

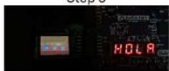

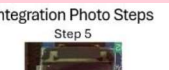





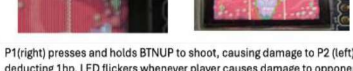




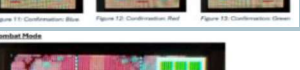
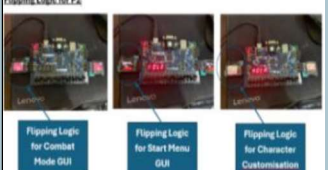





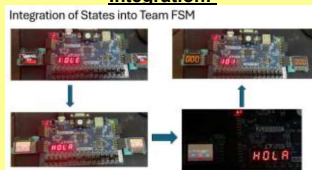
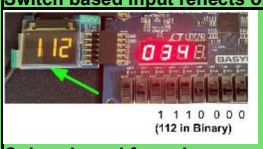



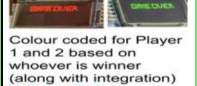


Student/ Improvement Name	Improvement Description	Images / Photos
<p>Team S1_21</p> <p>Veristrike (Overall User Guide)</p>	<p>Veristrike is an intense head-to-head first-person shooter (FPS) that blends fast-paced shooter action with a challenge system based on quizzes. The main objective of the game is to solve a Verilog conversion question in order to unlock combat capabilities in a "shoot-out" match. Before the game starts, players indicate their readiness to play through btnR (P1) and btnL (P2) to proceed to player customization.</p> <p>Next, they can choose between 3 different characters using the selection (P1: btnR, P2: btnL) and confirmation buttons (P1: btnU, P2: btnD), which would be indicated in red for the shoot-out (combat mode) happening later for a customised gaming experience. In quiz mode, a randomly generated decimal number is displayed at the beginning of the game to assess the players' understanding of converting decimals to binary.</p> <p>Players can enter combat mode and start a shooting match with the opponent if they provide accurate binary responses through SW0 – SW6 for P1 and SW7 – SW14 for P2. However, if the player finds it difficult to complete the quiz, they will be locked in quiz mode while the player that answers correctly is shown a correct prompt and proceeds to the combat mode first, losing important chances to survive and deal damage to the opponent through combat response.</p> <p>In the battlefield, the aim is to kill the opponent by depleting health points through successful shoot-out. Each player has the ability to shoot (P1: btnU, P2: btnD), switch character position (P1: btnR, P2: btnL) between 3 lanes and start with 3 Health Points (HP) each when they are in this mode. A successful shoot-out is determined when the player shoots in the same lane as the opponent. The hit player will face a 1-point deduction in his HP.</p> <p>For added difficulty, the players are unable to view each other's screens and lane change is delayed by 3 seconds to add game difficulty. Damage dealt is indicated by the LED lights (LD0 and LD15 for P1 and P2 respectively). Finally, the game is won by the player who first depletes the opponent's H/P bar. This is indicated by a Game Over screen that is shown in green for the victor and red for the loser. In conclusion, the game requires a strong mastery of decimal-to-binary conversion (theoretical knowledge) and fast response time for victory.</p>	<p>Integration Photo Steps</p> <p>Step 1  Step 2 </p> <p>Step 3  Step 4 </p> <p>Integration Photo Steps</p> <p>Step 5  Step 6 </p> <p>Step 7 </p>
<p>Student A:</p> <p></p> <p>Shooting Rainbow Stream</p> <p>Damage Handling</p> <p>HP Bar & Game Over</p> <p>Overall Integration of all game parts and development of FSM Algorithm</p>	<p>Shooting Rainbow Stream</p> <p>The Shooting Rainbow Stream offers continuous fire, activated by pressing and holding the shoot button, enabling sustained offense and dynamic gameplay. Bullets are rendered with a captivating rainbow effect, alternating RGB colors to create a vivid visual display. Synchronized precisely with the player's lane position, the stream ensures accurate alignment with movement.</p> <p>Pulse-width modulation (PWM) orchestrates seamless color transitions, heightening immersion and responsiveness in gameplay.</p> <p>Damage Handling</p> <p>The damage model employs a time-weighted mechanism, deducting HP only after prolonged exposure to bullets within the same lane, creating a realistic damage system. This cumulative approach reduces HP after 3 seconds of continuous exposure. A fllickering LED during damage events offers immediate visual feedback, helping players adapt their strategies.</p> <p>Controls for shooting & damage indicator: BTN_UP/LD0 for P1 (Right) and BTN_DOWN/LD15 for P2 (Left).