EEL 4712
Midterm 2 – Spring 2012
VERSION 1

Name: Solution

UFID: _____

Sign your name here if you would like for your test to be returned in class:

_____

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 (15 points)	
3 (14 points)	
4 (6 points)	
5 (6 points)	
6a (16 points)	
6b (16 points)	
6c (15 points)	

Total:

Regrade Info:

```
ENTITY _entity_name IS
PORT(__input_name, __input_name : IN STD_LOGIC;
__input_vector_name : IN STD_LOGIC_VECTOR(__high downto __low);
__bidir_name, __bidir_name : INOUT STD_LOGIC;
__output_name, __output_name : OUT STD_LOGIC);
END __entity_name;

ARCHITECTURE a OF __entity_name IS
SIGNAL __signal_name : STD_LOGIC;
BEGIN
-- Process Statement
-- Concurrent Signal Assignment
-- Conditional Signal Assignment
-- Selected Signal Assignment
-- Component Instantiation Statement
END a;

__instance_name: __component_name PORT MAP (__component_port => __connect_port,
__component_port => __connect_port);

WITH __expression SELECT
__signal <= __expression WHEN __constant_value,
__expression WHEN __constant_value,
__expression WHEN __constant_value,
__expression WHEN __constant_value;
__signal <= __expression WHEN __boolean_expression ELSE
__expression WHEN __boolean_expression ELSE
__expression;

IF __expression THEN
__statement;
__statement;
ELSIF __expression THEN
__statement;
__statement;
ELSE
__statement;
__statement;
END IF;

CASE __expression IS
WHEN __constant_value =>
__statement;
__statement;
WHEN __constant_value =>
__statement;
__statement;
WHEN OTHERS =>
__statement;
__statement;
END CASE;

<generate_label>: FOR <loop_id> IN <range> GENERATE
-- Concurrent Statement(s)
END GENERATE;


type __identifier is type_definition;

subtype __identifier is subtype_indication;
```

1) a. (3 points) Write a VHDL type declaration called MY_ARRAY that creates a 1D array with 50 elements, where each element is a 32-bit std_logic_vector. Alternatively, state that VHDL pre-2008 doesn't support this type.

type MY_ARRAY is (0 to 49) of std_logic_vector (31 downto 0);

b. (3 points) Write a VHDL type declaration called MY_ARRAY that creates a 2D array with 640x480 elements, where each element is a 16-bit std_logic_vector. Alternatively, state that VHDL pre-2008 doesn't support this type.

type MY_ARRAY is array (0 to 479, 0 to 639) of std_logic_vector(15 downto 0);

c. (3 points) Write a VHDL type declaration called MY_ARRAY with an unconstrained range, where each element is a 16-bit std_logic_vector. Alternatively, state that VHDL pre-2008 doesn't support this type.

type MY_ARRAY is array (natural range <>) of std_logic_vector(15 downto 0);

d. (3 points) Write a VHDL type declaration called MY_ARRAY with 25 elements, where each element is an unconstrained std_logic_vector. Alternatively, state that VHDL pre-2008 doesn't support this type.

not supported

2) a. (5 points) For the VGA lab, you were required to display the image in different locations. Assuming that the top-left pixel position of the image is represented as (X,Y), define a formula that translates the current pixel position (vcount, hcount) being drawn on the monitor with the pixel position in the image. In other words, show the equations for the row and column address translation. Recall that each image pixel is drawn on the monitor as 8x8 pixels.

