VHDL 2007 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) P1 inverts z on every edge of clk. It is therefore not synthesisable, since flip-flops must trigger on one edge not both.

(ii) P2 has identical semantics to a positive edge triggered T flip-flop, so it synthesises.

[4A]

b)

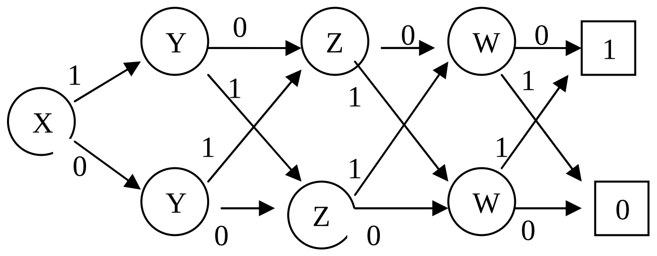
```
--from question
 ENTITY ram IS
 PORT(
      write, clk: IN std_logic;
      addr: IN std_logic_vector(3 DOWNTO 0);
      din: IN std_logic(7 DOWNTO 0);
      dout: OUT std_logic_vector(7 DOWNTO 0)
 );
 END ram;


 ARCHITECTURE synth OF ram IS
   TYPE ramtype IS ARRAY (0 TO 15) OF std_logic_vector( 7 DOWNTO 0);
   SIGNAL ramloc : ramtype;
 BEGIN
   dout <= ramloc(conv_integer(unsigned(addr));

   P1: PROCESS
   BEGIN
     WAIT UNTIL clk'EVENT AND clk='1';
     IF write = '1' THEN
     ramloc(conv_integer(unsigned(addr))<=din;
     END IF;
   END PROCESS P1;
 END ARCHITECTURE synth;
```

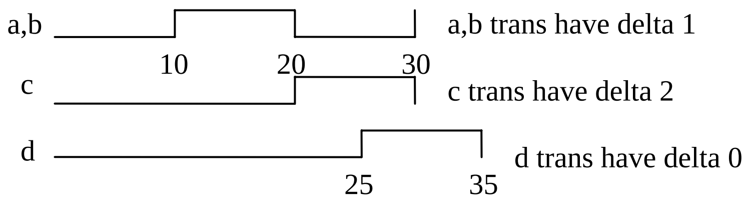
[4D]

c)


X
Y
Y
Z
Z
W
W
1
0
1
0
1
0
1
0
1
0
1
1
1
1
1
1
0
0
0
0

[4B]

d)


a,b
c
d
10
20
30
25
35
a,b trans have delta 1
c trans have delta 2
d trans have delta 0

[4A]

e)

```
--from question
ENTITY iobuff IS
    PORT(dir,clk:IN std;
     za,zb:INOUT std_logic
);
END iobuff;

ARCHITECTURE synth OF iobuff IS
   SIGNAL za_del, zb_del: std_logic;
BEGIN

P1: PROCESS
BEGIN
   WAIT UNTIL clk’EVENT AND clk=’1’;
   za_del <= za;
   zb_del <= zb;
END PROCESS P1;


P2: PROCESS(dir, za_del, zb_del)
BEGIN
   WAIT UNTIL clk’EVENT AND clk=’1’;
   IF dir=’1’ THEN
      zb <= za_del; za <= ’Z’;
   ELSE
      za <= zb_del; zb <= ’Z’;
   END IF;
END PROCESS P2;

END synth;
```

[4D]

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

Solution to Question 2

3.

a)

Must be in a package, so:

```
PACKAGE mypak IS
SUBTYPE word IS std_logic_vector(15 DOWNTO 0);
END PACKAGE;
```

[3B]

b)

```
ENTITY tram IS
PORT(
  din: IN word;
  dout: OUT word;
  clk, write: IN std_logic;
  addr: In std_logic_vector(3 DOWNTO 0)
);
END tram;
```

[2B]

