EEL 4712
Midterm 3 – 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:

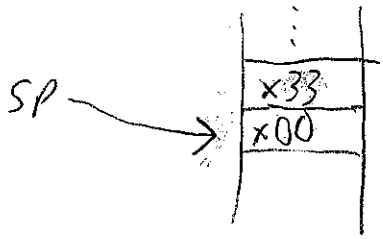
Problem#:	Points
1 (6 points)	
2 (10 points)	
3 (5 points)	
4 (5 points)	
5 (5 points)	
6 (12 points)	
7 (6 points)	
8 (12 points)	
9 (5 points)	
10 (5 points)	
11 (25 points)	
12 (5 points)	

Total:

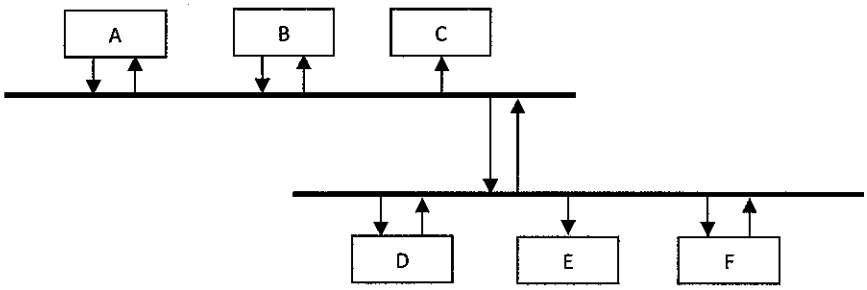
Regrade Info:

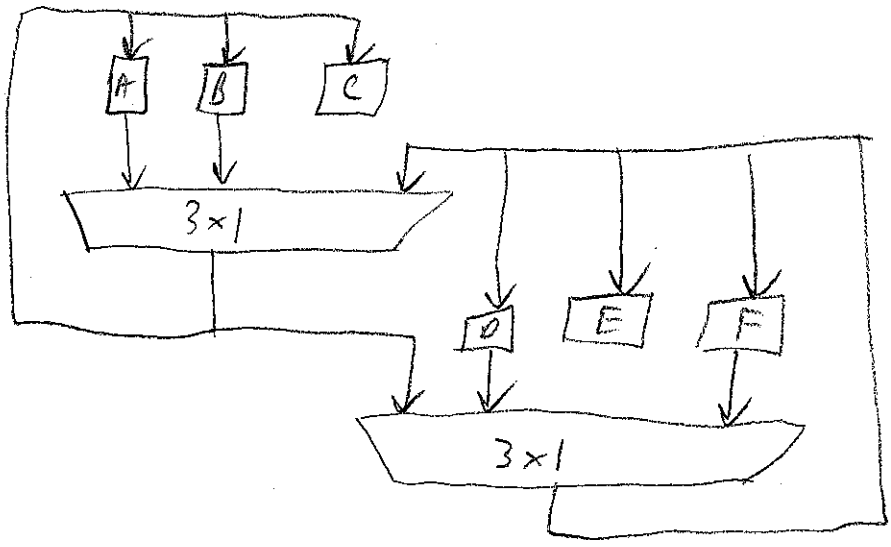
```
ENTITY _entity_name IS
PORT(__input_name, __input_name : IN STD_LOGIC;
__input_vector_name : IN STD_LOGIC_VECTOR(__high downto __low);
__bidir_name, __bidir_name : INOUT STD_LOGIC;
__output_name, __output_name : OUT STD_LOGIC);
END __entity_name;

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;
```

1) (5 points) For a call instruction at address 0x30, show the state of the stack after the call but before the corresponding return. You can omit any data that was on the stack prior to the call instruction. Instead of showing an explicit address in the stack pointer, just point to the top of the stack.



2) a. (5 points) For the following buses, show the synthesized circuit for an FPGA. Be sure to show inputs and outputs for all entities connected to the bus.





b. (5 points) What potential problem will be identified during synthesis for the circuit in part a?

combinational loop

3) (5 points) True/False. A dual-flop synchronizer addresses metastability problems by waiting one cycle to guarantee that the metastable output has stabilized.

false

4) (5 points) Name two situations where the input to a flip-flop may change during the setup and hold window.

