In The Name of God

COMPUTER ENGINEERING DEPARTMENT OF AMIRKABIR UNIVERSITY OF TECHNOLOGY

Iman Tabrizian (9331032)

April 24, 2017

1 PROBLEM 5

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

entity timer is
    port(
        pause: in std_logic := '0';
        clk: in std_logic;
        rst: in std_logic;
        counted: out integer
    );
end entity;

architecture rtl of timer is
begin
    process(clk)
        variable count: integer := 0;
    begin
        if(rising_edge(clk)) then
            if (rst = '1') then
                count := 0;
            elsif (pause = '0') then
                count := count + 2;
            else
                count := count;
            end if;
```

```
            counted <= count / 1000;
        end if;
    end process;
end rtl;
```

I have assumed that resume/pause options are one input. So the state of timer is either resuming or pausing.

2 Problem 6

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

entity timer is
    port(
        zero: in std_logic;
        one: in std_logic;
        rst: in std_logic;
        clk: in std_logic;
        hashtag: in std_logic;
        unlock: out std_logic
    );
end entity;

architecture rtl of timer is
    type STATE_TYPE is (s0, s1, s2, s3, s4, s5, s6, accept, trap);
    signal state: STATE_TYPE := s0;
begin
    process(clk)
    begin
        if (rising_edge(clk)) then
            if(rst = '1') then
                state <= s0;
            else
                case state is
                    when s0 =>
                        if(zero = '1') then
                            state <= s1;
                        else
                            state <= trap;
                        end if;
                        unlock <= '0';
                    when s1 =>
                        if(one = '1') then
```

```
            state <= s2;
        else
            state <= trap;
        end if;
        unlock <= '0';
    when s2 =>
        if(zero = '1') then
            state <= s3;
        else
            state <= trap;
        end if;
        unlock <= '0';
    when s3 =>
        if(one = '1') then
            state <= s4;
        else
            state <= trap;
        end if;
        unlock <= '0';
    when s4 =>
        if(one = '1') then
            state <= s5;
        else
            state <= trap;
        end if;
        unlock <= '0';
    when s5 =>
        if(one = '1') then
            state <= s6;
        else
            state <= trap;
        end if;
        unlock <= '0';
    when s6 =>
        if(hashtag = '1') then
            state <= accept;
        else
            state <= trap;
        end if;
        unlock <= '0';
    when accept =>
        if(zero = '1') then
            state <= s1;
        else
```

```
                        state <= trap;
                    end if;
                    unlock <= '1';
                when trap =>
                    state <= trap;
            end case;
        end if;
    end if;
end process;
end rtl;
```

I have used the default encoding which is sequential. Because it uses reduced number of bits and results in lower number of FFs. The report of number LUTs are attached
+—Adders :
2 Input 32 Bit Adders := 1
+—Registers :
64 Bit Registers := 3
32 Bit Registers := 1
4 Bit Registers := 1
+—Muxes :
2 Input 64 Bit Muxes := 2
4 Input 36 Bit Muxes := 1
3 Input 3 Bit Muxes := 1
3 Input 2 Bit Muxes := 1
3 Input 1 Bit Muxes := 4
2 Input 1 Bit Muxes := 3
5 Input 1 Bit Muxes := 1
Module halfadder
Detailed RTL Component Info :
+—XORs :
2 Input 1 Bit XORs := 1
Module bcdadder
Detailed RTL Component Info :
+—Muxes :
2 Input 4 Bit Muxes := 4
Module bcdaddersimple
Detailed RTL Component Info :
+—Muxes :
2 Input 4 Bit Muxes := 2
2 Input 1 Bit Muxes := 1

3 Problem 7

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

entity question is
end entity;

architecture test of question is
    signal a, b, c: std_logic := '0';
    signal clk: std_logic := '0';
begin
    c <= '1';
    b <= '0';
    clk <= not clk after 10 ns;
    process(clk)
        variable d: std_logic;
    begin
        a <= b;
        b <= c xor b;
        d := c and a;
        c <= not c;
    end process;
end test;
```

The outputs of all of the signals is X. Because there are multiple drivers for the all of the signals so the resulting signals are X.

4 Problem 8

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

entity question is
end entity;

architecture test of question is
    signal a, b, c: std_logic := '0';
    signal clk: std_logic := '0';
begin
    c <= '1';
    b <= '0';
    clk <= not clk after 10 ns;
    process(clk)
```

```
        variable d: std_logic;
    begin
        a <= b;
        b <= c xor b;
        d := c and a;
        c <= not c;
    end process;
end test;
```