ECE 385

Spring 2024

Experiment # 1

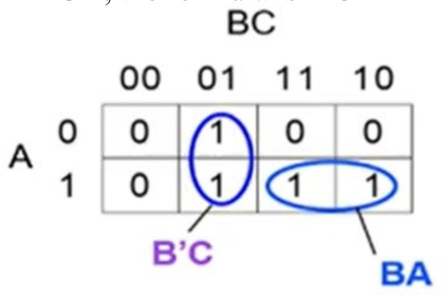
Introduction to TTL

Jie Wang 3200112404
Jan. 26, 2024, Friday D-225
TA: Jiebang Xia

Introduction

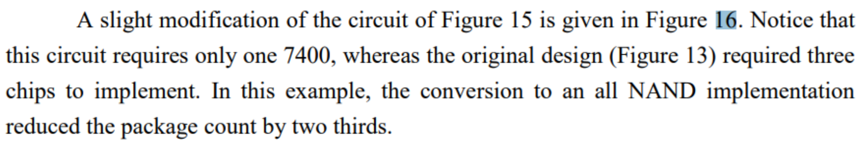
The objective of this lab is to let us get familiar with Quartus Prime + ModelSim Simulation. Building a simple 2-to-1 Multiplexer(MUX) with only NAND gates, we can visualize and understand the static hazards. Further, we get practice on avoiding it by adding a redundant component. The theoretical foundation was established through a Karnaugh Map (K-map) below:

For 2-to-1 MUX, the formula is $MUX = B'C + BA$



Implementation

Here is the NAND chip representation of the 2-to-1 MUX:



A slight modification of the circuit of Figure 15 is given in Figure 16. Notice that this circuit requires only one 7400, whereas the original design (Figure 13) required three chips to implement. In this example, the conversion to an all NAND implementation reduced the package count by two thirds.

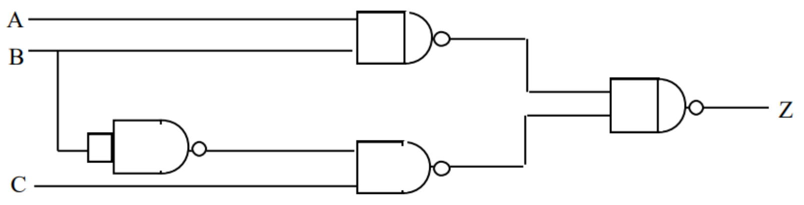
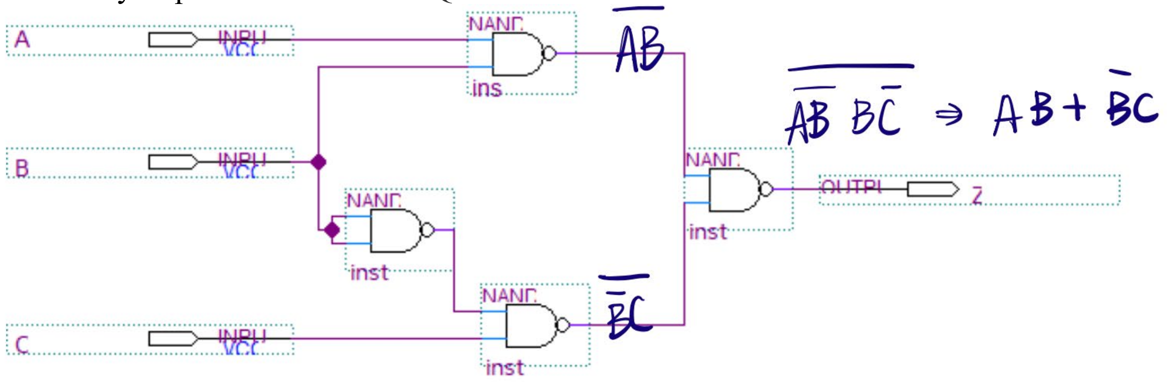


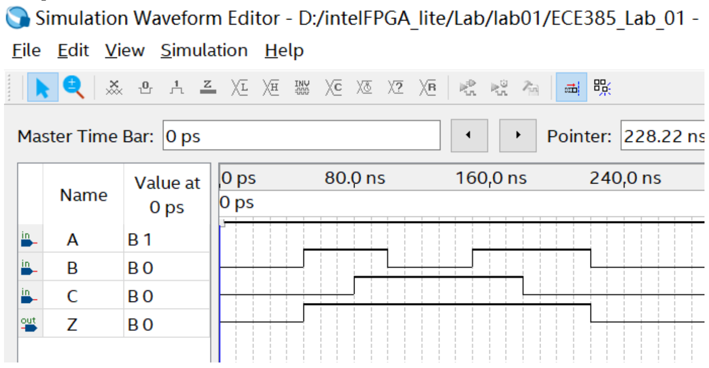
FIGURE 16. Single chip implementation using one 7400 Quad 2-input NANDs

