EC311

Final Project Report

Prof. Joshi

11/9/2025

Group 19

Whack-a-Mole On FPGA

An LED game based on Nexys A7

Hongming Du
Cankun Wang
Joe Nguyen
Parth Kheni

1. Introduction

FPGA, Field-Programmable Gate Array, is a powerful logic programmable device. For this project, we created a Whack-a-Mole reaction game based on the FPGA functions and parts. The goal is to use Verilog code to build a complete digital system that interacts with real hardware, rather than only running simulations. In the game, a set of LEDs represents different mole positions, and push-buttons act as the player's hammer. The system manages a short pre-game countdown, a timed game session, scoring based on the player's input, and a simple difficulty selection.

By creating this game, we consolidated our understanding of the course materials, including the design of finite state machines, module pipelines, basic hardware design, and team development. Beyond creating the project, we also gain experience in constructing a multi-module system design and behaviour-based hardware code design.

2. Project Overview

This project implements a Whack-a-Mole-style reaction game on the Nexys FPGA board using Verilog.

The basic game function design uses five on-board LEDs to represent "moles", a push-button as the player's hammer, and a switch button for position selection. At any given time, exactly one LED is active. The player should switch the switch that indicates the current mole position, and then press the matching button to "hit" the mole. Also, different difficulties are available.

The game is organised into three phases. After reset, a short pre-game countdown is displayed on the seven-segment display to give the player time to prepare. When the countdown reaches zero, the main game phase begins and runs for a fixed duration. During this period, the mole's position changes periodically according to a pseudo-random sequence and the player's hits. When the game time expires, the system enters a game-over state, and the final score remains visible on the seven-segment display until the player presses the reset button to start a new round.

From a digital design perspective, the project integrates the main topics of EC311 into a single finite system. The design includes debounced and edge-detected button inputs, clock division for human-scale timing and display scanning, dedicated counters for countdown and game time, a score counter, a pseudo-random mole controller, and anything we learned to build a central finite state machine that coordinates all subsystems and controls display output.

3. Design Specifications

Translating the functional requirements into concrete timing and interface specifications that guide the hardware implementation requires systematic module design. The aim is to define the exact functions of each module, including durations, count bit ranges, and signal conventions that each module must satisfy in order for the overall system to behave predictably on the FPGA board.

3.1 Timing and Behavioural Specifications

All logic in the design is clocked from the on-board 100 MHz system clock. While human-scale behaviour, such as the one-second countdown and the visible duration of each mole, is derived using clock dividers and synchronous counters. The key timing parameters are:

1. System clock
    a. Frequency: 100 MHz
    b. Source: on-board oscillator on the Nexys board
    c. All sequential modules are synchronous to this clock.
2. Countdown duration
    a. The pre-game countdown lasts 5 seconds.
    b. A one-second enable pulse, derived from the 100 MHz clock, is used to increment the countdown counter.
    c. The countdown counter starts from zero and counts up to 5. The value shown on the seven-segment display is 5 − countdown_sec, so the player sees 5, 4, 3, 2, 1, 0 at one-second intervals.
3. Game duration
    a. The main game phase lasts 30 seconds.
    b. A separate second counter, also driven by the one-second enable pulse, counts from 30 down to 0 while the game is in the playing state.
    c. When the game time reaches 30 seconds, the game terminates, and the system enters the game-over state.
4. Mole visibility window
    a. At any time, exactly one LED is active, representing the current mole position.
    b. The time that a mole remains visible depends on the selected difficulty level. Internally, a tick counter driven by the 100 MHz clock counts a fixed number of clock cycles for each mole:
        i. Easy: longest visible window (2 seconds).
        ii. Hard: intermediate visible window (1 second).

5. Hit registration and scoring.
    a. Button presses are first debounced and converted into one-cycle pulses.
    b. A hit is registered only if a debounced hammer pulse occurs while the switch of the mole is active and the corresponding position matches the lit LED.
    c. Each successful hit increments the score by one. The score is stored in an 8-bit counter and is constrained to the range 0–99, which fits on two seven-segment digits.
6. Mole position sequence
    a. After each mole window (either hit or timeout), the mole controller selects the next active LED position.
    b. The sequence of positions is generated pseudo-randomly from the five available locations in order to avoid repetitive and easily predictable patterns.

