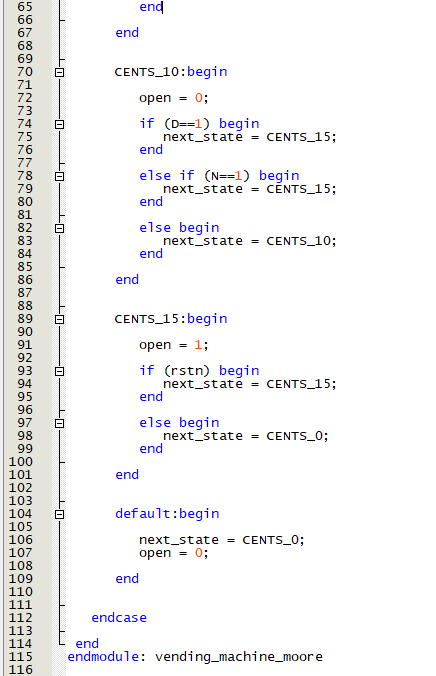
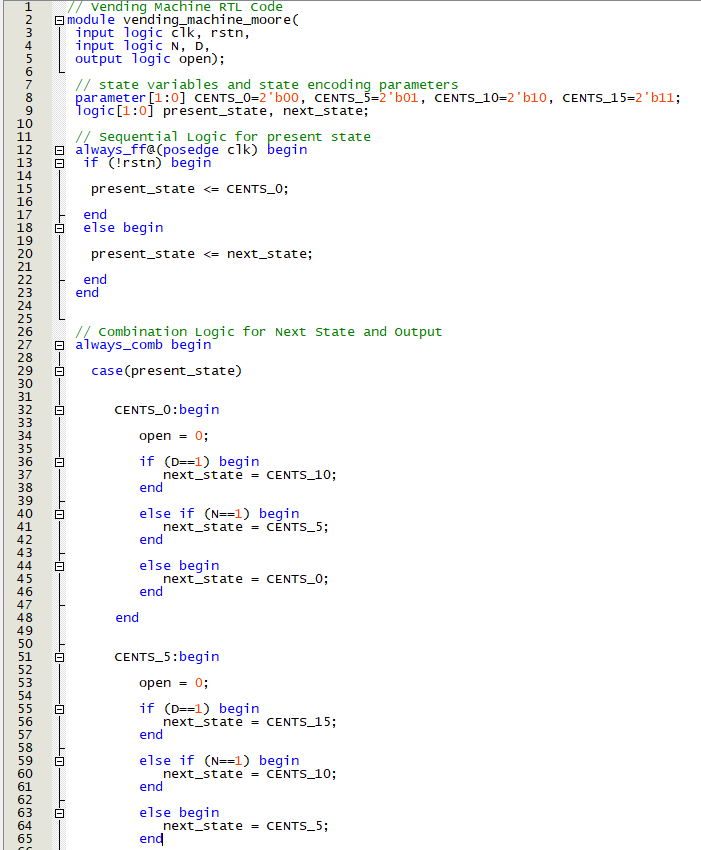
ECE 111 Winter 2022  
HW6  
Hao Le A15547504

