**0LAB4\_1**

**Construct a 4-bit synchronous binary up counter (b3b2b1b0) with the 1-Hz clock frequency from lab2 and use 4 LEDs for display.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **I/O** | **fcrystal** | **b3** | **b2** | **b1** | **b0** |
| **Site** | **W5** | **V19** | **U19** | **E19** | **U16** |

specification:  
output:

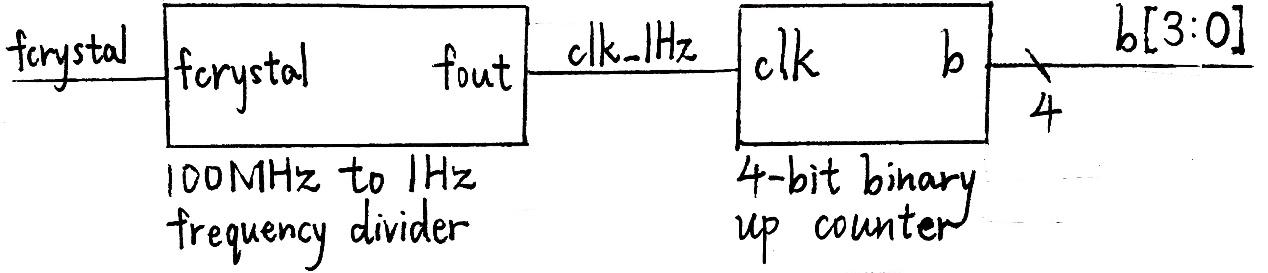
b[3:0] // binary value

input:

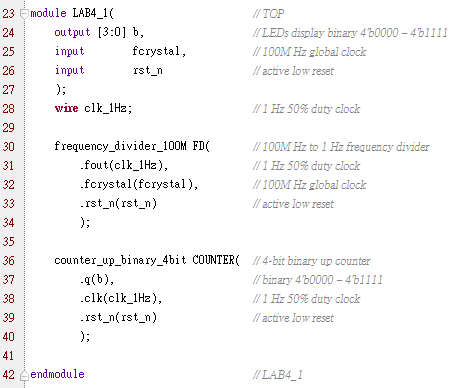
fcrystal // 100M Hz global clock

rst\_n // active low reset

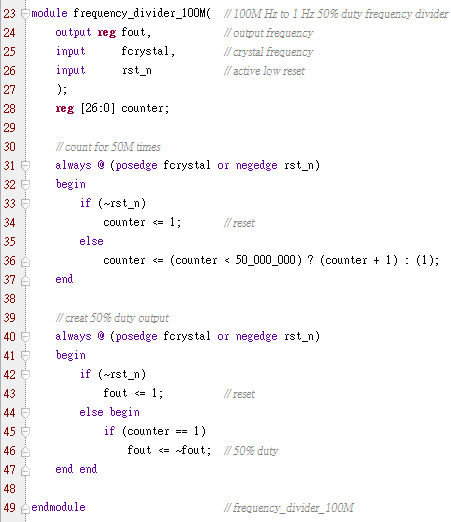
circuit diagram:



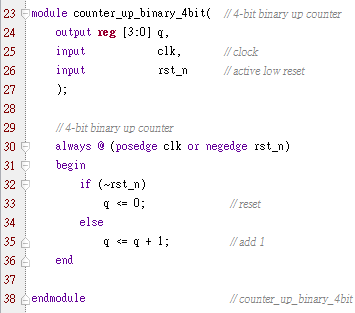
top: 把除頻器的一赫茲輸出連接到計數器作為時脈



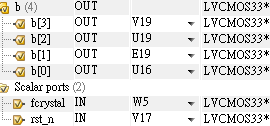
100M Hz to 1 Hz 50% duty frequency divider:

先數到50M，再連接到一個TFF即可

4-bit binary up counter:



implement:



conclusion:

這一題要做的是一個以一赫茲為時脈的四位元二進位上數計數器。解題思路如下：先拿一個計數器當作除頻器，利用FPGA內建的100M赫茲時脈產生一赫茲的時脈。再把該一赫茲時脈連接到一個四位元上數計數器。因為100M赫茲轉一赫茲的除頻器與四位元上數計數器在上次的LAB3就已經做過了，規格一模一樣，所以直接拿來用即可。

**LAB4\_2**

**Combine the 4-bit synchronous binary up counter from exp1 with a binary-to-seven-segment-display decoder (from lab2-exp3) to display the binary counting in 7-segment display.**

specification:

output:

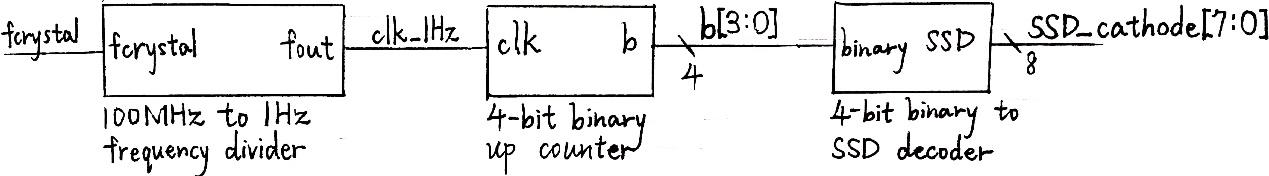
SSD\_cathode[7:0] // individual cathodes of seven segment display

