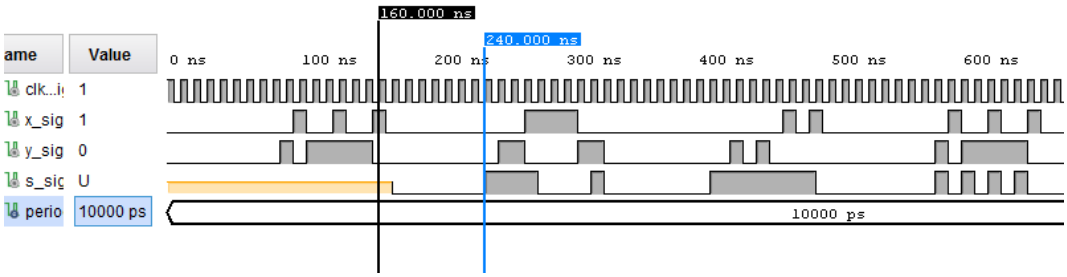
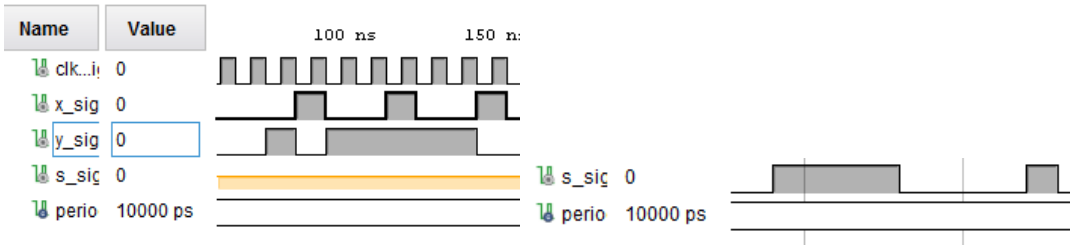
Spring 2019

Computer Assignment 3
Professor: Shahnam Mirzaei phD.
Author: Ridge Tejuco
March 24, 2019

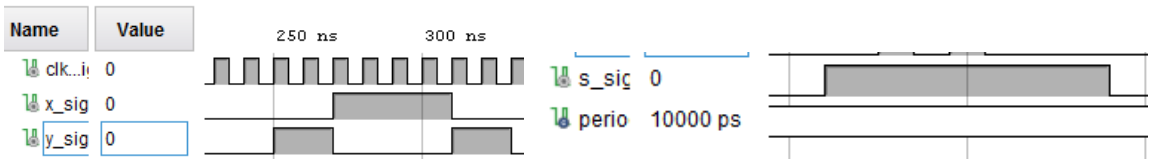
SA TOP RESULTS



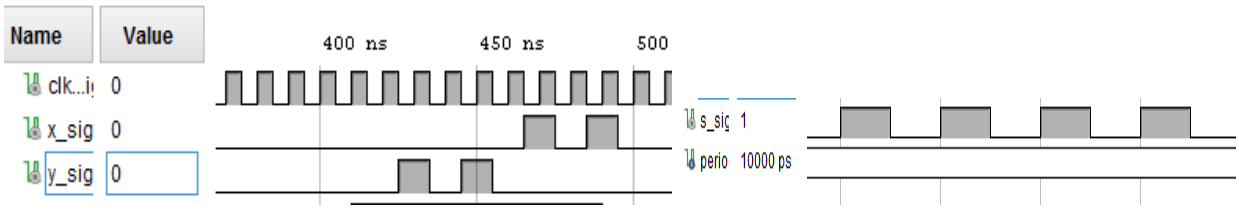
In the test bench 3 additions take place.
The result of each bit is displayed 8 clock cycles after the x and y bit is input.
Looking closer, the results are verified by reading s_sig right to left.




```
  0111 1101
+ 1001 0010
1 0000 1111
```




```
  1100 0011
+ 0011 1100
  1111 1111
```




```
  1010 0000
+ 0000 1010
  1010 1010
```

```
-- sa_top.vhd
-- Ridge Tejuco
library ieee;
use ieee.std_logic_1164.all;
entity sa_top is
     port (x, y, clk:  in std_logic;
     s: out std_logic);
end entity;

architecture siso of sa_top is
     signal x_out,y_out: std_logic;
     signal cout_sig, cin_sig, sum_sig: std_logic;
     signal output: std_logic;
     component shift_register_siso is
     port( clk, reset,d : IN STD_LOGIC;
               q: OUT STD_LOGIC);
     END component shift_register_siso;

     component FullAdder is
     Port ( cin,a,b : in STD_LOGIC;
               sum,cout : out STD_LOGIC);
     end component FullAdder ;

     component dff is
     port(d,clk,reset: in std_logic;
          q: out std_logic);
     end component dff;

     Begin
     X_REG: shift_register_siso
          port map(clk => clk,reset => '0',d => x,q => x_out);
     Y_REG: shift_register_siso
          port map(clk => clk,reset => '0',d => y,q => y_out);
     ADDER: FullAdder
          port map(cin => cin_sig, a => x_out,
               b => y_out, sum => sum_sig,cout => cout_sig);
     CARRY: dff
          port map(d => cout_sig,clk => clk,reset => '0',q =>
cin_sig);
     SUM_REG: shift_register_siso
          port map(clk => clk,reset => '0',d => sum_sig,q =>
output);
     s <= output;
end siso;
```

