CMPE 212

Spring, 2016

Project 2 Report

Sabbir Ahmed

Teaching assistant:

Ahmed Shahin

1. **Project description:**

The goal of this project is to design and implement a finite state machine through combinational and sequential circuits. With one 1-bit input corresponding to 25 ₵ or 10 ₵, the machine is to behave similarly to a vending machine dispensing an output after an amount of 70 ₵ or higher has been accumulated, except when the excess is either 5 ₵ or 15 ₵, in which case a change of 5 ₵ will also be outputted leaving the remaining as credit for the next simulation.

1. **Process:**
   1. **State diagram:**

The following state diagram was derived from the description:

C:\Users\Sabbir\Desktop\documents-export-2016-05-07\proj2.png

**State diagram for a vending machine finite state machine with outputs y0, y1**

The states were verified by an executable Python script, vending\_machine\_fsm.py attached to the report in the src directory. The script is written in Python 2.7 syntax.

**Usage:**

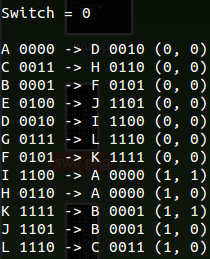
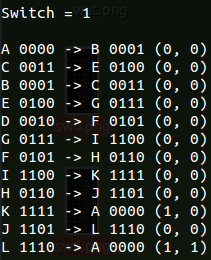
python vending\_machine\_fsm.py

Systems with a python2 compiler as default:

python2.x vending\_machine\_fsm.py

Systems with a newer Python compiler:

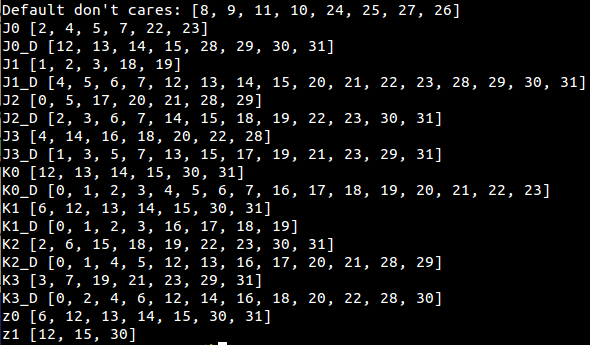
The script generates the following outputs:

**12 states with their corresponding code assignments**

* 1. **Functions:**

The script was also used to determine the minterms for each of the Boolean functions for the 4 flip flops and 2 outputs:



**Minterms for all the functions generated by simulating the state machine**

**(variables with ‘\_D’ represent don’t cares)**

The minterms were then used to reduce functions through use of Karnaugh maps:

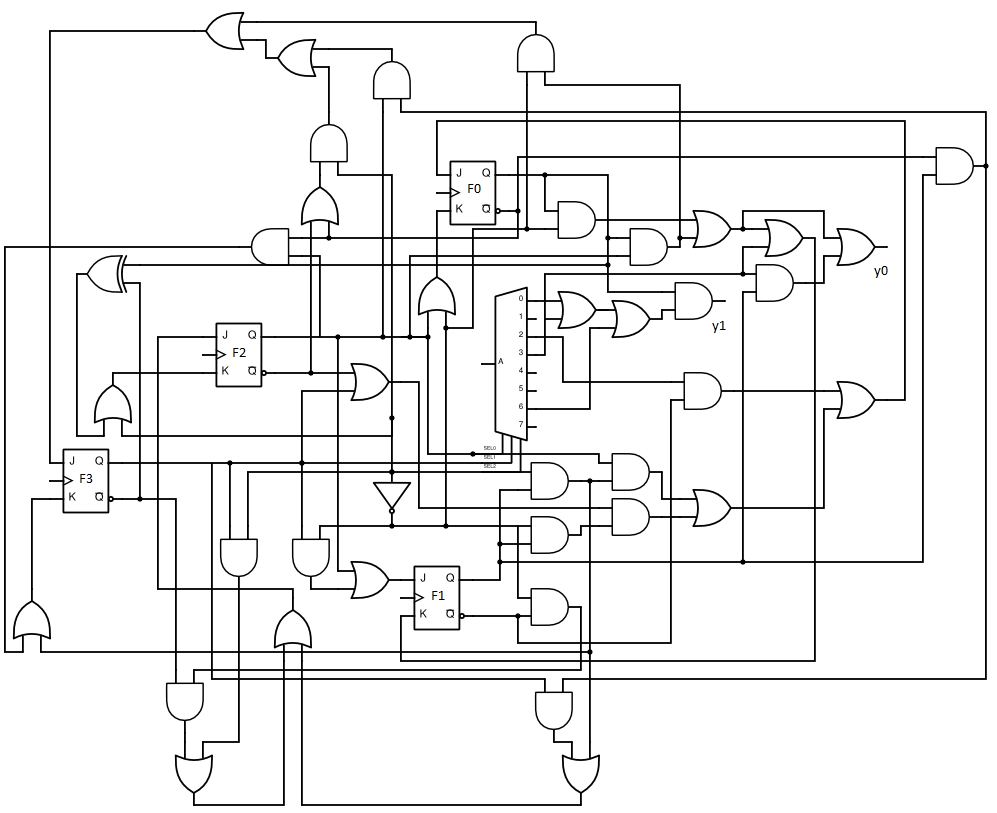
|  |  |  |
| --- | --- | --- |
| C:\Users\Sabbir\Desktop\ay-2016-05-08\ay\j0.png | | C:\Users\Sabbir\Desktop\ay-2016-05-08\ay\k0.png |
| j0 = | | k0 = |
| C:\Users\Sabbir\Desktop\ay-2016-05-08\ay\j1.png | C:\Users\Sabbir\Desktop\ay-2016-05-08\ay\k1.png | |
| j1 = | k1 = | |

|  |  |  |  |
| --- | --- | --- | --- |
| C:\Users\Sabbir\Desktop\ay-2016-05-08\ay\j2.png | | C:\Users\Sabbir\Desktop\ay-2016-05-08\ay\k2.png | |
| j2 = | | k2 = | |
| C:\Users\Sabbir\Desktop\ay-2016-05-08\ay\j3.png | | C:\Users\Sabbir\Desktop\ay-2016-05-08\ay\k3.png | |
| j3 = | | k3 = | |
|  | |  | |
| C:\Users\Sabbir\Desktop\ay-2016-05-08\ay\y0.png | | C:\Users\Sabbir\Desktop\ay-2016-05-08\ay\y1.png |
| y0 = | | y1 = |

