069 7003 6528 6358 6358 6290 6221 6188 6194 6179 6298	N2L06 N 8289 8421 8421 8657 8364 7901 8180 7398 7236 6668 6460 6510 6510 6510 65281 6264 6268 6259 6298	PPE Conflict 8938 8823 8791 8767 8768 8595 8564 9120 8899 8748 8899 8748 8890 8886 8840 8886 8704 8886 8704 8886 9777 77701	CHN PPE Conflict PPT 8999 9067 8915 9093 9072 9086 9338 9317 9123 9077 9146 9771 9077 9146 9771 9039 8569 8563	CHN PPE Non Conflict 8382 8510 8791 8375 8523 8403 7315 6574 6448 6205 6179 6149 6136 6095 6105 6137	CHN PPE Non Conflict PPT 8140 8348 8013 7937 7936 7936 7872 7096 6794 6567 6485 6533 6520 6476 6471 6445	BK_BT 6203 6369 6406 6703 6360 6259 6234 6239 6236 6233 6487 6233 6487 6231 5231 5231 5231 5231	BK_RS 6235 6506 6595 6237 6107 5959 6144 6108 5899 5597 5054 4914 4822 4831 4774 4795	HC_BT 6002 5689 5196 5041 5041 5041 5041 5041 5041 5040 5040	HC_BT Simple Mux 7293 7155 6477 6086 6003 5992 6039 6203 5995 5988 5990 5987 5989 5984 5984	HC_RS 6829 6644 6167 6168 6299 6738 6156 6031 5638 6366 5366 5366 5362 5328 5327 5283 6271 5220 5271 5220	KS_BT 9092 9092 8402 7913 7642 7636 7632 7629 7628 7626 7626 7623 7623 7623 7623 7623 7623	7234 6689 6986 6784 6785 6733 6215 6178 5976 6050 5970 5966 5964 6029 5962 5972	7669 7675 6923 6853 6801 6668 6402 6467 6397 6199 6130 6141 6141 6142 6141 6141 6141	6675 6505 6381 6889 6604 6386 6486 5967 5495 5495 5495 5495 5495 5495 5495 549	13693 13676 13673 13729 13724 13686 13686 13686 13882 13882 13882 13882 13882 13882 13882 13878 12908 12871 12820 12818 12376 12288
(ns) 10 11 12 13 14 15 16 17 17 18 19 20 21 22 23 24 25 26 27	N2N 8129 8396 8238 827 7919 7897 8192 7783 7648 6877 6522 6216 6196 6196 6069 6152 6898 8173 6948	PPT 8145 8285 8289 8199 8173 8074 8177 8111 8069 7003 6528 6358 6290 6221 6188 6194 6179 6298 6201	N2LOG N 8289 8172 8421 8667 8364 7901 8180 7398 7236 6668 6450 66651 66651 66561 6357 6281 6264 6268 6259 6298	PPE Conflict 8877 8938 8823 8791 8767 8767 8768 8594 9120 8899 8748 8899 8748 8840 8886 8704 8722 7977 77074 77977 77004	CHN PPE Conflict PPT 8916 9003 9072 9066 9338 9377 9123 9077 9123 9077 9146 9771 9077 91746 9771 9077 9176 9077 9176 9077 9077 9176 9077 9077 9176 9077 9176 9077 9077 9176 9077 9176 9077 9177 9077 9176 9077 9176 9077 9177 9177 9177 9177 9177 9177 9177	CHN PPE Non Conflict 83412 8382 8510 8791 8375 8523 8403 7315 8574 6448 6205 6179 6149 6136 6195 6105 6105	CHN PPE Non Conflict PPT 8140 8348 8136 8013 7937 7936 7937 7936 6794 8667 6485 6653 6520 6476 6485 6471 6485 6477 6482 6483	BK_BT 6203 6369 6406 5703 53500 