Bittar, Catherine (260687735)
Sorto-Ventura, Kevin-Rafael (260692767)

27 October 2017
McGill University

ECSE 323: Digital System Design
Lab #2: g21_RANDU, g21_pop_enable, g21_7_segment_decoder

Table of Contents

List of Figures

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 done 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:

$$rand = mod(65539 * seed, 2^{31})$$

where $seed$ is the input value and the mod function gives the remainder of the division of $(65539 * seed)$ by 2^{31}.

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 2^{31}.

A. Multiply by 65539
   As in Lab #1, the multiplication by 65539 can be simplified using shift and add operations. As $65539 = 2^{16} + 2^{1} + 1$, the operation is instead simplified by left-shifting the seed by 16,1, and summing these two values with the non-shifted input.
B. Modulo by 2^{31}
   The modulo by 2^{n}can be found by setting the bits beyond $n - 1$to 0. In this case, this is only the most significant bit.

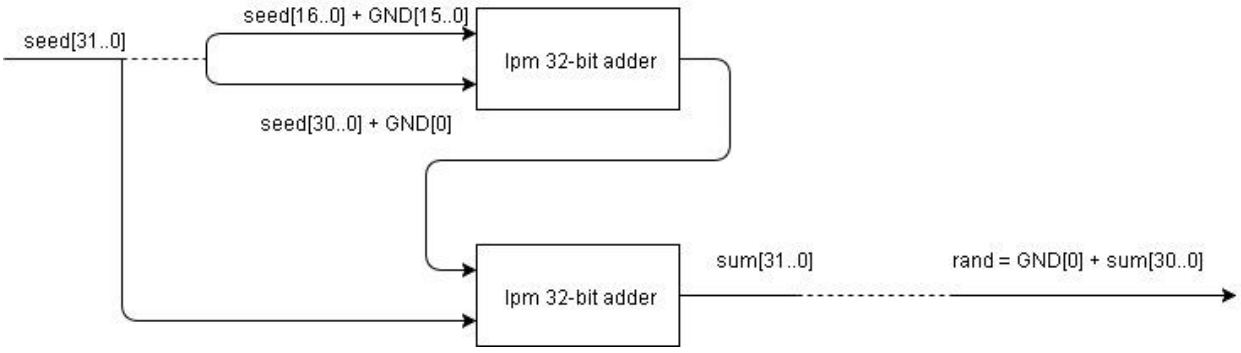

Figure 1 g21_RANDU Block Diagram

Schematic

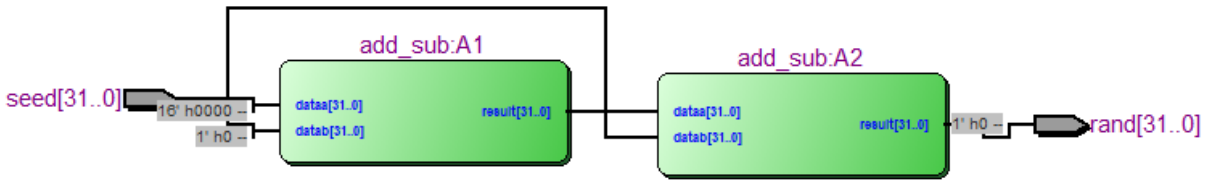

Figure 2 g21_RANDU schematic generated from VHDL file

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 [2]:

$$mod(R_n - R_{n-1} * 6 + R_{n-2} * 9,\ 2^{31}) = 0$$

Although the *RANDU* function works well enough for the current purposes of the lab, it is not a good random number generator [2] as it is easy to predict the next value using the relationship above.

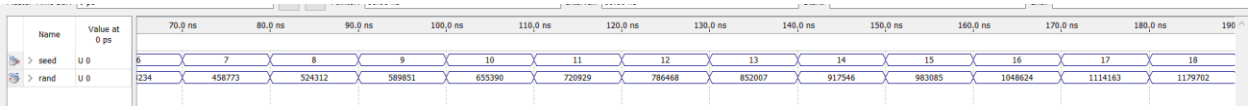

Figure 3 g21_RANDU functional simulation results

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 $N-1$ 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

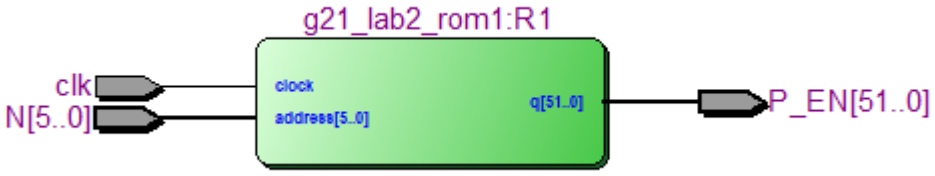

Figure 4 g21_pop_enable schematic generated from VHDL file

Simulation and Discussion

A functional simulation of the circuit was performed to verify its function. All 2^6possible addresses were simulated. The clock signal was simulated using high frequency waveform. It should be noted that a significant delay in response was found when longer the clock signal was longer. 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.

