Course Name: EMBEDDED SYSTEMS I / III
Course Number and Section: **14:332:493:03 / 16:332:579:05**
Year: Spring 2024

Lab Report #: Final Project

Lab Instructor: Milton Diaz

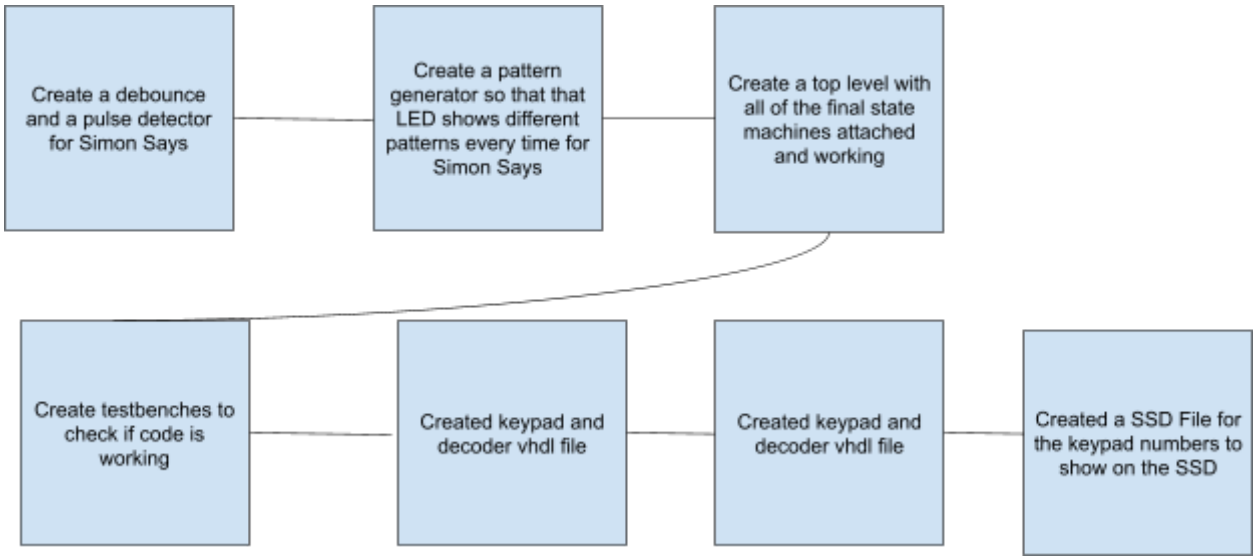
Student Name and RUID: Neha Murthy (207008813)

Date Submitted: May 06, 2024

GitHub Link:
https://github.com/embedded-systems-1-spring-2024-labs/final-project-NehaMurthy21

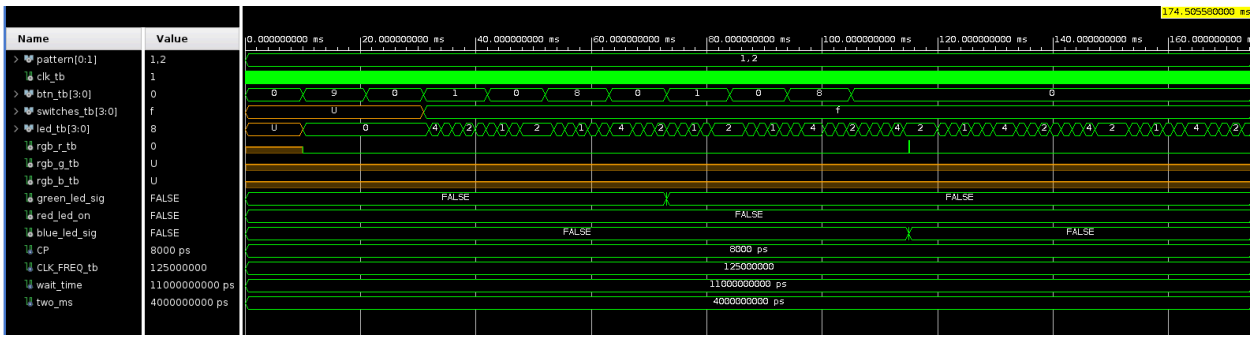
Introduction: **As an introduction, this project was intended as a lock and even though it isn't designed the way I wanted it to, it can still act as a lock in a way. The two pmods that I used are the SSD and the Keypad. The seven segment display allows one or two numbers/letters to be shown. The keypad, being arranged in a matrix form, causes the idea of rows and columns to be integrated into the design. To introduce my project, my project is essentially a lock that is fun and entertaining but also helpful in medical facilities and studies. The game, Simon Says, entertains people of all ages, but also is very beneficial when it comes to studies of cognitive abilities and development. I have mentioned about this in my presentation and have also linked an article that I saw about it. The keypad is an extension (was originally meant for another design that I couldn't finish). The idea is, Simon Says game is like a locking mechanism for a safe and someone needs to open it quickly, and not play the game, they are able to input the number of letters in order to open the lock. I couldn't integrate both the project into one, however, running the two .bit files would demo it.**

Theory of Operations:



Simulation:

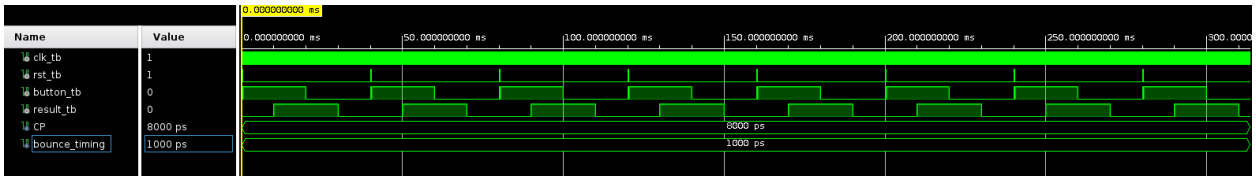
Simon Game Top Level Schematic



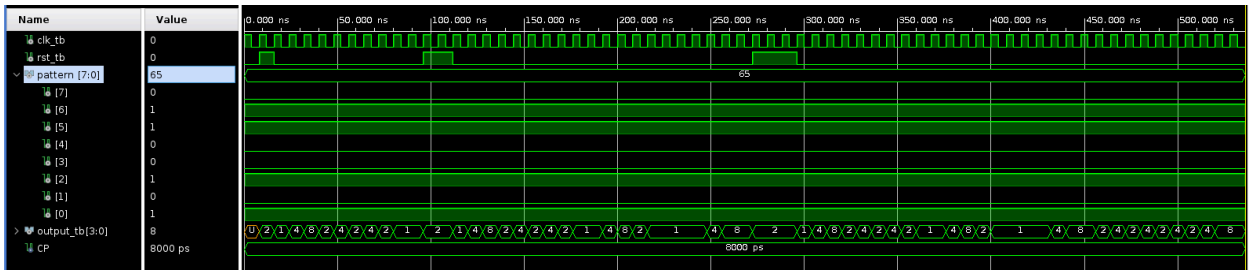
This testbench shows that there will be a time where the correct and that's why there is a small gap between the two falses in the gree_led_sig. Then, after a while, it continues until a red-led_light blinks (6th line) and then immediately, the blue_led_light will blink

as well (shown by the gap between the two falses.) The green and blue_led_light signals are showing undefined in my testbench, however, that is the reason I added other signals below like the green_led_sig that allows it to show if the led turns on or off.