5259 6244 5239 5236 5233 5487 5231 5231 5231 5231 5231 5231 5231	BK_RS 6235 6506 6237 6107 5959 6144 6108 5899 5697 5054 4914 4821 4821 4774 4821 4775 4789 4775 4789	HC_BT 6002 56899 5196 5046 5041 5041 5041 5041 5041 5041 5040 5040	HC_BT Simple Mux 7293 7156 6477 6086 64039 5992 6039 5992 6039 5989 6203 5995 5988 5990 5988 5990 5988 5990 5984 5984 5984 5984	HC_RS 6829 6644 6167 6158 6299 6738 6156 6031 5538 5366 5302 5385 5365 5302 5385 5367 5283 5271 5286 6224	KS_BT 9322 9092 8402 7913 7642 7635 7629 7628 7626 7626 7626 7623 7623 7623 7623 7623	7234 6689 6680 6986 6784 6785 6715 6215 6178 6215 6178 6050 5976 6050 5976 5964 6029 5962 5972 5972 5972	7669 7675 6923 6853 6801 6668 6407 6397 6199 6130 6143 6141 6126 6110 6080 6083 6089 6063	6675 6381 6381 6386 6386 6486 6486 6486 5967 5592 5495 5495 5495 5495 5495 5495 5495	13693 13676 13677 13724 13724 13686 13686 13769 13882 13842 13820 13842 13820 13842 13820 13842 13820 12871 12820 12871 12820 12871
(ns) 10 11 12 13 14 15 16 17 18 19 20 21 19 22 23 24 25 26 27 28	N2N 8129 8396 8238 8282 8077 7919 78192 7783 7648 6877 6522 6216 6196 6196 61652 63899 6153	PPT 8145 8285 8289 8199 8177 8171 8069 7003 6528 6358 6358 6290 6221 6188 6194 6179 6298	N2L06 N 8289 8421 8421 8657 8364 7901 8180 7398 7236 6668 6460 6510 6510 6510 65281 6264 6268 6259 6298	PPE Conflict 8938 8823 8791 8767 8768 8595 8564 9120 8899 8748 8899 8748 8890 8886 8840 8886 8704 8886 8704 8886 9777 77701	CHN PPE Conflict PPT 8909 9067 8915 9093 9072 9066 9338 9317 9123 9077 9123 9077 9146 9771 9039 8569 8563 7660 7080	CHN PPE Non Conflict 8382 8510 8791 8375 8523 8403 7315 6574 6448 6205 6179 6149 6136 6095 6105 6137	CHN PPE Non Conflict PPT 8140 8348 8136 8013 7937 7936 7937 7936 6794 6567 6485 6593 6520 6476 6471 6485 6471 6482	BK_BT 6203 6369 6406 6703 6259 6234 6239 6236 6233 6487 6233 6487 6231 5231 5231 5231 5231	BK_RS 6235 6606 6695 6237 6107 5959 6144 6108 5899 5697 5054 4914 4882 4831 4774 4785 4789 4775	HC_BT 6002 5689 5146 5046 5041 5041 5041 5041 5041 5041 5040 5040	HC_8T Simple Mux 7293 7166 6477 6086 6003 5992 6039 6039 5989 6203 5996 6203 5996 5988 5990 5989 5984 5984 5984	HC_RS 6829 6644 6167 6168 6299 6738 6156 6031 5638 6366 5366 5366 5362 5328 5327 5283 6271 5220 5271 5220	KS_BT 9092 9092 8402 7913 7642 7636 7632 7629 7628 7626 7626 7623 7623 7623 7623 7623 7623	7234 6689 6986 6784 6785 6733 6215 6178 5976 6050 5970 5966 5964 6029 5962 5972	7669 7575 6853 6853 6801 6868 6402 6467 6397 6199 6199 6130 6132 6141 6126 6141 6126 6141 6080 6083 6069	6675 6505 6381 6889 6604 6386 6486 5967 5495 5495 5495 5495 5495 5495 5495 549	13693 13676 13673 13729 13724 13686 13789 13862 13842 13842 13820 13800 128000 128000 128000 128000 1280000000000
