EE/CMPEN 417 – Field Programmable Devices

Lab 1 – Spring 2024

By: Fatma Alwasmi  
and Pauka Sy

# Description of Lab

In this lab, we created a series of different adder designs using the different techniques discussed in class and used static timing analysis to view the relative performance of each design. There are 5 parts to the lab and the extra credit.

In part 0, we will create a 1-bit Ripple Carry Full adder. Then, in part 1, we will create a variable bit-width ripple carry adder based on part 0. After that, we will make a 128-bit RCA with pipelining. We will insert the pipelining registers at every 16-bits and then instantiate eight 16-bit adders based on part 1’s design. Then, insert registers at every 16-bit. Last in part 3, we will create a variable bit-width behavioral adder. For each part we will have to simulate the design and take screenshots of the worst negative slack and the paths. Then in part 4, we will answer the questions that are asked.

# Calculating the Frequency

Part 1 128-bit max running frequency

Part 2 128-bit max running frequency

Part 3 128-bit max running frequency

Part 5 max running frequency

# Comparing Part 1, Part 2, Part 3

In Part 1, the ripple carry adder (RCA) operated at its lowest frequency. However, with the introduction of pipelining in Part 2, we observed a significant increase in frequency, enabling a higher rate of computations. This was due to the ability to execute multiple operations in parallel, rather than sequentially waiting for one operation to complete before initiating the next. The highest frequency was achieved in Part 3, because of the behavioral implementation that further optimized processing efficiency.

The utilization of a pipelined Ripple Carry Adder (RCA) in part 2 improved both throughput and frequency, as it allowed for an increased number of calculations within the same timeframe. This enabled concurrent operation execution, thus speeding up the computational process. Despite these benefits, the introduction of pipelining also resulted in increased latency. In Part 2, there was an 8-clock cycle delay between each input, representing a notable increase in wait time compared to Part 1. This trade-off shows the balance between achieving higher processing speeds and managing the latency introduced by advanced pipelining techniques.

# Question

In the same written report that you’ve been using to report the maximum clock frequency the other designs could run at, answer the following question: How could you make a 256-bit adder run at (or very near to) the same frequency as a 32-bit adder? Consider any impact on latency as well.

To achieve the same or similar frequency of a 32-bit adder for a 256-bit adder, we design a pipelined 256 adder, we can break down the design into stages, such as instantiating a 16 16-bit adder with pipelined registers every 16-bits. This will make the frequency close to the 32-bit adder. However, with this design, frequency and throughput might be higher but the tradeoff will be the latency. The latency will get worse with pipelining.

