|  |
| --- |
| ELEC 402 |
| Project 2 Report |
| Synthesized Version of Project 1; https://github.com/mchuahua/ELEC402/tree/master/Proj2 |

|  |
| --- |
| Martin Chua - 35713411  10-8-2021 |

**Mapped Verilog generated by RTL Compiler**

See ./fsm\_map.v

**Visual Waveforms showing state transitions from mapped Verilog:**

As the synthesized mapped Verilog doesn’t contain enum from the Proj1 testbench, it was decided that in order to verify the state transitions, the simplest method would be to utilize a visual comparison between the Proj1 waveforms and Proj2 waveforms. This was also the easiest method to confirm the state transitions, as the output/input signals of the DUT depend on the states and state transitions. Therefore, although assertions were not used in the Proj2 tb, just by having the waveforms match (disregarding delay) means that the state transitions also match, and work correctly.

Below is the waveform for the mapped Verilog (first):

Timeline

Description automatically generated

And below is the Proj1 waveform with verifiable states:

A picture containing diagram

Description automatically generated

There are various signals to note that will be listed below. One major point is that the Ready signal is asserted when state is in IDLE. The test\_num signal can also allow the user to glean information about what is happening. To explain why this is possible:

1. Test num 0 is in IDLE. Ready is asserted. This is correct.
2. Test num 1 sees that ready is asserted and deasserted, as it transitions back and forth due to invalid PIN (0-9). This is correct.
3. Test num 2 sees that after a valid PIN, state should not be in IDLE. This is correct. If it wasn’t the ready signal should be asserted just as in test\_num 1.
4. Test num 3 sees that withdrawal (0) makes FSM go to the correct state. Seeing that ready isn’t asserted, the state is not in IDLE and is proceeding along.
5. Test num 4 sees that reset works and if deposit (1) goes to correct state. Notice that the two waveforms are the same (albeit Proj2 waveform has delay). Therefore the deposit states are correctly being transitioned to.
6. Test num 5 checks to see if deposited correct. Notice the open atm output is being correctly triggered in the proj 2 waveform, matching the proj 1 waveform.
7. Test num 6 checks to see if reset + withdrawal was correct. Again, both waveforms match. At the end of test num 6, we see the open\_atm\_dispense was triggered properly, just as in proj 1 waveform. This suggests that the withdrawal states are correctly being transitioned to.
8. The next three tests can be understood by understanding the state functionality. The state for looping an amount check is before the state for looping the withdraw card check, which then goes to IDLE if card is withdrawn. Notice in test num 7 the input amount is changed. This should trigger a loop, as there are insufficient funds. This matches proj1 waveform. Now, if we change the input amount back to 1, the state should continue along and wait for the bank card to be withdrawn. In test num 8, the card is never withdrawn, and our waveforms match. Likewise, the bank card is then withdrawn, and in test num 9 we see Ready being asserted, indicating that we are indeed in IDLE.

**Report from RTL Compiler showing total number of cells in project**

See ./fsm\_area.rpt

Text

Description automatically generated

Seems as expected; passing above minimum threshold requirement of 100 cells. Expected to be bigger because FSM was slightly above trivial (10+ states, various state transitions).

See ./fsm\_timing.rpt

A picture containing graphical user interface

Description automatically generated

Timing slack is positive, which is good. Initial following according to the tutorial document gave a timing slack of 1ps, but after following Sean’s advice on piazza on how to change the timing unit, the slack increased. This is good because it allows more leeway for signals to propagate.

See ./fsm\_gates.rpt

Text

Description automatically generated

This is expected and matches expectations. NAND and NOR should be the highest because they are considered as universal gates and are easier and more economical to fabricate. There is more logic instances than sequential or inverter because Cadence is able to optimize some of the unused sequential logic to logic constants (LUTs). This effectively reduces power (for propagating unnecessary buses) and reduces size of area that might otherwise be used.

**Changed Designs from Project 1**

For testbench, all assertions were commented out. This can be seen in fsm\_tb.sv.

For fsm, 2 small changes were made:

Text

Description automatically generated with medium confidence

1. Last three coloured lines were fixes to allow the system Verilog to be able to be synthesized.
2. First three coloured lines switched to allow correct functionality in waveform. See below:

Working with fixes mentioned above:

Timeline

Description automatically generated

Not working without fixes:

Without the fixes, as indicated by the red circles there were some unintended assertions of the ready signal. It turns out that using the “==” produces a longer enough propagation time (cascading transistor delay) in comparison to “!=” that the states are not propagating fast enough for the state machine to function correctly.