**ELECTRICAL & COMPUTER ENGINEERING**

School of Engineering

**EGRE 365 – Digital Systems**

**Laboratory No. 4**

**Name: Luis Barquero**

**Major: Computer Engineering**

**Date: 10/05/17**

**Honor Pledge:** *I have neither given nor received any unauthorized help on this lab. Signed:*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**2) Component Description**

The purpose of this lab was to assume that the 8-bit adder from example 9 was to be used in an ALU for a new processor, and determine the worst-case delay of the adder. In order to determine the worst-case delay, the example 9 8-bit VHDL model alongside its testbench had to be modified with delays in order to measure the delay.

**3) Implementation**

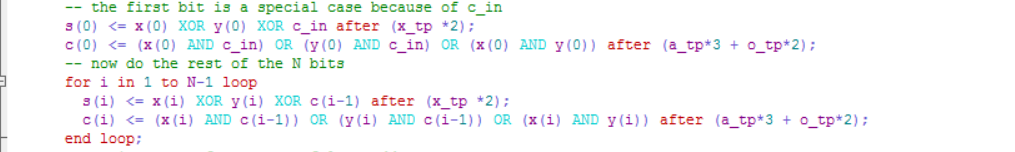
First, the addition of the following delays was added to the example 8-bit adder, as shown in Table 1:

***Table 1 – Table 1 shows the delays that were implemented in the example 9 model.***

|  |  |
| --- | --- |
| Logical Operation | Average Delay |
| NOT(Inversion) | 3 ns |
| AND | 4 ns |
| OR | 5 ns |
| XOR | 7 ns |

The delays were added in the following manner, as shown in Figure 1:

***Figure 1 – Figure 1 shows the implementation of the delays from Table 1 into the example 9 mode.***



From this, an equation can be used to determine which set of operations produces the most delays.

For the operations is s(0), there are two XOR operations, and since the XOR operation has a 7 ns delay, the total amount of delay is 14 ns. However, this is only for one bit, so to get the total for the 8 bits, the 14 ns is multiplied with 8, for a total of 112 ns.

For c(0), there are a total of three AND operations and two OR operations, and since the AND and OR operations produce a 4 ns and 5 ns delay respectively, the total amount of delay is 22, or 3(4) + 5(2) → 12 + 10 = 22. Once again, this is only for one bit, so to get the total for the 8-bits, 22 is multiplied with 8, for a total of 176 ns.

From this, it is evident that the maximum delay produced will be 176 ns, and from this, the maximum clock speed of the processor can be determined by using the following formula:

Using 176 ns for T, the maximum clock speed will be:

In the testbench, the new delay was modified from the 20 ns to 200 ns, to allow enough time for the outputs of the adder to settle to the correct values.

Next, the goal was changed to be able to run the processor at a clock speed of 20 MHz

In order to achieve this, the max delay had to be determined using the same equation from above, except this time T is the variable to solve for.

From there, the delays for each logical operation was determined. Given that the 50 ns is for the 8-bits, the delays has to be divided by 8 in order to determine the delays for each individual bit. Diving produces a delay of 6.25 for each bit.

Given that the logical operations that affects the delay the most are the AND and OR operations, and since there are three AND and two OR operations, the following equation was used to determine the individual delays:

Since the delay can only take integers, x and y were set to 1, meaning that the new delays for the AND and OR operations were both 1 ns. Subtracting 6.25 – 5 left , but since the delay has to be an integer, the 1.25 was rounded to 2 ns for a better estimation. Table 2 displays the new delays for each one of the logical operations.

***Table 2 – Table 2 displays the calculated delays for each logical operation.***

|  |  |
| --- | --- |
| Logical Operation | Average Delay |
| NOT(Inversion) | 3 ns |
| AND | 1 ns |
| OR | 1 ns |
| XOR | 2 ns |

After calculating the new delays, the old delays from the example 9 model were changed to the calculated delays, and the maximum delay of 200 ns from the testbench was modified to 55 ns, to allow enough time for the outputs of the adder to settle to the correct values.

**4) VHDL Code**

See Appendix A.

**5) Tests**

In order to verify that the calculated max delay worked as expected, the following six cases were constructed, as shown in Table 2:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case | x | y | s | c\_in | Delay(ns) |
| 1 | "00000000" | "00000000" | "00000000" | 0 | 200 |
| 2 | "01101011" | "10010100" | "00000000" | 1 | 176 |
| 3 | "11101110" | "00001000" | "11110110" | 0 | 80 |
| 4 | "00101001" | "00100101" | "01001111" | 1 | 36 |
| 5 | "11111111" | "00000001" | "00000001" | 1 | 156 |
| 6 | "11111111" | "11111111" | "11111111" | 1 | 200 |

***Table 2 – Table shows the six cases use for determining the max delay from example 9 testbench.***

As shown in the simulation waveforms from Appendix B and Table 2, the highest delay occurs during case 2, which produces a delay of 176 ns. The first and last case produces a 200 ns delay, but that is because they were the input delays from the testbench, which was set to 200 ns.

For the 200 MHz clock, the new max and individual delays were set, and the following six cases were constructed to verify that the processor works as expected:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case | x | y | s | c\_in | Delay(ns) |
| 1 | "00000000" | "00000000" | "00000000" | 0 | 55 |
| 2 | "01101011" | "10010100" | "00000000" | 1 | 40 |
| 3 | "11101110" | "00001000" | "11110110" | 0 | 36 |
| 4 | "00101001" | "00100101" | "01001111" | 1 | 22 |
| 5 | "11111111" | "00000001" | "00000001" | 1 | 36 |
| 6 | "11111111" | "11111111" | "11111111" | 1 | 55 |

Since the max delay acquired is 40, excluding the first and last delays, the processor works as expected, since it is within the bounds of 50 ns.

**6) Simulation Waveforms**

See Appendix B.

**7) Problems Encountered**

The main problem encountered were determining the delay, since it was a lot of trial and error in calculating the 176 ns delay. It wasn’t until we figured out that the AND and OR operations produces the most delay that we resolved this problem.

Appendix A

VHDL Code

Appendix B

Simulation Output Waveforms