Owen Shin

ECE 385 ABE

09/06/21

Lab 1

Introduction:

The purpose of this lab was to demonstrate static glitches in combinational logic and how they can be prevented. The main demonstration for this lab was a basic 2-to-1 multiplexer that was prone to experiencing a static-1 hazard. This brief glitch was observed through an oscilloscope. It was determined that the hazard, which was the result of a switch between two specific sets of inputs, was the result of gate delays which could be artificially extended. Redundant minterms were used as a solution to this issue. Additionally, the topics of propagation delays, debouncing, noise immunity, and use of LEDs were discussed.

Circuits:

Both logical circuits used in this lab accomplished the same task of a 2-to-1 multiplexer through combinational logic. The design begins as a basic boolean equation: Z = (AB) + (B’C). This expression resulted in the output Z matching the input A if B = 1 or matching the input C if B = 0. For the first part of the pre-lab, this expression was written with NAND gates. The term B’ was created by using a NAND gate as an inverter for B, and the terms A and B and B’ and C were connected by NAND gates, making the complement of the original minterms. These inverted minterms were connected by another NAND gate, which made the original expression. This was possible due to De Morgan’s laws, which states that an AND/OR expression is equivalent to the complement of the other operator (AND replacing OR and vice versa) applied to the complements of its original terms. In this case, both minterms were inverted, the OR was changed for a NAND gate, equivalent to the inverted outcome of an AND gate. Thus, the same expression was maintained using NAND gates.

The second version of the circuit included an extra minterm to prevent static-1 hazards. Though it is possible for the hazard to not occur due to small gate delays that don’t allow time for a glitch, it was necessary to account for this. These glitches occurred by means of gate delays that could briefly allow the last NAND gate to output an incorrect 0 when switching between ABC = 111 and ABC = 101. If the term A NAND B would change from 0 to 1 before the term B’ NAND C would change from 1 to 0, both terms would be high momentarily. This was made more possible since B’ NAND C required a sequence of two gates, while the other minterm only required one. This would result in the output becoming low for a short period of time, until the term B’ NAND C would become low.

(a) (b)



Figure 1: Oscilloscope screenshots of the (a) first circuit (b) second circuit. The top signal is a falling edge of the input B and the second signal is the output Z.

This glitch could be solved by adding a minterm AC such that B changing would not affect the outcome of A = 1 and C = 1. This redundancy resulted in the new equation: Z = (AB) + (B’C) + (AC). This was made into a circuit by inverting the output of the original circuit with a NAND gate, and using a NAND gate to obtain the complement of the other minterm AC. These two inverted terms as inputs to a NAND gate applied De Morgan’s laws, effectively using the original output and AC as terms for an OR expression. This resulted in the static-1 hazard becoming negligible (the output signal was slightly affected) and the logic remaining the same.

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | C | Z |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

(a) (b)

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | C | Z |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Figure 2: Truth table for the 2-to-1 multiplexer in (a) the first part of the pre-lab (b) the second part of the pre-lab. Both are identical and represent the logical expression (AB) + (B’C). The output Z matches the output A if B = 1 or the output C if B = 0.

(a) (b)

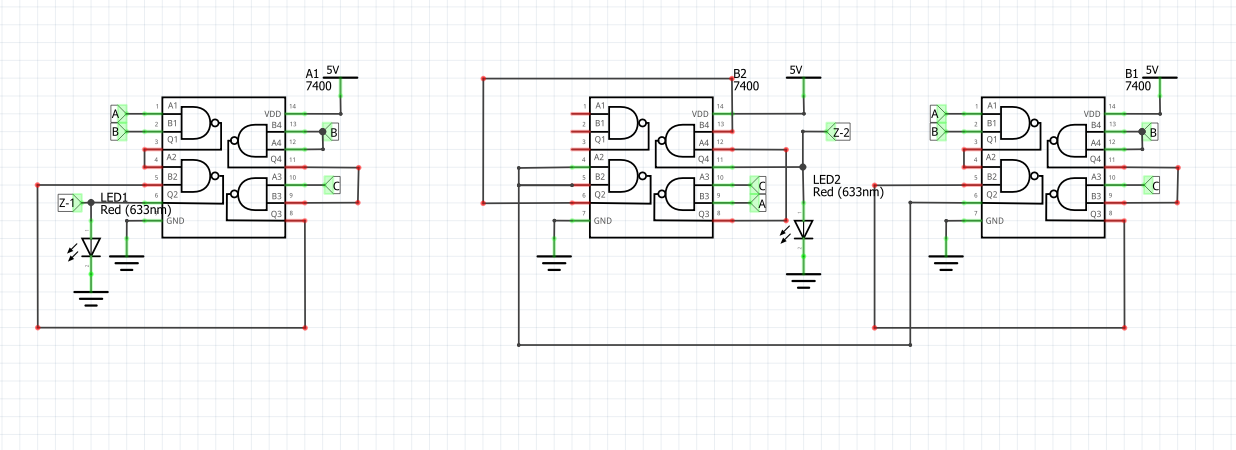


Figure 3: Schematics for (a) the first circuit (b) the second circuit. Both use NAND gates.

