EEL 4712
Midterm 2 – Spring 2015
VERSION 1

Name: Solution

UFID: _____

Sign here to give permission for your test to be returned in class, where others might see your score:

_____

IMPORTANT:
- Please be neat and write (or draw) carefully. If we cannot read it with a reasonable effort, it is assumed wrong.
- **As always, the best answer gets the most points.**

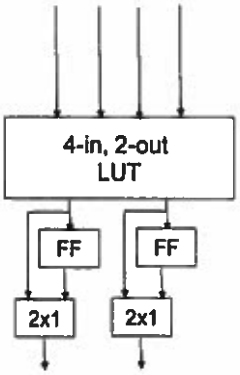
COVER SHEET:

Problem#:	Points
1 (12 points)	
2 (6 points)	
4 (6 points)	
5 (6 points)	
6 (6 points)	
6 (6 points)	
7 (6 points)	
8 (18 points)	
9a (17 points)	
9b (17 points)	

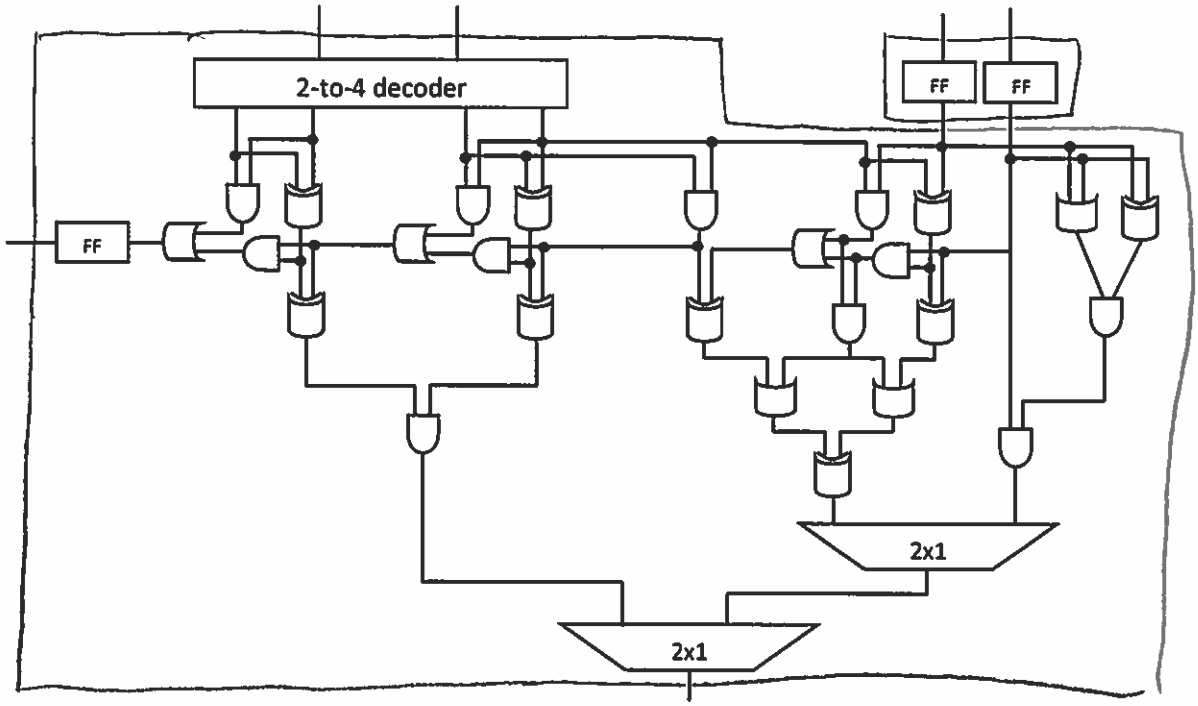
Total:

Regrade Info:

1) (12 points) Assume you are given an FPGA that consists of the following CLB structures:



Map the following circuit onto these CLBs by drawing rectangles to represent CLBs. Use the minimum number of CLBs.



2) (6 points) Given a 10-input, 2-output combinational logic function, how many SRAM bits are necessary to implement this function in a single lookup table (LUT)? Show your work.

