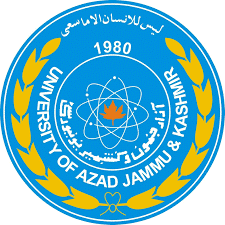
** The University of Azad Jammu & Kashmir, Muzaffarabad**

**Computer Architecture & Logic Design**

|  |  |
| --- | --- |
| **Student Name:** | Shahzad Ahmed Awan |
| **Roll No:** | 2024-SE-15 |
| **Course Title:** | C&LD |
| **Course Code:** | CS-1206 |
| **Instructor:** | Mam Sidra Rafique |
| **Submission Date:** | 09-August-2025 |
| **Department of Software Engineering** | |

**Lab 05: Types of Combinational Circuit**

**LAB Objective**

The primary objective of this lab is to design, analyze, and understand the working principles of Adder and Subtractor circuits in digital electronics. Through this practical work, students will gain both theoretical knowledge and hands-on experience in implementing arithmetic operations using combinational logic.

**Importance in Digital Systems**

* **Core Functionality**: Used in ALUs (Arithmetic Logic Units), microprocessors, and DSP systems.
* **High-Speed Operations**: Essential for real-time computation in embedded and high-performance applications.
* **Scalability**: Can be cascaded to handle multi-bit operations in larger data paths.

**Full Adder**

A **Full Adder** extends the half adder’s functionality by including a **carry-in (Cin)** input. This allows it to add three inputs: two significant bits (**A** and **B**) plus a carry from a previous addition. The outputs are:

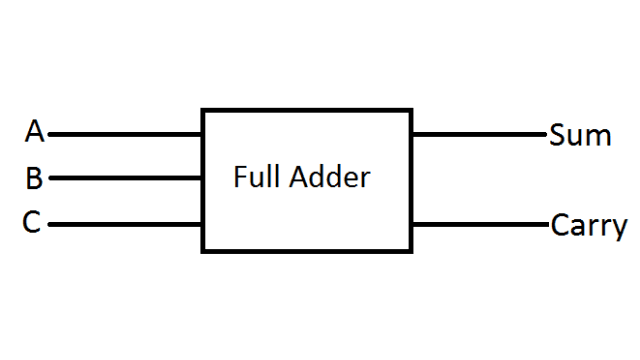
* **Sum** – final result of the addition.
* **Carry-out (Cout)** – carry generated for the next stage.