</p> <p>Created HP Bar and Game Over Screen</p> <p>The HP Bar conveys health status through segmented color blocks, instantly updating with each hit to provide real-time feedback. The Game Over screen is precisely coded to display "GAME OVER" in green for the victor and red for the defeated.</p> <p>Overall Integration of all game parts and development of FSM Algorithm</p> <p>Student A assisted Student D in integration and development of FSM algorithm, particularly for the Start Menu and Character Customization states. Refer to Student D section for more information.</p>	<p>Manual Drawing (Not Image) of Health Points Bar  Shooting Rainbow Stream (Not apparent due to lighting) </p> <p>P1(right) presses and holds BTUP to shoot, causing damage to P2 (left), deducting 1hp. LED flickers whenever player causes damage to opponent </p> <p>Manual Drawing (Not Image) of GAME OVER (Green for winner and red for loser) </p>
<p>Student B:</p> <p></p> <p>Pre-Game GUI</p> <p>Game Mode GUI</p> <p>Press-Detection</p>	<p>Start Menu GUI (Graphical User Interface) and Player-Ready Logic</p> <p>The game start menu appears on the screen when the game is initialized. The game's gloomy mood (shooting element) is hinted at by the reddish-brown noir background. White pixels are used to draw the term "Veristrike" for branding. To prompt players to press their respective ready (btnR and btnL for P1 and P2 respectively) button, the drawn "play" word will play an animation (expansion and contraction). When the button is pressed for respective players, their respective "play" word on the display will turn yellow as visual indication.</p> <p>Character Customisation GUI</p> <p>When btnR and btnL are pressed, the game switches from the Start Menu to Character Customization, displaying a sunset backdrop with selectable cowboy sprites (blue, red, yellow) and a "Pick your Character" prompt. Players use btnR and btnL to navigate and select their characters, with a yellow cursor appearing below to indicate the current choice. To confirm, P1 and P2 press btnL and btnD, respectively, turning the cursor red. The game advances to quiz mode only after both players confirm their selections which is passed to combat GUI.</p> <p>Combat Mode GUI</p> <p>The combat mode interface includes an interactive pixel-drawn HP bar on the top right for tracking progress and gradient borders to highlight the three player lanes. The border gradient (light to dark) intuitively indicates shooting direction. A red desert background maintains the Midwest theme and reflects time progression. The selected character sprite appears in the center of one of the lanes, with navigation buttons (btnR and btnL for players 1 and 2) enabling lane movement.</p> <p>Combat Mode Player Lane Transition Logic</p> <p>The combat player movement shows an animation of the player transitioning to the next lane on the right for 2s in combat mode.</p> <p>Specialised 2s Press Detection Logic for Lane Transition</p> <p>To maintain a 3-second buffer between lane transitions, button presses are ignored for 3 seconds after a lane change. This requires a counter that activates with each press, reaching and holding at 3 seconds, only resetting with the next valid press after the delay. This timer-based press-detection system will allow button presses only if the counter has reached 3 seconds, preventing animation glitches and excessive lane changes to maintain game dynamics.</p> <p>Improvement of General Press Detection Modules</p> <p>Enabled unlatched and latched presses to be recognised. Additionally, press count features for 2-bit and 3-bit values were added. The press count feature has been utilised for various features such as lane change and sprite selection. It can be reset through sending a HIGH (1) signal to the new state input momentarily.</p>	<p>Start Menu </p> <p>Character Customisation </p> <p>Combat Mode </p> <p>Figure 10: Character Customisation Right</p> <p>Figure 11: Character Customisation Right</p> <p>Figure 12: Character Customisation Right</p> <p>Figure 13: Character Customisation Right</p> <p>Figure 14: Character Customisation Right</p> <p>Figure 15: Character Customisation Right</p> <p>Figure 16: Character Customisation Right</p> <p>Figure 17: Lane Transition M=1</p> <p>Figure 18: Lane Transition M=2</p> <p>Figure 19: Lane Transition M=3</p>





