實用數位系統設計
期末專題設計報告

String Matching Engine

組別 ： 銀河守衛隊

組員 ： B103012001 陳胤琿

B103012002 林凡皓

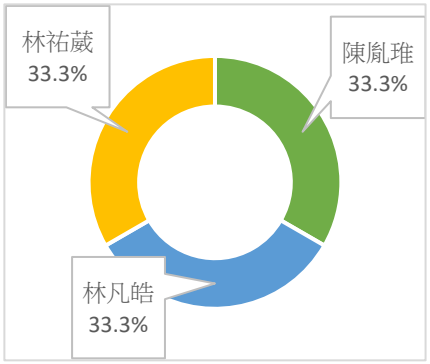
B103011043 林祐葳

組長 ： B103012001 陳胤琿

設計實作摘要

是否通過 RTL (合成前, pre-synthesis)驗證?	**YES**
是否通過 gate-level(post-synthesis)驗證?	**YES**
Clock period (ns)	**20**
Number of cycles	**1860**
Patterns score (Pattern 正確數目)	**100**
Area (um^2)	**23449.580629**
Power (mW)	**653.2817**
Simulation time (ns) (合成後)	**37200**
是否引用其他組的設計構想或架構?	no
Coding style check and enhancement?	**YES**
Code coverage evaluation and enhancement?	**no**
其他重要特色	**no**

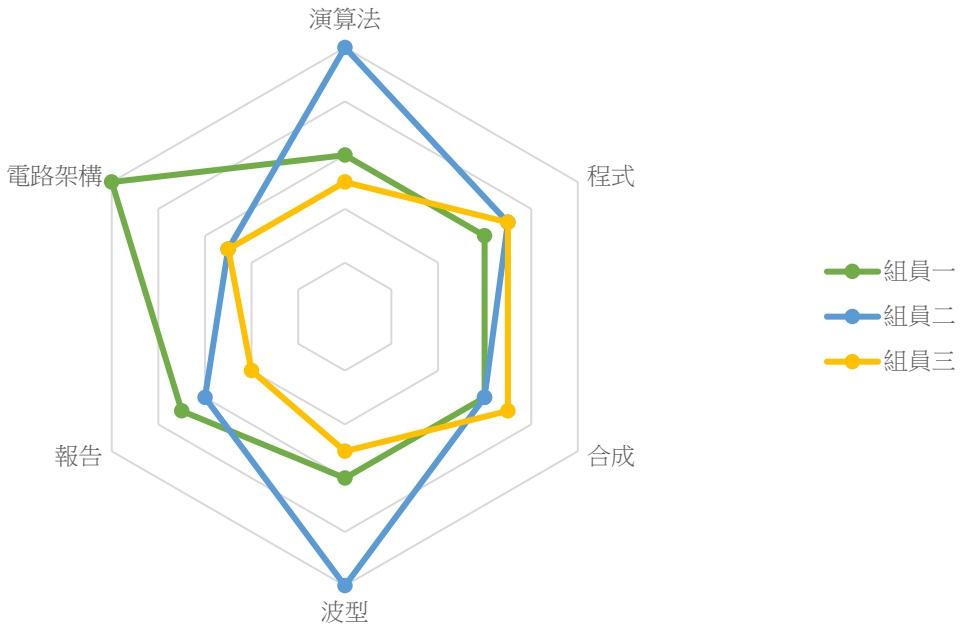
工作分配與貢獻



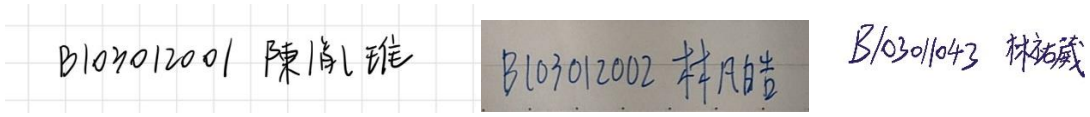
陳胤琲 33.3̄%：
PPT 錄製、電路整合、電路架構。

林祐葳 33.3̄%：
電路撰寫 part B、電路合成、報告撰寫。

林凡皓 33.3̄%：
演算法擬構、電路撰寫 part A、波型模擬、電路優化。

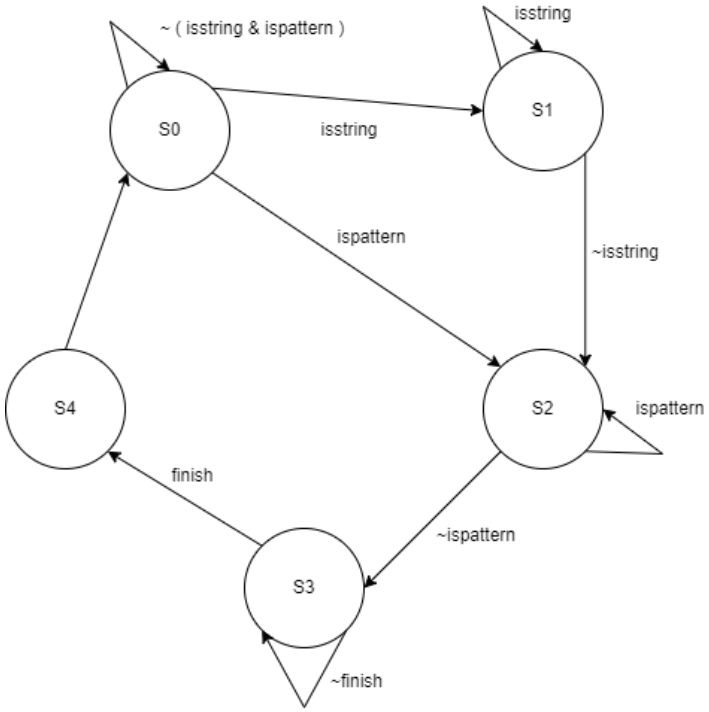


(附上簽名截圖、照片)



一、 演算法介紹與說明

我們這組預計會使用 Finite state machine 和 counter 的結合來實現作品。下圖為 FSM 的狀態圖。



(1) S0 : reset 後的初始狀態
(2) S1 : 得到的 chardata 為 string
(3) S2 : 得到的 chardata 為 pattern
(4) S3 : 執行 string 和 pattern 的比對
(5) S4 : 比對結束

FSM 中較為複雜的會是 S3 的部分，我們的想法是透過兩台 counter，一台為 pattern counter，負責計算目前對到 pattern 的第幾個字母，另一台為 string counter，負責計算目前對到 string 的第幾個字母，string counter 大致上就是一直 +1，然後 pattern counter 會看是不是有對到 string 的字母，如果有對到，那 pattern counter 會 +1，否則 pattern counter 會歸零從頭開始對。另外就是將 pattern counter 現在對到的字母做分類，用不同的狀況來討論怎麼樣是有對到。狀況分成以下幾種：

1. **Pattern 不是特殊字元：** 此情況為最基本的情況，只要此時 string 和 pattern 相同，

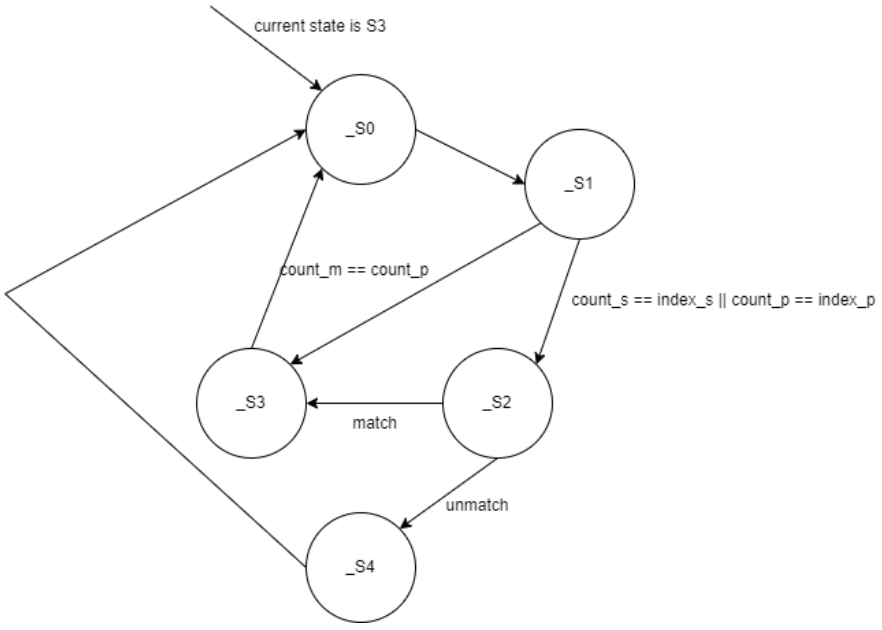
或是 pattern 為 . ，那 pattern counter 和 string patter 都 + 1。

