FPGA Car Game Part III: Rival Car Implementation and Collision Detection

Arnav Meena (2024CS50218) and Shivanshu Aryan (2024CS10237)

November 13, 2025

1 Introduction

This report details the design and implementation of Part III of the FPGA-based car game. The primary objectives were to introduce a rival car with autonomous vertical movement, generate its starting position randomly using a Linear Feedback Shift Register (LFSR), and augment the existing Finite State Machine (FSM) to detect collisions between the main car and the rival car.

2 Design Decisions and Implementation

2.1 8-bit Pseudo-Random Number Generator (LFSR)

To randomly determine the rival car's horizontal starting position, an 8-bit Linear Feedback Shift Register (LFSR) was implemented.

- **Implementation**: The LFSR consists of 8 D-flip-flops. The feedback function is a bitwise XOR of the outputs from the 4th, 5th, 6th, and 8th flip-flops, which is then fed into the first flip-flop.
- **Randomness**: The LFSR was seeded with the XOR of my partner's and my Kerberos IDs to ensure a unique pseudo-random sequence.

2.2 Rival Car ROM and Movement

The rival car's visual representation is stored in a dedicated ROM, and its movement is controlled by a frame counter.

- **ROM**: A 12-bit wide, 224-deep single-port ROM named `rival_car_rom` was created and initialized with the `Rival_car_testing.coe` file.
- **Horizontal Position**: The LFSR's output is used to set the rival car's initial horizontal coordinate (`rival_x_pos`) between pixels 44 and 104, ensuring it spawns within the road boundaries.
- **Vertical Movement**: The vertical coordinate (`rival_y_pos`) is updated downwards at a constant speed. A frame counter (`frame_reg_count`) increments at the end of each video frame. The `rival_y_pos` coordinate is incremented every 15 frames. This value was chosen as a design decision to provide smooth, visible movement without being excessively fast. Also, the rate of change of the y-coordinate was chosen to be 1 pixel per update. When the rival car reaches the bottom of the screen, it is respawned at the top with a new random horizontal position.

2.3 FSM Augmentation for Collision Detection

The main FSM from Part II was augmented to handle collisions between the main car and the rival car.

- **New Logic**: A new combinational block was added to continuously check if the hitboxes of the main car and rival car overlap.
- **Collision Logic**: Collision occurs when the rectangular bounding boxes of the two cars intersect. This is checked by comparing their respective x and y coordinates.

- **Game Over**: Once the hitboxes overlap, the game stops. The game can only be reset by pressing the reset button.

