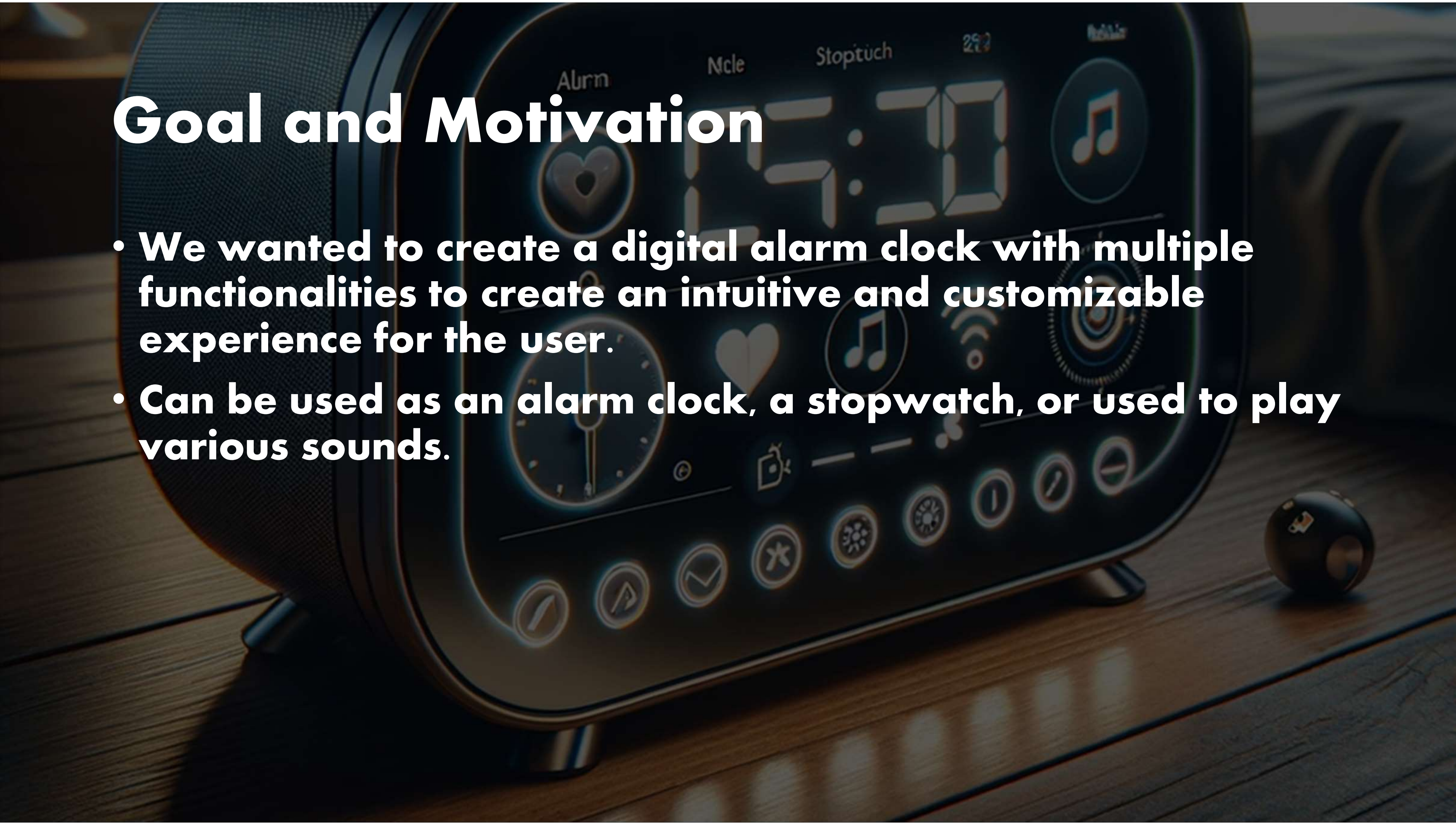
Digital Alarm Clock

Artix - 7

Santos Zuniga, Karl Carisme, Waseem Ridhuan, Michael Moran



Goal and Motivation

- We wanted to create a digital alarm clock with multiple functionalities to create an intuitive and customizable experience for the user.
- Can be used as an alarm clock, a stopwatch, or used to play various sounds.

Functionality

- Allow the user to choose between multiple modes on the device
- Use a stopwatch when needed
- Set an alarm time and customize the alarm sound that is output
- Play different sounds on demand
- Interactive minigame used to turn off the alarm.

