FPGA Reflex Timer

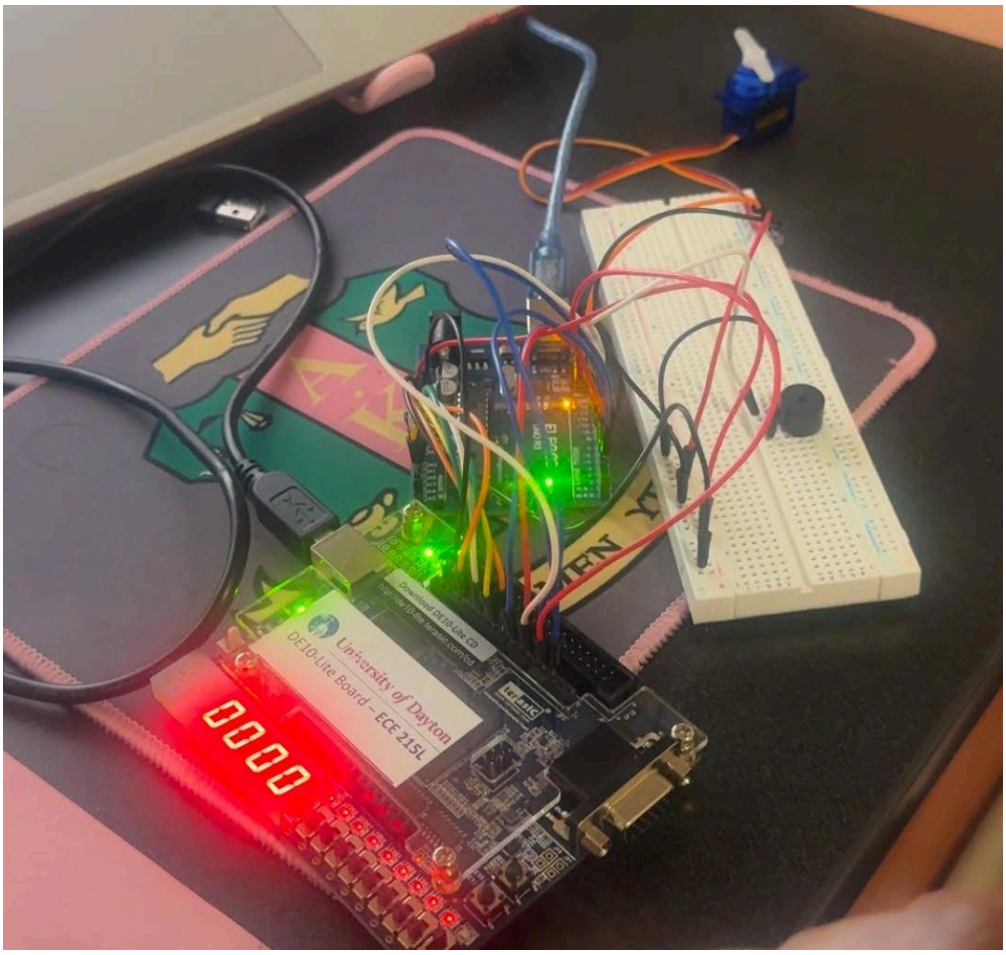
ECE 215L Final Project · Verilog + Arduino · DE10-Lite FPGA · University of Dayton

My Role: Primary developer: all Verilog code, Arduino interface code, hardware wiring, and advanced version with servo + buzzer feedback

Verilog / FPGA	Arduino	DE10-Lite Board	ECE 215L Final

CIRCUIT PHOTO

Built Hardware



DE10-Lite FPGA board with Arduino Nano, breadboard, servo motor, and buzzer — ECE 215L, University of Dayton.

PROJECT OVERVIEW

What I Built

A hardware reflex timer built on the DE10-Lite FPGA board using Verilog. The system measures a user's reaction time to 0.01-second precision using the board's 50 MHz clock. Press a button, wait for the LED to flash after a 3-second delay, then press again the 7-segment display shows your exact reaction time in SS.CC format.

I also designed and built an advanced version that connects the FPGA to an Arduino Nano via parallel GPIO. The Arduino reads the time in binary and drives a servo motor and buzzer with feedback that scales to how fast or slow the user reacts.

