EEL 4712
Midterm 3 – Spring 2013
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:

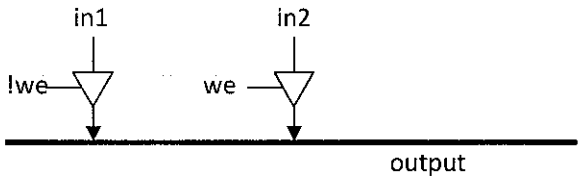
Problem#:	Points
1 (14 points)	
2 (7 points)	
3 (7 points)	
4 (7 points)	
5 (7 points)	
6 (7 points)	
7 (14 points)	
8 (6 points)	
9 (6 points)	
10 (21 points)	
11 (4 points)	4

Total:

Regrade Info:

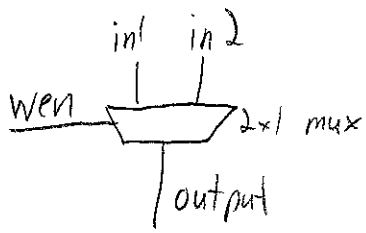
```
ENTITY _entity_name IS
PORT(__input_name, __input_name : IN STD_LOGIC;
__input_vector_name : IN STD_LOGIC_VECTOR(__high downto __low);
__bidir_name, __bidir_name : INOUT STD_LOGIC;
__output_name, __output_name : OUT STD_LOGIC);
END __entity_name;

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;
```

1) a. (7 points) Fill in the following ***behavioral*** architecture to implement the following bus.




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

entity small_bus is
  generic (
    width  :     positive := 8);
  port (
    in1    : in  std_logic_vector(width-1 downto 0);
    in2    : in  std_logic_vector(width-1 downto 0);
    wen    : in  std_logic;
    output : out std_logic_vector(width-1 downto 0));
end small_bus;

architecture BHV of small_bus is
begin

  process( in1, in2, wen )
  begin
        if (wen = '0') then
            output <= in1;
        else
            output <= in2;
        end if;

  end process;

end BHV;
```

b. (7 points) Show the synthesized structure when using an FPGA for the code in part a.

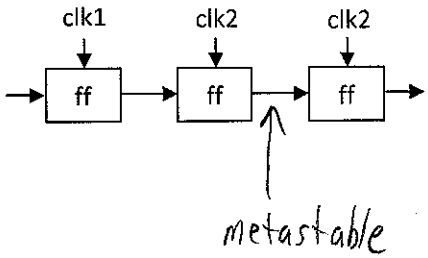


2) (7 points) Name two situations where the input to a flip-flop may change during the setup and hold window.

1) asynchronous inputs

2) clock domain crossing

3) (7 points) Show the location in the following circuit where metastability will occur.



4) (7 points) What synchronizer is most effective when trying to maximize bandwidth across clock domains?

FIFO

5) (7 points) All of the synchronizers discussed in class rely on the dual-flop synchronizer. Are these synchronizers guaranteed to always work? Explain why or why not.

No. There is a very small chance that a metastable signal will not stabilize in one cycle.

6) (7 points) Identify and describe the problem with the following combinational logic, which commonly occurs when creating 2-process FSMD entities. Note: the problem is not related to synthesis guidelines.

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

entity alu is
  generic (
    width  :     positive := 8);
  port (
    input1 : in  std_logic_vector(width-1 downto 0);
    input2 : in  std_logic_vector(width-1 downto 0);
    sel    : in  std_logic;
    output : out std_logic_vector(width-1 downto 0));
end alu;

architecture BHV of alu is

  signal feedback : std_logic_vector(width-1 downto 0);

begin

  process(input1, input2, sel, feedback)
  begin
    if (sel = '1') then
      feedback <= std_logic_vector(unsigned(input1)+unsigned(input2));
    else
      feedback <= std_logic_vector(unsigned(input1)+unsigned(feedback));
    end if;

    output <= feedback;

  end process;

end BHV;
```

the output defines an input, which creates a combinational loop

7) (14 points) Create a memory initialization file for the following assembly code. Add a comment to show the beginning of each instruction and each variable in memory. *You will likely need to break your answer up into two columns to fit on the page.*

```
OUTPORT0        EQU     $FFFE

BEGIN:
        LDAA    VALUE1
        STAR    D
        LDAA    VALUE2
        ANDR    D
        BEQA    THERE
        ADCR    D
THERE:
        ADCR    D
        STAA    OUTPORT0

INFINITE_LOOP:
        CLRC
        BCCA    INFINITE_LOOP

* Data Area
VALUE1: dc.b    $55
VALUE2: dc.b    $AA

        END     BEGIN
```

```
Depth = 256;
Width = 8;
Address_radix = hex;
Data_radix = hex;
% Program RAM Data %
Content
  Begin
0000: 88; //LDAA
0001: 14;
0002: 00;
0003: F1; //STAR
0004: 88; //LDAA
0005: 15;
0006: 00;
0007: 21; // ANDR
0008: B2; // BEQA
0009: 0C;
000A: 00;
000B: 01; //ADCR
000C: 01; //ADCR
000D: F6; //STAA
000E: FE;
000F: FF;
0010: F9; //CLRC
0011: B0; // BCCA
0012: 10;
0013: 00;
0014: 55; //VALUE1
0015: AA; //VALUE2
[       ..00FF] : 00;
End;
```

