North South University
Department of Electrical and Computer Engineering

CSE-332 PROJECT REPORT

Design of 17 Bit Datapath

Name	Nazmul Hasan
Student Id	1911742042
Course Code	CSE-332
Section	3
Date of Submission	January 27, 2021

Objective:

My task was to design 17 bit datapath in Logisim, which can work on R type and I type instruction. The operations that I have been given are ADD (addition), MULT (multiplication), X-nor and nor operations. S0, I the main objective is to design a logisim circuit that can execute these instructions.

Design process:

1 bit Arithmetic Logic Unit: As I have given 4 operations (addition, multiplication, x-nor and nor), I have designed a 1 bit ALU unit first. This ALU unit can execute all these four operations between two input bits and generates 1 bit output. The desired output is selected with the help of a 4x1 mux that has a 2 bit selection line, used as operation code.

Operation code 00 is for addition,

01 is for multiplication,

10 is for X-nor,

11 is for nor operation.

17 bit ALU: The same 1 bit ALU is used to design a 17 bit ALU, so that it can execute operations on 17 bit operands and generate a 17 bit output.

17 bit Register File: I have used a 17 bit register file for storing data that are used the operations in ALU. So, data will be read from here to execute an instruction, and data may be stored into register after instructions like load and R type instructions. The register file has RS, RT and RD that are used to select source and destination registers while decoding an instruction.

ISA: My ISA bit is 17 bits. I have divided it into 4 parts.

Bits 0-3 are for RD (4 bits),

Bits 4-7 are for RT (4 bits),

Bits 8-11 are for RS (4 bits),

Bit 12-16 are for Opcode (5 bits).

Datapath Design: It has five stages: fetch, decode, execute, memory access and write back. Fetch unit is for fetching instruction with the help of PC. Instructions are given into the ROM. For every clock pulse, the next instruction is fetched. A fetched instruction is then decoded and separated into RS, RT, RD and Opcode. Based on the type of instruction, RS, RT, RD are connected to the register file (for R type), or RS, RT are connected to register file (for I type). Based on the data of RS and RT registers, ALU executes different instructions. It is the execution unit. Generated result can be stored on the register (for R type), or an effective memory address can be produced (for I type instruction). It is done in memory access stage. ALU produced output can be written into register via the write data line.

In the designed datapath, there is no ALU control, and control unit. So, R type and I type instructions are selected manually.

Instruction Description:

ADD: It adds two register values and stores the result into destination register.

Operation: RD = RS + RT

Assembly code: add RD RS RT.

MULT: It multiplies two register values and stores the result into the destination register.

Operation: RD = RS * RT

Assembly code: mult RD RS RT.

X-nor: Performs x-nor operation between two register values and stores the result into destination register.

Operation: RD = RS X-nor RT

Assembly code: x-nor RD RS RT.

nor: Performs nor operation between two register values and stores the result into destination register.

Operation: RD = RS nor RT

Assembly code: nor RD RS RT.

Lw: Loads value from an effective memory address write it back to the destination register.

Operation: RD = Memory [RD+immediate]

Assembly code: lw RD RS immediate.

Sw: It stores value from register to an effective memory address.

Operation: Memory [RT + immediate] = RS

Assembly code: sw RS RT immediate.

For R type instructions:

[12-16] Opcode	[8-11] RS	[7-4] RT	[3-0] RD

For Instance: If 0123 is given to the ROM, it will be decoded into,

Opcode is 00000, which is ADD instruction.

RS is 0001, which is R1,

RT is 0010, which is R2,

RD is 0011, which is R3.

Value of R1 and R2 register will be added and ALU result will be stored in R3.

So, the assembly code is: add R3 R1 R2

And binary instruction is 00000 0001 0010 0011.

For I type instruction:

[12-16] Opcode	[8-11] RS	[7-4] RT	[3-0] Immediate

Some, I type instructions are load (lw), store (sw). Since, there is no ALU control added in the datapath, there is no specific opcode for these instructions. Rather, lw and sw signals for RAM are used for load and store type instruction. As, an effective memory address should be generated for

loading a data from a specific memory address or storing a data into a specific memory address, I have to use the opcode for add operation.

