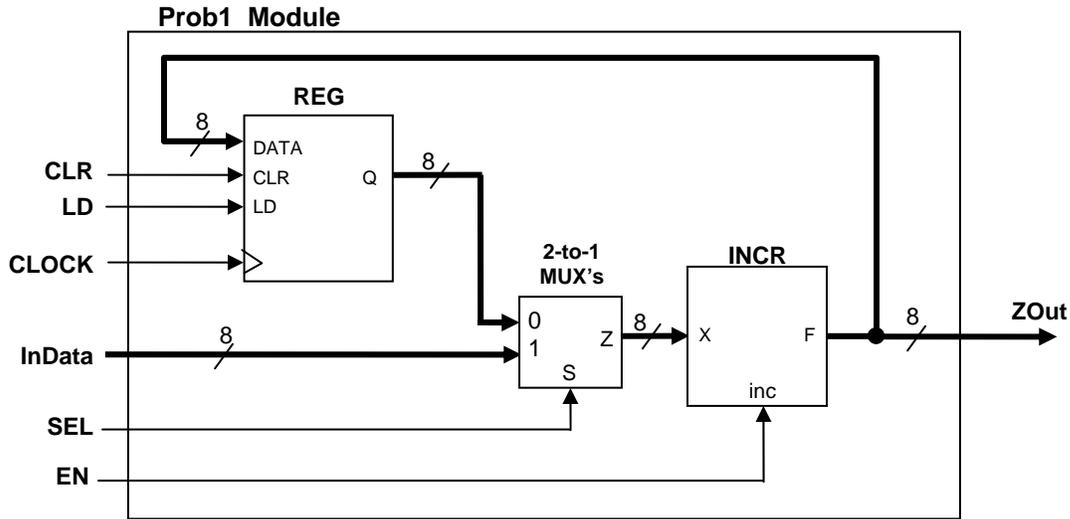


22 pts.

1. VHDL specification.

Complete the VHDL specification (below and on the next page) for the following circuit:



Notes:

- REG is an 8-bit storage register with an asynchronous CLR and synchronous LD inputs (CLR has priority over LD).
- There are eight 2-to-1 MUX's.
- INCR functions as follows:
 - If **inc** = 0, $F \leq 0$,
 - If **inc** = 1, $F \leq X + 1$ (i.e., increment X).
- All signals are active high.

(a) Complete the following Entity declaration for the Prob1 Module: (2 pts)

