Politecnico di Torino

Microelectronic Systems

DLX Microprocessor: Design & Development

Final Project Report

Master degree in Electronics Engineering
Master degree in Computer Engineering

Referents: Prof. Mariagrazia Graziano, Giulia Santoro

Authors: Group 37, Group 51
Luca Mocerino (37), Biagio Feraco (51)

October 14, 2016

Contents

1 Introduction **1**

2 Implementation **2**
2.1 Data Path . . . 2
2.1.1 IF: Instruction Fetch (Stage 1) . . . 4
2.1.2 ID: Instruction Decode (Stage 2) . . . 5
2.1.3 EXE: Execution (Stage 3) . . . 7
2.1.4 MEM: Memory Stage (Stage 4) . . . 8
2.1.5 WB: Write Back (Stage 5) . . . 8
2.2 Instruction ROM . . . 8
2.3 Data RAM . . . 8
2.4 Control Unit . . . 8

3 Synthesis and Optimization **9**
3.1 65nm tech . . . 9
3.2 45nm tech . . . 10
3.3 Physical Design . . . 11

4 Conclusions **12**

5 References **13**

A Instruction Set **14**

B Schematics and Figures **15**

Summary

In this project is defined the implementation of a 5-stage-pipelined DLX processor. It contains several additional features that extend the basic version, including:

- The basic instruction set plus some additional instructions (refer to Appendix A for the whole list)
- **Data path**: In particular for the ALU we exploited many different structural implementations learned during the class and lab session: the Pentium4 Adder, multiplier using Booth's algorithm, the logic unit of NiagaraT2 and a structural version of a comparator.
- **Data hazard** : An integrated Hazard Detection Unit able to solve RAR, WAR related to load instruction.
- **Forwarding**: In execution stage we have a Forwarding Unit able to propagate the ALU result in order to avoid EX/MEM and MEM/WB hazards.
- Synthesis and Optimization (using different methods and library) .
- Physical design.

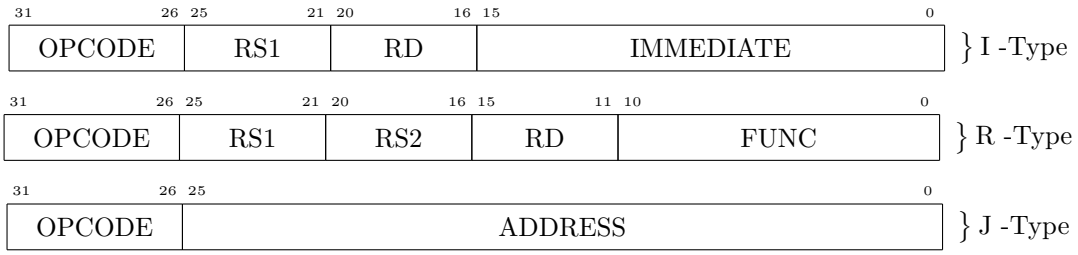
CHAPTER 1

Introduction

The DLX is a 32-bit RISC processor architecture designed by John L. Hennessy and David A. Patterson introduced in 1990s. The type is Register-Register & Load-store, with fixed encoding on 32-bits, based on a 5-stage in-order integer pipeline, that are:

- Fetch;
- Decode;
- Execute;
- Memory;
- Write back.

Moreover it allows two addressing mode, using a memory addressable Big-endian, with an immediate value or using displacement. The register file is composed by 32-bit general purpose register (R0,R1,...R31) in which R0 is always zero and R1 is used for return to subroutine. This architecture provides three groups of instruction with the following description and fields:



CHAPTER 2

Implementation

2.1 Data Path

The Figure 2.1 shows the schematic of the data path. It contains several units, each one related to the specific stage. The following paragraphs describe each of them, by listing all the main characteristics and their functionality.

