ECE 341 Digital Systems Design
Test 1

October 14, 2020

Instructions:

This test is organized into three sections:

Section 1: Problems (synchronous, Oct 14 from 4:20-5:35pm)
Section 2: Short answer (synchronous, Oct 14 from 4:20-5:35pm)
Section 3: Aldec problem (due in 24 hours: 5:35pm Thursday Oct 15)

For Sections 1&2, this test is closed book, closed notes, no calculators, no Aldec, unless otherwise noted or directed. You are permitted 1/2 of an 8.5 by 11 inch "cheat sheet", one side only (total area 8.5" × 5.5") with text size 6 point or larger. For Section 3, the problem statement and instructions will be provided in a separate document.

For Section 1, please show all of your work. For Section 2, provide the answer and partial credit will generally not be awarded in this section.

For the duration of the synchronous part of the test, leave your cameras turned on throughout the Zoom session and remain within the field of view of the camera.

If you have a question, "raise your hand" or send me your question through a private chat message. If necessary, we may go to a breakout room.

Your solutions to Sections 1&2 can be written on regular paper and be sure to accurately label each problem. You may use any color pencil or ink other than red. If you have a tablet computer and can enter written answers on this document, you may do so, provided the document you submit can be viewed using any PDF viewer. To be clear, only hand written solutions are permitted. Solutions that are typed, word processed, and/or diagrams composed using drawing tools are not acceptable.

When time is called, within 15 minutes of completing Sections 1&2 (by 5:50pm on Oct 14), upload an electronic copy (a clear scanned copy) of your test answers & "cheat sheet" in PDF format using the link provided. If you are unable to do so, (1) let me know immediately (email is fine), (2) photograph & upload individual pages from your test, again within the 15 minutes, and (3) upload a clear scanned copy as soon as possible afterwords.

Use the following link to upload your
https://www.dropbox.com/request/kGjwI8YeoT6MXkta5Cv5

Good Luck!

I pledge to support the Honor System at Old Dominion University. I will refrain from any form of dishonesty or deception, such as lying, cheating, or plagiarizing, which are Honor violations. I am further aware that as a member of the academic community, it is my responsibility to turn in all suspected violators of the Honor System. I will report to an Honor Council hearing if summoned.

(Print name) _____

(Sign here or write out pledge on your submission and sign) _____

Section 1: Problems (70 points)

P1 (25 points) Using the logic function $F(A, B, C) = \Sigma(2, 3, 5, 7)$, answer the following questions. For complete answers, you may need to perform some additional design work.

P1(a) (5) Give a signal assignment using only logical operations and expressions.

K MAP TO GET FUNCTION

F = $\bar{A}B + AC$

SIGNAL ASSIGNMENT

F <= (NOT A AND B) OR (A AND C);

P1(b) (5) Give a `when-else` signal assignment statement that specifically tests individual input values.

MORE THAN ONE SOLUTION POSSIBLE

F <= '1' WHEN ((A='0' AND B='1') OR (A='1' AND C='1')) ELSE '0';

P1(c) (5) Using a `case-when` statement, give the code that implements the logic function where the cases are the minterms.

```
CASE A&B&C IS
   WHEN "010" => F<='1';
   WHEN "011" => F<='1';
   WHEN "101" => F<='1';
   WHEN "111" => F<='1';
   WHEN OTHERS => F<='0';
END CASE;
```

P1(d) (5) Using `if` statements, give the code that implements the logic function that shows the signal changes to `'0'` in 5 ns whereas the signal changes to `'1'` in 10 ns.

MORE THAN ONE SOLUTION POSSIBLE

```
IF ((NOT A AND B) OR (A AND C))='1' THEN
    F<='1' AFTER 10 ns
ELSE
    F<='0' AFTER 5 ns;
END IF
```