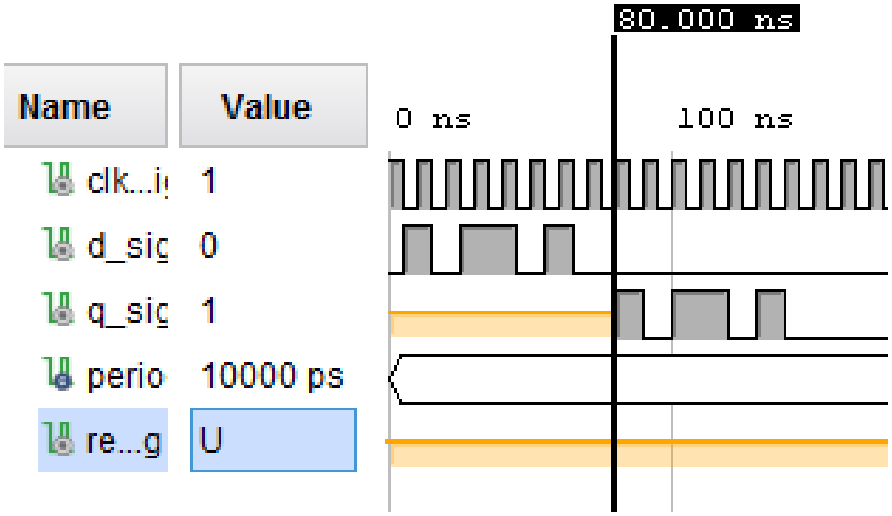
```
--sa_top_tb.vhd
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity sa_top_tb is
--  Port ( );
end sa_top_tb;
architecture Behavioral of sa_top_tb is
     signal clk_sig: std_logic := '0';
     signal x_sig,y_sig,s_sig: std_logic;
     constant period: time := 10ns;
     component sa_top
     port (x, y, clk:  in std_logic;
     s: out std_logic);
     end component;
     begin
     UUT: sa_top
          port map(x => x_sig,y => y_sig,clk => clk_sig,s => s_sig);
     clock: process
          begin
               clk_sig <= not clk_sig;
               wait for period/2;
          end process;
     STIM: process
          begin
               x_sig <= '0'; y_sig <= '0';
               wait for period/2;
               wait for period*8;
     -- next 8 periods is (0111 1101 + 1001 0010) = 1 0000 1111
               x_sig <= '0';y_sig <= '1'; wait for period;
               x_sig <= '1';y_sig <= '0'; wait for period;
               x_sig <= '0';y_sig <= '1'; wait for period;
               x_sig <= '0';y_sig <= '1'; wait for period;
               x_sig <= '1';y_sig <= '1'; wait for period;
               x_sig <= '0';y_sig <= '1'; wait for period;
               x_sig <= '0';y_sig <= '1'; wait for period;
               x_sig <= '1';y_sig <= '0'; wait for period;
               x_sig <= '0'; y_sig <= '0';
               wait for period*8;
               wait for period/2;
    -- next 8 periods is (0011 1100 + 1100 0011) =  1111 1111
               x_sig <= '0';y_sig <= '1'; wait for period;
               x_sig <= '0';y_sig <= '1'; wait for period;
               x_sig <= '1';y_sig <= '0'; wait for period;
               x_sig <= '1';y_sig <= '0'; wait for period;
               x_sig <= '1';y_sig <= '0'; wait for period;
```

```
                x_sig <= '1';y_sig <= '0'; wait for period;
                x_sig <= '0';y_sig <= '1'; wait for period;
                x_sig <= '0';y_sig <= '1'; wait for period;

                x_sig <= '0'; y_sig <= '0';
                wait for period*8;
                wait for period/2;
      -- next 8 periods is (1010 0000 + 0000 1010) =  1010 1010
                x_sig <= '0';y_sig <= '0'; wait for period;