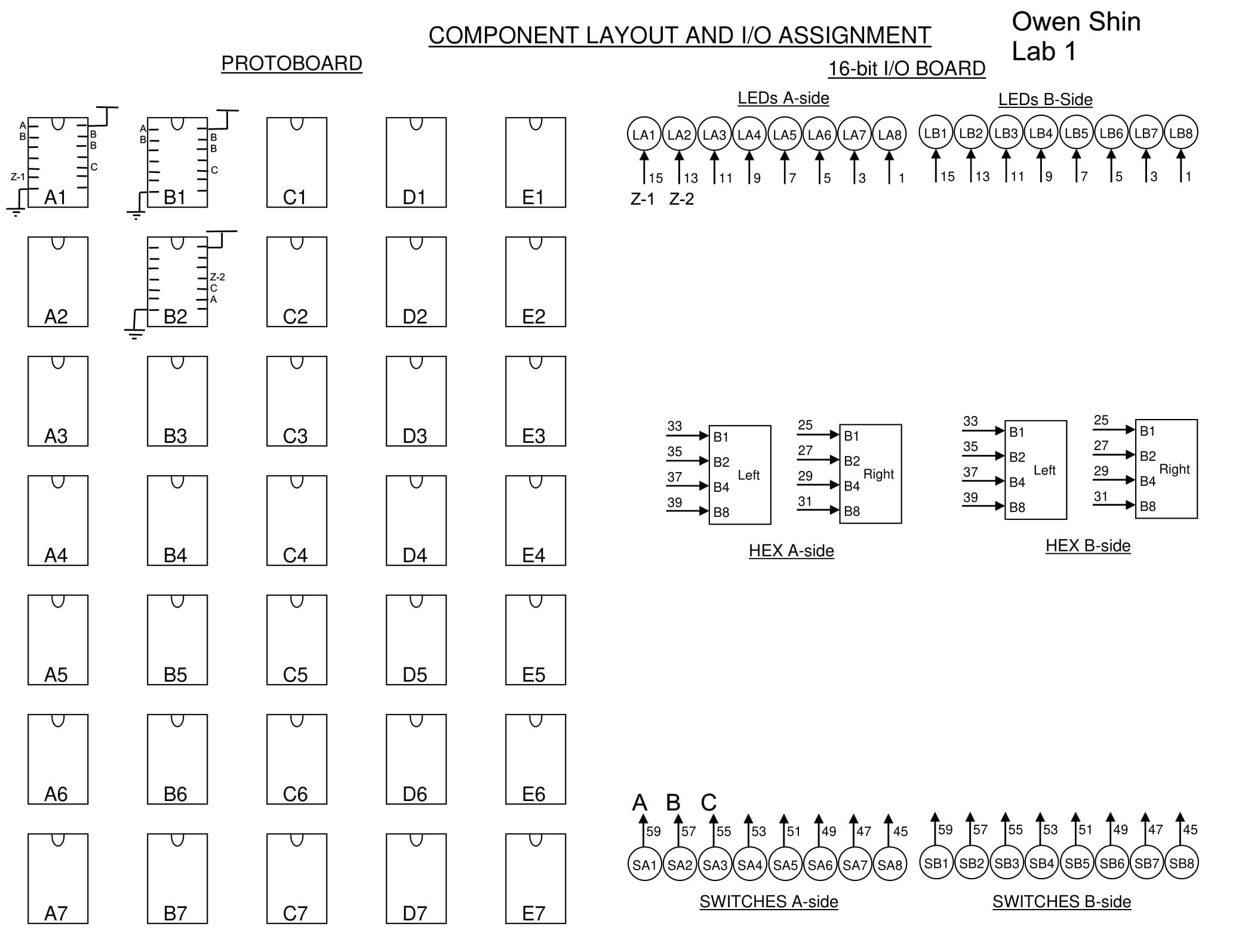


Figure 4: Component layout sheet for both circuits. The first circuit occupies location A1 and has output Z-1. The second circuit occupies locations B1 and B2 and has output Z-2. With the exception of the static-1 hazard, both circuits have the same behavior.

