Lab 1: Simple Circuit Design

Professor Gertner

Faheem Kamal

2019

**Table of Contents**

**Objective 2**

**Functionality and Specifications 3 – 6**

**Simulation 6 – 12**

**Conclusion 12**

**Appendix 12 - 36**

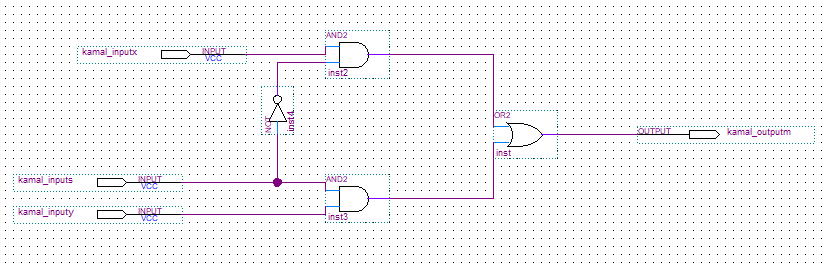
**Objective:**

The goal of this lab is to understand and design basic essential tools in digital electronics. Using Quartus and ModelSim, our task is to create and generate several types of circuits. In addition, we are also response to test and analyze each circuit.

* Part 2: We create a 2-to-1 Multiplexer using the Board Diagram Schematic file and we create a waveform file. In addition, we create a vhdl code and testbench file.
* Part 3: We create a 2-to-1 Multiplexer using LPM and we create a waveform file.
* Part 4: We create each circuit: 2-to-1 Mux, 1-bit Half Adder, 1-bit Full Adder using Half Adder, 3-to-8 Decoder and 8-to-3 Encoder.
  + We create each Board Diagram Schematic File.
  + We then add our own vhdl code and testbench file.
  + We also create a waveform file for simulation for each file.
* Part 5: We create each circuit using LPM except for Decoder and 2-to-1 Multiplexer(covered in Part 3)

**Functionality and Specifications:**

1. **2-to-1 Multiplexer**



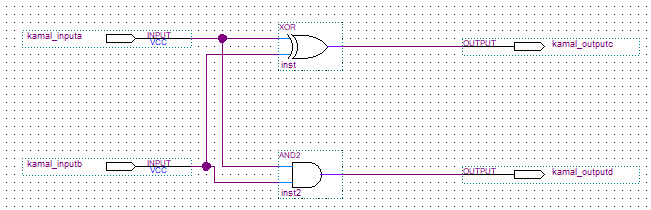
Inputs: kamal\_inputs, kamal\_inputx and kamal\_inputy

Outputs: kamal\_outputm

Pin Assignments (DE2-115): -

* kamal\_inputs: PIN\_AC27
* kamal\_inputx: PIN\_AB28
* kamal\_inputy: PIN\_AC28
* kamal\_outputm, PIN\_G19

1. **Half-Adder**



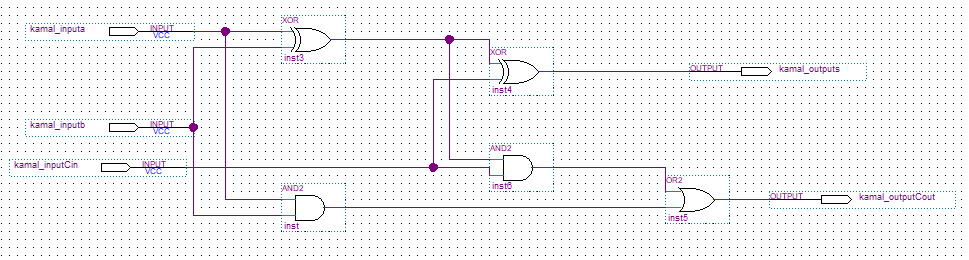
Inputs: kamal\_inputa and kamal\_inputb

Outputs: kamal\_outputc and kamal\_outputd

Pin Assignments (DE2-115): -

* kamal\_inputa: PIN\_AB28
* kamal\_inputb: PIN\_AC28
* kamal\_outputc: PIN\_G19
* kamal\_outputd: PIN\_F19

1. **Full-Adder**



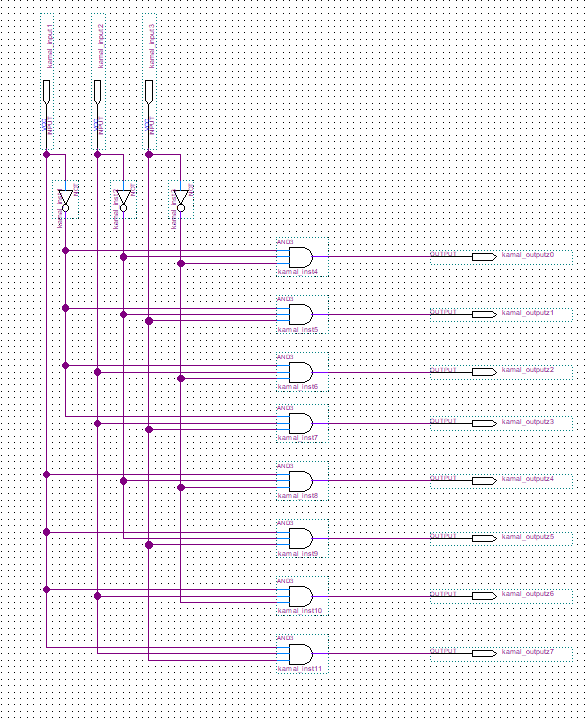
Inputs: kamal\_inputa, kamal\_inputb and kamal\_inputCin

