Vending Machine in Verilog & VHDL

Youssef Mouaddib

IEEE St. Mary's University

Spring 2023

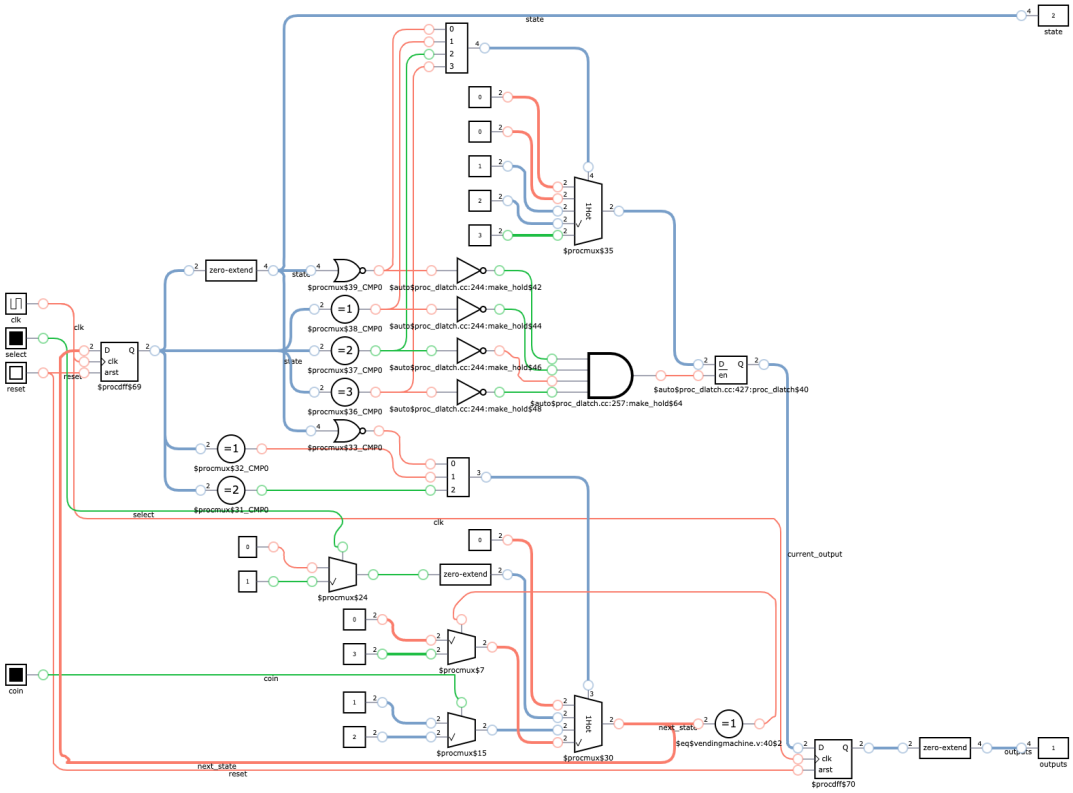


Figure a: The Vending Machine Simulation Output on DigitalJS

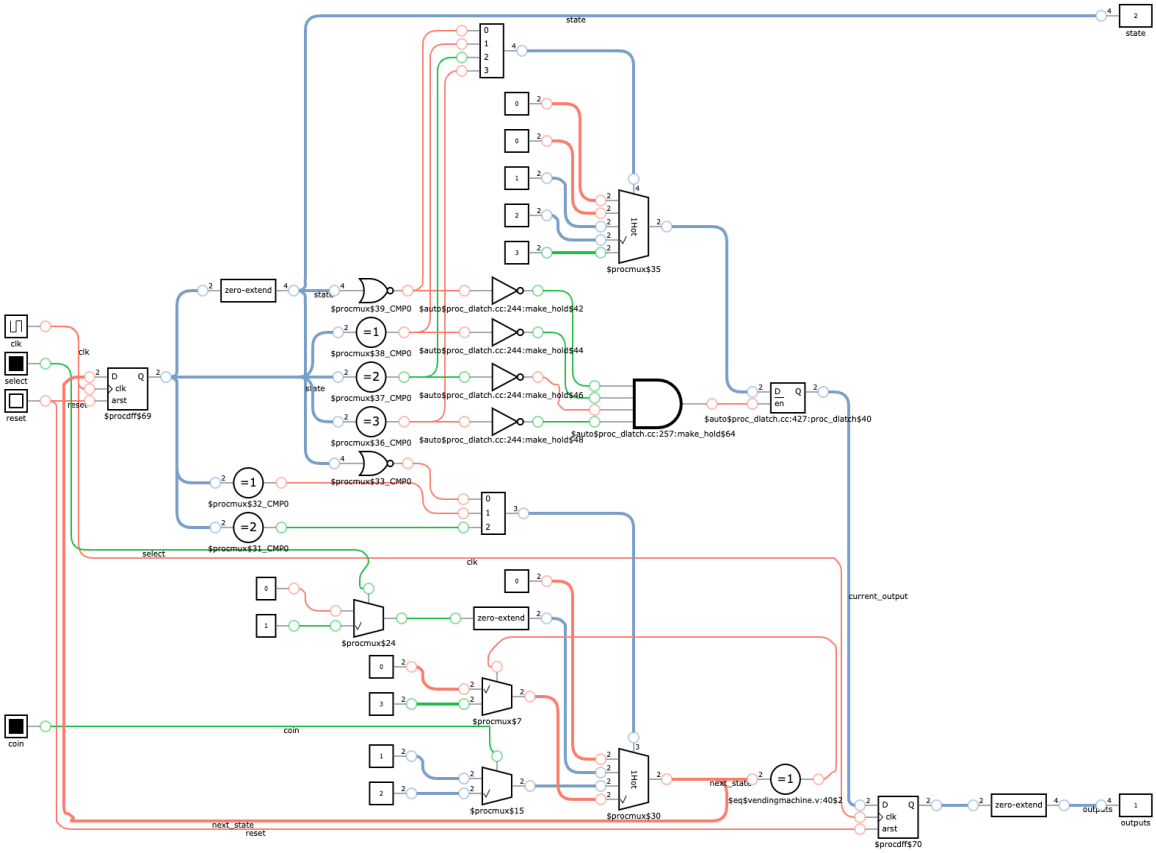
This project implements a simple digital logic vending machine using Verilog and DigitalJS. The vending machine has four states, which are defined as parameters in the code: S_IDLE, S_COLLECTING, S_DISPENSING, and S_CHANGE. The vending machine's state and output are controlled by two reg variables: next_state and current_output.

The logic for the vending machine is implemented in an always block that uses a case statement to determine what state the vending machine is in and what output should be displayed on the

vending machine's display. The vending machine transitions between states based on the inputs coin *and* select*. If the vending machine is in the* S_IDLE *state and the customer selects a product by pressing the* select *button, the vending machine transitions to the* S_COLLECTING *state. If the customer inserts enough coins to pay for the product, the vending machine transitions to the* S_DISPENSING *state and dispenses the product. If the vending machine is in the* S_DISPENSING *state, it transitions to the* S_CHANGE *state if the next state is* S_COLLECTING*, which means the customer has paid more than the price of the product and is owed change. Finally, if the vending machine is in the* S_CHANGE *state, it transitions back to the* S_IDLE *state.*

