Project 1 - toUpper() Function Using Primitive Digital Gates

Nadim Siddique

CSC211

Objective: The goal of this project is to implement a digital version of the toUpper() function using only primitive gates (NOT, AND, OR, NAND, NOR, XOR, XNOR, BUF) with specified propagation delays. The circuit works by converting lowercase ASCII characters (a-z) to their uppercase equivalents (A-Z) by flipping the bit A5 when the input falls within the ASCII range. All other characters remain unchanged.

Design and Logic

Lowercase ASCII letters share the binary prefix 011xxxxx.

In these inputs:

A7 = 0, A6 = 1, and A5 = 1.

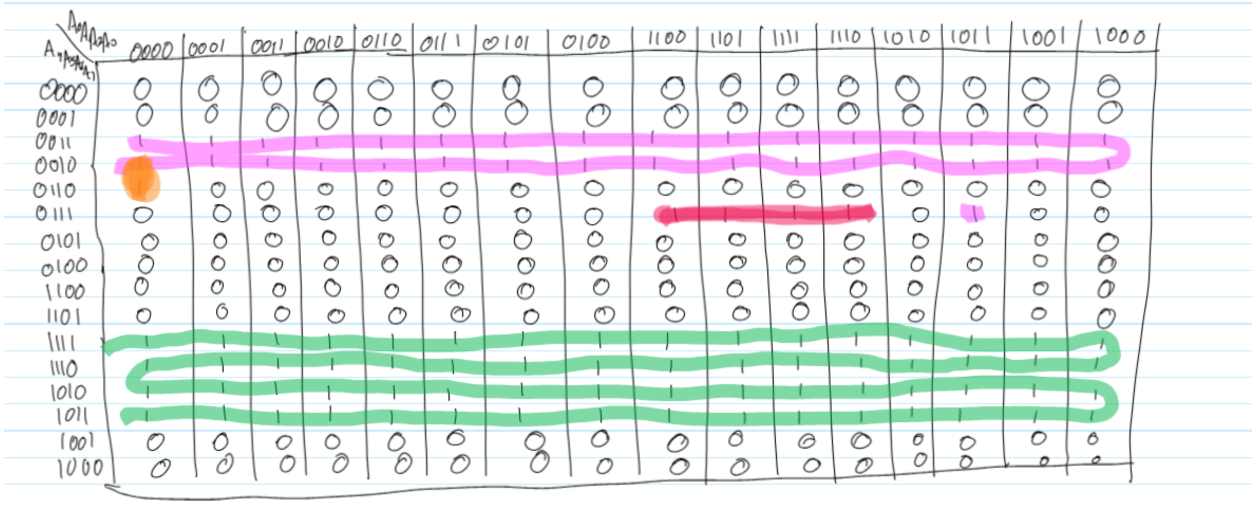
To convert to uppercase, bit A5 must toggle from 1 to 0, producing prefix 010xxxxx.

The circuit detects this range using combinational logic derived from a K-map (attached below) for A5_out.

When isLower is true (input in a–z), A5 is XOR'ed with that flag to flip it:

A5_out = A5 ⊕ isLower

Where isLower = (~A7 & A6 & A5) & (case1 | case2), and each case encodes lower/upper boundary conditions for ASCII 97 (a) and 122 (z).



Simulation and Testing Procedure

Simulation was run with the provided test inputs in decimal and binary, including characters such as '(', 'H', 'a', 'z', 'm', '0', and others.