Outputs: kamal\_outputs and kamal\_outputCout

Pin Assignments (DE2-115): -

* kamal\_inputa: PIN\_AB28
* kamal\_inputb: PIN\_AC28
* kamal\_inputCin: PIN\_AC27
* kamal\_outputs: PIN\_G19
* kamal\_outputCout: PIN\_F19

1. **3-to-8 Decoder**



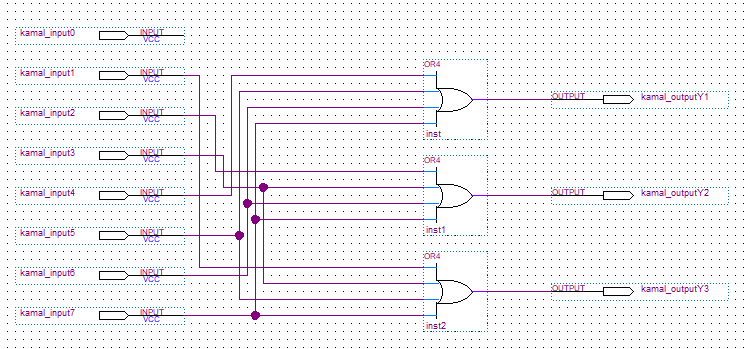
Inputs: kamal\_input1, kamal\_input2 and kamal\_input3

Outputs: kamal\_outputz0, kamal\_outputz1, kamal\_outputz2, kamal\_outputz3, kamal\_outputz4, kamal\_outputz5, kamal\_outputz6 and kamal\_outputz7

Pin Assignments (DE2-115): -

* kamal\_input1: PIN\_AB28
* kamal\_input2: PIN\_AC28
* kamal\_input3: PIN\_AC27
* kamal\_outputz0: PIN\_G19
* kamal\_outputz1: PIN\_F19
* kamal\_outputz2: PIN\_E19
* kamal\_outputz3: PIN\_F21
* kamal\_outputz4: PIN\_F18
* kamal\_outputz5: PIN\_E18
* kamal\_outputz6: PIN\_J19
* kamal\_outputz7: PIN\_H19

1. **8-to-3 Encoder**



Input: kamal\_input1, kamal\_input2, kamal\_input3, kamal\_input4, kamal\_input5, kamal\_input6 and kamal\_input7

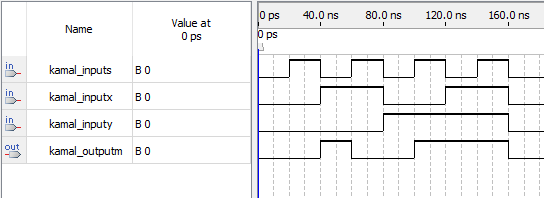
Output: kamal\_outputY1, kamal\_outputY2 and kamal\_outputY3

Pin Assignments (DE2-115): -

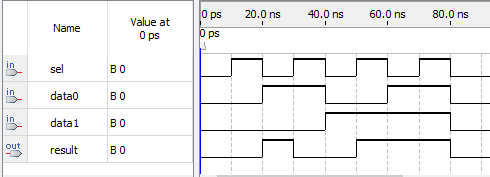
* kamal\_input1: PIN\_AB28
* kamal\_input2: PIN\_AC28
* kamal\_input3: PIN\_AC27
* kamal\_input4: PIN\_AD27
* kamal\_input5: PIN\_AB27
* kamal\_input6: PIN\_AC26
* kamal\_input7: PIN\_AD26
* kamal\_outputY1: PIN\_G19
* kamal\_outputY2: PIN\_F19
* kamal\_outputY3: PIN\_E19

**Simulation:**

1. **2-to-1 Multiplexer**

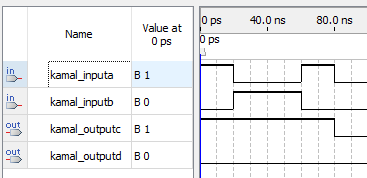


**(LPM Simulation Below)**

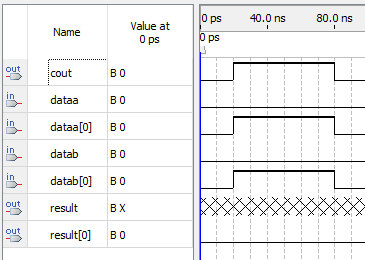


A 2-to-1 Multiplexer is a combinational logic circuit designed to switch one of the inputs to a single output line using a control signal. It’s also a digital switch where the select line is the throw on the switch, it chooses which of the many inputs get to be the output.

