**FACULTY OF SCIENCE AND TECHNOLOGY**

**Assignment (Laboratory) Coversheet**

|  |  |
| --- | --- |
| Student ID number | U3312590 |
| Student Name | Jasmine Bajracharya |
| Unit name | Introduction to Computer Engineering |
| Unit number | 10096 |
| Name of lecturer/tutor | Dr. Julio Romero |
| Assignment topic | HDL and Microprocessors |
| Due date | 2nd November, 2025 |
| Word Count | 1404 |

***You must keep a photocopy or electronic copy of your assignment.***

**Student declaration**

I certify that the attached assignment is my own work. Material drawn from other sources has been appropriately and fully acknowledged as to author/creator, source and other bibliographic details. Such referencing may need to meet unit-specific requirements as to format and style.

I give permission for my assignment to be copied, submitted and retained for the electronic checking of plagiarism.

**Signature of student: A close-up of a signature

AI-generated content may be incorrect. Date: 30th October 2025**

(Students submitting work electronically can type their name in the space for signature above, but

must produce a signed copy of this coversheet on request.)

**Date of submission: 30th October 2025**

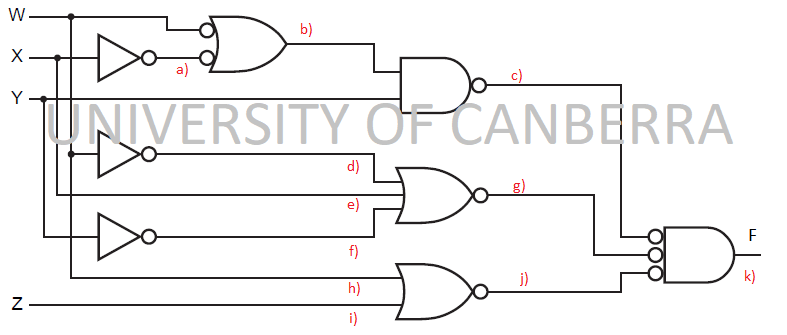


Figure 1: Boolean Circuit for Laboratory Week 12

# **1. Boolean Expression Derivation from Circuit (20 marks)**

Derive the Boolean expression that describes the circuit shown in Figure 1. Show your steps in each branch of the circuit given.

A diagram of a circuit

AI-generated content may be incorrect.

Here, we have for each intermediate expression we get,

# **2. Boolean Expression Simplification Using Theorems (10 marks)**

On simplification, the expression is given as:

# **3. Boolean Expression Simplification Using Karnaugh Maps (10 marks)**

The truth table for the above circuit is given as:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **W** | **X** | **Y** | **Z** | **F** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Now, using Karnaugh map for the simplification of the expression, we get:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | 0 | 0 | 1 | 0 |
|  | 0 | 0 | 1 | 0 |
|  | 0 | 0 | 1 | 1 |
|  | 0 | 0 | 0 | 0 |

On grouping the rows for 1’s we get:

The results from the simplification and Karnaugh map are different due to the redundancy in the terms in the manually simplified expression. The results are, however, equivalent.

# **4. VHDL Code representation SA (20 marks)**

From simplification obtained above, and using the structural approach, the code representation using structural approach is given as:  
  
considering , W -> IN1, X -> IN2, Y-> IN3, Z-> IN4 and F -> OUT

entity Logic is

port(IN1, IN2, IN3, IN4 : in bit; OUT5: out bit);

architecture LogicCircuit of Circuit is

component NOT\_gate is

port(A:in bit;X:out bit);

end component NOT\_GATE;

component AND\_gate3 is

port(A, B,C: in bit; X:out bit);

end component AND\_GATE;

component OR\_gate3 is

port (A,B, C : in bit; X:out bit);

end component OR\_GATE;

signal OUT1, OUT2, OUT3, OUT4: bit;

begin

G1: NOT\_gate port map( A=>IN1, X=>OUT1);

G2: And\_gate3 port map (A=>IN1, B=>IN2, C=>IN3, X=>OUT2)

G3: And\_gate3 port map (A=>IN2, B=>IN3, C=>IN4, X=>OUT3)

G4: And\_gate3 port map (A=>IN2, B=>OUT1, C=>IN4, X=>OUT4)

G5: OR\_gate3 port map (A=> OUT2 , B=>OUT3, C=>OUT4, X=>OUT5)

end architecture LogicCircuit;

# **5. VHDL Code representation DF (20 marks)**

From simplification obtained above, using the Data Flow approach, we get the code as:

entity Logic is

port(

W: in STD\_LOGIC;

X: in STD\_LOGIC;

Y : in STD\_LOGIC;

Z : in STD\_LOGIC;

F : out STD\_LOGIC;

);

End Logic;

Architecture Behavioral of Logic is

begin

F<= (W and X and Y) or (X and Y and Z) or (not W and Y and Z)

end Behavioral;

