Pattern Loader Guide

Daniel 2017/5/10 Initial Version

繼承於 PEB SRAM Memory 結構，本指南敘述實際上 Instruction Memory 與 Pattern Memory 所需填入的資料格式以及功能。Instruction Memory 為儲存 OP Code、TimingSet Index、Operand 等等。Pattern Memory 則用來儲存輸出 waveform 或輸入比較條件的參照位置。

Instruction SRAM (32-Bit) 每個 Slot 上的每個 Channel FPGA 均填入相同的值			Pattern SVM (48-bit) 每個 Slot 上的每個 Channel FPGA 都可以有不同設定
Command (3-bit)	Timing Set (6-bit)	Operand (23-bit)	3-bit Index per Channel (3x16=48) (48-bit) {ch16,ch15,…,ch2,ch1}

與 TimingSet Memory 之關係:
每行 Instruction + Pattern 在執行時，Timing Set(6-bit) + Index per Channel(3-bit) = 9-bit 即為 TimingSet Memory Guide 中所敘述的 512 深度 Timing Set Memory。

而 Instruction 中的 Command 支援下列 micro-instruction:

- NOP // (3'b000) No operation
  // 無動作，僅輸出 pattern，下一個位置為現在的位置+1
  // 但是當 operand 裡的值不為零的時候，為 Repeat instruction
  // 重複次數為 operand 裡的值+1，重複結束前下一個位置為現在的位
  // 置，重複結束後下一個位置為現在的位置+1
- JMP // (3'b001) Conditional Jump to specified address
  // 跳躍，下一個位置為 operand 裡指定的位置
  // 其條件為 CPU_FLAG = 1 (來自 Channel FPGA 的 Register Map)
- JSR // (3'b010) Jump to subroutine
  // 跳至副程式，下一個位置為 operand 裡指定的位置，並將現在的位置+1 存於
  // stack 裡，堆疊最多 16 層
- RTN // (3'b011) Return from subroutine
  // 結束副程式，下一個位置為當前 stack 裡儲存的位置
- PLC // (3'b100) Push loop counter
  // 迴圈次數儲存於 openand 中，並將迴圈次數推進 stack，堆疊最多 16 層
- EOL // (3'b101) End of loop
  // 迴圈的結束點，若當前迴圈次數不為零，則當前迴圈-1，
  // 下一個位置為 operand 裡指定的位置。
  // 若當前迴圈次數為零，則將 stack 推出，下一個位置為現在的位置+1
- END // (3'b110) Instruction halt
  // Burst 結束，將使 T0 不再輸出

- MCH // (3'b111) Immediate match
// maximum speed = 10MHz, 50ns dead zone at the end of period
// Match 指令，match 次數為 operand 裡的值，在得到 fail 時 match counter -1，
// 下一個位置為現在的位置。若 pass 時下一個位置為現在的位置+1。
// 若得到 fail 時 match counter 已為 0，則輸出真正的 fail 訊號，且下一個位置
// 為現在的位置+1

當 Operand 裡面儲存的是記憶體位置時，資料寬度為 23-bit，等同於 8M 記憶體深度
當 Operand 裡面儲存的是 Counter 時，資料寬度為 16-bit，最大數到 65536

Debug Tool 建議:
Debug Tool "Pattern Loader" 吃文字檔(可多次 load 不同檔案)

******.ins 內容:**

```
// 支援註解 "//"
// "#"開頭後面跟著 0xABCD 為記憶體起始位置(支援"0xABCD" 16 進制或"1234" 10 進制)
#0x1000
// 格式: 指令(文字) TimingSet(0~63) Operand (支援"0xABCD" 16 進制或"1234" 10 進制或不填就為 0 )
NOP     10              // 指令 NOP TS=10 Operand=0, Address = 0x1000
JMP     5   0x1000      // 指令 JMP TS=5 Operand=0x1000, Address = 0x1001
#1234
PLC     3   100         // 指令 PLC TS=3 Operand=0, Address = 100, Address = 1234
// Pattern Loader 不需要 check 記憶體位置是否衝突,只需將每個#後跟著的內容依序填入記憶體即可
// 未定義到的記憶體位置不用去理他，也不要去填 0
```

******.pat 內容:**

```
// 支援註解 "//"
// "[ ]" 中括弧內的 0~7 為 define，可對應轉換到後面跟著的文字或數字
[0] HZ
[1] H
[2] L
[3] 1
// "default" 後面跟著 0~8，當某一 Channel 未被定義，則預設為此數字
default 7
// "@" 開頭後面跟著 1~6 為 slot 編號
@3  // slot3
// "#"開頭後面跟著 0xABCD 為記憶體起始位置(支援"0xABCD" 16 進制或"1234" 10 進制)
#0x1000
// 格式: "chX" X=1~64 (channel)，後面跟著"[]"定義過的 pattern，同一行內可以連續定義個多個 ch
ch1 HZ ch3 L ch64 1 // ch1 = 3'd0, ch3 = 3'd2, ch64 = 3'd3, 其他 channel = 3'd7
// Pattern Loader 不需要 check 記憶體位置是否衝突,只需將每個#後跟著的內容依序填入記憶體即可
// 未定義到的記憶體位置不用去理他，也不要去填 0
```