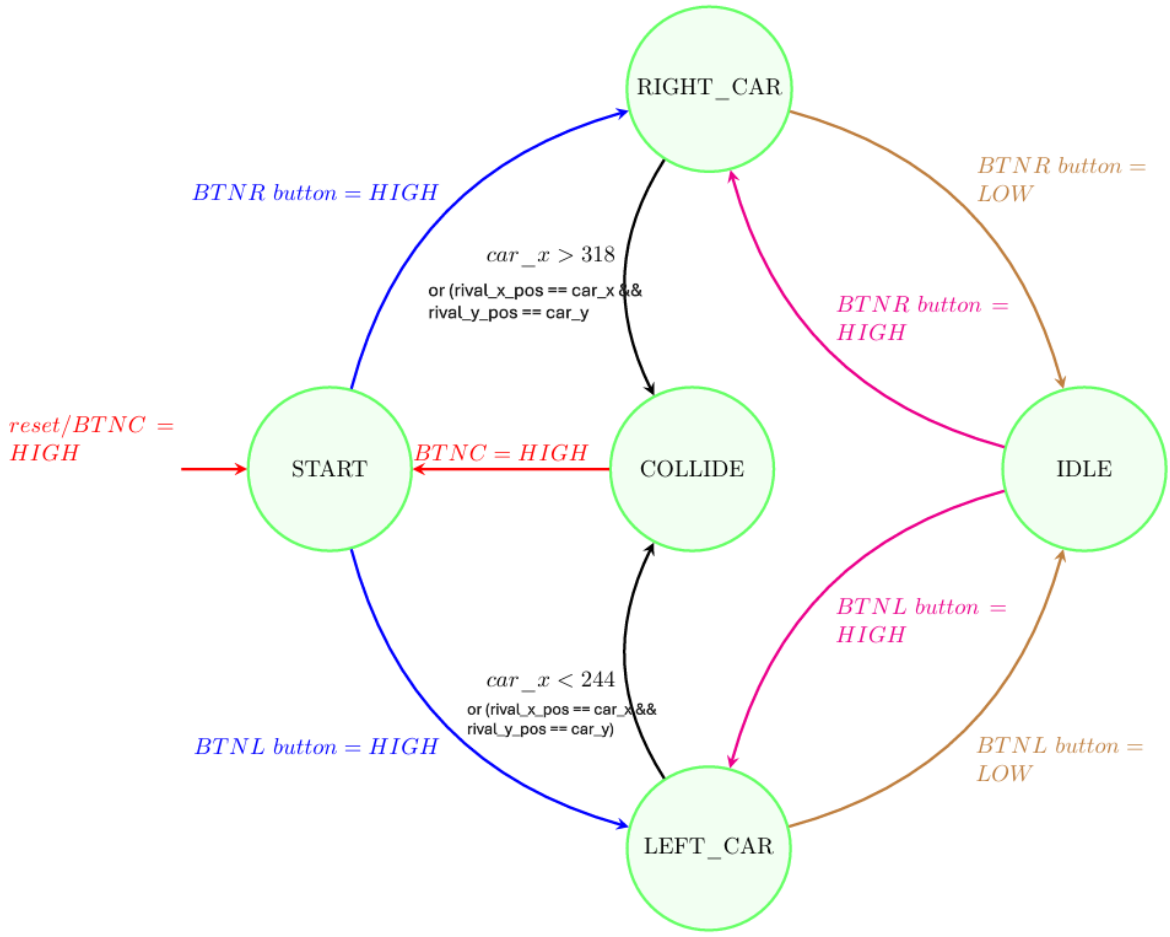

Figure 1: The augmented FSM which includes Rival Car Collision conditions

3 FPGA Synthesis and Implementation Report

The complete design was synthesized and implemented. The resource utilization is summarized below.

Resource Type	Used	Available	Util. (%)
Slice LUTs (Logic)	249	20,800	1.20
Slice Registers (FFs)	200	41,600	0.48
Block RAM Tiles	14.5	50	29.00
RAMB36E1/FIFO	12	50	24.00
RAMB18E1	5	100	5.00
DSP48E1 (DSPs)	2	90	2.22
Bonded IOBs	18	106	16.98
BUFGCTRL (Clock Buffers)	2	32	6.25

Table 1: FPGA Resource Utilization

4 Simulation and Verification

The design was extensively simulated to verify the functionality of the LFSR, rival car movement, and collision detection logic.

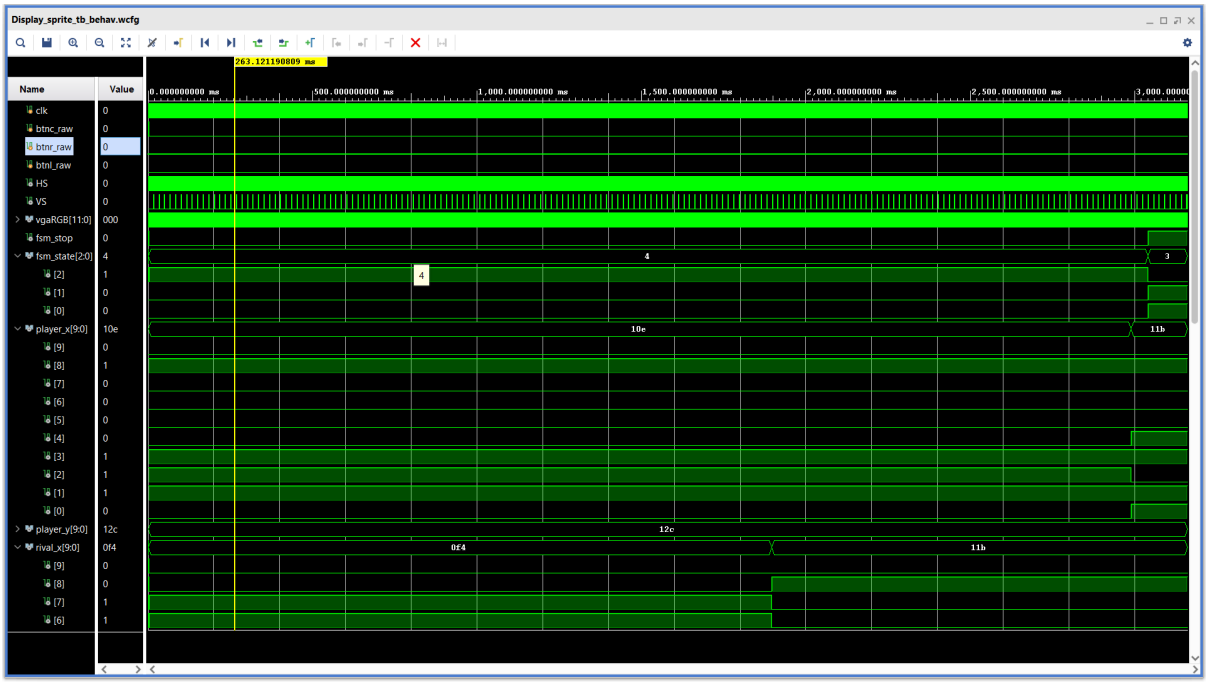

Figure 2: Simulation showing the fsm state transitioning to collision condition.

