1. VHDL analysis: timing diagram

Shown below is the VHDL specification of a circuit. Analyze the VHDL code and complete the timing diagrams on the next page.

ENTITY Prob1 IS
  PORT ( Clock, Resetn, X, R : IN  STD_LOGIC ;
         Q, Y, Z : OUT  STD_LOGIC ) ;
END Prob1 ;

ARCHITECTURE Behavior OF Prob1 IS
  SIGNAL tsig : STD_LOGIC_Vector (1 DOWNTO 0);
  BEGIN
      PROCESS (tsig, R)
      BEGIN
          CASE tsig IS
              WHEN "01" =>
                  IF R = '0' THEN
                      Z <= '0' ;
                  ELSE
                      Z <= '1';
                  END IF ;
                  Y <= '0';
              WHEN OTHERS =>
                  Y <= '1';
          END CASE;
      END PROCESS;

      PROCESS ( Resetn, Clock )
      BEGIN
          IF Resetn = '0' THEN
              tsig <= "01" ;
              Q <= '0';
          ELSIF (Clock'EVENT AND Clock = '1') THEN
              CASE tsig IS
                  WHEN "00" =>
                      IF X = '0' THEN tsig <= "01" ;
                      ELSE tsig <= "11" ;
                      END IF ;
                  WHEN "01" =>
                      IF R = '1' THEN tsig <= "00" ;
                      ELSE tsig <= "11" ;
                      Q <= '1';
                      END IF ;
                  WHEN "11" =>
                      tsig <= "01" ;
                      Q <= 0;
                  WHEN OTHERS =>
                      tsig <= "00";
          END CASE ;
      END PROCESS ;

  END Behavior ;
1. Put answer here.

Complete the timing diagram for tsig, Q, Y, and Z. For maximum credit, show delays.
(12 pts)
2. **VHDL specification.**

Create an 8-bit ALU using a behavioral architecture with the `numeric_std` package with the following operations:

<table>
<thead>
<tr>
<th>Sel</th>
<th>Output</th>
<th>Overflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td><code>input1 + input2</code> (just like Lab 3)</td>
<td><code>'1' if </code>input1 + input2<code>is bigger than the maximum number that can be written to</code>output<code>, </code>'0' otherwise (just like Lab 3)</td>
</tr>
<tr>
<td>01</td>
<td>Swap the high-half bits of <code>input1</code> with the low-half bits of <code>input1</code>, write this to <code>output</code> (just like Lab 3)</td>
<td>`'0' if the result is &quot;00000000&quot;, '1' otherwise (unlike Lab 3)</td>
</tr>
<tr>
<td>10</td>
<td>One's complement of <code>input1</code>: Invert each bit</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>Two's complement of <code>input1</code>: One's complement plus 1</td>
<td>0</td>
</tr>
</tbody>
</table>

