29/5/2017 CNG 232 LAB7 (Final)   
Objective:

In this lab we have to build a FSM to control a simple Calculator ALU, we built a datapath to connect our FSM to the ALU and display the output using a 7 segment decoder

7.4.1 MOP Table

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instruction  Mnemonic | S0 | S1 | S2 | S3 | | S4 | | |
| NOP (0000) | out\_state <= "0000" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | out\_state <="0001" ;  Input\_select <= "100" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | out\_state <= "0010" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | out\_state <= "0011" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | | out\_state <= "0100" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | | |
| CLR (0001) | out\_state <= "0000" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | out\_state <= "0001" ;  Input\_select <= "100" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '1' ; | out\_state <= "0010" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | out\_state <= "0011" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | | out\_state <= "0100" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | | |
| LOADA (0010) | out\_state <= "0000" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | out\_state <= "0001" ;  Input\_select <= "110" ;  Load\_A <= '1' ; Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | out\_state <= "0010" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | out\_state <= "0011" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | | out\_state <= "0100" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | | |
| LOADB (0011) | out\_state <= "0000" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | out\_state <= "0001" ;  Input\_select <= "110" ;  Load\_A <= '0' ;  Load\_B <= '1' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | out\_state <= "0010" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | out\_state <= "0011" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | | out\_state <= "0100" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | | |
| BCOPYA(0100) | out\_state <= "0000" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | out\_state<="0001";  Input\_select<="100”;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop<="000" ;  Reset <= '0' ; | out\_state <= "0010" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  load\_B <= '0' ;  Cin <= '0' ;  ALUop <= “000" ;  Reset <= '0' ; | out\_state <= "0011" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ; Cin <= '0' ;  ALUop <="000" ;  Reset <= '0' ; | | out\_state <= "0100" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | | |
| ADD2 (0101) | out\_state <= "0000";  Input\_Select <= "000";  Load\_A <= '0';  Load\_B <= '0';  Cin <= '0';  ALUop <= "000";  Reset <= '0'; | out\_state <= "0001" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | out\_state <= "0010" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | out\_state <= "0011" ; Input\_select <= "100" ;  Load\_A <= '0' ;  Load\_B <= '1' ;  Cin <= '0' ;  ALUop <= "001" ;  Reset <= '0' ; | | out\_state <= "0100";  Input\_Select <= "000";  Load\_A <= '0';  Load\_B <= '0';  Cin <= '0';  ALUop <= "000";  Reset <= '0'; | | |
| SUB (101) | out\_state <= "0000";  Input\_Select <= "000";  Load\_A <= '0';  Load\_B <= '0';  Cin <= '0';  ALUop <= "000";  Reset <= '0'; | out\_state <= "0001";  Input\_Select <= "000";  Load\_A <= '0';  Load\_B <= '0';  Cin <= '0';  ALUop <= "000";  Reset <= '0'; | out\_state <= "0010";  Input\_Select <= "000";  Load\_A <= '0';  Load\_B <= '0';  Cin <= '0';  ALUop <= "000";  Reset <= '0'; | out\_state <= "0011";  Input\_Select <= "101";  Load\_A <= '0';  Load\_B <= '1';  Cin <= '1';  ALUop <= "001";  Reset <= '0'; | | out\_state <= "0100" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | | |
| ADD21 (0110) | out\_state <= "0000" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | out\_state <= "0001" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | out\_state <= "0010" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | out\_state <= "0011" ;  Input\_select <= "100" ;  Load\_A <= '0' ;  Load\_B <= '1' ;  Cin <= '1' ;  ALUop <= "001" ;  Reset <= '0' ; | | out\_state <= "0100" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | | |
| ADD3 (0111) | out\_state <= "0000" ; Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | out\_state <= "0001" ;  Input\_select <= "100" ;  Load\_A <= '0' ;  Load\_B <= '1' ;  Cin <= '0' ;  ALUop <= "001" ;  Reset <= '0' ; | out\_state <= "0010" ;  Input\_select <= "100" ;  Load\_A <= '1' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | | out\_state <= "0011" ; Input\_select <= "110" ;  Load\_A <= '0' ;  Load\_B <= '1' ;  Cin <= '0' ;  ALUop <= "001" ;  Reset <= '0' ; | | out\_state <= "0100" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; |
| SUB (1000) | out\_state <= "0000" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | out\_state <= "0001" ;  Input\_select <= "101" ;  Load\_A <= '0' ;  Load\_B <= '1' ;  Cin <= '1' ;  ALUop <= "001";  Reset <= '0' ; | out\_state <= "0010" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000";  Reset <= '0' ; | | out\_state <= "0011" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | | out\_state <= "0100" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; |
| BINVB (1001) | out\_state <= "0000" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | out\_state <= "0001" ;  Input\_select <= "101" ;  Load\_A <= '0' ;  Load\_B <= '1' ;  Cin <= '0' ;  ALUop <= "000";  Reset <= '0' ; | out\_state <= "0010" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000";  Reset <= '0' ; | | out\_state <= "0011" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | | out\_state <= "0100" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; |
| BAND (1010) | out\_state <= "0000" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | out\_state <= "0001" ;  Input\_select <= "100" ;  Load\_A <= '0' ;  Load\_B <= '1' ;  Cin <= '0' ;  ALUop <= "101" ;  Reset <= '0' ; | out\_state <= "0010" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | | out\_state <= "0011" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | | out\_state <= "0100" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; |
| BOR (1011) | out\_state <= "0000" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | out\_state <= "0001" ;  Input\_select <= "100" ;  Load\_A <= '0' ;  Load\_B <= '1' ;  Cin <= '0' ;  ALUop <= "100" ;  Reset <= '0' ; | out\_state <= "0010" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | | out\_state <= "0011" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | | out\_state <= "0100" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; |
| BXOR (1100) | out\_state <= "0000" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | out\_state <= "0001" ;  Input\_select <= "100" ;  Load\_A <= '0' ;  Load\_B <= '1' ;  Cin <= '0' ;  ALUop <= "110";  Reset <= '0' ; | out\_state <= "0010" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000";  Reset <= '0' ; | | out\_state <= "0011" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | | out\_state <= "0100" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; |
| BLSRA (1101) | out\_state <= "0000" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | out\_state <= "0001" ;  Input\_select <= "010" ;  Load\_A <= '0' ;  Load\_B <= '1' ;  Cin <= '0' ;  ALUop <= "000";  Reset <= '0' ; | out\_state <= "0010" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000";  Reset <= '0' ; | | out\_state <= "0011" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | | out\_state <= "0100" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; |
| BLSLA (1110) | out\_state <= "0000" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | out\_state <= "0001" ;  Input\_select <= "001" ;  Load\_A <= '0' ;  Load\_B <= '1' ;  Cin <= '0' ;  ALUop <= "000";  Reset <= '0' ; | out\_state <= "0010" ; Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000";  Reset <= '0' ; | | out\_state <= "0011" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | | out\_state <= "0100" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; |
| BASRA (1111) | out\_state <= "0000" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | out\_state <= "0001" ;  Input\_select <= "011" ;  Load\_A <= '0' ;  Load\_B <= '1' ;  Cin <= '0' ;  ALUop <= "000";  Reset <= '0' ; | out\_state <= "0010" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000";  Reset <= '0' ; | | out\_state <= "0011" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; | | out\_state <= "0100" ;  Input\_select <= "000" ;  Load\_A <= '0' ;  Load\_B <= '0' ;  Cin <= '0' ;  ALUop <= "000" ;  Reset <= '0' ; |