                x_sig <= '0';y_sig <= '1'; wait for period;
                x_sig <= '0';y_sig <= '0'; wait for period;
                x_sig <= '0';y_sig <= '1'; wait for period;
                x_sig <= '0';y_sig <= '0'; wait for period;
                x_sig <= '1';y_sig <= '0'; wait for period;
                x_sig <= '0';y_sig <= '0'; wait for period;
                x_sig <= '1';y_sig <= '0'; wait for period;
           end process;
end Behavioral;
```

```
-- shift_register_siso.vhd
-- Ridge Tejuco
library ieee;
use ieee.std_logic_1164.ALL;
entity shift_register_siso is
    port(    clk, reset,d : IN STD_LOGIC;
            q: OUT STD_LOGIC);
END shift_register_siso;

architecture behavioral of shift_register_siso is
     signal r_reg: std_logic_vector(7 downto 0);
     signal r_next: std_logic_vector(7 downto 0);
     begin
     process(clk,reset)
     begin
          if(reset = '1') then
               r_reg <= (others => '0');
          elsif (clk' event and clk = '1') then
               r_reg <= r_next;
          end if;
     end process;
     -- next state logic;
     r_next <= d & r_reg(7 downto 1);
     -- output logic
     q <= r_reg(0);
end behavioral;
```

```
-- Shift_register_siso_tb
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity shift_register_siso_tb is
--  Port ( );
end shift_register_siso_tb;
architecture Behavioral of shift_register_siso_tb is
signal reset_sig,d_sig,q_sig: std_logic;
signal clk_sig: std_logic := '0';
constant period: time := 10ns;
component shift_register_siso is
    port(    clk, reset,d : IN STD_LOGIC;
            q: OUT STD_LOGIC);
END component shift_register_siso;
begin
uut: shift_register_siso
     port map(clk => clk_sig, reset => reset_sig,d => d_sig, q =>
q_sig);
clock: process
     begin
     clk_sig <= not clk_sig;
     wait for period/2;
     end process;
stim: process
     begin
     d_sig <= '0';
     wait for period/2;
     d_sig <= '1';
     wait for period;
     d_sig <= '0';
     wait for period;
     d_sig <= '1';
     wait for period;
     d_sig <= '1';
     wait for period;
     d_sig <= '0';
     wait for period;
     d_sig <= '1';
     wait for period;
     d_sig <= '0';
     wait;
     end process;
end Behavioral;
```