**Logic:**

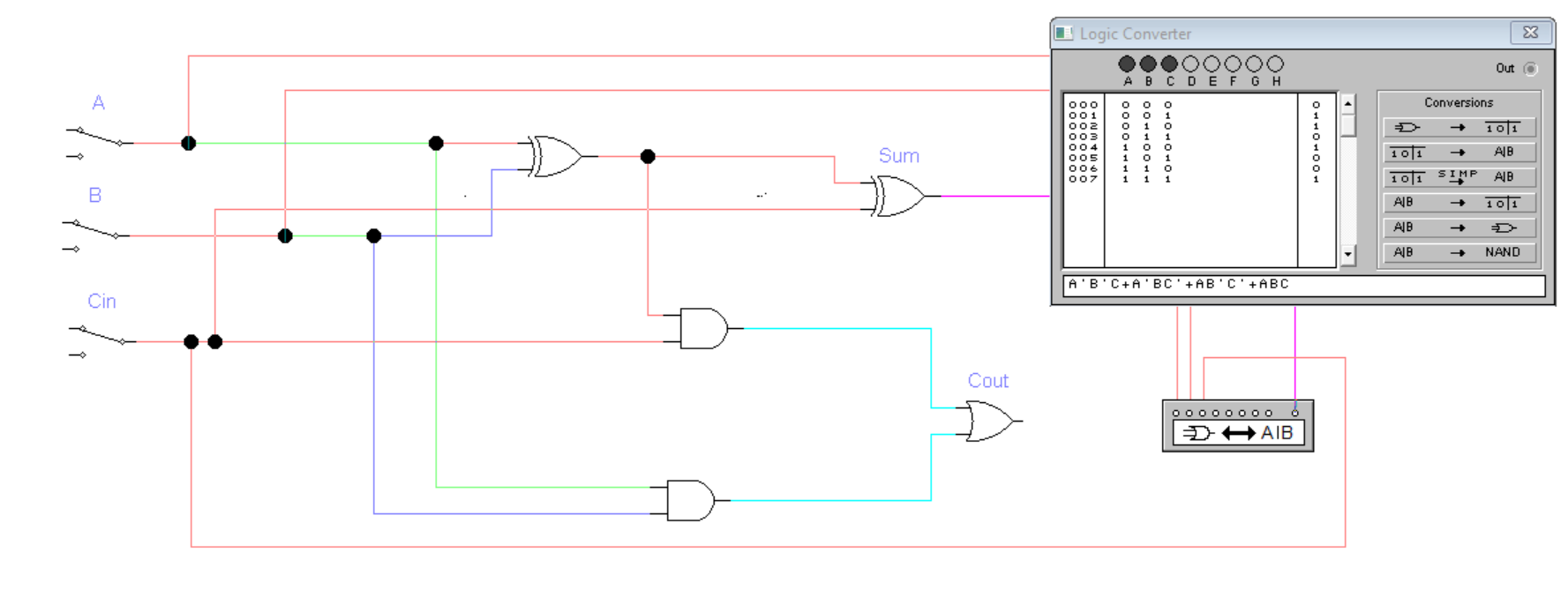
**Sum (S) = A ⊕ B ⊕ Cin**

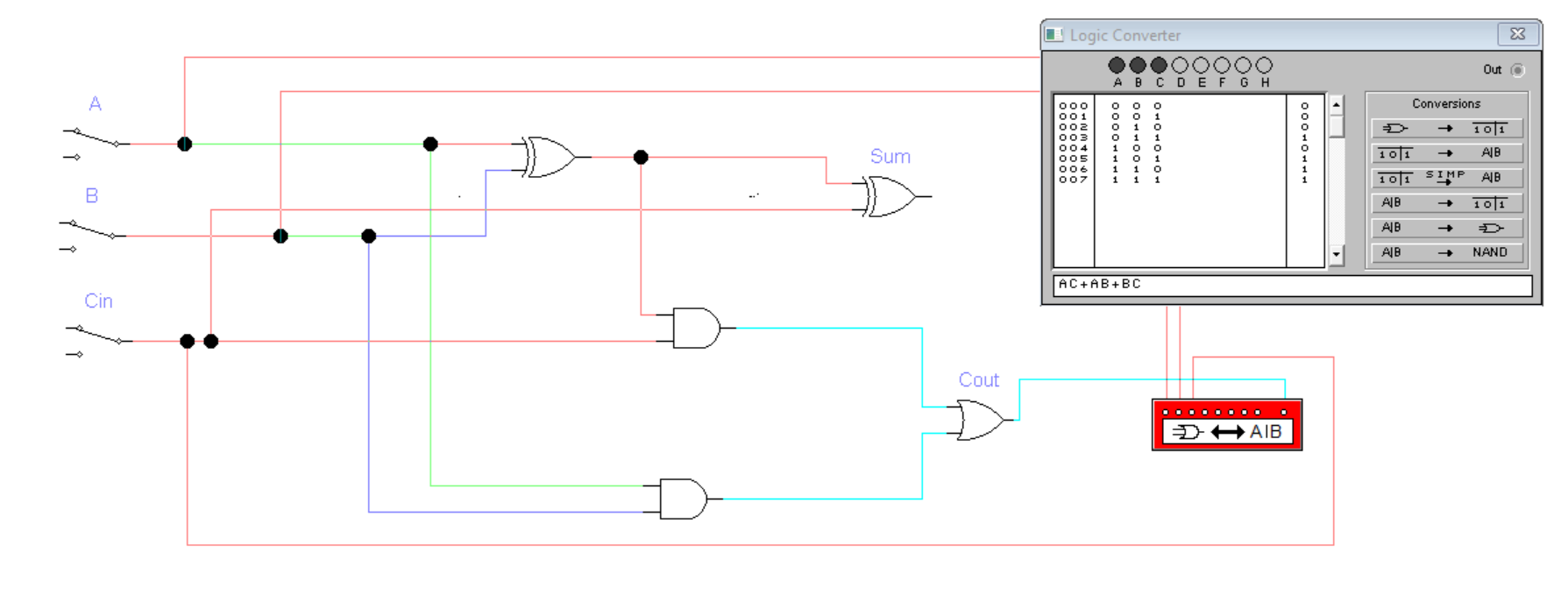
**Cout = (A · B) + (Cin · (A ⊕ B))**

**Block Diagram:**

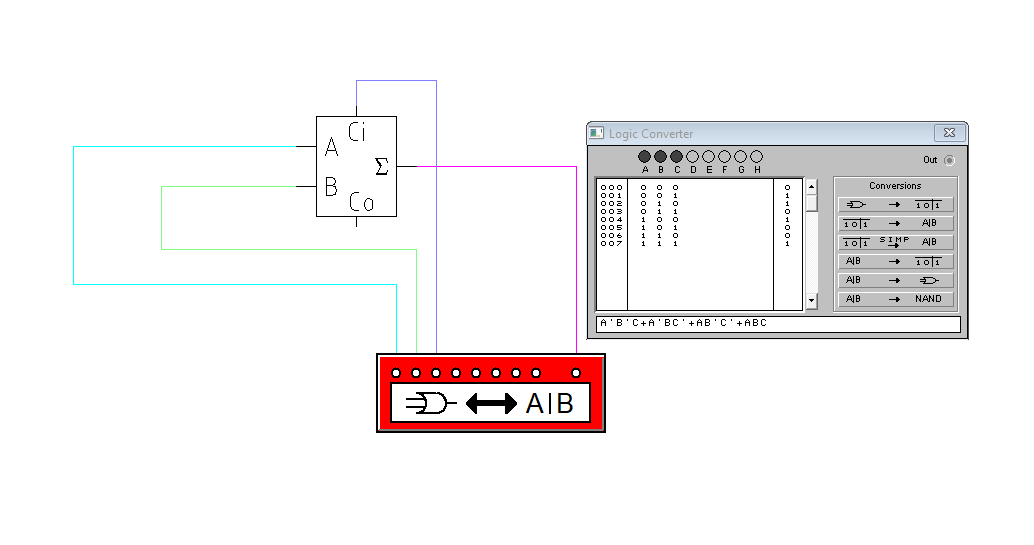


