EEL 4712 – Digital Design
Test 1 – Spring Semester 2007 Name _____

IMPORTANT: Please be neat and write (or draw) carefully. If we cannot read it with a reasonable effort, it is assumed wrong.

COVER SHEET:

Problem:	Points:
1 (15 pts)	
2 (20 pts)	
3 (15 pts)	
4 (18 pts)	
5 (18 pts)	
6. (14 pts)	

Total

Re-Grade Information:

1. VHDL Analysis (circuit synthesis).

15 pts.

Given the following ("non-sense") VHDL specification, draw the circuit diagrams for the corresponding components. Draw them on the next page.

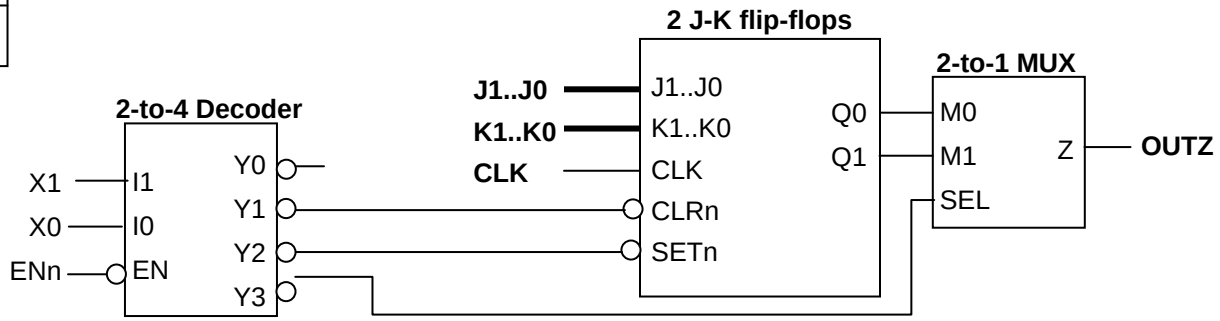
```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY Test1P1 IS
      PORT ( A, B, C, D, E, G, I, J, CLK : IN STD_LOGIC ;
            Y : INOUT STD_LOGIC_VECTOR (2 DOWNTO 1) );
END Test1P1 ;

ARCHITECTURE P1Arch OF Test1P1 IS

COMPONENT LCX
     PORT (A: IN STD_LOGIC;   X1, X2: OUT STD_LOGIC);
END COMPONENT;

SIGNAL TEMP1, TEMP2, TEMP3 : STD_LOGIC;

BEGIN
     PROCESS (A, C, E, TEMP1, CLK)
     BEGIN
          IF TEMP1 = '0'
               THEN Y <= "00";
          ELSIF (CLK'Event AND CLK = '1')
               Y(2) <= Y(1);
               CASE B IS
                    WHEN '0' =>
                         Y(1) <= TEMP3 ;
                    WHEN OTHERS =>
                         Y(1) <= D ;
               END CASE ;
          END IF;

          IF E = '0' THEN TEMP2 <= A;
               ELSE TEMP2 <= C;
          END IF;
     END PROCESS ;

     LCX port map(X1 =>J, X2 =>TEMP3, A =>TEMP2);
     TEMP1 <= G WHEN I = '0' ELSE J;

END P1Arch ;
```

Problem 1: VHDL Analysis (circuit synthesis): Put your answer here.

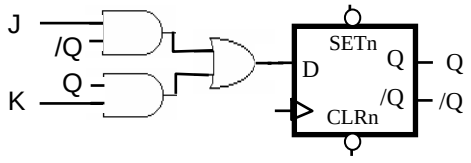
2. VHDL specification.

20 pts.

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



- The logic for the decoder **must** be specified using **a CASE statement**.
- The logic for the MUX **must** be specified using a **WITH-SELECT** assignment statement.
- Each JK flip-flop has an **asynchronous** clear and an **asynchronous** set. (Clear has priority.)
- Each JK flip-flop can be constructed using a D flip-flop as shown.