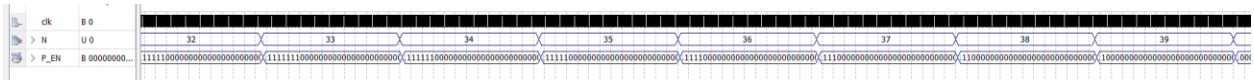

Figure 5 g21_pop_enable functional simulation results

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]:

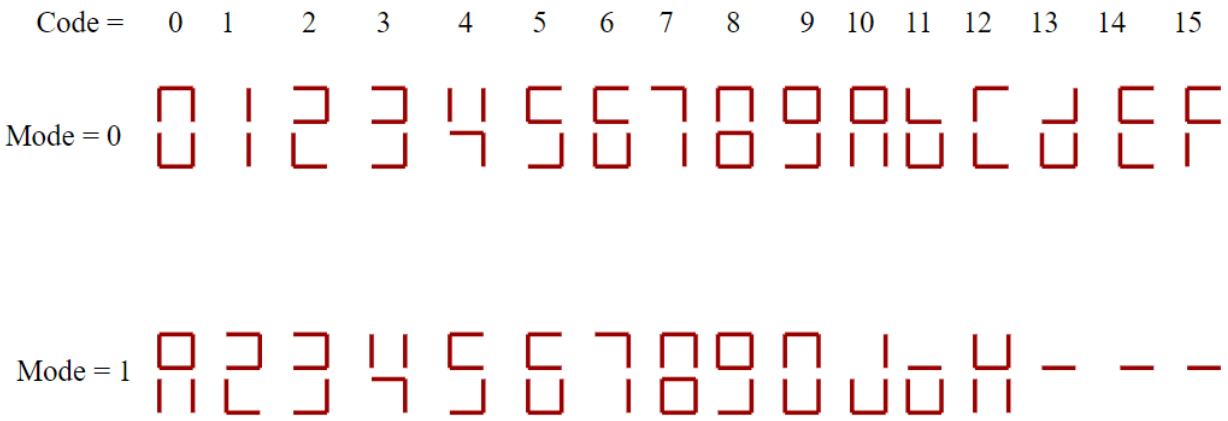

Figure 6 Seven segment LED output based on code and mode inputs

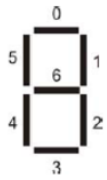

Figure 7 Mapping of the 7 segment LED output

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.

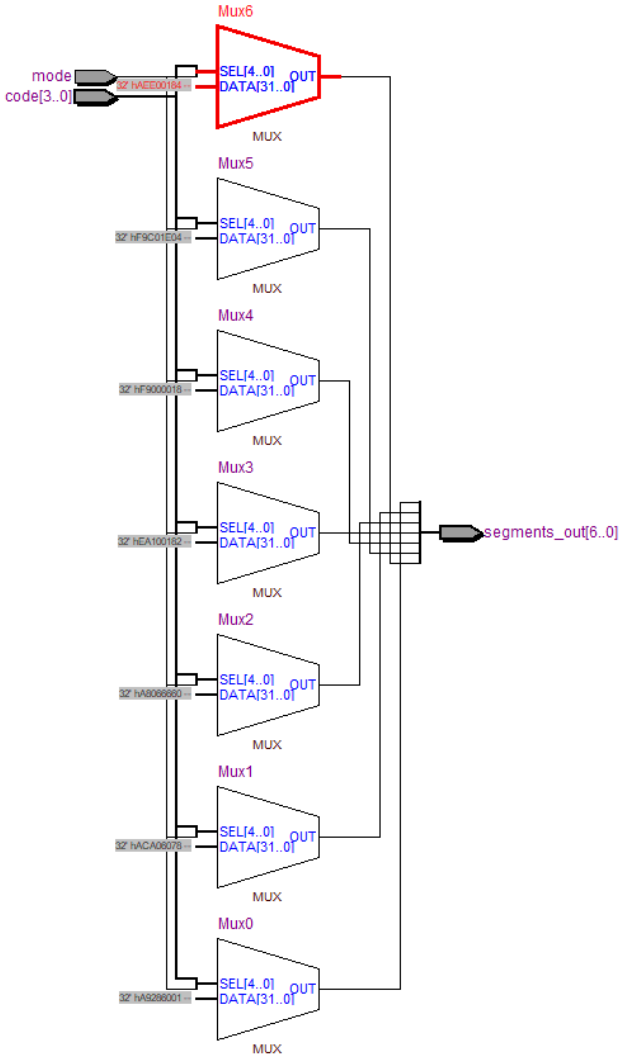

Figure 8 g21_7_segment_decoder schematic generated from VHDL file

Simulation and Discussion

After creating a symbol from the VHDL description, the circuit was simulated with a vector waveform file. All 32 combinations of the code and mode inputs were tested. 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.

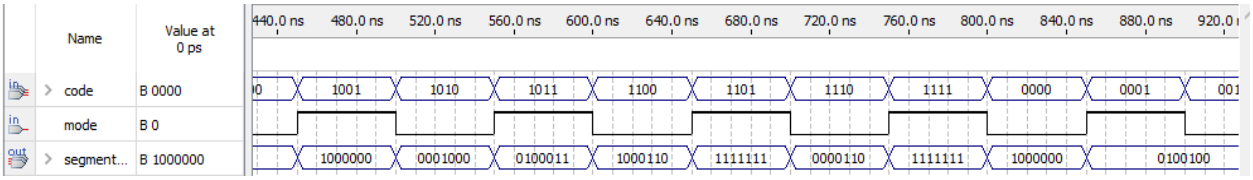

Figure 9 g21_7_segment_decoder functional simulation results

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 not requiring memory, but uses more extensive hardware as more inputs are to be mapped to outputs. The rom circuit requires a circuit with memory storage, but uses less hardware, and thus 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.