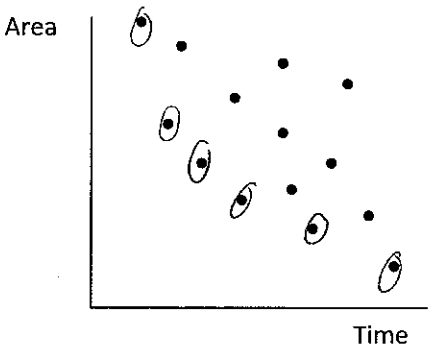
8) a. (3 points) Write a VHDL type declaration called MY_ARRAY that creates a 2D array with 800x600 elements, where each element is an 8-bit std_logic_vector. Alternatively, state that VHDL pre-2008 doesn't support this type.

type MY_ARRAY is array (0 to 799, 0 to 599) of std_logic_vector(7 downto 0);

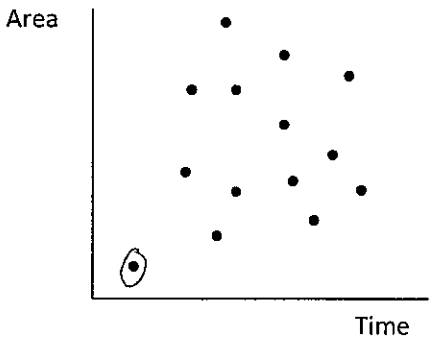
b. (3 points) When using an array type on the port definition of an entity, where do you need to define the type?

in a package that is included before the entity

9) a. (3 points) For the set of implementations shown below, circle the implementations that are Pareto optimal. List any assumptions.



b. (3 points) Do the same for the following set of implementations.



10) a. (7 points) For the following code, create a schedule for the provided datapath. Ignore muxes and other glue logic. Assume that address calculations are done *without* using the specified resources (i.e., address calculations cost nothing). Do not change the code.

```
for (int i=0; i < 1000000; i++) {
   a[i] = b[i]*5 + b[i+1]*12 + b[i+2]*55 + b[i+3]*42;
}
```

c d e f

Datapath
4 multipliers
3 adders
1 memory for b[] (can read 4 elements/cycle)
1 memory for a[] (can write 1 element/cycle)

cycle 1) i=0
2) i < 1,000,000, ld b[i .... i+3]
3) c, d, e, f
4) c+d, e+f
5) +
6) st a[i], i++

b. (7 points) Show the execution time in total cycles based on your schedule from part a.

1 + 5*1,000,000 = 5,000,001

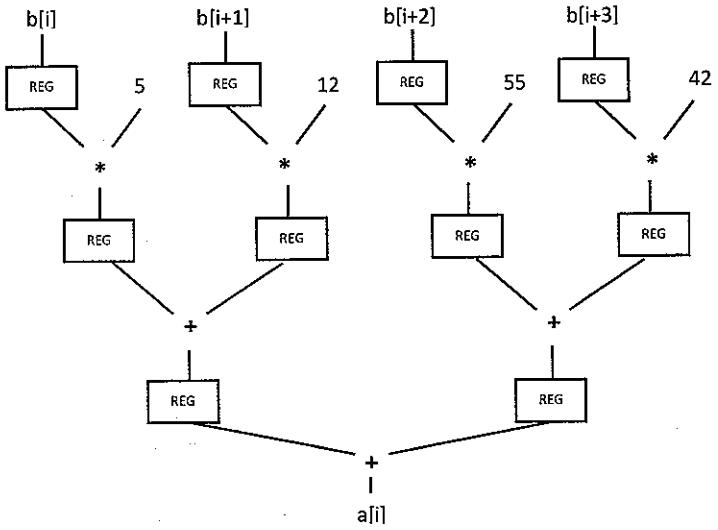
c. (7 points) Estimate the total cycles when using the following pipeline:

initial latency ≈ 4 cycles

cycles per iteration = 1

total cycles = 4 + 999,999

= 1,000,003



11) **(4 free points)** Write or draw something funny if you have time left.