Final Report

Ahmed Asif, Huzefa Soni, Jibran Sajid, Fowzi Ali

Description:

Our project is for the game “Frogger”. A player will control a frog attempting to cross through the screen which is filled with moving obstacles like cars and static obstacles like rocks etc. Features:

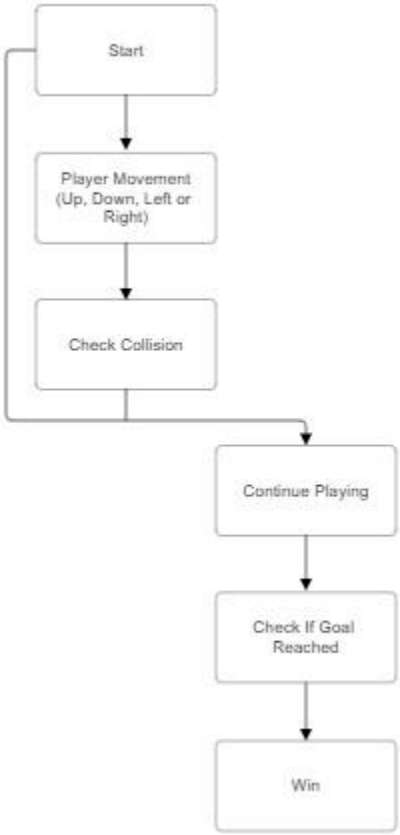
- FPGA input to control frog movement
- VGA Monitor: Shows the game area with the frog, cars and static obstacles.

Gameplay:

Initially, the frog will be at the bottom of the screen and the user’s task is to reach the top end of the screen safely. If reached, the user wins. The user can use vertical and horizontal motion to achieve this aim. If there is a collision or you reach the end safely, this will trigger a reset and you can start the game again.

To play, you have to use the 4 inputs on the FPGA (U17, M17, P17 and M18). There is also a reset button (U18) which will set the frog back to its initial starting position, on the bottom centered on the screen.

User Flow Diagram:



General Description of Modules:

top.v:

The top level module is where all the other modules are called and instantiated. It handles inputs for player controls, debounces the button presses, connects a VGA controller for video display, and manages the core game logic that is in the frogger module. RGB outputs are generated for the VGA display based on the current game state.

vga_controller.v:

This module generates VGA synchronization signals for a 640x480 resolution display. It calculates pixel coordinates (x, y) and signals whether the pixel is within the display area. A pixel clock derived from the 100MHz system clock drives these calculations.

frogger.v:

This module implements the core game logic, such as player movement, obstacle initialization and movement, collision detection, and win/loss conditions. It manages the layout, object colors, and determines the frog's position based on user inputs and collisions with obstacles.

btn_debounce.v: This module eliminates signal noise and stabilizes button input.

Explanation of Game Logic:

Checking for a win or a loss**:**

If the frog collided with an obstacle, which we could see by checking if the frog box was overlapping with an obstacle, it would be a loss.

If the frog reached the top of the screen (the maximum Y coordinate), it would be a win.

Collision Detection**:**

Whenever the frog collides with an obstacle (moving or static), it is sent to the initial starting position and the game restarts.

To compute both frog size and obstacle size, we had their x and y coordinates at the center of their box. We could get the top, bottom left and right bounds easily (for example top bound would be the y coordinate of the frog plus the frog size; similarly the right bound would be the x coordinate of the frog plus the frog size). The same process is true for the obstacles. This made it easier to detect collisions as we had to check if the bounds were overlapping with each.

Obstacle Movement**:**

Static obstacles were fixed in place whereas moving obstacles had different velocities and starting positions that were different and would bounce back (by changing direction) after reaching the horizontal edges of the playable screen.

Pixel Colour**:**

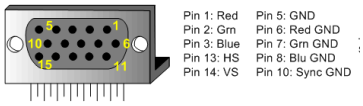
When the current pixel coordinates were the same as the frog's position or the obstacle's position, we would set the colour to the frog rgb and the obstacle rgb respectively.

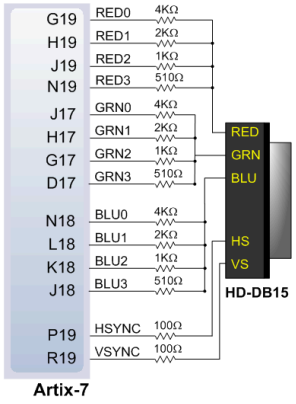
Input PIN Configuration:

Signal Name	Description	Pin
clk_100MHz	100MHz clock input	W5
reset	Reset button input (btnC)	U18
btn_Up	Button input for moving up	U17
btn_Down	Button input for moving down	M17
btn_Left	Button input for moving left	P17
btn_Right	Button input for moving right	M18

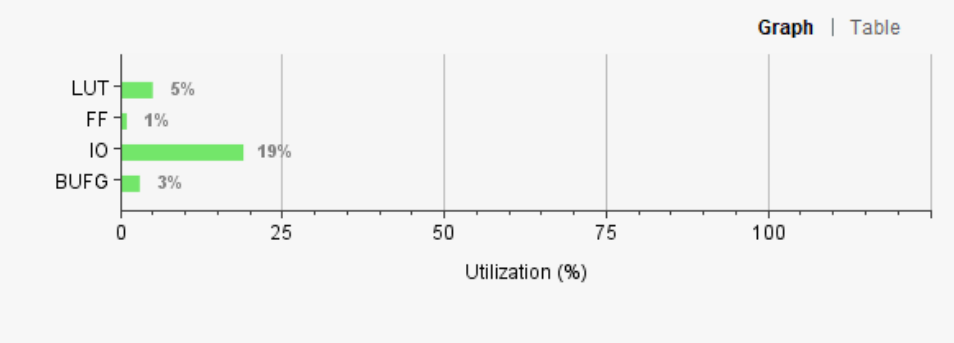
Output PIN Configuration:

Signal Name	Description	Pin
hsync	VGA horizontal sync output	P19
vsync	VGA vertical sync output	R19
rgb[11:0]	VGA RGB color output	R12, R11, T10, T9, V9, U12, U11, V12, V11, V10, U14, T14

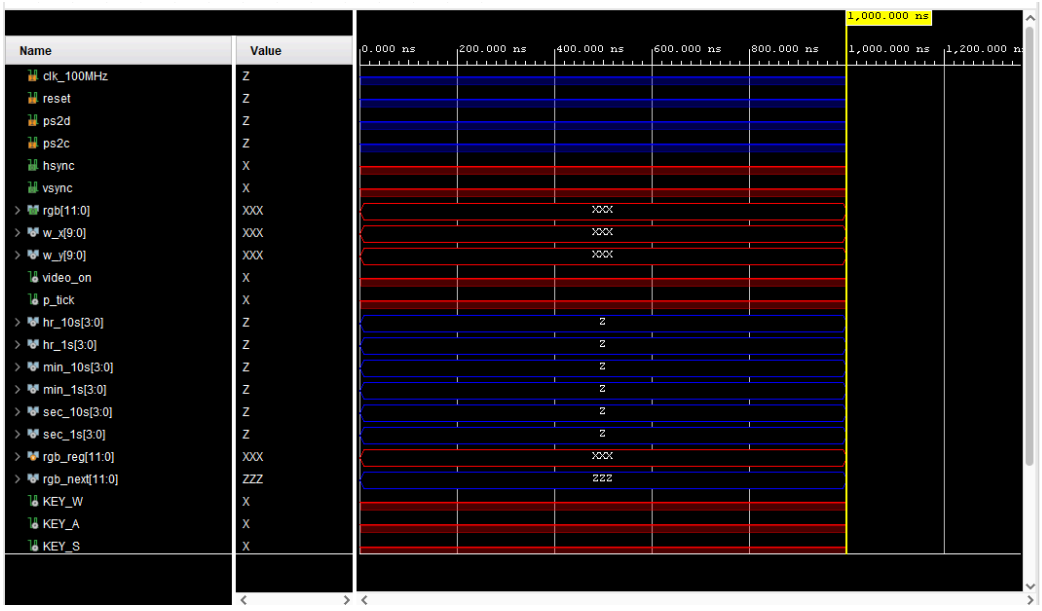




Hardware Utilization



Timing Diagram



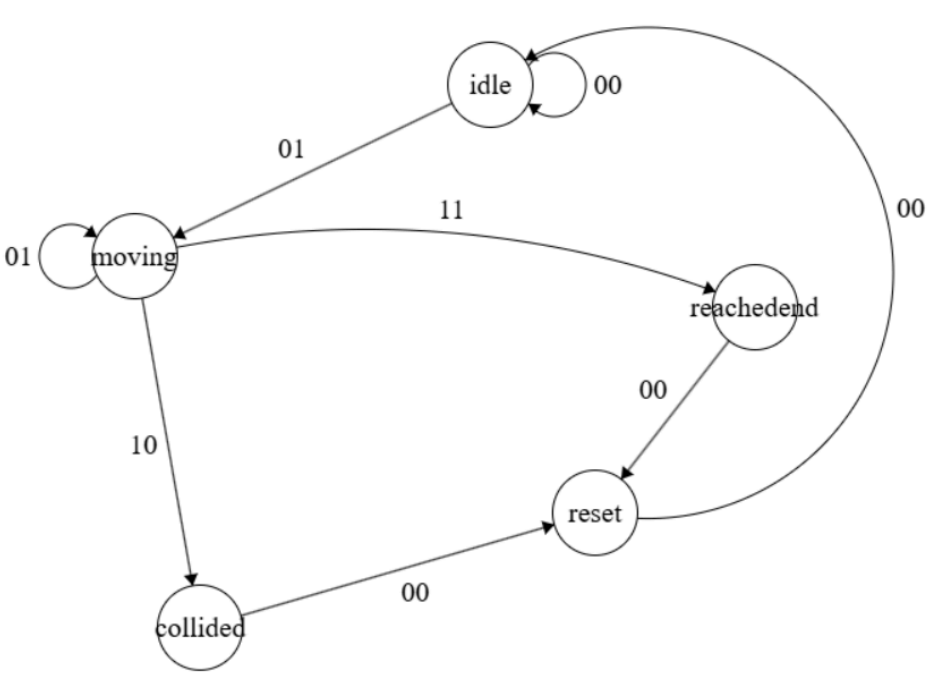
RTL Schematic

Finite State Machine:

Table

Current State	Input	Next State	Action/Output
000 (Idle)	01	001 (Moving)	Frog starts moving in the direction.
000(Idle)	00	000(Idle)	Stays idle
001 (Moving)	01	001 (Moving)	Frog continues moving (no change).
001 (Moving)	10	010 (Collided)	Handle collision.
001(Moving)	11	011(Reached end)	Trigger level completion logic
010 (Collided)	00	100 (Reset)	Reset frog's position and game state.
011 (ReachedEnd)	00	100 (Reset)	Prepare for the next level.
100 (Reset)	00	000 (Idle)	Return to idle state, ready for new input.

State Diagram



Game Picture