**National University of Computer and Emerging Sciences**

**Digital Logic Design Lab 03**

****

Fast School of Computing

FAST-NU, Lahore, Pakistan

Page **2** of **4**

**Objectives**

• Simulate, analyze, and integrate digital circuits with LogicWorks

• Laws of Boolean Algebra

• Minterms & Maxterms

• SOP & POS forms

Contents

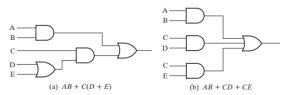
1. Circuit Optimization ..........................................................................................................................2 1.1 Literal Cost (L) ...............................................................................................................................2 1.2 Gate Input Cost (G) & Gate Input Cost with NOTs (GN) ...............................................................3 2. Parity Generator and Checker...........................................................................................................3 ACTIVITY 1.................................................................................................................................................3 ACTIVITY 2.................................................................................................................................................4 ACTIVITY 3.................................................................................................................................................4 ACTIVITY 4.................................................................................................................................................4 ACTIVITY 5.................................................................................................................................................4

**1. Circuit Optimization**

Circuit optimization is the process of achieving the simplest implementation for a given function through a systematic approach or algorithm. This optimization requires the use of specific cost criteria, such as literal cost (L), gate input cost (G), and gate input cost with NOTs (GN), to measure the simplicity of a circuit. The primary goal is to streamline the implementation while considering these cost factors, ensuring efficiency and effectiveness in the resulting circuit design.

**1.1 Literal Cost (L)**

A literal in the context of Boolean expressions represents either a variable or its complement. The literal cost is the number of literal appearances in a Boolean expression corresponding exactly to the logic diagram. For example, for the circuits in following diagram



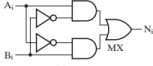
There are five literal appearances in the first equation and six in the second, so the first equation is the simplest in terms of literal cost.

Page **2** of **4**

© Hazoor Ahmad FAST NU LHR

Page **3** of **4**

**1.2 Gate Input Cost (G) & Gate Input Cost with NOTs (GN)** Gate Input Cost (G) refers to the count of inputs to gates in an implementation directly corresponding to the given equations, where inverters are not considered. Conversely, Gate Input Cost with inverters (GN) includes the count of inverters in the gate inputs. This cost can be determined easily from the logic diagram by simply counting the total number of inputs to the gates in the logic diagram. For G for circuit (a) and (b) are 8 and 9 respectively. However, consider the circuit below

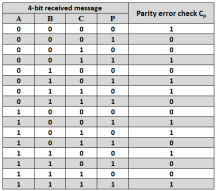
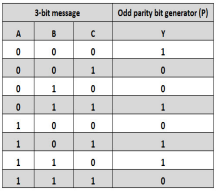


The G for ����is 6 but GN is 6+2 = 8.

**2. Parity Generator and Checker**

**A Parity Generator** functions as a combinational circuit designed to take in an n-1 bit data and produce an additional bit, known as the Parity Bit, to be transmitted alongside the bit stream. Conversely, a circuit responsible for verifying the parity at the receiver end is termed a **Parity Checker**. **Table 1** illustrates the truth table of an odd parity generator, where the parity bit is set to 1 to ensure an odd count of 1s in the truth table, even when the initial count is even. Meanwhile, **Table 2** presents the truth table for the Odd Parity Checker, indicating that PEC (Parity Error Check) equals 1 in the event of an error, specifically if the four received bits contain an even number of 1s; otherwise, PEC equals 0 if no errors occur, signifying that the 4-bit message contains an odd number of 1s.

**Table 1** Odd Parity Generator **Table 2** Odd Parity Checker

**Note for all activities:** Determine L, G & GN for every activity and report to the teacher.

**ACTIVITY 1**

Convert the following to **POS** form by using the **Distributive law,** implement on **Logic Works** and verify through truth table**.**

����′ + ����′

Page **3** of **4**

© Hazoor Ahmad FAST NU LHR

Page **4** of **4**

**Note: If you have already implemented this activity in previous lab on LogicWorks, Your Task is to implement this on Logic Trainer Board using Gate ICs and verify truth table.**

**ACTIVITY 2**

For the Boolean Functions **E** and **F**:

a) List min-term and max-terms of each function.

b) Express in sum-of-min-terms algebraic form.

c) Implement on **Logic works.**

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

**Note: If you have already implemented this activity in previous lab on LogicWorks, Your Task is to implement this on Logic Trainer Board using Gate ICs and verify truth table.**

**ACTIVITY 3**

Build a circuit for the Boolean function **F = AB + (A XOR B) C + B NAND D** and find the truth table by using LogicWorks. Implement **F** in Logic Works, and determine input & output waveforms.

**Note: If you have already implemented this activity in previous lab on LogicWorks, Your Task is to implement this on Logic Trainer Board using Gate ICs and verify truth table.**

**ACTIVITY 4**

Create a parity generator circuit for a 3-input message by following these steps: 1. Begin by constructing the truth table for the parity generator.

2. Utilize Karnaugh Maps (K-Maps) derived from the truth table to simplify the expression of P, the parity bit.

3. Proceed to implement the circuit on a Logic Trainer.

4. Verify the truth table of the implemented circuit to ensure accuracy and functionality. **ACTIVITY 5**

Extend the previously developed circuit to create a parity checker circuit by following these steps: 1. Begin by constructing the truth table for the parity checker based on the input and output requirements.

2. Use Karnaugh Maps (K-Maps) derived from the truth table to simplify the expression of C\_p, the parity check output.

3. Implement the parity checker circuit on a Logic Trainer, extending the existing circuit appropriately.

4. Verify the truth table of the implemented parity checker circuit to ensure accurate functionality

Page **4** of **4**

© Hazoor Ahmad FAST NU LHR