In The Name of God

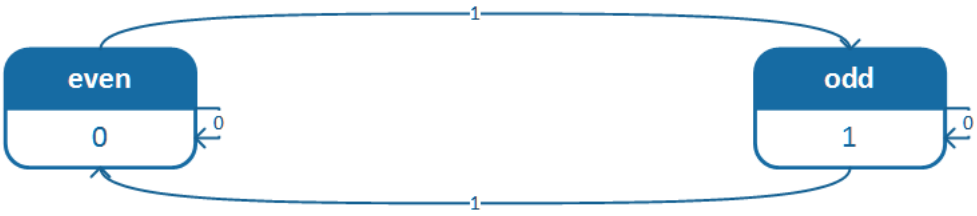
COMPUTER ENGINEERING DEPARTMENT OF AMIRKABIR UNIVERSITY OF TECHNOLOGY

FPGA Homework - 3

Parham Alvani (9231058)

April 26, 2016

1 PROBLEM 5




```
----------------------------------------------------------------------------------
-- Author:         Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:    25-04-2016
-- Module Name:    parity-generator.vhd
----------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

entity parity_generator is
        port (w, clk, reset : in std_logic;
                p : out std_logic);
end entity parity_generator;

architecture rtl of parity_generator is
```

```
        type state is (even, odd);
        signal current_state, next_state : state;
begin
        process (clk)
        begin
                if clk'event and clk = '1' then
                        if reset = '1' then
                                current_state <= even;
                        else
                                current_state <= next_state;
                        end if;
                end if;
        end process;

        process (current_state, w)
        begin
                if current_state = even then
                        if w = '1' then
                                p <= '1';
                                next_state <= odd;
                        else
                                p <= '0';
                                next_state <= even;
                        end if;
                else
                        if w = '1' then
                                p <= '0';
                                next_state <= even;
                        else
                                p <= '1';
                                next_state <= odd;
                        end if;
                end if;
        end process;
end architecture rtl;

----------------------------------------------------------------------------------
-- Author:         Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:    25-04-2016
-- Module Name:    parity-generator.vhd
----------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
```

```
entity parity_generator is
        port (w, clk, reset : in std_logic;
                p : out std_logic);
end entity parity_generator;

architecture rtl of parity_generator is
        type state is (even, odd);
        signal current_state, next_state : state;
begin
        process (clk)
        begin
                if clk'event and clk = '1' then
                        if reset = '1' then
                                current_state <= even;
                        else
                                current_state <= next_state;
                        end if;
                end if;
        end process;

        process (current_state, w)
        begin
                if current_state = even then
                        if w = '1' then
                                p <= '1';
                                next_state <= odd;
                        else
                                p <= '0';
                                next_state <= even;
                        end if;
                else
                        if w = '1' then
                                p <= '0';
                                next_state <= even;
                        else
                                p <= '1';
                                next_state <= odd;
                        end if;
                end if;
        end process;
end architecture rtl;
```

```
-- Author:         Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:    25-04-2016
-- Module Name:    p5_t.vhd
----------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

entity main_t is
end entity;

architecture behavioral of main_t is
        component main
                port (clk, load : in std_logic;
                        b : in std_logic_vector(7 downto 0);
                        serial : out std_logic);
        end component;

        for all:main use entity work.main;

        signal clk, reset : std_logic := '0';
        signal data : std_logic_vector(7 downto 0);
        signal serial : std_logic;
begin
        reset <= '1', '0' after 100 ns;
        clk <= not clk after 50 ns;
        data <= "10101010";

        m : main port map (clk, reset, data, serial);
end architecture;

----------------------------------------------------------------------------------
-- Author:         Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:    25-04-2016
-- Module Name:    p5.vhd
----------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

entity main is
        port (clk, load : in std_logic;
                b : in std_logic_vector(7 downto 0);
                serial : out std_logic);
```

```
end entity;

architecture rtl of main is
        component counter
                generic (N : integer := 4);
                port (clk, reset : in std_logic;
                        count : out std_logic_vector (N - 1 downto 0));
        end component;
        component parity_generator
                port (w, clk, reset : in std_logic;
                        p : out std_logic);
        end component;
        component shift_register
                generic (N : integer := 8);
                port (data_in : in std_logic_vector (N - 1 downto 0);
                        load, clk : in std_logic;
                        data_out : out std_logic);
        end component;

        for all:counter use entity work.counter;
        for all:parity_generator use entity work.parity_generator;
        for all:shift_register use entity work.shift_register;

        signal w, p : std_logic;
        signal c : std_logic_vector(2 downto 0);
begin
        sr:shift_register generic map (8) port map (b, load, clk, w);
        pg:parity_generator port map (w, clk, load, p);
        cn:counter generic map (3) port map (clk, load, c);
        serial <= p when c = "111" else w;
end architecture rtl;
```

