Politecnico di Torino

III Facoltà di Ingegneria

Laboratory 3
Design of a RISC-V-lite processor

Master degree in Electronic Engineering

Authors: Group 21

Dilillo Nicola S284963
Moncalvo Stefano S290315
Carrano Lorenzo S281565

GitHub Repository

Contents

CHAPTER 1

Introduction

The goal of this laboratory is to design and synthesize a RISC-V-lite processor. The ISA of this processor includes:

- R-Type instructions;
- I-Type instructions;
- S-Type instructions;
- SB-Type instructions;
- UJ-Type instructions;
- U-Type instructions.

The format of the instructions is summarized in the following image.

Name (Field Size)	Field						Comments
	7 bits	5 bits	5 bits	3 bits	5 bits	7 bits	
R-type	funct7	rs2	rs1	funct3	rd	opcode	Arithmetic instruction format
I-type	immediate[11:0]		rs1	funct3	rd	opcode	Loads & immediate arithmetic
S-type	immed[11:5]	rs2	rs1	funct3	immed[4:0]	opcode	Stores
SB-type	immed[12,10:5]	rs2	rs1	funct3	immed[4:1,11]	opcode	Conditional branch format
UJ-type	immediate[20,10:1,11,19:12]				rd	opcode	Unconditional jump format
U-type	immediate[31:12]				rd	opcode	Upper immediate format

Figure 1.1: **Instruction Format**

In particular, the processor is required to support a subset of the RV32I ISA, which corresponds to:

- **Arithmetic:**
  - **add:** R-Type instruction performing the addition between the source registers.
  - **addi:** I-Type instruction performing the addition between a source register and an immediate value.
  - **auipc:** U-Type instruction that adds the content of the program counter with an immediate value.
  - **lui:** U-Type instruction that stores the immediate value in the destination register.

- **Branches:**
  - **beq:** SB-Type instruction used to update the value of the program counter if the content of the two source registers is equal.
- **Loads:**
  - **lw:** I-Type instruction used to read a word from memory, computing the address using the immediate value. The word is stored in the destination register.
- **Shifts:**
  - **srai:** I-Type instruction performing the arithmetic right shift of the content of the source register, by a number of positions corresponding to the immediate value.
- **Logical:**
  - **andi:** I-Type instruction performing the logical AND between the source register and the immediate value.
  - **xor:** R-Type instruction performing the logical XOR between the two source registers.
- **Compare:**
  - **slt:** R-Type instruction that compares the content of the two source registers.
- **Jump and link:**
  - **jal:** J-Type instruction performs a jump to the instruction indicated by the sum between the PC and the immediate value. The value of the PC before the jump, increased by 4, is stored in the destination register.
- **Stores:**
  - **sw:** S-Type instruction that stores the value of the source register in the memory, whose address is computed using the immediate value.

The processor uses a 32-bit parallelism and is composed of 5 pipeline stages, instruction fetch, instruction decode, execute, access memory and write back. The stages will be described in the following chapters, as well as the techniques utilized to avoid hazards and in general guarantee the correct operation of the processor. Finally, the memories are not included in the design, instead they are implemented in the testbench for simulation purposes.

CHAPTER 2

Datapath

2.1 Fetch Stage

2.2 Decode Stage

2.3 Execute Stage

2.3.1 ALU

2.3.2 Forwarding Unit

2.4 Memory Stage

2.5 Write Back Stage

CHAPTER 3

Control Unit

3.1 Introduction

CHAPTER 4

Hazards

CHAPTER 5

Testbench

CHAPTER 6

Synthesis and Place & Route

6.1 Shynthesis

The two different designs have been synthesized using Synopsys Design Compiler, in order to obtain the maximum working frequency and an area estimation of the circuits. The following table summarizes the obtained results.

	RISCV	**Modified RISCV**
Min. Period	3.55 ns	3.6 ns
Max. Frequency	281.7 MHz	277.8 MHz
Area	14006.23 μm^2	14802.1 μm^2

Table 6.1: **Design Compiler Reports**

As expected, the modified design with the added component occupies more area. Moreover, since the added hardware is placed along the critical path, the maximum frequency is also slightly lower; however this is compensated by the reduced time needed for the computation of the instruction.

The report of the *elaborate* command lists the inferred memory devices inserted in the netlist. Every element was checked to avoid the insertion of latch instead of flip-flops. Finally, both synthesized netlists were simulated with Modelsim to confirm they still behaved correctly.

6.2 Place & Route

After the synthesis of the circuits, place & route was performed using Cadence Innovus. This requires to perform the following steps:

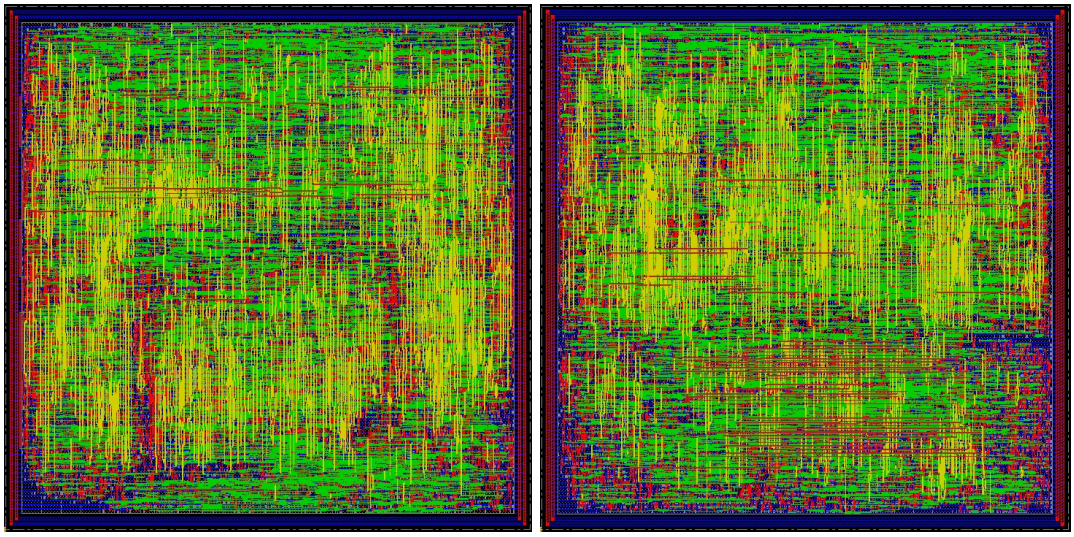
- Importing the design;
- Floorplanning;
- Power planning and routing;
- Cell placing;
- Signal routing;
- Timing and design analysis.

The procedure returned no violations in connections and geometry. The resulting netlist was saved together with the final gate count.

	RISCV	**Modified RISCV**
Area	13533.0 μm^2	14133.6 μm^2

Table 6.2: **Innovus Reports**

Innovus was able to optimize the netlists and save a noticeable amount of area on both designs, with respect to the value obtained with Synopsys. Also in this case, the produced netlists were simulated to verify the compliance with the original design.



(a) RISCV (b) Modified RISCV

Figure 6.1: **Place and route of the two designs**