TimingSet RAM Guide

Daniel 2017/5/10 Initial Version

Daniel 2017/7/19 Add "Compare_V"

New PEB Board 進行 digital pattern 測試時所需要的時序我們稱之為 TimingSet，TmingSet 之組成由下列表格顯示:

TS Period	D0 Loc	D0 Data	D1 Loc	D1 Data	C0 Loc	C1 Loc	C Data	C Mode	C Com Ignore	T0 Loc	T0 Data	T1 Loc	T1 Data	TS RAM Address
0	30-bit	2-bit	30-bit	2-bit	30-bit	30-bit	2-bit	1-bit	1-bit	30-bit	2-bit	30-bit	2-bit	0
30-bit														1
														2
														3
														4
														5
														6
														7
1														8
														9
														10
														11
														12
														13
														14
														15
…														…
63														504
														505
														506
														507
														508
														509
														510
														511

除 TS Period 欄位之外，D0、D1(Data Marker)、C0、C1(Compare Marker)、T0、T1(IO Marker)均為每一 pin 獨立設置。而 TS Period 欄位則以 Channel FPGA 為單位，每 16 channel 共用一個 TS Period Memory。雖然硬體上可以將每個 Channel FPGA 的 TS Period 欄位填入不同的值，但是目前僅需將所有 Slot 上的所有 Channel FPGA 的 TS Period 填入完全一樣的設定。

TS Period的值為整個TimingSet每八個位置共用，所以以深度512(address 9-bit)的TimingSet RAM而言，TS Period 的 Address 僅需 6-bit。由於 address 之 bit 數量不同以及其為 16-channel 共用，硬體實際上對於 TS Period 儲存空間是一個獨立於 TimingSet 的 Memory，只是在運作時會一起同步運算 address。

整個深度 512 的 TimingSet RAM 組合起來的寬度非常寬，在此解釋每個欄位的實際作用之前，先解釋跑 Pattern 時的 Address 運作:

1. TimingSet 對外的規格是 64 組，其實也就是 TS Period Address 6-bit，相當於整個 TimingSet RAM address 的 MSB 6-bit。這個 6-bit 會儲存在跑 Patterm 時的 Instruction Memory 的一個欄位之中。
2. 跑 Patterm 時每一行 Instruction Memory 可以隨時切換 64 組中的一組 TimingSet 設定值，但實際運作還需要 LSB 的 3-bit 來決定每個 Marker 設定。此 3-bit 就是儲存在每個 Channel 使用的 Pattern Memroy 中。這可以對應到之後解釋每顆 Channel FPGA 的 Pattern Memroy 寬度為 3x16=48bit。

了解到 TimingSet RAM address 運作後，再來解釋每個欄位的實際作用:

1. TS Period 為每個 cycle 的測試周期，寬度為 30-bit。亦即表示每個 instruction 的測試周期可運作範圍為 2^{30}ns(1 秒左右)。注意到 TS Period 最小設定值為 30 (30nsec)。
2. D0 Location 為 Data Marker 0 產生時的位置。例如 TS Period 為 100ns，而我想要在 100ns 中間實現一個位置在 15ns 的 Data 波形改變，所以 D0 Location 設定為 15ns。
3. D0 Data 為 Data Marker 0 產生時的輸出波形。Data = 2'b00 表示波形輸出 low，Data = 2'b01 表示波形輸出 high，Data = 2'b10 表示波形輸出由 high 至 low，Data = 2'b11 表示波形輸出由 low 至 high。
4. D1 Location 為 Data Marker 1 產生時的位置。
5. D1 Data 為 Data Marker 1 產生時的輸出波形。Data = 2'b00 表示波形輸出 low，Data = 2'b01 表示波形輸出 high
6. C0 Location 為 Compare Marker 0 作用時的位置。例如 TS Period 為 100ns，而我想要在 100ns 中間實現一個位置在 50ns 的瞬間(strobe)比較 high or low，所以 C0 Location 設定為 50ns。
7. C1 Location 為 Compare Marker 1 作用時的位置。C1 只有在後面敘述的"C Mode = 1"時有作用，另外 C1 Location 必須大於或等於 C0 Location。
8. C Data 為 Compare Marker 作用時比較的依據。Data = 2'b00 為比較"low"， Data = 2'b01 為比較"high"，Data = 2'b10 為比較"middle"(介於 VOH 與 VOL 之間)，Data = 2'b11 則為"valid" (不是 High 就是 Low，非 middle)。
9. C Mode = 0 時使用 Compare Marker 0 做瞬間比較(strobe compare)，C Mode = 1 時使用 Compare Marker 0 與 Compare Marker 1 所定義的時間區間中做時間範圍內的比較(window compare)。
10. C Comp Ignore = 0 時為使用 Compare Marker 正常比較，C Comp Ignore = 1 時為不比較(Don't Care)
11. T0 Location 為 IO Marker 0 產生時的位置。例如 TS Period 為 100ns，而我想要在 100ns 中間實現一個位置在 15ns 的輸出入方向改變，所以 T0 Location 設定為 15ns。
12. T0 Data 為 IO Marker 0 產生時用作控制 ADATE305 的"RCV"，為輸出入的設定。Data = 2'b00 表示 Hi-Z，Data = 2'b01 表示 Drive Enable，Data = 2'b1x 表示不做任何改變。
13. T1 Location 為 IO Marker 1 產生時的位置。
14. T1 Data 為 IO Marker 1 產生時輸出入的設定。

常用 Format (Drive 0 or 1 or High-Z) 範例:

P.S DMx_DATA = 2'b00 = d

DMx_DATA = 2'b01 = u

DMx_DATA = 2'b10 = f

DMx_DATA = 2'b11 = r

NF_0: DM0_LOC = 0, DM0_DATA = 2'b00, DM1_LOC = 0, DM1_DATA = 2'b00
TM0_LOC = 0, TM0_DATA = 2'b01, TM0_LOC = 0, TM0_DATA = 2'b01
C_IGNORE = 1, C_DATA = 2'b11, CM0_LOC = location_1, CM1_LOC = location_1

NF_1: DM0_LOC = 0, DM0_DATA = 2'b01, DM1_LOC = 0, DM1_DATA = 2'b01
TM0_LOC = 0, TM0_DATA = 2'b01, TM0_LOC = 0, TM0_DATA = 2'b01
C_IGNORE = 1, C_DATA = 2'b11, CM0_LOC = location_1, CM1_LOC = location_1

RZ_1: DM0_LOC = location_1, DM0_DATA = 2'b01, DM1_LOC = location_2, DM1_DATA = 2'b00
TM0_LOC = 0, TM0_DATA = 2'b01, TM0_LOC = 0, TM0_DATA = 2'b01
C_IGNORE = 1, C_DATA = 2'b11, CM0_LOC = location_1, CM1_LOC = location_1

RO_0: DM0_LOC = location_1, DM0_DATA = 2'b00, DM1_LOC = location_2, DM1_DATA = 2'b01
TM0_LOC = 0, TM0_DATA = 2'b01, TM0_LOC = 0, TM0_DATA = 2'b01
C_IGNORE = 1, C_DATA = 2'b11, CM0_LOC = location_1, CM1_LOC = location_1

NRZ_0: DM0_LOC = location_1, DM0_DATA = 2'b00, DM1_LOC = location_1, DM1_DATA = 2'b00
TM0_LOC = 0, TM0_DATA = 2'b01, TM0_LOC = 0, TM0_DATA = 2'b01
C_IGNORE = 1, C_DATA = 2'b11, CM0_LOC = location_1, CM1_LOC = location_1

NRZ_1: DM0_LOC = location_1, DM0_DATA = 2'b01, DM1_LOC = location_1, DM1_DATA = 2'b01
TM0_LOC = 0, TM0_DATA = 2'b01, TM0_LOC = 0, TM0_DATA = 2'b01
C_IGNORE = 1, C_DATA = 2'b11, CM0_LOC = location_1, CM1_LOC = location_1