2 Problem 6

```
----------------------------------------------------------------------------------
-- Author:         Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:    25-04-2016
-- Module Name:    p6-3.vhd
----------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
```

```
entity seq_detector_3 is
        port (reset, clk, w : in std_logic
             z : out std_logic);
end entity;

architecture rtl of seq_detector_3 is
        type state is (rst, s_1, s_10, s_11, s_100, s_111)
        signal current_state, next_state : state;
begin
        process (clk)
        begin
                if reset = '1' then
                        current_state <= rst;
                elsif clk'event and clk = '1' then
                        currnet_state <= next_state;
                end if;
        end process;

        process (current_state, w)
        begin
                if current_state = rst then
                        if w = '1' then
                                z <= '0';
                                next_state <= s_1;
                        else
                                z <= '0';
                                next_state <= rst;
                        end if;
                elsif currnet_state = s_1 then
                        if w = '1' then
                                z <= '0';
                                next_state <= s_11;
                        else
                                z <= '0';
                                next_state <= s_10;
                        end if;
                elsif current_state = s_11 then
                        if w = '1' then
                                z <= '0';
                                next_state <= s_111;
                        else
                                z <= '0';
                                next_state <= s_10;
                        end if;
```

```
                elsif current_state = s_10 then
                        if w = '1' then
                                z <= '0';
                                next_state <= s_1;
                        else
                                z <= '0';
                                next_state <= s_100;
                        end if;
                elsif current_state = s_100 then
                        if w = '1' then
                                z <= '1';
                                next_state <= s_1;
                        else
                                z <= '0';
                                next_state <= rst;
                        end if;
                elsif current_state = s_111 then
                        if w = '1' then
                                z <= '1';
                                next_state <= s_1;
                        else
                                z <= '0';
                                next_state <= S_10;
                        end if;
                end if;
        end process;
end architecture;

----------------------------------------------------------------------------------
-- Author:         Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:    25-04-2016
-- Module Name:    p6-3.vhd
----------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

entity seq_detector_3 is
        port (reset, clk, w : in std_logic
             z : out std_logic);
end entity;

architecture rtl of seq_detector_3 is
        type state is (rst, s_1, s_10, s_11, s_100, s_111)
```

```
        signal current_state, next_state : state;
begin
        process (clk)
        begin
                if reset = '1' then
                        current_state <= rst;
                elsif clk'event and clk = '1' then
                        currnet_state <= next_state;
                end if;
        end process;

        process (current_state, w)
        begin
                if current_state = rst then
                        if w = '1' then
                                next_state <= s_1;
                        else
                                next_state <= rst;
                        end if;
                elsif currnet_state = s_1 then
                        if w = '1' then
                                next_state <= s_11;
                        else
                                next_state <= s_10;
                        end if;
                elsif current_state = s_11 then
                        if w = '1' then
                                next_state <= s_111;
                        else
                                next_state <= s_10;
                        end if;
                elsif current_state = s_10 then
                        if w = '1' then
                                next_state <= s_1;
                        else
                                next_state <= s_100;
                        end if;
                elsif current_state = s_100 then
                        if w = '1' then
                                next_state <= s_1;
                        else
                                next_state <= rst;
                        end if;
                elsif current_state = s_111 then
```

```
                if w = '1' then
                        next_state <= s_1;
                else
                        next_state <= S_10;
                end if;
        end if;
end process;

process (current_state, w)
begin
        if current_state = rst then
                if w = '1' then
                        z <= '0';
                else
                        z <= '0';
                end if;
        elsif currnet_state = s_1 then
                if w = '1' then
                        z <= '0';
                else
                        z <= '0';
                end if;
        elsif current_state = s_11 then
                if w = '1' then
                        z <= '0';
                else
                        z <= '0';
                end if;
        elsif current_state = s_10 then
                if w = '1' then
                        z <= '0';
                else
                        z <= '0';
                end if;
        elsif current_state = s_100 then
                if w = '1' then
                        z <= '1';
                else
                        z <= '0';
                end if;
        elsif current_state = s_111 then
                if w = '1' then
                        z <= '1';
                else
```

```
                                z <= '0';
                        end if;
                end if;
        end process;
end architecture;
```

