EE314 Course Project's Final Report

Ahmet ÇIRA
METU EEE
ahmet.cira@metu.edu.tr
2515872

Alperen TUNÇKIRAN
METU EEE
alperen.tunckiran@metu.edu.tr
2517100

Hamza SOLMAZ
METU EEE
hamza.solmaz@metu.edu.tr
2516904

***Abstract*—The project demonstrates the FPGA development board implementation of a real-time two-player fighting game through Verilog HDL. The system duplicates FOOTSIES fighting game essentials by enabling directional movement and attacking functions and blocking capabilities and hitbox-hurtbox collision detection. The game runs at 60 Hz while showing its 640×480 VGA display. The game accepts player input through physical keypads which trigger animated sprite movements and visual feedback for each character. A deterministic finite state machine controls the game logic to handle state transitions between menu mode and gameplay and gameover mode. The system enables two gameplay modes: Player vs Player and Player vs Random-Bot while showing VGA and seven-segment display and LED-based game-over logic and menu interface. The design focuses on modular Verilog development while ensuring real-time processing and visual output synchronization with input control to demonstrate complete digital design principles. (*Abstract*)**

Keywords—Verilog HDL, FPGA, VGA Display, Digital Design, FSM, Fighting Game, Hitbox, Hurtbox, Real-Time Graphics, Game Logic, Collision Detection, Keypad Interface (*key words****)***

I. Introduction (*Heading 1*)

The laboratory project involved creating a modular 2D fighting game system through Verilog HDL programming on an FPGA platform. The main objective of this project involved designing and implementing digital game architecture that combined state machines with character rendering and health tracking and hit detection mechanisms under central control. The game logic received its responsiveness and sequence through the use of finite state machines (FSMs) that managed both game states and player actions.

The project divided into separate components which enabled both modularity and scalability. The system consists of character_fsm for state management of characters and game_fsm for game state control and hit_detector for collision detection and health_manager for health updates and renderer and counter for VGA interface output. The individual testing of each module followed by top-level integration verified the seamless operation of gameplay and input control synchronization with visual feedback.

The project allowed students to understand digital logic design principles including timing and synchronization and finite-state control while showing their practical application in hardware-based interactive system development. The development process focused on clarity and modularity and FSM-driven design principles which are fundamental for digital system development in both game and industrial control applications.

II. Theoretical Background

The project depends on four essential digital design principles which include finite state machines (FSMs), VGA signal generation, collision detection and hardware input handling. The FSMs control character states including idle and movement and attack and stun states which synchronize their state changes at 60 Hz to match the VGA refresh rate. The VGA controller functions at a 25 MHz pixel clock rate to produce horizontal and vertical synchronization signals that enable the display of 640×480 resolution graphics. The game uses hit detection through the interaction between hitboxes and hurtboxes which represent invisible regions that determine attack outcomes. The correct transitions between damage and block states depend on precise timing and detection during the active phase of the attack. The debouncing mechanism filters out signal noise from mechanical button bounce to ensure reliable physical button input. The game logic operates on an FPGA development board through Verilog HDL code which combines VGA output functionality with GPIO-connected keypad input reading and LED indicator and seven-segment display modules. These systems demonstrate how interactive applications with complex functionality can be built through digital hardware design at the lowest level..

III. Proposed Solution

The FPGA-based fighting game system architecture includes several Verilog components which operate under a top-level controller structure. The system contains a fundamental game_fsm module which controls four primary game states: menu, countdown, gameplay and game over. The system starts in menu mode when power is applied or the reset button is pressed before allowing players to choose between 1-player or 2-player game modes through a physical switch. The game moves to a countdown phase after mode selection and button confirmation shows "3", "2", "1", "START" on the VGA display. The game state enters gameplay after countdown where each player runs a separate character FSM that handles their movements as well as their attacks and hit reactions and stun states.

The game receives button inputs from players which trigger their FSMs to modify character positions along with state changes. The hit_detector module functions as an independent combat interaction system to detect collisions between hitboxes and hurtboxes by using character positions and attack states. The detected hit triggers a health or block count modification for the affected player. Two distinct display subsystems work together in the system to generate game outputs for seven-segment displays and VGA output for rendering characters with animations and backgrounds as well as user interface elements such as countdown timers and health bars and game result screens.

The gameplay system continuously monitors and displays both health levels and block point values of all active players. The game_fsm shifts into game over state when any player's health drops to zero which leads to VGA display of the winner combined with seven-segment display match duration and starting LED blinks. The system will return to menu state when any button is pressed following the game over state thus concluding the game loop. The system uses a modular synchronized architecture to deliver a responsive real-time gameplay experience through low-level digital logic operations.

Figure 1: Game FSM state chart.

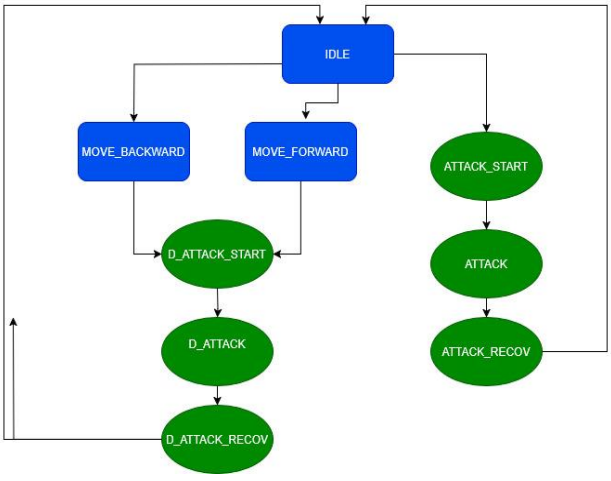

Figure 2: Character FSM state chart.