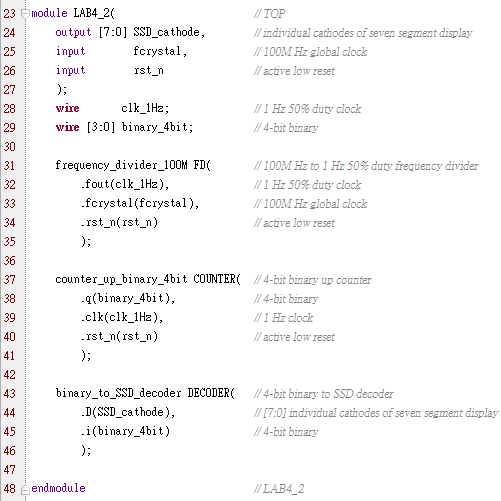
input:

fcrystal // 100M Hz global clock

rst\_n // active low reset

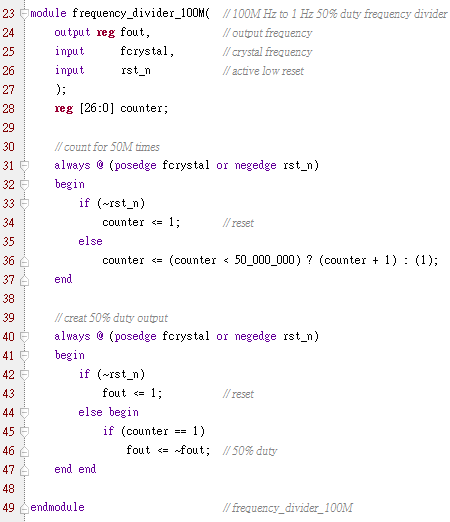
circuit diagram:

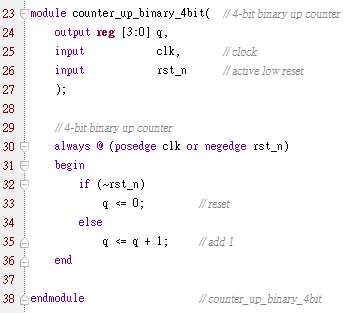
 top: 把除頻器產生的一赫茲時脈連接到四位元計數器，再接一個解碼器。



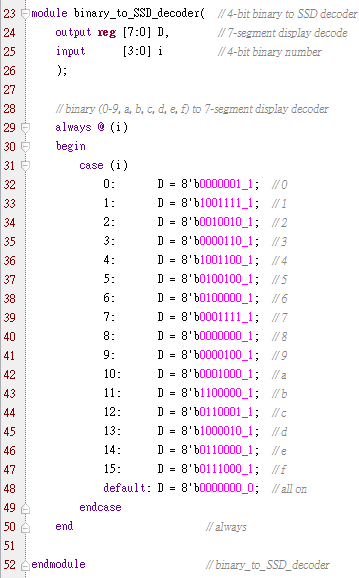
100M Hz to 1 Hz 50% duty frequency divider:

數到50M，再接一個TFF。

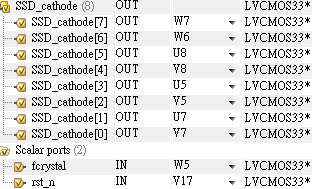


4-bit binary up counter:  


4-bit binary to SSD decoder:



implement:



conclusion:

這一題跟上一題一樣要做一個以一赫茲為時脈的四位元二進位上數計數器，但這題要求使用七段顯示器來監控計數器的數值。這題也很單純，只要將上一題的四位元計數器的輸出連接到一個二進位轉七段顯示器的解碼器即可。

**LAB4\_3**

**Construct a single digit BCD up counter with the divided clock as the clock frequency and display on the seven-segment display.**

**3.1 Construct a BCD up counter.**

**3.2 Construct a BCD-to-seven-segment display decoder (from lab2-exp2).**

**3.3 Combine the above two together.**

specification:

output:

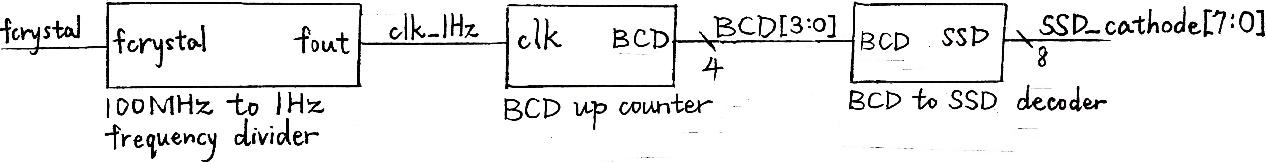
SSD\_cathode[7:0] // individual cathodes of seven segment display

input:

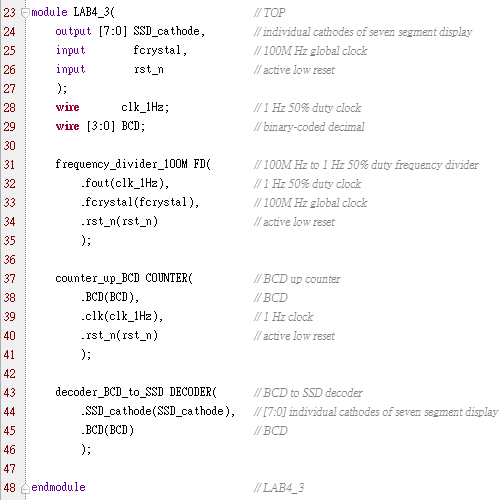
fcrystal // 100M Hz global clock

rst\_n // active low reset

circuit diagram:

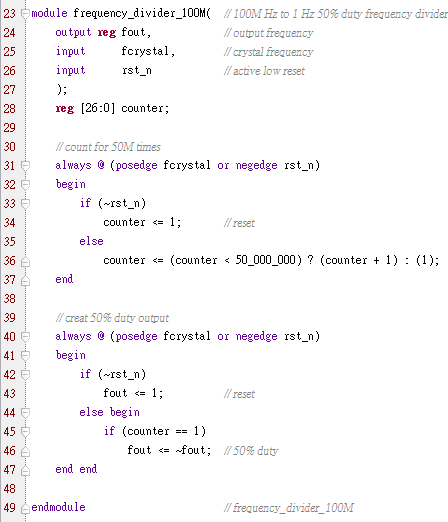


top: 將除頻器產生的一赫茲時脈連接到BCD上數計數器，再連接解碼器。

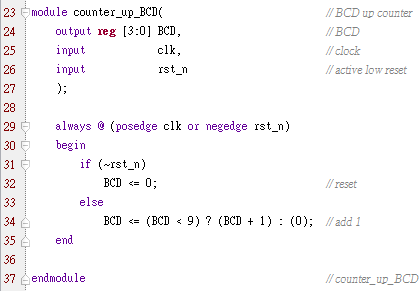


100M Hz to 1 Hz 50% duty frequency divider:

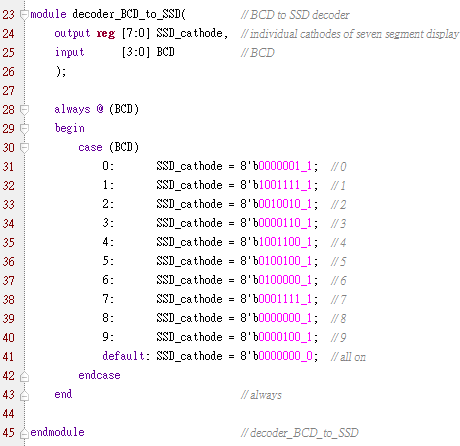
數到50M，然後連接到一個TFF。



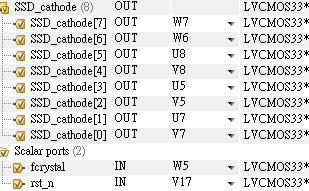
BCD up counter:



BCD to SSD decoder:



implement:



conclusion:

這一題跟上一題很像，要做的都是上數計數器，但是這題是BCD。也就是說，數到九之後的下一個時脈不是變成十，而是變成零，如counter\_up\_BCD的第34行所示。此外，因為計數器的數值不會大於九，因此不需要如上一題一樣把四位元二進位的所有可能情況都寫進解碼器，只要保留零到九就好，如decoder\_BCD\_to\_SSD的第31到41行所示。

**LAB4\_4**

**Construct a single digit BCD down counter with the divided clock as the clock frequency and display on the seven-segment display.**

**4.1 Construct a BCD up counter.**

**4.2 Construct a BCD-to-seven-segment display decoder (from lab2-exp2).**

**4.3 Combine the above two together.**

specification:

output:

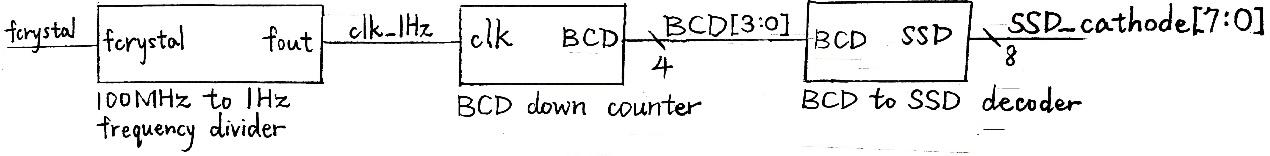
SSD\_cathode[7:0] // individual cathodes of seven segment display

input:

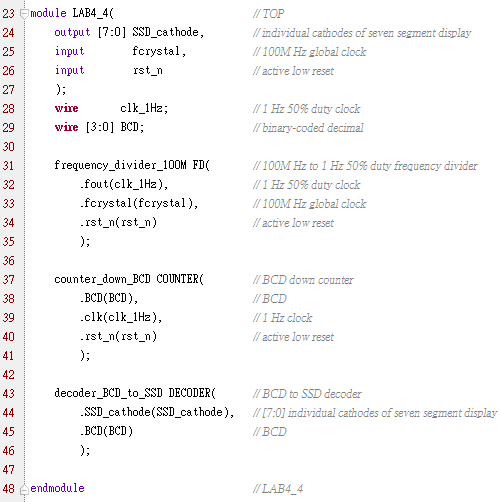
fcrystal // 100M Hz global clock

rst\_n // active low reset

circuit diagram:

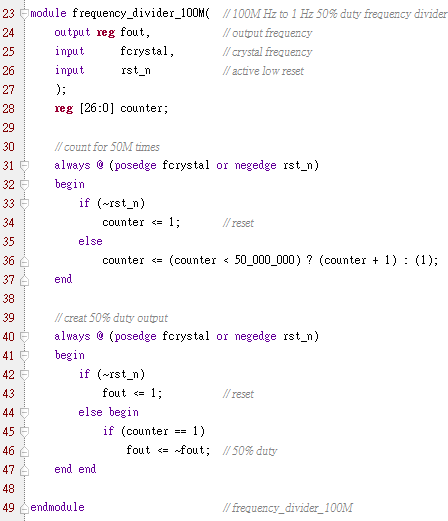


top: 將除頻器所產生的一赫茲時脈連接到BCD下數計數器，然後再接解碼器

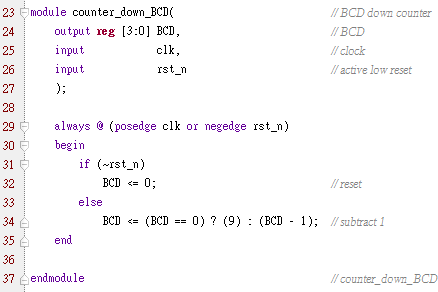


100M Hz to 1 Hz 50% duty frequency divider:

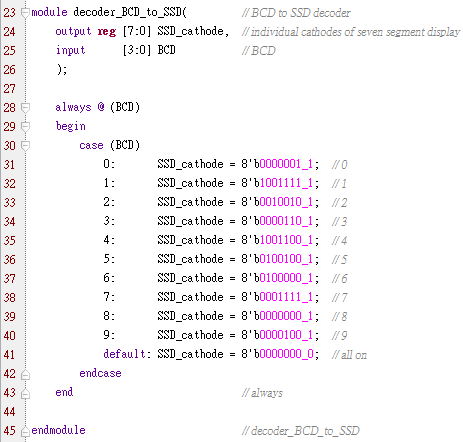
先數到50M，再連接到一個TFF。



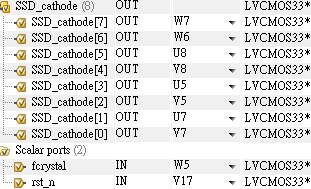
BCD down counter:



BCD to SSD decoder:



implement:



conslusion:

這一題幾乎跟上一題一模一樣，唯一的差別在於上一題是上數，這一題是下數。上一題是不斷加一直到數值為九，再過一時脈變為零，而這一題則是不斷減一直到數值為零，再過一時脈變為九，如counter\_down\_BCD的第34行所示。

**LAB4\_5**

**(Bonus) Construct a 30-second count down timer (stop at 00).**

specification:

output:

cathode[7:0] // cathodes of similar segments on all four displays

anode[3:0] // anodes of the seven LEDs forming each digit

input:

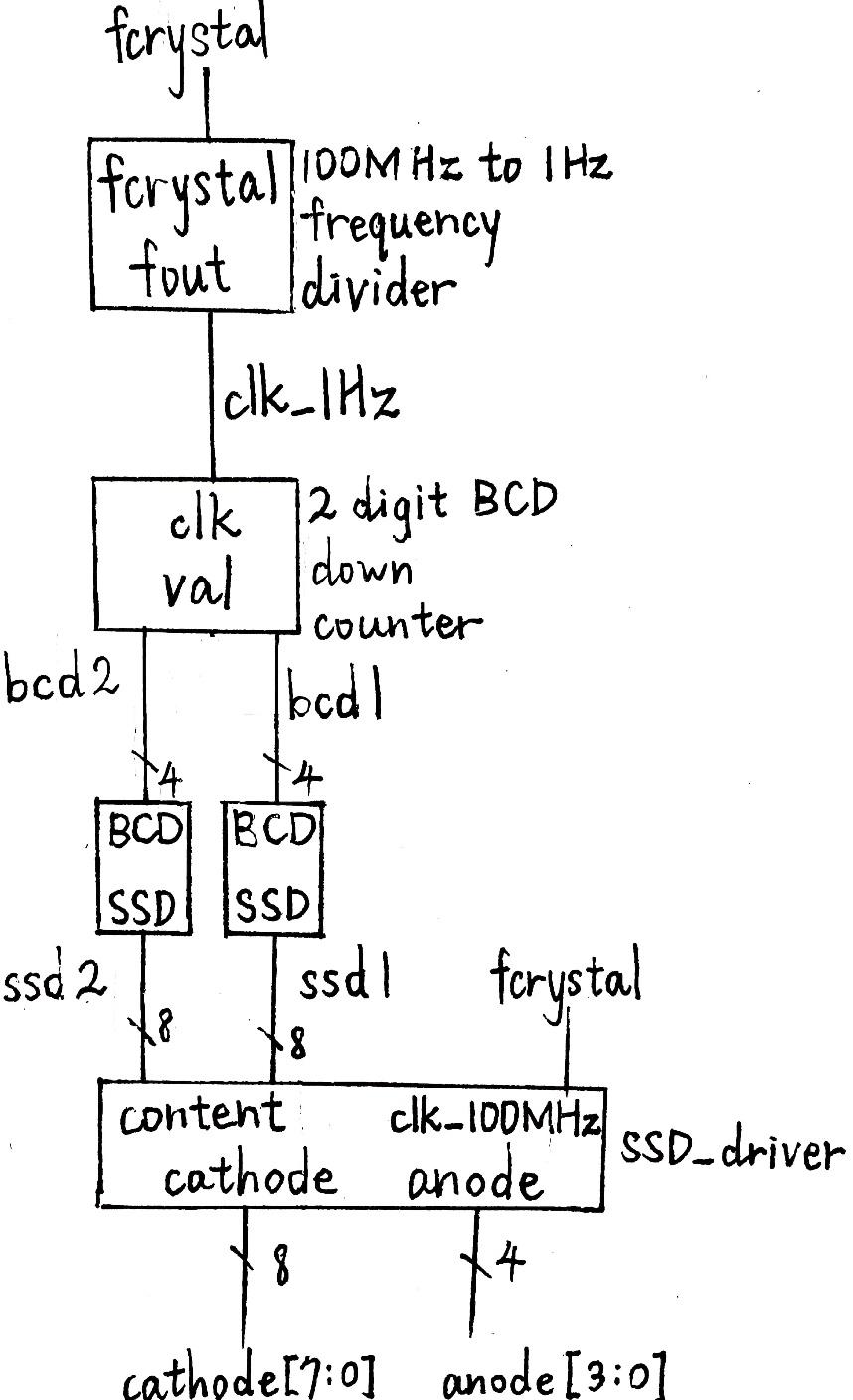
enable // start timing

fcrystal // 100M Hz global clock

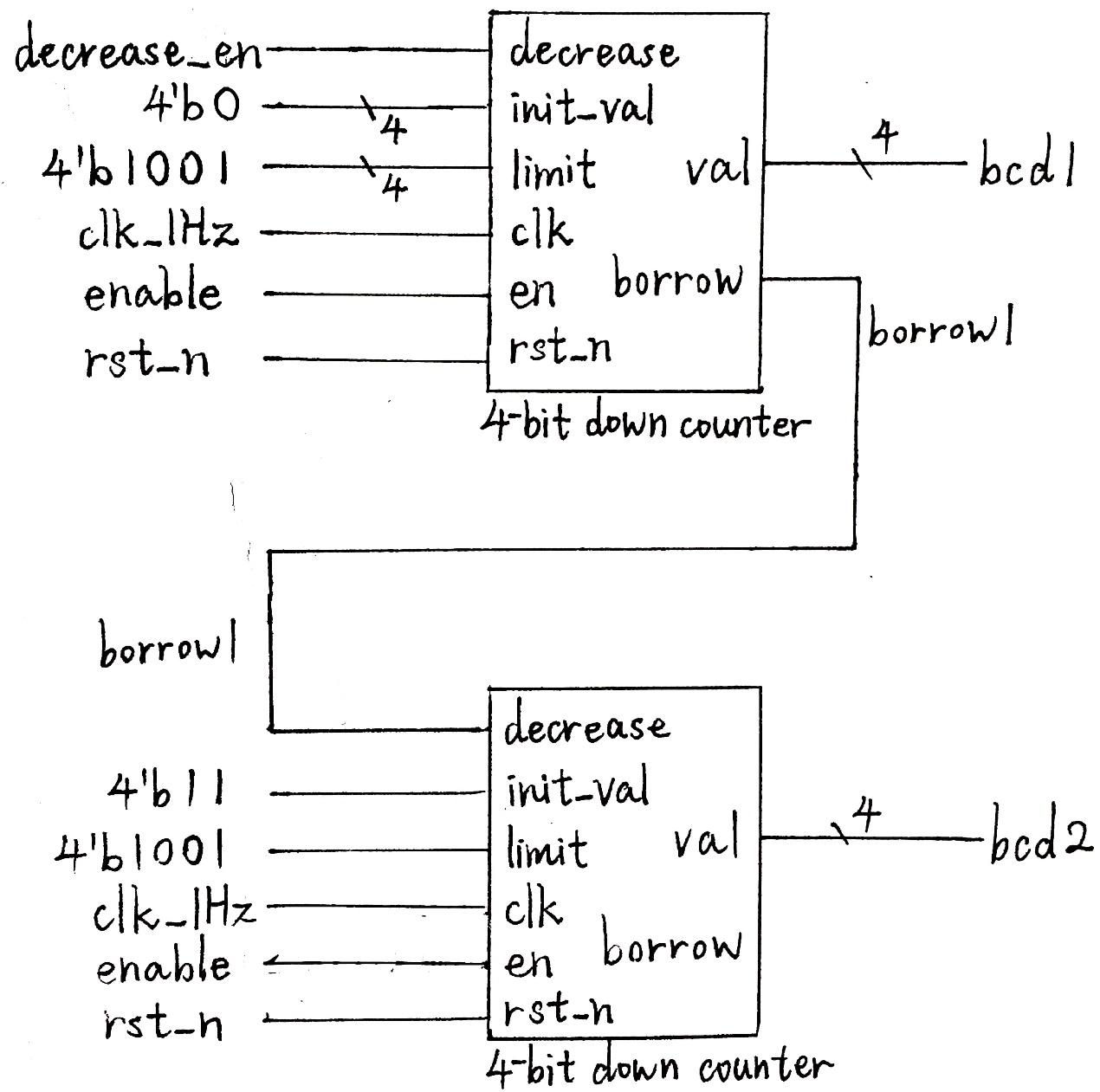
rst\_n // active low reset

