Finite State Machine (FSM) Vending Machine – Verilog HDL Implementation

Technical Report & FPGA Deployment Results

Abid Ahmad

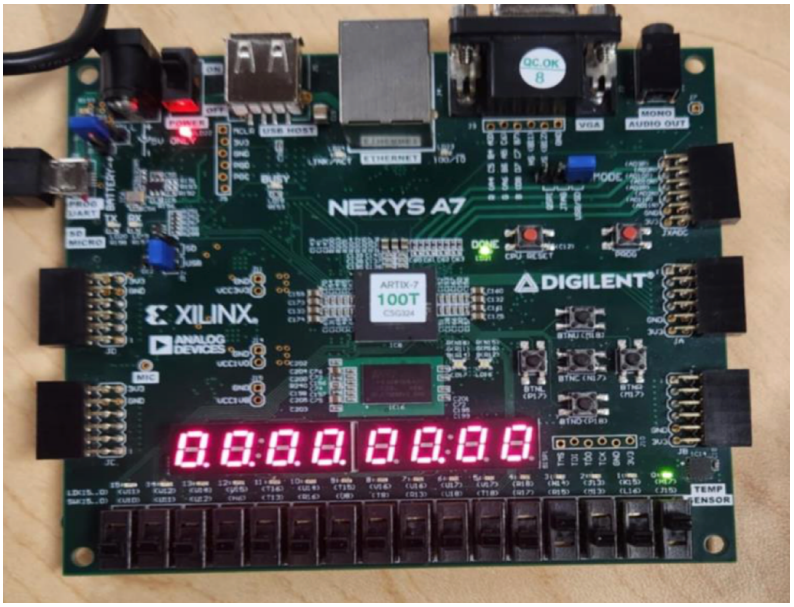
Degree: B.S. in Electrical and Computer Engineering, Wayne State University

Highlights:

Designed FSM from scratch for coin-based vending system

Implemented combo logic and LED feedback

Verified on Nexys A7 FPGA (Artix-7 100T)



*Full academic version with course details available in repository

Table of Contents

Abstract.....3

Introduction.....4

Summary.....4

Materials Used.....4

FPGA Switch Assignments.....4

FPGA LED Assignments.....4

Design.....5

- **State Diagram.....5**
- **Verilog Code.....6**
- **Test Bench.....10**
- **Top Module.....10**
- **Counter.....11**
- **disp_hex_mux.....11**
- **Timing Diagram.....12**
- **FPGA.....13**

Results.....21

Conclusion.....21

Abstract

The objectives of the project were to design a finite state machine (FSM) that simulates a vending machine and to implement the machine onto an FPGA board. The vending machine accepts up to 3 quarters, tracks and sums the amount inserted, and displays the total value in the 7-segment display of the FPGA board. The FSM transitions between 9 different states based on coin inputs and product selection. The FSM can return to its ideal or waiting state after a transaction so it can be used again without having to be manually reset.

The machine can show change and return a coin if funds are insufficient for purchase, at which point the machine can return to the ideal state. The machine can also dispense soda and candy at the same time if three-quarters are inserted. Furthermore, the machine is designed with a reset button which could be used to force-return the machine back to its ideal state. The machine is also designed to dispense chips of 0.75 cents. The FSM is Moore-type because the output of the system depends on the current states and not the primary inputs. The FSM is also synchronous, updating at every positive edge of the clock signal. For the initial design, a state diagram was created which was used to program and route everything to the appropriate state. The FSM is implemented using the Verilog HDL, and the FPGA was programmed and simulated using the Xilinx Vivado. The report includes an overview of the FSM design, the functionality of the FPGA and its hardware components, and the software simulation used to verify the system's design.

Introduction

a. Summary

Upon inserting the money, the FSM allows the user to select candy or soda. The FSM accepts and displays inputs of 25, 50, and 75 cents. Furthermore, the user has the option to buy a soda, candy, and chips. Depending on the amount inserted, FSM transitions through different states and communicates through the 7-segment display and LEDs which are programmed to three outputs: candy dispense, soda dispense, and coin ejection. Upon completing the transaction, the FSM returns to ideal state A, ready to be used again. Seven switches were assigned to the FPGA to manage primary inputs: three that each represent a quarter, one for candy, one for soda, one for chips and one for reset, which returns the FSM to the ideal state A.

Our FSM implementation consists of four flip-flops (FFs) to store the current state, with a total of 9 states. The goal of this design implementation was to create smooth transitions between the states and understand how FSMs are applied in real life.