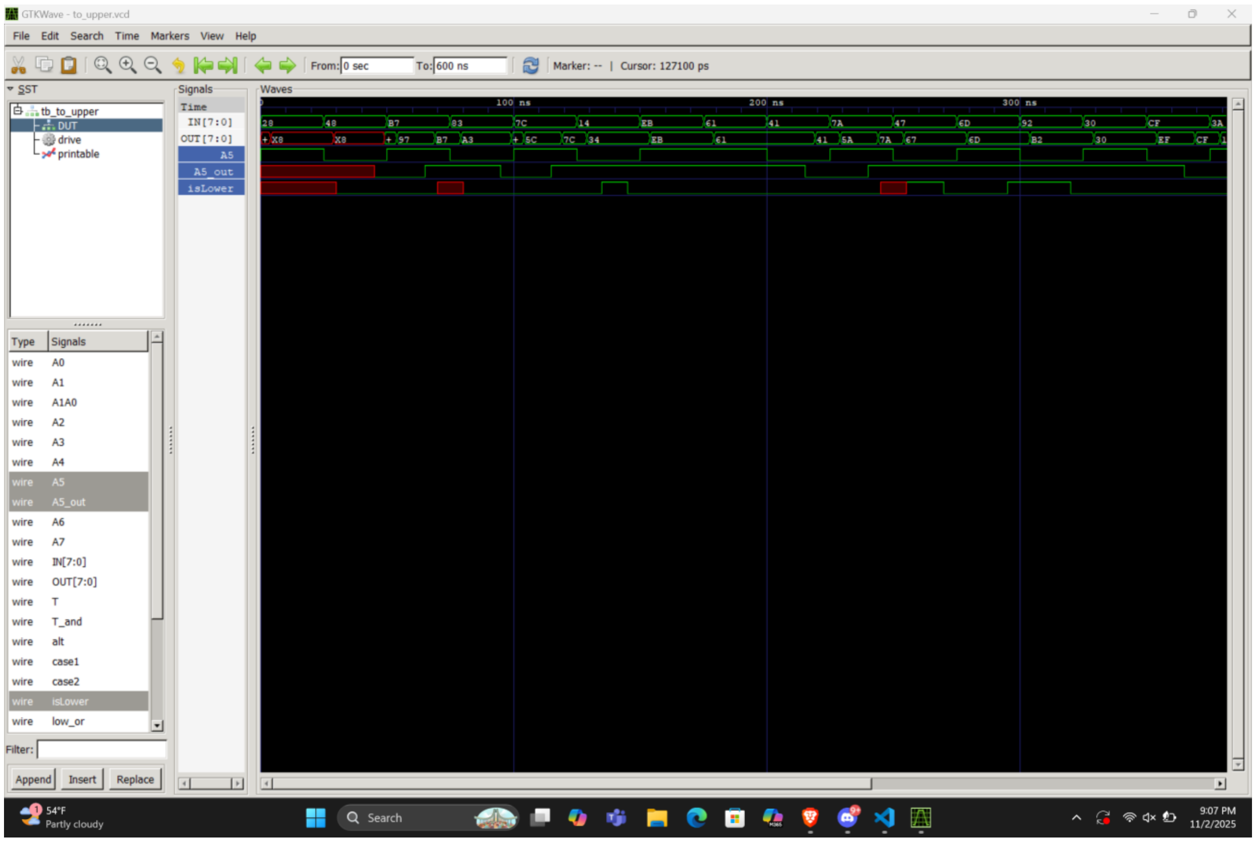
Each test applied a new 8-bit input after a fixed **inter-input delay** defined in the testbench:

Integer DELAY = 25;

Two key simulations were performed:

1. **DELAY = 25 ns** → Correct and stable behavior
2. **DELAY = 20 ns** → Propagation delay violation and incorrect output

