**Assignment Solution**

# Q1

1. Draw a block diagram of a digital circuit with the following specifications (20 Marks).

Interface:

Din – 16-bit data input

Dout – 16-bit data output

Addr – 3-bit address of the location (register) where input data is stored and output data is read

from

Read – control signal indicating read

Write – control signal indicating write

Clk – clock

Functionality:

• If Write = 1, then at the next rising edge of the clock, data from the input Din is stored

in the internal location given by the address Addr.

• If Read = 1, data from the location given by the address Addr is transferred to the output

Dout, and the contents of the internal memory (registers) does not change. If Read=0,

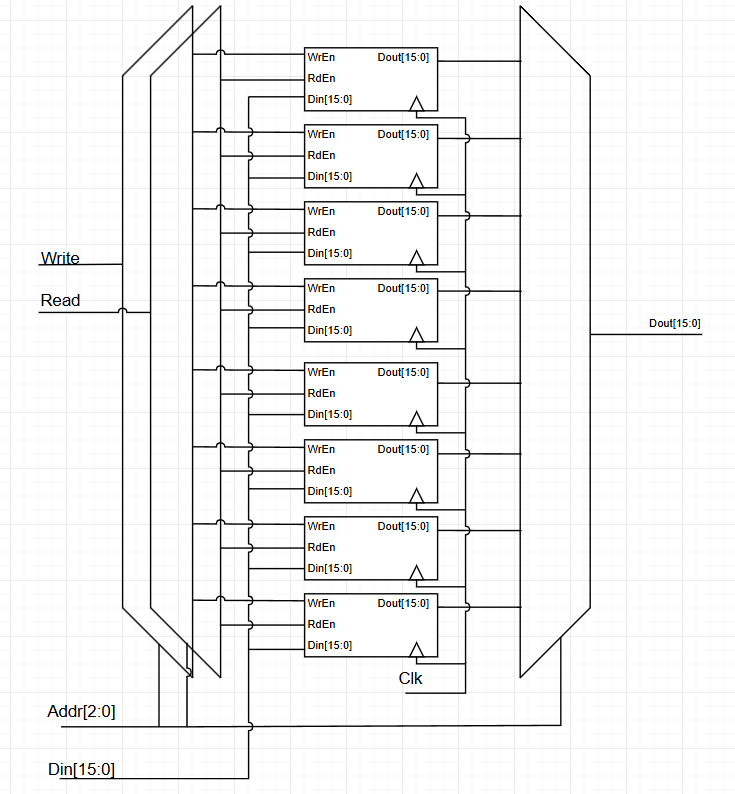
the output Dout should be set to the high impedance state. Assume that the internal

memory is implemented using registers. Use only medium scale components, such as

registers, multiplexers, encoders, decoders, buffers, etc.

You are not allowed to use RAM in your circuit.

# Answer Q1:



# Q2

The VHDL code given below is describing a block diagram of a circuit. Draw the

architecture RTL that is described by this code. (15 Marks)

LIBRARY ieee ;

USE ieee.std\_logic\_1164.all ;

ENTITY variable\_rotator is

PORT(

A : IN STD\_LOGIC\_VECTOR(15 downto 0);

B : IN STD\_LOGIC\_VECTOR(3 downto 0);

C : OUT STD\_LOGIC\_VECTOR(15 downto 0)

);

END variable\_rotator;

ARCHITECTURE structural OF variable\_rotator IS

TYPE array16 IS ARRAY (0 to 4) OF STD\_LOGIC\_VECTOR(15 DOWNTO 0);

SIGNAL Al : array16;

SIGNAL Ar : array16;

BEGIN

Al(0) <= A;

G:

FOR i IN 0 TO 3 GENERATE

Ar(i) <= Al(i)(15-2\*\*i downto 0) & Al(i)(15 downto 15-2\*\*i+1);

