# Experiment 3. 簡易音樂盒設計

## 【Purpose】製作撥放音樂的控制電路

i. 運用編碼解碼原理
ii. 運用記憶體掃描原理
iii. 了解音樂的產生原理
iv. 了解音樂輸出的控制方式
v. 了解樂譜的製作方式

## 【Experimental background】

Verilog 設計經驗

## 【Principle and explanation】

1. 音樂產生原理

喇叭發聲原理是根據提供頻率不同，產生不同的音高。依照此原理藉由板子上的頻率(50MHz)，經過除頻器控制，輸出我們所需要的音階範圍，而音譜之間頻率根據原音和高八度音相差 2 倍，以及兩個音之間共相差了 12 個音階的原理，可以計算出每個半因之間相差 $\sqrt[12]{2}$ 倍，如此可以很容易設計出每個音的音高所對照的頻率。音樂則可以被分成音高與音長，音長部分根據音樂的旋律，來決定出每個音需要有多長時間，並使用除頻器輸出節拍與利用計數器來數出每個音的音長。

| Hz | Scale | Hz | Scale | Hz | Scale |
|---|---|---|---|---|---|
| 195.99 | So(bass) | 207.65 | #So(bass) | 220 | La(bass) |
| 233.08 | #La(bass) | 246.94 | Si(bass) | 261.62 | Do |
| 277.18 | #Do | 293.66 | Re | 322.12 | #Re |
| 329.62 | Mi | 349.22 | Fa | 369.99 | #Fa |
| 391.99 | So | 415.30 | #So | 440 | La |
| 466.16 | #La | 493.88 | Si | 523.25 | Do(treble) |
| 554.36 | #Do(treble) | 587.32 | Re(treble) | 622.25 | #Re(treble) |
| 659.25 | Mi(treble) | 698.45 | Fa(treble) | 739.98 | #Fa(treble) |
| 783.99 | So(treble) | 830.66 | #So(treble) | 880 | La(treble) |
| 932.32 | #La(treble) | 987.76 | Si(treble) | 1046.50 | Do(Super high) |

Table 1. 頻率音高對照表

2. 音樂輸出的控制方式

在上一小節知道音高是由頻率的高低所決定，音長是由所在頻率維持的長短所決定，所以製作音樂上一個音要包含音高及音長兩個要素。

**音高：**

從數位的角度去看音高，一個八度(全音加半音)有 12 個音，在一般的音樂上 2 個八度已經很足夠，所以在設計上採用 5bits 來表示，最多則可以表示 2 個八度多。在實作上會先把每個音的除數儲存在記憶體中，要輸出那個音時，則利用除頻器除以記憶體的內容值來實現；舉例來說，使用實驗版內建時脈為 50MHz 要產生中央 La，從上一小節知道輸出頻率為 440Hz，所以記憶體中要儲存 113636。而輸出音的方式，則會先在記憶體每個位置儲存對應的音高，藉由位址線的選擇，決定哪個音的輸出，舉例來說，現今記憶體位置 0 儲存 La，記憶體位置 1 儲存 Si，如果要輸出 La 則位置要輸入 0。

**音長：**

一個音有分成 1/32、1/16、1/8 音符等；為了能表示這些長度又符合大部分歌曲的需求，設計上使用 3bits，共可以顯示 7 個不同的音長，例如 3'b000 表示 1/32 音符，3' b001 表示 1/16 音符。在電路上必須設計解碼器把 3bits 的資料轉換成相對的長度，再由後端的比較器決定什麼時候該換下一個音，舉例來說，先把最短的音符(1/32)當作基準長度設為 1，則 1/16 音符就是 2，1/8 音符就是 4，將音符轉換成比較熟悉的數值，這時電路會有一個上數器，上數的頻率為(1/32)音符的快慢，將上數器的值與解碼器產生出的值利用比較器做比對，如果值相同時，則表示這音長已經達到，發出控制訊號要求輸出下一個音，如此一來就完成了音長的控制。下表為音符對應解碼後的值。

| 音符 | 拍子 | 解碼後值 |
|---|---|---|
| 1/32 note | 1/8 beat | 1 |
| 1/16 note | 1/4 beat | 2 |
| 1/8 note | Half beat | 4 |
| 1/4 note | 1 beat | 8 |

| | | |
|---|---|---|
| 1/2 note | 2 beat | 16 |
| 3/4 note | 3 beat | 24 |
| 1 note | 4 beat | 32 |

