A Self-Checking Approach for FSM Based Design on the Replication of One-Hot Code

Modi Mahendra  
Department of VLSI & ESD,  
SKR College of Engineering & Technology,  
Konduru Satram [V], Manubolu [M],  
S.P.S.R.Nellore [Dt], Andhra Pradesh, India.

Y.V.Krishna Reddy  
Associate Professor,  
Department of VLSI & ESD,  
SKR College of Engineering & Technology,  
Konduru Satram [V], Manubolu [M],  
S.P.S.R.Nellore [Dt], Andhra Pradesh, India.

Abstract:

As technology scales, the protection of Finite State Machines’ (FSMs) states against single event upset (SEU) and multiple bit upsets (MBUs) becomes more difficult. In this paper, a self-checking approach to enhance the SEU/MBUs immunity of FSMs’ states by replicating One-Hot code times for state encoding is presented. This approach can correct less than bit-flip faults in the state register per cycle. Bit-flips are treated as random events and modeled by applying Poisson distribution. Two characteristics of this approach are obtained through probability analysis: first, this approach performs better with the increase of $n$, whereas worse when an FSM contains more states; second, this approach can offer more enhanced reliability than Binary or One-Hot state encoding with Triple Modular Redundancy (TMR). The former characteristic leads to the further improvement of this approach which is called state-reforming. The reliabilities of this proposed approach and its state-reformed solutions, as well as are all evaluated through simulations of fault injections.

Index Terms:

FSMs, MBUs, mean-time-between-failures, Poisson distribution, SEU.

I. INTRODUCTION:

Vending Machines are used to dispense various products like Coffee, Snacks, and Cold Drink etc. when money is inserted into it. Vending Machines have been in existence since 1880s. The first commercial coin operated machine was introduced in London and England used for selling post cards.

In this paper a new approach is proposed to design an FSM based Vending Machine with auto-billing features. The machine also supports a cancel feature means that the person can withdraw the request and the money will be returned back to the user. The user will get a bill of total number of products delivered with total price. This machine can be used at various places like Hotels, Restaurants and food streets. This reduces the time and cost.

1.1.OBJECTIVE :

In This Project I Implement the self-checking approach to enhance of FSMs’ states is presented. This approach replicates One-Hot code times for state encoding and has the ability to correct less than bit-flip faults in the state register per cycle. In This Project I Improved solutions state-reforming is proposed, which combine several One-Hot codes and then replicate them for state encoding. State-reforming solutions can offer higher reliabilities by reducing the number of flip-flops needed for storing state values. In this paper the process of four state (user Selection, Waiting for money insertion, product delivery and servicing) has been modeled using MEALY Machine Model.

1.2.MOTIVATION:

Nowadays, Vending Machines are well known among Japan, Malaysia and Singapore. The quantity of machines in these countries is on the top worldwide. This is due to the modern lifestyles which require fast food processing with high quality. This paper describes the designing of multi select machine using Finite State Machine Model with Auto-Billing Features. Finite State Machine (FSM) modeling is the most crucial part in developing proposed model.
The vending machines are more accessible and practical than the convention purchasing method. Nowadays, these can be found everywhere like at railway stations selling train tickets, in schools and offices vending drinks and snacks, in banks as ATM machines and provides even diamonds and platinum jewelers to customers. The FPGA based machine is also more flexible, programmable and can be re-programmed. But in microcontroller based machine, if one wants to enhance the design, he has to change the whole architecture again but in FPGA user can easily increase the number of products.

The documentation of the project mainly consist two parts one is design FSM machine and second one is draw the state diagram of one hot vending machine. Implementation of the project using ModelSim 6.5e and synthesis using XILINX ISE 10. This documentation helps one to understand the complete functional verification process of complex ASICs and SoC’s and it gives opportunity to try the latest verification methodologies, programming concepts like Object Oriented Programming of Hardware Verification Languages and sophisticated EDA tools, for the high quality verification. The last decade assertion-based verification (ABV) proved itself to be an efficient verification methodology based on instrumenting the design with assertions. These assertions are checked by methods including simulation, emulation, and formal verification. Assertions also document the design and play the role of executable comments.

