EEL 4712 – Digital Design
Test 1 – Spring Semester 2009

Name Solution

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 (18 pts)	
2 (18 pts)	
3 (12 pts)	
4 (24 pts)	
5 (18 pts)	
6. (10 pts)	

Total

Re-Grade Information:

15 pts

1. VHDL Analysis (circuit synthesis)

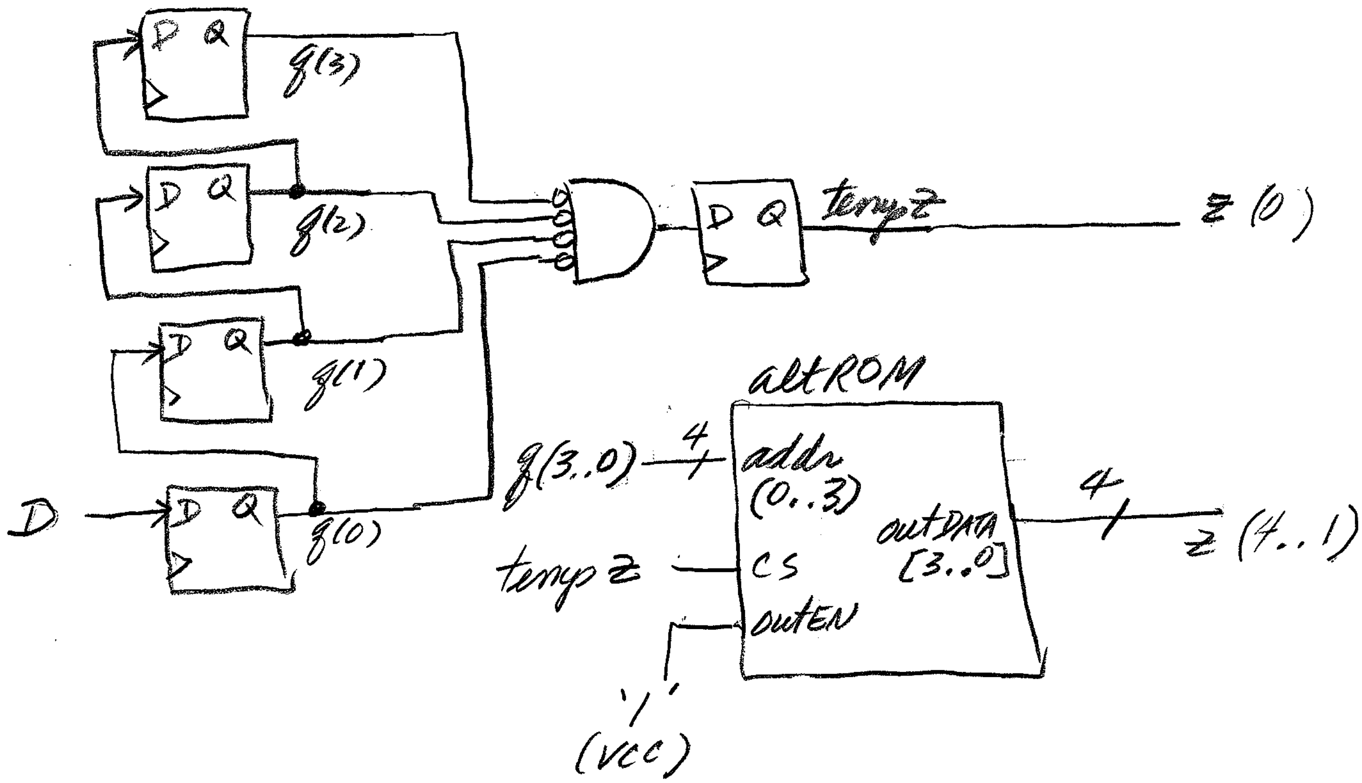
Given the following VHDL specification, draw the corresponding circuit.

```
ENTITY T1Prob1 IS
    PORT (sysClk : IN STD_LOGIC;
          D      : IN STD_LOGIC;
          Z      : OUT STD_LOGIC_VECTOR (4 DOWNTO 0));
END T1Prob1;

ARCHITECTURE Prob1Arch OF T1Prob1 IS

SIGNAL tempZ : STD_LOGIC;
SIGNAL q, x : STD_LOGIC_VECTOR(3 DOWNTO 0);

COMPONENT altROM
    PORT ( addr        : IN STD_LOGIC_VECTOR (0 TO 3);
           CS, outEn   : IN STD_LOGIC;
           outData     : OUT STD_LOGIC_VECTOR (3 DOWNTO 0));
END COMPONENT;

BEGIN

    PROCESS
        BEGIN
        Wait Until (sysClk'event AND sysClk = '1' );
            loop1: FOR i IN 0 TO 2 loop
                loop2: FOR j IN 0 to 2 loop
                    q(i+1) <= q(i);
                end loop;
            end loop;
            q(0) <= D;
            x <= (others =>'0');
            if q = x then
                tempZ <= '1';
            else
                tempZ <= '0';
            end if;
    END PROCESS;

    altROM PORT MAP (q, tempZ,'1', Z(4 DOWNTO 1));

    Z(0) <= tempZ;

END Prob1Arch;
```