$2^{10} * 2 = \boxed{2^{11}}$

3) (6 points) Briefly describe the purpose of a connection box.

connects CLB I/O to routing tracks

4) (6 points) Briefly describe the purpose of a switch box.

connects routing tracks from different channels

5) (6 points) Long channels in FPGA provide which of the following advantages (select one):
   (a) Increased data width
   (b) Direct connections between block RAM and DSP units
   (c) Shorter propagation delays between distant resources
   (d) PCIe controller logic

6) (6 points) In the VGA lab, your implementation drew four pixels on the screen (a 2x2 block) for every image pixel. Define the row address and column address logic as a function of vCount and hCount for drawing a 16x16 block for each image pixel. Assume the image should be displayed at the top-left portion of the screen and that hCount and vCount are in the proper range.

rowAddr = vCount/16

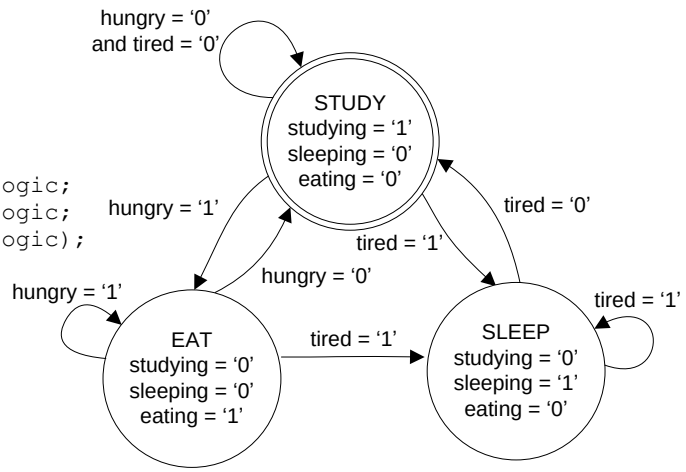
columnAddr = hCount/16

7) (6 points) You are designing a new monitor interface that is similar to VGA, where an h_sync signal is held low for specific amount of time. If the maximum allowable error for this signal is 80 ns, what is the slowest possible clock frequency you can use for this interface? List any assumptions.

max error = 1 cycle

$\frac{1\text{ cycle}}{80\text{ ns}} = \boxed{12.5\text{ MHz}}$

8) (18 points) Fill in the code to implement the following Moore finite state machine (FSM), *using the 2-process FSM model*. **Assume that input *tired* always takes priority over *hungry* when there is an option between two state transitions.** Assume that STUDY is the initial state. Use the next page if extra room is needed.

