Project One

Benny Wen
CSC 21100
11/4/25

Function Specification

The toUpper() function converts the lowercase ASCII letters to their corresponding uppercase letter. An 8-bit input is taken in representing an ASCII character from values 0 to 255 and produces an 8-bit output.

The function works in the following stages:

1. Check if the input is a lowercase letter (if the input ASCII values are between 97-122, for the letters ‘a’ through ‘z’)
2. If yes, subtract 32 from the input value to get the uppercase version
3. If no, output the input unchanged

Example:

- Input 97 (‘a’) > outputs 65 (‘A’)
- Input 65 (‘A’) > outputs 65 (‘A’) since the input is not between 97-122, already an uppercase letter
- Input 48 (‘0’) > outputs 48 (‘0’) since it’s not a letter

Circuit Design

Input: $I_7 I_6 I_5 I_4 I_3 I_2 I_1 I_0$ (8 bits)

Output: $O_7 O_6 O_5 O_4 O_3 O_2 O_1 O_0$ (8 bits)

Lower 5 bits:

$A = I_4$
$B = I_3$
$C = I_2$
$D = I_1$
$E = I_0$

Lowercase detection, bits $I_7 I_6 I_5$ must equal 011, and lower 5 bits (A-E) must represent values 1-26

Range check ABCDE:

$InRange(A,B,C,D,E) = \Sigma m(1,2,3,4,5,6,7 \ldots 26)$

Expanded canonical:

$$InRange = \bar{A}B\bar{C}\bar{D}E + \bar{A}B\bar{C}D\bar{E} + \bar{A}B\bar{C}DE + \bar{A}BC\bar{D}\bar{E} + \bar{A}BC\bar{D}E + \bar{A}BCD\bar{E} + \bar{A}BCDE + \bar{A}\bar{B}\bar{C}\bar{D}\bar{E} + \bar{A}\bar{B}\bar{C}D\bar{E} + \bar{A}\bar{B}C\bar{D}E + \bar{A}\bar{B}CDE + \bar{A}\bar{B}\bar{C}\bar{D}E + \bar{A}\bar{B}\bar{C}DE + \bar{A}\bar{B}C\bar{D}\bar{E} + \bar{A}\bar{B}CD\bar{E} + A\bar{B}\bar{C}\bar{D}\bar{E} + A\bar{B}\bar{C}D\bar{E} + A\bar{B}C\bar{D}E + A\bar{B}CD\bar{E} + A\bar{B}CDE + AB\bar{C}\bar{D}\bar{E} + AB\bar{C}\bar{D}E + AB\bar{C}D\bar{E}$$

$ISLower = \bar{I}_7 \cdot I_6 \cdot I_5 \cdot InRange$

Unmin Implementation

↳ 26 5 input and gates (each per minterm)
↳ 1 26 input OR gate
↳ gates for upper bits check

K-Map A=0 (values 0-15)

BC \ DE	00	01	11	10
00	0	1	1	1
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

→ all zeros = $\bar{A}\bar{B}\bar{C}\bar{D}\bar{E}$ (detect value 0)

A=1 (values 16-31)

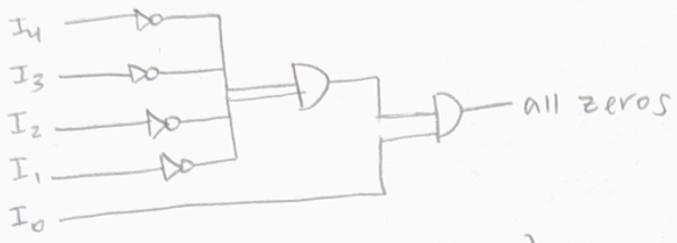
BC \ DE	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	0	0	0	0
10	1	1	0	1

→ TooLarge = $ABC + AB\bar{C}DE$

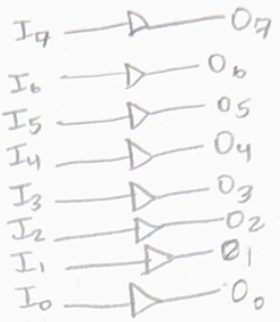
$AllZeros = \bar{A}\bar{B}\bar{C}\bar{D}\bar{E}$
$TooLarge = ABC + AB\bar{C}DE$
$InRange = (AllZeros + TooLarge)'$
$ISLower = \bar{I}_7 I_6 I_5 \; InRange$

$O_7 = I_7$
$O_6 = I_6$
$O_5 = I_5 \leftarrow (IsLower)'$
$O_4 = I_4$
$O_3 = I_3$
$O_2 = I_2$
$O_1 = I_1$
$O_0 = I_0$

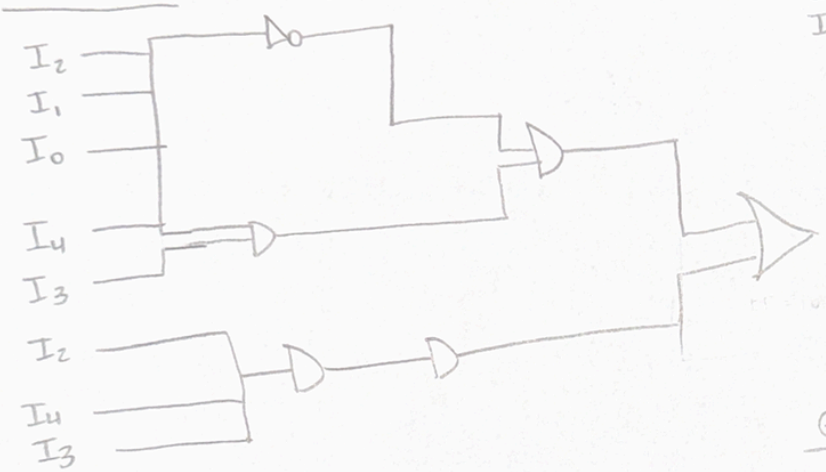
Circuit 1 (All zero detection)



Circuit 6



Circuit 2 (Too Large Detection)



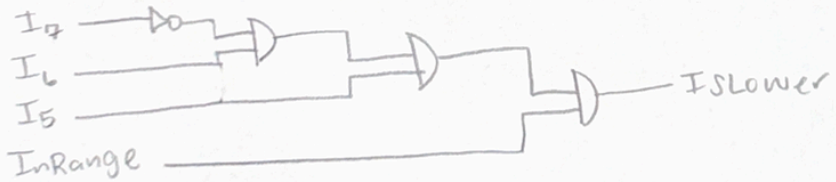
Circuit 3 (In Range)

Gate Count

All Zeros, 5 Not & 3 AND
TooLarge, 1 NOT & 4 AND & 1 OR
InRange, 1 OR & 1 NOT
ISLower, 1 NOT & 3 AND
Output, 7 BUF
Output(5), 1 NOT & 1 AND

29 gates used

Circuit 4 (ISLower)



Circuit 5 (Output Bit 5)

Minimum Delay Investigation

For toUpper_tb.v, a safe inner input delay of 100 ns was used between test cases to ensure the circuit had sufficient time to settle.

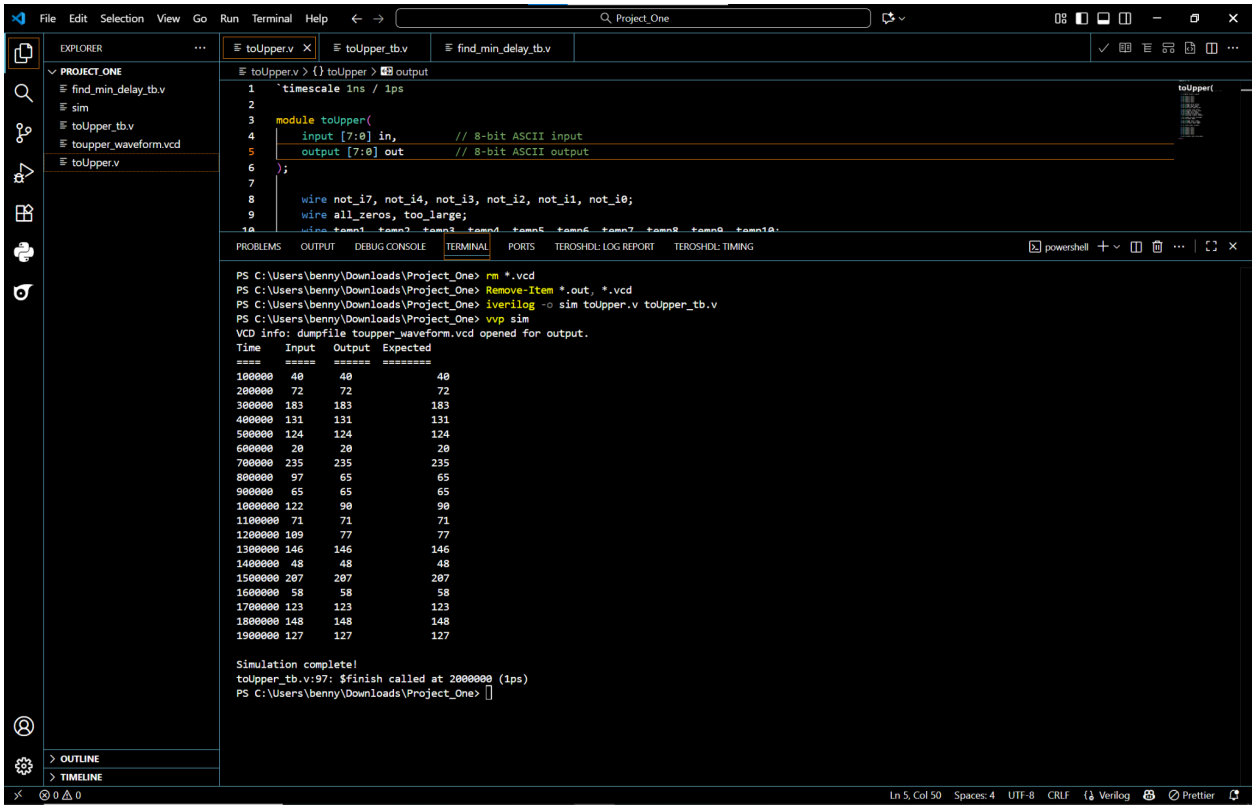
To find the minimum delay, I adjusted the delay parameter in delay_tb.v, starting from 100 ns and decreasing until failures occurred.