For a load (lw): If 0235 is given to the ROM, then

Opcode is 00000, which is add,

RS 0010, which is R2,

RT 0011, which is R3,

0101 is immediate value.

Immediate value and value of R3 register is added and it ALU generates an effective address from where the value is loaded into the register R2.

So, assembly code for this program is: addi R2 R3 0101

And binary instruction is: 00000 0010 0011 0101.

Logisim Circuits:

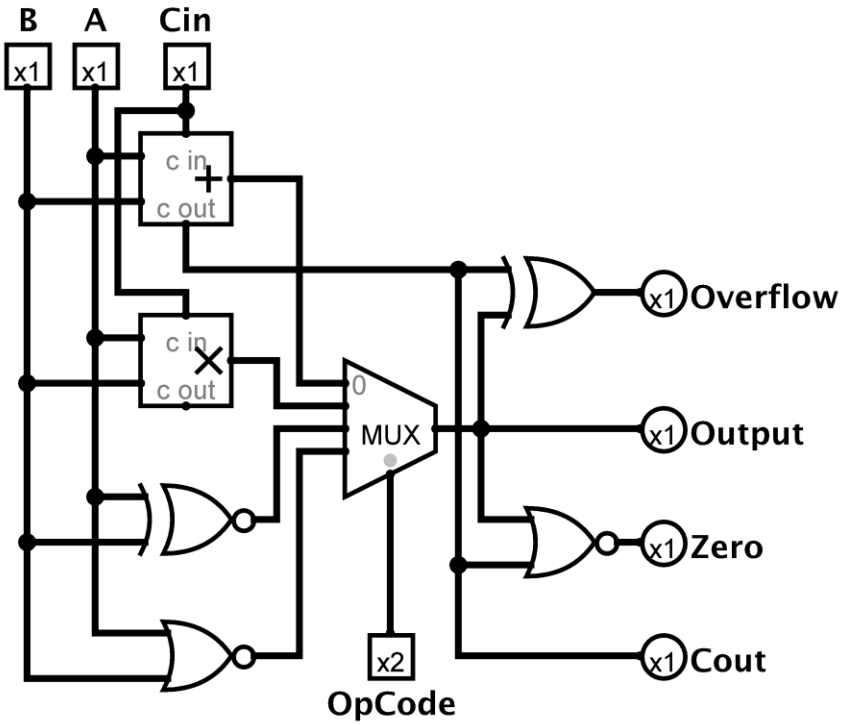

Figure: 1 bit ALU.