Table 2 音符解碼後的值對應表

**休止符、換音及結尾:**

當音樂正在播放時，難免會遇到連續相同的音以及休止符，為了避免連續相同的音聽起來會連在一起變成一個較長的音，在設計上必須把它斷開，斷開的原理是利用換下一個音時，同時對電路的總輸出發出 reset 訊號，使頻率短暫的變化為 0，聲音就會感覺稍微停頓一下；而休止符原理相同，在休止符期間，讓總輸出頻率為 0。結尾的控制是讓換下一個音的計數器不動，如此一來就不會切換音符，停止在結尾狀態。

| Bit[7:5] | 音長 | Bit[4:0] | 音高 | Bit[4:0] | 音高 | Bit[4:0] | 音高 | Bit[4:0] | 音高 |
|---|---|---|---|---|---|---|---|---|---|
| 000 | 1/32note | 11000 | #Fa(H) | 10000 | #La | 01000 | Re | 00000 | Rest |
| 001 | 1/16note | 11001 | So(H) | 10001 | Si | 01001 | #Re | 00001 | So(L) |
| 010 | 1/8note | 11010 | #So(H) | 10010 | Do(H) | 01010 | Mi | 00010 | #So(L) |
| 011 | 1/4note | 11011 | La(H) | 10011 | #Do(H) | 01011 | Fa | 00011 | La(L) |
| 100 | 1/2note | 11100 | #La(H) | 10100 | Re(H) | 01100 | #Fa | 00100 | #La(L) |
| 101 | 3/4note | 11101 | Si(H) | 10101 | #Re(H) | 01101 | So | 00101 | Si(L) |
| 110 | 1note | 11110 | Do(HH) | 10110 | Mi(H) | 01110 | #So | 00110 | Do |
| 111 | | 11111 | 音樂結束 | 10111 | Fa(H) | 01111 | La | 00111 | #Do |

Table 3. 音高語音長對照表

樂譜的製作主要就是要將音高與音長的資訊存入記憶體內，在設計上我 們採用了 3bits 的音長以及 5bits 的音高，所以樂譜是採用 8bits 的記憶體來存放，音長在前，音高在後，並且將 8 個 bits 全為 1 的情況設定為結束符號， 也就是說遇到 8’hFF 時就是一首歌的結尾，要做停止播放的動作；音高與音長的對應表如下：