b. Materials Used

1. **FPGA board**
2. **USB cable**
3. **Xilinx Vivado Software**

c. FPGA Switch Assignments

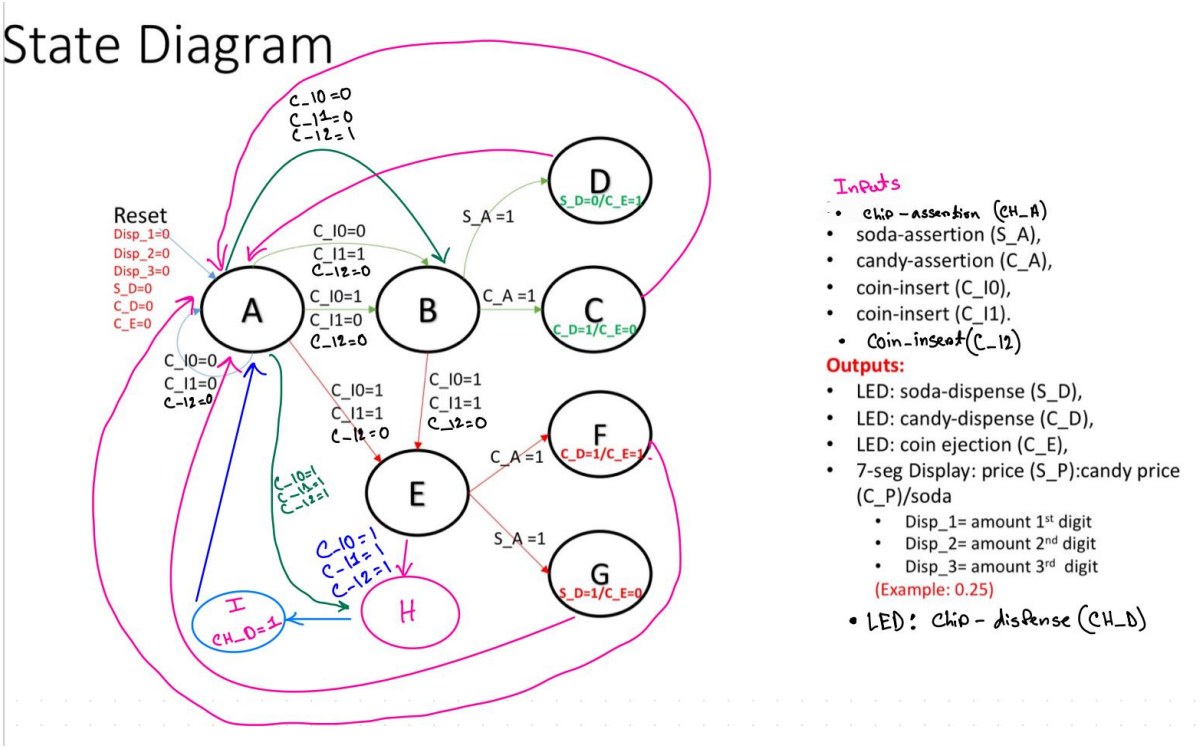
1. **J15 = coin C0 (25 cents)**
2. **L16 = coin C1 (25 cents)**
3. **M13 = coin C2 (25 cents)**
4. **R15 = candy assertion**
5. **R17 = soda assertion**
6. **T18 = chips assertion**
7. **V10 = reset**

d. FPGA LED Assignments

1. **H17 = candy dispensed**
2. **K15 = soda dispensed**
3. **J13 = chips dispensed**
4. **N14 = coin ejection**

Design

a. State Diagram



b. Verilog Code

```
285
286   module fsm(clock,reset,c0,c1,c2,c_a,s_a,ch_a,s_d,c_d,ch_d,c_e,disp_1,disp_2,disp_3,disp_4);
287   input clock,reset,c0,c1,c2,s_a,c_a,ch_a;
288   output reg [3:0] disp_1,disp_2,disp_3,disp_4;
289   output reg s_d,c_d,c_e,ch_d;
290   reg [3:0] y;
291   parameter [3:0] A=4'b0000, B=4'b0001, C=4'b0010, D=4'b0011, E=4'b0100, F=4'b0101, G=4'b0110, H=4'b0111, I=4'b1000;
292   wire [2:0] coin_input = {c2, c1, c0};
293
294   always @(posedge clock)
295
296   if (reset)
297   begin
298   y<=A;
299   s_d<=0; c_d<=0; c_e<=0;
300   disp_1=4'b0000; disp_2=4'b0000; disp_3=4'b0000; disp_4=4'b0000;
301   end
302
303   else
304   case (y)
305
306   A:
307
308   begin
309   disp_1=4'b0000; disp_2=4'b0000; disp_3=4'b0000; disp_4=4'b0000;
310   s_d <= 0; c_d <= 0; c_e <= 0; ch_d <= 0;
311
312   case (coin_input)
313
314   3'b100, 3'b010, 3'b001: begin
315   disp_1=4'b0101;
316   disp_2=4'b0010; //25 cents
317   disp_3=4'b0000;
318   y<=B;
319   end
```

The states of the FSM are listed in the parameters.

FSM is synchronous. The sensitivity list in line 294 communicates that the output is updated at every positive edge of the clock signal.

