Lab 8: Conway's Game of Life

Perry Chien, 2261656 3-June-2024

1. User's Manual

Overview

Conway's Game of Life is a cellular automaton where cells on a grid live, die, or multiply based on specific rules. This implementation on an FPGA board features user interaction for initial configuration, running the game, and recognizing specific patterns (blinker oscillations).

Controls & Components

1. Key FPGA Board Components
    - HEX5, HEX4: Seven-segment display.
    - KEY[3:0]: Push buttons for user input.
    - SW[9:0]: Toggle switches for various controls with the game.
    - GPIO_1: General-purpose I/O pins for LED expansion board.
2. LED Display
    - RedPixels and GrnPixels: Represent the cells on the grid.
        - RedPixels: Active cells displayed in red.
        - GrnPixels: Blinking cursor position displayed in green.
    - HEX4 and HEX5: Seven-segment display count of recognized blinker patterns.
3. Controls
    - SW9: Global reset switch. When toggled on, the system resets.
    - SW8: Start/pause switch for the game. When toggled on, the game starts or resumes; when off, it pauses.
    - KEY[3:0]: Navigation and action buttons:
        - KEY3: Move cursor right.
        - KEY2: Move cursor up.
        - KEY1: Move cursor down.
        - KEY0: Move cursor left.
    - SW0: Toggle cell state at the cursor position (light up or light off LED cell).

How to Play

1. Initial Configuration
    - Power on the FPGA board, and ensure the DE-1 LED Display Expansion Board is appropriately connected to the GPIO-1 pins on the FPGA.
    - Ensure SW9 (reset) is toggled on, then off to reset the system.
    - Use KEY[3:0] to navigate the cursor across the grid.
    - Toggle cell state with SW0 at the blinking cursor. An active cell will light up in red.

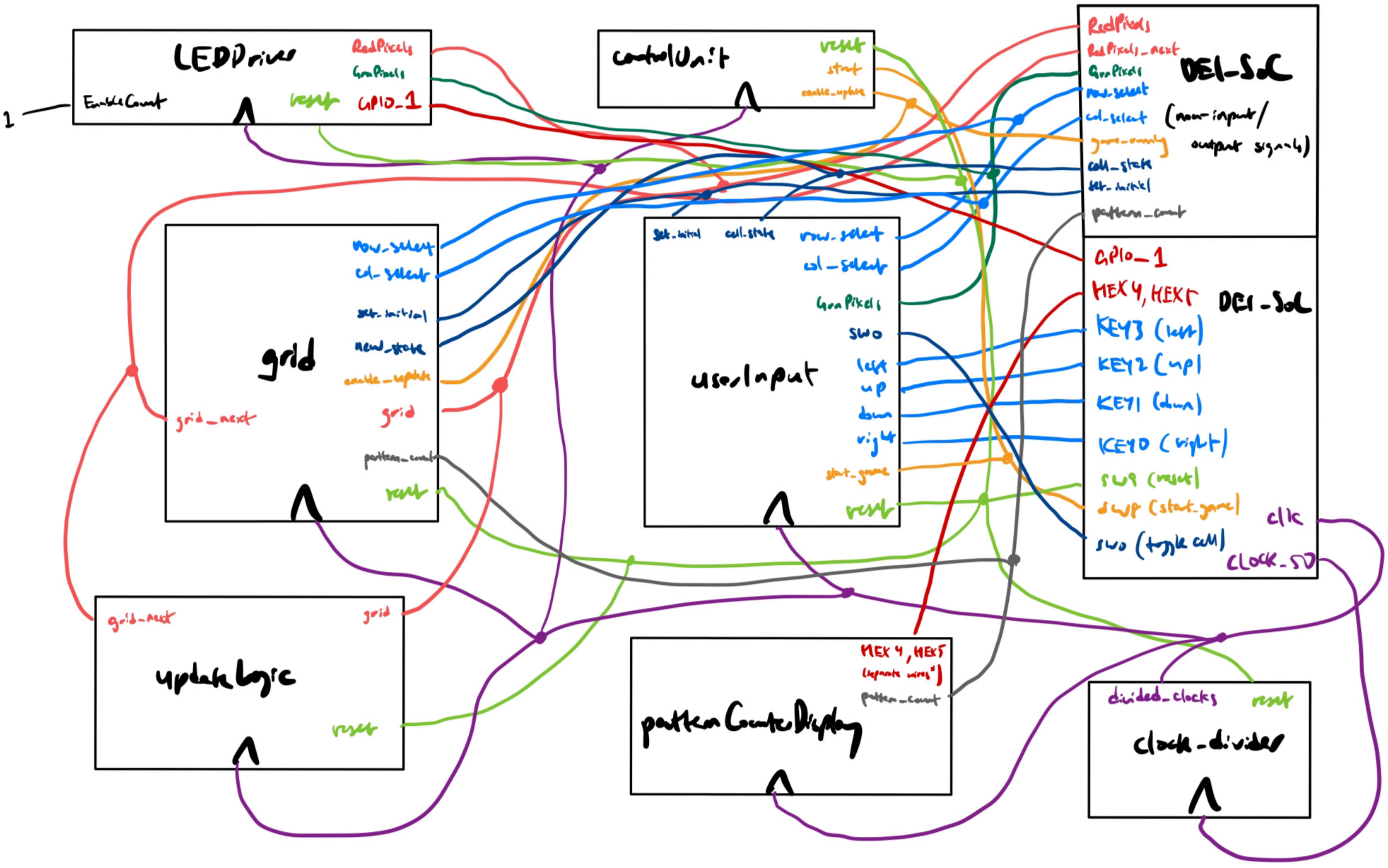
- ○ Configure the initial state of the grid by setting or clearing cells as desired. Populate the LED expansion board with red cells to set the starting cell configuration for the Game of Life.