row addr = (vcount - X) >> 3

col addr = (hcount - Y) >> 3

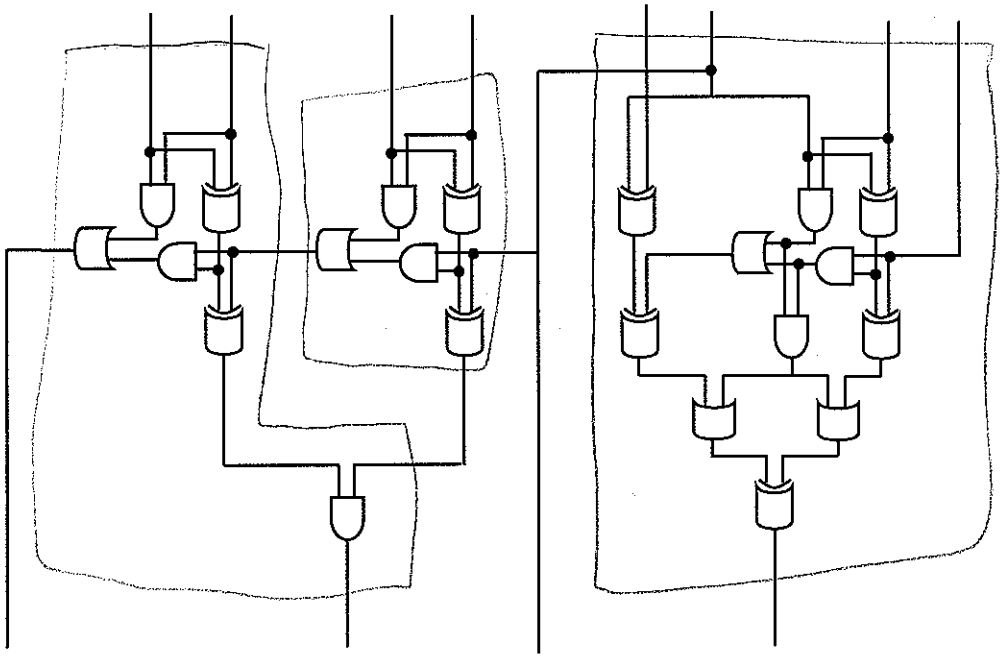
b. (5 points) How many possible colors can be used with the VGA interface used in lab? Explain your answer.

3 binary color signals, $2^3 = \boxed{8}$

c. (5 points) The ROM used in lab6 has a 1-cycle read latency. What changes in the rest of the circuit were required to account for this latency?

hsync, vsync, possibly enable signals must be delayed by 1 cycle

3) (14 points) Map the following circuit onto 4-input, 2-output LUTs by drawing shapes around each portion of the circuit that is mapped to an individual LUT. *Each input to the overall circuit is shown at the top and every output is at the bottom. Be careful not to miss any.* Use the minimum number of LUTs.



4) (6 points) In addition to LUTs and CLBs, what are 2 common resources provided by commercial FPGAs? Do not mention interconnect resources.

