Design and Implementation of IITB-CPU

Course: EE224 - Digital Systems
Instructor: Prof. Virendra Singh
Department of Electrical Engineering, IIT Bombay

Yaswanth Ram Kumar	**Sri Charan Raj**	**Manidheep Reddy**	**Bhanu Prasanna**
23B1277	23B1229	23B1311	23B1291

1 Components

1.1 Program Counter

The Program Counter is a special-purpose register that holds the address of the next instruction to be fetched and executed. It is updated sequentially during instruction execution or modified by control instructions such as jumps or branches.

1.2 Register File

A register file is a small, fast storage unit consisting of multiple registers. It provides read and write access to operands used by the processor. It typically has multiple read ports and one or more write ports to support parallel access. For this IITB-CPU, we have made this RF with 8 registers.

1.3 Instruction Register

The Instruction Register holds the instruction currently being decoded and executed. It is loaded with the fetched instruction during the fetch phase of the instruction cycle and feeds control signals for further execution.

1.4 Concatenator

The Concatenator combines an 8-bit input with additional bits to form a 16-bit output. It operates based on a selection signal (concat_sel):

- `concat_sel=1`: `LLIOutput` is the 8-bit input extended with leading zeros (`00000000 && input`).
- `concat_sel=0`: `LHIOutput` is the 8-bit input extended with trailing zeros (`input && 00000000`).

1.5 Arithmetic Logic Unit

The ALU performs arithmetic and logical operations on two 16-bit inputs (`A` and `B`) based on a 3-bit `opcode`. The result is stored in `C` (16-bit output), and a `z_flag` indicates if the output is zero.

- `000`: ADD (`C = A + B`)
- `010`: SUB (`C = A - B`)
- `011`: MUL (`C = A * B` – lower 16 bits)
- `100`: AND (`C = A AND B`)
- `101`: ORA (`C = A OR B`)
- `110`: IMP (`C = NOT A OR B`)

1.6 SE6

The SE6 module extends a 6-bit input to a 16-bit output by replicating the sign bit (most significant bit of the 6-bit input) across the upper 10 bits. This preserves the signedness of the value when used in arithmetic operations.

1.7 SE9

The SE9 module extends a 9-bit input to a 16-bit output in a similar manner to SE6. It replicates the sign bit (MSB of the 9-bit input) across the upper 7 bits to maintain the value's signedness in extended form.

1.8 Left Shifter

The Left Shifter shifts the bits of a 16-bit input (`A`) to the left by exactly one bit. The vacated bit is filled with zero, and the output is a 16-bit result, which is twice of the input (`A`).

1.9 Finite State Machine (FSM)

The CPU employs a Finite State Machine (FSM) with exactly 16 states to manage the execution of instructions. Each state corresponds to a specific phase of the instruction cycle, such as:

- **Fetch**: Retrieve the instruction from memory.
- **Decode**: Interpret the instruction and prepare control signals.
- **Execute**: Perform the operation specified by the instruction.
- **Memory Access**: Read or write data to/from memory.
- **Write-Back**: Store the result in the register file.

The FSM ensures precise control and coordination of the CPU's operations, transitioning systematically between states.