2. Running the Game
    - Toggle SW8 to start the game. The grid will update based on these rules:
        - Any live cell with fewer than two live neighbors dies (underpopulation).
        - Any live cell with two or three live neighbors lives on to the next generation.
        - Any live cell with more than three live neighbors dies (overpopulation).
        - Any dead cell with exactly three live neighbors becomes a live cell (reproduction).
    - The game will continue to run, updating the grid at each clock cycle.
3. Pausing/Reset the Game
    - Toggle SW8 to pause the game. The current state of the grid will be retained until the game resumes (by toggling SW8 again).

Pattern Recognition Extension

- Feature to recognize & count blinker patterns (three-cell horizontal/vertical oscillations).
- The system scans the grid to detect blinkers and counts when a blinker oscillation is detected. Count is displayed on HEX4 and HEX5, which can count up to 16 blinkers.

2. Block Diagram



3. Market & Usability Analysis

Usability and Ergonomics

The system demonstrates high usability and moderate ergonomic efficiency. The user interface allows intuitive user interaction through simple switch and key inputs. Users can easily configure the initial state of the game using directional keys and a switch that toggles between setting and removing red LED cells, making the setup process straightforward. The use of HEX displays for pattern count also provides clear and immediate feedback on the state of the game.

While the positioning of the keypad controls could be better, the design makes the most out of the provided FPGA hardware and ensures that users do not need technical knowledge to play the game. In particular, the blinking green LED cursor during the initial configuration phase provides visual guidance, which aids with user-game interaction.

Suitability for Goals

The system meets its primary goals of providing an engaging tool for understanding cellular automata. It presents a bug-free implementation of Conway's Game of Life, serving as a useful tool for demonstrating principles of computation, emergence, and system dynamics. In addition, the extension of blinker-pattern recognition adds a layer of complexity to the system. Despite the simplicity of blinker oscillations in the game’s context, this feature offers further educational value on how pattern recognition algorithms can be implemented in hardware design.

