Politecnico di Torino

Microelectronic Systems

DLX Microprocessor: Design & Development

Final Project Report

Masters degree in Computer Engineering

Referents: Prof. Mariagrazia Graziano, Giovanna Turvani

Authors: Group 37

Juan José Restrepo, Michael Elias

October 19, 2023

Contents

CHAPTER 1

DLX overview

This document describes the design of a DLX processor from the RTL level to the Floorplan level, considering aspects like synthesis, optimization, and performance. It also mentions design choices, trade-offs, and implementations selected during the conception of the architecture.

The DLX processor is an RISC processor designed by John L. Hennessy and David A. Patterson. It consists of a simple 32-bit load/store architecture and it is widely used for teaching purposes. In particular, its instructions set is divided into 3 different types of instructions: *R-type* (Load/Stores and conditional branches), *I-type* (Register-register ALU operations), and *J-type* (Jump and jump link instructions).

In all the three types, the 6-bit *OPCODE* field is always present. Then, the remaining 26 bits assume different meanings depending on the instruction type, as it is shown:

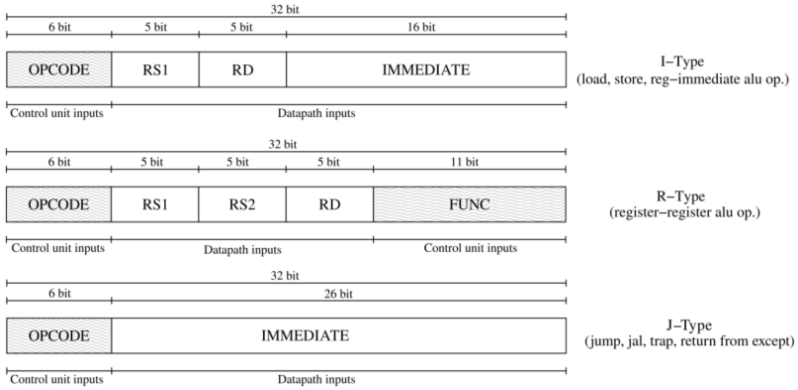

Figure 1.1: DLX instruction types

Figure 1.1 exhibits the different parts that each instruction type has. On the one hand, all of them have an *OPCODE* which identifies the instruction that is going to be executed by the processor and also goes to the control unit in order to generate the proper control signals for the instruction in execution. On the other hand, the *R-type* instructions have another field called *FUNC* which allows the architecture to know which kind of R-type operation is going to be performed by the processor. This appears as a design choice because all the R-type instructions have the same control signals except one called *ALU_OPCODE* which is generated depending on the *FUNC* field of the R-type instruction. In brief, here is a more detailed explanation of the instruction fields:

1. I-type

   (a) OPCODE: Identifies the instruction
   (b) RS1: Source register
   (c) RD: Destination register
   (d) IMMEDIATE: Offset value that is added to the value stored in the Source register

2. R-type
   (a) OPCODE: Identifies the instruction
   (b) RS1: Source register 1
   (c) RS2: Sourcer register 2
   (d) RD: Destination register
   (e) FUNC: Identifies the kind of R-type instruction

3. J-type
   (a) OPCODE: Identifies the instruction
   (b) IMMEDIATE: Offset value which will be used as an address

Additionally, the basic instruction set is composed of 27 instructions: add, addi, and, and, beqz, bnez, j, jal, lw, nop, or, ori, sge, sgei, sle, slei, sll, slli, sne, snei, srl, srli, sub, subi, sw, xor, xori. For further details about the instruction set, please refer to Appendix A.

The DLX processor has a 5-stages pipelined architecture. Thus, a new instruction is started at each clock cycle and different resources work on different instructions at the same time. Moreover, at every clock cycle, each resource can be used only for one purpose which means that separate instruction and data memories must be used, the register file has to be used in two stages(*Decode* and *Write back*), and the Program Counter must change in the first stage(*Instruction fetch*).

One important consideration is that this architecture does not handle Hazards. Therefore, the system is vulnerable to two types of them:

1. Data Hazards: an instruction depends on the result of a previous instruction.

2. Control Hazards: depend on branches and other instructions that change the Program Counter. In this case, this is present when executing *beqz*, *bnez*, *j* and *jal*.

A possible solution to this problem is to force the pipeline to stall, i.e. to block instructions for one or more clock cycles. However, the system does not do it by itself; therefore, the programmer should consider stalling the execution by using the *NOP* instruction(Please refer to Appendix C).

Finally, the architecture has three main parts: Datapath, Control unit and Memories(Instruction memory and Data memory). Figure 1.2 exhibits the connections among these parts.

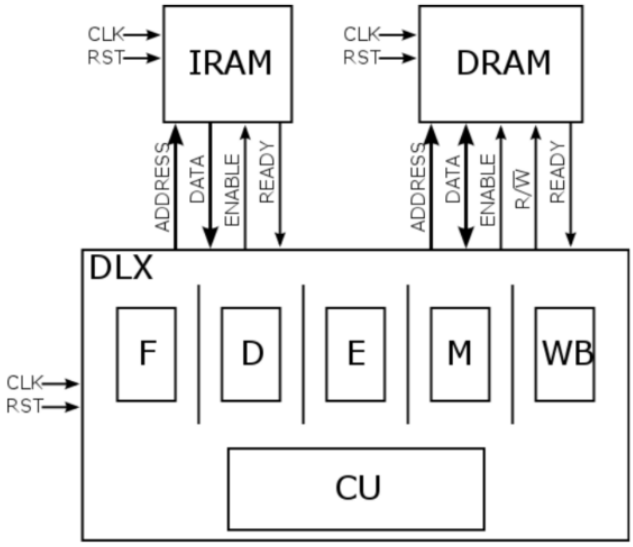

Figure 1.2: DLX architecture

CHAPTER 2

Datapath

2.1 General Description

The Datapath is shown in the Figure 2.1. Every instruction can be implemented in at most five clock cycles:

1. *Instruction fetch* cycle: Send out the *PC*(Program counter) and fetch the instruction from memory into the *IR*(instruction register) and increment the *PC* by 4 to address the next instruction. The other register called *NPC* is used to hold the next PC(PC+4).

2. *Instruction decode* cycle: Decode the instruction and access the *RF*(Register file) to read the registers. Then, the outputs of the *RF* are read into two temporary registers called *A* and *B*. In addition, the register *Imm* holds the sign-extended value coming from the output of the *IR*.

3. *Execution/effective address* cycle: It is where arithmetic and logic operations are executed. The execution unit interprets instructions fetched from the IR and carries out the necessary operations on data stored in registers or memory.

4. *Memory access/branch completation* cycle: Access memory if it is needed. Likewise:

   (a) If the instruction is a load, data is returned from *Data memory* and is placed in the *LMD*(Load Memory Data) register.

   (b) If the instruction is a store, the data from the register B is written into *Data memory*.

   (c) If the instruction is a branch, the PC is updated with the new address.

5. *Write back* cycle: Write the result into the *RF*.

Finally, an important aspect to consider is that not all the sequential logic is Rising-edge sensitive. In specific, they are organized as follows:

1. Falling-edge sensitive: *PC*, *LMD* and *Data memory*.

2. Rising-edge sensitive: *NPC*, *IR*, *Register File*, *NPC2*, *A*, *B*, *Imm*, *RD1*, *NPC3*, *Cond*, *ALU output*, *Data mem* and *RD2*.

2.2 Fetch stage

The fetch stage is composed of an *ADDER*, *IRAM* (external to the DLX) and three 32-bit registers. The role of each component is explained as follows:

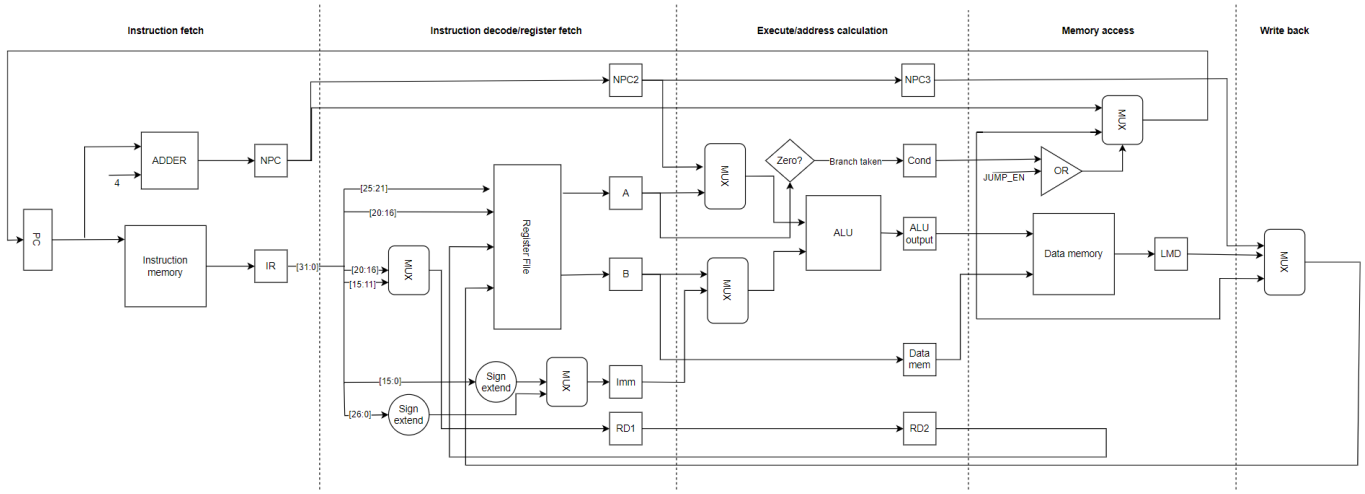

Figure 2.1: DLX Datapath

1. *PC*: Register that generates the address to read the *IRAM* and get the corresponding instruction.