fsm.v

C:/Users/ho9306/OneDrive - Wayne State University/ECE-2610/HONORSPROJECT/HONORSPRO

```
317     disp_3=4'b0000;
318     y<=B;
319     end
320
321     3'b110, 3'b101, 3'b011: begin
322     disp_1=4'b0000;
323     disp_2=4'b0101; //50 cents
324     disp_3=4'b0000;
325     y<=E;
326     end
327
328     3'b111: begin
329     disp_1=4'b0101;
330     disp_2=4'b0111; //75 cents
331     disp_3=4'b0000;
332     y<=H;
333     end
334
335     default: y<=A;
336
337     endcase
338     end
339
340     B: begin
341     disp_1=4'b0101;
342     disp_2=4'b0010; //25 cents
343     disp_3=4'b0000;
344
345     case (coin_input)
346     3'b110, 3'b101, 3'b011: y<=E;
347     3'b111: y<=H;
348     default: begin
349     if (s_a)
350     y<=D;
351     else if (c_a)
```

```
351     else if (c_a)
352     y<=C;
353     else
354     y<=B;
355     end
356     endcase
357     end
358
359     C: begin
360     disp_1=4'b0000; disp_2=4'b0000; disp_3=4'b0000; disp_4=4'b0000;
361     c_d<=1;
362     c_e<=0;
363     y<=A;
364     end
365
366     D: begin
367     disp_1=4'b0000; disp_2=4'b0000; disp_3=4'b0000; disp_4=4'b0000;
368     s_d<=0;
369     c_e<=1;
370     y<=A;
371     end
372
373     E: begin
374
375     disp_1=4'b0000;
376     disp_2=4'b0101; //50
377     disp_3=4'b0000;
378
379     case (coin_input)
380     3'b111: y<=H;
381     default: begin
382     if (c_a)
383     y<=F;
384     else if (s_a)
385     y<=G;
```

```
381     default: begin
382     if (c_a)
383     y<=F;
384     else if (s_a)
385     y<=G;
386     else
387     y<=E;
388     end
389     endcase
390     end
391
392     F: begin
393     disp_1=4'b0101;
394     disp_2=4'b0010; //25 cents
395     disp_3=4'b0000;
396     c_d<=1;
397     c_e<=1;
398     y<=A;
399     end
400
401     G: begin
402     disp_1=4'b0000; disp_2=4'b0000; disp_3=4'b0000; disp_4=4'b0000;
403     s_d<=1;
404     c_e<=0;
405     y<=A;
406     end
407
408     H: begin
409     disp_1=4'b0101;
410     disp_2=4'b0111; //75 cents
411     disp_3=4'b0000;
412     if (ch_a)
413     y<=I;
414     else
415     y<=H;
```

```
414     else
415     y<=H;
416     end
417
418     I: begin
419     disp_1=4'b0000; disp_2=4'b0000; disp_3=4'b0000; disp_4=4'b0000;
420     ch_d<=1;
421     c_d<=0;
422     s_d<=0;
423     c_e<=0;
424     y<=A;
425     end
426
427     endcase
428     endmodule
```

c. Test Bench

```
module FSM_TB();

reg clk, reset, s_a,c_a,c0,c1;
wire [3:0]disp_1,disp_2,disp_3,disp_4;
wire  s_d,c_d,c_e;
parameter[2:0] A=3'b000, B=3'b001, C=3'b010,D=3'b011, E=3'b100, F=3'b101, G=3'b110;

FSM uut(clk, reset, s_a,c_a,c0,c1,disp_1,disp_2,disp_3,disp_4,s_d,c_d,c_e);

initial
begin
clk=0;
s_a=0;c_a=0;c0=0;c1=0;
reset=1;
end

always #10 clk=~clk;

initial begin
#20; reset=0;
#20;c0=1;
#20;c_a=1;
end
endmodule
```

The test bench tests the system's behavior for functionality. The test bench offers insight from the timing diagram to how the code will work in the FPGA.

d. Top Module

fsm_top.v

C:/Users/ho9306/OneDrive - Wayne State University/ECE-2610/HONORSPROJECT/HONORSPROJECT.srcs/sources_1/new/fsm_top.v

```
`timescale 1ns / 1ps

module fsm_top(
input wire clock,
input wire reset,
input wire s_a, c_a,ch_a, c0, c1, c2,
output wire [3:0]an,
output wire [7:0] sseg,
output wire s_d, c_d, c_e,ch_d
    );

//signal declaration
//wire [3:0] a,b;
wire [3:0] disp_1, disp_2, disp_3, disp_4;
wire new_clock;
//instantiate 7-segment LED Display module
disp_hex_mux disp_unit
(.clock(clock), .reset(reset), .hex3(disp_4), .hex2(disp_3), .hex1(disp_2), .hex0(disp_1), .dp_in(4'b1011), .an(an), .sseg(sseg));
//instantiate the control counter to lower down the 50MHz to 10Hz
cntr cntr_unit (.clock(clock), .out(new_clock));
//instantiate the BCD counter
fsm fsm_call (.clock(new_clock), .reset(reset),.ch_a(ch_a), .s_a(s_a), .c_a(c_a), .c0(c0), .c1(c1),
.c2(c2), .disp_1(disp_1), .disp_2(disp_2), .disp_3(disp_3), .disp_4(disp_4), .ch_d(ch_d), .s_d(s_d), .c_d(c_d), .c_e(c_e));
endmodule
```

The top module is used for instantiating the 7-segment display, counter, and FSM.

e. Counter file

```
module cntr(input wire clk ,output wire out);
// signal declaration
reg [31:0] r_reg;
wire [31:0] r_next;
// Register
always @ (posedge clk)
r_reg <= r_next;
//next state logic
assign r_next = (r_reg == 100000000) ? 0 : r_reg +1; //output logic
assign out = ( r_reg == 100000000) ? 1'b1 : 1'b0;
endmodule
```

The counter file divides the clock's frequency to make the operation of the system human readable. Without it, the machine will be so fast that it will be impossible to see its operations.

