SAPIENZA
UNIVERSITÀ DI ROMA

FACOLTÀ DI INGEGNERIA DELL'INFORMAZIONE, INFORMATICA E STATISTICA

MASTER DEGREE IN ELECTRONICS ENGINEERING

Design of a single-cycle extended-RV32I processor

Student:

Mirko Lapozyk 2081548

Giovanni Panicini 1873903

Giuseppe Presti 1897020

ACADEMIC YEAR 2023-2024

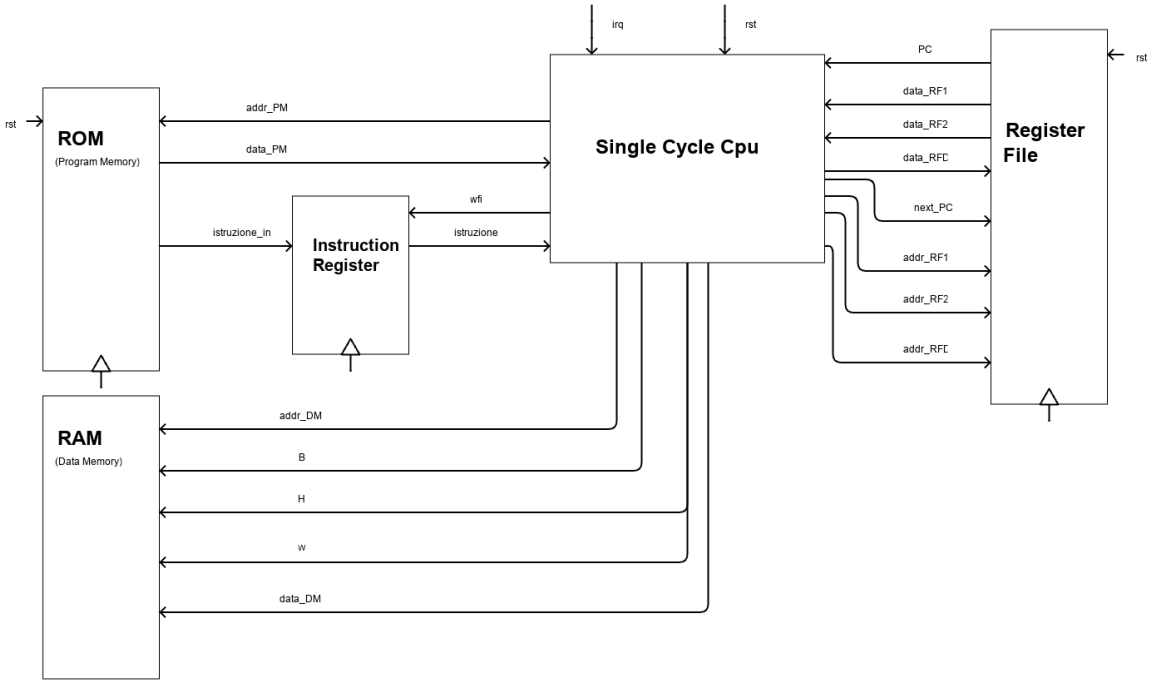
Introduction

This document provides a brief overview of the technical specification of a single-cycle processor implementing the RV32I instruction set, extended with integer multiplications, divisions and WFI instruction. The core processes a single instruction in one cycle without pipelining, so the stages of fetch, decode, execute, memory (if needed) and write are all executed within the same clock period.

The design uses external data memory, program memory, register file and also an instruction register, all implemented as separate entities with respect to the core logic.

Since the processor implements the RV32I instruction set, both the instructions format and the register file are compliant with the RISC-V standard: in particular the register file is composed of 32 registers of 32 bits each, with the addition of a program counter register dedicated to the memorization of the address of the current instruction.

The processor is designed to execute a program composed of all the supported instructions preloaded in the program memory. The architectural layout is as follows:



ROM is a 256 B memory in order to fit every possible instruction of the RV32I instruction set. This memory is composed of 256 locations of 8 bits each in order to make every instruction addressable in multiples of 4. Every 32 bit instruction is composed using 4 locations of memory, for a maximum of 64 instructions.

The data bus is 32 bits long and once the cpu has sent the program memory an address (addr_PM), the bus is loaded with the corresponding instruction: the 8 bits in the location pointed by the address are loaded in position 31 - 24 of the bus, and then the following positions (23 - 16, 15 - 8, 7- 0) are filled with the bits present in locations addr_PM + 1, addr_PM + 2 and addr_PM + 3 respectively.