(115) 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	N2N 8129 8396 8238 827 7919 7897 8192 7783 7648 6877 6622 6216 6196 6162 6196 6162 63899 6173 5948 6176	PPT 8146 8285 8289 8190 8130 8074 8130 8074 8130 8074 8130 8074 8130 8074 8130 8074 8130 8074 8145 8074 8145 8074 8145 8074 8145 8074 8145 8074 8145 8074 8150 8074 8150 8074 8150 8074 8150 8074 8150 8074 8150 8074 8150 8074 8150 8074 8150 8074 8150 8074 8150 8074 8150 8074 8150 8074 8150 8074 8150 8074 8074 8052 8074 8052 80552 8055 8055 8055 8055 8055 8055 8055 8055 8055 8055	N2L06 N 8289 8172 8421 8657 8364 7901 8180 7398 7236 6668 6450 6510 65510 6357 6281 6264 6268 6259 6298 6294 6287 6296	PPE Conflict 8877 8938 8823 8791 8767 8738 8596 8544 9120 8898 8544 9120 8898 8544 9120 8898 8748 8898 8704 8885 8704 8885 8704 8722 7977 7701 7404 7187 6810 6817	CHN PPE Conflict PPT 8999 9067 8916 9093 9072 9066 9338 9317 9123 9077 9146 9771 9077 9146 9771 9077 9077 9146 9771 9077 9077 9077 9077 9077 9077 9077	CHN PPE Non Conflict 8382 8510 8791 8375 8523 9403 7363 7363 7365 6574 6574 6574 6448 6205 6179 6136 6105 6105 6105 6144 6137	CHN PPE Non Conflict PPT 8140 8348 8136 8013 7937 7936 7937 7936 6794 6567 6485 6533 6520 6477 6485 6477 6482 6477 6482 6477 6482 6472	BK_BT 6203 6369 6406 6703 6369 6229 6224 6239 6233 6487 6232 6231 6231 6231 6231 6231 6231 6230 6230 6230	BK_RS 6235 6506 6595 6237 6105 6597 6144 6108 5899 6144 6108 5899 6597 5054 4914 4882 4831 4774 4785 4776 4789 4770 4770 4770	HC_BT 6002 5689 5196 5046 5041 5041 5041 5041 5041 5040 5040 5040	HC_BT Simple Mux 7293 7155 6477 6086 6003 5992 6039 6989 6989 6989 5985 5988 5990 5987 5988 5984 5984 5984 5984 5985	HC_RS 6829 6644 6167 6158 6298 6738 6156 6031 6538 6356 6302 6328 63367 6283 63271 6320 6271 6224 6224 6224 6224 6279	KS_BT 9322 9092 8402 7913 7642 7632 7629 7628 7628 7626 7880 7625 7624 7623 7623 7623 7623 7623 7623 7623	7234 6680 6986 6784 6785 6733 6216 6178 6976 6966 5970 5966 5964 6029 5962 5972 5972 5970 5961	7669 7576 6923 6853 6801 6868 6402 6467 6397 6199 6130 6132 6141 6126 6110 6080 6063 6063 6065 6058	6675 6505 6381 8889 6584 6386 6486 6486 6486 6486 6486 6485 6497 6419 6345 6346 6346 6346 6346 6344 6336 6342 6336	13693 13676 13677 13724 13724 13686 13686 13769 13882 13842 13820 13842 13820 13842 13820 13842 13820 12871 12820 12871 12820 12871