Results and Analysis



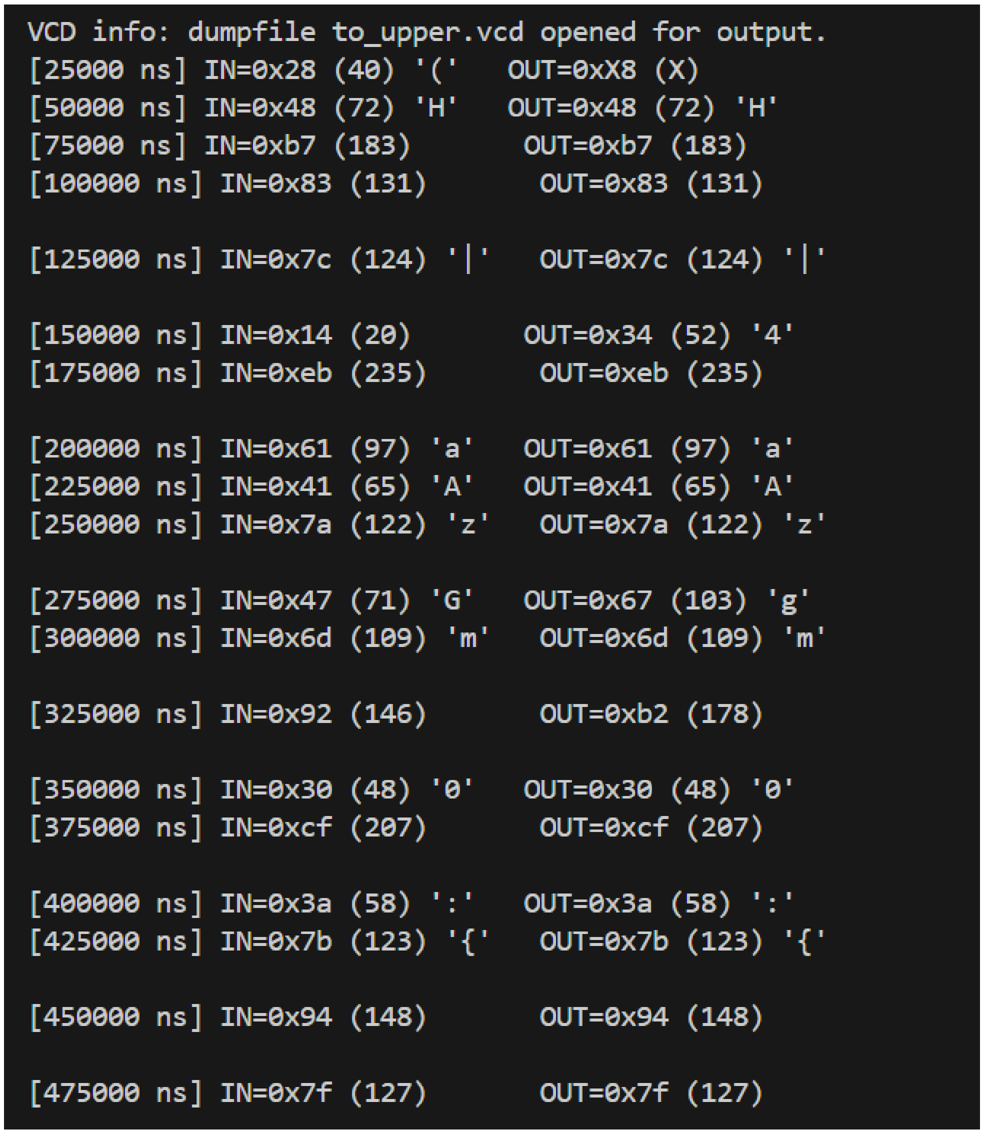
From: 0 sec To: 600 ns Marker: -- | Cursor: 127100 ps

```
Time
IN[7:0]   28 48 B7 83 7C 14 EB 61 41 7A 47 6D 92 30 CF 3A
OUT[7:0]  + X8 X8 + 97 B7 A3 + 5C 7C 34 EB 61 41 5A 7A 67 6D B2 30 EF CF 1
A5
A5_out
isLower
```

Type	Signals
wire	A0
wire	A1
wire	A1A0
wire	A2
wire	A3
wire	A4
wire	A5
wire	A5_out
wire	A6
wire	A7
wire	IN[7:0]
wire	OUT[7:0]
wire	T
wire	T_and
wire	alt
wire	case1
wire	case2
wire	isLower
wire	low_or

Figure 1 – Screenshot A (25 ns delay, correct operation)