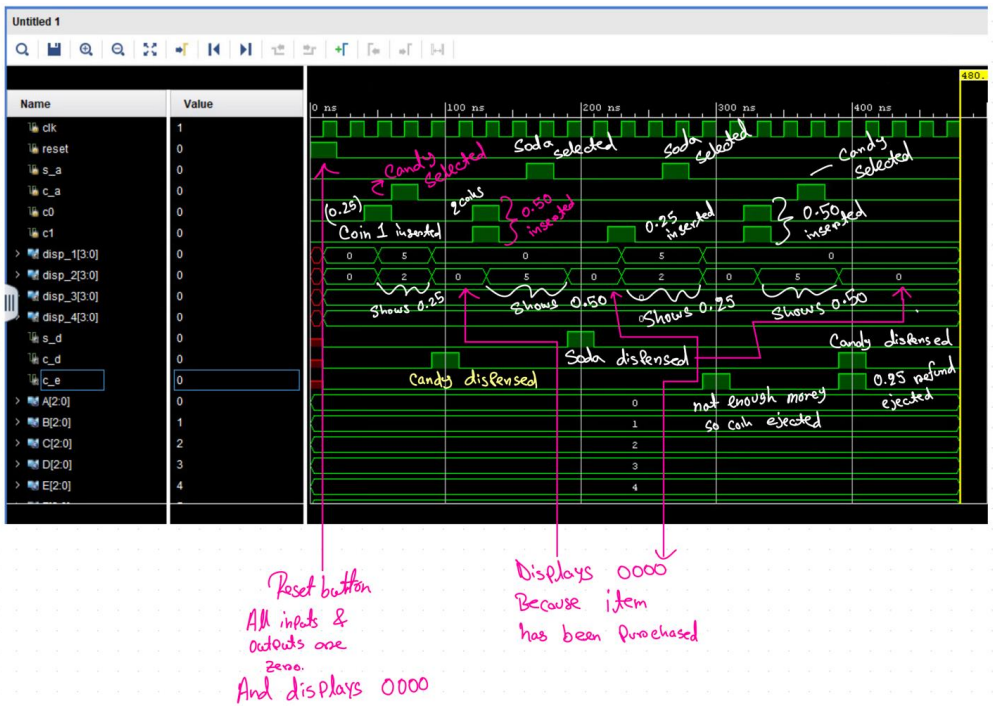
f. Disp_hex_mux module

```
module disp_hex_mux
(
input wire clk, reset ,
input wire [3:0] hex3 , hex2 , hex1 , hex0 , // hex digits
input wire [3:0] dp_in , // 4 decimal points
output reg [3:0] an , // enable 1-out-of-4 asserted low
output reg [7:0] sseg // led segments
);
// constant delaration
// refereshing rate around 800 Hz (50 MHz/2^16)
localparam N = 18 ;
// internal signal declaration
reg [N-1:0] q_reg ;
wire [N-1:0] q_next ;
reg [3:0] hex_in ;
reg dp ;
```

Note that some of the code was omitted to fit in the document. The purpose of this code is to drive a 4-digit 7-segment display. It takes 4 hexadecimal values and multiplexes them to show multiple values in the 7-segment display.

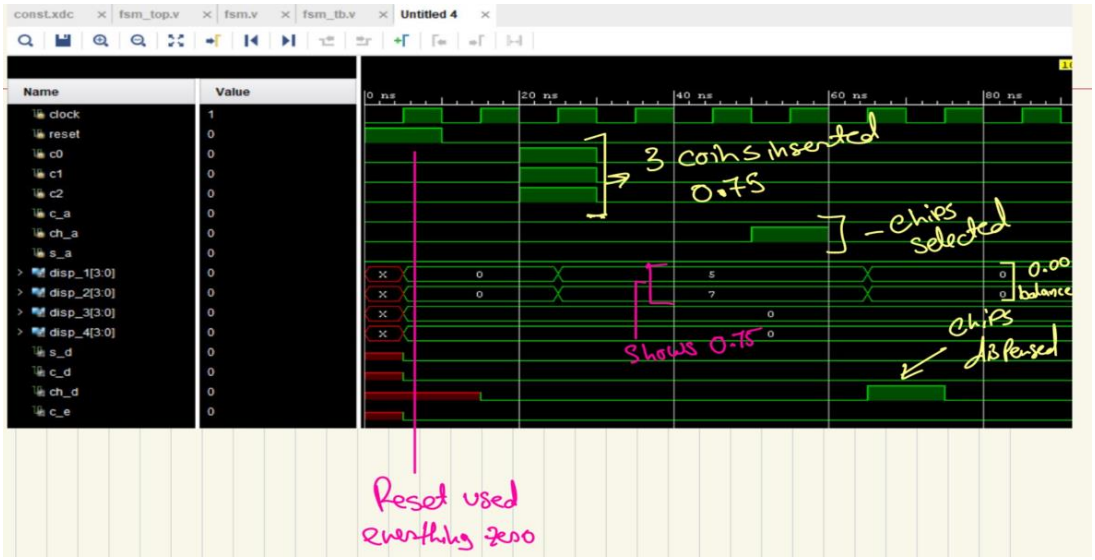
g. **Timing Diagram**

A)



The timing diagram shows how the system is operating.

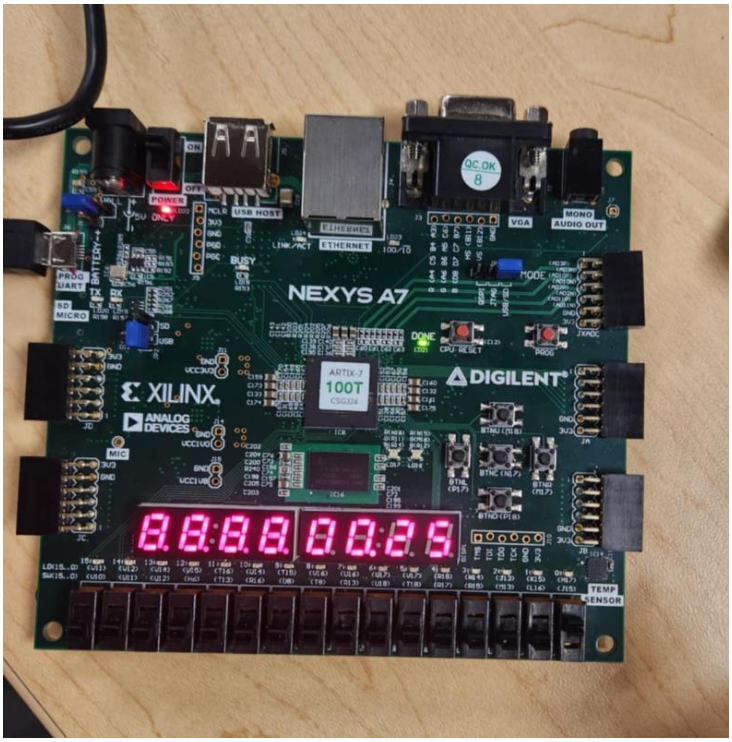
B)



