Digital Stopwatch Controller

Overview

Design and implement a **Digital Stopwatch Controller** that measures elapsed time in **minutes:seconds (MM:SS)**. The project is a **hardware–software co-design**: the hardware (Verilog RTL) implements all timekeeping and control logic, while the software (C/C++ program) controls and observes the hardware model generated using **Verilator**.

1. Problem statement (Hardware)

Design a Digital Stopwatch Controller that measures elapsed time in seconds and minutes. The stopwatch must support start, stop, and reset operations and must pause and resume counting correctly.

Counter requirement

- The stopwatch shall use **synchronous up-counters** for seconds and minutes.
- Ripple counters are **not permitted**.

Functional requirements

The hardware shall implement the following behaviors:

1. The system shall provide `start`, `stop`, and `reset` control inputs. These inputs may be assumed to be clean, single-cycle synchronous signals.
2. When started, the stopwatch shall increment time in seconds (00–59) and roll over seconds to minutes on overflow.
3. Minutes shall support a minimum range of 00–99.
4. When stopped, the stopwatch shall retain the current time until restarted.
5. Reset shall clear the time to 00:00 and return the FSM to an idle state.
6. Counting shall occur only when explicitly enabled by the control FSM.
7. The current time (minutes and seconds) and the FSM state shall be observable via dedicated output ports on the top-level module.

2. Expected Verilog files (required)

Provide a modular Verilog implementation. At minimum, include the following `.v` files. The file names listed below are required for evaluation.

- `rtl/stopwatch_top.v` – Top-level module
- `rtl/seconds_counter.v` – Synchronous seconds counter (0–59)
- `rtl/minutes_counter.v` – Synchronous minutes counter (0–99 or parameterized)
- `rtl/control_fsm.v` – Control FSM (IDLE / RUNNING / PAUSED)

Module hierarchy and roles

The recommended module hierarchy is shown below. A similar modular separation shall be followed.

```
stopwatch_top
  |
  +-- control_fsm
  +-- seconds_counter
  +-- minutes_counter
```

Each module shall have a single, well-defined purpose and a clean interface. All RTL must be synthesizable. Simulation-only constructs such as `#delays` shall not be used in RTL. Non-blocking assignments (<=) shall be used in sequential `always` blocks.

Top-level conventions (required)

The top-level module `stopwatch_top` shall expose the following ports.

```
module stopwatch_top (
    input  wire        clk,
    input  wire        rst_n,       // active-low reset
    input  wire        start,
    input  wire        stop,
    input  wire        reset,
    output wire [7:0]  minutes,
    output wire [5:0]  seconds,
    output wire [1:0]  status       // 00=IDLE, 01=RUNNING, 10=PAUSED
);
endmodule
```

Recommended tools for writing and verifying `.v` files

- **ModelSim-Intel**
- **Icarus Verilog (iverilog)** and **GTKWave**

3. Software Co-Design

A software component shall be developed to interact with the stopwatch hardware using **Verilator**. Verilator shall be used to generate a cycle-accurate C++ model of the Verilog design. A separate C/C++ program shall then control and observe the hardware exclusively through the top-level interface.

All timekeeping and state transitions shall be implemented in hardware. The software shall only issue control commands and read observable outputs.

Software co-design requirements

- The same Verilog source files used for hardware verification shall be used without modification for Verilator-based simulation.
- Verilator shall be used to generate a C++ model of `stopwatch_top`.
- A C/C++ control program shall be compiled and linked with the Verilated model.
- The control program shall demonstrate reset, start, stop (pause), resume, and periodic readout of minutes and seconds.
- The observed stopwatch time shall be reported in a human-readable format (MM:SS).
- The structure of the control program is intentionally left open to allow flexibility and independent design decisions.

4. Files expected in the submission

Provide a single ZIP archive named `<studentnameid>_stopwatch.zip` with the following directory structure.

```
<studentnameid>_stopwatch.zip
|
+-- rtl/
|   +-- stopwatch_top.v
|   +-- seconds_counter.v
|   +-- minutes_counter.v
|   +-- control_fsm.v
|
+-- tb/
|   +-- tb_stopwatch.v
|
+-- verilator_sw/
|   +-- main.cpp
|   +-- Makefile
|
+-- README.md
```

- The `rtl/` directory shall contain all Verilog source files.
- The `tb/` directory shall contain the Verilog testbench.

- The `verilator_sw/` directory shall contain the C/C++ control program.
- `README.md` shall document tool versions, build commands, and design choices.

Important notes

- Top-level ports must match the specified names and widths.
- Document your design choices, and submit that document during final submission

5. Resources

For guidance on Verilator installation, Digital Design fundamentals, and Verilog, the following resources may be helpful:

- **Verilator Installation Guide**: `https://youtu.be/J64kzzC79yM?si=q0Ql0ghSAbjDojOx`
- **Introduction to Verilator**: `https://youtu.be/ALuaCai1W_0?si=j0KFD4VXeEfraU2u`
- **Digital Design Basics**: `https://youtu.be/MMntCKUbOKs?si=WXZUyIg0u6FV3ij8`
- **Verilog Basics**: `https://youtu.be/OC8F0js5nrc?si=diADR5xOlfDB6AyT`