Project 92: Stopwatch Controller

A Comprehensive Study of Advanced Digital Circuits

By: Abhishek Sharma , Ayush Jain , Gati Goyal, Nikunj Agrawal

Documentation Specialist: Dhruv Patel & Nandini Maheshwari

Contents

1 Project Overview

The Stopwatch Controller project aims to design and implement a precise and efficient electronic stopwatch system. This system is crucial in applications requiring accurate time measurement, such as sports, healthcare, education, and industrial processes. The project involves designing the hardware and software components to create a reliable timer that can start, stop, and reset with high precision.

The core features include real-time digital display, start/stop/reset functionalities, and optional split timing. The system will be controlled through simple user inputs such as buttons or touch interfaces. The technical approach involves using a microcontroller (e.g., Arduino or Raspberry Pi), along with a display unit (like a 7-segment display or LCD). The software will be programmed in C/C++ or Python, with real-time clock (RTC) integration for accuracy.

Applications of the stopwatch controller span sports event timing, laboratory experiments, industrial cycle monitoring, and healthcare procedures. The project aims to deliver a portable, compact, and cost-effective solution with customizable features to meet the specific needs of users in various fields.

While the project offers advantages like reliability and ease of integration, challenges include ensuring timing accuracy, environmental robustness, and creating a user-friendly interface. The outcome will be a functional prototype and comprehensive documentation.

2 Stopwatch Controller

2.1 Key Components of Stopwatch Controller

Microcontroller (MCU):

- The central processing unit responsible for controlling the stopwatch operations, such as starting, stopping, and resetting the timer. Popular choices include Arduino or Raspberry Pi.

Display Unit:

- A visual output to show the time. This could be:

1. ***7-segment Display:*** Simple, digital format for hours, minutes, and seconds.
2. ***LCD/LED Screen:*** Larger, more detailed display with additional functionalities like lap time.

Input Buttons/Sensors:

- Physical or touch buttons to control the stopwatch. Common buttons include:

1. ***Start/Stop Button:*** To begin or stop the timer.
2. ***Reset Button:*** To reset the timer to zero.
3. ***Lap/Set Buttons:*** For recording split times in advanced models.

Real-Time Clock (RTC):

- An optional component for precise timekeeping, especially for long-term timing accuracy. It ensures accurate tracking of time intervals.

Power Supply:

- The power source for the stopwatch controller. This could be:

1. Battery (e.g., Li-ion or CR2032): For portability and mobile use.
2. USB Power: For continuous operation when plugged into a power source.

Buzzer or Alarm (optional):

- For alerting the user when a specific time interval has elapsed, useful for countdowns or timing limits.

Enclosure (optional):

- A protective casing to house the components, ensuring durability and a neat design.

2.2 Working of Stopwatch Controller

1. Power On

- When the system is powered on, the microcontroller initializes the components, including the display and input buttons. The display will show an initial time value (usually 00:00:00) or a blank screen depending on the design.

2. Start Function

- When the Start/Stop button is pressed, the microcontroller activates the timer function. It begins counting time by using an internal timer or real-time clock (RTC) module to increment the seconds, minutes, and possibly hours. The time is updated on the display in real-time, showing the elapsed time since the start.

3. Stop Function

- When the Start/Stop button is pressed again, the stopwatch stops the counting process. The time on the display freezes, allowing the user to record or view the elapsed time. The microcontroller halts the timer, and the user can either restart or reset the stopwatch.

4. Reset Function

- Pressing the Reset button will set the timer back to its initial state (usually 00:00:00). The microcontroller clears the stored time, preparing the system for a new session.

5. Lap Time (Optional)

- If the system supports Lap time, pressing a Lap button records the current time without stopping the stopwatch. The microcontroller stores the split time and continues counting. This feature is often used in sports or testing scenarios where multiple intervals need to be measured within the same session.

6. Display Update

- The microcontroller continuously updates the display during the timing process, showing the hours, minutes, seconds, and sometimes milliseconds. The system keeps track of elapsed time and adjusts the display accordingly.

7. Buzzer/Alarm (Optional)

- If integrated, the buzzer or alarm will trigger after a set time or upon pressing specific buttons, alerting the user. This feature is useful for timed activities like cooking or exercises.