Put solution for Problem 2 here:

```
ENTITY Test1P2 IS
        PORT ( ENn, CLK        : IN    STD_LOGIC;
               X, J, K         : IN    STD_LOGIC_VECTOR(1 DOWNTO 0);
               OUTZ            : OUT   STD_LOGIC);
END Test1P2 ;

ARCHITECTURE P2Arch OF T1Prob2 IS

SIGNAL Q: STD_LOGIC_VECTOR(0 TO 1);  -- NOTE (0 TO 1) and not (1 DOWNTO 0)
SIGNAL Y: STD_LOGIC_VECTOR(0 TO 3);  -- NOTE (0 TO 3) and not (3 DOWNTO 0)

SIGNAL

BEGIN
    PROCESS (                                                    )
    BEGIN









    END PROCESS ;



END P2Arch ;
```

15 pts.

3. Given the following VHDL code, draw the corresponding circuit and explain why it does or does not perform the switch-debouning function correctly on the push button signal (PB).

```
LIBRARY ieee;
USE ieee.std_logic_1164.all ;
USE ieee.std_logic_unsigned.all ;
ENTITY switch_debounce IS
      PORT ( PB, CLK          : IN STD_LOGIC;
                dbPB  : OUT STD_LOGIC );
END switch_debounce;

ARCHITECTURE Behavior OF switch_debounce IS
SIGNAL Q : STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL tempAND : STD_LOGIC;
BEGIN
      PROCESS(PB)
         BEGIN
            IF ( CLK'EVENT AND CLK = '1' ) THEN
                FOR i IN 0 TO 2 loop
                    Q(i) <= Q(i+1);
                    tempAND <= Q(i) AND tempAND;
                end loop;
                Q(M-1) <= PB;
                dbPB <= tempAND AND Q(3);
            END IF;
      END PROCESS;
END Behavior;
```

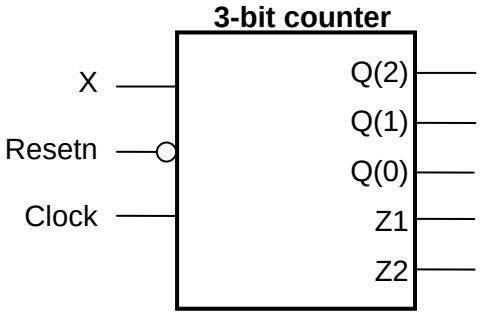
(a) Yes or No (debounced correctly)?
Explain: (1 pt.)

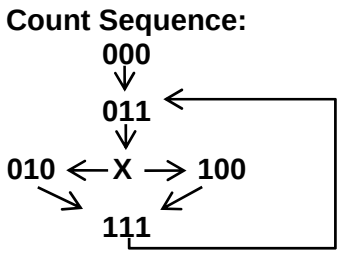
(b) <u>Draw you circuit here:</u> (14 pts.)

4. VHDL specification.

18 pts.

Complete the VHDL specification for the ARCHITECTURE part (on the next page) for the following circuit:





The 3-bit counter works as follows:

- It has a synchronous, active low Resetn to count Q[2..0] = 000.
- If Resetn = high, then it counts in a sequence as shown above: 000, 011. At Q[2..0] = 011, the next count is either 010 (if X = 0) or 100 (if X=1). In either case, it will then go to count 111, 011, etc. until Resetn is 0.
- From any illegal count (like 001 or 101), go to "000".
- Z1 = 1 whenever the count is 010.
- Z2 = 1 whenever the count is 100.