2. *Adder*: Calculates the value to update the *Program counter* with the value PC + 4.

3. *IRAM*: Even though the IRAM is placed outside the *Datapath*, it was mentioned for ease of explanation. The memory is filled once by reading from the assembly program after it has been compiled.

   Modern CPUs perform read and write with higher efficiency when the data is aligned, which means that the data's memory address is a multiple of the data size. In this case, the architecture is a 32-bit, hence the data is aligned if it is stored in four consecutive bytes and the first byte lies on a 4-byte boundary.

   Consequently, the memory has a data width of 1 byte(8 bits); therefore, the 32-bit instruction is read from the file and split into 4, and each byte is saved in a contiguous memory location. In this case, the system uses a Big Endian notion to store the bytes.

   Finally, the memory has asynchronous read and asynchronous reset.

2.3 Decode stage

The decode unit is composed of a Register File (RF), three 32-bit registers (A, B and IMM) holding the data of what could possibly be operands to the ALU, a 32-bit register named NPC2, a 5-bit register Rd, two sign-extend components (16 and 26 bits), three 2x1 MUXes (two for register Rd1 and the other for register IMM). See Figure 2.3 for a more detailed view of the Decode unit (the only difference in this implementation is that there is no "in1" register").

1. **Register NPC2**: This register plays a role in the jal instruction, it stores the value of NPC until this value is stored in the register 31.

2. **Register Rd1**: Retains and passes on the address where specific data will be written in the RF.

3. **Sign extend**: Simply extends "Immediate" from 16 to 32 bits or from 26 to 32 bits while keeping the same sign. The 16-bit one is used for I-type instructions and the 26-bit is used for J-type instructions.

4. **2x1 MUXes for Rd1**: Looking into the encoding of I-type and R-type instructions, it can be seen that Rd has different bits depending on the instruction type; therefore, this MUX is used to dedicate bits [20-16] (I-type) or bits [15-11] (R-type) to the Rd register. The second MUX is used only for jal instruction, the input "11111" is selected in that case to force the value of PC+4 to be written into register 32.

5. **2x1 MUX for IMM**: This MUX is used to choose the source of the immediate value stored in the IMM reg. The immediate could either be an immediate from an I-type instruction (16 bits) or a J-type instruction (26 bits).

2.3.1 Register File unit

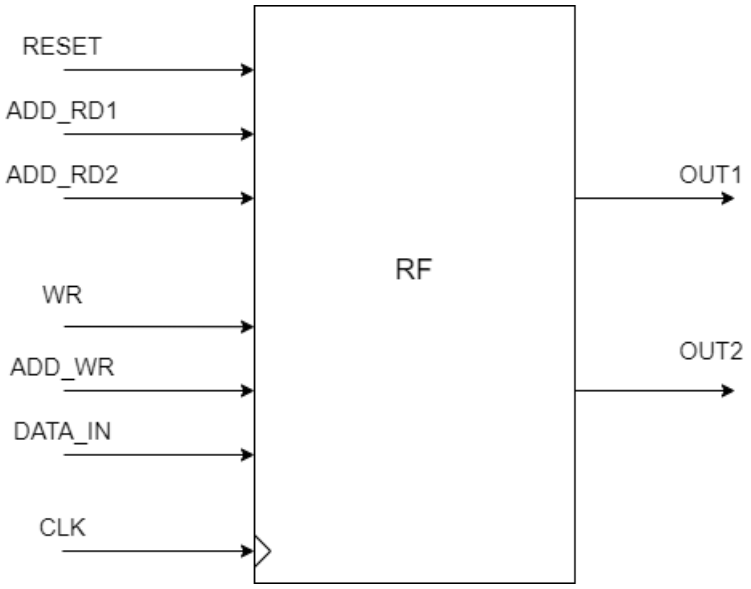

Figure 2.2: Register file general overview.

The RF is the core of the Decode stage, it has thirty two 32-bit general-purpose registers, named R0, R1, ... R31. R0 cannot be overwritten, it is always 0 (R0 is made inaccessible in hardware, it can only be read). It has 2 Read ports and 2 output ports, 1 write port, write enable, clock and reset. A general schematic of the RF is shown in Figure 2.2.
Asynchronous reading was implemented to avoid needing an extra clock cycle when filling the registers A and B. However synchronous writing was used as it does not affect the functioning of the DLX. f

2.4 Execution stage

The execution unit is composed of two 2x1 MUXes, a *Zero?* component that checks if the input is equal to 0, an ALU, a 1-bit register *Cond* (stores the output of zero?), a third MUX, an AND gate and a 32-bit register that stores the output of the ALU. The first two MUXes are controlled in a way such that the ALU can have the following input operands:

1. Output of register A and Output of register B (**Memory reference**)

2. Output of register A Output of register Imm (**ALU operation**)

3. Output of register A and Output of register Imm(**Register-Immediate ALU**)

4. Output of register *NPC* and Output of register *Imm*(**Branch Address calculation**)

The third MUX either selects the output of the *Zero?* or its negated output, depending on whether it is a beqz or bnez instruction, respectively.
The AND gate has the output of the previously mentioned MUX and the *EQ COND* control signal, this AND gate was placed there because whenever *EQ COND* is zero the *COND* register should also be zero (Enable signal for *COND* register is set to '1').
The unit *Zero?* checks if the output of the register A is equal to 0 (useful for working with branches) and sends the result to the register *Cond*.
A 5-bit register Rd2 was added, the sole purpose of the register is to retain the value of Rd (discussed in the decode stage). See Figure 2.3.

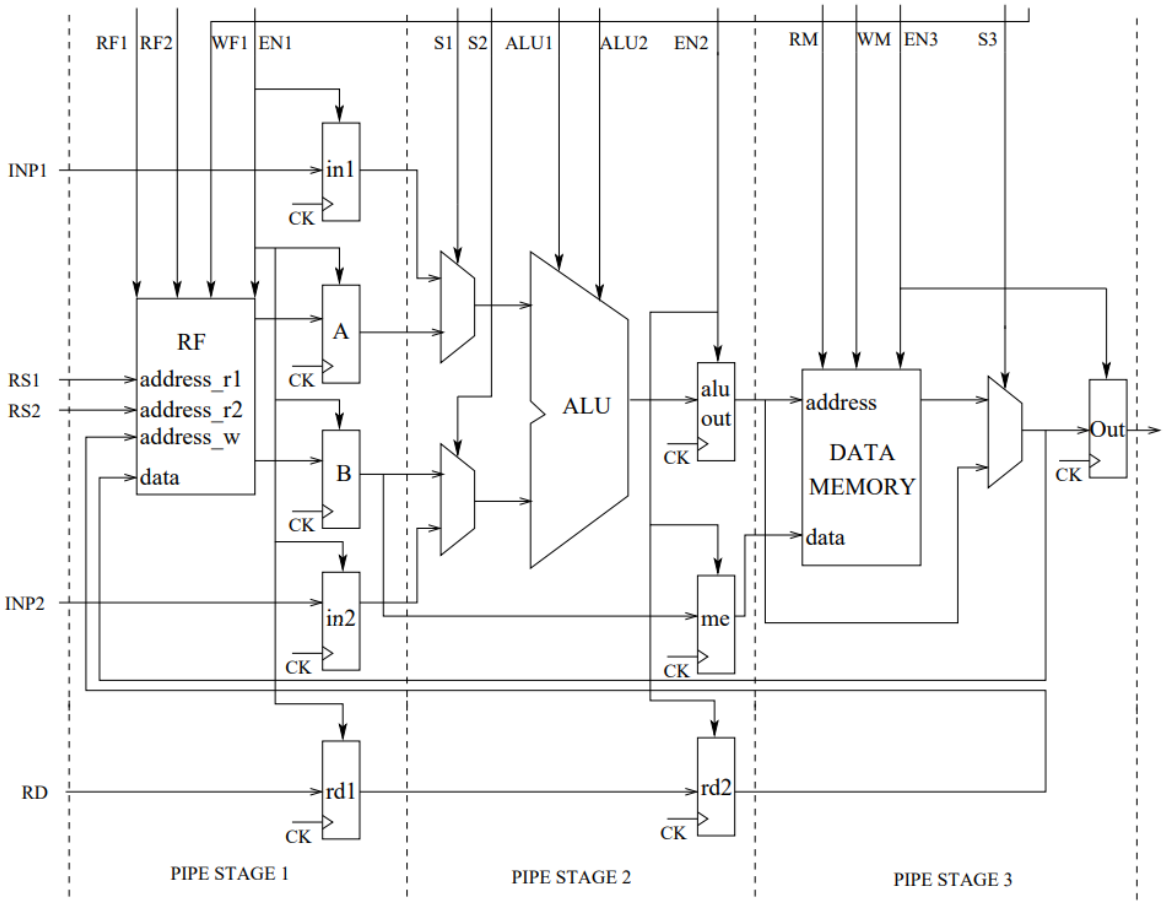

Figure 2.3: Detailed view of the Decode and Execution stages

2.4.1 ALU unit

The ALU (Arithmetic and Logic Unit) is responsible for performing arithmetic operations as well as logical operations, and it has two operands (32 bits each), a control signal of 4 bits, and an output of 32 bits. The general schematic of the ALU is shown in Figure 2.4. It can perform the following operations:

1. Addition

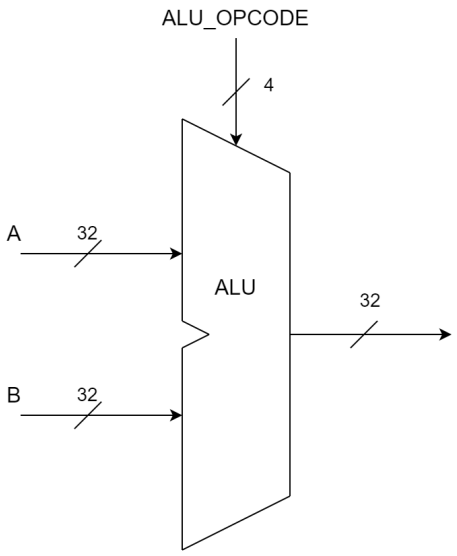

Figure 2.4: ALU general block diagram

2. Subtraction

3. Bitwise AND

4. Bitwise OR

5. Bitwise XOR

6. Comparison (>= , <= , !=)

7. Logical shift left

8. Logical shift right

See **table 3.1** for the encoding of each operation.
It should be noted that the addition and subtraction operations were not implemented by simple "+" and "-" operators in VHDL, rather it was chosen to implement and use a P4 adder (with small modifications to also perform subtraction) because it offers better performance(this is explained in detail later on.)
In order to be able to use the P4 adder, it was placed outside of the "basic" ALU where the other operations were implemented, and then both were wrapped inside a "complete" ALU, as seen in figure 2.5.

