**Task1:**

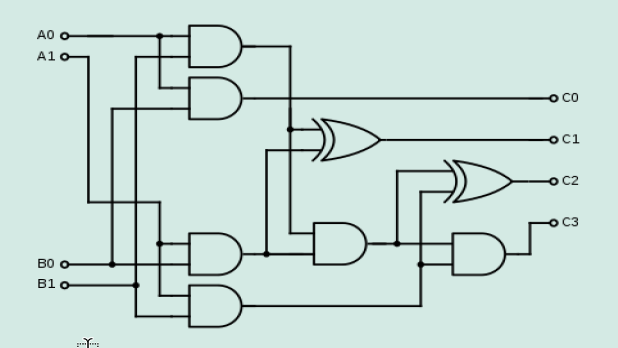
**1- Identify the main components of the given combinational logic system and extract the relations between each block.**

* **Main Components:**
  + - **AND gates**
    - **Partial Product**

**HALF ADDER:**

**XOR**

**AND GATE**



* **Main Components:**
  + - **AND gate**
    - **OR gate**
    - **XOR gate**

**Relations**

A0 B0 + 0 🡪**P0**

B0 A1 + B1 A0 🡪 **P1**, and C1 carryout in case that B0 A1 = 1 and B1 A0 = 1

C1 + B1 A1 🡪 **P2**, and C2 carryout in case that C1 = 1 and B1 A1 = 1

C2 🡪 **P3**

**Define the different types of logic components in the given combinational logic system.**

**Input:**

* **Switches**

- A0 - A1 - B0 - B1

* **System**

**- AND Gate**

Outputs 1 only if ALL inputs are 1. Think of it as a two-key lock requiring both keys to turn for the door to open.

Otherwise, it outputs 0. Even one missing key (0) keeps the door shut.

**- HALF ADDER**

It's a basic digital circuit that performs the addition of two single binary digits (bits). In this case the first half adder used to add (A0 B1 + A1B0), the sum is C1 and the carryout will be used as input to the second half adder which is adding (C1 + A1B1)

XOR gate

Outputs 1 if only ONE input is 1. Imagine an exclusive club door: you can only enter if either key (1) is used, not both.

Outputs 0 if BOTH inputs are the same (0 or 1). Both keys being the same (both 0s or both 1s) gets you denied entry.

AND Gate

* Outputs

- C0

- C1

- C2

- C3

**3. Explain the operation of the given combinational logic system as shown in Figure1 , make a good use of Truth Tables and Karnaugh Map (K-Map).**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A1 | A0 | B1 | B0 | P3 | P2 | P1 | P0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 1 | 1 |
| 11 | 0 | 1 | 0 | 1 |
| 10 | 0 | 1 | 1 | 0 |

