Bittar, Catherine (260687735) 27 October 2017

Sorto-Ventura, Kevin-Rafael (260692767) McGill University

ECSE 323: Digital System Design

Lab #2: g21\_RANDU, g21\_pop\_enable, g21\_7\_segment\_decoder

Table of Contents

[Introduction 2](#_Toc496473753)

[g21-RANDU 3](#_Toc496473754)

[Circuit Function 3](#_Toc496473755)

[Circuit Design 3](#_Toc496473756)

[Schematic 4](#_Toc496473757)

[Simulation and Discussion 4](#_Toc496473758)

[g21\_pop\_enable 5](#_Toc496473759)

[Circuit Function 5](#_Toc496473760)

[Circuit Design 5](#_Toc496473761)

[Schematic 5](#_Toc496473762)

[Simulation and Discussion 5](#_Toc496473763)

[g21\_7\_segment\_decoder 7](#_Toc496473764)

[Circuit Function 7](#_Toc496473765)

[Circuit Design 7](#_Toc496473766)

[Schematic 8](#_Toc496473767)

[Simulation and Discussion 8](#_Toc496473768)

[Conclusion 9](#_Toc496473769)

[References 9](#_Toc496473770)

List of Figures

Figure 1 g21\_RANDU Block Diagram 3

Figure 2 g21\_RANDU schematic generated from VHDL file 4

Figure 3 g21\_RANDU functional simulation results 4

Figure 4 g21\_pop\_enable schematic generated from VHDL file 5

Figure 5 g21\_pop\_enable functional simulation results 6

Figure 6 Seven segment LED output based on code and mode inputs 7

Figure 7 Mapping of the 7 segment LED output 7

Figure 8 g21\_7\_segment\_decoder schematic generated from VHDL file 8

Figure 9 g21\_7\_segment\_decoder functional simulation results 9

# Introduction

The purpose of this lab was to design a random number generator, a ROM based enable signal decoder circuit as well as a 7-bit segment LED decoder circuit using VHDL descriptions. More specifically, we designed a circuit based on the IBM *RANDU* random number generator algorithm which was implemented in their computers in the 1960s [1]. The second part was the design of a Pop-Enable circuit which takes a 6-bit input and generates a 52-bit output based on a lookup table (ROM module). Finally, the hardware description of the 7-bit segment decoder was through the VHDL equivalent of a truth table. Note that all three of these circuits will become important building blocks for the term project which involves playing a card game.

The design was completed using the FPGA design software Quartus. It was also used to compute simulations of the designed circuit. This design was part of the project file *g21\_lab2.QPF*.

# 

# g21-RANDU

## **Circuit Function**

The g21\_RANDU circuit has 1 input and 1 output.

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Length (bit)** |
| Input | seed | 32 |
| Output | rand | 32 |

The *RANDU* operation can be expressed mathematically as:

where is the input value and the function gives the remainder of the division of () by .

## **Circuit Design**

The circuit is described in the VHDL design file *g21\_RANDU.vhd*. The design was implemented with two 32-bit adders from the lpm library. See Figure 1 g21\_RANDU Block Diagram. According to the function described above, the seed should first be multiplied by 65539 and then find the modulo by .

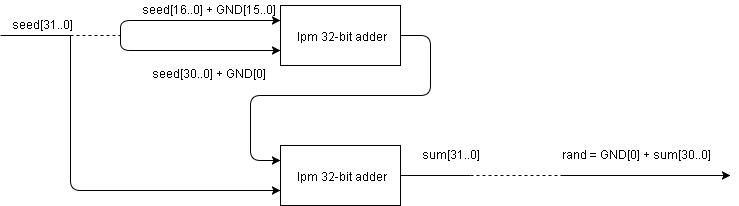
1. Multiply by 65539  
   As in Lab #1, the multiplication by 65539 can be simplified using shift and add operations. As, the operation is instead simplified by left-shifting the seed by 16,1, and 0 and summing them.
2. Modulo by   
   The modulo by can be found by setting the bits beyond to 0. In this case, this is only the most significant bit. 