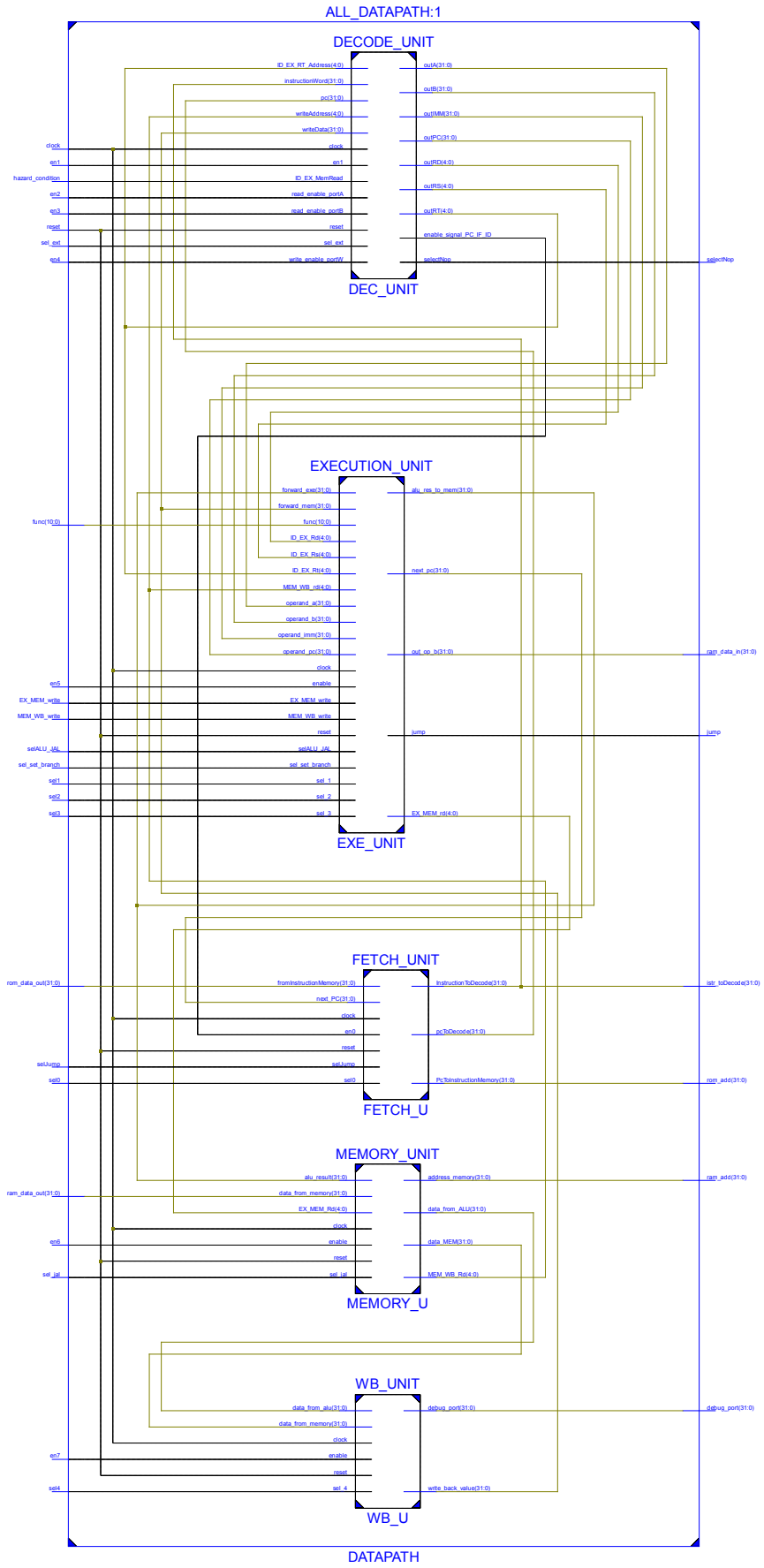


Figure 2.1: Data Path Schematic.