7.4.2 FSM CODE

LIBRARY ieee;

use IEEE.STD\_LOGIC\_1164.ALL;

ENTITY FSM IS

PORT (

Load\_A: OUT STD\_LOGIC;

Load\_B: OUT STD\_LOGIC;

Cin: OUT STD\_LOGIC;

ALUop: OUT STD\_LOGIC\_VECTOR (2 DOWNTO 0);

Input\_Select: OUT STD\_LOGIC\_VECTOR (2 DOWNTO 0);

Reset: OUT STD\_LOGIC;

out\_state: OUT STD\_LOGIC\_VECTOR (3 DOWNTO 0);

Clock: IN STD\_LOGIC;

RUN : IN STD\_LOGIC;

MOP: IN STD\_LOGIC\_VECTOR (2 DOWNTO 0)

);

END FSM;

ARCHITECTURE Behavior OF FSM IS “In here we specify how many states does our FSM have”

SIGNAL y\_present, y\_next : STD\_LOGIC\_VECTOR(4 DOWNTO 0);

CONSTANT S0 : STD\_LOGIC\_VECTOR(4 DOWNTO 0) := "00001";

CONSTANT S1 : STD\_LOGIC\_VECTOR(4 DOWNTO 0) := "00010";

CONSTANT S2 : STD\_LOGIC\_VECTOR(4 DOWNTO 0) := "00100";

CONSTANT S3 : STD\_LOGIC\_VECTOR(4 DOWNTO 0) := "01000";

CONSTANT S4 : STD\_LOGIC\_VECTOR(4 DOWNTO 0) := "10000";