These timing and behavioural specifications ensure that the game feels responsive and consistent while remaining fully synchronous and compatible with FPGA timing constraints.

3.2 Hardware Interface Specification

The design maps the abstract game behaviour onto specific physical resources of the Nexys FPGA board. Table 1 summarises the primary input and output signals at the top level.

1. Clock and reset
    a. Clk (input, 1 bit): 100 MHz system clock used by all sequential logic.
    b. Reset (input, 1 bit): active-high synchronous reset. When asserted, the system returns to the idle state, clears all counters, and turns off all LEDs.
2. Control buttons
    a. button_start (input, 1 bit): start/reset game button. A debounced pulse from this button triggers the 5-second countdown and, from other states, restarts the game sequence.
    b. button_clear (input, 1 bit): clear-score button. When pressed, it clears the score counter and resets the game timer without requiring a full power cycle.
    c. button_difficulty (input, 1 bit): difficulty selection button. Each pulse cycles through the available difficulty levels and updates the internal difficulty register when the system is in the idle or game-over state.

d. button_hammer (input, 1 bit): hammer button used to register hits. When pressed, the design samples the switch vector to determine which mole position the player is targeting.

3. Switches and LEDs

   a. Switch [4:0] (input, 5 bits): position selector switches, one under each LED. When the hammer button is pressed, this vector encodes which mole position the player is attempting to hit.

   b. LED[4:0] (output, 5 bits): mole indicator LEDs. Exactly one bit is high at any time during the game to indicate the active mole position; all bits are low when the game is idle or in reset.

4. Seven-segment display

   a. Seg [6:0] (output, 7 bits): segment control lines for the seven-segment display (active-low on the Nexys board). The pattern on these lines encodes the current digit being displayed.

   b. digit_select[3:0] (output, 4 bits): digit enable lines for the four seven-segment positions. A scan counter periodically activates one digit at a time while driving seg[6:0] with the corresponding digit code, creating the illusion of a static multi-digit display.

The combination of these timing and interface specifications defines a precise contract between the game logic and the physical hardware, and forms the basis for the module-level design described in the next section.

4. High-Level Architecture

The Whack-a-Mole game is implemented as a synchronous, modular design built around a single 100 MHz system clock. At the top level, the system is decomposed into several cooperating blocks: button interface logic, clock division and timing, a central game control finite state machine, a mole control and pseudo-random position generator, a score counter, and a multiplexed seven-segment display driver.

4.1 Top-Level Structure

At the highest level, the design can be viewed as five major subsystems:

- Button and difficulty interface (button_io, debounce_one_pulse)

This block receives raw push-button signals from the Nexys board (start, clear, difficulty, and hammer). It debounces the mechanical inputs and converts them into one-cycle pulses, ensuring that the rest of the design only sees clean, synchronous events. In addition, it maintains a small difficulty register that cycles through the available difficulty levels when the difficulty button is pressed.

- Clock division and timing (clock_divider, sec_counter)

  The on-board 100 MHz clock is divided to generate a 1 Hz enable pulse, used for human-scale timing, and a higher-frequency scan clock, used for seven-segment multiplexing. Two instances of a generic second counter module use the 1 Hz enable to implement the 5-second pre-game countdown and the 30-second game timer.

- Game control finite state machine (game_control_fsm)

  This is the central controller of the system. Based on the debounced start and clear pulses, the current countdown value, the game timer, and the score, it progresses through four states: IDLE, COUNTDOWN, PLAYING, and GAME_OVER. In each state, it asserts or de-asserts control signals to the countdown counter, game timer, score counter, and mole controller, and selects which numeric value should be shown on the seven-segment display.