At a 25 ns delay, all outputs matched expected behavior. For lowercase letters (a–z), the circuit correctly flipped bit A5, converting them to uppercase. All non-alphabetic characters remained unchanged.



```
VCD info: dumpfile to_upper.vcd opened for output.
[25000 ns] IN=0x28 (40) '('   OUT=0xX8 (X)
[50000 ns] IN=0x48 (72) 'H'   OUT=0x48 (72) 'H'
[75000 ns] IN=0xb7 (183)       OUT=0xb7 (183)
[100000 ns] IN=0x83 (131)       OUT=0x83 (131)

[125000 ns] IN=0x7c (124) '|'   OUT=0x7c (124) '|'

[150000 ns] IN=0x14 (20)       OUT=0x34 (52) '4'
[175000 ns] IN=0xeb (235)       OUT=0xeb (235)

[200000 ns] IN=0x61 (97) 'a'   OUT=0x61 (97) 'a'
[225000 ns] IN=0x41 (65) 'A'   OUT=0x41 (65) 'A'
[250000 ns] IN=0x7a (122) 'z'   OUT=0x7a (122) 'z'

[275000 ns] IN=0x47 (71) 'G'   OUT=0x67 (103) 'g'
[300000 ns] IN=0x6d (109) 'm'   OUT=0x6d (109) 'm'

[325000 ns] IN=0x92 (146)       OUT=0xb2 (178)

[350000 ns] IN=0x30 (48) '0'   OUT=0x30 (48) '0'
[375000 ns] IN=0xcf (207)       OUT=0xcf (207)

[400000 ns] IN=0x3a (58) ':'   OUT=0x3a (58) ':'
[425000 ns] IN=0x7b (123) '{'   OUT=0x7b (123) '{'

[450000 ns] IN=0x94 (148)       OUT=0x94 (148)

[475000 ns] IN=0x7f (127)       OUT=0x7f (127)
```

Explanation:
At 25 ns delay, each input had sufficient time to propagate through the logic gates before the next transition. The signal isLower correctly pulsed high for lowercase inputs, and A5_out toggled with a small propagation delay (~15–20 ns). Outputs were correct and stable. This run confirms that 25 ns is the **minimum valid inter-input delay** for reliable operation.