	<p>Flipping Display Logic for Combat Mode, Player Customisation and Start Menu GUI As P2's OLED screen is connected upside down due to pin layout limitations, a flipping logic has been implemented for intuitive display comprehension.</p> <p>Pre-Game Modules Integration Performed initial integration for Start Menu and Character Customisation GUI using modular coding conventions for simplified integration with other states (combat, question, correct state and game over) using a simplified early version of Finite State Machine</p>	
<p>Student C:</p> <p>7-Segment Display GUI & Logic</p> <p>Pre-Game Character Selection Logic</p> <p>Correct GUI Display and Flipped Logic</p> <p>State Machine Integration</p>	<p>7-Segment Display GUI & Logic At the initial state where the game has started, the word "IDLE" is being displayed to indicate that the game has not started. Once Player 1 and Player 2 presses the right button and the left button respectively, An "HOLA" sign on the 7-segment Display is being displayed. "HOLA" represents for Hello in Spanish. This allows the player to get to have a different type of interaction in terms of understanding a language while playing the game. This allows both players to be able to stay interacted within the game and serves as a checking mechanism if both players are ready to standby for the next stage of the game. To activate this task, Press Left and Right – HOLA Display Last: "OVER" display automatically Character Selection Logic This is also known as the "choosing character" and "confirming character" logic state. The module allows Player 1 and Player 2 to independently navigate and confirm character selection using designated buttons. A yellow cursor shows navigable selections, turning red once a character is confirmed. The selected character is locked and tracked for use in a later battle mode. Each player's button presses are handled independently, ensuring smooth, interference-free character selection. To activate this task, Press right/left button to navigate for Player 1 and Player 2 respectively. Press Up/Down button to confirm selection for Player 1 and Player 2 respectively. Once completed, it should transition to the quiz mode. Correct GUI Display (Mode 3) (Battle Mode) Displays an accurate measure for correct status by representing a tick. This is represented to show that the player has answered correctly. This has been applied for both Player 1 and Player 2. This logo will appear quickly for a while to show that player have managed to match the desired binary number during the quiz before transitioning to the shooting lane. To activate this task, one of the players must get the binary number correct before the tick logo is displayed. State Machine Integration in Pre-Game with Team FSM Integration on the Pre-Game Logic with the rest of the game. This includes being able to show the states of the pre-game status, being able to adapt to the full game for transition purposes. Integration from Start Menu > Character Customization Menu > Quiz Mode.</p>	<p>7-Segment Display Logic: Start Mode (Mode 0):  Press Left and Right (Mode 1): HOLA Display  Mode 6: Game Over State  Character Selection Logic: Image for Players 1 and 2 (Character Selection Logic) Player 1 (Right Screen) Press Right Button: Navigates Character (Yellow Cursor) Press Up Button: Confirms Character (Red Cursor) Player 2 (Left Screen) Press Left Button: Navigates Character (Yellow Cursor) Press Down Button: Confirms Character (Red Cursor)  Correct GUI:  Integration: Integration of States into Team FSM </p>
