# Lab 3
# Finite State Machines
# The Simon Game

With Basys 3 as your target platform, design, verify, implement, and test the Simon Game, defined below using the following specification:

**Game introduction:**

Simon is a game designed to test your memory. In this game, there are 4 distinct light sources and 4 corresponding inputs from the user. The game displays a sequence of patterns using available light sources, and the user has to repeat this sequence from memory using the corresponding buttons.

The game starts at level 1, where the sequence length is 1. Each subsequent level adds an additional pattern to the sequence, thus increasing the sequence length by 1. There can be a maximum of 15 levels. A level is cleared if the user enters the sequence of patterns precisely as displayed by the game before. If the user fails to do so, then the game ends at a particular level.

**Game patterns:**

The game mainly focuses on the user memorizing a sequence of patterns. There are 4 possible patterns as illustrated in the right half of Figs. 1 through. 4 We can name the patterns as

1. Left (Fig. 1)
2. Up (Fig. 2)
3. Right (Fig. 3)
4. Down (Fig. 4)

Fig. 1: Left Pattern

Fig. 2: Up Pattern

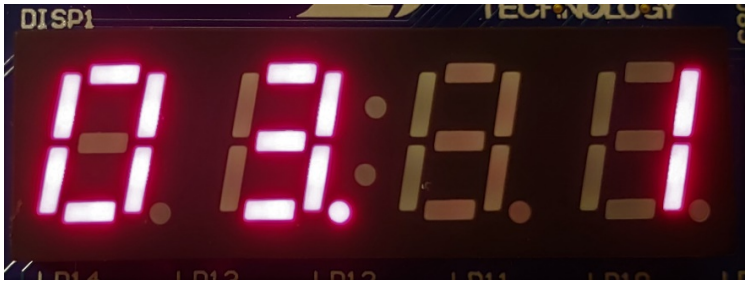

Fig. 3: Right Pattern


Fig. 4: Down Pattern

There are also 4 buttons on the Basys 3 board that will correspond to the patterns mentioned above.

1. Left : BTNL
2. Up : BTNU
3. Right : BTNR
4. Down : BTND

**Game score:**

The Game score is the same as the current level of the user. For example, if the user is at level 3 then the game score is 3. The level controls the number of patterns displayed in a given sequence. So, if the current level of the user is 3 then there will be 3 patterns in the sequence shown to the user.

**Game display:**

There are multiple messages that need to be displayed in the game.

1. The main display of the game is as follows
    - The two rightmost 7-segment displays show the current pattern
    - The two leftmost 7-segment displays show the current level
    - The decimal point of the second 7-seven display from the left is lit to show the division between the two sections

2. Before the start of the game the message “bEgn” (standing for “begin”) should be displayed using four 7-segment displays. It should be a blinking text, with each character occupying the corresponding 7-segment display.

Fig. 5: Begin Message

3. When a user clears a level, a message “PASS” should be displayed using the 7-segment displays. It should be a static text, displayed for 2 seconds, with each character occupying the corresponding 7-segment display.

Fig. 6: Pass Message

4. When the user fails to clear a level by entering the wrong pattern the message “FAIL” should be displayed. It should a static text, displayed for 2 seconds, with each character occupying the corresponding 7-segment display.

Fig. 7: Fail Message

5. At the end of the game the game-over animation should be shown, which is the message “End”. This message should blink for 2 seconds.

Fig. 8: End Message

**Game start and reset:**

The game should accept the following inputs related to the start of the game and reset:

- If BTNC is pressed shortly and the game is in the reset state, then, the game should start.
- If BTNC is pressed shortly and the game is not in the reset state, then the press of this button should have no effect on the functionality of the circuit.
- **Bonus Task:** If BTNC is pressed for more than 2 seconds, then the game should be stopped and enter the reset state.

**Game difficulty:**

The game has four levels of difficulty. The current level of difficulty controls the delay between the patterns displayed during the game. This level is set using the two leftmost toggle switches, SW 15 & 14, and corresponds to the following delays:

- 00: delay of 2.0 Seconds
- 01: delay of 1.5 Seconds
- 10: delay of 1.0 Seconds
- 11: delay of 0.5 Seconds

The level of difficulty should be set before the start of the game and should stay the same throughout a given game.