2. **Pattern 為 ^ :** 此時要開頭對到才算，有兩種情況滿足，第一為一開始就對到，第二為在 string 的中間對到。判斷為第一種情況的方法為看 string counter 數到的字母數是不是零，而判斷為第二種情況的方法為看 string counter 數道的前一個字母是不是空格。

3. **Pattern 為 $:** 此時要結尾有對到才算，有兩種情況滿足，第一為 string 在結尾對到，第二為 string 在中間對到。判斷第一種的方法為 string counter 數到的字母數等於 string 總共的字母數，判斷第二種的方法為 string counter 數到的字母為空格。

4. **Pattern 為 * :** 由於 * 可以代表不只一個字元，所以如果遇到 * 那會有一個訊號 pattern_is_star 變成 1，當 pattern_is_star 為 1 時，只要 pattern 不等於 string，pattern counter 會一直維持在 * 的下一個字母，而 string counter 會持續 + 1，直到 pattern 對到 string 為止。

此外，我們可以注意到要判斷結果是否有比對成功住要是看 pattern counter 是否有數到 pattern 的最後一個字母，所以我打算再用一台 finite state machine 來判斷結果是否有比對成功。以下為狀態圖。



(1) **_S0 :** 當 current state 為 S3，即在判斷是否有比對成功時會進入此狀態，而此狀態代表開始判斷是否有比對成功。

(2) **_S1 :** 此狀態為判斷是否比對結束，比對結束有兩種可能，第一種為比對成功的字母數等於 pattern 的總字母數，此情況代表全部的 pattern 都有對 string，也就是有比對成功；

第二種為 string counter 數到的字母數等於 string 的總字母數或是 pattern counter 數到的字母數等於 pattern 的總字母數，此時代表已經沒有可以比對的 pattern 和 string 了，但是無法代表是否比對成功，所以要進入下一個狀態進一步判斷結果。

(3) **_S2 :** 此狀態為進一步去判斷_S1 第二種狀況的結果。由於要考慮到特殊字元 $ ，所以要分成兩種狀況討論，第一種為 pattern 結尾為 $ ，第二種為 pattern 的結尾不是 $ 。第一種狀況中，因為 $ 代表最後一個字母要比對成功才可以，所以比對成功的字母數 + 1 要等於 pattern 的總字母數才算比對成功(因為 pattern 中有一個字母為 $ ，所以比對成功數目要 + 1 才會等於 pattern 的總字母數)。第二種狀況中，只要 pattern 的總字母數等於比對成功的總字母數就代表有比對成功。

(4) **_S3 :** 代表比對結束且有比對成功。

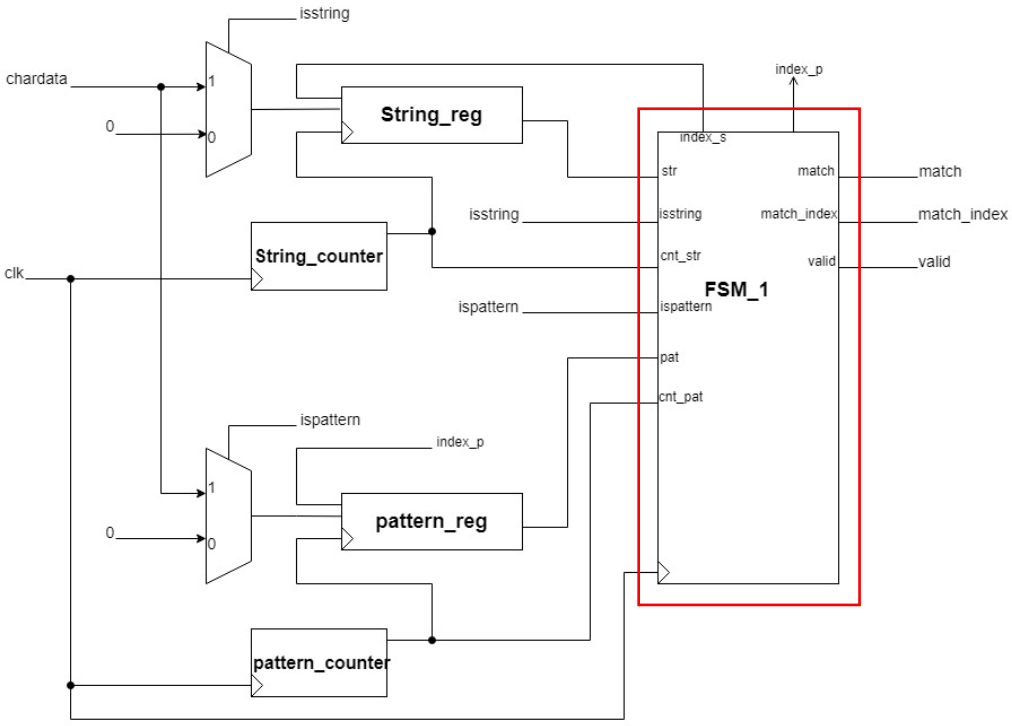
(5) **_S4 :** 代表比對結束但是沒有比對成功。

- **驗算法比較**

最初我們這組是打算使用爆破法和 BM 演算法，但是經過初步的評估，我們認為這次期末專題的 string 跟 pattern 的位元都很少，使用 BM 演算法的優勢並不會很大，甚至還有可能會讓面積變得非常大，因此我們最終選擇使用爆破法。另外就是終極夢幻屁眼派對組的演算法，雖然說他們的效能非常好，但是我們覺得那樣的寫法很不硬體導向，因此我們還是以原本的為主。

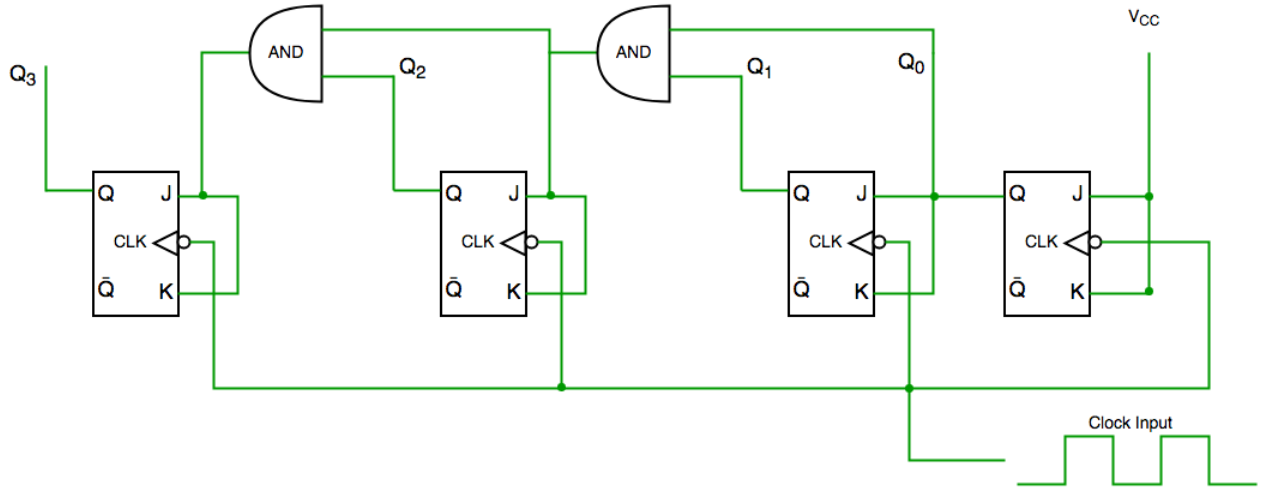
二、 架構介紹與說明

String Matching Engine



我們的電路架構由多工器、暫存器、計數器和幾台有限狀態機組成。其中多工器根據輸入的 isstring/ispattern 值來判斷是 string 還是 pattern，暫存器再將 string 或是 pattern 的值處存起來用作之後的運算。因為我們使用的是基本的暴力法，也就是一個一個比對，我們使用兩個計數器來讓我們比對字元的位置逐一進行。之後他們都會被送往有限狀態機裡進行狀態的轉移與實質上的字元比對。