**Moore Vending Machine**

Code



RTL netlist

Diagram

Description automatically generated

Resource usage

Graphical user interface

Description automatically generated

State diagram

Diagram

Description automatically generated

A picture containing text, electronics

Description automatically generated

Testbench simulation waveform

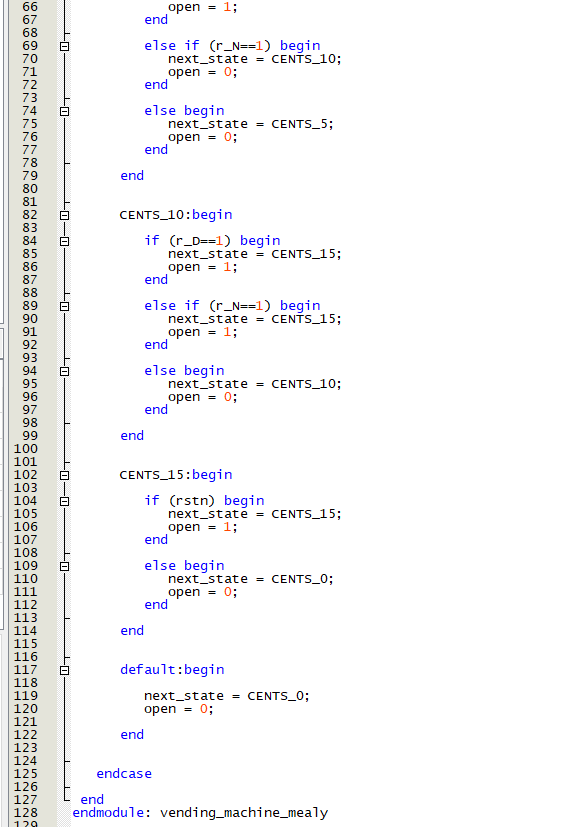
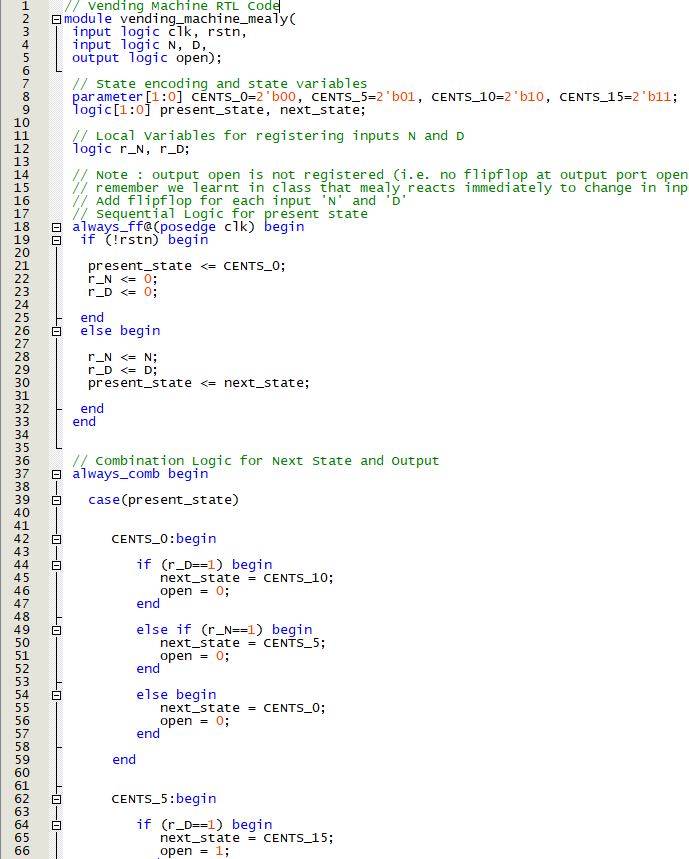
A screenshot of a computer

Description automatically generated with medium confidence

* A Moore state machine generates an output based on the current state, regardless of the inputs
* We see this behavior in the waveform; for example, in the first operation starting at around 30ns, a nickel is inserted, but since it was inserted at the falling edge of the clock, the state does not change from 00 to 01 (5 cents) until the next rising clock edge. We see this for the proceeding dime, where upon the rising edge, the state changes from 5 cents to 15 cents, thus bringing the output high
* This is distinct from a Mealy state machine in that the output did not go high immediately after the dime was inserted
* Lastly, once the 15 Cents (11) state has been reached, rstn goes low, but only at the rising clock edge does the present state switch back to 0 Cents, and the output goes low
* Described above is the correct behavior