Debounce:

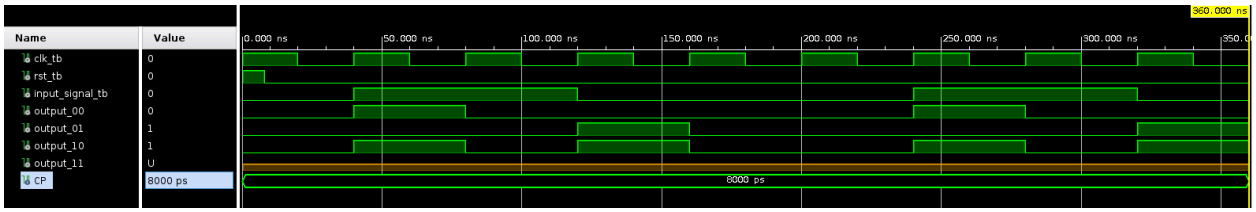


Random_Pattern_Generator



Shows a random pattern generated with respect to the clock. The output will always be 1,2,4,8 because the buttons that we would have to press correspond to 1,2,4,8. We can't have a pattern with the number 3 showing or then we would have to click two buttons at the same time.

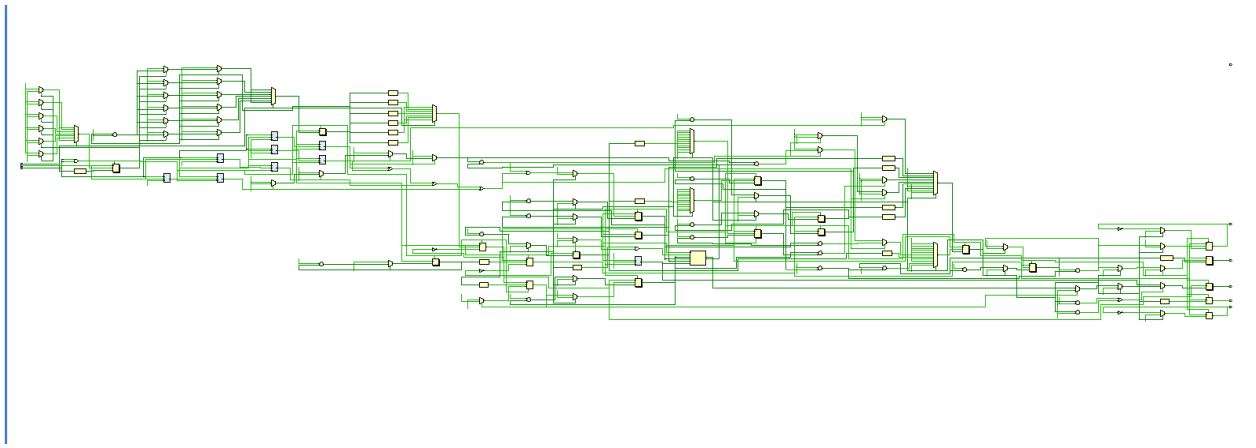
Pulsing_ Detector



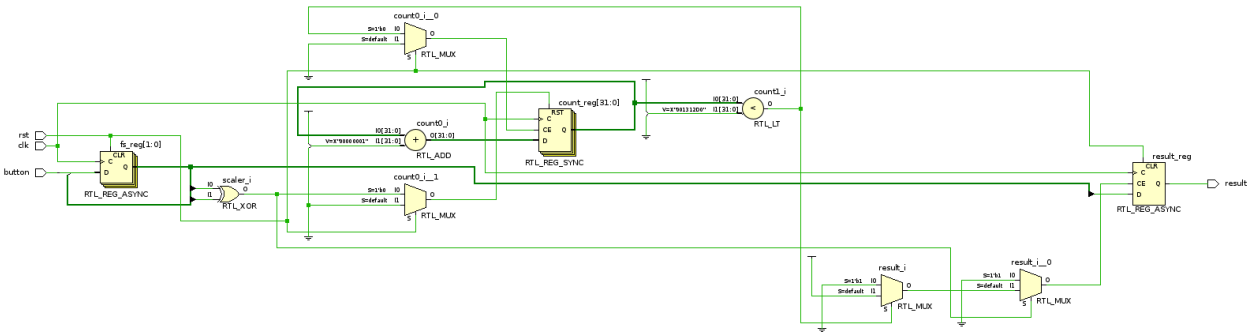
Schematics:

RTL

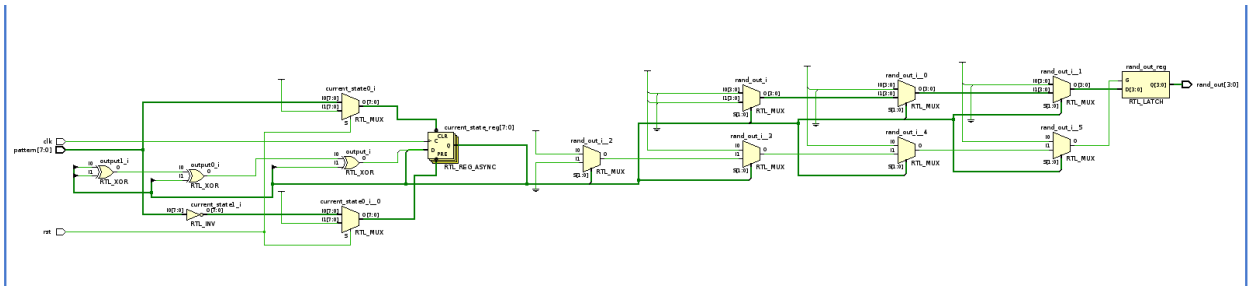
Simon Game Top



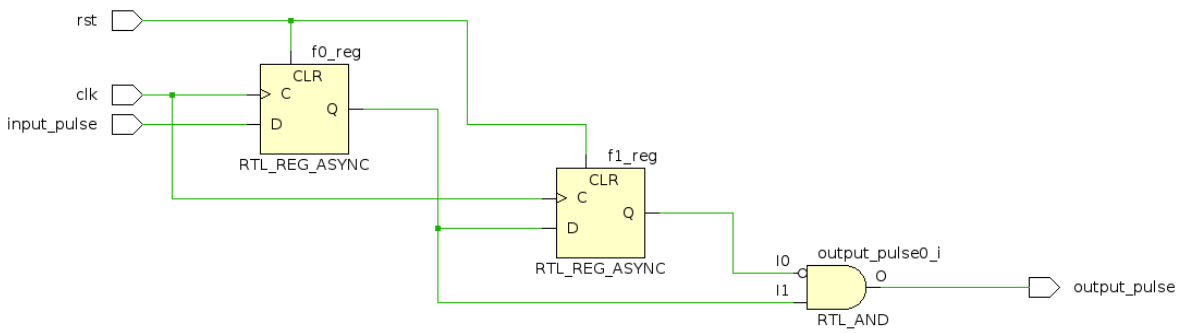
Debounce



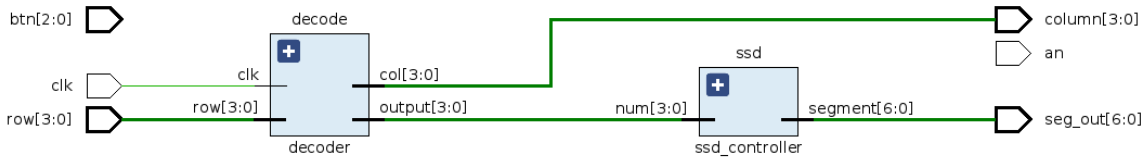
Random_Pattern_Generator



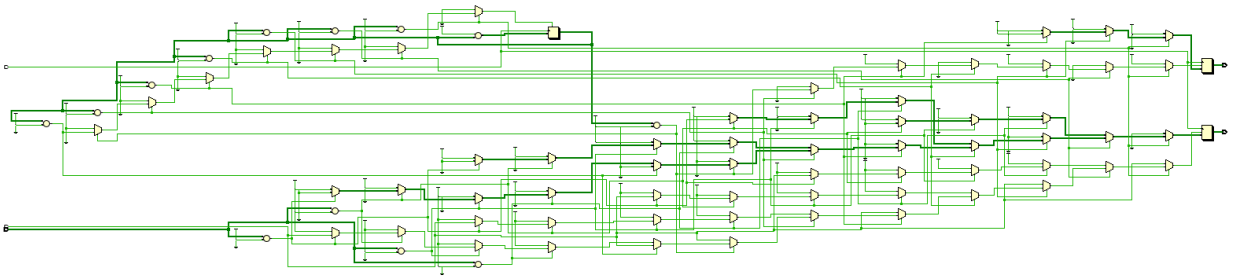
Pulse_Detector



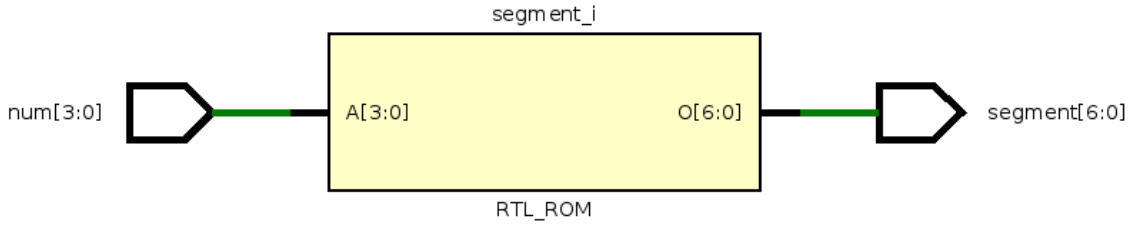
Keypad:



Decoder:

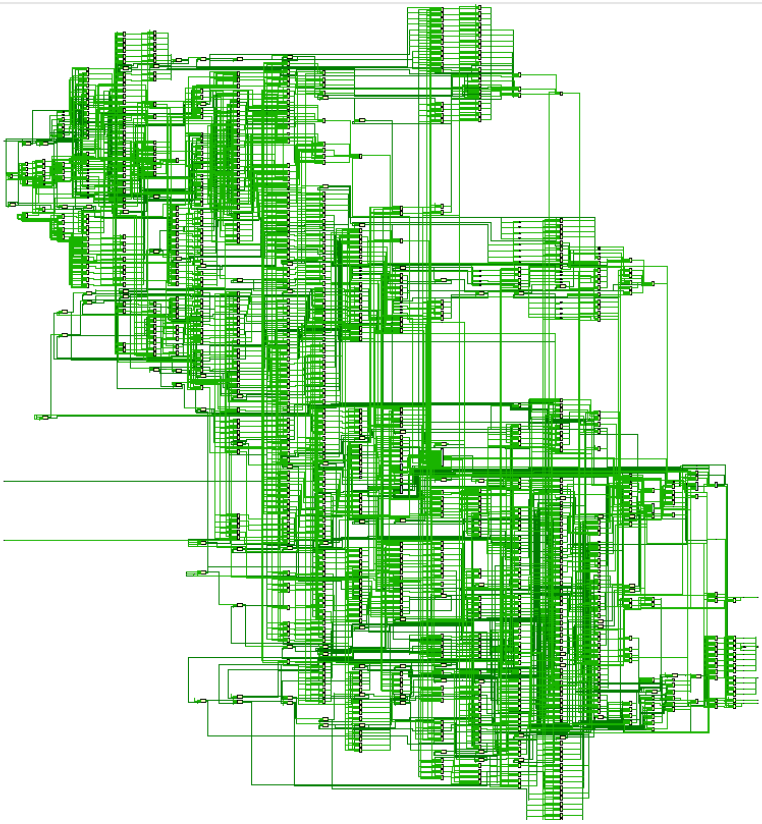


SSD:

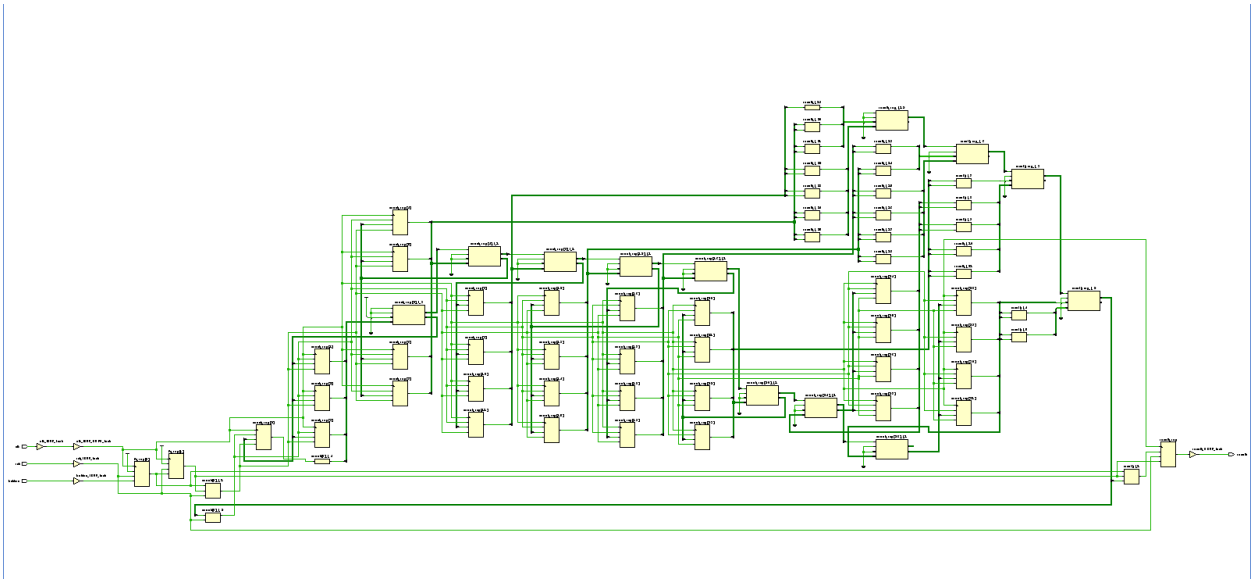


Synthesis

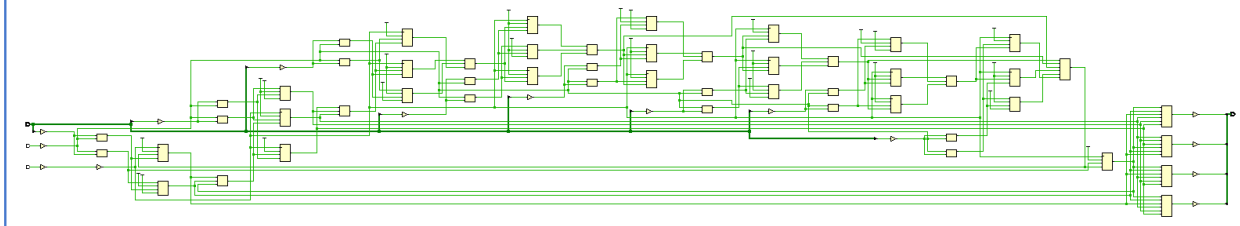
Simon Game Top



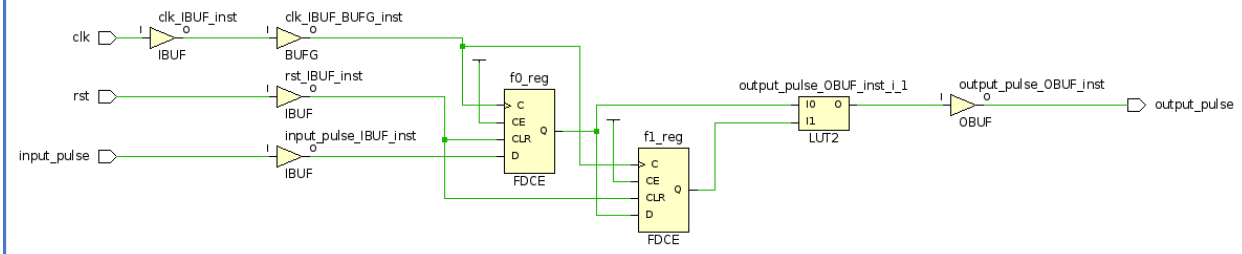
Debounce



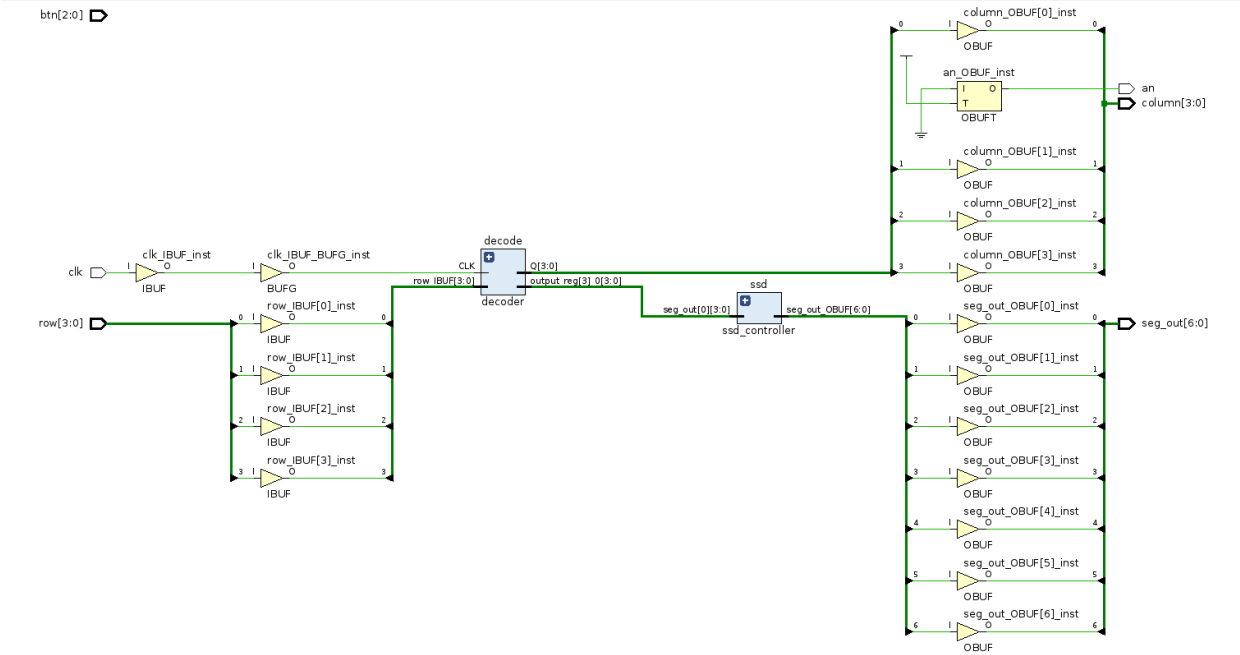
Random_Pattern_Generator



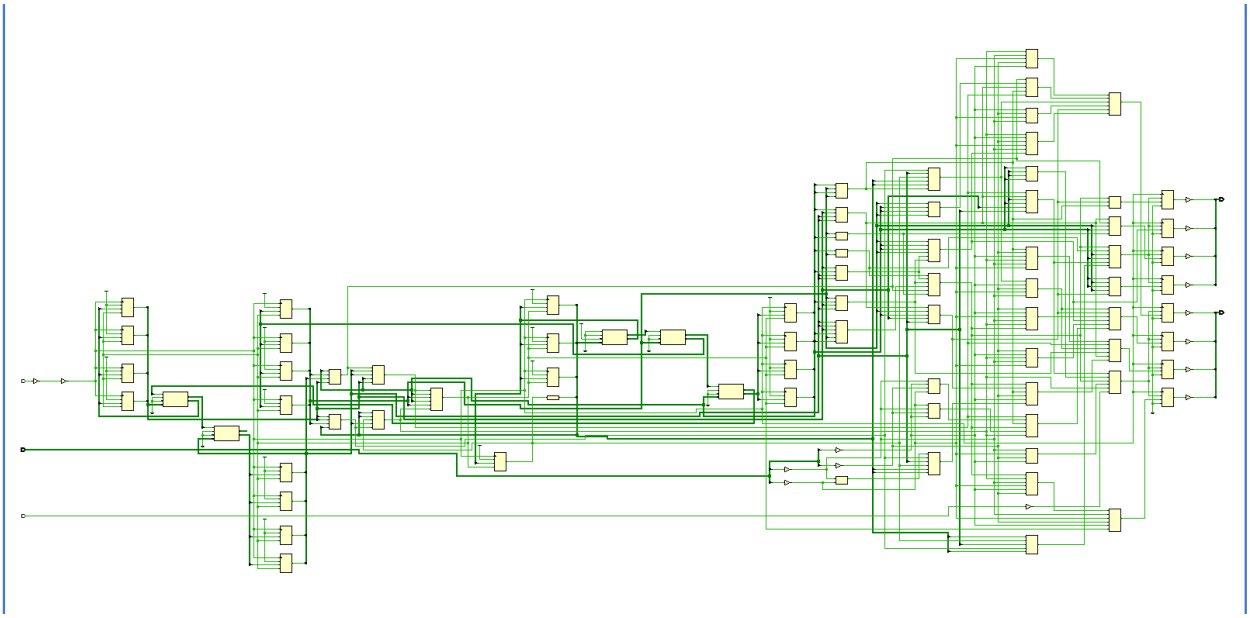
Pulse_Detector



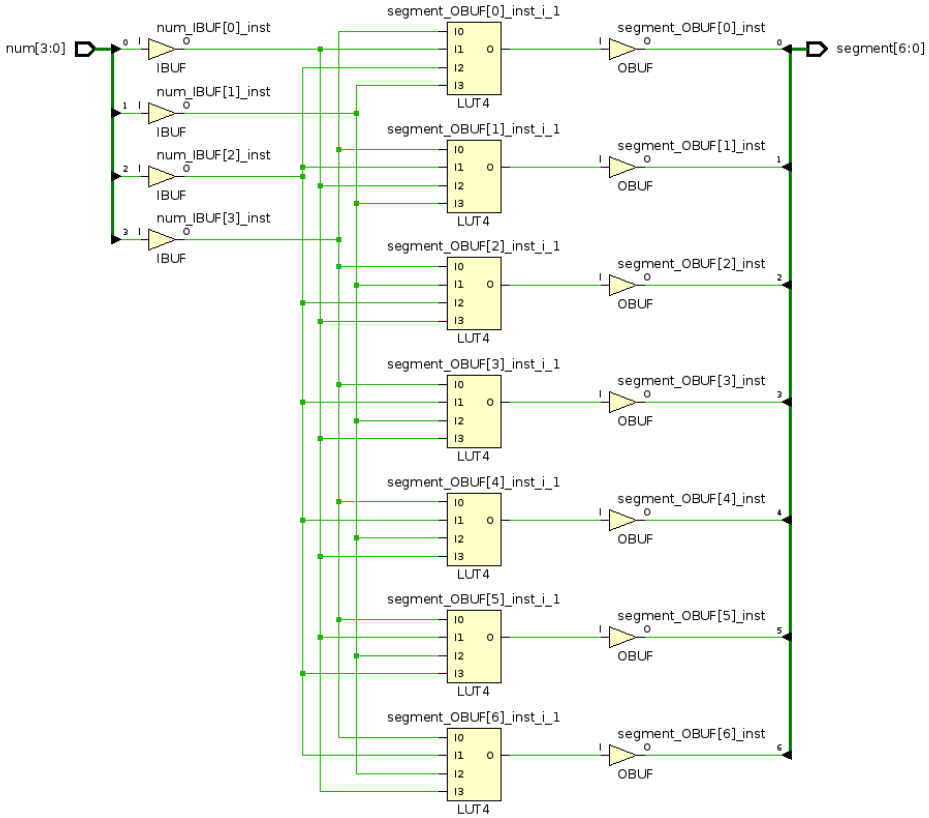
Keypad:



Decoder:



SSD:



Messages | Log | Reports | Design Runs

Post Synthesis Utilization

Simon Game Top

Utilization — **Post-Synthesis** | Post-Implementation

Graph | **Table**

Resource	Estimation	Available	Utilization %
LUT	582	17600	3.31
LUTRAM	8	6000	0.13
FF	528	35200	1.50
IO	24	100	24.00
BUFG	3	32	9.38

Debounce

Utilization — **Post-Synthesis** | Post-Implementation

Graph | **Table**

Resource	Estimation	Available	Utilization %
LUT	18	17600	0.10
FF	35	35200	0.10
IO	4	100	4.00
BUFG	1	32	3.13