Post-lab:

An analysis of the output of the combinational logic shows that changes between the inputs ABC = 111 and ABC = 101 on either the falling or rising edge of B can create static glitches. If one of the inverted minterms becomes high before the other one becomes low, both inputs to the last NAND gate are high. This results in the static-1 hazard of the output being momentarily low when it is not meant to be. On the falling edge of B, the output Z can take up to 60 mS to stabilize due to the term that falls to low taking up to 40 mS to stabilize. On the rising edge of B, the falling term takes up to 20 mS to stabilize, meaning it can take up to 40 mS for Z to stabilize.

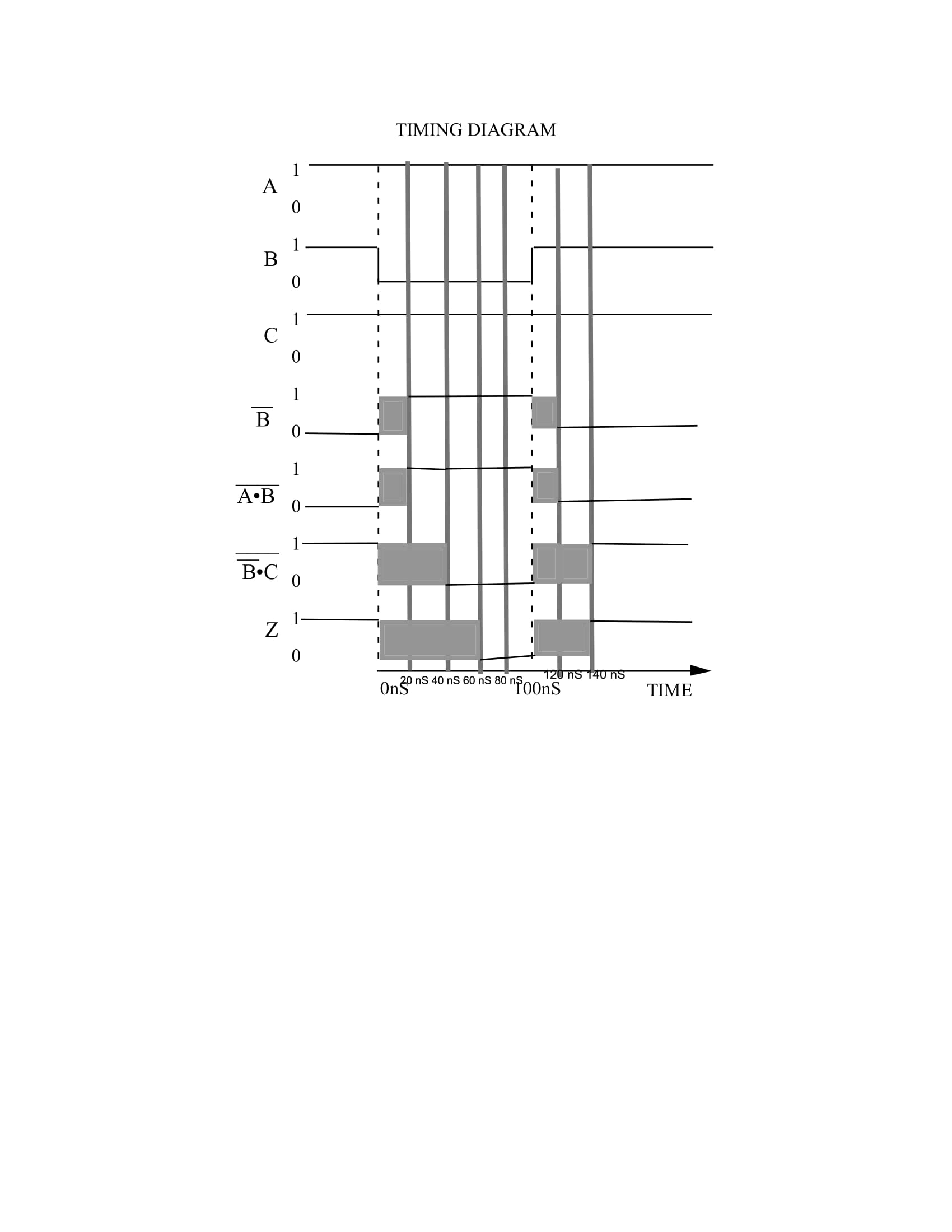


Figure 5: A timing diagram for the first version of the circuit. This was made with the maximum propagation delay for the NAND gate being 20 nS and the minimum delay being 0 nS.

The debouncing circuit provided in the general guide makes the SPDT switch behave like a single switch by pulling low one of the inputs to a NAND gate. Doing this ensures the other NAND gate has its floating input pulled high, and the prior NAND gate’s output rising ensures the other is low. This way, the state is maintained even if the switch floats, as the gates act as a latch. Changing the state of either NAND gate changes the state of the other, thus the output Q can be toggled with the switch contacting or either gate.