IITB-CPU

Guide: Prof. Virendra Singh

EE224 : Digital Design

Team 16

Aaditya Gupta	Dattaraj Salunkhe	Harsh S Roniyar	Pranav Prakash
22B3941	22B1296	22B3942	22B3945

CONTENTS

1 PROBLEM STATEMENT

Design a computing system, IITB-CPU, whose instruction set architecture is provided. Use VHDL as HDL to implement the CPU. IITB-CPU is a 16-bit very simple computer developed for teaching purpose based on the Little Computer Architecture. The IITB-CPU is an 8-register, 16-bit computer system, i.e., it can process 16 bits at a time. It should use point-to-point communication infrastructure.

The IITB-CPU is an 8-register, 16-bit computer system. It has 8 general-purpose registers (Ro to R7). Register R7 always stores the Program Counter (Instruction Pointer). PC points to the next instruction. All addresses are short word addresses. This architecture uses condition code register which has two flags Carry flag (C) and Zero flag (Z). The IITB-CPU is general enough to solve complex problems. There are three machine-code instruction formats (R, I, and J type) and a total of 14 instructions.

2 COMPONENTS

- **Multiplexers**: A multiplexer (or MUX) selects between several input signals and forwards the selected input to a single output line.
- **Sign Extenders**: Sign Extender performs an operation in which the number of bits representing a specific value is increased to 16 bits by padding while preserving the original sign and value.
- **ALU**: It stands for arithmetic logic unit and performs arithmetic and logic operations. It performs the functions **ADD, SUB, MUL, AND, ORA, IMP** along with **Zero (Z)** flag.
- **Left Shifter**: A Left Shifter multiplies by powers of 2. Thus, it shifts a binary number left by a specified number of positions.
- **Register File**: The RF contains 8 general purpose registers (R_0 to R_7). Register R_7 stores the Program Counter (Instruction Pointer).
- **FSM (Finite-State Machine)**: Used as a mathematical model of computation, it is an abstract machine that can be in exactly one of a finite number of states at any given time. Our design has 14 states.
- **Datapath**: A datapath is a collection of functional units such as arithmetic logic units or multipliers that perform data processing operations, registers. Along with the control unit, it composes the central processing unit (CPU).

3 INSTRUCTION SET ARCHITECTURE

3.1 Instruction Formats

Opcode	R_A	R_B	R_C	**Unused**	**Condition (CZ)**
4 bits	3 bits	3 bits	3 bits	1 bit	2 bit

Table 1: R Type Instruction format

Opcode	R_A	R_C	**Immediate**
4 bits	3 bits	3 bits	6 bits signed

Table 2: I Type Instruction format

Opcode	R_A	**Immediate**
4 bits	3 bits	9 bits signed

Table 3: J Type Instruction format

3.2 Instructions Encoding

ADD	00_00	R_A	R_B	R_C	0	00
SUB	00_10	R_A	R_B	R_C	0	00
MUL	00_11	R_A	R_B	R_C	0	00
ADI	00_01	R_A	R_B	6 bit Immediate		
AND	01_00	R_A	R_B	R_C	0	00
ORA	01_01	R_A	R_B	R_C	0	00
IMP	01_10	R_A	R_B	R_C	0	00
LHI	10_00	R_A	0 + 8 bit Immediate			
LLI	10_01	R_A	0 + 8 bit Immediate			
LW	10_10	R_A	R_B	6 bit Immediate		
SW	10_11	R_A	R_B	6 bit Immediate		
BEQ	11_00	R_A	R_B	6 bit Immediate		
JAL	11_01	R_A	9 bit Immediate			
JLR	11_10	R_A	R_B	000_000		

4 STATE DESCRIPTIONS

4.1 S0

State Operation	Control
PC → Mem_Addr	MDR
Mem_Data → T1	T1:E
PC → ALU_A	ADD
+2 → ALU_B	
ALU_C → RF_D3	WR_EN
X07 → RF_A3	

4.2 S1

State Operation	Control
r_A → RF_A1	
r_B → RF_A2	
RF_D1 → T2	T2:E
RF_D2 → T3	T3:E