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
ENTITY Prob2 IS
    -- Declare ZOut to be an OUT signal type.
    PORT(

```

END Prob1;

(b) On the next page, complete the architecture section to specify the behavior of the Prob1 Module in behavioral VHDL. (20 pts.)

Notes: Follow exactly the requirements below

- Every statement must be inside a PROCESS statement (you can use any number of PROCESS statements).
- IF and simple assignment statements only.
- The best answer gets the most points.

ARCHITECTURE behaviorArch OF Prob1 IS

SIGNAL

BEGIN

PROCESS

BEGIN

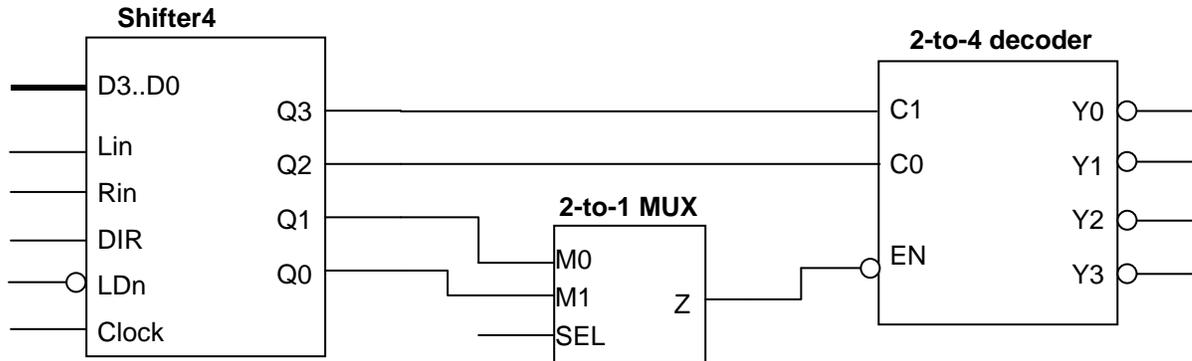
-- You must use a WAIT UNTIL statement to specify REG

END behaviorArch;

2. VHDL specification.

22 pts.

Complete the VHDL specification (on the next page) for the following circuit: (Note that a “bubble” indicates that the signal is active low.)



- Shifter4 is a 4-bit shift register that can shift left when (DIR = 0) or shift right (when DIR = 1):
 - When shifting left, $Q0 \leftarrow Lin$; when shifting right, $Q3 \leftarrow Rin$.
 - When $LDn = true$ (active low), then D3 is loaded into the Shifter4. LDn is synchronous and has priority over shifting.
- The 2-to-4 decoder (with an active low enable EN) must be specified using a WITH-SELECT assignment statement. (All decoder outputs are active low).
- The logic for the MUX must be specified using a conditional assignment statement.

Put solution for Problem 2 here:

ENTITY Test1P2 IS

PORT (Lin, Rin, DIR, LDn, Clock, SEL : IN STD_LOGIC;
D : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
Y : OUT STD_LOGIC_VECTOR (3 DOWNTO 0));

END Test1P2 ;

ARCHITECTURE P2Arch OF T1Prob2 IS

SIGNAL

BEGIN

PROCESS ()
BEGIN

END P2Arch ;

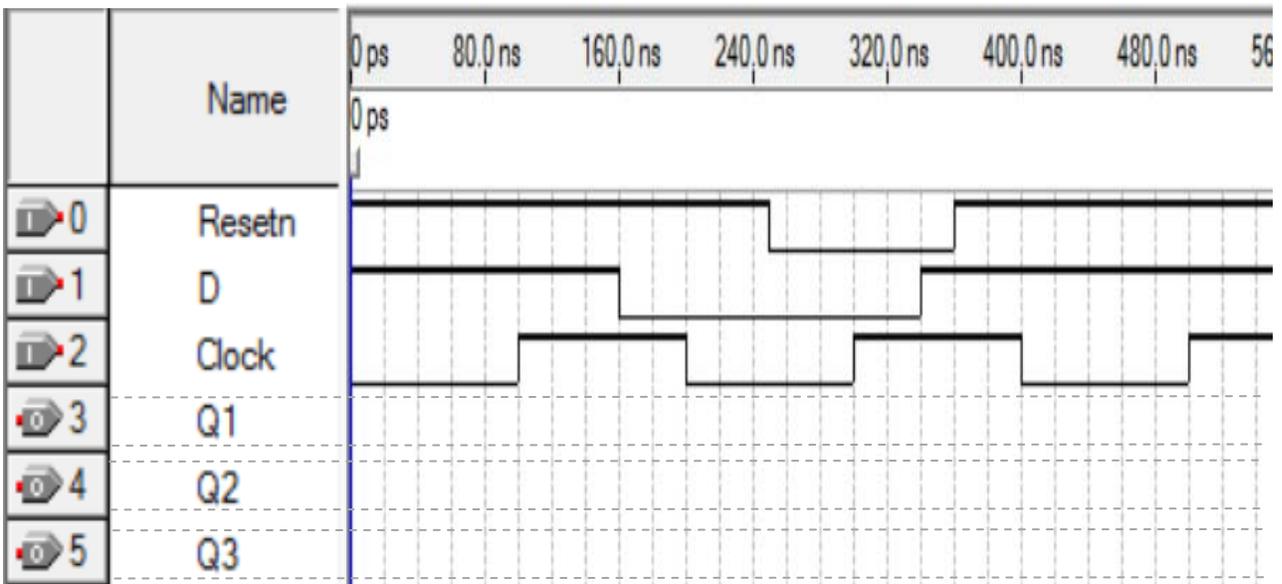
15 pts.

3. VHDL Analysis – Timing diagrams). Given the following VHDL specification, complete the following timing diagram for the outputs, **Q1, Q2, and Q3**.

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY Test1P3 IS
    PORT ( D,Resetn, Clock      : IN  STD_LOGIC ;
          Q1, Q2, Q3           : OUT STD_LOGIC);
END Test1P3 ;
```

```
ARCHITECTURE Behavior OF Test1P3 IS
BEGIN
    PROCESS ( D, Resetn, Clock )
    BEGIN
        Q3 <= '0';
        IF Resetn = '0' THEN
            Q1 <= '0' ;
        ELSIF Clock = '1' THEN
            Q1 <= D ;
        END IF ;
        IF Resetn = '0' THEN
            Q2 <= '0' ;
        ELSIF (Clock'event AND Clock = '1')THEN
            Q2 <= D ;
        END IF ;
        If (Resetn = '1' AND Clock = '1') THEN
            Q3 <= D ;
        END IF ;
    END PROCESS ;
END Behavior ;
```

Note: Please show delays. The initial contents of all flipflops are unknown. Also, go as far as you can.