Shift register Results

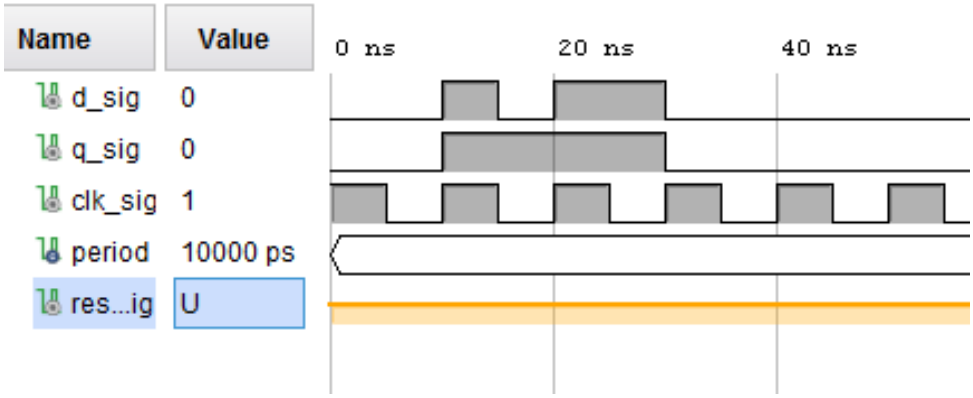


The simulation for the shift register is easily verifiable as the input of d_sig is shifted out after 8 cycles.

```
-- dff.vhd
-- simple d flip flop to hold the carry
-- Ridge Tejuco
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity dff is
      port(d,clk,reset: in std_logic;
      q: out std_logic);
end dff;
architecture Behavioral of dff is
begin
      process(clk,reset)
      begin
      if( reset = '1') then
            q <= '0';
      elsif (clk' event and clk = '1') then
            q <= d;
      end if;
      end process;
end Behavioral;
```

```
--dff_tb.vhd
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity dff_tb is
--  Port ( );
end dff_tb;
architecture Behavioral of dff_tb is
signal d_sig,reset_sig,q_sig: std_logic;
signal clk_sig: std_logic := '0';
constant period: time := 10ns;
component dff is
     port(d,clk,reset: in std_logic;
     q: out std_logic);
end component;
begin
     uut: dff
     port map(d => d_sig, clk => clk_sig, reset => reset_sig, q =>
q_sig);
     clock: process
     begin
          clk_sig <= not clk_sig;
          wait for period/2;
     end process;
     stim: process
     begin
          d_sig <= '0';
          wait for period;
          d_sig <= '1';
          wait for period/2;
          d_sig <= '0'; -- 0 should be skipped
          wait for period/2;
          d_sig <= '1';
          wait for period;
          d_sig <= '0';
          wait;
     end process;
end Behavioral;
```

DFF results



The results for the d flip flop is easy to verify.
The output for q follows the input of d only when the clock is rising edge.
For example q_sig is not set to 0 at 15ns while clk is falling.

```
-- Module Name: FullAdder
-- Ridge Tejuco
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity FullAdder is
     Port ( cin,a,b : in STD_LOGIC;
          sum,cout : out STD_LOGIC);
end FullAdder;

architecture Behavioral of FullAdder is

begin
     cout <= (a and b) or (a and cin) or (b and cin);
     sum <= (a xor b xor cin);
end Behavioral;

-- Full Adder Testbench
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity FullAdderTestBench is
end FullAdderTestBench;
architecture Behavioral of FullAdderTestBench is
signal cin_sig, a_sig,b_sig,cout_sig,sum_sig: STD_LOGIC;
component FullAdder is
     Port ( cin,a,b : in STD_LOGIC;
          sum,cout : out STD_LOGIC;
end component;
begin
UUT: FullAdder
port map (a => a_sig, b => b_sig,cin => cin_sig, cout => cout_sig,sum
=> sum_sig);
STIM: process
begin
     cin_sig <= '0';b_sig <= '0';a_sig <= '0';wait for 10ns;
     cin_sig <= '0';b_sig <= '0';a_sig <= '1';wait for 10ns;
     cin_sig <= '0';b_sig <= '1';a_sig <= '0';wait for 10ns;
     cin_sig <= '0';b_sig <= '1';a_sig <= '1';wait for 10ns;
     cin_sig <= '1';b_sig <= '0';a_sig <= '0';wait for 10ns;
     cin_sig <= '1';b_sig <= '0';a_sig <= '1';wait for 10ns;
     cin_sig <= '1';b_sig <= '1';a_sig <= '0';wait for 10ns;
     cin_sig <= '1';b_sig <= '1';a_sig <= '1';
     wait;
end process;
end Behavioral;
```

Full adder is the same as computer assignment 1