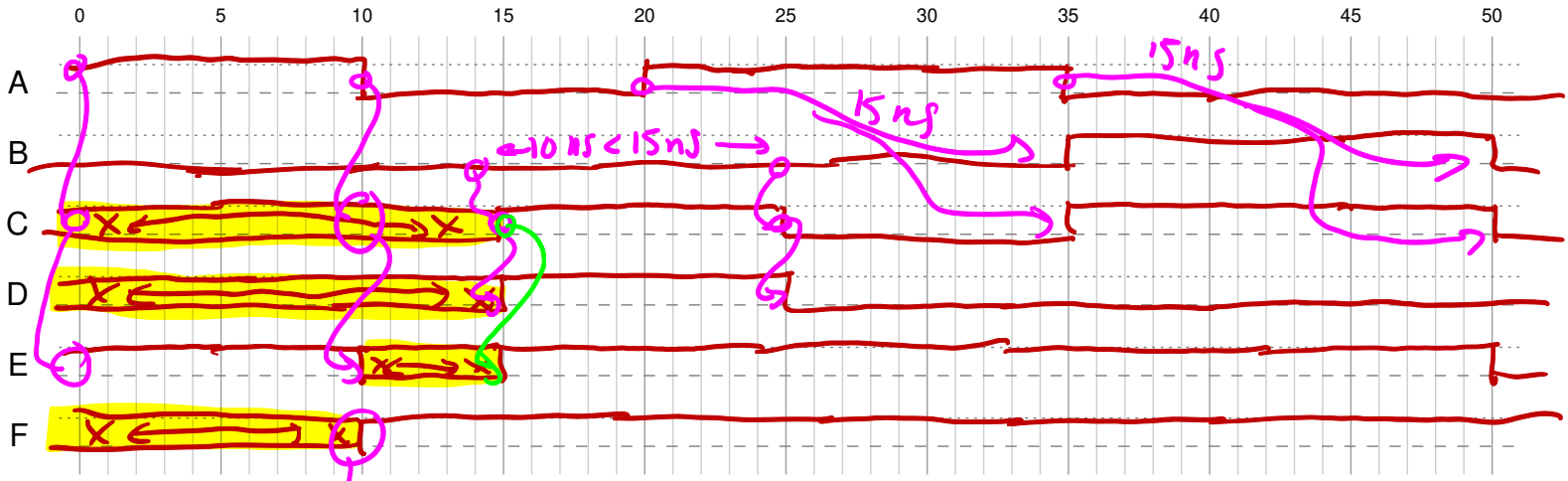
P1(e) (5) Give code that uses a look-up table to implement F.

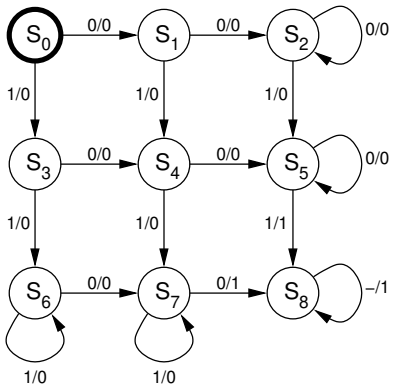
```
TYPE lutTYPE IS ARRAY (0 TO 7) OF STD_LOGIC
CONSTANT lut: lutTYPE:=('0','0','1','1','0','1','0','1');
  .
  .
  .
F <= lut(to_INTEGER(UNSIGNED(A&B&C)));
```

(ASSUMPTIONS – IEEE.NUMERIC_STD,
A, B, C, F DECLARED AS STD_LOGIC

P2 (20 points) Give the timing diagram for the following VHDL model. In your timing diagram, if any signals change at 50 ns, note the updated values. Points awarded are given in the comments. Give a justification for any signal that does not change.

```
entity Problem_2 is
end entity Problem_2;
architecture model of Problem_2 is
  signal A,B:std_logic:='0';
  signal C,D:std_logic:='X';
  signal E,F:std_logic;
begin
  -- Part A: 3 points
  A <= '1','0' after 10 ns, '1' after 20 ns, '0' after 35 ns;
  -- Part B: 3 points
  B <= A after 15 ns;
  -- Part C: 3 points
  C <= transport A after 15 ns;
  -- Part D: 5 points (3 for waveform, 2 answer the question)
  D <= B XOR C; -- What does D's value tell you?
  process(A,B,C,D)
  begin
    -- Part E: 3 points
    E <= A OR C;
    -- Part F: 3 points
    F <= E;
  end process;
end architecture;
```





P3 (25 points) Use the accompanying state diagram for this problem. In addition to an asynchronous active-low reset, X is the input, Z is the output, and S_0 is the reset state. Assume that all circuit inputs and outputs are declared as `std_logic` in the entity, and that the `numeric_std` library has been declared as well.

