The American University in Cairo

School of Sciences and Engineering

CSCE 3301-02 Computer Architecture

Project 1: femtoRV32

RISC-V FPGA Implementation and Testing

Milestone 3&4

Submitted By:

Dalia Elnagar	ID: 900191234
Kirolos Mikhail	ID: 900191250
Kareem A. Mohammed Talaat	ID: 900192903

Submitted Under the Supervision of Dr:

Cherif Salama

Date: 11/20/2021

Table of Contents

Milestone Description

- In this milestone, we have implemented the pipelined datapath (single-ported memory) for RV32IMC instructions, and did thorough testing on the design.

- Also we have implemented 3 bonuses:

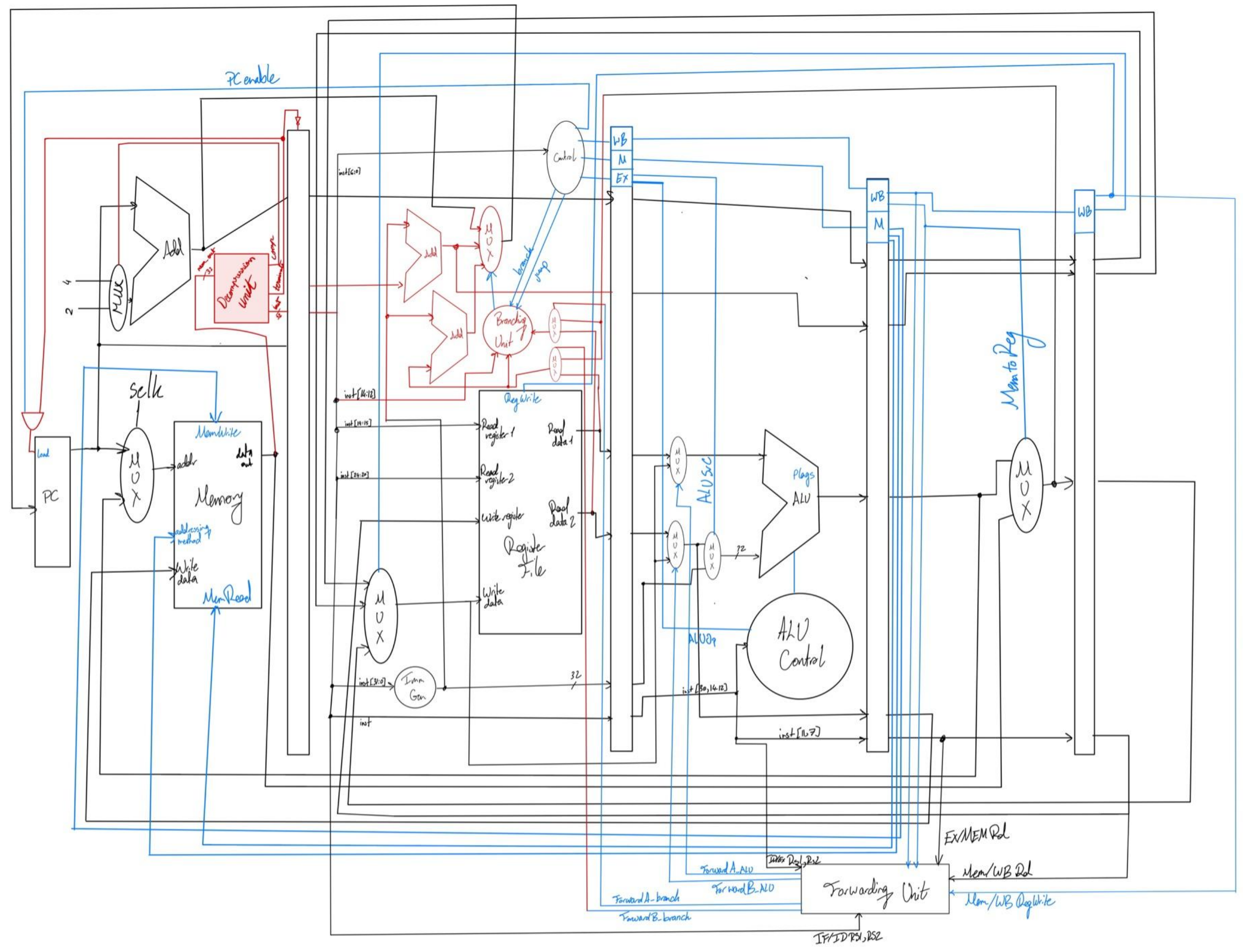
  1- Support for the integer multiplication and division instructions.