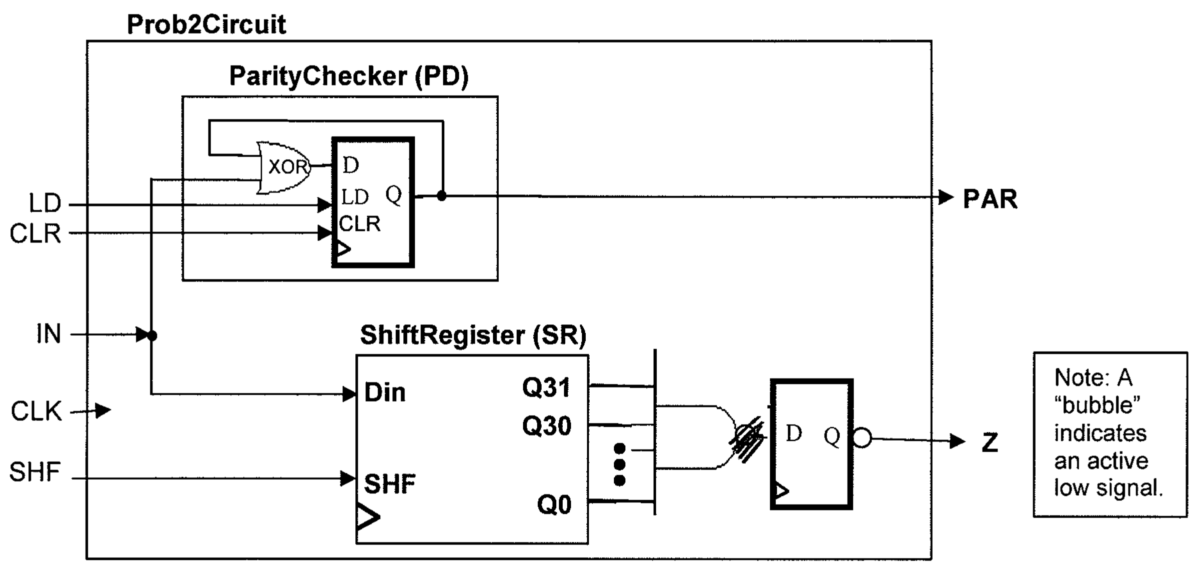
q(1) <= q(0);
q(1) <= q(0);
q(1) <= q(0);
q(2) <= q(1);
q(2) <= q(1);
q(2) <= q(1);
q(3) <= q(2);
q(3) <= q(2);
q(3) <= q(2);
q(0) <= D;

1. **(continued) Put answer for Problem 1 here:**

sysclk → all clock inputs



2. **VHDL specification**



- Put your answer for Probem 2 on the next page.
- PD is a D flip-flop (with an XOR gate) with synchronous LD and CLR. CLR has priority over LD.
- SR is a 32-bit shift register: If SHF = 1, it will synchronously shift right, else "hold".