For 0.75 cents and Chips

h. FPGA Implementation

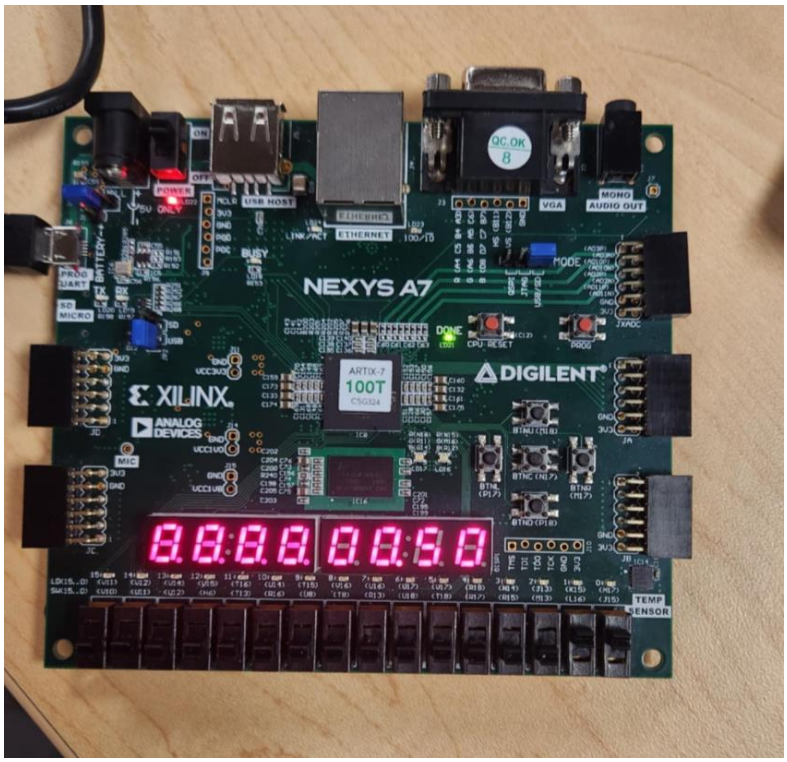
1.



State A.

From right-Switch 0 represents coin c0, switch 1 represents coin c1, and switch 2 represents coin c2. Only switch 0 is high; therefore 1 coin is inserted, and 25 cents is displayed.

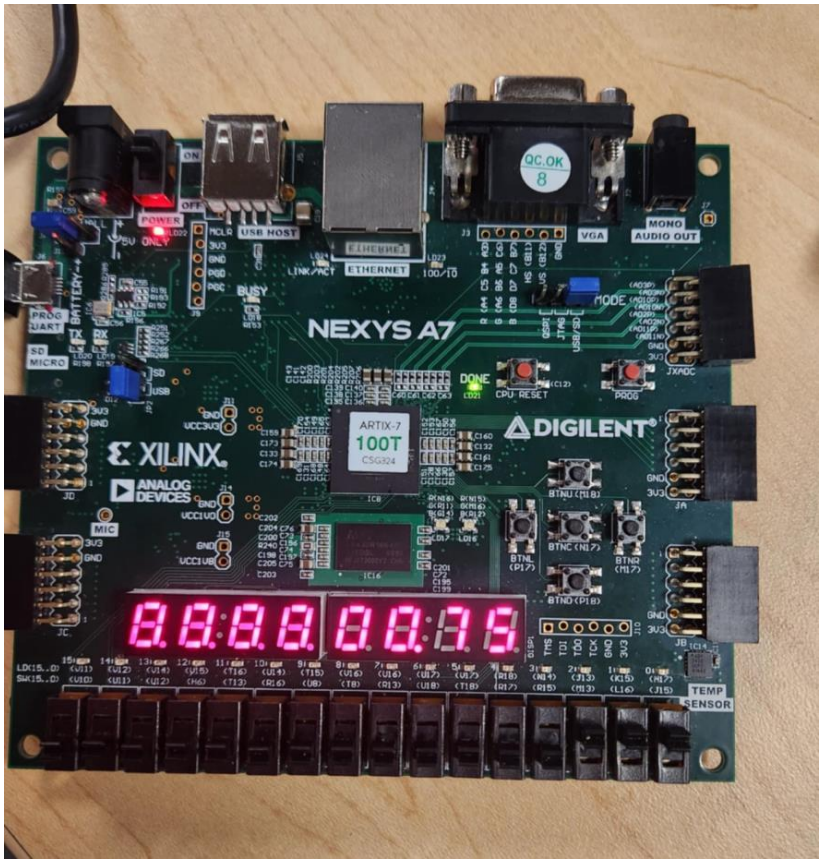
2.