**Mealy Vending Machine**

Code



RTL netlist

Diagram, schematic

Description automatically generated

Resource usage

Graphical user interface, application

Description automatically generated

State diagram

Diagram

Description automatically generated

Table

Description automatically generated

Testbench simulation waveform

Graphical user interface

Description automatically generated

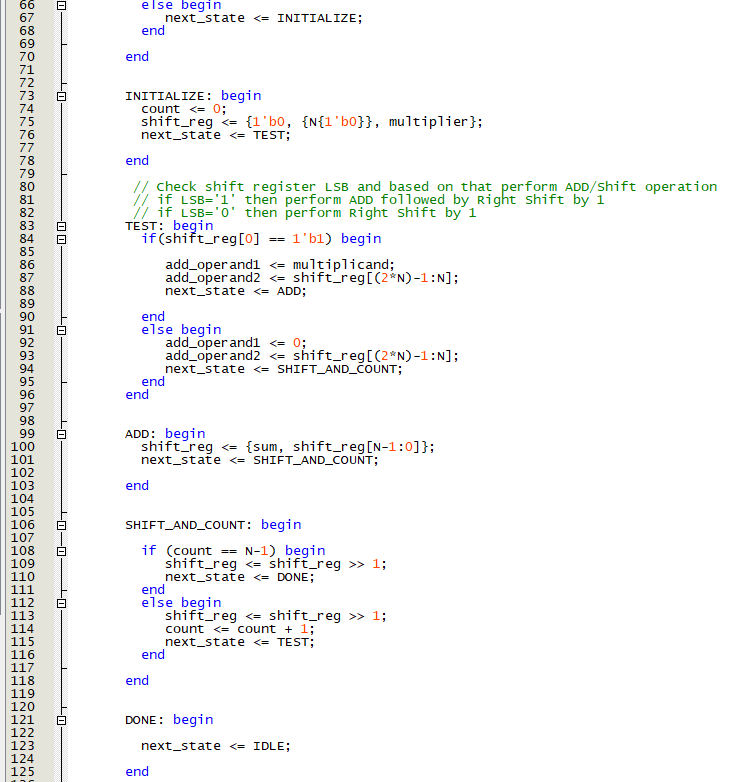
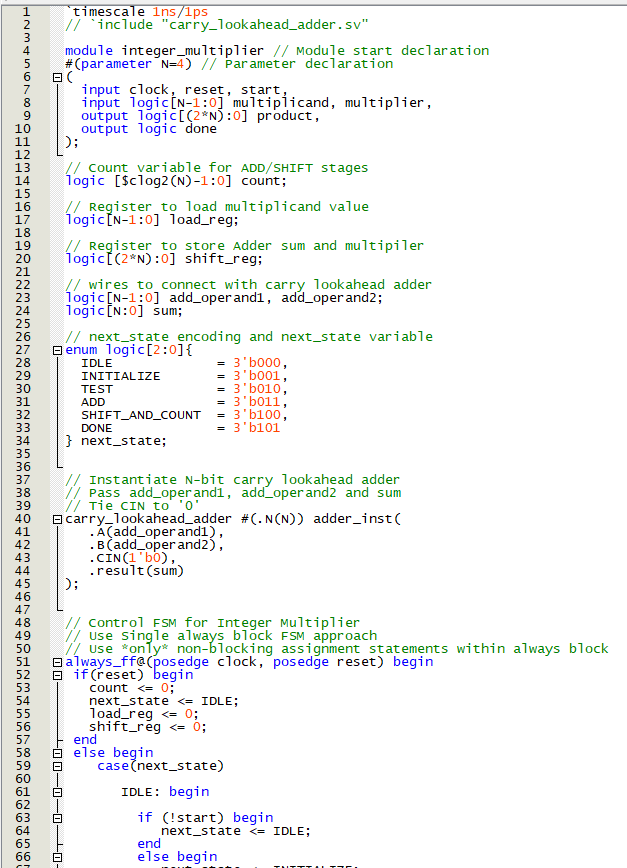
* A Mealy state machine generates an output based on the present state and the inputs; this differs from a Moore in that the output reacts immediately to a changing input
* In this implementation, flip flops were added to the inputs to delay them by half a clock cycle; this is to remedy the issue of double counting, especially during the third operation of the vending machine at around 260ns. As a side effect, the actual operation of this state machine is exactly like a Moore
* Without the flip flops, the inserted nickel would be counted three times, thus triggering the output before the dime even gets inserted which is incorrect behavior
* The “ideal” inputs would be the outputs of the registers
* Focusing on r\_N and r\_D as the inputs now, we see that at 50ns, the first nickel gets inserted, and the state is changed to 01 (5 Cents) at the positive clock edge
* Then, a dime is inserted, and the state changes to 11 (15 Cents) at the next positive clock edge
* We also observe that the output goes high as soon as the dime is inserted even though at that point, the present state is still 01 (5 Cents)
* This is because the Mealy machine knows the next state will be 15 Cents, so it triggers the output beforehand, saving a clock cycle.
* Described above is the correct behavior, assuming the inputs are timed correctly