BEGIN

PROCESS (RUN, y\_present)”In here this will allow us to cycle through our FSM states”

BEGIN

CASE y\_present IS

WHEN S0 =>

IF RUN='0' THEN y\_next <= S0;

ELSE y\_next <= S1;

END IF;

WHEN S1 =>

IF RUN='0' THEN y\_next <= S0;

ELSE y\_next <= S2;

END IF;

WHEN S2 =>

IF RUN='0' THEN y\_next <= S0;

ELSE y\_next <= S3;

END IF;

WHEN S3 =>

IF RUN='0' THEN y\_next <= S0;

ELSE y\_next <= S4;

END IF;

WHEN S4 =>

IF RUN='0' THEN y\_next <= S0;

ELSE y\_next <= S4;

END IF;

WHEN OTHERS =>

y\_next <= S0;

END CASE;

END PROCESS;

PROCESS (Clock, RUN) “In here we define what happens at each state i.e what happens in every MOP”

BEGIN

IF RUN='0' THEN

y\_present <= S0;

ELSIF (Clock'EVENT AND Clock='1') THEN

y\_present <= y\_next;

END IF;

END PROCESS;

PROCESS (y\_present, MOP)

BEGIN

Input\_select <= "000" ;

Load\_A <= '0' ;

Load\_B <= '0' ;

Cin <= '0' ;

ALUop <= "000" ;

Reset <= '0' ;

IF MOP = "000" then

IF y\_present = S0 then

out\_state <= "0000" ;

ELSIF y\_present = S1 then

out\_state <= "0001" ;

ELSIF y\_present = S2 then

out\_state <= "0010" ;

ELSIF y\_present = S3 then

out\_state <= "0011" ;

ELSIF y\_present = S4 then

out\_state <= "0100" ;

END IF;

ELSIF MOP = "001" then

IF y\_present = S0 then

out\_state <= "0000" ;

ELSIF y\_present = S1 then

out\_state <= "0001" ;

ELSIF y\_present = S2 then

out\_state <= "0010" ;

ELSIF y\_present = S3 then

out\_state <= "0011" ;

Reset <= '1' ;

ELSIF y\_present = S4 then

out\_state <= "0100" ;

END IF;

ELSIF MOP = "010" then

IF y\_present = S0 then

out\_state <= "0000" ;

ELSIF y\_present = S1 then

out\_state <= "0001" ;

ELSIF y\_present = S2 then

out\_state <= "0010" ;

ELSIF y\_present = S3 then

out\_state <= "0011" ;

Input\_select <= "110" ;

Load\_A <= '1' ;

ELSIF y\_present = S4 then

out\_state <= "0100" ;

END IF;

ELSIF MOP = "011" then

IF y\_present = S0 then

out\_state <= "0000" ;

ELSIF y\_present = S1 then

out\_state <= "0001" ;

ELSIF y\_present = S2 then

out\_state <= "0010" ;

ELSIF y\_present = S3 then

out\_state <= "0011" ;

Input\_select <= "110" ;

Load\_B <= '1' ;

ELSIF y\_present = S4 then

out\_state <= "0100" ;

END IF;

ELSIF MOP = "100" then

IF y\_present = S0 then

out\_state <= "0000" ;

ELSIF y\_present = S1 then

out\_state <= "0001" ;

ELSIF y\_present = S2 then

out\_state <= "0010" ;

ELSIF y\_present = S3 then

out\_state <= "0011" ;

Input\_select <= "100" ;

Load\_B <= '1';

Cin <= '1';

ALUop <= "001";

ELSIF y\_present = S4 then

out\_state <= "0100" ;

END IF;

ELSIF MOP = "101" then

IF y\_present = S0 then

out\_state <= "0000" ;

ELSIF y\_present = S1 then

out\_state <= "0001" ;

ELSIF y\_present = S2 then

out\_state <= "0010" ;

ELSIF y\_present = S3 then

out\_state <= "0011" ;

Input\_select <= "101" ;

Load\_B <= '1';

Cin <= '1';

ALUop <= "001";

ELSIF y\_present = S4 then

out\_state <= "0100" ;

END IF;

ELSIF MOP = "110" then

IF y\_present = S0 then

out\_state <= "0000" ;

ELSIF y\_present = S1 then

out\_state <= "0001" ;

ELSIF y\_present = S2 then

out\_state <= "0010" ;

ELSIF y\_present = S3 then

out\_state <= "0011" ;

Input\_select <= "101" ;

Load\_B <= '1';

ELSIF y\_present = S4 then

out\_state <= "0100" ;

END IF;

ELSIF MOP = "111" then

IF y\_present = S0 then

out\_state <= "0000" ;