2.3 RTL Code

Listing 1: Stopwatch Controller

```

module stopwatch_controller (
    input  logic clk, reset, start_stop, // Control signals
    output logic [7:0] count             // 8-bit stopwatch count
);

    // State variables
    typedef enum logic {STOPPED, RUNNING} state_t;
    state_t current_state, next_state;

    // Counter logic
    always_ff @(posedge clk or posedge reset) begin
        if (reset)
            count <= 8'b0; // Reset the count
        else if (current_state == RUNNING)
            count <= count + 1; // Increment the count
    end

    // State transition logic
    always_ff @(posedge clk or posedge reset) begin
        if (reset)
            current_state <= STOPPED; // Start in STOPPED state
        else
            current_state <= next_state;
    end

    // Next state logic
    always_comb begin
        case (current_state)
            STOPPED: next_state = (start_stop) ? RUNNING : STOPPED; // Start if start_stop = 1
            RUNNING: next_state = (start_stop) ? STOPPED : RUNNING; // Stop if start_stop = 1
        endcase
    end

endmodule
```

2.4 Testbench

Listing 2: Stopwatch Controller

```

module tb_stopwatch_controller();
    logic clk, reset, start_stop;
    logic [7:0] count;

    stopwatch_controller uut (
        .clk(clk),
        .reset(reset),
        .start_stop(start_stop),
        .count(count)
    );

    // Clock generation
    initial begin
        clk = 0;
        forever #5 clk = ~clk; // 10ns clock period
    end

    // Test scenario
    initial begin
        reset = 1; start_stop = 0; // Assert reset initially
        #10 reset = 0;             // Deassert reset

        // Start stopwatch
        #10 start_stop = 1; #10 start_stop = 0; // Transition to RUNNING state
        #50; // Let the stopwatch run for 50 time units

        // Stop stopwatch
        #10 start_stop = 1; #10 start_stop = 0; // Transition to STOPPED state

        // Restart stopwatch
        #10 start_stop = 1; #10 start_stop = 0; // Transition to RUNNING state
        #30;

        // Reset stopwatch
        #10 reset = 1; #10 reset = 0;

        #50 $stop; // Stop simulation
    end

    // Monitor outputs
    initial begin
        $monitor("Time: %0t | Reset: %b | Start/Stop: %b | Count: %d | State: %s",
                 $time, reset, start_stop, count, uut.current_state.name());
    end
endmodule
```