MY CORE CONTRIBUTION — VERILOG

1. Finite State Machine (4 States)

I designed the complete FSM that controls all game logic:

```
parameter DEFAULT = 2'b00; // Idle — waiting for first button press
parameter LOBBY   = 2'b01; // 3-second delay (150,000,000 clock cycles)
parameter GO      = 2'b10; // LED on, timer counting, waiting for press
parameter FINISH  = 2'b11; // Timer stopped, result displayed

// Transitions:
// DEFAULT → button_pressed → LOBBY
// LOBBY   → delay_count >= 150M-1 → GO
// GO      → button_pressed → FINISH
// FINISH  → button_pressed → DEFAULT
```

2. Clock Divider — Centisecond Timer

Dividing 50 MHz down to 0.01s ticks for precision timing:

```
parameter clock_PT = 500_000; // 50MHz / 500K = 0.01s per tick
parameter delay    = 150_000_000; // 3.0 second lobby delay

// In GO state, every 500,000 cycles:
cs_total <= cs_total + 1;   // 0-99 centiseconds
if (cs_total == 99) begin
    cs_total <= 0;
    s_0 <= s_0 + 1;
    if (s_0 == 9) s_T <= s_T ^ 1;
end
```

3. Button Edge Detector

Flip-flop based falling-edge detection to prevent multi-trigger bounce:

```
always @(posedge clock)
    button_previous <= button;

// Falling edge: 1→0 (active-low button)
assign button_pressed = (~button && button_previous);
```

4. 7-Segment Display Decoder

Wrote the seg7 function and wired all four digits in SS.CC format:

```
4'd0: seg7 = 7'b1000000;  4'd5: seg7 = 7'b0010010;
4'd1: seg7 = 7'b1111001;  4'd6: seg7 = 7'b0000010;
4'd2: seg7 = 7'b0100100;  4'd7: seg7 = 7'b1111000;
4'd3: seg7 = 7'b0110000;  4'd8: seg7 = 7'b0000000;
4'd4: seg7 = 7'b0011001;  4'd9: seg7 = 7'b0010000;
```

```
assign cs_0_disp = cs_total % 10;
assign cs_T_disp = cs_total / 10;
```

ADVANCED VERSION — ARDUINO

5. FPGA → Arduino Parallel Interface

I added 5 output ports to the Verilog and wrote the full Arduino sketch:

```
output [3:0] s_0_out,  // seconds ones (4 bits)
output       s_T_out,  // seconds tens (1 bit)
output [3:0] cs_0_out, // centisecond ones (4 bits)
output [1:0] cs_T_out  // centisecond quarter-band (2 bits)

// Arduino reconstructs time:
float time = (s_T*10) + s_0 + (cs_T*0.25) + (cs_0*0.01);

// Feedback:
time < 5  → 1800Hz + 180° servo  (FAST)
time < 10 → 1200Hz + 120° servo  (MEDIUM)
else      → 600Hz  +  60° servo  (SLOW)
```

TECHNICAL DETAILS

Technical Highlights

FPGA	DE10-Lite (Intel, 50 MHz system clock)
HDL Language	Verilog (synthesized in Quartus Prime)
Timer Precision	0.01 seconds (centisecond resolution)
FSM States	4: DEFAULT → LOBBY → GO → FINISH
Display	4-digit 7-segment (SS.CC format)
Clock Divider	500,000 cycles = 0.01s; 150M = 3s delay
Arduino	Nano — reads 11 GPIO pins, drives servo + buzzer
Course	ECE 215L — University of Dayton, Spring 2026

TAKEAWAYS

What I Learned

- Designing a 4-state FSM in Verilog from scratch: mapping real-world timing to synchronous hardware logic.
- Clock division at the hardware level, 50 MHz / 500,000 = 100 Hz gave 0.01s resolution.
- Falling-edge button detection using flip-flops to prevent bounce and multi-trigger issues.
- Designing a parallel binary GPIO protocol between FPGA and Arduino across 11 signal lines.
- Building a complete embedded system end-to-end: Verilog FSM → GPIO → Arduino → servo + buzzer.
- Physical hardware integration \breadboarding servo, buzzer, and Arduino to the DE10-Lite board.

Full Verilog and Arduino source available upon request — loganpride525@email.com