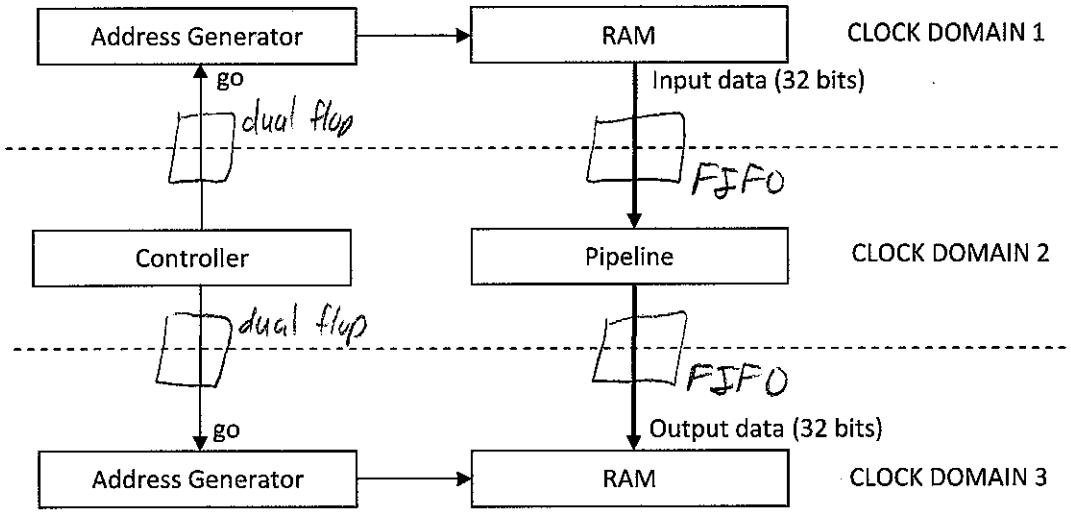
asynchronous inputs

clock domain crossing

5) (5 points) In what unique situation can a dual-flop synchronizer be used to synchronize multiple bits?

when a single input bit changes

6) (12 points) Show where synchronizers should be used in the following schematic to handle all communication across clock domains. Make sure to label the type of synchronizer. You do not need to show how the synchronizer is implemented.



7) (6 points) *Briefly* describe what will happen while simulating the following 2-process FSMD. Identify the problematic line(s) of code, if any.

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

entity fsmd is
  port( clk  : in  std_logic;
        rst  : in  std_logic;
        go   : in  std_logic;
        done : out std_logic);
end fsmd;

architecture bhv of fsmd is

  type STATE_TYPE is (S_START, S_COUNT, S_DONE);
  signal state, next_state : STATE_TYPE;
  signal count             : unsigned(3 downto 0);
  constant MAX_COUNT_VAL : natural := 10;

begin

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

  process(go, state, count)
  begin

    case state is
      when S_START =>

        done  <= '0';
        count <= to_unsigned(1, count'length);

        if (go = '0') then
          next_state <= S_START;
        else
          next_state <= S_COUNT;
        end if;

      when S_COUNT =>

        done  <= '0';
        count <= count + 1;

        if (count = MAX_COUNT_VAL) then
          next_state <= S_DONE;
        else
          next_state <= S_COUNT;
        end if;

      when S_DONE =>

        count      <= to_unsigned(MAX_COUNT_VAL, count'length);
        done       <= '1';
        next_state <= S_DONE;

      when others => null;
    end case;

  end process;
end bhv;
```

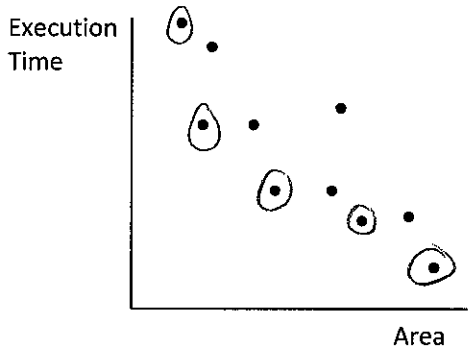
infinite simulation loop

8) (12 points) Create a memory initialization file for the following assembly code. Add a comment at the beginning of each instruction. *You will likely need to break your answer up into two columns to fit on the page.*