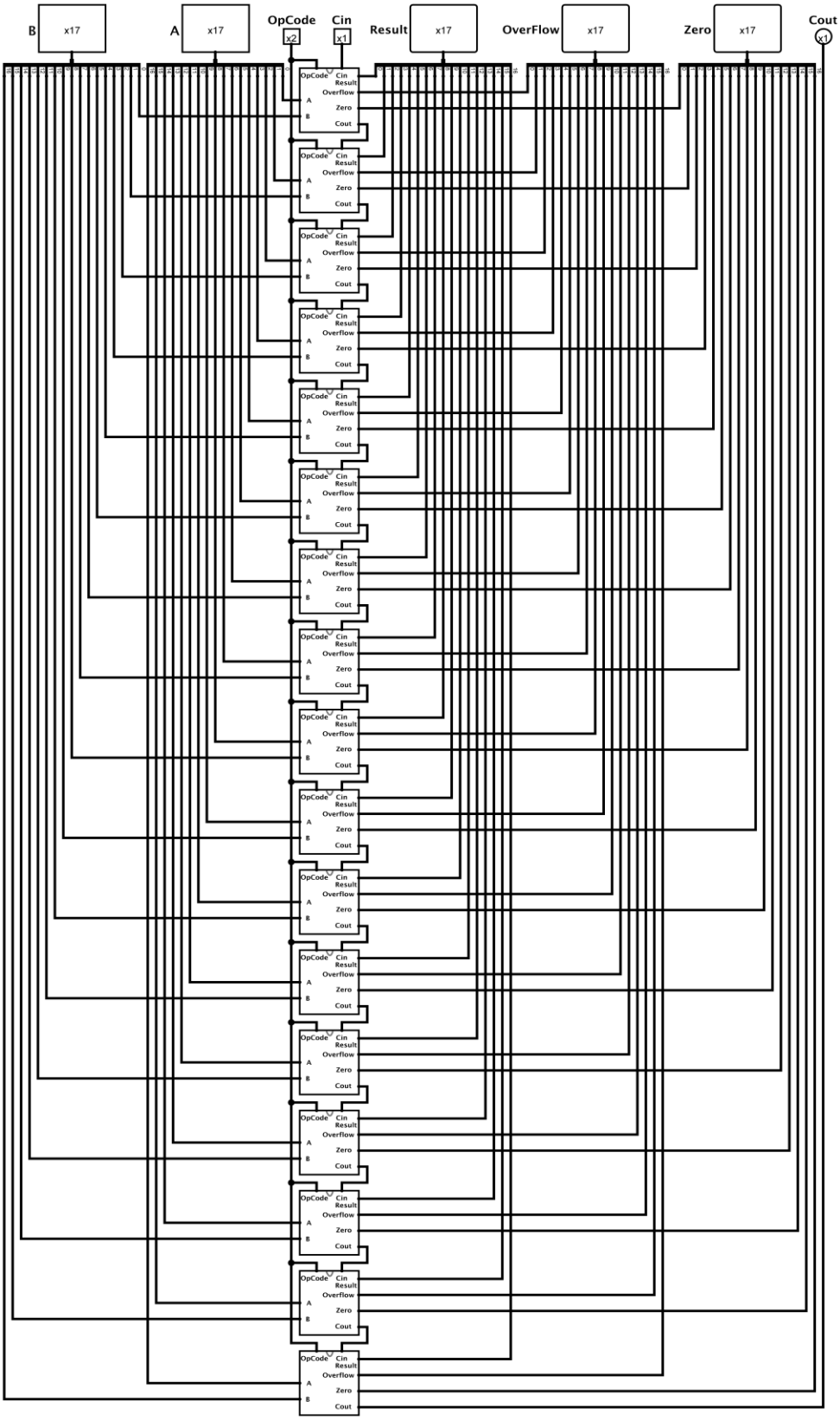

Figure: 17 Bit ALU.