3 Problem 7

Mealy machine outputs are not synchronous with clock, if we can synchronous them with clock we can use it's output better with our sequential logic. One way is to put some register after our output generator combinational circuit.

4 Problem 8

```
----------------------------------------------------------------------------------
-- Author:         Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:    26-04-2016
-- Module Name:    p8.vhd
----------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

entity p8 is
        port (i1, i2 : in std_logic;
                o : out std_logic;
                clk, reset : in std_logic);
end entity;

architecture sequential of p8 is
        type state is (s0, s1, s2, s3, s4, s5);
        signal current_state, next_state : state;

        attribute fsm_encoding : string;
        attribute fsm_encoding of current_state : signal is "sequential";
begin
        process (clk, reset)
        begin
                if reset = '1' then
                        current_state <= s0;
                elsif clk'event and clk = '1' then
                        current_state <= next_state;
                end if;
```

```
end process;

process (current_state, i1, i2)
begin
        case current_state is
                when s0 =>
                        if i1 = '0' and i2 = '0' then
                                next_state <= s0;
                                o <= '0';
                        elsif i1 = '1' and i2 = '1' then
                                next_state <= s0;
                                o <= '0';
                        elsif i1 = '0' and i2 = '1' then
                                next_state <= s2;
                                o <= '0';
                        elsif i1 = '1' and i2 = '0' then
                                next_state <= s4;
                                o <= '0';
                        end if;
                when s1 =>
                        if i1 = '0' and i2 = '0' then
                                next_state <= s1;
                                o <= '1';
                        elsif i1 = '1' and i2 = '1' then
                                next_state <= s1;
                                o <= '1';
                        elsif i1 = '0' and i2 = '1' then
                                next_state <= s3;
                                o <= '1';
                        elsif i1 = '1' and i2 = '0' then
                                next_state <= s5;
                                o <= '1';
                        end if;
                when s2 =>
                        if i1 = '0' and i2 = '0' then
                                next_state <= s0;
                                o <= '0';
                        elsif i1 = '1' and i2 = '1' then
                                next_state <= s0;
                                o <= '0';
                        elsif i1 = '0' and i2 = '1' then
                                next_state <= s2;
                                o <= '0';
                        elsif i1 = '1' and i2 = '0' then
```

```
                next_state <= s4;
                o <= '0';
        end if;
when s3 =>
        if i1 = '0' and i2 = '0' then
                next_state <= s1;
                o <= '1';
        elsif i1 = '1' and i2 = '1' then
                next_state <= s0;
                o <= '0';
        elsif i1 = '0' and i2 = '1' then
                next_state <= s3;
                o <= '1';
        elsif i1 = '1' and i2 = '0' then
                next_state <= s5;
                o <= '1';
        end if;
when s4 =>
        if i1 = '0' and i2 = '0' then
                next_state <= s0;
                o <= '0';
        elsif i1 = '1' and i2 = '1' then
                next_state <= s1;
                o <= '1';
        elsif i1 = '0' and i2 = '1' then
                next_state <= s3;
                o <= '1';
        elsif i1 = '1' and i2 = '0' then
                next_state <= s4;
                o <= '0';
        end if;
when s5 =>
        if i1 = '0' and i2 = '0' then
                next_state <= s1;
                o <= '1';
        elsif i1 = '1' and i2 = '1' then
                next_state <= s1;
                o <= '1';
        elsif i1 = '0' and i2 = '1' then
                next_state <= s2;