SBC_0: DM0_LOC = location_1, DM0_DATA = 2'b10, DM1_LOC = location_2, DM1_DATA = 2'b01
TM0_LOC = 0, TM0_DATA = 2'b01, TM0_LOC = 0, TM0_DATA = 2'b01
C_IGNORE = 1, C_DATA = 2'b11, CM0_LOC = location_1, CM1_LOC = location_1

SBC_1: DM0_LOC = location_1, DM0_DATA = 2'b11, DM1_LOC = location_2, DM1_DATA = 2'b00
TM0_LOC = 0, TM0_DATA = 2'b01, TM0_LOC = 0, TM0_DATA = 2'b01
C_IGNORE = 1, C_DATA = 2'b11, CM0_LOC = location_1, CM1_LOC = location_1

High-Z: TM0_LOC = 0, TM0_DATA = 2'b00, TM0_LOC = 0, TM0_DATA = 2'b00
C_IGNORE = 1, C_DATA = 2'b11, CM0_LOC = location_1, CM1_LOC = location_1

常用 Compare

Strobe_0: C_IGNORE = 0, C_MODE = 0, C_DATA = 2'b00, CM0_LOC = location_1, CM1_LOC = location_1
TM0_LOC = 0, TM0_DATA = 2'b00, TM0_LOC = 0, TM0_DATA = 2'b00

Strobe_1: C_IGNORE = 0, C_MODE = 0, C_DATA = 2'b01, CM0_LOC = location_1, CM1_LOC = location_1
TM0_LOC = 0, TM0_DATA = 2'b00, TM0_LOC = 0, TM0_DATA = 2'b00

Strobe_M: C_IGNORE = 0, C_MODE = 0, C_DATA = 2'b10, CM0_LOC = location_1, CM1_LOC = location_1
TM0_LOC = 0, TM0_DATA = 2'b00, TM0_LOC = 0, TM0_DATA = 2'b00

Strobe_V: C_IGNORE = 0, C_MODE = 0, C_DATA = 2'b11, CM0_LOC = location_1, CM1_LOC = location_1
TM0_LOC = 0, TM0_DATA = 2'b00, TM0_LOC = 0, TM0_DATA = 2'b00

Window_0: C_IGNORE = 0, C_MODE = 1, C_DATA = 2'b00, CM0_LOC = location_1, CM1_LOC = location_1
TM0_LOC = 0, TM0_DATA = 2'b00, TM0_LOC = 0, TM0_DATA = 2'b00

Window_1: C_IGNORE = 0, C_MODE = 1, C_DATA = 2'b01, CM0_LOC = location_1, CM1_LOC = location_1
TM0_LOC = 0, TM0_DATA = 2'b00, TM0_LOC = 0, TM0_DATA = 2'b00

Window_M: C_IGNORE = 0, C_MODE = 1, C_DATA = 2'b10, CM0_LOC = location_1, CM1_LOC = location_1
TM0_LOC = 0, TM0_DATA = 2'b00, TM0_LOC = 0, TM0_DATA = 2'b00

Window_V: C_IGNORE = 0, C_MODE = 1, C_DATA = 2'b11, CM0_LOC = location_1, CM1_LOC = location_1
TM0_LOC = 0, TM0_DATA = 2'b00, TM0_LOC = 0, TM0_DATA = 2'b00

Compare_X: C_IGNORE = 1, C_MODE = 0, C_DATA = 2'b11, CM0_LOC = location_1, CM1_LOC = location_1
TM0_LOC = 0, TM0_DATA = 2'b00, TM0_LOC = 0, TM0_DATA = 2'b00

Channel FPGA Register Map & Example:

REGISTER: TimingSet Period Data Access

W	0x07	TS_ADDR[5:0]	// Timing Set Address
R	0x07	CUR_TS_ADDR[5:0]	// Current Timing Set Address
W	0x08	TS_PERIOD[29:16]	// Timing Set Period MSB
R	0x08	TS_PERIOD[29:16]	// Timing Set Period MSB
W	0x09	TS_PERIOD[15:0]	// Timing Set Period LSB, Address auto increment
R	0x09	TS_PERIOD[15:0]	// Timing Set Period LSB, Address auto increment