Resource Usage and Cost

Based on the resource usage analysis, the design is optimized to fit within the constraints of the available hardware while maintaining functionality. The absence of block memory bits and DSP blocks indicates a reliance on combinational logic and registers, which aligns with the game’s behavior and the pattern recognition task. The following is a summary of the resource utilization:

- **Combinational ALUTs:** 2,543 (grid.sv uses 912, updateLogic.sv uses 1257)
- **Dedicated Logic Registers:** 856 (grid.sv uses 484, updateLogic.sv uses 256)
- **Block Memory Bits:** 0
- **DSP Blocks:** 0
- **Pins:** 103

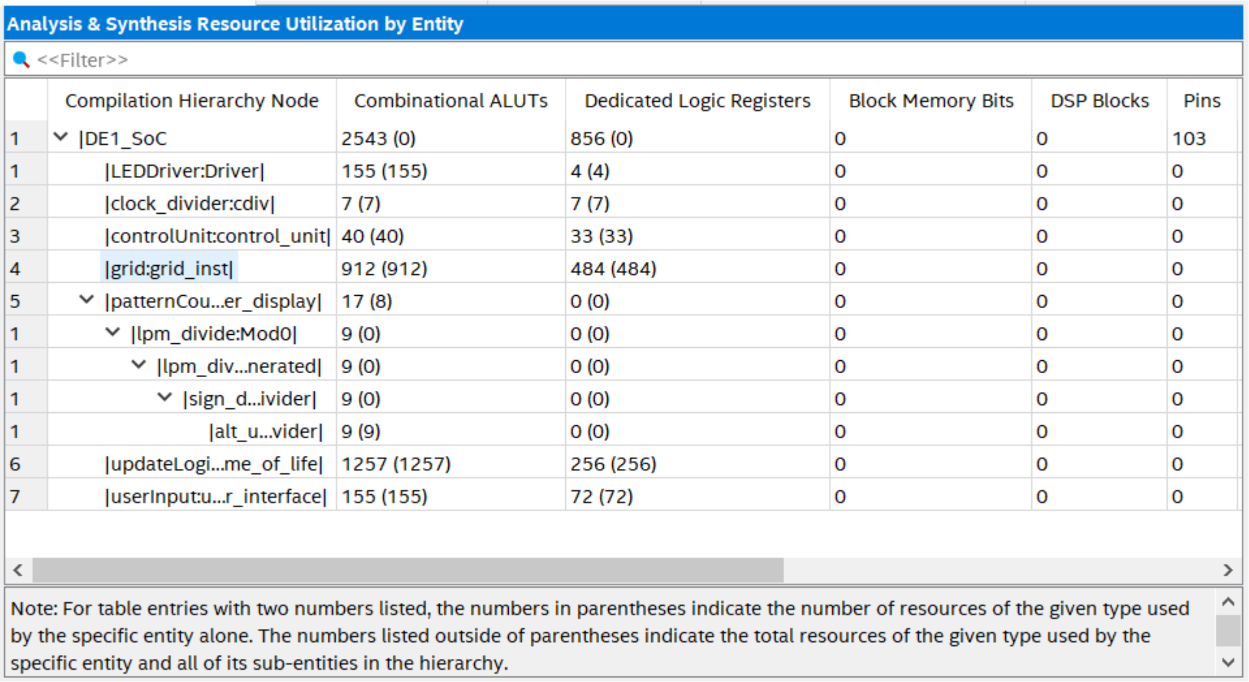
Safety and Social Considerations

From a safety perspective, the system poses no significant risks. It operates within the safe voltage and current levels typical of FPGA development boards and LED displays. The use of a stable hardware platform further prevents major potential safety concerns. From a social perspective, the system is designed with a global and culturally neutral approach, making it suitable for diverse educational environments. Its simplicity and ease of use ensure that it can be adopted in various cultural contexts without requiring localization.