                o <= '0';
        elsif i1 = '1' and i2 = '0' then
                next_state <= s5;
                o <= '1';
```

```
                        end if;
                when others =>
                        next_state <= s0;
                        o <= '0';
        end case;
    end process;
end architecture;

architecture medvedev of p8 is
    constant s0 : std_logic_vector (2 downto 0) := "000";
    constant s1 : std_logic_vector (2 downto 0) := "100";
    constant s2 : std_logic_vector (2 downto 0) := "001";
    constant s3 : std_logic_vector (2 downto 0) := "101";
    constant s4 : std_logic_vector (2 downto 0) := "010";
    constant s5 : std_logic_vector (2 downto 0) := "110";

    signal current_state, next_state : std_logic_vector (2 downto 0);
begin
    process (clk, reset)
    begin
        if reset = '1' then
            current_state <= s0;
        elsif clk'event and clk = '1' then
            current_state <= next_state;
        end if;
    end process;

    o <= current_state(2);

    process (current_state, i1, i2)
    begin
        case current_state is
            when s0 =>
                if i1 = '0' and i2 = '0' then
                    next_state <= s0;
                elsif i1 = '1' and i2 = '1' then
                    next_state <= s0;
                elsif i1 = '0' and i2 = '1' then
                    next_state <= s2;
                elsif i1 = '1' and i2 = '0' then
                    next_state <= s4;
                end if;
            when s1 =>
                if i1 = '0' and i2 = '0' then
```

```
                next_state <= s1;
        elsif i1 = '1' and i2 = '1' then
                next_state <= s1;
        elsif i1 = '0' and i2 = '1' then
                next_state <= s3;
        elsif i1 = '1' and i2 = '0' then
                next_state <= s5;
        end if;
when s2 =>
        if i1 = '0' and i2 = '0' then
                next_state <= s0;
        elsif i1 = '1' and i2 = '1' then
                next_state <= s0;
        elsif i1 = '0' and i2 = '1' then
                next_state <= s2;
        elsif i1 = '1' and i2 = '0' then
                next_state <= s4;
        end if;
when s3 =>
        if i1 = '0' and i2 = '0' then
                next_state <= s1;
        elsif i1 = '1' and i2 = '1' then
                next_state <= s0;
        elsif i1 = '0' and i2 = '1' then
                next_state <= s3;
        elsif i1 = '1' and i2 = '0' then
                next_state <= s5;
        end if;
when s4 =>
        if i1 = '0' and i2 = '0' then
                next_state <= s0;
        elsif i1 = '1' and i2 = '1' then
                next_state <= s1;
        elsif i1 = '0' and i2 = '1' then
                next_state <= s3;
        elsif i1 = '1' and i2 = '0' then
                next_state <= s4;
        end if;
when s5 =>
        if i1 = '0' and i2 = '0' then
                next_state <= s1;
        elsif i1 = '1' and i2 = '1' then
                next_state <= s1;
        elsif i1 = '0' and i2 = '1' then
```

```
                                        next_state <= s2;
                                elsif i1 = '1' and i2 = '0' then
                                        next_state <= s5;
                                end if;
                        when others =>
                                next_state <= s0;
                end case;
        end process;
end architecture;

--------------------------------------------------------------------------------
-- Author:         Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:    26-04-2016
-- Module Name:    p8_t.vhd
--------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

entity p8_t is
end entity;

architecture rtl of p8_t is
        component p8
                port (i1, i2 : in std_logic;
                        o : out std_logic;
                        clk, reset : in std_logic);
        end component;

        for all:p8 use entity work.p8(sequential);

        signal clk : std_logic := '0';
        signal reset, i1, i2, o : std_logic;
begin
        clk <= not clk after 50 ns;
        reset <= '1', '0' after 40 ns;
        i1 <= '1', '0' after 110 ns, '1' after 220 ns;
        i2 <= '0', '1' after 55 ns, '0' after 210 ns;
        m : p8 port map (i1, i2, o, reset, clk);
end architecture;
```