1. **Half-Adder**

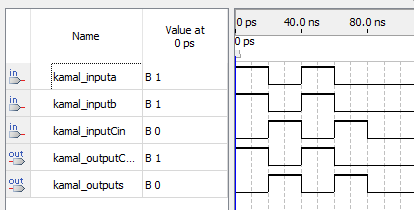


**(LPM Simulation Below)**

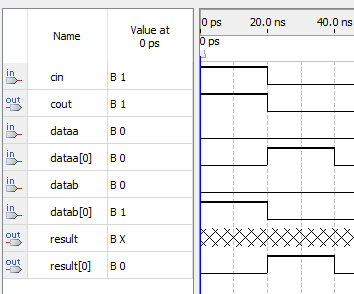


A Half-Adder is a type of adder that performs the addition of numbers. It can add two single binary digits and provide the output. It has two inputs, in this case, kamal\_inputa and kamal\_inputb. It also has two outputs, kamal\_outputc and kamal\_outputd. We can observe based on the simulation diagram that kamal\_outputc gets its signal values based on the addition of kamal\_inputa and kamal\_inputb.

1. **Full-Adder**

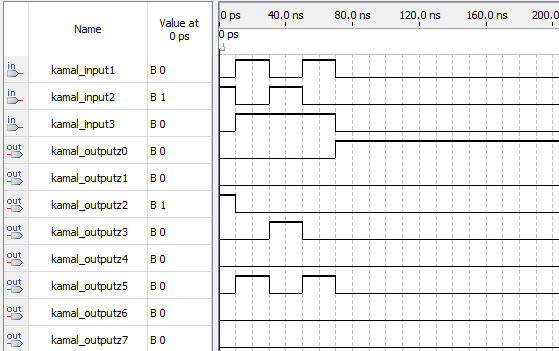


**(LPM Simulation Below)**

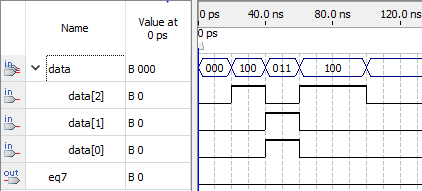


Full Adder is the adder that adds three inputs and produces two outputs. For this case, our three inputs are kamal\_inputa, kamal\_inputb and kamal\_inputCin. And our two outputs are kamal\_outputCout and kamal\_outputs. The output carry is kamal\_outputCout and the normal output sum is kamal\_outputs. We can analyze each output signal based on the logical expressions. The logical expression for the output sum is **kamal\_inputCin XOR (kamal\_inputa XOR kamal\_inputb)**. The logical expression for the output carry is **(kamal\_inputb AND kamal\_inputb) OR (kamal\_inputb AND kamal\_inputCin) OR (kamal\_inputa AND kamal\_inputCin).**

1. **3-to-8 Decoder**



**(LPM Simulation Below)**



A Decoder translates or decodes information from one format into another. In this case, a Binary Decoder transforms n binary input signals into an equivalent outputs. A 3-to-8 Decoder has 8 logic outputs for 3 logic inputs. The inputs are kamal\_input1, kamal\_input2 and kamal\_input3. The outputs are kamal\_outputz0, kamal\_outputz1, kamal\_outputz2, kamal\_outputz3, kamal\_outputz4, kamal\_outputz5, kamal\_outputz6 and kamal\_outputz7. The logical expression for the outputs are:

1. **kamal\_outputz0 = (NOT kamal\_input1) AND (NOT kamal\_input2) AND (NOT kamal\_input3)**

2. **kamal\_outputz1 = (NOT kamal\_input1) AND (NOT kamal\_input2) AND (kamal\_input3)**

3. **kamal\_outputz2 = (NOT kamal\_input1) AND (kamal\_input2) AND (NOT kamal\_input3)**

4. **kamal\_outputz3 = (NOT kamal\_input1) AND (kamal\_input2) AND (kamal\_input3)**

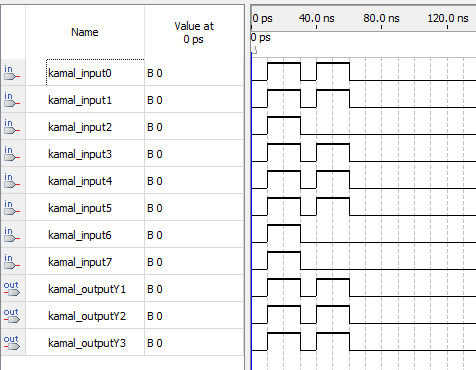
5. **kamal\_outputz4 = (kamal\_input1) AND (not kamal\_input2) AND (not kamal\_input3)**

6. **kamal\_outputz5 = (kamal\_input1) AND (not kamal\_input2) AND (kamal\_input1)**

7. **kamal\_outputz6 = (kamal\_input1) AND (kamal\_input2) AND (not kamal\_input3)**

8. **kamal\_outputz7 = (kamal\_input1) AND (kamal\_input2) AND (kamal\_input3)**

1. **8-to-3 Encoder**



An encoder takes all of the data inputs one at a time and converts them into a single equivalent binary code as an output. We have 8 inputs denoted by kamal\_input0 through kamal\_input7 in the circuit. And we also have 3 outputs denoted by kamal\_outputY1 through kamal\_outputY3. The input kamal\_input0 is generally ignored since it is not connected to any of the OR gates in the circuit.

**Conclusion:**

I learned various circuit designs such as the 2-to-1 MUX, Half-Adder, Full-Adder, 3-to-8 Decoder and 8-to-3 Encoder using Quartus II. I learned to also make block diagrams, code in VHDL and create Waveform files. In addition, I also learned the properties and functionalities of each circuits and to verify each one through simulation. Furthermore, I also learned to utilize Quartus II and Modelsim in future labs.