ELSIF y\_present = S1 then

out\_state <= "0001" ;

ELSIF y\_present = S2 then

out\_state <= "0010" ;

ELSIF y\_present = S3 then

out\_state <= "0011" ;

Input\_select <= "100" ;

Load\_B <= '1';

ALUop <= "101";

ELSIF y\_present = S4 then

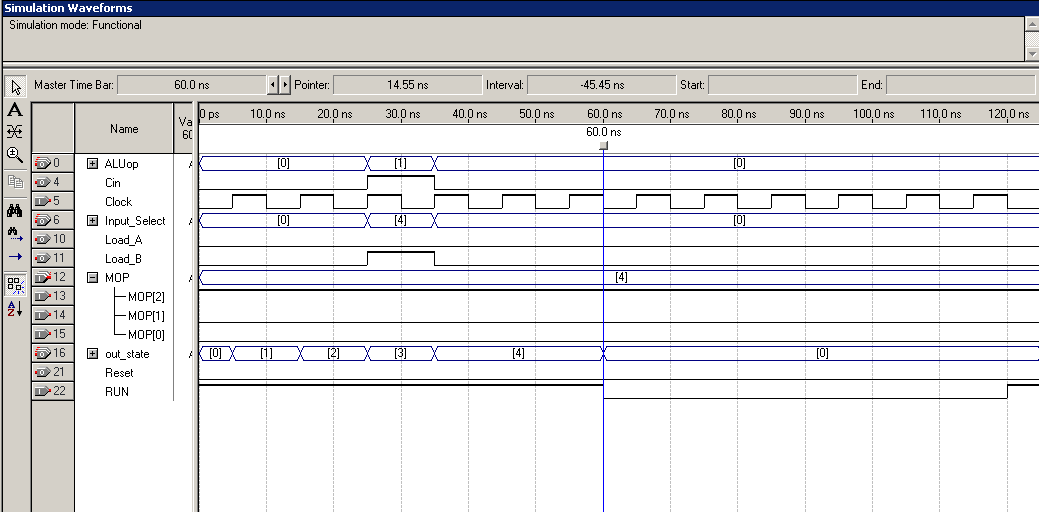
out\_state <= "0100" ;

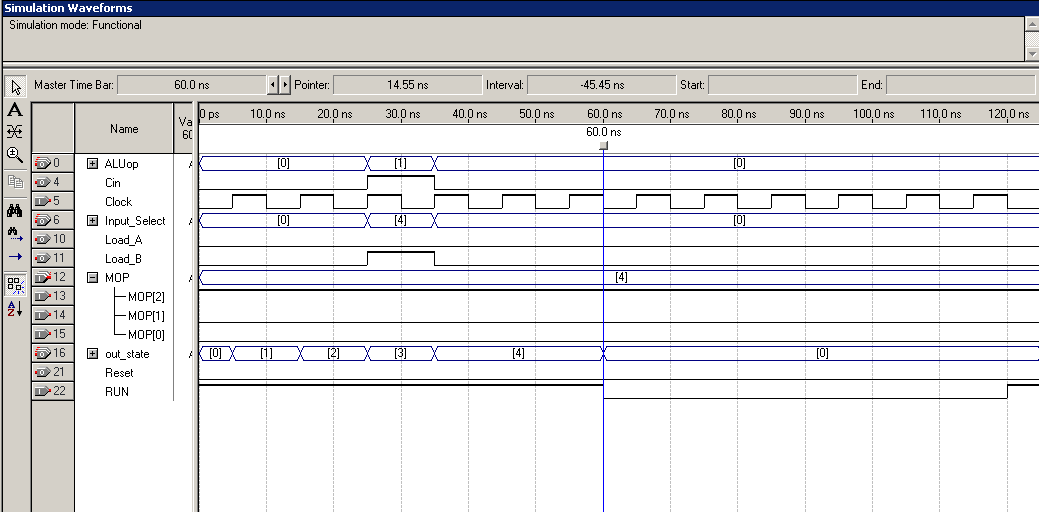
END IF;

END IF;