  2- Support for compressed instructions.

  3- Moved the branch outcome and target address computation to the ID stage, and handled resulting data hazards.

- Thorough testing on the bonuses are included.

Schematic


PC enable
PC
Load
Add
MUX
4
2
Decompression Unit
selk
Memory
MemWrite
MemRead
addr
data out
Write data
addressing method
Control
WB
M
EX
Add
MUX
Branching Unit
branch
jump
inst[6:0]
inst[19:15]
inst[24:20]
inst[11:7]
RegWrite
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Register File
Imm Gen
inst[31:0]
inst
32
ALUSrc
Flags
ALU
ALU Control
ALUOp
inst[30,14:12]
Mem to Reg
EX/MEM Rd
Mem/WB Rd
Mem/WB RegWrite
Forwarding Unit
ForwardA_ALU
ForwardB_ALU
ForwardA_branch
ForwardB_branch
IF/ID RS1, RS2

Modules Description

PC

Inputs and outputs

rst: a reset button to reset the value of the program counter to 0

load: an enabler which enables (PC_en) the program counter register to get incremented or decremented or obtain its current value in case we need to halt the program (terminate).

sclk: the system's slow global clock.

PC_input: the program counter input which is the value we desire the program counter to be next (for example, for a normal instruction (no jumps or anything) should be the old PC value + 4). This input is connected to the output of an MUX such that the branching control component controls the output of that MUX depending on the instruction at hand.

PC_out: the output of the program counter which is the instruction address to be used to fetch the instruction 32 bits.

Purpose

The program counter component is used to store the program counter values, and depending on the current program counter value, the instruction address is produced as an output.

Memory

Inputs and outputs

clk: the system's slow global clock.

mem_read: The memory read (boolean value) control which was saved in the EX_MEM register, indicating a memory read.

mem_write: The memory write (boolean value) control which was saved in the EX_MEM register, indicating a memory write.

AU_inst_sel: The Addressing Unit component instruction select control which was saved in the EX_MEM register, indicating the instruction on which the Addressing Unit component is going to handle.

signed_int: The signed instruction (boolean value) control which was saved in the EX_MEM register, indicating an unsigned or signed instruction.

mem_in: The address which the memory is going to operate on.

read_data2: The register (rs2) coming from EX_MEM register which was an output of the register file, acting as the data in for the memory which is the 32 bit input data to be written to the memory

mem_out: The output data resulted from the memory.

Purpose

This component memory is working as an instruction memory and as a data memory for this pipelined design.

During the positive clock cycle, the memory acts as an instruction memory and reads the word pointed at by the address. During the negative clock cycle, the memory acts as a data memory, doing 2 main operations:

1- Reads from memory by setting the data_out to be the concatenation of a group of 32 bits which differ according to the load instruction. LW loads the concatenation of 4 bytes. LH loads the concatenation of 2 bytes while sign-extending the most significant bit of the byte with the bigger address. LHU concatenates the 2 bytes and fills the uppermost 16 bits with zeros. LB and LBU are the same as LH and LHU with the difference being loading 1 byte.

2- Writes to memory by setting the concatenation of some bytes in the memory array to be either all or some bytes from the 32 bit data_in. The number of bytes concatenated is decided according to the AU_inst_sel

Decompression Unit

Inputs and outputs

rst: this input is the reset of the decompression unit

clk: this is the slow clock calculated in the datapath

instruction: this is the output of the memory.

new_inst: this is the output of the decompression unit and is the new instruction generated.

terminate: this is a flag that is one only when a condition of the compressed instructions condition is not satisfied.

comp: this is a flag that is set to one when a compressed instruction is detected.

Purpose

This unit takes the output of the memory, regardless of it being the instruction or the output of the data memory, and it checks if the input is a 32-bit instruction, data or a 16-bit instruction using the Opcode. If data or 32-bit instructions are supplied, then the value is passed as it is, otherwise, it is a 16-bit instruction, so it depending on which instruction it is, the instruction is then mapped to the bigger 32 bit instructions, and hence no change will be made in any other component than the adder of the PC to 4, as it either takes 2 or 4 depending on whether the instruction is compressed or not.