Here is my implementation in the Quartus Prime:



A
B
C
INPU VCC
NANC
ins
inst
OUTPU
Z
$\overline{AB}$
$\overline{\overline{B}C}$
$\overline{\overline{AB}\ \overline{BC}} \Rightarrow AB + \overline{B}C$

My waveform:


Simulation Waveform Editor - D:/intelFPGA_lite/Lab/lab01/ECE385_Lab_01 -

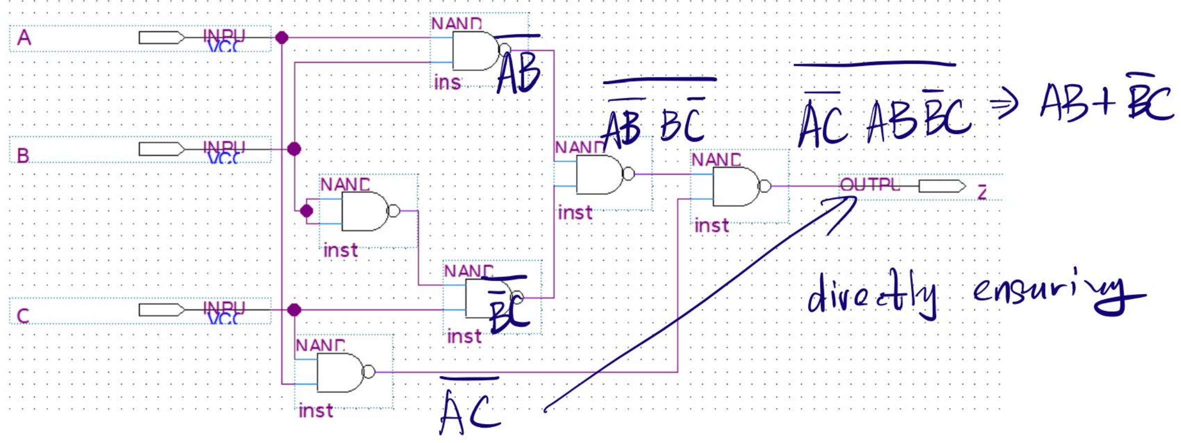
Master Time Bar: 0 ps   Pointer: 228.22 ns

Name	Value at 0 ps
A	B 1
B	B 0
C	B 0
Z	B 0

It can be clearly seen that my circuit runs exactly same as the formula.

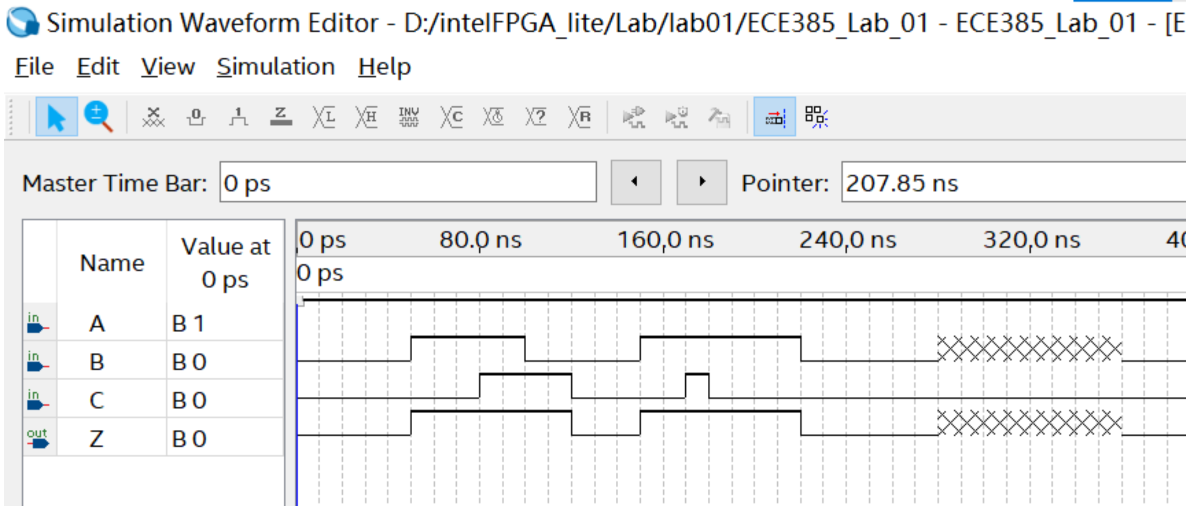
Redesign the circuit of part A

An improved circuit design was developed to address the static hazard caused by the delay inherent in B-Z's extra NAND gate. This redesign did not alter the fundamental operation of the circuit, as evidenced by the new waveform generated in the simulation. The revised design successfully maintained the expected output while eliminating the static hazard.


