EJ.06/Isc 3.5/ AC5

VHDL 2008 SOLUTIONS
A=Analysis, D = Design, B = Bookwork

Question 1 is COMPULSORY, and constitutes 40% of marks, 72 minutes time, 15 minutes per part.

Solution to Question 1.

1.

a)

(i) It has an incomplete sensitivity list – changes in the missing signals, e.g. y in P1 below will not change output pre-synthesis, but will be correctly reflected in output post-synthesis.

```
P1: PROCESS(x)
BEGIN
a <= x AND y;
END PROCESS P1;
```

(ii) It has an incomplete IF or CASE statement with a signal not driven on one branch e.g. y in P2 below:

```
P2: PROCESS(x)
BEGIN
IF x='1' THEN y <= '1'; END IF;
IF z='1' THEN y<='0'; END IF;
END PROCESS P2;
```

This will simulate and synthesise as a latch, not combinational logic.

[4A]

b)

```
ENTITY demult IS -from question
  GENERIC( K: INTEGER);
  PORT(
    addr: IN STD_LOGIC_VECTOR(K-1 DOWNTO
0);
    y: IN STD_LOGIC;
    x: OUT STD_LOGIC_VECTOR(2**K-1 DOWNTO
0)
    );
END demult;

ARCHITECTURE synth OF demult IS
BEGIN
  x(UNSIGNED(addr))<=y;
END ARCHITECTURE synth;
```

Instantiate with $K = 9$ for 512 outputs (*addr* will have width 9).

[4D]

c)

```
LIBRARY IEEE;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

-- from question
ENTITY mul IS
  PORT(
    a: OUT UNSIGNED(17 DOWNTO 0);
    b,c: IN UNSIGNED(5 DOWNTO 0);
    d: IN SIGNED(3 DOWNTO 0)
    );
END mul;


ARCHITECTURE synth OF mul IS
  SIGNAL t1: UNSIGNED(7 DOWNTO 0);

BEGIN
  t1 <= UNSIGNED(d*d)+c;
  a(13 DOWNTO 0)  <= t1*b;
  a(17 DOWNTO 14) <= "0000";
END ARCHITECTURE synth;
```

[4D]

d)

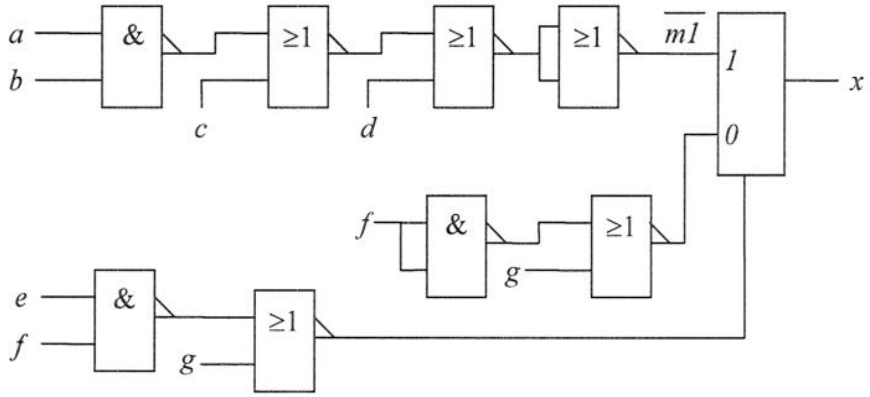
```
LIBRARY IEEE;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY count IS
  PORT(
    clk: IN STD_LOGIC;
    max,min: OUT STD_LOGIC;
    m: IN STD_LOGIC_VECTOR(1 DOWNTO 0);
    x: OUT STD_LOGIC_VECTOR(5 DOWNTO 0)
    );
END ENTITY count;

ARCHITECTURE synth OF count IS

  SIGNAL x_int: UNSIGNED(5 DOWNTO 0);

BEGIN

  x <= STD_LOGIC_VECTOR(x_int);

  P1: PROCESS(x_int)
  BEGIN
    max <= '0'; min <= '0';
    IF UNSIGNED(x_int)=63 THEN max<='1'; END IF;
    IF UNSIGNED(x_int)=0 THEN min <= '1'; END IF;
  END PROCESS P1;

  P2: PROCESS
  BEGIN
    WAIT UNTIL clk'EVENT AND clk='0';
    CASE m IS
      WHEN "00" => NULL;
      WHEN "01" => IF x_int /= 0 THEN x_int <= UNSIGNED(x_int)-1; END IF;
      WHEN "10" => IF x_int /= 63 THEN x_int <= UNSIGNED(x_int)+1; END IF;
      WHEN "11" => x_int <= (OTHERS=>'0');
    END CASE;
  END PROCESS P2;



END ARCHITECTURE synth;
```

[8D]

Students must answer two questions from questions 2-4, each question caries 30% of marks and takes 54 minutes.