4.3 S2

State Operation	Control
T2 → ALU_A	
T3 → ALU_B	
ALU_C → T4	T4:E
r_C → T_or	T_or:E

4.4 S3

State Operation	Control
T4 → RF_D3	WR_EN
T_or → RF_A3	

4.5 S4

State Operation	Control
Imm6 → SE6	
T2 → ALU_A	ADI
SE6 → ALU_B	
ALU_C → T4	T4:E
r_B → T_or	T_or:E

4.6 S5

State Operation	Control
Imm6 → SE6	
SE6 → ALU_A	
T_3 → ALU_B	
ALU_C → T_4	T_4:E

4.7 S6

State Operation	Control
T_4 → Mem_Addr	MDR
Mem_Data → T_4	T_4:E
r_A → T_or	T_or:E

4.8 S7

State Operation	Control
T_4 → Mem_Addr	MWR
T_2 → Mem_Data	

4.9 S8

State Operation	Control
T_2 → ALU_A	ADD
T_3 → ALU_B	
ALU_C → T_4	
ALU_F_Z → Z_IN	
r_C → T_or	T_or:E

4.10 S9

State Operation	Control
PC → ALU_A	
Imm6 → LS → SE6	
SE6 → ALU_B	
ALU_C → RF_D_3	WR_EN
X_{07} → RF_A_3	

4.11 S10

State Operation	Control
$r_A \rightarrow$ RF_A3	
PC $\rightarrow$ RF_D3	WR_EN

4.12 S11

State Operation	Control
$r_B \rightarrow$ RF_A2	
RF_D2 $\rightarrow$ RF_D3	WR_EN
X07 $\rightarrow$ RF_A3	

4.13 S12

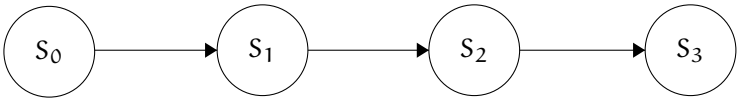
State Operation	Control
Imm9 $\rightarrow$ SE9 $\rightarrow$ LS	
LS $\rightarrow$ T4	T4:E
$r_A \rightarrow$ T_or	T_or:E

4.14 S13

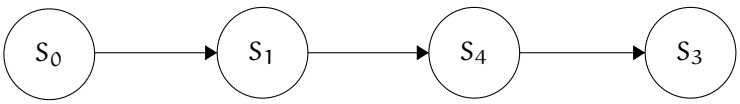
State Operation	Control
Imm9 $\rightarrow$ SE9	
SE9 $\rightarrow$ T4	T4:E
$r_A \rightarrow$ T_or	T_or:E

