In The Name of God

COMPUTER ENGINEERING DEPARTMENT OF AMIRKABIR UNIVERSITY OF TECHNOLOGY

FPGA Homework - 4

Parham Alvani (9231058)

May 7, 2016

1 PROBLEM 4

```
--------------------------------------------------------------------------------
-- Author:         Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:    06-05-2016
-- Module Name:    p4-1.vhd
--------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

entity logic1 is
        port (A, B, C : in std_logic;
                F : out std_logic);
end logic1;

architecture behavioral of logic1 is
begin
        process(A,B,C) begin
                if A = '1' then
                        F<= '1';
                elsif B = '1' and C = '1' then
                        F <= '0';
                -- providing else clause in order to
```

```
                -- preventing from transparent latch
                -- creation.
                else
                        F <= 'X';
                end if;
        end process;
end behavioral;

----------------------------------------------------------------------------------
-- Author:         Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:    07-05-2016
-- Module Name:    p4-2.vhd
----------------------------------------------------------------------------------


-- next state logic for a FSM
process (state, a, b, c, d, e)
begin
        case state is
                when IDLE =>
                        if a = '0' then
                                next_state <= INITIAL;
                        -- preventing from transparent latch creation
                        else
                                next_state <= IDLE;
                        end if;
                when INITIAL =>
                        if a = '1' then
                                next_state <= ERROR_FLAG;
                        else
                                next_state <= SCANNING;
                        end if;
                when SCANNING =>
                        if b = '1' then
                                next_state <= LOCKED;
                        elsif b = '0' then
                                if c = '0' then
                                        next_state <= TIME_OUT;
                                else
                                        next_state <= RELEASE;
                                end if;
                        -- following statement never happening ...
                        else
```

```
                                next_state <= CAPTURE;
                        end if;
                when CAPTURE =>
                        next_state <= ...
                when LOCKED =>
                        next_state <= ...
                when TIME_OUT =>
                        next_state <= ...
                when RELEASE =>
                        next_state <= ...
                when ERROR_FLAG =>
                        next_state <= some_function(a, d, e);
        end case;
end process;
```

2 Problem 5

```
----------------------------------------------------------------------------------
-- Author:         Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:    07-05-2016
-- Module Name:    p5.vhd
----------------------------------------------------------------------------------


process (sel, sel_2, sel_3, a, b)
begin
        if sel = '1' then
                f <= a;
                if sel_2 = '1' then
                        g <= not a;
                else
                        g <= not b;
                        if sel_3 = '1' then
                                g <= a xor b;
                        end if;
                end if;
        else
                if sel_2 = '1' then
                        g <= a and b;
                else
                        if sel_3 = '1' then
                                g <= a nand b;
```

```
                        -- preventing from transparent latch creation
                        else
                                g <= ...;
                        end if;
                end if;
                f <= b;
        end if;
end process;
```

3 Problem 6

3.1 A

If we use signals in clock sensitive process, we create flipflop. For variables we have 2 modes:
1) Just wire

```
----------------------------------------------------------------------------------
-- Author:         Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:    07-05-2016
-- Module Name:    p6-1-1.vhd
----------------------------------------------------------------------------------

process (clk)
        variable sum : integer;
begin
        sum := a + b;
        q <= sum;
end process;
```

2) FlipFlop

```
----------------------------------------------------------------------------------
-- Author:         Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:    07-05-2016
-- Module Name:    p6-1-2.vhd
----------------------------------------------------------------------------------

process (clk)
        variable sum : integer;
begin
        q <= sum;
        sum := a + b;
end process;
```

3.2 B

4 FlipFlop

3.3 C

1 FlipFlop

4 Problem 7

```
----------------------------------------------------------------------------------
-- Author:         Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:    05-05-2016
-- Module Name:    p7.vhd
----------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

entity p7 is
end entity;

architecture rtl of p7 is
        component ring_counter
                generic (N : integer := 4);
                port (clk, start : in std_logic;
                        Q : out std_logic_vector(N - 1 downto 0));
        end component;

        for all:ring_counter use entity work.ring_counter;

        signal clk : std_logic := '0';
        signal start : std_logic := '1';
        signal Q : std_logic_vector(3 downto 0);
begin
        clk <= not clk after 50 ns;

        start <= '0' after 75 ns;

        m:ring_counter port map (clk, start, Q);
end architecture;

----------------------------------------------------------------------------------
-- Author:         Parham Alvani (parham.alvani@gmail.com)
```

```
--
-- Create Date:   05-05-2016
-- Module Name:   d-flipflop.vhd
----------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

entity d_flipflop is
        port ( clk, reset, preset : in std_logic;
                d : in std_logic;
                q, qbar : out std_logic);
end entity d_flipflop;

architecture rtl of d_flipflop is
        signal b : std_logic;
begin
        process (clk)
        begin
                if clk = '1' and clk'event then
                        if reset = '1' then
                                b <= '0';
                        elsif preset = '1' then
                                b <= '1';
                        else
                                b <= d;
                        end if;
                end if;
        end process;
        q <= b;
        qbar <= not b;
end architecture;

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

entity ring_counter is
        generic (N : integer := 4);
        port (clk, start : in std_logic;
```

```
                Q : out std_logic_vector(N - 1 downto 0));
end entity ring_counter;

architecture rtl of ring_counter is
        component d_flipflop
                port ( clk, reset, preset : in std_logic;
                        d : in std_logic;
                        q, qbar : out std_logic);
        end component;

        for all:d_flipflop use entity work.d_flipflop;

        signal b : std_logic_vector (N - 1 downto 0);
        signal bbar : std_logic_vector (N - 1 downto 0);
begin
        dff:d_flipflop port map (clk, '0', start, b(N - 1), b(0), bbar(0));
        dffsg: for I in 1 to N - 1 generate
                dffs:d_flipflop port map (clk, start, '0', b(I - 1), b(I), bbar(I));
        end generate;
        Q <= b;
end architecture;
```