Figure 6.6: 256 Bit RRAs synthesis results

r		Rivals			Our Other Archs.		Proposed Arbiters								
Bitwidth	Criterion	STA N2N	CHN_PPE Conflict	CHN_PPE NonConf lict	BOWTIE	STA Logn2n	ВК_ВТ	BK_RS	НС_ВТ	HC_RS	KS_BT	KS_RS	LF_BT	LF_RS	
8	Area	176	186	173	387	196	165	155	174	160	177	187	169	157	
0	Timing	9	11	9	10	9	8	10	8	10	8	9	8	10	
16	Area	452	400	346	800	441	361	321	349	363	374	329	352	332	
10	Timing	12	14	11	12	10	9	12	9	11	9	11	9	11	
32	Area	1409	1096	893	1663	899	721	628	701	920	785	808	875	878	
32	Timing	14	16	12	14	12	10	14	10	12	10	12	9	12	
64	Area	1631	1842	1719	3276	2072	1500	1395	1680	1443	1835	1416	1631	1485	
04	Timing	15	20	14	17	13	11	15	10	14	10	14	10	14	
128	Area	3480	4631	3786	6475	3641	2862	2916	3679	3285	3960	3364	3420	2915	
120	Timing	17	>20 ns	15	20	15	12	17	11	15	11	15	11	16	
256	Area	7648	7977	7353	12908	8180	5703	5597	5689	6031	8402	6215	6923	5967	
200	Timing	19	25	17	22	16	13	19	11	17	12	17	12	17	

Figure 6.7: Synthesis results at first positive or zero slack

7. CONCLUSION AND FUTURE WORK

In this thesis, we proposed two RRA architectures PPT_RRA_RS and PPT_RRA_BT. We also described another new architecture – BOW-TIE_RRA which is still under development. We optimized and implemented our rival architectures. Various modifications and optimizations are applied on these architectures to strengthen them. Then, all RRA architectures are verified and synthesized. According to synthesis results we proved that our proposed designs are area efficient and faster than the rival architectures. The fastest architecture is HC_PPT_RRA_BT and the most area efficient one is BK_PPT_RRA_RS with respect to 256-bit synthesis results. These results are simplified as follows.

According to 256-bit synthesis results, our new architectures achieve 22% area improvement and 42% timing improvement over optimized STA_RRA_N2N which is the well-known RRA design in literature. We optimized Smpl_PE block which is described in Chapter 2.

Two CHN_RRA designs were published in 2006. The authors tried to enhance the STA_RRA architecture. However, their architectures had some drawbacks. We applied some optimizations on their work. These optimizations are explained in Chapter 2. When we compared our proposed architectures' synthesis results against CHN_RRAs' results for 256-bit, we saw that we outperform them by factor in 35% for speed and 23% for area.

Savin C.E., McSmythurs T., and Czilli J. published BTS_RRA in 2004. In their paper, they represented that their architecture has $(log_2N + 4)$ logic level, $(nlog_2N + 7n - 6)$ combinational gate count and it has n-bit flop. Our proposed architecture for best timing (with Ladner Fisher Pre_Thermo block — LF_PPT_RRA_BT) has $(log_2N + 4)$ logic level too. However, its gate count is equal to $(nlog_2N + 4n) + n$ -bit flop. Thus, our proposed architecture's gate count is better than BTS_RRA.

The most recent work was carried out by Zheng S. Q. and Yang M. in January 2007. They proposed two architectures: PRRA and IPRRA. They indicated that IPRRA achieved 30.8% timing improvement and 66.9% area improvement over PPE design which is the core block of STA_RRA. Our proposed design for best timing achieved 42% timing improvement over STA_RRA, which is better than both IPPRA and PPRA.

Other two different architectures SA and PPA use different algorithms and they cannot ensure the fairness for non-uniformly distributed requests. Thus, we did not implement these architectures for comparison.

In summary, PPT_RRA_BT and PPT_RRA_RS architectures both achieve important improvements in area and speed departments compared with former RRA architectures. Due to their performance results, these architectures will play important roles for high-speed switches, routers, and the systems where arbitration is required for various purposes.

It is possible to enhance the area and timing results of BOW-TIE_RRA architecture. It consists of complex RR Blocks. So, in the future we will work on to reduce the RR Blocks' complexity of that architecture. Also, unimplemented RRA architectures will be implemented for area and timing comparison.

REFERENCES

Books

Koren, I., 2001. Computer Arithmetic Algorithms, 2nd edition, A K Peters Ltd.