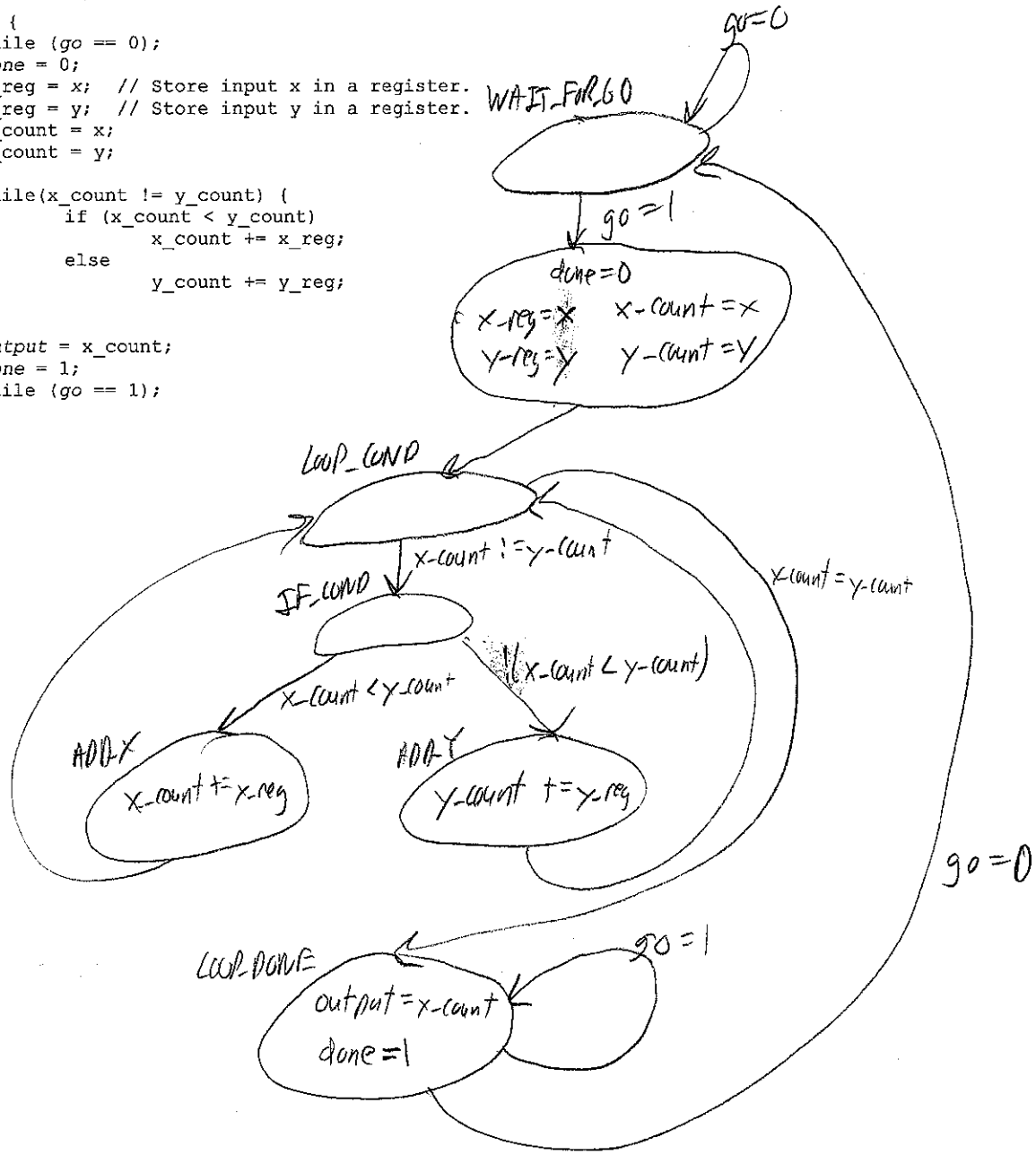
DSPs, block RAM

5) (6 points) ***Name*** the 3 reconfigurable interconnect resources provided by FPGAs that enable connections between LUTs and CLBs.