Register

Inputs and outputs

clk: an input clock

rst: a reset button to reset the values of the register entries.

terminate: to stall the pipeline and flush purposes.

n: The number of bits of the register inputs (size of the register).

Other inputs: These include inputs from the previous stage(s) (For example, control values, ALU outputs, memory data out, etc) to be stored in the register.

Other outputs: These include outputs of the register file to be used in the next stage(s) (for example, IF_ID_inst, ID_EX_gen_out, EX_MEM_mem_wrtie, etc).

Purpose

This component is used between different pipeline stages to store data between stages and allow the pipeline to occur through holding specific data until it is required in a following stage or until it is over-written.

Control Unit

Inputs and outputs

rst: a reset button to reset all the values to 0

inst: the 32 bit instruction which is coming from the IF_ID register to be used as an input.

pc_en: an enabler which enables the program count register to get incremented or decremented or obtain its current value in case we do not want to load the next instruction produced as an output.

Branch: a boolean enabled when the instruction being currently executed is a branching instruction.

Jump: a boolean which is enabled when the instruction being currently executed is a jumping instruction (jal, jalr).

Mem_read: a boolean which is enabled when the instruction being currently executed requires a memory read (lw for example)

Mem_to_reg: a boolean which is enabled when the instruction being currently executed requires saving a value from the memory to the register file (add for example).

Mem_write: a boolean which is enabled when the instruction being executed requires writing into the memory.

ALU_Src: a select which is used to select the input value into the ALU.

Reg_write: a boolean which is enabled when writing to the register file is

required.

Signed_inst: a boolean which is enabled when the instruction being executed is a signed instruction.

AU_inst_sel: 2 bit selection used to select the type of the loading or storing instruction (LH, LB, LBU, SW, etc…)

ALUOp: 3 bit selection used to select the type of operation the ALU is going to be executed.

RF_MUX_sel: a 2 bit selection used to select the value to be used as the 32 bits value to be written into the register file (write_data) based on the instruction at hand.

Purpose

The control unit is using the instruction bits to identify the instruction at hand, and therefore produce controlling outputs (like selects, branch, jump, etc) which regulate the other components depending on the instruction at hand.

Register File

Inputs and outputs

rst: a reset button to reset the values of the registers to 0

clk: the negative of the system's slow global clock.

read_reg1: the read register 1 address from the instruction coming from the IF_ID register (5 bits from bit 15 to bit 19 in the instruction)

read_reg2: the read register 2 address from the instruction coming from the IF_ID register (5 bits from bit 20 to bit 24 in the instruction)

write_reg: the destination register address from the instruction coming from the MEM_WB register (5 bits from bit 7 to bit 11 in the instruction)

write_data: the 32 bits register value to write into the register file in the destination register address. The input to this register is the output of a 3x1 MUX to support R-instructions, AUIPC, and JAL/JALR

reg_write: the 1 bit select which enables/disables writing in the register file, specifically in the destination register, and it is coming from the MEM_WB register.

read_data1: the 32 bits register value produced as an output from the register file which represents the first source register in the instruction.

read_data2: the 32 bits register value produced as an output from the register file which represents the second source register in the instruction.

Purpose

The register file is used to store the registers values, and there can be multiple operations executed on it:

We can read register values based on the input address.

We can modify the register value (the write_address is used to identify that register), and write_data is the value to be written into the register, and all of this is enabled by the reg_write, so if it is 1 then we can change that register's value, otherwise we cannot edit it.

Immediate generator

Inputs and outputs

IR (inst): the 32 bit instruction as an input coming from the IF_ID register.

gen_out (immediate): the immediate value produced depending on the instruction.

Purpose

The immediate generator is used to generate the immediate value based on the instruction to be used later in other components depending on the instruction at hand, for example, in a beq instruction, the immediate generator is used to generate the immediate value from the instruction bits, then shift it to the left (multiply by 2) to be the offset to be added to the current program counter to represent the new program counter value.