Synopsys, Inc., 2007. Design Compiler 1 Student Guide, v2007.03

Synopsys, Inc., 2008. Prime Time 1 Student Guide, v2006.06

Periodical Publications

- Gupta, P. & McKeown, N., 1999. Designing and implementing a fast crossbar scheduler, *Micro IEEE*, **19** (1), pp. 20 28.
- Chao, H.J., Lam, C.H. & Guo, X., 1999. A fast arbitration scheme for terabit packet switches, *Global Telecommunications Conference*, 1999. GLOBECOM '99, pp. 1236 – 1243.
- Shin, E.S., Mooney, V.J. III & Riley, G.F., 2002. Round-robin Arbiter Design and Generation, 15th International Symposium on System Synthesis, pp. 243 – 248.
- Lee, K., Lee, S.-J. & Yoo, H.-J., 2003. A high-speed and lightweight on-chip crossbar switch scheduler for on-chip interconnection networks, ESSCIRC '03. Proceedings of the 29th European Solid-State Circuits Conference, pp. 453 – 456
- Yoghigoe, K., Christensen, K.J. & Roginsky, A., 2003. Design of a high-speed overlapped round robin (ORR) arbiter, 28th Annual IEEE International Conference on Local Computer Networks, pp. 638 – 639.
- Savin, C.E., McSmythurs, T. & Czilli, J., 2004. Binary tree search architecture for efficient implementation of round robin arbiters, (ICASSP '04). IEEE International Conference on Acoustics, Speech, and Signal Processing, 6 (5), pp. V – 333.
- Gao, X., Zhang Z. & Long X., 2006. Round Robin Arbiters for Virtual Channel Router, IMACS Multiconference on Computational Engineering in Systems Applications, pp. 1610 – 1614.

- Si Q. Z. & Mei Y., 2007. Algorithm-Hardware Codesign of Fast Parallel Round Robin Arbiters, *IEEE Transactions on Parallel and Distributed Systems*, 18 (1), pp. 84 – 95.
- Harris, D. 2003. A taxonomy of parallel prefix Networks, *Conference Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers*, pp. 2213 – 2217.

Other Publications

- Shin, E.S., 2003. Automated Generation of Round-Robin Arbitration and Crossbar Switch Logic. *PhD Thesis*. Georgia Institute of Technology, School of Electrical and Computer Engineering.
- Weber, M. 2001. Arbiters: Design Ideas and Coding Styles, *SNUG Boston*, Silicon Logic Engineering, Inc.

Kuphaldt T. R., CMOS Gate Circuitry [online], http://www.allaboutcircuits.com/vol_4/chpt_3/7.html, [citied 2 May 2008]

VITA

Name Surname	: Onur BAŞKİRT
Address	: STMicroelectronics Elektronik Araştırma ve Geliştirme A.Ş. Büyükdere Cad. İTÜ Ayazağa Kampüsü Koru Yolu ARI2 B Blok Ofis No: 3–1 34469 Maslak, Sarıyer – İSTANBUL
Birth Place / Year	: Kars / 1982
Languages	: Turkish (native) – English
Elementary School	: Ahmet Paşa Elementary School – 1994
High School	: N. M. Baldöktü Anatolian High School – 2000
BS	: Bahçeşehir University – 2005
MS	: Bahçeşehir University – 2008
Name of Institute	: Institute of Science
Name of Program	: Electrical & Electronics Engineering
Publications	H.F. Uğurdağ, Y. Şahin, O. Başkirt , S. Dedeoğlu, S. Gören, Y.S. Koçak, "Population-based FPGA Solution to Mastermind Game," Proceedings of the IEEE/NASA Adaptive Hardware and Systems Conference, Istanbul, June 2006.
	L. Eren, O. Başkirt , and M. J. Devaney, "Rule Based Motor Fault Detection," Proceedings of the IEEE Instrumentation and Measurement Technology Conference, Ottawa, May 2005.
Work Experience	: STMicroelectronics Connectivity Division – Digital IC Design Engineer September 2007 – Ongoing
	University of California, Santa Cruz School of Engineering – Visiting Researcher June 2006 – August 2006
	Bahçeşehir University EE Engineering Department – Teaching & Research Asst. September 2005 – August 2007