**Appendix:**

1. **2-to-1 Multiplexer VHDL code**

library ieee;

use ieee.std\_logic\_1164.all;

entity kamal\_mux2to1code is

port(

kamal\_inputx: in std\_logic;

kamal\_inputy: in std\_logic;

kamal\_inputs: in std\_logic;

kamal\_outputm: out std\_logic

);

end kamal\_mux2to1code;

architecture arch of kamal\_mux2to1code is

begin

kamal\_outputm <= (kamal\_inputs and kamal\_inputy) or ((not kamal\_inputs) and kamal\_inputx);

end arch;

1. **2-to-1 Multiplexer Testbench VHDL code**

library ieee;

use ieee.std\_logic\_1164.all;

entity kamal\_mux2to1\_tb is

end kamal\_mux2to1\_tb;

architecture tb of kamal\_mux2to1\_tb is

signal kamal2\_inputx: std\_logic := '0';

signal kamal2\_inputy: std\_logic := '0';

signal kamal2\_inputs: std\_logic := '0';

signal kamal2\_outputm: std\_logic;

begin

UUT: entity work.kamal\_mux2to1code

port map(

kamal\_inputs => kamal2\_inputs,

kamal\_inputx => kamal2\_inputx,

kamal\_inputy => kamal2\_inputy,

kamal\_outputm => kamal2\_outputm

);

process is

begin

kamal2\_inputs <= '0';

kamal2\_inputx <= '0';

kamal2\_inputy <= '0';

wait for 10 ns;

kamal2\_inputs <= '0';

kamal2\_inputx <= '0';

kamal2\_inputy <= '1';

wait for 10 ns;

kamal2\_inputs <= '0';

kamal2\_inputx <= '1';

kamal2\_inputy <= '1';

wait for 10 ns;

kamal2\_inputs <= '1';

kamal2\_inputx <= '1';

kamal2\_inputy <= '1';

wait for 10 ns;

end process;

end tb;

1. **PIN ASSIGNMENT for 2-to-1 Multiplexer**

To, Location

kamal\_inputx,PIN\_AB28

kamal\_inputy,PIN\_AC28

kamal\_inputs,PIN\_AC27

kamal\_outputm,PIN\_G19

1. **2-to-1 MUX LPM VHDL code**

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

LIBRARY lpm;

USE lpm.lpm\_components.all;

ENTITY kamal\_muxLPM IS

PORT

(

data0 : IN STD\_LOGIC ;

data1 : IN STD\_LOGIC ;

sel : IN STD\_LOGIC ;

result : OUT STD\_LOGIC

);

END kamal\_muxLPM;

ARCHITECTURE SYN OF kamal\_muxlpm IS

-- type STD\_LOGIC\_2D is array (NATURAL RANGE <>, NATURAL RANGE <>) of STD\_LOGIC;

SIGNAL sub\_wire0 : STD\_LOGIC\_VECTOR (0 DOWNTO 0);

SIGNAL sub\_wire1 : STD\_LOGIC ;

SIGNAL sub\_wire2 : STD\_LOGIC ;

SIGNAL sub\_wire3 : STD\_LOGIC\_2D (1 DOWNTO 0, 0 DOWNTO 0);

SIGNAL sub\_wire4 : STD\_LOGIC ;

SIGNAL sub\_wire5 : STD\_LOGIC ;

SIGNAL sub\_wire6 : STD\_LOGIC\_VECTOR (0 DOWNTO 0);

BEGIN

sub\_wire4 <= data0;

sub\_wire1 <= sub\_wire0(0);

result <= sub\_wire1;

sub\_wire2 <= data1;

sub\_wire3(1, 0) <= sub\_wire2;

sub\_wire3(0, 0) <= sub\_wire4;

sub\_wire5 <= sel;

sub\_wire6(0) <= sub\_wire5;

LPM\_MUX\_component : LPM\_MUX

GENERIC MAP (

lpm\_size => 2,

lpm\_type => "LPM\_MUX",

lpm\_width => 1,

lpm\_widths => 1

)

PORT MAP (

data => sub\_wire3,

sel => sub\_wire6,

result => sub\_wire0

);

END SYN;

1. **PIN ASSIGNMENT for 2-to-1 Multiplexer LPM**

To, Location

data0,PIN\_AB28

data1,PIN\_AC28

sel,PIN\_AC27

result,PIN\_G19

1. **Half-Adder VHDL code**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity kamal\_lab1halfaddercode is

port(

kamal\_inputa : in std\_logic;

kamal\_inputb : in std\_logic;

--

kamal\_outputc : out std\_logic;

kamal\_outputd : out std\_logic

);

end kamal\_lab1halfaddercode;

architecture arch of kamal\_lab1halfaddercode is

begin

kamal\_outputc <= kamal\_inputa xor kamal\_inputb;

kamal\_outputd <= kamal\_inputa and kamal\_inputb;

end arch;