For addition/Subtraction operations, the basic ALU simply outputs the result of the P4 adder, no additional operations are performed (dotted line). The encoding of the add and sub operation are as follows "0001" and "0010" respectively, therefore the bit of index 1 is used to drive Cin of the P4

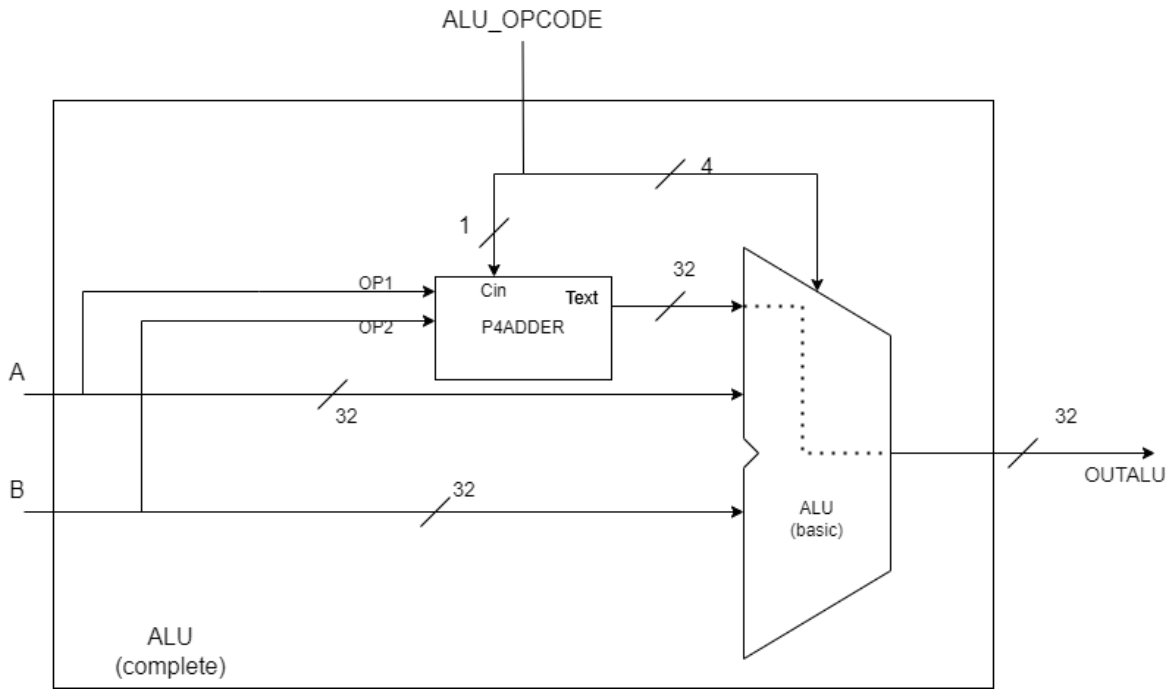

Figure 2.5: Detailed diagram of the ALU.

adder (Cin=0 =>Add, Cin=1 =>Sub). The remaining operations are implemented inside the "basic" ALU by using simple VHDL operators (AND, OR...).

Adder/Substractor The adder implemented in this DLX processor is based on the adder used in the Pentium4 adder. It is split into 2 major components:
1) Carry generator: Has both operands and carry in as inputs, and generates 8 carries (1 bit each).
2) Sum generator: Has both operands and the 8 bits generated by the carry generator as inputs, and generates the final result of the addition.
The general schematic is shown in Figure 2.6.

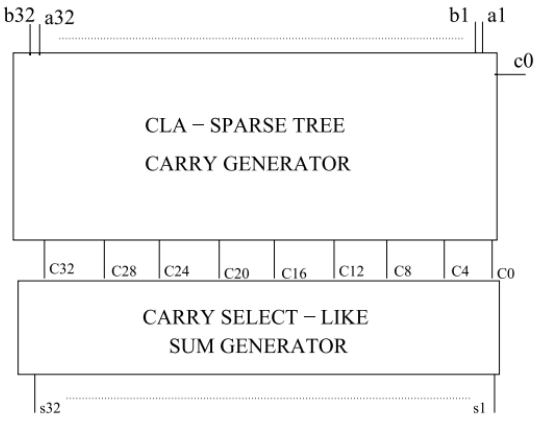

Figure 2.6: P4 adder general block diagram.

It is worth mentioning that the adder is also serving as a subtractor. This was made possible with

some modifications done to the carry select adder and the sparse tree carry generator (discussed in paragraph **Sparse tree carry generator**).

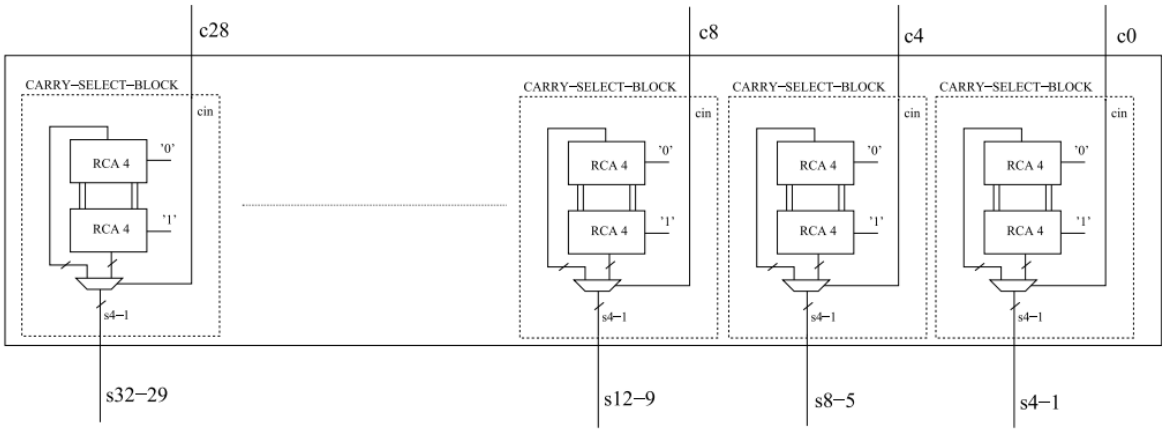

Figure 2.7: P4 Simplified Carry Select adder

Carry select adder The carry-select adder is made out of eight carry-select blocks, each having two ripple carry adders (RCA) and a 2x1 MUX (see figure 2.6). Each block computes the sum of 4 bits of the operands, with Cin = 0 and Cin = 1 (this is made possible due to having 2 RCAs in each block), then after their carry has been computed by the carry generator, the real sum is selected using the MUX.
To implement the subtractor, one operand needs to be XORED and operand B was chosen. This was implemented by adding a 32-bit output to the carry generator, which is simply XORING operand B with Cin of the whole adder, then using this XORED B signal as input for the carry select adder. When Cin = 0 perform sum, when Cin = 1 perform subtraction.

Sparse tree carry generator The sparse tree is composed of a PG network (first row), PG network is composed of PG blocks, and they work as follows:
g = a AND b
p = a XOR b
Therefore it has 2 inputs and 2 outputs.
The remaining rows are made out of G (Generate) and PG (Propagate and Generate) blocks, they are a bit different than the previous PG block mentioned as they are more general. Without going into much detail the general formulas for these blocks are:
For Generate block: (3 inputs and 1 output)
$G_{i:j} = G_{i:k} + P_{i:k} \cdot G_{k1:j}$

For Propagate and Generate block: (4 inputs and 2 outputs)
$G_{i:j} = G_{i:k} + P_{i:k} \cdot G_{k1:j}$
$P_{i:j} = P_{i:k} \cdot P_{k1:j}$
Placing these blocks as shown in Figure 2.7 allows to compute the carry for every 4 bits (thus radix-2), also explaining why the carry-select blocks are computing the sum for each 4 bits only.
To implement the subtractor, the same XORED B mentioned previously is used instead of the initial operand B. Another modification is needed which is the first block in the PG network with a general generate, having A, XORED B and Cin as inputs. This is shown in Figure 2.8.