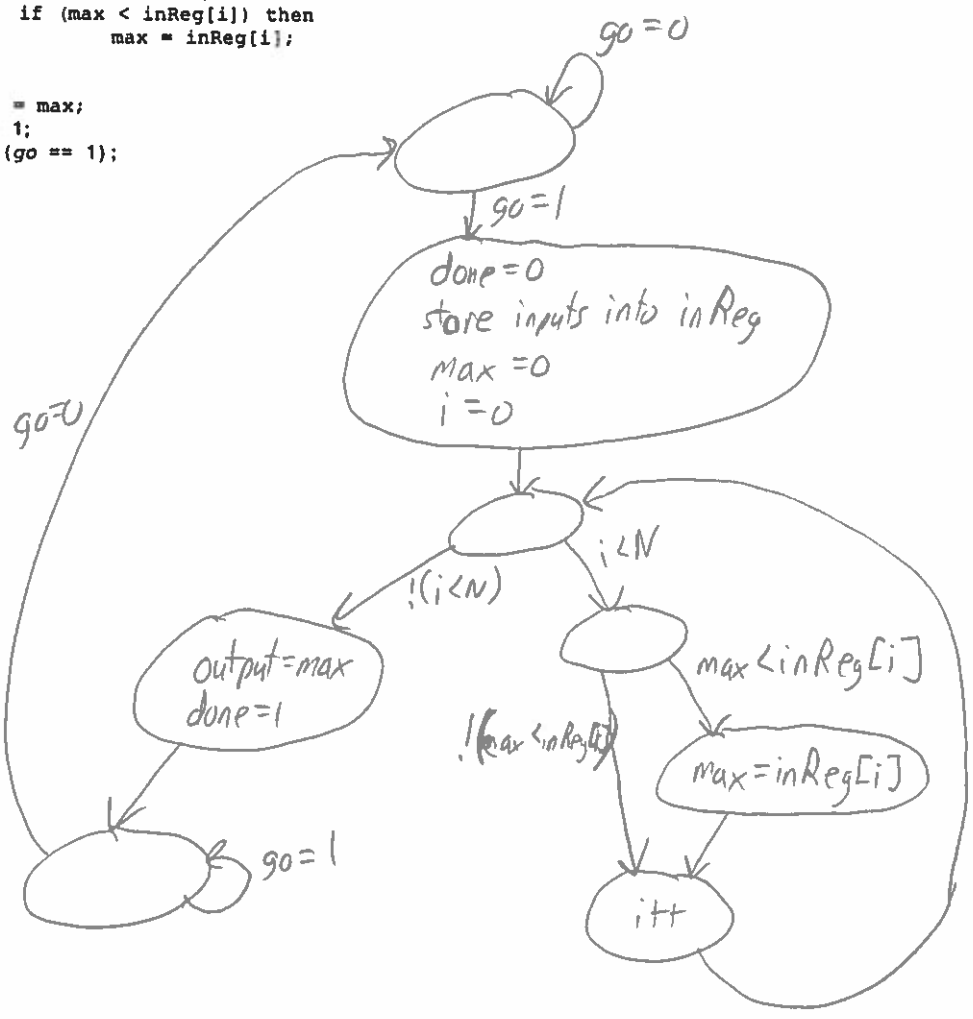



```
library ieee;
use ieee.std_logic_1164.all;

entity fsm is
    port (
        clk, rst                    : in  std_logic;
        hungry, tired               : in  std_logic;
        studying, sleeping, eating : out std_logic);
end fsm;

architecture PROC2 of fsm is

    type STATE_TYPE is (STUDY, EAT, SLEEP);
    signal state, next_state : STATE_TYPE;

begin

    process(clk, rst)
    begin
        if (rst = '1') then
            state <= STUDY;
        elsif (clk'event and clk = '1') then
            state <= next_state;
        end if;
    end process;

    process(hungry, tired, state)
    begin

        studying   <= '0';
        eating     <= '0';
        sleeping   <= '0';
        next_state <= state;

        case state is
            when STUDY =>
                studying <= '1';
                if (tired = '1') then
                    next_state <= SLEEP;
                elsif (hungry = '1') then
                    next_state <= EAT;
                end if;

            when EAT =>
                eating <= '1';
                if (tired = '1') then
                    next_state <= SLEEP;
                elsif (hungry = '0') then
                    next_state <= STUDY;
                end if;

            when SLEEP =>
                sleeping <= '1';
                if (tired = '0') then
                    next_state <= STUDY;
```

```
                end if;

            when others =>
                null;
        end case;
    end process;
end PROC2;
```

9) a. (17 points) Create an FSMD that implements the following pseudo-code. **Do not write VHDL and instead leave the FSMD in graphical form** (i.e., state machine with corresponding operations in each state). Make sure to specify all operations and state transitions. Note that *output*, *go*, *input*, and *done* are I/O. Assume that *input* and *inReg* both use the following array type in vhdl: `type input_array is array (0 to N-1) of unsigned(31 downto 0)`

```
const int N = 8; // In VHDL: generic( N : positive )

Inputs: go (std_logic), input (input_array)
Outputs: output (std_logic), done (std_logic)

int i, max;
int inReg[N]; // In VHDL: signal inReg : input_array;

// reset values for outputs
done = 0; output = 0;

while (1) {

        while (go == 0);
        done = 0;
        // Store N inputs from "input" into N registers "inReg"
        // NOTE: this can be done in one cycle, so your FSMD doesn't need a loop here.
        for (i=0; i < N; i++)
                inReg[i] = input[i];

        // calculate max value across all N inputs
        max = 0;
        for (i=0; i < N; i++) {
                if (max < inReg[i]) then
                        max = inReg[i];
        }

        output = max;
        done = 1;
        while (go == 1);
}
```



b. (17 points) Draw an FSM capable of controlling the illustrated datapath to perform the pseudo-code in part *a*, by assigning or reading from the underlined control signals. Assume that *go* is an input to the controller and that left mux inputs have a select value of 1. Note that this datapath assumes that *N=4*. **Do not write any VHDL code, just show the FSM and control signals. Be sure to mention default signal values to save space.**