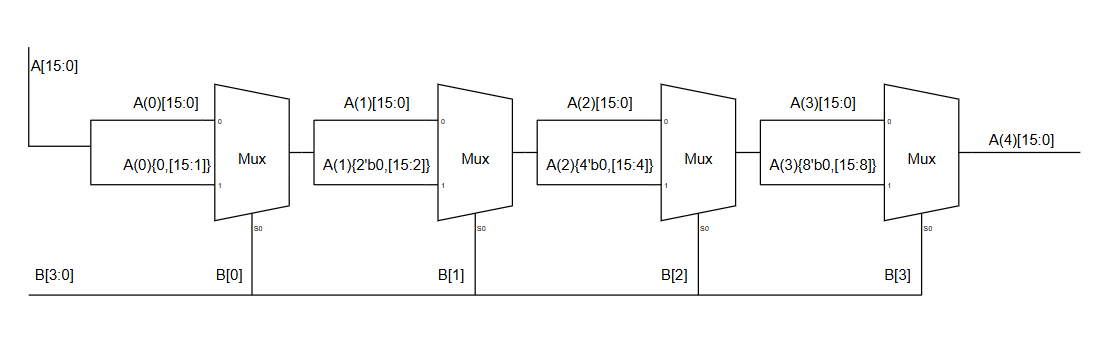
Al(i+1) <= Al(i) when B(i)=‘0’ else Ar(i);

END GENERATE;

C <= Al(4);

END dataflow;

# Answer Q2:



# Q3

Consider a cell-based 8-bit × 8-bit Cary-save array multiplier. (15 Marks)

a. Draw the architecture schematic of the RTL.

b. Estimate how many AND gates and Full adders it requires (Assume half adders

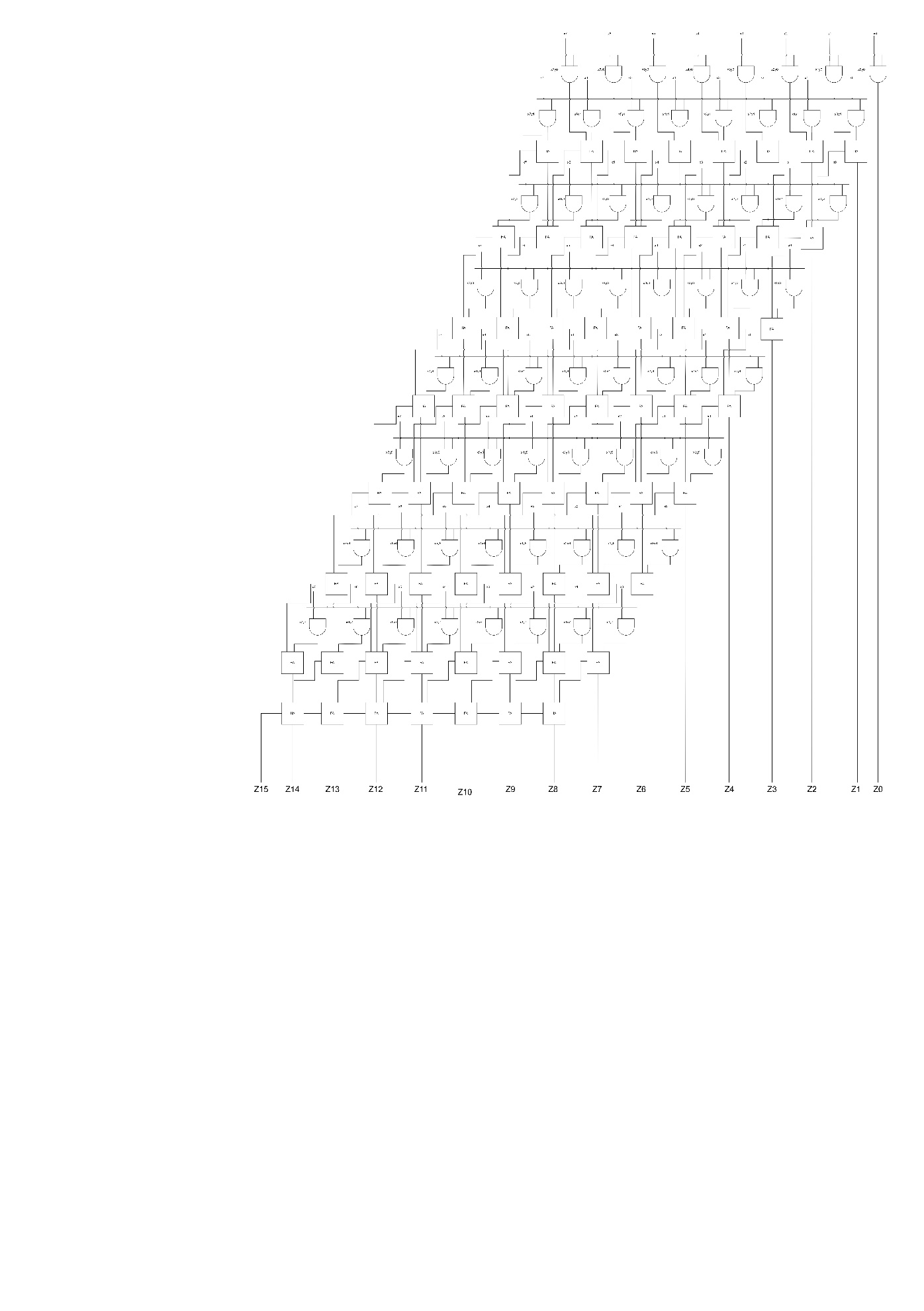
are implemented by full adders).

