Game Summary

My project is a single-player memory game implemented on the Nexys A7-100T FPGA board. The player starts the game by pressing the center button. If the game starts successfully, all the LEDs blink as visual confirmation.

Once the game starts, the player enters Level 1 (displayed as 00 on the 7-segment display). The LEDs turn off, and after a short delay, 6 random LEDs turn on. They remain on for a few clock cycles, then turn off: this is the signal for the player to replicate the pattern using the switches. When the player is ready, pressing the center button initiates the score evaluation phase.

Scoring works as follows:

- +1 for each correct match (LED was ON, switch is ON)
- -1 for each incorrect match (LED was OFF, switch is ON)
- No penalty if the LED was ON and the switch was OFF (Levels 1 and 2 only)

When the player reaches a score of 16, the game automatically proceeds to Level 2 (shown as 01). The flow is the same as Level 1, except that 8 LEDs are used instead of 6. The score resets to zero at the start of the level.
Upon reaching 16 points again, the player enters Level 3 (02). It also uses 8 LEDs and the same sequence of pattern display and evaluation. However, Level 3 introduces a stricter rule: the player is now penalized -1 if they forget to activate a switch corresponding to an ON LED.

After each evaluation, it is the player's responsibility to manually turn off all the switches. Once all switches are in the OFF state, a new pattern is generated and the game continues.

Scoring Rules Summary

- Pattern = 1 and Switch = 1 → +1
- Pattern = 0 and Switch = 1 → -1
- Pattern = 1 and Switch = 0 → 0 (Levels 1 & 2), -1 (Level 3)
- Pattern = 0 and Switch = 0 → 0

Pattern Generation

The random LED pattern is generated using the Fisher–Yates Shuffle algorithm, which selects a fixed number of unique positions (6 or 8 depending on the level) to turn ON.

A current limitation of the system is that the pattern generated is always the same each time the game starts. This is because the internal LFSR that seeds the randomness is always initialized with the same value at power-up or reset (eg. 0000000000111111).
I will dive more precisely in the behavior in the final report.

To increase randomness, I'm considering using the number of clock cycles between pattern generation and the moment the player presses the center button. This time interval could be used to influence the LFSR seed dynamically, making the pattern generation less predictable.

Final Considerations

I'm aware of the limitation related to pattern repeatability, and I'm currently waiting for your feedback before proceeding with further refinement.

Due to my pc being based on an ARM architecture, which is not compatible with Vivado, I've been working across two computers:

- One for writing the VHDL code, creating testbenches, and generating VCD files
- Another for synthesis and simulation with Vivado

That said, I've kept everything organized and up-to-date. All files have been progressively prepared and are stored in a private GitHub repository. If you like, I'd be happy to share access with you.