# ANALYSIS OF THE PROPOSED INTEGRATED CIRCUITS FOR PARITY, HAMMING AND MOD 3 CODING CODES USING LOGISIM

**Milotin Barakov – Student**
Faculty of Electrical Engineering, Electronics and Automation
Department of Computer Systems and Technologies
University of Ruse "Angel Kanchev"
Phone: +358 87 79 90 817
E-mail: milotinbarakov@gmail.com

**Assoc. Prof. Galina Ivanova, PhD**
Faculty of Electrical Engineering, Electronics and Automation
Department of Computer Systems and Technologies
University of Ruse "Angel Kanchev"
Phone: +359 82 888 827
E-mail: giivanova@uni-ruse.bg

***Abstract:*** *The paper compared the simulated circuits, used to solve problems in the discipline RELIABILITY AND DIAGNOSTICS OF COMPUTER SYSTEMS and attempts to recreate them inside of a real CAD environment. It shows the problems which can arise while looking at hardware components from a programmer's point of view. The purpose of this paper is to offer improvements which at the end will benefit the learning process.*

***Keywords****: Simulation, Logisimp, Integrated Circuits, Parity check, Hamming code, Mod 3code*

## INTRODUCTION

The discipline Reliability and diagnostics of computer systems introduces the students to the methods used to secure the integrity of transmitted digital information. It aims to give theoretical knowledge for the reasons causing loss of integrity as well as their theoretical mathematical solutions and the circuits used in practice to realize them. The acquired to knowledge is intended to be used in disciplines such as: Cryptography, Computer Security, Computer Networks as well as in the practical field by students who decide to pursue career in the field.

Having a practical knowledge of the principals of operations of the circuits used in the discipline will give a significant jump start to any such students. Therefore, accurate representation of the used **Integrated Circuits** (**IC**s) is mandatory.

## EXPOSITION

Three types of codes were examined – Parity check, Hamming and Module 3. The example ICs were created inside of a supplied **"Simulator of Logical Circuits"** (which will be referred to as **SLC**). For the purposes of this study the same ICs are recreated inside of the Computer Aided Development environment – Logisim, (Burch, C., 2002), changing as little as possible, while keeping the example ICs working properly (as much as possible).

### Parity check

Parity check is relatively simple and easy method for checking the amount of even/odd numbers in a given signal, capable of finding a single or odd number of errors, (Guruswami, V. 2010). It is widely used and its implementation in a simulation shouldn't be problematic. However, upon attempting to recreate the IC (Fig.1) in Logisim a problem with how flip-flops (synchronous, or clocked)/latches (asynchronous) are simulated inside of SLC is encountered.

  ***ISBN 978-954-712-793-7 (Print)***

The supplied bistable multivibrators (flip-flops) were not being simulated as expected (either as a synchronous or asynchronous component). All of them had a clock input to which they did respond however, they expected constant a logical high or low (1/0) to which they would react instantly (change their internal state and push out their previous state on their outputs). Upon starting the circuit inside of Logism instantly an oscillation became apparent (Fig 2.) The logic gates and the bistable multivibrators run at different clocks (as they would in real life) which produced oscillation in the ICs between the input and output of the flip-flop.

