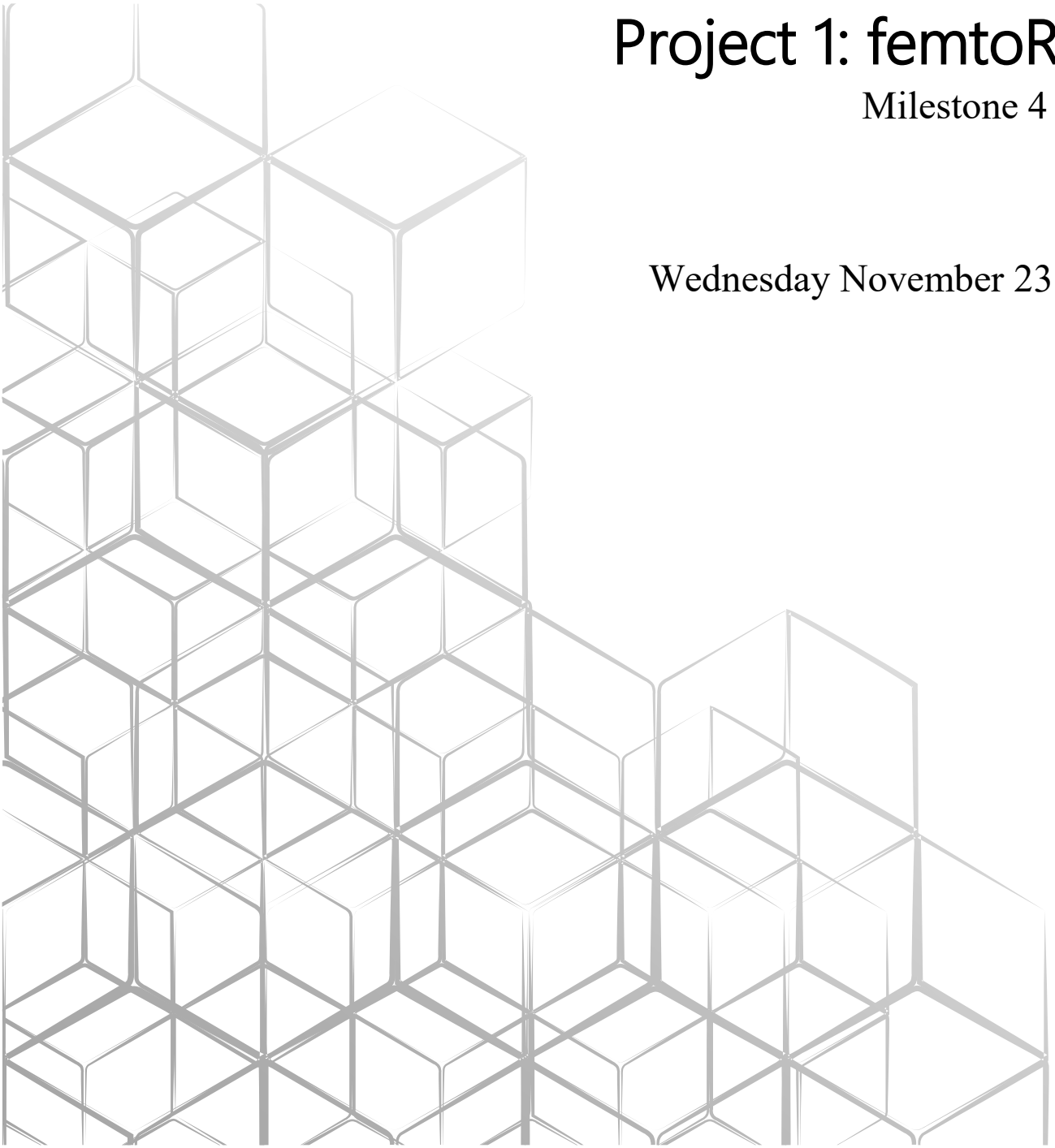
Farah Kabesh – 900191706
Karim Sherif – 900191474
Omar Fayed – 900191831

CSCE 3301 – Computer Architecture
Cherif Salama



Project 1: femtoRV32

Milestone 4 Report

Wednesday November 23, 2022

Milestone 4 Summary:

The aim of this milestone is to implement a pipelined RISC-V Processor that supports the RV32I base integer instruction set and RV32M instruction set using Verilog. The processor should be designed to execute all user-level instructions (including integer multiplication and division) which can be found below, as well has handle all hazard situations.

