Integration of Dynamic Power Management Strategies for System-Level Power Optimization (Low-Power Design) with Its Formal Verification and Implementation

A thesis submitted
for award of the degree of

Doctor of Philosophy

In

Electrical Engineering

By:
Bhatt Kiritkumar Ramanbhai

DEPARTMENT OF ELECTRICAL ENGINEERING
FACULTY OF TECHNOLOGY & ENGINEERING
(KALABHAVAN)
THE MAHARAJA SAYAJIRAO UNIVERSITY OF BARODA
VADODARA – 390 001 GUJARAT, INDIA
OCTOBER – 2012

“Integration of Dynamic Power Management Strategies for System-Level Power Optimization (Low-Power Design) with Its Formal Verification and Implementation”

A thesis submitted
for the award of the degree of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

By

Bhatt Kiritkumar Ramanbhai

DEPARTMENT OF ELECTRICAL ENGINEERING
FACULTY OF TECHNOLOGY & ENGINEERING
THE MAHARAJA SAYAJIRAO UNIVERSITY OF BARODA
VADODARA – 390 001 GUJARAT, INDIA
OCTOBER – 2012

CERTIFICATE

This is to certify that the thesis titled, **"Integration of Dynamic Power Management Strategies for System-Level Power Optimization (Low-Power Design) with Its Formal Verification and Implementation"** submitted by **Shri Bhatt Kiritkumar Ramanbhai** in fulfilment of the degree of **DOCTOR OF PHILOSOPHY** in Electrical Engineering Department, Faculty of Technology & Engineering, The M. S. University of Baroda, Vadodara is a bonafide record of investigations carried out by him in the Department of Electrical Engineering, Faculty of Technology & Engineering, M. S. University of Baroda, Vadodara under my guidance and supervision. In my opinion this has attained the standard fulfilling the requirements of the Ph.D. Degree as prescribed in the regulations of the University.

OCTOBER, 2012

Guide:

Prof. A. I. Trivedi
Department of Electrical Engineering,
Faculty of Technology & Engineering,
The Maharaja Sayajirao University of Baroda
Vadodara – 390 001

Head:

Prof. S. K. Shah
Department of Electrical Engineering,
Faculty of Technology & Engineering,
The Maharaja Sayajirao University of Baroda
Vadodara – 390 001

Dean:

Dr.Prof. A. N. Misra
Faculty of Technology & Engineering,
The Maharaja Sayajirao University of Baroda
Vadodara – 390 001

DECLARATION

I, **Shri Bhatt Kiritkumar Ramanbhai** hereby declare that the work reported in this thesis titled, *"Integration of Dynamic Power Management Strategies for System-Level Power Optimization (Low-Power Design) with Its Formal Verification and Implementation"* submitted for the award of the degree of **DOCTOR OF PHILOSOPHY** in Electrical Engineering Department, Faculty of Technology & Engineering, The M. S. University of Baroda, Vadodara is original and was carried out in the Department of Electrical Engineering, Faculty of Technology & Engineering, M. S. University of Baroda, Vadodara. I further declare that this thesis is not substantially the same as one, which has already been submitted in part or in full for the award of any degree or academic qualification of this University or any other Institution or examining body in India or abroad.

October, 2012 **Shri Bhatt Kiritkumar Ramanbhai**

Acknowledgements

First and foremost, I offer my utmost gratitude to the omnipresent God, for providing me inspiration, strength, energy and patience to start and accomplish my goal.

I sincerely thanks with deep sense of gratitude to my guide **Prof. A. I. Trivedi,** who has been the main source of inspiration. His guidance and motivation has been of the greatest help to me in bringing out this work in its present shape. The direction, advice, discussions and constant encouragement given by him have been so helpful that it enabled me to complete work successfully. He created and pointed to the path and helped in every way to reach this final destination.

I would like to show my gratitude towards **Prof. S.K. Shah**, Head of Electrical Engineering Department, Faculty of Technology & Engineering, M.S. University of Baroda, who always been a catalyst and the effective source of inspiration.

My **parents** showered blessings with love and affection on me. They guided me through my first steps to this milestone in my life and always supported me. They taught me to read, think and analyze.

I owe my deepest gratitude to **Dr.Prof. S K Joshi, Prof. Manish J Desai** and **Prof. P P Nehte** who constantly supported me during my entire time-span of Ph.D. work. Their mental and moral support encouraged me to all the time to do my work. I am also thankful to all the other staff members including the supporting staff of Electrical Engg. Dept., M S University of Baroda for their timely help.

I express my sincere thanks to my friends **Ekata Mehul** and **Saifee Muffadal** who were always ready to help especially during the project testing and verification phase.

I would also like to thank my **Institute authorities and management**, **colleagues** and other **faculty** members for their direct and indirect support.

I express my thanks to my wife **Manisha** and sweet children **Vedanshi** and **Bhavyang** for all the support, inspiration and love given to me despite of all inconvenience caused to them due to my preoccupation with Ph.D. work.

Finally, last but not least, I thank all persons, who helped me in my Ph.D. work, but whose names may have been missed.

October - 2012 **Bhatt Kiritkumar Ramanbhai**

Dedicated

To

My Parents Sri. Ramanbhai & Smt. Hansaben,

My Wife Manisha and

My Children Vedanshi & Bhavyang

Abstract

Minimization of power consumption in portable and battery operated embedded systems has become an important aspect in recent era of system design and there are several reasons why power efficiency is becoming increasingly important. Most importantly portable systems which are powered by batteries performing tasks need increasing computational performance. At the same time these systems are becoming physically smaller and battery weight is becoming more significant. Users demand longer battery life, this can only be obtained either by increasing the capacity of the battery or by increasing the efficiency of the logic. The rate of progress in battery technology is very slow; hence the focus is on the digital designer to improve the logic efficiency. Also researchers believe that there are lots of opportunities for power optimization and tradeoffs emphasizing low power are available across the entire design hierarchy.

The Ph.D. thesis described here addresses the problem of system level power optimization. In the early stage, a state-of-the-art consolidated review of many existing low – power techniques is presented, which can be applied at many levels of the design hierarchy. And in later stage, several power minimization techniques have been proposed along with some of the design decisions, which are implemented at the system hardware design level. The proposed techniques include memory access stage removal, resource sharing, novel RAM addressing scheme and clock gating. These suggested modifications are less expensive and are very useful to evaluate the system power consumption at the early stage of the system design. It also helps to limit the time to market.

For verification and validation of the proposed energy saving techniques, a conventional 32 – bit 5 – stage pipeline processor working on RISC principle is taken as system under consideration and its construction is discussed; upon which the proposed power reducing strategies are implemented and a modified processor with 4 – stage pipeline structure is developed, which results in power consumption improvement without affecting the performance of the system. The power analysis is done after implementation of each

strategy and the power improvement is achieved. This newly developed system is implemented on the Xilinx – SPARTEN – 3E FPGA. The power comparison between conventional 5 – stage CPU and modified 4 – stage CPU is presented at the end and also the system performance at different frequencies of operation has been verified.

The proposed work is simulated, synthesized, tested and verified by using tools such as VHDL simulator, Xilinx Sparten – 3E FPGA as target device, Xilinx XPower Estimator -11.1 for power estimation, Xilinx ISE Suit – 13.1 tool for simulation and synthesis, XPower Analyzer for power analysis and ModelSim SE PLUS-6.5 for simulated waveform generation.

Keywords:

32 – bit Low Power Processor , Power Efficient Embedded Processor, Power Optimization, Energy Efficient Design, Low-Power Architecture, Low – Power Design, Low – Power System, Low – Power Pipelined Processor, System Level Power Optimization, System Design, Power Optimization, FPGA Implementation.

Table of Contents

Certificate i
Declaration *ii*
Acknowledgement *iii*
Dedication *iv*
Abstract *v*
Table of Contents *vii*
List of Figures *xi*
List of Tables *xiv*

Chapter 1 Introduction 1

1.1 Introduction 1
1.1.1 Leakage Current 2
1.1.2 Short – circuit Current 2
1.1.3 Switching Current 2
1.1.4 Process Technology 3
1.1.5 Reduce Leakage Power 3
1.1.6 Reducing Supply Power 4
1.1.7 Higher Density of Integration 4
1.1.8 Reducing Switching Activity 4

1.2 Motivation 5

1.3 Research Objectives 7

1.4 Contribution to the Thesis 8

1.5 Thesis Organization 10

Chapter 2 Background Work 12

2.1 Commercially Available FPGAs 12
2.1.1 Introduction 12
2.1.2 FPGA Basis 12
2.1.2.1 FPGA Architectures 12
2.1.2.2 Logic Blocks 13
2.1.2.3 Interconnect Resources 13
2.1.2.4 Classes of commercial FPGAs 14
2.1.3 Currently available FPGAs Technology 15
2.1.3.1 Programming Technology 15
2.1.3.2 Logic Blocks Architecture 19
2.1.3.3 Interconnections 21
2.1.3.4 I/O Structures 24
2.1.3.5 Other Resources 24

2.2 Power Consumption Model of MOS-based Circuits 26

2.2.1 Introduction 26
2.2.1.1 The CMOS Inverter 26
2.2.2 Power Consumption of Complementary CMOS 29
2.2.2.1 Static Power 29
2.2.2.2 Dynamic Power Caused by Load Capacitance 31
2.2.2.3 Dynamic Power Caused by Short-Circuit Currents 32
2.2.3 Power Consumption of SRAM 33
2.2.4 Power Consumption of Input / Output Circuits 34
2.2.4.1 Input Circuits 34
2.2.4.2 Output circuits 35
2.2.5 Power Consumption in Clock Circuits 37
2.3 Power Consumption in SRAM-based FPGAs. *37*
2.4 Conclusion *38*
Chapter 3 Power Reduction Techniques for Embedded Systems 40
3.1 Introduction *40*
3.2 Defining Power Dissipation in CMOS Circuits *41*
3.3 Power Reduction Methodologies for Various Abstraction Levels *43*
3.3.1 Logic and Circuit Level Power Reduction Techniques 43
3.3.1.1 Transistor Sizing 43
3.3.1.2 Transistor Reordering 44
3.3.1.3 Half Frequency and Half Swing Clocks 45
3.3.1.4 Logic Gates Restructuring 45
3.3.1.5 Technology Mapping 46
3.3.1.6 Low Power Flip-Flops 46
3.3.1.7 Low – Power Control Logic Design 47
3.3.1.8 Delay-Based Dynamic Supply Voltage Adjustment 47
3.3.2 Low – Power Techniques for Interconnect 47
3.3.2.1 Bus Encoding and Cross Talk 48
3.3.2.2 Low Swing Buses 48
3.3.2.3 Bus Segmentation 49
3.3.2.4 Adiabatic Buses 50
3.3.2.5 Network-On-Chip 50
3.3.3 Low Power Techniques for Memories and Memory Hierarchies 51
3.3.3.1 Splitting Memories into Smaller Sub-systems 51
3.3.3.2 Augmenting the Memory Hierarchy with Specialized Cache Structures 52
3.3.4 Power Reduction at Architecture Level 53
3.3.4.1 Adaptive Cache 53
3.3.4.2 Adaptive Instructive Queues 54
3.3.4.3 Algorithms for Reconfiguring Multiple Structures 55
3.3.5 Dynamic Voltage Scaling (DVS) 55
3.3.5.1 Unpredictable Nature of Workloads 56
3.3.5.2 Indeterminism and Anomalies in Real Systems 56
3.3.5.3 Interval – Based Approaches 57
3.3.5.4 Inter task Approaches 57
3.3.5.5 Intra task Approaches 58
3.3.5.6 The Implications of Memory Bounded Code 59

3.3.5.7 Dynamic Voltage Scaling in Multiple Clock Domain Architectures 60
3.3.6 Algorithmic Level Power Reduction Techniques 60
3.4 Introduction to Emerging Technologies for Power Reduction 64
3.4.1 Fuel Cells 64
3.4.2 MEMS 65
3.5 Conclusion 66
Chapter 4 System Architecture 68
4.1 Introduction 68
4.2 Processor Architecture 68
4.2.1 IF Stage 70
4.2.2 DC Stage 70
4.2.3 EX stage 74
4.2.3.1 Data memory access 74
4.2.3.2 ALU 74
4.2.4 WB stage 75
4.3 Instruction Set Formation 75
4.4 Sub-modules of Processor 78
4.4.1 ALU Design 79
4.4.2 Register File Design 79
4.4.3 Data Memory Design 80
4.4.4 Instruction Memory Design 80
4.4.5 Instruction Decoder 80
4.4.6 Control Unit Design 83
4.5 Multiplier Unit & Its Logic 85
4.6 Clock Distribution Network 87
4.7 Data Forwarding and Data Dependency 91
4.7.1 Structural Hazards 92
4.7.2 Data Hazards 92
4.7.3 Control Hazards 93
4.8 External Interface 94
4.9 Instruction Simulation and Verification 94
4.10 FPGA Design Flow 94
4.11 Summary of Synthesis Report 95
4.12 Power Estimation Reports of Complete Architecture 96
4.13 Conclusion 99
Chapter 5 Proposed Strategies for Power Optimization 101
5.1 Introduction 101
5.2 Architecture of 32 – bit 5 – Stage Pipeline (Conventional/Standard) CPU 101
5.3 Proposed Strategies for Power Reduction 109
5.3.1 Memory Access Stage Removal Technique 109

5.3.2 Resource Sharing Technique 116
5.3.3 Novel RAM Addressing Scheme 119
5.3.4 Clock Gating 120

5.4 Verification of 4- Stages CPU After Implementation of Power Optimization Strategies 128
5.4.1 Performance Verification 128
5.4.2 Graphical Representation of Power Requirement 133

5.5 Conclusion 136

Chapter 6 Conclusions and Future Work 138

6.1 Summary and Contributions 138

6. 2 Future Work 139

6.3 Closing Remark 140

List of Publication 141

Bibliography 143

List of Figures

Figure 1.1: Power Reduction Opportunities 3

Figure 2.1: Anti Fuse - Switch 16

Figure 2.2: The EPROM Transistor 16

Figure 2.3: SRAM Cell with (a) Pass – Transistor, (b) Transmission Gate, (c) Multiplexer 18

Figure 2.4: HRL SRAM Cell 18

Figure 2.5: A 2- Input LUT 19

Figure 2.6: Multiplexer – Based Logic Cell 20

Figure 2.7: Multiplexer and Basic Gates LCELL Proposed by Atmel TM (Courtesy of Atmel™ Co.) 21

Figure 2.8: Actel™ ACT1 Interconnect Architecture (Row – Based) 22

Figure 2.9: XC4000E/XL/XV Interconnect Architecture (Segmented – Based) (Courtesy of Xilinx™ Corporation) 23

Figure 2.10: XC4000 Series Interconnect Resources (Courtesy of Xilinx™ Co) 23

Figure 2.11: Hierarchical Interconnect (Courtesy of Altera™ Corporation) 24

Figure 2.12: Embedded Memory (a) Block, (b) Distributed Cells 25

Figure 2.13: Standard CMOS Inverter 26

Figure 2.14: DC Transfer Characteristics of a CMOS Inverter, (a) Voltage and (b) Current 27

Figure 2.15: Input Voltage and Short – Circuit Current 32

Figure 2.16: TTL Input Buffer 35

Figure 2.17: Tri – State Output Buffer 36

Figure 3.1: Transistor Reordering 44

Figure 3.2: Gate restructuring (Figure adapted from the Pennsylvania State University Microsystems Design Laboratory's tutorial on Low Power Design) 45

Figure 3.3: Low Voltage Differential Signalling 49

Figure 3.4: Bus Segmentation 49

Figure 3.5: Two Bit Charge Recovery Bus 50

Figure 3.6: Dead Block Elimination 54

Figure 3.7: Performance Versus Power 62

Figure 4.1: Detailed Architecture of 4 – Stage Pipelined Processor Under Consideration 71

Figure 4.2: An ALU Architecture for 4 – Stage CPU 75

Figure 4.3: Formats for Various Instructions 77

Figure 4.4: Detailed Internal Architecture of Instruction Decoder 81

Figure 4.5: Internal Architecture of Multiplexer 82

Figure 4.6: Diagram of Instruction Decoder with all relevant Signals 83

Figure 4.7: Main Features of Multiplier Block 85

Figure 4.8: Pin Diagram of MULT18X18SIO 86
Figure 4.9: Four possible Configures for the B_INPUT Attribute and BREG Attribute 87
Figure 4.10: Xilinx SPARTAN – 3E Clock Distribution Network (Courtesy of Xilinx™ Co.) 88
Figure 4.11: Internal Element of 2 – to -1 Multiplexer (Courtesy of Xilinx™ Co.) 89
Figure 4.12: Quadrant – Based Clock Routing (Courtesy of Xilinx™ Co.) 90
Figure 4.13: Data Dependency and Data Forwarding 94
Figure 4.14: External Interface 94
Figure 4.15: FPGA Design Flow 95
Figure 4.16: Summary of Estimated Power Distribution Report 97
Figure 4.17: Graphical Representation of Estimated Power Requirements for Internal Modules and Effect of Various Parameters on Power Consumption 98
Figure 5.1: Detailed Architecture of 5 – Stage Pipelined Conventional CPU 102
Figure 5.2: Summary of Power Consumption Report for 5 – Stage Pipeline CPU 103
Figure 5.3: Instruction Fetch for 5 - Stage CPU 104
Figure 5.4: Instruction Decode for 5 – Stage CPU 105
Figure 5.5: RAM Address for 5 – stage CPU 106
Figure 5.6: Instruction Execute for 5 - Stage CPU 107
Figure 5.7: Write Back Stage for 5 – Stage CPU 108
Figure 5.8: Summary of Power Consumption Report for 4 – Stage Pipeline CPU (After Implementation of Memory Access Stage Removal 111
Figure 5.9: Instruction Fetch for 4 – Stage CPU 112
Figure 5.10: Instruction Decode and Operand Fetch for 4 – Stage CPU 113
Figure 5.11: Instruction Execute for 4 – Stage CPU 114
Figure 5.12: Write Back for 4 – Stage CPU 115
Figure 5.13: MUX for Resource Optimization 117
Figure 5.14: MUX with Opcode as Selection Logic 118
Figure 5.15: Summary of Power Consumption Report for 4 – Stage Pipeline CPU (After Implementation of Resource Sharing Strategy) 119
Figure 5.16: RAM Address Multiplexer 120
Figure 5.17: Clock – Enabled Global Buffer Resource 121
Figure 5.18: Gated Clock – Not Preferable 121
Figure 5.19: Clock Enable – Efficient way of Gating a Clock Signal 121
Figure 5.20: Summary of Power Consumption Report for 4 – Stage Pipeline CPU (After Implementation of RAM Addressing Scheme and Clock Gating along with earlier Techniques) 125
Figure 5.21: Simulated Waveforms for Clk_gating Signal Varification for ALU Related Instructions 126
Figure 5.22: Simulated Waveforms for Clk_gating Signal Verifcation during RAM Access Instructions 127
Figure 5.23: Verification of Instruction Fetch for 4 – Stage CPU 129
Figure 5.24: Verification of Instruction Decode and Operand Fetch for 4 – Stage CPU 130
Figure 5.25: Verification of Instruction Execute for 4 – Stage CPU 131

Figure 5.26: Verification of Write Back Operation of 4 – Stage CPU 132
Figure 5.27: Graphical Representation of Estimated and Actual Power Consumption of 5 – Stage CPU 133
Figure 5.28: Power Consumption Requirement after Implementation of Power Saving Techniques..... 134
Figure 5.29: Overall Power Consumption Comparisons 135
Figure 5.30: Graphical View of Power Consumption Comparison at Different Clock Frequencies 136

List of Tables

Table 1.1: Power Dissipation of Microprocessors (Source: UK Electronics Forum) 5
Table 1.2: Technological Evolutions (Source: Semiconductor Industry Association) 6
Table 1.3: Allowable maximum powers for the coming years (Source: ITRS) 6
Table 2.1: Commercially Available FPGA Architecture 15
Table 4.1: Summary of All the Instructions Supported by this Processor 77
Table 4.2: Summary of Control Signals 83
Table 4.3: Pipeline Registers Flow Through Different Stages of Pipeline 91
Table 4.4: Summary of Synthesis Report 96
Table 5.1: List of Instructions under Consideration 116
Table 5.2: Summary of Power Results 133
Table 5.3: Summarizes the Results of Power Requirements 134
Table 5.4: Power Consumption of CPUs at Different Frequencies 135

Chapter 1
Introduction

1.1 Introduction

There are many possible facts because of which the power efficiency is becoming important consideration. The most portable systems used in recent era, which are powered by batteries, are performing tasks requiring lots of computations. At the same time these systems are becoming physically smaller in size and battery weight is becoming more important factor. Users demand longer battery life and this can only be obtained either by increasing the battery capacity or by increasing the logic efficiency. The rate of development in battery technology is very slow; hence, to improve efficiency the focus is on the system designers. There are many other reasons because of which the system power consumption is becoming important aspect. As the heat dissipation of components increases it becomes more difficult to provide sufficient cooling through the good packages, heat sinks or fans and it also increases the cost. Furthermore, higher temperatures increase the strain on the component and hence reduce its trustworthiness. Other electrical issues are also need attention, to provide a supply with proper capacity demands a big number of bond wires between the chip and the package, and a huge amount of the potential signal routing space is occupied by power distribution. High current densities can lead to electro-migration and at the system level, higher power requirement demands larger and expensive power supplies. These factors, and many others which are presented in [1], together made power efficiency an important factor for the design of digital systems.

Designing low – power system requires some methodologies to be implemented at every level of abstractions such as system level, architecture, algorithm level and circuit level. The prime components of such methodologies are estimation and optimisation as discussed in [2], to understand these components one must know that how the energy is getting dissipated. It is understood that low-power design technology means system should dissipate lowest energy when actually it performs and in case of CMOS technology; it is proved that it consumes

considerably less amount of energy. There are three major sources of power consumption in CMOS circuits as described in [3]. Power dissipation is either static or dynamic. Static power dissipation is caused due to leakage and short circuit currents while dynamic power dissipation is due to occurrence of switching activities within the circuit. Dynamic power is the biggest contributor to the power dissipation within the system and hence it catches attention. The proposed power reduction strategies intend to reduce dynamic power by reducing unwanted transitions within the system and hence, the total power consumption.

1.1.1 Leakage Current

It is primarily determined by the technology used in its fabrication and consists of reverse bias current in the parasitic diodes formed between source and drain diffusions and the bulk region in a MOS transistor described in [4]. The Sub threshold current that arises from the inversion charges that exists at the gate voltage between the threshold voltages. This is also known as static power consumption and is proportional to the number of transistors which are in the OFF state.

1.1.2 Short – circuit Current

It is due to DC path between the supply rails during the output transitions explained in [5].

1.1.3 Switching Current

It is dissipated when capacitive loads are charged and discharged during logic changes. In any digital System, to understand the whole power estimation of a system one must understand the CMOS inverter and its internal structure presented in [6] [7].

A low level of the design space is not of much use for the designer, since the defined design flow ends at the gate level. Techniques that effect lower levels are out of the scope for this dissertation work. Even though, the given information is relevant for a complete understanding of the matter. At the higher level of abstraction at which a methodology is applied, the more promising and effective savings on power dissipation can be achieved which is described through Figure 1.1. This thesis focuses only to deal with system level (Behaviour Level) where up to 25% power reduction possibilities yet to be explored as per

ITRS reports and the other two levels i.e. transistor level and a layout levels are not within the scope of this work.

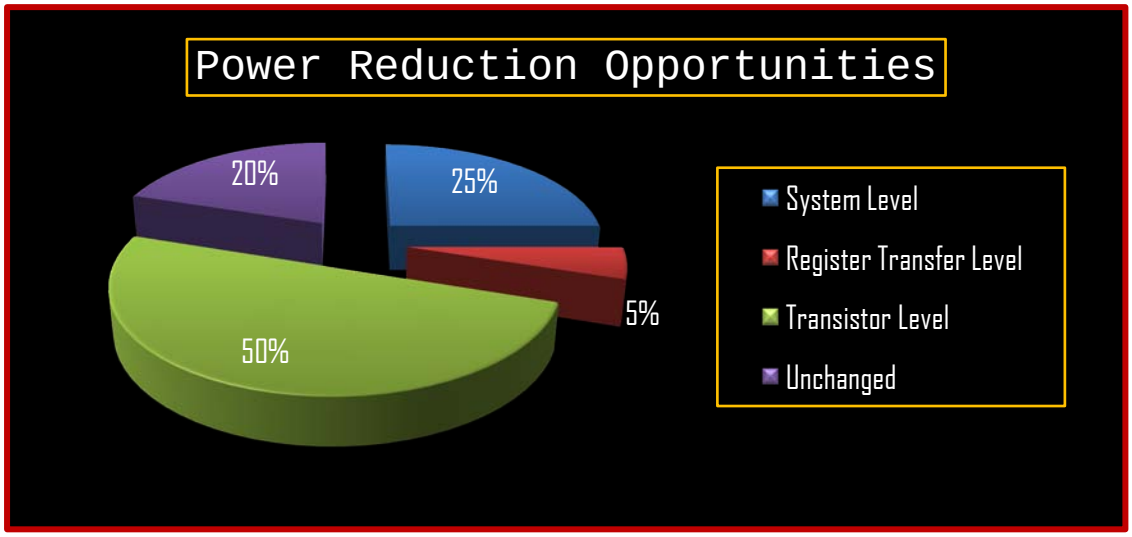


Figure 1.1: Power Reduction Opportunities

1.1.4 Process Technology

To give a complete picture on low power techniques, there is no relevance for the designers, as the described effects base on a level of design abstraction which is not in designer's scope. In reducing capacitance is the effective methodology of reducing power supply voltage. Power savings through higher density of integration can be done and the reducing Capacitance C_{out} can be described as the sum of three capacitances:

$$C_{out} = C_{f0} + C_w + C_p \qquad [1.1]$$

C_{fo} is the input capacitance of fan-out gates, C_w the wiring and C_p the parasitic capacitance. For deep sub-micron technologies C_w is the most dominant component and also difficult to estimate. And the effect of "cross-talk" have to be considered. Designers are not in charge of placing and routing a design below gate level and have therefore no major role to play. Only lay-designer and technology vendors are able to deal with this parameter.

1.1.5 Reduce Leakage Power

Generally, $P_{dynamic}$ outweighs $P_{leakage}$, if the design is idle most of the time and switching activity is low [8]; then these effects are out of our design flow. The technology vendor is responsible for the design flow at this level of abstraction.

1.1.6 Reducing Supply Power

Reducing supply voltage is the best way of saving power since its influence is quadratic; but the drawback is that it reduces the switching speed as suggested by equation 1.2.

$$P_{dynamic} = K\, C_{out}\, V_{dd}{}^2\, \mathrm{f} \qquad [1.2]$$

Usually a circuit is designed to meet certain timing constraints which will be violated when the supply voltage is reduced. The solution is called “architecture-driven voltage scaling”. The level of concurrency is raised by adding more hardware to the design. Typical methodologies are pipelining and parallelization. This eases the timing restrictions. In spite of having more hardware that is consuming power, the overall power dissipation is reduced because of the quadratic influence of V_{dd} [9] [10].

1.1.7 Higher Density of Integration

By minimizing the scale of a circuit, its capacitances and therefore its dynamic power dissipation can be reduced. The technology is fixed to the structures of the vendors’ technology; hence, there is no scope for designers.

1.1.8 Reducing Switching Activity

In order to reduce power dissipation effectively, the low power methodologies must target this source to control. As discussed, earlier the designers have no control on V_{dd} and only a minor one on C_{out}, then the switching activity is left and is a component upon which we can concentrate [11] [12]. Many existing along with the newly suggested methodologies can be tried to reduce the switching activity to a greater extent at the system level [13].

The existing techniques are Minimization of Glitches; Minimization of the Number of Operations; Low Power Bus/Bus Inversion; Charge Recovery and Adiabatic Systems;

Scheduling and Binding Optimization ; Power Down Modes ; Power Supply Shutdown ; Clock Gating; Enabled Flip-Flops; Memory Partitioning; Routing approach to reduce the Glitches; Priority Selection; Pipeline Structures ; Switching algorithm; Use of don't care conditions; Use of Gray coding in place of Binary coding; Logic Optimisation ; Supply Voltage Adjustment ; Retiming ; Pre-computation ; Clocking Schemes and Asynchronous Logic ; Data-path activity management, etc..

1.2 Motivation

From above discussion, it is clear that power is a key control for high-performance systems. With large integration density and improved speed of operation, systems with high clock frequency are emerging. These systems are based on high-speed products such as microprocessors. The cost associated with packaging, cooling and fans required by these systems are increasing significantly. The Table 1.1 shows the power consumption of various microprocessors that operate in a range of 50 to 300 MHz. These data shows that power consumption becomes too excessive at higher frequencies.

Another issue related to power consumption is reliability. An excessive increase in power dissipation can reduce the performance of the circuit [10], which may sometimes enables the failure mechanism such as silicon interconnect fatigue, package related failure, electrical parameter shift, electro-migration and junction fatigue. Reliability problems coupled with power consumption issues, when scaling down to 0.5µm, have driven the electronics industry

Table 1.1: Power Dissipation of Microprocessors (Source: UK Electronics Forum)

Processor	Clock (MHz)	Technology (µm)	VDD (Volts)	Power Peak (Watts)
Intel Pentium & Onwards	53	0.80	5.00	16
DEC Alpha 21064	200	0.75	3.30	30
DEC Alpha 21164	300	0.50	3.30	50
Power PC 620	133	0.50	3.30	30
MIPS R10000	200	0.50	3.30	30
UltraSparc	167	0.45	3.30	30

to adopt lower supply voltages. New standards for ICs operating voltage such as 3.3 volts, 2.5 volts and 1.8 volts are adopted. The effect of lowering the supply voltage results into low power consumption. But since size, density, frequency and the number of I/O per package are increasing drastically, power dissipation increases also. The Table 1.2 shows the evolution of ICs technology and the increment of power consumption.

Table 1.2: Technological Evolutions (Source: Semiconductor Industry Association)

Parameters	1995	1998	2001	2004	2007	2010
Technology (µm)	0.35	0.25	0.18	0.13	0.1	0.07
DRAM size Bits	64M	256M	1G	4G	16G	64G
Transistors per µP	12M	28M	64M	150M	350M	800M
Gates ASIC	5M	14M	26M	50M	210M	430M
Frequency (MHz)	300	450	600	800	1000	1100
Metal Layer	5	5	6	6	7	8
Supply (Volts)	3.3	2.5	1.8	1.5	1.2	0.9
Power (Watts)	80	100	120	140	160	180

We must consider that most recent processors can work at 1GHz or more. The power consumption trends for MPUs and high performance ASICs shown in the following Table 1.3 predicted by the ITRS; which are classified into three categories.

Table 1.3: Allowable maximum powers for the coming years (Source: ITRS)

Category	2012	2014	2016	2018	2020
High-Performance with Heat sink (W)	198	198	189	198	198
Cost – Performance (W)	125	137	151	151	157
Battery (W) – (Low Cost/Hand Held)	3.0	3.0	3.0	3.0	3.0

For High-performance desktop applications, the heat sink on package is permitted; for cost-performance, the economical power management solutions of the highest performance are the most important and the portable battery operations.

The power consumption is continued to increase even though the use of a low supply voltage. The increased power consumption is due to higher chip operating frequency; the higher interconnect overall capacitance and resistance, the increasing gate leakage which is exponentially growing and scaling on-chip transistors. The saturation in battery technology, the data given in Table 1.1, Table 1.2 & Table 1.3 and the high speed applications in current era demands the strategic development of system level designing methodology which meets the power requirement.

Dynamic power management strategies is the domain which has very strong potential to meet the objective and as mentioned in Figure 1.1 there are passages lies for further development. Many techniques have been developed in recent years and the conventional power techniques have been tried on most systems. But still there is a scope for development which covers many system specific techniques to overcome certain limitations and helps to optimize the average power consumption of the system to the greater extent.

Hence, main motivation of the work is to design, develop and implement various dynamic power saving strategies together upon the specific system, which can optimize the system level power consumption.

1.3 Research Objectives

In this work, Xilinx SPARTAN-3E FPGA platform is used for implementation and the main objectives behind the work are listed below:

- To understand the requirement for the processors and to design the 32 – bit processor with 4 – stage pipeline structure based on RISC Principle along with its RTL coding.

- Separate memory for both code and data is used and on chip Data memory (2048X32 bits) as well as code memory (2048X40 bits) are made using Xilinx block memory for both types of memory of the processor, complete architecture is to be developed along with Data Forward Unit which is required to provide proper data flow to the ALU and Hazard Detection Unit to sense the various data hazards because of which proper data forwarding is not restricted and the pipeline stages stalls for one or two cycles in order to ensure the instruction execution with the correct data set, Formation of instructions (not all but sub-

set) are mainly for three types i.e. register type, immediate type and the branch type, which are to be used to carry out the work.

- To develop Whole system using VHDL simulator and validated through waveforms generated using ModelSim SE – 6.5. The power estimation and analysis is to be carried by using Xilinx ISE – 13.1 using Xpower Estimator -11.1 and Xilinx Xpower Analyser. Also synthesized for Xilinx Family FPGA target board and synthesis report is produced.

- To develop and implement various low – power strategies to be implemented at hardware level up on the system under consideration for power reduction purpose.

- To verify the implementation to claim as low-power embedded system by making power comparisons using the results received from the Xpower Analyzer with and without power considerations.

- To implement a suggested novel strategies at system level and to carry overall Dynamic Power analysis for the developed system.

1.4 Contribution to the Thesis

- 32- bit processor has been developed with 5 and 4 - pipeline stages based on RISC principle comprising of Data forward unit and Hazard Detection Unit. RTL coding for processor has developed and verified. Formation of required instructions for the processor has been done with verification. These instructions are mainly of three types: Immediate, register and branch.

- System as a whole is developed using VHDL listing, synthesized and tested by down loading into Xilinx family FPGA and generated the synthesis reports for with and without modification of the implementation.

- Normal pipeline stages have been modified and reconfigured pipeline stages have been implemented with special data path activity management logic.

- Normally, recognition of dependency is carried in EX stage. In our processor design, we do it in DC stage and use pipeline registers to transfer to EX stage. DC stage save some hardware like logic gates by using common logic with other decode circuits in shared fashion. Also time utilized by EX stage will be reduces because the signals like ADEPEN and BDEPEN which are available immediately at the beginning of EX stage.

- The newly developed power reduction logic is employed along with multiplexers; which decide whether to bypass the data or to send to the next stage, the control block generate the control signal which act as select signal for the multiplexers.

- The controlled mechanism for clock signal is developed using a unique logic, which uses the status of the current instruction and the control signal generated by control unit to forward the signal to the concern pipeline stage only, that is the pipeline registers for write operation are to be disabled for the duration of execution cycle, it is employed at the architecture level also to prevent the clock signals to reach to various modules of the processors when it is not in use. The absence of clock signal prevents register and/or flip-flops from changing values, hence input to combinational circuits remains unchanged and no switching takes place during this period. It is possible because the architecture of ALU is designed in modular form; the execution logic is developed in a way so the operation performed by ALU is done in sub-part inside the ALU. As almost all the instructions use ALU, hence only those parts of the ALU should remain ON which is to be used by the current instruction and rest are to remain OFF. Each of the modules of the ALU is preceded by a set of transmission logic gates controlled through the ALU control unit; which allow the data to pass through, otherwise they simply put that portion of ALU in an electrically disconnected state.

- It is known that buses are the biggest source of power consumption, for the data to be transmitted over the bus; the care has been taken for hardware/software partitioning and system has been designed by keeping view that very less communication is to be done with IO and the most of all components are made available on chip, so power consumption load from the buses has been eliminated.

- Thus, Power results are achieved by implementing the various power saving techniques such as memory access stage removal, resource sharing, RAM Addressing Scheme and a

Clock Gating on the system under consideration at hardware level and finally the power dissipation comparison for modified 4 – stage pipeline CPU with the conventional 5 – stage CPU has been made to the satisfactory level.

1.5 Thesis Organization

The main goal in this thesis is to construct a complex system such as CPU and implement it on Xilinx FPGA family; also discusses the power consumption in FPGA and implements various proposed strategies at the design level to reduce the dynamic power to have system level low power design without making any change at the architecture level of the existing FPGA.

This dissertation thesis is organised in six chapters. This chapter has discussed introduction, research objectives and motivation for the low-power design and the detailed discussions on relevant issues are presented in the subsequent chapters.

Chapter 2 is based on a literature survey and it presents the brief description of different FPGAs technologies, its internal architectures and programming technologies and an overview of various static and dynamic power consumption sources in the MOS – Based circuits.

Chapter 3 describes the various abstraction levels of the system design and also discusses the various system level dynamic power reducing techniques which can be applied at different levels of the system, this chapter is also on the basis of literature survey and incorporates the survey of system level power reduction methodologies.

Chapter 4 includes the complete construction of modified 4 – stage pipelined CPU as a system under consideration, formation of its instruction set. The instructions considered here are only those which are useful to carry out the work, not a whole instruction set. The construction of CPU is represented in this chapter and its verification is discueed in Chapter 5 through the simulated waveforms. This chapter also presents the power estimation as power budget is essential component for designer to have power optimization.

Chapter 5 deals with the design of conventional 5 – stage CPU and the development & implementation of newer strategies called resource sharing and memory access stage removal, A Novel RAM Addressing Scheme and Clock Gating, which are applied on the system under consideration and derive the comparison of power dissipation for with and without implementation of these newer strategies. It also includes implementation and verification of low-power CPU designed using VHDL coding, power analysis has been carried by using Xpower Analysis and synthesized on Xilinx FPGA. Results from the experimental set – up is also a part of the chapter. It incorporates the summary of power reports and different power consumption comparisons for the system under consideration.

Finally, Chapter 6 incorporates our conclusions and the future work. This chapter is concluded by proposing some future research axes that can be explored by using this dissertation as start point in the area of low-power designs.

Chapter 2
Background Work

The proposed work is to be implemented on the Xilinx FPGA; hence it is necessary to study the FPGAs, its power consumption sources and the related mathematics to understand the system implementation completely, also the related work is presented in [14] [15] . The following sections of this chapter are based on a literature survey on FPGA technologies, its internal architectures and programming technologies. It also incorporates an overview of various static and dynamic power consumption sources.

2.1 Commercially Available FPGAs

2.1.1 Introduction

An FPGA comprises of an array of logic blocks which are interconnected through interconnection resources and has three main components: Logic Blocks, I/O blocks and programmable routing and recent FPGAs also contain the embedded memory cells and Phase-Locked Loop (PLL) blocks. This section discusses the different FPGA architectures, the basic technologies used to make FPGAs programmable, the different types of logic blocks, I/O elements, and interconnect elements are also included. It also covers most of programming technologies and SRAM based FPGA architectures.

2.1.2 FPGA Basis

2.1.2.1 FPGA Architectures

There are large number of varieties of FPGAs are available in the market with different internal architecture, such as the size and structure of logic blocks and that of the interconnect resources.

The FPGA architectures can be categorised based on the size and flexibility (or granularity) of the Logic Cell (LC), and based on the routing (or interconnect) architecture [16] . Also the FPGAs can be classified based only in their routing architecture [17]. The following discussion describes both types of classifications.

2.1.2.2 Logic Blocks

The structure and content of a LC (or logic cell/block) can be as simple as 2-input NAND gates or can be more complex structure with a Multiplexer or Look-Up Tables (LUT). For some FPGAs, a logic block (LB) corresponds to an entire PAL-like structure. There is number possibilities to define the logic block as a more complex circuit, consisting of several sub-circuits and having more than one output. But most logic cells contain D-type Flip-Flops in order to implement sequential circuits.

Logic blocks can often classified by their granularity. The granularity of a LC can be defined in different ways, such as the number of transistors, the number of Boolean operations that can be realized by the LB, or the number of inputs and outputs of the block, which can also be classified into two categories:

1. The fine-grain logic blocks architecture consists of few transistors with programmable interconnect resources. The advantage of this type is the high LB utilization can be achieved, but they need many wires and programmable switches which increases additional chip area, timing delay and power consumption.

2. Coarse-grain architectures are based on the ability of a multiplexer that connects each of its input to a constant or to a signal to implement different logic functions.

The most important advantage of Multiplexer-based LBs is that they provide a high degree of functionality with a less number of transistors which is achieved at the cost of big number of input requirements capable to place a high demand on routing resources e.g. ACTEL^{TM} FPGA Logic Block.

2.1.2.3 Interconnect Resources

The routing architecture of an FPGA is the way in which the programmable switches and the wires are placed into the circuit to allow the programmable interconnection between the logic and I/O blocks which includes wire segments of varying lengths and interconnection blocks.

The number of wires in a FPGA affects the density achieved by the device, if the used wires are inadequate, then a small number of logic blocks are achieved. On the other hand, an excessive number of wiring segments can increase the die size and reduce the utilization efficiency.

Routing architectures of FPGAs have to accomplice two constraints: routability and speed. The routability is the capability of the FPGA to accommodate all nets for a typical application even if the number of wire has to be predefined for blank (or unprogrammed) FPGA configuration. FPGAs are classified into three basic architectural groups based on the logic blocks size, functionality and, the structure of the interconnect resources [16] [17]:

1. Row-based FPGA architecture which consists of coarse-grain logic blocks in rows, which are divided horizontally by programmable routing channels. The programmable routing contains wire segments of different lengths e.g. ACTEL™ FPGAs.
2. Symmetrical FPGAs are based on large grain logic blocks called Configurable Logic Blocks (CLBs), it is a matrix of CLBs with horizontal and vertical routing channels and can also be visualized as a net of programmable wires for direct (or neighbourhood) connections, along with general purpose and long lines e.g. Xilinx™, Lucent™ and Atmel™ devices.
3. The cellular architecture consists of two-dimensional symmetrical arrays of Logic Cells with a hierarchical interconnect structure in which LCs can be connected directly about each other by using a low level of interconnect (or local interconnect). The longer connections use a high level of interconnect (formed by long wires) to reach one section from another, or one LCELL to an I/O cell e.g. Motorola™ and Altera™ FPGAs.

2.1.2.4 Classes of commercial FPGAs

In general, there are two kinds of FPGA architectures: fine-grained and coarse-grained. Fine-grained devices consist of a large number of relatively simple logic blocks such as either a two-input logic function or a 4-to-1 multiplexer and a D-type flip-flop (i.e. Actel™,

Atmel™); whereas coarse-grained comprises of large logic blocks having two or more look-up tables (LUT) and DFFs. Usually, these architectures are based on 4-input LUTs (i.e. Altera™, Lucent™, Vantis™ and Xilinx™). The Table 2.1 summarizes the classification of some FPGA architectures:

Table 2.1: Commercially Available FPGA Architecture

Architecture	Anti-Fuse	Flash	EPROM	SRAM
Coarse-grained	QuickLogic (pASIC)	Cypress (Delta)	Cypress (Ultra)	Altera (Flex, Apex) Atmel (AT40K) Lucent (Orca) Vantis (VFI) Xilinx (XC3000, XC4000, Spartan, Virtex)
Fine-grained	Actel (ACT)	Actel (ProAsic) GateField	GateField (GF260)	Actel (SPGA) Atmel (AT6K)

2.1.3 Currently available FPGAs Technology

2.1.3.1 Programming Technology

In this section we expose the different programming technologies. FPGAs consist of two layers: a programmable layer that contains programmable elements, such as low resistance and low-capacitance interconnect switches; and a logic layer which contains logic blocks, I/O elements and interconnect. The programming elements are used to implement the programmable connections among all the internal logic elements. Several different programming technologies are used to implement the programmable switches in FPGAs. In the following section we will describe some of the currently used technologies.

a) Anti-fuse

The anti-fuse switch or a silicon composition is a device with two-terminal having un-programmed state provides a very high resistance between its terminals. When a high voltage (from 11 to 21 volts) is applied between both terminals, the anti-fuse is blown to create a low-resistive and permanent link. The anti-fuse developed by Actel™ consists of three layers. The top layer is a conductor made of poly-silicon. The middle layer consists of an oxide-nitride-oxide (ONO) chemical composition used as insulator. The bottom layer is a conductor of

negatively doped diffusion. Un-programmed, the ONO anti-fuse insulates the top layer of metal from the bottom layer.

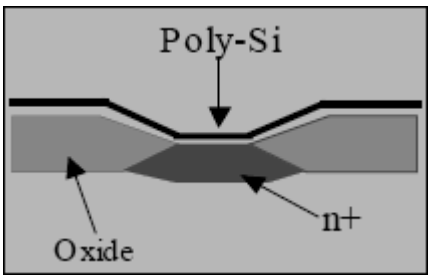


Figure 2.1: Anti Fuse - Switch

When the anti-fuse is programmed, a current of about 5 mA is passing through the device. This procedure generates enough heat in the dielectric to cause it to melt and form a conductive link between the poly-Si and the n+ diffusion shown in Figure 2.1. Both, the bottom layer and top layer of the anti-fuse are connected to metal wires. When the anti-fuse is programmed, a very low resistance connection is formed between the two metals wires. Many other vendors have developed the many other ways to form anti-fuse switch.

b) EPROM

EPROM programming technology is used by many developers such as Altera™, Atmel™, Cypress™ and Xilinx™ FPGAs, which is same as that used in EPROM memories. Figure 2.1 shows the EPROM transistor which is based on the NMOS transistor and has two gates [15]: a floating gate and a select gate. The floating gate is positioned between the selected gate and the transistor's channel which is called "floating" because it is not electrically connected to

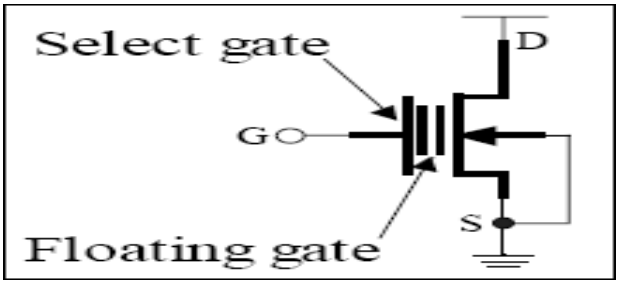


Figure 2.2: The EPROM Transistor

any circuit. During the unprogrammed state, no charge is stored on the floating gate and the EPROM transistor can be turned ON by applying a voltage in the selected gate (like a

NMOS). When the transistor is programmed, a charge is trapped under the floating gate. This charge forces the transistor to be permanently turning OFF.

An EPROM transistor gets re-programmed by eliminating the trapped charge from the floating gate. This can be done by exposing the gate to ultraviolet light exits the trapped electrons to the point where they can pass through the gate oxide into substrate. The EPROM transistor in an FPGA is used as a pull – down device for logic block inputs. This circuit scheme has one wire called “word line” which is connected to the select gate of the EPROM transistor. As long as the transistor has not been programmed into the OFF state, the word line can cause the "bit line", which is connected to a logic block input, to be pulled to logic zero. Since a pull-up resistor is present on the bit line, this scheme allows the EPROM transistor to not only implement connections but also to realize wired-AND logic functions. A disadvantage of this approach is that the resistor consumes static power.

The EEPROM approach is similar to the EPROM technology except that EEPROM transistor can be re-programmed in-circuit. The disadvantage of using EEPROM is that they consume twice the chip area as EPROM transistor. Also EEPROM requires multiple voltage sources.

c) Flash Memory

The flash memory is a type of non-volatile memory (NVM) and can retain information even when power supply is disconnected. The major advantages are like it can be erased and reprogrammed most of the time with no special voltages requirement; it is low cost and available with high density than EPROM.