circuit diagram:

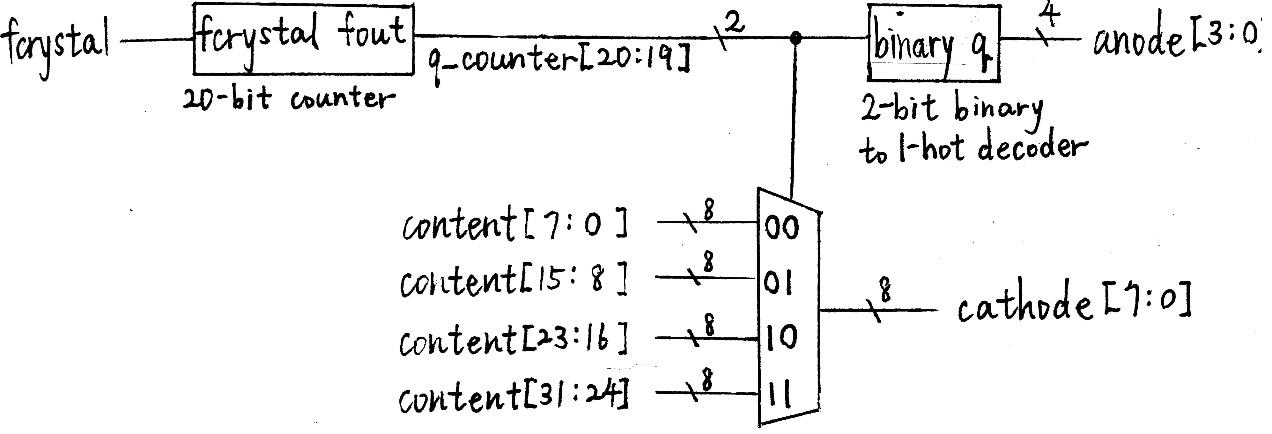
top:



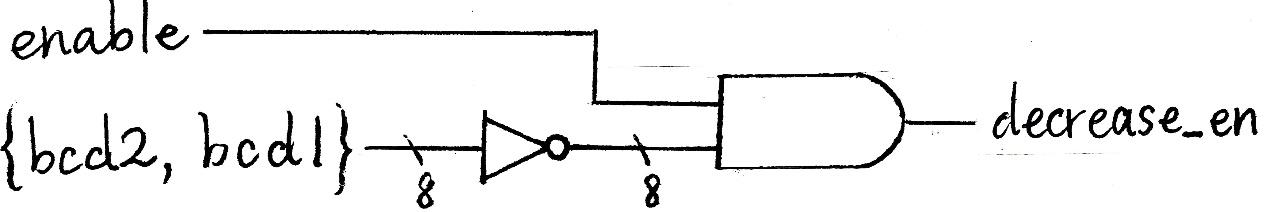
2-digit BCD down counter:



SSD driver:

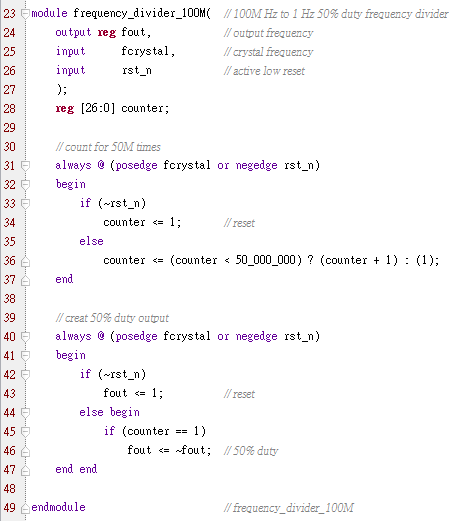


decrease enable:



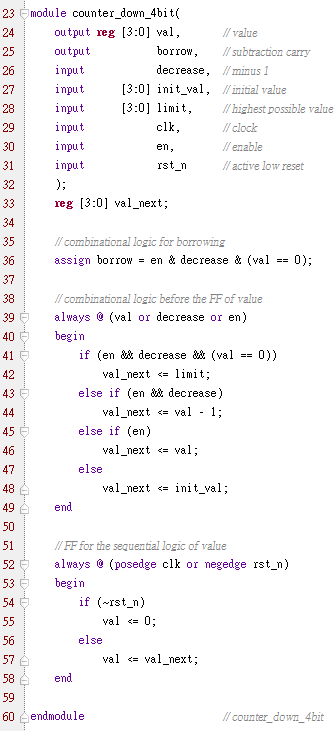
**100M Hz to 1 Hz 50% duty frequency divider:**

把FPGA內建的100M赫茲時脈連接到一個計數器，只要數到50M就重置為一，再將重置的訊號連接到一個TFF即可產生一赫茲的時脈。



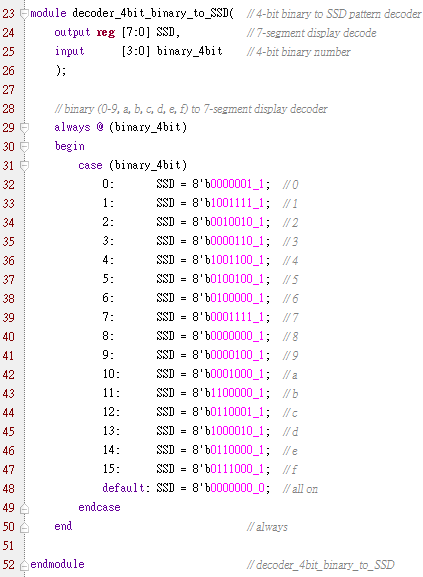
**4-bit down counter:**

四位元下數計數器的code如下圖。其中，借位的訊號較單純，條件為：啟用（enable）且要減一（decrease）且當前的數值為零（value==0）。使用一般的組合邏輯電路即可，如下圖第36行所示。而下一刻的數值（val\_next）則要使用sequential logic來設計：使用四個flip-flop來儲存當前的數值（val），過一個時脈後將下一個數值（val\_next）存入，如下圖第52到58行所示。下一個時脈的數值（val\_next）要使用組合邏輯電路來產生，可分為四種情況：如果不啟用（enable），則重設為初始數值（initial value）；如果啟用但不用減一（decrease），則維持當前數值；如果需要減一且當前數值非零，則直接減一；如果需要減一但當前數值為零，則將數值設為最大值（limit），如下圖第39到49行所示。



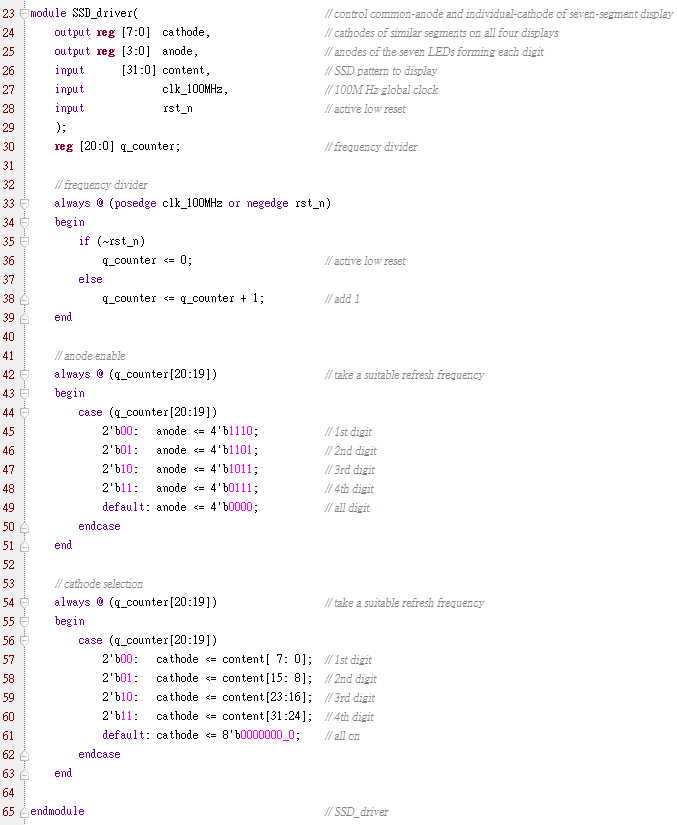
**4-bit binary to SSD pattern decoder:**

