Lab 6

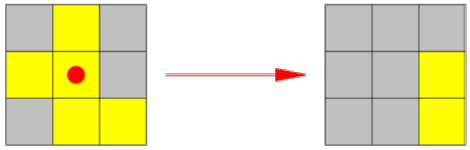
The LightsOut Puzzle with the Display on a Monitor Screen

Your task is to develop a system based on the FPro SoC that implements the LightsOut Puzzle with the display of the array of lights and other puzzle information on a monitor screen.

Introduction

The LightsOut Puzzle is a puzzle involving a rectangular (usually square) matrix of lights (squares) that can be turned on and off. A move consists of selecting a currently highlighted square, thereby toggling the state of this and all four vertically and horizontally adjacent squares.

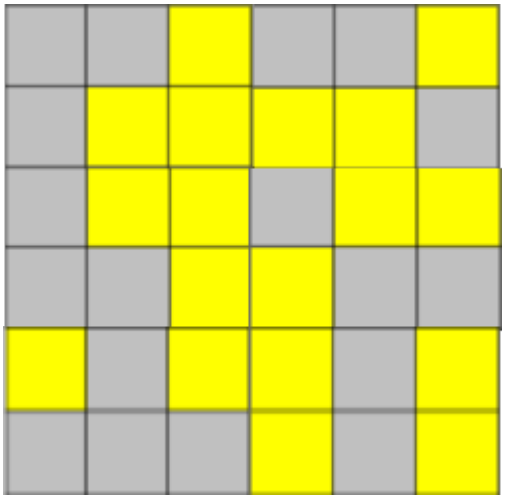
An example of a move for a 3x3 matrix is shown below:



Starting from a randomly chosen light pattern, the aim is to turn all the lamps off.

A demonstration of this puzzle for a 5x5 array of lights is available at
https://www.logicgamesonline.com/lightsout/

Your task is to implement a version of the LightsOut Puzzle with a **6x6** matrix of lights (squares), as shown below:

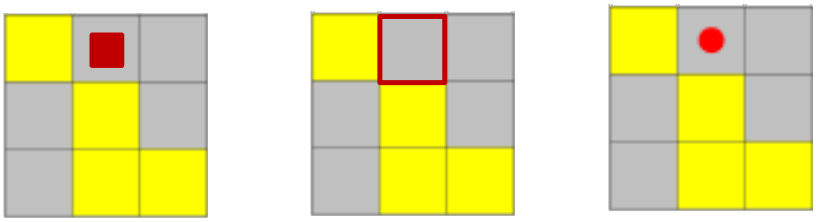


Display

A matrix of lights (squares) should be displayed on a VGA monitor.

The dimensions of each square should be 60 pixels by 60 pixels.
The matrix of squares should be displayed in the middle of the screen.
The widths of boundaries between squares should be 2 pixels.

Any of the following highlighters can be used to indicate the currently selected square (light).



Default for students working individually:
Implement the leftmost highlighter. Other versions will be rewarded with bonus points.

Default for students working in groups:
Implement the middle highlighter. The rightmost highlighter will be rewarded with bonus points.

Initial state and starting the puzzle

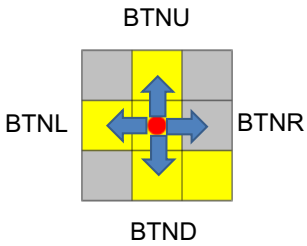
After asserting reset, all lights should be off, and the system should wait for BTNC to be pressed.

Four presses of BTNC should be used to initialize each 3x3 subarray, starting from the one in the top-left corner. The subsequent subarrays should be initialized in a clockwise fashion.

After the fourth press of BTNC, the highlighter should appear in the top-left corner of the 6x6 array.

Playing the game

Pressing buttons left, right, up, and down should naturally change the position of the highlighter, as shown in the figure below. All moves should be wrap-around moves.



During the game, pressing BTNC should toggle the state of the currently highlighted square and all four vertically and horizontally adjacent squares.

Difficulty Levels and Time Enforcement

The puzzle should have 7 difficulty levels. At each of the seven levels L (L=1..7), the user should have 8-L time units to complete the puzzle. The length of the time unit should be 30 seconds.

Thus, at level 1, the user should have 7 time units = 210 seconds to complete the puzzle. At level 7, this time should be reduced to 8-7=1 time unit = 30 seconds.

A current level, e.g., L5, and the remaining time, e.g., 84 seconds, should be displayed above the array of lights.

Allow doubling the time limit with SW15, using the convention:
SW15:
- 0: time unit = 30 seconds
- 1: time unit = 1 minute.

Winning and Losing Display Patterns (required for teams; bonus for students working individually)

After a user wins the game, all lights should be turned on and off five times for the duration of 1 second in each state.
After a user loses the game, all lights on both diagonals should be turned on and off five times for the duration of 1 second in each state.

Afterward, the system should return to the initial state (which it also enters after reset), i.e., all lights should be off, and the system should wait for BTNC to be pressed.

In order to allow demonstrating the winning and losing display patterns without finishing the game, SW0 can be used to force a win and SW1 to force a loss, using the following convention:
SW0:
- 0: normal operation
- 1: forced win

SW1:
- 0: normal operation
- 1: forced loss.

Deliverables:

1. New or revised VHDL Code [other than the original code of the FPro system made available to you], including testbenches (folders: src_rtl and src_tb)
2. New or revised C/C++ Code [drivers, test programs, applications] (folders: src_app and src_drv)
3. Video demonstrating the operation of your application. (folder: video)
4. A short report (folder: report) describing
   A. List of fully completed tasks, including the approach used (software, hardware, specific driver)
   B. I/O register map of all custom cores used
   C. List of tasks attempted but not completed (please describe shortly what is missing)
   D. List of any deviations from the original specification.
   E. Resource utilization after placing and routing (#Slices, #LUTs, #FFs, #BRAMs)
   F. Difficulties encountered and lessons learned.

Note: Make sure to create a separate folder for each deliverable mentioned above.
Do not submit any other files of the Vivado project!

Important Dates

	Friday Section	**Monday Section**
Related Lab Sessions	Friday 4/21, 4/28, 5/5 2023 8:40-11:20 AM	Monday 4/24, 5/1, 5/8 2023 9:00-11:40 AM
Deliverables Due	**Saturday** **5/13, 11:59 PM**	**Saturday** **5/13, 11:59 PM**
Demo and Q&A	**Friday, 5/12** **Monday, 5/15**	**Friday, 5/12** **Monday, 5/15**