2.1.1 IF: Instruction Fetch (Stage 1)

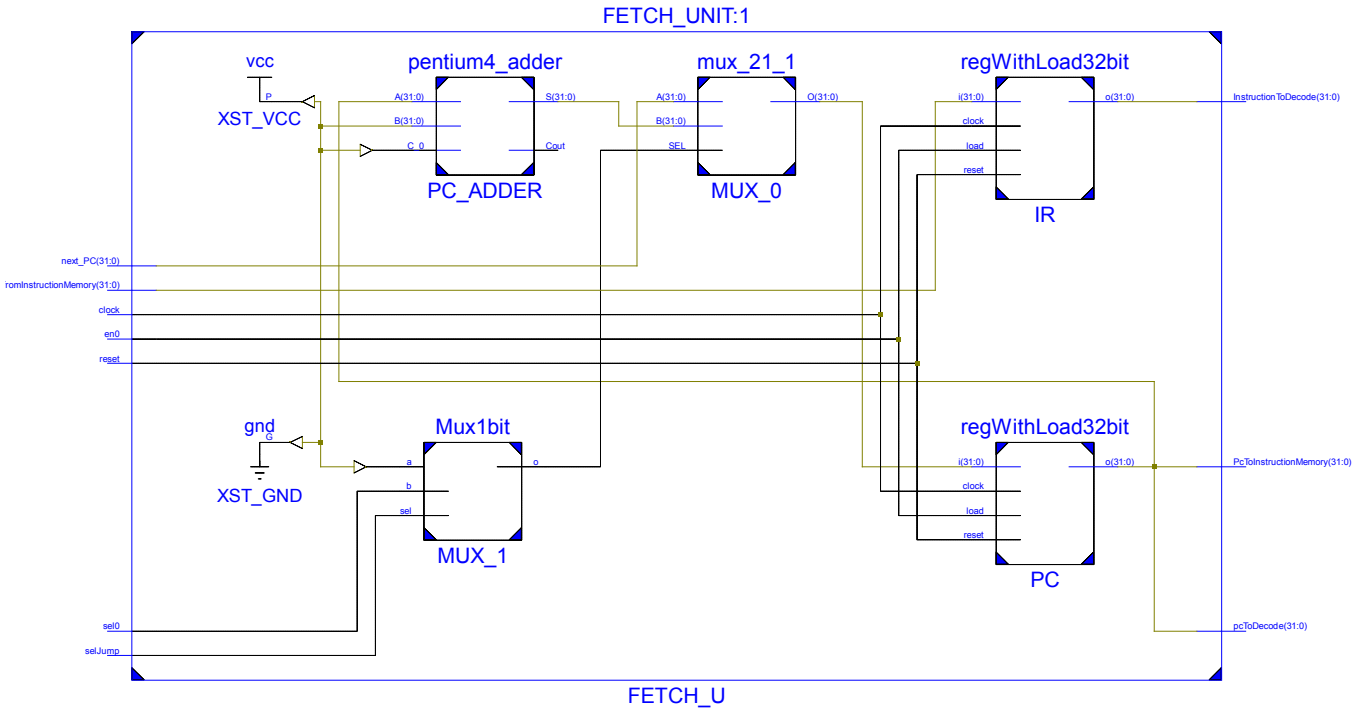


Figure 2.2: Fetch Stage Schematic.

The figure 2.2 shows the schematic of the Fetch Stage. It is composed of four parts:

- Program Counter (PC), that is the register storing the address for accessing the instruction memory;
- the Pentium 4 Adder, computing the Next Program Counter (NPC) Value (PC+4);
- Instruction Register (IR), that is the register storing the instruction coming from the Instruction Memory;
- two Multiplexers, that, depending on if the next instruction is a taken branch or a Jump, select among the Next Program Counter and the new value of the Program Counter

2.1.2 ID: Instruction Decode (Stage 2)

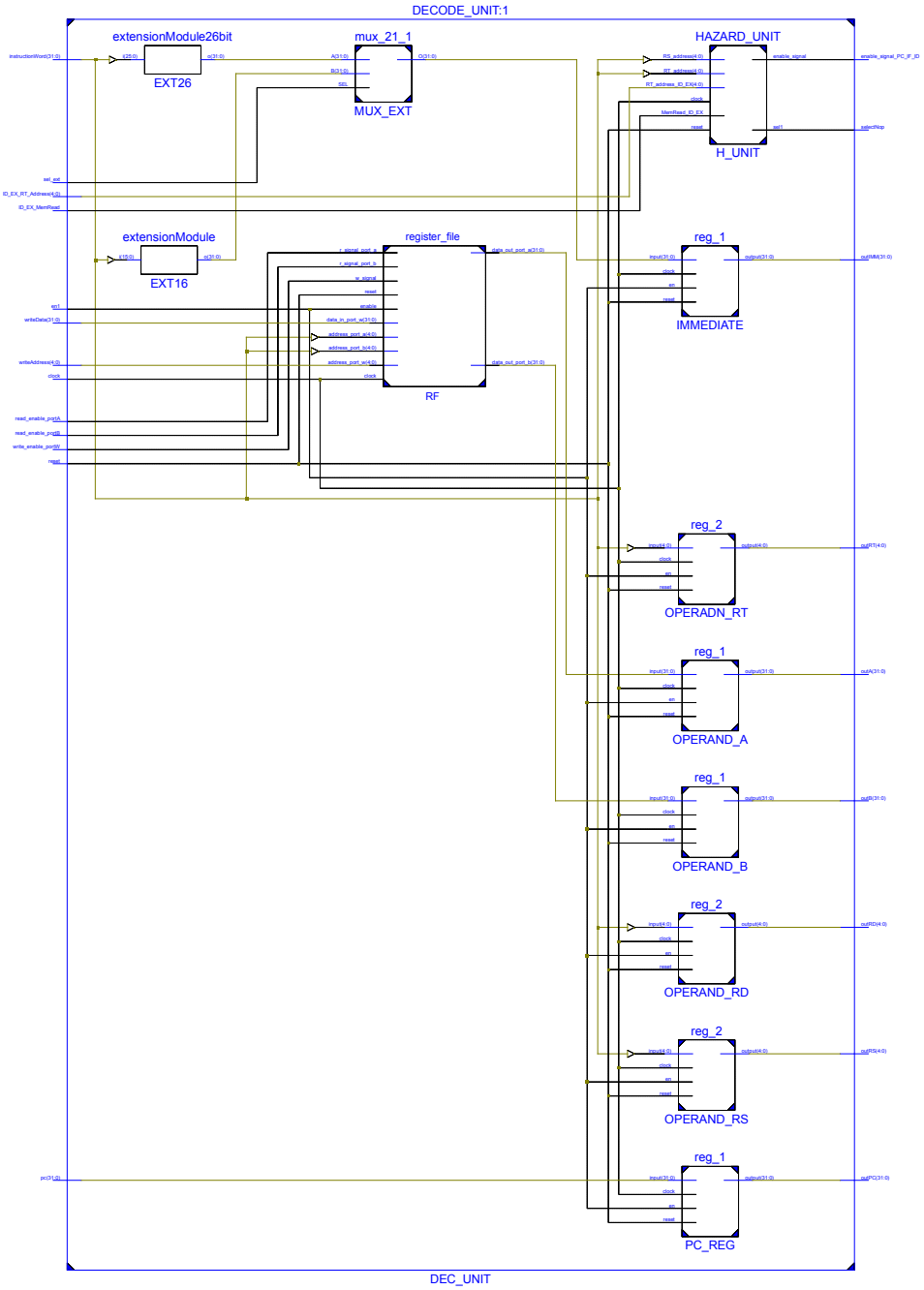


Figure 2.3: Decode Stage Schematic.