Delay (ns)	Outcome	Failed Tests	Notes
100	All pass	0	All conversions correct
65	Fail	3	Inputs 124, 123, 127 fail
90	Fail	1	Inputs 123 outputs 91 (wrong)
91	All pass	0	**Minimum Valid Delay Found**

Screenshots 3 & 4: Terminal output and waveform showing all tests pass with DELAY = 100 ns.
Screenshots 5 & 6: Terminal output and waveform showing 3 failures with DELAY = 65 ns.
Screenshots 7 & 8: Terminal output and waveform showing 1 failure with DELAY = 90 ns (input 123 outputted as 91 instead of 123)
Screenshots 9 & 10: Terminal output showing all tests pass with DELAY = 91 ns, confirming the minimum valid delay is 91 ns.

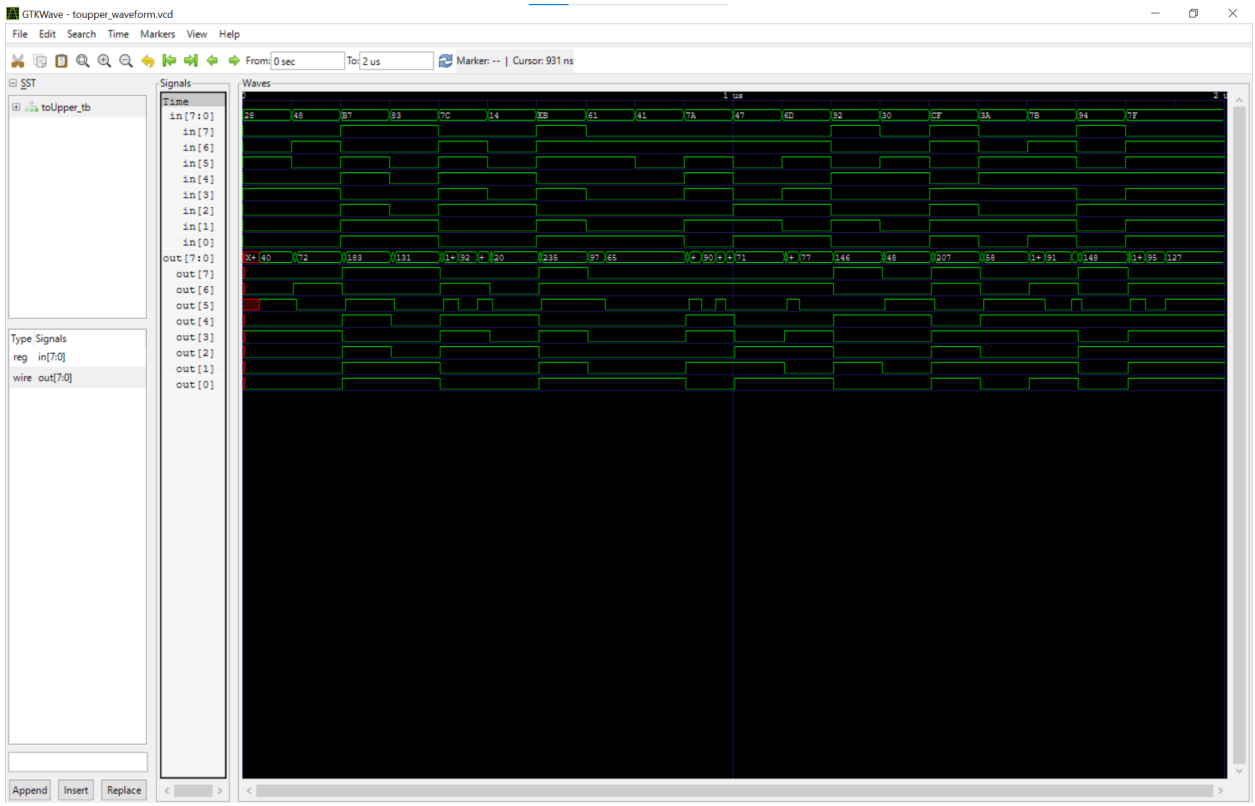
Screenshots

Screenshot 1



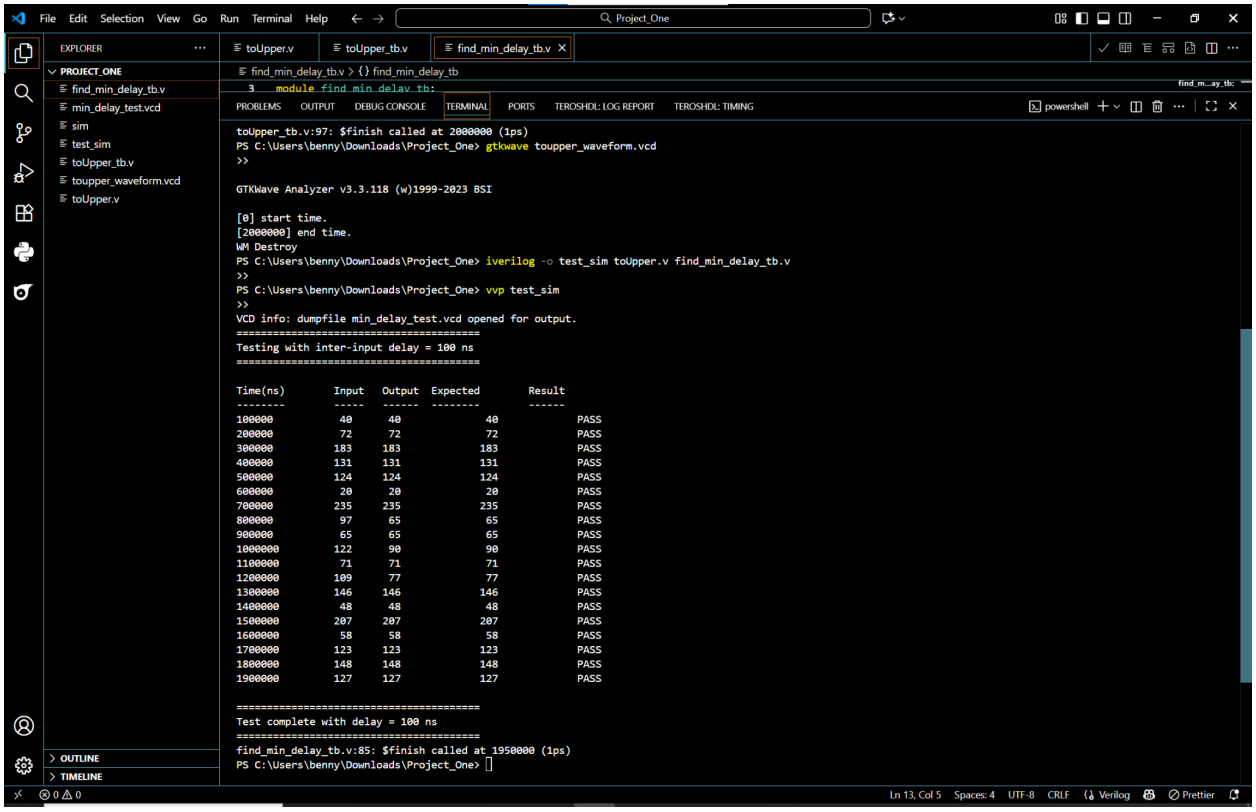
- Screenshot 1 shows the output from running toUpper_tb.v with the default 100 ns delay between inputs. It shows all test cases pass and the output values match the expected ASCII conversions, proving the logic inside toUpper.v works as intended.

