COL215_2018 Major

SAKSHAM DHULL

TOTAL POINTS

12.2 / 20

QUESTION 1

1 Q1 3 / 3

✓ + 3 pts **Correct**

+ 1 pts Correct timing

✓ + 1 pts **Inputs and outputs from DSP**

✓ + 1 pts **Breaking of 36*36 mul into 18 *18**

+ 0 pts Incorrect

QUESTION 2

2 Q2 3 / 3

✓ + 3 pts **Correct**

+ 0 pts Nothing is correct

QUESTION 3

3 Q3 3 / 3

✓ + 3 pts **Correct**

+ 0 pts Nothing is correct

+ 1 pts Major errors

+ 2 pts Minor errors

+ 2.5 pts Some errors

+ 1.5 pts Wrong

QUESTION 4

4 Q4 1 / 3

+ 3 pts Correct

+ 0 pts nothing is correct

✓ + 1 pts **Click here to replace this description.**

+ 2 pts Click here to replace this description.

+ 0.5 pts Click here to replace this description.

+ 2.5 pts 3 out of 4 signature are correct due to error

+ 1.5 pts 3 out of 4 signature are correct for error

+ 1.5 pts 1 out of 4 signature sre correct for error

No explanation given for initial signature

QUESTION 5

5 Q5 1 / 2

✓ + 0.5 pts **Part (a) is correct**

+ 1 pts Part (b) is correct

+ 0.5 pts Part (b) Partially Correct

✓ + 0.5 pts **Part (c) is correct**

+ 0 pts incorrect

- 0.25 pts Incomplete Explanations

+ 0.5 pts Part (a) and (c) partially correct

+ 0.25 pts Part(a) partially correct

QUESTION 6

6 Q6 0 / 2

+ 1 pts Identify all combinational circuit fragments

✓ + 0 pts **Incorrect/ Unattempted**

+ 0.5 pts Find the sources and destination and trace path

+ 0.5 pts Check existence of cycle

+ 1 pts Steps mentioned but not explained

+ 1.5 pts Steps mentioned and explained but not completely correct

+ 0.5 pts Partially correct steps mentioned and not explained

+ 1 pts Only cycle detection part correct

QUESTION 7

7 Q7 1 / 2

+ 2 pts Correct

+ 0.5 pts Vague Idea

+ 0 pts Blank

✓ + 1 pts **Partially Correct**

+ 1.5 pts Minor Errors

QUESTION 8

8 Q8 0.2 / 2

+ 2 pts Correct

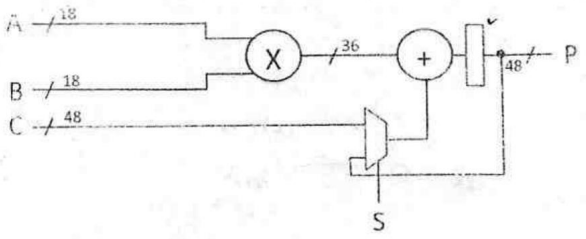
✓ + 0 pts nothing is correct

+ 0.2 Point adjustment

NAME: SAKSHAM DHULL	ENTRY NO.: 2017CS10370

24.11.2018 **COL215 Digital Logic and System Design : Major Test** Max Marks : 20

1. Design a circuit in VHDL to perform 36x36 integer multiplication using a single instance of the DSP module shown in the figure with inputs A, B, C and S and output P. S is the select input of the multiplexer. The register shown at the output of the adder is clocked in every cycle. No other adder or multiplier is to be used. Assume that the operands are non-negative. The circuit takes a fixed number of cycles to perform the operation and then starts again, repeating it forever. Include a component declaration for the DSP in your design. Its entity and architecture are not required. The following VHDL features are not to be used – Loops, Procedures, Wait statements, After clauses and Variables.