# Screenshots From the Lab

A screenshot of a computer

Description automatically generated

Figure 1: Part 1 16-bit WNS

A screenshot of a computer

Description automatically generated

Figure 2: Part 1 16-bit Waveform

A screenshot of a computer

Description automatically generated

Figure 3: Part 1 16-bit Paths

A screenshot of a computer

Description automatically generated

Figure 4: part 1 32-bit WNS

A screenshot of a computer

Description automatically generated

Figure 5: part 1 32-bit Waveform

A screenshot of a computer

Description automatically generated

Figure 6: part 1 32-bit Paths

A screenshot of a computer

Description automatically generated

Figure 7: Part 1 64-bit WNS

A screenshot of a computer

Description automatically generated

Figure 8: Part 1 64-bit Waveform

A screenshot of a computer

Description automatically generated

Figure 9: Part 1 64-bit Paths

A screenshot of a computer

Description automatically generated

Figure 10: part1 128-bit WNS

A screenshot of a computer

Description automatically generated

Figure 11: Part1 128-bit Waveform

A screenshot of a computer

Description automatically generated

Figure 12: Part 1 128 Paths

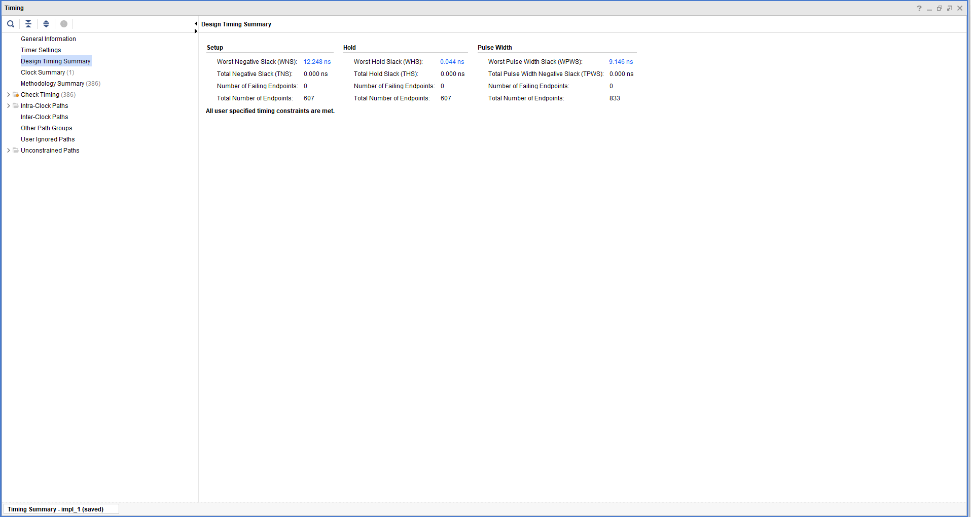


Figure 13: part 2 WNS

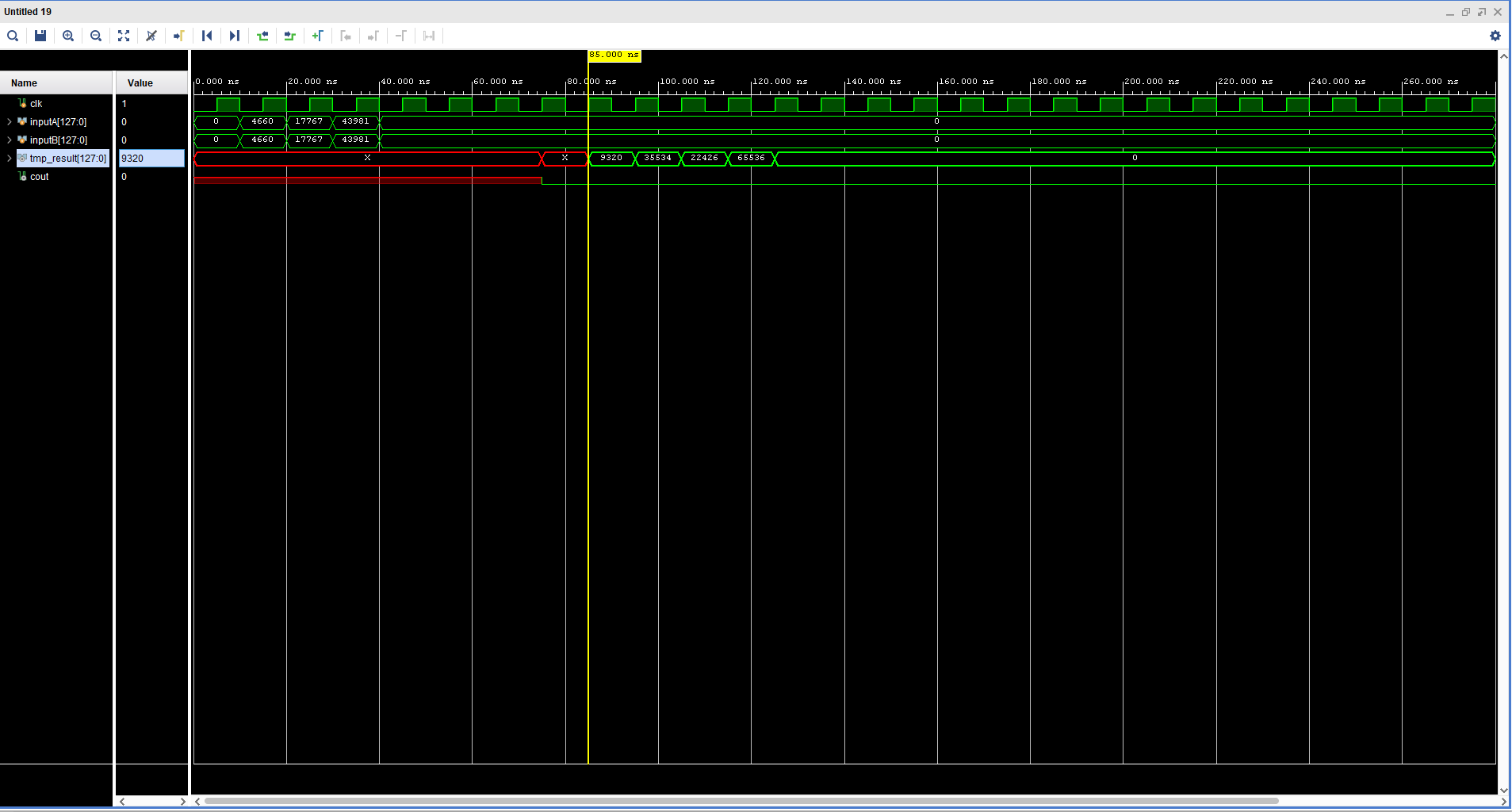


Figure 14: Part 2 Waveform

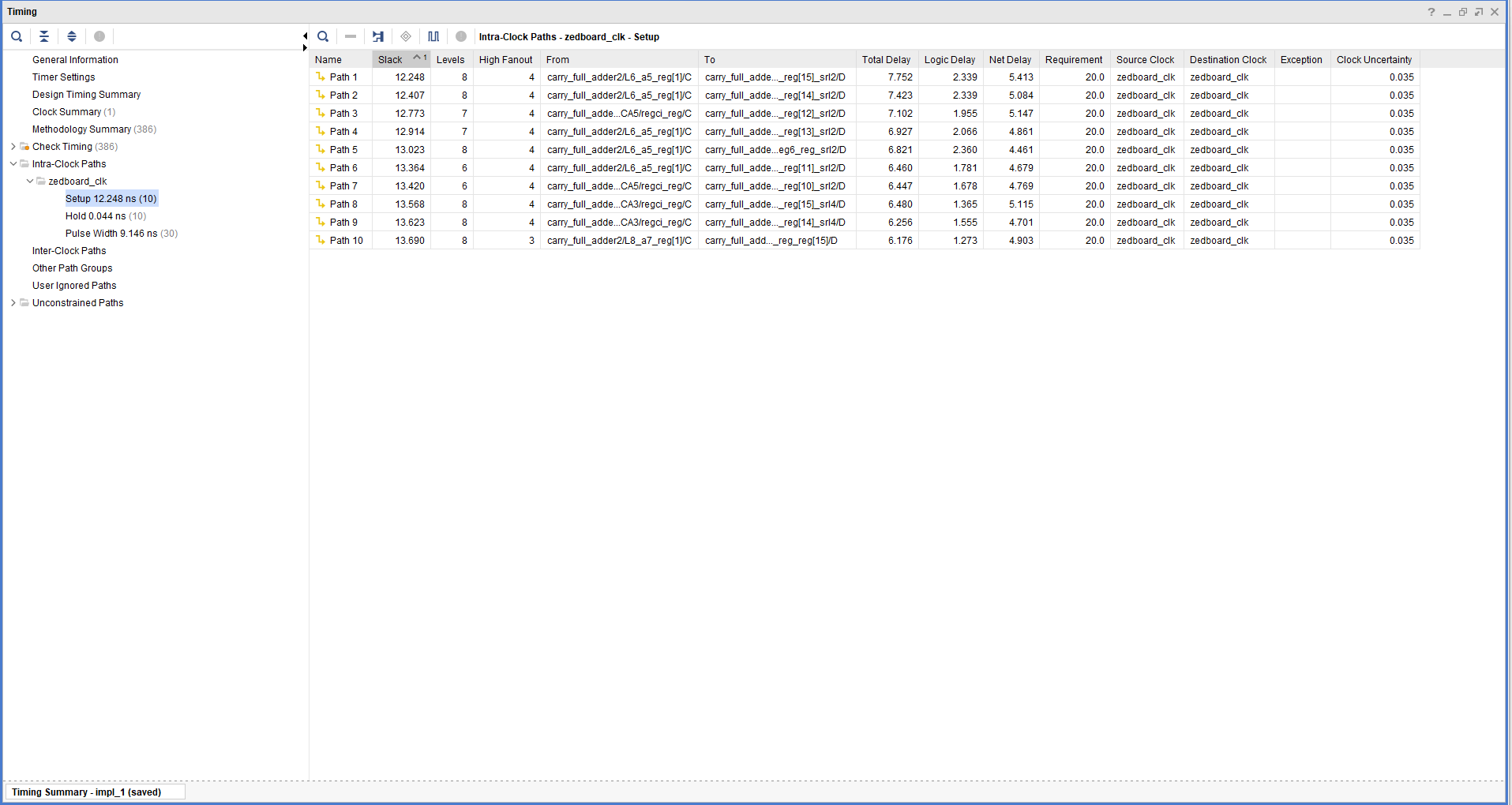


Figure 15: Part 2 Paths

A screenshot of a computer

Description automatically generated

Figure 16: Part 3 16-bit WNS

A screenshot of a computer

Description automatically generated

Figure 17: Part 3 16-bit Paths

A screenshot of a computer

Description automatically generated

Figure 18: Part 3 16-bit Waveform