5 Problem 8

```
--------------------------------------------------------------------------------
-- Author:        Parham Alvani (parham.alvani@gmail.com)
--
-- Create Date:   05-05-2016
-- Module Name:   p8.vhd
--------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

entity p8 is
        port (clk, reset_a, reset_s : in std_logic;
                d : in std_logic;
                q : out std_logic);
end entity p8;

architecture rtl of p8 is
begin
        -- the process sensitivity list must contain
        -- clock and asynchronous signals
```

```
        process (clk, reset_a)
        begin
                -- asynchronous reset, it must come
                -- before clock
                if reset_a = 1 then
                        q <= '0';
                elsif clk'event and clk = '1' then
                        -- synchronous reset
                        if reset_s = '0' then
                                q <= '0';
                        else
                                q <= d;
                        end if;
                end if;
        end
end architecture;
```

6 Problem 9

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

entity vending_machine is
        port(coin_in : in std_logic;
                coin_in_1 : in std_logic;
                coin_in_10 : in std_logic;
                coin_in_100 : in std_logic;
                buy_in : in std_logic;
                price : in std_logic_vector(7 downto 0);
                coin_return : out std_logic;
                coin_return_1 : out std_logic_vector(7 downto 0);
                coin_return_10 : out std_logic_vector(7 downto 0);
                coin_return_100 : out std_logic_vector(7 downto 0);
                clk : in std_logic);
end entity;
```

```
architecture rtl of vending_machine is
        type state is (COIN_IN_S, COIN_RETURN_100_S, COIN_RETURN_10_S, COIN_RETURN_1_S);

        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 (coin_in, buy_in, current_state)
                variable coin_return_total : std_logic_vector(7 downto 0);
                variable coin_return_100_var : std_logic_vector(7 downto 0);
                variable coin_return_10_var : std_logic_vector(7 downto 0);
                variable coin_return_1_var : std_logic_vector(7 downto 0);
                variable coin_in_total : std_logic_vector(7 downto 0);
        begin
                case current_state is
                        when COIN_IN_S =>
                                if coin_in = '1' then
                                        coin_return <= '0';
                                        next_state <= COIN_IN_S;
                                        if coin_in_1 = '1' and coin_in_10 = '0' and coin_i
                                                coin_in_total := coin_in_total + "00000001
                                        elsif coin_in_1 = '0' and coin_in_10 = '1' and coi
                                                coin_in_total := coin_in_total + "00001010
                                        elsif coin_in_1 = '0' and coin_in_10 = '0' and coi
                                                coin_in_total := coin_in_total + "01100100
                                        end if;
                                end if;
                                if buy_in = '1' then
                                        coin_return_total := coin_in_total - price;
                                        coin_return_100_var := "00000000";
                                        coin_return_10_var := "00000000";
                                        coin_return_1_var := "00000000";
                                        next_state <= COIN_RETURN_100_S;
                                end if;
                        when COIN_RETURN_100_S =>
                                if coin_return_total >= "01100100" then
                                        coin_return_total := coin_return_total - "01100100
                                        coin_return_100_var := coin_return_100_var + "0000
                                        next_state <= COIN_RETURN_100_S;
                                else
```