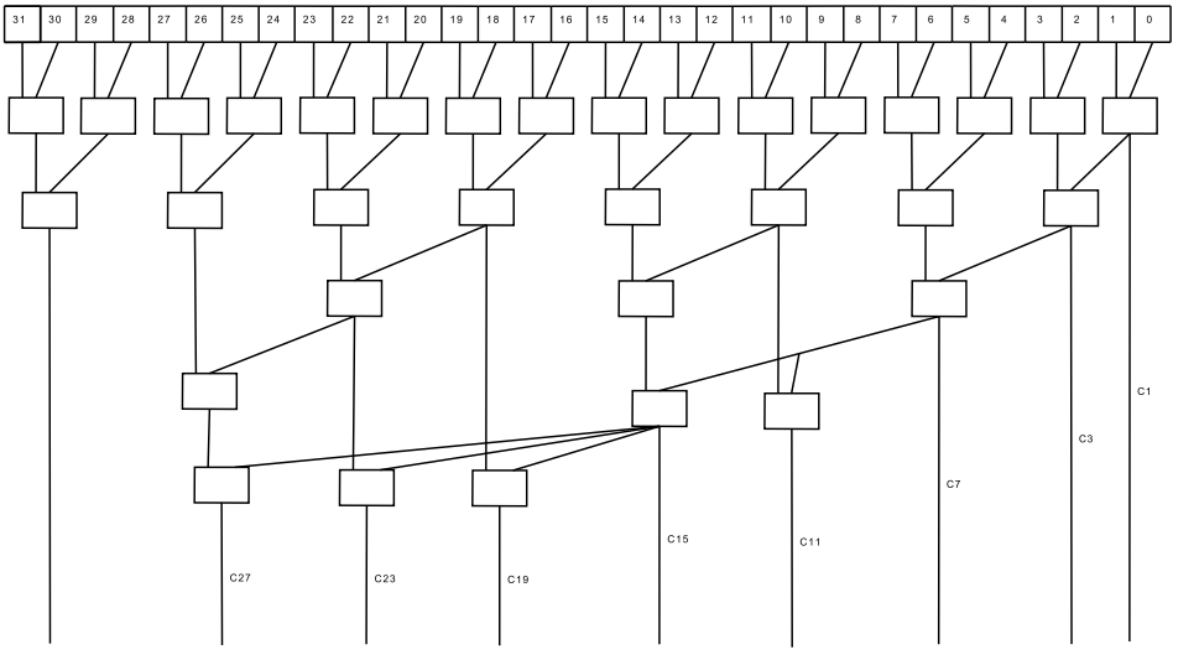

Figure 2.8: Pentium 4 sparse tree propagate network.

The P4 adder was chosen because it is based on the carry look-ahead adder, which allows for computation of the sum without waiting for the carry to propagate from the MSB to the following bits, which reduces delay and, therefore, has better performance.

2.5 Memory access stage

The memory access stage is composed of a 2x1 MUX, an OR gate, DRAM (external to the DLX) and a 32-bit register. The role of each component is explained as follows:

1. 2x1 MUX: The output of this MUX either writes NPC to PC (normal functioning) or writes the branch or jump address carried in a branch or jump instruction.

2. OR gate: The output of this gate controls the "SEL" signal of the MUX, jump or branch taken means "SEL" is equal to 1. The OR gate is 1 if the "COND" register is 1 or JUMP EN is one, which allows the branch or jump address to be written into PC.

3. DRAM: Although the DRAM is placed outside the data path, it was mentioned for ease of explanation. The memory is implemented with a data width of 32 bits and a data length of 16 (2**16 rows), 16 was chosen because using 32 bits of length is impossible to simulate and synthesis. It has a port dedicated to reading and writing, asynchronous read and synchronous write is implemented, a DATAIN port,and asynchronous reset.

4. 32-bit register LMD: Used to save the output of the DRAM, this value could be used in the next clock cycle to write in the RF.
   See Figure 2.1 for a better view of the memory access stage

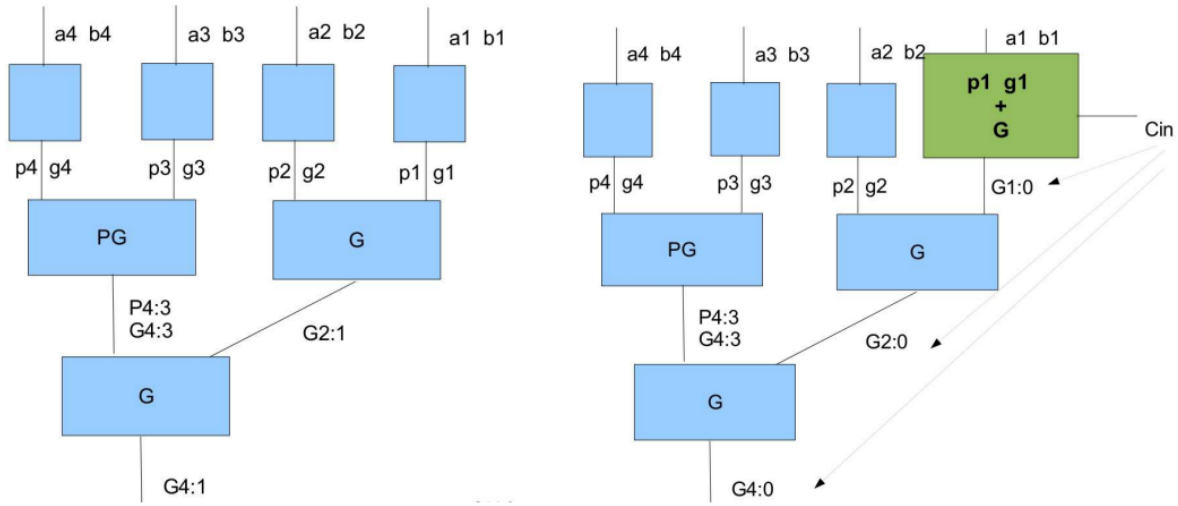

Figure 2.9: Detail for the first terms in the Pentium 4 sparse tree propagate network. On the right a modified version to allow the propagation of carry-in.

2.6 Write back stage

This stage is only composed of 2x1 MUX whose purpose is to select the output going to the *Register File*. It selects among the output coming from the register *NPC3*, the output coming from the register *LMD* or the output coming from the register *ALU output*.

CHAPTER 3

Control Unit

3.1 General description

The control unit is an essential step while designing a processor architecture because it controls the behavior of the Datapath.

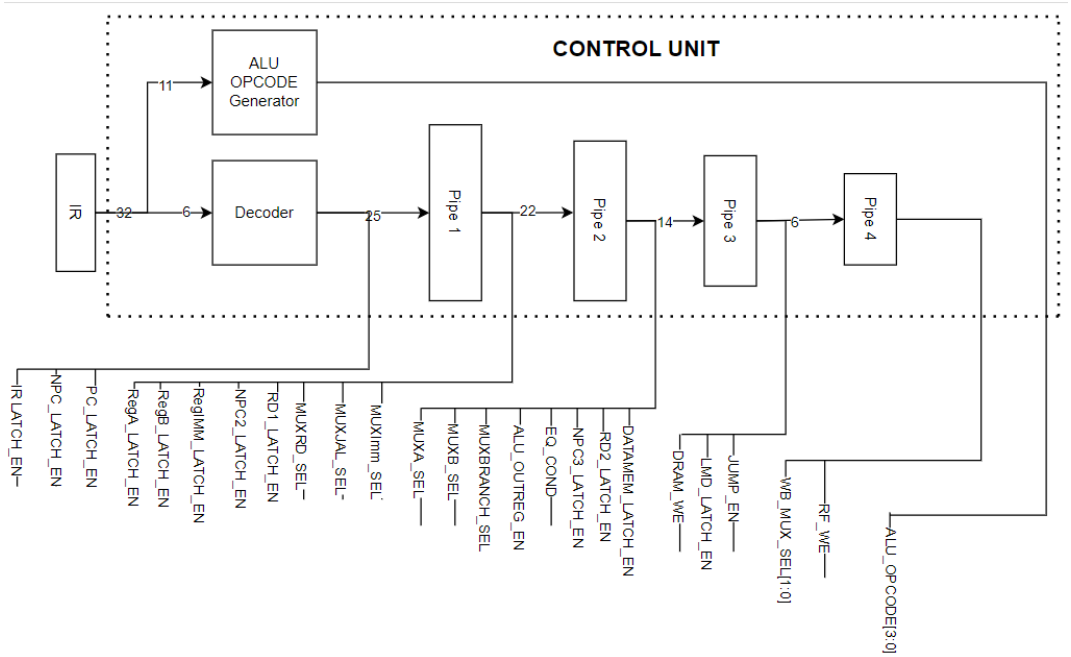

Figure 3.1: DLX control unit

In particular, Figure 2.1 shows the basic Datapath implementation for the DLX processor. Here, it is possible to see that the architecture is composed of 5 stages: *Instruction fetch*, *Instruction decode*, *Execution*, *Memory access*, *Write back*. Whereas, Figure 3.1 exposes the control unit structure Moreover, each component needs control signals that indicate which specific action should be done by them. The following list describes these signals:

1. Instruction fetch

   (a) IR_LATCH_EN: Enables the IR register

   (b) NPC_LATCH_EN: Enables the NPC register
   (c) PC_LATCH_EN: Enables the PC register

2. Instruction decode
   (a) RegA_LATCH_EN: Enables the A Register
   (b) RegB_LATCH_EN: Enables the B Register
   (c) NPC2_LATCH_EN: Enables the NPC2 Register
   (d) RD1_LATCH_EN: Enables the RD1 Register
   (e) MUXRD_SEL: Controls the mux selection between IR[20:16] for I-type instructions (0) and IR[15:11](1) for R-type instructions
   (f) MUXJAL_SEL: Controls the mux selection between the output of *MUX_RD*(0) and "11111"(1) for saving in *R31* for the *JAL* instruction
   (g) MUXImm_SEL: Controls the mux selection between sign-extension of IR[15:0] (0) for *beqz* and *bnez* and sign-extension of IR[26:0](1) for *Jump* and *JAL*

