

# Softcore System on a Chip Final Project:

## Motion Aware Study Timer

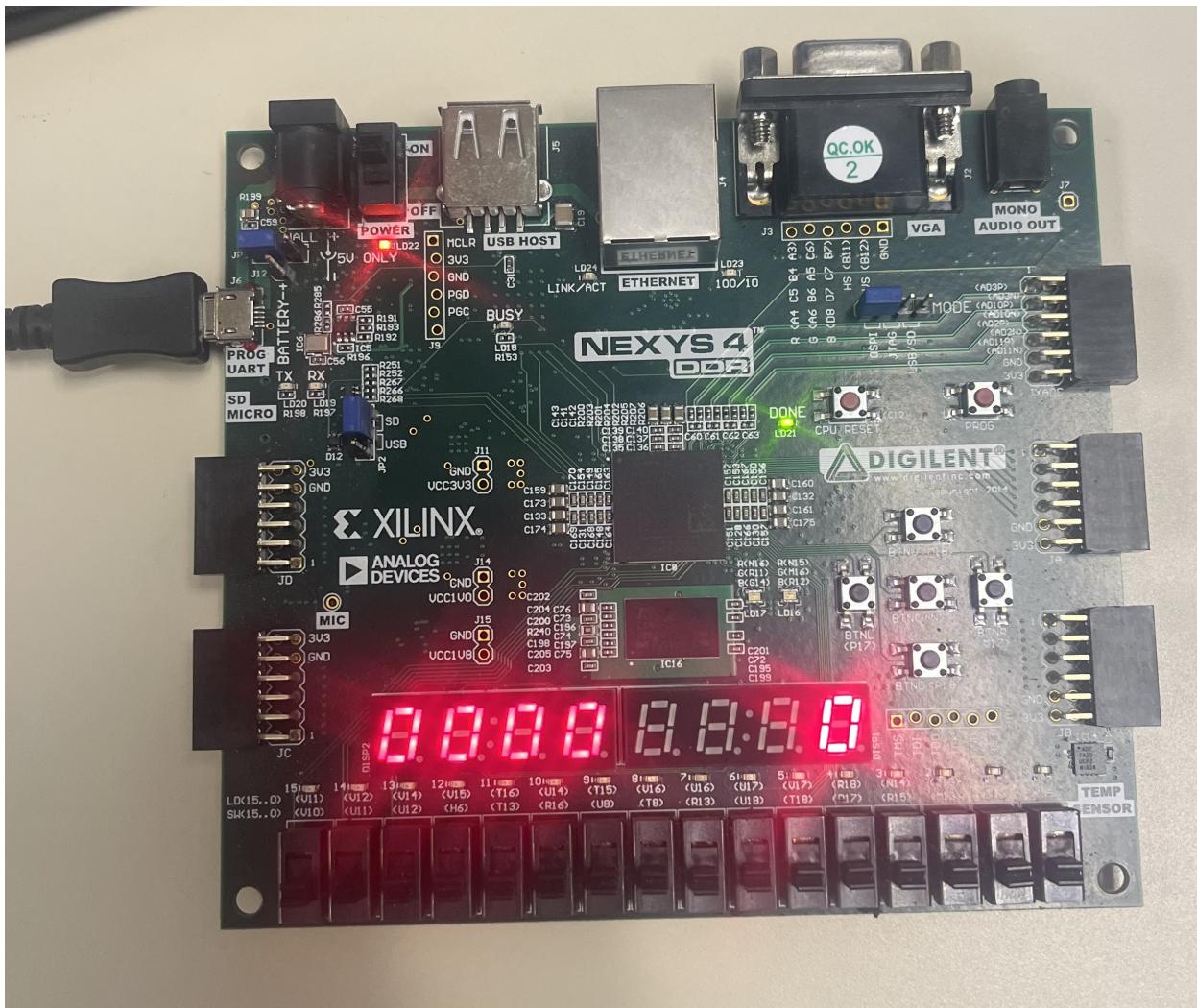
David Edeni

December 15th, 2025

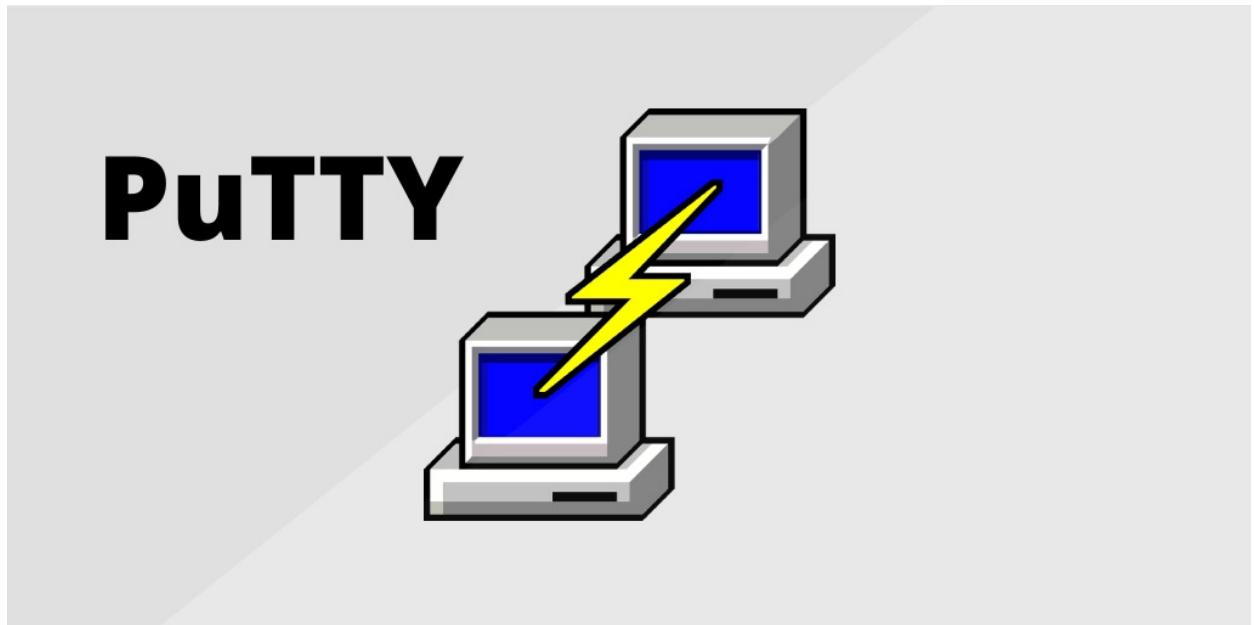
### Lab Description:

This project is a Verilog implementation of an FPGA Study Timer that tracks focused study time using motion detection. This timer pauses when motion is detected, and displays focused time on the Nexys DDR-4's seven-segment display and over a UART interface such as PUTTY.

Based on a specified hardware configuration, this is what the Nexys DDR-4 FPGA Board would look like in my project's default state.



PUTTY is a UART interface program that displays data read from sensors on FPGA Boards such as Nexys DDR-4 Boards and Atmel SAM4L Boards. PUTTY displays a wide variety of data, including temperature, distance, and light frequencies.



The temp sensor reader System Verilog module simulates a temperature sensor. It periodically produces a temperature value with small, random variation.

Figure 1: The temp sensor reader (temperature sensor reader) module:

```

`timescale 1ns / 1ps // Defines this module's simulation time unit (1 ns) and time precision unit (1 ps)

module temp_sensor_reader (
    input logic clk,
    input logic rst,
    output logic signed [11:0] temperature,
    output logic valid,
    output logic i2c_scl,
    inout wire i2c_sda
);

    logic [11:0] counter;
    logic [7:0] lfsr;

    assign i2c_scl = 1'b1;
    assign i2c_sda = 1'bs;

    always_ff@(posedge(clk), posedge(rst)) begin
        if(rst) begin
            counter <= 0;
            temperature <= 12'sd20;
            valid <= 0;
            lfsr <= 8'hA5;
        end
        else begin
            counter <= counter + 1;
            valid <= 0;

            if(counter == 1_000_000) begin
                counter <= 0;

                // 8-bit LFSR: x^8 + x^6 + x^5 + x^4 + 1