**Integer Multiplier**

Code

Text

Description automatically generated



RTL netlist

Diagram, schematic

Description automatically generated

Resource usage

Graphical user interface

Description automatically generated with medium confidence

State diagram

A picture containing sky, different, group, bunch

Description automatically generated

Graphical user interface, table

Description automatically generated

Testbench simulation waveform

Graphical user interface

Description automatically generated

Graphical user interface

Description automatically generated

Text

Description automatically generated with low confidence

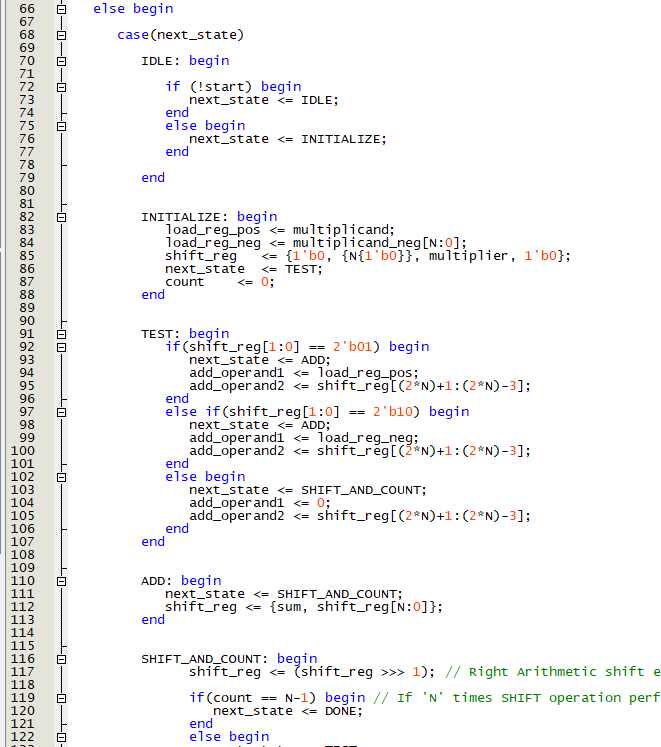
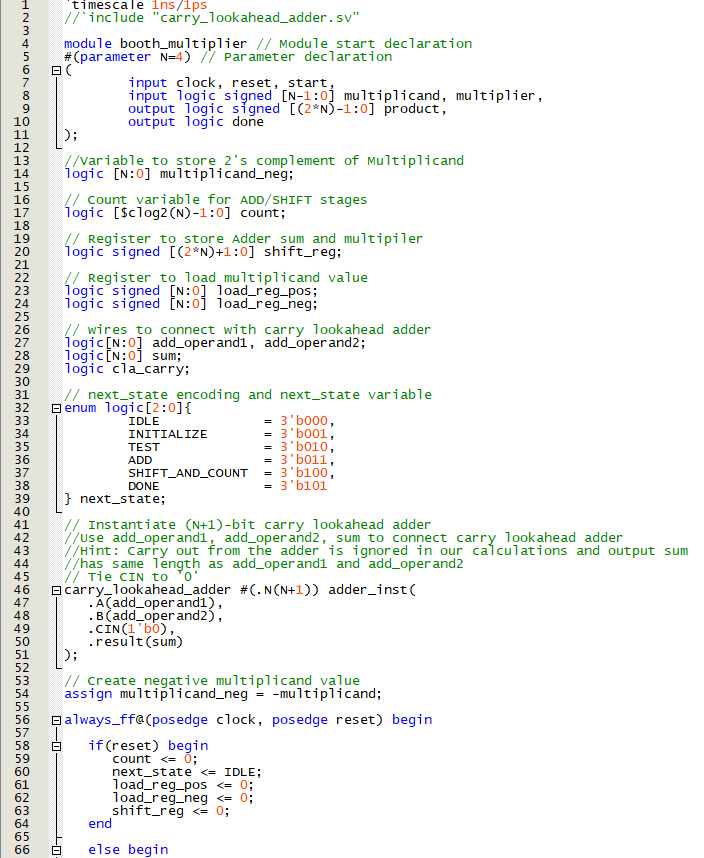
* The waveforms shown are of 3 views: the overall waveform, a zoomed in section to show the correct results from input pairs, and the transcript to demonstrate that each case yielded the correct result
* The multiplier behaves correctly because upon loading two numbers to be multiplied, and by setting start to high, this starts the internal cycle of states, and once the final product has been calculated, it is outputted with the done signal going high
* Then upon the next clock cycle, product resets to 0, and the multiplier idles until a new pair of numbers is loaded along with the pull up of the start input

**Booth Multiplier**

Code

Graphical user interface, text, application

Description automatically generated



RTL netlist

Diagram

Description automatically generated

Resource usage

Graphical user interface

Description automatically generated with medium confidence

State diagram

A diagram of a system

Description automatically generated with low confidence

Graphical user interface, table

Description automatically generated

Testbench simulation waveform

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface

Description automatically generatedTable

Description automatically generated with medium confidence

* The waveforms shown are of 3 views: the overall waveform, the zoomed in waveform to show 3 cases and their outputs, and the transcript to demonstrate that each case yielded the correct result
* The multiplier behaves correctly because upon loading two numbers to be multiplied, and by setting start to high, this starts the internal cycle of states, and once the final product has been calculated, it is outputted with the done signal going high
* Then upon the next clock cycle, product resets to 0, and the multiplier idles until a new pair of numbers is loaded along with the pull up of the start input