c. What is the critical-path delay of this multiplier, assuming an AND gate delay is

t\_a, and adder delay is t\_add

# Answer Q3:

## a.



The picture is attached in a pdf file.

## B

**1) Number of AND Gates:**

* For an 8x8 multiplier, each bit of the multiplicand (X) is ANDed with each bit of the multiplier (Y) to generate the partial products.
* Total partial products = .

**Total AND Gates = 64**

2) **Number of Full Adders**:

The carry-save adder (CSA) structure reduces the 8 rows of partial products into a final sum and carry. Let’s calculate the required number of full adders:

**Stage 1**:

* Add **Row 1**, **Row 2**, and **Row 3** using full adders:
  + Each column (from least significant to most significant) has 3 inputs.
  + For 8 bits, this requires full adders.

**Subsequent Stages**:

* For an 8×8 multiplier, there are 8 rows of partial products.
* CSA reduces the number of rows from 8 to 2 in stages.
* Each stage processes the sum and carry of the previous stage along with the next partial product row.
* Total Full Adders in CSA =

|  |  |  |
| --- | --- | --- |
| Stage | Rows Combined | Numbers of Adders |
| 1 | Row 1,2,3 | 7 |
| 2 | Sum1, Carry1, Row4 | 7 |
| 3 | Sum2, Carry2, Row5 | 7 |
| 4 | Sum3, Carry3, Row6 | 7 |
| 5 | Sum4, Carry4, Row7 | 7 |
| 6 | Sum5, Carry5, Row8 | 7 |

**Final Summation**:

* The two rows (sum and carry) from the last CSA stage are added using a 16-bit ripple-carry adder.
* A 16-bit ripple-carry adder requires full adders.

**Total Full Adders** =

## C

**3) Critical Path Delay**

**Delays:**

1. **AND Gates**:
   * All 64 partial products are generated in parallel.
   * Delay for partial products = ​.
2. **Carry-Save Adder Stages**:
   * Each CSA stage contributes a delay of .
   * Total CSA stages = 6.
   * Delay for CSA stages = 6⋅.
3. **Final Ripple-Carry Adder**:
   * The ripple-carry adder combines two rows (sum and carry) into the final product.
   * For a 16-bit ripple-carry adder, the delay = ​.