2. SYSTEM DESIGNING

Existing System:

• Binary Encoding
• Gray encoding

2.1 Binary Encoding:

The Binary Method for assigning states is a common method to use. It counts up in binary starting from 0 and going up assigning each state the next number. It will use \( \log_2 \) bits to assign the states. Binary is a common method to use because it is easy to think of what the bit code will be for each state. Students often learn binary coding as their first encoding sequence because of this reason. However, it is very inefficient. There are a minimum number of bits used to encode the machine, but the encoding equations derived for each bit are often large and complex.

The equations derived for table 1 is:

\[
Q_0^+ = (Q_0^' * Q_1) + (X * Q_0') + (Q_0 * Q_1' * X') \\
(2.1)
\]

\[
Q_1^+ = (X'^* Q_1') + (X'* Q_0) + (X * Q_0'* Q_1) \\
(2.2)
\]

Because of this reason, different methods have been created that shorten the glue logic needed between the different gates.

2.1.2 Gray Encoding:

The reflected binary code, also known as Gray code after Frank Gray, is a binary numeral system where two successive values differ in only one bit (binary digit). The reflected binary code was originally designed to prevent spurious output from electromechanical switches. Today, Gray codes are widely used to facilitate error correction in digital communications such as digital terrestrial television and some cable TV systems. The binary-reflected Gray code list for \( n \) bits can be generated recursively from the list for \( n-1 \) bits by reflecting the list, concatenating the original list with the reversed list, prefixing the entries in the original list with a binary 0, and then prefixing the entries in the reflected list with a binary 1. For example, generating the \( n = 3 \) list from the \( n = 2 \) list:

**Table 2.1 Example Gray Code**

The one-bit Gray code is \( G_1 = (0, 1) \). This can be thought of as built recursively as above from a zero-bit Gray
2.2. PROBLEMS IN EXISTING METHOD:

If we design a complex controllers it contains the different (2 or 3) FSM blocks If we use binary Or gray code state encoding methods the circuit is not working properly because same state encoding techniques are used here. They can take a long time to compile the entire program. Also the dependency issues can be slightly more difficult to decipher. They are big files, and take up more space. They are usually enabled with all the options on the install, and decipher. They are big files, and take up more space. They can take a long time to compile the entire program. Also cause same state encoding techniques are used here. They - encoding methods the circuit is not working properly be (2 or 3) FSM blocks If we use binary Or gray code state encoding methods the circuit is not working properly be (2 or 3) FSM blocks If we use binary Or gray code state encoding methods the circuit is not working properly be (2 or 3) FSM blocks If we use binary Or gray code state encoding methods the circuit is not working properly be (2 or 3) FSM blocks If we use binary Or gray code state encoding methods the circuit is not working properly be (2 or 3) FSM blocks If we use binary Or gray code state encoding methods the circuit is not working properly be (2 or 3) FSM blocks If we use binary Or gray code state encoding methods the circuit is not working properly be

2.3. PROPOSED METHOD

One-Hot Encoding

Using the standard method, the following equations can be easily derived by looking at the 1 on each next state. For state N0, the 1’s are on rows 3 and 6. By looking at the current states and the inputs on those rows, the equation for N0 is:

\[ N0 = (X' * Q0' * Q1 * Q2' * Q3') + (X * Q0' * Q1' * Q2 * Q3') \]  

(2.3)

By using the same method, the equations for the other next states are:

\[ N1 = (X' * Q0' * Q1' * Q2 * Q3') + (X * Q0' * Q1' * Q2' * Q3) \]  

(2.4)

\[ N2 = (X' * Q0' * Q1' * Q2 * Q3') + (X' * Q0 * Q1' * Q2' * Q3') \]  

(2.5)

\[ N3 = (X' * Q0' * Q1' * Q2' * Q3') + (X * Q0' * Q1' * Q2 * Q3') \]  

(2.6)

Only one of these states are Hot at any given time. This means if Q0 is 1, then Q1, Q2 and Q3 are all 0. Likewise with all bits, if one is a 1, then all others are 0. Because of this, the equations can be reduced to:

\[ N0 = (X' * Q1) + (X * Q2) \]  