**Implementations in EWB**

**Sum with Truth Table:**

**CARRY:**

**Implementation using Built in Circuit**

****

**2. Half Subtractor**

A **Half Subtractor** is used for subtracting two binary bits (**A - B**). It has two outputs:

* **Difference (D)** – result of the subtraction.
* **Borrow (Bo)** – indicates if borrowing is needed from the next higher bit.

It is called “half” because it does not handle a borrow from a previous stage.

**Logic:**

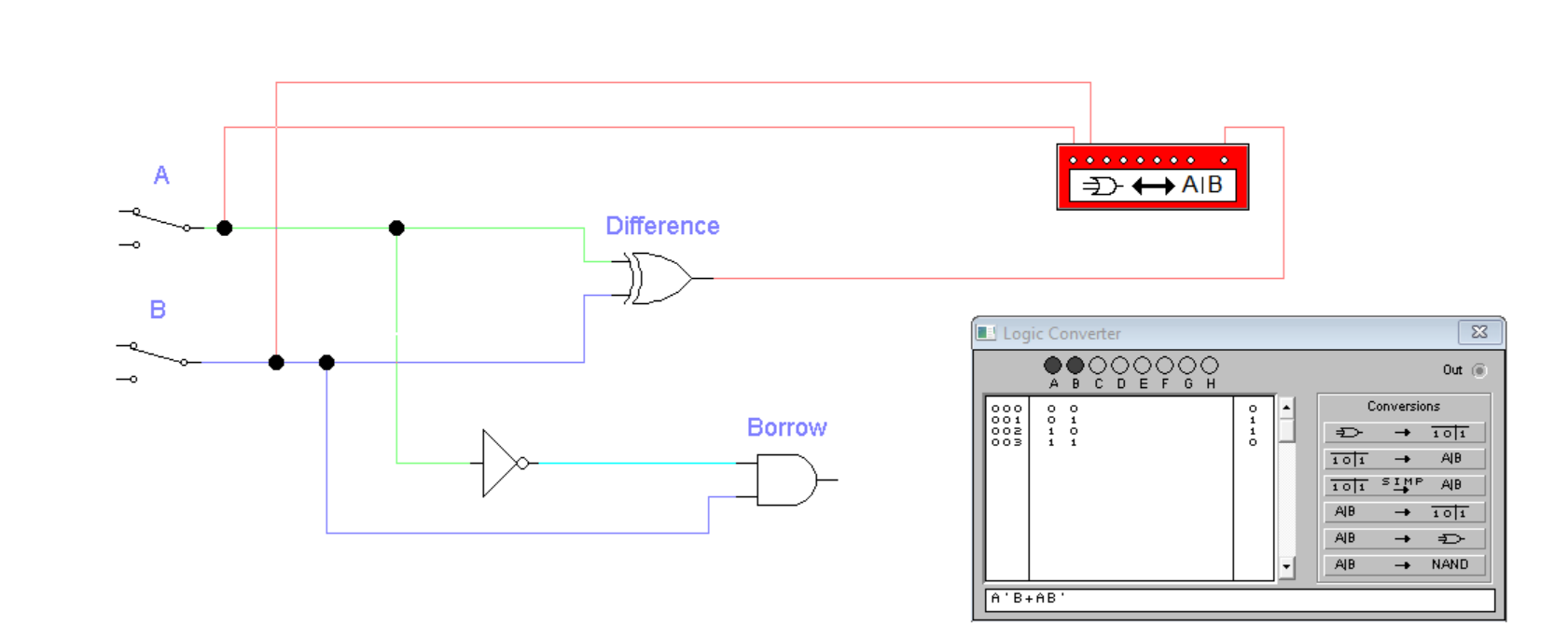
* **Difference (D) = A ⊕ B**
* **Borrow (Bo) = A̅ · B**