**Total Critical Path Delay**:

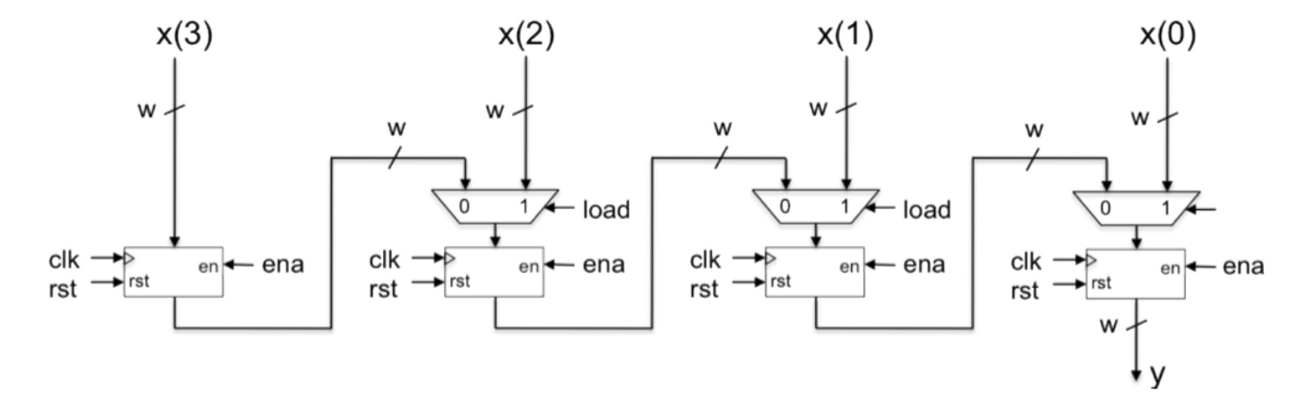
If then

# Q4

This circuit is called parallel-in-serial-out. Fill in the blanks (dashed lines) in the given

VHDL codes which is for entity declaration and architecture, with the size of the output

bus, w, treated as a generic with the default value equal to 8. (20 Marks)



|  |
| --- |
| library ieee;  use ieee.std\_logic\_1164.all;    entity piso is  generic (  w : integer := …………………);  port (  clk : in std\_logic;  ena : in std\_logic;  load : in std\_logic;  rst : in std\_logic;  x : in std\_logic\_vector(……………………. downto 0);  y : out std\_logic\_vector(w-1 downto 0));  end piso;    architecture behavioral of piso is  type bus\_array is array (3 downto 0) of std\_logic\_vector(w-1  downto 0);  signal reg, mux : bus\_array;  begin  mux(3) <= x(4\*w-1 downto ……………………);  mux\_gen : for i in 2 downto 0 generate  mux(i) <= x(………………………downto i\*w) when load = '1' else  reg(i+1);  end generate;    regx\_gen : for i in 3 downto 0 generate  regx : process ( clk )  begin  if ……………………………………. then  if rst = '1' then  reg(i) <= (……………………………);  elsif ena = '1' then  reg(i) <= mux(i);  end if;  end if;  end process;  end generate;    y <= reg(0);  end behavioral; |