**P0 P1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 1 | 1 | 0 |
| 11 | 0 | 1 | 1 | 0 |
| 10 | 0 | 0 | 0 | 0 |

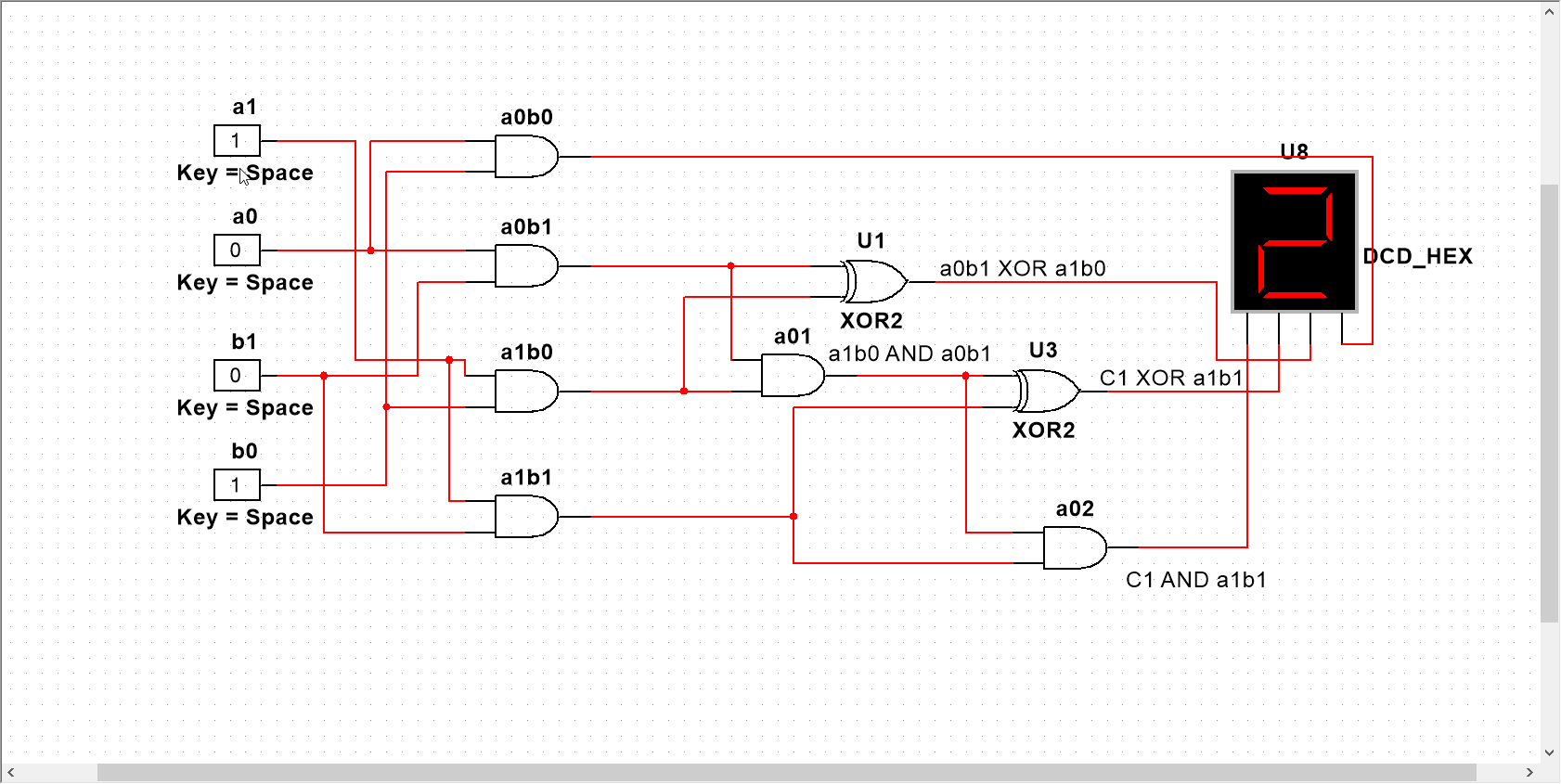
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 1 | 0 |
| 10 | 0 | 0 | 0 | 0 |