A flash memory cell is like a transistor with an extra gate between the source and drain, and the control gate, there is a second gate called "floating gate" which can be used as a charge storage mechanism. When a sufficiently large voltage goes across the source and the control gate, electrons tunnel through the oxide layer and accumulate in the floating gate called "channel hot electron injection". This extra-negative charge in the floating gate reaches the threshold voltage writing a zero in the cell. To erase the cell, the control gate must be connected to ground, and the programming voltage must be applied to the source. It removes electrons from the floating gate and turns the cell back to a one.

d) Static RAM

Static RAM is the widely accepted technology used to build programmable logic. These cells are used to configure programmable wires and logic cells. In case of the pass-transistor approach, the SRAM cell controls whether the pass-gate is on or off. When off, the pass-gate provides a very high resistance between the two wires to which it is attached, and when on, it forms a low resistance connection between the two wires.

Figure 2.3 (b) illustrates a transmission gate formed by two pass-transistors (NMOS and PMOS), where an SRAM cell controls both NMOS and PMOS transistors. For the Multiplexer approach, the SRAM cells allow the MUX to select one routing wire and connect it to a Logic Cell. This scheme would typically be used to optionally connect one of several wires to a single input of a logic block.

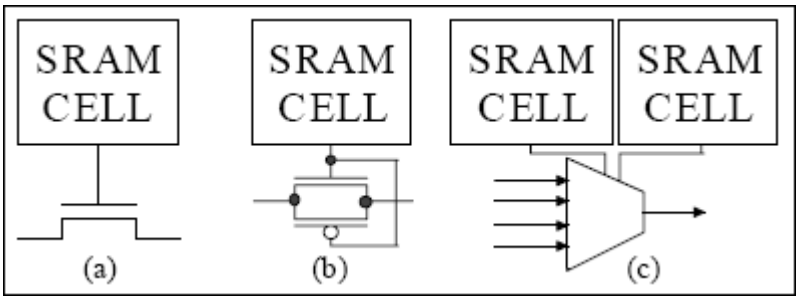


Figure 2.3: SRAM Cell with (a) Pass – Transistor, (b) Transmission Gate, (c) Multiplexer

Another memory cell configuration is shown in Figure 2.4, where, high-resistance polysilicon load replace the PMOS pull-up devices. The area of this cell could be about 40% smaller than the CMOS six-transistor memory cell. This cell is also called High Resistive Load (HRL) memory cell.

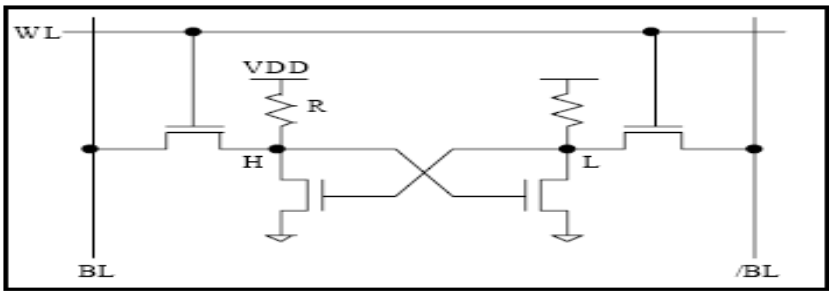


Figure 2.4: HRL SRAM Cell

The high-state storage node (H) can be pulled down with time due to two sources of leakage currents [15], the leakage currents flowing through the drain junction, and the subthreshold current.

A SRAM cell is re-programmable, unlike anti-fuse elements, which are physically altered when programmed, but are volatile in nature meaning is that the states of memory cells are lost when power is not applied. SRAM-based FPGAs must be programmed each time the circuit is powered up. Compared with other programming technologies the area required by SRAM is large, because of the number of transistors needed for each SRAM cell, as well as the additional transistor for the passgates or multiplexers.

The major advantage of this technology is that the FPGAs can be reconfigured very quickly and it can be produced using a standard CMOS process technology, which is advantageous for low power design compared to other technologies.

2.1.3.2 Logic Blocks Architecture

Logic blocks (or cells) have a great influence in the speed and area efficiency. There are a large number of possibilities for the design of a logic block. Here, some of the possibilities explored by the FPGA vendors are presented.

a) Look-Up Table based Logic Cell

Most recent FPGAs are based on Look-up Table (LUT) logic cells. A k-input LUT requires 2^k memory cells and a 2^k-input multiplexer to implement any Boolean function of k-inputs

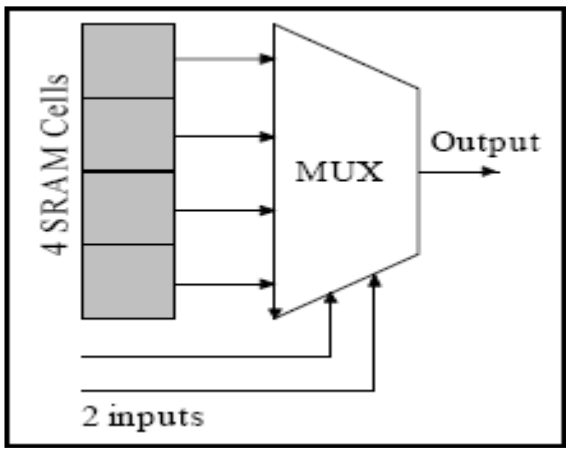


Figure 2.5: A 2- Input LUT

As mentioned in [18]. Figure 2.5 illustrates a 2-input LUT. In this case, 4 SRAM cells are required to store the truth table of the desired function. Since LUT-based logic cells are used, the FPGA research group from University of Toronto demonstrated that LUTs with 4 inputs lead to FPGAs with the greatest area-efficiency. The LUT-based cell developed by Altera™

in Flex™ and Apex™ devices consist, in general, of three elements: a 4-input LUT, a programmable DFF (D-type Flip Flop) and programmable resources that permits the logic element to implement cascade chain and carry chain operations. Xilinx™ devices contain LUT-based logic blocks. These blocks, called Configurable Logic Block (CLB) which are formed by two 4-input LUTs, one 3-input LUT that can be used as multiplexor, and two DFFs. This structure permits to build Boolean functions with 8 inputs. The CLB can be also configured as a 16 x 2 or a 32 x 1 memory cell.

b) Multiplexer-based

The Multiplexer-based logic block developed by Actel™ is presented in Figure 2.6 which contains three N-input Multiplexer controlled by a certain number of gates (AND, OR).

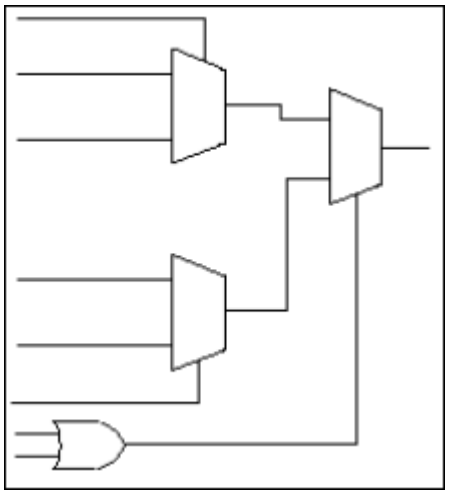


Figure 2.6: Multiplexer – Based Logic

These modules can implement any of several hundred functions of the inputs and any larger functions can be built by cascading several logic cells. The advantage of this structure is that it requires small area and hence, it allows the circuit area to be reduced.

c) Multiplexer and basic Gates or Symmetrical Cell

Figure 2.7 shows the logic block developed by Atmel™, which is similar to the Multiplexer-based one. It contains four multiplexers named X and Y (two input multiplexers and two output multiplexers), a DFF, and two 8-bit LUTs.

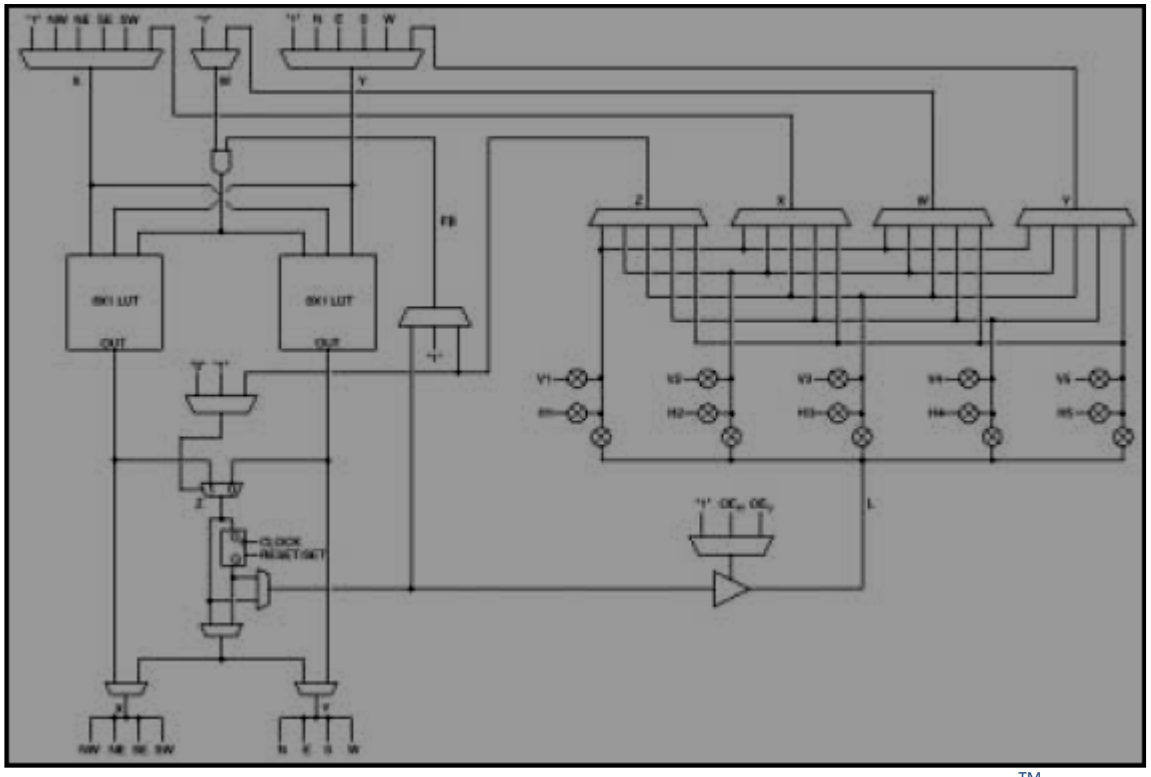


Figure 2.7: Multiplexer and Basic Gates LCELL Proposed by Atmel TM (Courtesy of Atmel^{TM} Co.)

This logic block has 8 inputs and 8 outputs, two pairs for each cardinal side and can be accessed by the four cardinal points; this represents an advantage for routing.

2.1.3.3 Interconnections

In general, FPGAs can be categorised in three groups based on their routing architecture: Row-based, segmented channel routing, and hierarchical routing.

a) Row-based FPGAs

A row-based FPGA proposed by Actel^{TM} consists of rows of logic blocks that are separated by horizontal routing channels, which are formed by wires of various lengths and separated by routing switches and adjacent wires can be connected to have longer segments if required as shown in Figure 2.8. Dedicated vertical segments are used to connect the inputs and outputs of logic blocks to the interconnect resources via the routing switches.

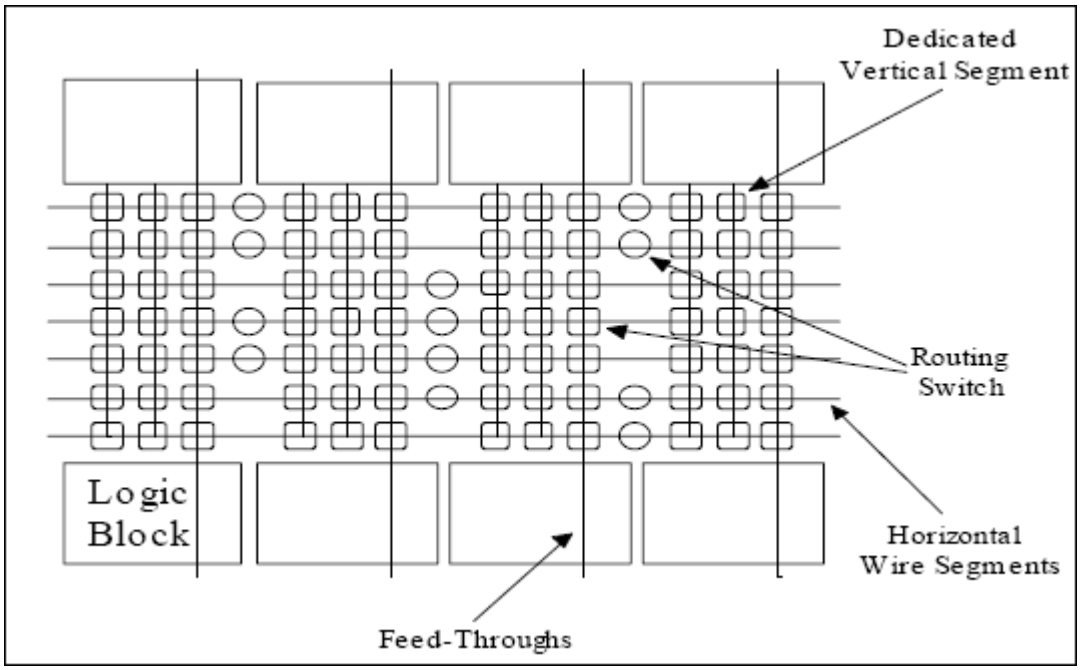


Figure 2.8: Actel™ ACT1 Interconnect Architecture (Row – Based)

There are also some vertical wires called "feed-through" used to connect one routing channel to another.

b) Segment-based FPGAs

Figure 2.9 shows a segment-based (also called island style) architecture developed in Xilinx™ devices in which the logic cells are surrounded by routing segments. Input or output pins of the LCELL can be connected to some or all of the wiring segments in the channel adjacent to it through a programmable connector switch.

These switches allow a segment wire to be connected to another to form longer segments or to connect a horizontal segment with a vertical segment and vice-versa. Long wires that traverse the entire device are dedicated to distribute some important signals like clock, reset, enable, etc.

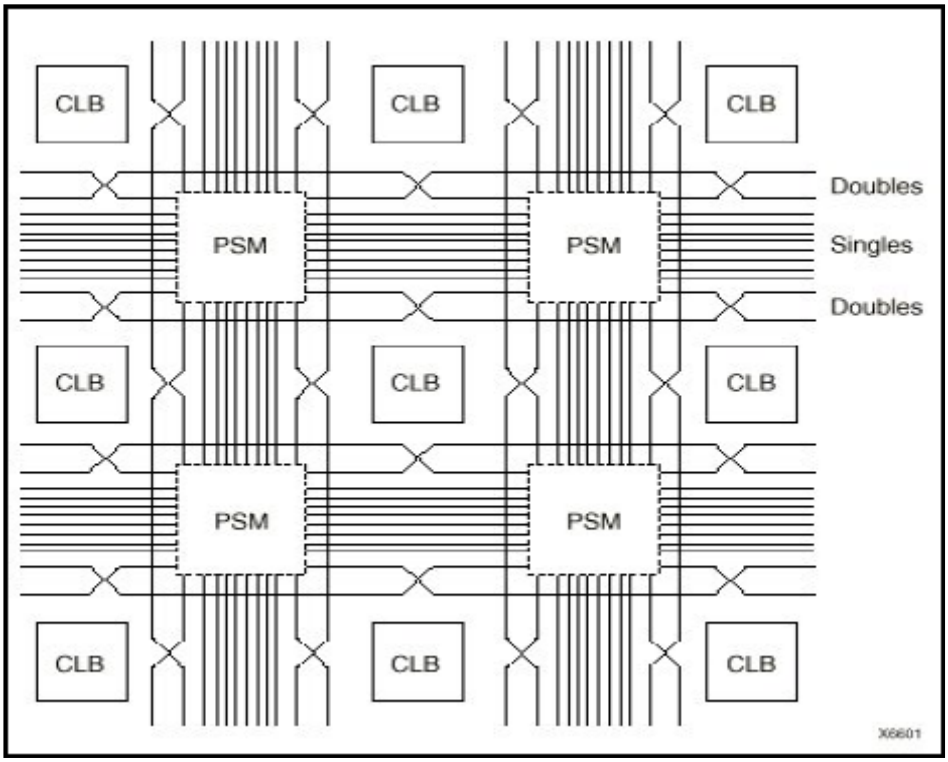


Figure 2.9: XC4000E/XL/XV Interconnect Architecture (Segmented – Based) (Courtesy of Xilinx™ Corporation)

The following figure shows the interconnect resources of the XC4000 series:

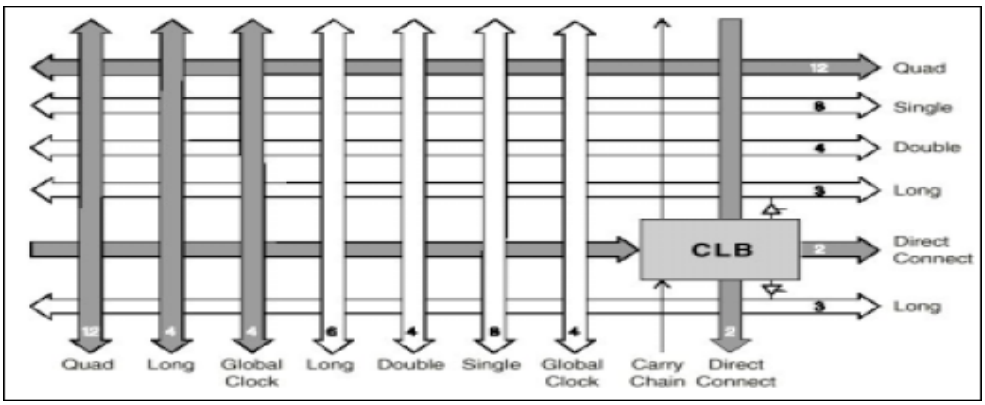


Figure 2.10: XC4000 Series Interconnect Resources (Courtesy of Xilinx™ Co)

c) Hierarchical Routing

Figure 2.11 shows the interconnect architecture developed by Altera™. Here, the FPGA is organized in long blocks called LABs (Logic Array Block) containing eight logic cells (LCELL). Each LCELL can communicate with the other LCELL of the same LAB by using local wires and can be connected to adjacent blocks by using direct connections.

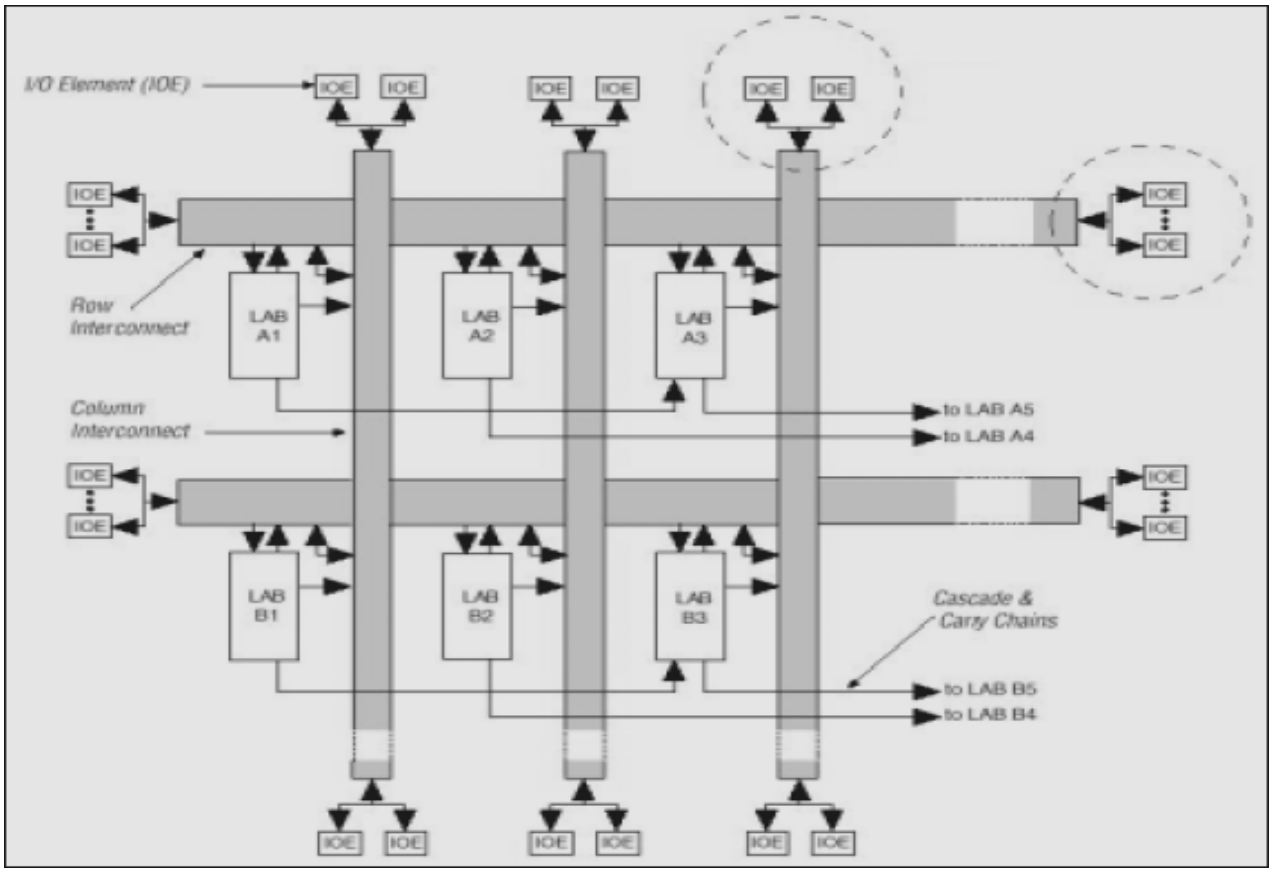


Figure 2.11: Hierarchical Interconnect (Courtesy of Altera™ Corporation)

LCELLS can reach other cells or I/O cells by lines, called "fast tracks" that traverse the entire device. This structure identifies three levels of interconnect resources: local (or LAB) interconnect, direct-paths (cascade and carry chain) and fast tracks.

2.1.3.4 I/O Structures

There could be different I/O block structures, but majority are of DFF-Multiplexer arrays with a slew rate control. The I/O blocks can be configured as Input, Output or bidirectional.
I/O units are directly connected to the routing resources of the device (Altera™). Some FPGAs have "dedicated inputs"; these inputs normally are used for Clocks, Reset, and Enable signals and are directly connected to the internal registers of device. The existing I/O elements use two DFF registers for Input and Output separately. This reduce control signals when use I/O as Bi-directional representing also an advantage for routing (Xilinx™).

2.1.3.5 Other Resources

Recent FPGAs contain other structures such as embedded memory cells, programmable Phase Lock-Loop (PLL) and, in some cases, thermal sensors. Embedded memory cells can be used to build large memory blocks or to implement logic operations. PLLs are used in FPGAs for generation of internal clock signals and also useful when implementing parallel architectures.

a) Embedded RAM Cells

Embedded memory cells can be placed inside the FPGA in big blocks and at the centre of the device such as Flex 10K devices, or in small blocks distributed in the device as used in the AT40K family with specific routing wires and a logic cell which are dedicated to optimize the memory access as in Figure 2.12. Some commercial FPGAs use logic blocks to build memory (i.e. Xilinx™ devices). In this kind of components, there are no embedded cells, and memory blocks are built at the cost of a reduction of the available number of logic blocks.

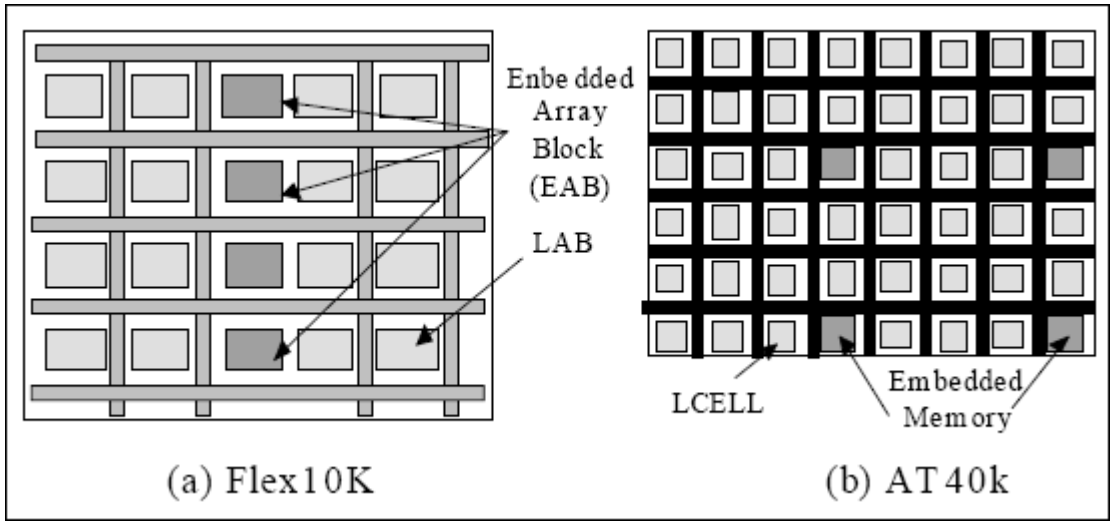


Figure 2.12: Embedded Memory (a) Block, (b) Distributed Cells

Embedded cells can be used to build SRAM, Dual-Port RAM, FIFO (First-in, First-out), LIFO (Last-in, First-Out), and other memory structures like CAM. They can be used to build big logic blocks such as state machines or long logic tables.

b) Phase Lock-loop

PLLs are normally used to generate internal clock signals from an external clock signal. They allow us to copy the global clock and change its phase. The clock phase can be adjusted by 90º increments for phase shifting of 90º, 180º, and 270º. PLLs are also used to generate two

or more internal clock signals with different frequencies by multiplying or dividing the clock frequency. The use of synchronous PLLs to generate the internal clock allows us to reduce the clock delay and skew within a device. This reduction minimizes clock-to-output and setup times while maintaining zero hold times. Internal PLLs can also be used to create an external clock signal to other devices.

2.2 Power Consumption Model of MOS-based Circuits

2.2.1 Introduction

Most of the models used to explain the power consumption behaviour of ICs are based on the equations derived from the analysis of the CMOS inverter. To understand its functionality and to introduce the equations and terms expressed in further sections, an overview of the CMOS inverter is discussed in [19].

2.2.1.1 The CMOS Inverter

The following Figure 2.13 shows the basic complementary CMOS inverter:

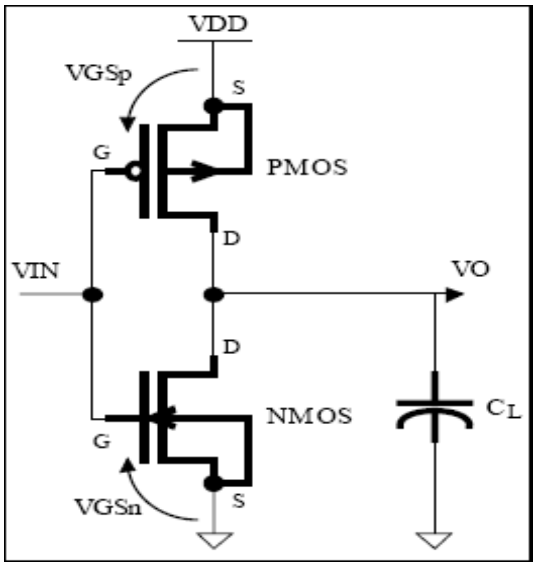


Figure 2.13: Standard CMOS Inverter

When $V_{IN}=V_{DD}$, $V_{GSn}=V_{IN}=V_{DD}$ and $V_{GSp}=V_{IN}-V_{DD}=0$. In this case, $V_{GSn}>V_{Tn}$, and $V_{GSp}|<|V_{Tp}|$. The NMOS is ON and the PMOS is OFF. The NMOS device provides a current path to GROUND (GND), and $V_O=0$. When the PMOS is OFF and the V_{DS} of NMOS device

is equal to zero. The DC current from V_{DD} to GND is controlled by the sub-threshold current of the PMOS device. If the V_{Tp} (extrapolated threshold voltage) is low enough, the sub-threshold current can be considered negligible; on the other hand, if V_{Tp} is high, the sub-threshold current is not negligible. In this case, the output voltage is not exactly at zero and can have values of tens of mV.

When V_{IN} is low (0 volts) $V_{GSn}<V_{Tn}$ and $|V_{GSp}|>|V_{Tp}|$, The PMOS device is ON and the NMOS transistor is OFF. The output voltage is $V_O=V_{DD}$. The Figure 2.14 shows the DC transfer characteristic of a CMOS inverter with the different regions of operation.

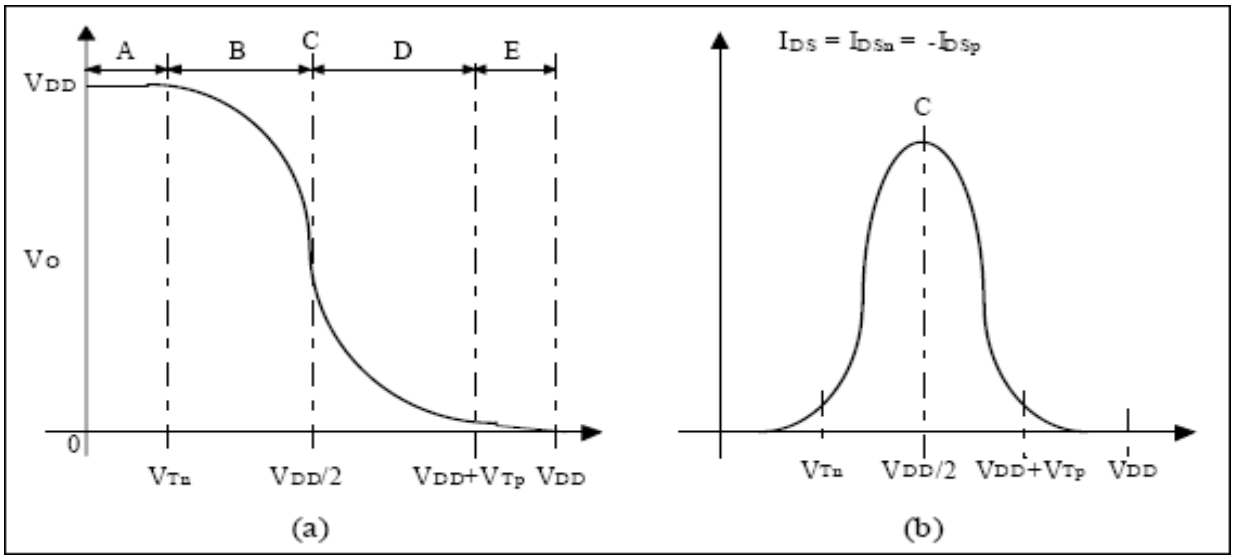


Figure 2.14: DC Transfer Characteristics of a CMOS Inverter, (a) Voltage and (b) Current

We can notice that the curve is divided into five regions of operation that can be described as follows:

For region A when $0 \leq V_{IN}<V_{Tn}$, the NMOS device is operating in the subthreshold region and the current is considered zero. The PMOS is in the linear region, and the current flowing through this device is also considered zero. Thus, $V_O=V_{DD}$. For region B When $V_{Tn}<V_{IN}<V_{INV}$ and V_{INV} is defined as the input voltage at which the gain of the inverter is maximum and is also defined as the gate threshold voltage. In this case, the NMOS device is operating in the saturation region and the PMOS is operating in the linear region. Since the current in both devices is the same, we have $I_{DSp}=I_{DSn}$. The current flowing through the PMOS device is given by [15]:

$$I_{DSp} = -\beta_p \left[(V_{IN} - V_{DD} - V_{Tn})(V_O - V_{DD}) - \frac{1}{2}(V_O - V_{DD})^2\right] \quad [2.2.1]$$

where $\beta_p = k_p \frac{W_{eff}}{L_{eff}}$ [2.2.2]

$$V_{GSp} = V_{IN} - V_{DD} \quad [2.2.3]$$

and $V_{DSp} = V_O - V_{DD}$ [2.2.4]

W_{eff} is the effective channel width, L_{eff} is the effective channel length, and K_p is a process-depend parameter that is defined as $K_p = \mu C_{OX}$, where μ is the mobility of electrons in the channel of the MOS transistor. C_{OX} is the gate oxide capacitance per unit area which is given by $C_{OX} = \frac{\varepsilon_0}{l_{OX}}$, where ε_0 is the oxide permittivity and ι_{ox} is the gate oxide thickness.

The saturation current of the NMOS device is:

$$I_{Dsn} = \beta_n \frac{(V_{IN} - V_{Tn})^2}{2} \quad [2.2.5]$$

Where $\beta_n = k_n \frac{W_{eff}}{L_{eff}}$ [2.2.6]

$$V_{GSp} = V_{IN} \quad [2.2.7]$$

Using equations [2.2.1,2.2.5], we can obtain an expression that represents the output voltage (Figure 2.14(a)):

$$V_O = (V_{IN} - V_{Tp}) + \sqrt{(V_{IN} - V_{Tp})^2 - 2\left(V_{IN} - \frac{V_{DD}}{2} - V_{Tp}\right)V_{DD} - \frac{\beta_n}{\beta_p}(V_{IN} - V_{Tn})^2} \quad [2.2.8]$$

In Region C, When $V_{IN} = V_{INV}$. In this case, both, NMOS and PMOS devices are in the saturation region. The PMOS current is given by:

$$I_{DSp} = -\beta_p \frac{(V_{IN} - V_{Tp})^2}{2} \quad [2.2.9]$$

The current flowing through the NMOS device is given by the equation [2.2.5]. By equalizing both equations [2.2.5, 2.2.9] we can obtain the expression that represents the V_{INV}:

$$V_{INV} = \frac{V_{DD} + V_{Tp} + V_{Tn}\sqrt{\beta}}{1 + \sqrt{\beta}} \quad [2.2.10]$$

Where $\beta = \frac{\beta_n}{\beta_p}$ and $V_{Tn} = V_{Tp}$. If we consider $\beta_n = \beta_p$ in a CMOS process defined by:

$$\frac{k_n}{k_p} = \frac{\mu_n}{\mu_p} \approx 2 - 3 \quad [2.2.11]$$

If we consider the following dimension ratio:

$$\left\{\frac{W_{eff}}{L_{eff}}\right\}p = 2.5 \left\{\frac{W_{eff}}{L_{eff}}\right\}n \quad [2.2.12]$$

We obtain $V_{IN} = V_{INV} = \frac{V_{DD}}{2}$ [2.2.13]

This kind of inverter is called "symmetrical gate". Nevertheless, the output voltage is not necessarily equal to $V_{DD}/2$ and is given by $V_{IN} - V_{Tn} < V_o < V_{IN} + V_{Tp}$.

In Region D, When $V_{INV} < V_{IN} < (V_{DD} + V_{Tp})$. In this case, the NMOS is in the linear region and the PMOS is in the saturation region. If we consider the same conditions from Region B, we can obtain:

$$V_o = (V_{IN} - V_{Tn}) - \sqrt{(V_{IN} - V_{Tn})^2 - \frac{\beta_p}{\beta_n}(V_{IN} - V_{DD} - V_{Tp})^2} \quad [2.2.14]$$

In Region E, When $(V_{DD} + V_{Tp}) < V_{IN} \leq V_{DD}$. In this case, the NMOS is ON and the PMOS is operating into the sub-threshold region. If we assume that the current flowing through this device is almost zero, then $V_O = 0$. From Figure 2.14(b), we can notice that when $V_{IN} = V_{INV}$, the DC power dissipation is maximal. It is also called short circuit power consumption.

2.2.2 Power Consumption of Complementary CMOS

The power consumed by the CMOS inverter, and by all CMOS circuits, can be divided into three components [20] [21] [22]:

1. The Static Power Consumption because of the leakage current I_{leak} and static current I_{ST} due to the supplied input voltage.
2. The Dynamic Power Consumption takes place due to the charge and discharge of the total output capacitance C_L.
3. The Dynamic Power Consumption caused due to the short-circuit current I_{SC} during the switching transient (also called short-circuit Power Consumption).

2.2.2.1 Static Power

There are two sources of static power in a complementary CMOS inverter: the leakage currents; and current drawn from the supply due to the input voltage. The total static power consumption can be expressed by the following equation:

$$P_{STAT} = P_{leak} + P_{dp} \quad [2.2.15]$$

The leakage currents are caused by the parasitic diodes in a CMOS inverter. When the input voltage is not changing, the parasitic diodes are not conducting. According to [19] the current in a diode is given by:

$$I_d = I_s \left(e^{\frac{qV_d}{nkT}} - 1\right) \quad [2.2.16]$$

Where n is the diode emission co-efficient (sometimes n=1) and V_d is the voltage applied to the diode. The total power consumption caused by the leakage currents is:

$$P_{leak} = \sum_i I_{di} V_{DD} \quad [2.2.17]$$

A typical value of I_d is 1 fA per device. In a pure CMOS circuit containing a million of devices, the total P_{leak} would be equals to 0.01 μW. The power dissipation due to the leakage currents could be neglected. Anyway in circuits containing memory cells, this power consumption could be more important and another component of the static power is a function of the input voltage. Assume that the input of the pull-down NMOS is at a voltage $0 \leq V_{IN} < V_T$. In this case, the current is given by the sub-threshold expression:

$$I_{dp} = I_o \frac{W_{eff}}{W_o} 10^{\frac{(V_{IN} - V_T)}{S}} \quad [2.2.18]$$

Where V_T is the constant-current threshold voltage, Io and Wo are the drain current and the gate width to define V_T, and S is the sub-threshold swing parameter. The current Io is related to V_{DS} by:

$$I_o = I_o' \left(1 - e^{\frac{V_{DS}}{V_t}}\right) \quad [2.2.19]$$

According to [23] the sub-threshold swing is given by:

$$S \approx 2.3\, V_t \left(1 + \frac{C_d}{C_{ox}}\right) \frac{V}{decade} \quad [2.2.20]$$

Where C_d is the depletion-layer capacitance of the source/drain junctions, S has a theoretical minimum limit of 60 mV/decade and When $V_{IN} > V_T$, the current can be expressed as follows:

$$I_{dp} = \frac{W}{LC_{ox}} (V_{IN} - V_T)^{1.5} \quad [2.2.21]$$

Where C_{OX} is the gate oxide capacitance, L and W are the average width and length of the device. The power dissipation caused by the direct-paths currents is:

$$P_{dp} = I_{Dmean} V_{DD} \quad [2.2.22]$$

For a CMOS circuit with more than a million of transistors, this source of static power consumption could be important. Static power consumption increases with temperature. Even if CMOS circuits have been designed to consume energy only during switching, in recent low-power applications with CMOS, the V_T is becoming low and the static power due to direct-pats current is becoming important.

2.2.2.2 Dynamic Power Caused by Load Capacitance

This source of power consumption is due to the currents needed to charge and discharge the effective load capacitance C_L of Figure 2.14. Let us assume a step input so neither the N and P devices are ON simultaneously. The average dynamic power P_d required to charge and discharges the C_L during a clock period T is:

$$P_d = \frac{1}{T} \int_0^T i_0 \ (t) V_0(t) dt \quad [2.2.23]$$

The output current needed to charge C_L is given by:

$$i_O = i_p = C_L \frac{dV_O}{dt} \quad [2.2.24]$$

And the current flowing through the NMOS during the discharge phase is:

$$i_O = i_n = -C_L \frac{dV_O}{dt} \quad [2.2.25]$$

The equation 2.2.23 becomes:

$$P_d = \frac{1}{T}\left[\int_0^{V_{DD}} C_L V_O dV_O - \int_{V_{DD}}^{0} C_L V_O dV_O\right] \quad [2.2.26]$$

The dynamic power dissipation can be expressed [15]as:

$$P_d = \frac{C_L V_{DD}^2}{T} = C_L V_{DD}^2 F \quad [2.2.27]$$

Where F is operation frequency and the dynamic power consumption is proportional to F and V_{DD}. If the supply voltage is reduced, power consumption will be reduced by a quadratic factor. This equation is only valid for the CMOS-Inverter, but it can be used to determine an equivalent expression for a any complex circuit.

2.2.2.3 Dynamic Power Caused by Short-Circuit Currents

Even if there are no load capacitance on the output and the parasitic capacitance are negligible, the CMOS-Inverter would still dissipate switching energy. If the Input voltage changes slowly, both the P and N devices are ON. An excess power is dissipated due to the short-circuit current. If we assume that the falling and rising times are equivalent, the power consumed by the short-circuit current is:

$$P_{SC} = I_{mean}\, V_{DD} \quad [2.2.28]$$

Where I_{mean} is estimated using the Figure 2.15.

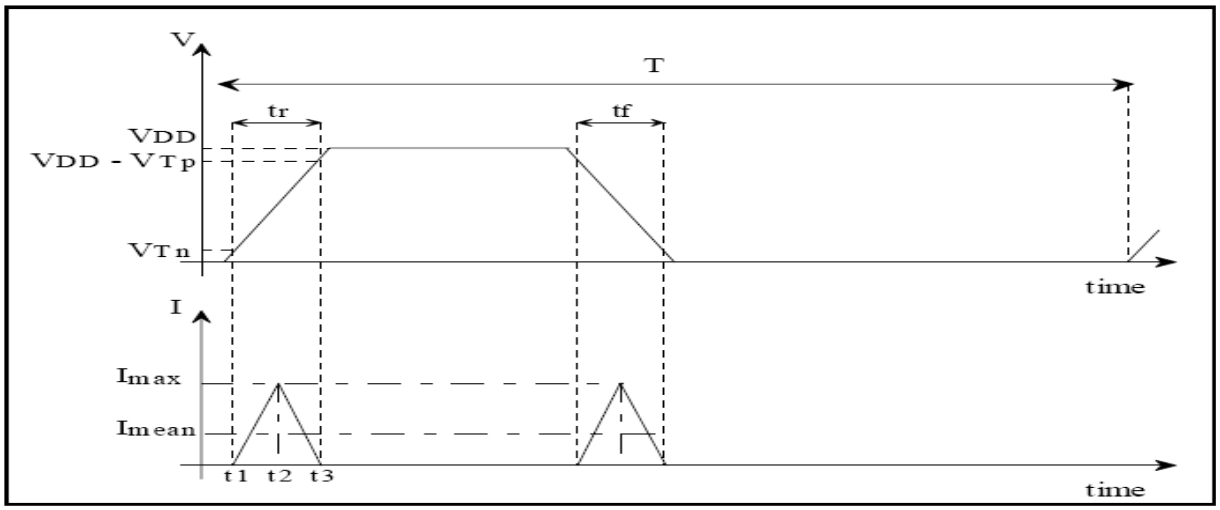


Figure 2.15: Input Voltage and Short – Circuit Current

Let us assume that $\beta_n = \beta_p = \beta$ And $V_{Tn} = V_{Tp} = V_T$ and if the rising time is equals to the falling time of the input signal ($\tau_r = \tau_f = \tau$). The mean short-circuit current is given by:

$$I_{mean} = 2X \frac{1}{T} \left[\int_{t_1}^{t_2} i(t)dt + \int_{t_2}^{t_3} i(t)dt\right] \quad [2.2.29]$$

Because property of symmetry, we have:

$$I_{mean} = \frac{4}{T} \left[\int_{t_1}^{t_2} i(t)dt\right] \quad [2.2.30]$$

Since the NMOS is operating in the saturation region, the above equation becomes:

$$I_{mean} = \frac{4}{T} \left[\int_{t_1}^{t_2} \frac{\beta}{2} (V_{IN}(t) - V_T)^2 \, dt\right] \quad [2.2.31]$$

The input voltage is:

$$V_{IN}(t) = \frac{V_{DD}}{\tau} t \quad [2.2.32]$$

It can be derived from figure 2.15 that $t_1 = \frac{V_T}{V_{DD}} \tau$ and$t_2 = \frac{\tau}{2}$. Then the integral leads to:

$$P_{SC} = \frac{\beta}{12} (V_{DD} - 2\,V_T)^3 \tau\, F \quad [2.2.33]$$

Eq. 2.2.33 describes that the short-circuit power consumption is proportional to the frequency. And for a system with equal rising and falling duration, the short-circuit power dissipation may be less than 20% of the total power consumption.

2.2.3 Power Consumption of SRAM

Different sources of power consumption can be identified using the SRAM architecture and the total power consumption can be divided in two components that are active and the standby power consumption. The active power is the sum of the power dissipated by the decoders and the memory array. If *m* is the number of memory cells connected to the same word line, the active power of the memory array in read mode can be expressed [15] as:

$$P_{mem-array} = mP_{act} + (n-1)m\,p_{leak} + mI_{DC}\Delta tFV_{DD} \quad [2.2.34]$$

Where P_{act} is the power dissipated in active mode when selecting the *m* cells and P_{leak} is the data retention (standby power) of the unselected memory cells in the *m* X *n* array. The third term is due to the DC current, I_{DC}, during the read operation. Δt is the activation time of the DC consuming parts and F is the operating frequency ($F = \frac{1}{t_{RC}}$). The standby power of an SRAM has a major contribution from the memory cells in the array if the sense amplifiers are disabled in this mode. It can be given by:

$$P_{standby} = mnP_{leak} \quad [2.2.35]$$

2.2.4 Power Consumption of Input / Output Circuits

The Input/Output (I/O) circuits allow the on-chip logic circuitry to communicate to the external world. The I/O circuits are important in the limitation of speed and power consumption of the entire circuit. There are several kinds of I/O circuits, such as input and output buffers, clock distribution, clock buffering and low-swing I/O. In this section, an overview of I/O circuits based on [24] is presented.

2.2.4.1 Input Circuits

In order to distribute an input signal to the whole circuit, an input buffer is needed which are formed by at least one inverter. In this section, the power consumption behaviour of a TTL to CMOS input circuit is discussed. The power consumed by this circuit is divided in two components: Static and dynamic.

a) Static Power Dissipation

The CMOS input buffer is used to translate the TTL (Transistor-Transistor Logic) or the Low-Voltage TTL levels to CMOS levels. The inverters that form the input buffer are designed by setting their W/L ratio such that the switching point of the buffer is near at middle of V_{IL} and V_{IH}. However, since the TTL voltage swing is limited to 1.2 volts, the

input buffer is always dissipating DC power. The circuit shown in Figure 2.16 is an example of a 2-stage input buffer.

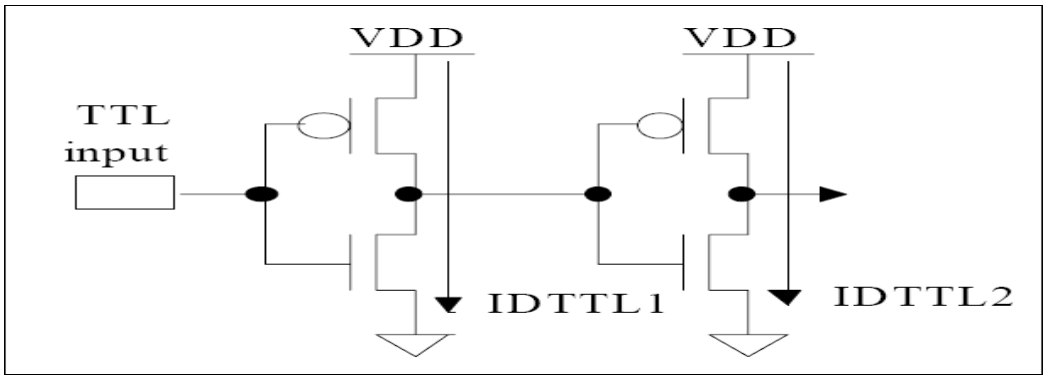


Figure 2.16: TTL Input Buffer

If the first inverter cannot be able to transfer the TTL level, the second one will dissipate some DC power. The static power consumption of this buffer is:

$$P_{TTL} = V_{DD} I_{DTTL} \quad [2.2.36]$$

Where, $$I_{DTTL} = I_{DTTL1} + I_{DTTL2} \quad [2.2.37]$$

I_{DTTL} is the average current flowing through both inverters when the input is at low and high levels. When the number of a TTL input pads is large, the DC current of the input buffers becomes important.

b) Dynamic Power Consumption

For a circuit that contains several I/O pads, the total dynamic power consumption of all input pads can be expressed as follows:

$$P_{inputs} = \alpha N_i E_{ii} F \quad [2.2.38]$$

Where α is the switching activity, N_i is the number if input pads, E_{ii} is the internal energy of the input pad in Watt/Hz and F is the operational frequency of the circuit.