Screenshot 2

- 

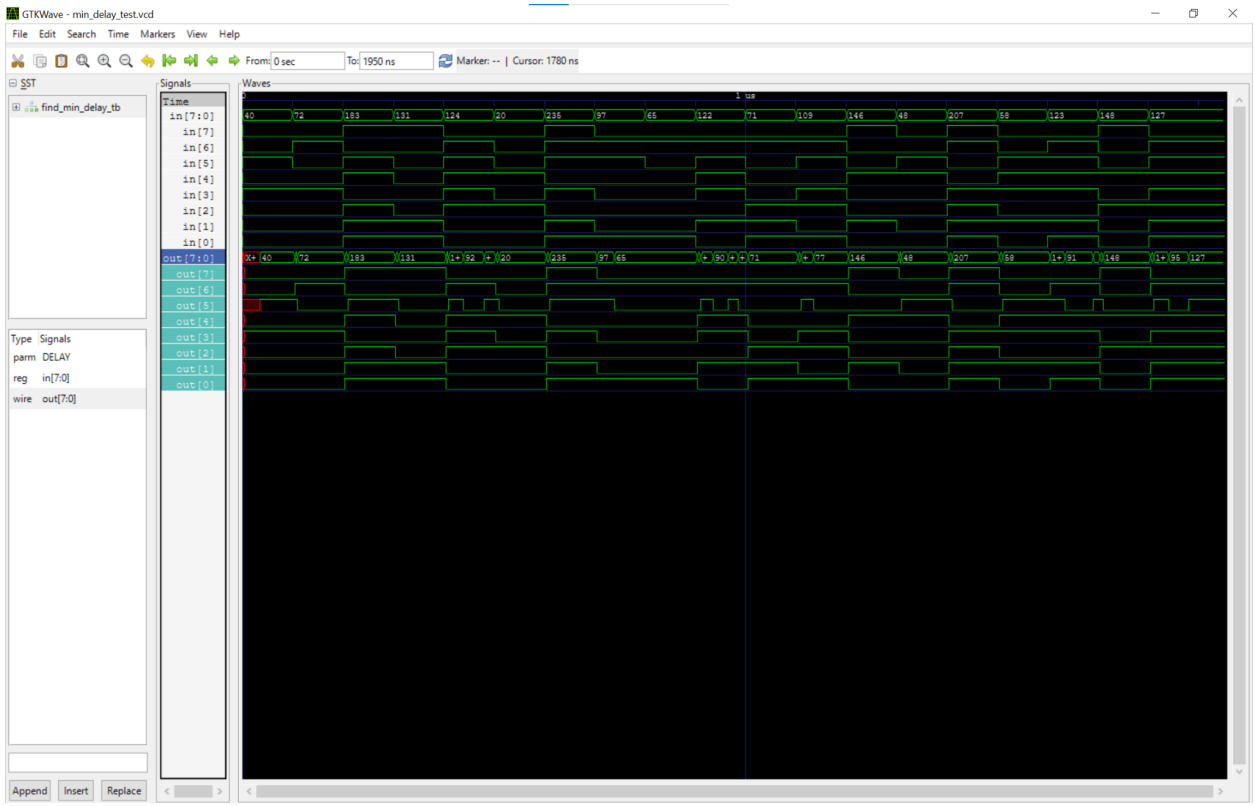
- GTKWave Waveform view of the 100 ns functional test using toupper_waveform.vcd. Where signals in[7:0] and out[7:0] are visible, shown in decimal format.

Screenshot 3



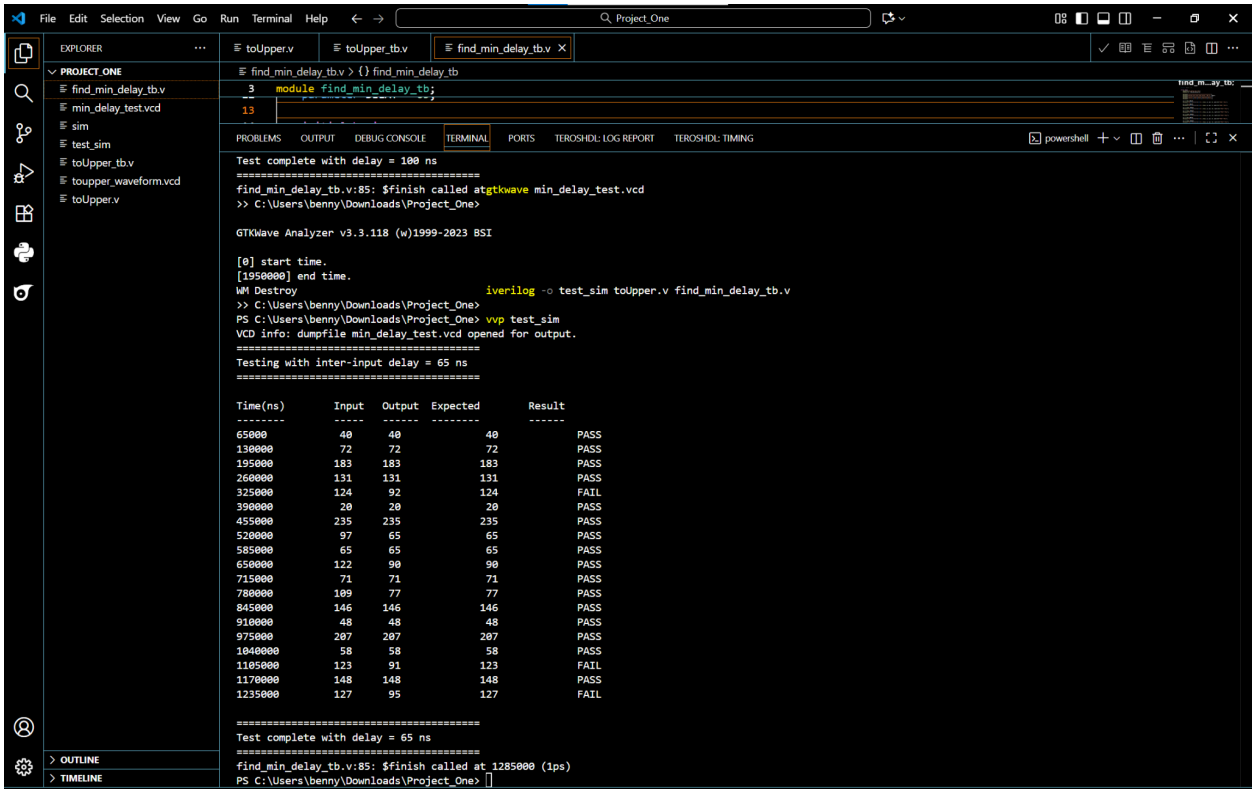
-
- Screenshot 3 shows the terminal output with all PASS values when parameter DELAY = 100 for find_min_delay_tb.v

Screenshot 4



- Screenshot 4 shows the GTKWave waveform view at 100 ns delay showing all signals settling properly between transitions.