```
                                        next_state <= COIN_RETURN_10_S;
                                end if;
                        when COIN_RETURN_10_S =>
                                if coin_return_total >= "00001010" then
                                        coin_return_total := coin_return_total - "00001010
                                        coin_return_10_var := coin_return_10_var + "000000
                                        next_state <= COIN_RETURN_10_S;
                                else
                                        next_state <= COIN_RETURN_1_S;
                                end if;
                        when COIN_RETURN_1_S =>
                                if coin_return_total >= "00000001" then
                                        coin_return_total := coin_return_total - "00000001
                                        coin_return_1_var := coin_return_1_var + "00000001
                                        next_state <= COIN_RETURN_1_S;
                                else
                                        next_state <= COIN_IN_S;
                                        coin_return <= '1';
                                        coin_return_1 <= coin_return_1_var;
                                        coin_return_10 <= coin_return_10_var;
                                        coin_return_100 <= coin_return_100_var;
                                end if;
                end case;
        end process;
end architecture;

Copyright 1986-2015 Xilinx, Inc. All Rights Reserved.
------------------------------------------------------------------------------------
| Tool Version : Vivado v.2015.4 (lin64) Build 1412921 Wed Nov 18 09:44:32 MST 2015
| Date         : Fri May  6 17:30:19 2016
| Host         : parham-xilinx running 64-bit Ubuntu 16.04 LTS
| Command      : report_utilization -file vending_machine_utilization_synth.rpt -pb vendin
| Design       : vending_machine
| Device       : 7z020clg484-1
| Design State : Synthesized
------------------------------------------------------------------------------------

Utilization Design Information

Table of Contents
-----------------
1. Slice Logic
1.1 Summary of Registers by Type
2. Memory
```

3. DSP
4. IO and GT Specific
5. Clocking
6. Specific Feature
7. Primitives
8. Black Boxes
9. Instantiated Netlists

1. Slice Logic

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	80	0	53200	0.15
LUT as Logic	80	0	53200	0.15
LUT as Memory	0	0	17400	0.00
Slice Registers	69	0	106400	0.06
Register as Flip Flop	2	0	106400	<0.01
Register as Latch	67	0	106400	0.06
F7 Muxes	0	0	26600	0.00
F8 Muxes	0	0	13300	0.00

* Warning! The Final LUT count, after physical optimizations and full implementation, is t

1.1 Summary of Registers by Type

Total	Clock Enable	Synchronous	Asynchronous
0	_	-	-
0	_	-	Set
0	_	-	Reset
0	_	Set	-
0	_	Reset	-
0	Yes	-	-
0	Yes	-	Set
67	Yes	-	Reset
0	Yes	Set	-
2	Yes	Reset	-

2. Memory

Site Type	Used	Fixed	Available	Util%
Block RAM Tile	0	0	140	0.00
RAMB36/FIFO*	0	0	140	0.00
RAMB18	0	0	280	0.00

* Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodat

3. DSP

Site Type	Used	Fixed	Available	Util%
DSPs	0	0	220	0.00

4. IO and GT Specific

Site Type	Used	Fixed	Available	Util%
Bonded IOB	39	0	200	19.50
Bonded IPADs	0	0	2	0.00
Bonded IOPADs	0	0	130	0.00
PHY_CONTROL	0	0	4	0.00
PHASER_REF	0	0	4	0.00
OUT_FIFO	0	0	16	0.00
IN_FIFO	0	0	16	0.00
IDELAYCTRL	0	0	4	0.00
IBUFGDS	0	0	192	0.00
PHASER_OUT/PHASER_OUT_PHY	0	0	16	0.00
PHASER_IN/PHASER_IN_PHY	0	0	16	0.00
IDELAYE2/IDELAYE2_FINEDELAY	0	0	200	0.00
ILOGIC	0	0	200	0.00
OLOGIC	0	0	200	0.00

+----------------------------+------+-------+-----------+-------+

5. Clocking

Site Type	Used	Fixed	Available	Util%
BUFGCTRL	1	0	32	3.13
BUFIO	0	0	16	0.00
MMCME2_ADV	0	0	4	0.00
PLLE2_ADV	0	0	4	0.00
BUFMRCE	0	0	8	0.00
BUFHCE	0	0	72	0.00
BUFR	0	0	16	0.00

6. Specific Feature

Site Type	Used	Fixed	Available	Util%
BSCANE2	0	0	4	0.00
CAPTUREE2	0	0	1	0.00
DNA_PORT	0	0	1	0.00
EFUSE_USR	0	0	1	0.00
FRAME_ECCE2	0	0	1	0.00
ICAPE2	0	0	2	0.00
STARTUPE2	0	0	1	0.00
XADC	0	0	1	0.00

7. Primitives

Ref Name	Used	Functional Category
LDCE	67	Flop & Latch
LUT6	40	LUT

```
| OBUF     |   25 |                  IO |
| LUT5     |   23 |                 LUT |
| LUT4     |   19 |                 LUT |
| LUT3     |   14 |                 LUT |
| IBUF     |   14 |                  IO |
| LUT2     |    6 |                 LUT |
| LUT1     |    3 |                 LUT |
| FDRE     |    2 |        Flop & Latch |
| CARRY4   |    2 |          CarryLogic |
| BUFG     |    1 |               Clock |
+----------+------+---------------------+