1. **Half-Adder Testbench VHDL code**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity kamal\_lab1halfadder\_tb is

end kamal\_lab1halfadder\_tb;

architecture tb of kamal\_lab1halfadder\_tb is

signal kamal2\_inputa: std\_logic := '0';

signal kamal2\_inputb: std\_logic := '0';

signal kamal2\_outputc: std\_logic;

signal kamal2\_outputd: std\_logic;

begin

UUT : entity work.kamal\_lab1halfaddercode

port map(

kamal\_inputa => kamal2\_inputa,

kamal\_inputb => kamal2\_inputb,

kamal\_outputc => kamal2\_outputc,

kamal\_outputd => kamal2\_outputd

);

process is

begin

kamal2\_inputa <= '0';

kamal2\_inputb <= '0';

wait for 10 ns;

kamal2\_inputa <= '0';

kamal2\_inputb <= '1';

wait for 10 ns;

kamal2\_inputa <= '1';

kamal2\_inputb <= '0';

wait for 10 ns;

kamal2\_inputa <= '1';

kamal2\_inputb <= '1';

wait for 10 ns;

end process;

end tb;

1. **PIN ASSIGNMENTS for Half-Adder**

To, Location

kamal\_inputa,PIN\_AB28

kamal\_inputb,PIN\_AC28

kamal\_outputc,PIN\_G19

kamal\_outputd,PIN\_F19

1. **Half-Adder LPM VHDL code**

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

LIBRARY lpm;

USE lpm.all;

ENTITY kamal\_lab1\_AdderLPM IS

PORT

(

dataa : IN STD\_LOGIC\_VECTOR (0 DOWNTO 0);

datab : IN STD\_LOGIC\_VECTOR (0 DOWNTO 0);

cout : OUT STD\_LOGIC ;

result : OUT STD\_LOGIC\_VECTOR (0 DOWNTO 0)

);

END kamal\_lab1\_AdderLPM;

ARCHITECTURE SYN OF kamal\_lab1\_adderlpm IS

SIGNAL sub\_wire0 : STD\_LOGIC ;

SIGNAL sub\_wire1 : STD\_LOGIC\_VECTOR (0 DOWNTO 0);

COMPONENT lpm\_add\_sub

GENERIC (

lpm\_direction : STRING;

lpm\_hint : STRING;

lpm\_representation : STRING;

lpm\_type : STRING;

lpm\_width : NATURAL

);

PORT (

cout : OUT STD\_LOGIC ;

dataa : IN STD\_LOGIC\_VECTOR (0 DOWNTO 0);

datab : IN STD\_LOGIC\_VECTOR (0 DOWNTO 0);

result : OUT STD\_LOGIC\_VECTOR (0 DOWNTO 0)

);

END COMPONENT;

BEGIN

cout <= sub\_wire0;

result <= sub\_wire1(0 DOWNTO 0);

LPM\_ADD\_SUB\_component : LPM\_ADD\_SUB

GENERIC MAP (

lpm\_direction => "ADD",

lpm\_hint => "ONE\_INPUT\_IS\_CONSTANT=NO,CIN\_USED=NO",

lpm\_representation => "SIGNED",

lpm\_type => "LPM\_ADD\_SUB",

lpm\_width => 1

)

PORT MAP (

dataa => dataa,

datab => datab,

cout => sub\_wire0,

result => sub\_wire1

);

END SYN;

1. **PIN ASSIGNMENTS for Half-Adder LPM**

To, Location

dataa,PIN\_AB28

datab,PIN\_AC28

cout,PIN\_G19

result,PIN\_F19

1. **Full-Adder VHDL code**

library ieee;

use ieee.std\_logic\_1164.all;

entity kamal\_lab1fulladdercode is

port(

kamal\_inputa : in std\_logic;

kamal\_inputb : in std\_logic;

kamal\_inputCin : in std\_logic;

kamal\_outputs : out std\_logic;

kamal\_outputCout : out std\_logic

);

end kamal\_lab1fulladdercode;

architecture arch of kamal\_lab1fulladdercode is

begin

kamal\_outputs <= kamal\_inputa xor kamal\_inputb xor kamal\_inputCin;

kamal\_outputCout <= (kamal\_inputa and kamal\_inputb) or (kamal\_inputCin and kamal\_inputa) or (kamal\_inputCin and kamal\_inputb);

end arch;

1. **Full-Adder Testbench VHDL code**

library ieee;

use ieee.std\_logic\_1164.all;

entity kamal\_lab1fulladder\_tb is

end kamal\_lab1fulladder\_tb;

architecture tb of kamal\_lab1fulladder\_tb is

signal kamal2\_inputa: std\_logic := '0';

signal kamal2\_inputb: std\_logic := '0';

signal kamal2\_inputCin: std\_logic := '0';

signal kamal2\_outputs: std\_logic;

signal kamal2\_outputCout: std\_logic;