2.2.4.2 Output circuits

The output buffer must be able to drive the load capacitance (fan-out) maintaining adequate rise and fall times. Usually, an inverter chain that can handle the large capacitance formed by

the pad, the package wiring and the off-chip load forms the output circuit. Figure 2.17 shows a tri-state output buffer.

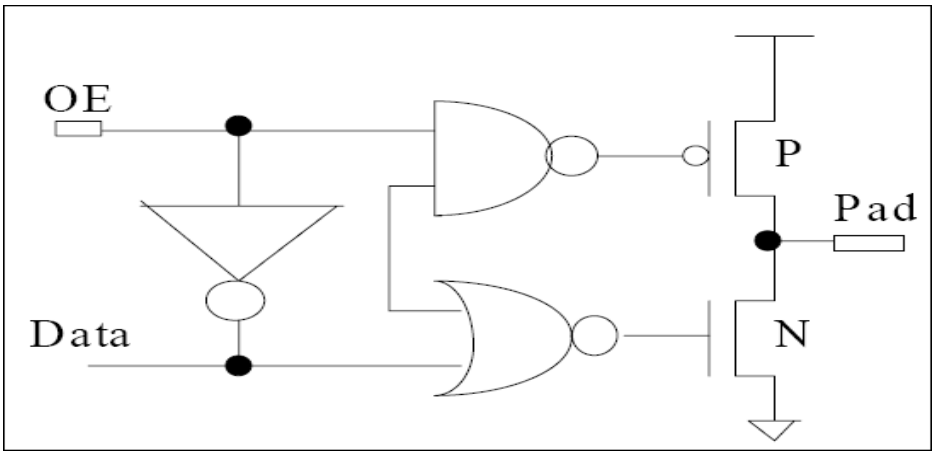


Figure 2.17: Tri – State Output Buffer

When the Output Enable (OE) is high, the output data is the same that the input data. When OE is low, the pad is set to high impedance (Z) and both (NMOS & PMOS) are cut-off.

a) Power Consumption of output circuits

The power consumed by the output pads can be divided into two components: static and dynamic. The static power consumption is caused by the junction leakage currents of the transistors that form the buffers and by the sub-threshold current from the input voltage. When V_T is small, the DC power dissipation becomes important due to the subthreshold currents. The static power consumption for the output pads when driving a CMOS TTL load is given by:

$$P_{static} = N_O V_{DD} (I_{DS\_mean} + I_{leak}) \quad [2.2.39]$$

Where N_O is the number of output pads, I_{DS_mean} is the average sub-threshold current when the input is low and high, and I_{leak} is the current caused by junctions. If the output pad has to drive a bipolar TTL charge, the output buffer is forced to sink significant amount of currents (due to the bipolar input transistor). The static power consumed by one output buffer driving a bipolar input pad is:

$$P_{Stat\_TTL} = V_{OL} I_{OL} \quad [2.2.40]$$

Where I_{OL} is the current sunk by the output buffer and is equals to the sum of the current from all the bipolar inputs. V_{OL} is the minimal output voltage when the output data is '0' (V_{OL} = 0.4 volts). The dynamic power consumed by the output circuits can be expressed [15] by the following equation:

$$P_{DYN} = \alpha\left(N_O E_{io} + N_O C_O {V_{DD}}^2\right)F \qquad [2.2.41]$$

Where E_{iO} is the internal switching energy of the output pad, C_O is the average output load capacitance, N_O is the number of output pads, F is the clock frequency and α is the average switching rate of all output pads.

2.2.5 Power Consumption in Clock Circuits

The current way to distribute the clock signal on-chip is using input buffers that have the ability to drive high internal load with fast fall/rise times. Consider a 3.3-volt micro-controller working at 200 MHz, with an internal load for the clock driver equals to 3.2 nF. In this case, the rise/fall times should be equal to 0.5 nS (T_{clock} = 5 nS); according to [25], the average transient current would be:

$$I_{AVE} = C\frac{\Delta V}{\Delta t} = \frac{3.2X10^{-9}X\ 3.3}{0.5X10^{-9}} = 21A \qquad [2.2.42]$$

And the dynamic power consumed only by this clock circuit is:

$$P_{dyn} = C{V_{DD}}^2 F \approx 7W \qquad [2.2.43]$$

This explains the importance of the clock circuit in terms of power. An architectural strategy should be used to distribute the clock signal to the whole circuit with minimum clock skew and low-power consumption. The clock distribution network can be planned in different ways to reduce the power dissipation of the clock buffer. The equivalent load capacitance of the clock buffer can be reduced by using low capacitance clock routing lines and the use of low-swing drivers at the top level of the clock tree can help in optimizing power consumption.

2.3 Power Consumption in SRAM-based FPGAs.

Recent FPGA architectures are formed by different types of technologies and elements. Logic Elements are formed by LUTs and DFFs. LUTs can be constructed using SRAM cells, and DFF is normally a CMOS device. Embedded Memory Cells must be constructed using SRAM cells. Finally, the interconnect resources must be programmed using SRAM cells that controls pass-transistors. Pass-transistors are used like switch to enable or disable all the internal elements of a FPGA. If we consider all these elements, the SRAM-based FPGAs are formed by three different technologies: SRAM, pure CMOS and Pass-Transistors. It means that the power consumption modelling in FPGAs becomes more complex than pure CMOS [26]. Power in FPGAs is not only a function of ${V_{DD}}^2$. Since they contain a lot of pass-transistors, short-circuit currents are not negligible in these devices. According to equation 2.2.33, power consumption must be also a function of${V_{DD}}^3$. Finally, the direct-paths current in pass-transistor structures becomes important, this factor added to the static current dissipated for all the SRAM cells used in a FPGA, makes that the static power consumed by a FPGA must be considered and it is not negligible. The Total power consumption of a FPGA can be represented using the following equation:

$$P_{FPGA} = \alpha {V_{DD}}^3 + \beta {V_{DD}}^2 + \delta V_{DD} \qquad [2.2.44]$$

Where α is the element that corresponds to the dynamic power consumption caused by short-circuit currents (V_{DD} - $2V_T$), and by a portion of the static power caused by direct-path currents and β corresponds to the dynamic power caused by the charge and discharge of the load capacitance and δ corresponds to the static power consumption.

2.4 Conclusion

In the first section of this chapter, we have described the different internal architectures of FPGAs in a generic way as any commercial architecture was used to explain the complexity of FPGA architectures. Initially we identified the internal elements that form all FPGA architectures:

- Logic Blocks which are formed by 4-input and 3-input Look-Up Tables with a programmable D-type flip-flop.
- Input / Output cells which can be programmed as input, output or bi-directional. Each I/O cell contains at least one programmable D-type flip-flop.

- Interconnect resources, formed by different types of wires with different lengths, and by programmable interconnect switches.
- The embedded memory cells which are used to build RAM, ROM, DPRAM, FIFO or LIFO blocks.

The next Section, is an overview of the power consumption in MOS-based circuitry. First of all, we have explained the CMOS-inverter, since this component is the base for most of the CMOS power consumption model.

Based on all theoretical elements, and the specific FPGA architectures formed by CMOS logic and pass-transistors, the power consumption behaviour in FPGAs is different to other circuits (like Micro-Controllers, DSPs or ASICs). Equation [2.2.44] summarizes all the theoretical aspects presented in this chapter.

In the following chapter we will present the state-of-the-art literature survey on different existing power management techniques which can be applied at the various abstraction levels of the system especially for the design of power optimized microprocessor and/or microcomputer.

Chapter 3
Power Reduction Techniques for Embedded Systems

3.1 Introduction

Minimization of power consumption in portable and battery – operated embedded systems has become an important aspect of the embedded system designing and power efficient design requires reducing power dissipation in all the parts of the design also during all stages of the design process without compromising the system performance and the quality of services. The opportunities for power optimization are available across the entire design hierarchy. Many techniques can be applied at various levels ranging from circuits to architectures, architectures to system software and system software to applications [27]. We believe that power management is many-sided discipline that is continually expanding with new techniques being developed at every level.

The increasing usage of portable electronics devices has become a driving force in the design of new computational elements in very large-scale integration (VLSI) systems on a chip. As the recent focus is on mobile appliance, a rethinking of design optimizations targeting increasing performance and high clock rates at any cost are required in order to optimize battery life and extend the utility of these devices. The trend in the embedded system of continuous growth in complexity and size in terms of micro-architecture needs to be re-examined, as the tradeoffs in energy consumption versus the improved performance obtained by different set of design choices. Power consumption arises as a third axis in the optimization space in addition to the traditional speed (performance) and area (cost) dimensions. Improvements in circuit density and the corresponding increase in heat generation must be addressed even for high-end systems and in recent era the CMOS circuits cannot be reliably sustained without considering power consumption issues. Environmental

concerns relating to energy consumption by computers and other electrical equipment are another reason for interest in low-power designs and design techniques.

Low-power design can be an important to reduce the system cost. Smaller packages, batteries, and reduced thermal management overhead result in less costly products, with higher reliability as an added benefit. Size, power budget, and weight of a device are important metrics, and the power source is the main factor of these metrics. In Power efficient design, the system minimizes the peak demands on the source and improves its operating efficiency. The rate of energy use can have a dramatic effect on the amount of energy available from a battery source as well as its cost [28] [29], which minimize average power consumption and peak power consumption as well.

Hence, opportunities for design tradeoffs emphasizing low power are available across the entire spectrum of design process for a portable system, and are effectively applied at many levels of the design hierarchy, from algorithm selection to silicon process technology. Generally, it is observed that power saving possibilities is lies with the higher the level of abstraction.

This chapter includes the study for different power aware design methodologies and techniques in broad range discussed in [30], which are targeted for an embedded systems development and can be apply at various abstraction levels. This dissertation work is carried out by considering dynamic power reduction techniques which are applied at the system level [31], architecture level and logic level without making any change at the circuit/device structure level.

3.2 Defining Power Dissipation in CMOS Circuits

Here, Power dissipated in CMOS circuits consists of several generic components [32] as indicated in equation (3.2.1) discussed to focus the power saving techniques.

$$P_{total} = P_{switching} + P_{shortcircuit} + P_{static} + P_{leakage} \quad [3.2.1]$$

All the components mentioned in Eq. 3.2.1 represent the power required to charge a capacitive load ($P_{switching}$), short circuit power consumed for CMOS gate as the input

switches ($P_{shortcircuit}$), static power consumed (P_{static}), and leakage power consumed is ($P_{leakage}$). $P_{switching}$ and $P_{shortcircuit}$ are active when a device is actively changing state, while P_{static} and $P_{leakage}$ are available regardless of state changes. The biggest contribution is due to $P_{switching}$, which is defined as

$$P_{switching} = C * V_{dd} * V_{swing} * \alpha * f \quad [3.2.2]$$

Where C is the capacitance, V_{dd} is the supply voltage, V_{swing} is related to the change in voltage level of the switching capacitance, α is switching activity factor depends on the probability of an output transition, and f is the frequency of operation. The product $\alpha * C$ is the effective switched capacitance, or C_{eff}. In most designs, V_{swing} is equal toV_{dd}, so (Eq. 3.2.2) can be

$$P_{switching} = C_{eff} * {V_{dd}}^2 * f \quad [3.2.3]$$

The term $P_{switching}$ occurs due to the overlapped conductance of both the PMOS and NMOS transistors forming a CMOS logic gate as the input signal transitions. This term has a complicated derivation, but in simplified form can be written as [6],

$$P_{short\,circuit} = I_{mean} * V_{dd} \quad [3.2.4]$$

where I_{mean} is the average current. I_{mean} is lowered for a single gate with short input rise/ fall times, and with long output transition times, and shows tradeoffs in device sizing. P_{static} related to pure CMOS designs, as it is not drawn by a CMOS gate, but few circuit architectures like voltage references, constant current sources, sense amplifiers, etc.. which exists in CMOS systems and add to overall power. $P_{leakage}$ is due to leakage currents from reversed biased PN junctions and subthreshold conduction currents and it is proportional to area and temperature. The subthreshold leakage component is dependent on device threshold voltages, and also important as supply voltage scaling is used to lower power. For systems having high ratio of standby operation to active operation, $P_{leakage}$ may be the principal factor in affecting overall battery life.

Optimization of all above discussed components in designing low-power systems and active power minimization involves reduction of the magnitude of each of the components in (Eq.

3.2.3). With its quadratic contribution in the power equation, reduction of supply voltage is an understandable method for power reduction, and can be applied to an entire implementation. Reducing supply voltage by a factor of two results in a factor of four reductions in $P_{switching}$. There are limitations to supply voltage scaling, however, since the performance of a gate is reduced as V_{dd} is lowered, due to the lowered saturation current available to charge and discharge load capacitance. Gate delay dependence on V_{dd} is approximated [19] by

$$T_{delay} \propto \frac{C_{load} * V_{dd}}{(V_{dd} - V_{threshold})^{1.5}} \quad [3.2.5]$$

The energy-delay product is minimized when V_{dd} is equal to$2 * V_{threshold}$. Reducing V_{dd} from $3* V_{threshold}$ (a typical value for 0.18 m technology) to $2 * V_{threshold}$ results in an approximate 50% decrease in performance while using only 44% of the power. It would seem that reducing threshold voltage of the devices and, thus, a corresponding reduction in V_{dd} offers low-power consumption. But, there are practical limits to the degree that $V_{threshold}$ can be lowered, due to reduced noise margins and since exponentially increased leakage current becomes a limiting factor in contribution to $P_{leakage}$ [33]. Controllability of variations in $V_{threshold}$ is also an issue in manufacturing, and provides a lower bound on supply voltage scaling [34] [35].

3.3 Power Reduction Methodologies for Various Abstraction Levels

Power reduction techniques may be applied at all the levels of the system design hierarchy, which include Algorithmic, Architectural, Logic and Circuit and Device Technology. A brief description of each is given in the following section with some suitable example, wherever it is needed.

3.3.1 Logic and Circuit Level Power Reduction Techniques

3.3.1.1 Transistor Sizing

Transistor sizing reduces the width of transistors to reduce their dynamic power consumption, but reducing the width also increases the transistor's delay; hence the transistors that lie away from the critical paths of a circuit are usually the best suited for this technique. Algorithms for applying this technique usually associate with each transistor a tolerable delay, which tries to scale each transistor to be as small as possible without violating its tolerable delay.

3.3.1.2 Transistor Reordering

The arrangement of transistors in a circuit affects energy consumption. Figure 3.1 shows two possible implementations of the same circuit that differ only in their placement of the transistors marked A and B. Suppose that the input to transistor A is 1, for B is 1 and for C is 0. Then transistors A and B will be on, allowing current from V_{dd} to flow through them and charge the capacitors C1 and C2.

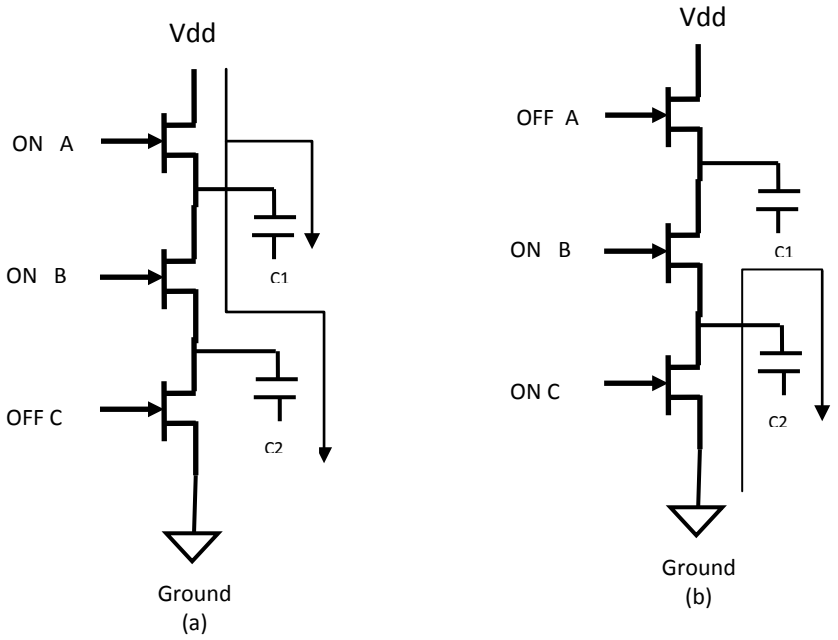


Figure 3.1: Transistor Reordering

Now, suppose the inputs change and that A's input becomes 0, and for C it is 1. Then A will be off while B and C will be on. Now the implementations in (a) and (b) will differ in the amounts of switching activity. In (a), current from ground will flow through B and C, discharging both the capacitors C1 and C2. However, in (b), the current from ground will only flow through C2 it will not pass through A since A is turned off. Thus it will only discharge the capacitor C2, rather than both C1 and C2 as in part (a). Thus the implementation in (b) will consume less power than that in (a). Transistor reordering [36] rearranges transistors to minimize their switching activity.

3.3.1.3 Half Frequency and Half Swing Clocks

Half-frequency and half-swing clocks reduce frequency and voltage, respectively. Traditionally, hardware events such as register file writes occur on a rising clock edge. Half-frequency clocks synchronize events using both edges, and tick at half the speed of regular clocks, thus cutting clock switching power in half. Reduced-swing clocks use a lower voltage signal and thus reduce power quadratically.

3.3.1.4 Logic Gates Restructuring

There are many ways to connect a circuit using logic gates but the way the gates and their signals are connected affects power consumption. Consider two implementations of a four-input AND gate shown in Figure 3.2 with signal probabilities (1 or 0) at each of the primary inputs (A,B,C,D) with the transition probabilities (0→1) for each output (W, X, F, Y, Z). If each input has an equal probability of being a 1 or a 0, then the calculation shows that implementation (a) is likely to switch less than the implementation (b). This is because each gate in (a) has a lower probability of having a 0→1transition . In (b) some gates may share a parent (in the tree topology) instead of being directly connected together. These gates could have the same transition probabilities. The circuit (a) do not necessarily save more energy than Circuit (b). There may be many other issues such as glitches or spurious transitions which occur when a gate does not receive all of its inputs at the same time.

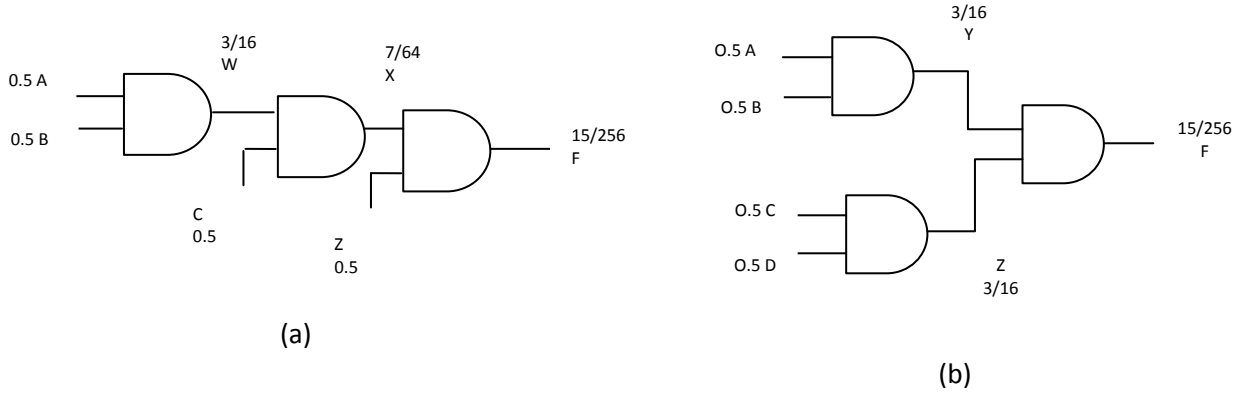


Figure 3.2: Gate restructuring (Figure adapted from the Pennsylvania State University Microsystems Design Laboratory's tutorial on Low Power Design)

These glitches are more common in (a) where signals can travel along different paths having widely varying delays. There are many solutions such as path balancing, retiming etc.

3.3.1.5 Technology Mapping

Technology mapping [37] [38] is the process which constructs a gate-level representation of a circuit with the constraints such as area, delay, and power and for power it depends on gate-level power models and a library which makes the gates available along with their design constraints. Initially circuit is described at logic level before it is presented in the form of gates. The problem is to design the circuit out of logic gates in a way so that it will reduce the total power consumption under delay and cost limitations. The remedy is to break the circuit into a set of trees and find the optimal mapping for each sub-tree using standard algorithms.

3.3.1.6 Low Power Flip-Flops

Flip-flops are the basic building blocks of small memories such as register files. A standard master-slave flip-flop requires two latches with clock and data as inputs to the master latch and for the slave latch that are the inverse of the clock signal and the output of master latch. Hence, when the clock is high, the master latch is turned ON, and the slave latch is turned OFF. In this phase, the master samples whatever inputs it receives and outputs them. The slave does not sample its inputs but outputs whatever it has most recently stored. On the falling edge of the clock, the master turns OFF and the slave turns ON. Thus the master saves its most recent input and stops sampling further inputs. The slave samples the new inputs it receives from the master and outputs it. The other options are the pulse-triggered flip-flop and sense-amplifier flip-flop. All these designs share some common sources of power consumption, such as power dissipated from the clock signal, internal switching activity and when the outputs transition occurs.

Researchers have proposed several alternative low power designs for flip-flops. Most of these approaches reduce the switching activity or the power dissipated by the clock signal as in case of the self-gating flip-flop. This design inhibits the clock signal to the flip-flop when the inputs will produce no change in the outputs proposed by [39].

Another low-power flip-flop is the conditional capture flip-flop. This flip-flop detects when its inputs produce no change in the output and stops these redundant inputs from causing any spurious internal current switches. There are many variations in this concept, such as the conditional discharge flip-flops and conditional pre-charge flip-flops [40] which eliminate unnecessary pre-charging and discharging of internal elements.

3.3.1.7 Low – Power Control Logic Design

The Finite State Machine (FSM) can be viewed as the control logic of a processor, which specifies the possible processor states and conditions for switching between the states and generates the signals that activate the appropriate circuitry for each state. Usually, control logic optimizations targeted performance, but in recent era also targeting power. One way of reducing power is to encode the FSM states in a way that minimizes the switching activity throughout the processor. Another approach is decomposing the FSM into sub-FSMs and activating only the circuitry needed for the currently executing sub-FSM.

3.3.1.8 Delay-Based Dynamic Supply Voltage Adjustment

Many processors run at multiple clock speed uses a lookup table built for worst-case analysis to decide what supply voltage to select for a given clock speed. But ARM Inc. has been developed a more efficient runtime solution known as the Razor pipeline [41], in which instead of using a lookup table, Razor adjusts the supply voltage based on the delay in the circuit. The purpose is that whenever the voltage is reduced, the circuit slows down, causing timing errors if the clock frequency chosen for that voltage is too high. Because of these errors, some instructions produce inconsistent results or fail altogether. The Razor pipeline periodically monitors how many such errors occur. If the number of errors exceeds a threshold, it increases the supply voltage. If the number of errors is below a threshold, it scales the voltage further to save more energy [42].

This solution requires extra hardware to detect and correct circuit errors. To detect errors, it adds the flip-flops in delay-critical regions of the circuit with shadow-latches which receive the same inputs as the flip-flops but are clocked more slowly to adapt to the reduced supply voltage. If the output of the flip-flop differs from that of its shadow-latch, then this signifies an error. In the event of such an error, the circuit propagates the output of the shadow-latch instead of the flip-flop, delaying the pipeline for a cycle if necessary.

3.3.2 Low – Power Techniques for Interconnect

Interconnect heavily affects power consumption as it is the medium of most electrical activity. Efforts to improve chip performance are resulting in smaller chips with more transistors and more densely packed wires carrying larger currents. The wires in a chip often use materials with poor thermal conductivity [43]. Many techniques are available to reduce the switching and few of them are surveyed and discussed below.

3.3.2.1 Bus Encoding and Cross Talk

Reducing the power consumed in buses the effective way is to reduce the switching activity through some bus encoding schemes, such as bus-inversion [44]. Buses consist of wires to transmit bits. For every data transmission on a bus, the number of wires that switch depends on the current and previous values transmitted. If the Hamming distance between these values is more than half the number of wires, then most of the wires on the bus will switch current. To prevent this from happening, bus-inversion transmits the inverse of the intended value and asserts a control signal alerting recipients of the inversion. It also ensures that at most half of the bus wires switch during a bus transaction. However, because of the cost of the logic required to invert the bus lines, this technique is mainly used in external buses rather than the internal chip interconnect. It is also possible that as chips become smaller, there arise additional sources of power consumption and one of these sources is crosstalk, which is spurious activity on a wire that is caused by activities in neighbouring wires. Also because of increasing delays and impairing circuit integrity, crosstalk can increase power consumption.
One of the ways to reduce crosstalk is to insert a shield wire between adjacent bus wires. Since the shield remains disserted, no adjacent wires switch in opposite directions, but it doubles the number of wires. However, the vision behind it, is to develop self-shielding codes [45], which are resistive to crosstalk. As in traditional bus encoding, a value is encoded and then transmitted, but the code chosen avoids opposing transitions on adjacent bus wires.

3.3.2.2 Low Swing Buses

Bus-encoding schemes used to reduce transitions on the bus. Alternatively, a bus can transmit the same information but at a lower voltage. This is the principle behind low swing buses. Traditionally, logical one is represented by +5 volts and logical zero is represented by 5 volts. In a low-swing system shown in Figure 3.3, logical one and zero are encoded using lower voltages, such as +300mV and 300mV. Typically, these systems are implemented

with differential signalling. An input signal is split into two signals of opposite polarity bounded by a smaller voltage range. The receiver sees the difference between the two transmitted signals as the actual signal and amplifies it back to normal voltage.

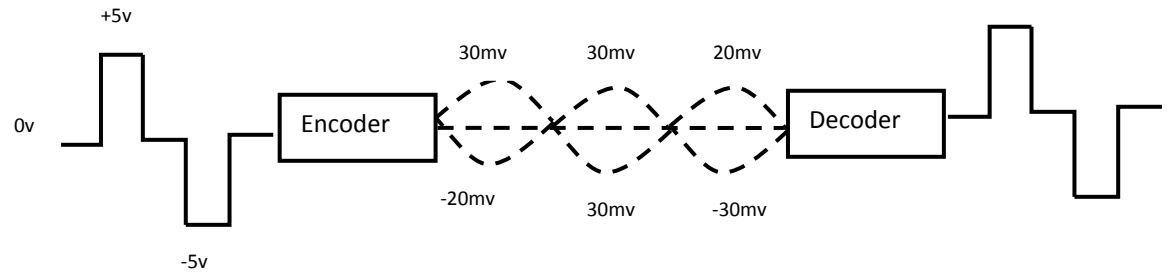


Figure 3.3: Low Voltage Differential Signalling

Low Swing Differential Signalling has several advantages in addition to reduced power consumption. It is immune to crosstalk and electromagnetic radiation effects. Since the two transmitted signals are close together, any spurious activity will affect both equally without affecting the difference between them. For implementation of this technique, one needs to consider the costs of increased hardware at the encoder and decoder.

3.3.2.3 Bus Segmentation

Bus segmentation is another strategy, in shared bus architecture; the entire bus is charged and discharged upon every access. Figure 3.4 shows that it splits a bus into multiple segments connected by links that regulate the traffic between adjacent segments and links connecting paths essential to a communication are activated, with most of the bus to remain powered down.

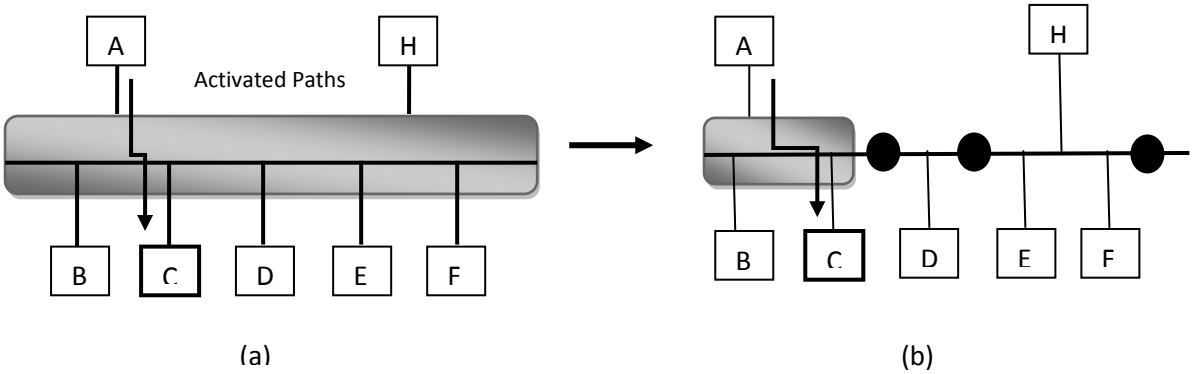


Figure 3.4: Bus Segmentation

Ideally, devices communicating frequently should be in the same or nearby segments to avoid powering many links. The algorithm discussed in [46] begins with an undirected graph whose

nodes are devices, edges connect communicating devices, and edge weights display communication frequency.

3.3.2.4 Adiabatic Buses

The techniques discussed above are targeting the reduction of switching activity or voltage, whereas, adiabatic circuits reduces total capacitance. These circuits reuse existing electrical charge to avoid creating new charge. Generally, when a wire becomes disserted, its previous charge is wasted. A charge-recovery bus recycles the charge for wires about to be asserted.

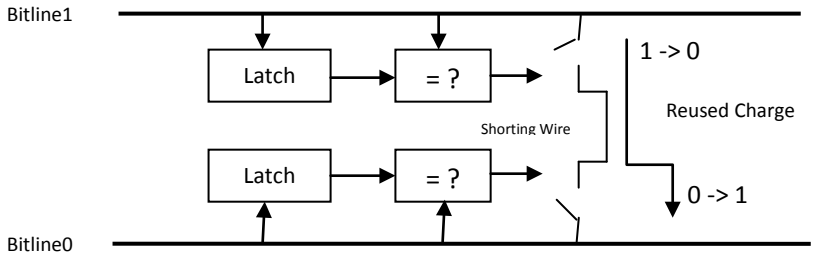


Figure 3.5: Two Bit Charge Recovery Bus

Figure 3.5 is a design for a two-bit adiabatic bus [47]. A comparator in each bit-line tests whether the current bit is equal to the previously sent bit. Inequality signifies a transition and connects the bit-line to a shorting wire used for sharing charge across bit-lines. For eg. Bit-line1 has a falling transition while Bit-line0 has a rising transition. As Bit-line1 falls, its positive charge transfers to Bit-line0, allowing Bit-line0 to rise. The power saving depends on transition patterns. No energy is saved when all lines rise. The most energy is saved when an equal number of lines rise and fall simultaneously. The biggest drawback of adiabatic circuits is a delay for transferring shared charge.

3.3.2.5 Network-On-Chip

In all above techniques multiple functional units share one or more buses having many drawbacks related to power and performance. The inherent bus bandwidth limits the speed and volume of data transfers which is not suitable for varying requirements of different execution units. Only one unit can access the bus at a time, even though many other may be requesting simultaneously. Unlike simple data transfers, every bus transaction involves multiple clock cycles of handshaking signals that increase power requirement and introduce delay.

The hardware-level optimizations for different sources of power consumption are discussed in [48] for an on-chip network. It has proposed a segmented crossbar network topology, in order to reduce the power consumption of data transfers. When data is transferred over a regular network, all rows and columns corresponding to the intersection points; end up switching current even though only parts of these rows and columns are actually traversed. To eliminate this switching, the said topology divides the rows and columns into segments using tri-state buffers. The different segments are selectively activated as the data traverses through them, hence confining current switches to only those segments along which the data actually passes.

3.3.3 Low Power Techniques for Memories and Memory Hierarchies

For computation purpose the storage is the essential, also the memory size optimization is the earlier focus, but with the technological advancement the cost per bit is too low; hence memory size is not an issue but its performance and power needs are the constraints in system design. Some strategies are discussed below to provide the solutions to these problems.

Memory can be classified into two categories, Random Access Memories (RAM) and Read-Only- Memories (ROM). There are two kinds of RAMs, static RAMs (SRAM) and dynamic RAMs (DRAM) which differ in a way they store data. SRAMs store data using flip-flops and DRAMs store each bit of data as a charge on a capacitor; thus DRAMs need to refresh their data periodically. SRAMs allow faster accesses than DRAMs but require more area and are more expensive. As a result, normally only register files, caches, and high bandwidth parts of the system are made up of SRAM cells, while main memory is made up of DRAM cells. Although these cells have slower access times than SRAMs, they contain fewer transistors and are less expensive than SRAM cells. The power reducing techniques are not confined to any specific type of memory. Rather they are high-level architectural principles that apply across the spectrum of memories to the extent that the required technology is available. They attempt to reduce the energy dissipation of memory accesses in two ways, either by reducing the energy dissipated in a memory accesses, or by reducing the number of memory accesses.

3.3.3.1 Splitting Memories into Smaller Sub-systems

An effective way to reduce the energy consumed in memory access is to activate only the needed memory circuits in each access, meaning is to partition memories into smaller, independently accessible components. This can be done as different granularities, at lowest granularity, one can design a main memory system that is split up into multiple banks each of which can independently transit into different power modes [30] [49] and at higher granularity, it is possible to partition the separate banks of a partitioned memory into sub-banks and activate only the relevant sub-bank in every memory access. In a normal cache access, the set-selection step involves transferring all blocks in all banks onto tag-comparison latches. Since the requested word is at a known offset in these blocks, energy can be saved by transferring only words at that offset. Cache sub-banking splits the block array of each cache line into multiple banks. During set-selection, only the relevant bank from each line is active. Since a single sub-bank is active at a time, all sub-banks of a line share the same output latches and sense amplifiers to reduce hardware complexity. Sub-banking saves energy without increasing memory access time. In addition to that, it is independent of program locality patterns.

3.3.3.2 Augmenting the Memory Hierarchy with Specialized Cache Structures

The other way to lower energy consumption in the memory is to reduce the number of memory hierarchy accesses. The paper [50] discussed a simple but effective technique for this and they integrate a specialized cache into the memory hierarchy of a today's processor, this hierarchy has one or more levels of caches. They proposed a catch deployed between the processor and first level cache and smaller in size than the first level cache and hence dissipate less energy. But when a data request misses this cache would it require searching higher levels of cache, and the penalty for a miss would be offset by redirecting more memory references to this smaller, more energy efficient cache. This was the principle behind the filter cache. Even though this is a simple idea, it is used by many of the low-power cache designs today, ranging from simple cache hierarchy to the scratch pad memories used in embedded systems, and the complex trace caches used in high-end processors. Trace caches were developed for performance but have been studied recently for their power benefits. Instead of storing instructions in their compiled order, a trace cache stores traces of instructions in their executed order. If an instruction sequence is already in the trace cache, then it need not be fetched from the instruction cache but can be decoded directly from the

trace cache. This saves power by lowering the number of instruction cache accesses. Low-power designs for trace caches intend to lower the number of instruction cache accesses. One alternative is the dynamic direction prediction-based trace cache discussed in [51], uses branch prediction to decide where to fetch instructions from. If the branch predictor predicts the next trace with high confidence and that trace is in the trace cache, then the instructions are fetched from the trace cache rather than the instruction cache.

3.3.4 Power Reduction at Architecture Level

Formation of instruction set, structure of pipelining and Parallelism have great impact in lowering the power consumption at the architectural level. Architecture-driven voltage scaling technique for power reduction is discussed in [52]. It lowers the voltage to reduce power consumption, and then to apply parallelism and/or pipelining to maintain throughput as the speed of a unit is decreased, it is used if enough parallelism exists at the application level to keep the pipeline full, but trades off increased latency and additional area overhead in the form of duplicated structures or pipeline register overhead. The overhead for these schemes results in extra energy consumption, and additionally, incorrect speculation results in discarding of operations, an additional waste of energy. Low-power designs tend to avoid these deeply pipelined approaches unless the amount of speculation is limited, the overhead for speculation is low, and the accuracy of speculation is high. Meeting required performance for an application without overdesigning a solution is a fundamental optimization. Additional circuitry designed to dynamically extract more parallelism can actually be detrimental, since the power consumption overhead of this logic is not controllable, and will be present even when the additional parallelism is absent from the application.

So far we have described the energy saving features of hardware. However, software exhibit wide variations in behaviour. Researchers have been developing hardware structures whose parameters can be adjusted on demand so that one can save energy by activating just the minimum hardware resources needed for the code that is executing.

3.3.4.1 Adaptive Cache

Caches whose storage elements can be selectively activated based on the application workload. One example of such a cache is the Deep-Submicron Instruction (DRI) cache,

which permits to deactivate its individual sets on demand by gating their supply voltages. To decide what sets to activate at any given time, the cache uses a hardware profiler that monitors the application's cache-miss patterns. Whenever the cache misses exceed a threshold, the DRI cache activates previously deactivated sets. Likewise, whenever the miss rate falls below a threshold, the DRI deactivates some of these sets by inhibiting their supply voltages, but the problem is that dormant memory cells lose data and need more time to be reactivated for their next use. So the solution is to reduce their voltages as low as possible without losing data. This is the aim of the drowsy cache, a cache whose lines can be placed in a drowsy mode [53] where they dissipate minimal power but retain data and can be reactivated faster. Figure 3.6 illustrates a typical drowsy cache line discussed in [54].

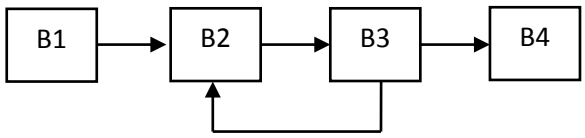


Figure 3.6: Dead Block Elimination

There are many other strategies for controlling cache lines. Dead-block elimination powers down cache lines containing basic blocks that have reached their final use. It is a compiler-directed technique that requires a static control flow analysis to identify these blocks. Since block B1 executes only once, its cache lines can power down once control reaches B2. Similarly, the lines containing B2 and B3 can power down once control reaches B4.

3.3.4.2 Adaptive Instructive Queues

One of the first adaptive instruction issue queues is presented in [55], a 32-bit queue consisting of four equal size partitions each one consists of wakeup logic that decides when instructions are ready to execute and readout logic that dispatches ready instructions into the pipeline. At any time, only the partitions that contain the currently executing instructions are activated. Many other methods have been developed for configuring these queues, few of them measures the rate at which instructions are executed per processor clock cycles, or IPC. And others find that when a program has little parallelism, the most recent part of the instruction issue queue contributes very little to the overall IPC in that instructions in this part are often committed late. Thus their heuristic monitors the contribution of the youngest part of this queue and deactivates this part when its contribution is minimal.

3.3.4.3 Algorithms for Reconfiguring Multiple Structures

In addition to this work, the problem of reconfiguring multiple hardware units simultaneously has also been taken care. The heuristics for combining hardware adaptations with dynamic voltage scaling for multimedia applications have been presented in [56] [57] with the strategy to run two algorithms while individual video frames are being processed. A global algorithm chooses the initial DVS setting and baseline hardware configuration for each frame, while a local algorithm tunes various hardware parameters (e.g., instruction window sizes) while the frame is executing to use up any remaining slack periods.

Another heuristics discussed in [58], which adjust the pipeline width and register update unit (RUU) size for different hotspots or frequently executed program regions. To detect these hotspots, the technique counts the number of times each branch instruction is taken. When a branch is taken enough times, it is marked as a candidate branch. To detect how often candidate branches are taken, the technique uses a hotspot detection counter. To measure energy at runtime, the heuristic relies on hardware that monitors usage statistics of the most power hungry units and calculates the total power based on energy per access values. [59] Discusses the other method, which provide hardware solution at the granularity of subroutines and uses offline profiling to plot an energy-delay trade-off curve. Each point in the curve represents the energy and delay trade-off of applying an adaptation to a subroutine. There are three regions in the curve. The first includes adaptations that save energy without impairing performance; these are the adaptations that will always be applied. The third represents adaptations that worsen both performance and energy; these are the adaptations that will never be applied. Between these two extremes are adaptations that trade performance for energy savings; some of these may be applied depending on the tolerable performance loss. The main idea is to trace the curve from the origin and apply every adaptation that one encounters until the cumulative performance loss from applying all the encountered adaptations reaches the maximum tolerable performance loss.

The heuristic for controlling other structures such as issue queues is occupancy based. It measures how often different components of these structures fill up with data and uses this information to decide whether to upsize or downsize the structure.

3.3.5 Dynamic Voltage Scaling (DVS)

The Dynamic voltage scaling addresses the problem about the way a processor's clock frequency gets modulated and supply voltage in lockstep as programs execute. The base is that a processor's workloads vary and that when the processor has less work, it can be slowed down without affecting performance adversely. For example, if the system has only one task and it has a workload that requires 10 cycles to execute and a deadline of 100 seconds, the processor can slow down to 1/10 cycles/sec, saving power and meeting the deadline right on time. This is presumably more efficient than running the task at full speed and idling for the remainder of the period. Though it appears straightforward, this task of how DVS works is highly simplified and hides serious real world complexities [60] [61].

3.3.5.1 Unpredictable Nature of Workloads

To predict workloads with accuracy require knowing what tasks will execute at any given time and the work required for these tasks. There are two issues, first one is that tasks can be pre-empted at arbitrary times because of user and I/O device requests and second one is that it is not possible to predict the future runtime of an arbitrary algorithm accurately every time. Especially in case of pipelining, hyper threading and out-of-order execution; it is difficult to predict execution times. Along with many other inputs a compiler also needs to know how instructions are interleaved in the pipeline, what the probabilities are that different branches are taken, and when cache misses and pipeline hazards are likely to occur which needs to develop a model for pipeline and memory hierarchy. A number of researchers have attempted to integrate complete pipeline and cache models into compilers. It remains nontrivial to develop models that can be used efficiently in a compiler and that capture all the complexities inherent in current systems.

3.3.5.2 Indeterminism and Anomalies in Real Systems

There is no direct relationship between clock frequency, execution time, and power consumption at the system level and theoretical studies are made on voltage scaling are based on certain assumptions which might be reasonable but are not guaranteed in real systems. It is also believed that total microprocessor system power is quadratic in supply voltage. But using the CMOS transistor model, the power dissipation of individual transistors is quadratic in their supply voltages, but there remains no precise way of estimating the power dissipation of an entire system.

Another misconception is that it is most power efficient to run a task at the slowest constant speed that allows it to exactly meet its deadline. Several theoretical studies attempt to prove this claim. These proofs rest on idealistic assumptions such as “power is a convex linearly increasing function of frequency”, assumptions that ignore how DVS affects the system as a whole. When the processor slows down, peripheral devices may remain activated longer, consuming more power. An important study of this issue was done by [62] who shown that, for specific DRAM architectures, the energy versus slowdown curve is “U” shaped. As the processor slows down, CPU energy decreases but the cumulative energy consumed in active memory banks increases. Thus the optimal speed is actually higher than the processor’s lowest speed; any speed lower than this causes memory energy dissipation to overshadow the effects of DVS.

A third one is that the clock frequency and execution time are in inversely proportion. Actually it is a problem defines how slowing down the processor affects the execution time of any application. DVS may result in nonlinearities.

All of these issues show that theoretical studies are insufficient for understanding how DVS will affect system state. One needs to develop an experimental approach driven by heuristics and evaluate the tradeoffs of these heuristics empirically. Most of the existing DVS approaches can be classified as interval-based approaches, inter task approaches, or intra task approaches.

3.3.5.3 Interval – Based Approaches

Interval-based DVS algorithms measure how busy the processor is over some interval or intervals, estimate how busy it will be in the next interval, and adjust the processor speed accordingly. These algorithms differ in how they estimate future processor utilization.

3.3.5.4 Inter task Approaches

Inter task DVS algorithms assign different speeds for different tasks. These speeds remain fixed for the duration of each task’s execution. Inter task DVS has two drawbacks. First, task workloads are usually unknown until tasks finish running. Thus traditional algorithms either

assume perfect knowledge of these workloads or estimate future workloads based on prior workloads. When workloads are irregular, estimating them is nontrivial.

The second drawback of inter task approaches is that they are unaware of program structure. A program's structure may provide insight into the work required for it, insight that, in turn, may provide opportunities for techniques such as voltage scaling. Within specific program regions, for example, the processor may spend significant time waiting for memory or disk accesses to complete. During the execution of these regions, the processor can be slowed down to meet pipeline stall delays.

3.3.5.5 Intra task Approaches

In this approach, the processor speed and voltage are adjusted within tasks. There are many approaches available, few of them splits each task into fixed length timeslots. The algorithm assigns each timeslot the lowest speed that allows it to complete within its preferred execution time which is measured as the worst case execution time minus the elapsed execution time up to the current timeslot. Moreover, the algorithm is pessimistic since tasks could finish before their worst case execution time. It is also insensitive to program structure. In addition to these schemes, there have been a number of intra task policies implemented at the compiler level.

[63] represents the one of the first of these algorithms, they noticed that programs have multiple execution paths, some more time consuming than others, and that whenever control flows away from a critical path, there are opportunities for the processor to slow down and still finish the programs within their deadlines. Based on this observation, a tool that profiles a program offline to determine worst case and average cycles for different execution paths was developed, and then inserting instructions to change the processor frequency at the start of different paths based on this information.

Another approach, program check-pointing, annotates a program with checkpoints and assigns timing constraints for executing code between checkpoints. It then profiles the program offline to determine the average number of cycles between different checkpoints. Based on this profiling information and timing constraints, it adjusts the CPU speed at each checkpoint.

The most well known approach is given in [64], which profile a program offline with respect to all possible combinations of clock frequencies assigned to different regions and then build a table describing how each combination affects execution time and power consumption. Using this table, selection of the combination of regions and frequencies are made that saves the most power without increasing runtime beyond a threshold.

3.3.5.6 The Implications of Memory Bounded Code

Applications with memory are on target for DVS algorithms because the time for a memory access is independent of processor speed. But if the memory is required to access frequently, it affects the time for program execution and due to this “memory wall”, the processor can actually run slower and save a lot of energy without losing as much performance as it would if it were slowing down a through compute-intensive code.

But there are many assumptions used at work, like the assumption that a DVS algorithm can predict with complete accuracy a program’s future behaviour and switch the clock frequency without any hidden costs. But the fundamental problem is how to detect program regions during which the processor stalls, waiting for a memory, disk, or network access to complete. Modern processors contain counters that measure event such as cache misses, but it is difficult to extrapolate the nature of these stalls from observing these events.

The researchers are developing techniques for modulating the processor frequency based on memory access patterns. Because it is difficult to reason abstractly about the complex events occurring in modern processors, the research in this area has a strong experimental flavour; many of the techniques used are heuristics that are validated through detailed simulations and experiments on real systems.

Unlike the previous approaches which attempt to monitor memory boundedness, a recent technique expressed in [65] attempts to monitor CPU boundedness. Their algorithm periodically measures the rate at which instructions are executed to determine how compute-intensive the workload is. The authors find that this approach is as accurate as other approaches that measure memory boundedness, but may be easier to implement because of its simpler cost model.