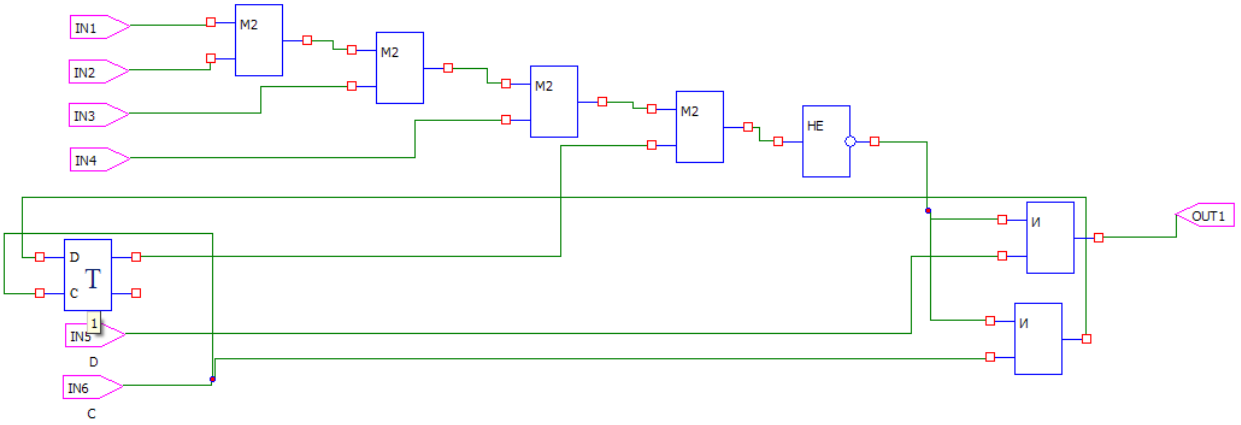


Fig. 1 Example circuit in SLC

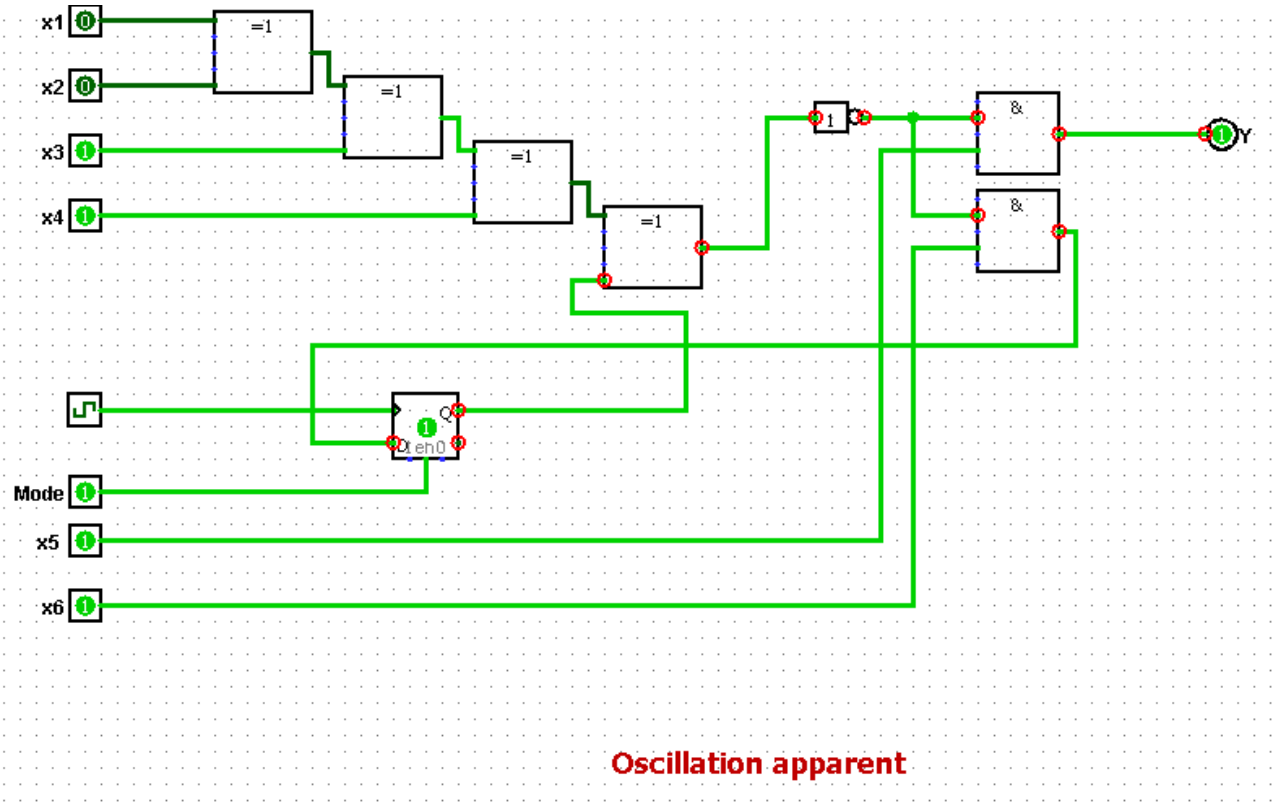


Fig. 2 Logisim simulates the clock frequency at which the elements run as well as the delays they experience internally

The IC is recreated as asynchronous one (Fig. 3) which removed the oscillation but that solution meant that the Latches had to be controlled manually by the user.

