July 2019

Simon Game

Instruction Manual

Louis Alfred Tacata

Table of Contents

1. Overview

The project described herein is an FPGA/SoC implementation of the Simon game based of the original 1978 variant by Milton Bradley (now Hasbro). Using the Zybo Z7-20 Development Board, it features the same buttons, sounds, and skill level options the Simon game is known for. With over 550,000 total sequences, this implementation allows for a fun and fair gameplay for people of all ages.

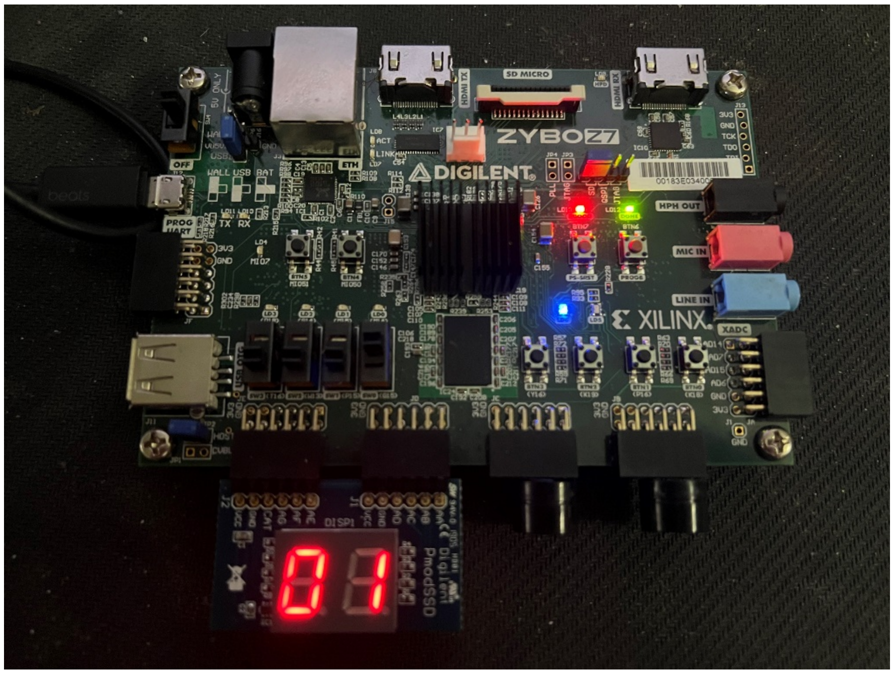

Figure 1: Simon game after first correct input

2. Setup

2.1. Requirements

- <u>Software</u>

  You must have the Vivado Design Suite 2019.1 installed in your device in order to load and modify the game.

- <u>Hardware</u>

  The Zybo Z7-20 Development Board is used as the primary medium of the game. In addition, to allow for level displays and sound generation, the seven-segment display PMOD and two passive buzzers are also utilized.

- <u>Source Files</u>

  The project files can be found at https://github.com/takatz28/Simon-Game.

2.2. Board Setup

Figure 2 demonstrates how the peripherals are placed in the PMOD ports of the Zybo board. In the case of the buzzers, make sure that the pin under the (+) symbol is oriented to the left.

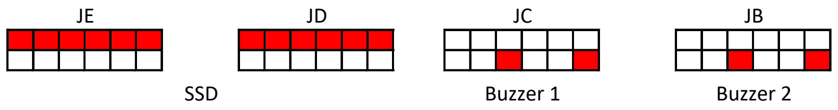

Figure 2: *Used (red) and unused pinouts in the onboard PMOD ports*

2.3. Loading the Game

There are two ways of loading the Simon game. The microSD card option allows the user to play the game as soon as the board is turned on. On the other hand, loading the project through the Vivado Design Suite lets the user perform upgrades and modifications.

2.3.1. Using the microSD card

Once the source files are extracted, copy the contents of the 'sdcard' folder in the microSD card. Afterwards, insert it into the port underneath the Zybo board. Before turning the board on, make sure that the JP5 jumper is set to SD. When the board is finally powered on, the game loads into idle mode.

2.3.2. Using the Vivado Design Suite:

To load the project:

1. Open Vivado 2019.1.
2. Using the tcl console, type the following:

```
cd <change to extracted_folder>/<verilog or vhdl>
source ./Simon_VHDL.tcl <if using the VHDL IPs, or>
source ./Simon_Verilog.tcl <if using the Verilog IPs>
```

3. Create a new HDL wrapper for the block design.
4. Run synthesis and implementation, then generate the bitstream.
5. Go to File > Export > Export Hardware. Make sure that the "Include Bitstream" box is marked.
6. Go to File > Launch SDK.
7. Once the SDK is launched, go to File > New > Application Project.
8. Fill up the form with the following details:

Field	Value
Project Name	"Your preferred name"
OS Platform	Standalone
Hardware Platform	design_1_wrapper_hw_platform_0
Processor	ps7_cortex a9_0
Language	C

9. Click on Next, select the *Empty Application* template, then click Finish.
10. Copy the contents of `sdk_files` into the `src` folder under "Project Name".
11. Under Project Explorer, right-click on the project folder, click on *C/C++ Build Settings*.
12. Under *Libraries*, click on *Add*, then type `m`.
13. Program the FPGA.
14. Right-click on the project folder, click on *Run As > Launch On Hardware* (GDB).

3. Gameplay

3.1. Skill Levels

To set the game's skill level, toggle the leftmost switch. Once toggled, the two RGB LED lights will turn cyan. The pushbuttons on the board's bottom-right side represent the maximum sequence length for each of the four levels: 8, 14, 20, and 31, respectively.

3.2. Simon Says

Toggle the rightmost switch to turn on the game. Simon will give the first signal. When the signal is played correctly by the user, Simon will duplicate the signal, then add a new one. If the sequence is repeated correctly, the game continues, otherwise, the game is over and produces a single buzz. Every fourth signal starting at the 5th signal, Simon's tempo increases. When the game is won, for skill levels 1-3, six beeps will play, with the LEDs and SSD outputs flashing accordingly, and the game restarts after the sixth beep. On the other hand, for level 4, once Simon is beaten, it will play a descending scale and shut down.

3.3. LED and SSD Status

Status	LED6	LED5	SSD
Board is in idle mode	🟪	🟪	N/A
Simon is in skill select mode	🩵	🩵	S1, S2, S3, or S4
Welcome message	⬜	⬜	H-E-L-L-O
Simon is showing the sequence pattern/waiting for player input	🟦	⬜	Current level between 00 and 31
Player is in the process of repeating pattern	⬜	🟨	
User's pattern input is correct	⬜	🟩	Increment above number
User's pattern input is wrong (game over)	🟥	🟥	FF (blinking)
User wins the game	🟩	🟩	88 (blinking)

4. Future Improvements

Unlike its original counterparts, this implementation of Simon is only limited to one single game. Also, the time limit between input presses was eliminated, allowing players to keep track of the current sequence easier. Therefore, these improvements are suggested for the project's next version:

- Addition of a 4- or 5-second timer to each button press for a faster game pace.
- Include the game modes "Player Add" and "Choose Your Color".
- Use an audio codec for sound generation instead of pulse wave modulators.