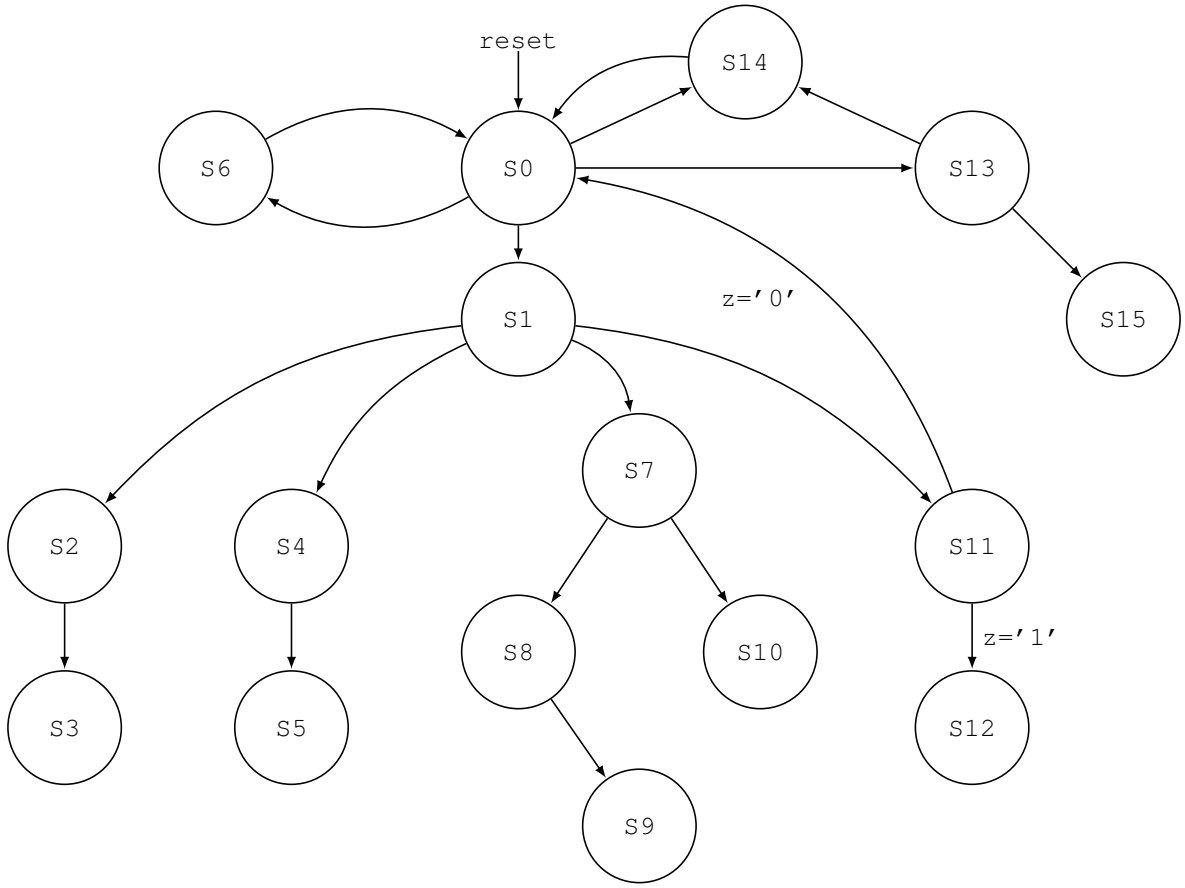
1.10 Datapath

The datapath comprises the functional units that process and transfer data within the CPU. Key components include:

- **Arithmetic Logic Unit (ALU)**: Executes arithmetic and logical operations.
- **Register File**: Stores intermediate and final results.
- **Program Counter (PC)**: Tracks the address of the next instruction.
- **Instruction Register (IR)**: Temporarily holds the current instruction.
- **Multiplexers and Shifters**: Facilitate data routing and manipulation.
- **Sign Extenders**: Extend smaller bit-width data to the required size.

The datapath works in tandem with the FSM to execute instructions efficiently.

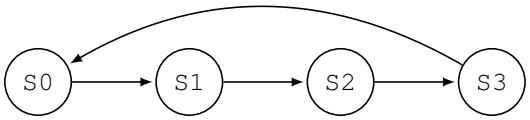
2 Finite State Machine



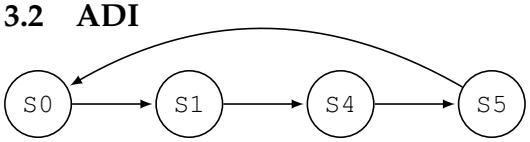
- Any FSM state without outgoing transitions is assumed to reset to `S0`.