3 Results

3.1 Simulation

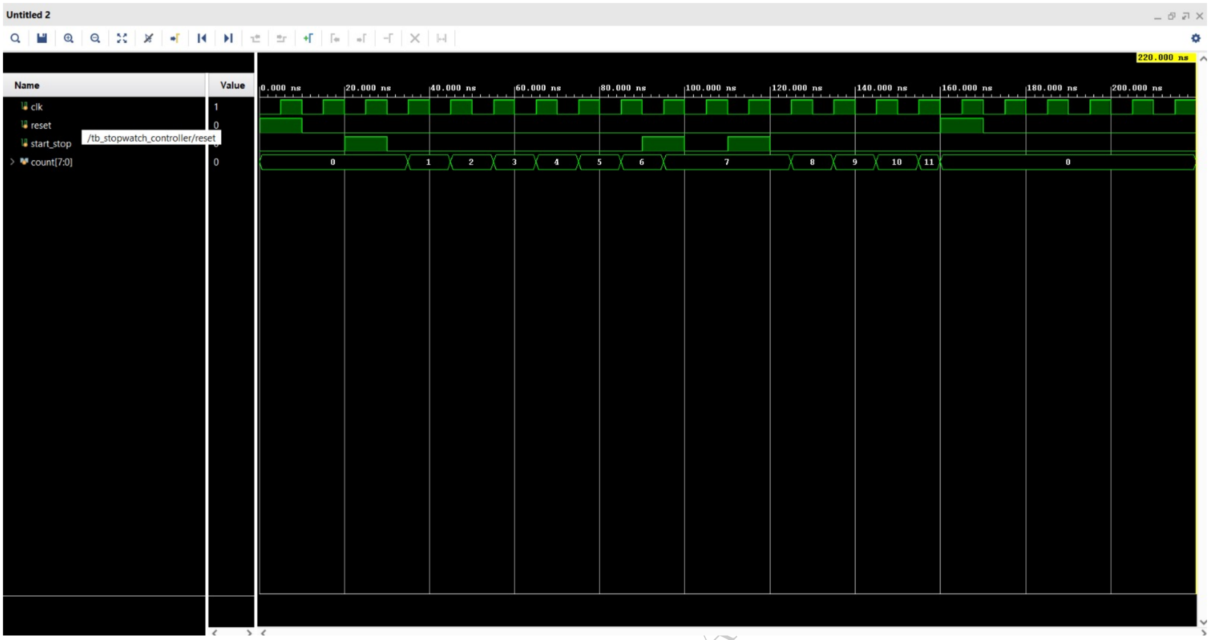

Figure 1: Simulation of Stopwatch Controller

3.2 Schematic

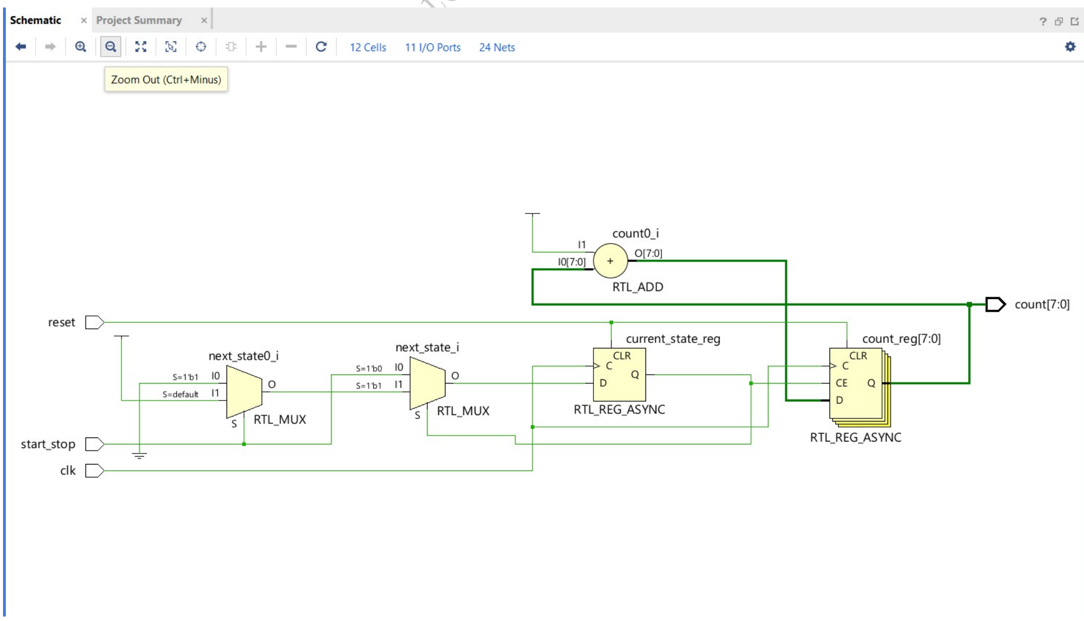

Figure 2: Schematic of Stopwatch Controller

3.3 Synthesis Design

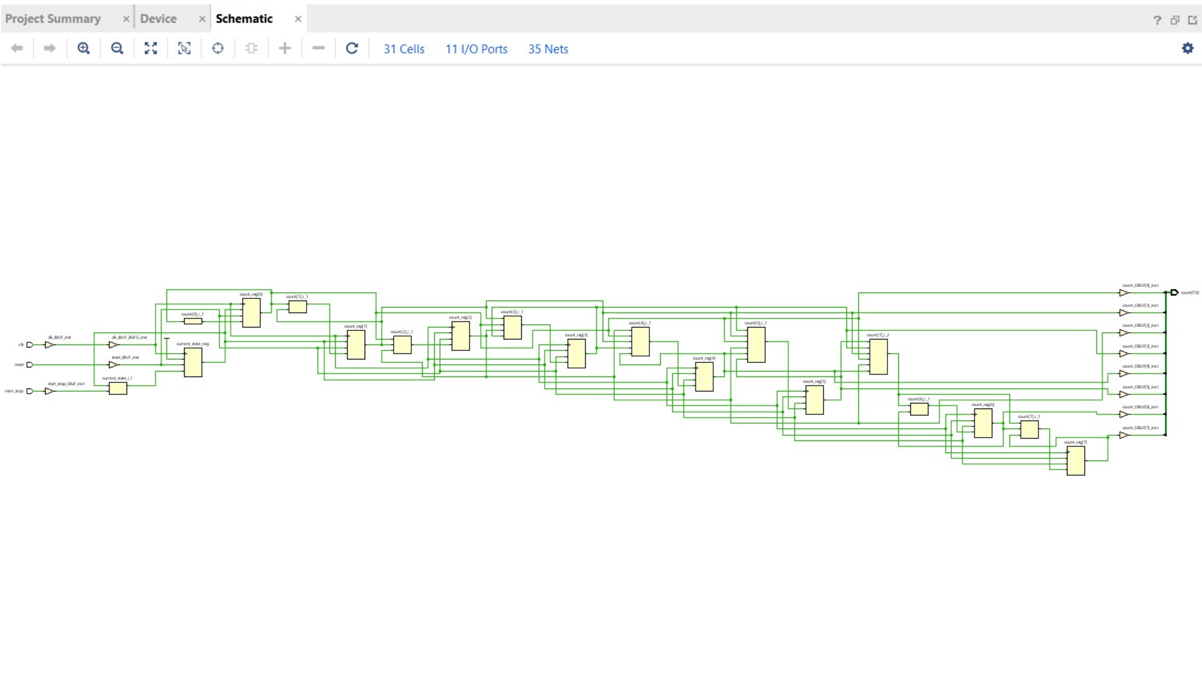

Figure 3: Synthesis Design of Stopwatch Controller

4 Advantages of Stopwatch Controller

- **Precise Timing:** Enables accurate measurement of time intervals.
- **Automation:** Simplifies timing operations, reducing manual intervention.
- **Compact Design:** Optimized for integration into various devices.
- **Customizable:** Can be programmed for specific time intervals and functionalities.
- **Low Power Consumption:** Efficient use of energy, especially in embedded systems.
- **Reliability:** Provides consistent performance in timing-critical applications.
- **Versatility:** Useful in diverse fields, including sports, testing, and industrial processes.

5 Disadvantages of Stopwatch Controller

- **Limited Functionality:** Designed primarily for timing, with few additional features.
- **Complex Programming:** May require expertise for customization.
- **Hardware Constraints:** Performance depends on the quality of components.
- **Power Dependency:** Requires a reliable power source to function accurately.
- **Maintenance:** Precision may degrade over time, needing recalibration.
- **Environmental Sensitivity:** Performance can be affected by extreme conditions like temperature or humidity.
- **Cost:** Advanced models can be expensive for simple applications.

6 Applications of Stopwatch Controller

- **Sports:** Timing races, games, and athletic events.
- **Education:** Monitoring test durations and lab experiments.
- **Healthcare:** Timing medical procedures and physical therapy sessions.
- **Manufacturing:** Tracking production cycles and machine operation times.
- **Research:** Measuring precise time intervals in experiments.
- **Automotive:** Testing vehicle performance and reaction times.
- **Everyday Use:** Time management in activities like cooking, workouts, or games.

7 Conclusion