3. Execution
   (a) MUXA_SEL: Controls the mux selection between the NPC register output(0) and the A Register output(1)
   (b) MUXB_SEL: Controls the mux selection between the B Register output(0) and the Imm Register output(1)
   (c) MUXBRANCH_SEL: Controls the mux selection between the output of *Zero?*(0) for the *beqz* and the negated output of *Zero?*(1) for the *bnez*
   (d) ALU_OUTREG_EN: Enables the ALUOUT register
   (e) EQ_COND: Enables the COND register
   (f) NPC3_LATCH_EN: Enables the NPC3 Register
   (g) RD2_LATCH_EN: Enables the RD2 Register
   (h) DATAMEM_LATCH_EN: Enables the Data mem Register

4. Memory access
   (a) DRAM_WE: Enables writing in the Data Memory
   (b) LMD_LATCH_EN: Enables the LMD Register
   (c) JUMP_EN:Controls the mux selection between the NPC register output(0) and the the COND register(1)

5. Write back
   (a) WB_MUX_SEL: Controls the mux selection among the output of NPC3(00), the LMD register output(01) and the the ALUOUT register(10)
   (b) RF_WE: Enables writing in the Register File

Furthermore, the control unit has to generate the *ALU_OPCODE* signal that indicates which operation should be performed by the ALU depending on the instruction. Thus, the following table shows the encoding for the operations needed to execute the instructions of the instructions set:

It is important to mention that the control signals and the *ALU_OPCODE* generation are managed separately. For further information related to the control signals, please refer to B

Operation	Opcode
nop	0x0
add	0x1
sub	0x2
and	0x3
or	0x4
xor	0x5
sge	0x6
sle	0x7
sne	0x8
sll	0x9
srl	0xA

Table 3.1: *ALU_OPCODE* encoding

3.2 Implementation

The DLX processor implements 5 stages pipeline and this has to be considered when choosing the control unit implementation. As it is seen in Figure 3.1, the *General control unit* could be implemented with three different approaches: Hardwired, encoded FSM and microprogrammed. In particular, the technique selected was to have a Hardwired control unit in order to obtain good behavior while working with the pipelined stages. Figure 3.2 describes the structure of the control unit.

It is composed of 4 blocks. First, the *LUT* works as a memory in which different values are stored. Particularly, the *IR* provides the instruction *OPCODE*(6 MSB's) to a decoder whose purpose is to translate the *OPCODE* into an address. Then, the LUT receives this address and searches the row(word) corresponding to it. Finally, the LUT gives *CW* as output.

On the one hand, the decoder understands which instruction is being executed, using the *OPCODE* value, and generates the address to the *LUT*. On the other hand, the LUT contains *CW*(control word) which is a word whose values are bits that correspond to control signal values. Therefore, the input of the LUT is the *OPCODE* of the current instruction and its output is *CW* that contains the control signal values that will go to the Datapath to correctly execute the instruction.

In consequence, the designer should be careful while defining the values for the *LUT* because they will have an important impact on the processor's behavior. One thing to mention is that the *LUT* has a width equal to the number of signals required to properly control the Datapath.

A last point to mention is that the *ALU_OPCODE* generator is considered apart from the *General control unit*. Likewise, the output called *ALU_OPCODE* is considered separately as well(see Figure 3.1. Moreover, the *ALU_OPCODE* generator works as a combinational circuit that receives two inputs(*OPCODE* and *FUNC*), processes them, and sets an output(*ALU_OPCODE*) depending on the operation required from the current instruction to be performed by the *ALU*. Thus, Table 3.1 shows the encoding for each operation.

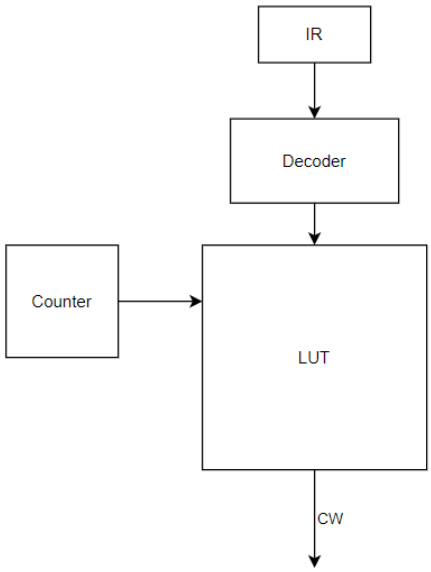

Figure 3.2: Hardwired control unit in detail

CHAPTER 4

Synthesis of DLX processor

4.1 Scripts

In order to synthesise the DLX processor 5 tcl scripts were written which are **SYN_script.tcl**, **compile0.tcl**, **compile1.tcl**, **compile2.tcl** and **compile3.tcl** and they work as follows:

1. SYN_script.tcl: This script analyses all VHDL files and then elaborates the DLX processor.
2. compile0.tcl: Compiles the design without any optimizations, a simple *"compile -exact_map"* command is used. It saves the power and timing reports in files named *"optimised0_power.rpt"*, *"optimised0_timing.rpt"* and *"optimised0_area.rpt"*. It also generates a .ddc and .v files of the synthesized design.
3. compile1.tcl: Compiles the design with clock gating and high effort for mapping, area and power, the command is *"compile -exact_map -map_effort high -area_effort high -power_effort high -gate_clock"*. It saved and generated the same files as in "compile0.tcl" but with different names (replacing 0 by 1).
4. compile2.tcl: Compiles the design with even more optimizations by using the command *"compile_ultra -gate_clock"*, this implements clock gating plus other optimizations offered by compile_ultra. The same reports and files are saved but with different names.
5. compile3.tcl: Compiles the design with the same optimizations as "compile2.tcl" but with constraints. The same reports and files are saved but with different names.

It should be noted that using the compile scripts one after the other does not work. In order to test them all, the script **SYN_script.tcl** should be run between them.

4.2 Synthesis results

Different optimization levels result in different power, timing and area reports. The detailed report for each optimisation level can be found in appendix C, for now the summaries will be enough.

1. Summary for level 0 optimization (compile0.tcl):
   1) Area = 14395.388249
   2) Power = 1.4320e+03 uW
   3) Timing = 0.43

2. Summary for level 1 optimization (compile0.tcl):
   1) Area = 12176.948276
   2) Power = 711.8751 uW
   3) Timing = 0.30

3. Summary for level 2 optimization (compile0.tcl):
   1) Area = 11260.844305
   2) Power = 619.5353 uW
   3) Timing = 0.41

4. Summary for level 3 optimization (compile0.tcl):
   1) Area = 11244.352305
   2) Power = 512.8225 uW
   3) Timing = 0.68

4.3 Conclusion

The following table summarizes all the results obtained in the previous section:

Optimization level	Area	Power	Timing
Level 0	14395.388249	1.4320e+03 uW	0.43
Level 1	12176.948276	711.8751 uW	0.30
Level 2	11260.844305	619.5353 uW	0.41
Level 3	11244.352305	512.8225 uW	0.68

More optimizations (such as clock gating) mean better results in terms of area, power and timing. This can be seen in the table as the area and power decrease when increasing the optimization level. However, in some cases, not all parameters can be improved because there is a trade-off between these parameters, for example, the timing increases going from level 1 to level 2 (worse); even so, the area and power decrease. Another example is the difference going from level 2 to level 3 (better area but worse timing).

CHAPTER 5

Floorplan

The Floorplan is a schematic representation of the placement of the major functional blocks of an integrated circuit. The process of designing the Floorplan consists of multiple steps as it is shown below:

1. Configuring
2. Structuring of the Floorplan
3. Power rings insertion
4. Stripes insertion
5. Standard Cell Power routing
6. Placement
7. Exploring placed design
8. Placing I/O Pins
9. Post-Clock-Synthesis optimization
10. Place filler
11. Routing
12. Post routing optimization
13. Analysis of parasitics delay
14. Verification

Figures 5.1, 5.2, 5.3 and 5.4 show the Floorplan obtained after the previous steps were done.

Finally, if one is interested in looking at the reports of the Floorplan, he can refer to B. Thus, one can notice that this report shows the same area as shown in 5 for the last version of the synthesis described.