5 Problem 9

```
----------------------------------------------------------------------------------
-- Author:         Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:    25-04-2016
-- Module Name:    p9.vhd
----------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

entity robot is
        port (key1, key2, key3, clk : in std_logic);
end entity;

architecture rtl of robot is
        type state is (stop, move_forward_slowly, move_forward_fast,
                move_backward_fast, move_backward_slowly, turn_right,
                turn_left);
        signal current_state, next_state : state;
        signal command : std_logic_vector (2 downto 0);
begin
        command <= key1 & key2 & key3;
        process (command)
        begin
                case command is
                        when "000" => next_state <= stop;
                        when "001" => next_state <= move_forward_slowly;
                        when "010" => next_state <= move_forward_fast;
                        when "011" => next_state <= move_backward_slowly;
                        when "100" => next_state <= move_backward_fast;
                        when "101" => next_state <= turn_right;
                        when "110" => next_state <= turn_left;
                        when "111" => next_state <= current_state;
                        when others => next_state <= stop;
                end case;
        end process;
        process (clk)
        begin
                if clk'event and clk = '1' then
                        current_state <= next_state;
                end if;
        end process;
end architecture;
```

6 Problem 10

```
----------------------------------------------------------------------------------
-- Author:         Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:    25-04-2016
-- Module Name:    p10.vhd
----------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity alu is
        port (a, b : in std_logic_vector (3 downto 0);
                alucode : in std_logic_vector (2 downto 0);
                result : out std_logic_vector (3 downto 0);
                z, o : out std_logic);
end entity;

architecture rtl of alu is
begin
        process (alucode, a, b)
                variable im : std_logic_vector (4 downto 0);
        begin
                case alucode is
                        -- ADD
                        when "000" =>
                                im := ('0' & a) + ('0' & b);
                                result <= im(3 downto 0);
                                z <= im(4);
                        -- SUB
                        when "001" =>
                                im := ('0' & a) - ('0' & b);
                                result <= im(3 downto 0);
                                z <= im(4);
                        -- AND
                        when "010" =>
                                im(3 downto 0) := (a(0) and b(0)) & (a(1) and b(1)) & (a(2
                                if im(3 downto 0) = "111" then
                                        z <= '1';
                                else
                                        z <= '0';
                                end if;
                                result <= im(3 downto 0);
```

```
                        -- CMP
                        when "011" =>
                                if a < b then
                                        result <= a;
                                        z <= '0';
                                elsif a = b then
                                        result <= a;
                                        z <= '1';
                                else
                                        result <= b;
                                        z <= '0';
                                end if;
                        -- RT
                        when "100" =>
                                result <= '0' & a(2 downto 0);
                                z <= a(3);
                        -- RR
                        when "101" =>
                                result <= a(3 downto 1) & '0';
                                z <= a(0);
                        -- Parity
                        when "110" =>
                                result <= a;
                                z <= a(0) xor a(1) xor a(2) xor a(3);
                        when others =>
                                result <= (others => '0');
                                z <= '0';
                                o <= '0';
                end case;
        end process;
end architecture;
```

