Team Members:
Saveetha Venkatesan (SXV200028), Vishnuvaradhan Moganarengam (VXM210090)

EEDG6302: Microprocessor and Embedded Systems Wednesday Lab Report

Introduction:

MCU, we have design is 8-bit instruction set computer architecture. It is a collection of multiple interconnected modules which are controlled by control signal.

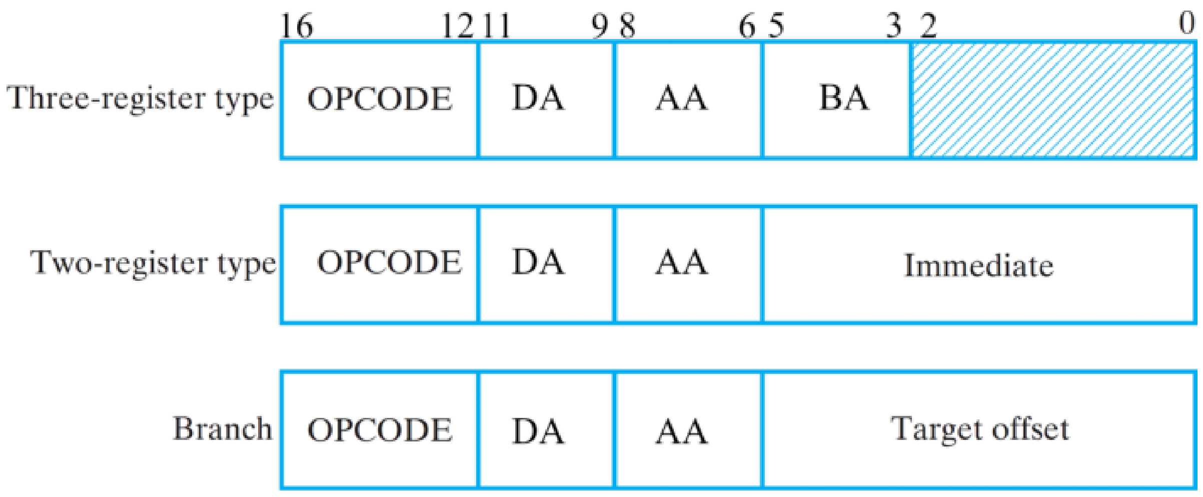
The fundamental building blocks of the module are:

- Program Memory
- Data Memory
- Register File
- Arithmetic and Logical Unit
- Multiplexors
- Instruction Decoder
- Constant Unit

MCU consists of four stages of Pipeline:

- Fetch
- Decode
- Execute
- Writeback

MCU has 17-bit instruction, where each instruction starts with 5-bit OPCODE. Followed by 3-bit destination address, and 3-bit source address. Certain Operations are three-register type, while others are two-register type.



Team Members:
Saveetha Venkatesan (SXV200028), Vishnuvaradhan Moganarengam (VXM210090)

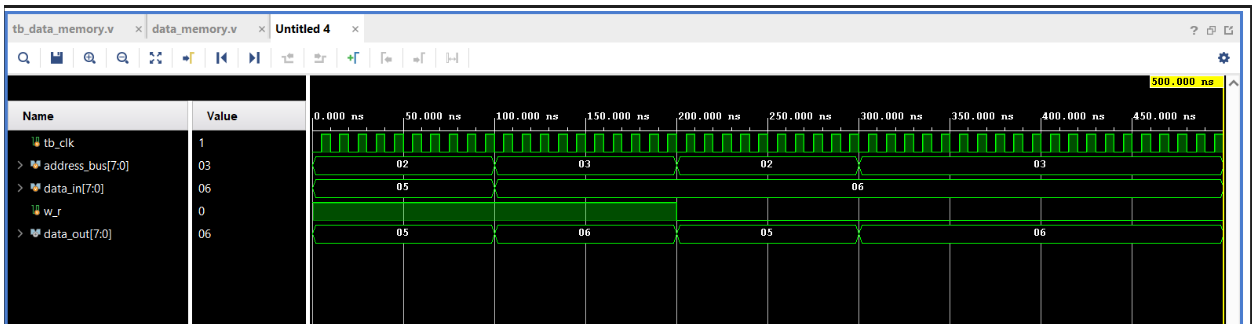
Following are the OPCODEs and their functions:

OPERATION	INSTRUCTION	ALU FS	ID CASE
No Operation	NOP	XXXX	00000
Branch if Nonzero	BNZ	0001	00001
Add	ADD	0010	00010
Compliment	NOT	0011	00011
Load	LD	XXXX	00100
Set if Less Than	SLT	0100	00101
Branch if Zero	BZ	0001	00110
Subtract	SUB	0100	00111
Output	OUT	XXXX	01000
Store	ST	XXXX	01001
Logical Shift Right	LSR	0101	01010
Add Immediate	ADI	0010	01011
Jump Register	JMR	XXXX	01100
Input	IN	0110	01101
Jump and Link	JML	0001	01110
Exclusive-OR	XOR	0111	01111
Logical Shift Left	LSL	1000	10000
OR Immediate	ORI	1001	10001
Move	MOVA	0001	10010
AND Immediate	ANI	1010	10011
Input Keyboard	INK	1011	10100

Team Members:
Saveetha Venkatesan (SXV200028), Vishnuvaradhan Moganarengam (VXM210090)

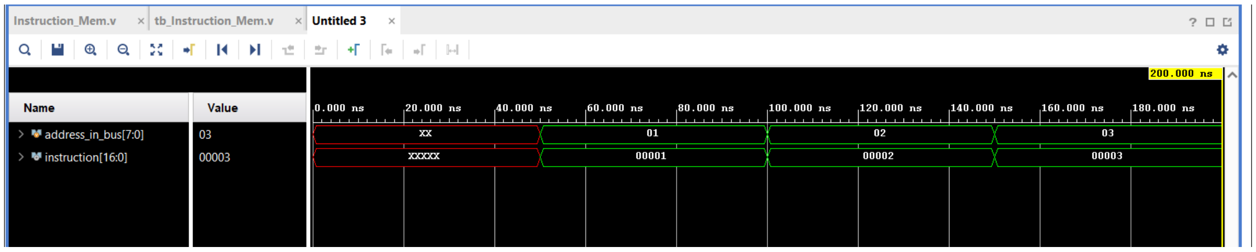
Data Memory:

Data Memory (RAM) is used to store the data created by MCU. It takes 8-bit input data, 8-bit address as input, we have assigned 256 bytes to our Data Memory. When the write enable bit is high, it stores the input data at the address received at the positive edge of the clock pulse. To test this module, we have first enabled write operation by given w_r=1, then write operation is performed on the given address. When w_r=0, then read operation is performed where we can see the output value at the given address locations.



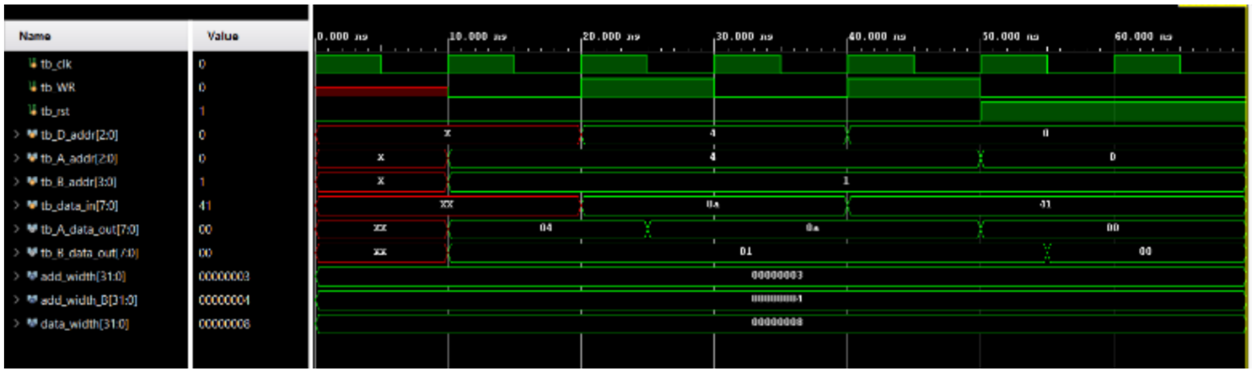
Instruction Memory:

Program memory is used for storing instructions. It takes in an 8-bit address as input and outputs a 17-bit instruction bus according to the address given. The program will create an array of size 17-bits as mentioned in the bus size provided. Since we have an 8-bit address bus, it results in 256 words each of size 17-bits. An 8-bit address with 8 arrays results in 256 words. To test Instruction memory, we hard code random instructions at certain memory locations. In a test bench we call those address. Following waveform shows memory address and their corresponding 17-bit instruction. To test this module, we gave address bit as 8-bit address i.e. 1 , the value is saved at the particular memory location as 17-bits.



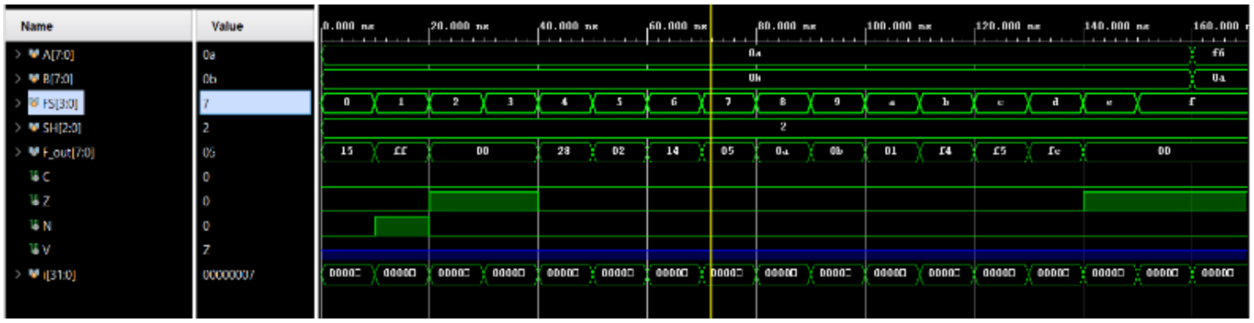
Register File:

Register File works similar to Data Memory. It has several General-purpose Registers(8-bit). It has Register file takes clk, Address A, B address, D address, 8-bit input data and clock along with write enable. If the write is enabled, the input data will be written to address given by D address register. Data output A and B are controlled by address register A and B.



Arithmetic and Logical Unit (ALU):

As the name suggests, it performs arithmetic and logical operations. This performs all the computations fed into MCU. It takes function select as input which determines which operation to be performed. Along with this it takes shift value, two ALU input A and B. It gives 8-bit output F along status flag like carry, zero, negative and overflow. We have included addition, subtraction, multiplication, division, OR, AND, NOR, NAND, NOT, XOR, greater comparison, equal comparison, logical shift left and logical shift right. To test it we have passed arguments in A and B, we get correct result in ALU output. To test this module, we will select each operation in a loop after specific delay (i.e. #10) and we will give data values for register A and B. According to the assigned operation different computations takes place and finally output is stored in F_out register.



Multiplexors:

For Multiplexors, we have developed 4 different modules and tested each. Multiplexor A and B have 2 inputs, hence used a 2:1 Multiplexor to build it. We passed random values to check the functionality of the MUX with select lines. Similarly, we built a 4:2 multiplexor for Multiplexor C and D. To verify those, we passed the values for inputs and verified with the respective select line outputs.