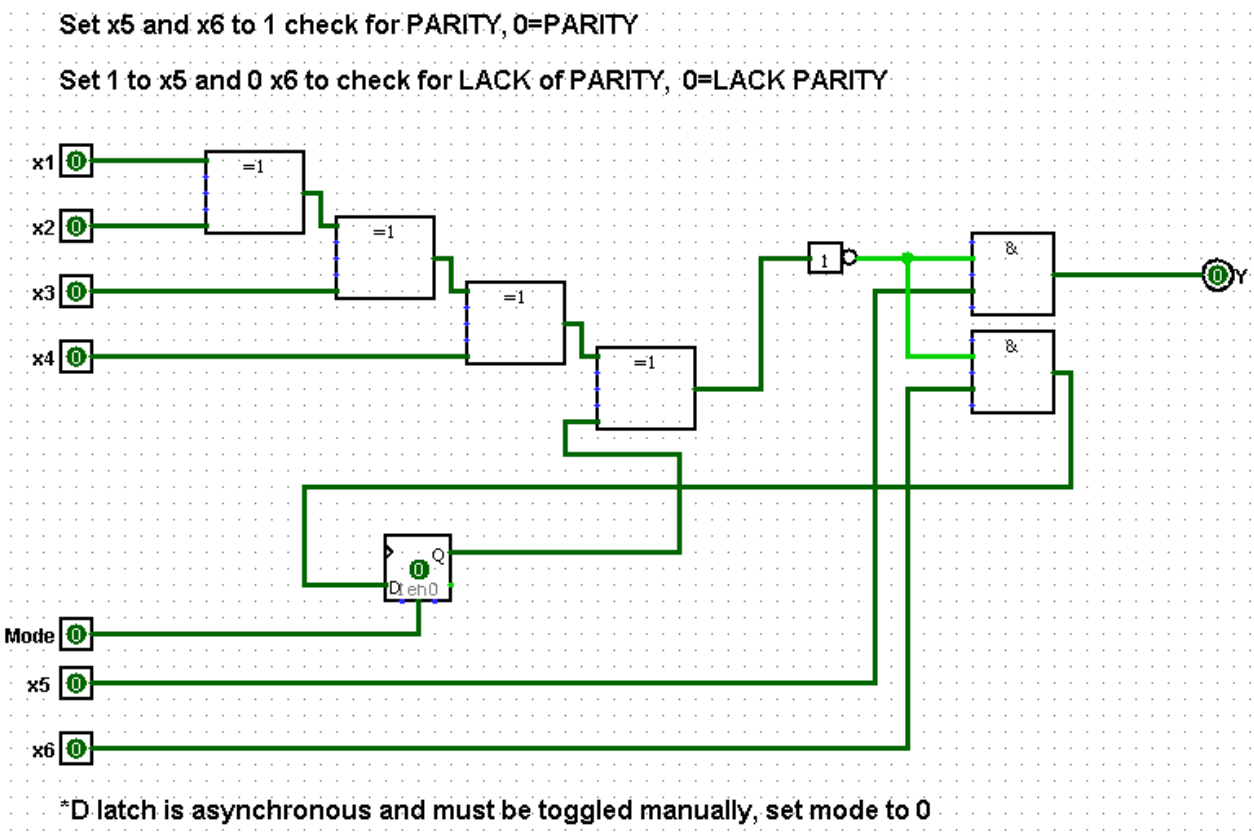


Fig. 3 The IC works but the controls are changed.