BEGIN

UUT: entity work.kamal\_lab1fulladdercode

port map(

kamal\_inputa => kamal2\_inputa,

kamal\_inputb => kamal2\_inputb,

kamal\_inputCin => kamal2\_inputCin,

kamal\_outputs => kamal2\_outputs,

kamal\_outputCout => kamal2\_outputCout

);

process is

begin

wait for 10 ns;

kamal2\_inputa <= '0';

kamal2\_inputb <= '0';

kamal2\_inputCin <= '0';

wait for 10 ns;

kamal2\_inputa <= '0';

kamal2\_inputb <= '0';

kamal2\_inputCin <= '1';

wait for 10 ns;

kamal2\_inputa <= '0';

kamal2\_inputb <= '1';

kamal2\_inputCin <= '0';

wait for 10 ns;

kamal2\_inputa <= '0';

kamal2\_inputb <= '1';

kamal2\_inputCin <= '1';

wait for 10 ns;

kamal2\_inputa <= '1';

kamal2\_inputb <= '0';

kamal2\_inputCin <= '0';

wait for 10 ns;

kamal2\_inputa <= '1';

kamal2\_inputb <= '0';

kamal2\_inputCin <= '1';

wait for 10 ns;

kamal2\_inputa <= '1';

kamal2\_inputb <= '1';

kamal2\_inputCin <= '0';

wait for 10 ns;

kamal2\_inputa <= '1';

kamal2\_inputb <= '1';

kamal2\_inputCin <= '1';

wait for 10 ns;

end process;

END;

1. **PIN ASSIGNMENTS for Full-Adder**

To, Location

kamal\_inputa,PIN\_AB28

kamal\_inputb,PIN\_AC28

kamal\_inputCin,PIN\_AC27

kamal\_outputs,PIN\_G19

kamal\_outputCout,PIN\_F19

1. **Full-Adder LPM VHDL code**

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

LIBRARY lpm;

USE lpm.all;

ENTITY kamal\_lab1\_fulladderlpm IS

PORT

(

cin : IN STD\_LOGIC ;

dataa : IN STD\_LOGIC\_VECTOR (0 DOWNTO 0);

datab : IN STD\_LOGIC\_VECTOR (0 DOWNTO 0);

cout : OUT STD\_LOGIC ;

result : OUT STD\_LOGIC\_VECTOR (0 DOWNTO 0)

);

END kamal\_lab1\_fulladderlpm;

ARCHITECTURE SYN OF kamal\_lab1\_fulladderlpm IS

SIGNAL sub\_wire0 : STD\_LOGIC ;

SIGNAL sub\_wire1 : STD\_LOGIC\_VECTOR (0 DOWNTO 0);

COMPONENT lpm\_add\_sub

GENERIC (

lpm\_direction : STRING;

lpm\_hint : STRING;

lpm\_representation : STRING;

lpm\_type : STRING;

lpm\_width : NATURAL

);

PORT (

cin : IN STD\_LOGIC ;

datab : IN STD\_LOGIC\_VECTOR (0 DOWNTO 0);

cout : OUT STD\_LOGIC ;

dataa : IN STD\_LOGIC\_VECTOR (0 DOWNTO 0);

result : OUT STD\_LOGIC\_VECTOR (0 DOWNTO 0)

);

END COMPONENT;

BEGIN

cout <= sub\_wire0;

result <= sub\_wire1(0 DOWNTO 0);

LPM\_ADD\_SUB\_component : LPM\_ADD\_SUB

GENERIC MAP (

lpm\_direction => "ADD",

lpm\_hint => "ONE\_INPUT\_IS\_CONSTANT=NO,CIN\_USED=YES",

lpm\_representation => "UNSIGNED",

lpm\_type => "LPM\_ADD\_SUB",

lpm\_width => 1

)

PORT MAP (

cin => cin,

datab => datab,

dataa => dataa,

cout => sub\_wire0,

result => sub\_wire1

);

END SYN;

1. **PIN ASSIGNMENTS for Full-Adder LPM**

To, Location

cin,PIN\_AB28

dataa,PIN\_AC28

datab,PIN\_AC27

cout,PIN\_G19

result,PIN\_F19

1. **3-to-8 Decoder VHDL code**

library ieee;

use ieee.std\_logic\_1164.all;

entity kamal\_lab1\_3to8Decodercode is

Port(

kamal\_input1 : in std\_logic;

kamal\_input2 : in std\_logic;

kamal\_input3 : in std\_logic;

kamal\_outputz0 : out std\_logic;

kamal\_outputz1 : out std\_logic;

kamal\_outputz2 : out std\_logic;

kamal\_outputz3 : out std\_logic;

kamal\_outputz4 : out std\_logic;

kamal\_outputz5 : out std\_logic;

kamal\_outputz6 : out std\_logic;

kamal\_outputz7 : out std\_logic

);

end kamal\_lab1\_3to8Decodercode;

architecture gate of kamal\_lab1\_3to8Decodercode is

begin

kamal\_outputz0 <= (not kamal\_input1) and (not kamal\_input2) and (not kamal\_input3);

kamal\_outputz1 <= (not kamal\_input1) and (not kamal\_input2) and (kamal\_input3);

kamal\_outputz2 <= (not kamal\_input1) and (kamal\_input2) and (not kamal\_input3);

kamal\_outputz3 <= (not kamal\_input1) and (kamal\_input2) and (kamal\_input3);

kamal\_outputz4 <= (kamal\_input1) and (not kamal\_input2) and (not kamal\_input3);

kamal\_outputz5 <= (kamal\_input1) and (not kamal\_input2) and (kamal\_input1);

kamal\_outputz6 <= (kamal\_input1) and (kamal\_input2) and (not kamal\_input3);

kamal\_outputz7 <= (kamal\_input1) and (kamal\_input2) and (kamal\_input3);

end gate;