18 pts

2. **(continued)** Complete the following VHDL code to define the above circuit. All your code in the architecture ***must*** be inside the **one PROCESS statement**.

```
ENTITY Prob2Circuit IS
    PORT( LD, CLR, IN, CLK, SHF : IN STD_LOGIC;
          PAR, Z : OUT STD_LOGIC);
END Prob4Circuit;

ARCHITECTURE behaviorArch OF Prob4Circuit
SIGNAL  Q : STD_LOGIC_VECTOR (31 DOWNTO 0);
        PDQ : STD_LOGIC;
BEGIN

PROCESS ( CLK ) -- All your code must be inside this one PROCESS statement.
                -- The best answer gets the most points.

IF (CLK'EVENT AND CLK = '1') THEN
   IF CLR = '1' THEN
      PDQ <= 0;
   ELSIF LD = '1' THEN
      PDQ <= IN XOR PDQ;
   END IF;                                  PD
   IF SHF = '1' THEN
     FOR i IN 0 TO 31 LOOP
       Q(i) <= Q(i+1)
     END LOOP;                              SR
     Q(31) <= IN;
   END IF
   IF (Q = OTHERS => 1) THEN
      Z <= 0; -- active low                 AND gate
   ELSE Z <= 1; -- active low               and
                                            DFF
   END IF;
ENDIF

PAR <= PDQ

END PROCESS
END behaviorArch;
```

~~10~~ pts.
12 pts

3. Using the GENERIC feature of VHDL, complete the following code that will define a "generic" component named genMUXDff shown in Figure 3(a). The generic component has "N" slices, each of which contains a D flip-flop and a 2-to-1 MUX as shown in Figure 3(b). Also, the output ZFlag is true when all the D flip-flops contain '0'.

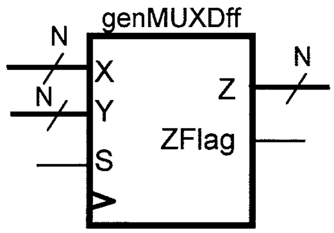


Figure 3(a)

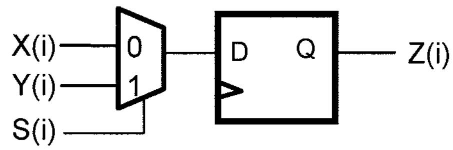


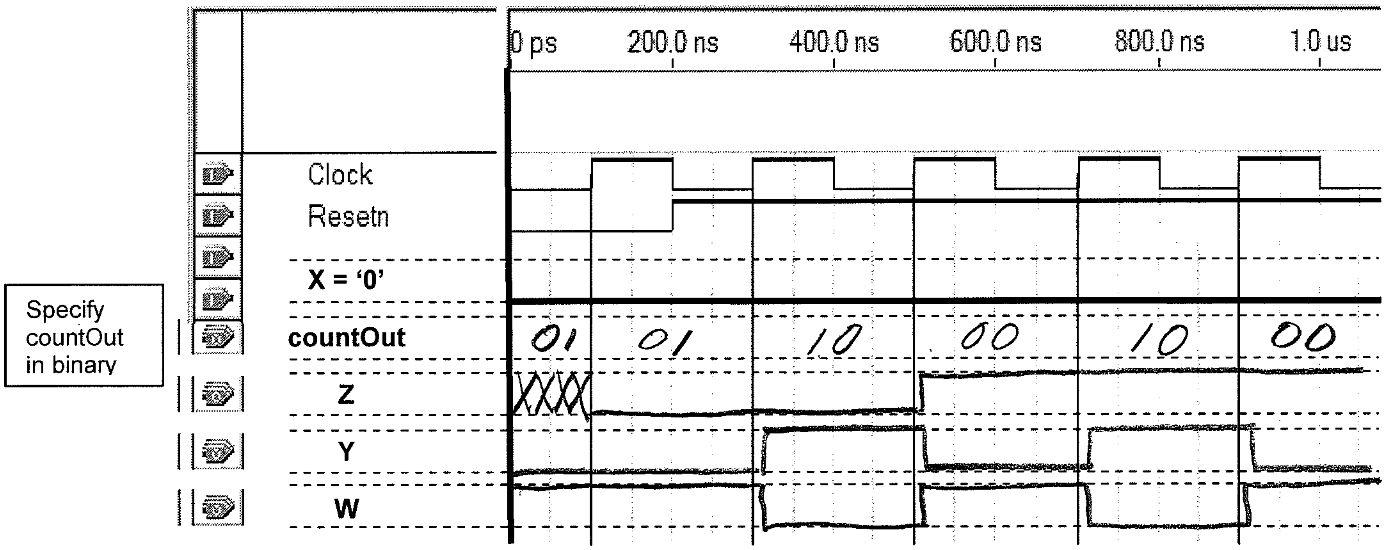
Figure 3(b)