State A.

Switches 0 and 1 are high; therefore 2 coins are inserted, and 50 cents is displayed.

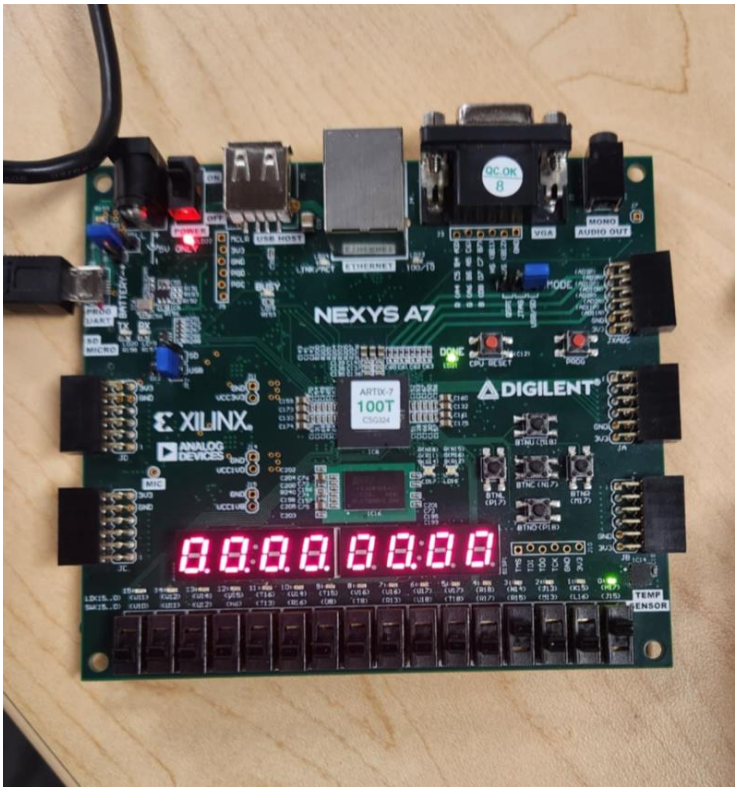
3.



State A.

Switches 0, 1, and 2 are high; therefore 3 coins are inserted, and 75 cents is displayed.

4.



State C.

Switch 3 (fourth from left) represents candy_assertion (i.e. a candy is selected).

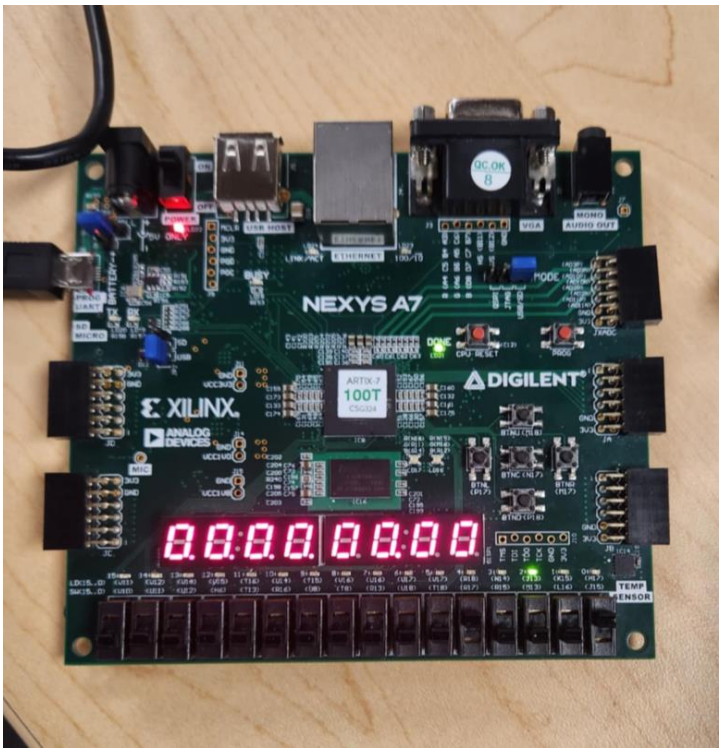
One candy cost 25 cents, and one soda cost 50 cents.

LEDs 0-3 represent candy_dispense, soda_dispense chips_dispense, and coin_ejection, respectively.

Switches 0 and 3 are high. 25 cents is inserted, and candy is selected. LED 0 is turned on to indicate a candy was dispensed. Zeros here were chosen to be displayed to show change after the transaction.

The LED goes off after a second, and zeros are displayed.

5.



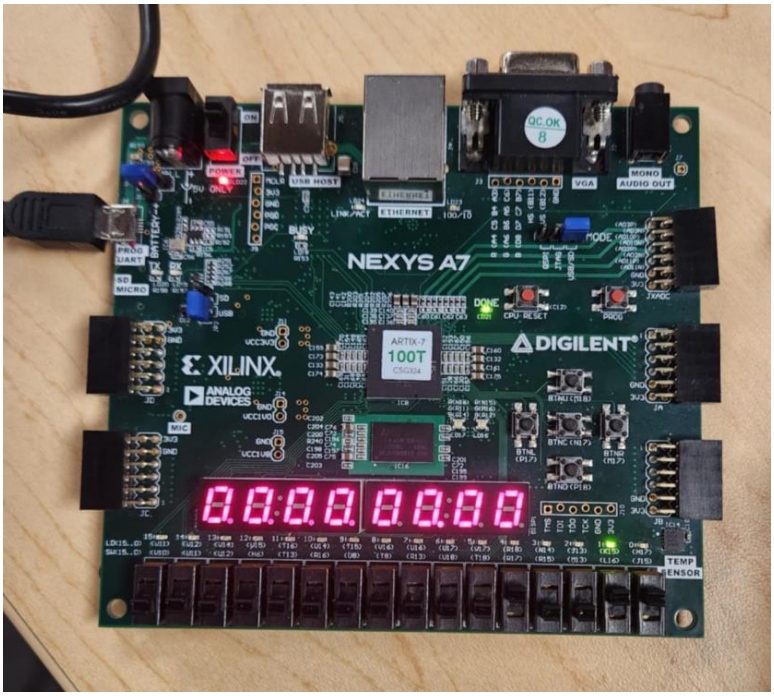
State D.