RV32I Base Instruction Set

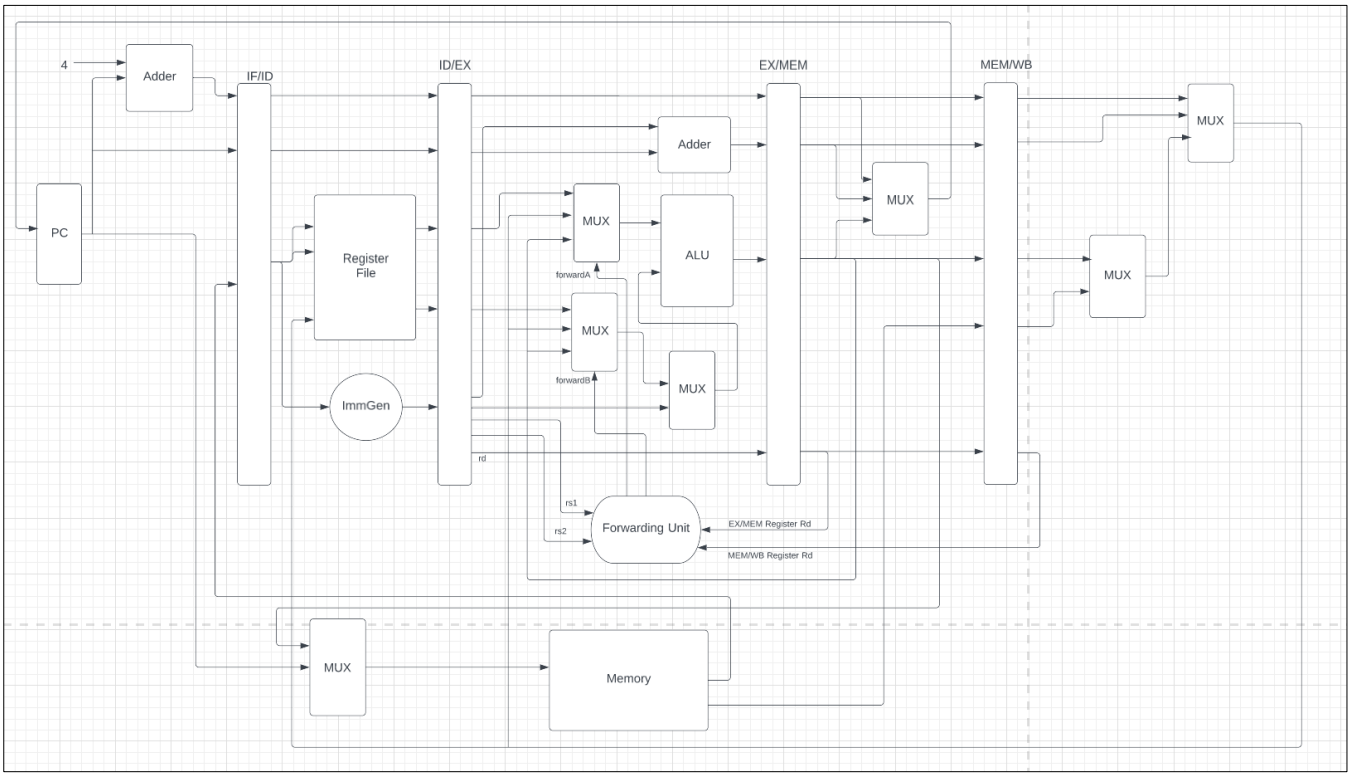
imm[31:12]				rd	0110111	LUI
imm[31:12]				rd	0010111	AUIPC
imm[20|10:1|11|19:12]				rd	1101111	JAL
imm[11:0]		rs1	000	rd	1100111	JALR
imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011	BEQ
imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011	BNE
imm[12|10:5]	rs2	rs1	100	imm[4:1|11]	1100011	BLT
imm[12|10:5]	rs2	rs1	101	imm[4:1|11]	1100011	BGE
imm[12|10:5]	rs2	rs1	110	imm[4:1|11]	1100011	BLTU
imm[12|10:5]	rs2	rs1	111	imm[4:1|11]	1100011	BGEU
imm[11:0]		rs1	000	rd	0000011	LB
imm[11:0]		rs1	001	rd	0000011	LH
imm[11:0]		rs1	010	rd	0000011	LW
imm[11:0]		rs1	100	rd	0000011	LBU
imm[11:0]		rs1	101	rd	0000011	LHU
imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	SB
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	SH
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	SW
imm[11:0]		rs1	000	rd	0010011	ADDI
imm[11:0]		rs1	010	rd	0010011	SLTI
imm[11:0]		rs1	011	rd	0010011	SLTIU
imm[11:0]		rs1	100	rd	0010011	XORI
imm[11:0]		rs1	110	rd	0010011	ORI
imm[11:0]		rs1	111	rd	0010011	ANDI
0000000	shamt	rs1	001	rd	0010011	SLLI
0000000	shamt	rs1	101	rd	0010011	SRLI
0100000	shamt	rs1	101	rd	0010011	SRAI
0000000	rs2	rs1	000	rd	0110011	ADD
0100000	rs2	rs1	000	rd	0110011	SUB
0000000	rs2	rs1	001	rd	0110011	SLL
0000000	rs2	rs1	010	rd	0110011	SLT
0000000	rs2	rs1	011	rd	0110011	SLTU
0000000	rs2	rs1	100	rd	0110011	XOR
0000000	rs2	rs1	101	rd	0110011	SRL
0100000	rs2	rs1	101	rd	0110011	SRA
0000000	rs2	rs1	110	rd	0110011	OR
0000000	rs2	rs1	111	rd	0110011	AND
fm | pred | succ		rs1	000	rd	0001111	FENCE
000000000000		00000	000	00000	1110011	ECALL
000000000001		00000	000	00000	1110011	EBREAK