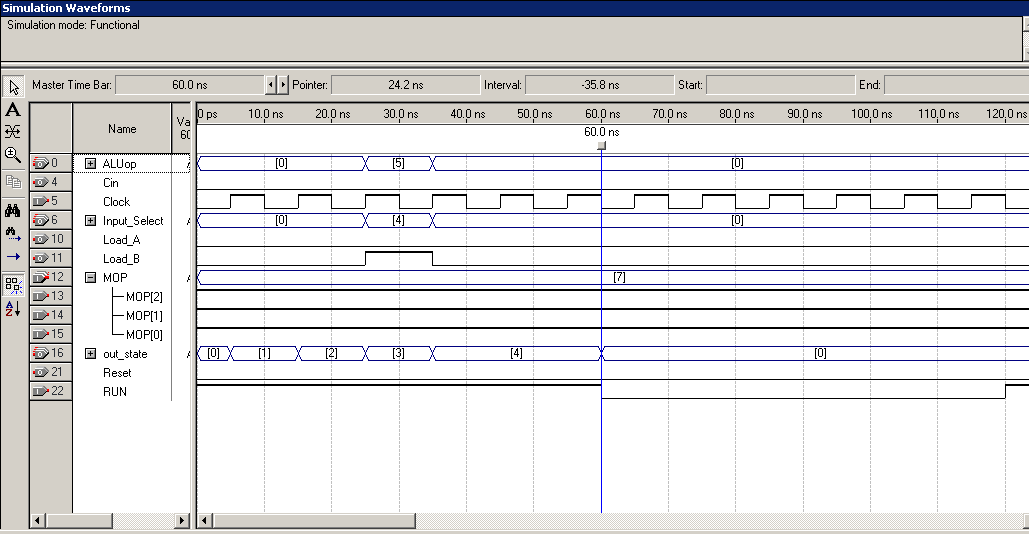
END PROCESS;

END Behavior;



ADD Simulation  
  


BAND Simulation

  
Same as above with BAND, changes are MOP now is (7) “111” to make BAND operation and ALU\_OP is (5) “101”.

7.4.3 Datapath Design

ALU VHDL

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_SIGNED.ALL;

entity alu is

Port( Operation: in STD\_LOGIC\_VECTOR (2 downto 0);

X: in STD\_LOGIC\_VECTOR (3 downto 0);

Y: in STD\_LOGIC\_VECTOR (3 downto 0);

Cin: in STD\_LOGIC;

Cout: out STD\_LOGIC;

Output: out STD\_LOGIC\_VECTOR (3 downto 0) );

end alu;

architecture Behavioral of alu is

signal Outputt : STD\_LOGIC\_VECTOR (4 Downto 0) ;

begin

process(Operation, X, Y, Cin)

begin

case Operation is

WHEN "000" => Outputt <= ('0' & X);

Output <= Outputt(3 Downto 0);

Cout <= Outputt(4);

WHEN "001" => Outputt <= ('0' & X) + Y + (Cin);

Output <= Outputt(3 Downto 0);

Cout <= Outputt(4);

WHEN "010" => Outputt <= ('0' & X)- Y - (NOT Cin);

Output <= Outputt(3 Downto 0);

Cout <= Outputt(4);

WHEN "011" => Outputt <= Y- ('0' & X) - (NOT Cin);

Output <= Outputt(3 Downto 0);

Cout <= Outputt(4);

WHEN "100" => Outputt <= ('0' & X) OR Y;

Output <= Outputt(3 Downto 0);

Cout <= Outputt(4);

WHEN "101" => Outputt <= ('0' & X) AND Y;

Output <= Outputt(3 Downto 0);

Cout <= Outputt(4);

WHEN "110" => Outputt <= ('0' & X) XOR Y;

Output <= Outputt(3 Downto 0);

Cout <= Outputt(4);

WHEN OTHERS => Outputt <= ('0' & X) XNOR Y;

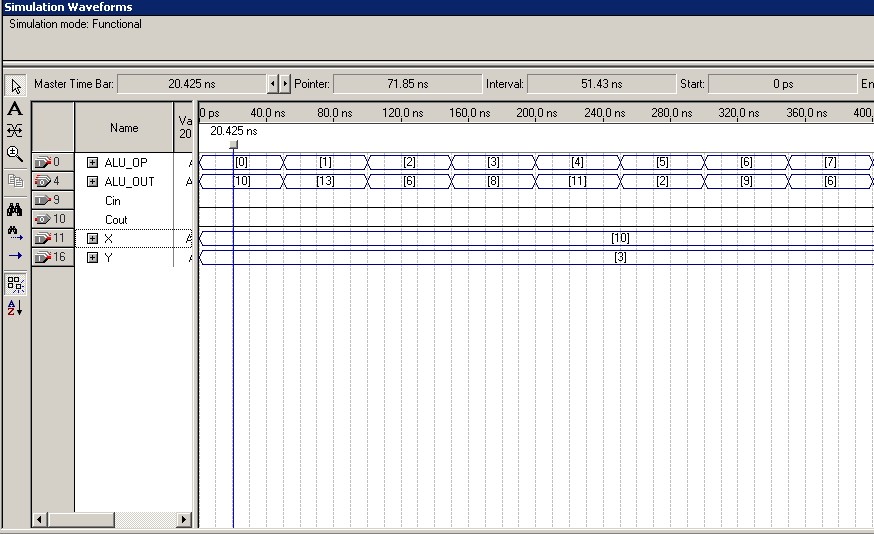
Output <= Outputt(3 Downto 0);

Cout <= Outputt(4);

end case;

end process;

end Behavioral;

ALU Simulation  


As requested the ALU is going through all operations for numbers 10 and 3.