The vending machine's state and output are assigned in a second always *block that sets the vending machine's initial state and output to* S_IDLE *and 0, respectively, when the* reset *signal is high. Otherwise, the block updates the vending machine's state and output based on the values of* next_state *and* current_output*.*

This vending machine project could be useful for learning digital logic, Verilog, and DigitalJS. It could also be extended to include additional features, such as a product inventory, a coin dispenser, or a user interface with buttons and LEDs.


state
state
clk
select
reset
clk
D
clk
arst
Q
$procdff$69
zero-extend
state
$procmux$39_CMP0
$procmux$38_CMP0
$procmux$37_CMP0
$procmux$36_CMP0
$procmux$33_CMP0
$procmux$32_CMP0
$procmux$31_CMP0
$auto$proc_dlatch.cc:244:make_hold$42
$auto$proc_dlatch.cc:244:make_hold$44
$auto$proc_dlatch.cc:244:make_hold$46
$auto$proc_dlatch.cc:244:make_hold$48
$auto$proc_dlatch.cc:257:make_hold$64
$procmux$35
$auto$proc_dlatch.cc:427:proc_dlatch$40
D
en
Q
select
clk
current_output
$procmux$24
zero-extend
$procmux$7
coin
$procmux$15
$procmux$30
next_state
eqvendingmachine.v:40$2
next_state
reset
$procdff$70
zero-extend
outputs
outputs