RV32M Standard Extension

0000001	rs2	rs1	000	rd	0110011	MUL
0000001	rs2	rs1	001	rd	0110011	MULH
0000001	rs2	rs1	010	rd	0110011	MULHSU
0000001	rs2	rs1	011	rd	0110011	MULHU
0000001	rs2	rs1	100	rd	0110011	DIV
0000001	rs2	rs1	101	rd	0110011	DIVU
0000001	rs2	rs1	110	rd	0110011	REM
0000001	rs2	rs1	111	rd	0110011	REMU

Description of Implementation:

Our implementation is designed to build the following pipelined datapath block diagram to support the 6 RISC-V instruction formats R-type, I-type, S-type, B-type, U-type, and J-type. The control unit, AND gate, branch control signals and multiplexer which implements instruction flushing (to handle control hazards) are not shown in the diagram.



The RISC-V pipeline consists of 5 stages:

1. IF: instruction fetched from memory.
2. ID: instruction decoded and register read.
3. EX: instruction executed or memory address calculated for store/load operations.
4. MEM: access memory operand.
5. WB: write result back to register.

Verilog modules were written to represent each component of the datapath. A module named *pipeline* was created to connect the components and construct the pipelined datapath by using wire initializations and module instantiations. This module was selected as the top module in the project. Modifications were made to the single cycle processor implementation to add IF/ID, ID/EX, EX/MEM, and MEM/WB registers. These registers were created using

parametrized *nbitreg* module instantiations. These registers exist in between stages to hold information produced in the previous cycle.

Our implementation uses a single memory for both data and instructions. Memory is single-ported and byte addressable (1 byte = 8 bits). Structural hazards occur as a result of having only one memory. In order to overcome this issue, instructions will be issued every 2 clock cycles. Memory access of different instructions will not occur on the same clock cycle.

Instructions are executed in 6 cycles and divided into 3 stages. Each stage uses 2 clock cycles, which is why *sclk* (slow clock) used in our implementation, in addition to clk which already existed in the single cycle implementation. A *clock divider* module was created to create *sclk*.

- Stage 0: IF (sclk) and ID (~sclk).
- Stage 1: EX (sclk) and MEM (~sclk).
- Stage 2: WB (sclk).

Since instructions are issued every 2 clock cycles, a hazard detection unit is not needed to handle load-use hazards. Data hazards were handled by creating a forwarding unit. On the other hand, control hazards were handled by implementing instruction flushing. The outcome of the branch module (PCsel) serves as a flag that indicates the need for instruction flushing. A multiplexer was added to the main implementation module to provide the control signals to be stored in ID/EX pipeline register. PCsel is the selection line of this multiplexer. If the branch is to be taken, zeroes will be stored instead of all control values.

The following section provides a brief description of the design and implementation of the main components of the datapath using Verilog.