8. Black Boxes
--------------

+----------+------+
| Ref Name | Used |
+----------+------+


9. Instantiated Netlists
------------------------

+----------+------+
| Ref Name | Used |
+----------+------+

Copyright 1986-2015 Xilinx, Inc. All Rights Reserved.
-------------------------------------------------------------------------------------------
| Tool Version : Vivado v.2015.4 (lin64) Build 1412921 Wed Nov 18 09:44:32 MST 2015
| Date         : Fri May  6 17:21:56 2016
| Host         : parham-xilinx running 64-bit Ubuntu 16.04 LTS
| Command      : report_utilization -file vending_machine_utilization_synth.rpt -pb vendin
| Design       : vending_machine
| Device       : 7z020clg484-1
| Design State : Synthesized
-------------------------------------------------------------------------------------------

Utilization Design Information

Table of Contents
-----------------
1. Slice Logic
```

1. Slice Logic

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	112	0	53200	0.21
LUT as Logic	112	0	53200	0.21
LUT as Memory	0	0	17400	0.00
Slice Registers	69	0	106400	0.06
Register as Flip Flop	2	0	106400	<0.01
Register as Latch	67	0	106400	0.06
F7 Muxes	2	0	26600	<0.01
F8 Muxes	0	0	13300	0.00

* Warning! The Final LUT count, after physical optimizations and full implementation, is t

1.1 Summary of Registers by Type

Total	Clock Enable	Synchronous	Asynchronous
0	_	-	-
0	_	-	Set
0	_	-	Reset
0	_	Set	-
0	_	Reset	-
0	Yes	-	-
0	Yes	-	Set
67	Yes	-	Reset
0	Yes	Set	-
2	Yes	Reset	-

+-------+-------------+------------+-------------+

2. Memory

+----------------+------+-------+-----------+-------+
| Site Type | Used | Fixed | Available | Util% |
+----------------+------+-------+-----------+-------+
Block RAM Tile	0	0	140	0.00
RAMB36/FIFO*	0	0	140	0.00
RAMB18	0	0	280	0.00
+----------------+------+-------+-----------+-------+
* Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodat

3. DSP

+-----------+------+-------+-----------+-------+
| Site Type | Used | Fixed | Available | Util% |
+-----------+------+-------+-----------+-------+
| DSPs | 0 | 0 | 220 | 0.00 |
+-----------+------+-------+-----------+-------+

4. IO and GT Specific

+-----------------------------+------+-------+-----------+-------+
| Site Type | Used | Fixed | Available | Util% |
+-----------------------------+------+-------+-----------+-------+
Bonded IOB	39	0	200	19.50
Bonded IPADs	0	0	2	0.00
Bonded IOPADs	0	0	130	0.00
PHY_CONTROL	0	0	4	0.00
PHASER_REF	0	0	4	0.00
OUT_FIFO	0	0	16	0.00
IN_FIFO	0	0	16	0.00
IDELAYCTRL	0	0	4	0.00
IBUFGDS	0	0	192	0.00
PHASER_OUT/PHASER_OUT_PHY	0	0	16	0.00
PHASER_IN/PHASER_IN_PHY	0	0	16	0.00
IDELAYE2/IDELAYE2_FINEDELAY	0	0	200	0.00

| ILOGIC | 0 | 0 | 200 | 0.00 |
| OLOGIC | 0 | 0 | 200 | 0.00 |

5. Clocking

Site Type	Used	Fixed	Available	Util%
BUFGCTRL	1	0	32	3.13
BUFIO	0	0	16	0.00
MMCME2_ADV	0	0	4	0.00
PLLE2_ADV	0	0	4	0.00
BUFMRCE	0	0	8	0.00
BUFHCE	0	0	72	0.00
BUFR	0	0	16	0.00

6. Specific Feature

Site Type	Used	Fixed	Available	Util%
BSCANE2	0	0	4	0.00
CAPTUREE2	0	0	1	0.00
DNA_PORT	0	0	1	0.00
EFUSE_USR	0	0	1	0.00
FRAME_ECCE2	0	0	1	0.00
ICAPE2	0	0	2	0.00
STARTUPE2	0	0	1	0.00
XADC	0	0	1	0.00

7. Primitives

Ref Name	Used	Functional Category

LDCE	67	Flop & Latch
LUT6	52	LUT
OBUF	25	IO
LUT5	24	LUT
LUT3	16	LUT
LUT4	14	LUT
IBUF	14	IO
LUT2	6	LUT
MUXF7	2	MuxFx
FDRE	2	Flop & Latch
CARRY4	2	CarryLogic
BUFG	1	Clock

8. Black Boxes

Ref Name	Used

9. Instantiated Netlists

Ref Name	Used