**Register VHDL**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity registerr is

Port(

Data: in STD\_LOGIC\_VECTOR (3 downto 0);

load: in STD\_LOGIC;

Reset: in STD\_LOGIC;

Clk: in STD\_LOGIC;

Output: out STD\_LOGIC\_VECTOR (3 downto 0));

end registerr;

architecture Behavioral of registerr is

begin

process (Clk, load, Reset)

begin

if Clk'event and Clk ='1' then

if Reset = '1' then

Output <= "0000";

elsif Reset = '0' then

if load = '1' then

Output <= Data;

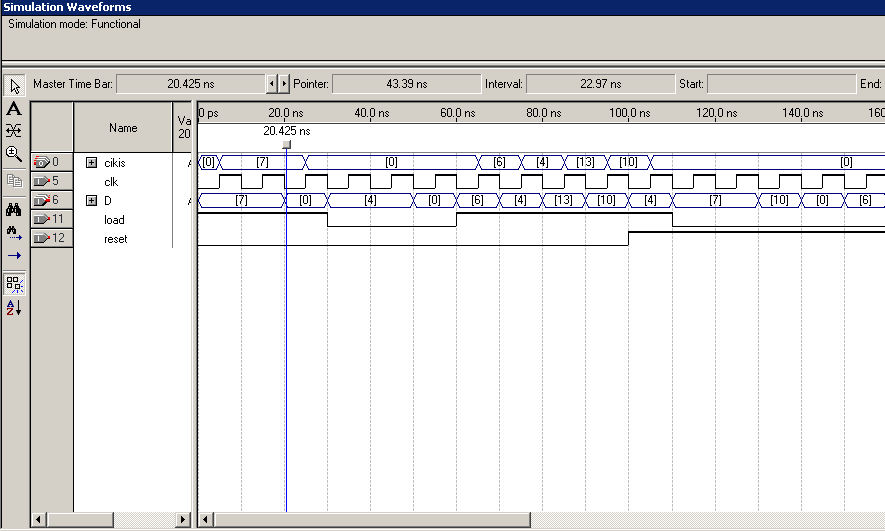
end if;

end if;

end if;

end process;

end Behavioral;

**Register Simulation**  


**8x1MUX VHDL**  
  
library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity mux8to1 is

Port ( “in here defining 8 inputs and 3 select lines”

s : in STD\_LOGIC\_VECTOR (2 DOWNTO 0);

w0: in STD\_LOGIC\_VECTOR (3 DOWNTO 0);

w1: in STD\_LOGIC\_VECTOR (3 DOWNTO 0);

w2: in STD\_LOGIC\_VECTOR (3 DOWNTO 0);

w3: in STD\_LOGIC\_VECTOR (3 DOWNTO 0);

w4: in STD\_LOGIC\_VECTOR (3 DOWNTO 0);

w5: in STD\_LOGIC\_VECTOR (3 DOWNTO 0);

w6: in STD\_LOGIC\_VECTOR (3 DOWNTO 0);

w7: in STD\_LOGIC\_VECTOR (3 DOWNTO 0);

f : out STD\_LOGIC\_VECTOR (3 DOWNTO 0));

end mux8to1 ;

architecture Behavior of mux8to1 is “using when command to act as the select lines behavior”