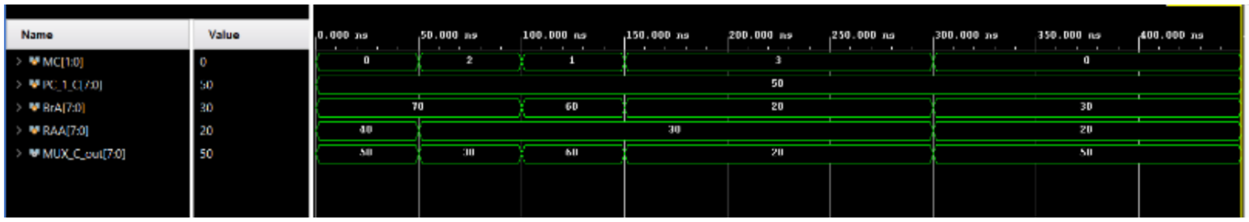
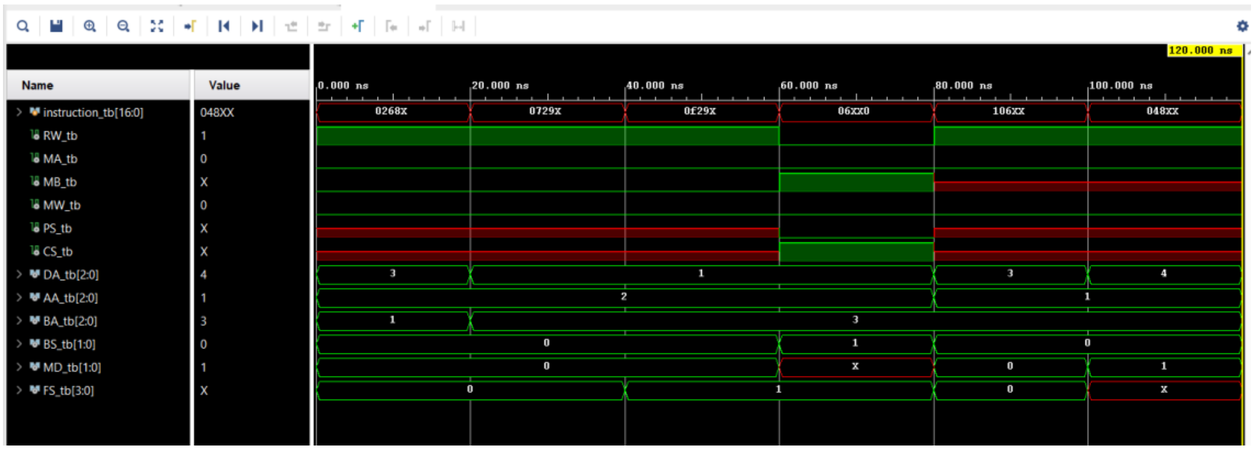


Figure shows MUX C simulation

Instruction Decoder:

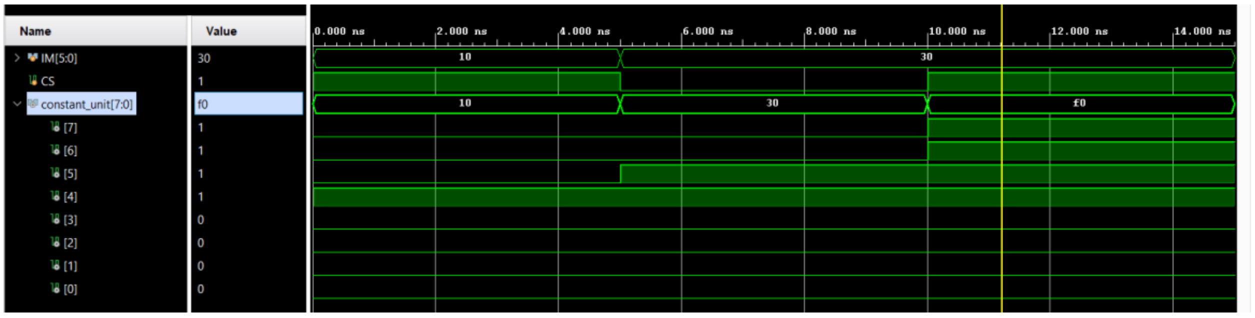
Instruction Decoder takes instruction as input and decodes the instruction and generates necessary control signal for MCU to function properly. It takes 17-bit instruction from instruction memory and generates control signals to have the MCU function properly with respect to the instruction. Verilog module decodes the 5-bit opcode from instructions. We are using a switch case to execute the corresponding opcode. We have implemented the given opcodes. To test this module, we have hardcoded few instructions like No operation, Add, logical left shift in a test bench to verify the corresponding control signal in simulation.