The following VHDL code is used for Problem 4 and Problem 5:

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.std_logic_unsigned.all ;

ENTITY Test1P4_5 IS
  PORT (   Clock, Resetn : IN   STD_LOGIC ;
          Z1, Z2, Z3     : OUT STD_LOGIC;
          Q, QQA, QQB, QQC : OUT STD_LOGIC_VECTOR (3 DOWNTO 0)) ;
END Test1P4_5 ;

ARCHITECTURE Behavior OF Test1P4_5 IS
  SIGNAL Count : STD_LOGIC_VECTOR (3 DOWNTO 0) ;

BEGIN
  QQA <= Count ;

  PROCESS ( Clock, Resetn )
  BEGIN
    Q <= Count ;

    IF Resetn = '0' THEN
      Count <= "0000" ;
    ELSIF (Clock'EVENT AND Clock = '1') THEN
      forloop: FOR i IN 0 TO 3 LOOP
        Count <= Count + 1 ;
        Z1 <= Count(0);
      END LOOP;
      {
        IF (Count = "0000") THEN -- This IF statement is used for Problem 5(b)
          Z2 <= '1';
        ELSE
          Z2 <= '0';
        END IF;
      }
      QQB <= Count;
    ELSE
      Count <= Count ;
    END IF ;

    QQC <= Count;

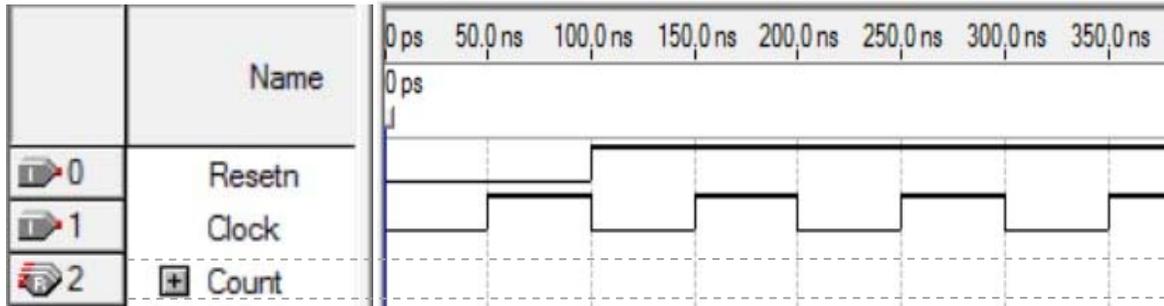
    {
      IF (Count = "0000") THEN -- This IF statement is used for Problem 5(a)
        Z3 <= '1';
      ELSE
        Z3 <= '0';
      END IF;
    }