Restriction:

- **All your statements** must be contained **inside the PROCESS statement**.
- You **must use** a **CASE** statement.

```
ENTITY Counter3 IS
    PORT (      Clock, Resetn, X    : IN    STD_LOGIC;
                Q                   : OUT   STD_LOGIC_VECTOR(2 DOWNTO 0);
                Z1,Z2               : OUT   STD_LOGIC );
END Counter3 ;
```

```
ARCHITECTURE P4Arch OF Counter3 IS
SIGNAL

BEGIN
    PROCESS (                                    )
    BEGIN




    END PROCESS ;
END P4Arch ;
```

5. Generic VHDL and FOR statement

18 pts.

Complete the code below to specify a generic M bit shift register that can shift left or right:

- When EN = 0, then the function is "hold".
- When EN = 1 and LR = 0, it will shift left one bit, with LeftIN going into Q[0].
- When EN = 1 and LR = 1, it will shift right one bit, with RightIN going into Q[M-1].

```
LIBRARY ieee;
USE ieee.std_logic_1164.all ;
ENTITY GenShiftReg IS
   GENERIC(M : INTEGER := 8);
   PORT ( EN, LR, LeftIN, RightIN, Clk  : IN STD_LOGIC;
          Q : OUT STD_LOGIC_VECTOR (M-1 DOWNTO 0) );
END GenShiftReg;
ARCHITECTURE GenShiftArch OF GenShiftReg IS
SIGNAL

BEGIN
  PROCESS(                                                    )
     BEGIN  -- You have to use a WAIT UNTIL statement.
        WAIT UNTIL (                                                  )
```

```
        END PROCESS;
ENDGenShiftArch;
```

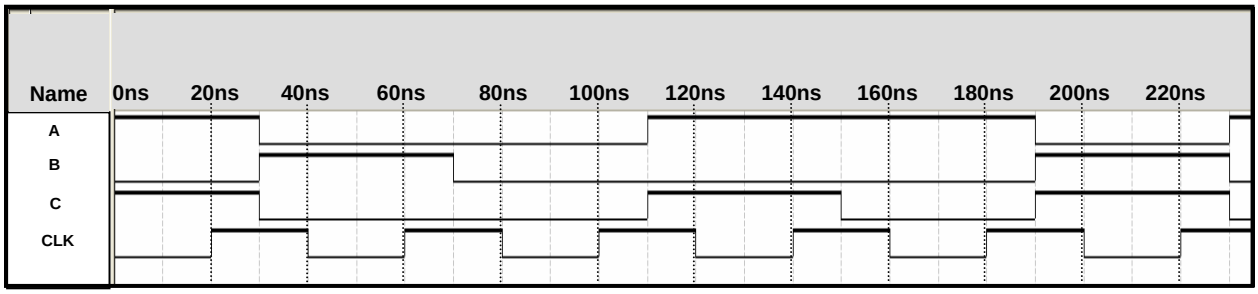
6(a and b) Lab (LSA) Question

14 pts.

The signals A, B, C, and CLK represent synchronous outputs from your BT-U board. The LSA is connected to your board in the following fashion:

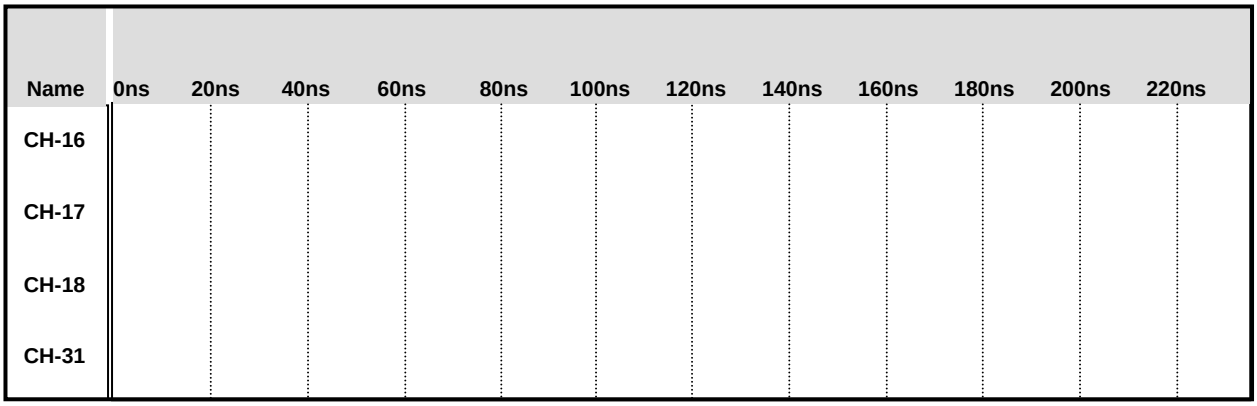
LSA channel	Signal name
16	A
17	B
18	C
31	CLK

