StopWatch

Marco Losso

December 16, 2024

Introduction

This report describes the design of a stopwatch developed in Verilog. The stopwatch counts seconds, tenths, and hundredths of a second and includes features such as reset, start/stop, partial time display, and countdown functionality. Additionally, functions were implemented to count minutes and hours and to switch between different display modes.

The Verilog project was simulated using ModelSim and finally loaded onto the Terasic DE1-SoC board to showcase the final result on its six seven-segment displays.

1 Stopwatch Description

The basic stopwatch is designed to count seconds, tenths, and hundredths of a second. Two main buttons have been implemented: **reset** and **start/stop**. The reset button clears the count, while the start/stop button allows the counting process to start and stop. When the stopwatch is reset, it remains at zero until the start/stop button is pressed again to begin counting.

2 Clock Divider

The project utilizes a **clock divider** to generate a 100 Hz clock signal from a 50 MHz internal signal of the DE1-SoC board. This ensures an appropriate update frequency for counting hundredths of a second. This is implemented by counting the number of rising edges of the 50 MHz clock, which totals 500,000 in 0.01 seconds: thus 250,000 for the high state and the same for the low state. At the end of the count, the value of the output 100 Hz clock is toggled to generate the high and low states. Below is the complete module that implements the **clock divider**, and it is easy to recognize the operations just described.

Listing 1: Clock Divider

```
module clock_divider (
    input clk_in,     // Clock at 50 MHz
    output reg clk_out    // Clock at 100 Hz
);
    reg [18:0] counter;
    initial begin
        counter = 0;
        clk_out = 0;
    end

    always @(posedge clk_in) begin
        if (counter == 250000) begin
            counter <= 0;
            clk_out <= ~clk_out;
        end else begin
            counter <= counter + 1;
        end
    end
endmodule
```

2.1 Description of Variables

- **clk_in**: 50 MHz input clock signal.
- **clk_out**: 100 Hz output clock signal.
- **counter**: Counter that tracks clock cycles to generate the output clock.

3 Counter

Subsequently, the actual counting mechanism was implemented, and this was achieved using nested `if` statements, which also introduced the counting of minutes and hours.

Below is an excerpt from the module that implements the **counter**, highlighting the structure of the `if` statements. Additionally, it shows how the `reset` sets the count value to 0, and the use of `enable` to stop or restart the counting process, thereby implementing the start/stop function.

Listing 2: Estratto del Modulo del Contatore

```
always @(posedge clk_100Hz or posedge rst) begin
    if (rst) begin
        // Reset del conteggio
        centisec <= 0;
        sec <= 0;
        min <= 0;
        hr <= 0;
        running <= 0;
    end else begin
        if (enable && !enable_reg) begin
            running <= ~running; // Cambia lo stato del contatore
        end
        enable_reg <= enable; // Aggiorna lo stato del tasto enable
        if (running) begin
            if (centisec == 99) begin
                centisec <= 0;
                if (sec == 59) begin
                    sec <= 0;
                    if (min == 59) begin
                        min <= 0;
                        if (hr == 23)
                            hr <= 0;
                        else
                            hr <= hr + 1;
                    end else
                        min <= min + 1;
                end else
                    sec <= sec + 1;
            end else
                centisec <= centisec + 1;
        end
    end
end
```

3.1 Description of Variables

- **centisec**: Hundredths of a second counter.
- **sec**: Seconds counter.
- **min**: Minutes counter.
- **hr**: Hours counter.
- **running**: State of the counter (started/stopped).
- **enable_reg**: Previous state of the enable button register.

3.2 First Simulation

Before proceeding with downloading the project onto the board, a simulation of this initial part was conducted using ModelSim to verify the correct operation of the counter as well as the `reset` and `enable` signals.

The figure below shows the simulation results, highlighting how the `reset` and `enable` signals interact with the counter.

Figure 1: Simulation of the counter in ModelSim

4 Partial Time Implementation

The partial time display functionality was added using a dedicated button. When the partial time button is pressed, the current stopwatch values are frozen and displayed on the 7-segment displays for 5 seconds, without interrupting the actual counting. This allows the partial time to be viewed at any moment while keeping the stopwatch running.

The following Verilog code snippet shows how the stopwatch values are stored and displayed for 5 seconds when the partial time button is pressed:

Listing 3: Gestione del Tempo Parziale

```
always @(posedge clk) begin
    if (parziale_mode) begin
        if (parziale_counter < 500) begin // 5 secondi a 100 Hz
            parziale_counter <= parziale_counter + 1;
        end else begin
            parziale_mode <= 0;
            parziale_counter <= 0;
        end
    end

    if (parziale && !parziale_mode) begin
        // Attiva la visualizzazione del tempo parziale
        // quando il tasto e' premuto
        parziale_mode <= 1;
        parziale_counter <= 0;
        // Memorizza i valori attuali del contatore
        parziale_centisec <= centisec;
        parziale_sec <= sec;
        parziale_min <= min;
        parziale_hr <= hr;
    end
end
```

4.1 Description of Variables

- **parziale_mode**: State of the partial time.
- **parziale_counter**: Counter that tracks the display time of the partial time.
- **parziale_centisec**, **parziale_sec**, **parziale_min**, **parziale_hr**: Store the current stopwatch values when the partial time button is pressed.

5 Countdown Implementation

A functionality was added to allow the stopwatch to count down using a dedicated switch. When the switch is activated, the stopwatch counts down; when the switch is deactivated, the stopwatch counts up.

5.1 Verilog Code for Countdown

The following Verilog code snippet shows how the counting is managed based on the state of the switch:

Listing 4: Gestione del Conteggio all'Indietro

```
if (direction) begin
    // Conteggio inverso
    if (centisec == 0) begin
        centisec <= 99;
        if (sec == 0) begin
            sec <= 59;
            if (min == 0) begin
                min <= 59;
                if (hr == 0)
                    hr <= 23;
                else
                    hr <= hr - 1;
            end else
                min <= min - 1;
        end else
            sec <= sec - 1;
    end else
        centisec <= centisec - 1;
end
```

5.2 Description of Variables

- **direction**: Switch that determines the counting direction (0 = up, 1 = down).
- **centisec**, **sec**, **min**, **hr**: Variables that store the stopwatch counting values.

6 Final Project Simulation

The final simulation of the project was implemented in a top entity module that includes the use of actual buttons and switches. This allowed testing the system's behavior under realistic conditions, using physical inputs to control the counter operation and the display on the 7-segment displays.

The counter values were converted to BCD (Binary-Coded Decimal) to be correctly displayed on the 7-segment displays.

6.1 Description of Signals Used

In the final simulation, various signals and buses were used to control the system's operation:

- **CLOCK_50**: 50 MHz clock signal.
- **clk_100Hz**: 100 Hz clock signal generated by the clock divider.
- **KEY[0]**: Reset button.
- **KEY[1]**: Enable button to start and stop the counting.
- **KEY[2]**: Partial time button to freeze the display on the 7-segment displays.
- **HEX**: Bus for the 7-segment displays.
- **disp**: Signal that controls the 7-segment displays.
- **sec, centisec, min**: Variables that store the counter values.

6.2 Simulation Results

The following images show the simulation results:

- In Figure 2, besides using the reset button KEY[0] at the beginning, the enable button KEY[1] is used to stop both the counter (yellow lines) and the display on the 7-segment displays (pink lines).
- In Figure 3, the partial button `KEY[2]` is used to freeze only the display on the 7-segment displays (pink lines), while the counter continues counting (yellow lines).

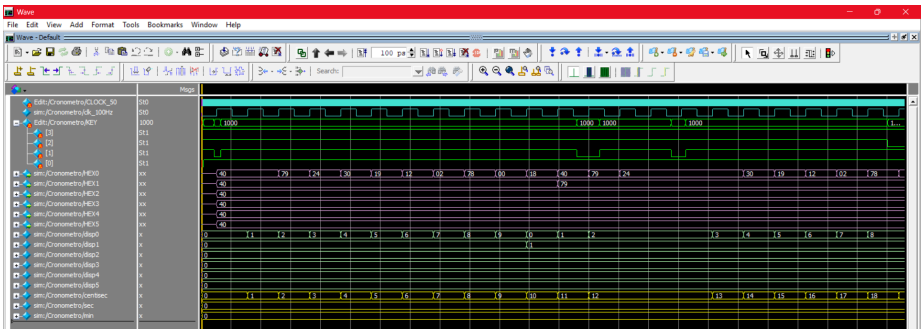

Figure 2: Simulation showing the use of the reset button (KEY[0]) and the enable button (KEY[1]) to stop the counting.

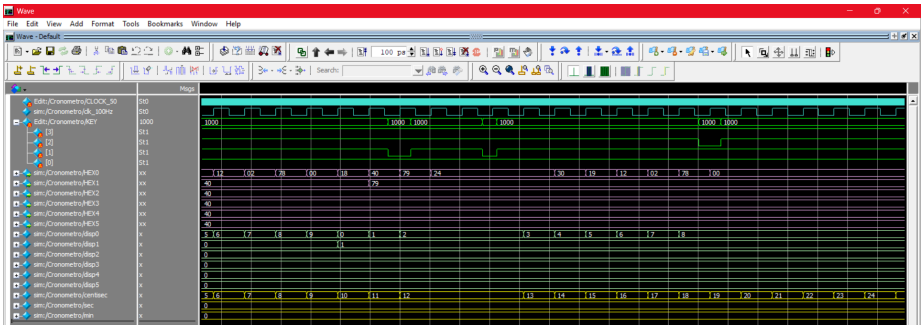

Figure 3: Simulation showing the use of the partial button (KEY[2]) to freeze only the display on the 7-segment displays (pink lines), while the counter continues counting (yellow lines).

7 Display Modes on 7-Segment Displays

Since there are only 6 available 7-segment displays, a functionality was implemented to switch between two display modes using the KEY[3] button. This button allows switching from displaying hundredths of a second, seconds, and minutes (**centisec-sec-min**) to displaying seconds, minutes, and hours (**sec-min-hr**).

7.1 Verilog Code for Display Modes

Below is an excerpt of the Verilog code that manages the display modes on the 7-segment displays:

Listing 5: Gestione delle Modalità di Visualizzazione

```
always @(posedge clk) begin
    if (key3) begin
        mode <= ~mode;
        // Cambia modalita' quando KEY[3] e' premuto
    end
end

always @(*) begin
    if (mode == 0) begin
        // Visualizza centisec-sec-min
        disp0 = centisec % 10;
        disp1 = centisec / 10;
        disp2 = sec % 10;
        disp3 = sec / 10;
        disp4 = min % 10;
        disp5 = min / 10;
    end else begin
        // Visualizza sec-min-hr
        disp0 = sec % 10;
        disp1 = sec / 10;
        disp2 = min % 10;
        disp3 = min / 10;
        disp4 = hr % 10;
        disp5 = hr / 10;
    end
end
```

The division by 10 is done to convert the values into BCD (Binary-Coded Decimal) format. On the 7-segment displays, numbers need to be shown digit by digit in decimal format. The division by 10 separates the units from the tens for each value to be displayed, making it possible to show each digit individually on each display.

7.2 Description of Variables

- **key3**: Button used to change display modes.
- **mode**: Variable that determines the display mode (0 = centisec-sec-min, 1 = sec-min-hr).
- **disp0, disp1, disp2, disp3, disp4, disp5**: Variables that control the 7-segment displays.
- **centisec, sec, min, hr**: Variables that store the counting values.

8 Conclusions

The project developed in Quartus successfully achieved the goal of creating a stopwatch using Verilog. The implemented functionalities include counting hundredths of a second, seconds, minutes, and hours, using buttons for reset and start/stop, partial time display, countdown, and switching between different display modes on the 7-segment displays.

The final simulation of the project was conducted using ModelSim, and the implementation was verified on the Terasic DE1-SoC board. All necessary tests were performed on the hardware board, and the system worked perfectly in every aspect. This validated the stopwatch's functionality in a realistic environment, ensuring that all features operated correctly.

Through this project, essential skills were acquired in hardware design using Verilog, digital systems simulation, and hardware-software integration. The ability to manage clock signals, implement counting logic, and display data on 7-segment displays were crucial skills developed during the project.

The project's success opens the door to potential future improvements, such as adding additional counting modes, integrating with other external devices, and optimizing performance.

These enhancements could further expand the capabilities and applications of the developed stopwatch.