- Mole and scoring logic (mole_led_and_random, score_counter)

  The mole controller uses the system clock and the selected difficulty level to determine how long each mole remains visible, and selects the next mole position using a pseudo-random or pseudo-random-looking sequence over the five LED positions. When the hammer button pulse occurs, the design samples the switch vector, compares it to the currently active mole position, and generates a hit pulse if they match. The score counter increments on each hit pulse while it is enabled by the game control FSM and can be cleared by the clear-score button or when a new game starts.

- Display subsystem (scan_counter, seven-segment decoder, 7-segment wrapper)

  The display subsystem receives an 8-bit display_value from the game control FSM and converts it into the appropriate seven-segment codes. A scan counter, driven by the scan clock, periodically activates each digit in turn while driving the shared segment lines with the correct pattern. This time-multiplexed approach allows multiple digits (e.g., tens and ones of the score, or remaining countdown time) to be displayed using a single set of segment outputs.

4.2 Data and Control Flow

The overall data and control flow can be summarised as follows. Raw button presses from the Nexys board enter the button interface block, which produces debounced pulses and a current difficulty level. These signals, together with the outputs of the countdown and game-time counters and the current score, feed into the game control FSM. The FSM determines the current game phase and, based on that phase, enables or clears the two-second counters, the score counter, and the mole controller.

In parallel, the mole controller uses the system clock and difficulty level to control the visible duration of each mole, and drives the five LED outputs such that exactly one LED is active at a time. When the hammer button pulse occurs, the design uses the switch vector to decide which mole the player attempted to hit, and generates a hit pulse if it matches the active LED. This hit pulse drives the score counter, which increments the stored score value as long as scoring is enabled by the FSM.

The game control FSM also selects which numeric value should be displayed at each moment. During the countdown phase, it derives a countdown value (5 down to 0) from the countdown counter; during the playing and game-over phases, it forwards the current score. This 8-bit display_value is passed to the display subsystem, which converts it into digit values and segment patterns and scans the physical seven-segment display at a high enough rate that the human eye perceives a stable multi-digit number.

By structuring the system in this modular way, each block has a clear and limited responsibility, and all interactions occur through well-defined synchronous interfaces. This high-level architecture makes the design easier to reason about, to simulate, and to debug on real hardware.

5. Detailed Module Design

The design is organised as a set of small, reusable modules, each with a single clear responsibility:

- Button interface and debouncing

  Raw push-buttons (start, clear, difficulty, hammer) are filtered by a debouncing module that waits for the input to remain stable for a fixed number of clock cycles and then produces a clean, one-cycle pulse. The difficulty button also updates a small 2-bit register that encodes the current difficulty level.

- Clock divider and second counters

  A clock divider derives a 1 Hz enable tick and a faster scan clock from the 100 MHz system clock. Two generic second counters use the 1 Hz tick: one implements the 5 s pre-game countdown, the other implements the 30 s game timer. Each counter has enable and clear inputs so the game-control FSM can start, stop, and reset them.

- Score counter

  The score counter is an 8-bit up-counter that increments on each hit pulse while enabled and is synchronously cleared at the beginning of a game or when the clear-score button is pressed. The score is limited to 0–99 so that it fits on two seven-segment digits.

- Mole controller and pseudo-random selection

  The mole controller tracks the currently active mole index (0–4) and decodes it into a one-hot LED vector so that exactly one LED is lit. A tick counter, parameterised by the difficulty level, determines how long each mole remains visible. When a visibility window ends, or a hit occurs, the next mole index is chosen from a pseudo-random-looking sequence (e.g., via an LFSR or counter + modulo). The module compares the hammer selection to the active index and generates a hit pulse on correct hits.

- Seven-segment display system

  The display subsystem receives an 8-bit display_value and splits it into tens and ones digits. A scan counter periodically selects which digit is active, and a decoder maps each 4-bit digit to the corresponding seg[6:0] pattern. Because the digits are scanned at a high rate, the display appears static to the user.

All of these modules are instantiated and connected in the top-level module, which maps logical signals to the physical pins of the Nexys board.

6. Finite State Machines

The central finite state machine in the design is the game control FSM. It coordinates timers, scoring, and mole control using four states:

- IDLE: after reset, all counters are cleared, the mole controller is disabled, and the display shows zero. Difficulty can be changed in this state. A start pulse moves the FSM to COUNTDOWN.