begin

WITH s SELECT

f <= w0 WHEN "000",

w1 WHEN "001",

w2 WHEN "010",

w3 WHEN "011",

w4 WHEN "100",

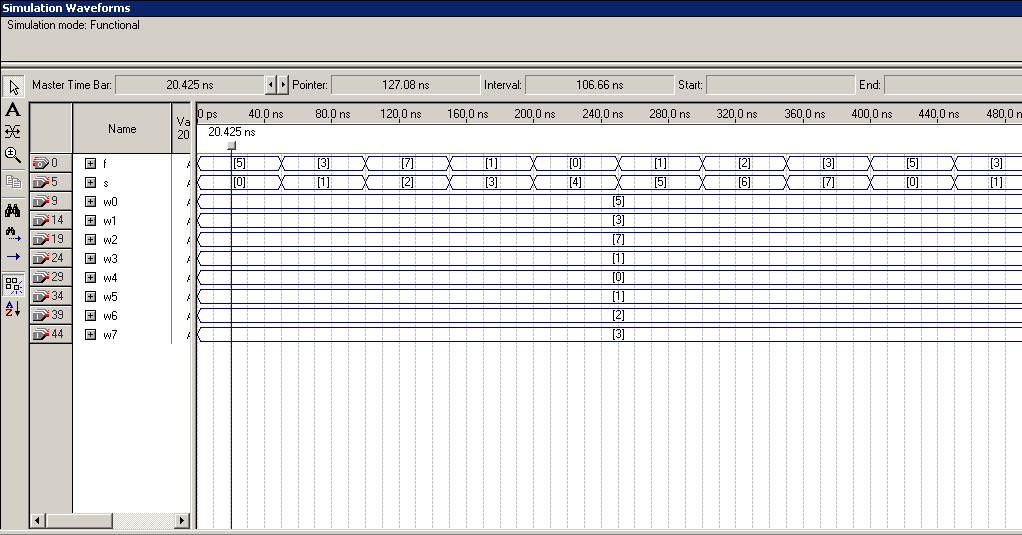
w5 WHEN "101",

w6 WHEN "110",

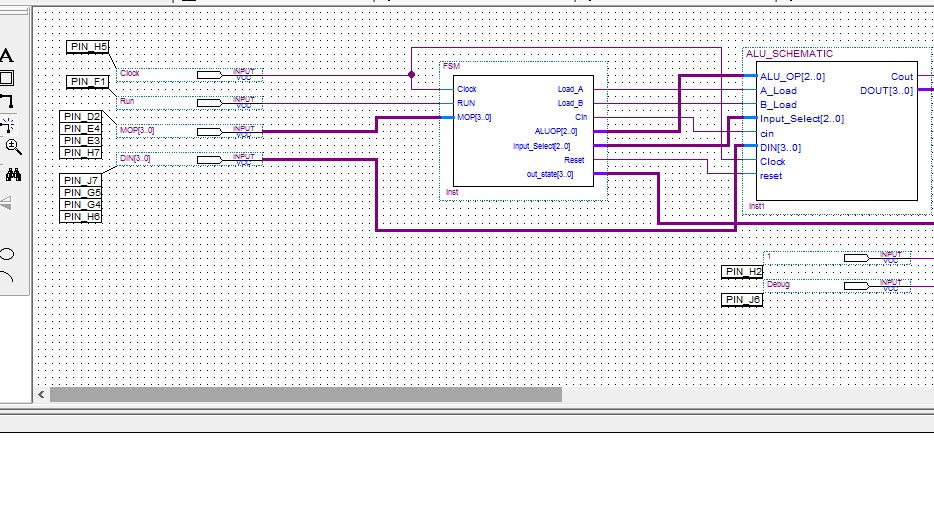
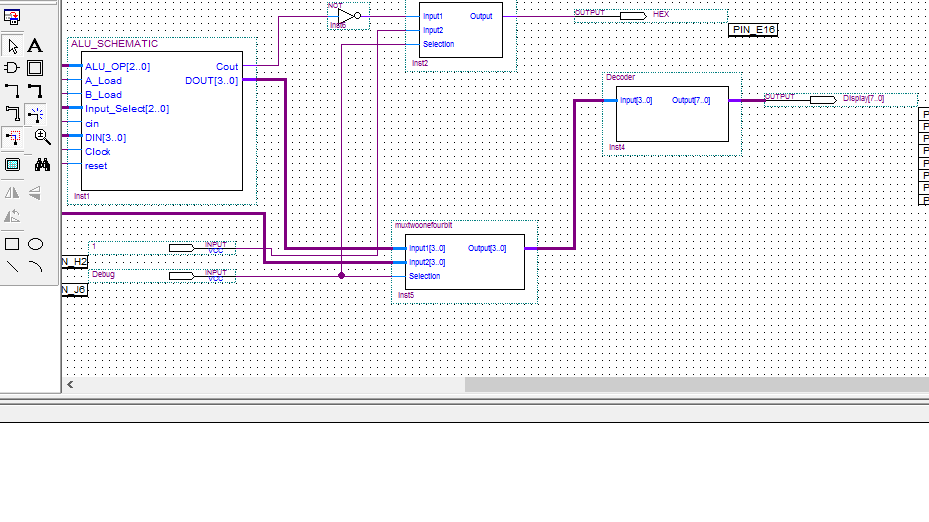
w7 WHEN "111";

end Behavior;