RAM is a 512 *B* memory, composed of 512 locations of 8 bits each in order to be able to load/store data from/to the memory in Byte, Half-word (16 bits) or Word (32 bits) format. Data to be memorized is coming or it needs to be stored into the registers so each data is composed of 32 bits and so the relative RAM data bus (data_DM) is also 32 bits long. This memory uses the same principle as the ROM to load data into the bus: it uses the address (addr_DM) to find the corresponding starting byte and then depending on the type of data required it fills the bus with the corresponding number of bits. RAM is also connected to the core via 3 control signals B, H and W sent by the core in order to let the data memory know which kind of data is required: 8, 16 or 32 bits respectively.

Register file as already mentioned is composed of 32 locations of 32 bits each and a register dedicated to the program counter address. It's connected to the core through 6 different wires: 3 are 5 bits each and are used to send the register file the addresses of rs1, rs2 and rd (operands of the instruction and the destination register to store the data) and the other 3 are the data buses (32 bits) relative to the operand or the results of the instruction. The program counter is connected both to the program memory and to the core: the first connection is necessary in order to give the ROM the address of the instruction to be fetched, meanwhile the core needs the program counter to increment it in accord to the instruction being executed (+ 4 in case of a normal instruction or + an offset in case of a jump/branch instruction).

IR is a register that sends the fetched instruction to the core only if the WFI signal is low. This is used to implement the WFI instruction feature.

The ROM, the register file and the instruction register are connected to the same clock signal.

Signal description

Core

irq: flag in case of an interrupt request (active when high).
wfi: flag signal (active when high),out signal,to ir in case of a wfi instruction
istruzione: 32 bit long, in signal, instruction from instruction register.
addr_PM: 6 bit long, out signal, address of the ROM .
data_PM: 32 bit long, in signal, instruction read from the ROM.
addr_DM: 9 bit long, out signal, address of the RAM cell.
data_DM: 32 bit long, inout signal, from or to store in the RAM.
data_RF1: 3 bit long, in signal, first operand from register file.
data_RF2: 32 bit long, in signal, second operand from register file.
data_RFD: 32 bit long, out signal, destination operand from register file.
addr_RF1: 5 bit long, out signal, address of first operand from register file.
addr_RF2: 5 bit long, out signal, address of second operand from register file.
addr_RFD: 5 bit long, out signal, address of destination operand from register file.
PC: 32 bit long, in signal, from program counter register inside of register file.
next_PC: 32 bit long, out signal, to program counter register inside of register file.

Register file

rst: reset signal (active when high).
clk: clock signal.
addr_RF1: 5 bit long, in signal, address of first operand from core.
addr_RF2: 5 bit long, in signal, address of second operand from core.
addr_RFD: 5 bit long, in signal, address of destination operand from core.
data_RF1: 32 bit long, out signal, first operand from core.
data_RF2: 32 bit long, out signal, second operand from core.
data_RFD: 32 bit long, in signal, destination operand from core.
PC: 32 bit long, out signal, to core.
next_PC: 32 bit long, in signal, from core.

Data memory

addr_DM: 9 bit long, in signal, address of the RAM cell from core.
data_DM: 32 bit long, in out signal, from or to store in the RAM.

B, H, W: flag input signals respectively in the case of reading/writing bytes, halfwords, words.

Program memory

rst: reset signal (active when high).
clk:clock signal.
addr_PM: 6 bit long, in signal, address of the ROM from core .
data_PM: 32 bit long, out signal, instruction read.

Instruction register

instruction_in: 32 bit long, in signal, instruction from program memory.
instruction_out: 32 bit long, out signal, instruction to the core.
wfi: flag signal (active when high) from core in case of a wfi instruction.
clk: clock signal.

Testbench and simulations

In order to simulate the processor behavior we used the following VHDL files:

Single_cycle_CPU.vhd - it contains the description of our core architecture.

Program_memory.vhd - it's the description of the ROM memory and it's pre-loaded with all the instructions to be verified.

Data_memory.vhd - provides the VHDL description of the RAM behavior.

IR.vhd - contains the VHDL description of the instruction register.

RegFile.vhd - provides the description of the register file, including the program counter.

In the "testbench.vhd" file, we interconnected all the components, allowing us to perform the tests. Specifically, we need to simulate the execution of a program composed of all the supported instructions, so we created a process to generate a

5 MHz clock and an additional process where we set the reset signal to '1' for 100 ns and then reset it to '0' to initialize all the components. In the same process, we also modified the “irq” signal to test the WFI instruction.

We also present a diagram showing the core behavior, with the decoding process of each instruction and the following execution. the WFI instruction makes the core stop until an IRQ signal is received.