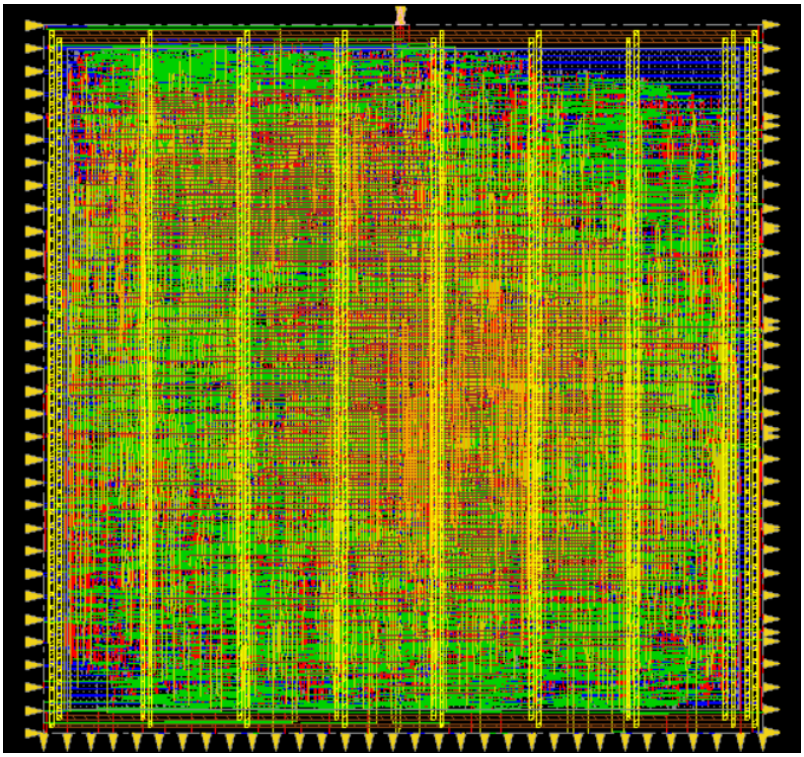

Figure 5.1: Floorplan: General view of the Chip

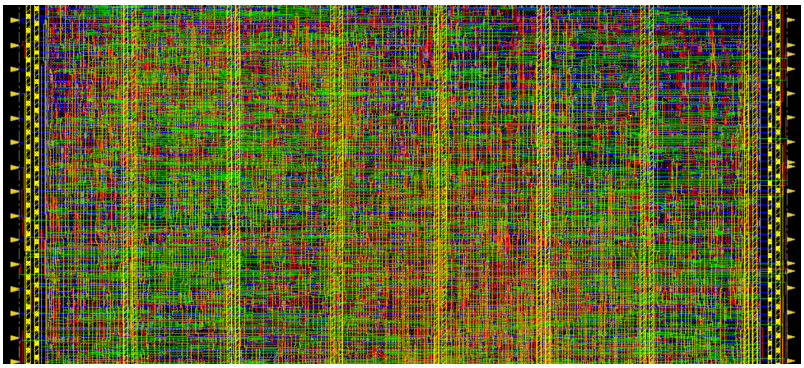

Figure 5.2: Floorplan: DLX Processor interconnections

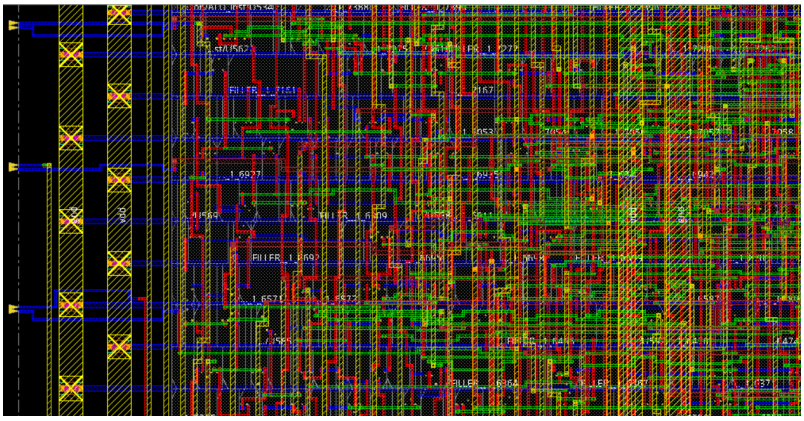

Figure 5.3: Floorplan: DLX Processor interconnections in detail

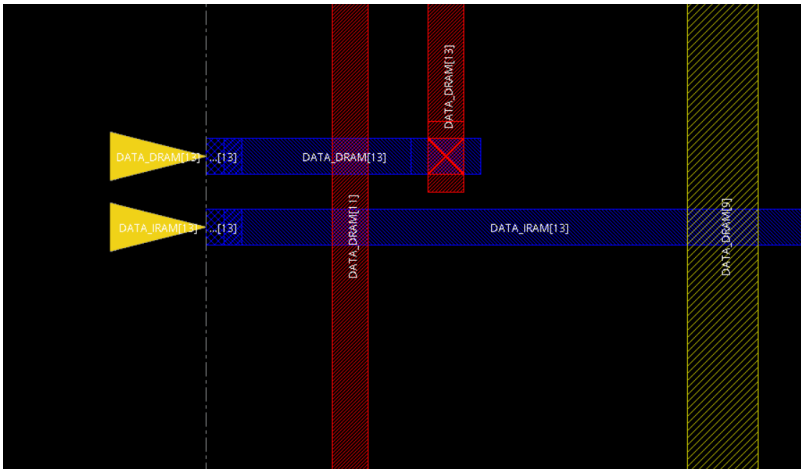

Figure 5.4: Floorplan: Example of pin connections

APPENDIX A

DLX instruction set

Instruction	Description	OPCODE	FUNC	Type	Explanation
add	add	0x00	0x20	R-type	R[RD] = R[RS1] + R[RS2]
addi	add immediate	0x08	–	I-type	R[RD] = R[RS1] + Imm
and	and	0x00	0x24	R-type	R[RD] = R[RS1] AND(bitwise) R[RS2]
andi	and immediate	0x0C	–	I-type	R[RD] = R[RS1] AND(bitwise) Imm
beqz	branch if equal to zero	0x04	–	I-type	PC = PC + (R[RS1] == 0 ? extend(Imm) : 0)
bnez	branch if not equal to zero	0x05	–	I-type	PC = PC + (R[RS1] != 0 ? extend(Imm) : 0)
j	jump	0x02	–	J-type	PC = PC + extend(Imm)
jal	jump and link	0x03	–	J-type	PC = PC + extend(Imm); R[R31] = PC + 4
lw	load word	0x23	–	I-type	R[RD] = MEM[RS1 + extend(immediate)]
nop	no operation	0x15	–	I-type	–
or	or	0x00	0x25	R-type	R[RD] = R[RS1] OR(bitwise) R[RS2]
ori	or immediate	0x0D	–	I-type	R[RD] = R[RS1] OR(bitwise) Imm
sge	set if greater than or equal	0x00	0x2D	R-type	R[RD] = (R[RS1] $\geqq R[RS2]$?1 : 0)
sgei	set if greater than or equal	0x1D	–	I-type	R[RD] = (R[RS1] $\geqq extend(Imm)$?1 : 0)
sle	set if less than or equal	0x00	0x2C	R-type	R[RD] = (R[RS1] $\leqq R[RS2]$?1 : 0)
slei	set if less than or equal immediate	0x1C	–	I-type	R[RD] = (R[RS1] $\leqq extend(Imm)$?1 : 0)
sll	shift left logical	0x00	0x04	R-type	R[RD] = R[RS1] $\ll (R[RS2]\%8)$
slli	shift left logical immediate	0x14	–	I-type	R[RD] = R[RS1] $\ll (Imm\%8)$
sne	set if not equal	0x00	0x29	R-type	R[Rd] = (R[RS1] != R[RS2] ? 1 : 0)
snei	set if not equal immediate	0x19	–	I-type	R[Rd] = (R[RS1] != extend(Imm) ? 1 : 0)
srl	shift right logical	0x00	0x06	R-type	R[RD] = R[RS1] $\gg (R[RS2]\%8)$
srli	shif right logical immediate	0x16	–	I-type	R[RD] = R[RS1] $\gg (Imm\%8)$
sub	sub	0x00	0x22	R-type	R[RD] = R[RS1] - R[RS2]
subi	sub immediate	0x0A	–	I-type	R[RD] = R[RS1] - Imm
sw	store word	0x2B	–	I-type	MEM[R[RS1] + extend(Imm)] = R[RD]
xor	exclusive xor	0x00	0x26	R-type	R[RD] = R[RS1] XOR(bitwise) R[RS2]
xori	exclusive xor immediate	0x0E	–	I-type	R[RD] = R[RS1] XOR(bitwise) Imm

Table A.1: DLX Instruction set

APPENDIX B

Area, Power, and Timing Reports

B.1 Synthesis reports

1. **Level 0 optimisation (compile0.tcl):**

   (a) **Area report:**

```
****************************************
Report : area
Design : dlx_processor
Version: S-2021.06-SP4
Date   : Thu Oct 19 11:17:09 2023
****************************************

Number of ports:                         4827
Number of nets:                         11259
Number of cells:                         6853
Number of combinational cells:           5226
Number of sequential cells:              1367
Number of macros/black boxes:               0
Number of buf/inv:                        935
Number of references:                       2

Combinational area:               7154.868016
Buf/Inv area:                      667.659988
Noncombinational area:            7240.520234
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                 14395.388249
Total area:                 undefined
```

   (b) **Power report:**

```
****************************************
```

```
Report : power
        -analysis_effort low
Design : dlx_processor
Version: S-2021.06-SP4
Date   : Thu Oct 19 11:17:09 2023
****************************************



Operating Conditions: typical   Library: NangateOpenCellLibrary
Wire Load Model Mode: top

Design        Wire Load Model            Library
------------------------------------------------
dlx_processor          5K_hvratio_1_4    NangateOpenCellLibrary


Global Operating Voltage = 1.1
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000ff
    Time Units = 1ns
    Dynamic Power Units = 1uW    (derived from V,C,T units)
    Leakage Power Units = 1nW


  Cell Internal Power  = 680.4598 uW   (57%)
  Net Switching Power  = 510.5707 uW   (43%)
                         ---------
Total Dynamic Power    =   1.1910 mW  (100%)

Cell Leakage Power     = 240.9919 uW


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  A
-----------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000            0.0000            0.0000  (   0.00%)
memory             0.0000           0.0000            0.0000            0.0000  (   0.00%)
black_box          0.0000           0.0000            0.0000            0.0000  (   0.00%)
clock_network      0.0000           0.0000            0.0000            0.0000  (   0.00%)
register           0.0000           0.0000            0.0000            0.0000  (   0.00%)
sequential       509.1975           2.2977        1.0819e+05          619.6898  (  43.27%)
combinational    171.2599         508.2732        1.3280e+05          812.3287  (  56.73%)
-----------------------------------------------------------------------------------------------
Total            680.4573 uW       510.5708 uW     2.4099e+05 nW     1.4320e+03 uW
```