Screenshot 5

- 

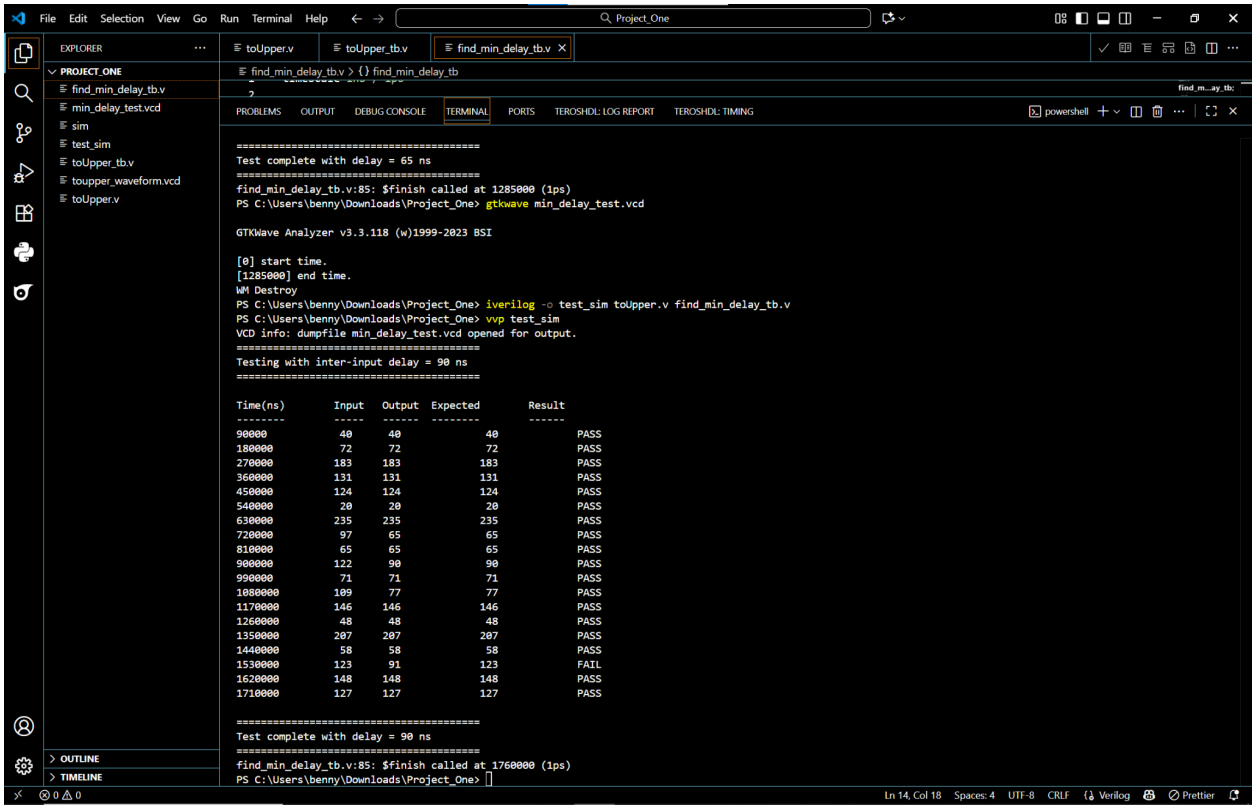
- Screenshot 5 shows the terminal output when parameter DELAY = 65 ns with three failures: inputs 123, 124, and 127.