3 Instruction State-Flow Diagrams

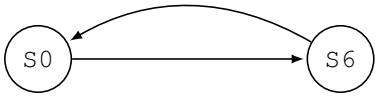
3.1 ADD/SUB/MUL/ORA/AND/IMP



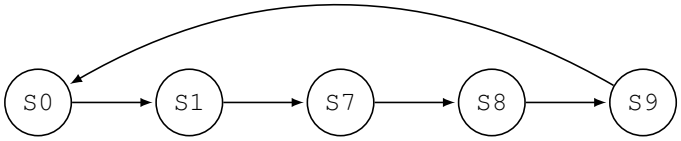
3.2 ADI



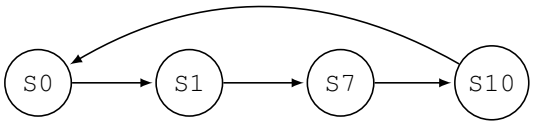
3.3 LHI/LLI



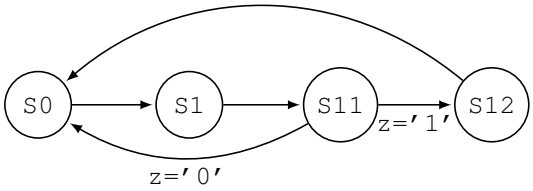
3.4 LW



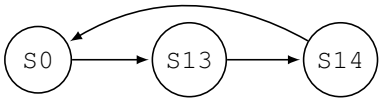
3.5 SW



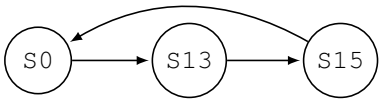
3.6 BEQ



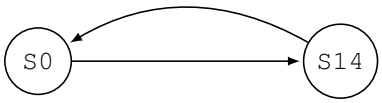
3.7 JAL



3.8 JLR



3.9 J



4 State Descriptions

4.1 S0

State Operation	Control Signal
PC ⇒ Mem_address	Mem_read
Mem_data ⇒ IR	IR_en
PC ⇒ RF_D3	RF_write
RF_A3="111"	

4.2 S1

State Operation	Control Signal
PC ⇒ ALU_A	PC_en
+2 ⇒ ALU_B	ALU_sel = "000"
ALU_C ⇒ PC	T1_write
IR(11-9) ⇒ RF_A1	T2_write
IR(8-6) ⇒ RF_A2	
RF_D1 ⇒ T1	
RF_D2 ⇒ T2	

4.3 S2

State Operation	Control Signal
T1 ⇒ ALU_A	T3_write
T2 ⇒ ALU_B	ALU_sel = IR(14-12)
ALU_C ⇒ T3	

4.4 S3

State Operation	Control Signal
IR(5-3) ⇒ RF_A3	RF_write
T3 ⇒ RF_D3	

4.5 S4

State Operation	Control Signal
T1 ⇒ ALU_A	T3_write
IR(5-0) ⇒ SE6 ⇒ ALU_B	ALU_sel = "000"
ALU_C ⇒ T3	

4.6 S5

State Operation	Control Signal
IR(8-6) ⇒ RF_A3	RF_write
T3 ⇒ RF_D3	

4.7 S6

State Operation	Control Signal
IR(12) ⇒ concat_sel	RF_write
IR(11-9) ⇒ RF_A3	PC_en
IR(7-0) ⇒ concat_in	ALU_sel = "000"
concat_out ⇒ RF_D3	
PC ⇒ ALU_A	
+2 ⇒ ALU_B	
ALU_C ⇒ PC_in	

4.8 S7

State Operation	Control Signal
T2 ⇒ ALU_A	T3_write
IR(5-0) ⇒ SE6 ⇒ ALU_B	ALU_sel = "000"
ALU_C ⇒ T3	

4.9 S8

State Operation	Control Signal
T3 ⇒ Mem_address	Mem_read
Mem_data ⇒ T3	T3_write

4.10 S9

State Operation	Control Signal
IR(11-9) ⇒ RF_A3	RF_write
T3 ⇒ RF_D3	

4.11 S10

State Operation	Control Signal
T3 ⇒ Mem_address	Mem_write
T1 ⇒ Mem_data	

4.12 S11

State Operation	Control Signal
T1 ⇒ ALU_A	ALU_sel = "010"
T2 ⇒ ALU_B	
Z_flag ⇒ Z_sel	

4.13 S12

State Operation	Control Signal
RF_D1 ⇒ ALU_A	PC_en
IR(5-0) ⇒ SE6 ⇒ left_1s ⇒ ALU_B	ALU_sel = "000"
ALU_C ⇒ PC_in	RF_A1="111"

4.14 S13

State Operation	Control Signal
IR(11-9) ⇒ RF_A3	RF_write
PC ⇒ RF_D3	

4.15 S14

State Operation	Control Signal
RF_D1 ⇒ ALU_A	PC_en
IR(8-0) ⇒ SE9 ⇒ left_1s ⇒ ALU_B	RF_A1="111"
ALU_C ⇒ PC_in	

