In The Name of God

COMPUTER ENGINEERING DEPARTMENT OF AMIRKABIR UNIVERSITY OF TECHNOLOGY

FPGA Homework - 6

Parham Alvani (9231058)

May 31, 2016

1 PROBLEM 1

Intellectual property core, IP core, or IP block is a reusable unit of logic, cell, or chip layout design that is the intellectual property of one party.
We use IP cores in order to design our system modular.
IP cores must be:

- understandable behavioural code of IP
- testability
- register output
- customizable and configurable
- proper number of I/O ports

USB interface, CORDIC, MD5, ...
IP cores categorize into following groups: (flexibility and portability decrease from top to down)

- Soft IP
- Firm IP
- Hard IP

2 Problem 2

FPGA special-purpose blocks like DSP,...

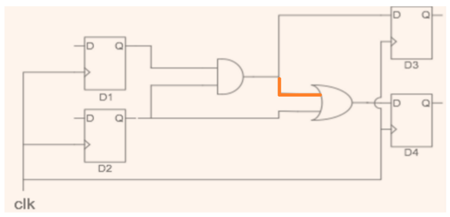
3 Problem 3

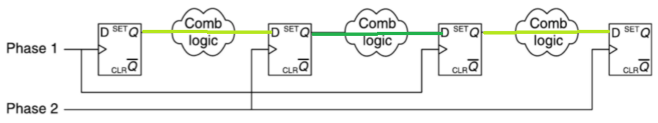
We use floorplaning for distributes design modules in FPGA, we assign specific area of FPGA to design modules.

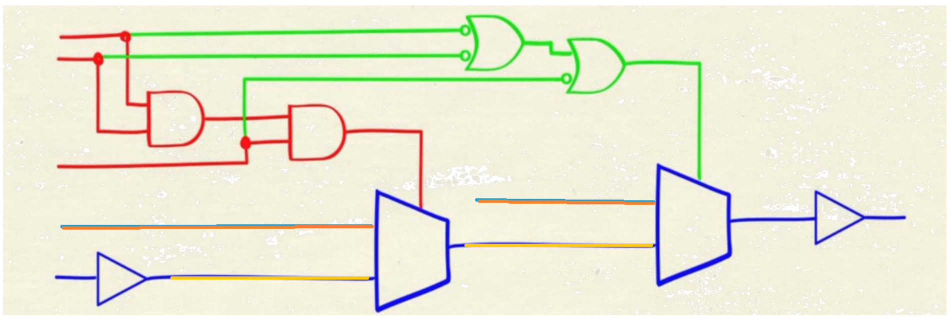
4 Problem 4

The critical path is defined as the path between an input and an output with the maximum delay.
The false path is defined as the path that never happend.
Some combinational paths from FF to another have more time than a one clock cycle.

5 Problem 5







6 Problem 6

2 Full Adder (without resource sharing)

```
--------------------------------------------------------------------------------
-- Author:          Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:     30-05-2016
-- Module Name:     p6-1.vhd
--------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;


entity adder_2_mux is
        port (a, b, c : in std_logic;
                sel : in std_logic;
                o : out std_logic);
end entity;

architecture rtl of addr_2_mux is
begin
        o <= a + c when sel = '0' else b + c;
end architecture;
```

1 Full Adder (with resource sharing)

```
--------------------------------------------------------------------------------
-- Author:         Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:    30-05-2016
-- Module Name:    p6-2.vhd
--------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;


entity adder_1_mux is
        port (a, b, c : in std_logic;
                sel : in std_logic;
                o : out std_logic);
end entity;

architecture rtl of addr_1_mux is
        signal t : std_logic;
begin
        t <= a when sel = '0' else b;
        o <= c + t;
end architecture;
```

7 Problem 7

- Memory Block Size: $1K * 10$
- Address: $X\&Y$
- Memory Block Size: $4K * 2$
- Address: $X\&Y$