Random_Pattern_Generator

Utilization — **Post-Synthesis** | Post-Implementation

Graph | **Table**

Resource	Estimation	Available	Utilization %
LUT	28	17600	0.16
FF	16	35200	0.05
IO	14	100	14.00
BUFG	1	32	3.13

Pulse_Detector

Utilization — **Post-Synthesis** | Post-Implementation

Graph | **Table**

Resource	Estimation	Available	Utilization %
LUT	1	17600	0.01
FF	2	35200	0.01
IO	4	100	4.00
BUFG	1	32	3.13

Keypad:

Utilization — **Post-Synthesis** | Post-Implementation

Graph | **Table**

Resource	Estimation	Available	Utilization %
LUT	38	17600	0.22
FF	28	35200	0.08
IO	17	100	17.00
BUFG	1	32	3.13

Decoder:

Utilization — **Post-Synthesis** | Post-Implementation

Graph | **Table**

Resource	Estimation	Available	Utilization %
LUT	36	17600	0.20
FF	28	35200	0.08
IO	14	100	14.00
BUFG	1	32	3.13

SSD:

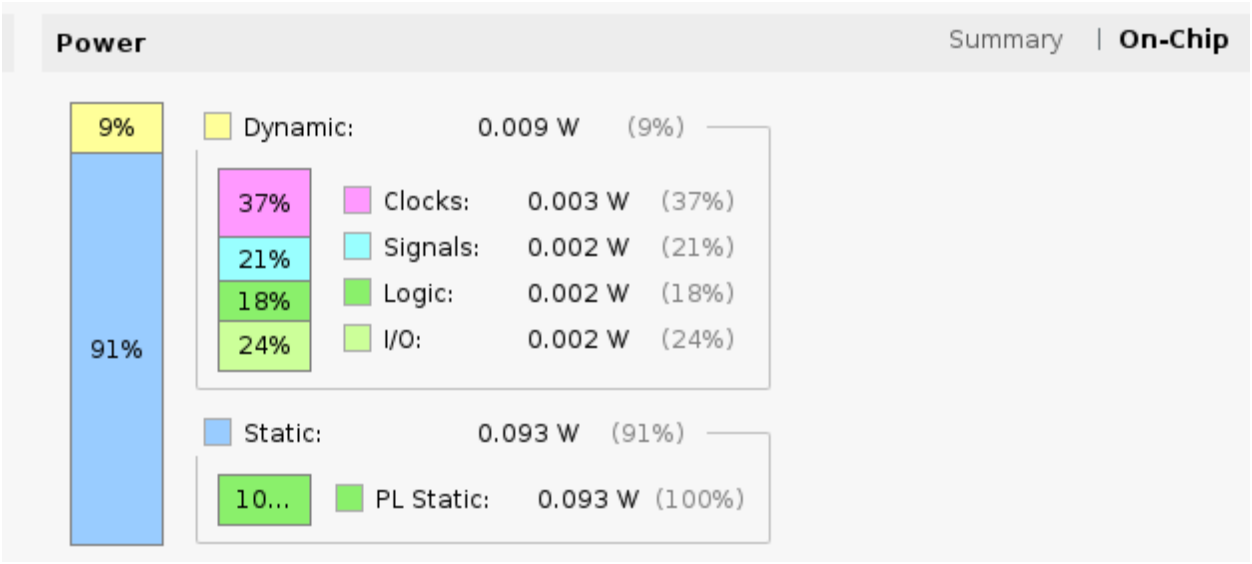
Utilization — **Post-Synthesis** | Post-Implementation

Graph | **Table**

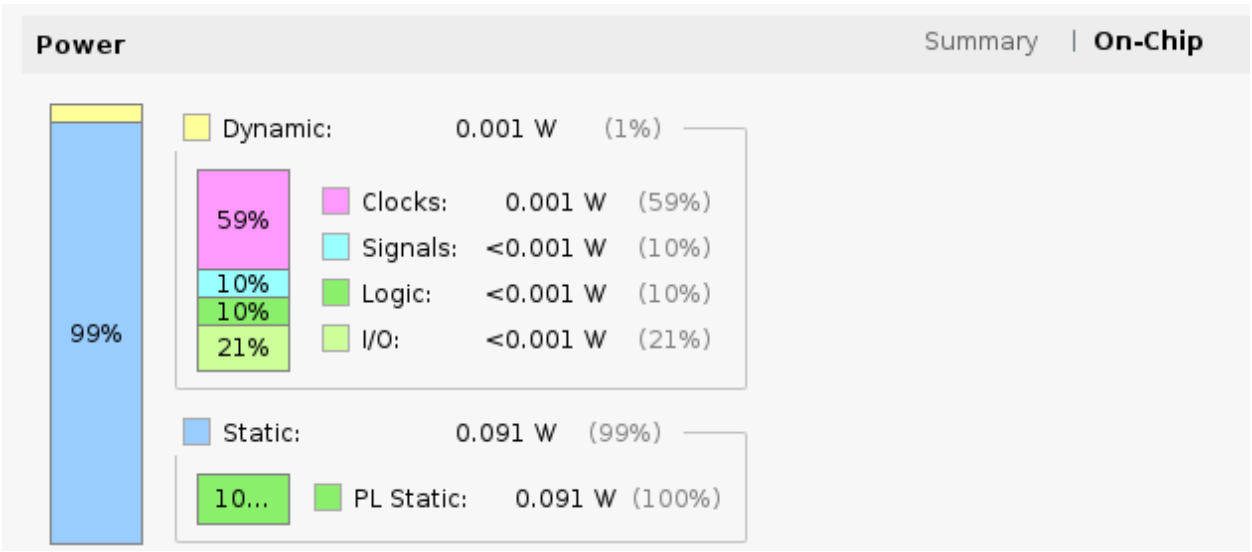
Resource	Estimation	Available	Utilization %
LUT	4	17600	0.02
IO	11	100	11.00