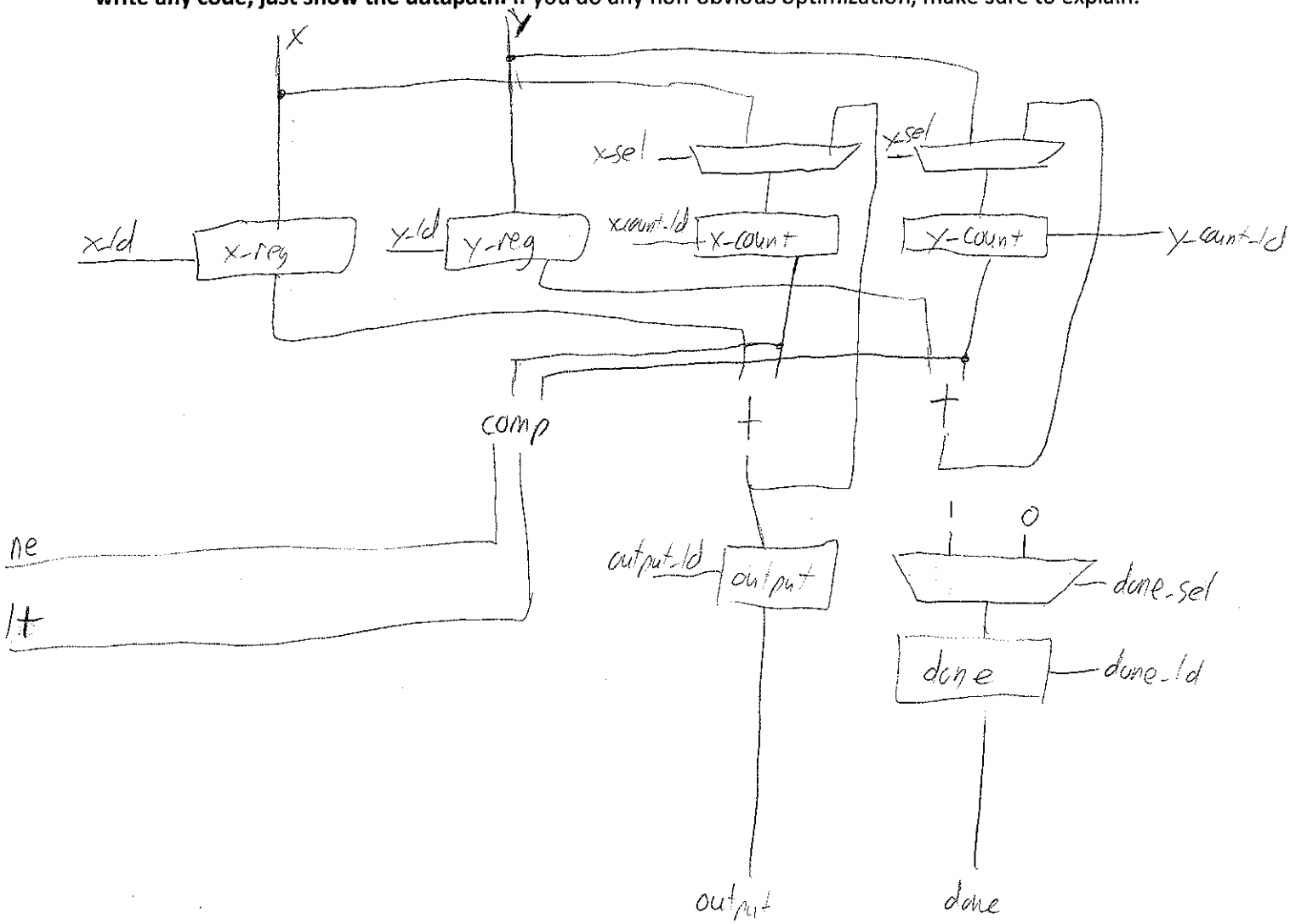
tracks, connection boxes, switch boxes

6) a. (16 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 *x*, *y*, *output*, *go*, and *done* are I/O.

```
Inputs: go, x, y
Outputs: output, done

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

while (1) {
        while (go == 0);
        done = 0;
        x_reg = x;  // Store input x in a register.
        y_reg = y;  // Store input y in a register.
        x_count = x;
        y_count = y;

        while(x_count != y_count) {
                if (x_count < y_count)
                        x_count += x_reg;
                else
                        y_count += y_reg;
        }

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



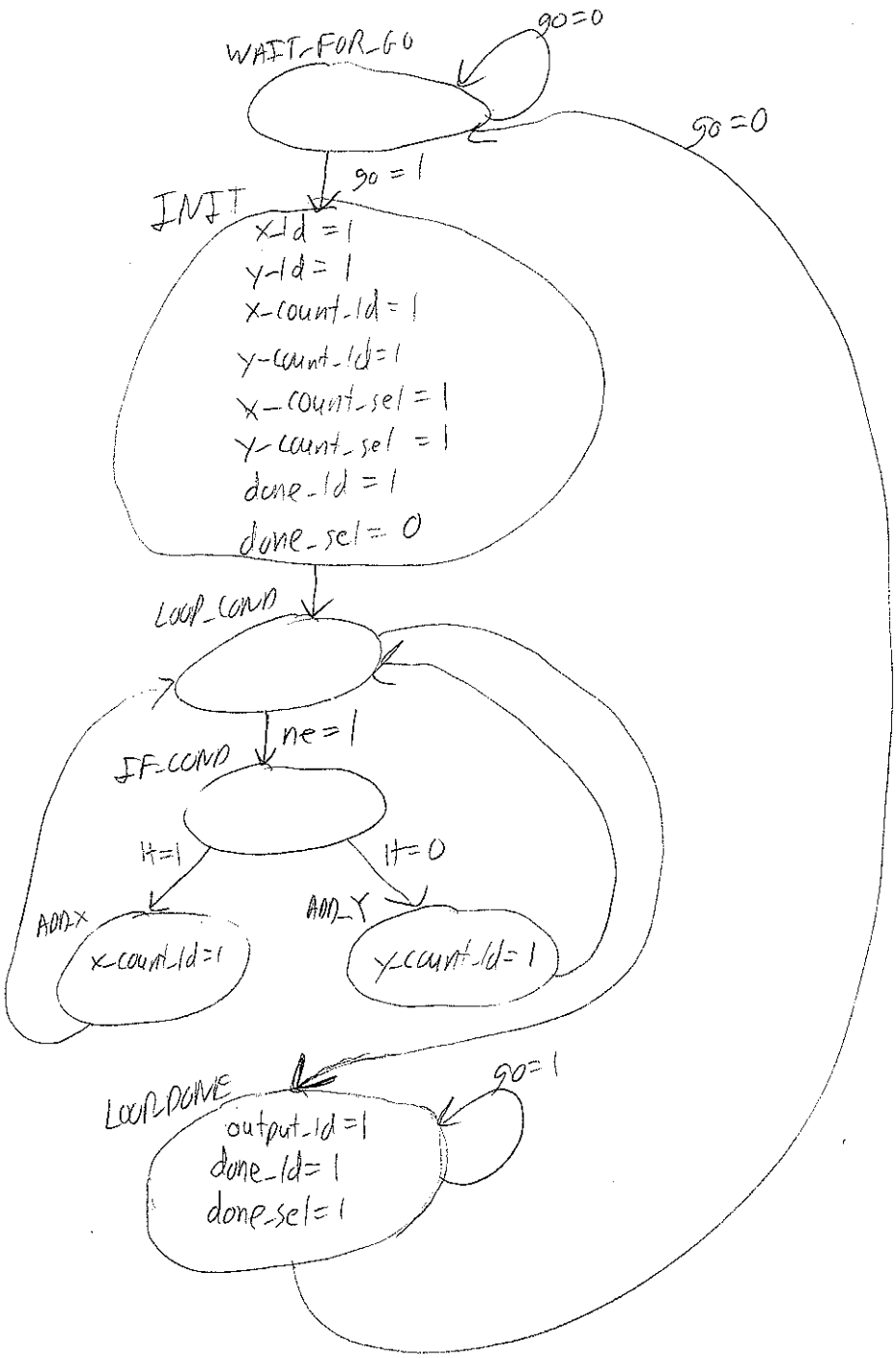