EXAMPLE: 寫入 Timing Set RAM

W	0x07	0	// 由 address 0 開始
W	0x08	period_index_0[29:16]	// 寫入第 1 組 period
W	0x09	period_index_0[15:0]	// address 會自動 +1
W	0x08	period_index_1[29:16]	// 寫入第 2 組 period
W	0x09	period_index_1[15:0]	// address 會自動 +1
…			
W	0x08	period_index_63[29:16]	// 寫入第 64 組 period
W	0x09	period_index_63[15:0]	// address 會自動 +1

EXAMPLE: 讀取 Timing Set RAM

W	0x07	0	// 由 address 0 開始
R	0x08	period_index_0[29:16]	// 讀取第 1 組 period
R	0x09	period_index_0[15:0]	// address 會自動 +1
R	0x08	period_index_1[29:16]	// 讀取第 2 組 period
R	0x09	period_index_1[15:0]	// address 會自動 +1
…			
R	0x08	period_index_63[29:16]	// 讀取第 64 組 period
R	0x09	period_index_63[15:0]	// address 會自動 +1

REGISTER: TimingSet Marker Data Access

```
W   0x0C    {TS_MARKER_SEL[3:0], CHANNEL_SEL[3:0]}  // Select MARKER & Channel
                                                   // TS_MARKER_SEL:
                                                   // 4'd0: DM, 4'd1: IO, 4'd2: CM
                                                   // CHANNEL_SEL[3:0] = CH 1~16
R   0x0C    {TS_MARKER_SEL[3:0], CHANNEL_SEL[3:0]}

W   0x0D    {MARKER_ADDR[8:0], MSB_SEL}   // MARKER_ADDR = 0~512
                                         // MSB_SEL = 0 "LSB 32-bit"
                                         // MSB_SEL = 1 "MSB 32-bit"
R   0x0D    {MARKER_ADDR[8:0], MSB_SEL}
W   0x0E    MARKER_RAM_DATA_MSB          // MARKER_RAM_DATA 依照 TS_MARKER_SEL
W   0x0F    MARKER_RAM_DATA_LSB          //  的不同而不同，參考下面 Example
```

EXAMPLE: 寫入 Channel[3]的 Data Marker

```
W   0x0C    {8'd0, 4'd0, 4'b0011}          // Select Data Marker, Channel[3]
W   0x0D    0                             // Address Start @ address = 0, LSB
W   0x0E    {DM0_DATA_index_0[1:0], DM0_LOC_index_0[29:16]}
W   0x0F    DM0_LOC_index_0[15:0]      // MSB_SEL 會自動由 LSB 轉成 MSB
W   0x0E    {DM1_DATA_index_0[1:0], DM1_LOC_index_0[29:16]}
W   0x0F    DM1_LOC_index_0[15:0]      // MSB_SEL 會自動由 LSB 轉成 MSB, 並且 address + 1
W   0x0E    {DM0_DATA_index_1[1:0], DM0_LOC_index_1[29:16]}
W   0x0F    DM0_LOC_index_1[15:0]
W   0x0E    {DM1_DATA_index_1[1:0], DM1_LOC_index_1[29:16]}
W   0x0F    DM1_LOC_index_1[15:0]
…
W   0x0E    {DM0_DATA_index_511[1:0], DM0_LOC_index_511[29:16]}
W   0x0F    DM0_LOC_index_511[15:0]
W   0x0E    {DM1_DATA_index_511[1:0], DM1_LOC_index_511[29:16]}
W   0x0F    DM1_LOC_index_511[15:0]
```