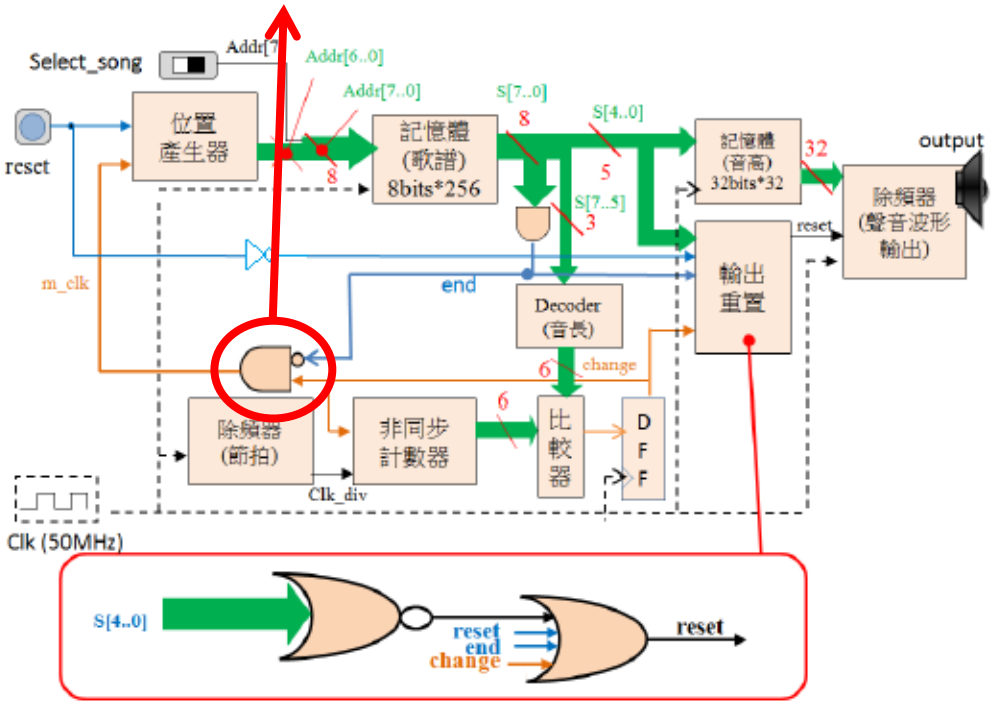
Song:butterfly

| | +0 | +1 | +2 | +3 | +4 | +5 | +6 | +7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 60 | 4A | 4A | 4F | 2F | 4F | 51 | 72 |
| 8 | 60 | 52 | 32 | 52 | 54 | 76 | 60 | 74 |
| 16 | 54 | 51 | 6D | 71 | 52 | 54 | 52 | 51 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 24 | 6F | 40 | 4F | 4A | 2F | 4F | 51 | 72 |
| 32 | 60 | 52 | 32 | 52 | 54 | 76 | 60 | 74 |
| 40 | 54 | 51 | 6D | 71 | 52 | 54 | 52 | 51 |
| 48 | 6F | 6A | 76 | 56 | 59 | 7B | 56 | 59 |
| 56 | 5B | 5E | 5B | 59 | 76 | 60 | 74 | 54 |
| 64 | 56 | 79 | 56 | 52 | 54 | 56 | 54 | 52 |
| 72 | 6F | 60 | 76 | 56 | 59 | 7B | 56 | 59 |
| 80 | 5B | 5E | 5B | 59 | 76 | 60 | 74 | 54 |
| 88 | 56 | 79 | 56 | 52 | 54 | 56 | 54 | 52 |
| 96 | 6f | 60 | FF | | | | | |

## 【實作】

做出簡易音樂盒的控制電路，可以經過指撥開關選擇曲目，將記憶體中對音歌譜解碼並輸出到喇叭上。並且在歌曲撥放完成後將輸出 8’hFF 停止 address generator 更新記憶體位置。



此架構圖如上圖所示，包含:

**2 個除頻器(divider):**

節拍除頻器(Beat frequency divider) :轉換板子上提供的 50MHz 轉換成最短音符長度(32 分音符)。
“1/4 音符為 1 拍則 1/32 音符為 1/8 拍。而我們設定樂曲節奏的快慢為 100 拍/ 分鐘，也就可以推算說每 1 拍為 6/10 sec 或是說 10/6 HZ，則除頻器的基準為 1/32 音符就是 1/8 拍又等同於(1/8*6/10) sec = 80/6HZ。”

**記憶體位置產生器(Address Generator):**

將所選取歌曲的起始位置傳送給歌譜 memory(Sheet Memory)，並當 m_clk 訊號觸發時，將 address+1，使 memory 傳送下一個音符到輸出進行解碼，按下 reset 後 address 將會為 0。Sel 會併入輸出的 MSB，{sel,addr[6:0]}。
“請注意，本次電路設計，如果音樂撥放結束，將會停止位置產生器工作，找助教檢查前請先開啟 reset 否則無法工作!!”

**2 個記憶體 memory:**

歌譜記憶體(Sheet music memory):保存兩首音樂歌譜的記憶體，記憶體大小為 8bits*256，記憶體檔案甲班請選擇 song_A.mif、乙班請選擇 song_B.mif。
節拍倍率記憶體(Pitch music memory): 保存所有音符對應的除數，讓除頻器能產生對應的音符頻率，記憶體大小為 32bits*32，選用 sound1.mif。

**解碼器 Decoder(音長):**

將音符前 3bit 解碼成對應節拍提供比較器做比對。

**比較器 Comparator:**

檢查計數器輸出與解碼器輸出是否相同。

**非同步計數器 Asynchronous counter:**

當 CLK_div 正緣觸發時+1，change 訊號正緣時 reset，為非同步劑暑器(reset 與 clk 不同步)。

**Output Reset:**

當 reset(重頭撥放)，END(歌曲結束)、change(換音)訊號為 1 時或是 S[4:0]為 0(休止符)，拉起 reset 訊號。

# 課堂上實作內容

1. 完成上述架構圖之所有模組，各模組請用 verilog 語法來撰寫。

2. 街上喇叭實現音樂盒

額外補充:

- 本實驗為音樂盒控制電路，所以記憶體模組的寫入致能以及資料輸入皆為接地即可
- m_clk 的訊號來源為~end & change，目的是為了音樂結尾時，不會繼續輸出下一音符，就停止在結尾符號上。
- 除頻器(節拍)輸出為 13.33Hz，因為歌曲節拍為 100 拍/分。
- 歌譜與過去幾屆不同，請勿抄襲，抓到就嚴重扣分。

**額外 bonus:**

做出任何有創意或額外功能....。

**實驗報告:**

報告內容包含：整體架構圖、各模組之波形模擬(須解釋如何驗證功能正 確)、各模組的 Verilog code (記憶體模組除外)包含註解、創意介紹(有實作創 意者)以及實驗心得。