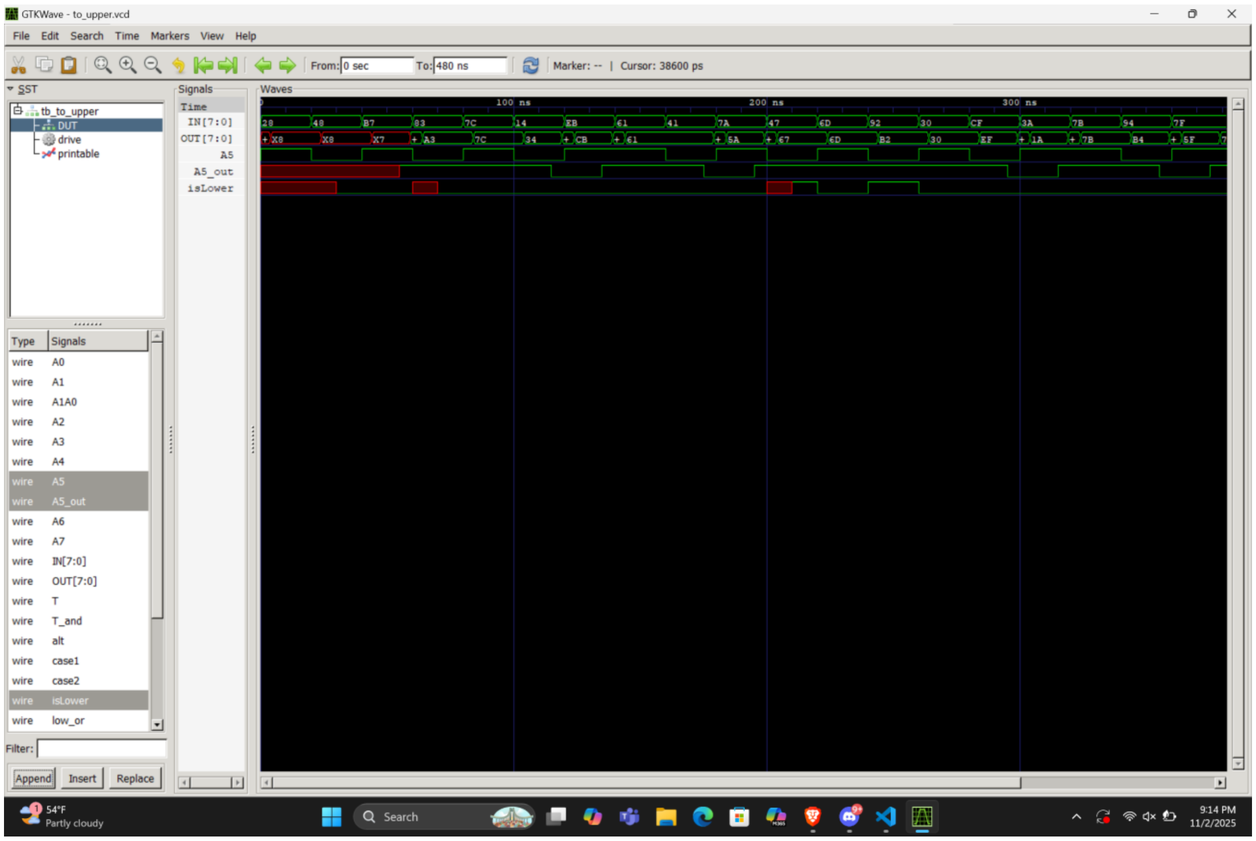
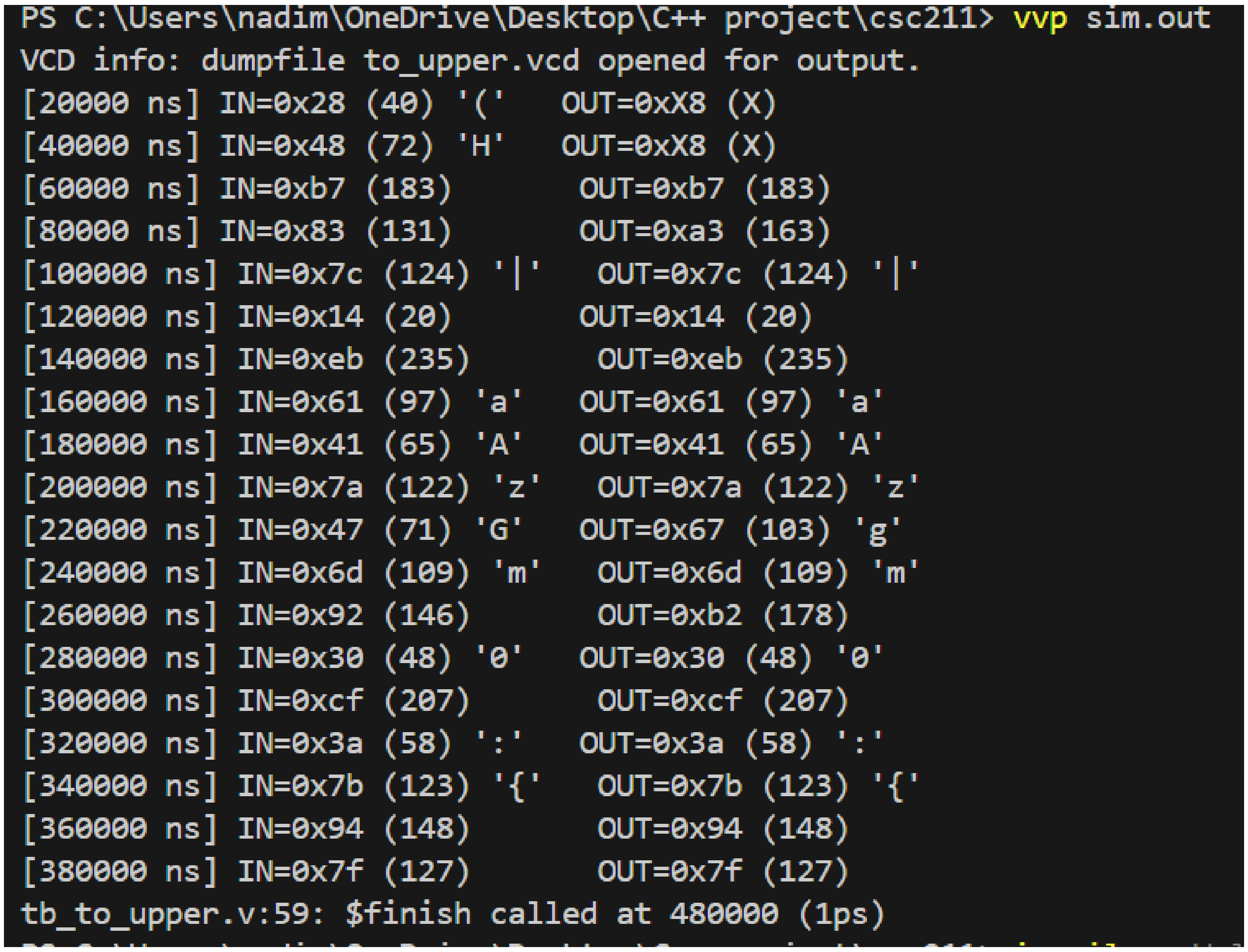