As well as the functionality problems Logisim required controlling the trigger manually, which made the use of simulation more complicated to use. In order to verify the results, the IC was recreated (Fig. 4) using the 7400 Library a collection of circuits made by Texas instruments which a simulated with 100% accuracy inside of Logisim. Surprisingly the IC was able to work synchronously since Logisim automatically adjusts the clock of all 7400 components to be compactable.

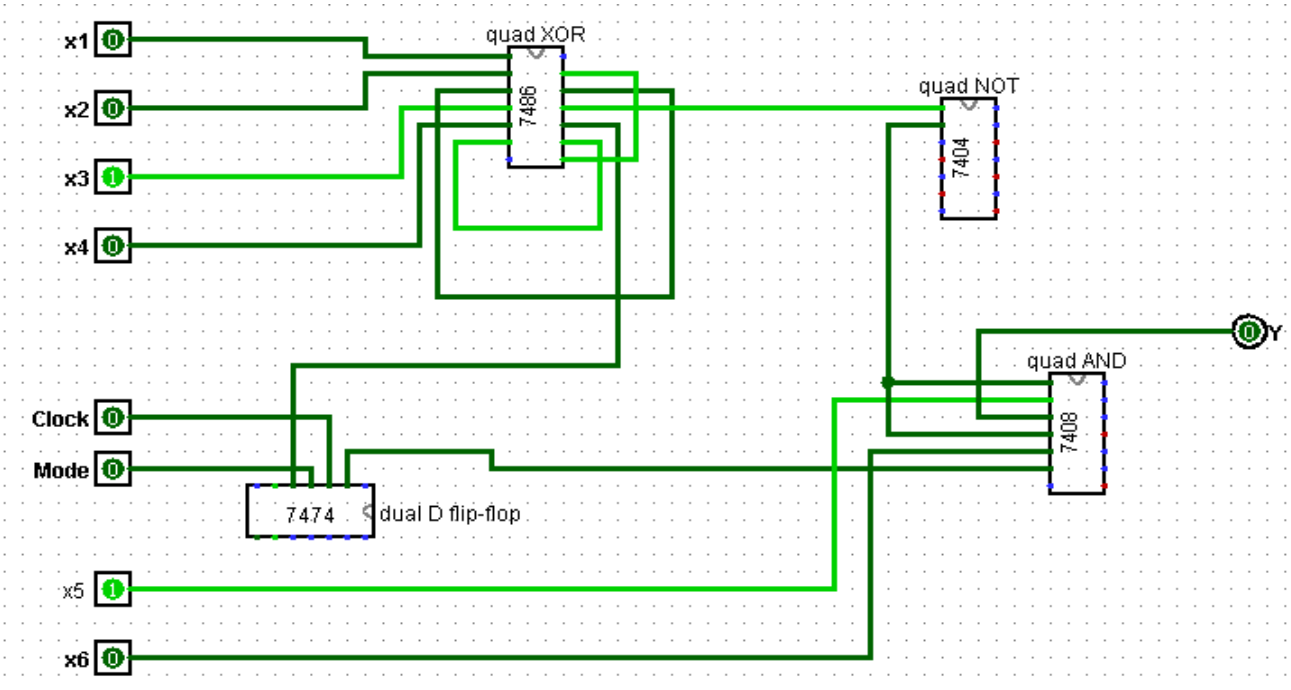


Fig. 4 Parity check using the 7400 library

 

Overall recreating Parity check code inside proved somewhat difficult due to the unexpected behavior of the flip-flops. Its usage was slightly more difficult than in SLC.

**Hamming code**

Hamming code (Hamming & Wesley R.,1950) is much more complicated than Parity code – it is able to detect up to two single bit errors or correct a single bit one therefore the code is widely used especially in Error-Correction Code (ECC) computer RAM, (Moon, T. K. , 2005). It is logical to expect that recreating Hamming code will be significantly more difficult, keeping in mind that only asynchronous triggers are usable and the code requires the use 3 flip-flops. (Fig. 5) Another problem which could arise is the need for decoder 3-to-8.