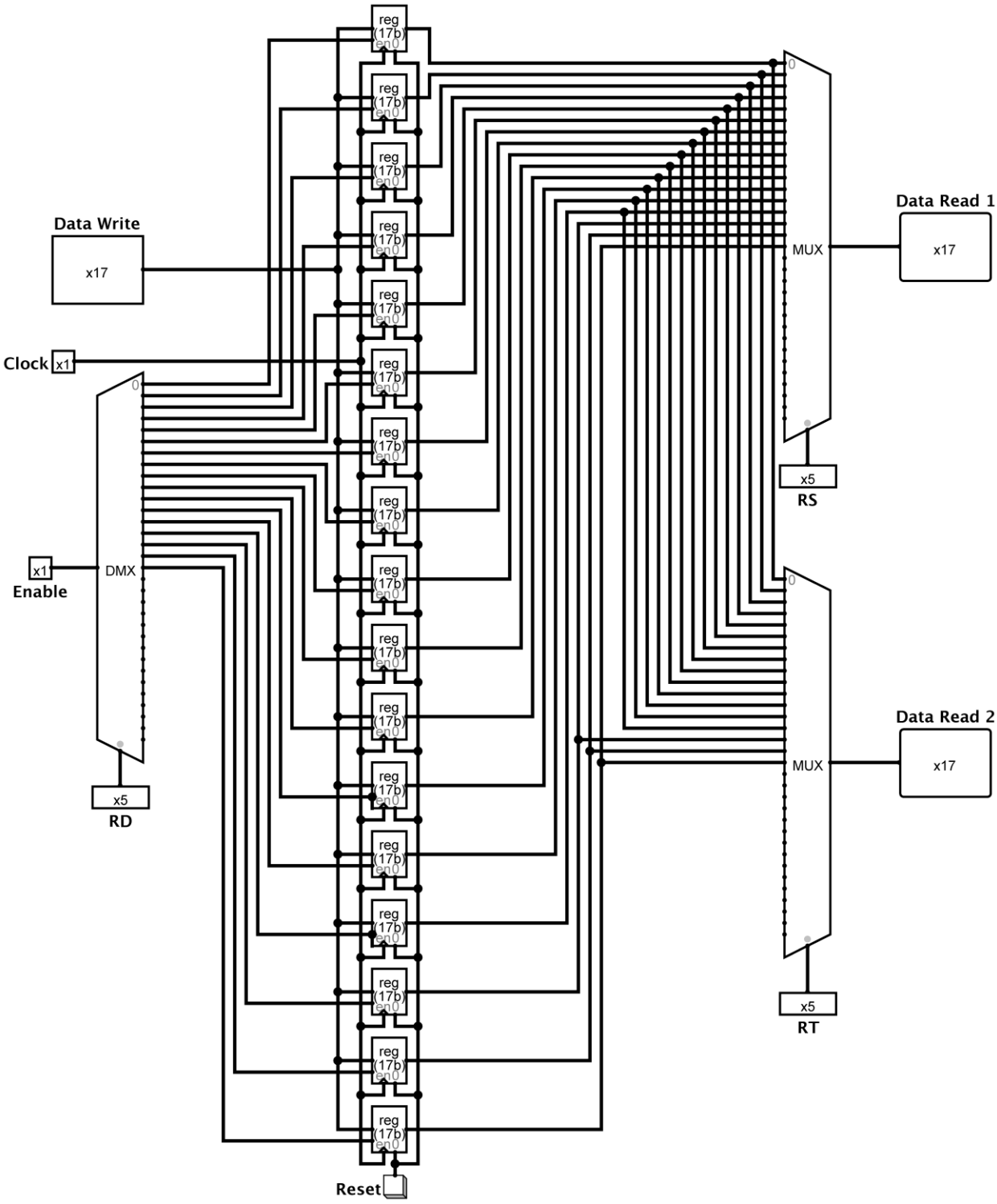

Figure: 17 bit register file.

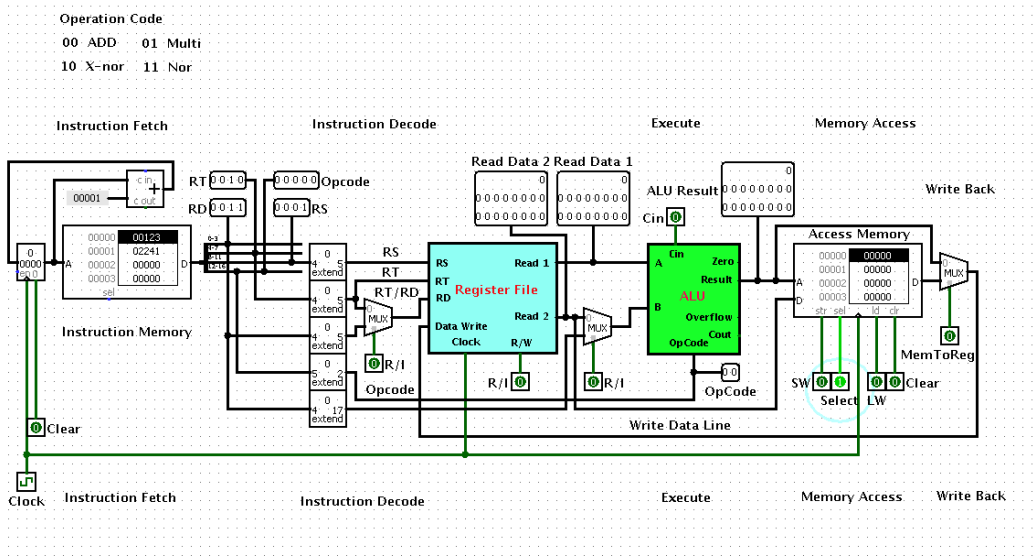

Figure: Datapath of 17 bit ISA.

END