Switch 4 represents soda_assertion (i.e. a soda is selected).

Switches 0 and 4 are high. 25 cents is inserted, and a soda is selected. LED 2 is turned on to indicate the coin was ejected because of insufficient funds. Zeros here are displayed per assignment requirements.

The LED goes off after a second, and zeros are displayed.

6.

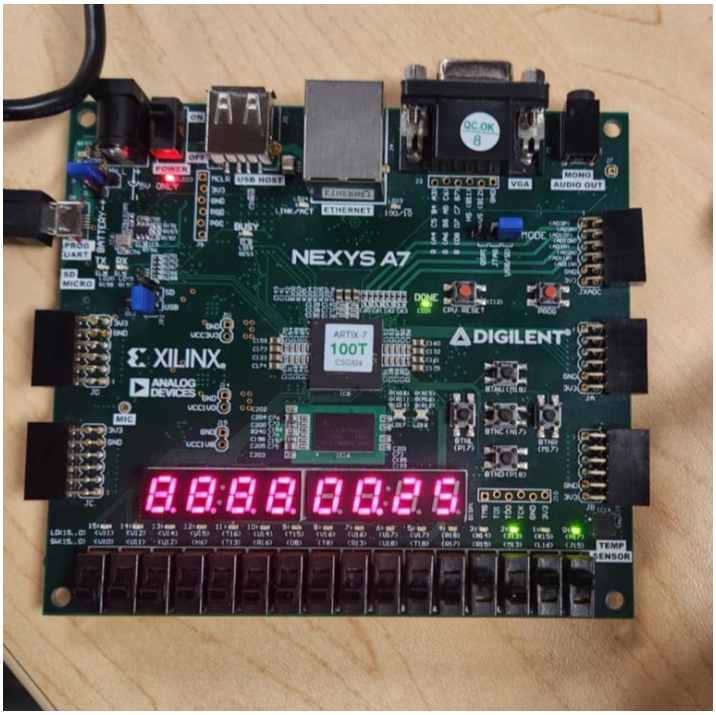


State G.

Switches 0, 1, and 4 are high. 50 cents is inserted, and a soda is selected. LED 1 is on to indicate a soda is dispensed. Zeros were chosen to be displayed to show change after the transaction.

The LED goes off after a second, and zeros are displayed.

7.

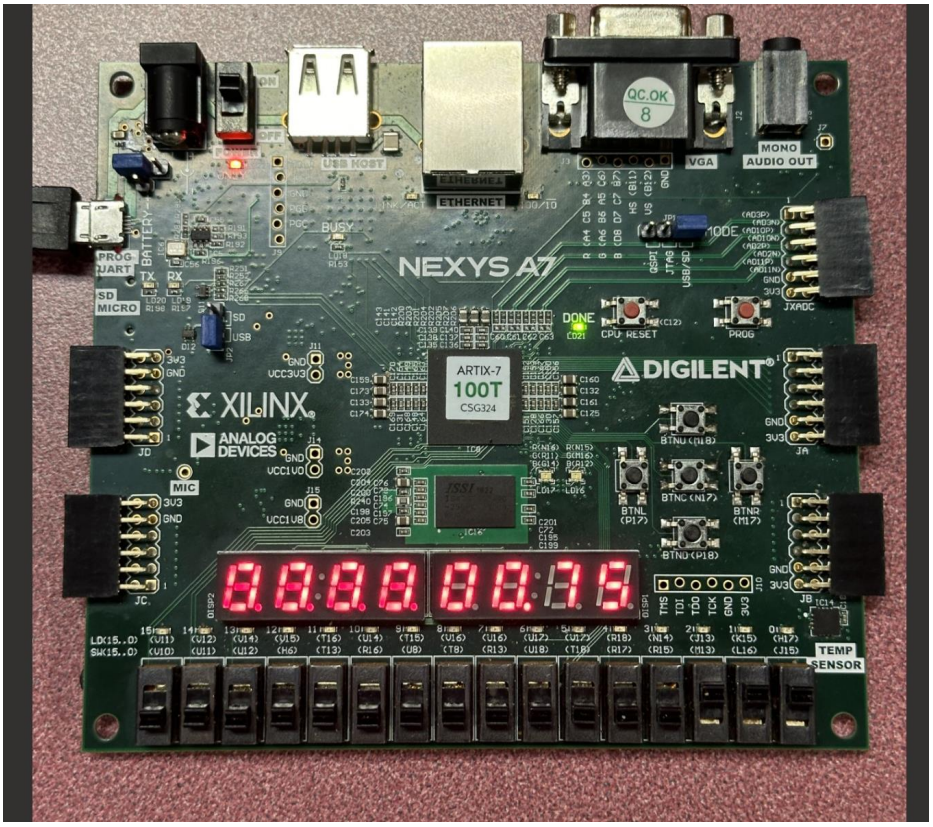


State F.