The Decode Stage, that is shown in the figure 2.3, is composed of four parts:

- a 2R1W Register File (two read ports and one write port), composed of 32 General Purpose Registers (R0, R1, ..., R31) each of them of 32-bit. The registers are accessed asynchronously both for the Read and Write Operation;
- a block of seven registers that are:

  - OPERAND_RD;
  - OPERAND_RS;
  - OPERAND_RT;

  each of them receives the corresponding instruction word field;

  - OPERAND_A;
  - OPERAND_B;

  each of them receives the operands coming from the output of the Register File;

  - the Pipeline Register (PC_REG);

  taking the Next Program Counter to forward it to the next stage;

- the IMMEDIATE Register, that receives the Immediate Value extended on 32 bit from the two extension modules;
- the Hazard Detection Unit (H_UNIT), that, depending on four inputs signals, generates two outputs:
  - the first one is used to disable the IF/ID register if an hazard is detected;
  - the second one is sent to the Control Unit, that will force a NOP instruction if this signal is asserted.

2.1.3 EXE: Execution (Stage 3)

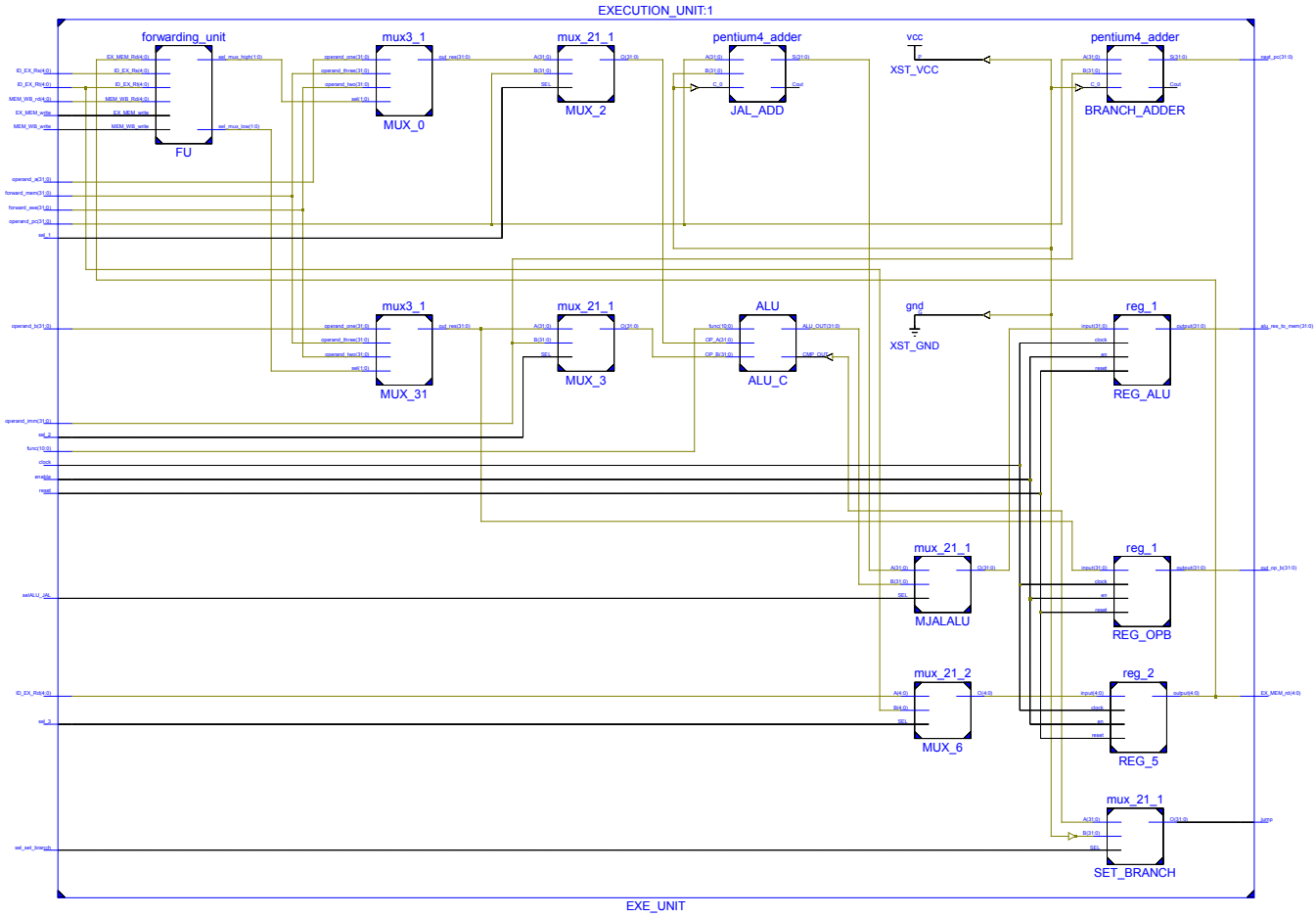


Figure 2.4: Execution Stage Schematic.

As it is shown in the figure 2.4, the Execution Stage is composed of several parts:

- a structural ALU, in which five different modules are defined, some of them following the implementation explained during the lessons:
  - the Pentium 4 Adder, performing Addiction and Subtraction on two 32-bits operands with a fast propagation of the carry;
  - the Booth Multiplier, performing the Multiplication of two 16-bits operands and returning a 32-bits result;
  - the T2 Logical Unit(Figure Appendix B.1), that performs all the set of logical operations with two levels of NAND gates, described in a structural way;
  - a Shifter, performing arithmetical and logical shift operations;
  - a Comparator (Figure Appendix B.2), also described in a structural way.

  The ALU receives the FUNC signal, and according to its value the proper configuration to all the sub-units are selected.

- The Forwarding Unit, that detects data hazards involving arithmetic instructions by comparing the destination registers of the previous instructions with the source registers of the current one. Taking the results from the pipeline registers before they are written back into the RF, Hazards are therefore avoided.

2.1.4 MEM: Memory Stage (Stage 4)

Memory Stage consists of four registers:

- Three of them are pipeline register that stores ALU result and address and data from memory
- **M_JAL**: is used for instruction JAL, in order to select the last register of the register file

2.1.5 WB: Write Back (Stage 5)

In this stage there is a pipeline register in order to allow a signal for debug and a multiplexer used to select the write back signal that comes fro ALU or Data RAM.