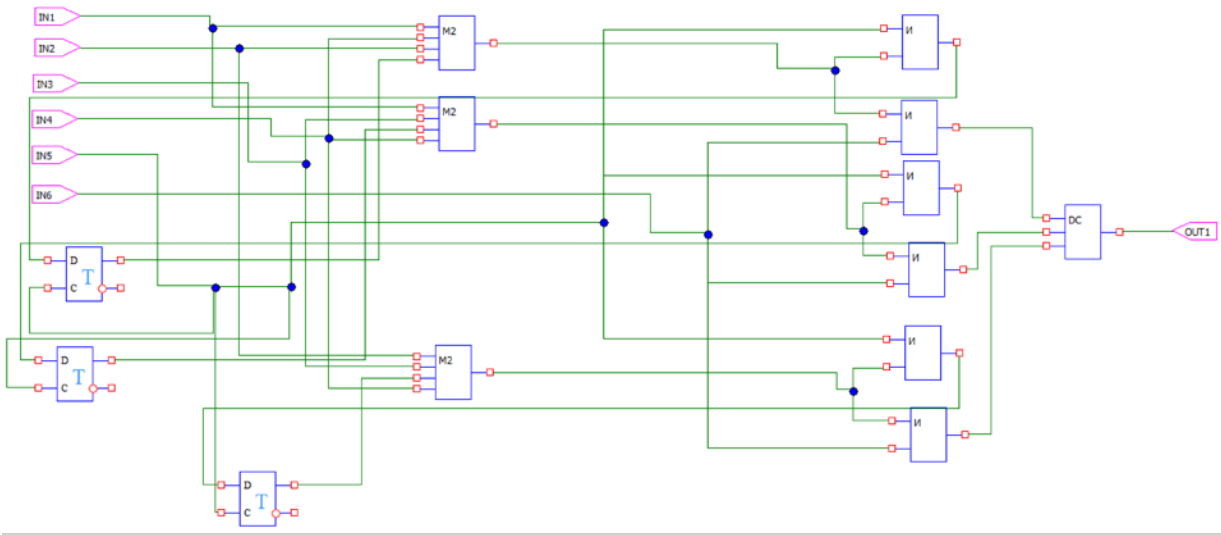


Fig. 5 Example Hamming circuit

Having found a way around the flip-flop limitations to which my ICs had to conform recreating a Hamming code was mostly straight forward and not particularly difficult, (Fig. 6). Only problem to overcome was the lack of information SLC of what type of quad input XOR was used. By default, Logisim automatically creates an Even parity (if the number of zeroes is even then there is 1 on the output) IC when selecting XOR element with more than 2 inputs, which is the reason each quad input XOR was replaced by 3 dual input XORs.

In order to decode the signal, a newly created 3-to-8 priority decoder is proposed (Fig. 7), using a Truth Table (Table 1) with the help of the auto generation tool in Logisim.

Table 1. Truth Table of 3-to-8 priority decoder

| **Inputs** | | | **Outputs** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **a2** | **a1** | **a0** | **y0** | **y1** | **y2** | **y3** | **y4** | **y5** | **y6** | **y7** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