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
ENTITY genMUXDff IS
    GENERIC (N: INTEGER :=8)
    PORT ( X, Y : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
           S, CLK : IN STD_LOGIC;
           Z : OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0);
           ZFLAG : OUT STD_LOGIC);
END genMUXDff;
ARCHITECTURE genArch OF genMUXDff IS
SIGNAL
BEGIN
  PROCESS ( CLK )  -- All code should be inside a single
                   -- PROCESS block
FF    IF (CLK'EVENT AND CLK='1') THEN
         IF S = '0' THEN
            Z <= X;
         ELSE
            Z <= Y;
         END IF;
      END IF;
MUX   IF Z (OTHERS => '0') THEN
            ZFLAG <= '1';
      ELSE
            ZFLAG <= '0';
      END IF;
  END PROCESS;
END genArch;
```

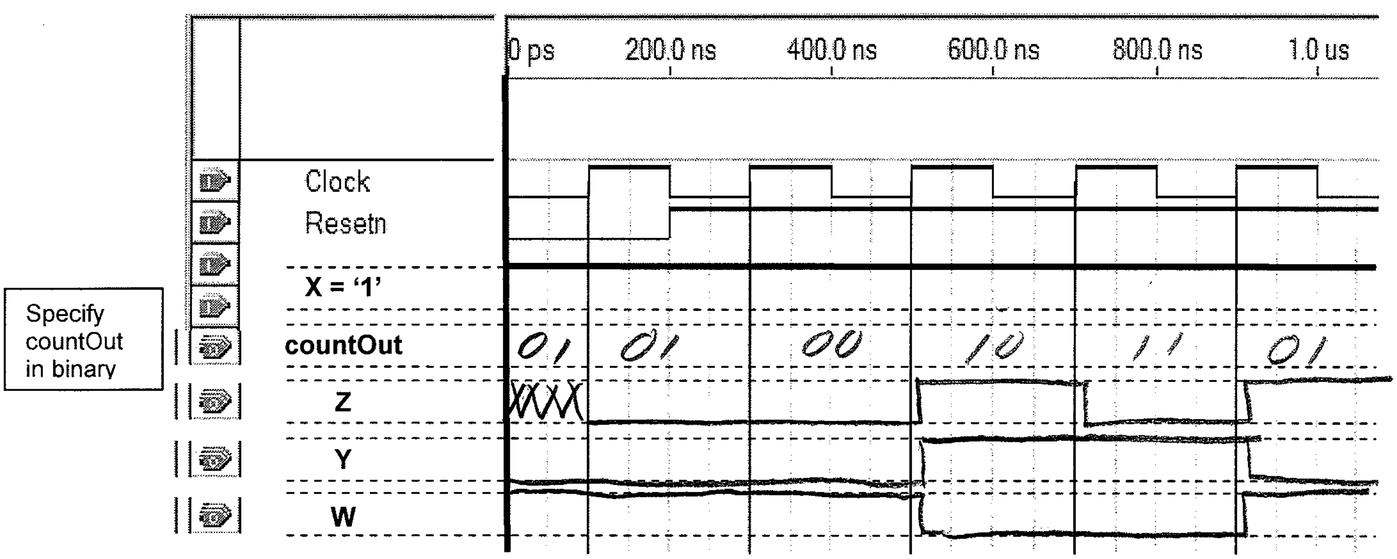
24 pts

4. Based on on the VHDL code on the next page, complete the following timing diagrams.

(a) Complete the following timing diagram for X = ‘0’. Please show delays and go as far as you can. 12 pts



(a) Complete the following timing diagram for X = ‘1’. Please show delays and go as far as you can. 12 pts



EEL 4712 – Digital Design
Test 1 – Spring Semester 2009 Name ______________________________

Code used for Problem 4:

```
ENTITY T1Prob4 IS
	PORT ( Clock, Resetn, X		: IN	STD_LOGIC ;
		countOut		: OUT	STD_LOGIC_Vector (1 DOWNTO 0);
		Y,Z,W			: OUT	STD_LOGIC ) ;
END T1Prob4 ;

ARCHITECTURE Behavior OF T1Prob4 IS
	SIGNAL count : STD_LOGIC_Vector (1 DOWNTO 0);
BEGIN
	WITH count SELECT
		W <= '1' WHEN "00",
		     '1' WHEN "01",
		     '0' WHEN OTHERS;
	PROCESS ( Resetn, Clock )
	BEGIN
		IF Resetn = '0' THEN
			count <= "01" ;
		ELSIF (Clock'EVENT AND Clock = '1') THEN
			Z <= '0';
			CASE count IS
				WHEN "10" =>
					IF X = '0' THEN
						count <= "00" ;
						Z <= '1';
					ELSE  count <= "11" ;
					END IF ;
				WHEN "11" =>
					count <= "01" ;
					Z <= '1';
				WHEN "01" =>
					IF X = '0' THEN count <= "10";
					ELSE count <= "00";
					END IF ;
				WHEN "00" =>
					count <= "10";
					Z <= '1';
				WHEN OTHERS =>
					count <= "10";
			END CASE ;
		END IF ;
	END PROCESS ;

	PROCESS (count, X)
	BEGIN
		CASE count IS
			WHEN "10" => Y <= '1';
			WHEN "11" => IF X = '1' THEN Y <= '1';
			             ELSE Y <= '0';
			             END IF;
			WHEN OTHERS => Y <= '0';
		END CASE;
		countOut <= count;