2.2 Instruction ROM

This is the memory fully asynchronous storing instructions to be executed.

2.3 Data RAM

This is the memory accessed by Load/Store instructions. It is an asynchronous *reset* and *read*, with a synchronous *write*. The implementation allows to register R0 to be always at zero.

2.4 Control Unit

In the Figure Appendix B.3 it is shown the top entity schematic of the DLX.
The type of Control Unit designed is the **Hardwired** style. Therefore the control words are stored in a LUT and at each clock event, according to instruction, they are propagated through the pipeline. Moreover, there is a FSM controlling the relative and absolute jumps/branches according some input values coming from the pipeline. The control word is 21 bit and contains all control signal to datapath. Among that signals we can notice:

- 2 signals used for forwarding conditions
- 1 signal used for hazard condition

The remaining signal are all to drive multiplexers and enable signals for register in datapath.
Other signals are *OPCODE* (6 bit) in order to identify the type of instruction and the *FUNC* (11 bit). With this input the control word is generated. Moreover there are 2 additional signals that detect when it is necessary to stall the pipeline after a load, stalling the pipeline for one clock cycle or the signal that enable to stall the pipeline for 2 clock cycles after the branch/jump. Also the FUNC signal for the ALU was generated by control unit but it is not included in the control word.

CHAPTER 3

Synthesis and Optimization

In this chapter we want to describe all steps of the synthesis and optimization phase of our DLX.
We used two different technologies for synthesis: **65nm** provided by STMicroelectronic and **45nm**.
Moreover, we have chosen clocks accordingly to three main categories of use:

- **High performance**: 333 - 200 MHz
- **Medium performance**: 100 MHz
- **Low performance**: 20 MHz

3.1 65nm tech

For this technology we ran different kinds of synthesis in order to underline the differences in terms of power consumption among the implementations.
In the following table are reported all values and kind of techniques used for this technology:

Clock period (ns)	Dynamic Power	Leakage Power	Clock gating
3	2587.6 uW	16.7396 uW	No
3	2561.2 uW	16.7498 uW	Yes
10	714.1 uW	10.2889 uW	Yes
50	144.4 uW	10.6942 uW	No
50	142.5 uW	10.2347 uW	Yes

For this technology is used another technique more, **dual−Vth assignment**: are used two different kind of cells for mapping with a low (**LVT**) and high (**HVT**) value of threshold voltage. The

mapping use LVT cells the critical path and HVT all the rest.

3.2 45nm tech

For this technology we ran different kinds of synthesis, using *clock gating* technique in order to underline the differences in terms of power consumption among the implementations.