- COUNTDOWN: The countdown counter increments from 0 to 5 using the 1 Hz tick, and the display shows the remaining time (5 down to 0). The game timer and score counter remain cleared. When the countdown reaches 5, the FSM enters PLAYING. Pressing start again restarts the countdown.

- PLAYING: the game timer counts from 0 to 30 seconds, the score counter and mole controller are enabled, and the display shows the current score. When the game timer reaches 30, the FSM moves to GAME_OVER. Pressing start early forces a restart (back to COUNTDOWN with cleared counters).

- GAME_OVER: timers and mole controller are disabled, and the final score is frozen on the display. Difficulty can again be adjusted. A start pulse initiates a new round by returning to COUNTDOWN.

The mole controller can also be viewed as a simple two-mode FSM: in an ACTIVE window, it holds one mole on and waits for a hit or timeout; then it ADVANCES to pick the next mole and starts a new window.

7. Implementation and Timing Considerations

All sequential logic in the design is synchronous to the 100 MHz system clock. Human-scale timing is derived using enable pulses instead of separate clock domains: the 1 Hz tick drives the second counters, while the mole visibility windows are implemented by counting raw clock cycles up to difficulty-dependent thresholds. This single-clock approach avoids clock-domain crossing issues and simplifies timing closure.

The seven-segment display uses a scan clock in the kilohertz range to cycle through the active digit index. At this rate, the digits appear continuously lit to the human eye while keeping the hardware simple. FPGA implementation reports indicate that the design meets the 100 MHz timing constraint with comfortable slack, and all top-level I/O pins are assigned explicit locations and I/O standards in the constraint file to satisfy design-rule checks.

8. Verification and Testing

Verification was carried out in both simulation and on hardware.

At the module level, the second counter, score counter, game control FSM, and mole controller were each tested in isolation. Testbenches applied sequences of enables, clear, and input pulses to confirm correct counting, state transitions, and hit detection. For example, the FSM was driven through a shortened game cycle (reduced countdown and game durations) to verify the sequence IDLE → COUNTDOWN → PLAYING → GAME_OVER and to check that enable and clear signals were asserted only in the intended states.

At the system level, the integrated design was loaded onto the Nexys board. Hardware tests checked that:

- The reset and start buttons reliably return the system to the idle state and initiate a new game.
- The countdown lasts approximately 5 seconds, and the game phase lasts approximately 30 seconds.
- Exactly one mole of LED is lit at a time, and the speed changes with difficulty.
- The score increments only on correct hits during an active mole window, and
- The final score remains stable after the game ends and can be cleared by the clear-score button.

Observed hardware behaviour matched the expected design, giving confidence in the correctness of the implementation.

9. Design Challenges and Lessons Learned

Several practical issues arose during development:

- Raw button inputs caused multiple unintended events due to bounce, which was resolved by adding systematic debouncing and one-cycle pulse generation.
- Misunderstanding the active-low digit on the seven-segment display initially led to only some digits lighting; a proper scan counter and correct polarity fixed this.
- Choosing suitable timings for countdown, game duration, and mole visibility required some experimentation to make the game both playable and demonstrative.
- Early versions suffered from missing or incorrect pin constraints, leading to non-responsive buttons or LEDs until the constraint file was carefully aligned with the Nexys board documentation.

From this project, we learned the importance of designing FSMs and block diagrams before coding, keeping modules small with clear interfaces, and verifying behaviour in simulation before moving to hardware.

10. Conclusion

In conclusion, the project successfully produced a fully functional Whack-a-Mole reaction game on the Nexys FPGA board using Verilog. The final system includes a 5-second pre-game countdown, a 30-second timed gameplay period, single active moles with difficulty-based visibility, reliable hit detection, accurate scoring, and real-time output on a multiplexed seven-segment display. Structuring the design into clear, modular components and coordinating them through a central FSM allowed us to build, verify, and debug a complete synchronous digital system. Overall, the project demonstrates a strong understanding of key EC311 concepts and showcases how digital design principles translate into a tangible, interactive hardware application.