b. (16 points) For the same pseudo-code, create a datapath capable of executing the code (ignore the controller in this step). Make sure to show all control signals (i.e., mux select signals, register load signals, comparator output signals). Hint: to make things easier, do not try to share resources. **Do not write any code, just show the datapath.** If you do any non-obvious optimization, make sure to explain.



15

c. (~~12~~ points) For the datapath in the previous step, draw an FSM capable of controlling the datapath to perform the pseudo-code. In each state of the FSM, show the values of your control signals from the previous step that configure the ~~FSM~~ datapath to do the corresponding operations. Hint: to save yourself time, try to use the same states as the FSMD, and just change the operations to the corresponding control signals. **Do not write any code, just show the FSM and control signals.** Mention defaults.

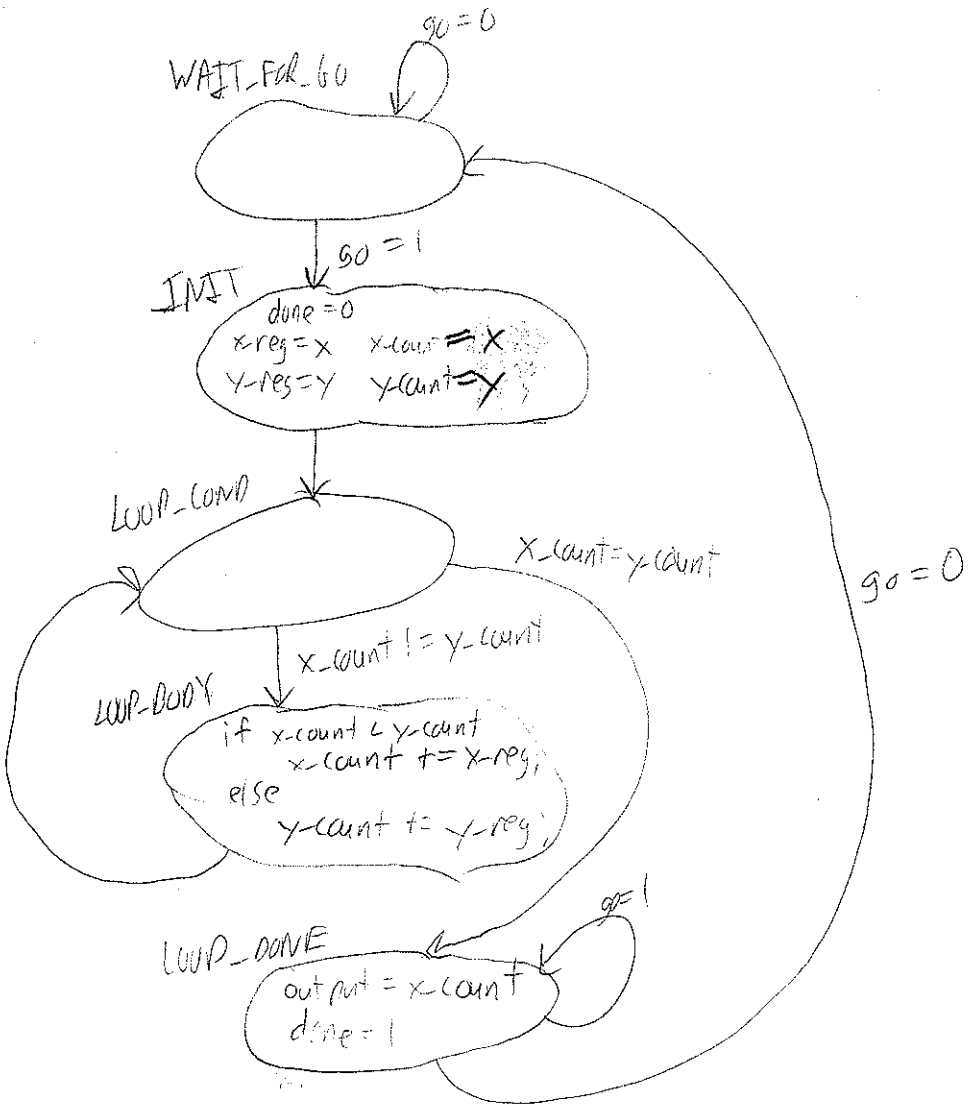
Defaults:
all ld signals = 0
all sel signals = 0