Clock period (ns)	Dynamic Power	Leakage Power	Area	Clock gating
No	5.3747 mW	430.7693 uW	7435.23	No
5	1.6796 mW	494.45 uW	24177.80	No
5	1.8457 mW	482.02 uW	23596.32	Yes
10	811.007 uW	479.70 uW	23150.50	Yes
50	159.925 uW	470.27 uW	22880.25	Yes

It can be observed that clock gating reduces power consumption as we expected.

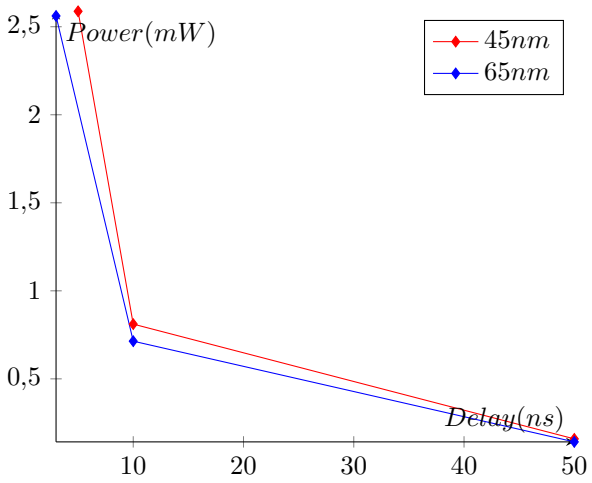


Figure 3.1: Dynamic Power-Delay variation

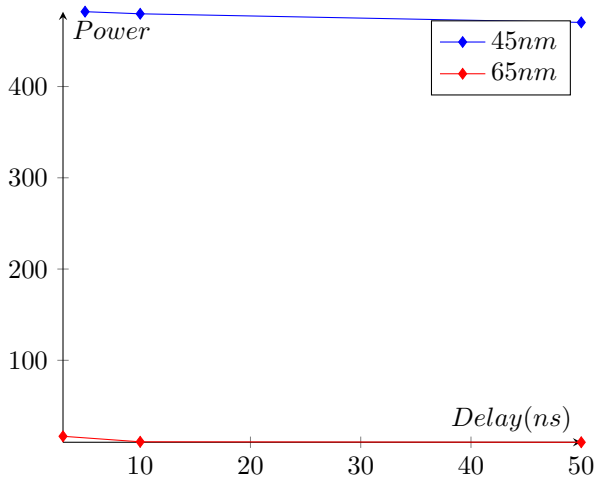


Figure 3.2: Leakage Power-Delay variation

The figures 3.1 and 3.2 represents the relative variation between the two technologies, changing clock period. An interesting thing we can observe is that the dynamic power between the two technologies is more or less the same but we have a meaningfull increase for newer technology.

3.3 Physical Design

Using the Cadence Encounter Tool, we generated the layout of the DLX and checked its signal integrity critical issues. After the synthesis of the DLX, fixing the clock at 5 ns, we performed step-by-step the procedure used during the laboratory and looking at the results and the timing reports, we observed that it did not incurred in violations. In the Figure Appendix B.4 it is shown the final layout.

CHAPTER 4

Conclusions

After the implementation we tested the DLX using some benchmarks in order to exploit the wide range of instructions and hazard that could be occurred. All that we implemented works according specification described in this document. Moreover, we can say that the implementation of DLX is well structured and modular, in order to allow some improvement. But during our design and implementation we followed the actual main thread, that is designed for low power. For a future work:

- Floating Point Unit and others arithmetic units such as Division and Square Root;
- Branch Prediction Mechanism *(Saturating counter, two-level adaptive predictor, etc...)*;
- **MMU** (Memory Management Unit) and different level caches;
- Interrupt and Exception handling;
- Multi-core organization.

CHAPTER 5

References

1. Mariagrazia Graziano. *"Microelectronic systems"*. Politecnico di Torino, Master's degree in Computer Engineering. Torino, March - June 2016. Lectures.

2. D. Patterson, J. Hennessy. *"Computer Organization and Design, Fourth Edition, Fourth Edition: The Hardware/Software Interface"* (4th ed.) Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 2008. Book

3. J. Hennessy, D. Patterson. *"Computer Architecture, Fifth Edition: A Quantitative Approach (The Morgan Kaufmann Series in Computer Architecture and Design)"* (5th ed.) Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 2011. Book.

APPENDIX A

Instruction Set

Basic	Pro
add	seq
addi	seqi
and	sgt
andi	sgt
beqz	sgti
bnez	slt
j	slti
jal	mul
lw	
nop	
or	
ori	
sge	
sgei	
sle	
slei	
sll	
slli	
sne	
snei	
srl	
srli	
sub	
subi	
sw	
xor	
xori	