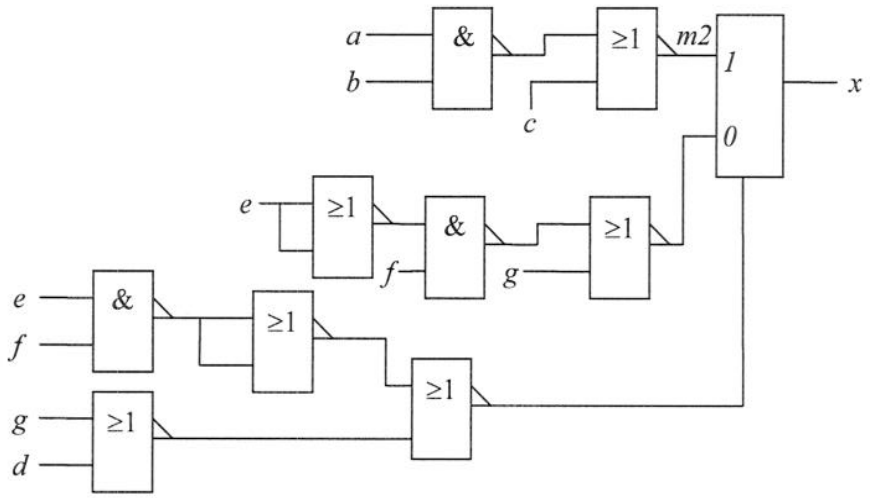
Solution to Question 2

a)



[8A]

b)



[8A]

c)

`original 6, m1 5, m2 4.`

[4A]

Solution to Question 3

a)

```
LIBRARY IEEE;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY digit_count IS
  PORT(
    clk, cin: IN STD_LOGIC;
    cout: OUT STD_LOGIC;
    count: OUT STD_LOGIC_VECTOR(3 DOWNTO 0)
    );
END ENTITY digit_count;
-- architecture of digit_count not required

ARCHITECTURE struct OF bcd12 IS
  SIGNAL carry: STD_LOGIC_VECTOR(0 TO 12);
BEGIN
  carry(0) <= cin;
  cout <= carry(12);

  G1: FOR i IN 0 TO 11 GENERATE
    I1: ENTITY digit_count PORT MAP(clk=>clk, cin=>carry(i),cout=>carry(i+1),count=>count(i));
  END GENERATE G1;

END struct;
```

NB – could use two PROCESSs with two FOR LOOPs instead.

[8D]

b)

```
ARCHITECTURE behav OF bcd12 IS
  SIGNAL count_int: digits;
  SIGNAL carry: STD_LOGIC_VECTOR(0 TO 12);
BEGIN
  carry(0) <= cin;
  cout <= carry(12);

  count <= count_int;

  G1: FOR i IN 0 TO 11 GENERATE

    P1: PROCESS
    BEGIN
      WAIT UNTIL clk'EVENT AND clk='1';
      IF carry(i)='1' THEN
        count_int(i)<=STD_LOGIC_VECTOR(UNSIGNED(count_int(i))+1);
      END IF;
      IF count_int(i)="1001" THEN count_int(i)<="0000"; END IF;
    END PROCESS P1;

    P2: PROCESS(carry,count_int)
    BEGIN
      carry(i+1) <= '0';
      IF carry(i)='1' AND count_int(i)="1001" THEN
            carry(i+1)<='1';
      END IF;
    END PROCESS P2;

  END GENERATE G1;

END ARCHITECTURE behav;
```

[8D]

c) Add a reset input – use this in architecture to set count to 0.

[4D]

Solution to Question 4

a)

Enumeration types have initial value their first state, whereas real hardware implementations have undefined initial state value. Adding a reset signal will allow correct operation but simulation will not check that reset is correctly implemented. Solution is to add an extra initial state "bad" which should never happen but will be the initial value when simulated pre-synthesis.

[5B]

b)

```
ENTITY timing IS - from question
  PORT (
    clk, reset    : IN STD_LOGIC;
    n, m  : IN STD_LOGIC_VECTOR(9 DOWNTO 0);
    x : OUT STD_LOGIC
    );
END timing;

ARCHITECTURE behav OF timing IS

  TYPE states IS (bad, init, xisone, waitloop);
  TYPE statearr IS ARRAY (states) OF INTEGER;
  SIGNAL state: states;
  CONSTANT statelength: statearr := (bad=>0, init=>100, xisone=>33,waitloop=>60);
  SIGNAL count: UNSIGNED(6 DOWNTO 0);
BEGIN

  P1: PROCESS
  BEGIN
    WAIT UNTIL clk'EVENT AND clk='1';
    IF reset='1' THEN
      state<=init;
      count <=(OTHERS=>'0');
    ELSE
      count <= count + 1;
      IF statelength(state)-1=count THEN
        count <= (OTHERS=>'0');
        CASE state IS
          WHEN init => IF UNSIGNED(n) < 10 AND UNSIGNED(n) > UNSIGNED(m)
                       THEN
                         state <= waitloop;
                       ELSE
                         state <= xisone;
                         x <= '1';
                       END IF;
          WHEN waitloop => state <= init;
          WHEN xisone => state <= init; x <= '0';
          WHEN bad => NULL;
        END CASE;
      END IF;
    END IF;
  END PROCESS P1;
END behav;
```

[15D]