3.3.5.7 Dynamic Voltage Scaling in Multiple Clock Domain Architectures

The GALS (Globally Asynchronous, Locally Synchronous) chips are split into multiple domains, each of which has its own local clock. Each domain is synchronous with respect to its clock, but the different domains are mutually asynchronous in that they may run at different clock frequencies, having certain advantages such as the clocks that power different domains are able to distribute their signals over smaller areas, thus reducing clock skew, the effects of changing the clock frequency are felt less outside the given domain. This is an important advantage that GALS has over conventional CPUs. When a conventional CPU scales its clock frequency, all of the hardware structures that receive the clock signal slow down causing widespread performance loss. In GALS, one can slow down some parts of the circuit, while allowing others to operate at maximum frequencies which provides more opportunities for saving energy. In compute bound applications, GALS system can keep the critical paths of the circuit running as fast as possible but slow down other parts of the circuit.

One of the first compiler-based algorithms for controlling GALS systems is presented in [66] [67]. The main algorithm is called the shaker algorithm due to its resemblance to a salt shaker. It repeatedly traverses the Directed Acyclic Graph (DAG) representation of a program from root to leaves and from leaves to root, searching for operations whose energy dissipation exceeds a threshold. It stretches out the workload of such operations until the energy dissipation is below a given threshold, repeating this process until either no more operations dissipate excessive energy or until all of the operations have used up their slack. The information provided by this algorithm would be used by a compiler to select clock frequencies for different GALS domains in different program regions.

3.3.6 Algorithmic Level Power Reduction Techniques

Algorithmic-level power reduction techniques focus on minimizing the number of operations weighted by the cost of those operations. Selection of an algorithm is generally based on implementation such as the energy cost of an addition versus a logical operation, the cost of a memory access, and whether locality of reference, both spatially and temporally can be maximized. The presence and structure of cache memory, for example, may cause a different set of operations to be selected, since the cost of a memory access relative to an arithmetic operation changes. In general, reducing the number of operations to be performed is a first-order goal, although in some situations, re-computation of an intermediate result may be

cheaper than spilling to and reloading from memory. Techniques used by optimizing compilers, such as strength reduction, common sub-expression elimination, and optimizations to minimize memory traffic are also useful in most circumstances in reducing power [68]. Loop unrolling may also be of benefit, as it results in minimized loop overhead as well as the potential for intermediate result reuse.

In addition all above, there are many ways a compiler can help reduce power, also reconfigures hardware units or activates power reduction mechanisms, compilers can apply common performance-oriented optimizations that also save energy by reducing the execution time, optimizations such as Common Sub-expression Elimination, Partial Redundancy Elimination, and Strength Reduction. However, some performance optimizations increase code size or parallelism, sometimes increasing resource usage and peak power dissipation. Researchers have developed models for relating performance and power, but these models are relative to specific architectures.

There is no fixed relationship between performance and power across all architectures and applications. The Figure 3.7 compares the power dissipation and execution time of doing two additions in parallel [69](as in VLIW architecture) (a) and doing them in succession (b).

Doing two additions in parallel activates twice as many adders over a shorter period. It induces higher peak resource usage but fewer execution cycles and is better for performance.
To determine which scenario saves more energy, one needs more information such as the peak power increase in scenario Figure 3.7(a), the execution cycle increase in scenario Figure 3.7(b), and the total energy dissipation per cycle for Figure 3.7(a and b). For ease of illustration, it is expressed that the peak power in Figure 3.7(a) to be twice that of Figure 3.7(b), but all of these parameters may vary with respect to the architecture.

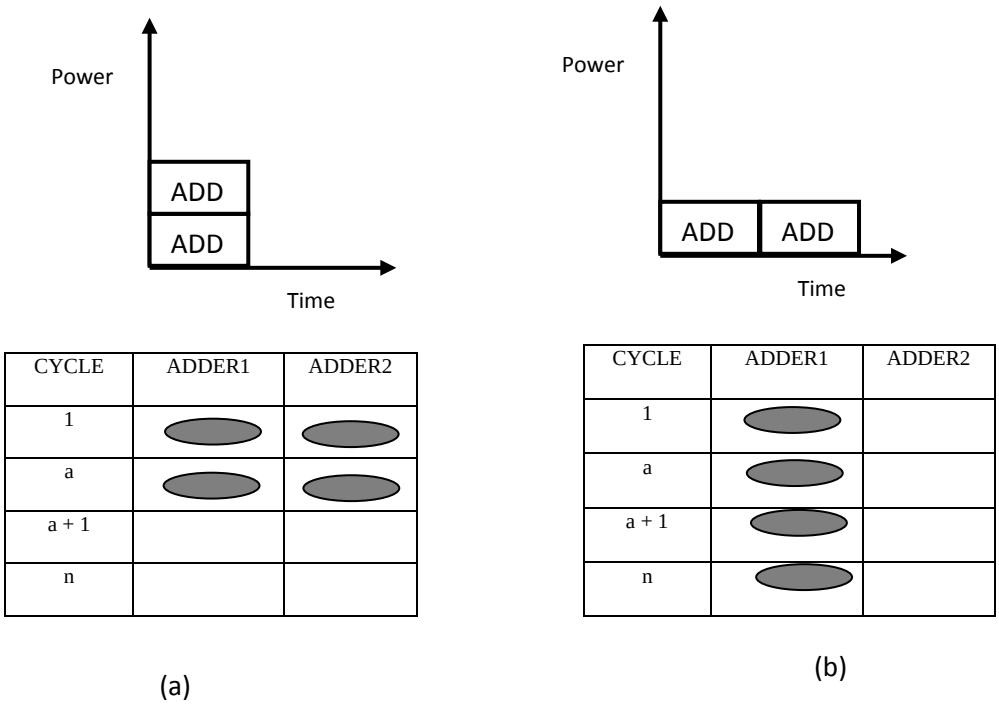


Figure 3.7: Performance Versus Power

Number representations offer another domain for algorithmic power optimization. For example, using a fixed point or a floating-point representation for data types can make significant difference in power consumption during arithmetic operations. Selection of sign-magnitude versus two's complement representation for certain signal processing applications can result in significant power reduction if the input samples are uncorrelated and dynamic range is minimized . Operator precision, or bit length, can be selected to minimize power at the cost of accuracy. In floating point algorithms, full precision can be avoided, and

mantissa and exponent width reduced below the standard 23 and 8 bits, respectively, for single precision IEEE floating point. In [70], the authors show that for an interesting set of applications involving speech recognition, pattern classification, and image processing, mantissa bit width may be reduced by more than 50% to 11 bits with no corresponding loss of accuracy. In addition to improved circuit delays, energy consumption of the floating point multiplier was reduced 20%–70% for mantissa reductions to 16 and 8 bits, respectively. Truncation of low-order bits of partial sum terms when performing a 16-bit fixed-point multiplication has been shown to result in power savings of 30% due mainly to reduction in area [71] [72].

Compilers can reduce memory accesses by eliminating redundant and load and store operations to reduce energy dissipated. Another way is to keep data as close as possible to the processor, may be in the registers and lower-level caches, using aggressive register allocation techniques and optimizations improving cache usage. Several loop transformations alter data traversal patterns to make better use of the cache. The assignment of data to memory locations also influences how long data remains in the cache. Though all these techniques reduce power consumption, they might be suboptimal along others. For example, to exploit the full bandwidth, banked memory architecture provides, one may need to disperse data across multiple memory banks at the expense of cache locality and energy.

A new domain of research for compilers involves compiling and executing applications jointly, having challenges such as the decision of what program sections to compile or execute remotely, between mobile devices and powerful servers to reduce execution time and increase battery life. Other issues include partitioning between multiple servers and handhelds, application migration, and fault tolerance.

The compiler-based approaches have demerits as a compiler's view is usually limited to the programs it is compiling and can address two problems. (1) Compilers have incomplete information about how a program will actually behave. Hence, compilers with static optimization usually depend on profiling data which is collected prior to execution, to determine which optimization should be applied in various program regions. A program's actual runtime behaviour may be different from its behaviour during simulation. (2) The compiler optimizations treat programs as if they work in a vacuum. But this may be ideal for embedded systems execution behaviour is predictable, real systems tend to be more complex. The events occurring within them continuously change and compete for processor and memory resources.

Dynamic compilation addresses some of these problems by introducing a feedback loop. A program is compiled but is then also monitored as it executes. As the behaviour of the program changes, possibly along with other changes in the runtime environment (e.g., resource levels), the program is recompiled to adapt to these changes. Since the program is continuously recompiled in response to runtime feedback, its code is of a significantly higher quality than it would have been if it had been generated by a static compiler.

There are a number of scenarios where continuous compilation can be effective. For example, as battery capacity decreases, a continuous compiler can apply more aggressive transformations that trade the quality of data output for reduced power consumption. (For example, a compiler can replace expensive floating point operations with integer operations, or generate code that dims the display.) However, there are tradeoffs. For example, the cost function for a dynamic compiler needs to weigh the overhead of recompiling a program with the energy that can be saved.

3.4 Introduction to Emerging Technologies for Power Reduction

A brief discussion on two of the upcoming technologies has been represented here, which may likely to prove as important technologies over the next decade. The focus of these techniques is on improving energy efficiency [30]. The techniques include fuel cells and MEMS systems. The researchers believe that these technologies will eventually resolve the energy constraint.

3.4.1 Fuel Cells

Fuel cells are being developed with the objective to replace the batteries used in mobile devices, as they have limited charge storage capacity and once depleted, they must be discarded or recharged if rechargeable. Also recharging can take several hours and its quality eventually slips down. In addition to all these, battery technology has low development rate and building more efficient batteries are not suited for small mobile devices as it may be bigger in size and weight.

Fuel cells are alternatives to batteries in which they generate electricity by means of a chemical reaction but they can supply energy indefinitely principally. The main components of a fuel cell are an anode, a cathode, a membrane separating the anode from the cathode, and a link to transfer the generated electricity. The fuel enters the anode, where it reacts with a catalyst and splits into protons and electrons. The protons diffuse through the membrane, while the electrons are forced to travel through the link generating electricity. When the protons reach the cathode, they combine with Oxygen in the air to produce water and heat as by-products. If the fuel cell uses a water-diluted fuel, then this waste water can be recycled back to the anode.

Fuel cells have a number of advantages such as fuels (e.g., hydrogen) are abundantly available from a wide variety of natural resources, and many offer energy densities high enough to allow portable devices to run far longer than they do on batteries. Another advantage is that refuelling is significantly faster than recharging a battery. In some cases, it merely involves spraying more fuel into a tank. A third advantage is that there is no limit to how many times a fuel cell can be refuelled. As long as it contains fuel, it generates electricity.

Several companies took very aggressive initiative to develop fuel cell technologies for portable devices including Micro Fuel Cell Inc., NEC, Toshiba, Medis, Panasonic, Motorola, Samsung, and Neah Systems and already a number of prototype cells have emerged as well as prototype mobile computers powered by hydrogen or alcohol based fuels. Even with the rapid advancement, few industries believe that it will take another 5-10 years for fuel cells to make it available in mobile applications.

Fuel cells also have several drawbacks and also very risky because they can get very hot (e.g., 500- 1000 Celsius). Also they need very expensive metallic materials and mechanical components that they are composed of. The fuel cells are highly flammable. In particular, fuel-powered devices will require high degree of safety measures as well as more flexible laws allowing them inside airplanes.

3.4.2 MEMS

MEMS (Micro-electrical and Mechanical Systems) are miniature versions of large scale devices that convert mechanical energy into electrical energy. Researchers are exploring the ways of using them to solve the energy problem. They are developing prototype millimetre scale versions of the gigantic gas turbine engines that power airplanes and drive electrical generators. These micro engines will give mobile computers unprecedented amounts of lifetime.

The micro engines work using similar principles as their large scale counterparts. They suck air into a compressor and ignite it with fuel. The compressed air then spins a set of turbines

that are connected to a generator to generate electrical power. The fuels used could be hydrogen, diesel based, or more energy-dense solid fuels.

Made from layers of silicon wafers, these tiny engines are supposed to output the same levels of electrical power per unit of fuel as their large scale counterparts. Their proponents claim they have two advantages. First, they can output far more power using less fuel than fuel cells or batteries alone. In fact, the ones under development are expected to output 10 to 100 Watts of power almost effortlessly and keep mobile devices powered for days. Moreover, as a result of their high energy density, these engines would require less space than either fuel cells or batteries.

However, it is too early to tell whether it will in fact replace batteries and fuel cells. One problem is that jet engines produce hot exhaust gases that could raise chip temperatures to dangerous levels, possibly requiring new materials for a chip to withstand these temperatures. Other issues include flammability and the power dissipation of the rotating turbines.

3.5 Conclusion

Power and energy management has grown into a multifaceted effort that brings together researchers from such diverse areas as physics, mechanical engineering, electrical engineering, design automation, logic and high-level synthesis, computer architecture, operating systems, compiler design, and application development. Above discussion is on how the power problem arises and how the problem has been addressed along multiple levels ranging from transistors to applications. Because Low-power design requires taking care of the power dissipation problem at all levels of the design hierarchy, No single target will be sufficient to extract the efficiency required for future handheld products [73].

Voltage scaling has limits that will require additional advanced techniques to be applied at the algorithmic and architectural level for additional power savings. Dynamic voltage scaling based on system loading and processing requirements is an emerging technique with great promise. Clock power optimizations will remain a challenge as higher frequencies and increased pipelining are applied to extract increased performance. Parallelism must be efficiently extracted without sacrificing the goal of low power. Software generation strategies that are based on power cost functions will be increasingly common in these future systems.

Even though a broad range of power reducing techniques have been proposed, the challenge still remains to integrate them into a design flow in which power plays as large a role as performance. Here in this chapter an in depth survey of major commercial power management technologies has been presented and also through some light on emerging technologies.

It is concluded with the understanding that the field is still active, and researchers are continually developing new algorithms, hardware level techniques and heuristics along each level as well as exploring how to integrate algorithms from multiple levels. Given the wide variety of micro-architectural and software techniques available today and the few that will be available in the future, it is highly preferred to overcome the limits imposed by high power consumption and continue to build processors offering greater levels of performance and versatility.

However, only time will tell which approaches will ultimately succeed in solving the power problem.

The following chapters will be based on complete construction of 32 – bit processor working on RISC principle with 4 – pipeline stages having many other features, this processor will be used as the system under consideration and will be implemented on FPGA as low power implementation through incorporation of low power design strategies and hence, the power saving features will be taken care and a power consumption comparison will be made with the conventional 5 – stage pipeline processor.

Chapter 4
System Architecture

4.1 Introduction

The system under consideration for this Ph.D. work is a prototype model of 32 –bit processor unit, which is designed considering the base of RISC principle and is targeted for implementation on Xilinx SPARTAN – 3E FPGA device. The embedded processors found in everyday appliances such as cell phones, personal digital assistants, and handheld game systems, are far more powerful compared to earlier ones. The embedded processor is a processor that has been embedded into a target device such as FPGA and it can be programmed to interact with different pieces of hardware. It is essential to have these embedded processors a low power processor as it becomes a part of everyday life and the expansion of battery life time, is an important aspect for the mobility of modern electronic gadgets, it also increases the device reliability [74] [75]. Embedded processors are accepted widely because they are small in size and are cost effective to fabricate, also very flexible as one can change the application or the specification very easily just by changing the software only. Low power design strategies, implemented at various abstraction levels of the system, also reduces the heat dissipation to a greater extent, which in turn reduces the packaging cost of the system as cooling arrangement may be simpler [76] [77].

This chapter includes the design of 32 – bit processor using RISC principle with 4 – stage pipeline [78] which allow the simpler implementation using the load / store mechanism and supports the predefined instruction set [79] .

4.2 Processor Architecture

The processor to be discussed here is 4 – stage pipelined processor with instruction and data memory within the FPGA chip and 32 – bits are the width of instructions and data. Due to the

difference in time taken to access a register as compared to a memory location, it is much faster to perform the operation on-chip register rather than memory. To eliminate the latency of memory operations, MIPS (Microprocessor without interlocked pipeline stages) processor uses the load/store architecture where the access to memory is only through load and store instructions. There are total 16 general purpose registers identified as R_0 to R_{15} and the size of each one is 32 – bit. Most of the instructions have two operands, one is a register operand (RD for destination) and the other is a register operand (RS for source) or an immediate or a direct or an indirect address in case of load store instruction. The operand result is written back to RD. The RD and RS registers can be one of the 16 general purpose 32 – bit registers i.e. from R_0 to R_{15}. Other special function registers discussed below are also designed which are essential for the pipeline operations.

Program counter (PC): It is a 32 – bit long register, holds the address of the next instruction which is to be fetched from the memory during the next clock cycle and to be executed. Normally PC is incremented by one during each clock cycle unless a branch instruction executed. In case of branch instruction is encountered the PC will jump to the branch offset address and start pointing to fetch the next instruction to be executed [80].

Instruction Register (IR): It is a 32 – bit register used by CPU itself. The instruction pointed by PC and fetched from the memory is loaded into this register. IR is not programmable and cannot be accessed.

Two Registers (A and B): The size of both the registers is 32 – bit, and used to hold two source operands.

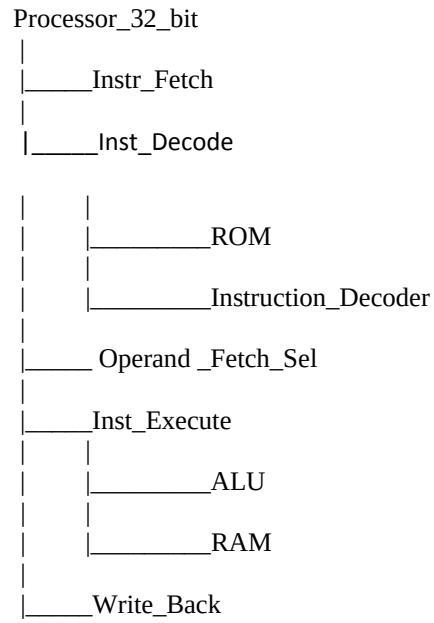
Register (C): It is 32 – bit register used to hold the result of the operations, it is a destination operand.

Single bit register (Z): It is used to hold the zero flag for conditional branching. The branch instructions will be evaluated using this register.

The detailed architecture of a processor is given in Figure 4.1, it explains the 4 – pipeline stages of the processor, which are Instruction Fetch (IF), Decode and Operand Fetch (DC), Execution or Memory Access (EX) and Write Back (WB). All these stages [81]can be detailed hierarchically as shown below.

Pipelining structures are used in the processors to permit overlapping execution of multiple instructions within the same circuitry. This system is divided into the number of stages which

includes instruction decoding, arithmetic, registers fetching stages, and also has a pipeline structure, where one instruction is processed in each stage at a time.



Along with the pipeline structure, the processor architecture also incorporates the data forward unit and the hazard detection unit to maintain the proper data flow through the pipeline stages. Each of the stage of the pipeline along with the data forward and hazard detection unit are described in detailed as follows:

4.2.1 IF Stage

This stage consists of Program counter (PC), Instruction Memory and the Branch detection Unit. In this stage, the content of PC is sent to ROM location from which the next instruction to be executed is to be fetched. At the same time the PC predictor predicts the next instruction. If in previous instructions decode stage a branch taken is detected then, according to the sign, immediate value is incremented to or decremented from PC else PC is incremented by 1.

4.2.2 DC Stage

In this stage the instruction is now in the instruction register to be decoded and corresponding operand is to be fetched. The control unit generates the control signals, which are utilized for proper synchronization and operation of the overall system. The various signals decoded by the instruction decoder, which can be used for read and write operations for register bank and

program memory, which are as described in the following section. It also generates the signals which decide the usage of multiplier and ALU, and also generates the flags used by

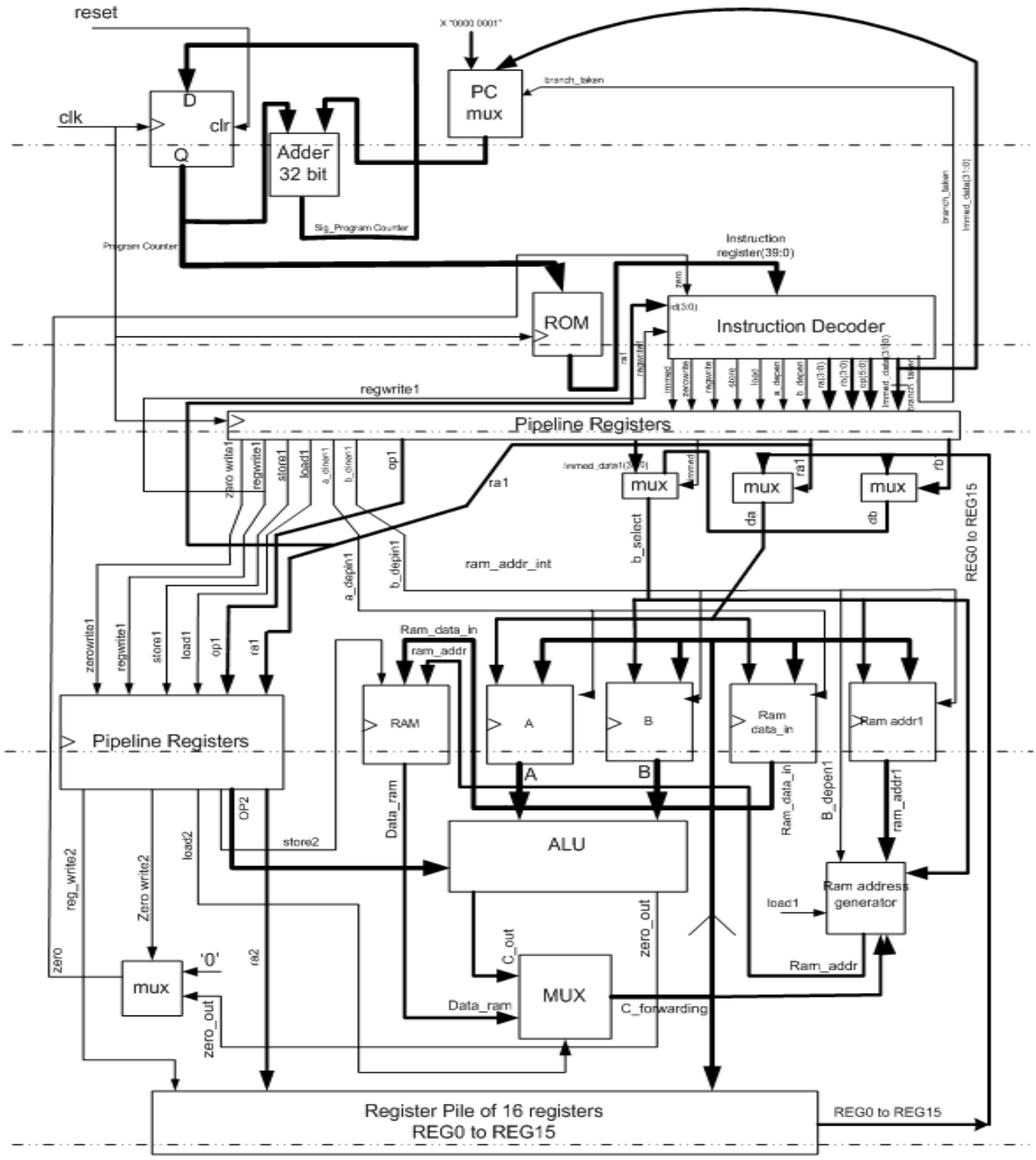


Figure 4.1: Detailed Architecture of 4 – Stage Pipelined Processor Under Consideration

branch unit and produces the clock gating signals for ALU control. Instruction decoder decodes the following signals:

- Btaken: 1 bit signal Branch taken to indicate that current instruction is one of the three branch instruction.
- Regwrite: 1 bit signal indicating that current instruction will write to a destination register. Nop, store and all three branch instruction don't generate this signal. For all other instructions this bit is decoded to logic 1.
- Load: 1 bit signal indicating that current instruction is a load instruction and data will be loaded from data memory to the destination register.
- Store: 1 bit signal indicating that current instruction is a store instruction and data will be stored to data memory from the source register.
- Op: 6 bit opcode is decoded in to this register. Opcode is detected to be either Mov, Add, Sub, Mul, Or, And, Xor, Ror, Rol, Slr, Sll, Inc, Dec, Cmp and Clr.
- a_depen: It's a 1 bit destination register dependency signal indicating that current instruction destination register is same as the previous instruction source or destination register. It is used by ALU operand A as selection signal; and if it go high, the data of register C is selected and forwarded, otherwise, the data of register A is selected.
- b_depen: It's a 1 bit source register dependency signal indicating that current instruction source register is same as the previous instruction source or destination register. It is used by ALU operand B as selection signal; and if go high, the content of register C is selected and forwarded, otherwise, the data of register B is selected.
- Immed: This 1 bit signal indicates that the current instruction is one of the instructions that operates on the immediate data using immediate addressing mode.
- Immed_data: This 32 bit register holds the 30 bit immediate value for the instruction involving operations on immediate data or 5 bit of shift or rotate for the two shift and two rotate instruction or 32 bit branch data for the three branch instruction.
- Sign: As Sign is a bit controllable signal, it indicates whether the branch will cause the program counter to be incremented or decremented by the immediate value.
- Ra: Destination register operand is decoded into this 4 bit register.
- Rb: Source register operand is decoded into this 4 bit register.

Data dependencies are detected in this stage so accordingly data forwarding can be done from write back stage to execute stage. Branch prediction is also done in this stage. For the data dependency, consider the following sequence of operations of pipeline executions mentioned as I1, I2, and I3.

I1: ADD R1, R2 ; R1 ⟵ R1 + R2

I2: SUB R3, R1 ; R3 ⟵ R3 - R1

I3: SUBI R1, 1 ; R1 ⟵ R1 – 1

In case of instruction I2, it reads R3 and R1 from the register file in the Decode and Operand Fetch stage, and writes them to A and B registers respectively and at the same time, the instruction I1 add R1 and R2 and write it to C; then store the sum to R1 in the next stage. Hence, I2 will receive the previous data from register file and write it to B. If I2 uses it to perform subtraction, it will result into wrong output. But when I3 reading R1, there will not be any trouble. The remedy is to insert NOP instruction between I1 and I2 to introduce delay for the execution of I2, but the performance will be affected. Hence, a data path is to be designed in a way that it should solve this problem and a dependency detection block should be able to identify the dependencies if any. Once the data dependency is detected, the source data required for ALU operation is passed from C via multiplexer, instead of from A or B. The data dependencies occurs under certain conditions, which are like (a) the operand A / B of the current instruction is a register operand (cRD / cRS), (b) the result of the previous instruction will be written into register file (pRD) and (c) cRD/cRS and pRD are the same register.

The detection of dependency is done in EX stage. In this processor design, we do this operation in DC stage, and make use of pipeline registers to transfer to EX stage. This arrangement offers few advantages, which are (1) The dependency detection in DC stage will reduce the use of number of gates because a common logic can be shared with other decode circuits. (2) The time required by EX stage will be shortened because the signals a_depen and b_depen are made available immediately at the beginning of EX stage. For the control dependency, we use a delay branch method. In the proposed processor design, the branch target address is evaluated in DC stage and it introduce one delay cycle for which an additional adder is required for address evaluation. This technique is implemented in this wok to achieve the optimization by rearranging the instruction codes described in detail in Chapter 5.

Operation fetch module fetches the data from the register file corresponding to source and destination operand.

- Sig_da: This 32 bit register holds the value of register pointed by destination operand.
- Sig_db: This 32 bit register holds the value of register pointed by source operand.
- B_select: This 32 bit register holds the value of register pointed by source operand or 32 bit immediate value depending on the status of immed signal.

4.2.3 EX stage

In the execution stage depending on the instructions either data is fetched from the data memory or stored into it or an ALU operation is performed. This stage includes ALU, ALU control Unit and Multiplier.

4.2.3.1 Data memory access

Block RAM of Xilinx is used as data memory. For writing data to memory, address is provided along with data, but the data reading operation has latency of one clock. So when there is load instruction to load data from memory to register, RAM address is generated one clock cycle earlier. Also depending on the dependency signal for RAM address, address is given by b_select or forwarded through c_forward, it depends on the status of b_depen signal. Similarly data to be written to memory is either given by sig_da or c_forward depending on a_depen signal if there is dependency between two consecutive instructions.

4.2.3.2 ALU

ALU is responsible for all arithmetic and logic operations that take place within the processor. These operations can have one or two operands, and these values are coming either from the registers or may be immediate value from the instruction directly. Either A or B register is selected according to the a_depen and b_depen signals and given to ALU. ALU then performs operation depending on the opcode and generate the result into register C. Zero flag is generated if the result in C is zero. A complete architecture of ALU unit is shown in Figure 4.2.

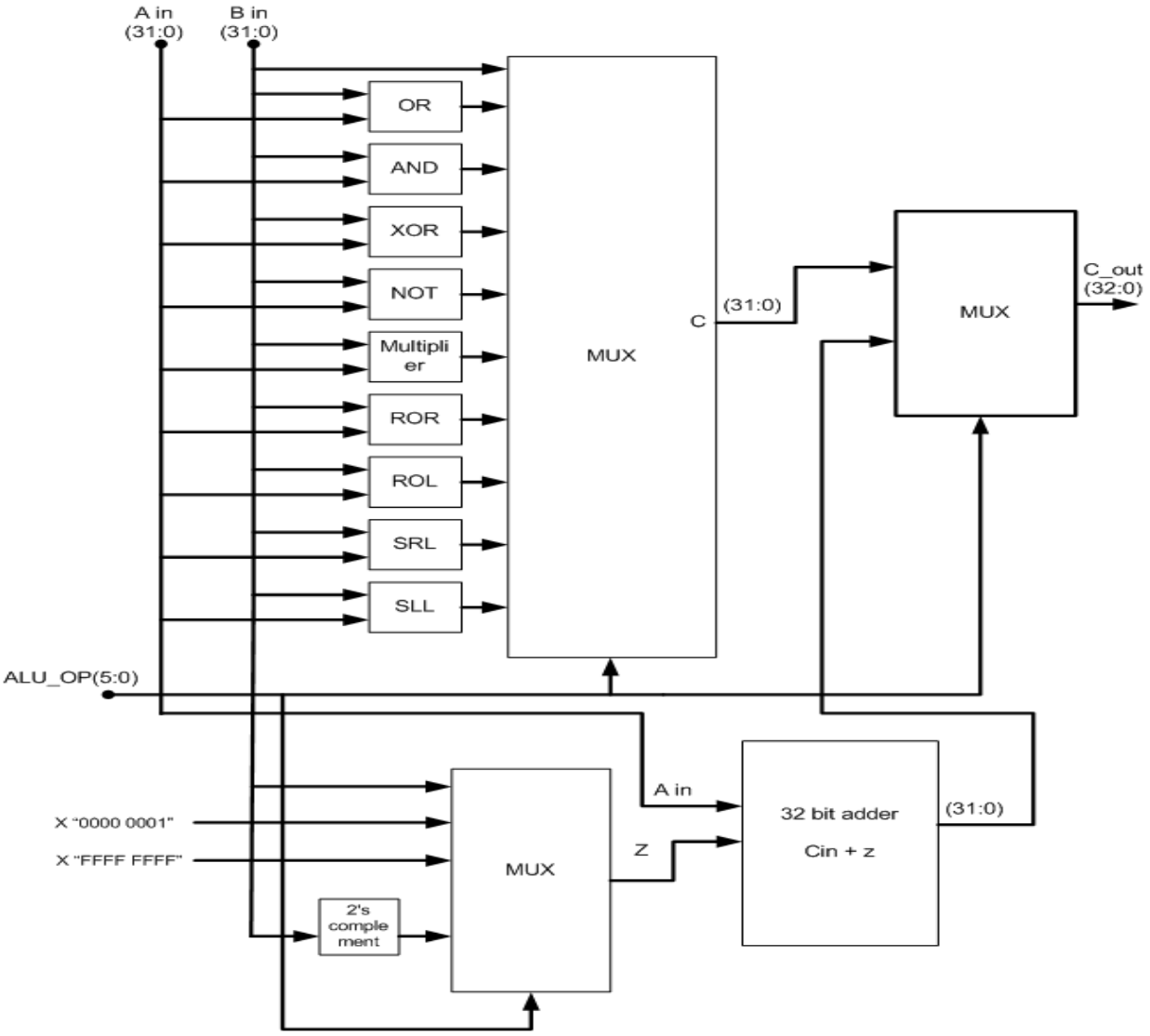


Figure 4.2: An ALU Architecture for 4 – Stage CPU

4.2.4 WB stage

During this stage the result generated by the instruction is written back into the one of the general purpose registers. In this stage the regwrite signal pipelined to this stage is checked for 1, if it is one indicating that destination register is to be updated. Thus the value in C register is updated to the corresponding destination register out of 16 general purpose registers.

4.3 Instruction Set Formation

The set of instructions is interpreted directly by the CPU. These instructions are encoded as bit strings in memory and are fetched and executed one by one by the processor. They perform primitive operations such as “add 2 to register i1”, “store contents of R6 into memory location 0xFF32”, This processor architecture supports mainly three types of the instructions such as (a) register type, for which both the operands are registers; (b) the immediate type, which consists of register as one of the operands and an immediate value as another operand; and the third one is (c) branch type instructions. The instruction formats for all these types of instructions are as given below. The format length is 40-bit.

39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

(a) Register Type: Format for all instructions where both operands are registers. It includes the instructions such as

add, sub, mul,

or, and, xor, move,

load, store,

rotate right, rotate left, shift right, shift left.

6 bits opcode	4 bits Operand 1	4 bits Operand 2	26 bits All 0’s

(b) Immediate Type: Format for all instructions where one operand is register and other is an immediate data it includes the instructions such as

add, sub, mul,

or, and, xor,

move, load, store,

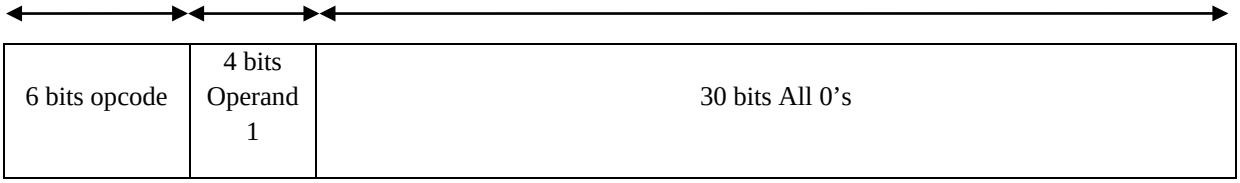
rotate right, rotate left, shift right, shift left.

6 bits opcode	4 bits Operand 1	30 bits immediate value

Format for instructions given below which operates on a following register.

Increment, decrement,

Clear, Complement



(c) Branch Type: Below is the instruction format for the three predefined branch instruction and two update instructions.

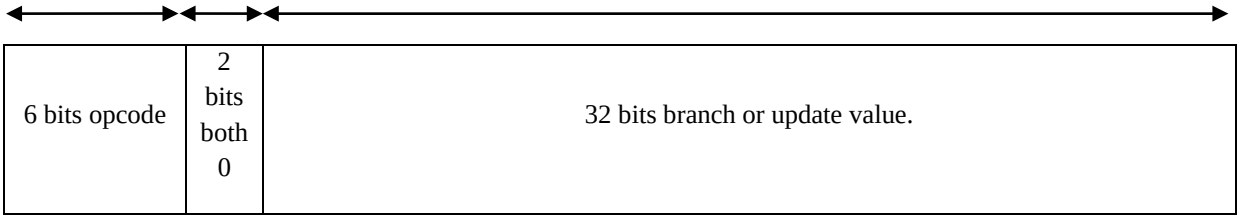


Figure 4.3: Formats for Various Instructions

Following Table 4.1 summarizes all the instruction supported by this processor. It is not a complete instruction set for this processor, only a part of the complete instruction has been developed to deal with this processor to carry out the work in this thesis. In list op1, op2 and op are of 4 – bits indicating the register operand. The immediate and direct address is 30 bit value and the branch address and update are 32 – bit value.

Table 4.1: Summary of All the Instructions Supported by this Processor

No	Instruction	Operation	Opcode	Instruction Format
1	Nop	No operation	000000	(000000)(All 0's)
2	Add	Add operand 2 to 1	000001	(000001)(op1)(op2)(all 0's)
3	Sub	Subtract operand 2 to 1	000010	(000010)(op1)(op2)(all 0's)
4	Mul	Multiply operand 2 to 1	000011	(000011)(op1)(op2)(all 0's)
5	Or	Oring operand 2 to 1	000100	(000100)(op1)(op2)(all 0's)
6	And	Anding operand 2 to 1	000101	(000101)(op1)(op2)(all 0's)
7	Xor	Xoring operand 2 to 1	000110	(000110)(op1)(op2)(all 0's)
8	Mov	Move operand 2 to 1	000111	(000111)(op1)(op2)(all 0's)
9	Ror	Rotate right operand 1 by amount specified in operand 2	001000	(001000)(op1)(op2)(all 0's)
10	Rol	Rotate left operand 1 by amount specified in operand 2	001001	(001001)(op1)(op2)(all 0's)
11	Slr	Shift right operand 1 by amount specified in operand 2	001010	(001010)(op1)(op2)(all 0's)
12	Sll	Shift left operand 1 by amount	001011	(001011)(op1)(op2)(all 0's)

		specified in operand 2		
13	Load	Load value to register pointed by operand 1 from memory location pointed by operand 2	001100	(001100)(op1)(op2)(all 0's)
14	Store	Store value from register pointed by operand 1 to memory location pointed by operand 2	001101	(001101)(op1)(op2)(all 0's)
15	Addi	Add 30 bit immediate value to operand 1	001110	(001110)(op)(immediate)
16	Subi	Subtract 30 bit immediate value to operand 1	001111	(001111)(op)(immediate)
17	Muli	Multiply 30 bit immediate value to operand 1	010000	(010000)(op)(immediate)
18	Ori	Or 30 bit immediate value to operand 1	010001	(010001)(op)(immediate)
19	Andi	And 30 bit immediate value to operand 1	010010	(010010)(op)(immediate)
20	Xori	Xor 30 bit immediate value to operand 1	010011	(010011)(op)(immediate)
21	Movi	Move 30 bit immediate value to operand 1	010100	(010100)(op)(immediate)
22	Rori	Rotate right operand 1 by amount specified by 5 bit immediate	010101	(010101)(op)(immediate)
23	Roli	Rotate left operand 1 by amount specified by 5 bit immediate	010110	(010110)(op)(immediate)
24	Slri	Shift right operand 1 by amount specified by 5 bit immediate	010111	(010111)(op)(immediate)
25	Slli	Shift left operand 1 by amount specified by 5 bit immediate	011000	(011000)(op)(immediate)
26	Loadi	Load value to register pointed by operand 1 from memory location indicated by 11 bit immediate	011001	(011001)(op)(direct address)
27	Storei	Store value from register pointed by operand 1 to memory location indicated by 11 bit immediate	011010	(011010)(op)(direct address)
28	Bz	Branch if Zero and start fetching instructions from location specified by 32 bit immediate address	011011	(011011)(2 bit 0's)(Branch Add)
29	Bnz	Branch if not Zero and start fetching instructions from location specified by 32 bit immediate address	011100	(011100)(2 bit 0's)(Branch Add)
30	Br	Branch and start fetching instructions from location specified by 32 bit immediate address	011101	(011101)(2 bit 0's)(Branch Add)
31	Sr0l	Update R0 register least significant word	011110	(011110)(2 bit 0's)(update value)
32	Sr0h	Update R0 register most significant word	011111	(011111)(2 bit 0's)(update value)
33	Inc	Increment operand by 1	100000	(100001)(op)(All 0's)
34	Dec	Decrement operand by 1	100001	(100010)(op)(All 0's)
35	Cmp	Complement operand	100010	(100011)(op)(All 0's)
36	Clr	Clear operand	100011	(100100)(op)(All 0's)

4.4 Sub-modules of Processor

The processor is designed in modular fashion and includes ALU, Register File, Instruction Memory, Data Memory, Decoder, Pipeline registers, and multiplexors as major modules. The four pipeline stages are presented in the Figure 4.1. Following section discusses the major sub-modules incorporated in this processor.

4.4.1 ALU Design

The ALU of the processor implemented here performs 14 different operations. A 6 - bit signal ALU_op selects the ALU operation shown in Figure 4.2. The 14 operations along with the possible variations defined whish are to be performed by this ALU are listed below.

1) MOV (for MOV, MOVI and SR0H)
2) OR (for OR, ORI and SR0L)
3) XOR (for XOR and XORI)
4) AND (for AND and ANDI)
5) SUB (for SUB and SUBI)
6) ADD (for ADD and ADDI).
7) MUL (for MUL and MULI)
8) INC (for INC)
9) DEC (for DEC)
10) CMP (for CMP)
11) CLR (for CLR)
12) ROR (for ROR and RORI)
13) ROL (for ROL and ROLI)
14) SLR (for SLR and SLRI)
15) SLR (for SLL and SLLI)

4.4.2 Register File Design

Register is a storage location directly on the CPU, used for temporary storage of small amount of data during processing. In this processor implementation, we have constructed 16 general purpose registers $R_0 - R_{15}$, each one is 32 bit wide. Hence, four bits are required to use for addressing the register file i.e. 16 × 32 bits having two read ports and a write port. It can be implemented with 16, 32-bit registers and a pair of 16-to-1 multiplexors and each one is of 32 bits wide for read ports and uses a 4-to-16 decoder for write control.

4.4.3 Data Memory Design

Memory array is randomly accessible to memory bytes, each one identified by a unique address. Flat memory models, segmented memory models, and hybrid models exists which are distinguished by the way the locations are referenced and potentially divided into sections. As proposed processor development is based on Harvard architecture, there must be provisions for separated instruction memory module and data memory module. Data memory has 2048 x 32 bits. We use Xilinx block RAM modules to build on-chip memory. For this implementation the Xilinx coregen block memory generator is used for generating RAM. It made memory access possible once at each cycle. The load and store instructions are used to access this module.

4.4.4 Instruction Memory Design

This unit contains the instructions that are executed by the processor. Instruction memory has 2048 × 40 bits. For the instruction memory design, we have used Xilinx coregen block memory generator. The ROM is initialized to a known value with the coe file at the time of ROM generation. Coe file hold the instruction that are dumped into ROM at the time of core generation.

4.4.5 Instruction Decoder

The internal structure of instruction decoder is made up of multiplexers, comparators and the logic gates. Figure 4.4 explain the detailed internal architecture of instruction decoder, the operational role of the multiplexer is defined in Figure 4.5 and the useful signals which are utilized during the operations such as instruction fetch execution and write back as shown in Figure 4.6 .

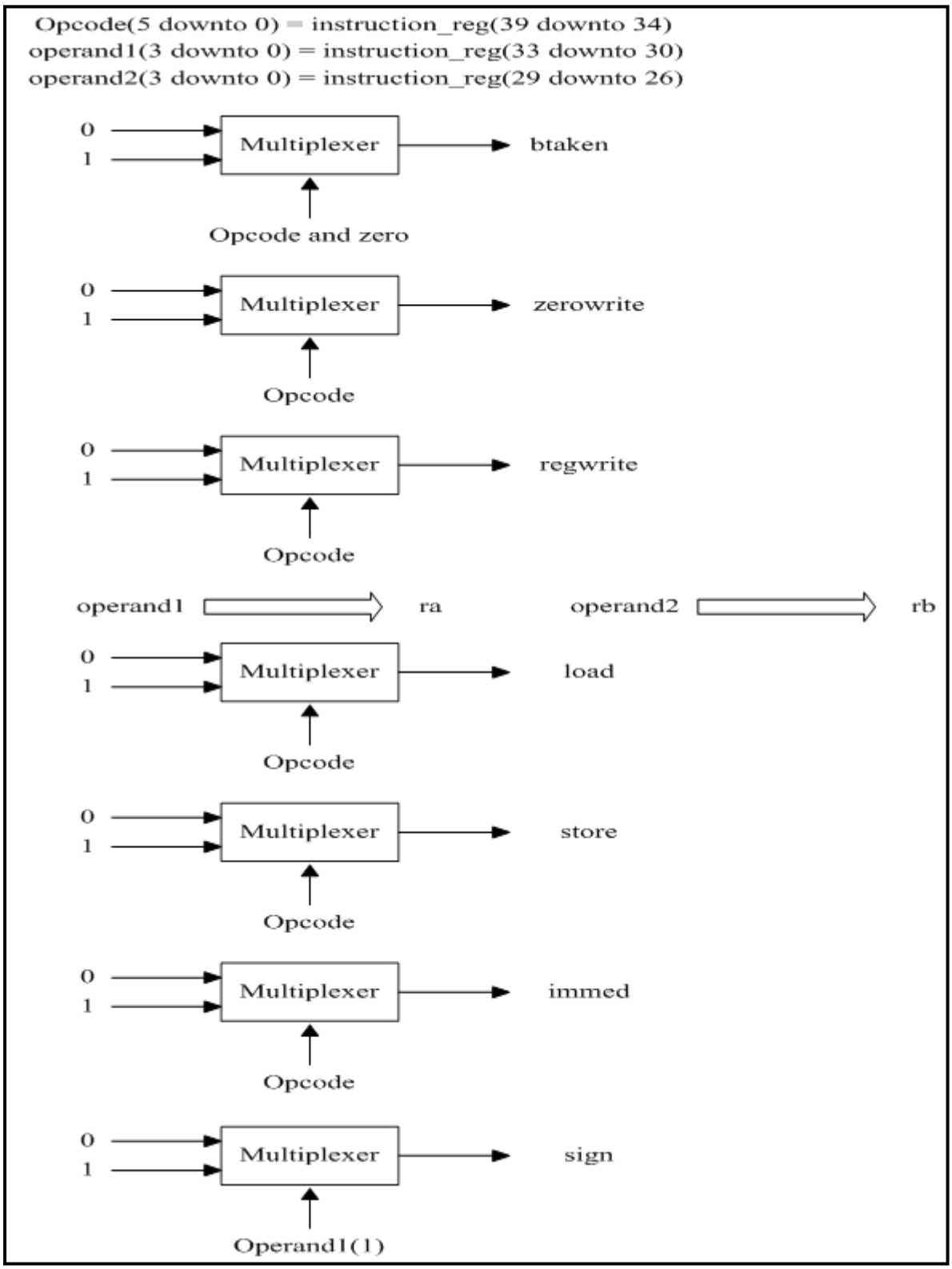


Figure 4.4: Detailed Internal Architecture of Instruction Decoder

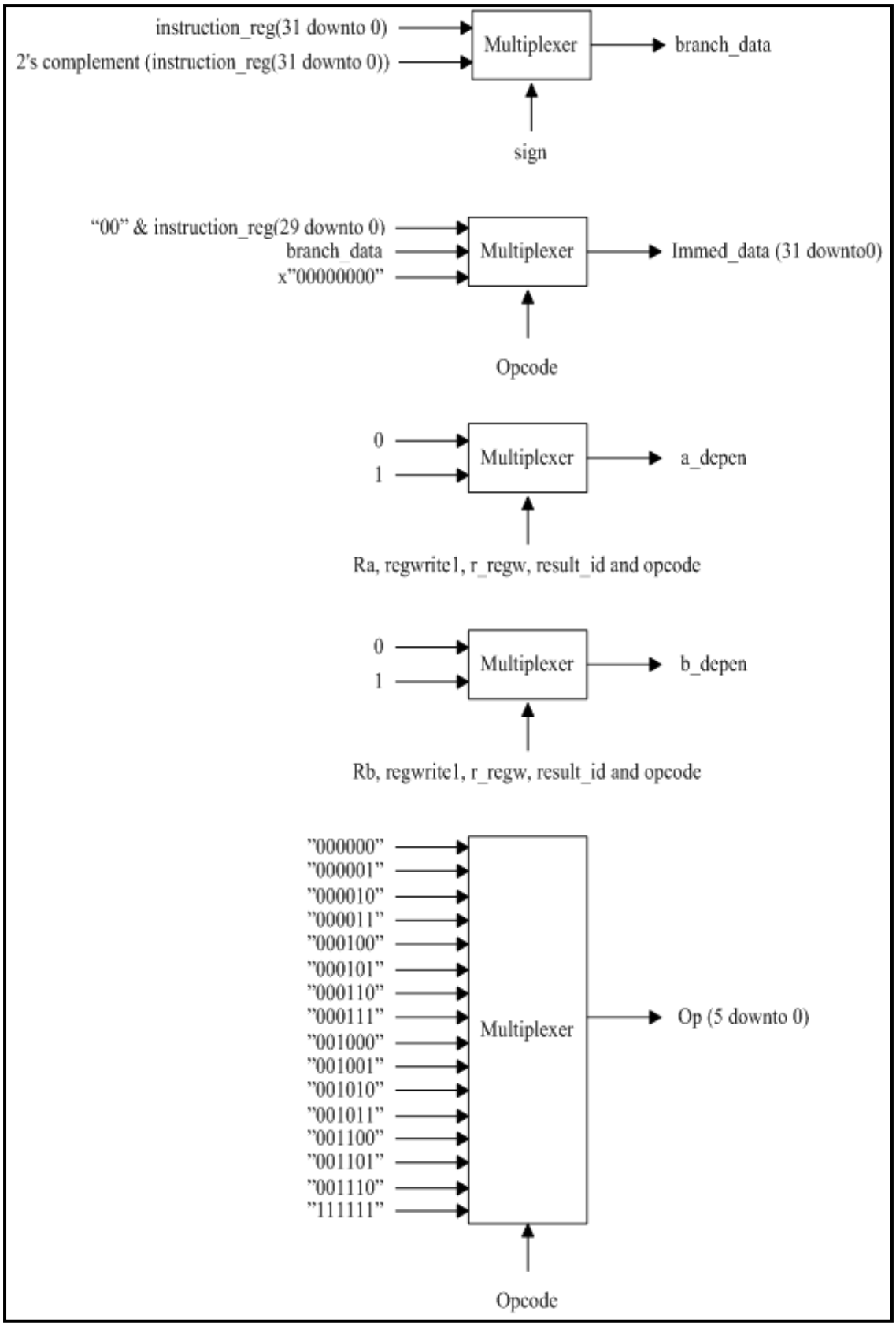


Figure 4.5: Internal Architecture of Multiplexer

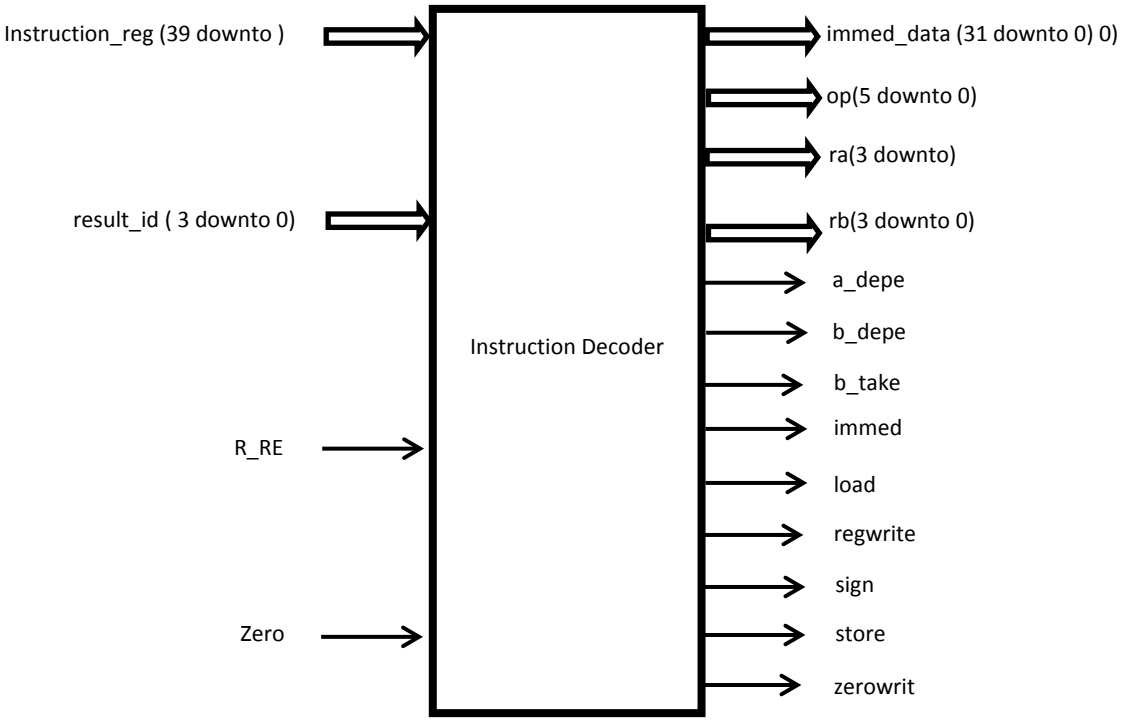


Figure 4.6: Diagram of Instruction Decoder with all relevant Signals

4.4.6 Control Unit Design

The control unit produce all the necessary control signals which are to be utilized for the synchronization among all the components of the processor. This unit also provides the signals that control all the read and write operations of the register file and also for decision about when to use the multiplier and when to use the ALU. It also generates appropriate branch flags that are used by the Branch Decide Unit. In addition to all these, this unit provides clock gating signals for the ALU control and the Branch Adder module. In short, this unit generates all the control signals for controlling all the data path activities. Summary of all the control signals is given in Table 4.2.