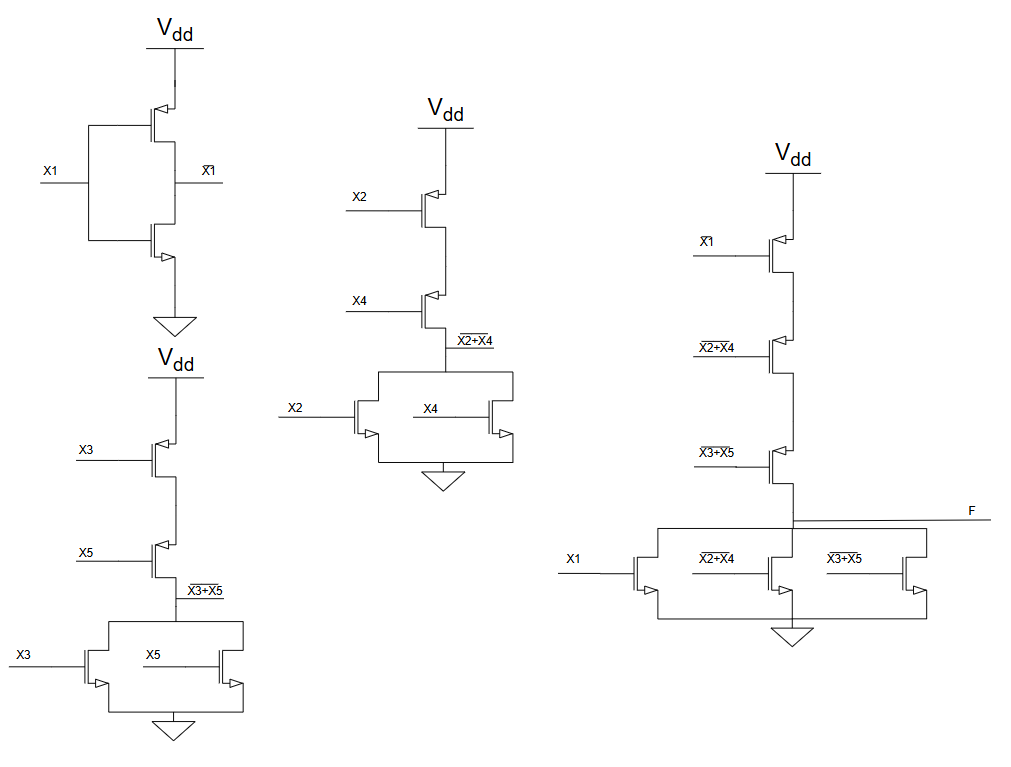
# Answer Q4

|  |
| --- |
| library ieee;  use ieee.std\_logic\_1164.all;  entity piso is  generic (  w : integer := 8); -- Width of each segment in the input vector  port (  clk : in std\_logic;  ena : in std\_logic;  load : in std\_logic;  rst : in std\_logic;  x : in std\_logic\_vector(4\*w-1 downto 0); -- Input vector of size 4\*w  y : out std\_logic\_vector(w-1 downto 0)); -- Output vector of size w  end piso;  architecture behavioral of piso is  type bus\_array is array (3 downto 0) of std\_logic\_vector(w-1 downto 0);  signal reg, mux : bus\_array; -- Internal signals to hold registers and multiplexer outputs  begin  mux(3) <= x(4\*w-1 downto 3\*w); -- Assign the highest segment to mux(3)  mux\_gen : for i in 2 downto 0 generate  mux(i) <= x((i+1)\*w-1 downto i\*w) when load = '1' else reg(i+1); -- Generate mux logic  end generate;  regx\_gen : for i in 3 downto 0 generate  regx : process ( clk )  begin  if rising\_edge(clk) then -- Check for clock's rising edge  if rst = '1' then  reg(i) <= (others => '0'); -- Reset all bits to 0  elsif ena = '1' then  reg(i) <= mux(i); -- Load mux output into the register if enabled  end if;  end if;  end process;  end generate;  y <= reg(0); -- Output the serialized data from reg(0)  end behavioral; |

# Q5

Implement the following function using CMOS: f = x + (x + x ) + (x + x ). (10 Marks)

# Answer Q5



# Q6

For the VHDL code given bellow, perform the following TWO tasks (20 Marks):

a. Supplement the code with the proper contents of the sensitivity list (10 Marks).

b. Draw a block diagram of the corresponding digital circuit (10 Marks).

Process

(…………………………………………………………………………………………………………

………………………………………………………………………………………………………….)

# Answer Q6

## a.

|  |
| --- |
| process(video\_on, wall\_on, bar\_on, sq\_ball\_on, wall\_rgb, bar\_rgb, ball\_rgb)  begin  if video\_on = '0' then  graph\_rgb <= "000"; -- blank  else  if wall\_on = '1' then  graph\_rgb <= wall\_rgb;  elsif bar\_on = '1' then  graph\_rgb <= bar\_rgb;  elsif sq\_ball\_on = '1' then  graph\_rgb <= ball\_rgb;  else  graph\_rgb <= "110"; -- yellow background  end if;  end if;  end process; |

## b.

