

1 Background

Consider the following 3-input,2-output truth table:

a	b	c	y	z
1	1	1	0	1
1	1	0	1	1
1	0	0	0	0
otherwise			0	1

where - denotes don't care

Implement the table as a module using the following variations. Synthesize each version and submit a screen-capture of the RTL view.

- using behavioral code with if statements
- using concurrent continuous assignment statement(s)
- using behavioral code with case statements

2 Using Behavioral Code With 'If' Statements

The module implementation along with its testbench can be found in the 'scripts' directory. A sample of the waveform generated is provided:

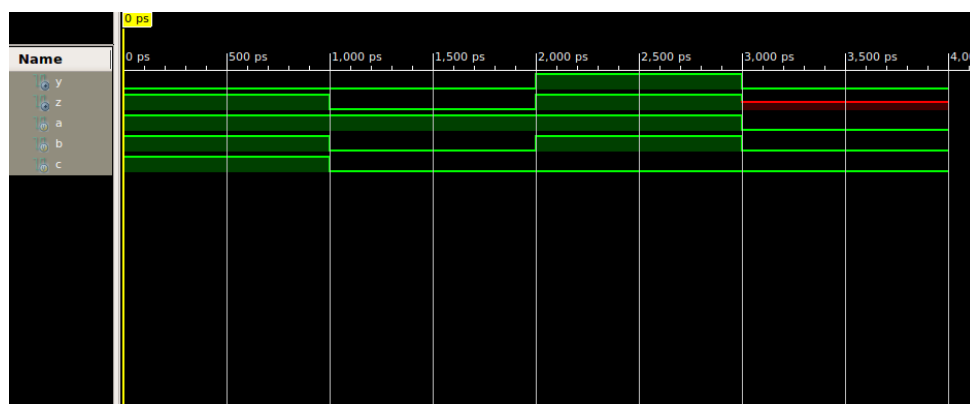


Figure 1: Waveform Generated from Part 1 Version A Test Bench

The module was synthesized and its RTL view was generated:

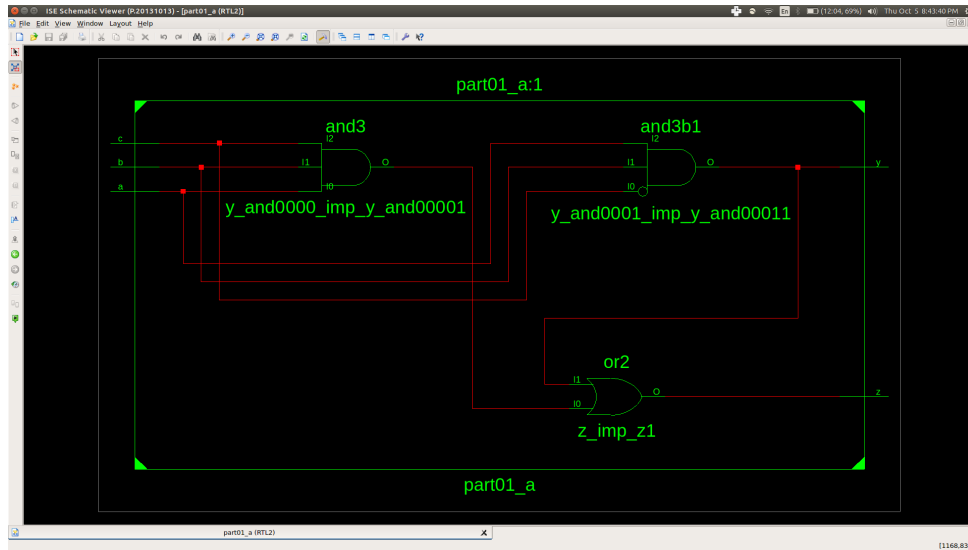


Figure 2: RTL View of Part 1 Version A

3 Using Concurrent Continuous Assignment Statement(s)

The module implementation along with its testbench can be found in the 'scripts' directory. A sample of the waveform generated is provided:

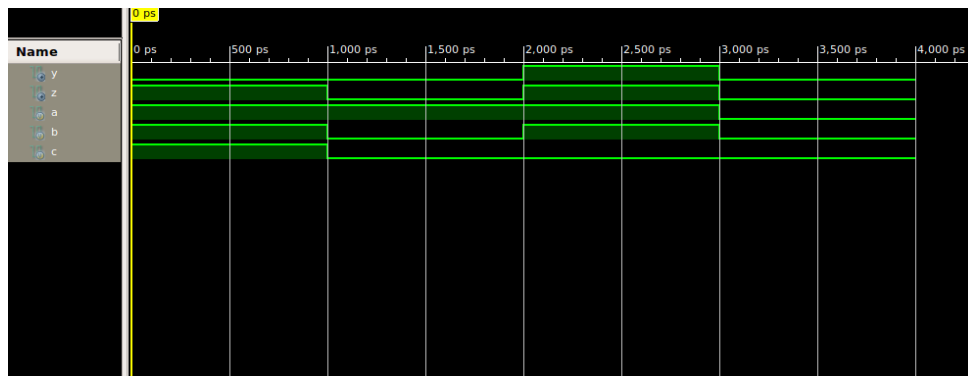


Figure 3: Waveform Generated from Part 1 Version B Test Bench

The module was synthesized and its RTL view was generated:

4 Using Behavioral Code With Case Statements

The module implementation along with its testbench can be found in the 'scripts' directory. A sample of the waveform generated is provided:

The module was synthesized and its RTL view was generated:

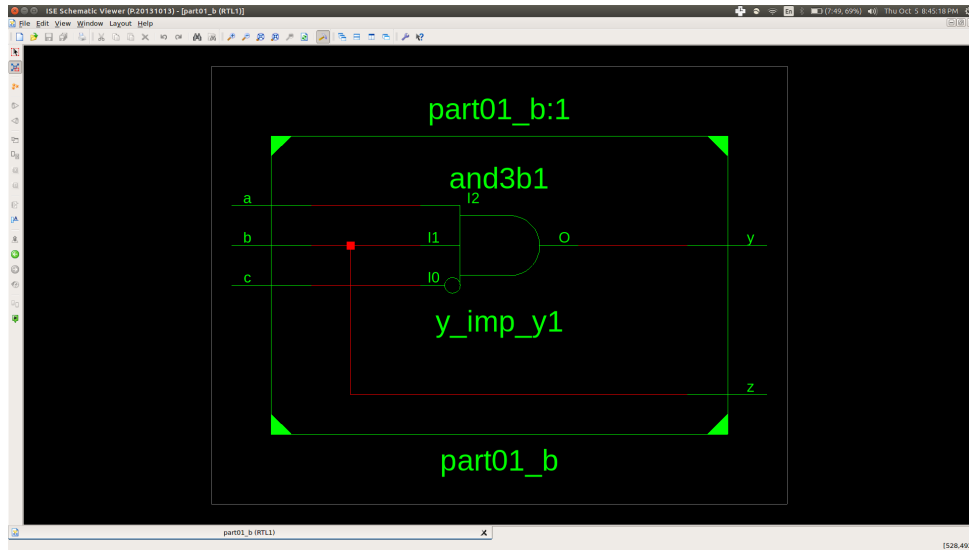


Figure 4: RTL View of Part 1 Version B

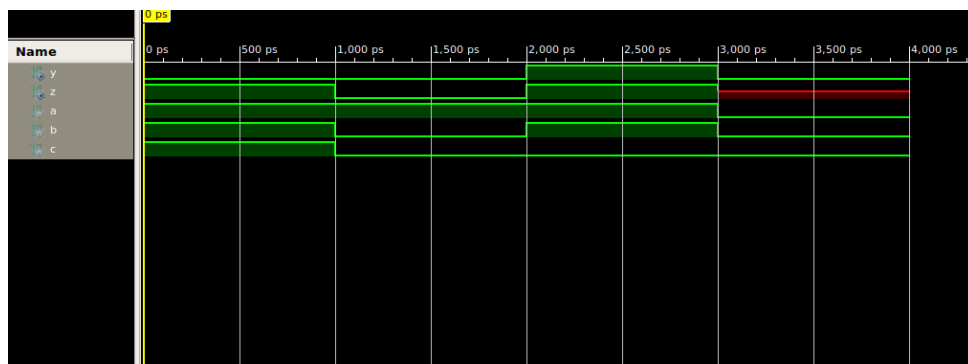


Figure 5: Waveform Generated from Part 1 Version C Test Bench

5 Observations

The Register Transfer Level of Version B appears to have the simplest hardware implementation due to its minimized equations. However, using concurrent continuous assignment statements do not allow any explicit 'don't care' assignments.

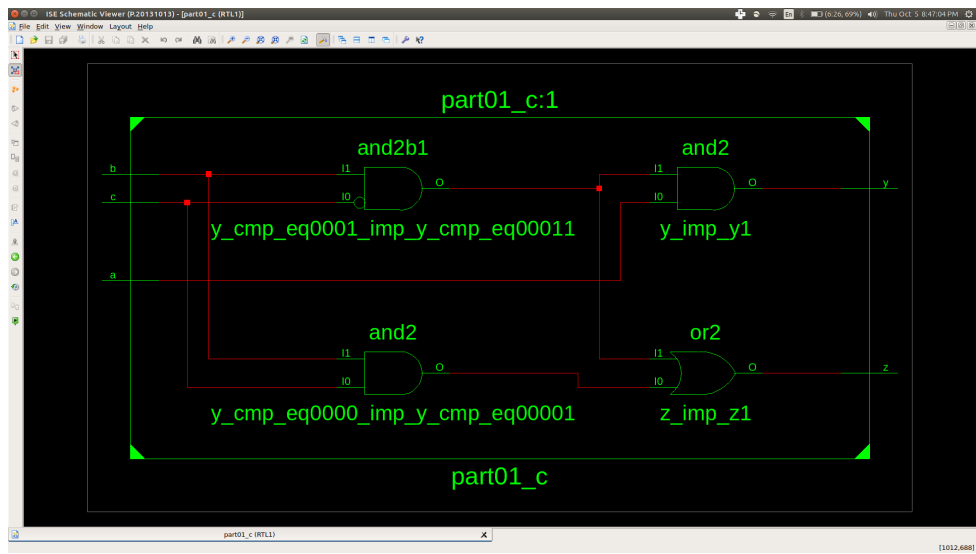


Figure 6: RTL View of Part 1 Version C