Table 4.2: Summary of Control Signals

Sr. No.	Signal	Description of Signal
01	BTAKEN Branch Taken	If branch is taken, BTAKEN go high to select the branch address for instruction to be fetched in the next clock cycle.
02	RA<3:0>	Used to point out the number of the destination register RD, but RD can also be source 1 register (RD = RD op RS); almost all the instructions keep the number of RD in a fixed position for the instruction format, but for SR0L and SR0H instructions, the number

		of RD is always 0 for these instructions.
03	RB<3:0>	It shows the register number of RS which is the source 2 register. The Most of the instructions put the number of RS in a fixed position, but few instructions use it as immediate data in that position.
04	IMMED Immediate Selection	When the source 2 operand is an immediate, IMMED go high for the selection of immediate, not register operand.
05	IMMED_DATA<31:0> Immediate	When it is 32-bit immediate data. Depending upon instructions, it can be produced with different extension techniques. The IMMED_DATA<31:0> is used for the source 2 operand of ALU operations and also for the branch target address calculation.
06	OP<5:0> ALU Operation Control	This signal will be generated based on instructions.
07	ADEPEN (A Dependent)	It is the selection signal for ALU operand A, if high; the forwarding data of register C is selected, else the data of register A is selected.
08	BDEPEN (B Dependent)	It is the selection signal for ALU operand B; if high, the forwarding data of register C is selected, else the data of register B is selected.
09	STORE Store	It is called memory-write control signal, if it go high, a memory write operation performed.
10	LOAD Load	When this signal LOAD is high, the register C will be written with the loaded data from data memory.
11	ZEROWRITE Zero Register Write	When ZR WRITE is high, zero flag register will be updated.
12	RA2<3:0>	It is the same signal as RA<3:0> of DC stage and is pipelined to retain the number of destination register for the purpose to be written in the write back stage.
13	REGWRITE (Register Write)	It is the register-write control signal. If it goes high, then the register C content will be shifted into register file.

All of the control signals are generated within the DC stage. However, few of them are actually utilized in EX and WB stages. To handle this situation, pipeline registers are used to assign a proper signal to the corresponding stages. In our processor design the control unit is divided into five blocks and each block generates the relevant signal. These blocks are described as follows with related signals.

- Branch block generates BTAKEN,
- Register address block generates RA<3:0> and RB<3:0>,
- ALU control block generates OP<5:0>, ZEROWRITE, and REGWRITE,
- Immediate block generates IMMED and IMMED_DATA <31:0>,
- Dependent block generates A_DEPEN and B_DEPEN.

4.5 Multiplier Unit & Its Logic

For multiplication to be done in the Spartan 3e FPGA, the dedicated multipliers are used, it provides 4 to 36 dedicated multiplier blocks per device. The multipliers are located together with the block RAM in one or two columns depending on device density. The multiplier blocks primarily perform two's complement numerical multiplication and it can also perform some less obvious applications, such as simple data storage and barrel shifting. Logic slices also implement efficient small multipliers and thereby supplement the dedicated multipliers. Each multiplier performs the principle operation P = A X B ; where 'A' and 'B' are 18 – bit words in two's complement form, and 'P' is the full-precision 36 – bit product, also in two's complement form. The 18 – bit inputs represent values ranging from -131,07210 to +131,07110 with a resulting product ranging from –17,179,738,11210 to + 17,179,869,18410.

Wider multiplication operation can be possible by combining the dedicated multipliers and slice-based logic in any viable combination or by time-sharing a single multiplier, perform unsigned multiplication by restricting the inputs to the positive range. Tie the most-significant bit low and represent the unsigned value in the remaining 17 lesser-significant bits. Figure 4.7 show that each multiplier block has optional registers on each of the multiplier inputs and the output. The registers are named AREG, BREG and PREG and can

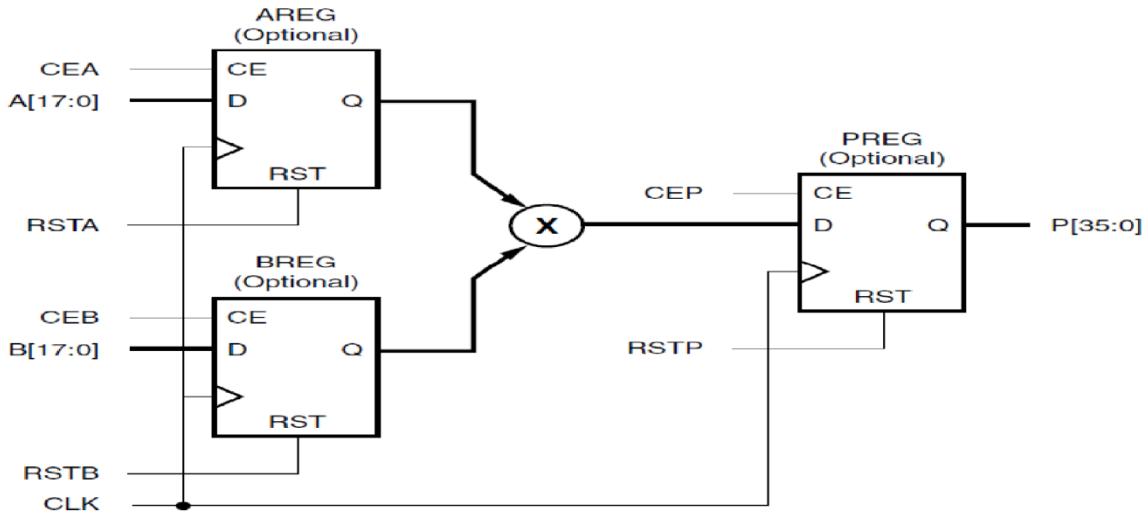


Figure 4.7: Main Features of Multiplier Block

be used in any combination. The clock input is common to all the registers within a block, but each register has an independent clock enable and synchronous reset controls making them

ideal for storing data samples and coefficients. When used for pipelining, the registers boost the multiplier clock rate, which is beneficial for higher performance applications. The MULTI8X18SIO primitive shown in Figure 4.8 is used to instantiate a multiplier within a design. Although high-level logic synthesis software usually automatically infers a multiplier, adding the pipeline registers might require the MULT18X18SIO primitive. Connect the appropriate signals to the MULT18X18SIO multiplier ports and set the individual AREG, BREG, and PREG attributes to '1' to insert the associated register, or to 0 to remove it and make the signal path combinatorial.

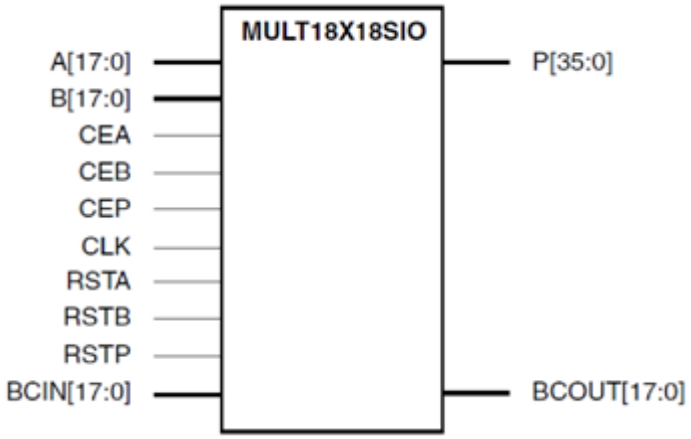


Figure 4.8: Pin Diagram of MULT18X18SIO

The MULT18X18SIO primitive has two additional ports called BCIN and BCOUT to cascade or share the multiplier's 'B' input among several multiplier blocks. The 18-bit BCIN "cascade" input port offers an alternate input source from the more typical 'B' input. The B_INPUT attribute specifies whether the specific implementation uses the BCIN or 'B' input path. Setting B_INPUT to DIRECT chooses the 'B' input. Setting B_INPUT to CASCADE selects the alternate BCIN input. The BREG register then optionally holds the selected input value, if required. BCOUT is an 18-bit output port that always reflects the value that is applied to the multiplier's second input, which is either the 'B' input, the cascaded value from the BCIN input, or the output of the BREG if it is inserted. Figure 4.9 illustrates the four possible configurations using different settings for the B_INPUT attribute and the BREG attribute.

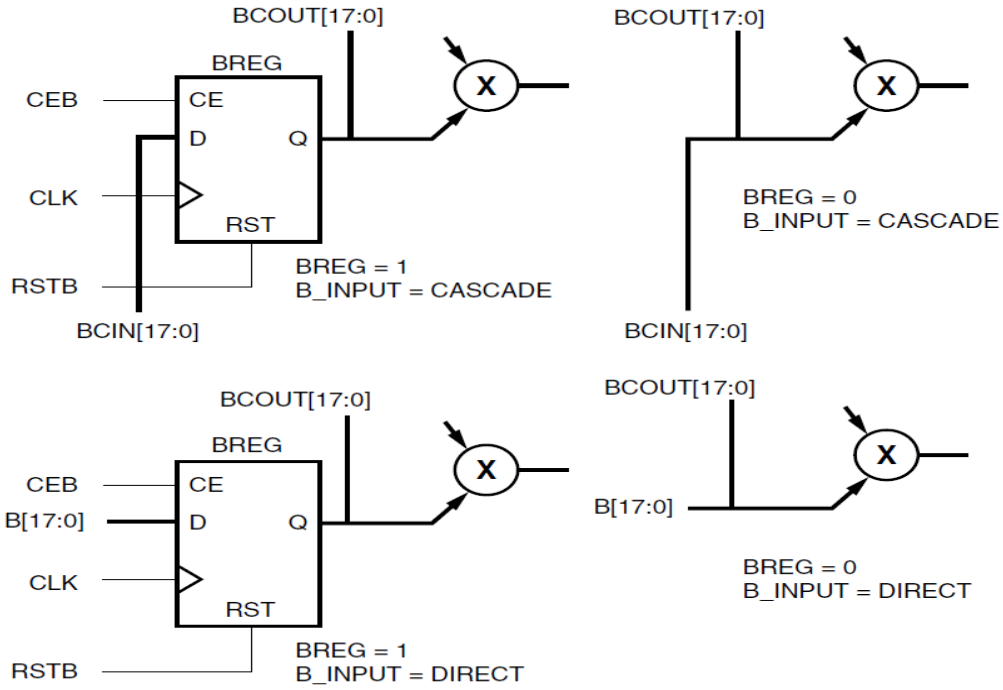


Figure 4.9: Four possible Configures for the B_INPUT Attribute and BREG Attribute

The BCIN and BCOUT ports have associated dedicated routing that connects adjacent multipliers within the same column. Via the cascade connection, the BCOUT port of one multiplier block drives the BCIN port of the multiplier block directly above it. There is no connection to the BCIN port of the bottom-most multiplier block in a column or a connection from the BCOUT port of the top-most block in a column.

4.6 Clock Distribution Network

In this processor design Spartan 3e xc3s500e FPGA device is used for its implementation and its clock distribution network is utilized for the functionality. The Spartan-3E clock distribution network is as shown in Figure 4.10, it presents a series of low-capacitance and low-skew interconnect lines which is suitable to carry high-frequency signals throughout the FPGA's blocks. The infrastructure also includes the clock inputs and BUFGMUX clock buffers/multiplexers. The Xilinx Place-and-Route (PAR) software automatically routes high-fan out clock signals using these resources. Clock pins can be connected directly to external clock signals and also to DCMs and BUFGMUX elements. Each Spartan-3E FPGA has 16 Global Clock inputs (GCLK0 through GCLK15) located along the top and bottom edges of the FPGA. Out of which 8 are Right-Half Clock inputs (RHCLK0 through RHCLK7) located along the right edge and 8 are Left-Half Clock inputs (LHCLK0 through LHCLK7) located

along the left edge. Each clock input is also optionally a user-I/O pin and connects to internal interconnect. Some clock pad pins are input-only pins.

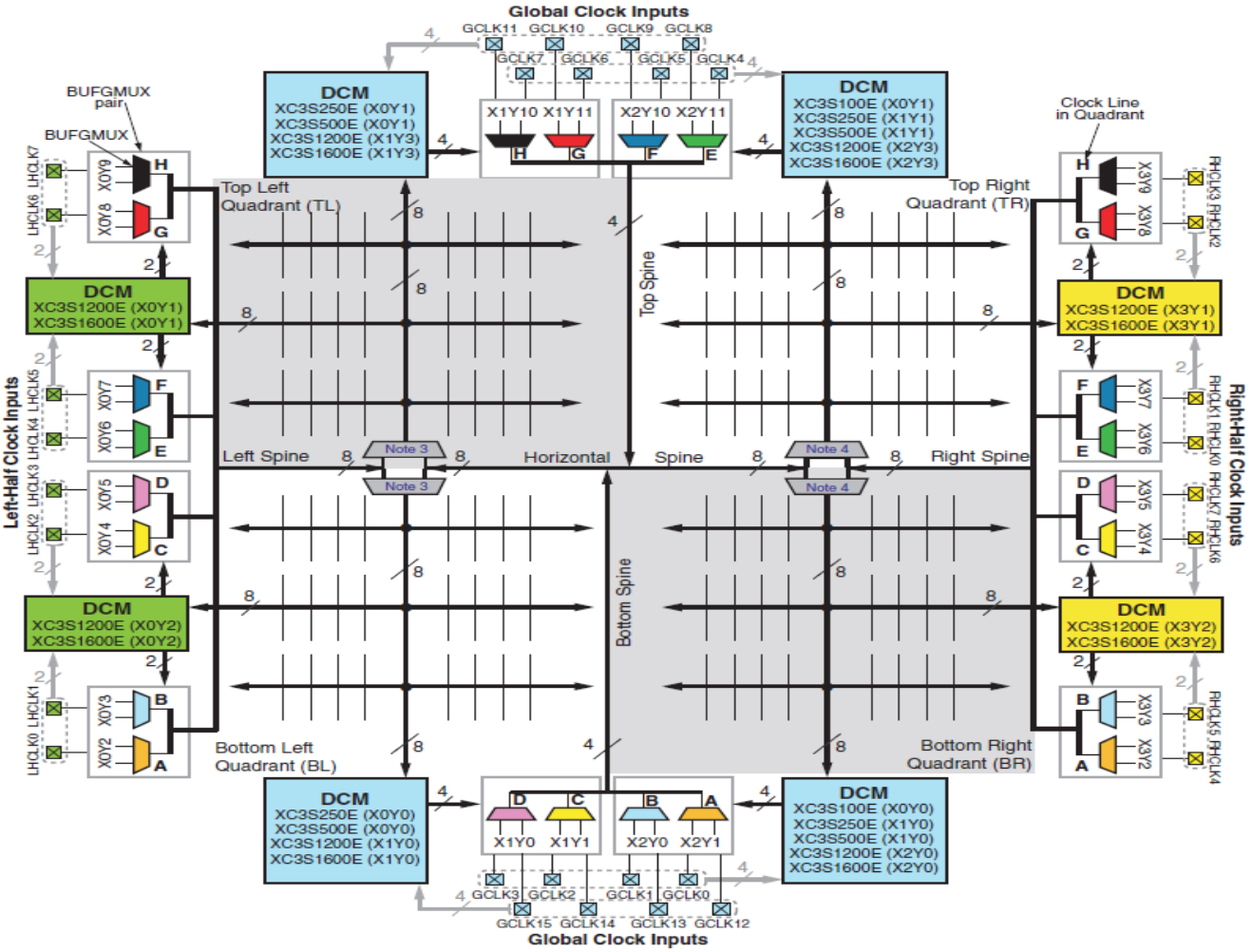


Figure 4.10: Xilinx SPARTAN – 3E Clock Distribution Network (Courtesy of Xilinx™ Co.)

Clock Buffers/Multiplexers either drive clock input signals directly onto a clock line (BUFG) or optionally provide a multiplexer to switch between two unrelated, possibly asynchronous clock signals (BUFGMUX). Each BUFGMUX element, shown in Figure 4.10, is a 2-to-1 multiplexer. The select line, S, chooses which of the two inputs, I0 or I1, drives the BUFGMUX's output signal, O. The switching from one clock to the other is glitch-less, and done in such a way that the output High and Low transition times are never shorter than the shortest High or Low time of either input clock. The two clock inputs can be asynchronous with regard to each other, and the S input can change at any time, except for a short setup time prior to the rising edge of the presently selected clock (I0 or I1).

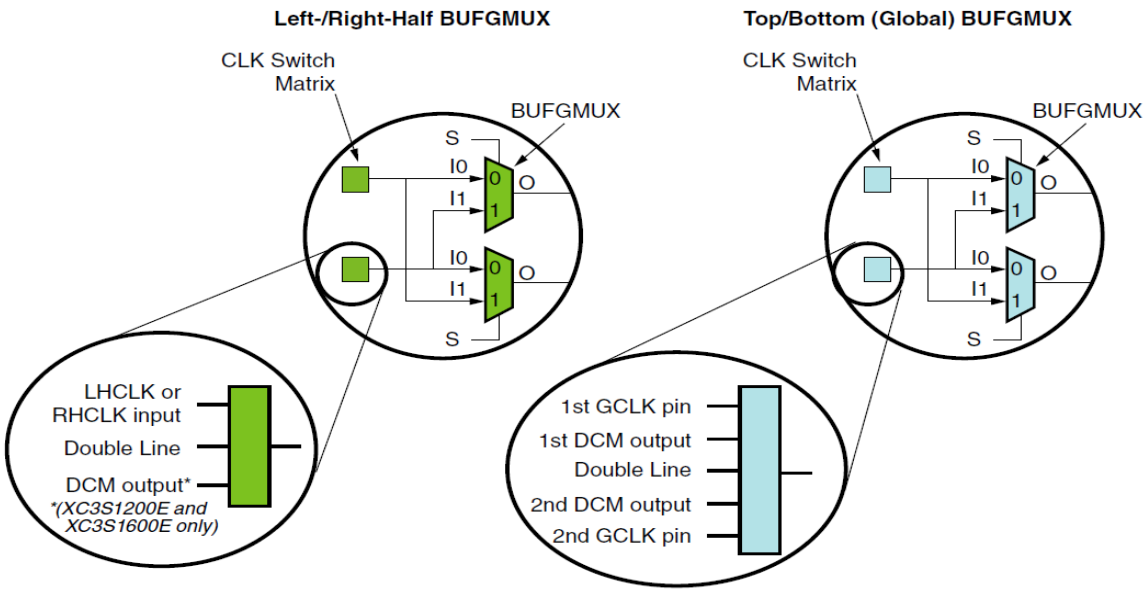


Figure 4.11: Internal Element of 2 – to -1 Multiplexer (Courtesy of Xilinx™ Co.)

The BUFG clock buffer primitive drives a single clock signals onto the clock network and is essentially the same element as a BUFGMUX, just without the clock select mechanism. Similarly, the BUFGCE primitive creates an enabled clock buffer using the BUFGMUX select mechanism. The I0 and I1 inputs to BUFGMUX element originate from clock input pins, DCMs, or Double-Line interconnect, as shown in above Figure 4.11. As depicted in Figure 4.10, there are total 24 BUFGMUX elements, which are distributed around the four edges of the device. Clock signals from the four BUFGMUX elements at the top edge and the four at the bottom edge are truly global and connect to all clocking quadrants. The eight left-edge BUFGMUX elements only connect to the two clock quadrants in the left half of the device. Similarly, the eight right-edge BUFGMUX elements only connect to the right half of the device. BUFGMUX elements are organized in pairs and share I0 and I1 connections with adjacent BUFGMUX elements from a common clock switch matrix as shown in Figure 4.11. For example, the input on I0 of one BUFGMUX is also a shared input to I1 of the adjacent BUFGMUX. The clock switch matrix for the left- and right-edge BUFGMUX elements receives signals from any of the three following sources: an LHCLK or RHCLK pin as appropriate, a Double-Line interconnect, or a DCM in the XC3S1200E and XC3S1600E devices.

By contrast, the clock switch matrixes on the top and bottom edges receive signals from any of the five following sources: two GCLK pins, two DCM outputs, or one Double-Line interconnect. The four BUFGMUX elements on the top edge are paired together and share

inputs from the eight global clock inputs along the top edge. Each BUFGMUX pair connects to four of the eight global clock inputs, as shown in Figure 4.10. This optionally allows differential inputs to the global clock inputs without wasting a BUFGMUX element. The connections for the bottom-edge BUFGMUX elements are similar to the top-edge connections (see Figure 4.11). On the left and right edges, only two clock inputs feed each pair of BUFGMUX elements.

The clock routing within the FPGA is quadrant-based, as shown in Figure 4.10. Each clock quadrant supports eight total clock signals, labelled 'A' through 'H' as shown in Figure 4.12. The clock source for an individual clock line originates either from a global BUFGMUX element along the top and bottom edges or from a BUFGMUX element along the associated edge, as shown in Figure 4.10. The clock lines provide the synchronous resource elements (CLBs, IOBs, Block RAM, multipliers and DCMs) within the quadrant. The four quadrants of the device are: Top Right (TR), Bottom Right (BR), Bottom Left (BL) and Top Left (TL).

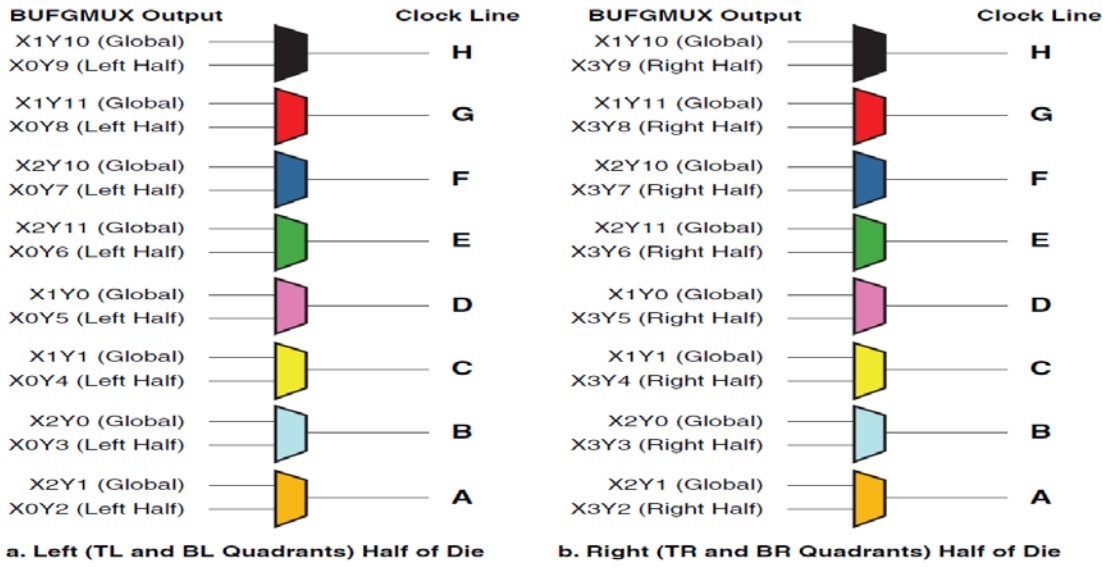


Figure 4.12: Quadrant – Based Clock Routing (Courtesy of Xilinx™ Co.)

The outputs of the top or bottom BUFGMUX elements connect to two vertical spines, each comprising four vertical clock lines as shown in Figure 4.10. At the centre of the die, these clock signals connect to the eight-line horizontal clock spine. Outputs of the left and right BUFGMUX elements are routed onto the left or right horizontal spines, each comprising eight horizontal clock lines. Each of the eight clock signals in a clock quadrant derives either from a global clock signal or a half clock signal. In other words, there are up to 24 total potential clock inputs to the FPGA, eight of which can connect to clocked elements in a

single clock quadrant. Figure 4.12 shows how the clock lines in each quadrant are selected from associated BUFGMUX sources. For example, if quadrant clock 'A' in the bottom left (BL) quadrant originates from BUFGMUX_X2Y1, then the clock signal from BUFGMUX_X0Y2 is unavailable in the bottom left quadrant. However, the top left (TL) quadrant clock 'A' can still solely uses the output from either BUFGMUX_X2Y1 or BUFGMUX_X0Y2 as the source. To minimize the dynamic power dissipation of the clock network, the Xilinx development software automatically disables all clock segments which are not in use.

4.7 Data Forwarding and Data Dependency

This unit is responsible for proper data forwarding to ALU and multiplier. The primary function of this unit is to compare the destination register address of the data waiting in the EX and WB pipeline registers to be written back to the register file with the current data needed by the ALU or multiplier and forward the most up-to-date data to these units. By forwarding the data at the appropriate time, this unit makes sure that the pipeline works smoothly and does not stall as a result of data dependencies [82]. All pipeline registers shown below are involved to have smooth flow through different stages of pipeline

Table 4.3: Pipeline Registers Flow Through Different Stages of Pipeline

IF Stage	DC Stage	EX Stage	WB Stage
Zerowrite	Zerowrite	Zerowrite1	
Regwrite	Regwrite	Regwrite1	Regwrite2
Ra(3 : 0)	Ra(3 : 0)	Ra1(3 : 0)	Ra2(3 : 0)
Rb(3 : 0)	Rb(3 : 0)	Rb1(3 : 0)	
Load	Load	Load1	
Store	Store	Store1	
Op(5 : 0)	Op(5 : 0)	Op1(5 : 0)	
A_depen	A_depen	A_depen1	
B_depen	B_depen	B_depen1	
Immed	Immed	Immed1	
Immed_data(31 : 0)	Immed_data(31 : 0)		

Hazards are the situations in which a pipeline may produce wrong answers by providing incorrect data. The remedy to this is instructions stalls and/or data forwarding to be used to overcome such pipeline hazards. Hazard detection unit detects such scenario in which correct data forwarding is not possible and stalls the pipeline for one or two clock cycles in order to assure that instructions are executed with the correct data set. If it found that a stall is necessary, it disables the operation in the instruction decode pipeline registers, stops the program counter from incrementing, and clears all the control signals generated by the control unit. There are mainly three types of hazards discussed below.

4.7.1 Structural Hazards

Structural Hazards occur when more than one instruction attempt simultaneously to make use of same resources or wrong inputs are given to hardware. Specifically, branch instructions could make use of the same ALU to figure out the target branch address. If the ALU were to use in the decode stage for the same purpose, an ALU related instructions followed by a branch would have seen that both the types of instructions tries to use the ALU concurrently. This conflict is resolved by designing a specialized branch target adder into decode stage. Float value given to integer instruction is also a structural hazard.

4.7.2 Data Hazards

These types of Hazards observed when an instruction scheduled blindly and if it tries to make use of data before the data is actually available in the register file. Data dependencies are mainly of three types for which there could be three types of possibilities for data hazards.

1) Read after Write:

Suppose instruction 1 writes a value which is used later by the next instruction 2. Instruction 1 must come first otherwise instruction 2 will read the older value instead of the newer one. For example, Instruction J tries to read operand before Instruction I writes it. Caused by Dependences occur in compiler nomenclature.

I: add r1,r2,r3

J: sub r4,r1,r3

2) Write after Read:

Suppose the situation is such that an instruction 1 reads a location which is to be later overwritten by instruction 2. It is mandatorily required that the instruction must come first, or it should read the newer updated value instead of the previous one. For example, Instruction J writes operand before Instruction I read it which is called an anti dependence by compiler writers. This results because of reusing the name r1.

I: sub r4,r1,r3
J: add r1,r2,r3
K: mul r6,r1,r7

3) Write after Write:

Consider two instructions and both of which tries to write the same location. It is mandatory requirement that instructions must take place as per their original order. For example, Instruction J writes operand before the instruction I writes it, Called an output dependence by compiler writers. This also results due to repetition of name r1.

I: sub r1,r4,r3
J: add r1,r2,r3
K: mul r6,r1,r7

4.7.3 Control Hazards

Control Hazards take place because of conditional and unconditional branching and jumps. The pipeline structure based on RISC provides solution for branches in the Decode stage, in which the branch resolution and reoccurrences are two cycles long. For any branch taken, the instruction which is immediately to be executed after the branch should always be fetched from instruction memory. If this instruction is being ignored, there will be an introduction of one cycle per taken branch as penalty, which is quite large. There are four techniques used to solve this performance problem with branches which are named as (1) Predict not taken; (2) Branch likely; (3) Branch delay slot and (4) Branch prediction.

In the proposed processor design we have used branch delay slot scheme to prevent control hazard. After every branch instruction a NOP instruction is there to provide a delay slot in the pipeline. Also this processor as being a fixed pipeline processor has only read after write data dependency. This dependency has been identified in the decode stage and data forwarding is

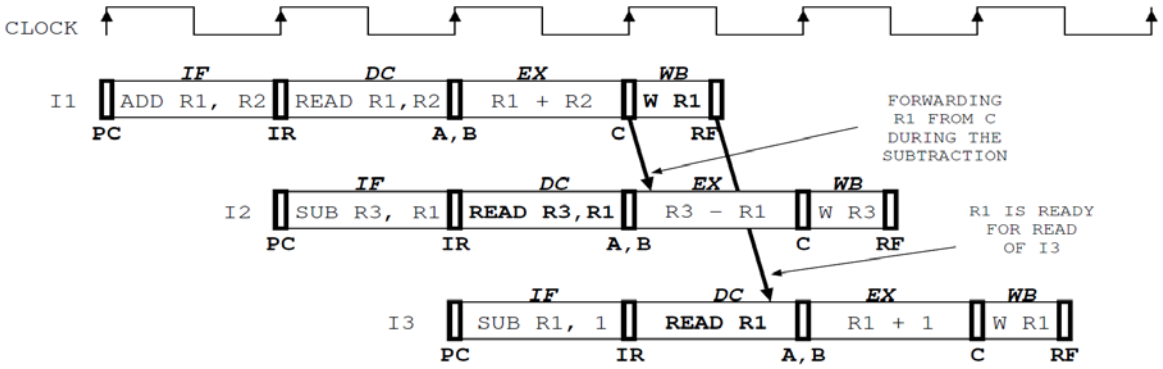


Figure 4.13: Data Dependency and Data Forwarding

done from write back stage to execution stage to eliminate this hazard. The data dependency and data forwarding are explained by Figure 4.13.

4.8 External Interface

This processor is given two signals as an external input signals which are Clk (System Clock) and reset_n (Active high Asynchronous Reset) and the interface is as shown in Figure 4.14.

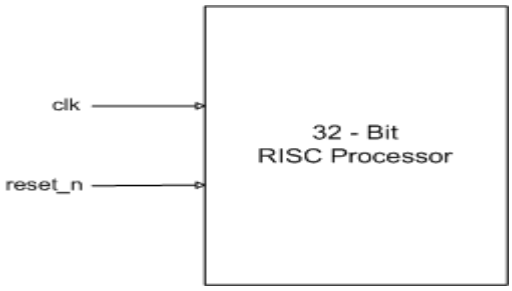


Figure 4.14: External Interface

4.9 Instruction Simulation and Verification

All the instructions supported by the processor are simulated using VHDL coding and verified through the generation of waveforms. Thus the performance of the processor as whole system has been checked, tested and verified by simulating variety of instructions. The overall performance of the CPU is discussed in detailed in the Chapter 5.

4.10 FPGA Design Flow

Whole system has been developed by using the VHDL and then implemented into the FPGA. A typical FPGA design flow has been followed and each and every step of the flow is shown in the following Figure 4.15.

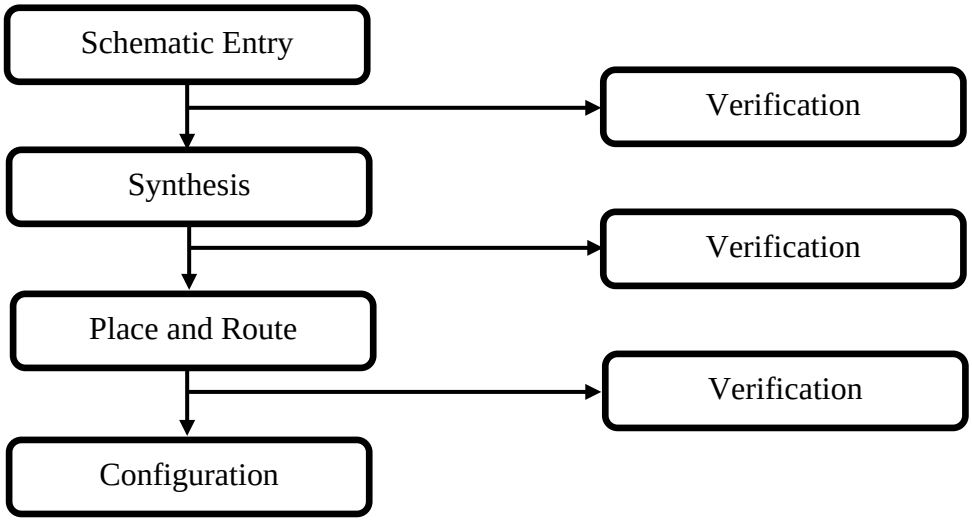


Figure 4.15: FPGA Design Flow

Schematic Entry: The design is entered into a synthesis design system using a hardware description language. The language used for this research work was VHDL and editor used for this purpose is one which is provided by Xilinx Integrated Environment (ISE Version 13.1).

Synthesis : A netlist is generated using VHDL code and Xilinx synthesis tool.

Place and Route: The place process decides the best location of the cells, the best routing strategy for the given design and for desired performance. The route process makes the connections between the cells and the blocks. This process is done by using Xilinx ISE.

Configuration: Creates the PLD configuration data and downloads the configuration data to the PLD (FPGA) and enables the configuration on the PLD to be verified for correctness.

Verification: At each and every step of the design process, the processor architecture has been verified using the software simulation. ModelSim XE – II has been used for simulating the VHDL coding.

4.11 Summary of Synthesis Report

Whole architecture has been synthesized for FPGA implementation using the Xilinx ISE – 13.1 suit and the summary of synthesis report has been presentation in Table 4.4.

Table 4.4: Summary of Synthesis Report

Device Utilization Summary			
Selected Device : XC3s500efg320 – 4 (SPARTAN – 3E FPGA)			
Devices	Device Utilization		
	Devices Used	Out of	% Utilization
Number of Slices	2017	4656	43%
Number of Slice Flip Flops	697	9312	7%
Number of 4 Input LUTs	3066	9312	32%
Number of bonded IOBs	2	232	0%
Number of BRAMs	9	20	45%
Number of MULT18X18SIOs	1	20	5%
Number of GCLKs	1	24	4%
Number of IOs	2		
Timing Summary	Speed Grade	Minimum Period	Maximum Frequency
	4	23.934ns	41.782 MHz

4.12 Power Estimation Reports of Complete Architecture

The XPower Estimator (XPE) – 11.1spreadsheet is a power estimation tool typically used in the pre-design and pre-implementation phases of a project. XPE assists with architecture evaluation, device selection, appropriate power supply components and thermal management components specific for the targeted application. XPE considers design's resource usage, toggle rates, I/O loading, and many other factors which it combines with the device models to calculate the estimated power distribution. The accuracy of XPE is dependent on two primary sets of inputs (1) Device utilization, component configuration, clock, enable, and toggle rates, and other information related to project entered into the tool and (2) Device data models integrated into the tool. It is a pre-implementation tool to be used in the early stages of a design cycle or when the RTL description is incomplete. After implementation, the XPower Analyzer (XPA) tool (available in the ISE® Design Suite software) can be used for more accurate estimations and power analysis.

The Summary sheet shown in Figure 4.16 is the default sheet on launch and allows designer to enter all device and environment settings. On this sheet the tool also reports estimated power; rail-wise and block-wise, so one can quickly review thermal and supply power

distribution required within the design. It also gives summary related to clock, logic, IO, BRAM, DCM and MUL. Important factors in dynamic power calculation are the activity and the load capacitance that needs to be switched by each net in the design. Some of the factors in determining the loading capacitance are fanout, wire length, etc. With clocks typically having higher activity and fanouts, the power associated with clock nets can be significant and also can be reported in a separate worksheet sheet along with many other reports. The consolidated summary of estimation report for the design under consideration is shown in Figure 4.16.

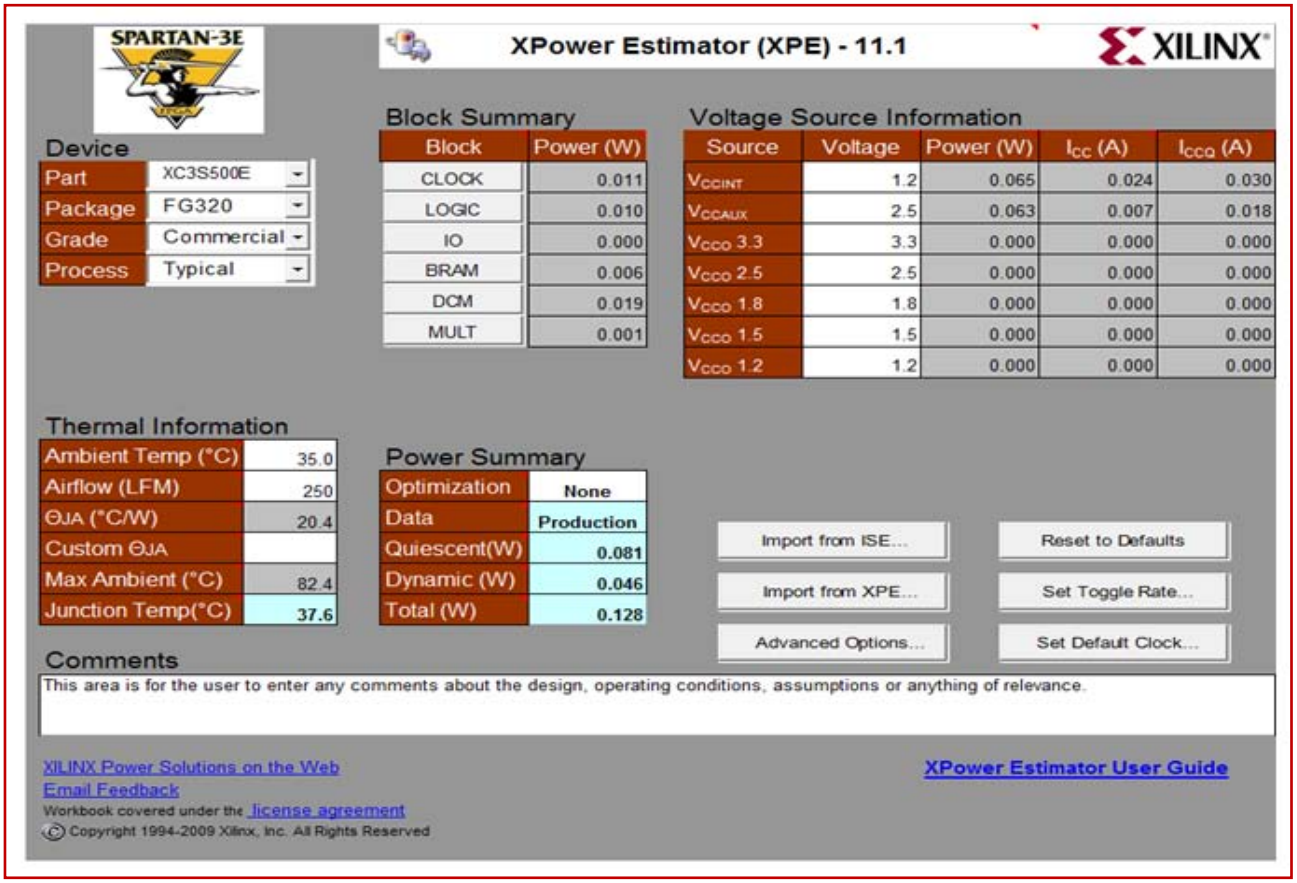


Figure 4.16: Summary of Estimated Power Distribution Report

Clock Fanout Column is the number of synchronous elements driven by this clock and for this design the maximum achievable frequency is 54.7 Mhz and clk fanout is 668. Using XPower Estimator – 11.1 one can have the separate sheets for each components such as Logic sheet is used to estimate the power consumed in the CLB resources. The estimated power accounts for both the logic components and the routing. Here two types of information are to be entered (1) Utilization – Enter the number of LUTs, Shift Registers and LUT-based

RAMs and ROMs and (2) Activity – Enter the Clock domain as per the logic and then enter the Toggle Rate the logic is expected to switch and the Average Fanout.

Note: The default setting for Toggle Rate (12.5%) and Average Fanout (3) are based on an average extracted from a suite of customer designs. In the absence of a better estimation for a specific design, Xilinx recommends to use the default setting provided in Xpower Estimator. In our design total no of flip flops are 659 and LUT's are 3423 used.

