VHDL Design and Modeling of Combination Circuits

Introduction

The purpose of this lab is to develop VHDL models for

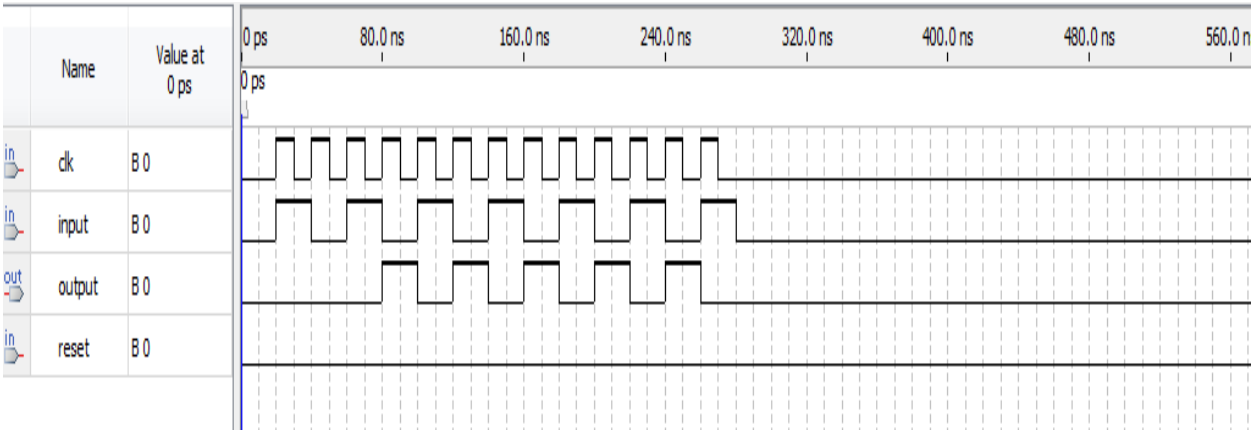
Experiment and Results

Part 1: 0101 State Machine

```
library ieee;
use ieee.std_logic_1164.all ;
entity state_machine is
port(reset,clk,input:in bit;
                                        output: out bit);
end state_machine;

architecture state_machine of state_machine is
type Statetype is (S0,S1,S2,S3);
signal state, next_state: Statetype;
begin
state_comb: process (state,input) begin
case state is
        when S0 =>
                if input = '0' then
                next_state <= S1;
                output <= '0';

                else
                next_state <= S0;
                output <= '0';
                end if;


        when S1 =>
                if input <= '1' then
                next_state <= S2 ;
                output <='0';
                else
                next_state <=S1 ;
                output <= '0';
                end if;

        When S2 =>
                if input <='0' then
                next_state <= S3 ;
```

```
                output <='0';
                else
                next_state <= S0;
                output <= '0';
                end if;

        When S3 =>
                if input <='1' then
                next_state <= S2;
                output <='1';
                else
                next_state <=S1;
                output <= '0';
                end if;
end case;
end process state_comb;

state_clocked: process
begin
wait until (clk'event and clk='1');
                if reset='1' then
                state <= s0;
                else
                state <= next_state;
end if;
end process state_clocked;
end state_machine;
```