8 Problem 8

```
--------------------------------------------------------------------------------
-- Author:         Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:    30-05-2016
-- Module Name:    control.vhd
--------------------------------------------------------------------------------
library IEEE;
```

```vhdl
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity control is
        port (ent, ext : out std_logic;
                a, b : in std_logic;
                clk : in std_logic);
end entity;

architecture mem of control is
        type state is array (natural range <>) of std_logic_vector (3 downto 0);

        signal next_state, current_state : std_logic_vector (1 downto 0);
        signal tt : state (15 downto 0) := (
                8 => "0001",
                12 => "0001",
                0 => "0000",
                4 => "0000",
                5 => "1010",
                13 => "1010",
                1 => "0001",
                9 => "0001",
                2 => "0011",
                6 => "0011",
                10 => "0010",
                14 => "0010",
                3 => "0100",
                11 => "0100",
                7 => "0011",
                15 => "0011"
        );
begin
        process (clk)
        begin
                if clk'event and clk = '1' then
                        current_state <= next_state;
                end if;
        end process;
        process (current_state, a, b)
                variable addr : std_logic_vector (3 downto 0);
        begin
                addr := a & b & current_state;

                ent <= tt(to_integer(unsigned(addr)))(3);
                ext <= tt(to_integer(unsigned(addr)))(2);
                next_state <= tt(to_integer(unsigned(addr)))(1 downto 0);
        end process;
end architecture;

architecture rtl of control is
        type state is (S0, S1, S2, S3);
        signal current_state, next_state : state;
begin
        process (clk)
        begin
                if clk'event and clk = '1' then
                        current_state <= next_state;
                end if;
        end process;
        process (current_state, a, b)
        begin
                case current_state is
                        when S0 =>
                                if a = '1' then
                                        next_state <= S1;
                                end if;
                                ent <= '0';
                                ext <= '0';
                        when S1 =>
                                if b = '1' then
                                        next_state <= S2;
                                        ent <= '1';
                                        ext <= '0';
                                end if;
                        when S2 =>
                                if a = '0' then
                                        next_state <= S3;
                                end if;
                                ent <= '0';
                                ext <= '0';
                        when S3 =>
                                if b = '0' then
                                        next_state <= S0;
                                        ent <= '0';
                                        ext <= '1';
                                end if;
                        when others =>
                                next_state <= S0;
                                ent <= '0';
                                ext <= '0';
                end case;
        end process;
end architecture;

--------------------------------------------------------------------------------
-- Author:        Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:   30-05-2016
-- Module Name:   counter.vhd
--------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity counter is
        generic (N : integer := 4);
        port (inc, dec : in std_logic;
                output : out std_logic_vector (N - 1 downto 0);
                clk : in std_logic);
end entity;

architecture rtl of counter is
        signal count : std_logic_vector (N - 1 downto 0) := (others => '0');
begin
        output <= count;
        process (clk)
        begin
                if clk'event and clk = '1' then
                        if inc = '1' then
                                count <= count + (0 => '1');
                        elsif dec = '1' then
                                count <= count - (0 => '1');
                        end if;
                end if;
        end process;
end architecture;

--------------------------------------------------------------------------------
-- Author:        Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:   30-05-2016
-- Module Name:   datapath.vhd
```

```vhdl
--------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;


entity datapath is
        port (clk, ent, ext : in std_logic;
             output : out std_logic_vector (3 downto 0));
end entity;

architecture rtl of datapath is
        component counter
                generic (N : integer := 4);
                port (inc, dec : in std_logic;
                        output : out std_logic_vector (N - 1 downto 0);
                        clk : in std_logic);
        end component;

        for all:counter use entity work.counter;
begin
        cntr : counter port map (ent, ext, output, clk);
end architecture;

--------------------------------------------------------------------------------
-- Author:        Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:   31-05-2016
-- Module Name:   main.vhd
--------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

entity main is
        port (clk, a, b : in std_logic;
             output : out std_logic_vector (3 downto 0));
end entity;

architecture rtl of main is
        component datapath
                port (clk, ent, ext : in std_logic;
                     output : out std_logic_vector (3 downto 0));
        end component;

        component control
```

```
                port (ent, ext : out std_logic;
                        a, b : in std_logic;
                        clk : in std_logic);
        end component;

        for all:control use entity work.control(rtl);
        for all:datapath use entity work.datapath;

        signal ent, ext : std_logic;
begin
        ctrl : control port map (ent, ext, a, b, clk);
        dp : datapath port map (clk, ent, ext, output);
end architecture;

--------------------------------------------------------------------------------
-- Author:         Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:    31-05-2016
-- Module Name:    main_t.vhd
--------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

entity main_t is
end entity;

architecture rtl of main_t is
        component main
                port (clk, a, b : in std_logic;
                     output : out std_logic_vector (3 downto 0));
        end component;

        signal clk, a, b : std_logic := '0';
        signal status : std_logic_vector (3 downto 0);
begin
        clk <= not clk after 25 ns;

        a <= '1' after 250 ns, '0' after 400 ns;
        b <= '1' after 300 ns, '0' after 450 ns;

        m : main port map (clk, a, b, status);
end architecture;
```