# **6. VHDL Implementation (20 marks)**

From simplification obtained above, and using the data flow approach,

**6.1** Correct structure of the code written in Vivado. It includes proper I/O mapping.

The code written in Vivado is as follows:

library IEEE;

use IEE.STD\_LOGIC\_1164.ALL;  
  
entity tute\_12 is

port(

W: in STD\_LOGIC;

X: in STD\_LOGIC;

Y : in STD\_LOGIC;

Z : in STD\_LOGIC;

F : out STD\_LOGIC;

);

End tute\_12;

Architecture Behavioral of tute\_12 is

begin

F<= (W and X and Y) or (X and Y and Z) or (not W and Y and Z)

end Behavioral;

Similarly, the testbench for the above is written as:

Library IEEE;

Use IEEE.STD\_LOGIC\_1164.ALL;

Entity tb\_and\_gate is

End tb\_and\_gate;

architecture behavior of tb\_and\_gate is

Signal A, B, Y; STD\_LOGIC;

Begin

uut.entity work.tute\_12

port map(

W => W,

X => X,

Y=> Y,

Z=>Z,

F=> F);

-- *Test process*

process

begin

W <= ‘0’ ; X <= ‘0’ ; Y <= ‘0’ ; Z<=’0’ wait for 10 ns;

W <= ‘0’ ; X <= ‘0’ ; Y <= ‘0’ ; Z<=’1’ wait for 10 ns;

W <= ‘0’ ; X <= ‘0’ ; Y <= ‘1’ ; Z<=’0’ wait for 10 ns;

W <= ‘0’ ; X <= ‘0’ ; Y <= ‘1’ ; Z<=’1’ wait for 10 ns;

W <= ‘0’ ; X <= ‘1’ ; Y <= ‘0’ ; Z<=’0’ wait for 10 ns;

W <= ‘0’ ; X <= ‘1’ ; Y <= ‘0’ ; Z<=’1’ wait for 10 ns;

W <= ‘0’ ; X <= ‘1’ ; Y <= ‘1’ ; Z<=’0’ wait for 10 ns;

W <= ‘0’ ; X <= ‘1’ ; Y <= ‘1’ ; Z<=’1’ wait for 10 ns;

W <= ‘1’ ; X <= ‘0’ ; Y <= ‘0’ ; Z<=’0’ wait for 10 ns;

W <= ‘1’ ; X <= ‘0’ ; Y <= ‘0’ ; Z<=’1’ wait for 10 ns;

W <= ‘1’ ; X <= ‘0’ ; Y <= ‘1’ ; Z<=’0’ wait for 10 ns;

W <= ‘1’ ; X <= ‘0’ ; Y <= ‘1’ ; Z<=’1’ wait for 10 ns;

W <= ‘1’ ; X <= ‘1’ ; Y <= ‘0’ ; Z<=’0’ wait for 10 ns;

W <= ‘1’ ; X <= ‘1’ ; Y <= ‘0’ ; Z<=’1’ wait for 10 ns;

W <= ‘1’ ; X <= ‘1’ ; Y <= ‘1’ ; Z<=’0’ wait for 10 ns;

W <= ‘1’ ; X <= ‘1’ ; Y <= ‘1’ ; Z<=’1’ wait for 10 ns;

Wait;

end process;

end behavior;

**6.2** Correct simulation of the circuit in Vivado (time diagram).

The circuit was simulated, and the timing diagram was obtained as:

A computer screen with a diagram

AI-generated content may be incorrect.  
  
A computer screen with a green and black graph

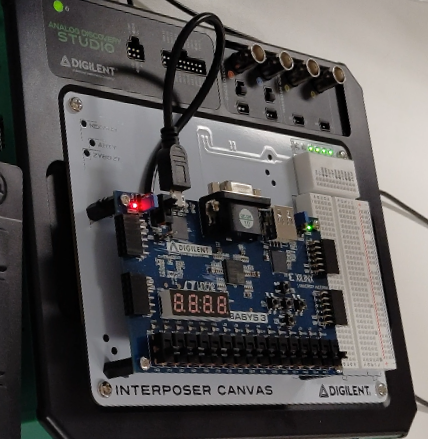
AI-generated content may be incorrect.  
  
*\*Note: The simulation video (timing diagram) can be accessed through the file timing\_diagram\_video/ timing\_diagram.mp4*

**6.3** Correct functioning of the FPGA by verifying that your circuit works as intended using a truth table.

The truth table for the circuit is as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **W** | **X** | **Y** | **Z** | **F** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

The FPGA implementation of the circuit is given as:

  
  
*\*Note: The FPGA implementation can be accessed through the file : FPGA\_video/ FPGA\_implementation.mp4*

# **REFERENCES**

[1] Floyd, Th.L. Digital Fundamentals, Pearson, 2015

[2] Boolean algebra website: https://www.boolean-algebra.com