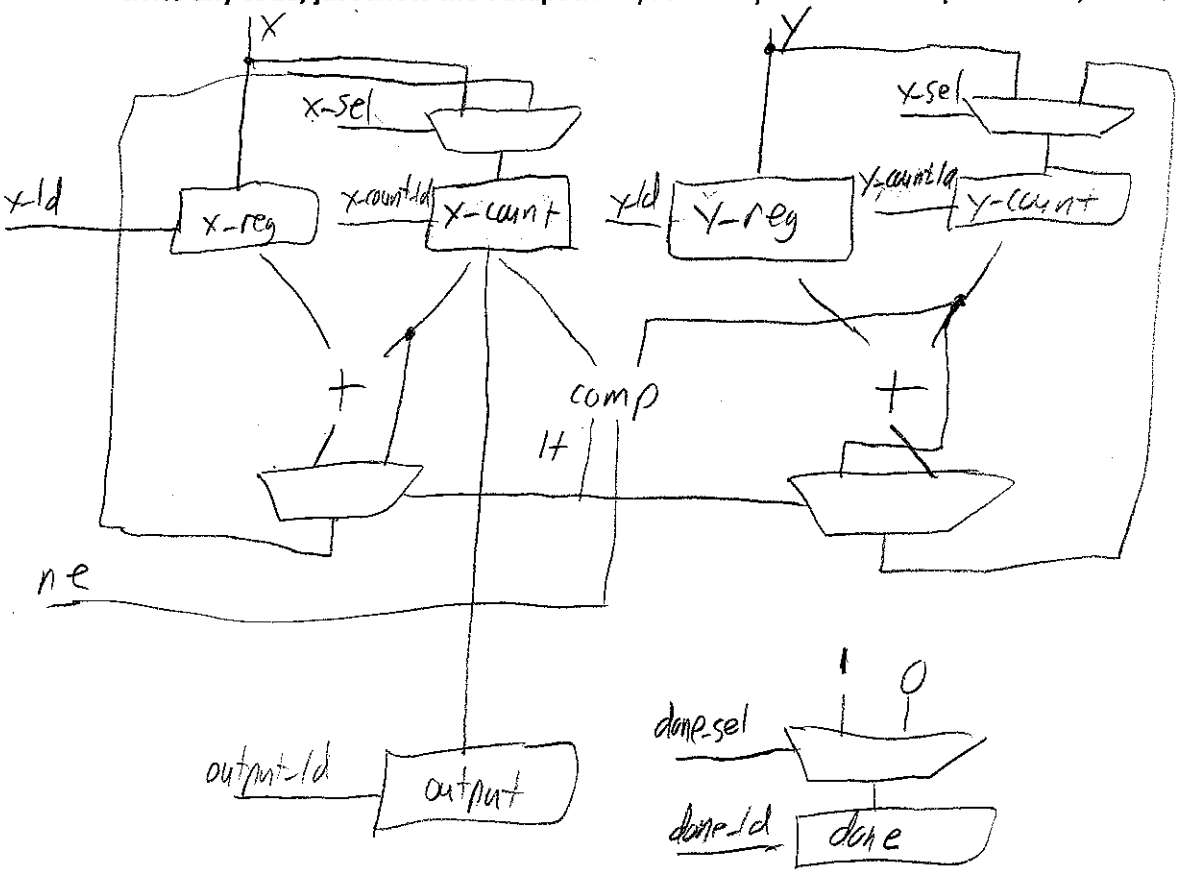
Alternate answer

6) a. (16 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 *x*, *y*, *output*, *go*, and *done* are I/O.

```
Inputs: go, x, y (width specified by generic width)
Outputs: output (width specified by generic width), done (1 bit)

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

while (1) {
        while (go == 0);
        done = 0;
        x_reg = x;  // Store input x in a register.
        y_reg = y;  // Store input y in a register.

        x_count = x_reg;
        y_count = y_reg;

        while(x_count != y_count) {
                if (x_count < y_count)
                        x_count += x_reg;
                else
                        y_count += y_reg;
        }

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



b. (16 points) For the same pseudo-code, create a datapath capable of executing the code (ignore the controller in this step). Make sure to show all control signals (i.e., mux select signals, register load signals, comparator output signals). Hint: to make things easier, do not try to share resources. **Do not write any code, just show the datapath.** If you do any non-obvious optimization, make sure to explain.



c. (15 points) For the datapath in the previous step, draw an FSM capable of controlling the datapath to perform the pseudo-code. In each state of the FSM, show the values of your control signals from the previous step that configure the datapath to do the corresponding operations. Hint: to save yourself time, try to use the same states as the FSMD, and just change the operations to the corresponding control signals. **Do not write any code, just show the FSM and control signals. Be sure to mention default signal values to save space.**

Defaults:
all ld signals = 0
all sel signals = 0