(c) **Timing report:**

```
****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : dlx_processor
Version: S-2021.06-SP4
Date   : Thu Oct 19 11:17:09 2023
****************************************

Operating Conditions: typical   Library: NangateOpenCellLibrary
Wire Load Model Mode: top

  Startpoint: DATA_IRAM[28]
              (input port)
  Endpoint: RW_DRAM (output port)
  Path Group: (none)
  Path Type: max

  Des/Clust/Port     Wire Load Model       Library
  ------------------------------------------------
  dlx_processor      5K_hvratio_1_4        NangateOpenCellLibrary

  Point                                                   Incr       Path
  --------------------------------------------------------------------------
  input external delay                                    0.00       0.00 f
  DATA_IRAM[28] (in)                                      0.00       0.00 f
  control_unit/IR_IN[28]
                                                          0.00       0.00 f
  control_unit/U105/ZN (INV_X1)                           0.05       0.05 r
  control_unit/U103/ZN (AND3_X1)                          0.07       0.11 r
  control_unit/U102/ZN (NAND3_X1)                         0.05       0.17 f
  control_unit/U100/ZN (NOR2_X1)                          0.06       0.22 r
  control_unit/U64/ZN (NAND2_X1)                          0.05       0.27 f
  control_unit/U14/ZN (INV_X2)                            0.15       0.42 r
  control_unit/DRAM_WE
                                                          0.00       0.42 r
  RW_DRAM (out)                                           0.01       0.43 r
  data arrival time                                                  0.43
  --------------------------------------------------------------------------
  (Path is unconstrained)
```

2. **Level 1 optimisation (compile1.tcl):**

   (a) **Area report:**

```
****************************************
Report : area
```

```
Design : dlx_processor
Version: S-2021.06-SP4
Date   : Thu Oct 19 11:20:33 2023
****************************************


Number of ports:                         5007
Number of nets:                         10095
Number of cells:                         5554
Number of combinational cells:           3791
Number of sequential cells:              1444
Number of macros/black boxes:               0
Number of buf/inv:                        820
Number of references:                       2

Combinational area:               4586.638037
Buf/Inv area:                      567.909992
Noncombinational area:            7590.310240
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                 12176.948276
Total area:                 undefined
```

(b) **Power report:**

```
****************************************
Report : power
        -analysis_effort low
Design : dlx_processor
Version: S-2021.06-SP4
Date   : Thu Oct 19 11:20:32 2023
****************************************



Operating Conditions: typical   Library: NangateOpenCellLibrary
Wire Load Model Mode: top

Design        Wire Load Model            Library
------------------------------------------------
dlx_processor          5K_hvratio_1_4    NangateOpenCellLibrary


Global Operating Voltage = 1.1
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000ff
    Time Units = 1ns
    Dynamic Power Units = 1uW    (derived from V,C,T units)
```

```
    Leakage Power Units = 1nW


  Cell Internal Power  = 182.4049 uW   (36%)
  Net Switching Power  = 324.4007 uW   (64%)
                         ---------
Total Dynamic Power    = 506.8056 uW  (100%)

Cell Leakage Power     = 205.0694 uW


                 Internal         Switching           Leakage             Total
Power Group      Power            Power               Power               Power   (   %    )  A
-----------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000              0.0000              0.0000  (   0.00%)
memory             0.0000           0.0000              0.0000              0.0000  (   0.00%)
black_box          0.0000           0.0000              0.0000              0.0000  (   0.00%)
clock_network      0.0000           0.0000              0.0000              0.0000  (   0.00%)
register           0.0000           0.0000              0.0000              0.0000  (   0.00%)
sequential        87.0620          10.7373          1.1089e+05            208.6848  (  29.31%)
combinational     95.3429         313.6640          9.4184e+04            503.1903  (  70.69%)
-----------------------------------------------------------------------------------------------
Total            182.4048 uW       324.4013 uW     2.0507e+05 nW         711.8751 uW
```

(c) **Timing report:**

```
****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : dlx_processor
Version: S-2021.06-SP4
Date   : Thu Oct 19 11:20:33 2023
****************************************

Operating Conditions: typical   Library: NangateOpenCellLibrary
Wire Load Model Mode: top

  Startpoint: DATA_IRAM[28]
              (input port)
  Endpoint: RW_DRAM (output port)
  Path Group: (none)
  Path Type: max

  Des/Clust/Port     Wire Load Model       Library
  ------------------------------------------------
  dlx_processor      5K_hvratio_1_4        NangateOpenCellLibrary

  Point                                                   Incr       Path
```

```
--------------------------------------------------------------------------
input external delay                                    0.00       0.00 f
DATA_IRAM[28] (in)                                      0.00       0.00 f
control_unit/IR_IN[28]
                                                        0.00       0.00 f
control_unit/U104/ZN (INV_X1)                           0.05       0.05 r
control_unit/U102/ZN (AND3_X1)                          0.07       0.11 r
control_unit/U101/ZN (NAND3_X1)                         0.05       0.17 f
control_unit/U99/ZN (NOR2_X1)                           0.06       0.22 r
control_unit/U62/ZN (NAND2_X1)                          0.04       0.26 f
control_unit/U61/ZN (INV_X1)                            0.03       0.30 r
control_unit/DRAM_WE
                                                        0.00       0.30 r
RW_DRAM (out)                                           0.00       0.30 r
data arrival time                                                  0.30
--------------------------------------------------------------------------
(Path is unconstrained)
```

3. **Level 2 optimisation (compile2.tcl):**

   (a) **Area report:**

```
****************************************
Report : area
Design : dlx_processor
Version: S-2021.06-SP4
Date   : Thu Oct 19 11:28:14 2023
****************************************

Number of ports:                          526
Number of nets:                          5096
Number of cells:                         4621
Number of combinational cells:           3138
Number of sequential cells:              1436
Number of macros/black boxes:               0
Number of buf/inv:                        420
Number of references:                      41

Combinational area:               3710.434067
Buf/Inv area:                      265.999998
Noncombinational area:            7550.410239
Macro/Black Box area:               0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                 11260.844305
Total area:                 undefined
```

   (b) **Power report:**

```
****************************************
Report : power
        -analysis_effort low
Design : dlx_processor
Version: S-2021.06-SP4
Date   : Thu Oct 19 11:28:13 2023
****************************************

Operating Conditions: typical   Library: NangateOpenCellLibrary
Wire Load Model Mode: top

Design        Wire Load Model            Library
------------------------------------------------
dlx_processor          5K_hvratio_1_4    NangateOpenCellLibrary


Global Operating Voltage = 1.1
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000ff
    Time Units = 1ns
    Dynamic Power Units = 1uW    (derived from V,C,T units)
    Leakage Power Units = 1nW


  Cell Internal Power  = 130.0274 uW   (30%)
  Net Switching Power  = 302.8427 uW   (70%)
                         ---------
Total Dynamic Power    = 432.8701 uW  (100%)

Cell Leakage Power     = 186.6660 uW

Information: report_power power group summary does not include estimated clock tree power. (PW

                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  A
-----------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000            0.0000             0.0000  (   0.00%)
memory             0.0000           0.0000            0.0000             0.0000  (   0.00%)
black_box          0.0000           0.0000            0.0000             0.0000  (   0.00%)
clock_network      0.0000           0.0000            0.0000             0.0000  (   0.00%)
register           0.0000           0.0000            0.0000             0.0000  (   0.00%)
sequential        85.3860          10.1768        1.1025e+05           205.8091  (  33.22%)
combinational     44.6404         292.6656        7.6420e+04           413.7262  (  66.78%)
-----------------------------------------------------------------------------------------------
Total            130.0264 uW       302.8424 uW     1.8667e+05 nW       619.5353 uW
```

(c) **Timing report:**

```
****************************************
```

```
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : dlx_processor
Version: S-2021.06-SP4
Date   : Thu Oct 19 11:28:14 2023
****************************************

Operating Conditions: typical   Library: NangateOpenCellLibrary
Wire Load Model Mode: top

  Startpoint: DATA_IRAM[30]
              (input port)
  Endpoint: RW_DRAM (output port)
  Path Group: (none)
  Path Type: max

  Des/Clust/Port     Wire Load Model       Library
  ------------------------------------------------
  dlx_processor      5K_hvratio_1_4        NangateOpenCellLibrary

  Point                                    Incr       Path
  -----------------------------------------------------------
  input external delay                     0.00       0.00 f
  DATA_IRAM[30] (in)                       0.00       0.00 f
  U381/ZN (NOR2_X1)                        0.07       0.07 r
  U396/ZN (INV_X1)                         0.03       0.10 f
  U401/ZN (NOR3_X1)                        0.10       0.20 r
  U402/ZN (NAND2_X1)                       0.05       0.25 f
  U403/ZN (NOR2_X1)                        0.06       0.31 r
  U404/ZN (INV_X1)                         0.04       0.35 f
  U405/ZN (NOR2_X1)                        0.06       0.40 r
  RW_DRAM (out)                            0.00       0.41 r
  data arrival time                                   0.41
  -----------------------------------------------------------
  (Path is unconstrained)
```

4. **Level 3 optimisation (compile3.tcl):**

   (a) **Area report:**

```
 ****************************************
Report : area
Design : dlx_processor
Version: S-2021.06-SP4
Date   : Thu Oct 19 21:55:48 2023
****************************************
```

```
Number of ports:                          526
Number of nets:                          5071
Number of cells:                         4596
Number of combinational cells:           3113
Number of sequential cells:              1436
Number of macros/black boxes:               0
Number of buf/inv:                        395
Number of references:                      40

Combinational area:               3693.942067
Buf/Inv area:                      250.571998
Noncombinational area:            7550.410239
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                 11244.352305
Total area:                 undefined
```