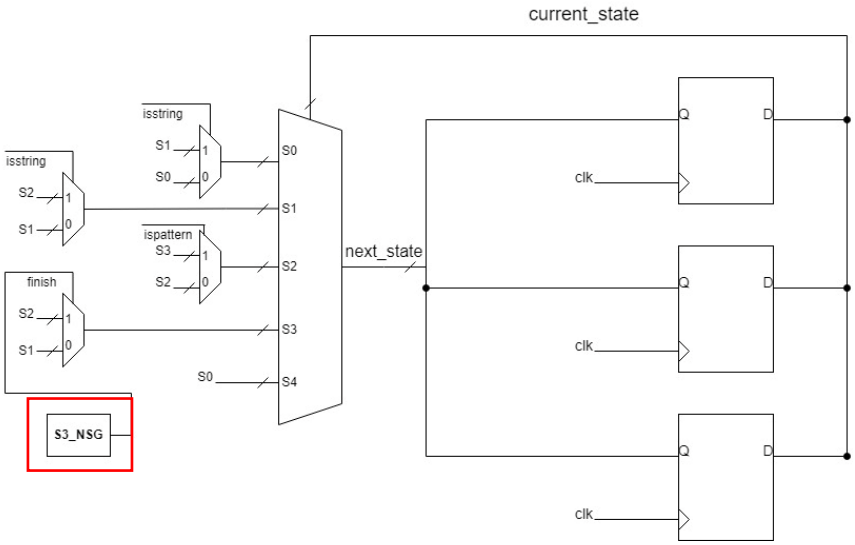
Synchronous Counter



以 4-bits 計數器為例，我們使用的是 5-bits 和 6-bits 的計數器來數 chardata 比對的位

置。所有正反器都使用同一個 posedge 時鐘脈衝來觸發，所以其輸出狀態會在同一時刻一起改變。少了累積延遲，因此可以在更高的頻率上運作，速度更快。

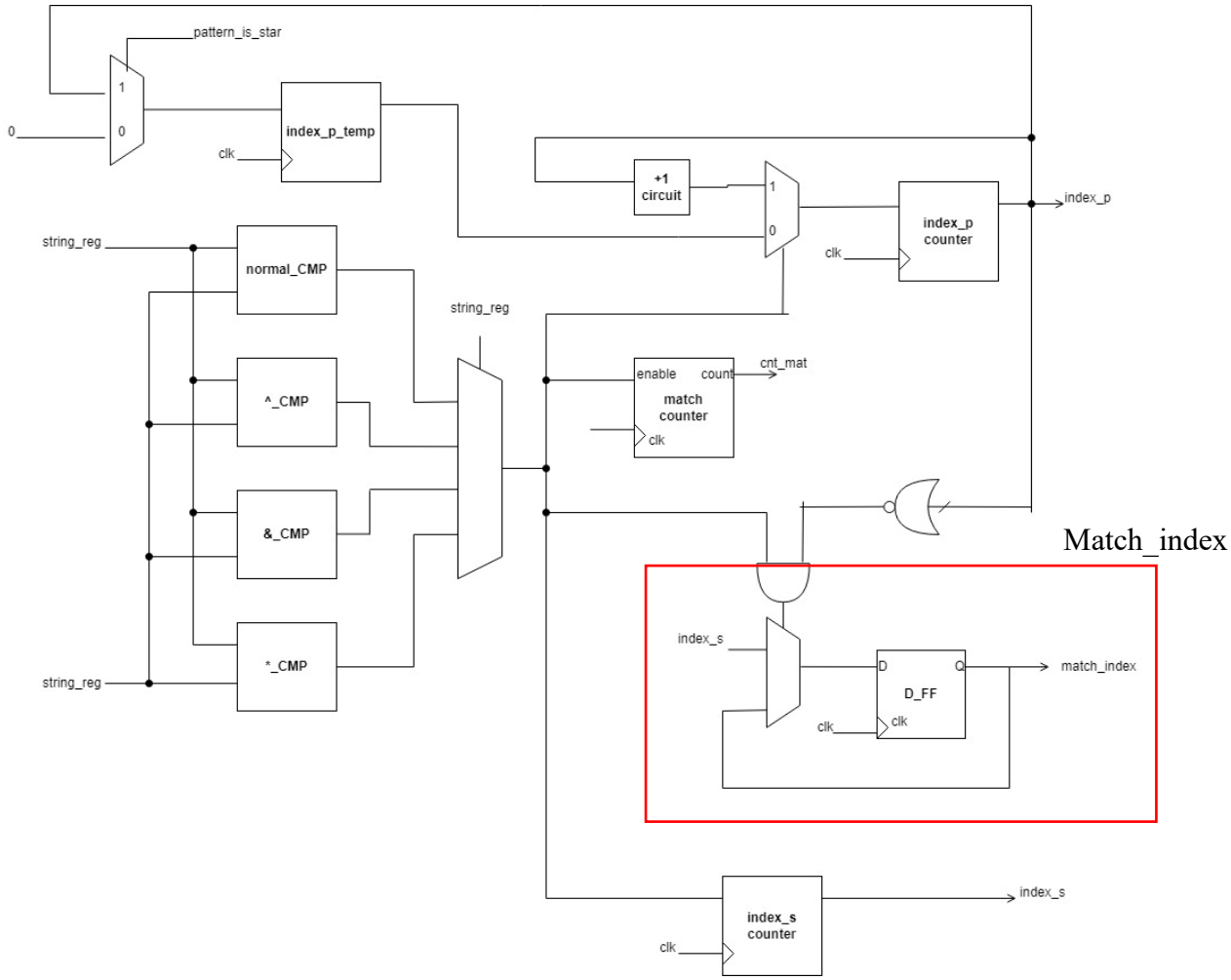
Finite State Machine



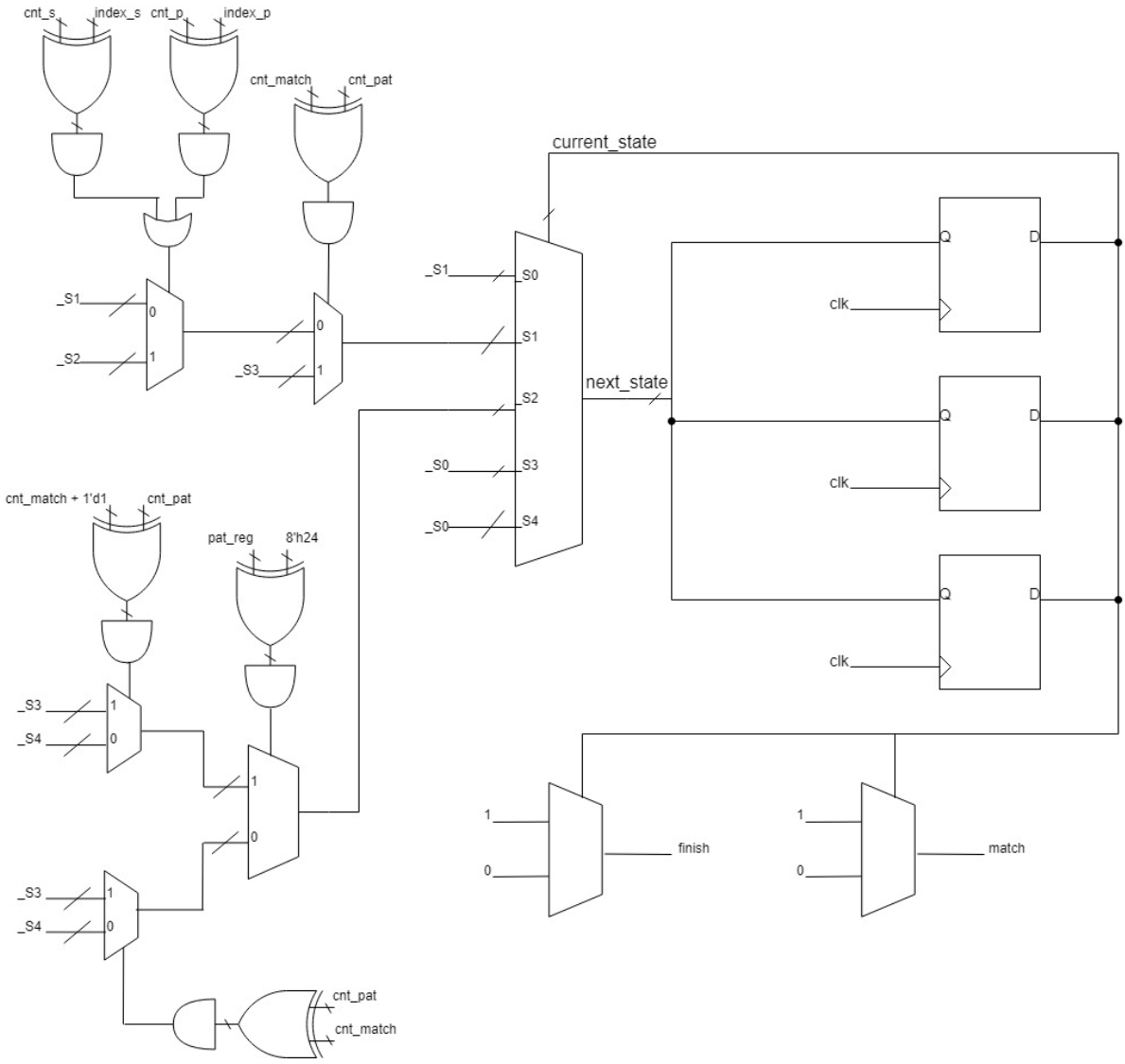
我們的有限狀態機根據不同的情況分成 5 種狀態，包括初始狀態 S0、字串狀態 S1、模式狀態 S2、比對狀態 S3 和結束狀態 S4。系統根據輸入的 isstring/ispattern，從初始狀態開始進行狀態轉換，並根據 chardata 的類型進行相應的處理。最終，系統在 S3 完成比對操作，進入結束狀態。而在 S3 狀態裡其實還有一個 FSM 負責處理實際上的字元比對，用於判別是否比對成功。