	END PROCESS;
 END Behavior ;
```

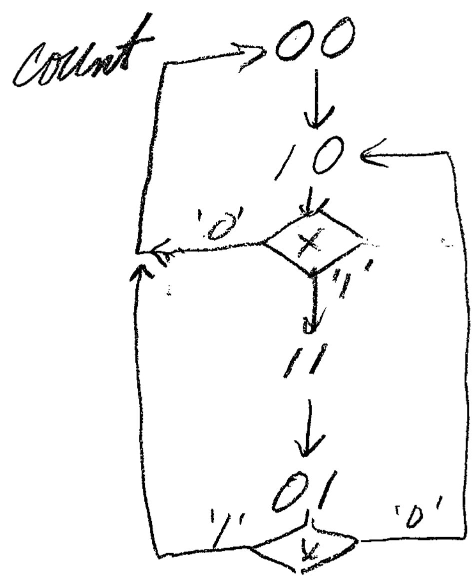
W is true in 00, 01

W is combinatorial

asynch reset to count = "01"

clock'event

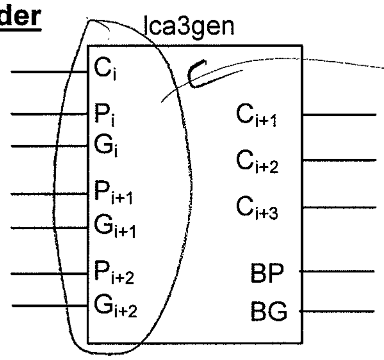
Z is through flipflop for count = "00" "10" AND X='0' "11"



Y is combinatorial

Y = '1' in count = 10, 11 (X=1)

~~18 pts.~~ 18pts **5.** **3-bit LCA Adder**



Some useful equations:

S_{i+1} = A_i XOR B_i XOR C_i
C_{i+1} = A_i AND B_i OR (A_i OR B_i) AND C_i
G_i = A_i AND B_i
P_i = A_i OR B_i

Show above is a 3-bit look-ahead carry LCA generator (lca3gen). It functions exactly like the lca2gen you designed in Lab 3, except it is a 3-bit slice.

(a) Give the necessary equations for lca3gen. (in SOP and functions of inputs)

(5 pts)

$$C_{i+1} = G_i + P_i C_i$$

$$C_{i+2} = G_{i+1} + P_{i+1} C_{i+1} = G_{i+1} + P_{i+1} G_i + P_{i+1} P_i C_i$$

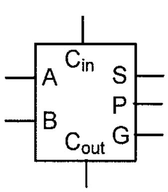
$$C_{i+3} = G_{i+2} + P_{i+2} C_{i+2} = G_{i+2} + P_{i+2} G_{i+1} + P_{i+2} P_{i+1} G_i + P_{i+2} P_{i+1} P_i C_i$$

$$BP = P_{i+2} \cdot P_{i+1} \cdot P_i$$

$$BG = G_{i+2} + P_{i+2} G_{i+1} + P_{i+2} P_{i+1} G_i$$

(13 pts)

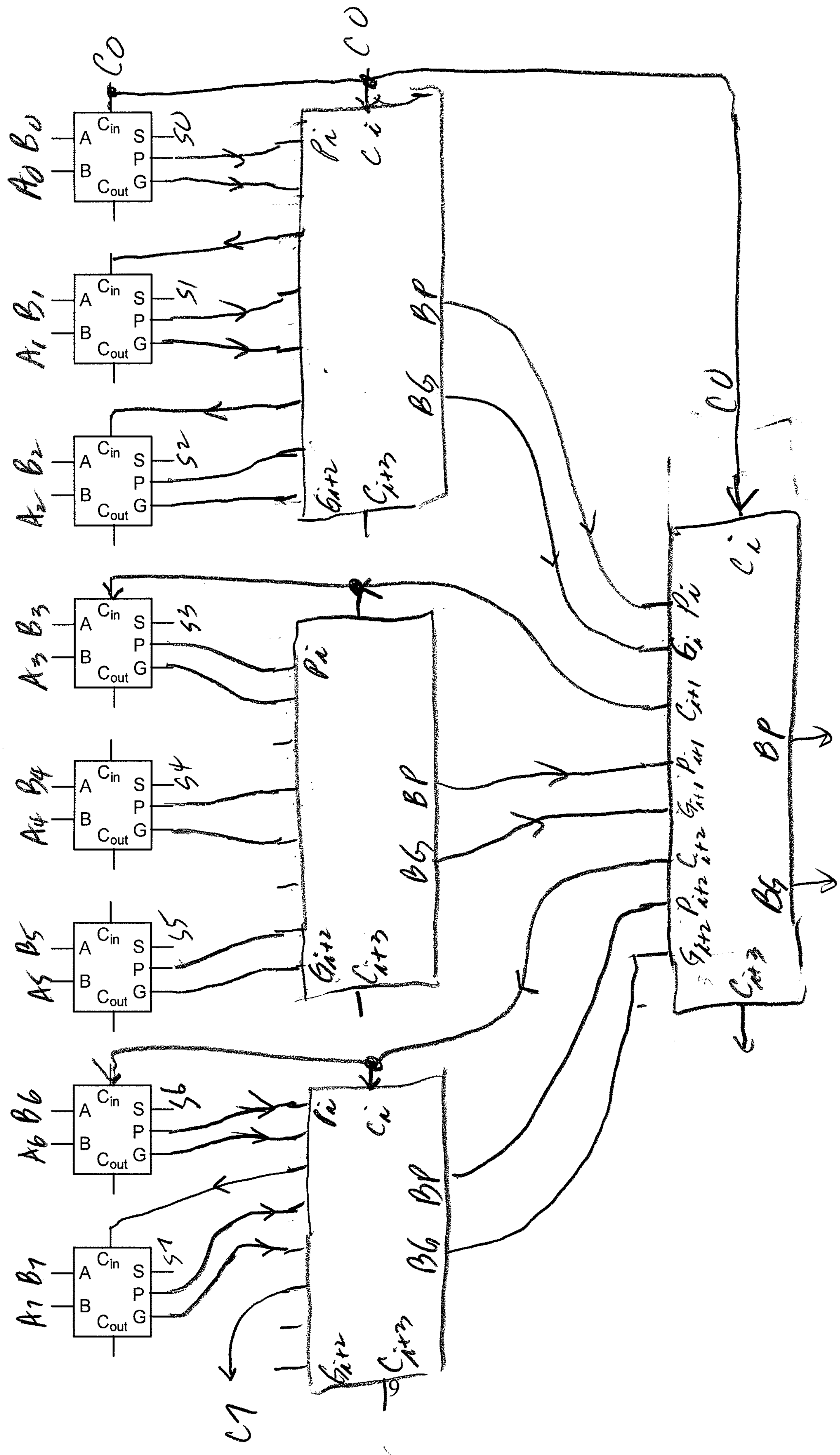
(b) Using 8 of the following full adder components, any number of lca3gen, and any additional gates, draw the circuit diagram of an 8-bit adder with carry lookahead (**on the next page**).



Full adder, with P and G

- The circuit diagram should show exactly how the components are connected. However, you can use labels when appropriate, instead of drawing the connections.

EEL 4712 – Digital Design
Test 1 – Spring Semester 2009 Name ______________________________



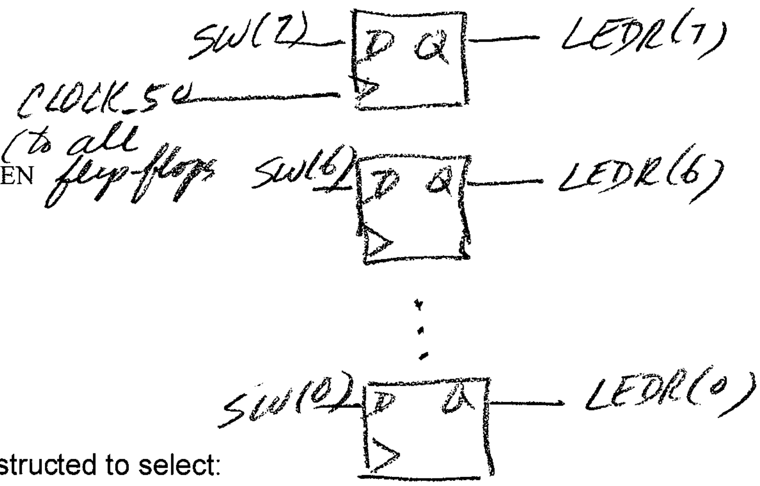
EEL 4712 – Digital Design
Test 1 – Spring Semester 2009 Name ______________________________

[10 pts) 6. EEL4712 Trivia.

(3 pts) (a) The following is the VHDL file given in the SignalTap II tutorial. Draw the corresponding cirucuit. Note that RISING_EDGE(CLOCK_50) is equivalent to (CLOCK_50'EVENT AND CLOCK_50 = '1')