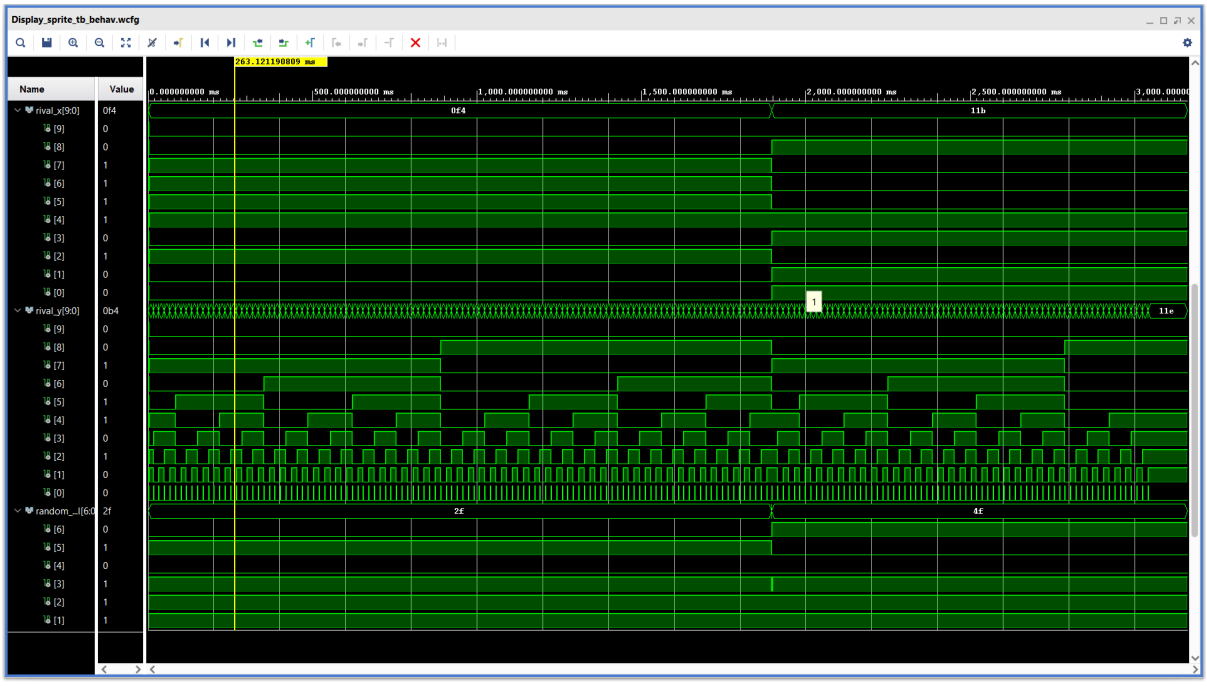

Figure 3: Waveform demonstrating the vertical movement of the rival car. The x-coordinate remains constant until the rival car hits the bottom.