S3_NSG

Index_p



此為上述電路之 S3_NSG 的前半部電路，這邊主要是根據不同情況比較的結果得到總共匹配到的字數(cnt_mat)，另外在這個電路中，分別有 index_s 和 index_p 的 counter 做計數，index_s 為一直+1，但在 index_p 中，我們還要考慮是否有 star 的出現和沒匹配到情況，若沒匹配到就會使 index_p 歸 0，而若 star 訊號有將 index_p 的值所進 index_p_temp 中，此時沒匹配到的時候會將 index_p 歸為 index_p_temp，其餘有配到的情況則是 index_p 加 1，而在電路的中右方為輸出 match_index 的地方，若同時有匹配到和 index_p 為 0 時，則會將現在的 index_s 輸出至 match_index 中。



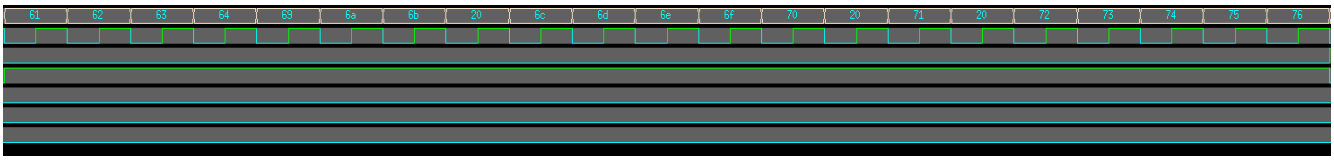
有了匹配的數量(cnt_mat)之後，就可以利用第二台 FSM 來判斷何時會比較結束，當 FSM_1 到 S3 時，FSM_2 就會開始運作，在_S1，_S2 為主要在計算何時結束的狀態，在_S1 中，我們會先檢查 cnt_mat 是否等於 cnt_pat，若有符合則 next_state 為_S3，若沒有的話則會檢查 index_s 和 index_p 是否已經比到底，若已經比到底就會進_S2 狀態，檢查 pattern 是否有$的出現，若有$的話，就檢查 cnt_mat + 1 是否有等於 cnt_pat，若有的話就會進入_S3 狀態，無的話則進入_S4 狀態。

最後此狀態機會根據目前的狀態，輸出 match 值和 finish 值，而這邊的 finish 值將會在 FSM_1 去當作改變狀態的依據。

三、 驗證流程說明

波形結果 (用十六進位表示) :

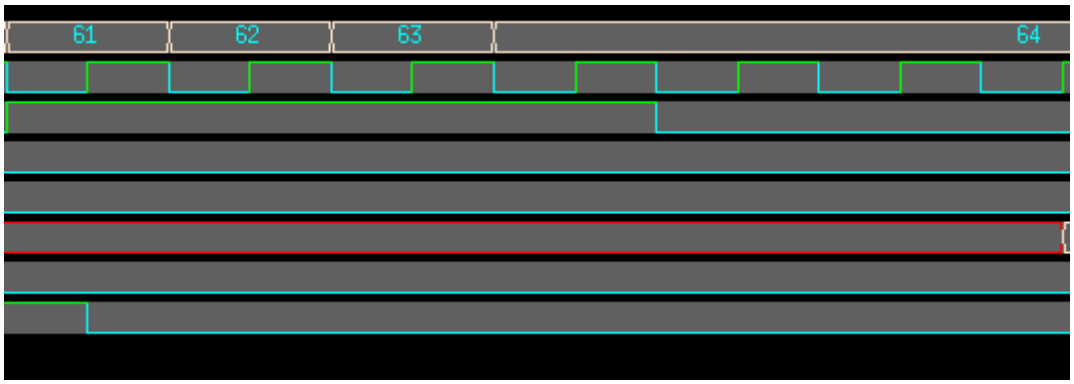
(1) 第一組 string : abcdijk lmnop q rstuv



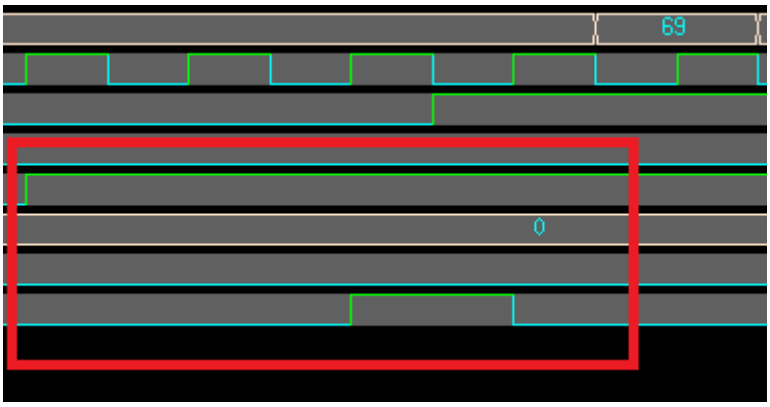
- 第一個 pattern : **1234**

此時當 valid 拉起時，match 應該為 0，如右圖所示。

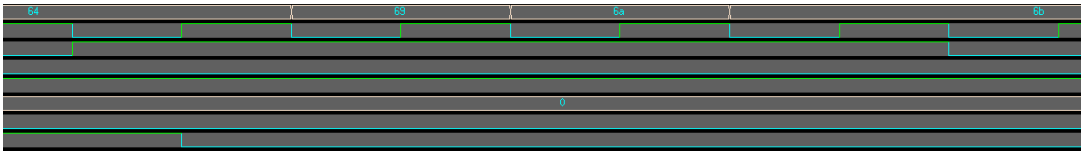
- 第二個 pattern : **abcd**



此時當 valid 拉起時，match 會是 1 且 match_index 為 0，如下圖所示。

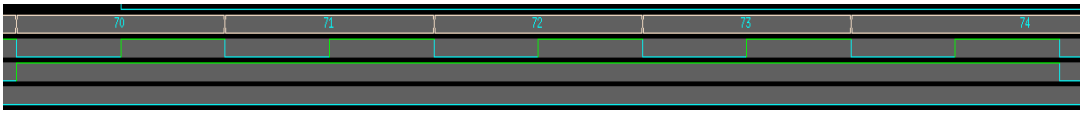


- 第三個 pattern : **dijk**



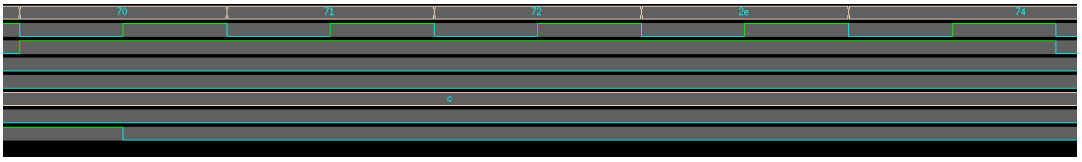
此時當 valid 拉起時，match 會是 1 且 match_index 為 3，如下圖所示。