On Chip Power

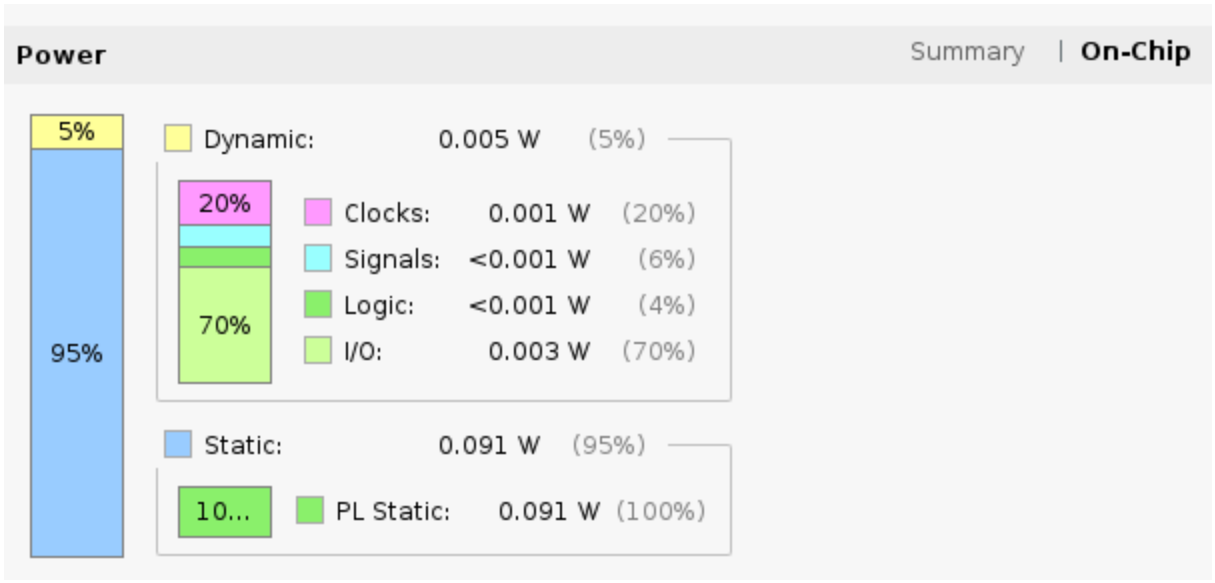
Simon Game Top



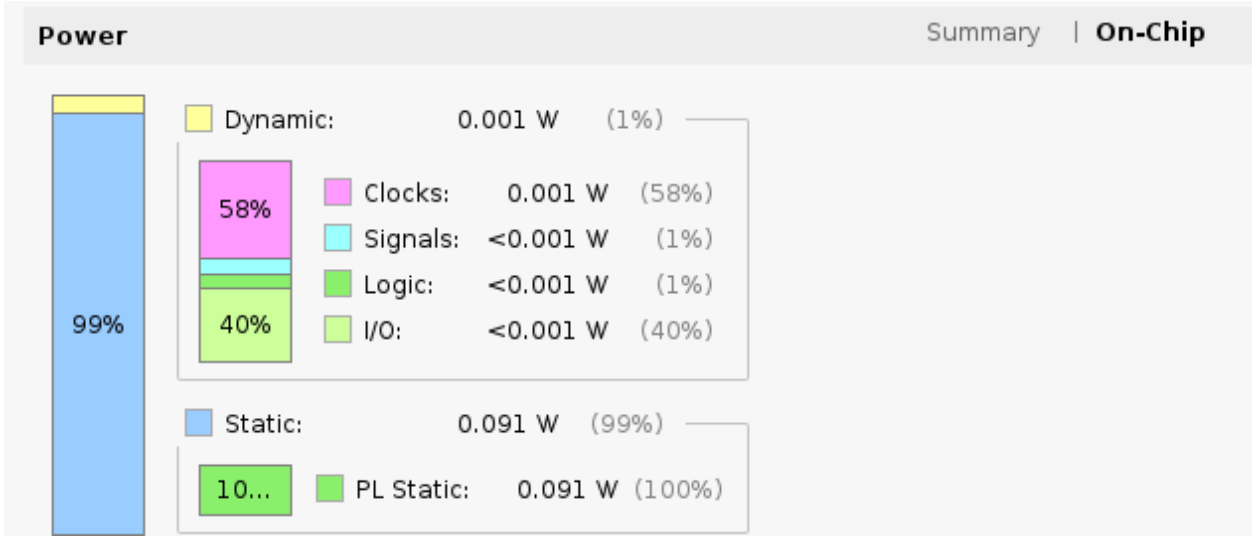
Debounce



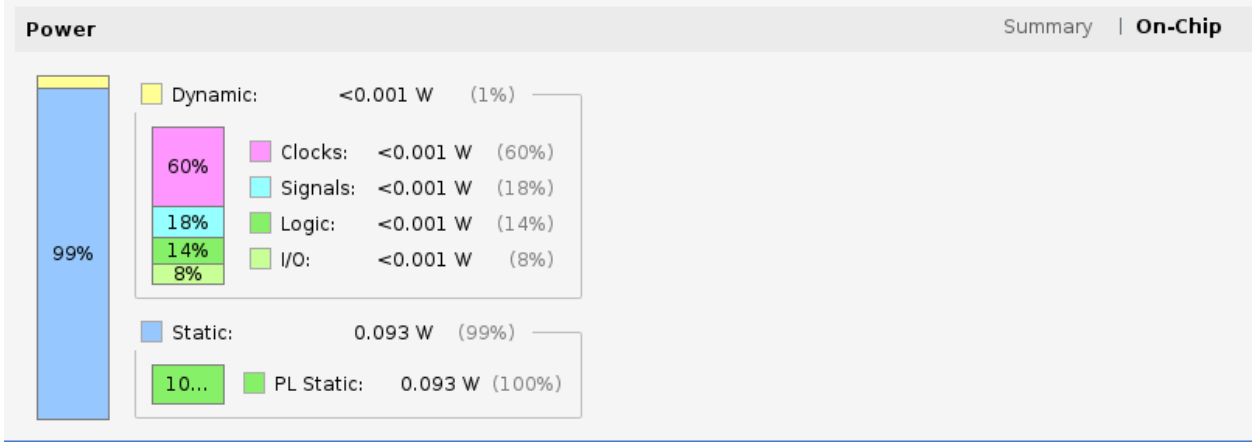
Random_Pattern_Generator



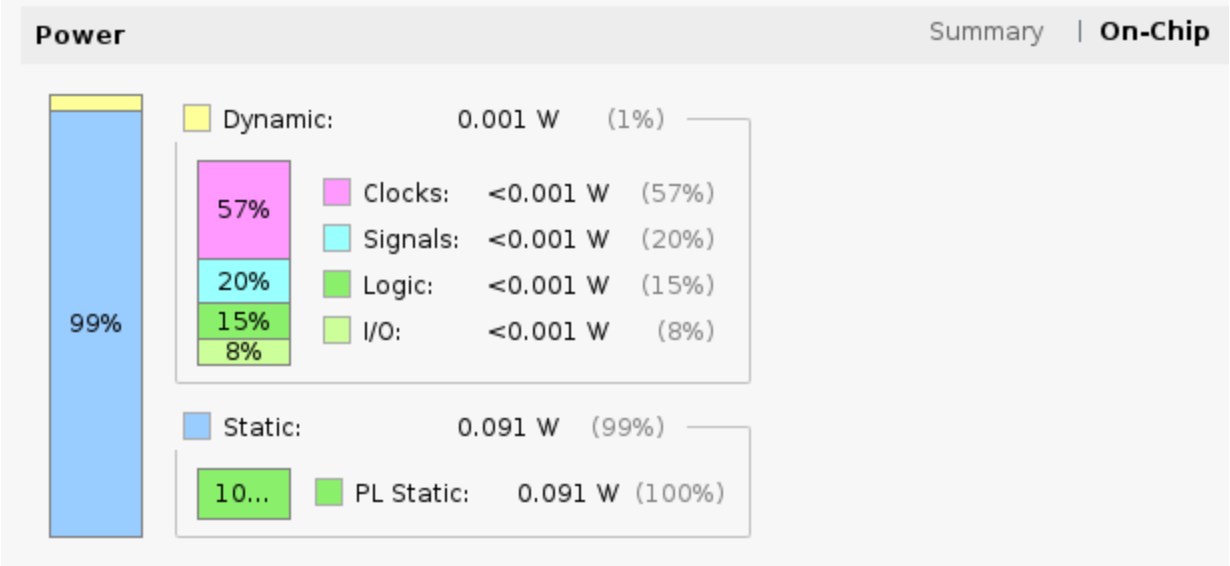
Pulse_Detector



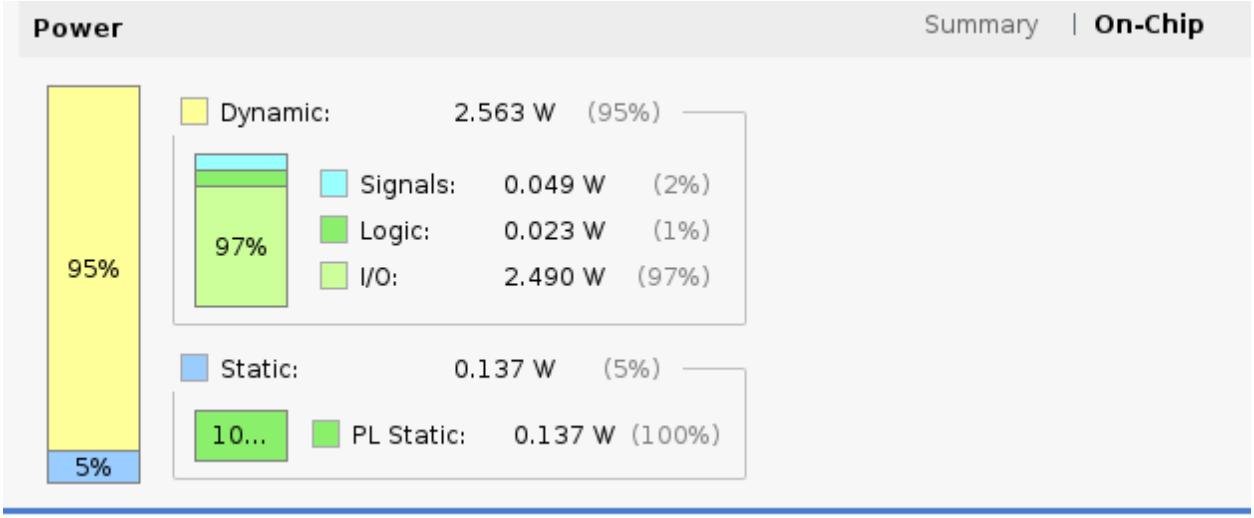
Keypad:



Decoder:



SSD:



Nonfunctional requirements: Using a device that has enough resources to work with a design is very important. Say, a fpga board doesn't have enough LUT's or IO's or flip flops, assume that they are very low, but the design requires a high amount, these are some factors that need to be kept in mind. In this project, looking at the total on chip power, I saw that the highest dynamic power was the SSD with a dynamic power of 2.563W. If the FPGA didn't support it, then this design would not have worked. The top level (Simon_says_game) had 582 LUT's having the highest amount, which indicates that it has the highest level with the main logic of the many components.

Since the two PMODS are SSD and KYPD, we can see the requirements that are needed

SSD:

- Two 6-pin PMODS connectors
- Voltage must be within 2.7V and 5.25 V (recommended that it be operated at 3.3V)

KYPD

- 3.3V is the recommended voltage
- 16 button keypad with hexadecimal format from 0-F.
- Pressing three or more keys together can cause false negatives or positives, as resistors are connected in parallel which could alter the voltage divider circuit.

XDC Constraint File Changes: In my XDC file, I had to open up the switches for the speed of the LED showing the pattern. I opened up the buttons as I needed the buttons to press the pattern that the LED gives us. I opened up the LED as that shows the pattern. After researching a bit about the boards to understand what each part did, I found out that the