A
B
C
INPU
VCC
NAND
inst
OUTPU
Z
$\overline{AB}$
$\overline{\overline{AB}\,\overline{BC}}$
$\overline{\overline{AC}\,\overline{\overline{AB}\,\overline{BC}}} \Rightarrow AB + \overline{B}C$
$\overline{BC}$
$\overline{AC}$
directly ensuring

New waveform:

It behaves the same way as last circuit as it is simulation. Still, my circuit works as expected.


Simulation Waveform Editor - D:/intelFPGA_lite/Lab/lab01/ECE385_Lab_01 - ECE385_Lab_01 - [E

Master Time Bar: 0 ps   Pointer: 207.85 ns

Name	Value at 0 ps
A	B 1
B	B 0
C	B 0
Z	B 0

Truth table of circuits

The center column separates the truth tables for the original circuit (Part A) and the modified circuit (Part B). In simulations, both circuits will output the same result since ModelSim treats all TTL components as ideal with zero delay. However, due to propagation delays in actual devices, the output will exhibit a temporary "0" when A and C are "1" and B transitions from "1" to "0" in the original circuit. This static hazard is addressed in the modified circuit, preventing the glitch from occurring.

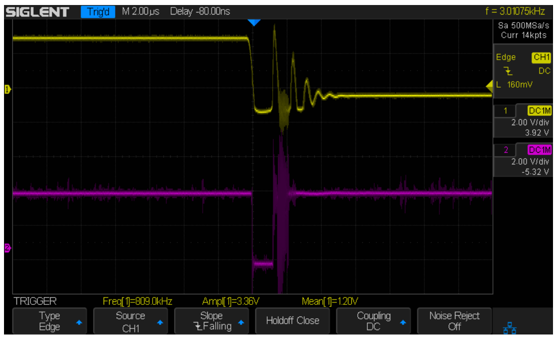
Truth Table for Circuit Part A (without AC term) and Part B (with AC term)

A	B	C	Z = AB + BC	A	B	C	Z = AB + BC + AC
0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	1
0	1	0	0	0	1	0	0
0	1	1	1	0	1	1	1
1	0	0	0	1	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Answers to Lab Questions

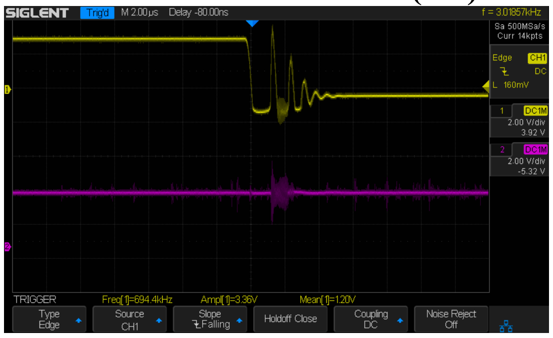
In the lab, we have two images provided by Prof. Chen about the glitch effect observed using oscilloscope.

- **2-1-mux naïve:**



This setup demonstrated the presence of glitches due to static hazards.

- **2-1-mux redundant term (AC)**



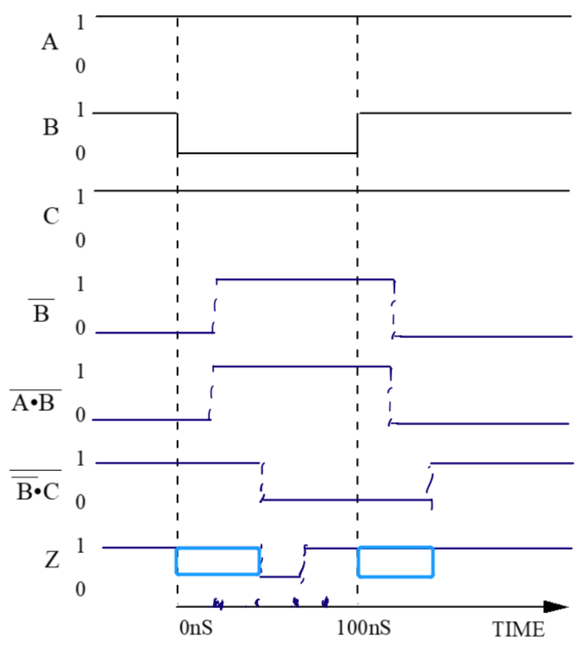
the redundant term effectively mitigated the glitch observed in the naïve implementation.