- 第四個 pattern : **pqrst**

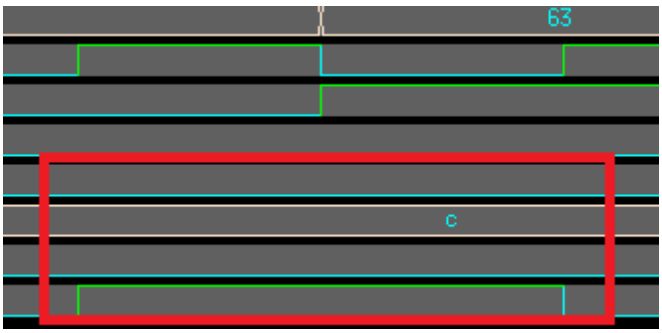


此時當 valid 拉起時，match 會是 0，如下圖所示。

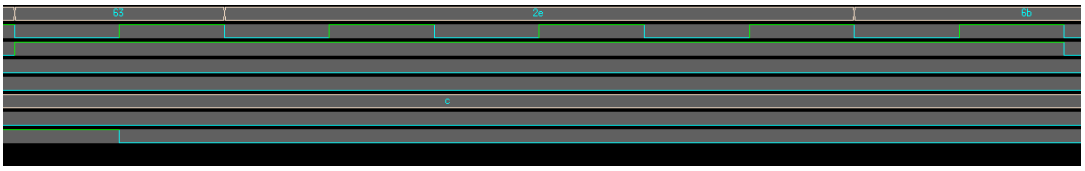
- 第五個 pattern : **pqr.t**



此時當 valid 拉起時，match 會是 0，如下圖所示。

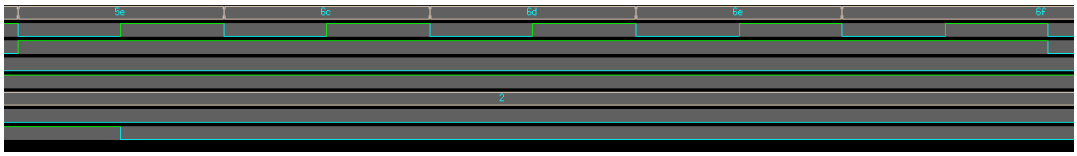


- 第六個 pattern : **c…k**

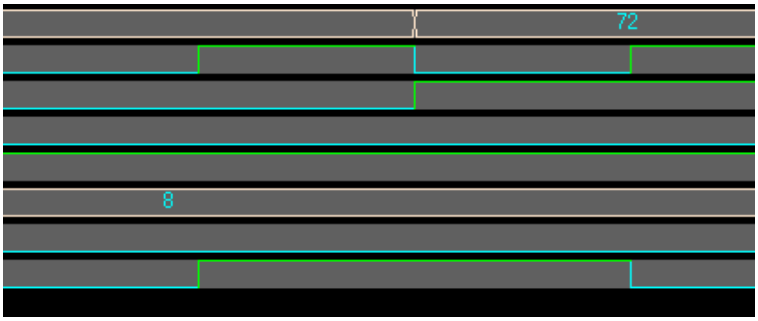


此時當 valid 拉起時，match 會是 1 且 match_index 為 2，如下圖所示。

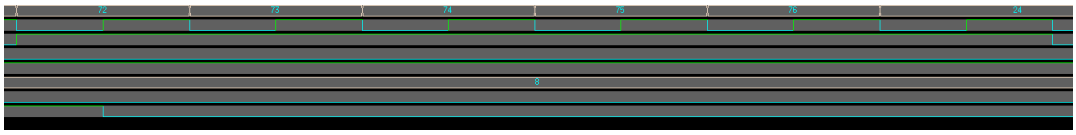
- 第七個 pattern : **^lmno**



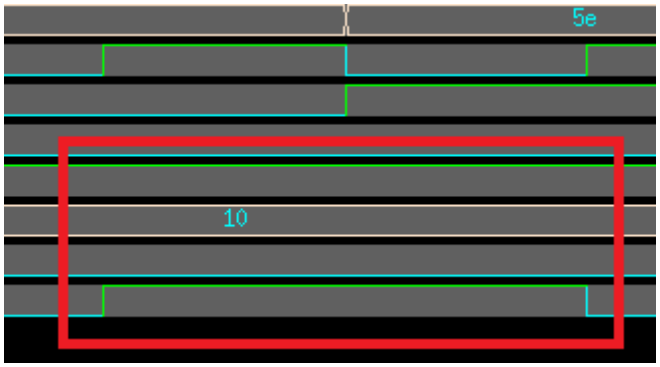
此時當 valid 拉起時，match 會是 1 且 match_index 為 8，如下圖。



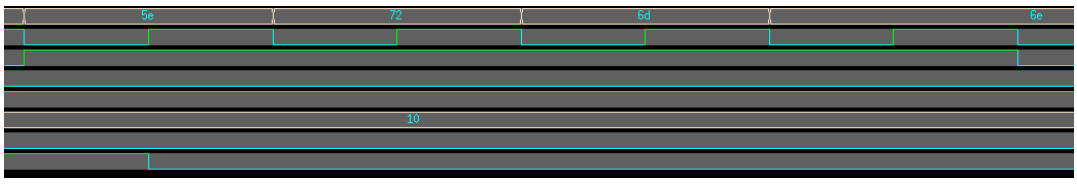
- 第八個 pattern : **rstuv$**



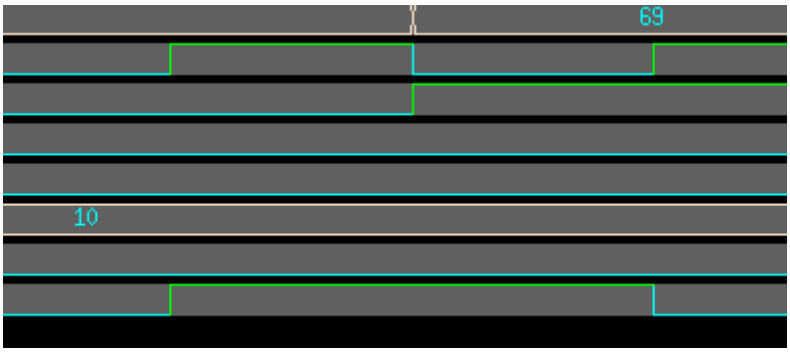
此時當 valid 拉起時，match 會是 1 且 match_index 為 10，如下圖。