Zybo 7010 had an extra RGB LED port that can turn red (for incorrect), green (correct) and blue (score). Lastly, I integrated the seven segment display in my XDC. I also had to include an "an" as an anode for this, which was tied to a floating signal. For the keypad, since it is designed like a matrix, I had to split it into rows and columns and then map it to the Zybo XDC.

Conclusion : In conclusion, this project was a Simon Says where the user would have to copy the exact same pattern that the LED shows. If the pattern is the exact same, then the led will blink a green light and if it isn't right, it will blink a red light. After the red light, a blue light will blink. My original project idea was actually a locking mechanism. The original idea was Simon Says would be a game but a lock for small kids. In my presentation, I had explained that Simon Says has been used in multiple researches where researchers have said that the game improves cognitive abilities and development. In that case, this can serve as a game, but also a lock, which I will discuss in the follow up. These two components are able to integrate with each other, however, getting close to the deadline and errors occurring, I didn't want the stuff working to be affected. That is the reason why I couldn't integrate the two components together.

Follow-up: As a follow-up, I wanted this project to become a fun locking mechanism that can be helpful in the medical world, in the entertainment world and the security one. However, since I couldn't get the validations for the locking mechanism, this is an aspect that would need to be improved upon. Due to time constraints with finals from other classes, I wasn't able to work on it with full effort which is why there is no validation for the locking and unlocking mechanism.