(2.7)

\[ N1 = (X' * Q2) + (X * Q3) \]  

(2.8)

\[ N2 = (X' * Q3) + (X' * Q0) \]  

(2.9)

\[ N3 = (X' * Q1) + (X * Q0) \]  

(2.10)

<table>
<thead>
<tr>
<th>Input</th>
<th>Current State</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Q0</td>
<td>Q1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table2.2: FSM State Encoding Transition Table

This type of encoding uses more bits to encode the states, but reduces the logic needed to glue the bits together. It is a very common coding to use, especially for many states.
One Hot Encoding is an encoding sequence that uses as many registers as there are states. With One Hot Encoding, only one of the bits are 1 or “Hot” at any given state. All the other bits are 0. This makes sure only two bits change when moving from one state to the next. With a maximum of two bits changing at any given time less power is consumed. In addition, One Hot Encoding reduces the logic needed to implement each bit. However, it also uses many more logic gates to implement the state design. One Hot is often used where there are many states that need to be implemented in the design. Using binary encoding, the complexity of the design grows with each bit added. With One Hot, the number of flip-flops grows with each state added, but the complexity of each equation does not. Because of this, it is more difficult to accidentally mess up the logic needed to implement the bits. K-Maps are not needed for this type of encoding either. Instead, a truth table is created and used to find the equation for each bit.

In this paper a new approach is proposed to design an FSM based Vending Machine with auto-billing features. The machine also supports a cancel feature means that the person can withdraw the request and the money will be returned back to the user. The user will get a bill of total number of products delivered with total price. This machine can be used at various places like Hotels, Restaurants and food streets. This reduces the time and cost.

3. ONE HOT - VENDING MACHINES:

Vending Machines are used to dispense various products like Coffee, Snacks, and Cold Drink etc. when money is inserted into it. Vending Machines have been in existence since 1880s. The first commercial coin operated machine was introduced in London and England used for selling post cards. The vending machines are more accessible and practical than the convention purchasing method. Nowadays, these can be found everywhere like at railway stations selling train tickets, in schools and offices vending drinks and snacks, in banks as ATM machine and provides even diamonds and platinum jewelers to customers. The FPGA based machine is also more flexible, programmable and can be re-programmed. But in microcontroller based machine, if one wants to enhance the design, he has to change the whole architecture again but in FPGA user can easily increase the number of products.

3.2. OPERATION OF VENDING MACHINE:

I. When the user puts in money, money counter tells the control unit, the amount of money inserted in the Vending Machine.
II. When the user presses the button to purchase the item that he wants, the control unit turns on the motor and dispenses the product if correct amount is inserted.
III. If there is any change, machine will return it to the user.
IV. The machine will demand for servicing when the products are not available inside the machine.
Money is an in/out signal which can be updated with the total money of all products delivered at a time. Money signal is seven bits wide. Money_count is an internal signal which can be updated at every transition. This signal is also seven bits wide. If the inserted money is more than the total money of products then the change will be returned through the change output signal. The products with their prices are shown by table 1. There are also two input signal clk and reset. The machine will work on the positive edge of clock and will return to its initial state when reset button is pressed. The proposed vending machine is designed using FSM modeling and is coded in Verilog HDL language.

### 4. DESIGN METHODOLOGY:

The state diagram mainly consists of four states (User Selection, Waiting for the money insertion, product delivery and servicing (when product_not_available='1')). Initially when the reset button is pressed, the machine will be ready for the users to select the product. This state is the initial state of the design. After this the user will select the product to be dispensed. This state can be one of the select1, select2, select3 and select 4. The machine can accept only two types of notes i.e. rupees 1/- and 2/-. Let us suppose that the user selects sel1 input. The machine will firstly check that whether the products are available in the machine or not. After this the control unit will move to the waiting state, where it will wait for the money to be inserted. Then if rupees 1/- note is inserted then the machine will go to state_1 and wait until the desired money is inserted. And if rupees 2/- note is inserted the machine will move to state_2 and then wait until 30/- rupees are inserted to the machine. When the desired amount is inserted the machine will go to the snacks state and snacks will be delivered at the product output.