  END PROCESS ;
END Behavior ;
```

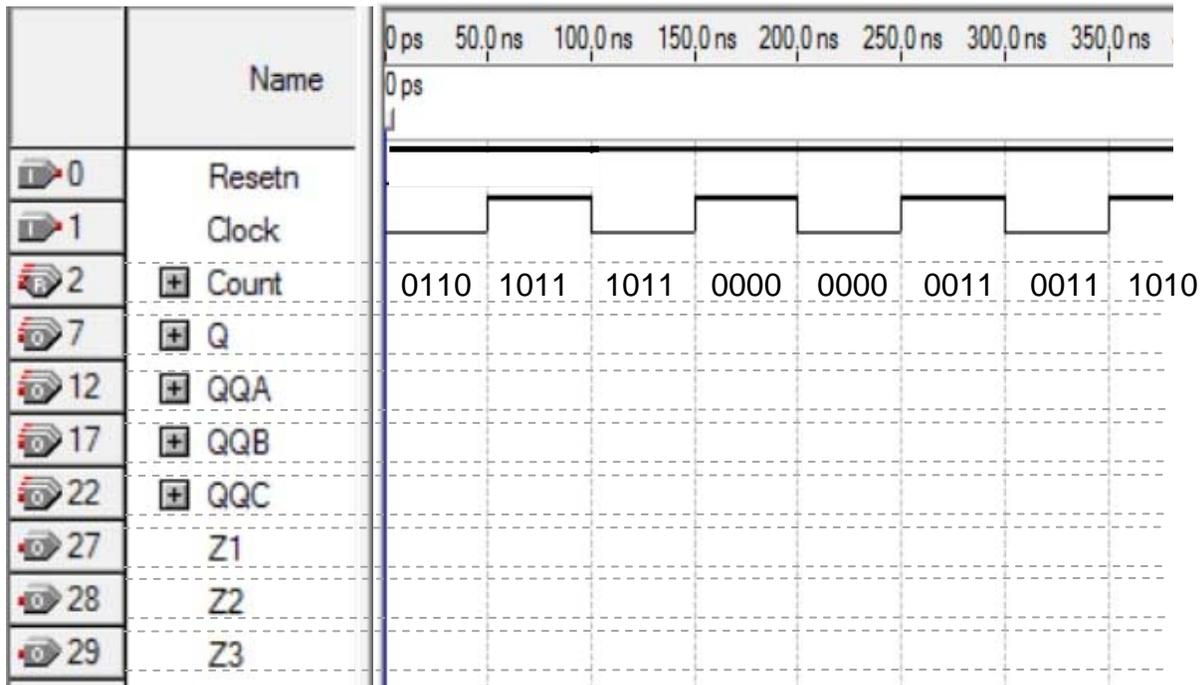
4. VHDL analysis – Timing diagram.

18 pts.

- (a) Based on the code shown on Page 7, complete the following timing diagram for the values for Count (in binary): (The initial contents of all flipflops are unknown. Also, go as far as you can.) (3 pts)



- (b) Assume the given Count values, complete the following timing diagram based on the code shown on Page 7 (i.e., don't use the code to figure out the Count values, they are given to you for this part of the problem). Give all values in binary. (The initial contents of all flipflops are unknown. Also, go as far as you can.) (15 pts)



5. VHDL Analysis and re-code

9 pts.

(a) Given the IF statement at the bottom of the code shown on Page 7, convert it into a SELECT assignment statement that can be specified outside the PROCESS block. (3 pts)

IMPORTANT: If necessary, you need to tell me what else you need to add to the code (if anything) to maintain the behavior of the original code.

(b) Given the IF statement in the middle of the code shown on Page 7, convert it into a conditional assignment statement that can be specified outside the PROCESS block. (6 pts)

IMPORTANT: If necessary, you need to tell me what else you need to add to the code (if anything) to maintain the behavior of the original code.

6. Other topics.

14 pts.

(a) Let us assume you have the components from Lab 3: adder2lca (2-bit adder) and lca2gen (2-bit look-ahead carry generator). (3 pts.)

For a 64-bit adder, how many lca2gen components will you need? _____

For a 64-bit adder, how many levels of lca2gen will you need? _____

(b) Let us assume you have the components from Lab 3: adder2lca (2-bit adder) and lca2gen (2-bit look-ahead carry generator). (3 pts.)

For a 48-bit adder, how many lca2gen components will you need? _____

For a 48-bit adder, how many levels of lca2gen will you need? _____

(c) Assume you are designing an 8-bit adder to perform two's complement addition. Given me the logic table and equation for OV (two's complement overflow) as a function of the sign bits of the two operands and the sum. (3 pts)

Put logic table here:

Put equation for OV here:

(d) Compare/contrast the function of a logic state analyzer (LSA) vs. the function of an oscilloscope. (2 pts)

(e) Compare/contrast the functions of an LSA, Quartus simulation, and Quartus SignalTap. (3 pts)

```
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;
    SIGNAL __signal_name : STD_LOGIC;
BEGIN
    -- Process Statement
    -- Concurrent Signal Assignment
    -- Conditional Signal Assignment
    -- Selected Signal Assignment
    -- Component Instantiation Statement
END a;

SIGNAL __signal_name : __type_name;

__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;

WAIT UNTIL __expression;

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