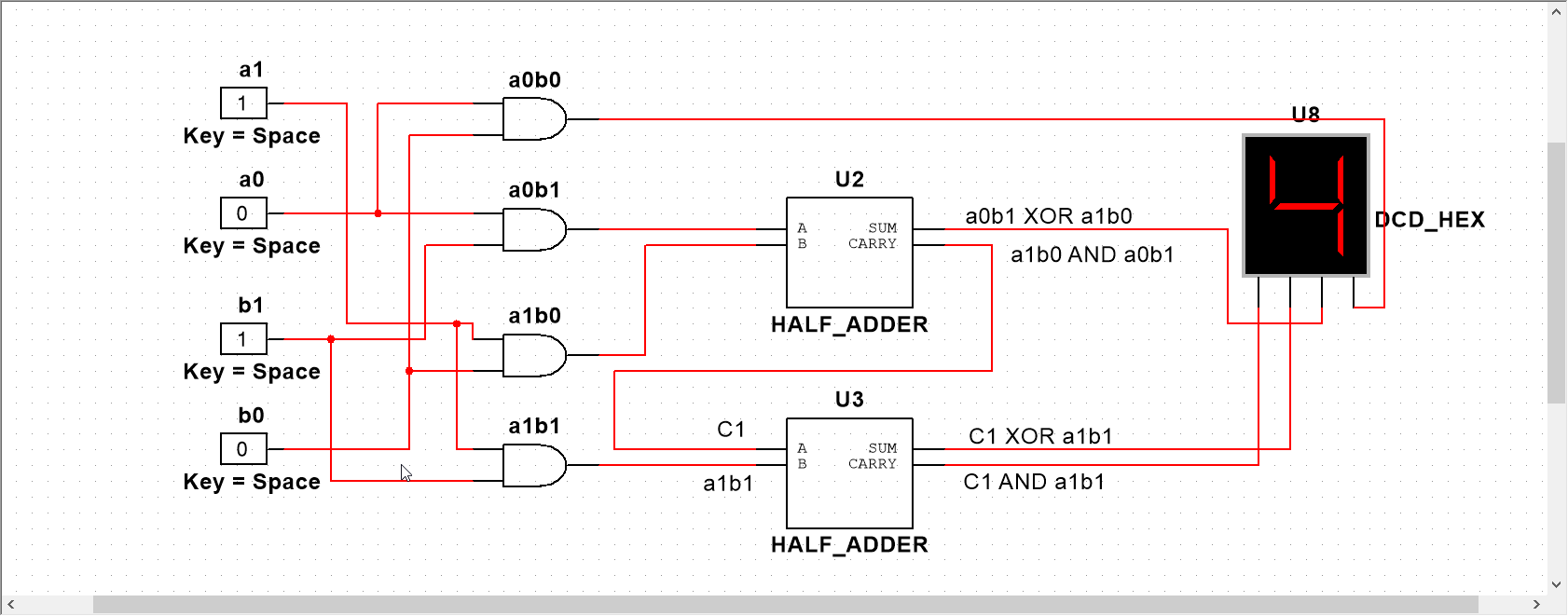
**P2 P3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 1 |
| 10 | 0 | 0 | 1 | 1 |

**4. Explain and Sketch the logic gates diagram of the 2-bits binary multiplier for**

**Structural Design. and compare it with the Behaviour Design.**

****

****

**5. Analyze the main differences between Structural Design and Behavior Design for 2-bits binary multiplier as shown in Figure 2**.

**Structural Design:**

**Focus:** Building the multiplier by directly implementing its internal hardware structure.

**Complexity**: Can be more complex for larger multipliers, requiring detailed understanding of gate-level logic and careful manual wiring.

**Method:** This involves manually defining the logic gates and their connections. You directly specify how each gate operates on the input bits to generate the partial products and final product.

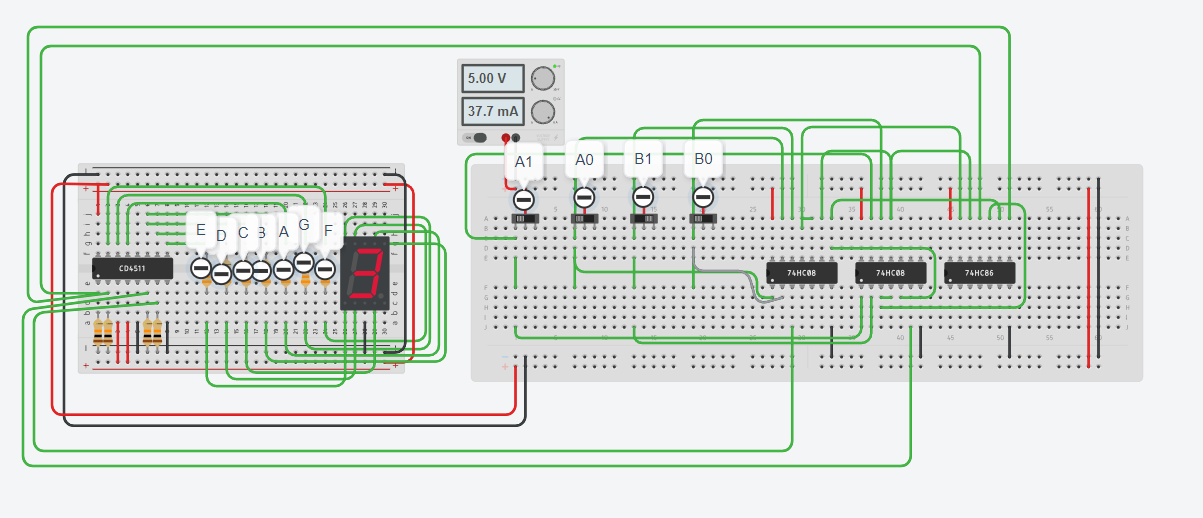
**Behavioral Design:**

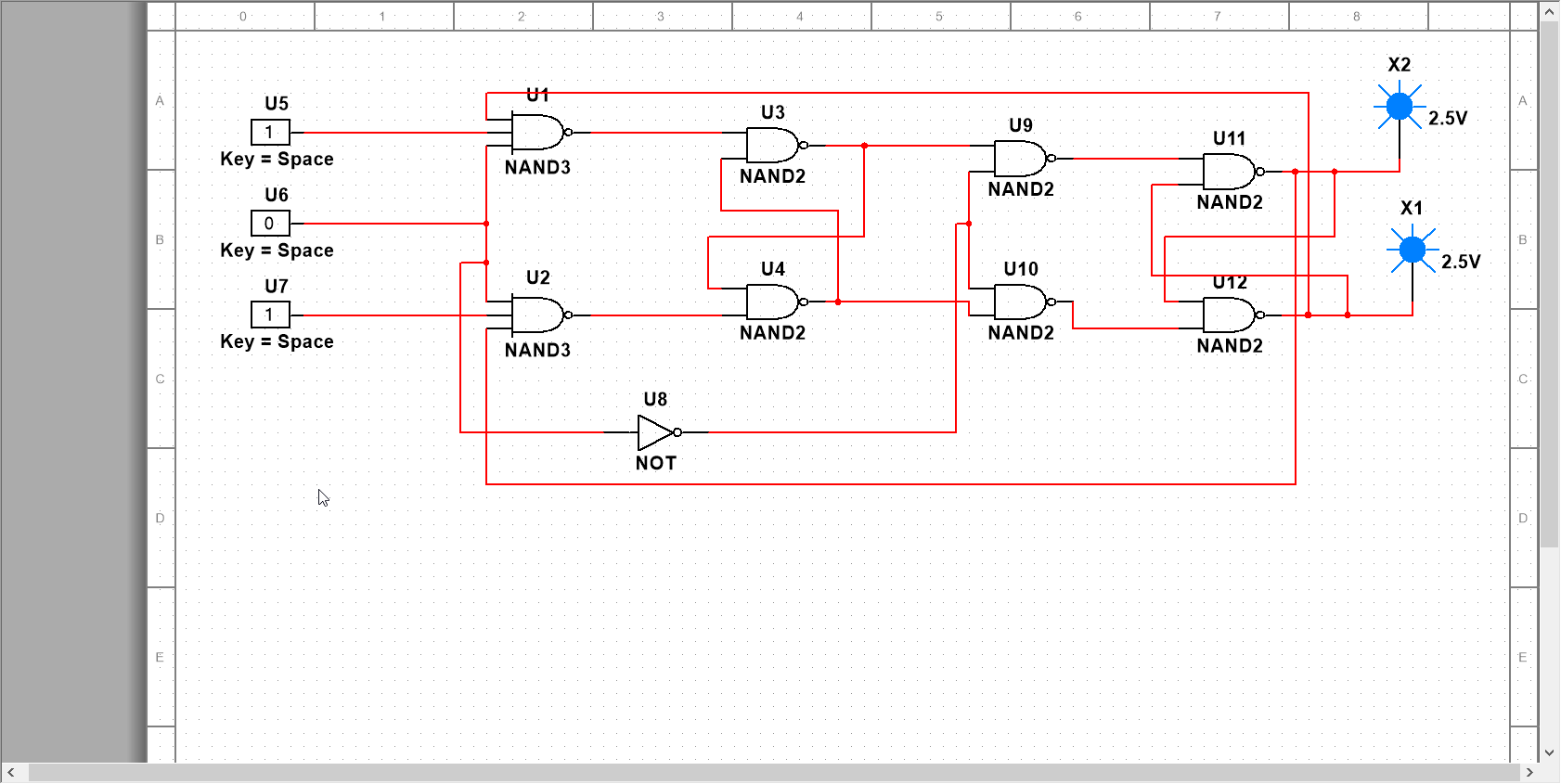
**Focus:** Describing the desired behavior of the multiplier without explicitly defining its hardware structure.

**Complexity:** Simpler to design, especially for large multipliers, as you focus on the algorithm rather than intricate gate-level wiring.

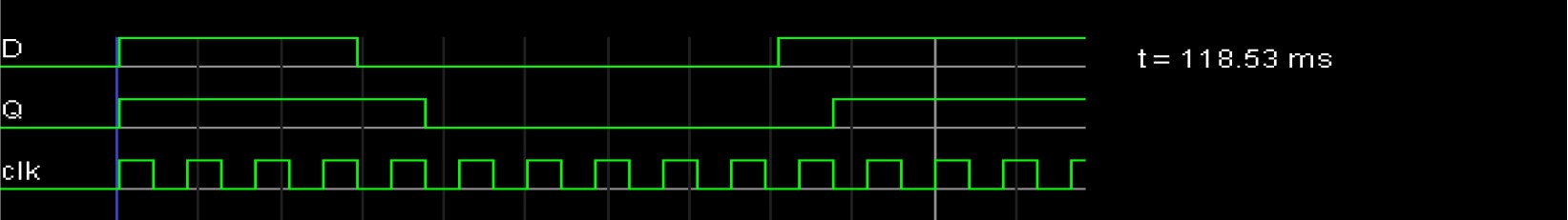
**Method:** This involves using a higher-level language like Verilog or VHDL to specify the multiplication algorithm. You define the steps and operations without necessarily specifying the exact gates and connections.

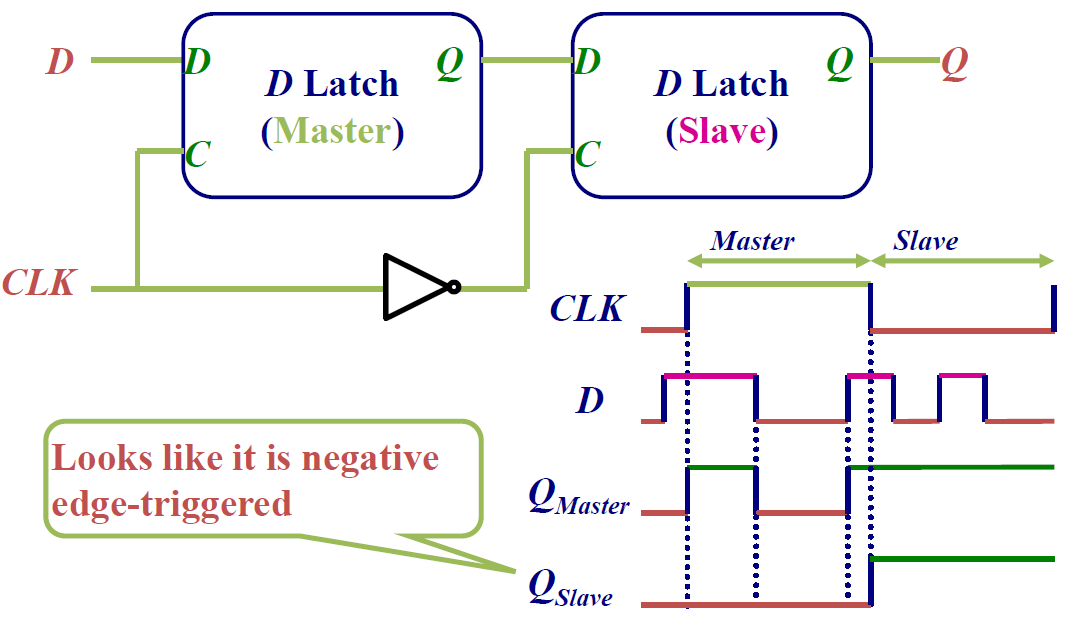
**6- Design a hardware implementation schematic and define the number of IC chips requirements for a Binary 2-bits binary multiplier with Binary to Decimal Converter to Drive Two Seven-Segment Indicators. The Four Inputs to the Converter Circuit (A, B, C, D) as in the Figure 3. Then, examine the suitable number of the Seven segment displays.**

****

**Task2**

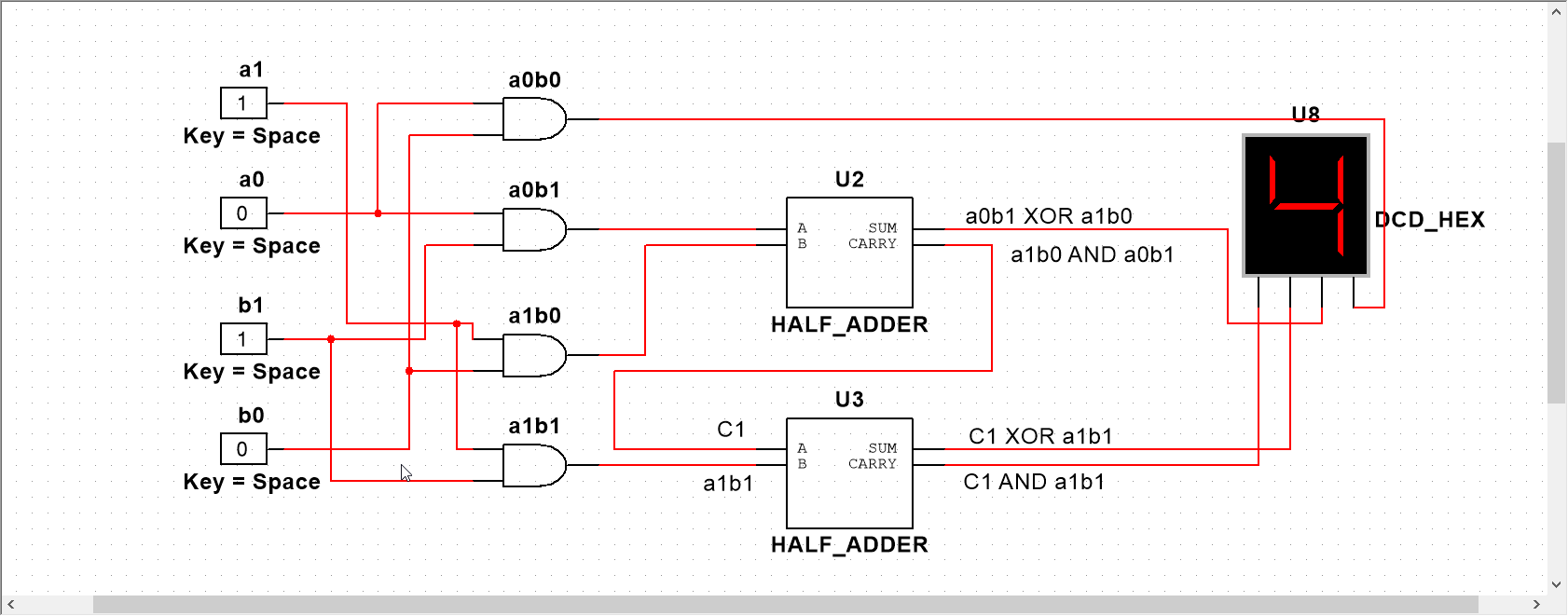
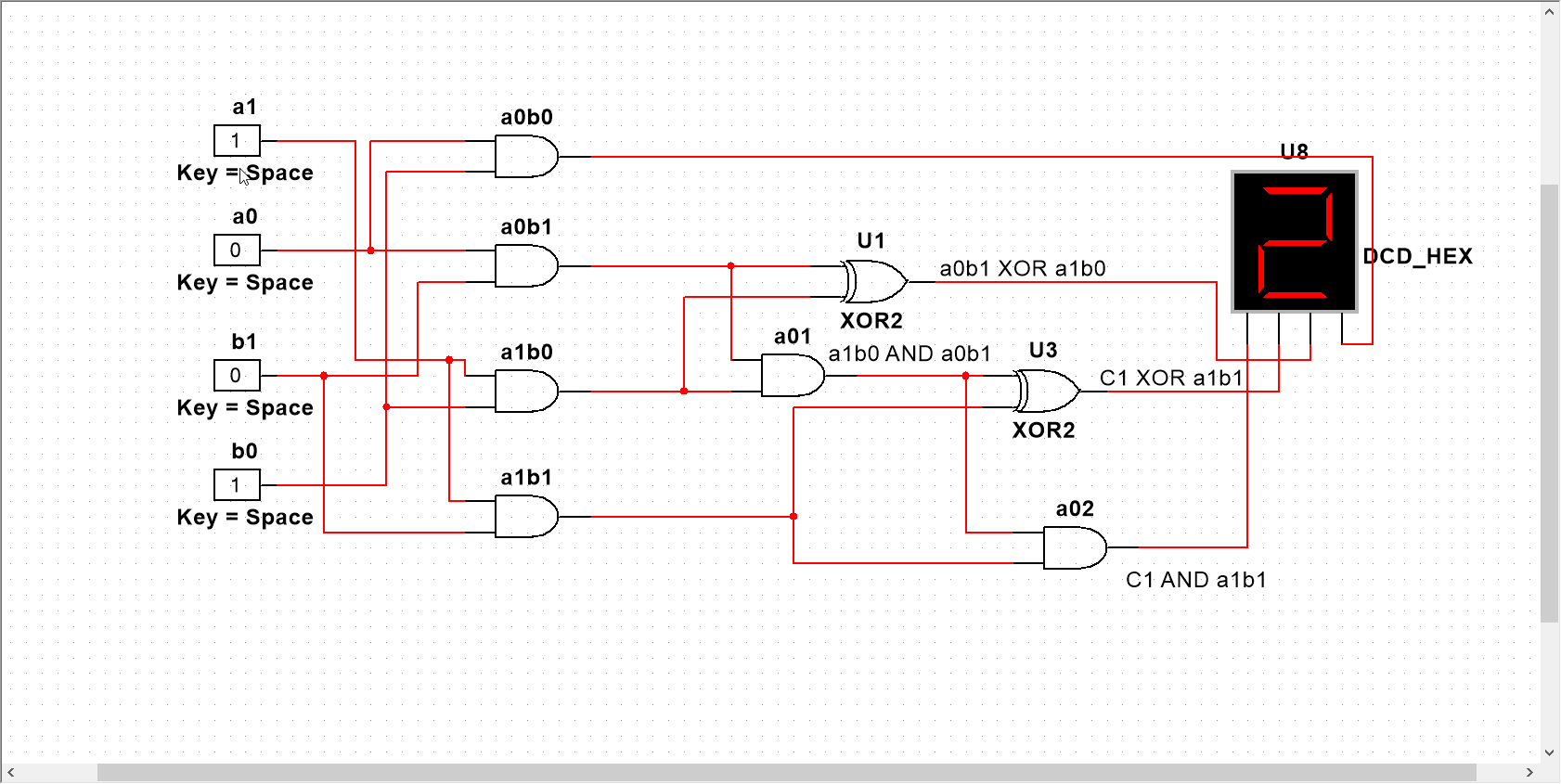
|  |
| --- |
| **1. From your exploration of the sequential logic systems, Explain the logic operation of two Master-Slave D Flip-Flop which is a sequential logic system with two stable states that can be used to store binary data. Analyze the system for logic gates, truth table, and Timing Diagram.** |