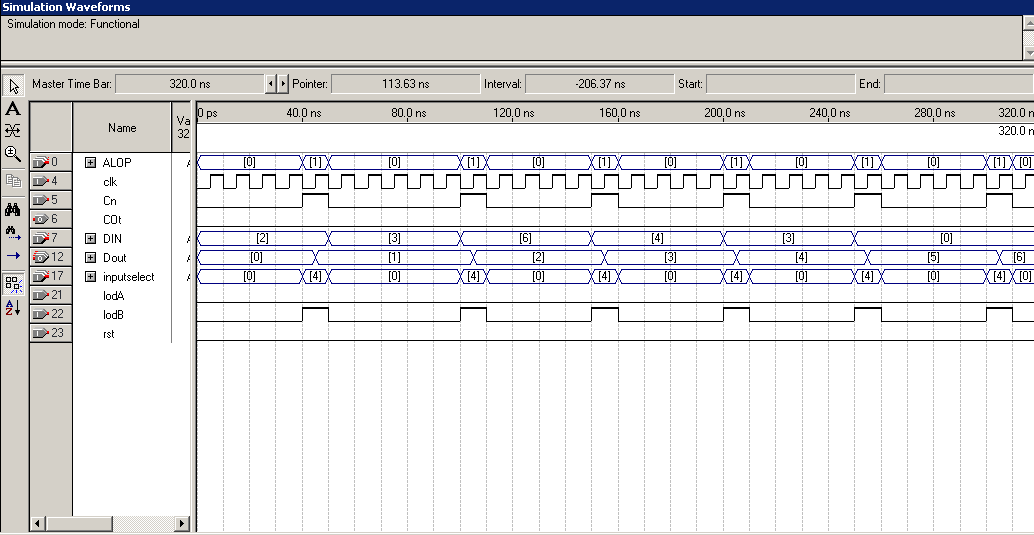
**8x1MUX Simulation**



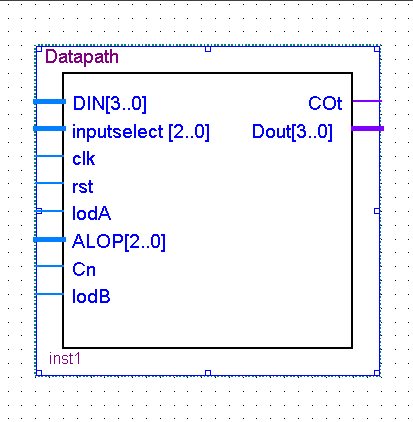
**Datapath Schematic**

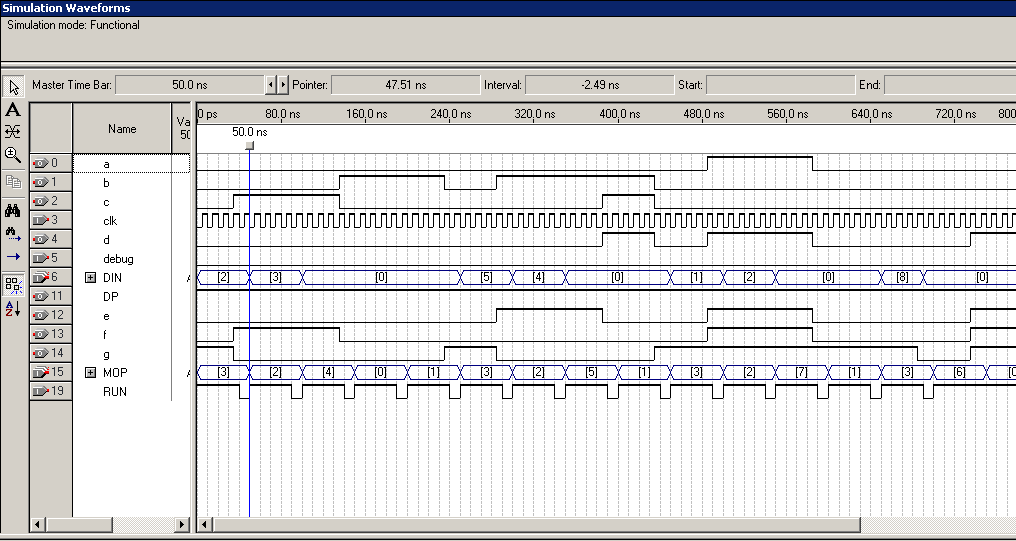
**Datapath Simulation**



**Datapath Block**



7.4.4 Calculator Simulation



**Conclusion:**

This lab is showing how to connect the knowledge of datapaths and FSM (finite state machines) also including combinational logic (MUX, decoder). So we can design any operation (shift,..) rather than being only a calculator.