P3(a) (5) Give a suitable VHDL entity for the state machine. The entity name should be `Problem_3`.

```
ENTITY PROBLEM_3 IS
  PORT ( RESETN : IN STD_LOGIC;
         CLK : IN STD_LOGIC;
         X : IN STD_LOGIC;
         Z : OUT STD_LOGIC );
END ENTITY PROBLEM_3;
```

P3(b) (18) Complete the following partial VHDL model for the above state machine. The state assignment presumes a one-hot finite state machine, e.g,

S_0="100000000" S_1="010000000" S_2="001000000"
S_3="000100000" S_4="000010000" S_5="000001000"
S_6="000000100" S_7="000000010" S_8="000000001"

Any white space can be used to add any other code you feel is necessary and appropriate.

```
architecture behavioral of Problem_3 is
  signal State,nextState: std_logic_vector(0 to 8);
begin

  -- if you are having problems with the following process, for reduced
  -- credit, rewrite in a way that makes sense to you
  process( STATE, X                    )

  begin

    Z<='0';                  -- DEFAULT - NOT DETECTED
    nextState <= (others => '0'); --DEFAULT: ALL ZEROS NOTE THAT
    if state(0)='1' then     -- THIS IS AN ILLEGAL
       if X='0' then
          nextState(1)<='1'; -- not a typo
       else
          nextState(3)<='1'; -- not a typo
       end if;

    elsif state(1)='1' then
       if X='0' then
         nextState(2)<='1';
       else
         nextState(4)<='1';
       end if;

    elsif state(2)='1' then
```

-- FOR FPGAS
-- IMPLEMENTING STATE
-- MACHINES USING ONE
-- HOT STATE ASSIGNMENTS
-- IS GENERALLY MORE EFFICIENT
-- ON THAT PLATFORM,
-- SYNTHESIS SOFTWARE WILL
-- GENERALLY DO THIS FOR YOU
-- NOT PART OF SOLUTION

```
    if x='0' then
      nextState(2)<='1';
    else
      nextState(5)<='1';
    end if;

elsif state(3)='1' then
    if x='0' then
      nextState(4)<='1';
    else
      nextState(6)<='1';
    end if;
elsif state(4)='1' then
    if x='0' then
      nextState(5)<='1';
    else
      nextState(7)<='1';
    end if;
elsif state(5)='1' then
    if x='0' then
      nextState(5)<='1';
    else
      nextState(8)<='1'; z='1';
    end if;
elsif state(6)='1' then
    if x='0' then
      nextState(7)<='1';
    else
      nextState(6)<='1';
    end if;
elsif state(7)='1' then
    if x='0' then
      nextState(8)<='1'; z<='1';
    else
      nextState(7)<='1';
    end if;

elsif state(8)='1' then
    nextState(8)<='1';
    z<='1';

else
    -- SHOULD NEVER HAPPEN
    null;
```

```
    end if;

  end process;

  process(resetn,clk)
  begin

    if resetn= '0'          then
       State<= (0=>'1', others=>'0');

    elsif CLK'EVENT AND CLK='1' THEN
        STATE <= nextState;


   end if;

end architecture;
```

P3(c) (2) What does this state machine do?

DETECTS WHEN TWO '0'S AND TWO '1'S OCCUR IN ANY ORDER

Section 2: Short answer (10 points)

Each short answer question is worth 2 points.

S1 Give one reason why you might want to use an active low reset.

IN SYSTEMS WITH RESETS, A SIMPLE RESISTOR/CAPACITOR CIRCUIT CAN BE USED TO PROVIDE A ROBUST RESET SIGNAL THAT IS ACTIVE LOW AND NATURALLY WORKS WHEN POWER IS TURNED ON IN THE SYSTEM

S2 What is the meaning of the following VHDL code snippet mean?

```
unsigned'(0 => S) --where S is a signal of type std_logic
```

UNSIGNED STD_LOGIC VECTOR OF LENGTH 1 WHERE THE VALUE OF THE BIT IS S

S3 How much simulation time passes when a VHDL simulation completes a delta cycle?

0 NS

S4 Which is more general (circle (A) or (B) or both if there is no difference)

(A) a process with a sensitivity list
(B) a process with wait statements

S5 State one capability that the type `std_logic` has that the type `bit` does not have.

STD_LOGIC CAN REPRESENT SOME SITUATIONS THAT OCCUR IN CIRCUITS AND SIMULATIONS THAT ARE NOT THE ORDINARY LOGIC VALUES '0', '1'
FOR EXAMPLE
'U' UNINITIALIZED
'X' UNKNOWN
'Z' HIGH IMPEDANCE

GRADING NOTE
FOR CREDIT, NEED TO GIVE A SPECIFIC EXAMPLE BECAUSE THAT IS WHAT CLEARLY STATES THE CAPABILITY

Section 3: Aldec Problem (25 points)

A1 (25 points) Aldec Programming Problem. Please see separate handout for instructions.