1. **3-to-8 Decoder Testbench VHDL code**

library ieee;

use ieee.std\_logic\_1164.all;

entity kamal\_lab1\_3to8Decoder\_tb is

end kamal\_lab1\_3to8Decoder\_tb;

architecture tb of kamal\_lab1\_3to8Decoder\_tb is

signal kamal2\_input1: std\_logic := '0';

signal kamal2\_input2: std\_logic := '0';

signal kamal2\_input3: std\_logic := '0';

signal kamal2\_outputz0: std\_logic;

signal kamal2\_outputz1: std\_logic;

signal kamal2\_outputz2: std\_logic;

signal kamal2\_outputz3: std\_logic;

signal kamal2\_outputz4: std\_logic;

signal kamal2\_outputz5: std\_logic;

signal kamal2\_outputz6: std\_logic;

signal kamal2\_outputz7: std\_logic;

begin

UUT: entity work.kamal\_lab1\_3to8Decodercode

port map(

kamal\_input1 => kamal2\_input1,

kamal\_input2 => kamal2\_input2,

kamal\_input3 => kamal2\_input3,

kamal\_outputz0 => kamal2\_outputz0,

kamal\_outputz1 => kamal2\_outputz1,

kamal\_outputz2 => kamal2\_outputz2,

kamal\_outputz3 => kamal2\_outputz3,

kamal\_outputz4 => kamal2\_outputz4,

kamal\_outputz5 => kamal2\_outputz5,

kamal\_outputz6 => kamal2\_outputz6,

kamal\_outputz7 => kamal2\_outputz7

);

process is

begin

kamal2\_input1 <= '0';

kamal2\_input2 <= '0';

kamal2\_input3 <= '0';

wait for 10 ns;

kamal2\_input1 <= '0';

kamal2\_input2 <= '0';

kamal2\_input3 <= '1';

wait for 10 ns;

kamal2\_input1 <= '0';

kamal2\_input2 <= '1';

kamal2\_input3 <= '0';

wait for 10 ns;

kamal2\_input1 <= '0';

kamal2\_input2 <= '1';

kamal2\_input3 <= '1';

wait for 10 ns;

kamal2\_input1 <= '1';

kamal2\_input2 <= '0';

kamal2\_input3 <= '0';

wait for 10 ns;

kamal2\_input1 <= '1';

kamal2\_input2 <= '0';

kamal2\_input3 <= '1';

wait for 10 ns;

kamal2\_input1 <= '1';

kamal2\_input2 <= '1';

kamal2\_input3 <= '0';

wait for 10 ns;

kamal2\_input1 <= '1';

kamal2\_input2 <= '1';

kamal2\_input3 <= '1';

end process;

end tb;

1. **PIN ASSIGNMENT for 3-to-8 Decoder**

To, Location

kamal\_input1,PIN\_AB28

kamal\_input2,PIN\_AC28

kamal\_input3,PIN\_AC27

kamal\_outputz0,PIN\_G19

kamal\_outputz1,PIN\_F19

kamal\_outputz2,PIN\_E19

kamal\_outputz3,PIN\_F21

kamal\_outputz4,PIN\_F18

kamal\_outputz5,PIN\_E18

kamal\_outputz6,PIN\_J19

kamal\_outputz7,PIN\_H19

1. **3-to-8 Decoder LPM VHDL code**

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

LIBRARY lpm;

USE lpm.all;

ENTITY kamal\_Lab1\_Decoder IS

PORT

(

data : IN STD\_LOGIC\_VECTOR (2 DOWNTO 0);

eq7 : OUT STD\_LOGIC

);

END kamal\_Lab1\_Decoder;

ARCHITECTURE SYN OF kamal\_lab1\_decoder IS

SIGNAL sub\_wire0 : STD\_LOGIC\_VECTOR (7 DOWNTO 0);

SIGNAL sub\_wire1 : STD\_LOGIC ;

COMPONENT lpm\_decode

GENERIC (

lpm\_decodes : NATURAL;

lpm\_type : STRING;

lpm\_width : NATURAL

);

PORT (

data : IN STD\_LOGIC\_VECTOR (2 DOWNTO 0);

eq : OUT STD\_LOGIC\_VECTOR (7 DOWNTO 0)

);

END COMPONENT;

BEGIN

sub\_wire1 <= sub\_wire0(7);

eq7 <= sub\_wire1;

LPM\_DECODE\_component : LPM\_DECODE

GENERIC MAP (

lpm\_decodes => 8,

lpm\_type => "LPM\_DECODE",

lpm\_width => 3

)

PORT MAP (

data => data,

eq => sub\_wire0

);

END SYN;

1. **PIN ASSIGNMENT for 3-to-8 Decoder LPM**

To, Location

data[2],PIN\_AB28

data[1],PIN\_AC28

data[0],PIN\_AC27

eq7,PIN\_G19

1. **8-to-3 Encoder VHDL code**

library ieee;

use ieee.std\_logic\_1164.all;

entity kamal\_lab1\_8to3Encodercode is

port(

kamal\_input0 : in std\_logic;

kamal\_input1 : in std\_logic;

kamal\_input2 : in std\_logic;

kamal\_input3 : in std\_logic;

kamal\_input4 : in std\_logic;

kamal\_input5 : in std\_logic;

kamal\_input6 : in std\_logic;

kamal\_input7 : in std\_logic;

kamal\_outputY1 : out std\_logic;

kamal\_outputY2 : out std\_logic;

kamal\_outputY3 : out std\_logic

);

end kamal\_lab1\_8to3Encodercode;

architecture gate of kamal\_lab1\_8to3Encodercode is

begin

kamal\_outputY1 <= (kamal\_input7) or (kamal\_input4) or (kamal\_input5) or (kamal\_input6);

kamal\_outputY2 <= (kamal\_input3) or (kamal\_input2) or (kamal\_input6) or (kamal\_input7);

kamal\_outputY3 <= (kamal\_input7) or (kamal\_input5) or (kamal\_input3) or (kamal\_input1);

end gate;