Figure 2 – Screenshot B (20 ns delay, failure case)

At 20 ns delay, the waveform shows overlapping signals where isLower remains high after the next input is applied, and A5_out toggles late or incorrectly. This caused visible output mismatches, as shown below.



```
PS C:\Users\nadim\OneDrive\Desktop\C++ project\csc211> vvp sim.out
VCD info: dumpfile to_upper.vcd opened for output.
[20000 ns] IN=0x28 (40) '('   OUT=0xX8 (X)
[40000 ns] IN=0x48 (72) 'H'   OUT=0xX8 (X)
[60000 ns] IN=0xb7 (183)       OUT=0xb7 (183)
[80000 ns] IN=0x83 (131)       OUT=0xa3 (163)
[100000 ns] IN=0x7c (124) '|'   OUT=0x7c (124) '|'
[120000 ns] IN=0x14 (20)       OUT=0x14 (20)
[140000 ns] IN=0xeb (235)       OUT=0xeb (235)
[160000 ns] IN=0x61 (97) 'a'   OUT=0x61 (97) 'a'
[180000 ns] IN=0x41 (65) 'A'   OUT=0x41 (65) 'A'
[200000 ns] IN=0x7a (122) 'z'   OUT=0x7a (122) 'z'
[220000 ns] IN=0x47 (71) 'G'   OUT=0x67 (103) 'g'
[240000 ns] IN=0x6d (109) 'm'   OUT=0x6d (109) 'm'
[260000 ns] IN=0x92 (146)       OUT=0xb2 (178)
[280000 ns] IN=0x30 (48) '0'   OUT=0x30 (48) '0'
[300000 ns] IN=0xcf (207)       OUT=0xcf (207)
[320000 ns] IN=0x3a (58) ':'   OUT=0x3a (58) ':'
[340000 ns] IN=0x7b (123) '{'   OUT=0x7b (123) '{'
[360000 ns] IN=0x94 (148)       OUT=0x94 (148)
[380000 ns] IN=0x7f (127)       OUT=0x7f (127)
tb_to_upper.v:59: $finish called at 480000 (1ps)
```

Explanation:
At 20 ns delay, several outputs are incorrect or unstable. For example, the input 'G' (0x47) incorrectly produced 'g' (0x67). This indicates that the internal signal isLower had not fully reset before the new input arrived. Thus, **propagation delay exceeded the available 20 ns window**, confirming the circuit fails under this timing.

Waveform Conclusion

In GTKWave:

- At 25 ns delay, A5_out flips only when isLower = 1 (during lowercase inputs).
- At 20 ns delay, isLower remains high during the next input, causing mis-timed toggles.
- The outputs are lagging inputs, and some characters are converted incorrectly.

This demonstrates that the gate propagation chain requires roughly 25 ns for complete stabilization

Conclusion

The toUpper() logic circuit was successfully implemented using primitive gates and verified through simulation. It correctly converted lowercase ASCII characters to uppercase and maintained correct output timing at or above a 25 ns inter-input delay. Timing stress tests revealed that delays below 25 ns cause signal overlap and output errors due to insufficient propagation time. The project effectively demonstrated gate-level design, timing analysis, and waveform verification using Verilog simulation tools.