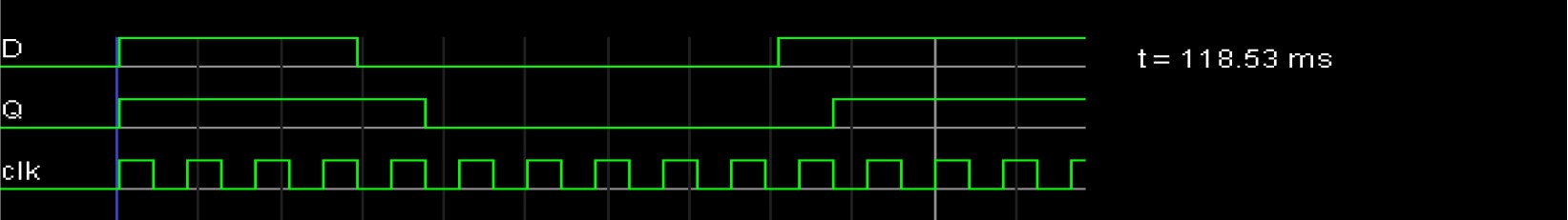
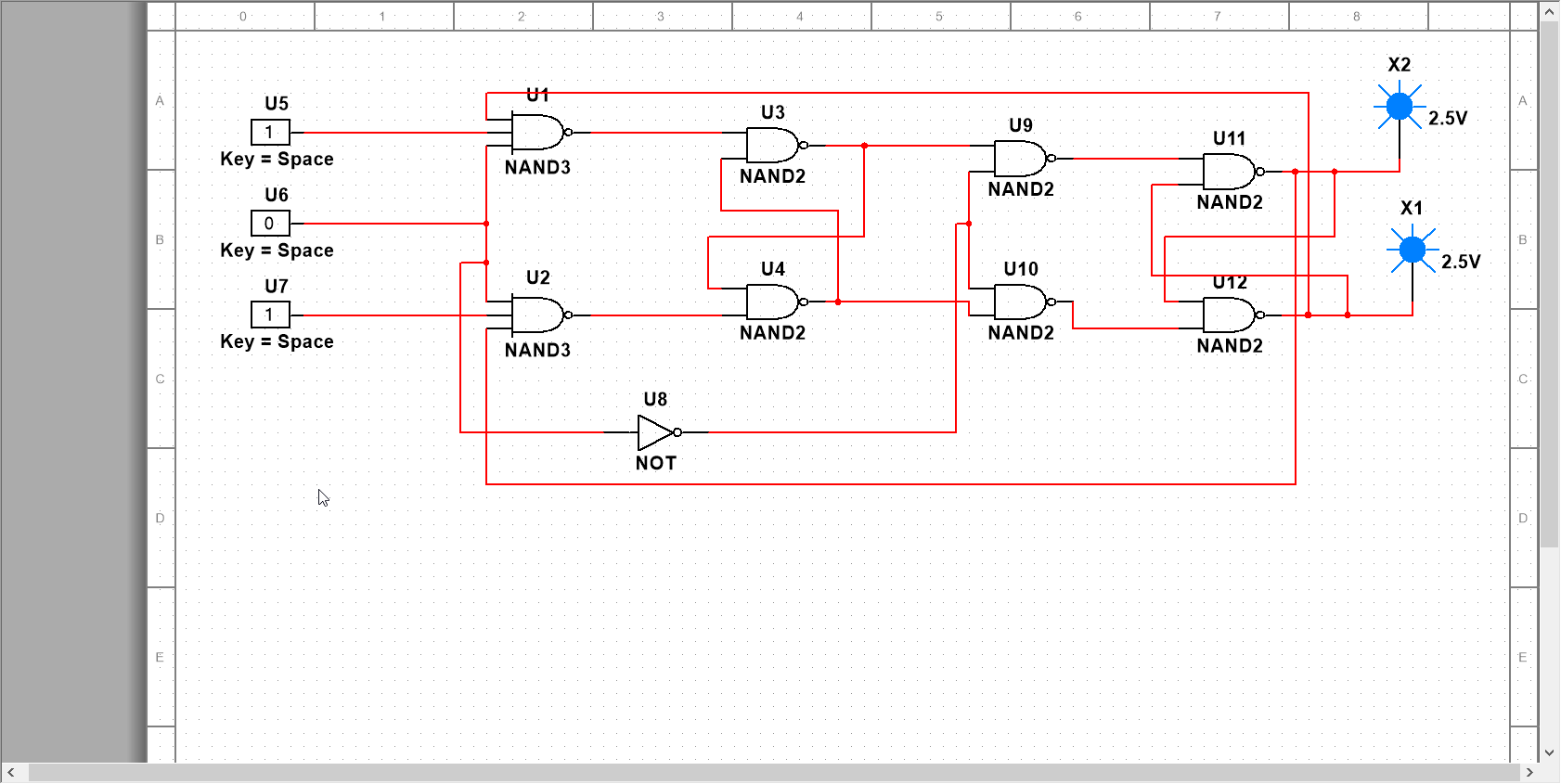


**Task 3**

1. **Use Multisim simulator to simulate the given combinational logic system and to test its operation.**

****

**Use Multisim simulator to simulate the Master-Slave D Flip-Flop and show its clock and Timing Diagram.**

****