Build a module which controls a stopwatch timer. The timer starts counting when the start button (start) is pressed (pulses) and increases by 1 every clock cycle. When the stop button (stop) is pressed, the timer stops counting. When the reset button (reset) is pressed, the count resets to 0 and the timer stops counting. If count ever reaches MAX, then it restarts from 0 on the next cycle. stop's functionality takes priority over start's functionality, and reset's functionality takes priority over both stop and start's functionality.

Input and Output Signals

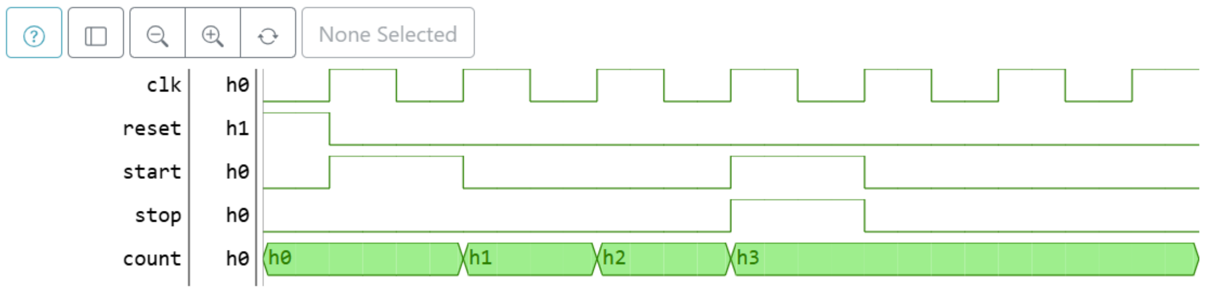
- clk - Clock signal
- reset - Synchronous reset signal
- start - Start signal
- stop - Stop signal
- count - Current count

Output signals during reset

- count - 0 when reset is active

We begin counting up from `1` when the `start` signal is pulsed. Once the `stop` signal is pulsed, we stop counting.

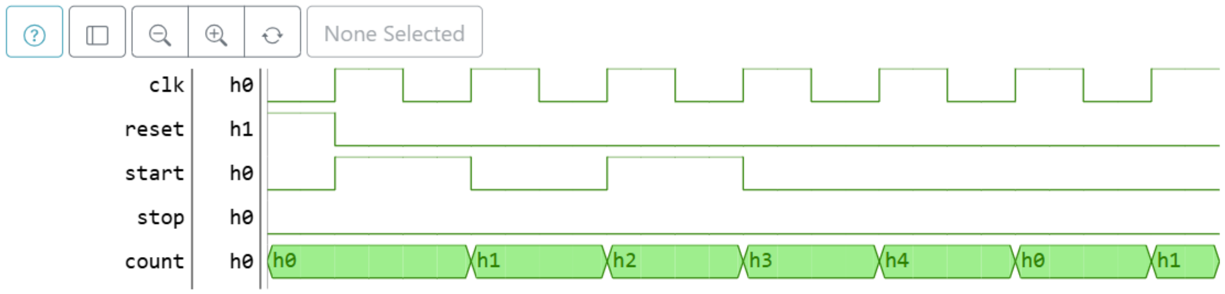
Note that even though `start` is pulsed at the same time, `stop`'s functionality takes priority over `start`'s functionality.



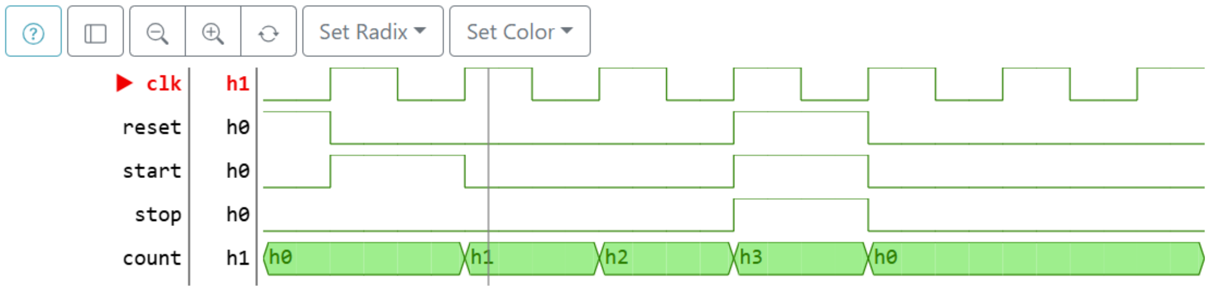
Assume `MAX = 4` in this example.

We begin counting up from `1` when the `start` signal is pulsed. Once the `count` reaches the `MAX` of `4`, `count` wraps back around to `0` and continues counting.

Note that pulsing `start` when we are already counting has no effect.



An active `reset` resets the counter to `0` despite the lower priority signals (`start` and `stop`) also pulsing at the same time.


Set Radix
Set Color
clk h1
reset h0
start h0
stop h0
count h1
h0
h1
h2
h3
h0