In conclusion, a Stopwatch Controller is a versatile and essential tool across various fields due to its ability to measure time intervals accurately and reliably. While it has some limitations, such as hardware constraints and programming complexity, its advantages, including precision, automation, and ease of integration, make it invaluable in applications ranging from sports and education to healthcare and industrial processes. Its role in ensuring efficiency and consistency highlights its importance in modern technology and daily life.

8 FAQs

1. **What is a Stopwatch Controller?**
   - A Stopwatch Controller is a digital or electronic device designed to measure and control time intervals with precision. It is often used in various applications like sports, research, and manufacturing.
2. **How does a Stopwatch Controller work?**
   - It uses electronic circuits, counters, or microcontrollers to start, stop, and reset the timer. Inputs from buttons or sensors trigger these actions, and the time is displayed on a screen.
3. **What are the main features of a Stopwatch Controller?**
   - Start, stop, and reset functions
   - High accuracy in time measurement
   - Compact design
   - Optional advanced features like split timing and countdown
4. **What are the advantages of using a Stopwatch Controller?**
   - Precise time tracking
   - Reliable and easy to use
   - Automates time management tasks
   - Can be integrated into larger systems
5. **What are the common limitations of Stopwatch Controllers?**
   - Limited to time-based operations
   - Dependence on power supply and components
   - May require recalibration over time
6. **In which industries are Stopwatch Controllers commonly used?**
   - ***Sports:*** For timing races and games
   - ***Education:*** During exams or lab experiments
   - ***Healthcare:*** To time treatments or procedures
   - ***Manufacturing:*** To monitor production cycles
7. **How can Stopwatch Controllers be customized?**
   - Using microcontrollers like Arduino or Raspberry Pi, they can be programmed for specific applications, including automated triggers, alarms, and data logging.
8. **Are Stopwatch Controllers suitable for real-time applications?**
   - Yes, they are designed for real-time use, especially in scenarios where precise timing is critical.