7 Problem 11

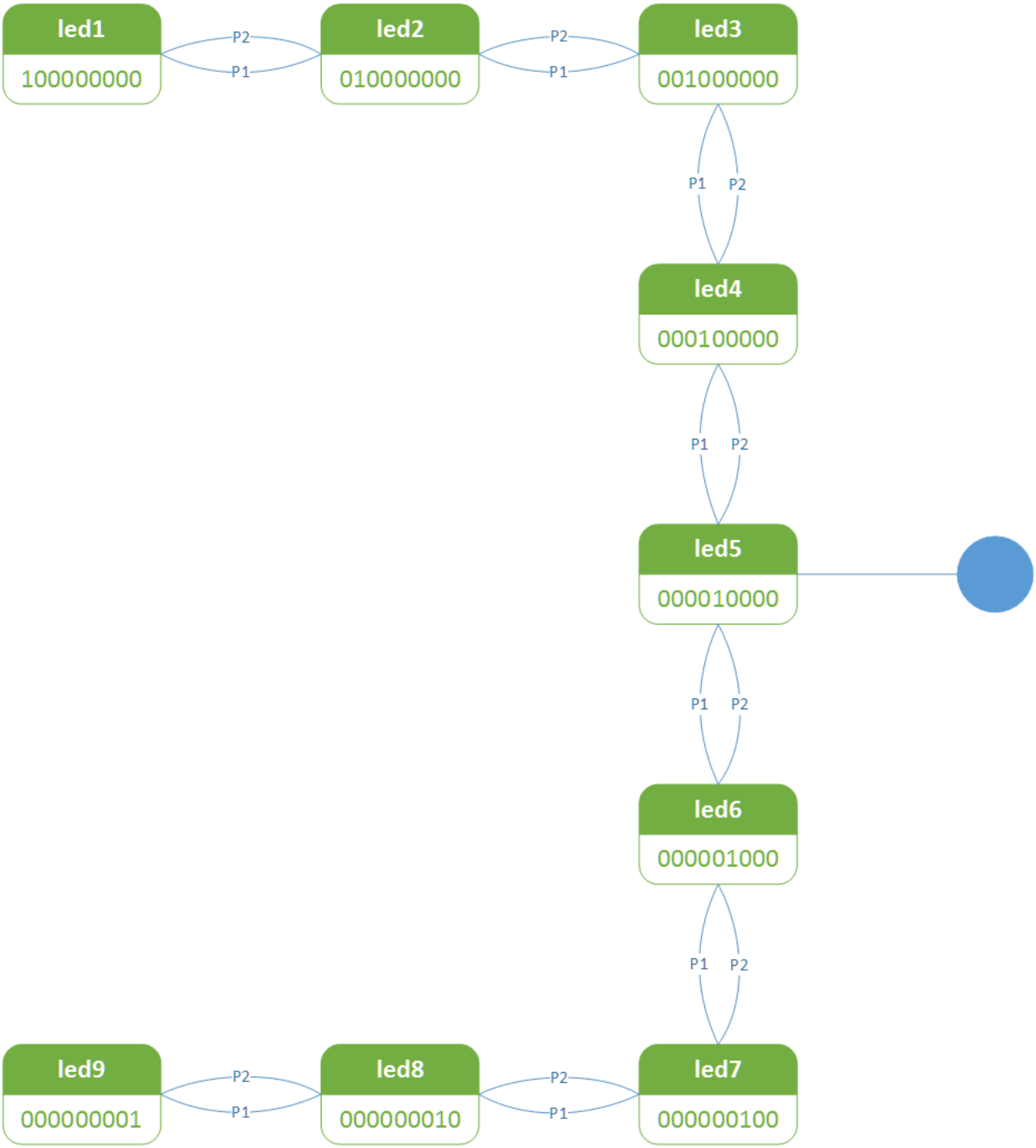
```
----------------------------------------------------------------------------------
-- Author:         Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:    26-04-2016
-- Module Name:    p11-in.vhd
----------------------------------------------------------------------------------
library iEEE;
use IEEE.std_logic_1164.all;
```

```
entity p11_in is
end entity;

architecture rtl of p11_in is
        entity sevsegv is
                port (d0, d1, d2, d3 : in vlbit;
                        a, b, c, d, e, f, g : out vlbit);
        end component;

        for sv1, sv2:sevsegv use entity work.sevsegv(behave1);
        for sv3, sv4:sevsegv use entity work.sevsegv(behave2);
begin
        sv1:sevsegv port map ();
        sv2:sevsegv port map ();
        sv3:sevsegv port map ();
        sv4:sevsegv port map ();
end architecture;

----------------------------------------------------------------------------------
-- Author:         Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:    26-04-2016
-- Module Name:    p11-out.vhd
----------------------------------------------------------------------------------
library iEEE;
use IEEE.std_logic_1164.all;

entity p11_out is
end entity;

architecture rtl of p11_out is
        entity sevsegv is
                port (d0, d1, d2, d3 : in vlbit;
                        a, b, c, d, e, f, g : out vlbit);
        end component;
begin
        sv1:sevsegv port map ();
        sv2:sevsegv port map ();
        sv3:sevsegv port map ();
        sv4:sevsegv port map ();
end architecture;

configuration conf of p11_out is
        for rtl
```

```
                for sv1, sv2:sevsegv
                        use entity work.sevsegv(behave1;
                end for;
                for sv3, sv4:sevsegv
                        use entity work.sevsegv(behave2);
                end for;
        end for;
end configuration;
```