1. **8-to-3 Encoder Testbench VHDL code**

library ieee;

use ieee.std\_logic\_1164.all;

entity kamal\_lab1\_8to3Encoder\_tb is

end kamal\_lab1\_8to3Encoder\_tb;

architecture tb of kamal\_lab1\_8to3Encoder\_tb is

signal kamal2\_input0: std\_logic := '0';

signal kamal2\_input1: std\_logic := '0';

signal kamal2\_input2: std\_logic := '0';

signal kamal2\_input3: std\_logic := '0';

signal kamal2\_input4: std\_logic := '0';

signal kamal2\_input5: std\_logic := '0';

signal kamal2\_input6: std\_logic := '0';

signal kamal2\_input7: std\_logic := '0';

signal kamal2\_outputY1: std\_logic;

signal kamal2\_outputY2: std\_logic;

signal kamal2\_outputY3: std\_logic;

begin

UUT: entity work.kamal\_lab1\_8to3Encodercode

port map(

kamal\_input0 => kamal2\_input0,

kamal\_input1 => kamal2\_input1,

kamal\_input2 => kamal2\_input2,

kamal\_input3 => kamal2\_input3,

kamal\_input4 => kamal2\_input4,

kamal\_input5 => kamal2\_input5,

kamal\_input6 => kamal2\_input6,

kamal\_input7 => kamal2\_input7,

kamal\_outputY1 => kamal2\_outputY1,

kamal\_outputY2 => kamal2\_outputY2,

kamal\_outputY3 => kamal2\_outputY3

);

process is

begin

kamal2\_input0 <= '1';

kamal2\_input1 <= '0';

kamal2\_input2 <= '0';

kamal2\_input3 <= '0';

kamal2\_input4 <= '0';

kamal2\_input5 <= '0';

kamal2\_input6 <= '0';

kamal2\_input7 <= '0';

wait for 10 ns;

kamal2\_input0 <= '0';

kamal2\_input1 <= '1';

kamal2\_input2 <= '0';

kamal2\_input3 <= '0';

kamal2\_input4 <= '0';

kamal2\_input5 <= '0';

kamal2\_input6 <= '0';

kamal2\_input7 <= '0';

wait for 10 ns;

kamal2\_input0 <= '0';

kamal2\_input1 <= '0';

kamal2\_input2 <= '1';

kamal2\_input3 <= '0';

kamal2\_input4 <= '0';

kamal2\_input5 <= '0';

kamal2\_input6 <= '0';

kamal2\_input7 <= '0';

wait for 10 ns;

kamal2\_input0 <= '0';

kamal2\_input1 <= '0';

kamal2\_input2 <= '0';

kamal2\_input3 <= '1';

kamal2\_input4 <= '0';

kamal2\_input5 <= '0';

kamal2\_input6 <= '0';

kamal2\_input7 <= '0';

wait for 10 ns;

kamal2\_input0 <= '0';

kamal2\_input1 <= '0';

kamal2\_input2 <= '0';

kamal2\_input3 <= '0';

kamal2\_input4 <= '1';

kamal2\_input5 <= '0';

kamal2\_input6 <= '0';

kamal2\_input7 <= '0';

wait for 10 ns;

kamal2\_input0 <= '0';

kamal2\_input1 <= '0';

kamal2\_input2 <= '0';

kamal2\_input3 <= '0';

kamal2\_input4 <= '0';

kamal2\_input5 <= '1';

kamal2\_input6 <= '0';

kamal2\_input7 <= '0';

wait for 10 ns;

kamal2\_input0 <= '0';

kamal2\_input1 <= '0';

kamal2\_input2 <= '0';

kamal2\_input3 <= '0';

kamal2\_input4 <= '0';

kamal2\_input5 <= '0';

kamal2\_input6 <= '1';

kamal2\_input7 <= '0';

wait for 10 ns;

kamal2\_input0 <= '0';

kamal2\_input1 <= '0';

kamal2\_input2 <= '0';

kamal2\_input3 <= '0';

kamal2\_input4 <= '0';

kamal2\_input5 <= '0';

kamal2\_input6 <= '0';

kamal2\_input7 <= '1';

wait for 10 ns;

end process;

end tb;

1. **PIN ASSIGNMENT for 8-to-3 Encoder**

To, Location

kamal\_input1,PIN\_AB28

kamal\_input2,PIN\_AC28

kamal\_input3,PIN\_AC27

kamal\_input4,PIN\_AD27

kamal\_input5,PIN\_AB27

kamal\_input6,PIN\_AC26

kamal\_input7,PIN\_AD26

kamal\_outputY1,PIN\_G19

kamal\_outputY2,PIN\_F19

kamal\_outputY3,PIN\_E19