[3]

```
entity dsp_vhdl is
PORT ( a,b : in bit_vector (35 downto 0);
       c,ans : out bit_vector (71 downto 0) );

architecture dsp of vhdl is
signal one,two,three : std_logic (48 downto 0);
signal al : std_logic_vector(17 downto 0);
signal ar : std_logic_vector(17 downto 0);
signal bl : std_logic_vector(17 downto 0);
signal br : std_logic_vector(17 downto 0);
signal ain, bin : std_logic_vector(17 downto 0);
signal c,p : std_logic_vector (47 downto 0);
signal s : std_logic;
signal count : std_logic_vector(1 downto 0) := "000"
signal done : std_logic := '0';
begin
disp : work.entity (dsp)
       port map ( ain, bin, c, p, s);
al <= a(35 downto 18);
ar <= a(17 downto 0);
bl <= b(35 downto 18);
br <= b(17 downto 0);
process(clk)
  if (clk='1' & clk'event)
    if (count="00" & done="00") then
      ain <= ar, bin <= br, s<=0,
    if (count="00").
```

→ Continued on last pg ;

a → al, ar

b → bl, br

(ar, br), c = 0, s = 0

→ P

ar. br

36×36

$(A_L/A_R)^{18} \times (B_L/B_R)^{18}$

$(A_L B_L)\, 2^{36} + (A_L B_R + A_R B_L) \times 2^{18} + A_R B_R$

(10000) (000001) (00010) 000001

10000000

100000

NAME: SAKSHAM DHULL	ENTRY NO.: 2017CS10370

24.11.2018 **COL215 Digital Logic and System Design : Major Test** Max Marks : 20

2. Design a circuit in VHDL to display a character on an 8x6 LED matrix display. You may assume the anodes of the LEDs to be connected to the rows and cathodes to be connected to the columns (or vice versa) in the matrix. The characters are coded in 8 bits. Assume that the dot patterns for the characters are stored in a memory row-wise and reading from memory involves one cycle delay (that is, the data output in a cycle corresponds to the address input in the previous cycle). Include a component declaration for the memory in your design. Its entity and architecture are not required. The following VHDL features are not to be used – Loops, Procedures, Wait statements, After clauses and Variables. P.T.O → [3]

```
Entity led is
PORT ( num : in bit_vector (7 downto 0);
       anode : out bit_vector (7 downto 0);
       cathode : out bit_vector (5 downto 0);
       en : in bit );

Architecture light of led is
signal an : std_logic_vector (7 downto 0) := "00000001";
signal lastan : std_logic_vector (7 downto 0) := "10000000";
signal checker : std_logic_vector (8 downto 0);
Begin

mem : work.entity (memory)
      Portmap ( cathode, lastan, num);

anode <= an;
checker <= an & en;

Case checker is
  when "000000011" => lastan <= an, an <= "10000000";
  when "000000101" => lastan <= an, an <= "00000001";
  when "000001001" => lastan <= an, an <= "00000010";
  when "000010001" => lastan <= an, an <= "00000100";
  when "000100001" => lastan <= an, an <= "00001000";
  when "001000001" => lastan <= an, an <= "00010000";
  when "010000001" => lastan <= an, an <= "00100000";
  when "100000001" => lastan <= an, an <= "01000000";
  when others => an <= "00000010";
  when others => lastan <= "10000000", an <= "00000001";
                 en <= '0';
end case;

end architecture light;
```

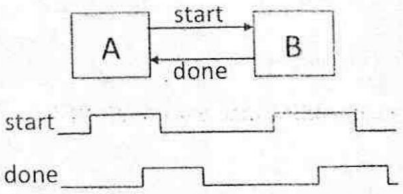
Q-2.