APPENDIX B

Schematics and Figures

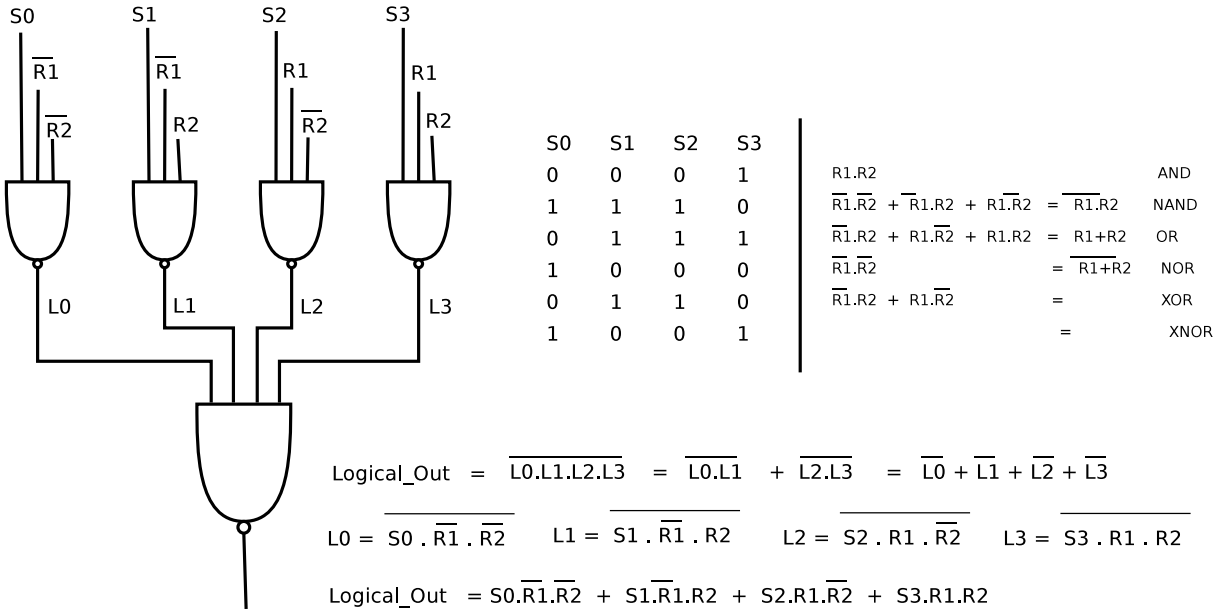


Figure B.1: Logicals T2.

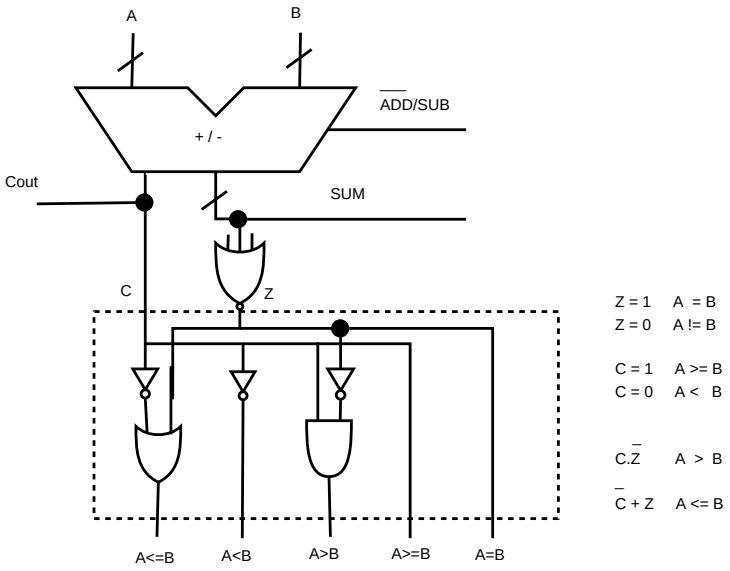


Figure B.2: Schematic of Comparator and Corresponding Results.