**The entity specification is as follows:**

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

entity alu_ns is
port ( input1  : in std_logic_vector(7 downto 0);
       input2  : in std_logic_vector(7 downto 0);
       sel     : in std_logic_vector(1 downto 0);
       output : out std_logic_vector(7 downto 0);
       overflow : out std_logic );
end alu_ns;
```

You are to complete the following ARCHITECTURE specification (below and on the next page) to implement this ALU.

**ARCHITECTURE** Behavior OF alu_ns IS -- answer for Problem 2

-- SIGNAL definitions
2. (continued)
   BEGIN – You **cannot** use a **PROCESS** block
   **WITH sel SELECT** – You **must** use this statement to define the (temp) ALU output.

   -- You **must** use a **conditional assignment** to define the **overflow** signal.
   -- You can use **simply assignment** statements for other signals.

END Behavior;
The following VHDL code is used for Problem 3 and Problem 4:

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

ENTITY Test1P3_4 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 Test1P3_4 ;

ARCHITECTURE Behavior OF Test1P3_4 IS
    SIGNAL Count : STD_LOGIC_VECTOR (3 DOWNTO 0) ;
BEGIN

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

        IF (Count = "0000") THEN -- This IF statement is used for Problem 4(a)
            Z1 <= '1';
        ELSE
            Z1 <= '0';
        END IF;

        IF Resetn = '0' THEN
            Count <= "0000" ;
        ELSIF (Clock'EVENT AND Clock = '1') THEN
            forloop: FOR i IN 0 TO 3 LOOP
                Count <= Count + 1 ;
                Z2 <= Count(0);
            END LOOP;
            IF (Count = "0000") THEN -- This IF statement is used for Problem 4(b)
                Z3 <= '1';
            ELSE
                Z3 <= '0';
            END IF;
        ELSE
            Count <= Count ;
        END IF;

        QQA <= Count;
        QQB <= Count;
        QQC <= Count ;

    END PROCESS ;

END Behavior ;
3. VHDL analysis – Timing diagram.

(a) Based on the code shown on Page 5, complete the following timing diagram for the values for Count (in binary): (5 pts)

(b) Assume the given Count values, complete the following timing diagram based on the code shown on Page 8 (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.** (18 pts)
4. VHDL Analysis and re-code

(a) Given the IF statement at the top of the code shown on Page 5 (for Z1), convert it into a conditional assignment statement. (4 pts)

**IMPORTANT:** If necessary, you need to tell me what else you need to do to maintain the behavior of the original code (e.g., location of the new code, add additional code, etc.).

(b) Given the IF statement in the middle of the code shown on Page 5 (for Z3), convert it into a SELECT assignment statement. (6 pts)

**IMPORTANT:** If necessary, you need to tell me what else you need to do to maintain the behavior of the original code (e.g., location of the new code, add additional code, etc.).
5. VGA Lab

![Horizontal Refresh Cycle](image)

Given above is the horizontal refresh cycle for the VGA lab. Briefly define the following:

(a) A _______________________________________________________________________

(b) B _______________________________________________________________________

(c) C _______________________________________________________________________

(d) D _______________________________________________________________________

(e) E _______________________________________________________________________

<table>
<thead>
<tr>
<th>Parameters</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>31.77 μs</td>
<td>3.77 μs</td>
<td>1.69 μs</td>
<td>25.17 μs</td>
<td>0.94 μs</td>
</tr>
</tbody>
</table>

Figure 1. Horizontal Refresh Cycle.

Given above is the horizontal refresh cycle for the VGA lab. Briefly define the following:

(a) A _______________________________________________________________________

(b) B _______________________________________________________________________

(c) C _______________________________________________________________________

(d) D _______________________________________________________________________

(e) E _______________________________________________________________________

6. VHDL test bench.

Given the following entity definition for a generic adder:

```vhdl
entity adder is
  generic (WIDTH : positive := 8);
  port map (input1, input2 : in std_logic_vector (WIDTH-1 downto 0);
    carry_in : in std_logic;
    sum: out std_logic_vector (WIDTH-1 downto 0);
    carry_out: out std_logic);
end adder;
```

Modify the testbench on the next page:

- To test the “adder” entity as a **16-bit** adder.
- Make the testbench code **general**; i.e., can be used to test for any number of bits by changing only the value of **NBITS**.
6. (continued). This is a copy of the adder_tb that you used in the labs, except:
   - I made changes to it (everything that is **bolded**).
   - You need to make all necessary changes to make it work with the generic adder from
     the previous page (**put changes right this page**)

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

entity adder_tb is
end adder_tb;

architecture TB of adder_tb is
constant NBITS : positive: 16;
signal input1, input2, sum : std_logic_vector(3 downto 0);
signal carry_in, carry_out : std_logic;
begin  -- TB
UUT : entity work.adder
   port map (input1    => A,
              input2    => B,
              carry_in  => C,
              sum       => D,
              carry_out => E);
end TB;

function X (constant a, b, c: integer) return std_logic is
begin
   if (a + b + c > 15) then return '1';
   else return '0';
   end if;
end X;

function Y (constant a, b, c: integer) return std_logic is
begin
   return std_logic_vector(to_unsigned((a+b-c) mod 16, 4));
end Y;

function Z (constant a, b, c: integer) return std_logic_vector is
begin
   return std_logic_vector(to_unsigned((a+b+c) mod 16, 4));
end Z;
ENTITY __entity_name IS
  PORT(__input_name, __input_name, __input_vector_name, __bidir_name, __bidir_name, __output_name, __output_name
      : IN STD_LOGIC;
      : IN STD_LOGIC_VECTOR(__high downto __low);
      : INOUT STD_LOGIC;
      : 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
  GENERIC MAP (__component_par =>__connect_par)
  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;
1.3 OVERLOADED OPERATORS

<table>
<thead>
<tr>
<th>Description</th>
<th>Left Operator</th>
<th>Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>bitwise-and</td>
<td>u&amp;&amp;v/v</td>
<td>and,&amp;&amp;</td>
</tr>
<tr>
<td>bitwise-or</td>
<td>u||v/v or</td>
<td>or,||v/v</td>
</tr>
<tr>
<td>bitwise-xor</td>
<td>u^^v/v xor</td>
<td>xor,^^v/v</td>
</tr>
<tr>
<td>bitwise-not</td>
<td>not u/v</td>
<td>not</td>
</tr>
</tbody>
</table>

1.4 CONVERSION FUNCTIONS

<table>
<thead>
<tr>
<th>From</th>
<th>To Bit</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>b</td>
<td>TO_BIT(from u, xmap)</td>
</tr>
<tr>
<td>uv</td>
<td>bv</td>
<td>TO_BITVECTOR(from uv, xmap)</td>
</tr>
<tr>
<td>b</td>
<td>u</td>
<td>TO_STD_LOGIC(from)</td>
</tr>
<tr>
<td>bv</td>
<td>uv</td>
<td>TO_STD_LOGIC_VECTOR(from)</td>
</tr>
<tr>
<td>bv</td>
<td>uv</td>
<td>TO_STD_LOGIC_VECTOR(from)</td>
</tr>
</tbody>
</table>

2. IEEE'S NUMERIC_STD

2.1 PREDEFINED TYPES

<table>
<thead>
<tr>
<th>UNSIGNED na</th>
<th>Array of STD_LOGIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIGNED na</td>
<td>Array of STD_LOGIC</td>
</tr>
</tbody>
</table>

2.2 OVERLOADED OPERATORS

<table>
<thead>
<tr>
<th>Left</th>
<th>Op</th>
<th>Right</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs</td>
<td></td>
<td>sg</td>
<td>sg</td>
</tr>
<tr>
<td>-</td>
<td></td>
<td>sg</td>
<td>sg</td>
</tr>
<tr>
<td>un</td>
<td>+,*</td>
<td>rem, mod</td>
<td>un un</td>
</tr>
<tr>
<td>sg</td>
<td>+,*</td>
<td>rem, mod</td>
<td>sg sg</td>
</tr>
<tr>
<td>na</td>
<td>+,*</td>
<td>rem, mod</td>
<td>na na</td>
</tr>
<tr>
<td>in</td>
<td>+,*</td>
<td>rem, mod</td>
<td>in in</td>
</tr>
<tr>
<td>un</td>
<td>&lt;,&gt;</td>
<td>eq, ne</td>
<td>un bool</td>
</tr>
<tr>
<td>sg</td>
<td>&lt;,&gt;</td>
<td>eq, ne</td>
<td>sg bool</td>
</tr>
<tr>
<td>na</td>
<td>&lt;,&gt;</td>
<td>eq, ne</td>
<td>na bool</td>
</tr>
<tr>
<td>in</td>
<td>&lt;,&gt;</td>
<td>eq, ne</td>
<td>in bool</td>
</tr>
</tbody>
</table>

2.3 PREDEFINED FUNCTIONS

| SHIFT_LEFT | un |
| SHIFT_RIGHT | un |
| ROTATE_LEFT | un |
| ROTATE_RIGHT | un |
| RESIZE | un |

3.3 PREDEFINED FUNCTIONS

| SHIFT_LEFT(tun, na) | un |
| SHIFT_RIGHT(tun, na) | un |
| ROTATE_LEFT(tun, na) | un |
| ROTATE_RIGHT(tun, na) | un |
| RESIZE(tun, na) | un |

3.4 CONVERSION FUNCTIONS

<table>
<thead>
<tr>
<th>From</th>
<th>To Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>buv</td>
<td>sg</td>
</tr>
<tr>
<td>sg</td>
<td>bv</td>
</tr>
<tr>
<td>sg</td>
<td>in</td>
</tr>
<tr>
<td>na</td>
<td>un</td>
</tr>
<tr>
<td>in</td>
<td>sg</td>
</tr>
</tbody>
</table>
**IMPORTANT:** Please be neat and write (or draw) carefully. If we cannot read it with a reasonable effort, it is assumed wrong.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(22 pts)</td>
</tr>
<tr>
<td>2</td>
<td>(20 pts)</td>
</tr>
<tr>
<td>3</td>
<td>(23 pts)</td>
</tr>
<tr>
<td>4</td>
<td>(10 pts)</td>
</tr>
<tr>
<td>5</td>
<td>(5 pts)</td>
</tr>
<tr>
<td>6</td>
<td>(20 pts)</td>
</tr>
</tbody>
</table>

**Total**

---

**Re-Grade Information:**

_______________________________________________________________________
_______________________________________________________________________
_______________________________________________________________________
_______________________________________________________________________
_______________________________________________________________________
_______________________________________________________________________
_______________________________________________________________________
_______________________________________________________________________
_______________________________________________________________________
_______________________________________________________________________
_______________________________________________________________________
_______________________________________________________________________
_______________________________________________________________________