N-Bit Register with Reset and Load Control (nbitreg.v)

The N-bit register was created and instantiated in the *Register File* module as well as in the *Pipeline* module to design the program counter and registers needed for the pipeline. Modules for the *D Flip-Flop* and a parameterized *2x1 Multiplexer* were written in order to

implement their functions and instantiate them inside a generate block in the *N-bit register* module.

Register File with Reset (regfile.v)

The register file contains 32 registers with 2 reading ports and 1 writing port. The writing port is only active if RegWrite Signal and Load are equal to 1. In order to write data, a generate block is used to loop the code for the 32 registers. The generate block included an instantiation of *nbitreg* module in order to write data and assign it to Q (output). In order to read data, 2 assign statements were used to read register 1 and read register 2.

Immediate Generator (rv32ImmGen.v)

The Immediate Generator was created in order to take in different instruction formats and accordingly generate the immediate constant for I-type, S-type, U-type and J-type formats. The Verilog module provided for the Immediate Generator was used along with the *defines.v* module.

Control Unit (controlUnit.v)

The control unit is used to read instructions and generate control signals based on their 7-bit opcode. The control unit supports all instruction format types and outputs the following signals: Branch, MemRead, MemtoReg, MemWrite, ALUSrc, RegWrite, ALUOp as well as New (which was created in order to include the LUI, AUIPC, JAL and JALR instructions).

ALU Control Unit (alucontrol.v)

The ALU Control processes instruction bits [14:12] (funct3) and instruction bit [30] of the instruction and assigns it to the corresponding alusel (ALU select line) value based on the 2-bit ALUOp (00, 01, 10, 11). Load/Store instructions have ALUOp 00 and their function is addition. Branch instructions have ALUOp 01 and their function is subtraction. R-Format instructions and the RV32IM instruction set instructions have

ALUOP 10. I-Format Instructions have ALUOp 11. Verilog case statements and if-Statements were used for the design of this module.

ALU (prv32_ALU.v)

The Arithmetic Logic Unit carries out the arithmetic, logic, and shift operations as well as the operations for set-less-than and set-less-than-unsigned (SLT and SLTU). It also generates the flags needed for implementing the conditional branches, which are Zero: Z, Carry: C, Overflow: V, and Sign: S. The Verilog description provided for the ALU was used.

Branch Module (branch.v)

This module was created to implement the conditional branches (BEQ, BNE, BLT, BGE, BLTU, BGEU). It checks the following flags that were generated by the *ALU* module: Zero: Z, Carry: C, Overflow: V, and Sign: S. The table below summarizes the flag conditions for each branching instruction.

	Branch if
BEQ	Z
BNE	~Z
BLT	S != V
BGE	S == V
BLTU	~C
BGEU	C

The output of this module (PCsel) is the select line of the multiplexers associated with the program counter and instruction flushing.

Memory (newDM.v)

This is where instructions are fetched and data is read from and written to. The memory is designed to be byte addressable. Instructions are fetched at every positive edge of the clock and data is written using the Store instructions (SB, SH, SW) at every negative edge of the clock and when the control signal MemWrite is equal to 1. When the control signal MemRead is equal to 1, data is read using the Load instructions (LB, LH, LW, LBU, LHU). Funct3 (instruction bits [14:12]) is provided as input and if-statements are used to distinguish between storing/loading a byte (8 bits), halfword (16 bits) or word (32 bits). RISC-V is little endian, so the least significant bytes are stored before the most significant bytes.

Forwarding Unit (forwardingUnit.v)

The forwarding unit is designed to handle data hazards by doing the following:

1. Forward the result from the EX/MEM pipeline register when:
    a. EX/MEM Register Rd = ID/EX Register Rs1
    b. EX/MEM Register Rd = ID/EX Register Rs2
    c. EX/MEM RegWrite
    d. EX/MEM Register Rd $\neq$ 0

2. Forward the result from the MEM/WB pipeline register when:
    a. MEM/WB Register Rd = MEM/WB Register Rs1
    b. MEM/WB Register Rd = MEM/WB Register Rs2
    c. MEM/WB RegWrite
    d. MEM/WB Register Rd $\neq$ 0

Additional Modules needed in constructing the datapath: Adder, 2x1 Multiplexer and 4x1 Multiplexer.

A simulation source (testbench) was created where the top module (*pipeline*) is instantiated in order to run test cases.