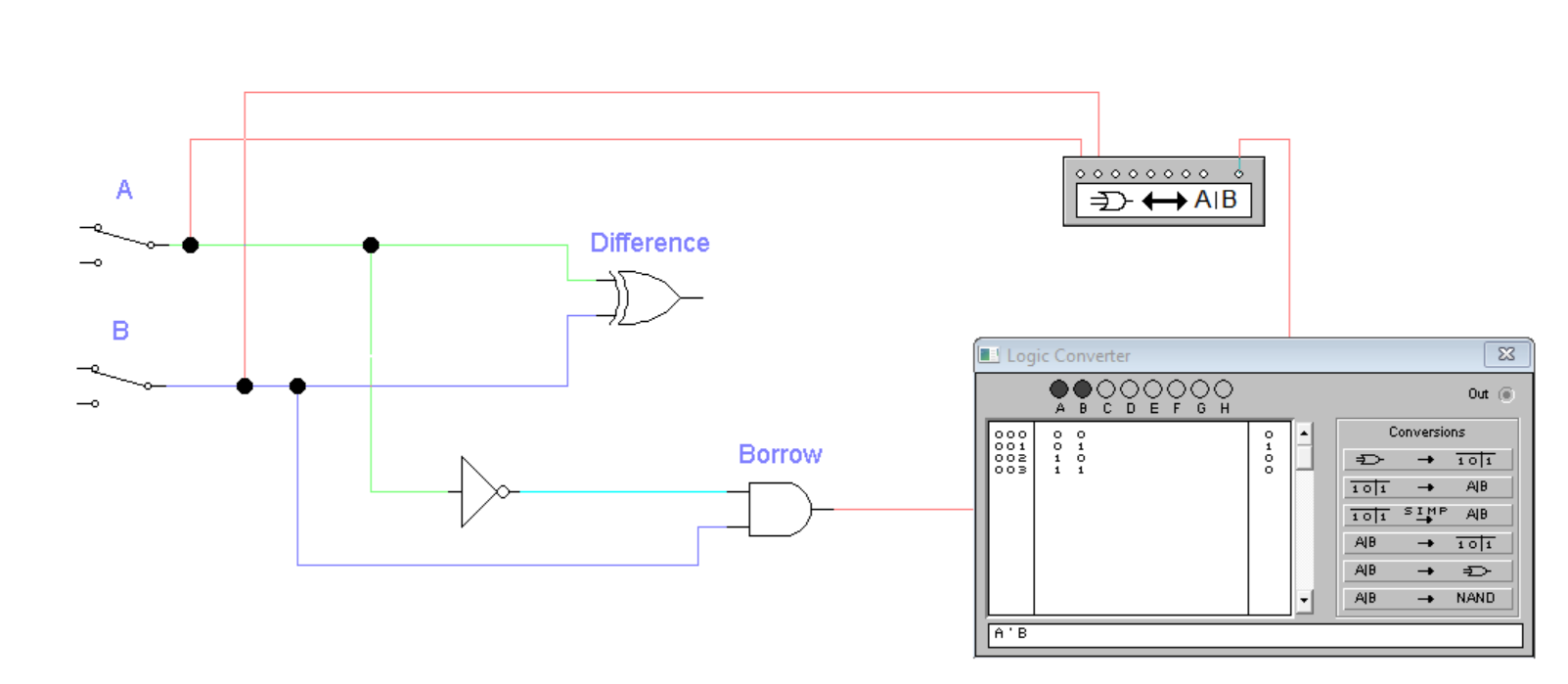
**Block Diagram:**

A black and white screen with text

AI-generated content may be incorrect.

**Implementation in EWB**

**Difference:**

**Borrow:**

**Full Subtractor**

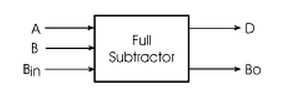
A **Full Subtractor** performs the subtraction of two binary bits with an additional **borrow-in (Bin)** input. This allows it to handle multi-bit subtraction by cascading multiple stages. It produces:

* **Difference (D)** – result of subtraction.
* **Borrow-out (Bout)** – borrow passed to the next stage.

**Logic:**

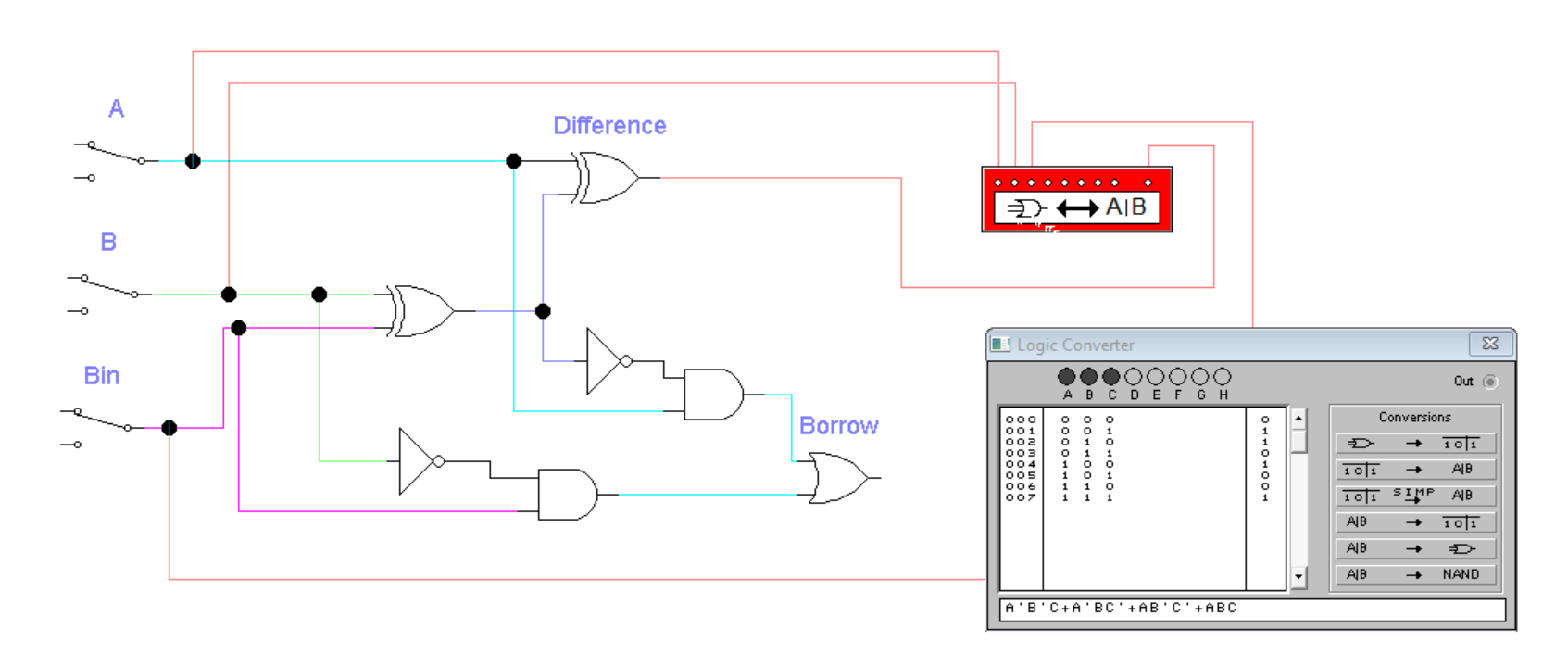
* **Difference (D) = A ⊕ B ⊕ Bin**
* **Bout = (B · Bin) + (A̅ · B) + (A̅ · Bin)**