Screenshot 6

- 

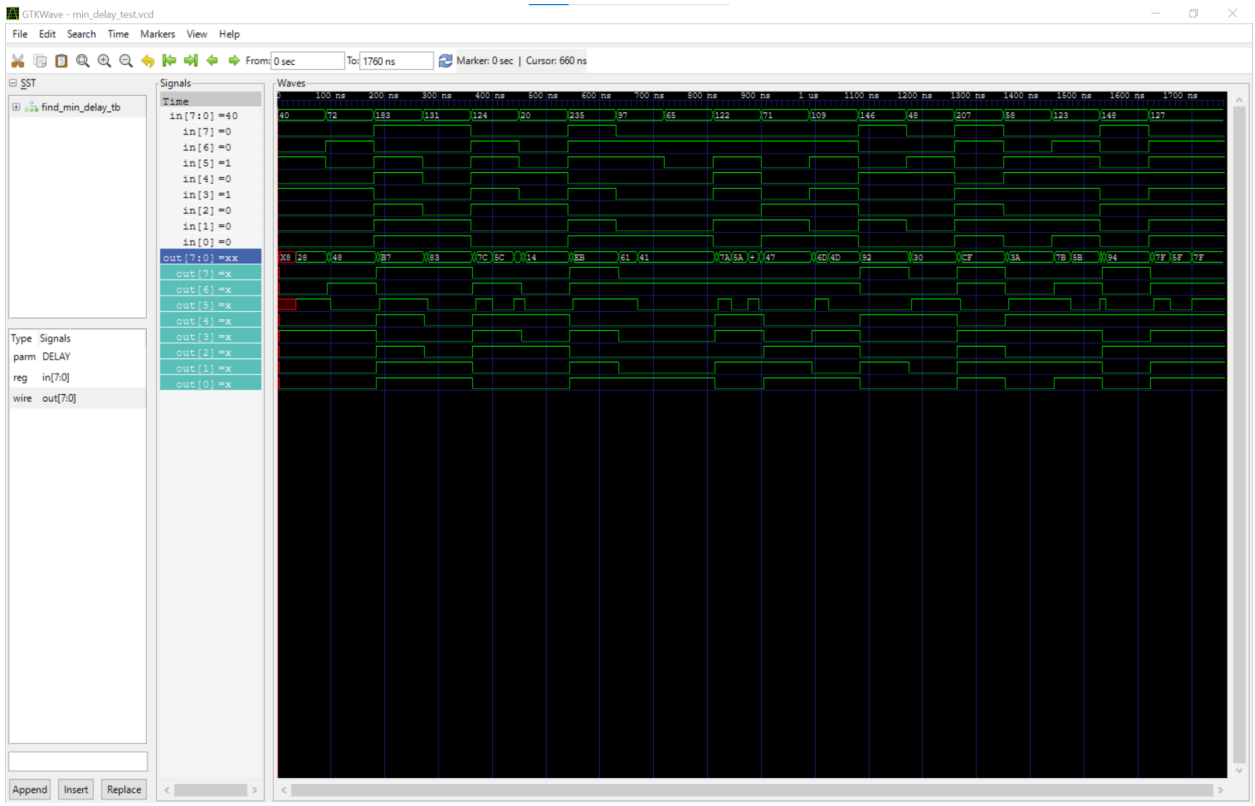
- Screenshot 6 shows GTKWave waveform at 65 ns delay showing failed conversions.

Screenshot 7



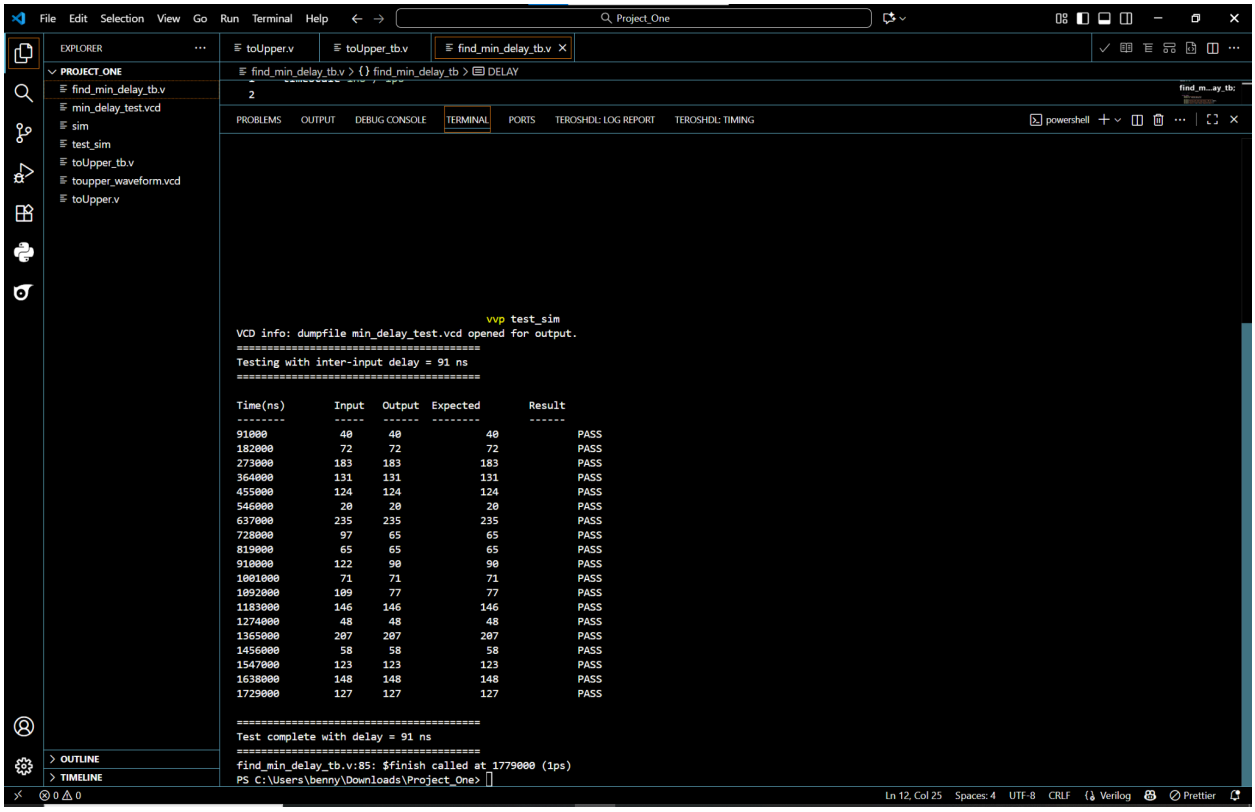
- Screenshot 7 shows terminal output when parameter DELAY = 90 ns showing a single failure (input 123 outputs as 91 instead of 123).