8 PROBLEM 12

In this problem I prefer to use one-hot coding because its good on FPGA and do not create redundant states and safer than other methods.




```
--------------------------------------------------------------------------------
-- Author:         Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:    26-04-2016
```

```
-- Module Name:   p12.vhd
----------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

entity drawstring is
        port (p1, p2 : in std_logic;
                clk, reset : in std_logic;
                led : out std_logic_vector (9 downto 1));
end entity;

architecture rtl of drawstring is
        type state is (led1, led2, led3, led4, led5, led6, led7, led8, led9);
        signal current_state, next_state : state := led5;
begin
        process (clk, reset)
        begin
                if reset = '1' then
                        current_state <= led5;
                elsif clk'event and clk = '1' then
                        current_state <= next_state;
                end if;
        end process;

        process (current_state, p1, p2)
        begin
                case current_state is
                        when led1 =>
                                if p1'event and p1 = '1' then
                                        next_state <= led2;
                                elsif p2'event and p2 = '1' then
                                        next_state <= led1;
                                end if;
                        when led2 =>
                                if p1'event and p1 = '1' then
                                        next_state <= led3;
                                elsif p2'event and p2 = '1' then
                                        next_state <= led1;
                                end if;
                        when led3 =>
                                if p1'event and p1 = '1' then
                                        next_state <= led4;
                                elsif p2'event and p2 = '1' then
                                        next_state <= led2;
```

```
                end if;
            when led4 =>
                if p1'event and p1 = '1' then
                        next_state <= led5;
                elsif p2'event and p2 = '1' then
                        next_state <= led3;
                end if;
            when led5 =>
                if p1'event and p1 = '1' then
                        next_state <= led6;
                elsif p2'event and p2 = '1' then
                        next_state <= led4;
                end if;
            when led6 =>
                if p1'event and p1 = '1' then
                        next_state <= led7;
                elsif p2'event and p2 = '1' then
                        next_state <= led5;
                end if;
            when led7 =>
                if p1'event and p1 = '1' then
                        next_state <= led8;
                elsif p2'event and p2 = '1' then
                        next_state <= led6;
                end if;
            when led8 =>
                if p1'event and p1 = '1' then
                        next_state <= led9;
                elsif p2'event and p2 = '1' then
                        next_state <= led7;
                end if;
            when led9 =>
                if p1'event and p1 = '1' then
                        next_state <= led9;
                elsif p2'event and p2 = '1' then
                        next_state <= led8;
                end if;
        end case;
end process;

process (current_state)
begin
        case current_state is
            when led1 => led <= (1 => '1', others => '0');
```

```
                        when led2 => led <= (2 => '1', others => '0');
                        when led3 => led <= (3 => '1', others => '0');
                        when led4 => led <= (4 => '1', others => '0');
                        when led5 => led <= (5 => '1', others => '0');
                        when led6 => led <= (6 => '1', others => '0');
                        when led7 => led <= (7 => '1', others => '0');
                        when led8 => led <= (8 => '1', others => '0');
                        when led9 => led <= (9 => '1', others => '0');
                end case;
        end process;
end architecture;

----------------------------------------------------------------------------------
-- Author:         Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:    26-04-2016
-- Module Name:    p12_t.vhd
----------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

entity drawstring_t is
end entity;

architecture rtl of drawstring_t is
        component drawstring
                port (p1, p2 : in std_logic;
                        clk, reset : in std_logic;
                        led : out std_logic_vector (9 downto 1));
        end component;

        for all:drawstring use entity work.drawstring;

        signal reset, p1, p2 : std_logic;
        signal clk : std_logic := '0';
        signal led : std_logic_vector (9 downto 1);
begin
        reset <= '1', '0' after 50 ns;
        clk <= not clk after 50 ns;

        p1 <= '1' after 50 ns, '0' after 110 ns, '1' after 220 ns;
        p2 <= '0', '1' after 55 ns, '0' after 65 ns;

        m : drawstring port map (p1, p2, clk, reset, led);
```

```
end architecture;
```