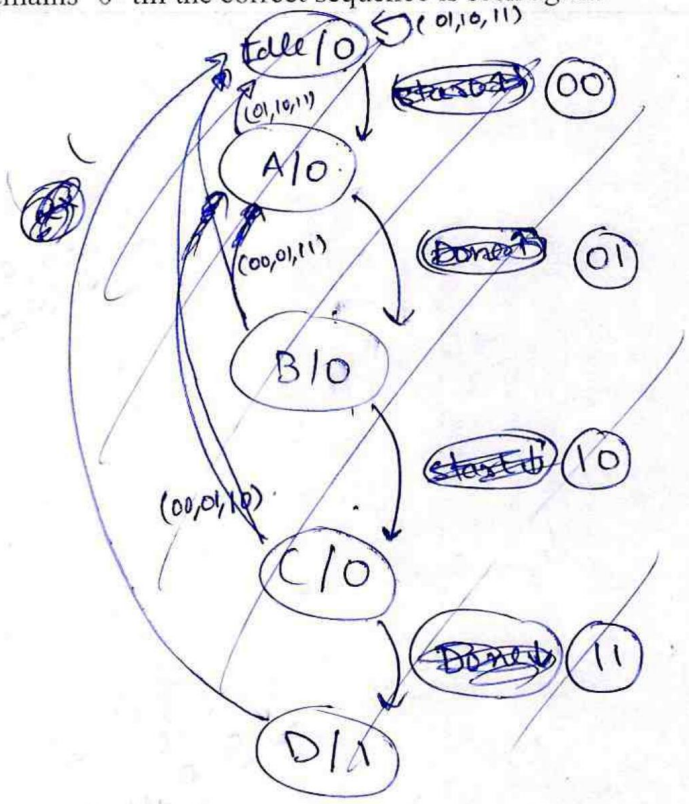
```
entity led is
PORT( clk: in bit
      num: in bit_vector (7 downto 0);
      anode: out bit_vector (7 downto 0);
      cathode: out bit_vector (5 downto 0) );

architecture light of led is
signal an : std_logic_vector (7 downto 0) := "00000001";
signal lastan : std_logic_vector (7 downto 0) := "00000010";
Begin
  Process (clk).
    if ( clk='1' & clk'event ) then.
        lastan <= an;
        an <= an & "0";
      if(an = "10000000") then
          an <= "0000 0001"
      else
          an <= an & "0";
      end if;
      anode <= an;
      lastan <= an & "0";
      if (an = "10000000") then
          lastan <= "00000001";
      else
          lastan <= an & "0";
      end if;
  end if;
  end process;
mem: work.entity (memory)
     PORT map ( cathode, lastan, num)
end.
End architecture;
```

NAME: Saksham Dhull	ENTRY NO.: 2017CS10370

24.11.2018 **COL215 Digital Logic and System Design : Major Test** Max Marks : 20

3. Two modules communicate with asynchronous handshaking signals 'start' and 'done' as shown. Design an asynchronous FSM (only flow table is required) that looks at these two signals and checks whether a correct sequence of transitions in the handshaking signals has been followed (start ↑, done ↑, start ↓, done ↓). Its output becomes '1' after seeing the correct sequence of four transitions. It becomes '0' on the next transition and remains '0' till the correct sequence is seen again. [3]