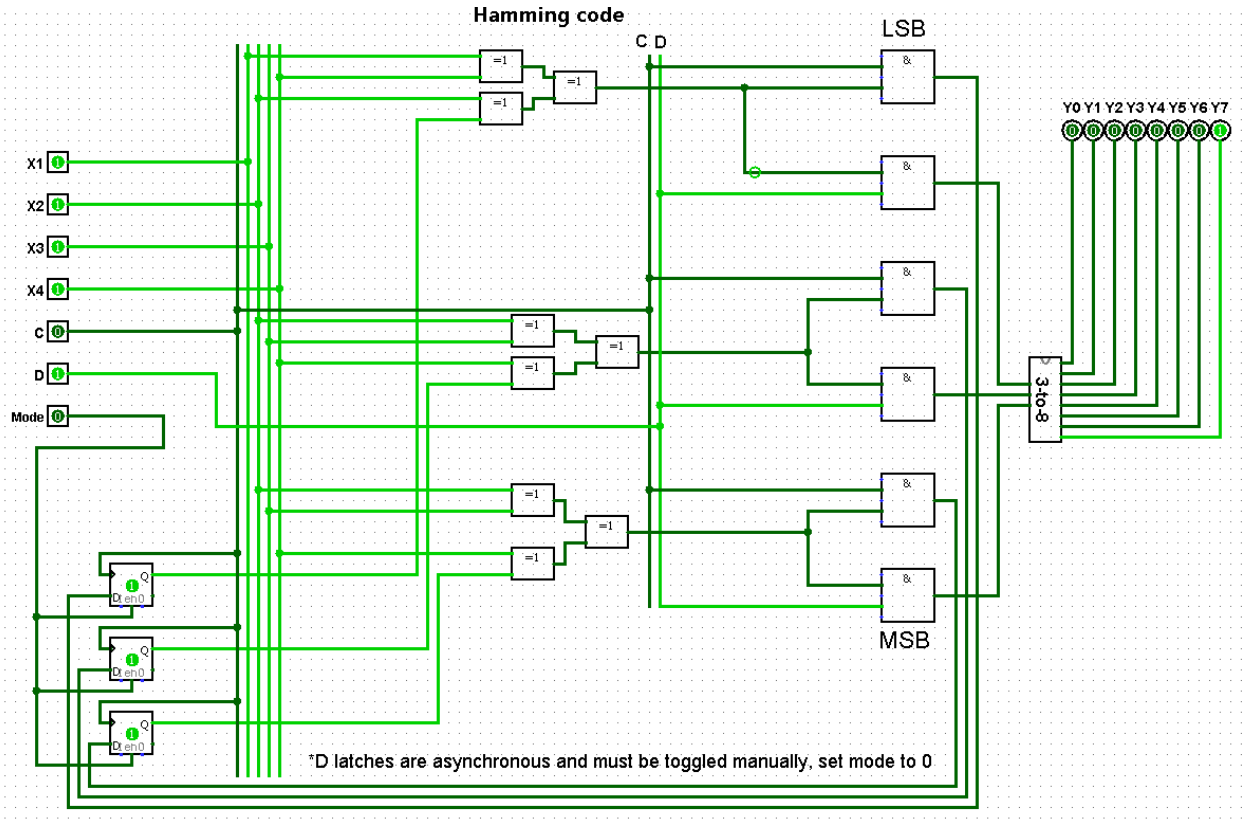


Fig. 6 Hamming code with custom decoder

Overall recreating Hamming code was not difficult. The usage of the generated circuit was slightly more difficult than the example in SLC.

**Mod 3 code**

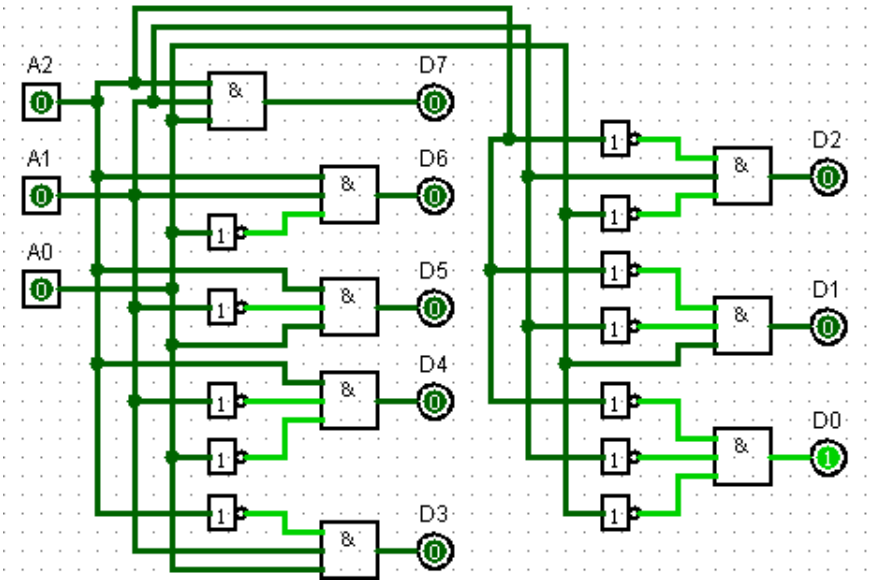


Fig. 7 Custom made 3-to-8 priority encoder

Mod 3 code is created to be easily executable by machines and there shouldn't be significant difficulties in its implementation, (Klove, T. 2007).