With high switching speed and capacitive load, there will be major contribution to the total power consumption of an FPGA is from switching I/O power. Because of this, it is important to accurately define all I/O related parameters. Using the I/O sheet, the XPE helps to calculate the on-chip and, eventually, off-chip power for the system I/O interfaces. There are three main types of information entered on the I/O sheet (1) IO Settings, (2) Activity and (3) bank voltage levels and voltage standards. In this design only two input pins of the processor are used, which are the clock and the reset pin. FPGA devices have dedicated block RAM resources. The details about the Enable Rate and Write Rate columns used in the Block RAM sheet can be described as (1) Use the Enable Rate to specify the percentage of time each block RAM's ports are enabled for reading and/or writing. To save power, the RAM enable can be driven low on clock cycles when the block RAM is not used in the design. BRAM

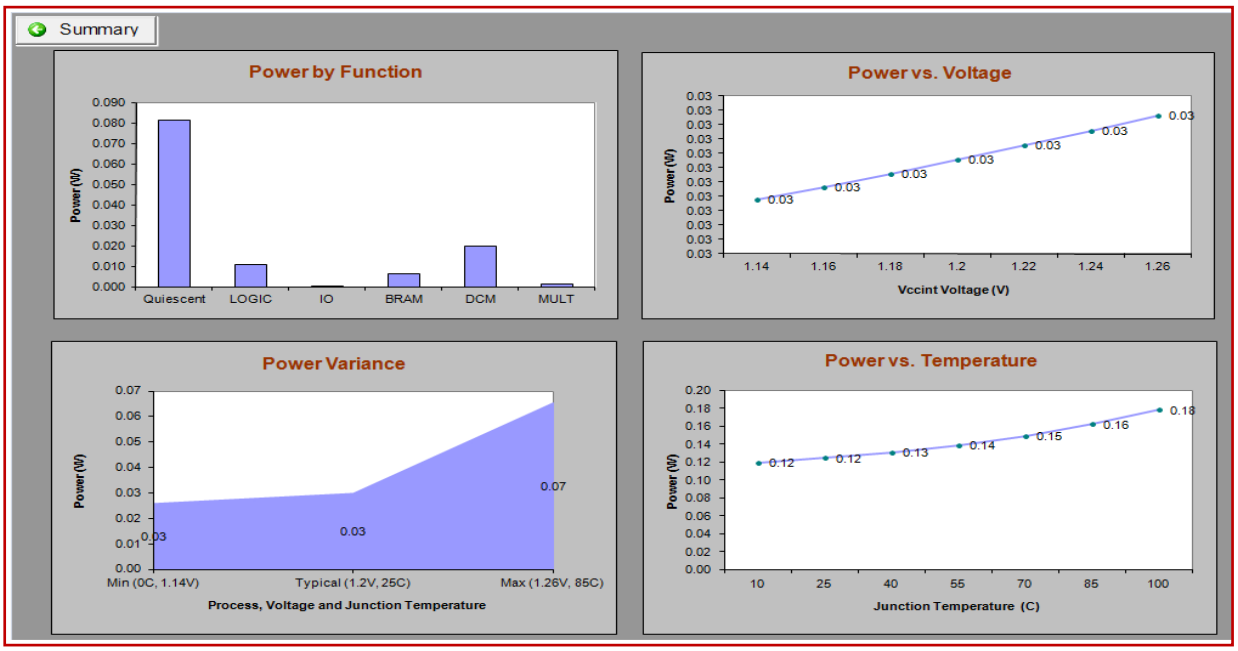


Figure 4.17: Graphical Representation of Estimated Power Requirements for Internal Modules and Effect of Various Parameters on Power Consumption

Enable Rate, together with Clock rate is important parameters that must be considered for

power optimization. (2) The Write Rate represents the percentage of time that each block RAM port performs write operations. The read rate is understood to be 100% – write rate. We have used in all 9 block RAMs of 18K: 4 for ROM and 5 for RAM and only 1 Digital Clock Manage (DCM) is used as there is only one clock source and also present its related electrical parameters and one multiplier is used as there is only one multiplication instruction. Graphs are plotted according to the data given as input and are well described in Figure 4.17, which represents the need of power of each individual module within the system and also provides the information for power need with reference to junction temperature and the voltage supplied to the device. It is noted that power consumption increases with the junction temperature and the increase in the voltage as well.

4.13 Conclusion

This chapter provides the complete details of construction of 4 – stage pipeline processor architecture and also discusses the types of instruction and their construction along with the detailed formation of instruction.

Processor's architecture design and its instruction formation are thoroughly tested using number of programs and validated through the simulated results in terms of waveform to solidify the design and observed that the correct functionality is achieved and the simulated waveforms are discussed in Chapter 5.

It covers the complete construction of decoder unit and multiplier unit along with many other useful modules such as data forwarding and data hazard detection unit along with the types of various data hazards and provides the remedies for the same.

We also included the complete clock distribution network and the structure of multiplier unit along with the details of all the related signals, as the system is targeted to be implemented up on SPARTAN – 3E FPGA.

At last the power estimation is done using Xpower Estimator -11.1, which will be used for power comparisons to be done in the next chapter.

The following chapter describes the conventional 5 – stage CPU, its power analysis and then its conversion to 4 – stage CPU along with its power analysis and suggests the newly developed power reduction techniques such as Memory Access Stage Removal, Resource Sharing, RAM Addressing Scheme and Clock Gating in order to reduce the dynamic power consumption and are implemented successfully on FPGA. Also the outcomes are analysed and verified in order to lower the overall system power consumption.

These techniques are applied at the hardware design level and analyzed for power requirements. Also the power consumption comparison between conventional 5 – stages CPU with the newly developed low power CPU with 4 – pipeline stages has been made by generating the power reports.

Chapter 5
Proposed Strategies for Power Optimization

5.1 Introduction

The proposed work suggested four out of many possible strategies; and all are implemented at hardware level for dynamic power reduction on the 32 – bit, 4- stage pipelined CPU whose architecture is discussed in previous Chapter, and the power comparisons are made with the conventional 5 – stage pipelined conventional CPU by taking the power reports for both the CPUs using Xilinx Xpower Analyzer. This chapter also discusses the 5- stage pipeline processor architecture in brief (for understanding purpose only) and using this processor the power comparisons are made.

With the prime focus to control the dynamic power along with maintaining the performance of the processor; we are implementing four strategies out of many other possible strategies upon the conventional 5 – stage CPU; which are (1) Memory stage Access Stage Removal, (2) Resource Sharing in 4 – stage CPU, (3) Novel RAM Addressing Scheme and (4) Clock Gating. After implementation of each technique the system is analysed for the power dissipation and at last the power comparisons with and without the implementation of these power reducing techniques on CPU are made and verified.

5.2 Architecture of 32 – bit 5 – Stage Pipeline (Conventional/Standard) CPU

Following Figure 5.1 describes the detailed architecture of 5- stage pipeline CPU, designed on the basis of RISC principle. This CPU is not described here in detailed as it is used only for the purpose of taking power results and then to apply the power reducing strategies to make it 4 – stage CPU which is discussed in Chapter 4. The power reports are generated using the Xpower Analyser (Xilinx ISE 13.1 Suit) for both the CPUs' to compare the power

consumption and the performance, which will verify the proposed techniques for dynamic power reduction. Here, the conventional architecture of 5 – stage CPU includes one more stage in addition to those of 4 – stage CPU, which are Instruction Fetch (IF), Decode and

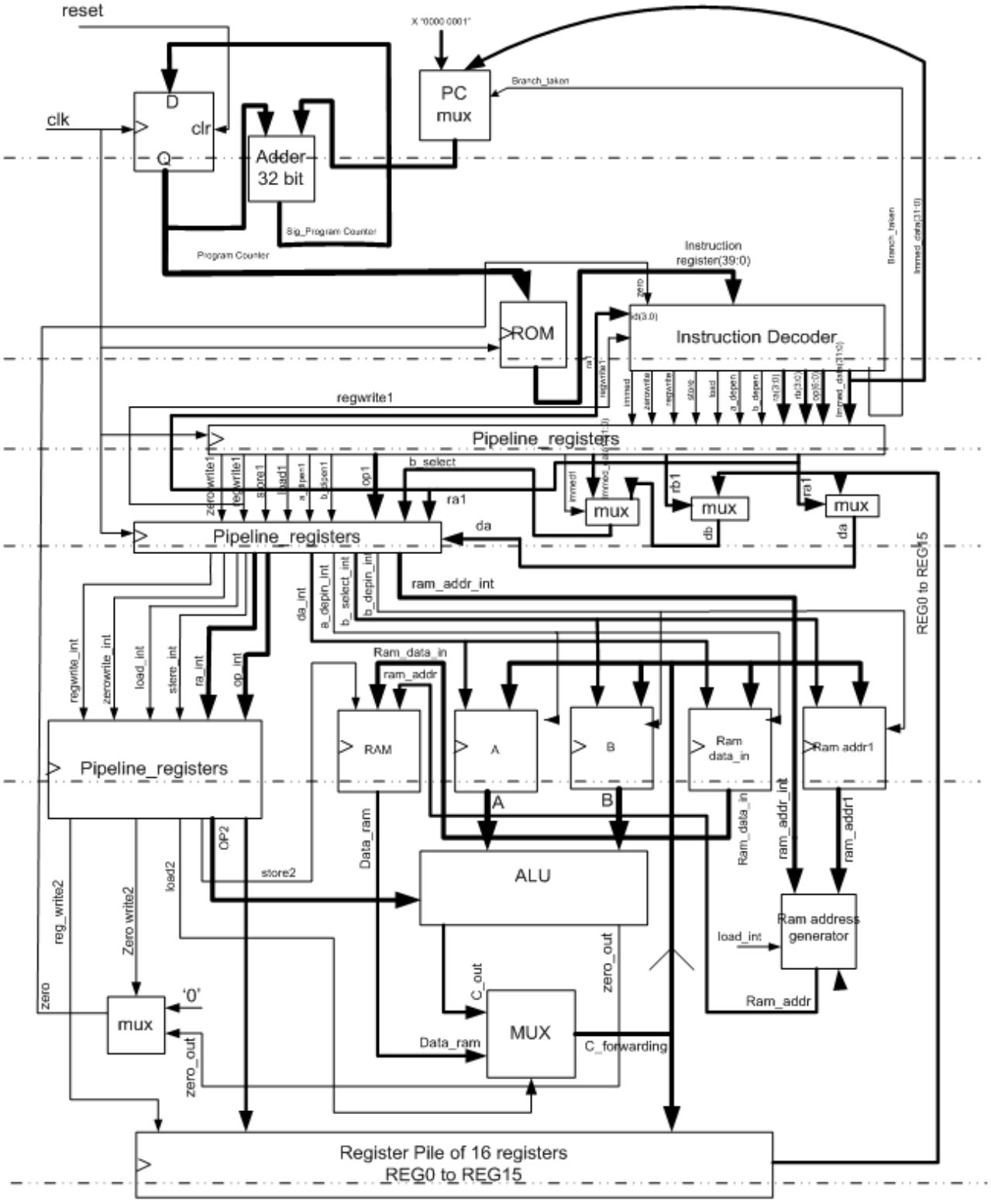


Figure 5.1: Detailed Architecture of 5 – Stage Pipelined Conventional CPU

Operand Fetch (DC), Execution (EX) and an additional one is Memory Access (MEM) and

Write Back (WB). The memory access (MEM) stage is observed additional in 5 – stage CPU. For whole system the power analysis has been done and checked the power requirement using Xpower Analyser of Xilinx ISE – 13.1 suit, the following Figure 5.2 provides the details of power consumption of normal CPU with 5 – stage pipeline structure.

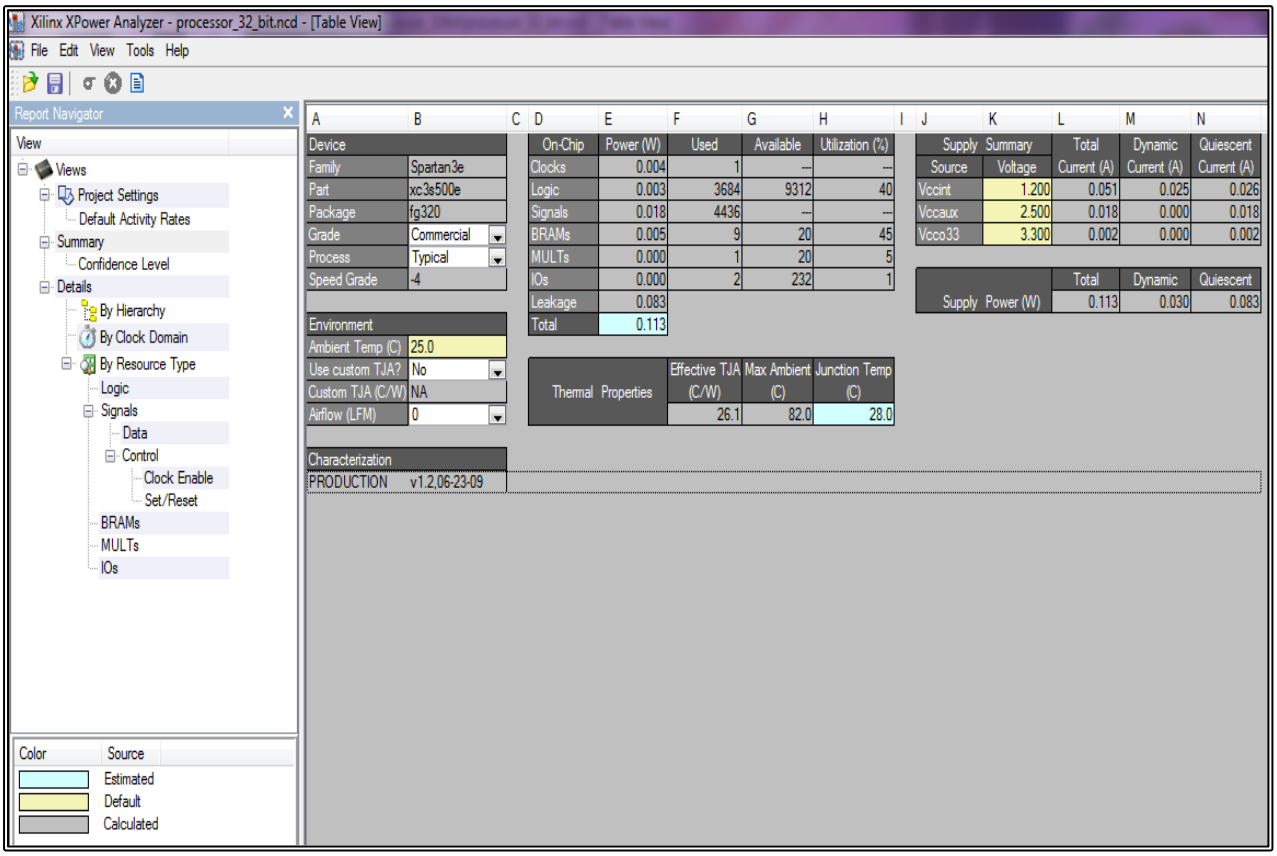


Figure 5.2: Summary of Power Consumption Report for 5 – Stage Pipeline CPU

The performance of above described 5 – stage CPU implementation has been verified for all the pipeline stages through the waveform generation using ModelSim6.5 and it can be represented as shown in following figures from Figure 5.3 to Figure 5.7. These simulated waveforms verifies the CPU performance in terms of execution of all types of instructions and the behaviour of all the related signals in all the pipeline stages.

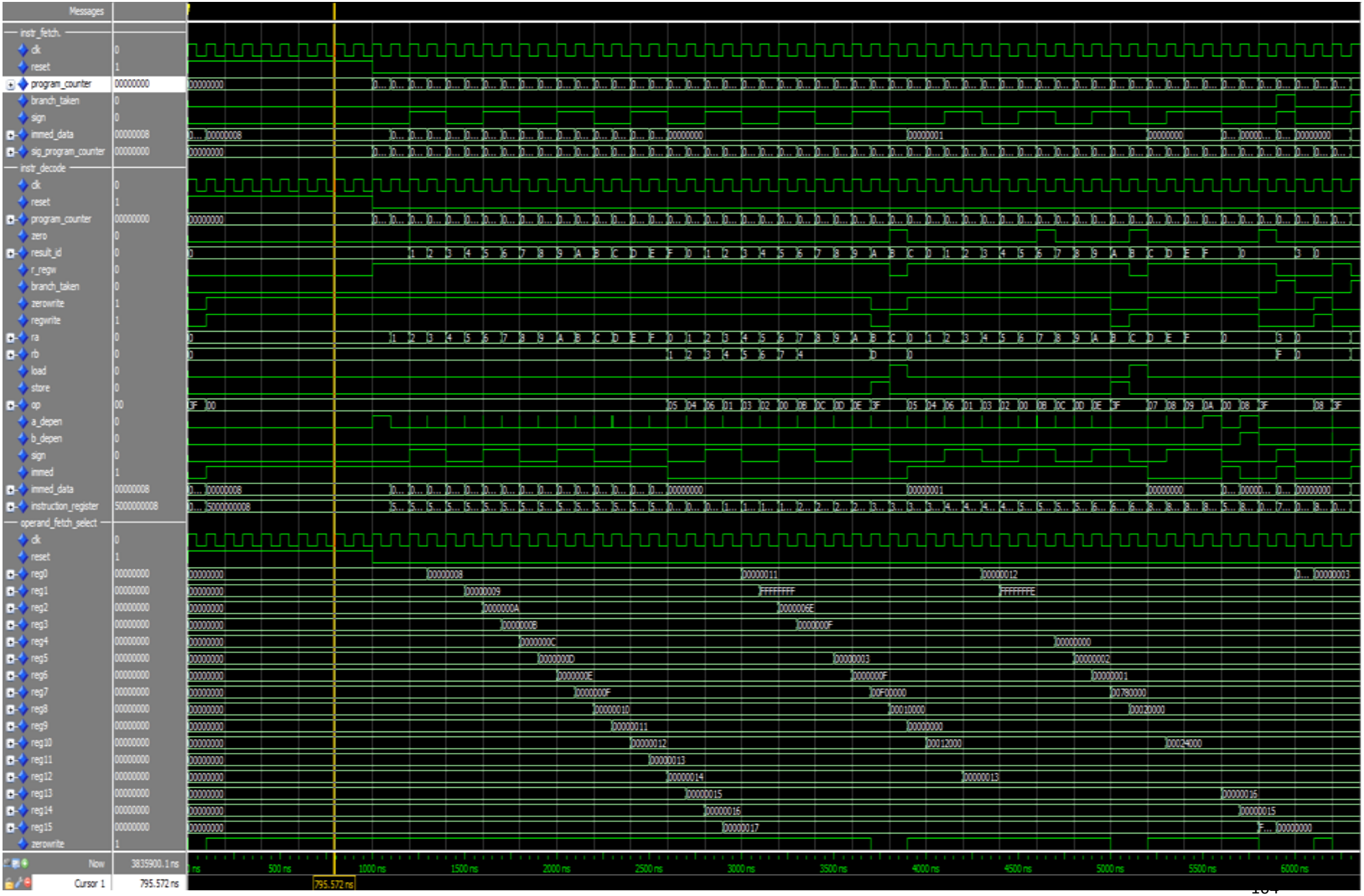


Figure 5.3: Instruction Fetch for 5 - Stage CPU

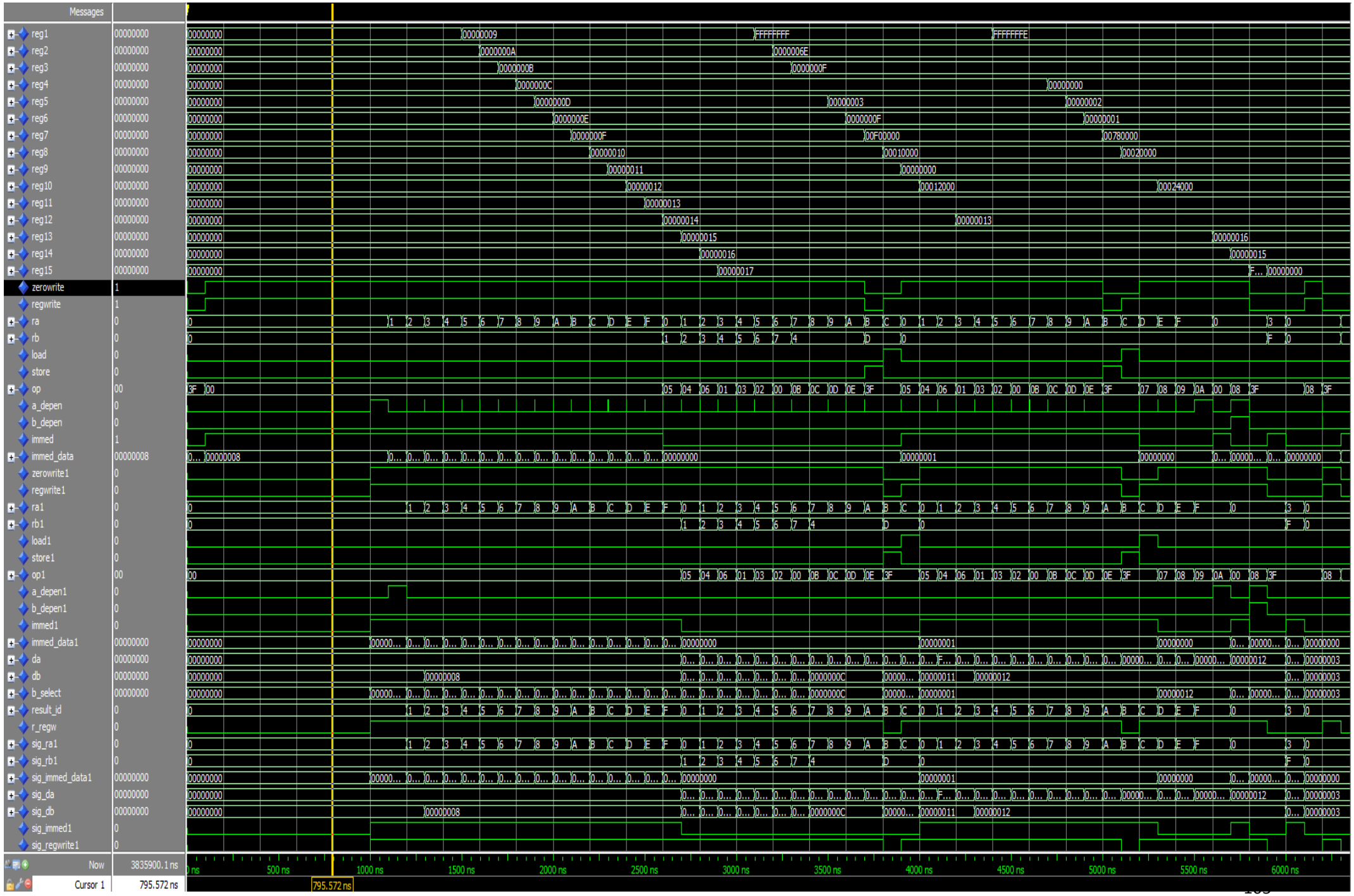


Figure 5.4: Instruction Decode for 5 – Stage CPU

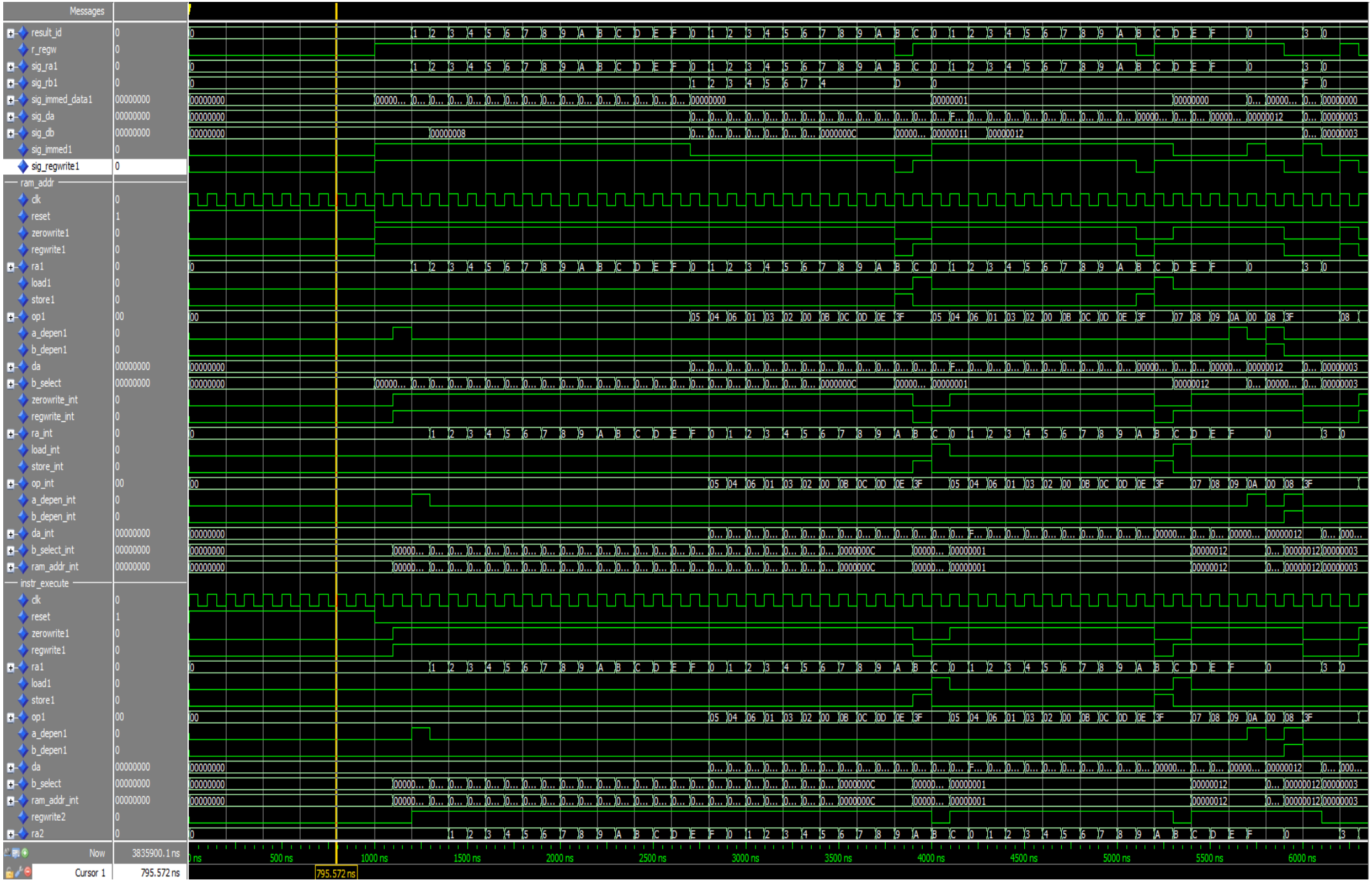


Figure 5.5: RAM Address for 5 – stage CPU

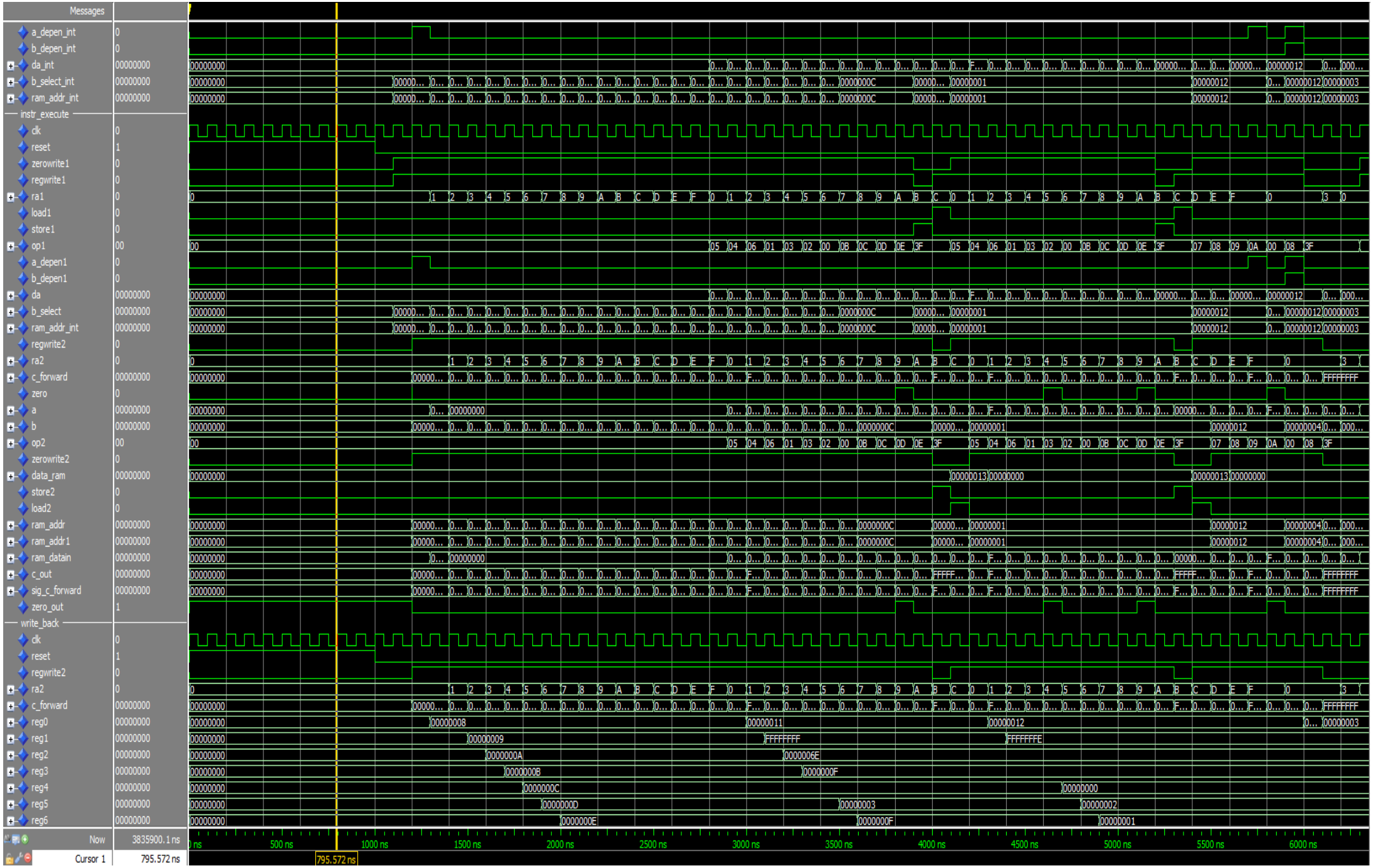


Figure 5.6: Instruction Execute for 5 - Stage CPU

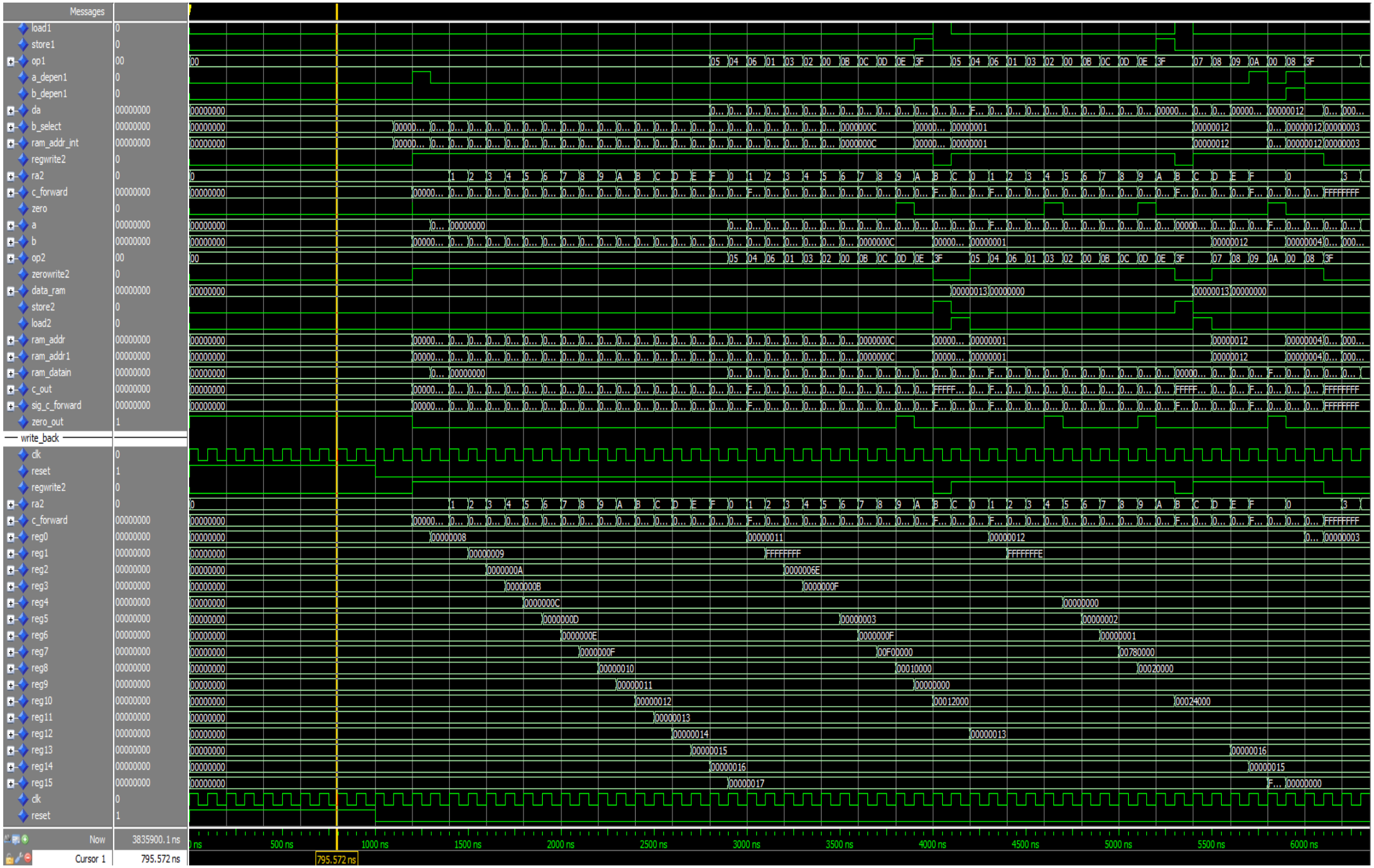


Figure 5.7: Write Back Stage for 5 – Stage CPU

For the Instruction Fetch stage waveform depicted in Figure 5.3 shows the start of processor execution. And also shows that after the reset is de-asserted the program counter starts incrementing. In the Instruction Decode stage described by Figure 5.4, the instructions are decoded and corresponding signals are generated. As shown above waveforms the various relevant signals performs the operations.

The signals such as dependency signals (a_depen and b_depen) for handling hazards, branch signals to indicate a branch, register select for selecting registers, load and store signals for accessing block RAM, immed signal for indicating that current instruction is immediate, zerowrite and regwrite for indicating an update of a register after execution of instruction are involved in this stage.

Here in RAM Address stage in Figure 5.5 the Ram address signals are generated and fed to RAM, so that data can be made available in next clock cycle.

Instruction is executed during the Instruction Execute stage shown in Figure 5.6 and depending on types of instructions, either ALU operation is performed or RAM data loading or storing is done which is described in above waveforms. In Write Back stage detailed by Figure 5.7 the register bank of 16 registers is updated with the instructions.

5.3 Proposed Strategies for Power Reduction

5.3.1 Memory Access Stage Removal Technique

We are using Xilinx FPGA's block RAMs to be utilised for RAM requirements of our system. Any write or storing of data to block RAM requires both address and data to be given at the same clock edge while data to be read or loaded from block RAM requires address to be given first and corresponding data will appear on the data line on the consecutive active clock edge. As data will be available on the next clock edge after issuing the address, one must keep a stage called memory access in the pipeline where the address is provided and data made available during the execute stage and should be stored in the corresponding register. This stage is available in 5- stage CPU.

Even though in 5 stage pipeline the memory access stage is available, it can be seen that this data memory access stage is not used by any of the arithmetic instructions or branch instructions. It is used only for memory access instructions. Arithmetic and branch instructions have a 'NOP' in the memory access stage. That is all its data are just passed-on through memory access and gets executed in the execute stage. Transitions during this unused state cause extra power dissipation. To avoid this wastage, the pipeline is reconfigured to bypass memory stage by using strategic address forwarding logic from instruction decode and operand fetch stage.

Thus address is provided to the RAM at instruction decode and operand fetch stage, if instruction is decoded to be the load instruction. Hence, if the data corresponding to that address is available in the execute stage and no extra clock or memory access stage is required. This address forwarding doesn't affect the other instructions as RAM address generation logic is independent and does not directly or indirectly affect the logic of the consecutive or previous instruction in the pipeline.

For example consider add, load and sub instruction in the pipeline. Now while add instruction is getting executed, load instruction gets decoded and RAM address is provided. If RAM address is dependent on previous instruction the result generated of add instruction is directly given to RAM as address through MUX selection. Now if next sub instruction also has data dependency with the load instruction, then also there is no problem as data will be available in the execute stage which will be forwarded to sub instruction through data forwarding, implemented to remove data hazard. Similarly for any instruction in the pipeline there are no hazards with respect to address forwarding.

Above Figure 5.1 shows the architecture normal five pipeline stages where the memory access is required by the load instruction because address provided to read the memory in execute stage causes the read information to come in next clock cycle that is memory access stage. But for all other instructions these unused transitions do happen for memory access stage without affecting the normal operation; but these transitions during the unused stage cause extra power consumption. To avoid this wastage, the normal pipeline structure is required to modify during the design of architecture of CPU; in a way, so that the correct data forwarding takes place even for the load/store instructions when actually the memory access stage is required.

Thus, the normal 5 – stage pipeline structure is modified to 4 – stage pipeline structure to reduce the dynamic power consumption, whose detailed architecture has been explained in previous chapter in Figure 4.1 and power analysis has been done for this new architecture. It verifies the justifiable reduction in power dissipation. The summary of power analysis is shown in Figure 5.8.

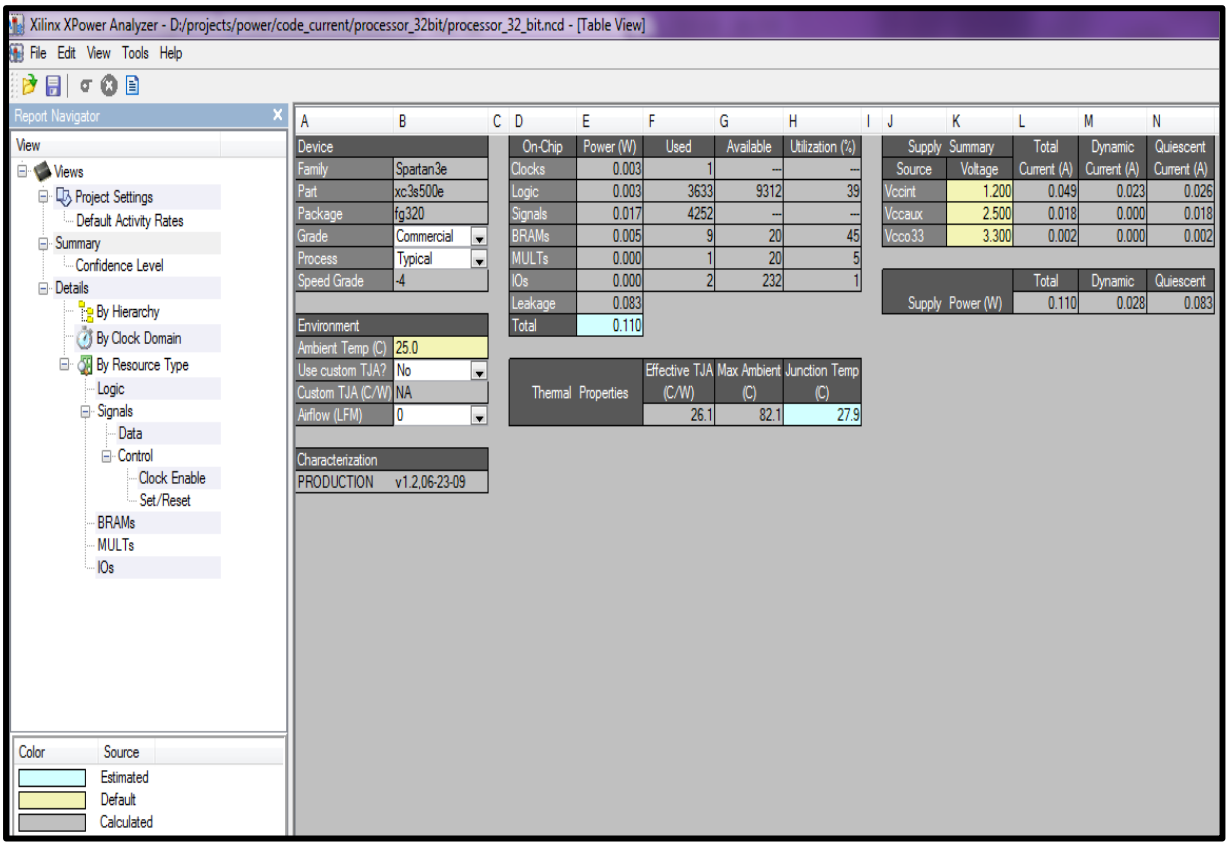


Figure 5.8: Summary of Power Consumption Report for 4 – Stage Pipeline CPU (After Implementation of Memory Access Stage Removal

Comparing the total power consumption requirement of the standard 5 – stage pipeline CPU, the 3mW (2.65%) improvement is achieved due to implementation of memory access stage removal technique.

Up on reduction of one pipeline stage, which was used for memory access, from the standard 5- stage pipeline, it is required to verify the performance of the 4 – stage CPU through the waveforms for all the stages. It is verified and noted that the power reduction is achieved without affecting the performance of the implementation. Explanations for waveforms for all the 4 – stages of CPU given through the Figure 5.9 to Figure 5.12 are same as that of for 5 – stages given above except the address forwarding is done from decode stage for RAM Access related instructions as discussed above.