```
INPORT0     EQU   $FFFE
OUTPORT0    EQU   $FFFE

BEGIN:
        LDAA    INPORT0
        STAA    COUNT
AGAIN:
        LDAA    VALUE
        CLRC
        RORC
        STAA    VALUE
        LDAA    COUNT
        DECA
        STAA    COUNT
        BNEA    AGAIN
        LDAA    VALUE
        STAA    OUTPORT0

INFINITE_LOOP:
        CLRC
        BCCA    INFINITE_LOOP

* Data Area
VALUE: dc.b   $AA
COUNT: ds.b   1

        END     BEGIN
```

```
Depth = 256;
Width = 8;
Address_radix = hex;
Data_radix = hex;
% Program RAM Data %
Content
  Begin
0000 : 88; //LDAA
0001 : FE;
0002 : FF;
0003 : F6; //STAA
0004 : 23
0005 : 00
0006 : 88  //LDAA
0007 : 22
0008 : 00
0009 : F9  //CLRC
000A : 62  //RORC
000B : F6  //STAA
000C : 22
000D : 00
000E : 88  //LDAA
000F : 23
0010 : 00
0011 : FB  //DECA
0012 : F6  //STAA
0013 : 23
0014 : 00
0015 : B4  //BNEA
0016 : 06
0017 : 00
0018 : 88  //LDAA
0019 : 22
001A : 00
001B : F6  //STAA
001C : FE
001D : FF
001E : F9  //CLRC
001F : B0  //BCCA
0020 : 1E
0021 : 00
0022 : AA  // VALUE
0023 : 01  // count

[       ..00FF] : 00;
End;
```

9) (5 points) For the set of implementations shown below, circle the implementations that are Pareto optimal. List any assumptions.



10) (5 points) Although loop unrolling (wide parallelism) and pipelining (deep parallelism) can sometimes achieve the same performance, briefly explain why loop unrolling by itself might not be Pareto optimal.

pipelining uses fewer resources

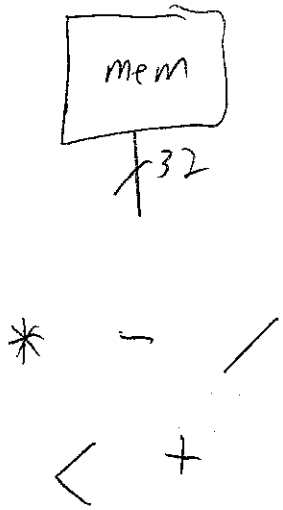
11) a. (20 points) For the following pseudo-code, create a non-pipelined implementation. List the datapath resources and the corresponding schedule. You do ***not*** need to show all datapath connections, just the computational resources. **Assume all memory accesses take 1 cycle. Assume all operations take 1 cycle, except for the divide, which takes 5 cycles. Show the estimated execution time in cycles assuming the if branch is always taken.** If your schedule is non-obvious, make sure to explain.

```
float q_val = q[50];
float m_val = m[50];
for (k=0; k < 10000; k++) }
      float diff = q[k] - q_val;
      float diff3 = diff * diff * diff;
      if (diff3 < 0) diff3 = diff3 * -1.0;
      a[i] = m_val*m[k]*diff / diff3;
}
```

Schedule:

1) load q_val
2) load m_val, K=0
3) K < 10000, load q[K]
4) q[K] - q_val, load m[K]
5) diff * diff
6) diff² * diff
7) diff3 < 0
8) diff3 * -1
9) m_val * m[K]
10) ↓ * diff
11-15) / diff3
16) store a[i], K++, go to 3)

Datapath:



execution time = 2 + 10000 (14)

≈ 140000 cycles

b. (5 points) Assume you are given a pipelined datapath that implements the loop in part a. The pipeline has a latency of 32 cycles and does not use any unrolling. What is the execution time in cycles when using this pipeline?

32 + 9999

10031 cycles

12) (5 points) Application design productivity using HDLs is an order of magnitude lower than approaches for other devices (CPUs, GPUs). What is the name of the research area that aims to enable FPGA design from high-level languages?

high-level synthesis