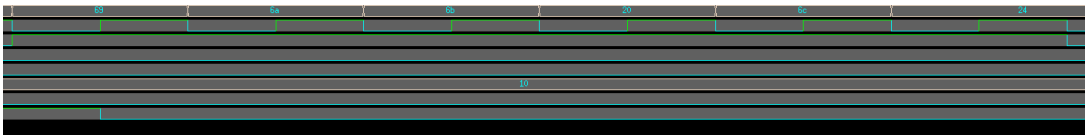
- 第九個 pattern : **^rmn**



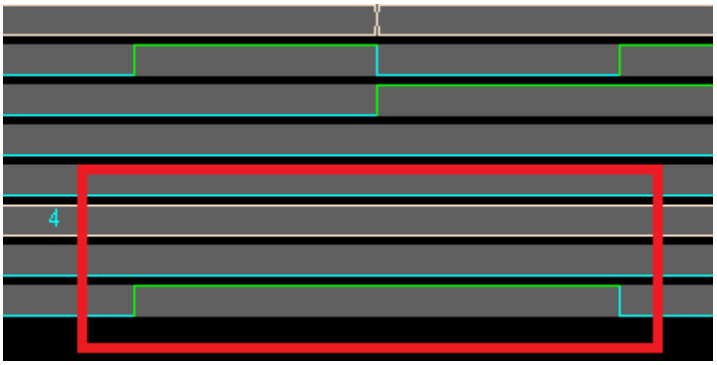
此時當 valid 拉起時，match 會是 0，如下圖。



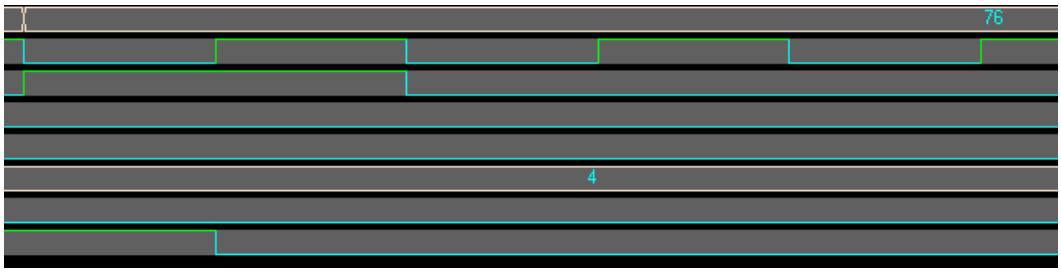
- 第十個 pattern : **ilk l$**



此時當 valid 拉起時，match 會是 0，如下圖。

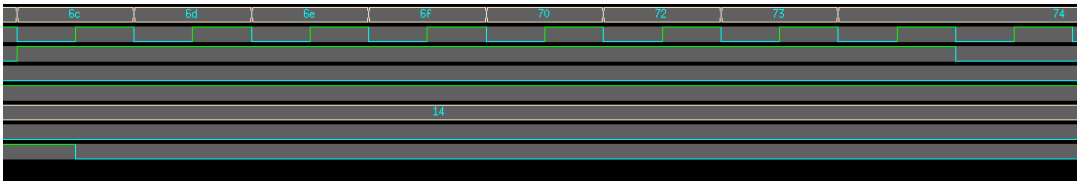


- 第十一個 pattern : **v**

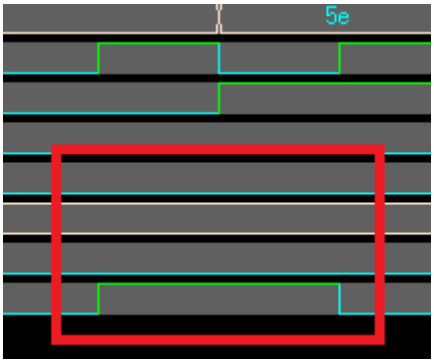


此時當 valid 拉起時，match 會是 1 且 match_index 為 14 (十六進位表示)，如下圖。

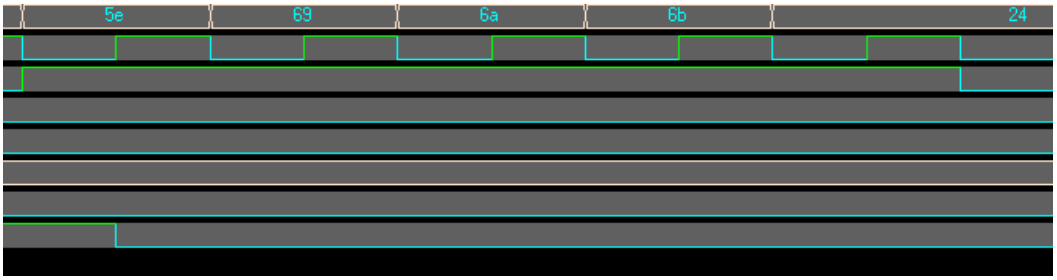
- 第十二個 pattern : **lmnoprst**



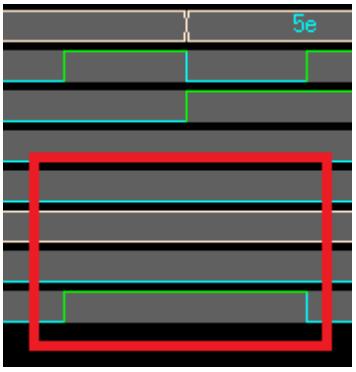
此時當 valid 拉起時，match 會是 0，如下圖所示。



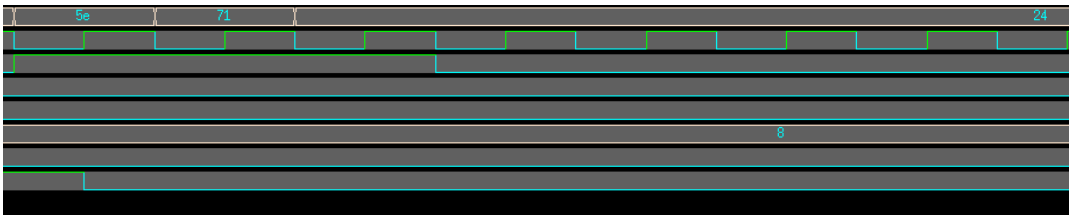
- 第十三個 pattern : **^ijk$**



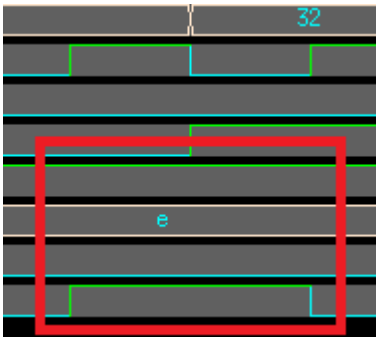
此時當 valid 拉起時，match 會是 0，如下圖所示。



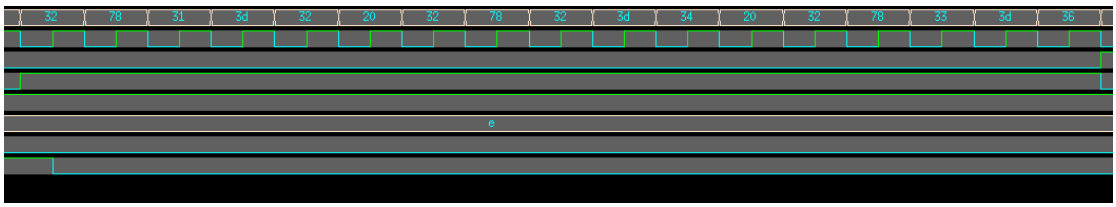
- 第十四個 pattern : ^q$



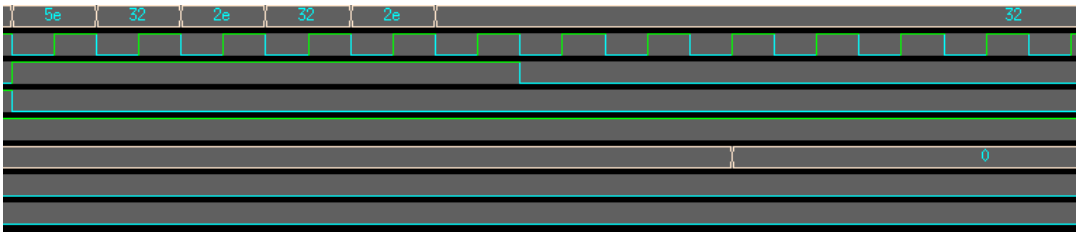
此時當 valid 拉起時，match 會是 1 且 match_index 為 e，如下圖。



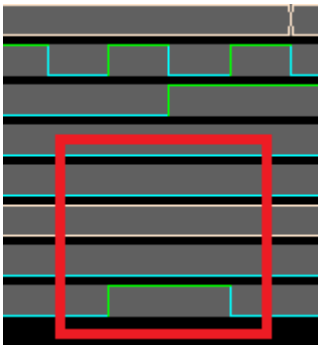
(2) 第二組 string : 2x1=2 2x2=4 2x3=6



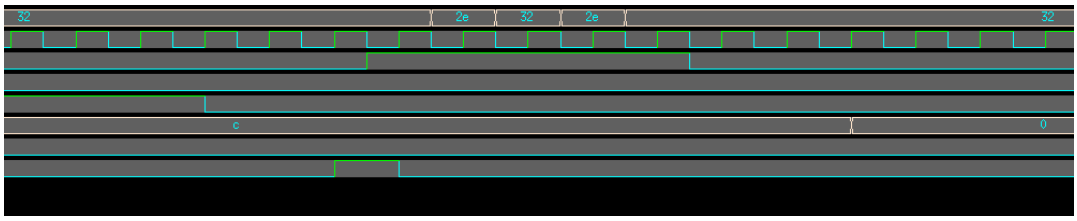
- 第一個 pattern : **^2.2.2**



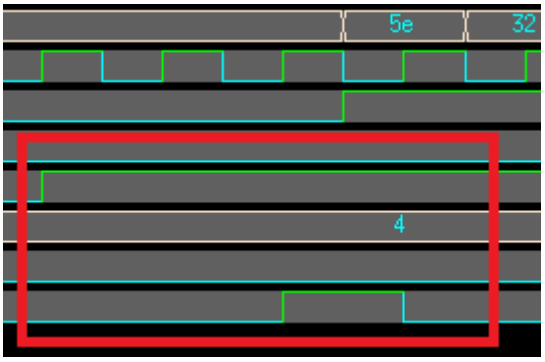
此時當 valid 拉起時，match 會是 0，如右圖所示。



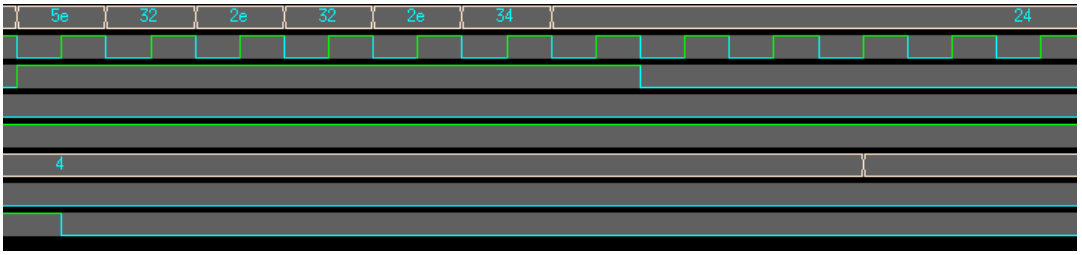
- 第二個 pattern : **2.2.2**



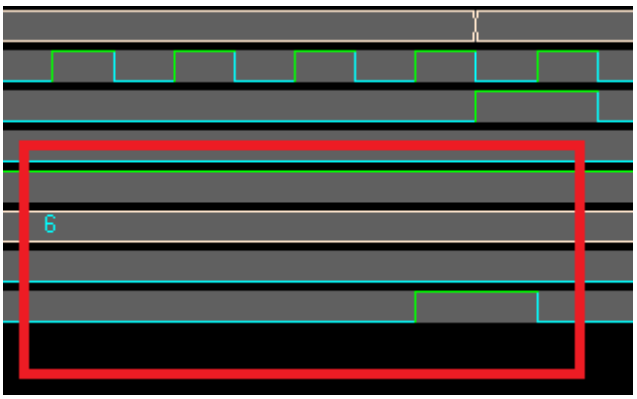
此時當 valid 拉起時，match 會是 1 且 match_index 為 4，如下圖所示。



- 第三個 pattern : **^2.2.4$**

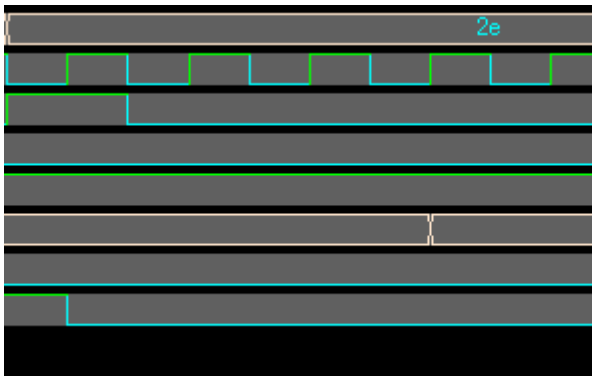


此時當 valid 拉起時，match 會是 1 且 match_index 為 6，如下圖所示。

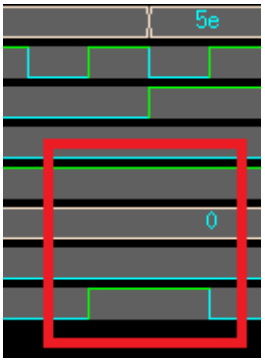


.