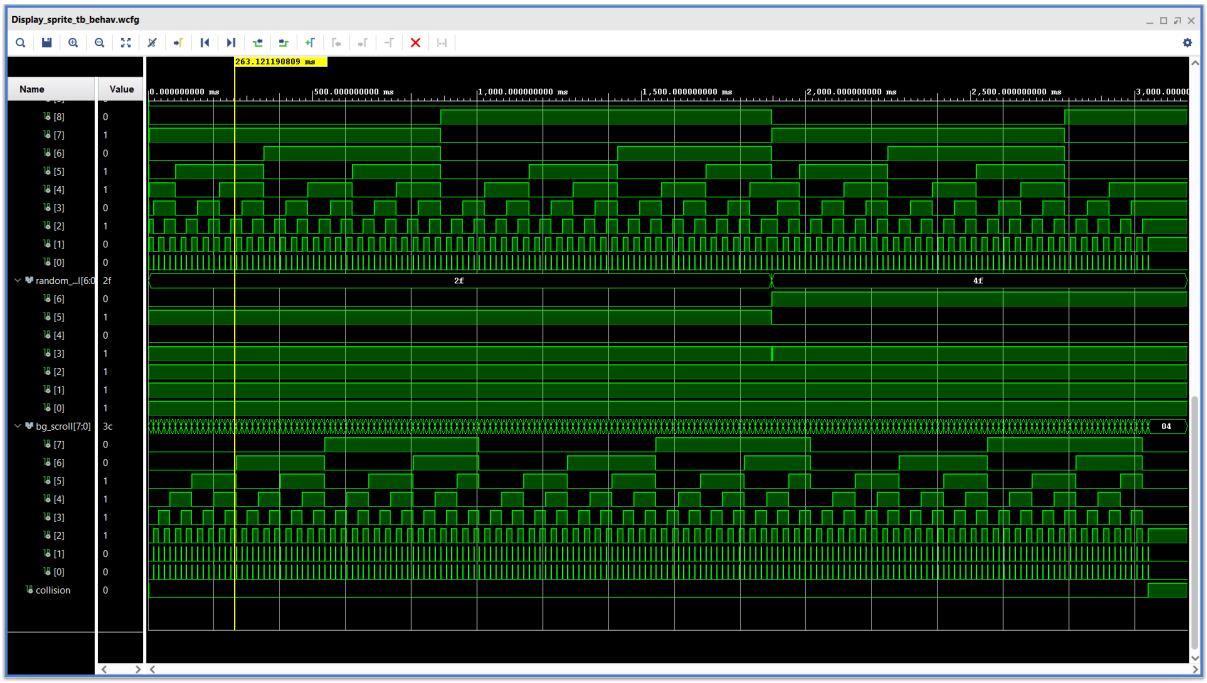

Figure 4: Collision detection signal, PRNG signal, and background stopping at collision are showcased.

5 Generated Schematics

The RTL and post-synthesis schematics generated by Vivado illustrate the hierarchical structure of the design, including the LFSR, ROMs, and control FSM.

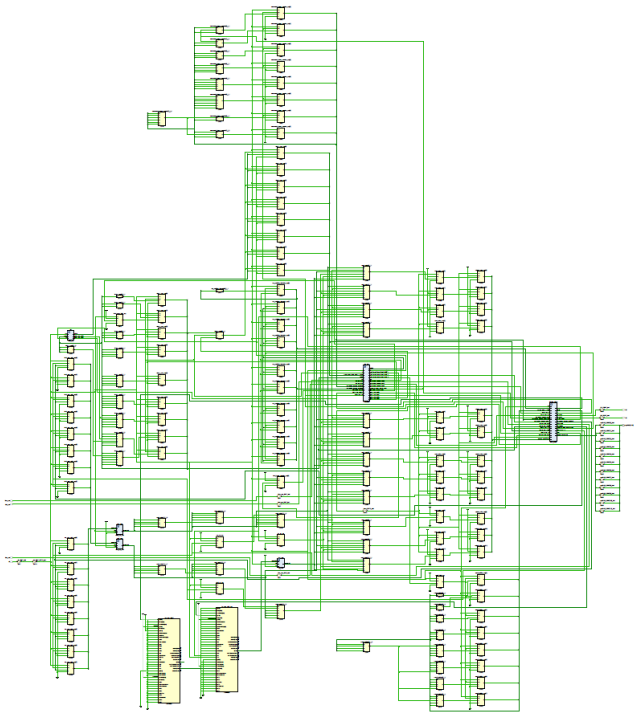

Figure 6: Synthesis Schematic

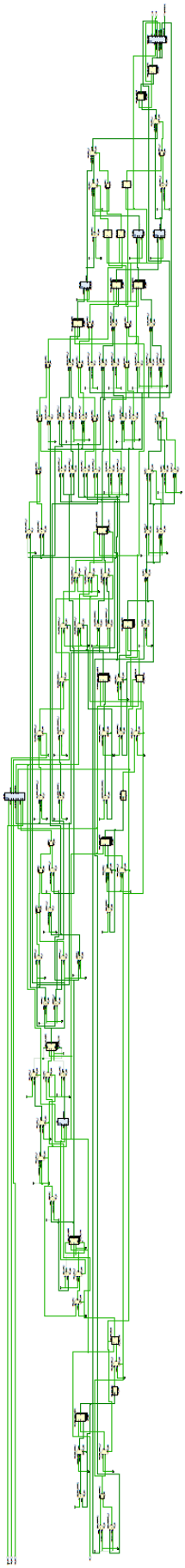

Figure 5: RTL Schematic

6 Conclusion

Part III of the project was successfully completed. An autonomous rival car was implemented with random horizontal positioning via an LFSR and constant vertical movement. The primary FSM was successfully augmented to detect and handle collisions between the player's car and the rival car. Simulation results confirmed correct logical behavior, and the design was efficiently implemented on the target FPGA.