A screenshot of a computer

Description automatically generated

Figure 19: Part 3 32-bit WNS

A screenshot of a computer

Description automatically generated

Figure 20:: Part 3 32-bit Waveform

A screenshot of a computer

Description automatically generated

Figure 21: Part 3 32-bit Paths

A screenshot of a computer

Description automatically generated

Figure 22: Part 3 64-bit WNS



Figure 23: Part 3 64-bit Waveform

A screenshot of a computer

Description automatically generated

Figure 24: Part 3 64-bit Paths

A screenshot of a computer

Description automatically generated

Figure 25: Part 3 128-bit WNS

A screenshot of a computer

Description automatically generated

Figure 26: Part 3 128-bit Waveform

A screenshot of a computer

Description automatically generated

Figure 27: Part 3 128-bit Paths

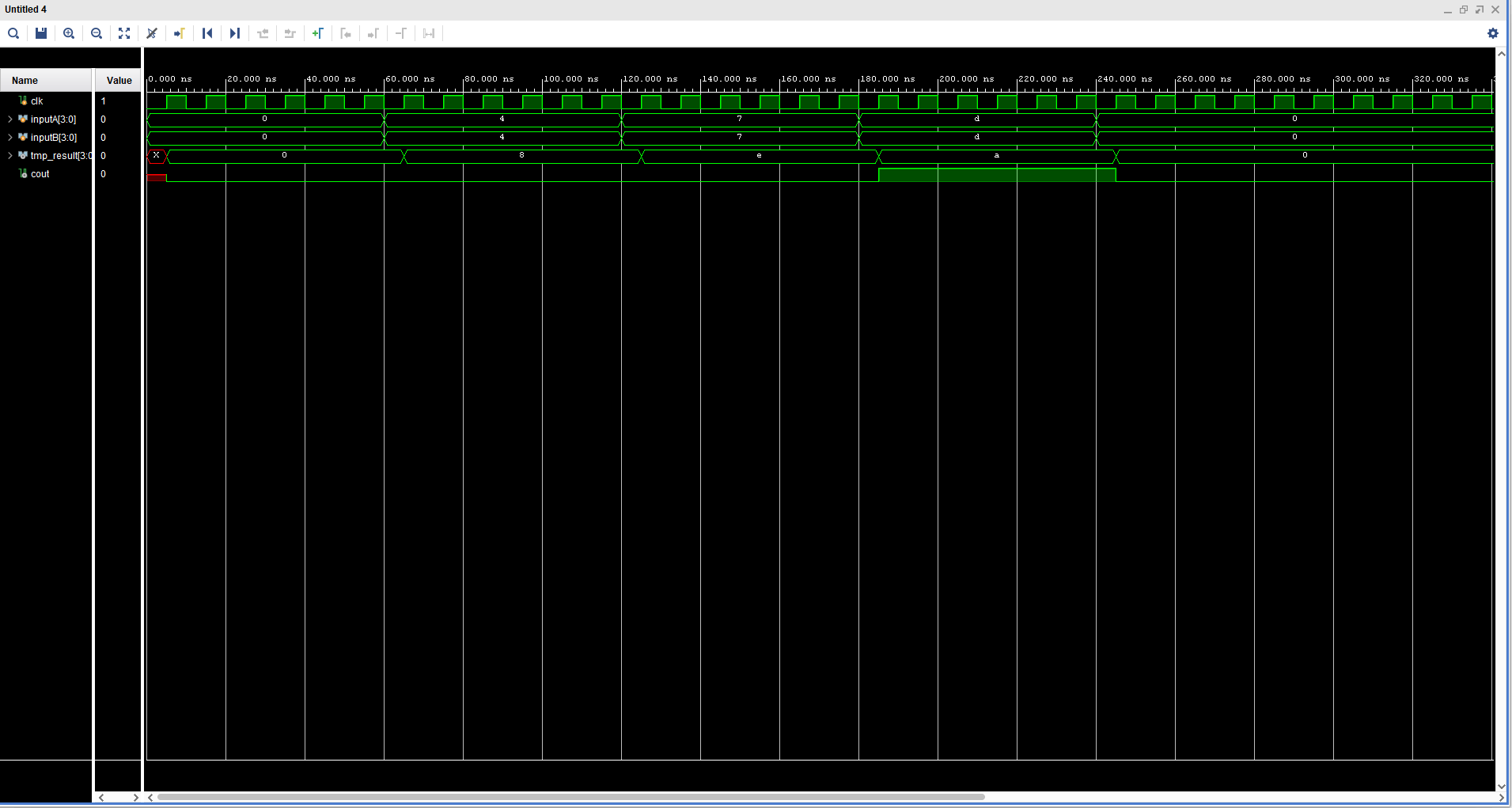


Figure 28: Part 5 4-bit Waveform

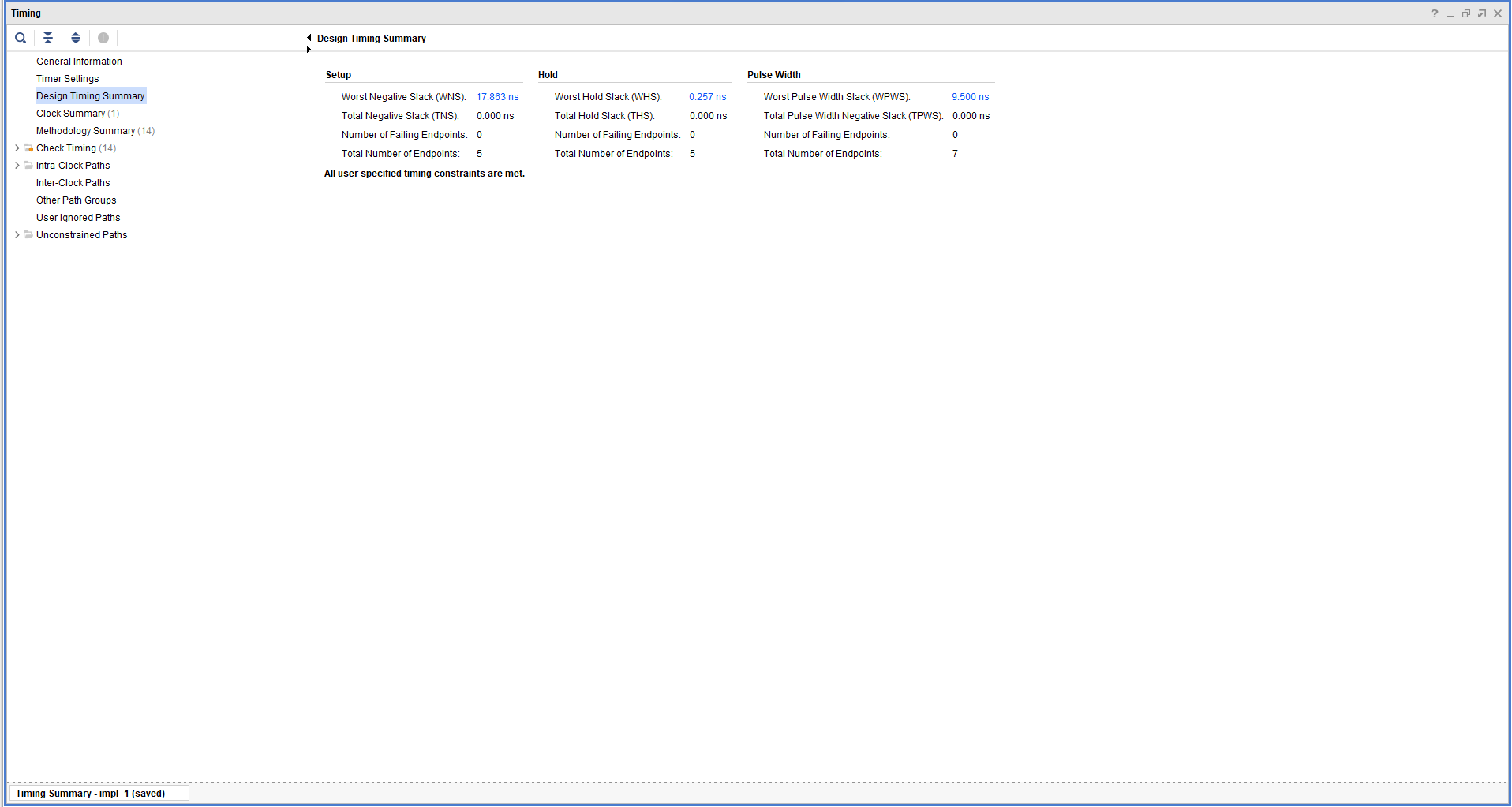


Figure 29: Part 5 4-bit WNS

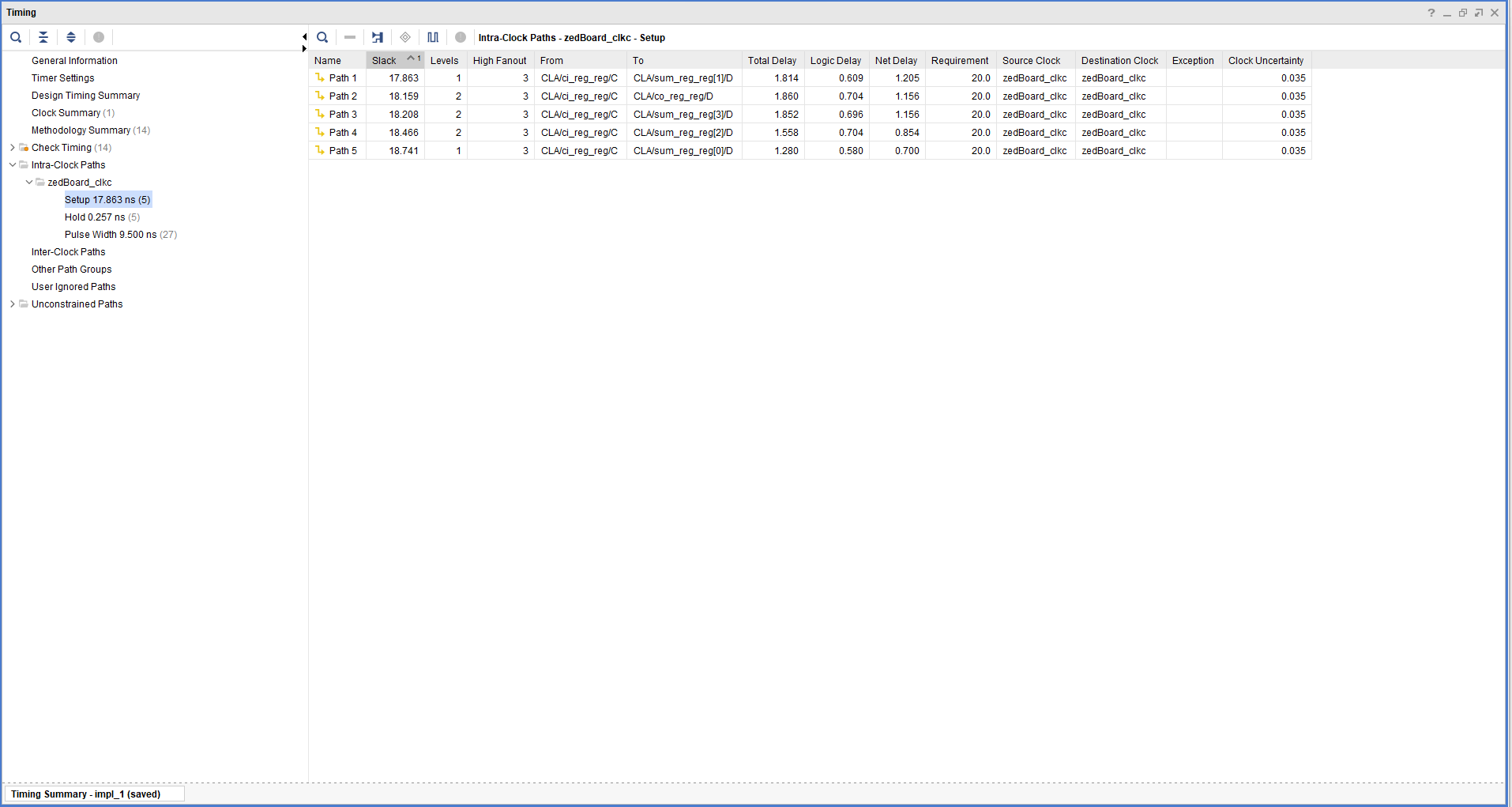


Figure 30: Part 5 4-bit setup

# Contributions

Fatma: Part 0

Pauka: Part 3

Both: Part 1, Part 2, Lab Report