Let' assume that "actual" signals behave as follows (vs. what is captured by an LSA).

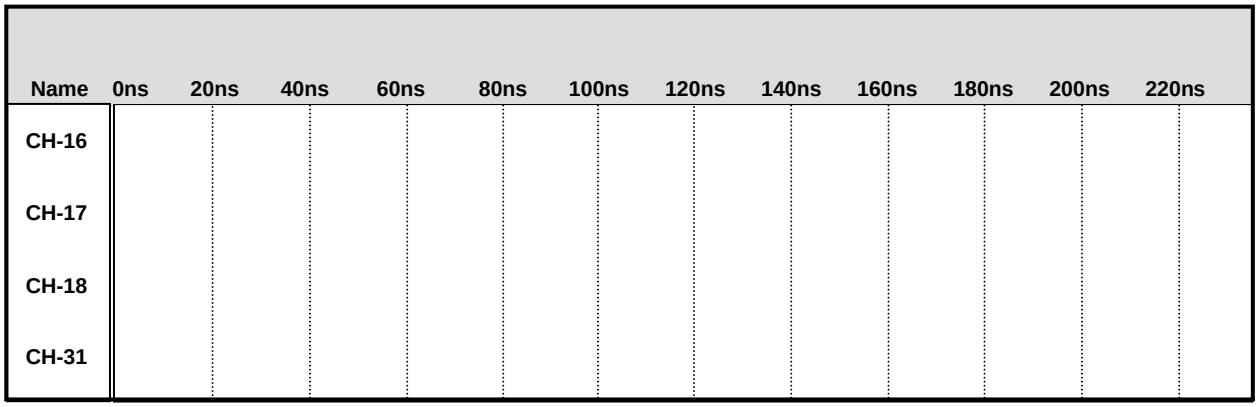


Draw the data that would be captured by the LSA, as it would appear on the screen, for each of the following scenarios:

6(a) The LSA is sampling using "Channel 17" (signal B) as the trigger. Assume the first sample is taken at 30 ns. Draw your answer for part a: (4 pts.)



6(b) The LSA is sampling using an internal 25 MHz clock source (i.e. 40 ns). Assume the first sample is taken just a 10 ns. Draw your answer for part b: (4 pts.)



6(c) LCA adder (4 pts.)

For an n-bit adder, the inputs are A(n-1)..A(0) and B(n-1)..B(0) and carry-in C(0). The outputs are SUM(n-1)..SUM(0) and carry-out C(n). (5 pts.)

For an n-bit look-ahead carry generator, the equation for carry-out of stage "i" is:

C(i+1) = G(i) OR P(i) AND C(i)

What is the equation for C(3) **as a SOP function of P(i)'s, G(i)'s, and C(0)**?

C(3) =

What is the equation for C(n) **as a SOP function of P(i)'s, G(i)'s, and C(0)**?

C(n) =

6(d) Addition overflow (2 pts.)

Complete the following two sentences concerning overflow in English (no formulas):

When adding 2 unsigned binary number, there is an overflow when

_____

When adding two 2's complement numbers, there is an overflow when

_____

_____

```
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;
```

```
loop_label:
FOR index_variable IN range_low TO range_high LOOP
        statement1;
        statement2;
END LOOP loop_label;
```