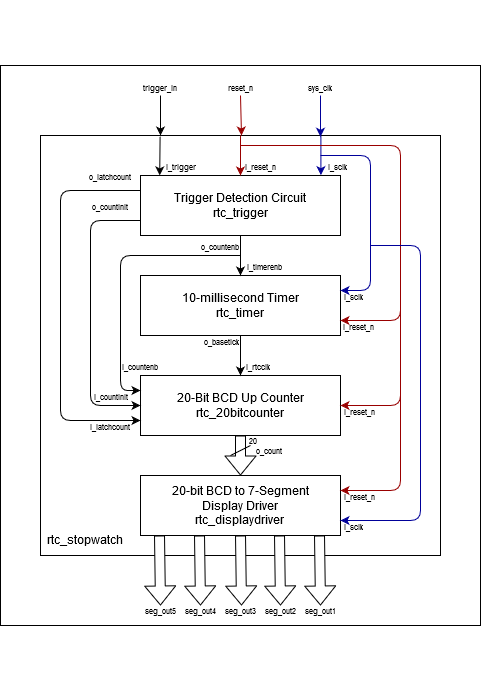
**General Specifications:**

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



Stopwatch will use 7 segment displays to show time down to tens of milliseconds (M: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 9m 59s 99ms before rolling back to 0m 00s 00ms.

4 modules will be used in a top-level wrapper to achieve this functionality. A trigger detection controller will enable a clock divider, which in turn will drive the counter module driving the display controller.

* System clock of 100Mhz
* Active-low reset

*Inputs:*

1. ‘trigger\_in’
2. 'reset\_n’
3. ‘sys\_clk’

*Outputs:*

1. ‘seg\_out1’ (vector [7:0])
2. ‘seg\_out2’ (vector [7:0])
3. ‘seg\_out3’ (vector [7:0])
4. ‘seg\_out4’ (vector [7:0])
5. ‘seg\_out5’ (vector [7:0])

**Trigger Detection:**

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’

**10ms Timer:**

Clock divider takes 100Mhz system clock input and ‘o\_base\_tick' should toggled every 5 millisecond 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\_reser\_n’

*Outputs:*

1. ‘o\_basetick’

**20-Bit Counter:**

A generic 4-bit counter component 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 digit will roll-over at a value of 6 to correctly count 60 seconds per minute.

*Inputs:*

1. ‘i\_rtcclk’
2. ‘i\_reset\_n’

*Outputs:*

1. ‘o\_count’ (vector [19:0])

**7-Segments Display Controller:**

Instantiates 5 7-segment display units, and alternates between the units one fifth of the time for each using an internal counter. 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\_sys\_clk’
2. ‘i\_reset\_n’
3. ‘i\_count’ (vector [19:0])

*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])