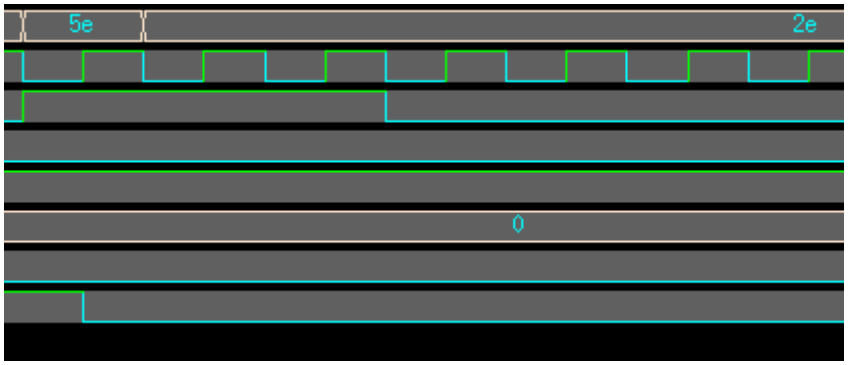
- 第四個 pattern : .



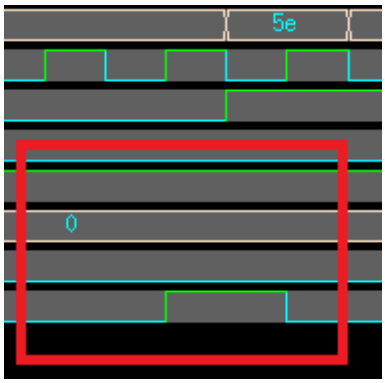
此時當 valid 拉起時，match 會是 1 且 match_index 為 0，如下圖。



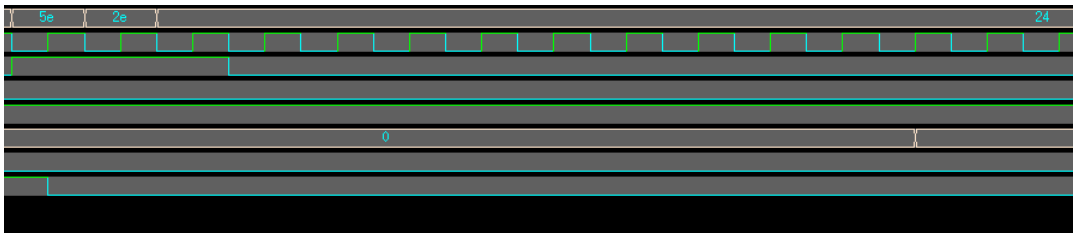
- 第五個 pattern : ^..



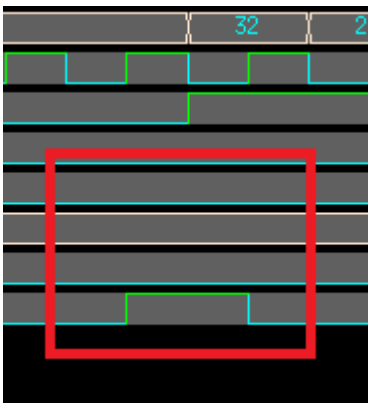
此時當 valid 拉起時，match 會是 1 且 match_index 為 0，如下圖所示。



