DE PROJECT

Function

- Timekeeping
- Stopwatch
- Alarm
- Time Zone

To implement timekeeping for a digital watch in Xilinx (using Verilog or VHDL), we are creating a design that tracks hours, minutes, and seconds. This involves using a clock signal to drive counters for timekeeping. Below are the steps we are following to design a basic digital watch:

Steps to Design a Digital Watch

We are using binary inputs to manage different functionalities like timekeeping, stopwatch, alarm, and time zone. The binary signals are processed through various integrated circuits (ICs) to give accurate outputs on the seven-segment displays. The system is clock-driven, and each function (timekeeping, stopwatch, alarm) operates with precise timing, displaying results such as hours, minutes, and seconds using 7 segment displays.

1. Clock divider: Divide the system clock (e.g., 50 MHz) down to 1 Hz using a counter. This 1 Hz clock will increment the time.
2. Seconds counter: A 6-bit counter that increments every time the 1 Hz clock pulses. When it reaches 59, it resets to 0 and triggers the minute counter.
3. Minutes counter: A 6-bit counter that increments every time the seconds counter resets to 0. When it reaches 59, it resets to 0 and triggers the hours counter.
4. Hours counter: `A 5-bit counter that increments every time the minutes counter resets to 0. We are designing a 24-hour clock, it will reset to 0 after reaching 23.

F1) Time keeping:

To implement timekeeping for a digital watch in Xilinx using Verilog or VHDL, we need to handle time (hours, minutes, seconds) through a clock signal. Below are the typical inputs and outputs and the structure of the digital watch.

Inputs:

1. System Clock (clk**)**: This is a high-frequency clock signal (e.g., 50 MHz or 100 MHz) provided by the Xilinx board.It will be divided down to 1 Hz for timekeeping purposes.

2. Reset (reset): This is used to reset the time counters (hours, minutes, seconds) back to 0. It's typically triggered by a button to restart the watch or initialize the time.

3. Set Time Buttons: Additional inputs like buttons to increment hours or minutes manually.

4. Enable **:**An enable signal to start or stop the timekeeping (like a start/stop button).

Outputs:

1. Hours: Output the current hour (0-23 for 24-hour format or 1-12 for a 12-hour format).

2. Minutes: Output the current minute (0-59).

3. Seconds: Output the current second (0-59).

4. AM/PM Indicator (am_pm):An output indicating AM or PM for designing a 12-hour format watch.

5. Overflow/Error:In case of abnormal conditions (e.g., counters exceeding valid ranges), an overflow or error flag can be output.

F2) Stopwatch:

To implement a stopwatch as part of a digital watch in Xilinx (using either Verilog or VHDL in an FPGA), here's a general guide:

Inputs:

1.Clock (clk): A high-frequency clock signal used for timing purposes.

2. Reset (rst)**:**A signal to reset the stopwatch to zero.

3. Start/Stop:A signal to start or stop the stopwatch.

5. Lap**:**Optional input to capture the current time without stopping the stopwatch.

6. Enable:A signal to allow or disable counting when high.

7. Set Time:If setting the time is required manually.

Outputs:

1. Elapsed Time (seconds, minutes, hours): The current count of the stopwatch in binary or decimal format, typically divided into seconds, minutes, and hours.

2. Lap Time If lap functionality is implemented, this output stores the time when the lap button is pressed.

3. Overflow Flag **:** If the stopwatch reaches its maximum count, this flag is set.

F3) Alarm Functionality:

The alarm functionality in a digital watch involves setting a specific time at which the alarm will trigger and producing an output (e.g., an alarm signal) when the current time matches the set alarm time. In a hardware description language (HDL) like Verilog or VHDL, the alarm logic compares the current time with the preset alarm time and raises a signal when they match.

1.Components of Alarm Functionality

The alarm function needs the following components:

1. Alarm Time Setting: Inputs to set the desired alarm time (hours, minutes, seconds).
2. Current Time: The time the digital watch keeps, provided by the timekeeping module.
3. Alarm Enable: A way to turn the alarm ON or OFF.
4. Comparison Logic: Logic to compare the current time with the set alarm time.

5. Alarm Output: An output that is triggered when the current time matches the alarm time (e.g., an alarm signal).

Alarm Set Inputs:

1. set_alarm:This input activates the alarm setting mode, allowing the user to adjust the alarm time.
2. alarm_hours: An input to set the alarm hours (5-bit input for 24-hour format, 0-23).
3. alarm_minutes: An input to set the alarm minutes (6-bit input for 0-59).
4. alarm_seconds: An input to set the alarm seconds (6-bit input for 0-59). If your alarm is more precise and allows for seconds, include this. If not, it can be omitted.

Current Time Inputs:

1. current_hours: The current hours value from the timekeeping module.
2. current_minutes: The current minutes value from the timekeeping module.
3. current_seconds: The current seconds value from the timekeeping module.

Alarm Enable Input:

enable_alarm: A signal that enables or disables the alarm. If enable_alarm is high (1), the alarm will be active and will trigger when the current time matches the alarm time.

Reset input:

reset_alarm: An optional input to reset the alarm time to 00:00:00.

Outputs for Alarm Functionality

The main output of the alarm module is the alarm signal, which indicates whether the alarm is triggered or not.

1. Alarm Signal Output:

alarm_signal: A 1-bit output that goes high (1) when the alarm is triggered, i.e., when the current time matches the set alarm time. This signal could be used to control an external component like an LED, buzzer, or any alert mechanism.

F4) Time Zone Feature:

1. **Inputs:**
2. **Time Zone Selection (Binary Input):**
    a. **Binary input** (4-bit) will represent different time zones.
    b. Each binary value will correspond to a specific time zone offset (e.g., UTC+5, UTC-3, etc.).
    c. For example:
        i. `0000` = UTC+0 (Universal Coordinated Time)
        ii. `0001` = UTC+1
        iii. `1111` = UTC-1 (or similar).

3. **Current Time (hours and minutes):**
    a. Input from the internal clock that keeps the base time (can be set to UTC or local time).

Outputs:

1. **Adjusted Time (hours and minutes):**
    - The clock will output the **time adjusted** according to the selected time zone.
    - This adjusted time will be displayed on the **7-segment display**.

Implementation:

Binary Input for Time Zone

1.User Selecting Time Zone through binary inputs .

2.for example If User Selects binary value 0101 means 5 (Which Could Correspond the UTC+5) means clock add 5 hours in the current time .

3.If Binary Value is 1101 (correspond to UTC-3)means clock subtracts 3 hours in the current time .

Time Zone Adjusting Logic:

1. The internal clock generates the current time (let's assume it's in UTC).
2. The **binary input** adjusts this time by either adding or subtracting the correct number of hours.
3. This adjustment happens by modifying the **hours** value in the clock's register.
4. For instance:
    a. Current time: 12:00 (UTC)
    b. Binary input: `0101` (UTC+5)
    c. Adjusted time: 17:00 (12 + 5 = 17).
    d. This value is then displayed.

Output to 7-Segment Display:

1. The adjusted hours and minutes are converted from binary to **7-segment display format** using a decoder like the **BCD to 7-segment decoder** (e.g., **74LS47**).
2. Each digit of the hours and minutes is displayed on the corresponding **7-segment display** for the user to see the time in the selected time zone.