EXAMPLE: 讀取 Channel[3]的 Data Marker

```
W   0x0C    {8'd0, 4'd0, 4'b0011}          // Select Data Marker, Channel[3]
W   0x0D    0                             // Address Start @ address = 0, LSB
R   0x0E    {DM0_DATA_index_0[1:0], DM0_LOC_index_0[29:16]}
R   0x0F    DM0_LOC_index_0[15:0]      // MSB_SEL 會自動由 LSB 轉成 MSB
R   0x0E    {DM1_DATA_index_0[1:0], DM1_LOC_index_0[29:16]}
R   0x0F    DM1_LOC_index_0[15:0]      // MSB_SEL 會自動由 LSB 轉成 MSB, 並且 address + 1
R   0x0E    {DM0_DATA_index_1[1:0], DM0_LOC_index_1[29:16]}
R   0x0F    DM0_LOC_index_1[15:0]
R   0x0E    {DM1_DATA_index_1[1:0], DM1_LOC_index_1[29:16]}
R   0x0F    DM1_LOC_index_1[15:0]
```

…

R 0x0E {DM0_DATA_index_511[1:0], DM0_LOC_index_511[29:16]}

R 0x0F DM0_LOC_index_511[15:0]

R 0x0E {DM1_DATA_index_511[1:0], DM1_LOC_index_511[29:16]}

R 0x0F DM1_LOC_index_511[15:0]

EXAMPLE: 寫入 Channel[3]的 IO Marker

W 0x0C {8'd0, 4'd1, 4'b0011} // Select IO Marker, Channel[3]

W 0x0D 0 // Address Start @ address = 0, LSB

W 0x0E {IO0_DATA_index_0[1:0], IO0_LOC_index_0[29:16]}

W 0x0F IO0_LOC_index_0[15:0] // MSB_SEL 會自動由 LSB 轉成 MSB

W 0x0E {IO1_DATA_index_0[1:0], IO1_LOC_index_0[29:16]}

W 0x0F IO1_LOC_index_0[15:0] // MSB_SEL 會自動由 LSB 轉成 MSB, 並且 address + 1

W 0x0E {IO0_DATA_index_1[1:0], IO0_LOC_index_1[29:16]}

W 0x0F IO0_LOC_index_1[15:0]

W 0x0E {IO1_DATA_index_1[1:0], IO1_LOC_index_1[29:16]}

W 0x0F IO1_LOC_index_1[15:0]

…

W 0x0E {IO0_DATA_index_511[1:0], IO0_LOC_index_511[29:16]}

W 0x0F IO0_LOC_index_511[15:0]

W 0x0E {IO1_DATA_index_511[1:0], IO1_LOC_index_511[29:16]}

W 0x0F IO1_LOC_index_511[15:0]

EXAMPLE: 寫入 Channel[3]的 CM Marker

W 0x0C {8'd0, 4'd2, 4'b0011} // Select CM Marker, Channel[3]

W 0x0D 0 // Address Start @ address = 0, LSB

W 0x0E {CM_DATA_index_0[1:0], CM0_LOC_index_0[29:16]}

W 0x0F CM0_LOC_index_0[15:0] // MSB_SEL 會自動由 LSB 轉成 MSB

W 0x0E {CM_IGNORE_index_0, CM_MODE_index_0, CM1_LOC_index_0[29:16]}

W 0x0F CM1_LOC_index_0[15:0] // MSB_SEL 會自動由 LSB 轉成 MSB, 並且 address + 1

W 0x0E {CM_DATA_index_1[1:0], CM0_LOC_index_1[29:16]}

W 0x0F CM0_LOC_index_1[15:0]

W 0x0E {CM_IGNORE_index_1, CM_MODE_index_1, CM1_LOC_index_1[29:16]}

W 0x0F CM1_LOC_index_1[15:0]

…

W 0x0E {CM_DATA_index_511[1:0], CM0_LOC_index_511[29:16]}

W 0x0F CM0_LOC_index_511[15:0]

W 0x0E { CM_IGNORE_index_511, CM_MODE_index_511, CM1_LOC_index_511[29:16]}

W 0x0F CM1_LOC_index_511[15:0]