Screenshot 8



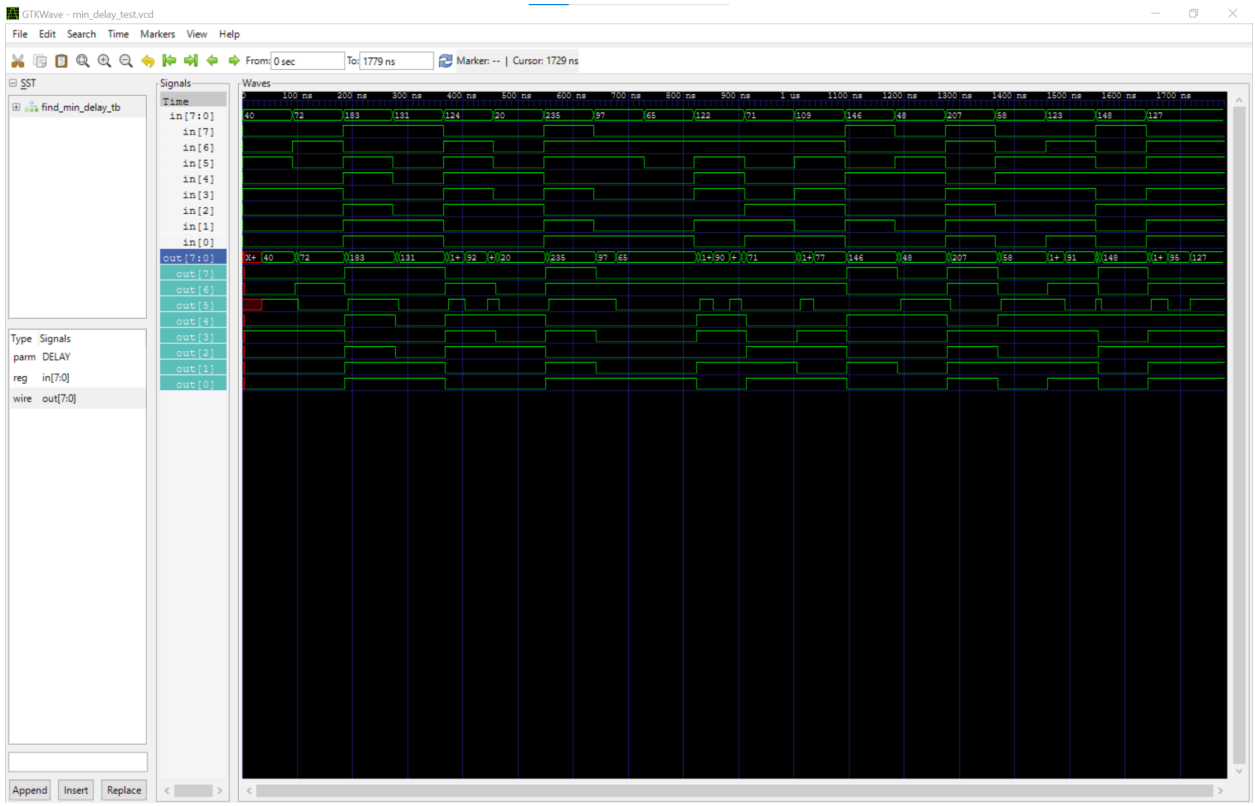
- Screenshot 8 shows GTKWave waveform failure when parameter DELAY = 90 ns.

Screenshot 9



-
- Screenshot 9 shows terminal output when parameter DELAY = 91 ns with all 19 test cases passing. This confirms 91 ns as the minimum delay.

Screenshot 10



- Screenshot 10 shows GTKWave waveform at 91 ns delay showing correct operations.