```
ENTITY switches IS
    PORT ( CLOCK_50 : IN STD_LOGIC;
        SW : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
        LEDR : OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
    END switches;
ARCHITECTURE Behavior OF switches IS
BEGIN
    PROCESS (CLOCK_50)
        BEGIN
        IF(RISING_EDGE(CLOCK_50)) THEN
            LEDR <= SW;
        END IF;
    END PROCESS;
END Behavior;
```

Put circuit for Part (a) here.



(3 pts) (b) In the SignalTap II tutorial, you were instructed to select:

- SW[7..0] as the nodes to probe
- CLOCK_50 as clock to run the SignalTap module
- SW[0] as the trigger condition and select "high" for it

After you downloaded the design onto the board, briefly explain how the SignalTap works when the analysis is run.

Initially, SignalTap will be waiting for the trigger condition to be true (SW(0) = '1'). When SW(0) is set to '1', the data window of SignalTap will begin the display of the probed signals, SW(7..0) at the rate of CLOCK_50.

(2 pts) (c) In performance point of view, for combinatorial circuits, is it better to use a Process block or concurrent statements (e.g., WITH SELECT)? For credit, please explain.

We don't know. It depends on how Quartus will synthesize the circuit

(2 pts) (d) In performance point of view, for sequential circuits (e.g., registers), is it better to use a Process block or concurrent statements? For credit, please explain.

We cannot specify sequential circuits with a concurrent statement like WITH SELECT.

```
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;
```