                lfsr <= {lfsr[6:0], lfsr[7] ^ lfsr[5] ^ lfsr[4] ^ lfsr[3]};

                // Small signed variation: -2 to +1
                temperature <= temperature + $signed({1'b0, lfsr[1:0]}) - 12'sd2;
            end
            valid <= 1;
        end
    end
end
endmodule

```

The sseg driver drives a seven-segment display and makes it flash when motion is detected.

Figure 2: The sseg driver (seven-segment display driver) module:

---

```
'timescale 1ns / 1ps // Defines this module's simulation time unit (1 ns) and time precision unit (1 ps)

module sseg_driver (
    input logic clk,
    input logic rst,
    input logic [15:0] value,
    input logic moving,
    output logic [6:0] seg,
    output logic [3:0] an
);

    logic [23:0] flash_counter;
    logic flash_state;

    // Flash counter (~0.1s period at 100MHz)
    always_ff@(posedge(clk), posedge(rst)) begin
        if(rst) begin
            flash_counter <= 0;
            flash_state <= 0;
        end
        else if(moving) begin
            flash_counter <= flash_counter + 1;
            if(flash_counter == 5_000_000) begin
                flash_counter <= 0;
                flash_state <= ~flash_state;
            end
        end
        else begin
            flash_counter <= 0;
            flash_state <= 1; // steady ON when still
        end
    end

    // Seven-segment display
    always_ff@(posedge(clk), posedge(rst)) begin
        if(rst) begin
            an <= 4'b1110;
            seg <= 7'b1000000;
        end
        else begin
            an <= 4'b1110;
            if(moving) begin
                seg <= flash_state ? 7'b1111001 : 7'b0000000;
            end
            else begin
                seg <= 7'b1000000; // 0 when still
            end
        end
    end
endmodule
```

The motion classifier System Verilog module decides whether the system is "moving" based on changes between consecutive temperature samples.