Control Unit (CU):

We have written Verilog module for Constant Unit for MCU. As described in the project description, we need to “extend” 6 bits to 8 bits. The duty of the Constant Unit in your MCU is to perform this extension. To test this module, we have passed hard coded inputs in a test bench to verify the sign extended output signal in simulation, where we get the desired results.

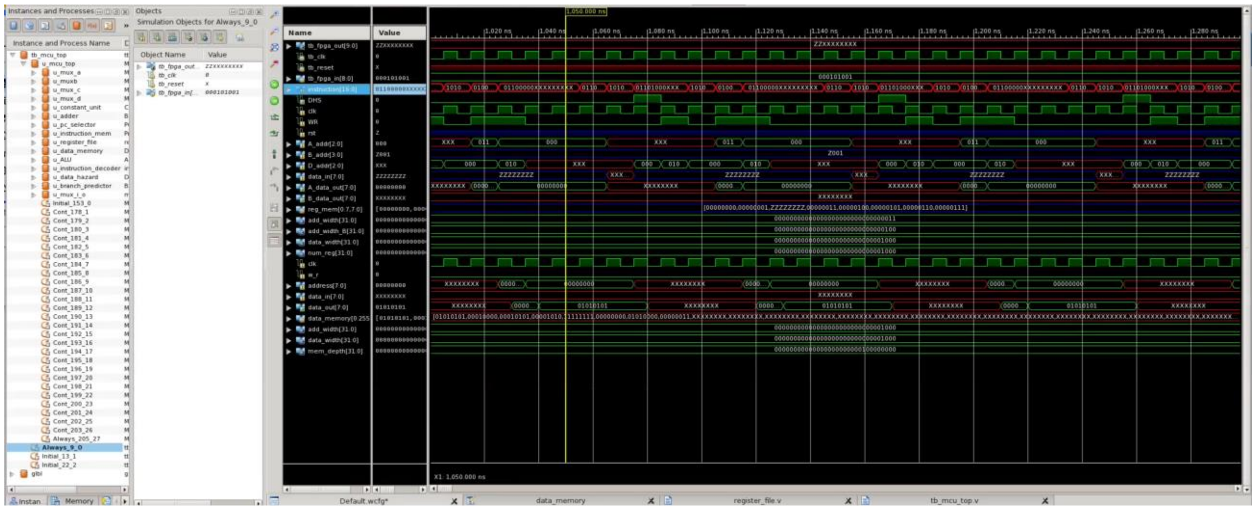
Team Members:
Saveetha Venkatesan (SXV200028), Vishnuvaradhan Moganarengam (VXM210090)



Name	Value
IM[5:0]	30
CS	1
constant_unit[7:0]	f0
[7]	1
[6]	1
[5]	1
[4]	1
[3]	0
[2]	0
[1]	0
[0]	0

MCU:

After creating all the above modules, for the MCU to function we must connect all the modules. Each module is required at various stages of pipeline. Output of one stage is input to following stages. At each stage of pipeline, registers are created to hold the values and status of result and signal generated at that stage. We have checked each function by adding the signals to the waveform.



Inference:

This project provides us the experience to build our own MCU step by step. With different drivers provided, we were able to observe different colors on the monitor. There are so many rewarding things that can be learnt from this project, as listed below.

1.Understanding the hardware design while creating the blocks
2.Improved the knowledge of coding skills while facing issues
3.Debugging skills to identify and fix the problems during the pipeline stages.

Overall, it helps us to build a wide range of technical skills, creativity, teamwork skills that can be applied to many areas. The MCU we built can be interfaced with various drivers to expand the scope of usage.