Forwarding Unit

Inputs and Outputs

ID_EX_RegisterRs1: the read register 1 address from the instruction coming from the ID_EX register (5 bits from bit 15 to bit 19 in the instruction)

ID_EX_RegisterRs2: the read register 1 address from the instruction coming from the ID_EX register (5 bits from bit 20 to bit 24 in the instruction)

IF_ID_RegisterRs1: the read register 1 address from the instruction coming from the IF_ID register (5 bits from bit 15 to bit 19 in the instruction)

IF_ID_RegisterRs2: the read register 1 address from the instruction coming from the IF_ID register (5 bits from bit 20 to bit 24 in the instruction)

MEM_WB_reg_write: the 1 bit select which enables/disables writing in the register file, specifically in the destination register, and it is coming from the MEM_WB register.

EX_MEM_reg_write: the 1 bit select which enables/disables writing in the register file, specifically in the destination register, and it is coming from the EX_MEM register.

MEM_WB_RegisterRsd: the destination register address coming from the MEM_WB register.

EX_MEM_RegisterRsd: the destination register address coming from the EX_MEM register.

forwardA_ALU: A select wire used in the MUX (the one that decides the first input to the ALU) to select whether a value is going to be forwarded or not.

forwardB_ALU: A select wire used in the MUX (the one that decides the second input to the ALU) to select whether a value is going to be forwarded or not.

forwardA_branch: A select wire used in the MUX (the one that decides the MUX first input to the branching unit) to select whether a value is going to be forwarded or not.

forwardB_branch: A select wire used in the MUX (the one that decides the MUX second input to the branching unit) to select whether a value is going to be forwarded or not.

Purpose

This component is used to handle data hazards. Forwarding is used to handle RAW hazards.

The ALU Control Unit

Inputs and Outputs

ALUOp: The ALUOp is the operation determined by the control unit and is passed to the ALU control unit to decide the instruction format. This comes from the ID_EX register.

Inst: This is the full instruction, which is used by the ALU Control unit to decide the ALU operation. (Mainly function three, from bits 12 to 14 and bit 30 are used) This comes from the ID_EX register.

ALU_selection: This output decides the operation to be done by the ALU.

Purpose

The ALU Control unit tells the ALU what to do in terms of operations, which can be narrowed down to a group of simple operations. It takes the ALUOp variable from the Control Unit, which determines the type of the instruction passed to it. Mainly, the I and S type and the Jalr instruction are grouped under (000), the branching and the Jal instruction under (001), the R-format instructions under (010), the AUIPC and LUI instructions are under (011), the M-format instructions are under (100) (Later on where removed from the ALU completely).

Depending on the ALUOp, the Alu Control Unit checks function three and bit number 30, if needed. Even though there will be no other cases than the 5 ALUOp cases, a default case is added to pass ALU ADD.

The ALU

Inputs and Outputs

A and B: These inputs are the inputs the operation is going to be done on. A is the output of the MUX handling forwarding (if any) and B is the result of the multiplexing of output of the MUX handling forwarding (if any) and the output of the immediate generator.

alu_control: This input is the control produced by the Alu Control unit and it decides the operation of the ALU.

ALUout: This output is the result of the operations done on A and B based on the ALU Control.

Z, V, C, S: These are the flags mainly used by the branching unit to decide on whether to branch or not.

Purpose

This component takes three main inputs and gives out just a single output.

The ALU takes the two operands, and performs an operation (included in the I or M set of instructions) on them and results in an output based on the instruction at hand.

Branching Unit

Inputs and Outputs

B and jump: These are the flags set by the Control Unit based on the instructions sent to the control unit.

Funct3: These are the three bits of function 3 and are used to know which branching condition we should look for and how to handle them. This is coming from the IF_ID register (instruction bits 12 to 14).

data1: value at rs1

data2: value at rs2

decision: This is a 2-bit output and

Purpose

This unit decides on whether to branch, Jal, Jalr or increment normally. If B is zero and jump is one, it chooses Jalr. If both are one, then it jumps. If both are zero, it decrements normally. Otherwise, function 3 is checked and based on it compares the data for said branching condition.

Testing

We did a total of 5 test cases. The main 3 test cases cover the 40 I instructions. The simulation output graph file is present in the zip file along with an excel sheet recording all the values at each cycle, colour-coded according to instructions. The other 2 test cases are for the M instructions and C instructions. There results are present in the zip file as screenshots.