Figure 1 g21\_RANDU Block Diagram

## **Schematic**

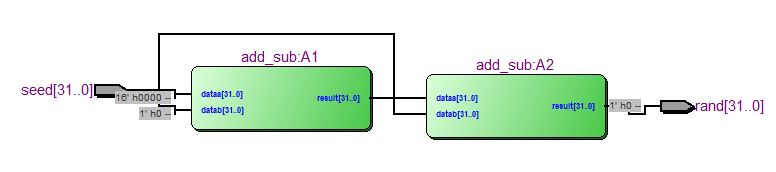


Figure 2 g21\_RANDU schematic generated from VHDL file

## **Simulation and Discussion**

A functional simulation was done on the circuit to ensure it was functioning correctly. The results of the simulation can be found in Figure 3 g21\_RANDU functional simulation results. All 2^31 possible inputs were simulated. These results were further verified using a known *RANDU* relationship:

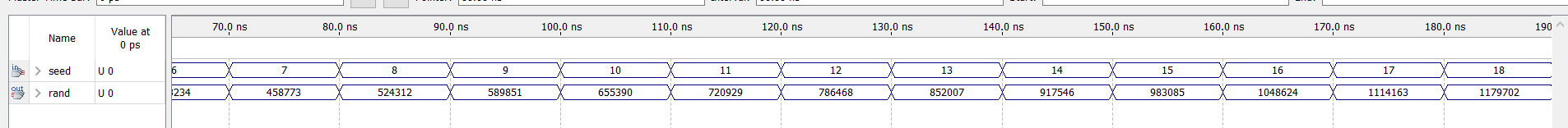


Figure 3 g21\_RANDU functional simulation results

# g21\_pop\_enable

## **Circuit Function**

The g21\_pop\_enable circuit has 2 inputs and 1 output.

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Length (bit)** |
| Input | N | 6 |
| Input | clk | 1 |
| Output | P\_EN | 52 |

The pop-enable circuit takes an integer N and returns a 52-bit output where the bits 0 to are 0 and bits N to 51 are 1.

## **Circuit Design**

The circuit is described in the VHDL design file *g21\_pop\_enable.vhd*. The design was implemented with a ROM look-up table (LUT) from the lpm library. The look-up table was chosen as opposed to a select statement as it is less hardware intensive when there are numerous statements. See Figure 4 g21\_pop\_enable schematic generated from VHDL file for the schematic.

## **Schematic**

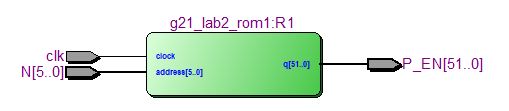


Figure 4 g21\_pop\_enable schematic generated from VHDL file

## **Simulation and Discussion**

A functional simulation of the circuit was performed to verify its function. All possible addresses were simulated. The clock signal was simulated using high frequency waveform. It should be noted that a significant delay in response was found the longer the clock signal was. In this simulation, the clock signal is set to 10 ns and a slight delay can still be perceived. This is a limitation of the circuit to be considered if it is implemented in a larger circuit.

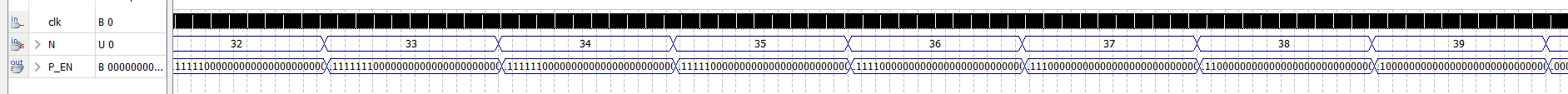


Figure 5 g21\_pop\_enable functional simulation results

# g21\_7\_segment\_decoder

## **Circuit Function**

The g21\_7\_segment decoder circuit has 2 inputs and 1 outputs.

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Length (bit)** |
| Input | code | 4 |
| Input | mode | 1 |
| Output | segments\_out | 7 |

The circuit is meant to output the set of segments to form the characters described in Figure 6. The output bits are defined in [2]:

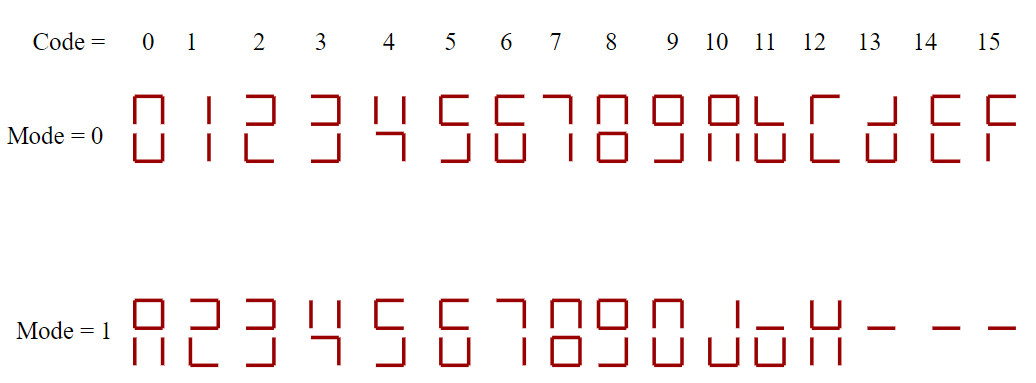


Figure 6 Seven segment LED output based on code and mode inputs

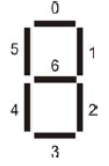


Figure 7 Mapping of the 7 segment LED output

## **Circuit Design**

The circuit is described in the VHDL design file *g21\_7\_segment\_decoder.vhd*. A *select when* statement was used in vhdl to assign proper outputs to inputs. This component was selected as opposed to a lookup table (ROM) because it was simple to implement with this few (32) different pairs of inputs and outputs.

## **Schematic**

Below is the block diagram generated by the Altera Quartus software from the VHDL file. It is composed of 7 multiplexors: which is a logical solution as the output has 7 bits.

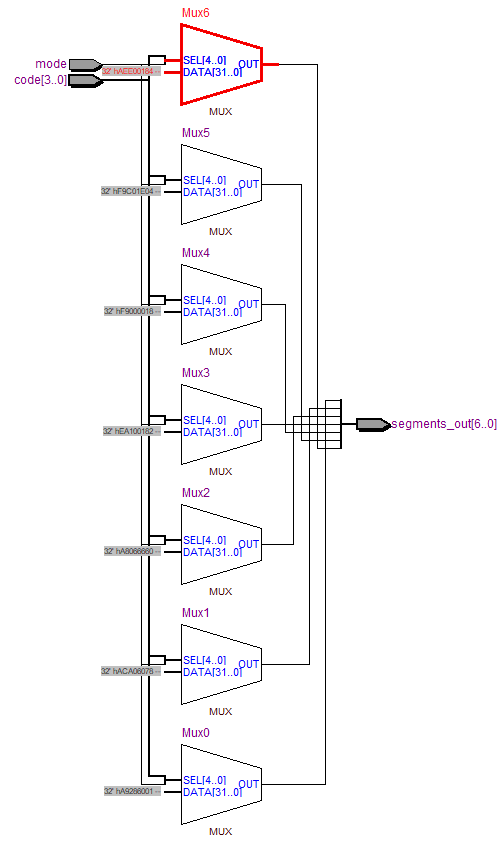


Figure 8 g21\_7\_segment\_decoder schematic generated from VHDL file

## **Simulation and Discussion**

After creating a symbol from the VHDL description, the circuit was simulated with a vector waveform file. The results obtained are illustrated below. As expected, the 7-bit outputs match the ones described in the *g21\_7\_segment\_decoder.vhd* design file. Note that the circuit diagram in the previous section is already rather large but it still works well in this case due to the relatively few inputs and small output size in terms of bits. The efficiency of this method would naturally decrease with larger datasets.

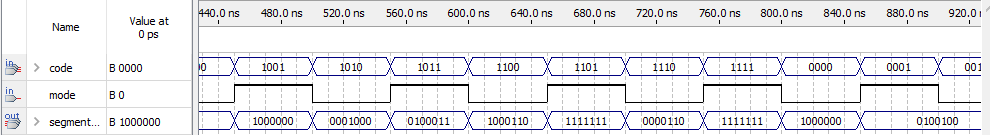


Figure 9 g21\_7\_segment\_decoder functional simulation results

# Conclusion

This lab consisted of two types of circuits: the random number generator and input-output assignments. In the latter case, two circuits were designed: one using a multiplexor approach, the other with rom. The multiplexor circuit has the advantage of being quicker, but requires more extensive hardware as more inputs are to be mapped to outputs. The rom circuit is a slower alternative which requires a circuit with memory storage, but is more effective for larger sets of input-outputs. All three circuits were successfully simulated, but it was found that when a clock signal was used, it necessitated a high frequency to limit delays.

# 

# References

[1] System/360 Scientific Subroutine Package, Version III, Programmer's Manual. IBM, White Plains, New York, 1968, p. 77

[2] Prof. J. Clark, Lab Instructions, “ECSE-323 Digital Systems Design: Lab #2 - Combinational Circuit Design with VHDL ”, Department of Electrical and Computer Engineering, McGill University, Oct. 2017.