Figure 3: The motion classifier module:

```
'timescale 1ns / 1ps // Defines this module's simulation time unit (1 ns) and time precision unit (1 ps)

module motion_classifier (
    input logic clk,
    input logic rst,
    input logic sample_valid,
    input logic signed [11:0] value,
    output logic moving
);

    logic signed [11:0] last;
    logic signed [12:0] diff;

    always_ff@(posedge(clk), posedge(rst)) begin
        if(rst) begin
            last <= 12'`sd20;
            moving <= 0;
        end
        else if(sample_valid) begin
            diff <= value - last;

            if(diff < 0)
                diff <= -diff;

            moving <= (diff > 13'`sd2);
            last <= value;
        end
    end
endmodule
```

The uart tx2 module sends a UART character periodically based on motion read from DDR-4 Board sensors.

Figure 4: The uart tx2 module:

```

'timescale 1ns / 1ps // Defines this module's simulation time unit (1 ns) and time precision unit (1 ps)

module uart_tx2 #(
    parameter CLK_FREQ = 100_000_000,
    parameter BAUD    = 115200
) (
    input logic clk,
    input logic rst,
    input logic moving,
    output logic tx,
    output logic busy
);

    localparam DIV = CLK_FREQ / BAUD;
    logic [15:0] div_count;
    logic [3:0] bit_count;
    logic [9:0] shift;

    logic send_pulse;
    logic [23:0] send_counter;

    // Generate a send pulse periodically (~0.1s)
    always_ff@(posedge(clk), posedge(rst)) begin
        if(rst) begin
            send_counter <= 0;
            send_pulse <= 0;
        end
        else begin
            send_counter <= send_counter + 1;
            if(send_counter >= 10_000_000) begin // ~0.1s
                send_counter <= 0;
                send_pulse <> 1;
            end
            else begin
                send_pulse <= 0;
            end
        end
    end

    // UART transmission
    always_ff@(posedge(clk), posedge(rst)) begin
        if(rst) begin
            tx <= 1'b1;
            busy <= 0;
            div_count <= 0;
            bit_count <= 0;
            shift <= 10'b1111111111;
        end
        else if(send_pulse && !busy) begin
            shift <= {1'b1, (moving ? "M" : "S"), 1'b0}; // start/stop + data
            busy <= 1;
            div_count <= 0;
            bit_count <= 0;
        end
        else if(busy) begin
            if(div_count == DIV-1) begin
                div_count <= 0;
                tx <= shift[0];
                shift <= {1'b1, shift[9:1]};
                bit_count <= bit_count + 1;
                if(bit_count == 9)
                    busy <= 0;
            end
            else
                div_count <= div_count + 1;
        end
    end
endmodule

```

Finally, the motion aware study timer top (system implementation) System Verilog module is the glue that ties all our sub-modules (temp sensor reader, sseg driver, motion classifier) together to drive the DDR-4's seven-segment display. This module connects the FPGA's hardware inputs (clock, switches) and outputs (seven-segment LEDs). It instantiates the four functional modules: temp sensor reader, sseg driver, motion classifier; and wires them together so the motion timer system works properly.

Figure 5: The motion aware study timer top (system implementation) module:

```
'timescale 1ns / 1ps // Defines this module's simulation time unit (1 ns) and time precision unit (1 ps)

module motion_aware_study_timer_top #(parameter BRG_BASE = 32'h0000_0000)
(
    input logic clk,
    input logic rst,
    output logic [6:0] seg,
    output logic [3:0] an,
    output logic uart_tx_o,
    output logic i2c_scl,
    inout wire i2c_sda
);

    logic signed [11:0] temperature;
    logic sample_valid;
    logic moving;

    // Temperature sensor reader (simulated with proxy)
    temp_sensor_reader temp (
        .clk(clk),
        .rst(rst),
        .temperature(temperature),
        .valid(sample_valid),
        .i2c_scl(i2c_scl),
        .i2c_sda(i2c_sda)
    );

    // Motion classifier using temperature changes
    motion_classifier mc (
        .clk(clk),
        .rst(rst),
        .sample_valid(sample_valid),
        .value(temperature),
        .moving(moving)
    );

    // Seven-segment display driver with flashing when moving
    sseg_driver sseg {
        .clk(clk),
        .rst(rst),
        .value(moving ? 16'd1 : 16'd0),
        .moving(moving),
        .seg(seg),
        .an(an)
    };

    // UART output printing M/S repeatedly
    uart_tx2 uart (
        .clk(clk),
        .rst(rst),
        .moving(moving),
        .tx(uart_tx_o),
        .busy()
    );

endmodule
```