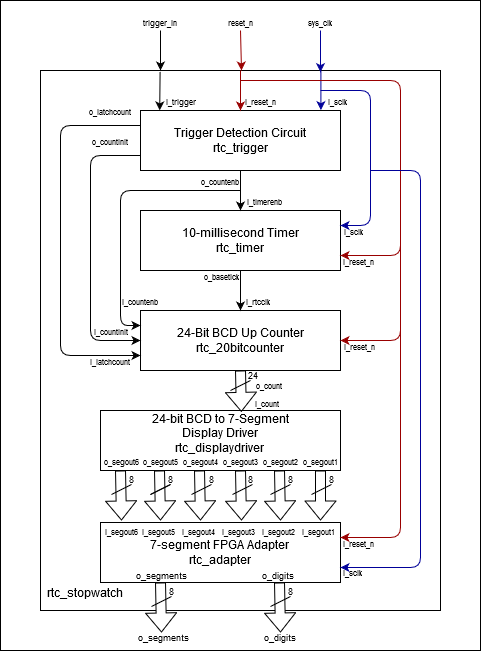
**System Specification Rev 1**

**General Specifications:**

This design is based on the RTC Stopwatch Design specification document with additional functionality as described here.



Stopwatch will use 7-segment displays to show time down to tens of milliseconds (MM:ss:mm). The stopwatch is started/resumed and stopped via a push-button input. It can be reset via another push button input. The stopwatch can count to 59m 59s 99ms before rolling back to 00m 00s 00ms.

Five modules will be used in a top-level wrapper to achieve this functionality. A trigger detection circuit will enable a clock divider, which in turn will generate clock signal for the counter module. The display driver then outputs the counter output values onto 7-segment display through FPGA Display adapter.

* System clock of 100Mhz
* Active-low reset

*Inputs:*

1. ‘trigger\_in’ (referred to as i\_trigger as per coding standards)
2. 'reset\_n’ (referred to as i\_reset\_n as per coding standards)
3. ‘sys\_clk’ (referred to as i\_sclk as per coding standards)

*Outputs:*

1. ‘o\_segments’ (vector [7:0])
2. ‘o\_digits’ (vector [7:0])

**Trigger Detection Circuit:**

Enables the timer and counter modules and sends a latch signal to stop/resume counting based on user input, it also takes care of de-bouncing the input signal from the push-button.

If reset is not ‘0’, one push of the push-button would start the count. If the count is active pushing the button would pause it, and if the count is paused pushing the button would resume it.

*Inputs:*

1. ‘i\_trigger’
2. ‘i\_reset\_n’
3. ‘i\_sclk’

*Outputs:*

1. ‘o\_latchcount’
2. ‘o\_countinit’
3. ‘o\_countenb’

**10-millisecond Timer:**

Clock divider takes 100Mhz system clock input and ‘o\_base\_tick' should toggled every 5 milliseconds after counting a maximum count sequence when enabled. Clock divider resets internal counter to its initial value and outputs ‘0’ when reset signal is low.

*Inputs:*

1. ‘I\_timerenb’
2. ‘i\_sclk’
3. ‘i\_reset\_n’

*Outputs:*

1. ‘o\_basetick’

**24-Bit BCD Up Counter:**

An array of generic 4-bit counter components will be parameterized to accept an enable signal and roll-over value. A roll-over flag will be pulsed for one clock cycle every time the counter is reset to its initial value.

One of these components will be instantiated per displayed digit, with the rollover flag of the less significant digits serving as the enable signal for the more significant digit above it. The milliseconds, seconds, and minutes digits will roll-over at a value of 9. The tens of seconds and tens of minutes digits will roll-over at a value of 6 to correctly count 60 seconds per minute.

*Inputs:*

1. ‘i\_rtcclk’
2. ‘i\_reset\_n’
3. ‘i\_latchcount’
4. ‘i\_countinit’
5. ‘i\_countenb’

*Outputs:*

1. ‘o\_count’ (vector [23:0])

**24-bit BCD to 7-Segment Display Driver:**

Translates the input BCD number to be displayed on six 7-segment displays. Each unit gets a 4-bit input from the counter representing the digit to be displayed. On reset, all units should display 0.

*Inputs:*

1. ‘i\_count’ (vector [23:0])

*Outputs:*

1. ‘o\_segout1’ (vector [7:0])
2. ‘o\_segout2’ (vector [7:0])
3. ‘o\_segout3’ (vector [7:0])
4. ‘o\_segout4’ (vector [7:0])
5. ‘o\_segout5’ (vector [7:0])
6. ‘o\_segout6’ (vector [7:0])

**7-segment FPGA Adapter:**

Converts the six o\_segout signals generated by the display driver for use in Digilent 7-segment arrays. The digit period of each seven-segment display change shall be 1 millisecond, with total refresh period of 6 milliseconds. Upon reset, all eight digits present on the FPGA board should have all of their segments lit, regardless of the segout inputs, to indicate said reset to the user.

*Inputs:*

1. ‘i\_sclk’
2. ‘i\_reset\_n’
3. ‘i\_segout1’ (vector [7:0])
4. ‘i\_segout2’ (vector [7:0])
5. ‘i\_segout3’ (vector [7:0])
6. ‘i\_segout4’ (vector [7:0])
7. ‘i\_segout5’ (vector [7:0])
8. ‘i\_segout6’ (vector [7:0])

*Outputs:*

1. ‘o\_segments’ (vector [7:0])
2. ‘o\_digits’ (vector [7:0])