Specification

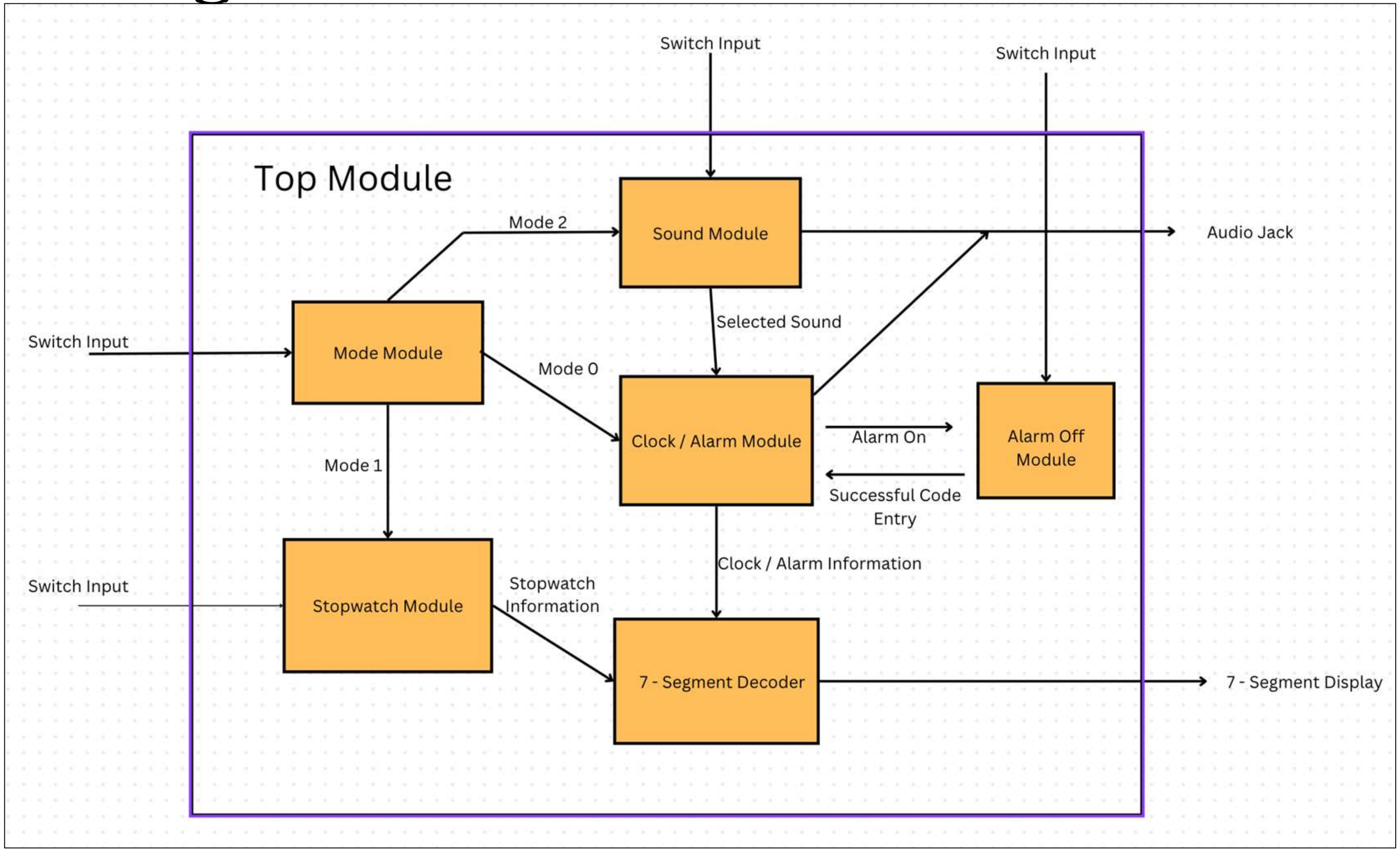
Requirements

- Display the time correctly
- Have alarm working
- Output sounds

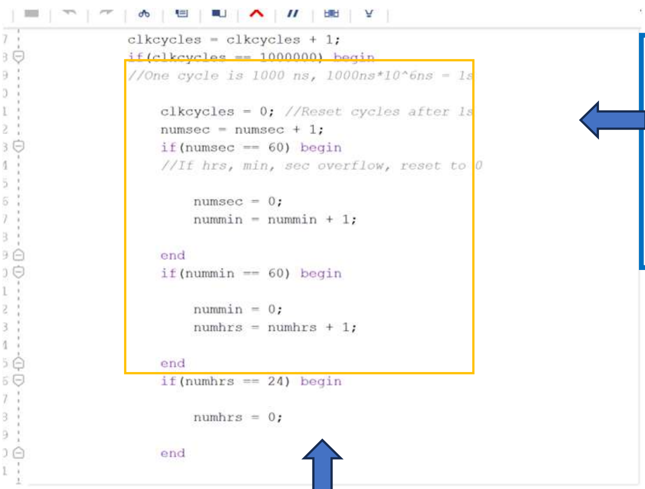
Constraints

- Deadline
- Lab issues
- Knowledge of audio output on FPGA

Block Diagram



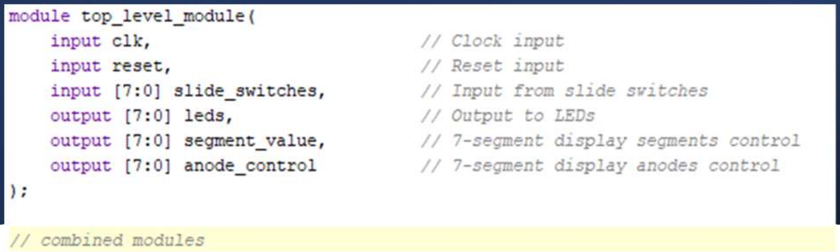
Code Snippet



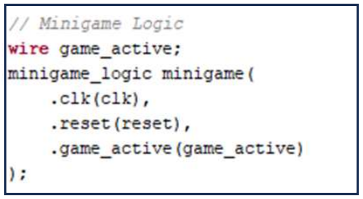
```
        clkcycles = clkcycles + 1;
        if(clkcycles == 1000000) begin
        //One cycle is 1000 ns, 1000ns*10^6ns = 1s

            clkcycles = 0; //Reset cycles after 1s
            numsec = numsec + 1;
            if(numsec == 60) begin
            //If hrs, min, sec overflow, reset to 0

                numsec = 0;
                nummin = nummin + 1;

            end
            if(nummin == 60) begin

                nummin = 0;
                numhrs = numhrs + 1;

            end
            if(numhrs == 24) begin

                numhrs = 0;

            end
```

One clock cycle is 1000 ns, so every 10^6 s, increment clock by 1s

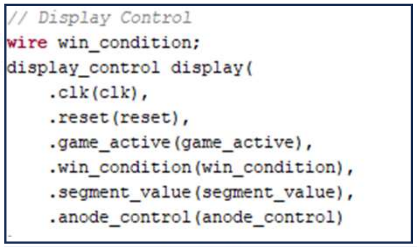
Every time the number of seconds, minutes, or hours overflows, resets value to 0 and increments the next biggest time unit if available.



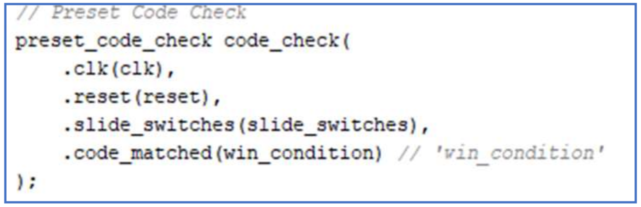
```
module top_level_module(
    input clk,                          // Clock input
    input reset,                        // Reset input
    input [7:0] slide_switches,         // Input from slide switches
    output [7:0] leds,                  // Output to LEDs
    output [7:0] segment_value,         // 7-segment display segments control
    output [7:0] anode_control          // 7-segment display anodes control
);

// combined modules
```



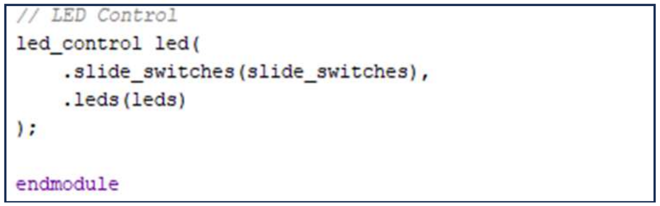
```
// Minigame Logic
wire game_active;
minigame_logic minigame(
    .clk(clk),
    .reset(reset),
    .game_active(game_active)
);
```



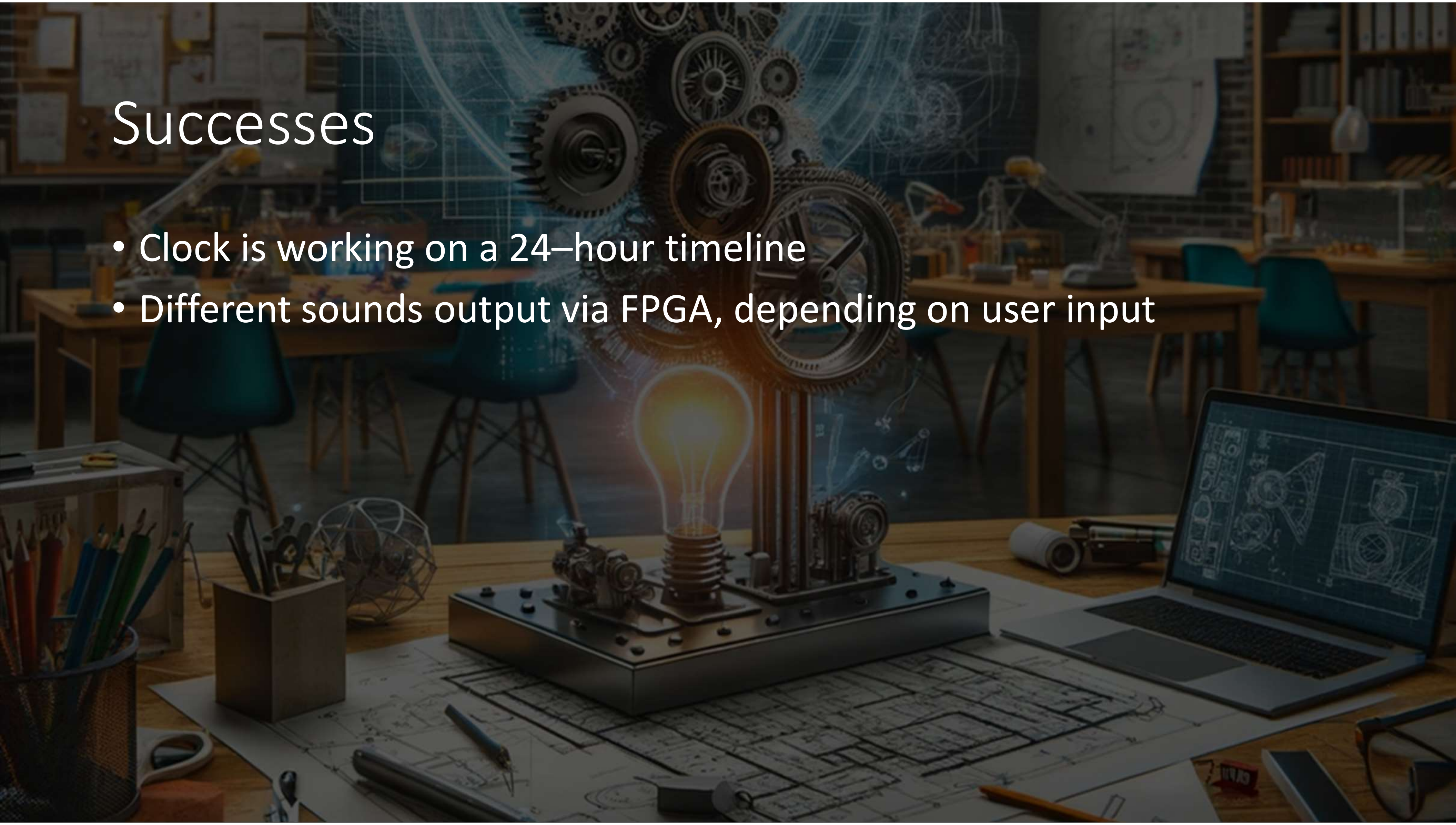
```
// Display Control
wire win_condition;
display_control display(
    .clk(clk),
    .reset(reset),
    .game_active(game_active),
    .win_condition(win_condition),
    .segment_value(segment_value),
    .anode_control(anode_control)
```



```
// Preset Code Check
preset_code_check code_check(
    .clk(clk),
    .reset(reset),
    .slide_switches(slide_switches),
    .code_matched(win_condition) // 'win_condition'
);
```

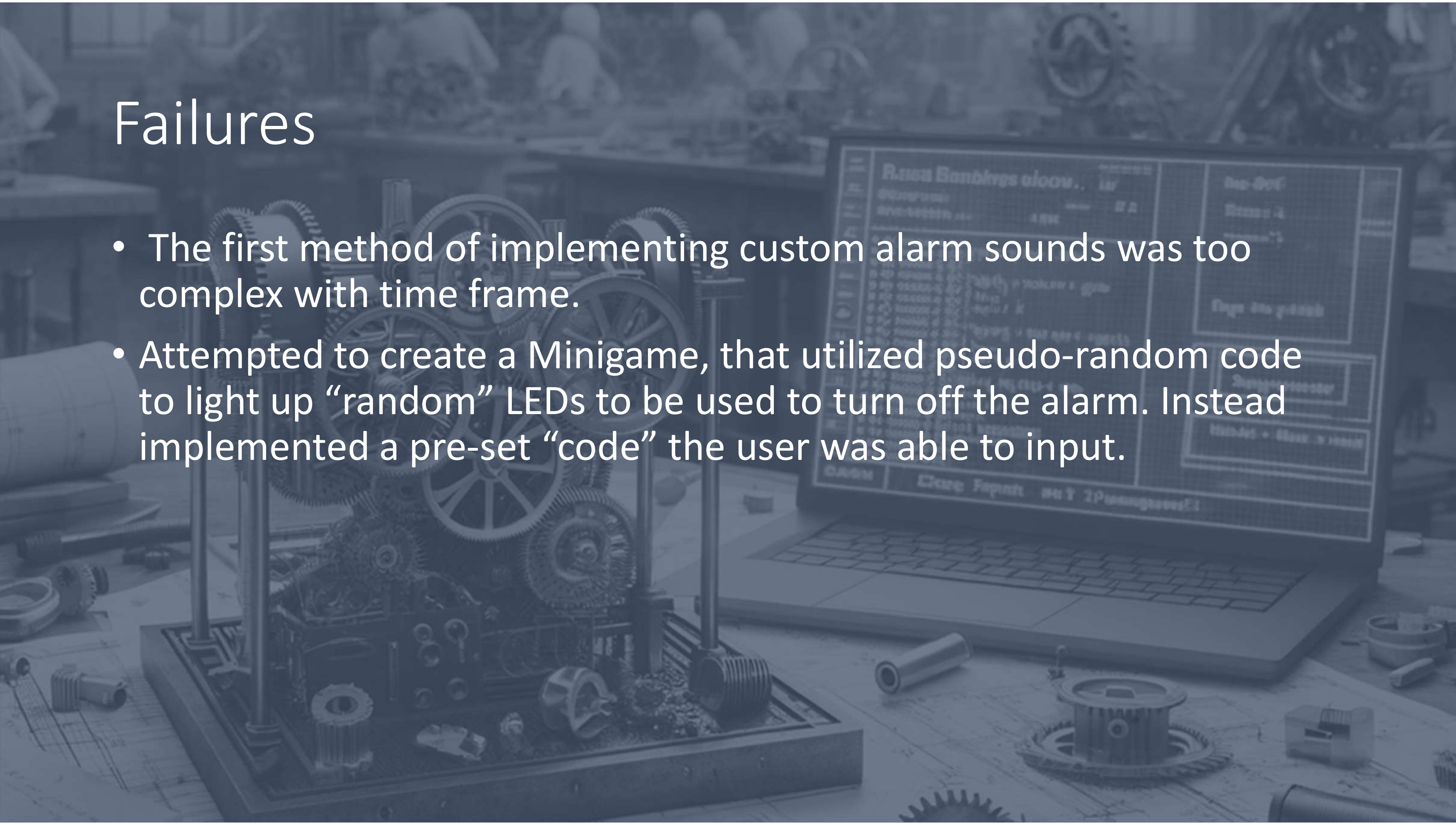


```
// LED Control
led_control led(
    .slide_switches(slide_switches),
    .leds(leds)
);

endmodule
```



Successes

- Clock is working on a 24–hour timeline
- Different sounds output via FPGA, depending on user input



Failures

- The first method of implementing custom alarm sounds was too complex with time frame.
- Attempted to create a Minigame, that utilized pseudo-random code to light up “random” LEDs to be used to turn off the alarm. Instead implemented a pre-set “code” the user was able to input.