4.16 S15

State Operation	Control Signal
IR(8-6) ⇒ RF_A2	PC_en
RF_D2 ⇒ PC_in	

5 Simulation Plots

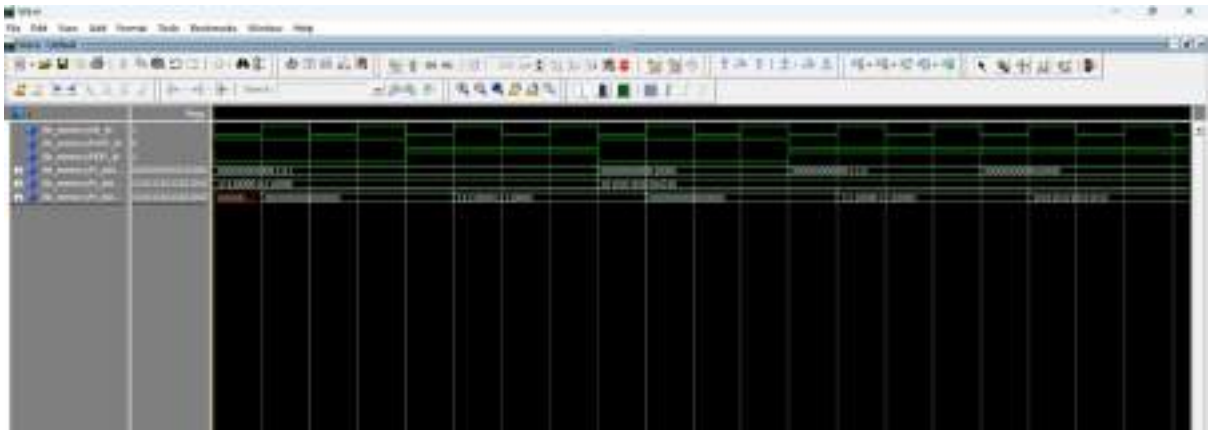


Figure 1: Memory

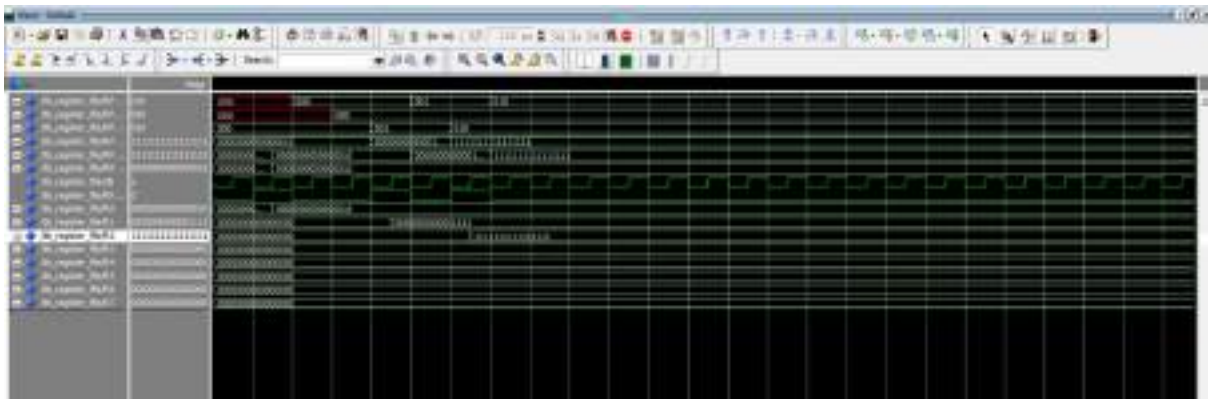


Figure 2: Register File

Figure 3: FSM

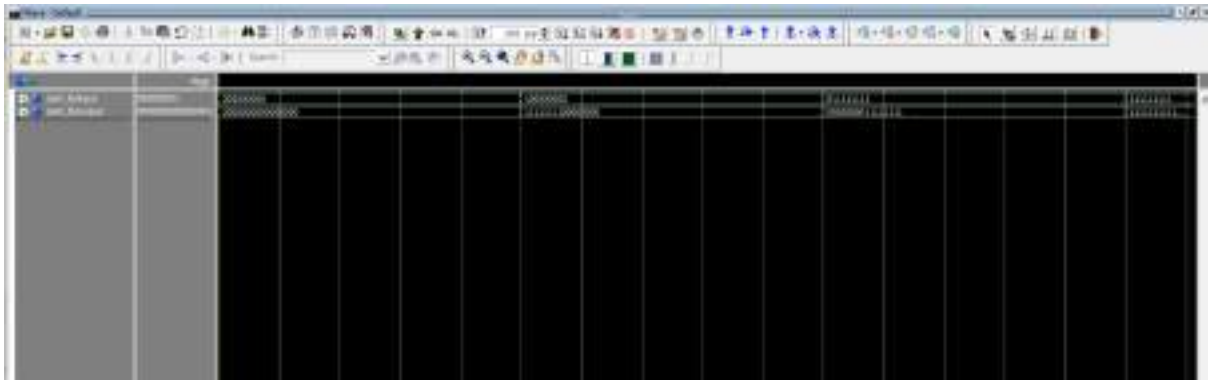


Figure 7: SE9

Figure 4: ALU

Figure 5: Left_1s

Figure 6: SE6

Figure 8: Concatenator

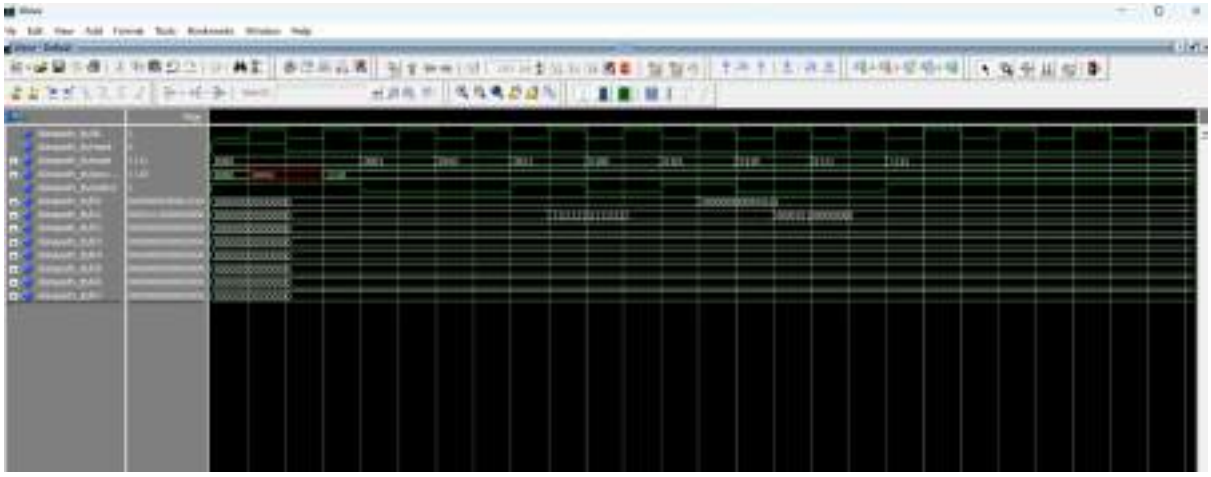


Figure 9: Datapath

6 Machine Constraints

The IITB-CPU we have designed features a memory unit with 25 registers, each being 16 bits wide, and a Register File comprising 8 registers. Among these, the 8th register (R7) is dedicated to storing the instruction PC value.

To ensure proper operation, the following constraints must be observed:

- Instructions must not attempt to access or write data to registers beyond the available storage capacity.
- Registers R0 to R6 are the only registers allowed for general-purpose operations; R7 is strictly reserved for the PC and must not be overwritten.
- Jump instructions and PC updates must ensure that the PC value remains within the valid range (0 to 48), preventing it from exceeding the bounds of the accessible instruction memory.