其實這裡使用BCD to SSD decoder就好，也就是說，下圖第42到47行是不必要的，因為前面連接的是一個只有可能產生0~9的下數計數器。

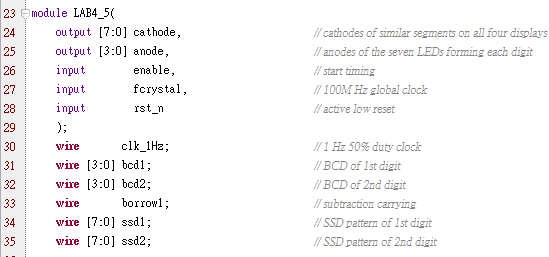


**controller for common-anode and individual-cathode of seven-segment display:**

有了前面那幾個模組之後，從三十開始倒數到零就只差最後一步了。這個模組主要是為了解決FPGA的七段顯示器的那四個digits無法同時顯示四個不同pattern的問題。我們在這裡可以利用人眼視覺暫留，以足夠快的更新頻率輪流在不同digit顯示各個pattern。實作如下：先使用除頻器將FPGA內建的100M赫茲時脈處理成適當的頻率後（100M÷219），連接到一個1-hot解碼器與一個多功器，分別用以控制七段顯示器的anode與cathode。



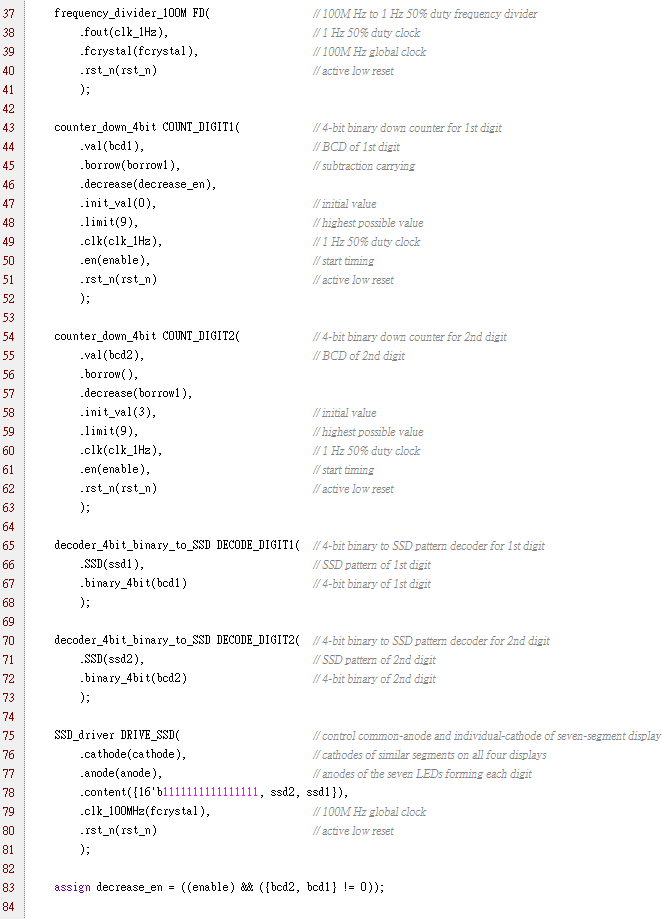
**top:**



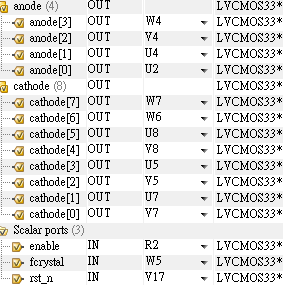
// some modules



最後，只要將剛剛建立好的那些模組串起來就大功告成了。由於三十秒的倒數計時器需要使用到七段顯示器的其中兩個digits，所以我在下圖第78行處將連接到SSD\_driver的剩下的另外兩個digits用1補滿。另外，在第83行處，我將decrease\_en設計成僅當有需要下數的時候才將開始下數的訊號傳入第46行的計數器。



implement:



conclusion:

這一題要做的是一個三十秒的倒數計時器。這一題與前幾題相較之下複雜許多，考驗分析問題的能力。如果碰到像這種功能較複雜的題目，可以先試著將大問題分解成數個小問題，在針對各個小問題構造出對應的模組，最後再整合起來。雖然寫硬體語言和寫軟體語言有很大的不同，但卻有類似的解題思路：例如寫軟體時可以把大問題解剖成數個小問題，再針對各個小問題寫出function，最後用main()整合起來。