c)

```
ARCHITECTURE synth OF transpose IS
  TYPE fsmtype IS (state1, state2, state3);
  SIGNAL state: fsmtype;
  SIGNAL regs: word;
  SIGNAL write, wen, dsel, asel: std_logic;
  SIGNAL data, adata : word;
  SIGNAL count, addr : STD_LOGIC_VECTOR(3 DOWNTO 0);

BEGIN

  I1: ENTITY tram PORT MAP(datain, dataout, clk, write, addr);
   ALU : PROCESS
  BEGIN
    WAIT UNTIL clk'EVENT AND clk = '1';
    IF state=state2 THEN
      regs <= SIGNED(regs)+SIGNED(data);
    END IF;
    IF reset='1' THEN regs<= (OTHERS=>'0'); END IF;
  END PROCESS ALU;

  COUNT : PROCESS
  BEGIN
    WAIT UNTIL clk'EVENT AND clk = '1';
    IF reset='1'OR state=state3 THEN
      count <= "0000";
    ELSE
      count <= UNSIGNED(count) + 1;
    END IF;
  END PROCESS COUNT;


  MUX : PROCESS(state, data, adata, count)
  BEGIN
     IF state=state3 THEN dataout <= adata; ELSE dataout <=data; END IF;
     addr <= count;
     IF state=state2 THEN
        addr <= (count(1), count(0), count(3), count(2));
     END IF;
  END PROCESS MUX;

  FSM : PROCESS
  BEGIN
    WAIT UNTIL clk'EVENT AND clk='1';
    IF reset THEN state <= state1;
    ELSE
       IF count=std_logic_vector'("1111") THEN
              CASE state IS
                     WHEN state1 => state <= state2;
                     WHEN state2 => state <= state3;
              END CASE;
       END IF
    END IF;
  END PROCESS FSM;

END ARCHITECTURE synth;  -- of transpose
```

[15D]

Solution to Question 3

a)

```
FUNCTION roundn(n: INTEGER; x: SIGNED) RETURN SIGNED IS
VARIABLE xx:SIGNED(x'LENGTH DOWNTO 1);
CONSTANT zeros: SIGNED(N DOWNTO 1):= (OTHERS => '0');
BEGIN
  xx := x;
  xx := signed(xx)+ unsigned(xx(n) & zeros);
  FOR i IN n DOWNTO 1 LOOP xx(i) := '0'; END LOOP;
  RETURN xx;
END;
```

[4D]

b) cb, ca, $cb+cb^2$, $ca+ca^{2.}$. The circuit computes the Mandelbrot recurrence for two values of c in parallel.

[4D]

c)

```
TYPE complex IS ARRAY (1 DOWNTO 0) OF SIGNED(m-1 DOWNTO 0);
(Element type could be std_logic_vector)
```

[2B]

d)

```
ARCHITECTURE synth OF mandelbrot IS
  SIGNAL z_int, c : complex;
  SIGNAL zii, zir, zrr : SIGNED(2*m-1 DOWNTO 0);
  SIGNAL count         : STD_LOGIC;

FUNCTION roundn(n: INTEGER; x: SIGNED) return SIGNED IS
VARIABLE xx:SIGNED(x'LENGTH DOWNTO 1);
CONSTANT zeros: SIGNED(N DOWNTO 1):= (OTHERS => '0');
BEGIN
  xx := x;
  xx := signed(xx)+ unsigned(xx(n) & zeros);
  FOR i IN n DOWNTO 1 LOOP xx(i) := '0'; END LOOP;
  RETURN xx;
END;

BEGIN
   P1: PROCESS
   BEGIN
      zrr <= z_int(0)*z_int(1);
      zii <= z_int(1)*z_int(1);
      zir <= z_int(1)*signed(z_int(0));
      z_int(0) <= roundn(zrr-zii)+c(0);
      z_int(1) <= roundn(zir)+c(1);
       count <= NOT count;
       IF reset = '1' THEN
         z_int(0) <= (OTHERS=>'0');
         z_int(1) <= (OTHERS=> '0');
         count <= '0';
    END IF;
   END PROCESS P1;

   P2: PROCESS(count,ca,cb)
   BEGIN
      IF count='1' THEN c <= ca; ELSE c <= cb; END IF;
   END PROCESS P2;

   z <= z_int;

END ARCHITECTURE synth;
```

[10D]

Solution to Question 4

a)
A constant index selects a bit at zero cost. A variable index turns into a multiplexer to select the given bit.

As operand of a logical operator constant values simplify to '0', '1' or straight connection, or (XOR & XNOR only) an inverter. Variable values must invoke the relevant operator.

Constant expressions can be derived from:
CONSTANTs
FOR indexes
GENERICs

[4B]

b)
(i) 2 X 2-input multiplexers
(ii)4 X 4-input multiplexers

[2B]

c)

```
ARCHITECTURE synth OF switch IS
   TYPE grid IS ARRAY (0 TO n+1) OF std_logic_vector(n-1 DOWNTO 0);
   SIGNAL x: grid;
BEGIN
   G1: FOR a IN 0 TO n-1 GENERATE
      G2: FOR b IN 0 TO 2**(N-1)-1 GENERATE
          P: ENTITY permute GENERIC MAP(n=>n, m=>1)
                 PORT MAP( p=>sel(a DOWNTO a),
                   xin  =>(x(a)(x0(a,b)), x(a)(x1(a,b))),
                   xout =>(x(a+1)(x0(a,b)),x(a+1)(x1(a,b)) ) );
      END GENERATE G2;
   END GENERATE G1;
END synth;
```

[14D]