7 Circuit Diagram and FSM Netlist

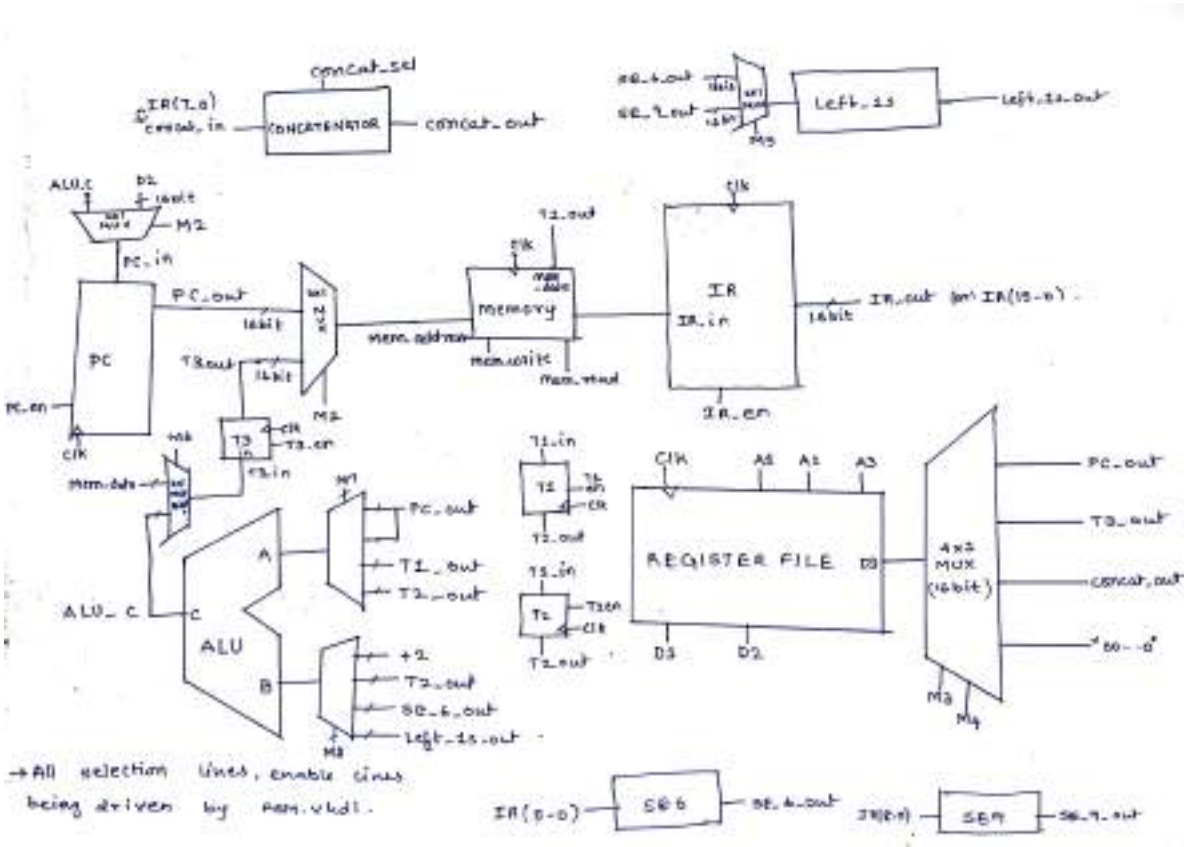


Figure 10: Circuit Diagram

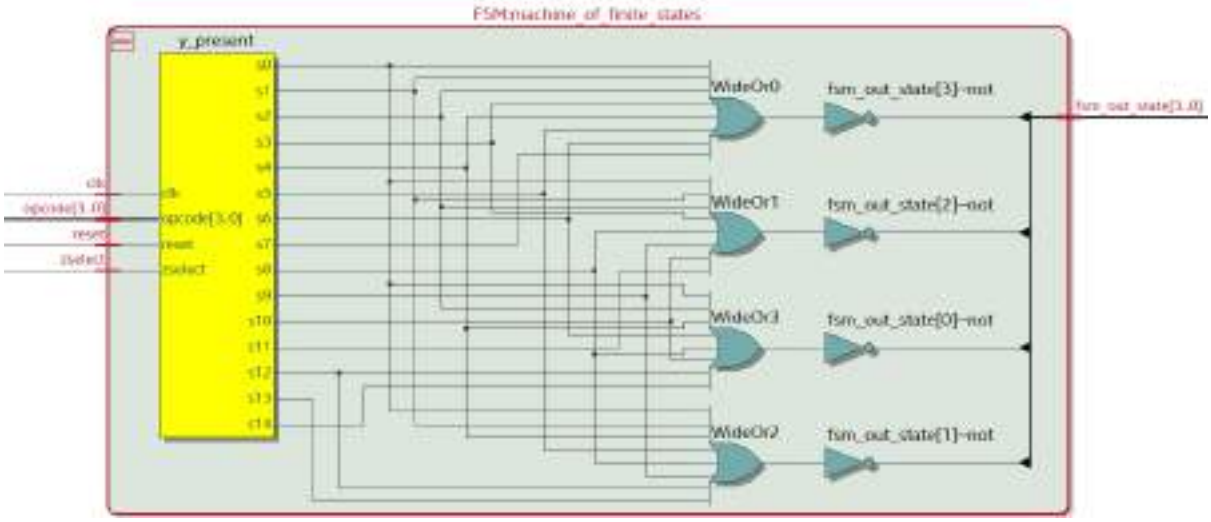


Figure 11: FSM Netlist