Since the current version of SLC had a bug which made usage of system buses impossible the given example is somewhat unorganized (Fig. 8) and hard to process. It recreation in Logisim

 

(Fig. 9) was mostly effortless however, a 3-to-2 coder IC for Logisim doesn't exist. Fortunately, the included circuit generation tool was able to easily autogenerate a circuit just by inputting the Truth Table of the Decoder. The example IC was hard to read. The shown example only processes 3 bits at a time compared to Fig. 8, which processes 4

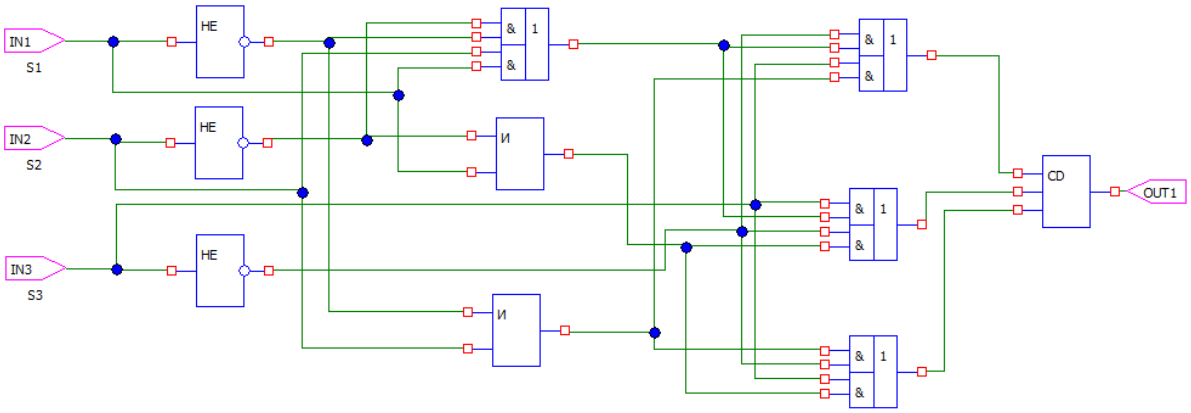


Fig. 8 Circuit control code module 3

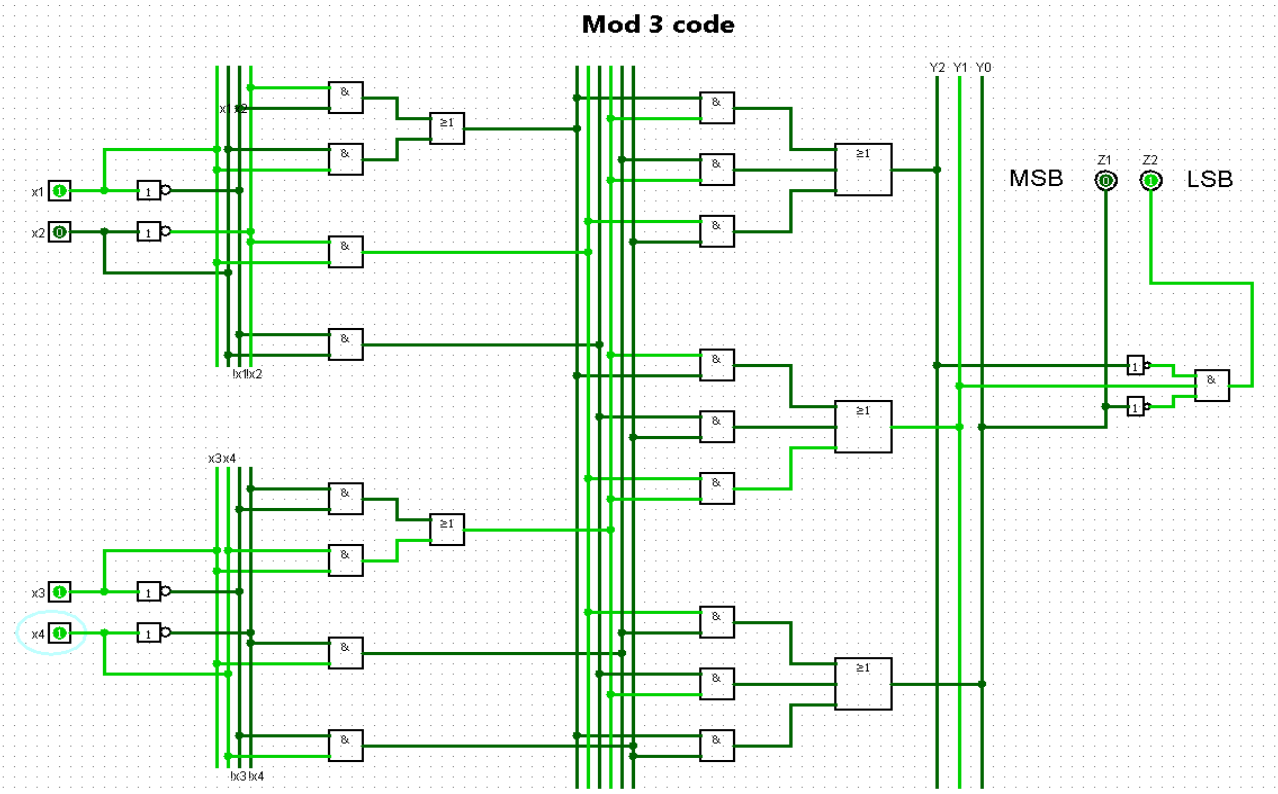


Fig. 9 Recreated IC was well organized

Overall recreating Parity check inside proved mostly easy. Its usage was definitely easier due to the ability to better arrange the components.

## CONCLUSION

The results from the experiment show a type of *pseudo-synchronous bistable multivibrators* which instantly update a single time once a signal is received on the Clock input. It is apparent the simulation is only *ran* once the "start" button is pressed and is paused the rest of the time.

The demonstrated behavior points towards not actually simulating the components but rather calculating the logical equations and displaying the result. This is very likely since actually implementing a real simulation of the components would be very difficult.

**As well as more accurate simulation Logisim offers the following benefits:**

Ability to arrange items freely and easily move them without risk of breaking the entire circuit.