Analysis

The naive approach cause a subsequent glitch as the signal drops for 1 to 0 this is caused by the delay within circuit To suppress this glitch reducing the static hazards,we involves a new redundant term avoiding this happens As for the noise it's hard to handle we may include filter or trigger to reduce.

Answers to Post-Lab Questions

1. Given that the guaranteed minimum propagation delay of a 7400 is 0ns and that its guaranteed maximum delay time is 20ns, complete the timing diagram below for the circuit of part A. (See GG.23 if you are not sure how to proceed.)



In summary, for the circuit of part A:

- **Falling edge of B**: Z stabilizes at most after 40 ns.
- **Rising edge of B**: Z stabilizes at most after 40 ns.

2. How long does it take the output Z to stabilize on the falling edge of B (in ns)? How long does it take on the rising edge (in ns)? Are there any potential glitches in the output, Z? If so, explain what makes these glitches occur.

For the **falling edge of B**:

- When B goes from 1 to 0, both inputs to the NAND gate forming the A•B term will be at logic level 1 briefly, due to the propagation delay in the gate that takes B as an input. This causes the A•B output to go to logic level 1 temporarily.
- Simultaneously, the B•C output will also go to logic level 1 as both B and C are at logic level 0 momentarily because of the delay.
- The output Z, which is an OR function of A•B and B•C, will thus see both inputs at logic level 1 briefly, causing Z to go to logic level 0.
- This glitch will last until the NAND gate outputs stabilize to their correct values after the propagation delay. So, Z will stabilize after at most 20ns, which is the maximum delay of the NAND gate.

For the **rising edge of B**:

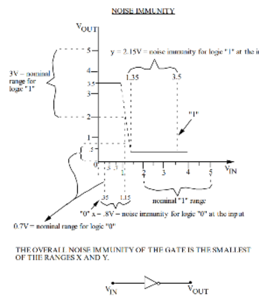
- When B goes from 0 to 1, the A•B term will not immediately reflect this change due to the propagation delay. If A is at logic level 1, the A•B output will temporarily be at logic level 0.
- The B•C term will similarly not immediately reflect the change, causing a temporary logic level 0 if C is at logic level 1.
- Z will stabilize to logic level 1 after the NAND gates have propagated the change, which again takes at most 20ns.

Potential Glitches in Output Z:

- Glitches in the output Z occur due to the propagation delays of the gates. When B changes state, there's a brief period where the old values of A and C are still 'seen' by the NAND gates due to the delay.
- On the falling edge of B, if A and C were previously at logic level 1, Z may glitch to 0 before stabilizing at 1 due to the temporary outputs of the NAND gates.
- On the rising edge of B, a similar situation can occur depending on the values of A and C. If either A or C is at logic level 1, Z may temporarily go to 0 before stabilizing at 1.

General Guide Questions

1. **(GG.6)** What is the advantage of a larger noise immunity? Why is the last inverter observed rather than simply the first? Given a graph of output voltage (VOUT) vs. input voltage (VIN) for an inverter, how would you calculate the noise immunity for the inverter? See the following figure.



Answer:

- The noise immunity of a gate indicates its ability to **withstand voltage variations due to noise without altering its logic state.** A gate with higher noise immunity is **more robust** against electrical noise, ensuring more reliable performance in noisy environments.
- To calculate the noise immunity from the V_OUT vs. V_IN graph for an inverter, assess the voltage ranges where the output remains stable. For the high-level (or positive) noise immunity, find the difference between the median of the nominal "1" input range (3.5V) and the V_IN at which V_OUT starts to drop towards "0" (1.35V), which is 2.15V. For the low-level (or negative) noise immunity, calculate the difference between the median of the nominal "0" input range (0.35V) and the V_IN at which V_OUT begins to rise towards "1" (0.95V), resulting in **0.6V.**

2. **(GG.29)** If we have two or more LEDs to monitor several signals, why is it bad practice to share resistors?

Answer:

Because each LED requires a specific amount of current to operate correctly. Sharing a resistor does not guarantee that each LED will receive the correct current, especially if the forward voltage drops across the LEDs are not identical, which may cause some LEDs being dimmer or not lighting up at all.

Conclusions

Lab1 is an effective refresher of foundational concepts from ECE110, ECE120, and ECE220, with a specific focus on Quartus Prime configuration. The lab emphasized the practical aspects of digital system design, particularly in identifying and resolving static hazards in a 2-to-1 MUX configuration. The hands-on experience with Quartus Prime and ModelSim simulations not only reinforced theoretical knowledge but also highlighted the significance of meticulous design and simulation in digital systems engineering.