(b) **Power report:**

```
****************************************
Report : power
        -analysis_effort low
Design : dlx_processor
Version: S-2021.06-SP4
Date   : Thu Oct 19 21:55:48 2023
****************************************

Operating Conditions: typical   Library: NangateOpenCellLibrary
Wire Load Model Mode: top

Design        Wire Load Model            Library
------------------------------------------------
dlx_processor          5K_hvratio_1_4    NangateOpenCellLibrary


Global Operating Voltage = 1.1
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000ff
    Time Units = 1ns
    Dynamic Power Units = 1uW    (derived from V,C,T units)
    Leakage Power Units = 1nW


  Cell Internal Power  = 209.2814 uW   (64%)
  Net Switching Power  = 117.2279 uW   (36%)
                         ---------
Total Dynamic Power    = 326.5093 uW  (100%)
```

```
Cell Leakage Power     = 186.3135 uW


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  A
-----------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000            0.0000           0.0000  (   0.00%)
memory             0.0000           0.0000            0.0000           0.0000  (   0.00%)
black_box          0.0000           0.0000            0.0000           0.0000  (   0.00%)
clock_network     54.0452          33.8287         2.6319e+03          90.5057  (  17.65%)
register         135.3273           2.5465         1.0766e+05         245.5309  (  47.88%)
sequential         0.9463           0.5278           470.8958           1.9450  (   0.38%)
combinational     18.9629          80.3251         7.5553e+04         174.8410  (  34.09%)
-----------------------------------------------------------------------------------------------
Total            209.2816 uW      117.2281 uW     1.8631e+05 nW      512.8225 uW
```

(c) **Timing report:**

```
****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : dlx_processor
Version: S-2021.06-SP4
Date   : Thu Oct 19 21:55:48 2023
****************************************

  Startpoint: dlx_datapath/execution/COND/Q_reg
              (rising edge-triggered flip-flop clocked by CLK_DLX)
  Endpoint: dlx_datapath/fetch/PC/Q_reg[0]
            (rising edge-triggered flip-flop clocked by CLKNEG_DLX)
  Path Group: CLKNEG_DLX
  Path Type: max

  Des/Clust/Port     Wire Load Model       Library
  ------------------------------------------------
  dlx_processor      5K_hvratio_1_4        NangateOpenCellLibrary

  Point                                                    Incr  Path
  ---------------------------------------------------------------------------
  clock CLK_DLX (rise edge)                                0.00  0.00
  clock network delay (ideal)                              0.00  0.00
  dlx_datapath/execution/COND/Q_reg/CK (DFFR_X1)           0.00  0.00 r
  dlx_datapath/execution/COND/Q_reg/Q (DFFR_X1)            0.08  0.08 f
  U605/ZN (NOR2_X1)                                        0.16  0.24 r
  U606/Z (CLKBUF_X1)                                       0.21  0.46 r
  U607/ZN (INV_X1)                                         0.13  0.59 f
  U608/ZN (OAI22_X1)                                       0.08  0.67 r
```

```
dlx_datapath/fetch/PC/Q_reg[0]/D (DFFS_X1)         0.01  0.68 r
data arrival time                                        0.68
```

B.2 Floorplan reports

General summary report:

```
==============================
Floorplan/Placement Information
==============================
Total area of Standard cells: 18758.054 um^2
Total area of Standard cells(Subtracting Physical Cells): 11252.332 um^2
Total area of Macros: 0.000 um^2
Total area of Blockages: 0.000 um^2
Total area of Pad cells: 0.000 um^2
Total area of Core: 18758.054 um^2
Total area of Chip: 21647.133 um^2
Effective Utilization: 1.0000e+00
Number of Cell Rows: 97
% Pure Gate Density #1 (Subtracting BLOCKAGES): 100.000%
% Pure Gate Density #2 (Subtracting BLOCKAGES and Physical Cells): 59.987%
% Pure Gate Density #3 (Subtracting MACROS): 100.000%
% Pure Gate Density #4 (Subtracting MACROS and Physical Cells): 59.987%
% Pure Gate Density #5 (Subtracting MACROS and BLOCKAGES): 100.000%
% Pure Gate Density #6 (Subtracting MACROS and BLOCKAGES for insts are not placed): 59.987%
% Core Density (Counting Std Cells and MACROs): 100.000%
% Core Density #2(Subtracting Physical Cells): 59.987%
% Chip Density (Counting Std Cells and MACROs and IOs): 86.654%
% Chip Density #2(Subtracting Physical Cells): 51.981%
# Macros within 5 sites of IO pad: No
Macro halo defined?: No

==============================
Wire Length Distribution
==============================
Total metal1 wire length: 4219.0800 um
Total metal2 wire length: 33191.3650 um
Total metal3 wire length: 39917.0200 um
Total metal4 wire length: 20476.1200 um
Total metal5 wire length: 10442.8800 um
Total metal6 wire length: 2710.6800 um
Total metal7 wire length: 0.0000 um
Total metal8 wire length: 0.0000 um
Total metal9 wire length: 0.0000 um
Total metal10 wire length: 0.0000 um
Total wire length: 110957.1450 um
Average wire length/net: 23.5378 um
Area of Power Net Distribution:
```

```
------------------------------
Area of Power Net Distribution
------------------------------
Layer Name  Area of Power Net  Routable Area  Percentage
metal1  1186.6918  18758.0540  6.3263%
metal2  0.0000  18758.0540  0.0000%
metal3  0.0000  18758.0540  0.0000%
metal4  0.0000  18758.0540  0.0000%
metal5  0.0000  18758.0540  0.0000%
metal6  0.0000  18758.0540  0.0000%
metal7  0.0000  18758.0540  0.0000%
metal8  0.0000  18758.0540  0.0000%
metal9  227.9360  18758.0540  1.2151%
metal10  1008.2880  18758.0540  5.3752%  For more information click here
```

APPENDIX C

Other considerations

C.1 Project organization

During a project, it is fundamental to have good file organization to work efficiently. Hence, the files are organized depending on their level of hierarchy. With that purpose, the names of the files follow the pattern *orderingPrefixLevel0.subLevel.subLevel-nameOfComponent.vhd* to have a good perspective of the main components and their respective sub-components.

Figure C.1: Project organization: Main folder

Figure C.1 displays the two main folders of this project. On the one hand, *DLX_processorSIMULATION* consists of all the folders and files required to simulate the processor and analyze its behavior. On the other hand, *DLX_processorSYNTHESIS* contains all the files and folders to synthesize the processor. Moreover, it includes Area, Power and Timing reports. Finally, the *DLX_processorFLOORPLAN* has all the files related to the floorplan designing, including Timing, Area and Power reports, as well.

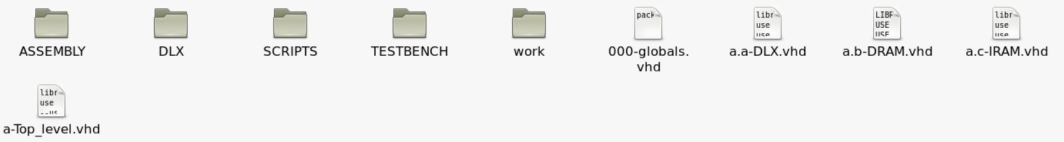

Figure C.2: Project organization: Inside *DLX_processorSIMULATION*

Figure C.2 shows the main files and folders that are part of the project for simulation. The *DLX* folder includes all the hardware components required to build the DLX processor and it has two main subfolders: *CU* and *DATAPATH*. Likewise, the *ASSEMBLY* folder includes the files that have assembly language code to test the architecture. In addition, one can find the testbenches to test the system behavior inside the *TESTBENCH* folder. Finally, *SCRIPTS* contains the scripts that can be used to speed up the process of simulating the architecture. Thus, if one is interested in using the script *a-Toplevel_script.txt*, he should run the command *"do SCRIPTS/a-Toplevel_script.txt"* in the QuestaSim command window.

C.2 Assembly and Simulation

In order to compile and/or simulate an assembly file change directory location to *ASSEMBLY*, these are the files that should be observed:

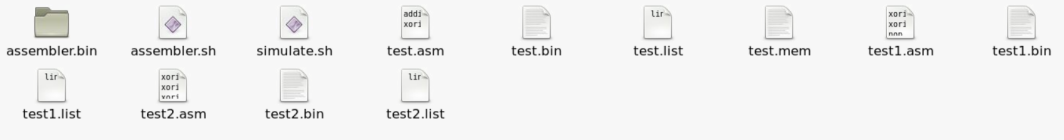

Figure C.3: *ASSEMBLY* files

Before simulating, make sure that the assembly file (.asm) is located in *ASSEMBLY* folder. From here on there are 2 ways to simulate the assembly file, either by using the *simulate.sh* (recommended) or *assembler.sh* script:

1. Using *simulate.sh*:
   (a) In *ASSEMBLY* folder run ". ./simulate.sh [name_of_file].asm"
2. Using *assembler.sh*:
   (a) In *ASSEMBLY* folder run ". ./assembler.sh [name_of_file].asm"
   (b) Go back to *DLX_processorSIMULATION]* and run vsim
   (c) In command line of vsim run the command "do SCRIPTS/a-Toplevel_script.txt"

It should be noted that hazards are not managed, so data dependency is an issue here. Therefore, in the case of data dependency, **nop** instruction had to be introduced between the instruction writing to the register and the instruction reading from it, specifically 2 instructions are needed.
Furthermore, jump and branch instructions are managed by freezing the pipeline, by simply adding, by hand, 2 nop instructions after each jump or branch. This could also be managed by the compiler instead of adding the "dummy" instructions by hand.