* 1. **Design:**

After several Boolean function manipulation, a design for the circuit was realized. Since the number of flip flops can be found by computing , 4 JK flip flops were used. A decoder was used as well to generate 4 3-bit minterms that were used multiple times.

The following schematic was designed and used as the working blueprint during the implementation:



**Circuit design of the vending machine finite state machine**

* 1. **Verification and simulation:**

The logic of the entire circuit was verified using Verilog. Several testbenches have been included along with a makefile for convenience. The scripts were compiled using Icarus Verilog (iverilog) successfully, and ncverilog on GL, where it showed occasional runtime errors.

**Usage:**

make jk

Compile testbench for JK flip flop module:

make compile

Compile testbench for the machine:

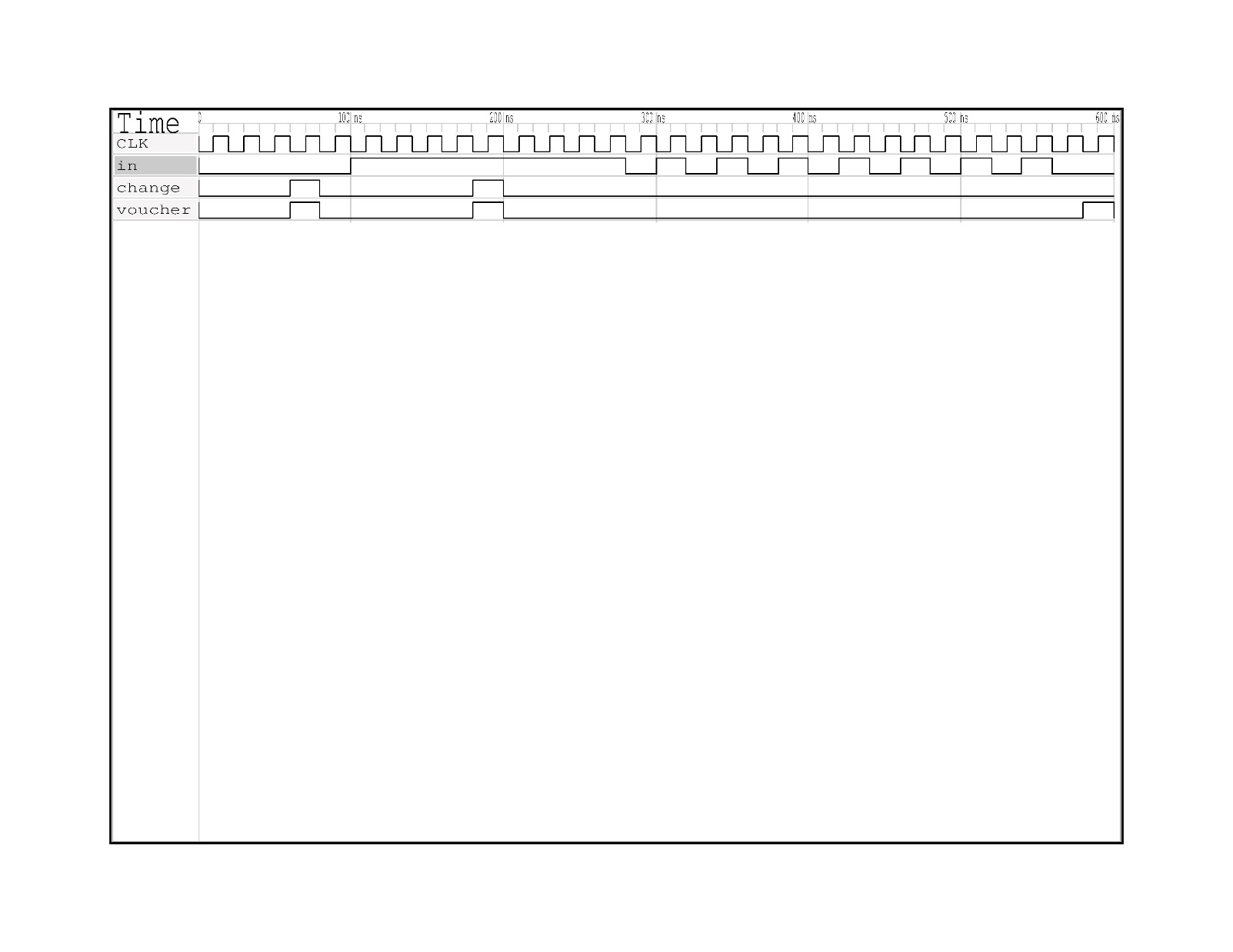
make run

Run executable:

make simulate

Simulate the testbench (after running script):

The following was simulated using the main testbench:



**Timing diagram simulated on GTKWave on a Debian Linux system**

1. **Conclusion:**

After the designing processes were completed, the circuit was constructed on a breadboard, and the logic above was verified again. Due to possible pin mapping errors and/or flawed logic, the state machine was not able to be implemented correctly and debugging the entire machine was not possible.