Ability to work with multiple circuits in the same project.

Ability to load 3rd party libraries with components.

Ability to create and save custom ICs.

**Using the facts stated in this paper allows me to objectively conclude the following:**

SLC is usable and somewhat capable of achieving the desired purposes however, there are better alternatives.

The students would benefit from using more advanced CAD environment.

Logisim is a very capable environment and is definitely more suited the purposes of the education process, (Lee, A, 2009).

In conclusion Logisim is capable of fulfilling the requirements of the course Reliability and diagnostics of computer systems however, there is a multitude of free CAD solutions on the market which could prove better. It is only logical to continue researching the problem.

**REFERENCES**

Moon, T. K. (2005). Error correction coding. Mathematical Methods and Algorithms. Jhon Wiley and Son, 2001-2006.

Richardson, T., & Urbanke, R. (2008). Modern coding theory. Cambridge university press.

Burch, C. (2002). Logisim: A graphical system for logic circuit design and simulation. ACM Journal, Vol.2, No.1

Lee, A. (2009). Conway's Game of Life, made in Logisim. University of California, Berkeley

Logisim (2014). Product documentation

Hamming & Wesley R (1950). Error detection and error correction codes. Bell System Technical Journal

Guruswami, V. (2010). Introduction to coding theory. Carnegie Mellon University

Klove, T. (2007). Codes for Error Detection. University of Bergen, Norway

 