Switches 0, 1, and 3 are high. 50 cents is inserted, and candy is selected. LEDs 0 and 2 are on to indicate a candy is dispensed and a coin is ejected. 25 is displayed to show change after the transaction.

The LEDs go off after a second, and zeros are displayed.

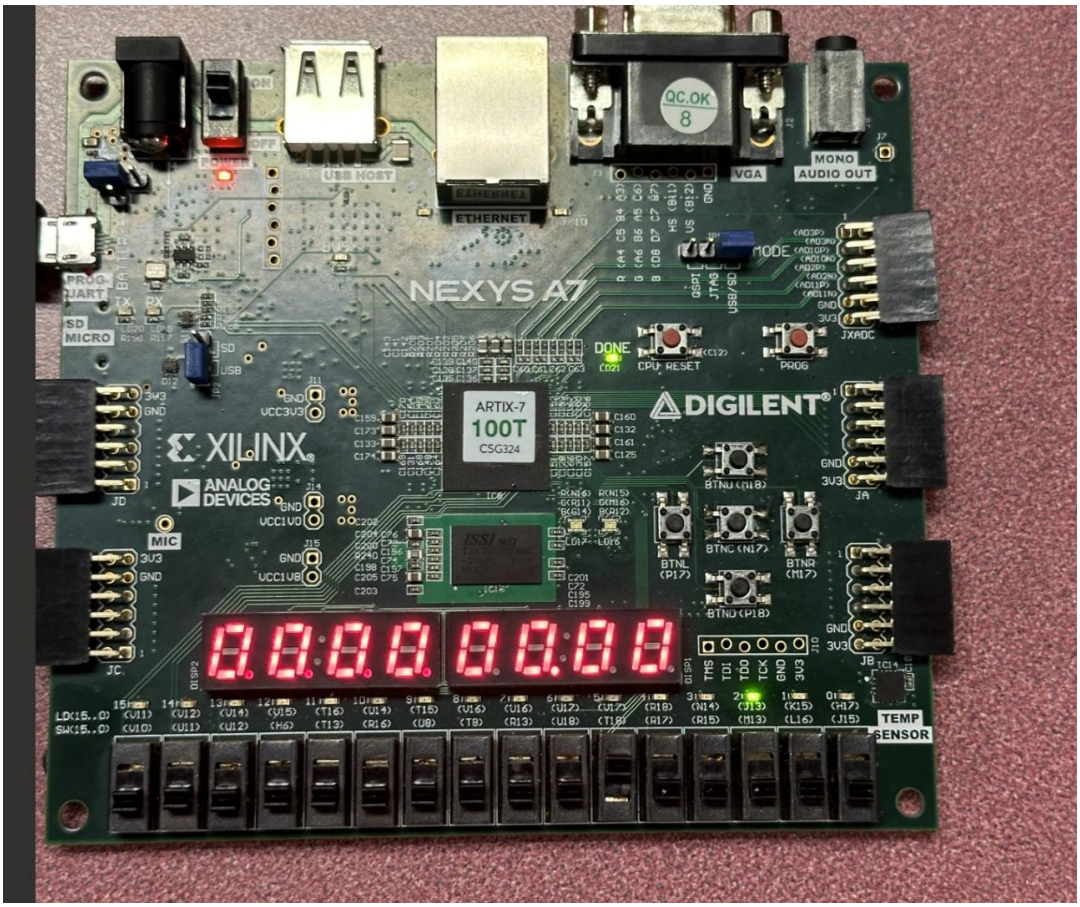
8.



State H.

It shows that $0.75 has been inserted and it is waiting for response

9.



State I.

The chips assertion button T18 is high, and we know chips have been dispensed because J13 LED is high which indicates chips has been dispensed. The LEDs go off after a second, and zeros are displayed.

Results

Design section h shows the implementation of the FPGA and how it communicates with the user. Some things to note here, per the state diagram in design section a:

- 25 will display regardless of which coin you choose if only 1 is chosen. If any 2 coins are high, 0.50 will display. When 3 coins are inserted. 0.75 will display.
- Any combination of coins will work with product assertions so that the outputs in design section c, parts 4-8 will not change. This indicates smooth transitions between states A, B, E, and H.
- With amounts ranging 0.25 to 0.75, 3 items can be purchased, and they are chips, candy, and soda.

Part of our design intent is to be able to make the machine return to state A, so the user can repeatedly use the machine without needing to use the reset button.

In design section h, parts 4-8, it is noted that the LEDs go off after a second, and zeros are then displayed. This is how we indicate that the FSM has returned to state A, as was intended from the state diagram in design section A.

For this to work, the user must return the coin switches to low before selecting their products. After the machine communicates by LED, the user should return the product switches to low. The user is indeed in state A because they can begin using the coin switches, where the FPGA will display the amount inserted accordingly. Then, selections can be made and the FPGA functions again as in design section h parts 4-8.

Conclusion

The FSM created is of Moore type because the output is a function of the current state of the machine and is not directly affected by the input. This is verified in design section part a because the output is updated once the machine is in states C, D, F, G, or I, and the output is not a function of inputs. The outputs are updated at the positive edge of the clock signal. Overall, the project provided a better understanding of FSMs and how to know whether you are working with a Moore or Mealy-type FSM. A state diagram is a great way to be able to tell if you do not have access to the circuit diagram.