5 INSTRUCTION STATE-FLOW DIAGRAM

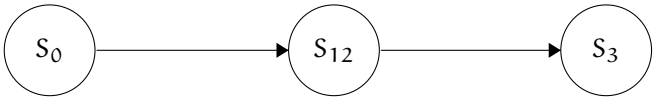
5.1 ADD/SUB/MUL/AND/ORA/IMP



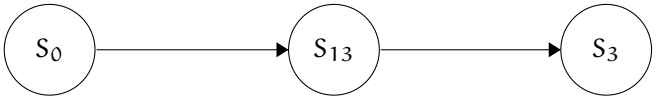
5.2 ADI



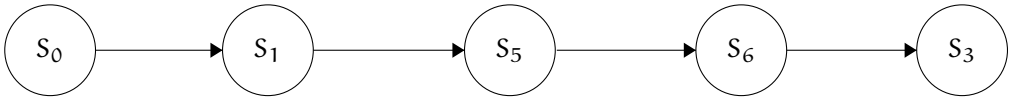
5.3 LHI



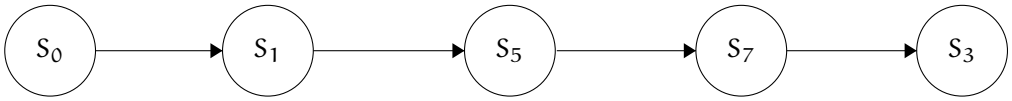
5.4 LLI



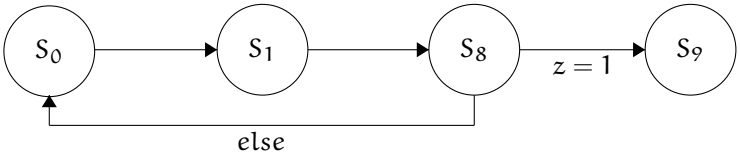
5.5 LW



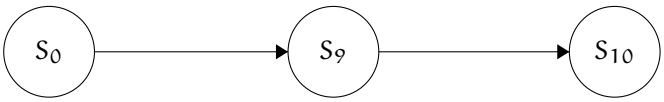
5.6 SW



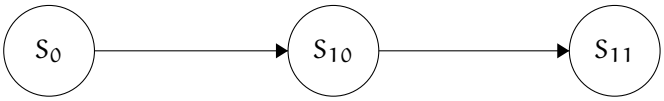
5.7 BEQ



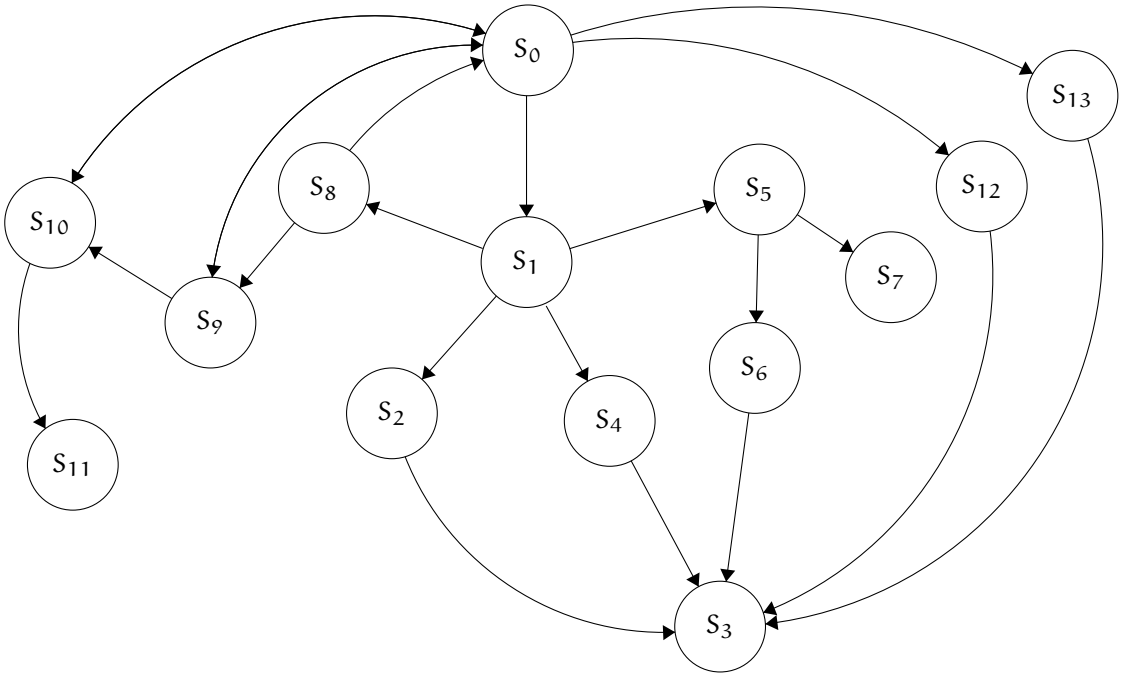
5.8 JAL



5.9 JLR



6 FINITE STATE MACHINE



Any state in the FSM with no outward arrows has been implicitly assumed to go to S_0 upon RESET.