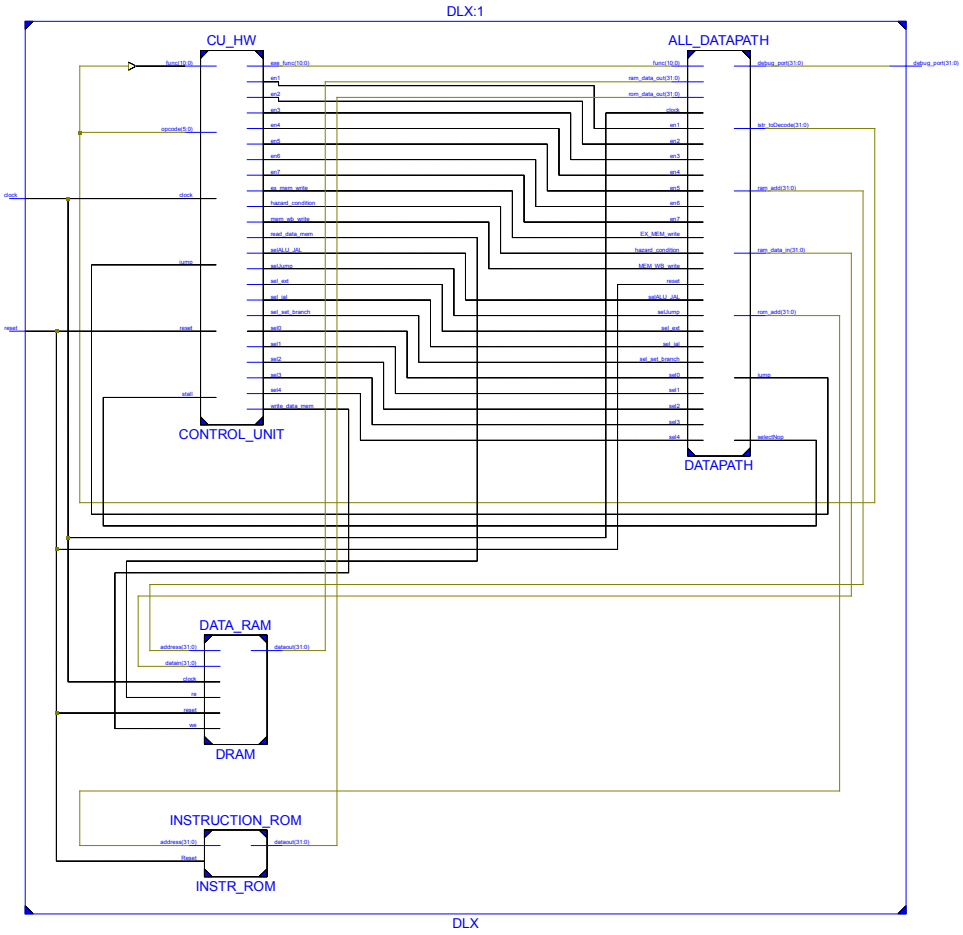


Figure B.3: DLX Top Entity Schematic.

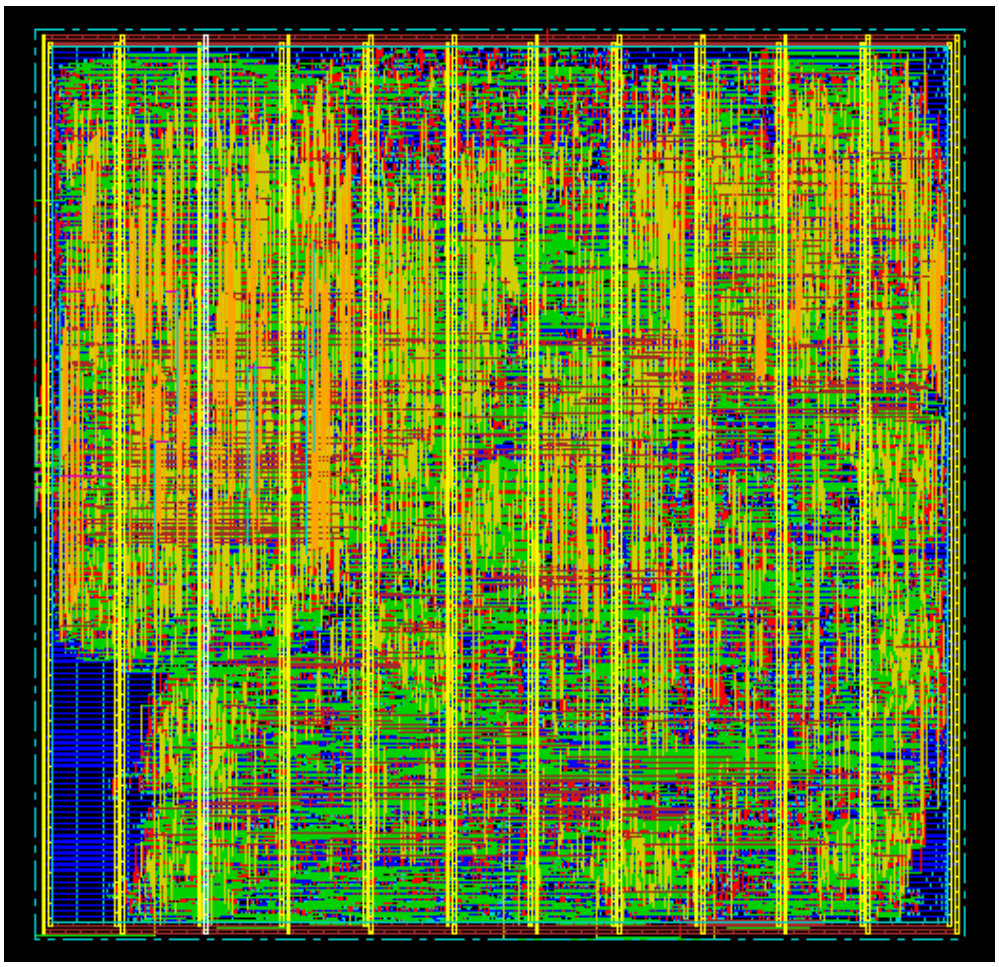


Figure B.4: Physical Design of the DLX