4. Verilog Design Files

**Verilog files start from page 5 onwards. (these screenshots were then merged into the official submission pdf; the report itself does not include these)*

Resource Utilization by Entity full summary:



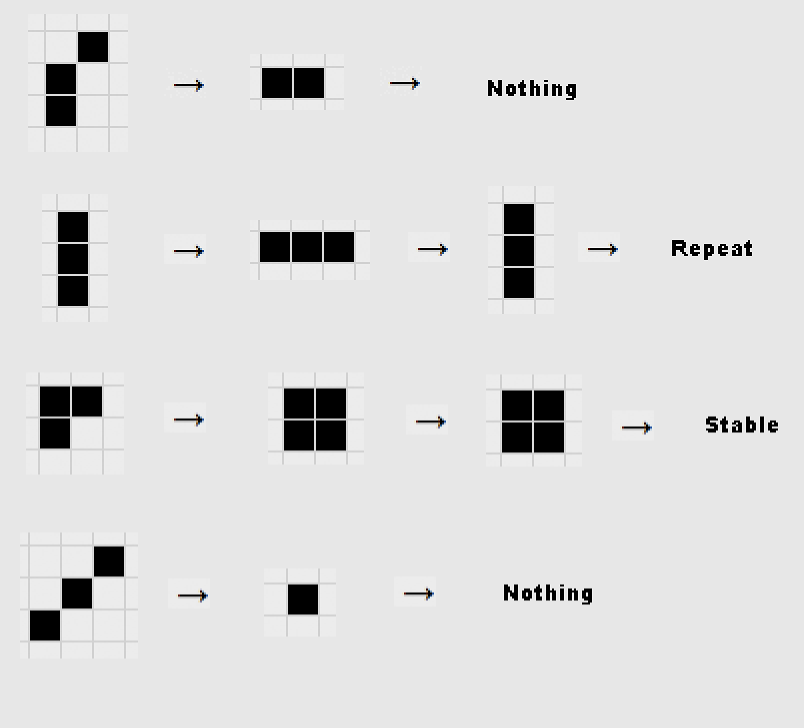
Analysis & Synthesis Resource Utilization by Entity

<<Filter>>

	Compilation Hierarchy Node	Combinational ALUTs	Dedicated Logic Registers	Block Memory Bits	DSP Blocks	Pins
1	|DE1_SoC	2543 (0)	856 (0)	0	0	103
1	|LEDDriver:Driver|	155 (155)	4 (4)	0	0	0
2	|clock_divider:cdiv|	7 (7)	7 (7)	0	0	0
3	|controlUnit:control_unit|	40 (40)	33 (33)	0	0	0
4	|grid:grid_inst|	912 (912)	484 (484)	0	0	0
5	|patternCou...er_display|	17 (8)	0 (0)	0	0	0
1	|lpm_divide:Mod0|	9 (0)	0 (0)	0	0	0
1	|lpm_div...nerated|	9 (0)	0 (0)	0	0	0
1	|sign_d...ivider|	9 (0)	0 (0)	0	0	0
1	|alt_u...vider|	9 (9)	0 (0)	0	0	0
6	|updateLogi...me_of_life|	1257 (1257)	256 (256)	0	0	0
7	|userInput:u...r_interface|	155 (155)	72 (72)	0	0	0

Note: For table entries with two numbers listed, the numbers in parentheses indicate the number of resources of the given type used by the specific entity alone. The numbers listed outside of parentheses indicate the total resources of the given type used by the specific entity and all of its sub-entities in the hierarchy.

5. Classic Conway’s GOL Test Cases (for demo)



6. Citations

- Online simulator + rulebook: https://playgameoflife.com/info
- General info about GOL: https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life
- Relevant Video #1: https://youtu.be/CgOcEZinQ2I?si=8zoHRuixiDQYu-po
- Relevant Video #2: https://youtu.be/R9Plq-D1gEk?si=xls30TQH1K1Ijm7-
- Relevant Video #3: https://youtu.be/ouipbDkwHWA?si=IOU2Z8GWL6Ms2bah
- Test cases image: https://pi.math.cornell.edu/~lipa/mec/lesson6.html