State Machine: Simulation

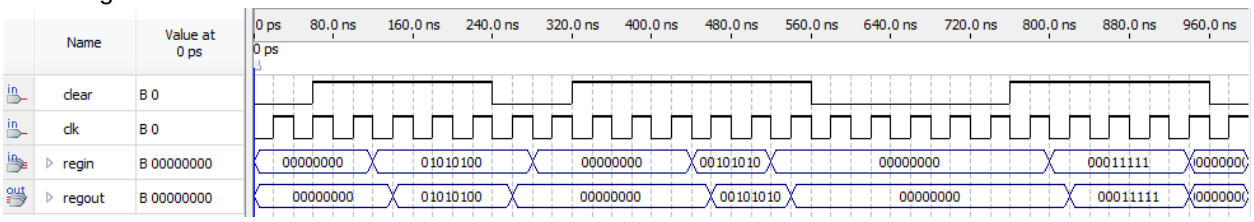


Part 2 Step 1: 8-Bit Adder

8-Bit Adder: Simulation

Part 2 Step 2: 8-Bit Register

8-Bit Register: Simulation



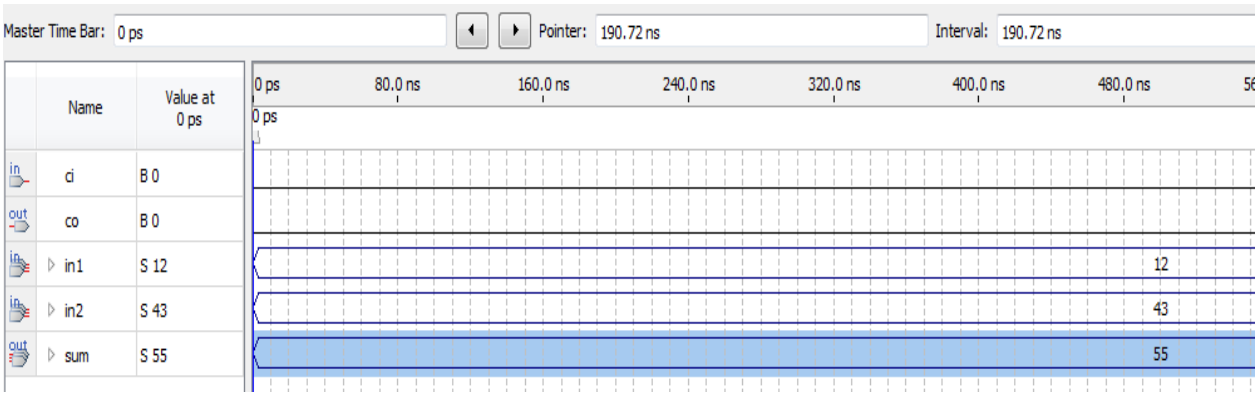
8-Bit Register: VHDL Code

```
--synchronous 8-bit register
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity register8bit is
  port(
  clk:    in std_logic;
  clear:  in std_logic;
  regin:  in std_logic_vector(7 downto 0);
  regout: out std_logic_vector(7 downto 0)
  );
end register8bit;

architecture register8bit of register8bit is
begin
  process
  constant zero: std_logic_vector(7 downto 0):="00000000";
  begin
  --synchronous reset
    wait until (rising_edge(clk));

          if clear = '0' then
            regout <= zero;
    elsif clear = '1' then
            regout <= regin;
          elsif clear = 'X' or clk = 'X' then
            regout <= (others => 'X');
          end if;
  end process;
end register8bit;
```

Part 2

Step 1: 8-Bit Full Adder

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity adder8bit is

Port(
in1: in std_logic_vector (7 downto 0);
in2: in std_logic_vector (7 downto 0);
ci: in std_logic;
sum: out std_logic_vector (7 downto 0);
co: out std_logic);

end adder8bit;

architecture structure_view of adder8bit is

component adder1bit

port (A,B,Cin : in std_logic  ;

 Sum,Cout : out std_logic);

 end component;

 signal co0, co1, co2, co3, co4, co5,co6 : std_logic;

 begin

 FA0: adder1bit port map(in1(0),in2(0),ci,sum(0),co0);
 FA1:  adder1bit port map(in1(1),in2(1),co0,sum(1),co1);
 FA2: adder1bit port map(in1(2),in2(2),co1,sum(2),co2);
 FA3: adder1bit port map(in1(3),in2(3),co2,sum(3),co3);
 FA4: adder1bit port map(in1(4),in2(4),co3,sum(4),co4);
 FA5: adder1bit port map(in1(5),in2(5),co4,sum(5),co5);
 FA6: adder1bit port map(in1(6),in2(6),co5,sum(6),co6);
 FA7: adder1bit port map(in1(7),in2(7),co6,sum(7),co);

 end structure_view;
```



Step 3: 8-Bit Accumulator

8-Bit Accumulator: Simulation

8-Bit Accumulator: VHDL Code