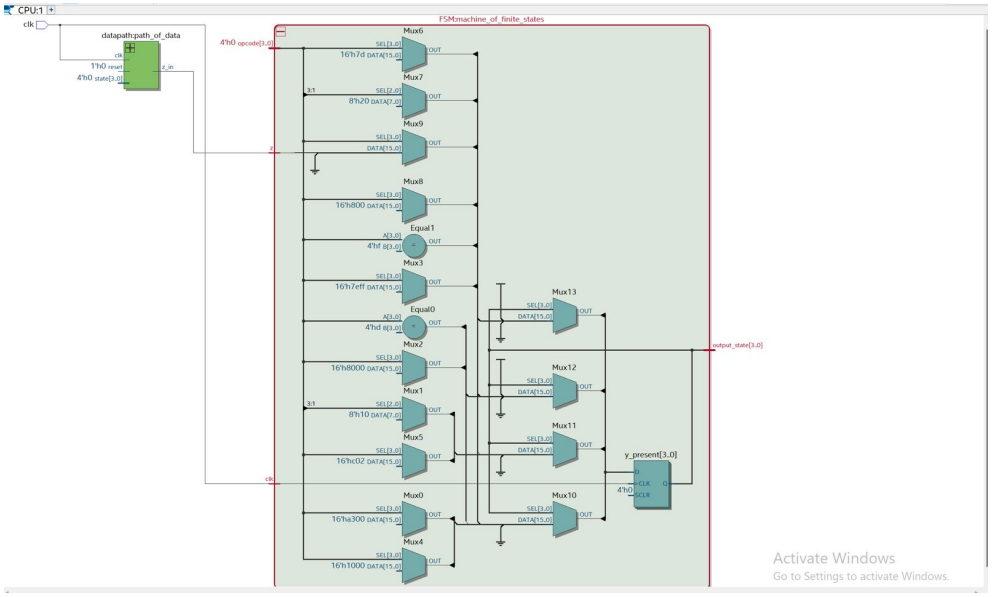


Figure 1: FSM_RTL

7 DATAPATH

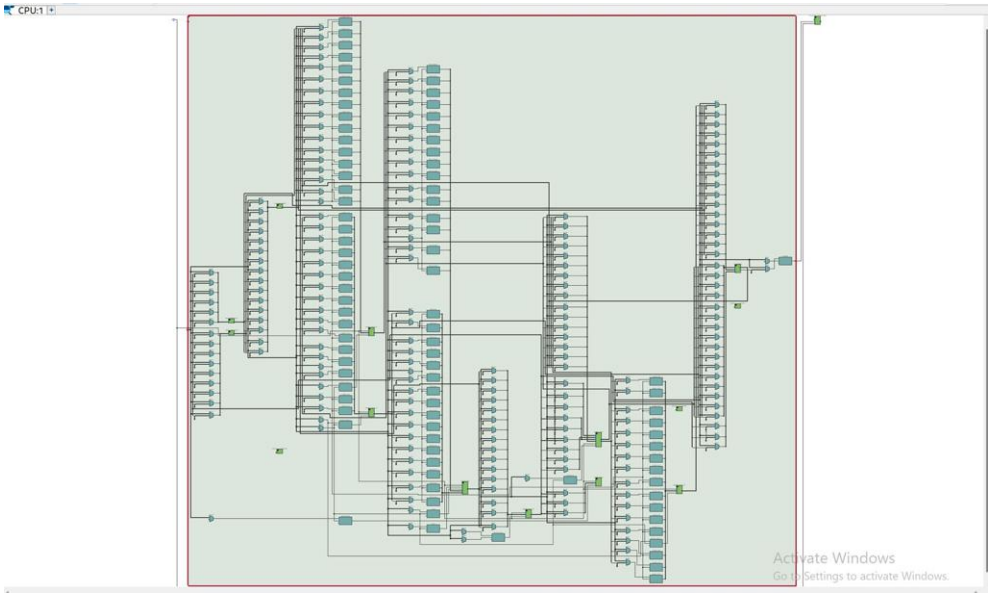


Figure 2: Data Path_NetList

8 RTL SIMULATION

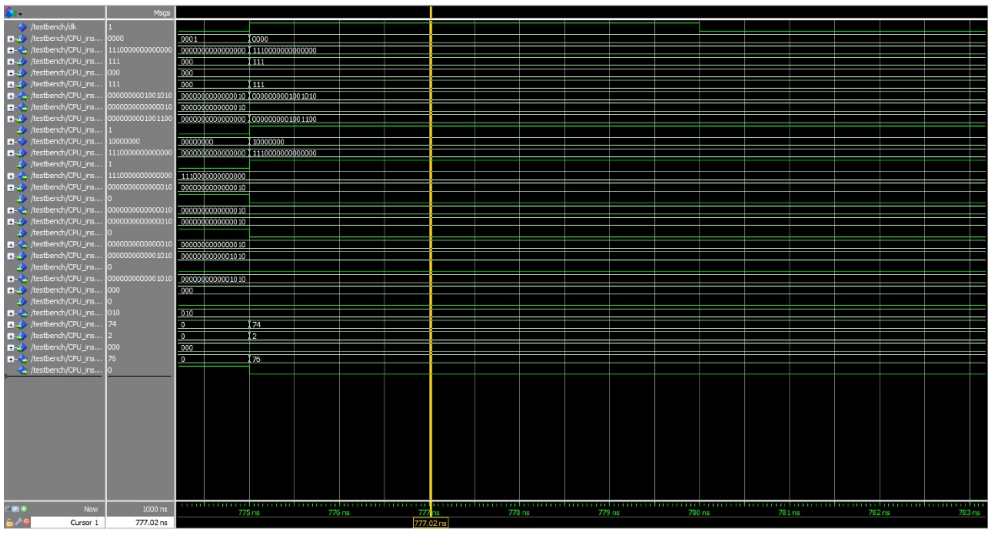


Figure 3: CPU_Testing

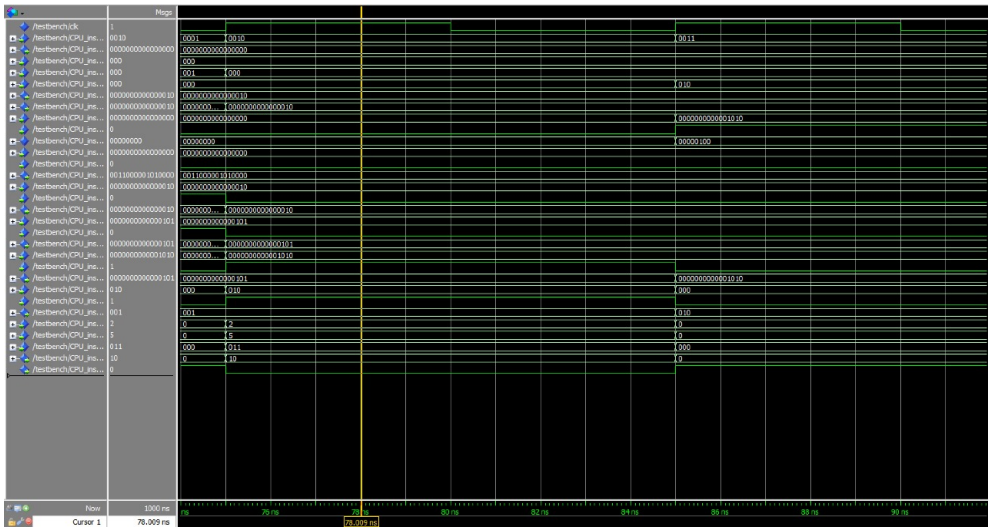


Figure 4: CPU_Testing

9 CONCLUSION

- Thus using an FSM of 14 states, we implemented 14 instructions each of which can be distinguished using the given op-codes and the Z bit.
- The implementation of the DataPath was achieved by utilizing the state and clock as inputs. This resulted in outputs that included the interconnections among various components and the status of control signals (such as memory read enable, memory write enable, Program Register write enable, etc.) relevant to that state.
- The entity CPU was designed including both the FSM and the DataPath using the clock as input.

10 MEMBER CONTRIBUTIONS

- **Aaditya Gupta (22B3941)**: Responsible for creating some of the Components and also helped in designing FSMs for the report.
- **Dattaraj Salunkhe (22B1296)**: Implemented the logic and testing of Datapath and helped in integration of the CPU.
- **Harsh Sanjay Roniyar (22B3942)**: Created the State Transition tables and integrated the individual state-flow diagrams into the FSM. Also, created the report and in the integration of the CPU.
- **Pranav Prakash (22B3945)**: Implemented the crucial memory blocks and also integrated the Components together in the CPU. Also, helped in designing the State Transition tables.