**Game dynamics:**

Before the start of the game the system is in the reset state. In this state, the message “bEgn” keeps blinking. All games start at the reset state and at level 1. The game proceeds as follows once the game start button is pushed:

- The game generates a sequence of patterns of the length defined by the current level. Each sequence of patterns is a superset of the sequence displayed at the previous level, extended with one additional pattern.
- This sequence is shown to the user with a delay between patterns defined by the game difficulty level.
- The game then waits for user input. The user must enter the pattern as seen previously and the game continuously monitors the input for errors from the user.
- If any error is detected, then the user has failed, and the message “FAIL” is shown for 2 seconds. The game then shows the game-over animation and goes to the reset state.
- If the user has successfully entered the correct sequence, then a message “PASS” is shown for 2 seconds. The game then proceeds to the next sequence, of the length increased by one.

- If the user has cleared the level 16, then after the "PASS" message, the game-over animation is shown, and the game proceeds to the reset state. If the level is less than 16 then the game repeats all the above steps.

**Pseudo random number generator:**

The game should use an LFSR-based pseudo random number generator that generates patterns for each level. The implemented LFSR should have the length L=8, and be described by the Connection polynomial, $C(D)= 1+D^4+D^5+D^6+D^8$, with the period $2^8-1=255$.
Before the generation of each sequence, LFSR should be initialized with the values determined by the positions of switches SW 7..0. The initial state equal to 8 consecutive zeros is not permitted.
Each pattern is determined by the current two least significant positions of LFSR, $s_1$ & $s_0$, according to the following convention: 00=UP, 01=RIGHT, 10=LEFT, 11=DOWN. After one pattern is generated, LFSR is shifted by two positions.
This design ensures that the sequences of patterns look random and are supersets of sequences from previous levels.

**Button debounce:**

Each button used in the game must have an associated debouncing circuit to ensure the accurate reading of the user input.

**Required tasks:**

1. Draw a block diagram describing the required Datapath.
2. Draw Algorithmic State Machine (ASM) charts describing the required Controller. Please note that multiple state machines, working in parallel, can be used to achieve the required behavior.
3. Translate the block diagram and ASM charts to VHDL.
4. Develop a simple testbench with two versions of timing constants, one used for simulation, and the other used for the actual operation of the circuit on the board.
5. Perform functional simulation of your code (up to the Simon level 4).
6. Synthesize your code using Xilinx Vivado.
7. Prepare the correct XDC (Xilinx Design Constraint) file.
8. Implement your circuit using Xilinx Vivado.
9. Check thoroughly all implementation reports. Pay attention to timing, resource usage, and pin allocations.
10. Perform post-synthesis simulation of your circuit using ModelSim or Vivado Simulator.
11. Perform static timing analysis.
12. Check very carefully your pin allocations listed in the report files, and only if these pin allocations are correct, download your bitstream to the FPGA board.
13. Test the operation of your circuit experimentally using the Basys 3 FPGA Board.

**Deliverables**

1. All block diagrams and ASM charts.
2. All source files used for synthesis and implementation of your circuit.
3. A simple testbench.
4. User constraint files.
5. All synthesis and implementation report files.
6. RTL netlist.
7. Simulation waveforms from the functional and post-synthesis simulations, proving the correct operation of your circuit (in the PDF format).

8. Report file from the static timing analysis.
9. Your own report containing at least the following additional information:
    - Resource utilization.
    - Minimum clock period and maximum clock frequency after synthesis and after implementation.
    - List of any deviations from the original specification.
    - Difficulties encountered and lessons learned.

**Important Dates**

| | **Tuesday Section** | **Wednesday Section** | **Friday Section** |
|---|---|---|---|
| **Hands-on Sessions and Introductions to the Experiment** | 02/12/2019<br>02/19/2019 | 02/13/2019<br>02/20/2019 | 02/15/2019<br>02/22/2019 |
| **Deliverables Due** | **02/26/2019**<br>**8:30am** | **02/27/2019**<br>**6:50pm** | **03/01/2019**<br>**8:00am** |
| **Demo and Q&A** | **02/26/2015**<br>**9:00-11:40am** | **02/27/2019**<br>**7:20-10:00pm** | **03/01/2019**<br>**8:40-11:20am** |