If products are not available in the machine then the control unit will demand for servicing and after service the machine will get reset. There is also an additional feature of withdrawing the request if the user doesn’t want to take the product. When cancel button is pressed then the money inserted will be returned to the user through the return output. A money count signal is used for calculating the total money inserted in the machine. And if the money inserted is more than the money of the product then the extra change will be returned to the user. The total amount of the product taken at a time is shown by the money signal. Similarly the user can select and get the other products following the above procedure.
A peer reviewed open access international journal

The selection of products and all the states are given below:

- When initialize=>
- money_count=0;
- Change=0;
- Product=0;
- When select1=>Sel1&!sel2&!sel3
  When product_available=1=>
  nx_st1<= waiting1;
  When product_available=0 =>
  nx_st1<= service1;
- When waiting1=>
  When rs_1&!rs_2=> nx_st1<= state_1;
  When !rs_1&rs_2=> nx_st1<= state_2;

5. SIMULATION RESULTS:

The state diagram shown in figure is simulated using Modelsim 6.5 e. Simulation Waveforms for the selection of three products like Coffee, Tea, Milk is shown in figure respectively with servicing feature when products are not available in the machine and change return features when the money inserted is more than the money of the product. Let us take an example that the user wants to take Tea. When one selects sel1 button, the machine will check that whether the products are available or not, if available then it will go to the waiting state and wait for total money insertion. If rs_1 note is inserted it will go to state_1 and if rs_2 note is inserted it will so to state_2 and check whether money_count>=3 or not. If the money_count >= 3 then machine will go to state snacks and vend the product.
5.2 SYNTHESIS REPORTS:

5.2.1 BINARY ENCODING:

Figure 5.1: Synthesis waveform for Binary Code

5.2.2 ONE HOT ENCODING:

Figure 5.2: Simulation waveform for Tea Selection

Figure 5.3: Simulation wave for Milk Selection

Figure 5.4: Synthesis waveform for Binary Code

Figure 5.5: Synthesis waveform for One-hot Code

RTL Schematic: One Hot Encoding:

Figure 5.6 RTL View of One Hot Vending Machine

Figure 5.7 RTL Schematic of One Hot Vending Machine

TIMING REPORT

Clock Information:

<table>
<thead>
<tr>
<th>Clock Signal</th>
<th>Clock buffer(FF name)</th>
<th>Load</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Asynchronous Control Signals Information:

No asynchronous control signals found in this design.

Timing Summary:

Speed Grade: -4

Minimum period: 3.263ns (Maximum Frequency: 306.466MHz)

Minimum input arrival time before clock: 4.959ns

Maximum output required time after clock: 4.310ns

Maximum combinational path delay: No path found

Timing Detail:

Data Path: m_in to state_0

<table>
<thead>
<tr>
<th>Gate</th>
<th>Net</th>
<th>Cell:in-&gt;out</th>
<th>fanout</th>
<th>Delay</th>
<th>Delay</th>
<th>Logical Name (Net Name)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>IBUF:I-&gt;O</td>
<td>4</td>
<td>1.218</td>
<td>0.762</td>
<td>m_in_IBUF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LUT3:I0-&gt;O</td>
<td>1</td>
<td>0.704</td>
<td>0.455</td>
<td>next_state&lt;0&gt;128</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LUT3_L:I2-&gt;LO</td>
<td>1</td>
<td>0.704</td>
<td>0.104</td>
<td>next_state&lt;0&gt;147_SW0 (N142)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LUT4:I3-&gt;O</td>
<td>1</td>
<td>0.704</td>
<td>0.000</td>
<td>next_state&lt;0&gt;1601</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FDS:D</td>
<td>0.308</td>
<td></td>
<td></td>
<td>state_0</td>
</tr>
</tbody>
</table>

Total 4.959ns (3.638ns logic, 1.321ns route)

(73.4% logic, 26.6% route)

Binary Coded Design:

Final Results

RTL Top Level Output File Name : fsmdesign_binaryngr

Top Level Output File Name : fsmdesign_binary

Output Format : NGC

Optimization Goal : Speed

Keep Hierarchy : NO

Device utilization summary:

Selected Device : 3s100etq144-4

Number of Slices: 22 out of 960 2%

Number of Slice Flip Flops: 27 out of 1920 1%
Number of 4 input LUTs: 39 out of 1920
2%
Number of IOs: 14
Number of bonded IOBs: 14 out of 108
12%
Number of GCLKs: 1 out of 24
4%

TIMING REPORT
Clock Information:
+---------------------+------------------------+------+
| Clock Signal       | Clock buffer(FF name) | Load |
+---------------------+------------------------+------+
| clk                 | BUFGP                  | 27   |
+---------------------+------------------------+------+
Timing Summary:
Speed Grade: -4
Minimum period: 3.474ns (Maximum Frequency: 287.853MHz)
Minimum input arrival time before clock: 5.060ns
Maximum output required time after clock: 4.283ns
Maximum combinational path delay: No path found

Comparison Table:

<table>
<thead>
<tr>
<th>FSM State Encoding</th>
<th>Area in Slices</th>
<th>Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>22</td>
<td>5.06ns</td>
</tr>
<tr>
<td>One hot</td>
<td>10</td>
<td>3.43ns</td>
</tr>
</tbody>
</table>

Table 5.1 : Synthesis and Delay Comparison

CONCLUSION & FUTURE SCOPE:

It takes more time and money initially to use these methods to reduce the number of states and logic to implement these states. Because of this, when designing a simple logic circuit, these methods are not needed. If a complex design with many states is required however, it takes more time and money in the end to not use these methods. When dealing with state machines that have many inputs and states, the circuit can become very large and complex.

The more complex the circuit, the easier it is to make a mistake in the logic. The present One Hot Encoding FSM vending machine controller is implemented using FSMs with the help of Xilinx ISE Design Suite 9.2 State machines based vending Systems enhances productivity, reduces system development cost, and accelerates time to market. Also FSM based vending machine give fast response and easy to use by an ordinary person. The designed machine can be used for many applications and we can easily enhance the number of selections. Compare to Binary Encoded based Design One- Hot Method is High Speed Shown in Compassion Table 5.1

REFERENCES:


Mr. Y. V. Krishna Reddy was born in Andhra Pradesh, India. He received the B.Tech degree in Electronics and Communication Engineering from Audhishankara College of Engineering and Technology affiliated to JNT University, Anantapur, and M.Tech degree in Embedded Systems from Nova College of Engineering & Technology, Ibrahimpatnam, Telangana, affiliated to JNT University-Hyderabad. He is working as an Associate Professor in the Dept. of Electronics and Communication Engineering and Specialization in Embedded System Design at Skr College of Engineering & Technology, Konduru Satram [V], Manubolu [M], S.P.S.R Nellore [Dt], Andhra Pradesh, India.

About Authors:

Modi Mahendra was born in Musthapuram [V], Ananthasagaram [M], S.P.S.R. Nellore [Dt], Andhra Pradesh, India. He received the B.Tech degree in Electronics and Communication Engineering from Priyadarshini Institute of Technology affiliated to JNT University, Anantapur, in 2013 and is pursuing M.Tech degree in VLSI and Esd from Skr College of Engineering & Technology, Konduru Satram [V], Manubolu [M], S.P.S.R. Nellore [Dt], Andhra Pradesh, India, affiliated to JNT University, Anantapur.


[REFERENCES:]

[1] Fauziah Zainuddin, Norlin Mohd Ali, Roslina Mohd & Esd From Skr College Of Engineering & Technology, Konduru Satram [V], Manubolu [M], S.P.S.R Nellore [Dt], Andhra Pradesh, India, Affiliated To Jnt University, Anantapur In 2013 And Pursuing M.Tech Degree In Vlsi & Esd From Skr College Of Engineering & Technology, Konduru Satram [V], Manubolu [M], S.P.S.R.Nellore [Dt], Andhra Pradesh, India, Affiliated To Jnt University, Anantapur.