<p>Student D:</p>	<p>Overall Integration of all game parts and Development of FSM Algorithm Student D designed an Adaptive FSM to manage game stages—Start Menu, Character Customization, Quiz Mode, Tick Display, Combat, and Round Over—using a Dynamic Event Handling Algorithm. Key state variables (<i>state_tracker_p1</i>, <i>state_tracker_p2</i>) track player actions independently, allowing concurrent transitions within shared modes. The Pulse-based Reset Algorithm manages temporary timers (<i>tick_p1_timer</i>, <i>tick_p2_timer</i>) to control timed displays, with round_over_timer regulating Round Over resets. This design enables responsive and precise gameplay transitions, using priority-based state handling and cycle-counted resets OLED Number Display with Dynamic Color Feedback Algorithm: Student D implemented a 7-segment display on the OLED, leveraging a coordinate mapping technique to simulate a 3-digit 7-segment display. The display is divided into hundreds, tens, and ones places by mapping specific OLED coordinates to each segment of a digit. By dividing each number input into these three components, the module displays each digit in its correct position on the OLED. Additionally, a Dynamic Color Feedback Algorithm adjusts the display colors based on the difference between the user's input and the target value, providing immediate feedback. The algorithm calculates the difference between the player's input value and the target number, with the formula: $red_level = (31 * normalized_diff) / 255$; $green_level = (63 * (255 - normalized_diff)) / 255$; Quiz Algorithm Development and Integration: Student D developed the entire Quiz Algorithm, which takes player inputs (sw[0:6] for Player 1 and sw[7:13] for Player 2) to display their values on the OLED. The algorithm breaks down each input into hundreds, tens, and ones, assigning each segment predefined OLED coordinates to replicate a 7-segment display. A dynamic color formula aids players by changing the displayed color based on input accuracy—green for closer guesses, red for greater difference—providing immediate feedback until users match the random number. Once correct, the system proceeds to the Tick Mode, which Student D also integrated to signal quiz completion. Also, colour coded "Game Over" sign based on whoever won (green/red) 7 Segment Display + True RNG Algorithm (Switch Timing Based) Student D implemented a 7-segment display algorithm that maps each input number (hundreds, tens, ones) to specific OLED coordinates, ensuring clarity and accurate numeric representation. The True RNG algorithm leverages the 100MHz clock's timing with sw[15] state changes, creating a genuinely random number based on the precise moment of switch interaction. This approach captures minor timing variations, ensuring randomness while adding unpredictability to the quiz's target number. Created Standardized Image Display Protocol: Student D created the protocol which involves background removal, scaling, and format conversion. It starts with preparing images on GIMP, removing backgrounds, and resizing them to fit the OLED. The images are then converted to hexadecimal using online tools, ensuring they display in Vivado without distortion.</p>	<p>Switch based input reflects on OLED display  1 1 1 0 0 0 0 (112 in Binary) Colour-based formula Red Level: $31 \times normalized_diff / 255$ Green Level: $63 \times (255 - normalized_diff) / 255$ Truly random number generator  Interaction between CLOCK and SW15 generate truly random number State tracker for the whole game state_tracker_p1/p2: • 0 - Start Menu • 1 - Character Customization Mode • 2 - Quiz Mode • 3 - Tick Mode (1 second per player, sequentially) • 4 - Combat Mode • 5 - Round Over (2 seconds, both players sit/cancelled) • 6 - Game Over (game over state) Integration of tick mode  Tick Mode Integration (with timer) Colour gradient feedback algorithm  Game colour module coloured  Colour coded for Player 1 and 2 based on whoever is winner (along with integration) Winner Loser</p>

Student A: [REDACTED]

Student B: [REDACTED]

Student C: [REDACTED]

Student D: [REDACTED]

Student A: 
Student B: 
Student C: 
Student D: 

Lab Session and Group No: Session 1, S1_21

References:

1. <https://jav1.github.io/image2cpp/>