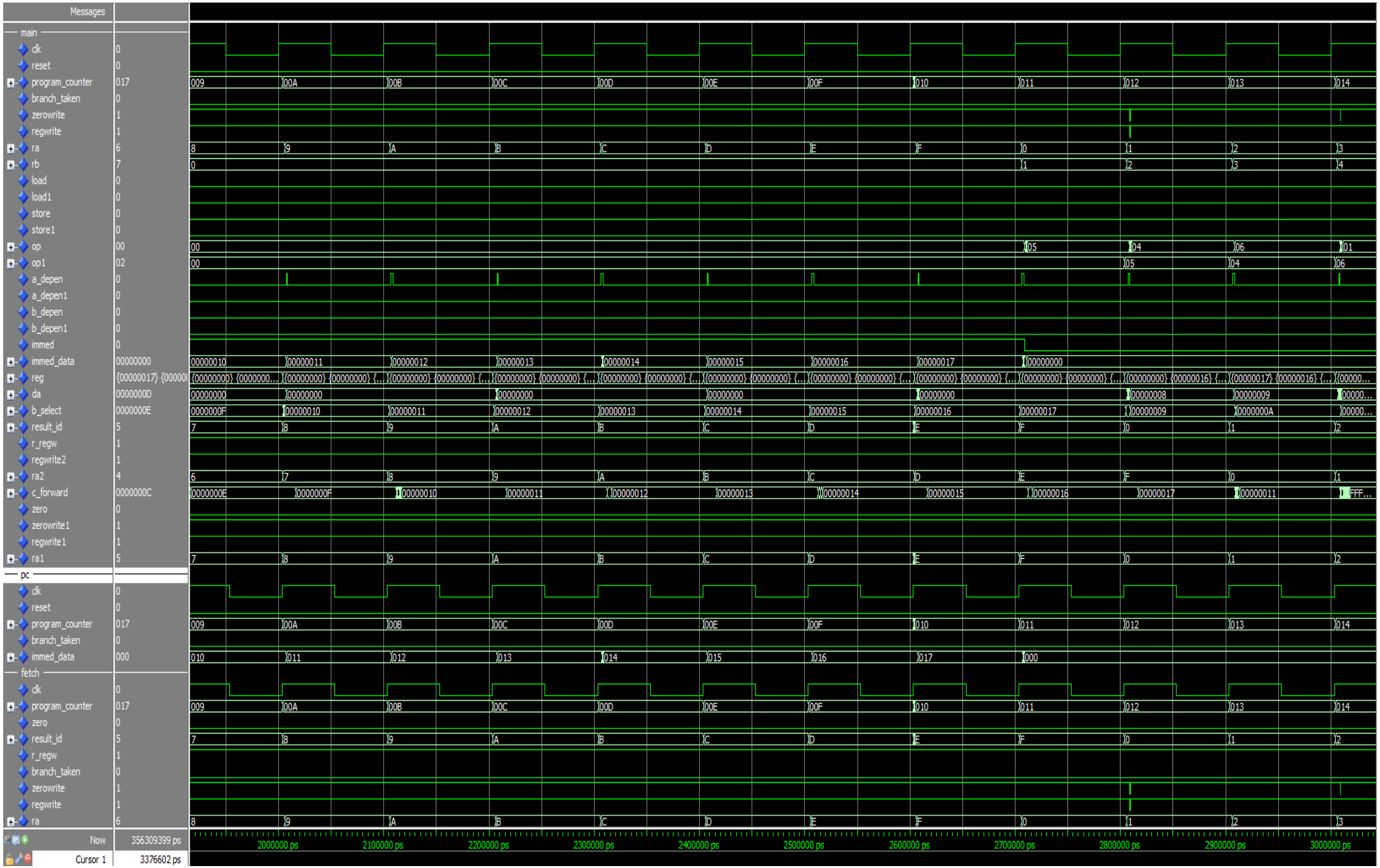


Figure 5.9: Instruction Fetch for 4 – Stage CPU

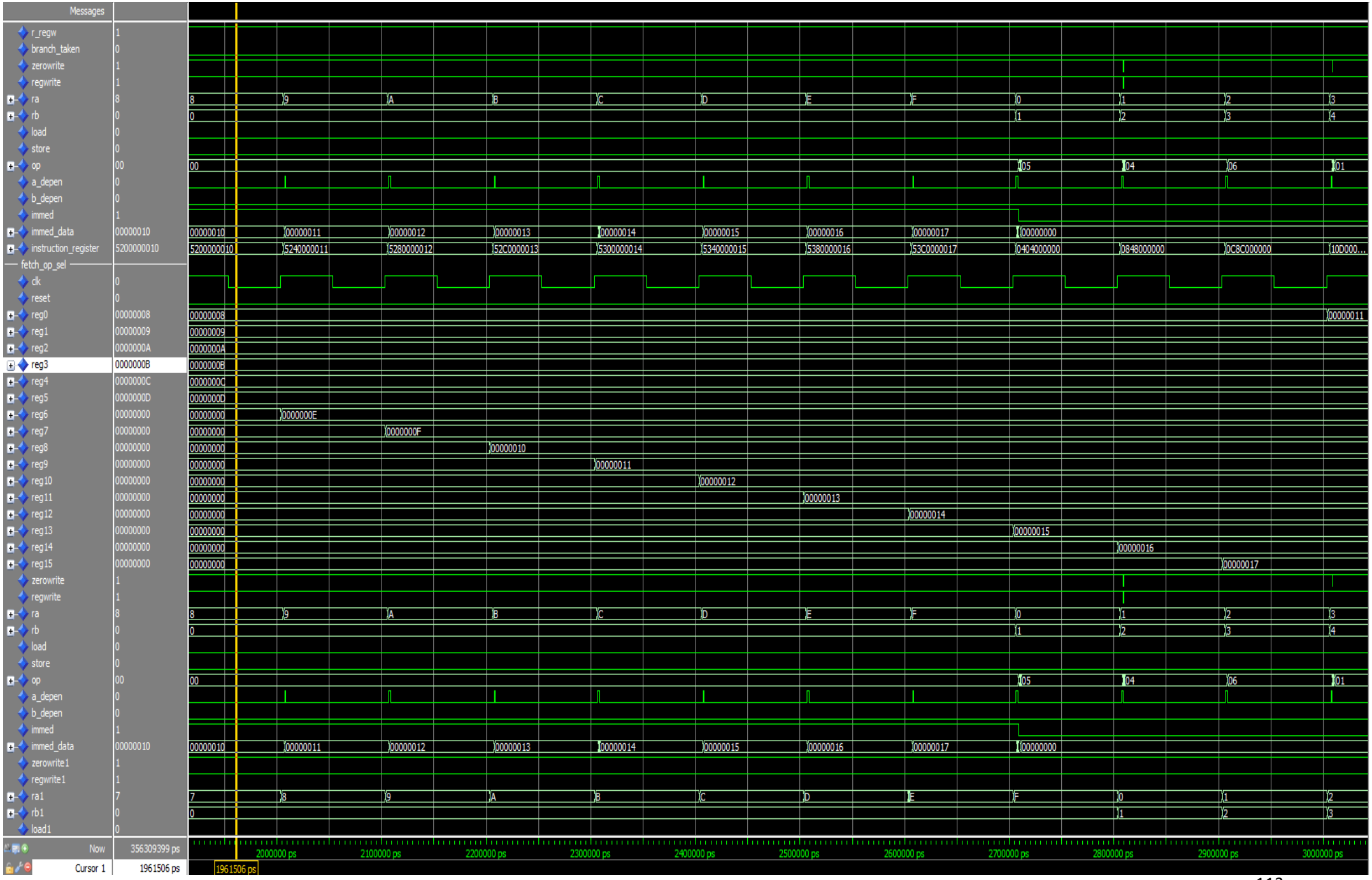


Figure 5.10: Instruction Decode and Operand Fetch for 4 – Stage CPU

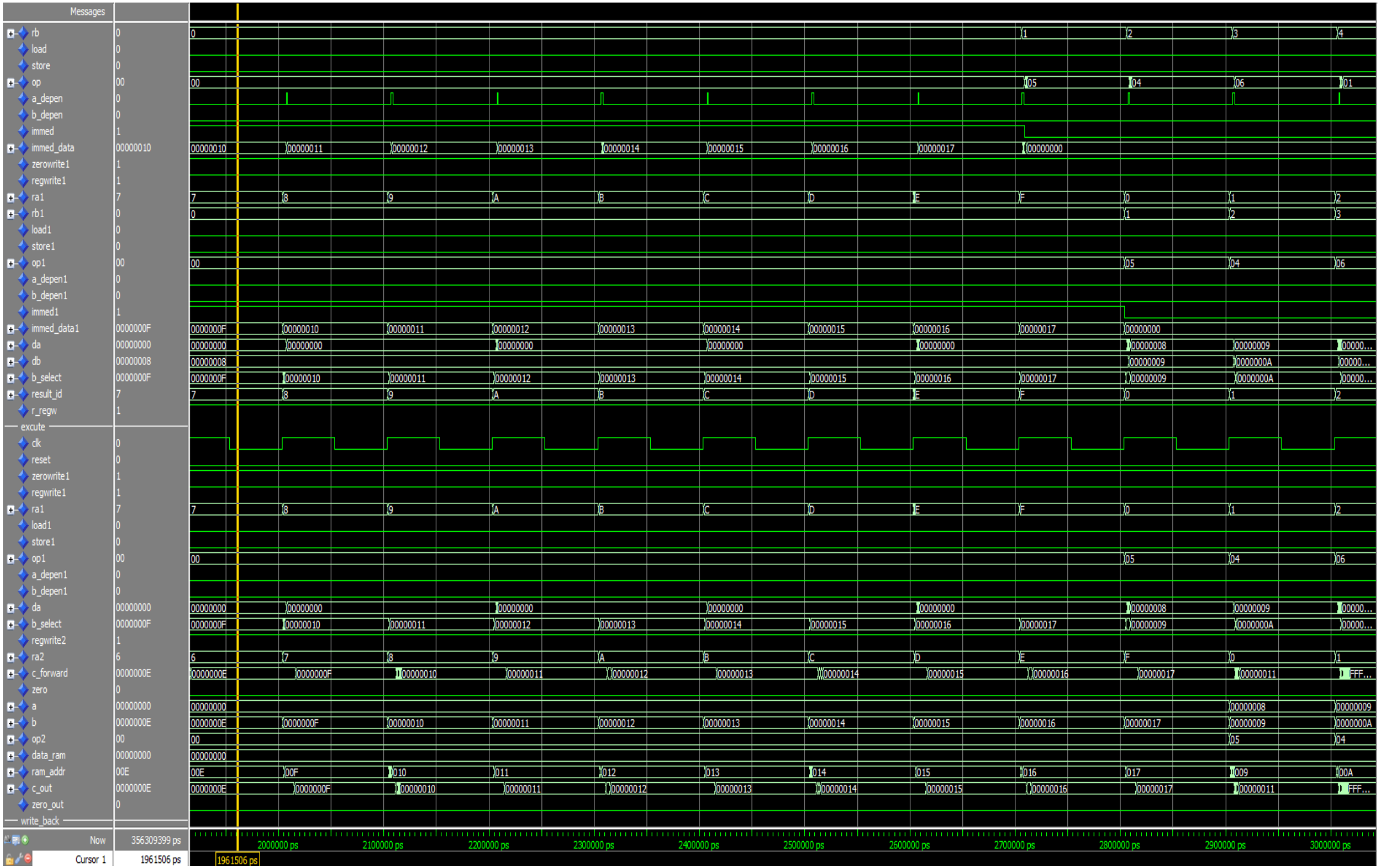


Figure 5.11: Instruction Execute for 4 – Stage CPU

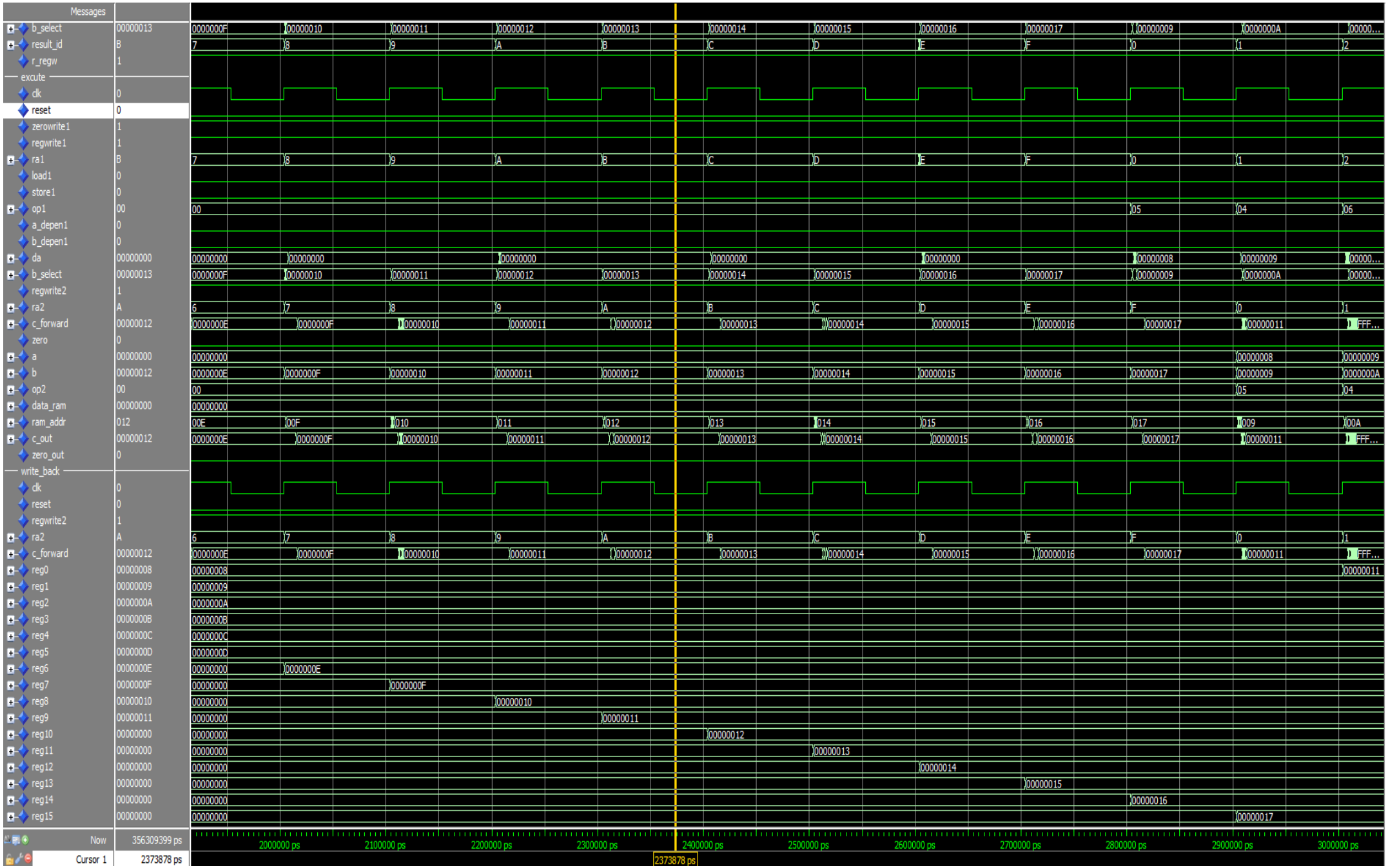


Figure 5.12: Write Back for 4 – Stage CPU

5.3.2 Resource Sharing Technique

The proposed technique is implemented during the designing of the decode stage of the processor. Most of the processors' design supports two types of the instructions addressing modes which are immediate addressing and direct register addressing types. For example, the arithmetic operation addition can be performed by both the types of addressing modes as shown below:

Immediate Addressing Type Operation: Addi r1 , 30 – bit Immediate Data
Direct Addressing Type Operation: Add r1 , r2

Here, both the instructions performs almost the same operation i.e. addition where the operand 1 is common i.e. register r1. The operand 2 can be either immediate data value or a register depending on the type of instruction. To introduce resource sharing technique both the opcode is required to decode to same operation that is for addition Addi the opcode is 001110 and for Add the opcode is 000001 is decoded to perform the addition operation and assigned an opecode which is 000101.

Instead of using the separate resources for both the instructions which perform the same operation, this technique channelize the instruction into one and hence the saving of half the resources is achieved from the decode stage which would have been required if both the instructions were executed differently using their separate resources. Thus in execute stage only one adder does the work for both types of instructions. Thus much amount of power was saved by resource sharing just by adding some control signals in the decode stage and eliminating many flip – flops, comparators and muxes which would have been required in the later stage.

Through Resource sharing following instructions listed in Table 5.1 are channelized to one instruction and the considerable power saving was achieved. This technique can be applied to all the stages with different logic as applied to decode stage here.

Table 5.1: List of Instructions under Consideration

Instruction 1	Instruction 2	Operation	Decoded to Opcode
Add (000001)	Addi (001110)	Addition	Addition (000101)

Sub(000010)	Subi(001111)	Subtraction	Subtraction(000100)
Mul(000011)	Muli(010000	Multiplication	Multiplication(000110)
Or(000100)	Ori(010001)	OR	Orring(000001)
And(000101)	Andi(010010)	AND	Anding(000011)
Xor(000110)	Xori(010011)	XOR	Xorring(000010)
Mov(000111)	Movi(010100)	Move	Move(000000)
Ror(001000)	Rori(010101)	Rotate Right	Rotate Right(001000)
Rol(001001)	Roli(010110)	Rotate Left	Rotate Left(001100)
Srl(001010)	Srli(010111)	Shift Right	Shift Right(001101)
Sll(001011)	Slli(011000)	Shift Left	Shift Left(001110)
Load(001100)	Loadi(011001)	Load	Same Load Signal for Both
Store(001101)	Stori(011010)	Store	Same Store signal for Both

Thus from fetching, executing and write back stages considerable amount of resources are saved. During the instruction fetch operation the Program Counter (PC) is either incremented by 1 or incremented/decremented by the content of branch address from the current position of PC. To perform this operation two adders and one subtractor is required. During the design of processor 2's complement logic was used for instruction fetching during the branching operation, which removes one subtractor.

Further, the resource sharing is also used by implementing 2:1 MUX with inputs of value 1 and 2's complement branch address and a branch signal as select line. This MUX output is then added to current PC to generate the next PC value. This reduces further one adder as shown in following Figure 5.13.

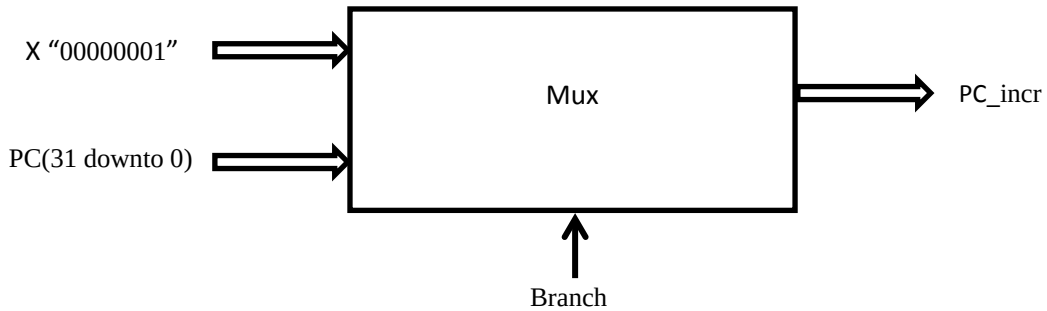


Figure 5.13: MUX for Resource Optimization

The Program Counter operation can be expressed as follows:

PC = PC + PC_incr;

In Execute Stage the ALU required 2 Adders for the operations such as addition and increment i.e. one for addition (Add, Addi) and one for Increment (Inc) and 2 subtractors for subtraction and decrement i.e. one for subtraction(Sub, Subi) and one for Decrement (Dec).
In this proposed work during execute stage instead of using 4 Adders/Subtractors dedicated to each operation one adder and a multiplexer have been implemented during the design of the CPU by which the resource utilization is optimized and the power reduction is achieved. The design and the functionality of the newer logic which is implemented in the CPU can be explained through the following example and the Figure 5.14 explains the implementation logic.

Consider four operations which are to be executed:

(1) Add : a + b; (2) Inc : a + 1;
(3) Sub : a – b ; (4) Dec : a – 1;

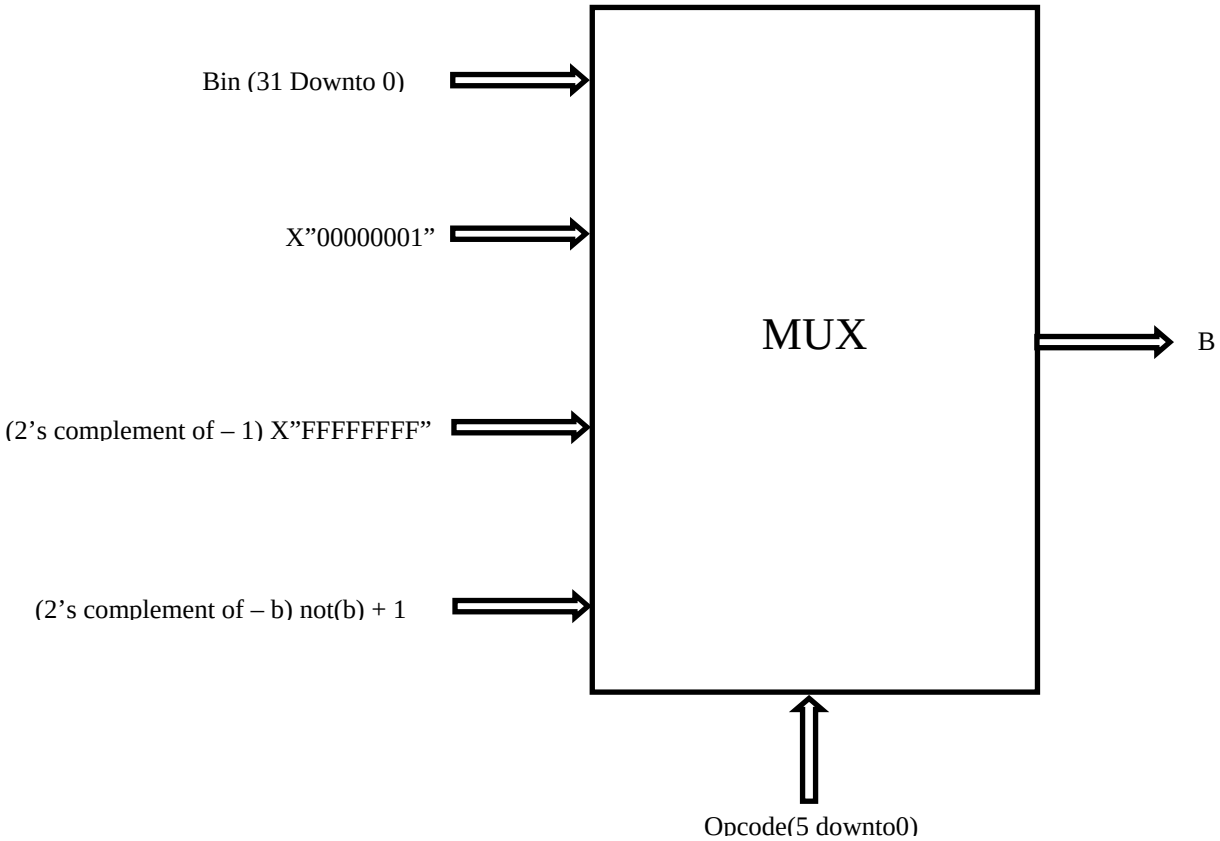


Figure 5.14: MUX with Opcode as Selection Logic

To achieve the resource sharing, B register is given the input value through the 4:1 MUX with the opcode as select line, which is as shown in Figure 5.14. It is concluded after the implementation of this logic that the single adder is used to generate the results and the logic used for all the cases can be presented as C = a + b as operation.

After implementation of this logic and the overall power requirements are analysed for the modified 4- stage CPU and it is observed that 4mW (3.63%) further improvement in the total power consumption is achieved, the screen shot of the summary of power analysis is show in Figure 5.15.

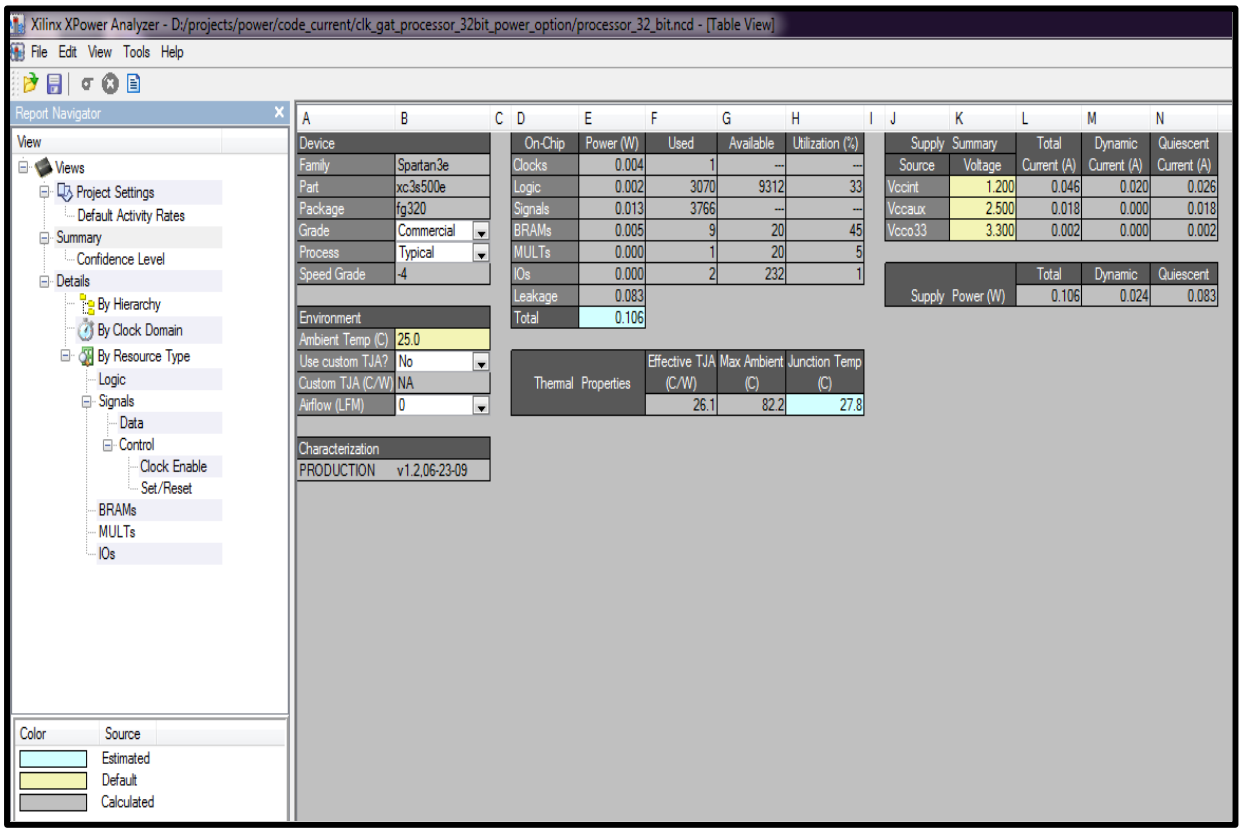


Figure 5.15: Summary of Power Consumption Report for 4 – Stage Pipeline CPU (After Implementation of Resource Sharing Strategy)

5.3.3 Novel RAM Addressing Scheme

Xilinx FPGA's have asynchronous block RAM. We are using it as simple dual port RAM through coregen generator for this architecture which provides synchronous interface to this block RAM as dual port RAM. For write operation write_en is also generated with address and data is fed to the RAM, but for read, only address is provided which causes data to be

read from that memory location. Power will be dissipated for each new address fed to the RAM as it will require appropriate read or write to that memory location addressed by RAM. Thus we see that even though instruction is not a RAM related instruction, RAM address will be generated and corresponding data will be given out by RAM. But as instruction is not a RAM related instruction, this data will not be propagated to write back stage. Thus what was done in this power reduction strategy was giving a fixed address of x"7ff" to RAM when instruction was not a RAM access instruction. Thus the changes in the RAM address generation were MUXed out and as there was no unnecessary switching takes place the dynamic power reduction is achieved.

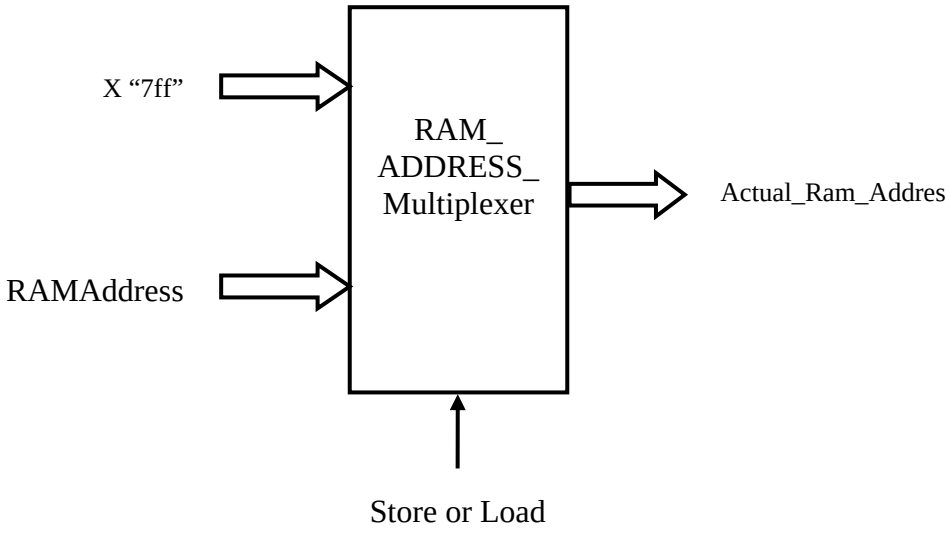


Figure 5.16: RAM Address Multiplexer

As seen in Figure 5.16 ram_address generated in the execute stage is fed to RAM, if there is load or store instruction else constant hexadecimal data 7FF is fed. Thus considerable amount of power is saved by preventing the occurrences of unnecessary switching by feeding constant address during execution of other instructions.

5.3.4 Clock Gating

Xilinx recommends for using the CLB clock enable pin instead of gated clocks. Gated clocks can cause glitches, increased clock delay, clock skew and other undesirable effects. Using clock enable saves clock resources and can improve timing characteristic and analysis of the design. But for the power reduction purpose if it is required to use a gated clock, most of the FPGA devices are facilitates with a clock enabled global buffer resource called BUFGCE.

However, a clock enable is still a better and preferred method to reduce or stop the clock to reach to various portions of the design and hence, subsequently to reduce the power.

There are several ways to use clock-enable resources which are available on devices, but to gate entire clock domains for power reduction purpose; it is preferable to use the clock-enabled global buffer resource called BUFGCE shown in Figure 5.17.

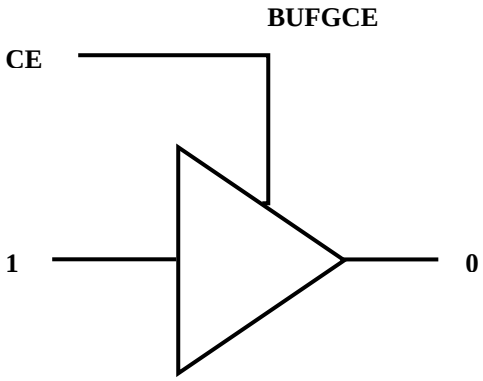


Figure 5.17: Clock – Enabled Global Buffer

Now, for applications that only attempt to pause the clock for a few cycles on small areas of the design, the preferred method is to use the clock-enable pin of the FPGA register. The first example demonstrated in Figure 5.18 which illustrates an inefficient way of gating clock signals, while the second example in Figure 5.19 shows a modified version of the code that map efficiently into the clock-enable pin.

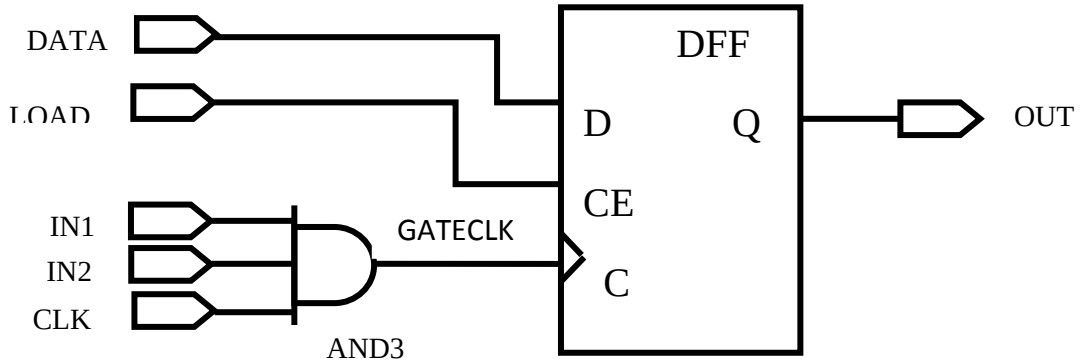


Figure 5.18: Gated Clock – Not Preferable

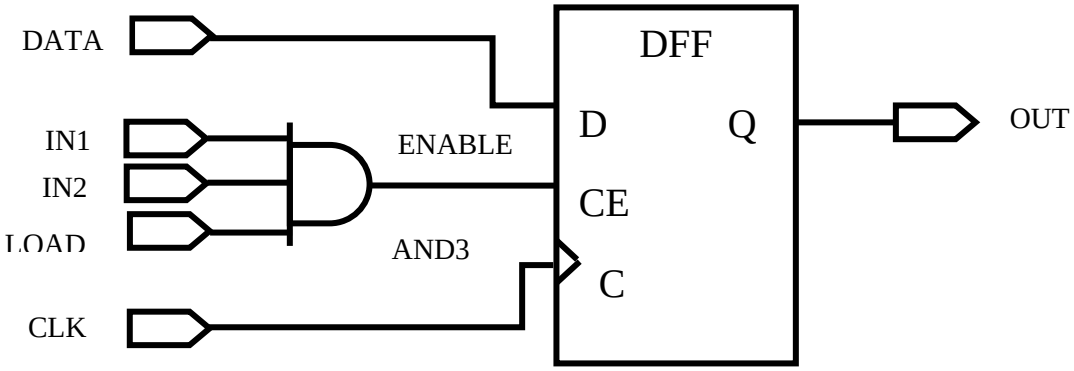


Figure 5.19: Clock Enable – Efficient way of Gating a Clock Signal

For global clock gating of design, one may use the clock-enable port of global clock buffers to stop the clock on entire clock domains. And for local level requirement, if clock gating of few registers of design is the required then following approach can be applied.

- Use the clock-enable port of registers to locally disable the clock.
- Consider replicating the clock-enable signal if it appears to be part of the paths that do not meet the timing requirements.

In this proposed design there is no scope for global clock gating as processor increments program counter on every clock and thus some portion of design is always active corresponding to the instruction fetched and executed. Thus for our processor design local level clock gating through clock enable pin is achieved. The basic idea over here is that for different types of instructions, different sets of registers are used, so when a particular instruction is getting decoded and executed the register corresponding to the other instructions toggle and get updated but are of no use but contributes heavily to consume the power. Thus whenever such situation occurs, it is better to do clock gating of registers to prevent them from toggling, in each pipeline stage, which are not in use by the current instruction. Following description explains that how the clock gating technique is implemented at all the stages of the pipelines.

Strategies to achieve the Clock Gating at different stages of pipeline

(1) Instruction Fetch

As program counter is computed and gets updated at every clock cycle; hence, there is no scope of Clock Gating.

(2) Instruction Decode and Operands Fetch:

In this stage instruction corresponding to program counter is fetched from ROM, since this is done at every clock cycle and the instruction fetched is combinatorially decoded. Hence there is no scope of Clock Gating.

(3) Execute Stage:

In this stage, the type of instruction is known; hence, accordingly registers corresponding to other instructions are Clock Gated through clock enable. There are three types of instruction getting executed in this stage, which are (a) RAM Instructions, (b) ALU Based Instructions and (c) Branch Instructions or NOP.

(a) RAM Instructions:

There are 4 RAM based instructions, so while these are loading or storing data to or from RAM, registers corresponding to ALU are Clock Gated. Instructions which require ALU operations generate regwrite signal in decode stage. This regwrite signal is used as clock enable signal for all registers corresponding to ALU based instructions. Thus when RAM instructions are getting executed regwrite will be zero and all the registers whose clock enable is fed by this regwrite signal will be disabled and considerable amount of power is saved. Registers which are Clock Gated during RAM instructions are as per the list given below:

A (32 bit register)
B (32 bit register)
Regwrite2 (1 bit register)
Zerowrite2 (1 bit register)
Ra2 (4 bit register)

(b) ALU Based Instructions:

There are 28 instructions which use ALU during their execution, so all the registers which are used by RAM instructions can be Clock Gated. Thus clock enable of registers corresponding to RAM instructions is tied to ANDing of store and load signals generated in decode stage. Now if instructions are ALU based, load and store both will be zero and thus registers using ANDing of them as clock enable will remain disabled as shown in Figure 5.18 and Figure 5.19, and subsequently this arrangement turns into saving of power consumption. Registers Clock Gated during ALU based instructions are listed below:

ram_datain (32 bit register)
ram_addr1 (32 bit register)
load2 (1 bit register)

store2 (1 bit register)

(c) Branch Instructions OR NOP:

Three branch instructions and an NOP instruction have no execution in execute stage so registers corresponding to RAM access and ALU all are Clock Gated. Regwrite, load and store signals all will be zero and thus above all registers will be clock gated. The Registers Clock Gated during Branch or NOP instructions are:

A (32 bit register)
B (32 bit register)
Regwrite2 (1 bit register)
Zerowrite2 (1 bit register)
Ra2 (4 bit register)
ram_datain (32 bit register)
ram_addr1 (32 bit register)
load2 (1 bit register)
store2 (1 bit register)

(4) Write back:

In write back stage register bank of 16 registers of 32 bits are updated if an ALU based instructions was performed. Hence, in case of RAM, Branch or NOP instructions this update is not required and consumes power. So these registers are Clock Gated by connecting regwrite signal passed on from execute stage to clock enable pin of these registers. So if instruction was ALU based then only registers will be updated else they will remain disabled due to Clock Gating. Registers which are Clock Gated during Write Back Stage are:

R0 (32 bit register)
R1 (32 bit register)
.
.

R15 (32 bit register)

After successful implementation of all the logic strategies for power optimization techniques called Novel RAM Addressing Scheme discussed in section 5.3.3 and Clock Gating described in section 5.3.4 at the processor architecture level, the CPU functionality has been tested and verified again to ensure its correctness. The power analysis of complete system has been done and the considerable amount of power reduction is achieved. It is resulted in reduction of 17mw (16%) of system power consumption. It is also noted that the Clock Gating technique is the major contributor in reducing the dynamic power consumption at the system level. The screen shot of the summary of power analysis is shown in Figure 5.20. Here the power analysis has been carried after implementation of two techniques together which are RAM Addressing Scheme and the Clock Gating along with earlier techniques.

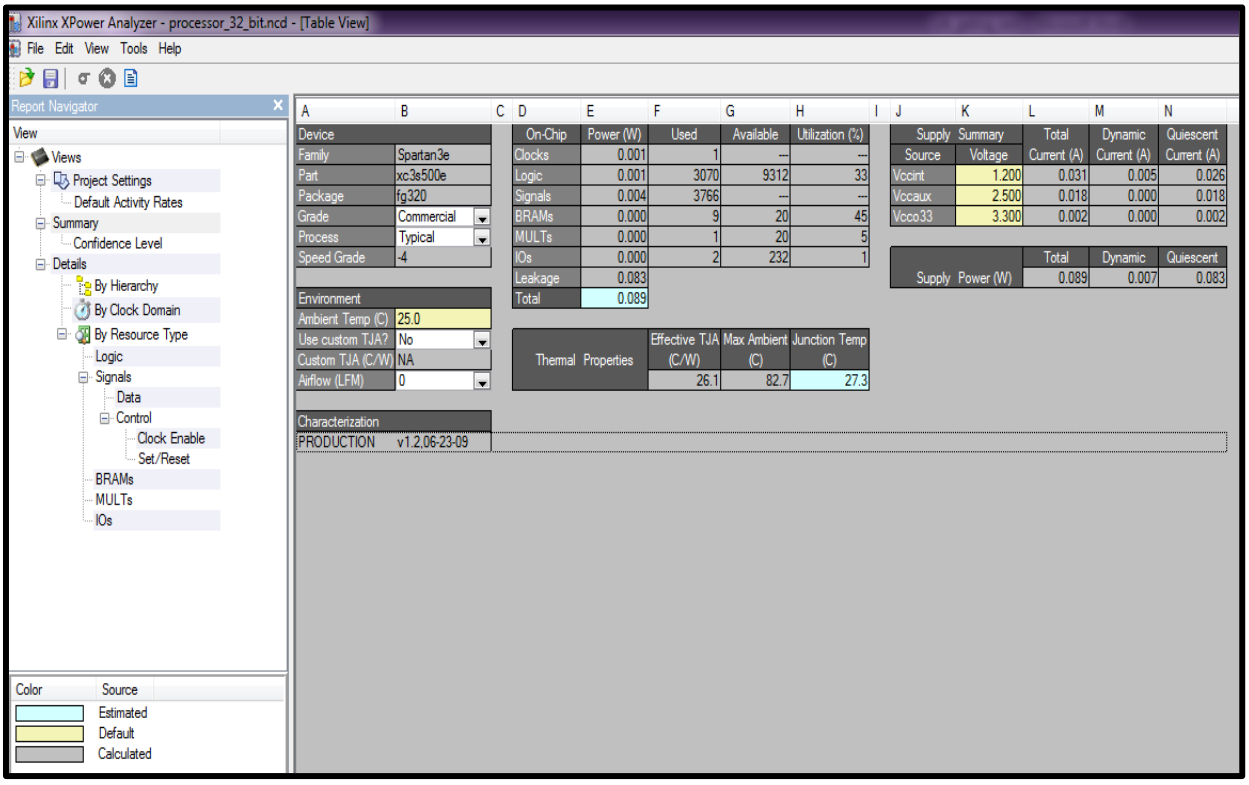


Figure 5.20: Summary of Power Consumption Report for 4 – Stage Pipeline CPU (After Implementation of RAM Addressing Scheme and Clock Gating along with earlier Techniques)

The testing of Clock Gating has been done separately for all types of the instructions and are represented with the relevant waveforms as shown in following Figure 5.21 and Figure 5.22. Following waveforms describes the Clk_gating during for ALU related instructions.

Figure 5.21: Simulated Waveforms for Clk_gating Signal Varification for ALU Related Instructions

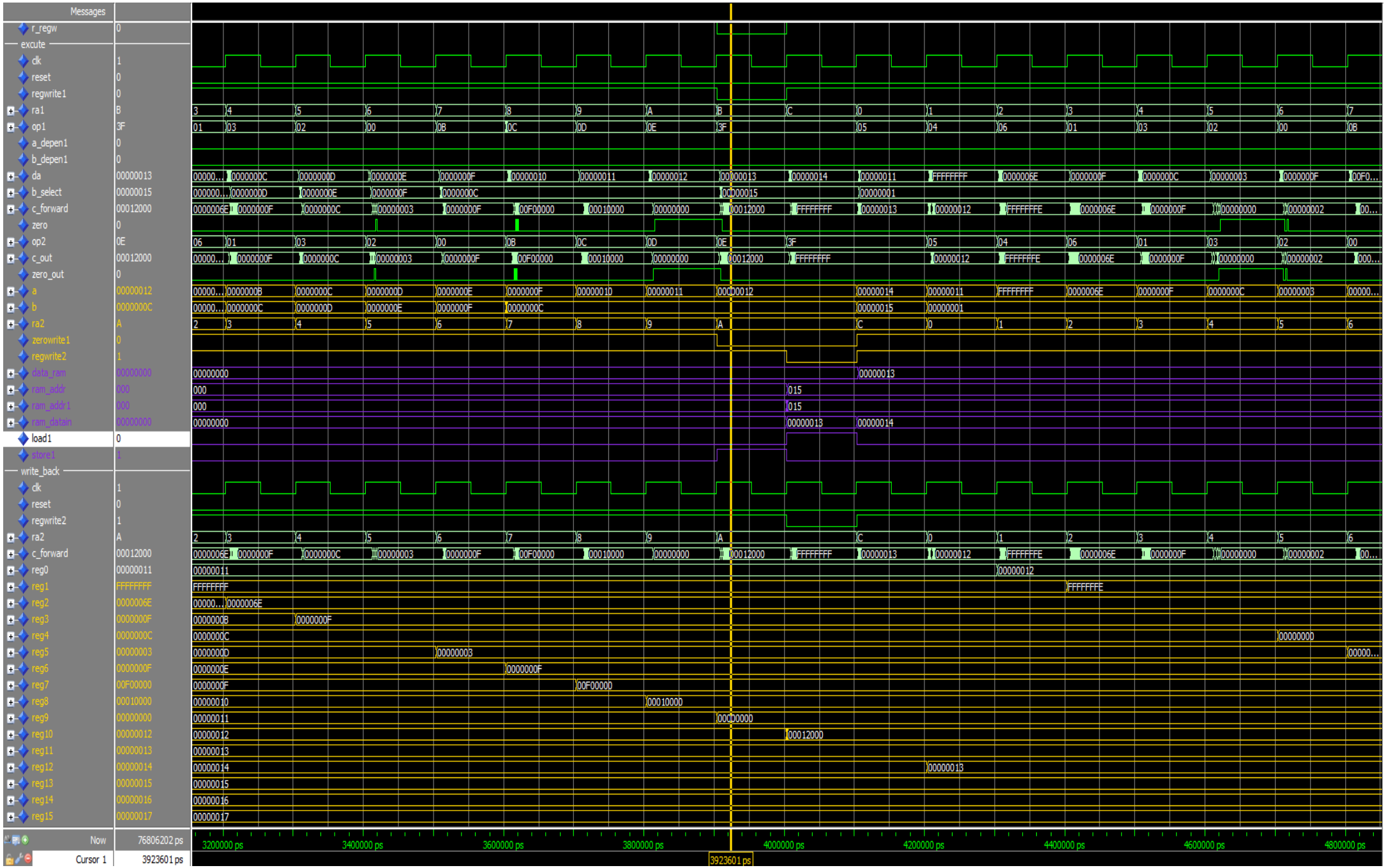


Figure 5.22: Simulated Waveforms for Clk_gating Signal Verifcation during RAM Access Instructions

It is shown in above Figure 5.21 that the registers such as data_ram, ram_addr, ram_data_in and load and store registers are clock gated during the execution of ALU related instructions and it also provides the constant hexademimal “7FF” address which is fed to RAM during ALU based instructions, because of RAM Address technique implementation which provides ram_data_in to zero, the details of Clk_gating during RAM Access instruction can be better explained through the waveforms shown in Figure 5.22.

As can be seen due to store instruction to RAM there is no change in a, b, ra, zerowrite, regwrite and register bank of 16 registers. When there is a Branch or NOP related instruction all the registers for both RAM Access and ALU are clock gated shown in Figure 5.22.

5.4 Verification of 4- Stages CPU After Implementation of Power Optimization Strategies

5.4.1 Performance Verification

All four power optimization strategies Memory Access Stage Removal, Resource Sharing Strategy, RAM Addressing Scheme and the Clock Gating have been implemented successfully at the hardware level and the whole system architecture has been tested and verified by executing many flavours of instructions and the performance of all of them has been checked for all the 4 – stages of the CPU. Following Figure 5.23 to Figure 5.26 demonstrates the performance of implementation after implementing all the power saving proposed techniques under different pipeline stages through the waveforms generated by using ModelSim SE 6.5. All these waveforms are self explanatory. Then after the power consumption comparison has also been made, in which the power consumption and estimated power for 5 – stage CPU has been discussed. Then the 5 – stage CPU is converted to 4 – stage CPU by implementing the Memory Access Stage Removal technique for which the power consumption is measured. Similarly, after implementation of each technique the power measurement is done and finally the power saving is achieved which is graphically represented through the comparison charts. The simulated waveforms for verification of 4 – stage CPU with Resource Sharing,Clock Gating and RAM Addressing Technique are demonstrated through following Figure 5.23 to Figure 5.26.