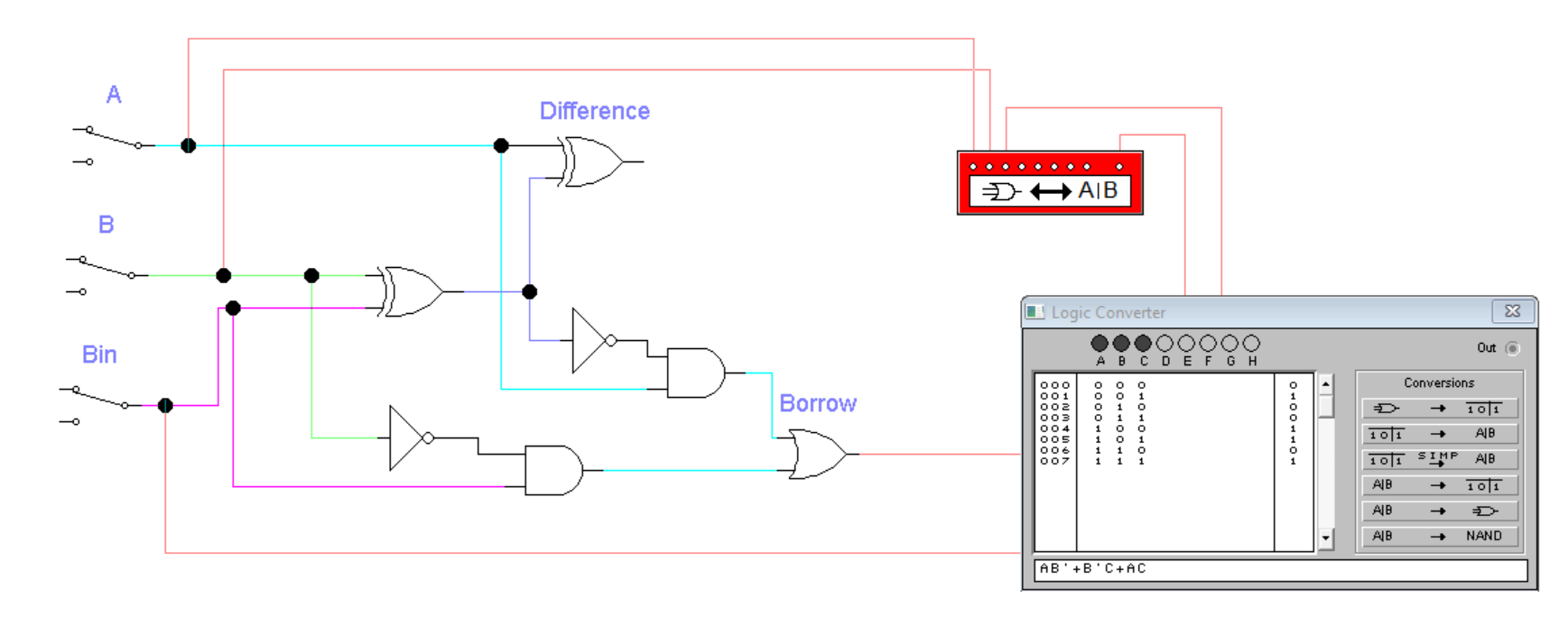
**Block Diagram:**

****

**Implementation in EWB**

**Difference:**

****

**Borrow:**

**LAB CONCLUSION WITH TRUTH TABLES**

**Full Adder Final Truth Table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | Cin | Sum | Cout |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

**HALF Subtractor Final Truth Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | Difference | Borrow |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

**Full Subtractor Final Truth Table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | Bin | Difference | Bout |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |