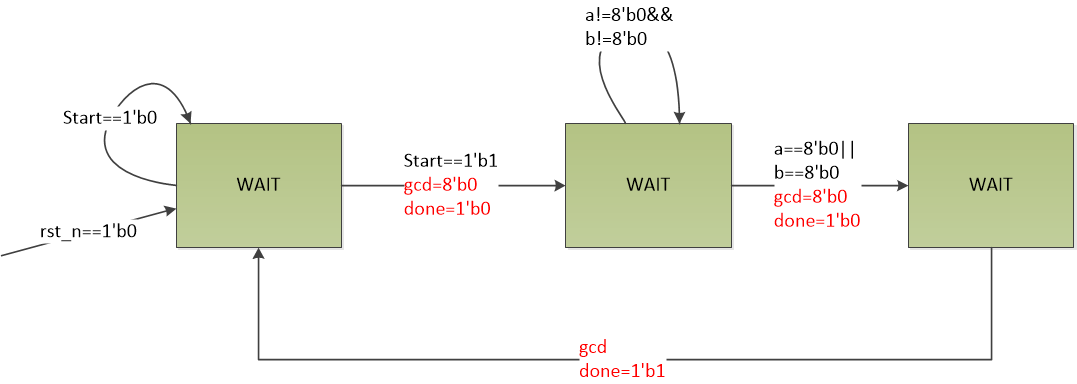
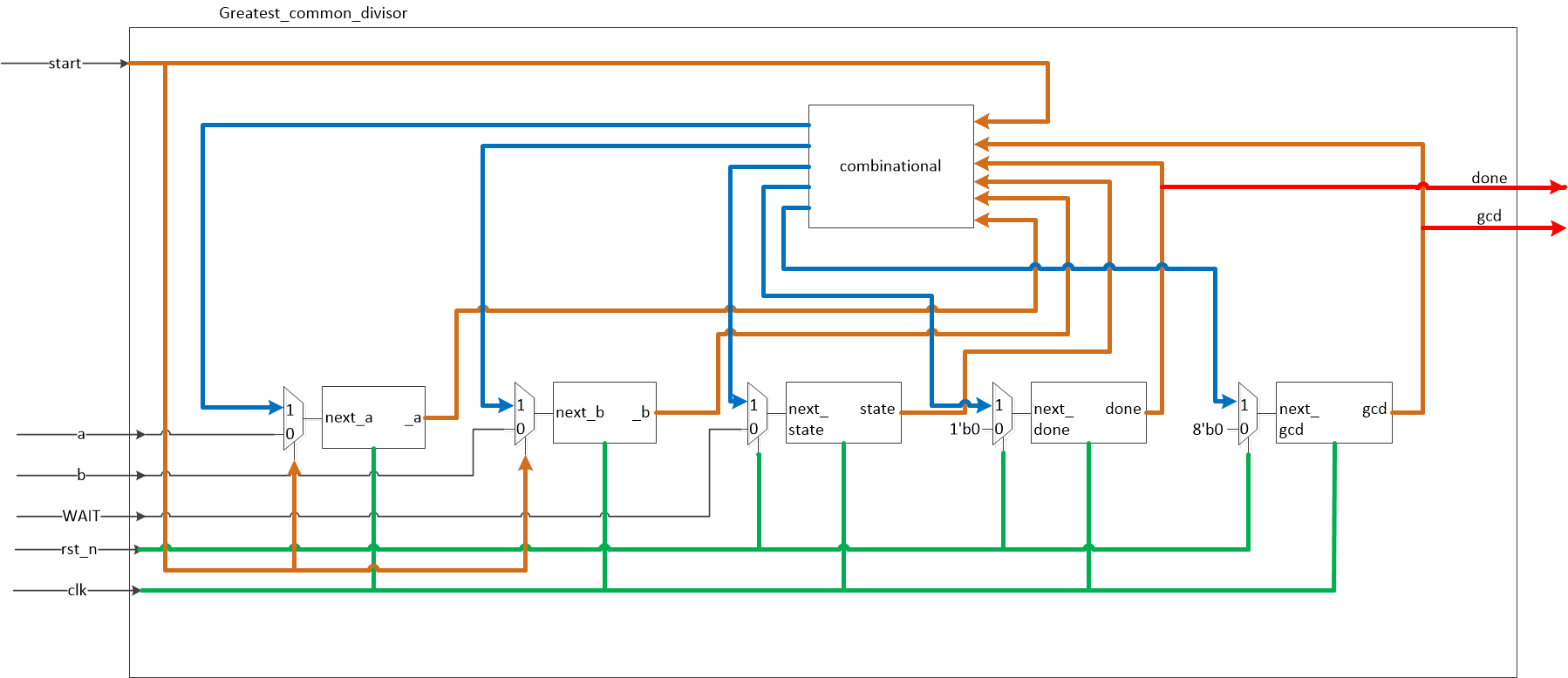
Greatest common divisor

* State transition



* Block diagram



在state transition diagram中。首先由rst=0使系統進入WAIT state，因此rst\_n也是初始所有的DFF的信號。在rst=0時，會初始state為WAIT、done=0、gcd=0。若start等於0就會一直在WAIT state中，當start=1，會由combinational電路馬上讀進a、b，存入next\_a、next\_b。指定next\_state=CAL state，此時的gcd和done都是output=0。在CAL state中，若a!=0&&b!=0，就會一直在CAL state中互相相減。直到a或b等於0，就會將不等於零的一方設為next\_gcd，並且next\_done=1，next\_state=FINISH。會需要next\_gcd和next\_done的原因在於，由於spec的CAL state到FINISH state的輸出皆為0，而在FINISH state才output gcd與done，因此要放在DFF中。最後將next state設為WAIT state。

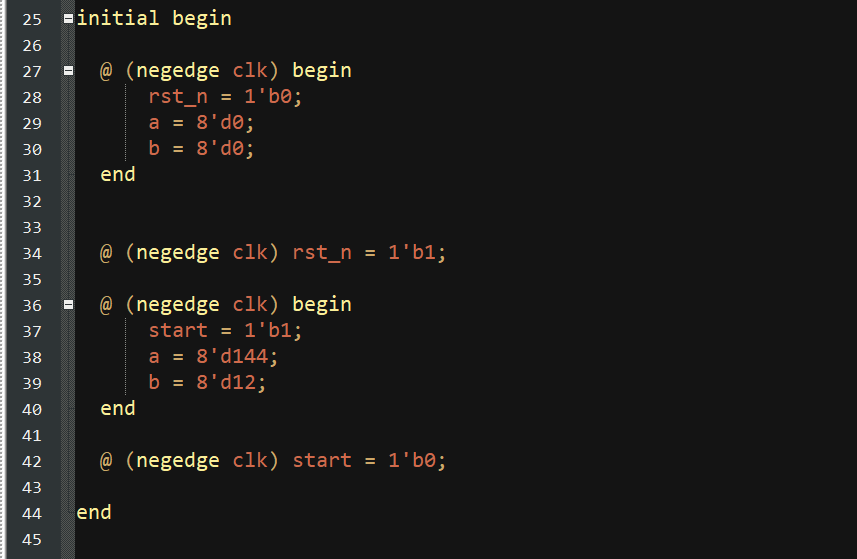
有想過要不要打default。照理說，宣告兩個bit的state，卻只有三個state應該要打default。不過，我在default中，將next\_state=WAIT會使的就算還沒reset\_n，一開始state就會是WAIT，這樣會跟spec不符，所以還是刪去了。

測試的方法為在negedge clk給出rst的初始信號，將state固定在WAIT，給出a與b，此時的a與b不是真正的a與b，因為start!=1，只是為了測試在start=1時能不能拿到對的a與b。

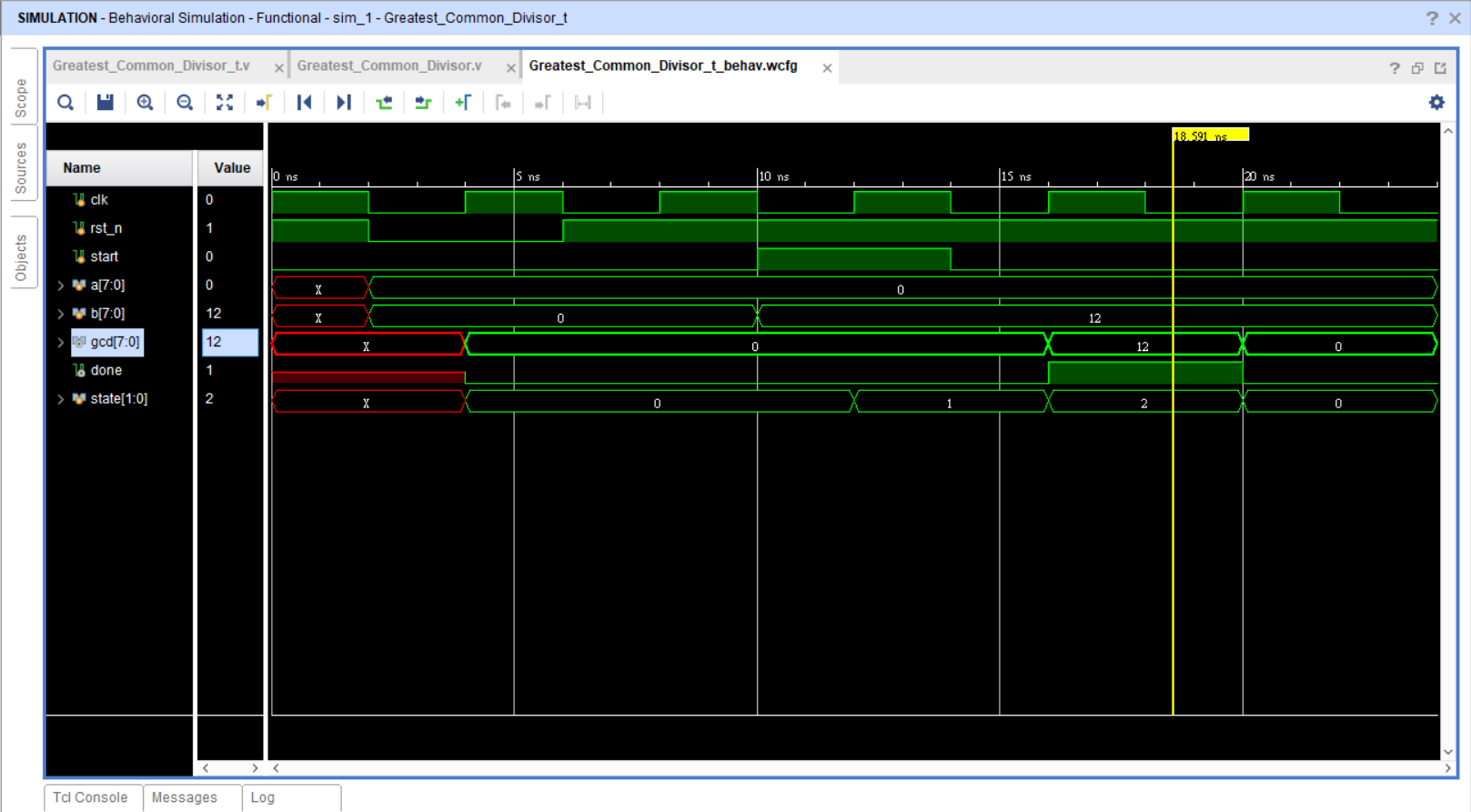
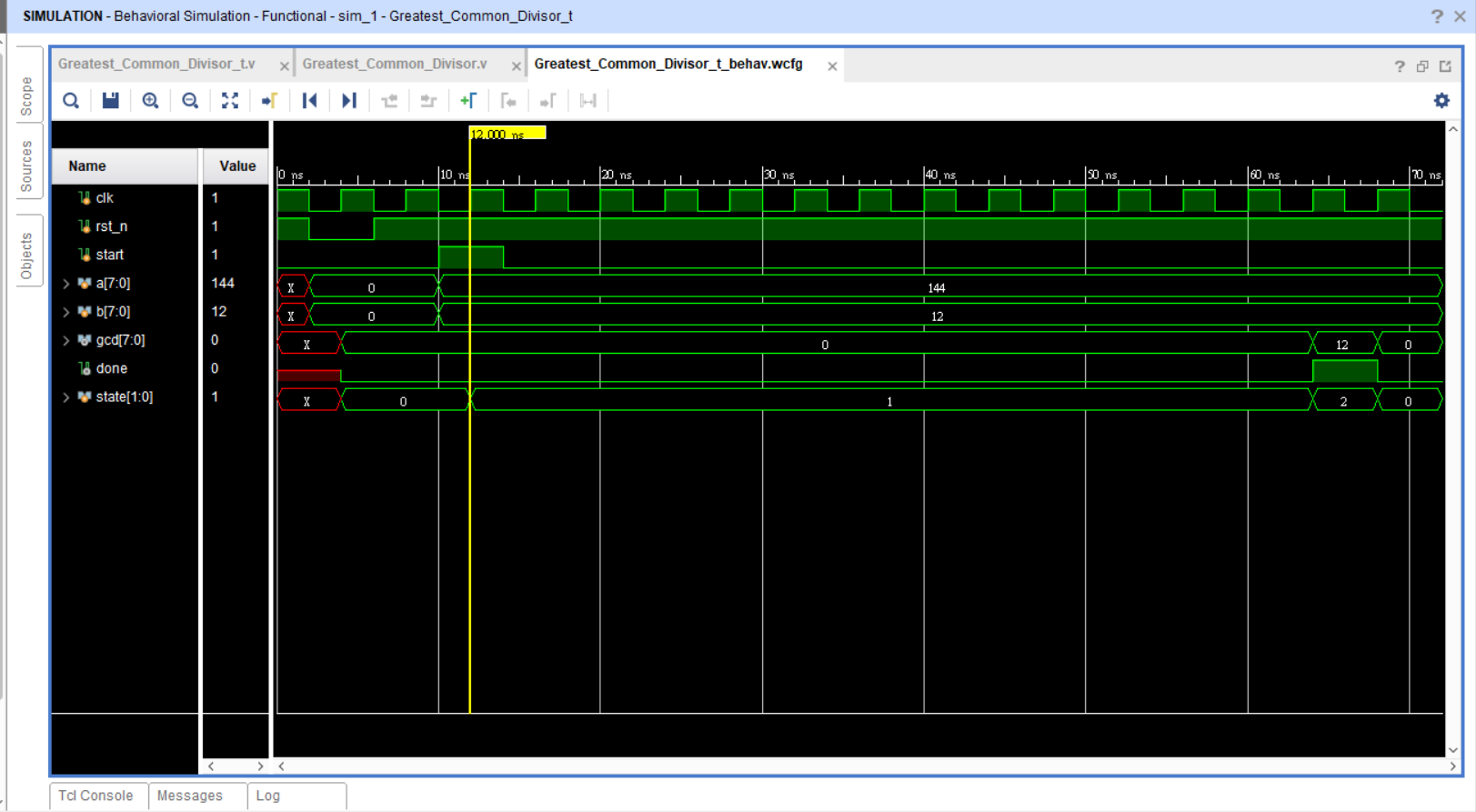
過一個cycle的negedge clk，將rst=1。再過一個cycle的negedge clk，讓start=1，使state能進入CAL，此時給出將要計算的a、b。

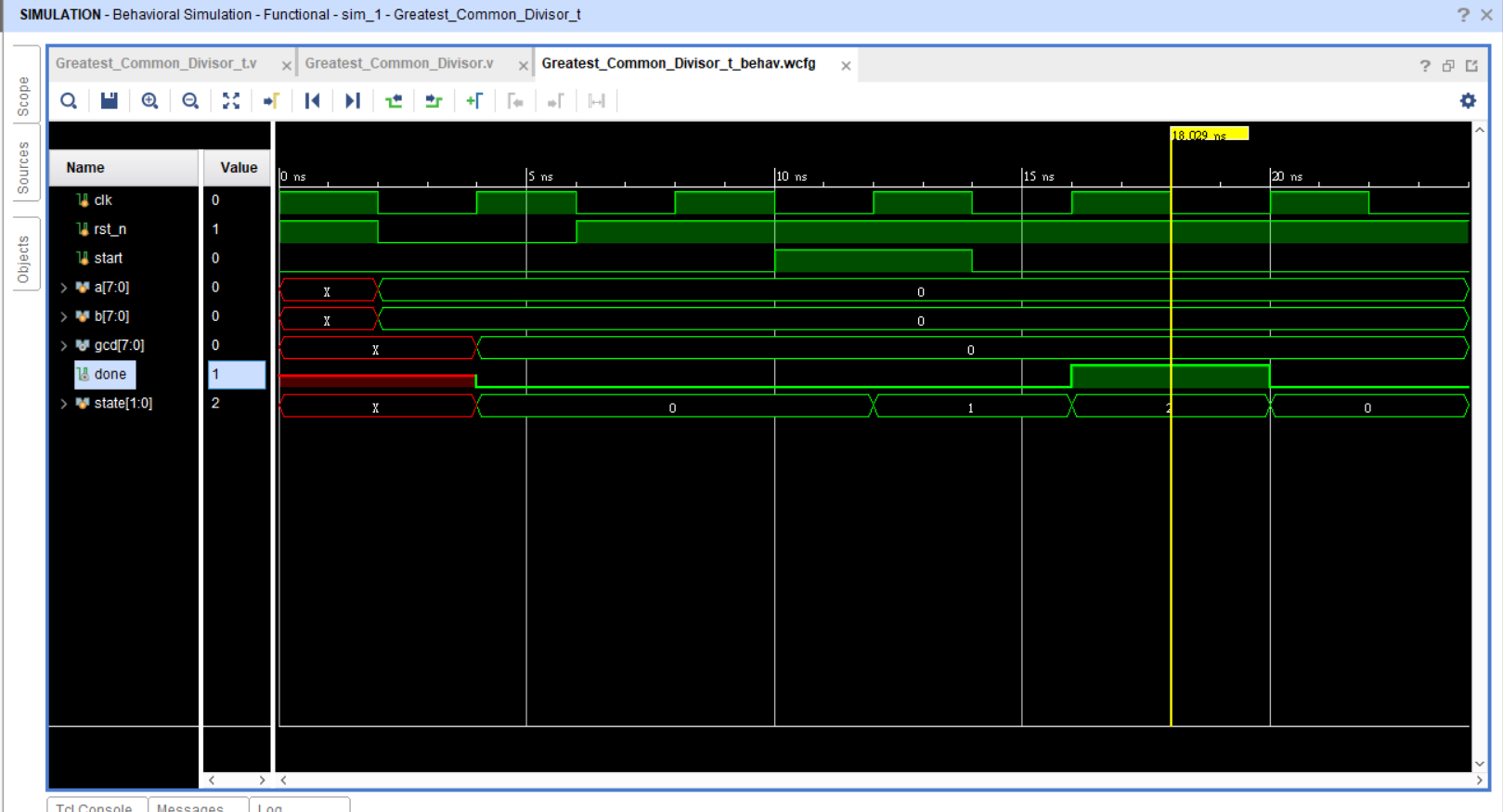
再過一個cycle讓start=0。最後，因為結果易於檢查，僅用人工檢查結果。

如圖：



Waveform





* 心得

這次的lab感覺比較輕鬆。比較有釐清上問題的是，我本來搞不太清楚reset跟start的差異，後來想清楚應該是，reset是初始的信號，而start是把state固定在WAIT或是CAL開始計算的信號。這樣也比較符合reset與start的字詞意義。Reset是在系統發生奇怪錯誤時，可以按下，讓系統不管是現在在何state，都能回到最初的狀態，也是WAIT，而start是一種從IDLE(WAIT)開始的信號。