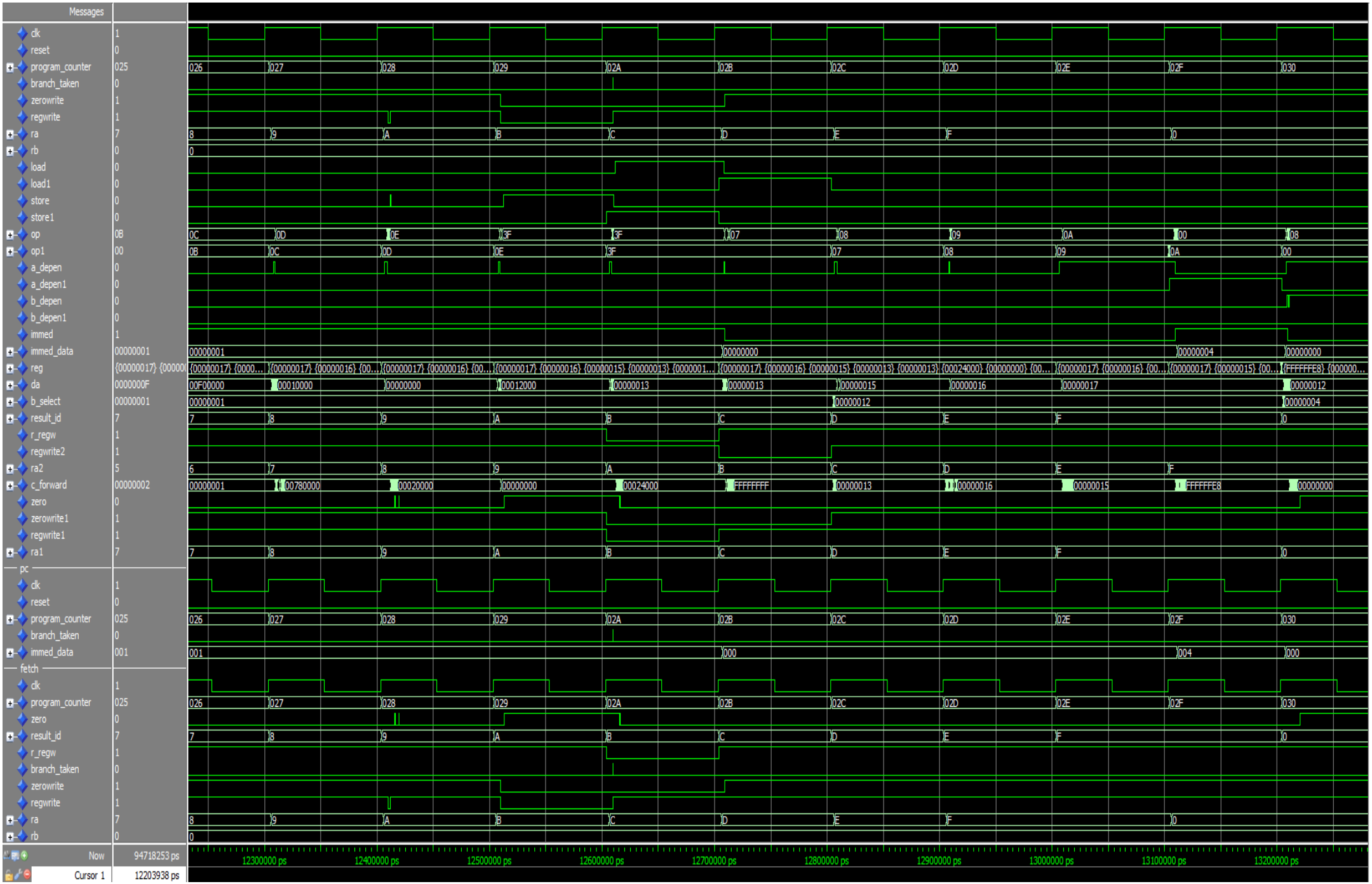


Figure 5.23: Verification of Instruction Fetch for 4 – Stage CPU

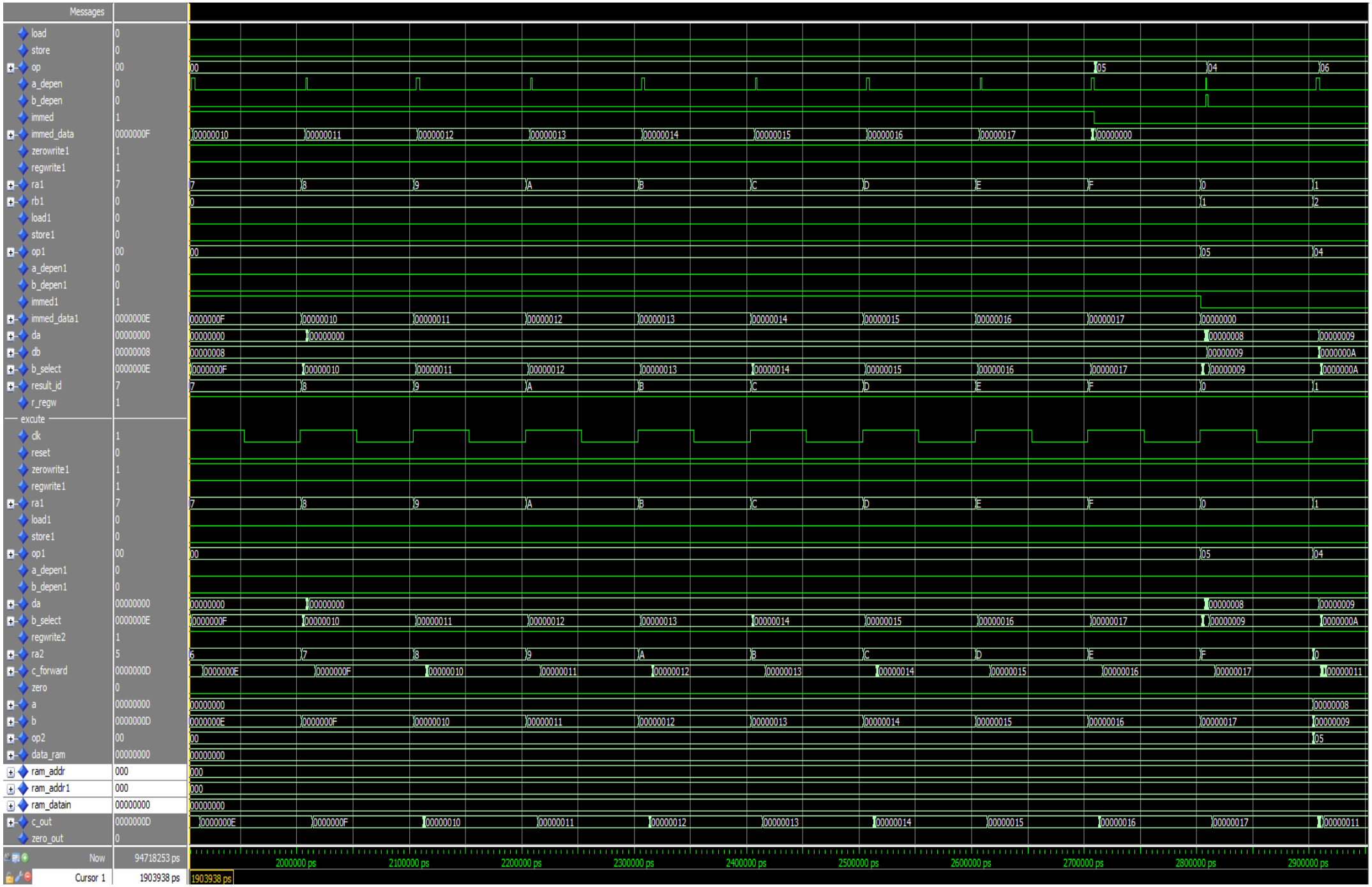


Figure 5.24: Verification of Instruction Decode and Operand Fetch for 4 – Stage CPU

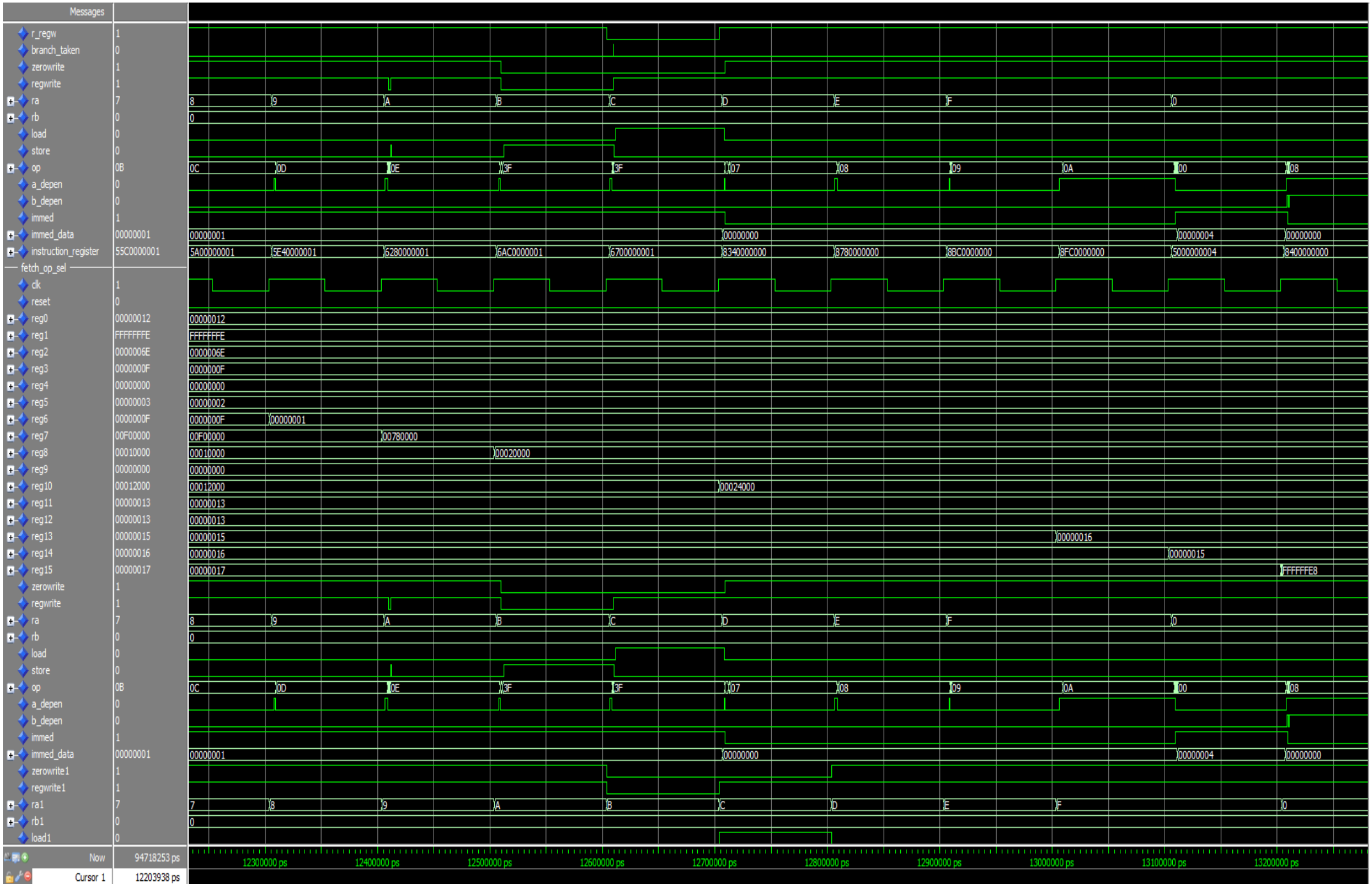


Figure 5.25: Verification of Instruction Execute for 4 – Stage CPU

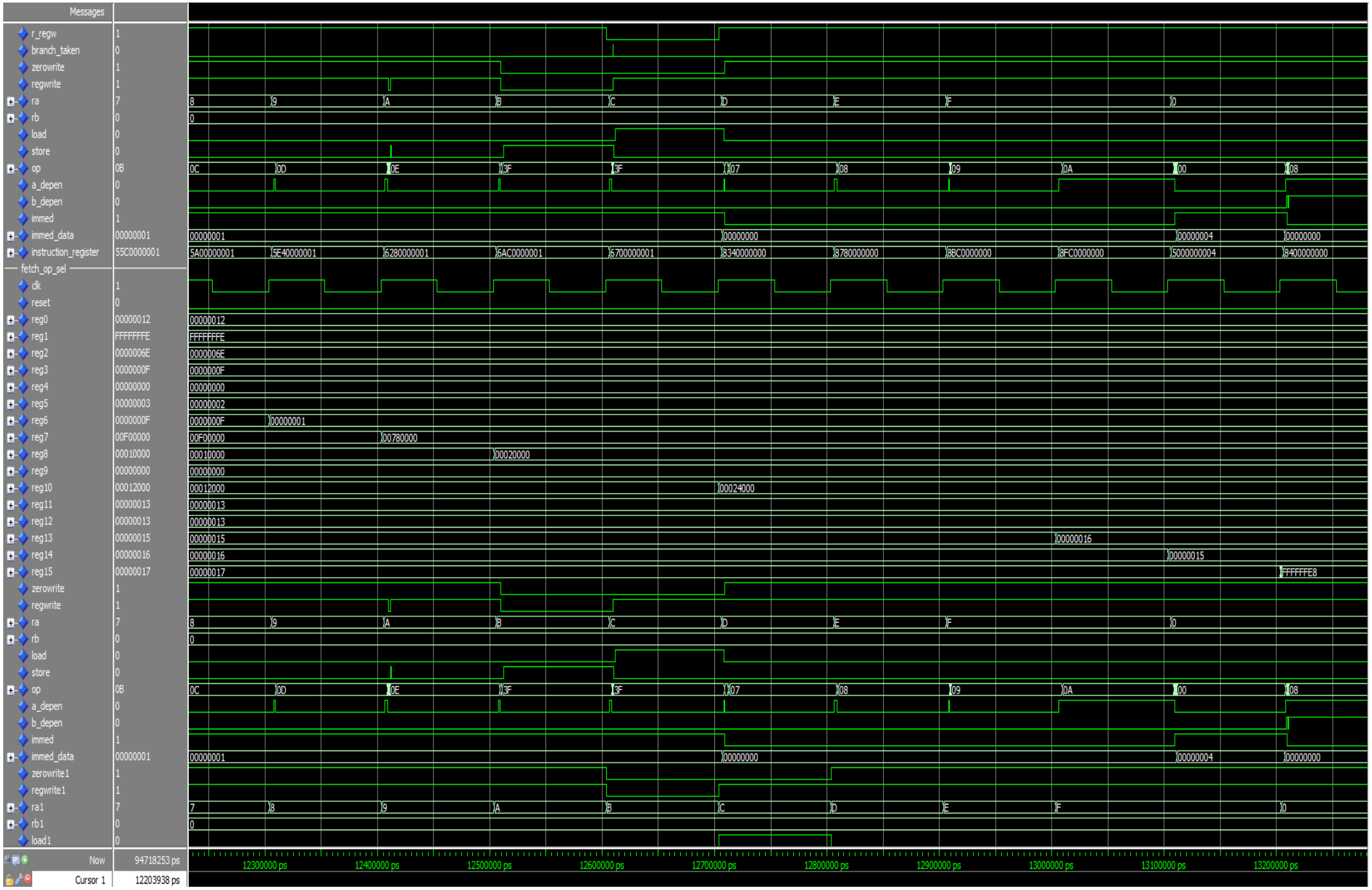


Figure 5.26: Verification of Write Back Operation of 4 – Stage CPU

5.4.2 Graphical Representation of Power Requirement

Following Figure 5.27 shows the graphical comparison of power requirements of the various modules of the systems for both the conventional (5 – Stage) and CPU with modified pipelining structure (4 – Stage), i.e. after implementation of Memory Access Stage Removal Technique. It is noted that the improvement in power consumption is achieved by 3mW (2.65%) in the system after the incorporation of this technique at the hardware design step.

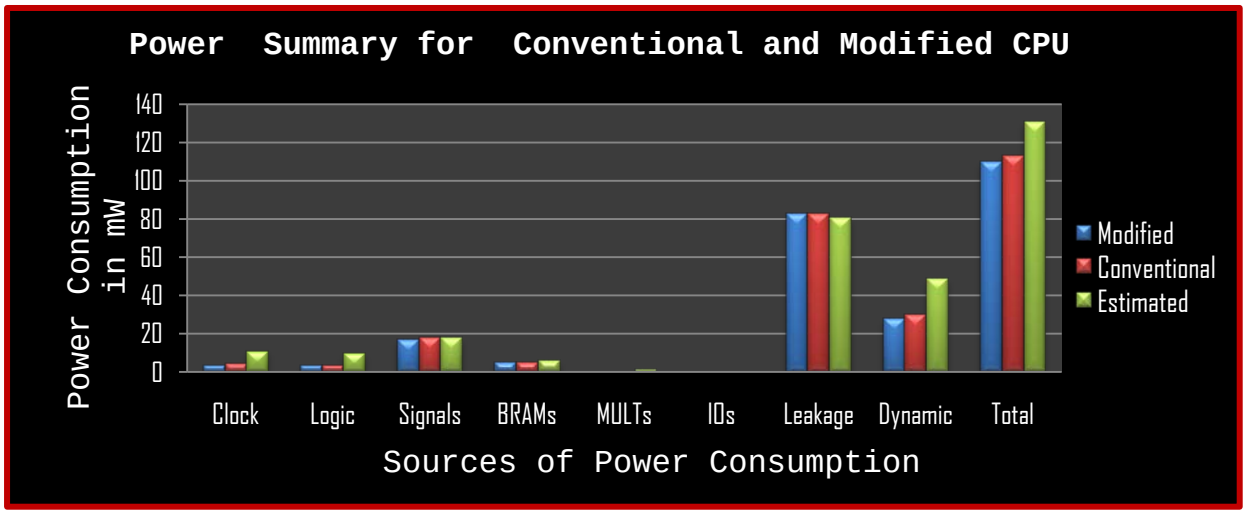


Figure 5.27: Graphical Representation of Estimated and Actual Power Consumption of 5 – Stage CPU

Table 5.2 summarizes the results obtained from the power analysis carried on conventional 5 – Stage CPU and 4 – Stage CPU after implementation of Memory Access Stage Removal Technique. It represents the power consumed by the various modules of the system under consideration and the results are also compared with the estimated values.

Table 5.2: Summary of Power Results

Category	Clock mW	Logic mW	Signals mW	BRAMs mW	MULTs mW	IOs mW	Leakage Power mW	Dynamic Power mW	Total Power mW
Estimated Requirement	11	10	18	06	01	00	81	49	131
Conventional CPU (5- Stage)	04	03	18	05	00	00	83	30	113
Modified CPU (4 – Stage)	03	03	17	05	00	00	83	28	110

The following Figure 5.27 shows the graphical view of the power consumption in different modules of the modified CPU after implementation of all the newly suggested power reduction techniques such as Memory Access Stage Removal, Resource Sharing, RAM Addressing Scheme and Clock Gating and finally the power dissipation comparison has made between the conventional 5 – Stage CPU and the modified 4 – Stage CPU modules. The power results indicating power consumption requirements are summarizes in Table 5.3.

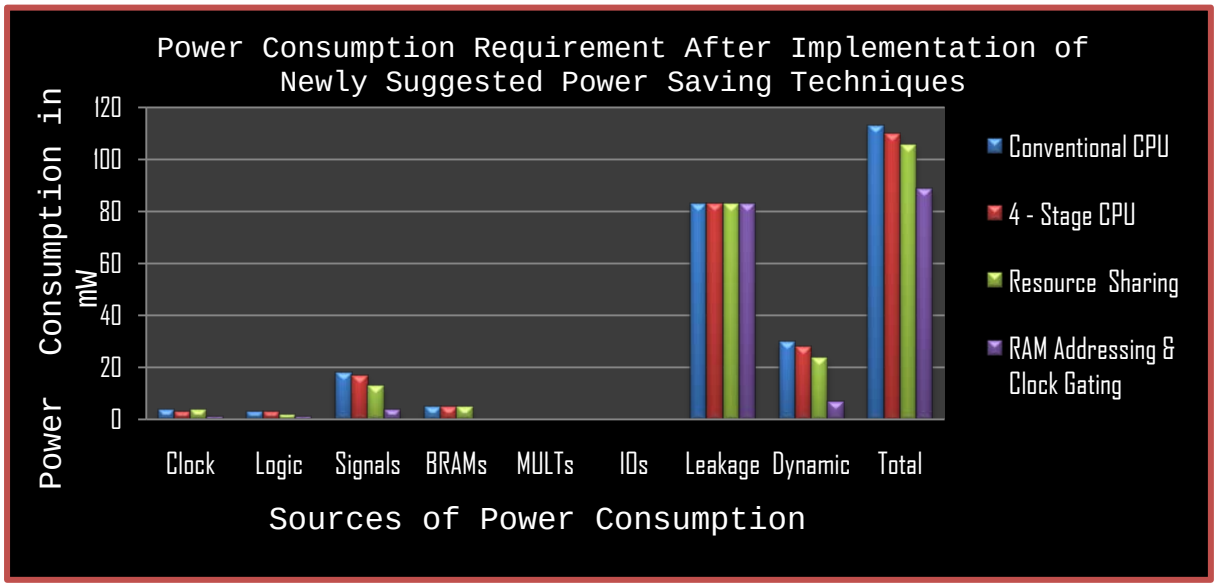


Figure 5.28: Power Consumption Requirement after Implementation of Power Saving Techniques

Table 5.3: Summarizes the Results of Power Requirements

Category	Clock mW	Logic mW	Signals mW	BRAMs mW	MULTs mW	IOs mW	Leakage Power mW	Dynamic Power mW	Total Power mW
Conventional CPU (5- Stage)	04	03	18	05	00	00	83	30	113
Modified CPU (4 – Stage)	03	03	17	05	00	00	83	28	110
Resource Sharing	04	02	13	05	00	00	83	24	106
RAM Addressing & Clock Gating	01	01	04	00	00	00	83	07	89

Hence, the total power requirement of conventional 5 – stage pipeline CPU has been reduced to 89 mW from its original need of 113 mW i.e. overall 21% of improvement in the power consumption is achieved. The overall power comparison of the system has been presented in

the below Figure 5.29, which represents clearly that the power reduction is achieved successfully.

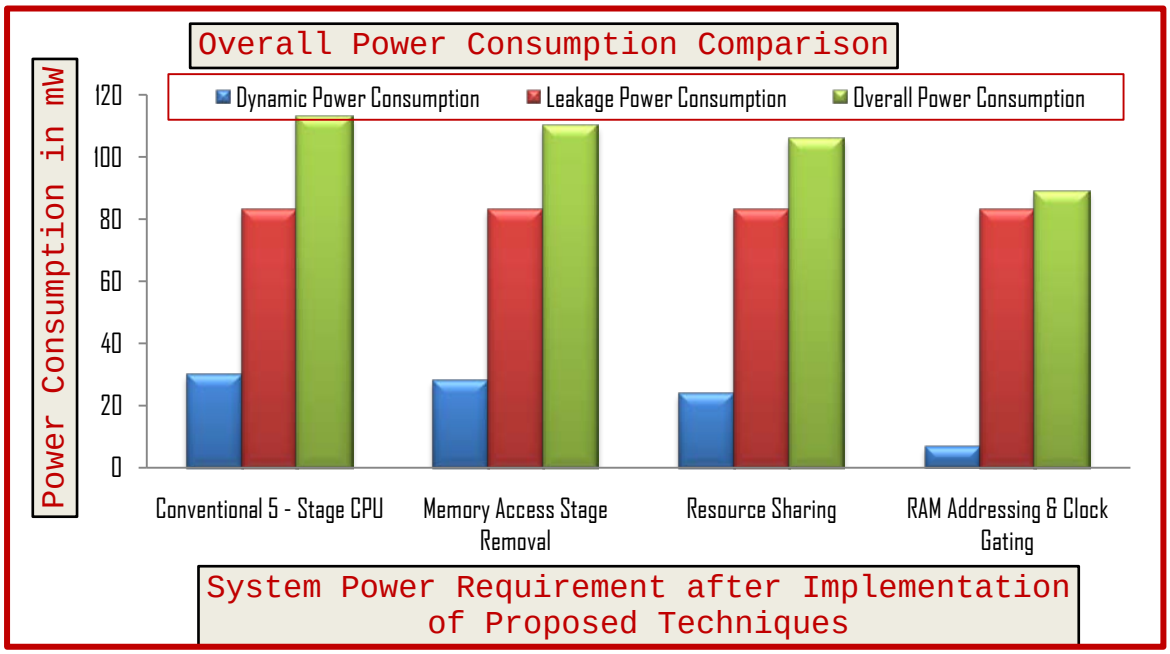


Figure 5.29: Overall Power Consumption Comparisons

The power consumption of these CPUs is also analyzed at different frequencies applied to the conventional CPU and the CPU with modified pipeline and also to the CPU after implementation of all the power saving proposed strategies, the related power figures are shown in Table 5.4.

Table 5.4: Power Consumption of CPUs at Different Frequencies

Frequency MHz	Power Consumption in 5 - Stage CPU (mW)	Power Consumption in 4 - Stage CPU (mW)	Power Consumption in 4 - Stage CPU after Implementation of all the Techniques (mW)
10	95	95	82
20	102	102	84
30	109	109	87
40	113	110	89
50	119	121	95
60	121	124	98
70	129	127	101
80	132	130	104
90	134	133	106
100	138	137	108

The plot for power consumption versus different clock frequencies for standard and the modified processors is shown in Figure 5.30.

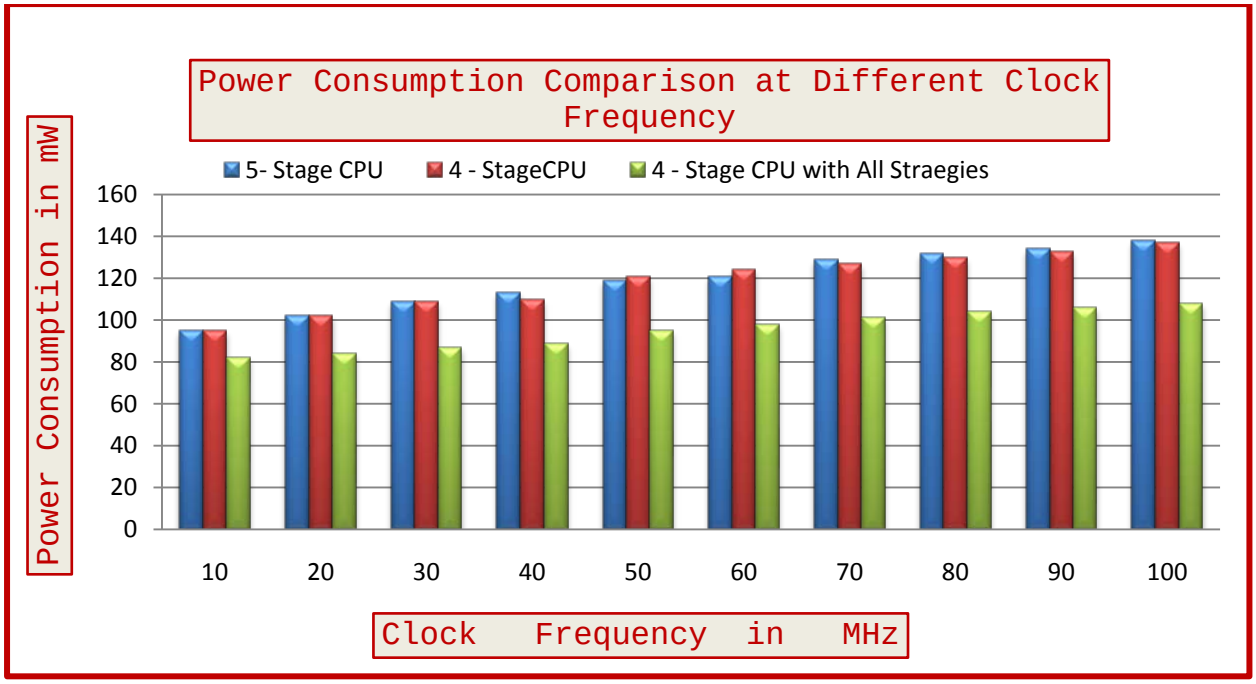


Figure 5.30: Graphical View of Power Consumption Comparison at Different Clock Frequencies

It is noted from the Figure 5.30 that the power consumption increases as the clock frequencies increases for all the categories such as standard, modified CPU and the CPU after implementation of power saving techniques. But the power consumption of modified and optimized CPU implementation is always less than the standard CPU for all the clock frequencies.

5.5 Conclusion

In the beginning of this chapter the architecture of 5- stage standard CPU is discussed and its performance has been successfully verified through the waveform generation and analyzed for its power requirement using the Xilinx Xpower Analyzer tool.

In order to achieve the prime objective this 5 – stage pipeline structure of CPU is modified to have 4 – stage pipeline structure by implementing the memory access stage removal technique. Its performance is verified and power analysis has been done successfully and the reduction in dynamic power requirement is achieved without affecting the performance of the processor. Then all other newly suggested techniques which are Resource Sharing, RAM Addressing Scheme and the Clock Gating are implemented on the modified CPU and finally power analysis is carried on this CPU successfully.

It is concluded that the proposed techniques have impressively helped to reduce the dynamic power consumption of the implementation up to approximately 21% and that to without compromising the performance of the system.

Finally the various power comparisons are made using the data received from the analysis as an input and are represented graphically. Also the system behaviour is observed as per the expectation and the considerable amount of power reduction is achieved.

The implementation also analyzed for its power requirement at different clock frequencies and it is concluded that the power requirement for standard and the modified CPU is increased with the increase in the frequency but the modified CPU consumes always less power compared to the standard one.

Following chapter discusses the conclusions and the future scope of the work.

Chapter 6
Conclusions and Future Work

6.1 Summary and Contributions

In early days the system performance was only considered to be an important factor and hence, the designers were developing the implementations by focusing only on speed and the performance of the system and the concern for power consumption requirements of the system was not given sufficient attention. But because of advancement of device technology, which has doubled the system complexity after almost every 18 – 24 months; and along with the performance, the system power consumption has also raised to considerable level. This increased power requirement has introduced newer challenges in connection with the system power dissipation, thermal parameters and hence the overall system reliability. The power reduction methodologies can be implemented at various abstraction levels of the implementation designing.

This dissertation work has proposed very unique power saving techniques which are implemented at the hardware design level up on the implementation under consideration, in our case, it is five stage pipelined processor (Standard Processor) based on RISC principle and the modification in the pipeline architecture of the standard processor has been done and various proposed dynamic power saving techniques are incorporated in this processor. Then the system is implemented on SPARTEN – 3E FPGA for testing and verification purpose.

As the system is targeted to be implemented on the FPGA the state-of-the-art literature survey has been presented in Chapter 2 which includes various FPGA technologies, internal architectures and also discussed the various dynamic and static power consumption sources of the MOS based systems. It also discusses the mathematical relevance of the power consumption resources.

The dynamic power reduction possibility lies at different abstraction levels and many techniques are available which can be applied at device level, architecture, logical or system architectures level. The detailed survey on existing power reducing techniques is included in Chapter 3. The span of this Ph. D. work is restricted to the application of newly derived energy saving techniques implemented at architecture level i.e. at hardware level.

A standard processor with five pipeline stages is constructed in modular fashion. The formation of subset of instructions is tested and validated through the simulation waveform for number of the programs and correct functionality is recorded. It is estimated and analysed for its power requirement using Xilinx Xpower Estimator and Xpower Analyzer, the reported power consumption is 113 mW. This dissertation work has suggested four unique dynamic power reducing techniques named as Memory Access Stage Removal, Resource Sharing, A novel RAM Addressing Scheme and a Clock Gating arrangement.

By applying memory access stage removal the pipeline structure of the standard processor is modified and made it of four stage pipelined processor. Then the newly formed processor is analyzed for its power requirement which is 110 mW. Hence, the power reduction of 2.65% is achieved compared to that of standard processor. The other technique resource sharing is applied up on the newer processor and analyzed for power consumption which is recorded as 106 mW; means it further reduces the 4 mW (3.63%). The RAM Addressing Scheme and the Clock Gating is applied together on this improved design which consumes the 89 mW only without affecting the performance of the processor. Hence, this strategy again reduces the 17 mW (16%). Here Clock Gating has played a major role in reducing the power consumption and the overall reduction in system power consumption is achieved up to 21%.

6. 2 Future Work

The work presented in this thesis provides suggestions to be implemented at hardware level to correct the whole system that strongly impact the power consumption of some complex embedded system and come out with some ideas that efficiently reduces the system power consumption.

However, some interesting points of future research have emerged during the evolution of this work. Many of them are related to the designed approaches, some of them are referenced to their implementation trends in recent era.

On the other hand, the approaches presented in this work have been designed from power optimization view – point. However, the thermal implications of these approaches are needed to be studied and optimized. It is also required to have in-depth study for these approaches with reference to their manufacturing processes, fabrication processes and its actual feasibility. The proposed system can be further optimized by applying many other existing power saving techniques at different abstraction level.

Finally, the concept of low – power is still in very early stages. It still has to understand the complex mechanism that appears in the system behaviour when these newly proposed techniques are applied.

Authors believe that the power management needs multidimensional inputs which are continually expanding with new techniques being developed at every abstraction levels.

6.3 Closing Remark

In summary, we believe that the system level power optimization is an active research area in the years ahead. The techniques proposed in this dissertation provide power reduction on the order of tens of percent, and this is clearly a good beginning. However, further improvements of the same or even with larger magnitude will be needed as processor usage especially in mobile applications increases very rapidly.

The research directions discussed above convinced us to judge that there is considerable space for improvement in the domain of system power consumption.

Yet, it is too early to say that which methodology will help the society to solve the problem of power dissipation. But it is for sure that this work can be taken as base and one can keep developing and adding more and more techniques to the proposed implementation at different levels and can design low – power implementations.

List of Publication

Following is the list of our publications which includes the most important papers and relevant to the work included in this thesis.

- Kiritkumar Bhatt, Prof. A I Trivedi *"Memory Access Stage Removal Technique for Dynamic Power Reduction in Embedded Processors"* accepted and to be presented/published in IEEE International Conference on Computer Applications Technology, ICCAT'2013, 20-22 Jan'2013, Souse, Tunisia.

- Kiritkumar Bhatt, Prof. A I Trivedi, *"Power Estimation and Optimization in Embedded Processors and Its Implementation"* accepted and to be published in International Journal of Innovative Systems Design and Engineering, in Jan'2013.

- Kiritkumar Bhatt, Prof. A I Trivedi, "Implementation of Resource Sharing Strategy for Power Optimization in Embedded Processors", International Journal of Computer Engineering and Intelligent Systems, Vol.3, No.8, pp. 35 – 45, 2012. Print: ISSN 2222-1719, Online: ISSN 2222-2863

- Kiritkumar Bhatt, Prof. A I Trivedi, *" Power Optimized Embedded Processor Design with Parallel Pipelining"* International Journal of Programmable Circuits and Systems Vol. 3, No. 1, pp 6 – 9, Jan- 2012. Print: ISSN 0974 – 973X, Online: ISSN 0974 – 9624

- Kiritkumar Bhatt, Prof. A I Trivedi, *"Low – Power Pipelined Processor Design, Its Verification and Implementation on FPGA"* 2012 – 4th IEEE Sponsored International Conference on Electronics and Computer Technology – ICECT 2012, 6 – 8April'2012, Int. Conf. Proceedings Vol.1, No.1, pp. 91 – 95, Kanyakumari, India. ISBN: 978-1-4673-1850-1

- Kiritkumar Bhatt, Prof. A I Trivedi, *"Power Estimation of Switching Activity for Low – Power Implementation on FPGA"* International Journal of Programmable Circuits and Systems Vol. 3, No. 14, pp 803 – 807, Nov - 2011. Print: ISSN 0974 – 973X, Online: ISSN 0974 – 9624

- Kiritkumar Bhatt, Prof. A I Trivedi, *" Power Computation Model of CMOS Based FPGA used for Power Optimization"* International Journal of Programmable Circuits and

Systems Vol. 3, No. 13, pp 759 – 762, Oct- 2011. Print: ISSN 0974 – 973X, Online: ISSN 0974 – 9624

- Kiritkumar Bhatt, Rajshree Jetani, *"Low Voltage Resized Design of Programmable Current Mirror Using 180 NM Technology"*, International conference on Information, Knowledge & Research in Engineering, Technology & Science – 2012 (ICIKR – ETS – 2012) at G K Bharad Institute of Engineering, Rajkot. Pp. 910 – 914. ISBN : 978-81-906220-3-5/24-25/3,2012

- Kiritkumar Bhatt, Rajshree Jetani, Devang Shah, *"Analysis of Bulk-Driven Technique for Low Voltage/Power Analog Circuit Design",* National Conference on Power Systems, Embedded Systems, Power Electronics, Communication, Control and Instrumentation – PEPCCI -2012, pp. 187 – 192 Jan'2012, Vasad, India. ISBN: 978-93-81286-06-7

- Kiritkumar Bhatt, Priya Kulkarni, Rachana Jani "A Survey on *Power Management Model for ARM SA – 1100 CPU"* in proceedings of National Conference on "Exploring Potentialities of Women in Engineering- EPWIE -2009" 3 – 4 July'2009, pp. 1 – 4, CIT - Changa, India.

- Kiritkumar Bhatt, Priya kulkarni, Rachana Jani, *"Dynamic Power Optimization Technique: A Case Study"*, National Paper Contest on Advanced Embedded Networking, Processing and Communication-AENPC – 2009, IETE Center - Baroda Jan'2009.

- Kiritkumar Bhatt, Prof. A I Trivedi, *"Low Power Processor Design and Its Formal Verification"* National Paper Contest on Advanced Embedded Networking, Processing and Communication-AENPC – 2009, IETE Centre, Baroda Jan'2009.

- Kiritkumar Bhatt, Prof. A I Trivedi, *"A Study of Various General Techniques for Reducing the Dynamic Power Consumption of Integrated Circuits"*, State Level Paper Contest on "Networking: Technology and Applications", IETE Centre, Baroda April'2008.

Bibliography

[1] P B Endecott, *Processor Architectures for Power Efficiency and Asynchronous Implementation,* UK: University of Manchester, 1993.

[2] D Brooks, V Tiwari and M Mortonosi, “A framework for architecture - level power analysis and optimizations, set processor design and embedded systems,” *Int. Symp.on Computer Architecture,* 2000.

[3] T Osmulski, et al., “A probabilistic power prediction tool for Xilinx FPGA,” *Int. Wksp. on Computer Embedded/Distributed HPC Systems and Applications,* pp. 776 - 783, May 2000.

[4] J Anderson and F. Najm, “Active leakage power Optimization for FPGAs,” *ACM/SIGDA Int. Symp. on Field Programmable Gate Array,* pp. 33 - 41, Monterey, 2004.

[5] S. Turgis, N. Azemard, D. Auvergne, “Explicit evaluation of short - circuit power dissipation for CMOS logic structures,” in *Proc. of ISLPD*, 1995.

[6] Jan M Rabey and M Pedram, Low Power Design Methodologies, Boston: Kluwer Academic, 1996.

[7] J P Uyemura, Fundamentals of MOS Digital Integrated Circuits, Addison Wesley, 1988.

[8] J H Anderson, F N Najm, “Active Leakage Power Optimization for FPGA,” *IEEE Trans. on Computer - Aided Design of Integrated Circuits and Systems,* vol. 25, no. 3, pp. 63 - 76, March 2006.

[9] J M Chang, M Pedram, “Emergy minimization using multiple supply voltages,” *IEEE Trans. on VLSI Systems,* vol. 5, pp. 1-8, 1997.

[10] J L Ayala and M Lopez - Vallejo, “A unified framework for power-aware design of embedded systems,” in *Int. Wksp. on Power and Timing Modelling, Optimization and Simulation*, Sept'2003.

[11] I Brozozowski and A Kos, “Minimization of Power Consumption in Digital Integrated Circuit by Reduction of Switching Activity,” in *25th Euromico Conf.*, Sept'1999.

[12] Kiritkumar Bhatt, A I Trivedi, “Power Estimation and Switching Activity for Low-Power Implementation on FPGA,” *Int. Journal of Programmable Devices, Circuits and Systems,* vol. 3, no. 14, pp. 803-806, 2011.

[13] Kiritkumar Bhatt, A I Trivedi, “Power estimation of switching activity for low-power implementation on FPGA,” *Int. Journal of PDCS,* vol. 3, no. 14, pp. 803-806, 2011.

[14] Jason Helge Anderson, *PhD thesis on Power Optimization and Prediction Techniques for FPGAs,* Toronto: Department of Electrical and Computer Engineering, University of Toronto, 2005.

[15] Andres David Garcia Garcia, *PhD thesis on Power Consumption and Optimization in Field Programmable Gate Array,* National Superior Telecommunications of Paris, 2000.

[16] A K Sharma, Programmable Logic Handbook, Mc Graw Hill, 1988.

[17] V. Betz, J. Rose, A. Marquardt, Architecture and CAD for Deep Submicron FPGAs, Boston: Kluwer Academic Publisher, 1999.

[18] V. Betz, J. Rose, A. Marquardt, Architecture and CAD for Deep Submicron FPGAs, Kluwer Acaademic Publisher, 1999.

[19] A. Bellaouar, M I Flmassy, Low - Power Digital VLSI Design, Circuits and systems, Kluwer Academic Publisher, 1995.

[20] L.Benini, A. Bogliolo and G. Micheli, "A Survey of design techniques for system - level dynamic power management," *IEEE Trans.on VLSI Systems,* pp. 299 - 316, June 2000.

[21] A P Chandrakasan, S. Sheng and R W Brodersen, "Minimizing Power Computation in Digital CMOS Circuits," *IEEE Trans. on VLSI Systems,* vol. 83, no. 4, pp. 498 - 523, 1995.

[22] J.Lamoureux, G.Lemieux and S.Wilton, "Glitchless: An active glitch minimization technique for FPGA," *Int. Symp. on Field-Programmable Gate Array,* pp. 156-165, 2007.

[23] R Jevtic, C.Carreras and G.Caffarena, "Fast and accurate power estimation of FPGA DSP components based on high-level switching activity models," *Int. Journal of Electronics,* vol. 95, no. 7, pp. 653-668, 2008.

[24] V. George and J. Rabey, Low - Energy FPGAs: Architecture and Design, Boston: Kluwer Academic Publisher, 2001.

[25] Fei Li, Deing Chen, Lei He, Jason Cong, "Architecture Evolution for Power Efficient FPGAs," *Int.Symp. on FPGA,* pp. 175 - 184, August 2003.

[26] F.Li, Y.Lin, L.He, D.Chen and J.Cong., "Power Modeling and Characteristics of Field Programmable Gate Arrays," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems,* vol. 24, no. 11, pp. 1712-1724, 2005.

[27] Luca Benini, Giovanni De Micheli, "System - Level Power Optimization : Techniques and Tools," *ACM Trans. on Design Automation of Electronic systems,* vol. 5, no. 2, pp. 115-192, 2005.

[28] R A Powers, "Batteries for Low Power Electronics," *Proc. of IEEE,* vol. 83, no. 7, pp. 687-693, 1995.

[29] Bill Moyer, “Low-Power Design for Embedded Processors,” *Proc. of IEEE,* vol. 89, no. 11, pp. 1576-1587, 2001.

[30] Vasanth Venkatachalam, Michael Franz, “Power Reduction Techniques for Microprocessor Systems,” *ACM Trans. on Coputing Surveys,* vol. 37, no. 3, pp. 195-237, 2005.

[31] L. Benini, et al., “A Survey of Design Techniques for System-Level Dynamic Power Management,” *IEEE Trans. on VLSI,* vol. 8, no. 3, pp. 113-130, 2000.

[32] S R Vemuru and N.Scheinberg, “Short circuit power dissipation estimation for CMOS logic gates,” *IEEE Trans. on Circuits and Systems,* vol. 43, no. 10, pp. 762-765, 2009.

[33] D.Dal, A.Nunez and N.Mansouri, “Power island: A high-level techniquefor counteracting leakage in deep sub-micron,” *Int. Symp. on Quality of Electronic Design,* pp. 165-170, 2005.

[34] S.Yaldiz, A.Demir and S. Tasiran, “Stochastic modeling and optimization for energy management in multicore systems: a video decoding case study,” *IEEE Trans. on CAD of Integrated Circuits and Systems ,* vol. 27, no. 9, pp. 1264-1277, 2008.

[35] E.Nurvitadhi, B.Lee, C.Yu and M.Kim, “A compartative study of dynamic voltage scaling technique for low-power video decoding,” in *Int. Conf. on Embedded Systems and Applications*, 2003.

[36] Kursun, E.Ghiasi and Sarrafzadeh.M, “Transistor level budgeting for power optimization,” in *Proc. of 5th Int. Symp. on Quality of Electronic Design*, 2004.

[37] Chen D., Cong J., Li F. and He.L., “Low power technology mapping for FPGA architectures with dual supply voltages,” in *Proc. of ACM/SIGDA 12th Int.Symp. on Field Programmable Gate Array*, 2004.

[38] Li.H, Katkoori and Mak W.K., “Power minimization algorithms for LUT-based FPGA technology mapping,” *ACM Trans. on Design Automation and ElectronicsSystems,* vol. 9, no. 1, pp. 33-51, 2004.

[39] Strollo.A., Napoli.E., Caro.D.D., “New clock-gating techniques for low-power flip-flops,” *Proc. of Int. Symp. on Low Power Electronics and Design,* pp. 114-119, 2000.

[40] Zhao.P., Darwshi.T., and Bayoumi.M., “High-performance and low-power conditional discharge flip-flop,” *IEEE Trans. on VLSI Systems,* vol. 12, no. 5, pp. 477-484, 2004.

[41] Ernst.D., Kim.N.,Das.S., “A low-powerpipeline based on circuit-level timing speculation,” *Proc. of Int. Symp. on Microarchitecture, IEEE Computer Society,* pp. 7-18, 2003.

[42] Pouwelse, et al., “Application Performance Directed Voltage Scaling,” *IEEE Trans. on VLSI,* vol. 9, no. 4, pp. 76-85, 2002.

[43] Banerhee.K and Malhotra.A., “Global interconnect warming,” *IEEE Circuits and Devices Magazine,* vol. 17, no. 5, pp. 16-32, 2001.

[44] Stan. M. and Burleson.W., “Bus-invert coding for low-power I/O,” *IEEE Trans. on VLSI,* pp. 49-58, 1995.

[45] Patel K. N. and Markov. I. L., “Error-correction and crosstalk avoidance in dsm busses,” *Proc. of Int. Wksp. on System-Level Interconnect Prediction,* pp. 9-14, 2003.

[46] Jone.W.B, Wang.J.S, Lu.H.I.,Hsu.I.P., “Design theory and implementation for low-power segmented bus systems,” *ACm Trans.on Design Automation in Electronics Systems,* vol. 8, no. 1, pp. 38-54, 2003.

[47] Bishop.B. and Irwin M.J., “Databus charge recovery: Practical considerations,” *Int. Symp on Low Power Electronics and Design,* pp. 85-87, 1999.

[48] Wang.H., Peh.L.S., and Malik.S., “Power driven desing of router microarchitectures in on-chip networks,” *Proc. of Int. Symp. on Microarchitecture, IEEE Computer Society,* pp. 105-116, 2003.

[49] Luz. V.D.L., Kademir.M and Kolcu.I., “Automatic data migration for reducing energy consumption in multi-bank memory systems,” *Proc. of Conf. on Design Automation,* pp. 213-218, 2002.

[50] Kin.J.,Gupta.M. and Mangione-Smith.W.H., “The filter cache: An energy efficient memory structure,” *Proc. of Int. Symp. on Microarchitecture, IEEE Computer Society,* pp. 184-193, 1997.

[51] Hu.J.,Vijayakrishnan.N.,Irwin.M.and Kandemir.M., “Using dynamic branch behavior for power-efficient instruction fetch,” *Proc. of IEEE Computer Society Annual Symp. on VLSI,* pp. 127-132, 2003.

[52] Raja.T., Agrawal V.D. and Bushnell M.L., “CMOS Circuit Design for Minimum Dynamic Power and Highest Speed,” *Proc. of Int. Conf. on VLSI Design ,* pp. 1035-1040, 2004.

[53] Powell.M, Yang. S.H.,Falsafi.B, Roy.K., “Reducing leakage in a high-performance deep-submicron instruction cache,” *IEEE Trans. on VLSI Systems,* vol. 9, no. 1, pp. 77-90, 2001.

[54] Kim. N.S., Flautner.K., Blaauw.D. and Mudge.T., “Drowsy instruction caches: leakage power reduction using dynamic voltage scaling and cache sub-bank prediction,” *Proc. of ACM/IEEE Int. Symp. on Microarchitecture, IEEE Computer Society,* pp. 219-230, 2002.

[55] Buyuktosunoglu.A., Schuster.S., Brooks.D., Bose. P., “An adaptive issue queue for reduced power at high performance,” *Proc. of Int. Wksp. on Power-Aware Computer Systems,* pp. 25-39, 2001.

[56] Huges.C.J,Adve.S.V., “A formal approach to frequent energy adaptions for multimedia applications,” *Procs.Int. Symp. on Computer Architecture (ISCA'04), IEEE Computer Society,* pp.

138-152, 2004.

[57] Huges.C.J., Srinivasan. J. and Adve. S.V., “Saving energy with architectural and frequency adaption for multimedia applications,” *Proc. of Int. Symp. on Microarchitecture, IEEE Computer Society,* pp. 250-261, 2001.

[58] Iyer.A. and Marculescu. D., “Power aware microarchitecture resource scaling,” *Proc. of Conf. on Design automation and Test in Europe, IEEE Press,* pp. 190-196, 2001.

[59] Huang.M.C.,Renau.J. and Torrellas.J., “Potional adaptation of processors: application to energy reduction,” *Proc. of Int. Symp. on Computer Architecture(ISCA'03),* pp. 157-168, 2003.

[60] Grunwald, et al., “Policies for Dynamic Clock Scheduling,” *Proc. of Int. Symp. on Operating System Design and Implementation, Usenix Association,* 2000.

[61] Simunic, et al., “Dynamic Voltage Scaling and Power Management for Portable Systems,” *Proc. of Int. Conf. on Design Automation,* 2001.

[62] Fan.X,Ellis.C.S. and Lebeck.A.R., “Synergy between power-aware memory sytems and processor voltage scaling,” *Proc. of Int. wksp. on Power-Aware Computer Systems,* pp. 164-179, 2003.

[63] Shin.D., Kim.J. and Lee.S., “Low-energy intra-taskvoltage scheduling using static timing analysis,” *Proc. of Int. Conf. on Design Automation, ACM Press,* pp. 438-443, 2001.

[64] Hsu.C.H. and Kremer.U., “The design, implementation and evaluation of compiler algorithm for CPU energy reduction,” *Proc.of Cong. on Programming Language Design and Implementation,* pp. 38-48, 2003.

[65] Hue.C. and Feng.W., “Effective dynamic voltage Scaling through cpu-boundedness detection,” *Proc. of Int. Wksp. on Power Aware Computing Systems, IEEE Computter Society,* pp. 122-131, 2004.

[66] Marklis.G.,Scott.M.,Semeraro.G. and Albonesi D.H., “Profile-based dynamic voltage and frequency scaling for a multiple clock domain microprocessor,” *Proc. of Int. Symp. on Computer Architecture,ACM Press.,* pp. 14-27, 2003.

[67] Magklis.G.,Semeraro.G.,Albonesi.D.,Dropsho.S.,Dwarkadas.S. and Scott.M., “Dynamic frequency and voltage scaling form multiple clock-domain microprocessor,” *IEEE Trans. on Microprocessors,* vol. 23, no. 6, pp. 62-68, 2003.

[68] X. Guan and Y.Fei, “Reducing power consumption of embedded processors through register file partitioning and compiler support,” *Proc. of Int. Conf. on Application-Specific Systems, Architectures and Processors,* pp. 269-274, 2008.

[69] T.J.Lin,S.K.Chen, T.Y.Kuo,C.W.Liu and P.C.Hsiao, “Design and implementation of a high-performance and complexx-effective VLIW DSP for multimedia applications,” *Int.Journal of VLSI*

Signal Processing, vol. 51, no. 3, pp. 209-223, 2008.

[70] D.Scoones, “Power Management Technology for Portable Devices,” *Proc. of Int. Conf. on Energy - Efficient Devices,* pp. 123-129, 2007.

[71] X.Ma, M.Dong, L.Zhong and Z. Deng, “Statistical Power Consumption Analysis and Modeling for GPU-Based Computing,” *Proc. of Int. Wksp. on Power Aware Computing and Systems,* 2009.

[72] D.Snowdon, “OS-Level Power Management,” *Ph.D. Thesis, School of Computer Science and Engineering, University of New South Wales,* 2010.

[73] Findlay Shearer, Power Management in Mobile Devices, Newnes, 2008.

[74] K.Scoones , “Power Management tchnologies for portable devices,” *Int. Conf. on Energy - Efficient Design ,* 2007.

[75] Brad Smith, “ARM and Intel battle over the mobile's chip's future,” *IEEE Trans. on Computers,* vol. 41, no. 5, pp. 15-18, 2008.

[76] Intel, “High-performance energy efficient processors for embedded market segments,” http://www.intel.com/design/embedded/downloads/315336.pdf, 2011.

[77] Steve Furber, ARM-System on-chip Architecture, 3rd Edition, Pearson Education, 2009.

[78] Kiritkumar Bhatt, A I Trivedi, “Power optimized embedded processor design with parallel pipelining,” *Int. Journal of PDCS,* vol. 4, no. 1, pp. 54-60, 2012.

[79] Kiritkumar Bhatt, A I Trivedi, “Low-power pipelined processor design, its verification and implementation on FPGA,” *IEEE Int. Conf. on Electronics and Computer Technology,* vol. 1, no. 1, pp. 91-95, 2012.

[80] P.Francesco, P.Antonio, et.al., “Energy efficient microprocessor systems-on-chip for embedded computing: exploring programming models and their architectural support,” *IEEE Trans. on Computers,* vol. 56, pp. 606-621, 2007.

[81] Gautam P.,Parthsarthy R.and Karthi Balasubhramanian, “Low-power pipelined MIPS processor design,” *Proc. of Int. Conf. on ISIC,* pp. 462-465, 2009.

[82] A A S Pejman Lofti Kamran, Amir Mohammad Rohmani and A A Kusha, “Stall power reduction in pipelined atchitecture processor,” *Proc. of Int. Conf. on VLSI Design ,* pp. 541-546, 2008.