IV. Results

As a result, we completed most of the tasks which are asked, except hit box for neutral attack and a few details.

Figure 3: IDLE State.

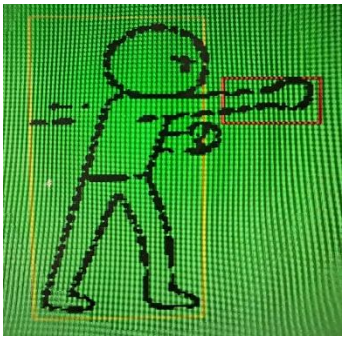

Figure 4:Directional Attack State.

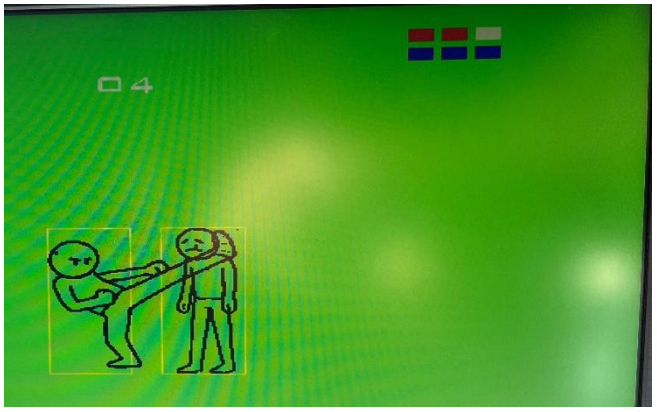

Figure 5: Neutral Attack State and Working Heart and Block Bar

Figure 6: Working Timer.

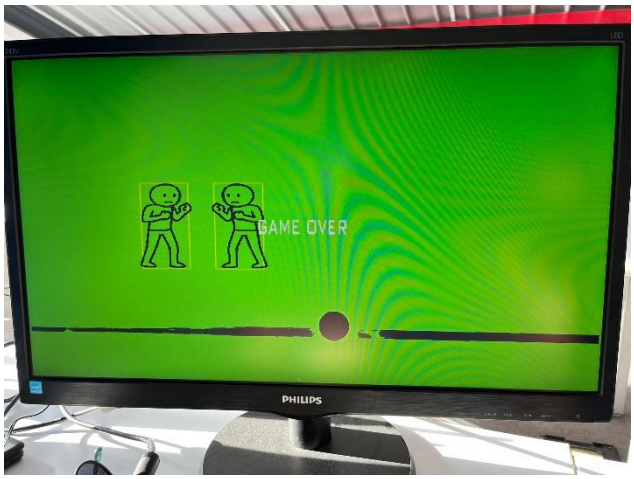

Figure 7: Game Over Shown When Time is Up.

Figure 8: Menu display.

V. Challenges and Discussion

The development process presented major obstacles which mainly affected hit detection systems and finite state machine coordination and rendering performance.

The main obstacle during development involved creating precise hit detection systems. The system required both character attack state management and precise modeling of their hitboxes and hurtboxes for accurate detection. The system required active attack detection alongside opponent hurtbox intersection checks and simultaneous attack handling capabilities. The system needed to handle simultaneous attacks by both characters when their hitboxes intersected by applying damage to both players at the same time. We created a hit_detector module to solve this problem and established precise definitions for attack active phases and startup and recovery phases in each character's FSM.

The main difficulty arose from managing two autonomous character FSMs to preserve game-wide consistency. The separate FSMs for each player needed complex interconnection logic to properly respond to each other's actions because each handled movement and attack behavior and hit reaction. The system needed to freeze character FSMs during stun states while blocking illegal state transitions that occurred during collisions. The development of this control flow needed extensive planning and multiple simulation tests for verification.

The last challenge we encountered involved performance issues when rendering character sprites. The synthesis and compilation process in the FPGA toolchain became excessively long because of the large sprite sizes and resolutions which made iterative development and debugging more time-consuming. The tool constraints together with hardware resource usage prevented us from completely resolving this limitation. We solved this issue by reducing the number of full recompilations and performing modular testing to detect bugs without needing complete builds.

VI. Conclusion

Our team successfully created a complete two-player fighting game for FPGA hardware through Verilog HDL programming with VGA output capabilities. Our design contained all required elements which included player-controlled finite state machines together with character rendering modules and collision detection mechanisms and health management systems and dynamic clock control. The development team tested each module separately to achieve perfect hardware integration and responsive gameplay.

The development process followed a modular approach which led to better system clarity and testability and maintainability. The FSMs enabled us to create detailed models of player interactions while the rendering and position control modules delivered smooth and accurate visual animations. The addition of hit detection and health tracking systems brought interactive elements to the game which improved the gaming experience.

We addressed the problems of timing mismatches and screen artifacts and input synchronization issues through a methodical approach of simulation testing and hardware verification and debugging. Our experiences taught us valuable lessons about real-time digital systems and VGA signal processing and hardware-software co-design principles.

The project met all course requirements while giving us practical experience in developing complex digital systems. Future work could include enhancements such as animated sprites, more refined AI behaviors for CPU-controlled players, and the implementation of additional gameplay mechanics. The project establishes a solid base for developing more complex FPGA-based games in the future.

References

[1] Adams, V. H. (n.d.). VGA driver in Verilog. Van Hunter Adams. https://vanhunteradams.com/DE1/VGA_Driver/Driver.html