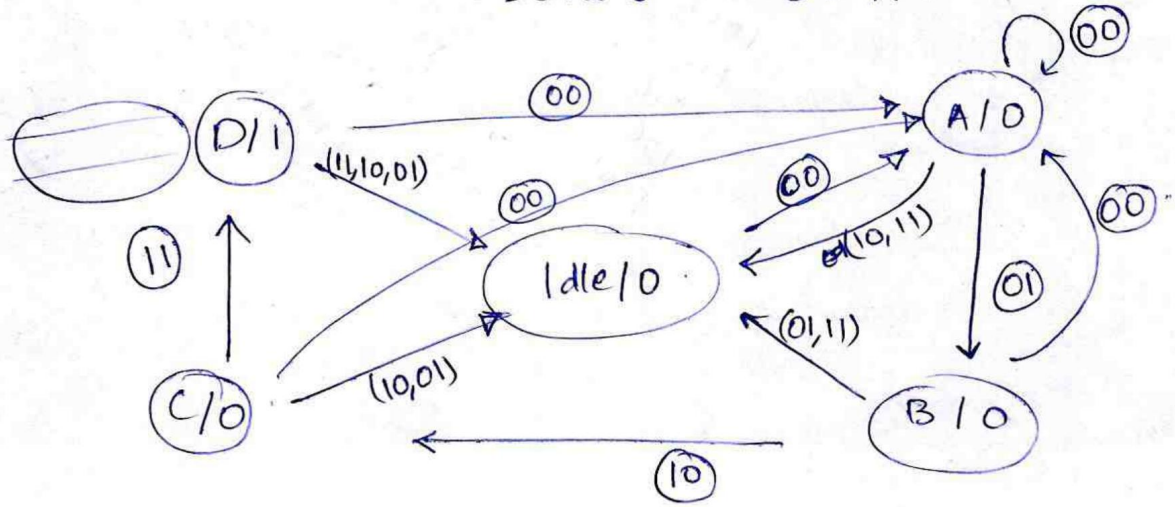


Let's denote signals start ↑ → 00

start ↓ → 10

Done ↑ → 01

Done ↓ → ~~00~~ 11.



1110
0110
1010
1100

1110
0110 → 0010
1000

0001
0001
1001
1101

1000
1000
1000

0011
0001
1001
0101
0011
1001
1011

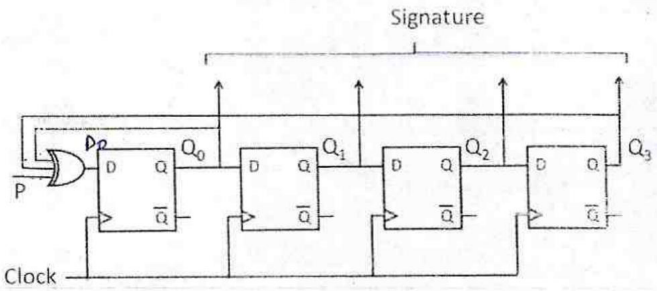
1110
1111
0111

0100

NAME: Salesham Dhull	ENTRY NO.: 2017CS10370.

24.11.2018 **COL215 Digital Logic and System Design : Major Test** Max Marks : 20

4. Consider the signature analyzer shown in the figure. What is the signature produced by the input sequence 1001100101, assuming that initially all the flip-flops contain '0'? If there were a single bit error in the sequence, what signature would be produced (consider any 4 of the 10 possible locations of the error in the sequence)?



[3]

~~If all~~

With no error → signature produced by input sequence is 1110.

~~Assuming~~

Errors :

$Q_0/0$: ~~seq~~ Signature : 0000.

$Q_0/1$: signature : 1111.

$Q_1/0$: Signature : 1000

$Q_1/1$: Signature : 1111

D_0 ~~Q_2~~ $/0$: signature : 0000

D_0 ~~Q_2~~ $/1$: signature : 1111

$Q_3/0$: Signature : 1000

$Q_3/1$: Signature : 1101.

Let the input to first flip flop be D_0.

NAME: Sakshama Dhull	ENTRY NO. 2017CS10370

24.11.2018 **COL215 Digital Logic and System Design : Major Test** Max Marks : 20

5. For the FSM shown in the figure, find an example of each of the following cases.
   - A pair of states that are compatible and can be merged unconditionally
   - A pair of states that are compatible but can be merged subject to merger of another pair
   - A group of three states illustrating non-transitivity of compatibility

Present state	Next state				Output z
	DN = 00	DN = 01	DN = 10	DN = 11	
S1	S1	S3	S2	*	0
S2	S2	S4	S5	*	0
S3	S3	S6	S7	*	0
S4	S1	*	S7	*	1
S5	S3	*	S2	*	1
S6	S6	S8	S9	*	0
S7	S1	*	S2	*	1
S8	S1	*	*	*	1
S9	S3	*	*	*	1

[2]

a) A pair of ~~Compatabite~~ Compatible states → $\{S_4, S_8\}$

b) $\{S_4, S_7\}$ can only be merged iff $\{S_7, S_2\}$ can be merged.

c) A group of 3 states illustrating non-transitivity
- $\{S_4, S_8, S_7\}$.

NAME: SAKSHAM DHULL	ENTRY NO.: 2017CS10370

24.11.2018 **COL215 Digital Logic and System Design : Major Test** Max Marks : 20

6. Given a design description in VHDL, how would you check whether there is a cycle (or cycles) formed in combinational circuit? Write your answer in algorithmic form, as if you were to give instructions to someone who understands VHDL syntax only, but not the meaning of various constructs. one entity & arch [2]

```
entity cycle
    PORT ( en : in bit; datain : in bit
           dataout : out bit )

architecture isthere of cycle
  signal a : std_logic;
begin
    with en select
        a <= data when '1';
             a <= '0' otherwise;

    data <= a;
    dataout <= a;
end architecture isthere;
```

when a signal is driving & being driven by both by a same signal. then there's a cyclic dependence.

NAME: SAKSHAM DHULL	ENTRY NO.: 2017CS10370

24.11.2018 **COL215 Digital Logic and System Design : Major Test** Max Marks : 20

7. Given a design description in VHDL, how would you check whether a latch (or latches) would be inferred by the synthesizer? Write your answer in algorithmic form, as if you were to give instructions to someone who understands VHDL syntax only, but not the meaning of various constructs.

[2]

~~arch~~
entity devices is
PORT(d: in bit ; clk: in bit;
~~out~~ ans : out bit);

architecture ~~#~~ latch of devices is
Begin
PROCESS(clk ~~d~~)
begin
if (clk = '1') then
~~if d="1" then ans <= d;~~ { signal assignments eg. ans <= d; for d latch }
end if;
~~else~~
end process
end architecture.

If any ~~of~~ of the ~~state is pre~~ signal assignment of any state is dependent on the clock level then a latch is being used.

NAME: Saketham Dhull	ENTRY NO. : 2017CS10370

24.11.2018 **COL215 Digital Logic and System Design : Major Test** Max Marks : 20

8. What is the significance of different colours in which waveforms are shown during simulation using Xilinx software? What do you need to do eliminate the undesired colours? [2]

Green color is used for outputs that depend upon inputs and are correctly connected.

Reddish Brown color is shown for undefined outputs in the circuits.

Red is shown for outputs ~~that~~ that are stuck at some value, ~~either due to wrong logic~~

One must eliminate all the reddish-brown & red as they might cause some errors.

Elimination of undesired colors needs to be done to get a good circuit that obeys logics.

Q-1 contd.

```
process (clk)
  if (clk = 1 & clk'event) then
    if (count = "000") then
      if (done = "0") then
        ain ⇐ ar, bin ⇐ br, s ⇐ 0, done ⇐ '1';
      else
        one ⇐ p; count ⇐ "001", done ⇐ '0';
      end if
    else if (count = "001") then
      if (done = "0") then
        ain ⇐ al, bin ⇐ br, s ⇐ 0, done ⇐ '1';
    elseif (done = "1") then
        ain ⇐ ar, bin ⇐ bl; s ⇐ 1, count ⇐ "010", done ⇐ '0',
    end if.
    else if (count = "010") then
        two ⇐ p; count ⇐ "011", done ⇐ '0';
    elsif (count = "011") then
        if (done = "0") then
            ain ⇐ ar, bin ⇐ br, s ⇐ 0, done ⇐ "1";
        else
            three ⇐ p; count ⇐ "100";
        end if
    else if (count = "100") then
```

~~ans ⇐ one~~
~~(count 17 down to 0) ⇐ one;~~
~~end if~~
~~ans ⇐ one + two~~
~~two ⇐ two~~
~~ans ⇐ one + two & "00000000 0000000000" +~~
~~if done = "0" then~~ there

ain ⇐
ans ⇐ one + two & " 0000000000000 " ~~three~~
(18)
+ three & "0 - - - 0"
(36)

end if;
end process this architecture;