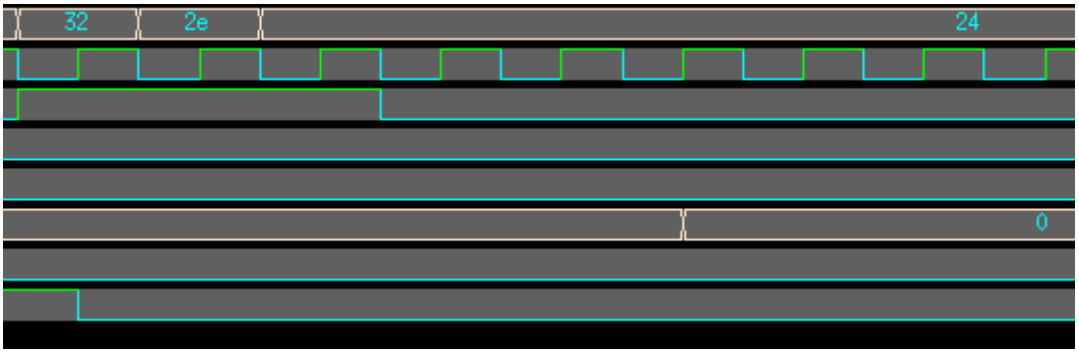
- 第六個 pattern : **^.$**



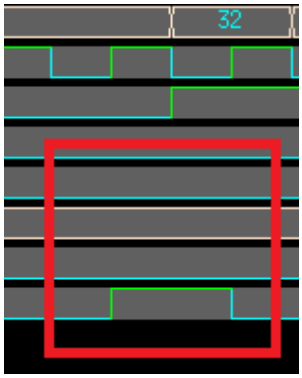
此時當 valid 拉起時，match 會是 0，如右圖所示。



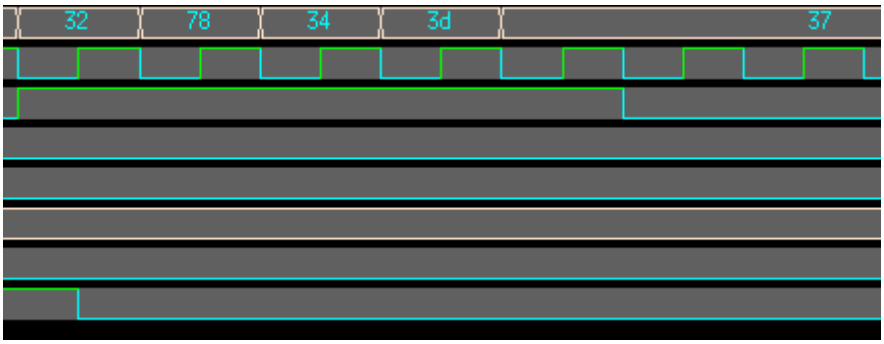
- 第七個 pattern : **2.$**



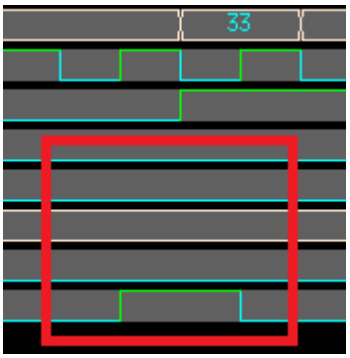
此時當 valid 拉起時，match 會是 0，如右圖所示。



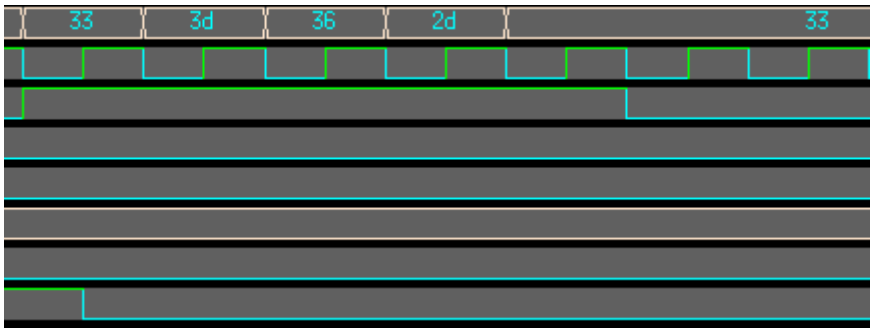
- 第八個 pattern : **2x4=7**



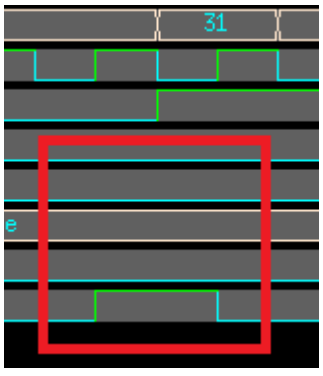
此時當 valid 拉起時，match 會是 0，如右圖。



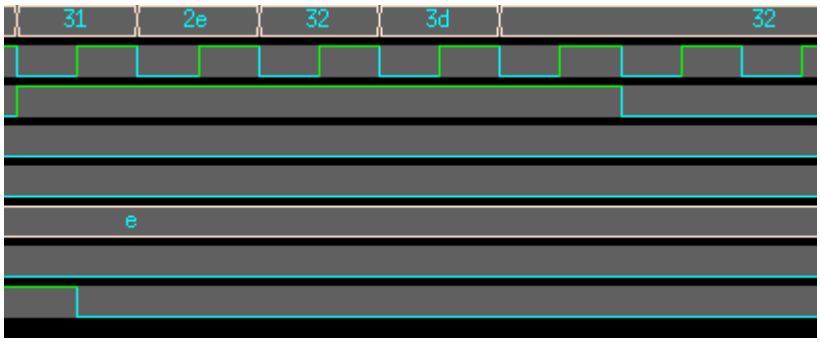
- 第九個 pattern : **3=6-3**



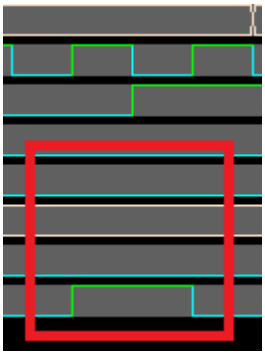
此時當 valid 拉起時，match 會是 0，如右圖。



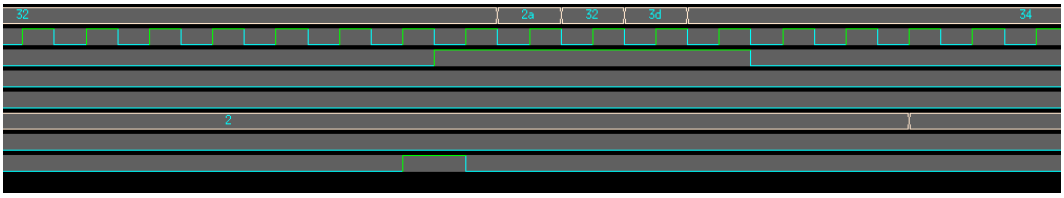
- 第十個 pattern : **1.2=2**



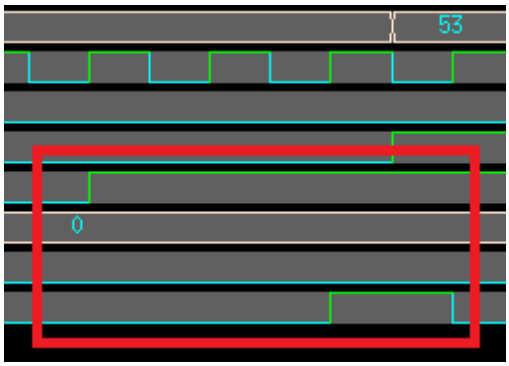
此時當 valid 拉起時，match 會是 0，如右圖。



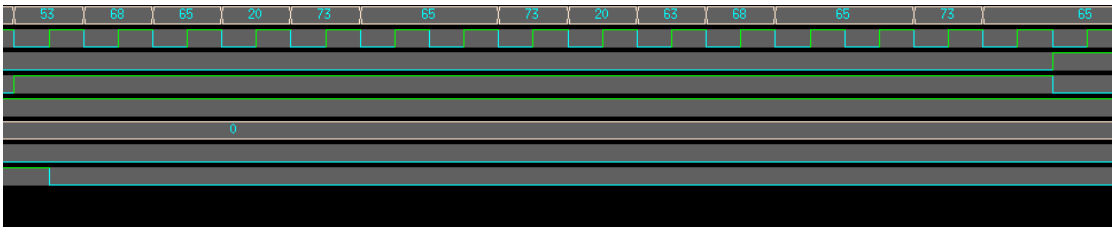
- 第十一個 pattern : **2*2=4**



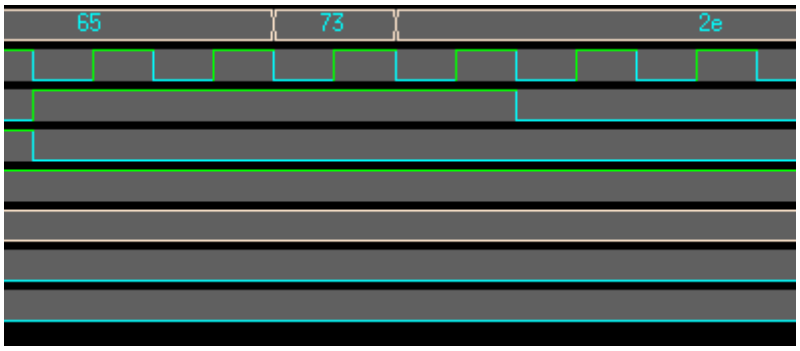
此時當 valid 拉起時，match 會是 1 且 match_index 為 0，如下圖所示。



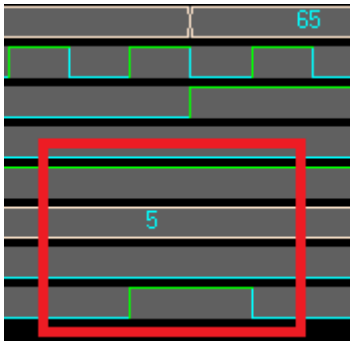
(3) 第三組 string : she sees cheese



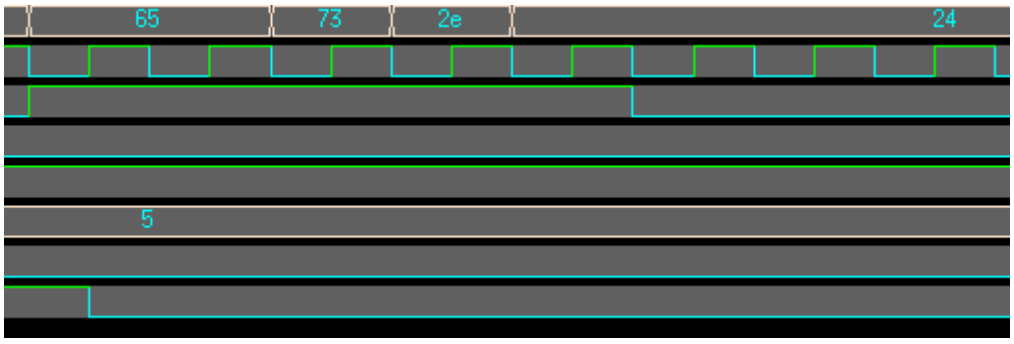
- 第一組 pattern : **ees.**



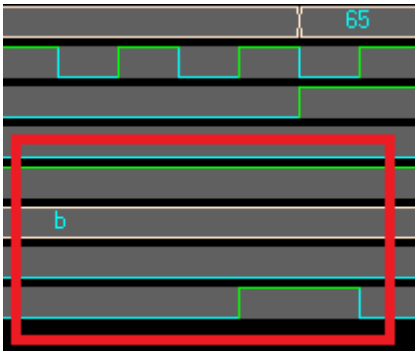
此時當 valid 拉起時，match 會是 1 且 match_index 為 5，如下圖所示。



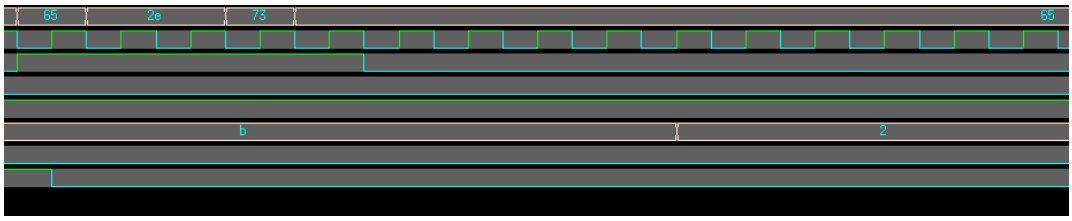
- 第二組 pattern : **ees.$**



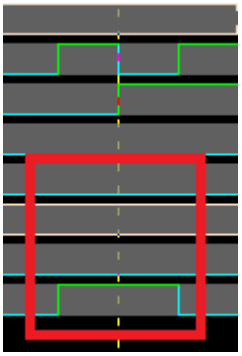
此時當 valid 拉起時，match 會是 1 且 match_index 為 b，如下圖所示。



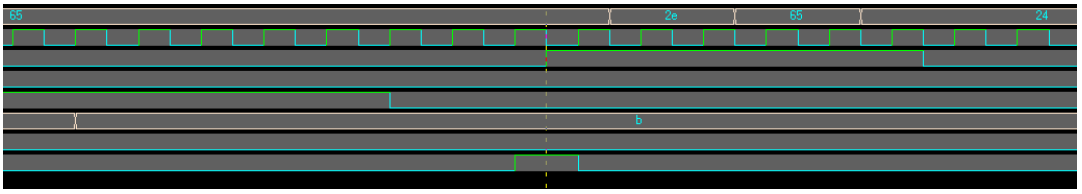
- 第三組 pattern : **e..se**



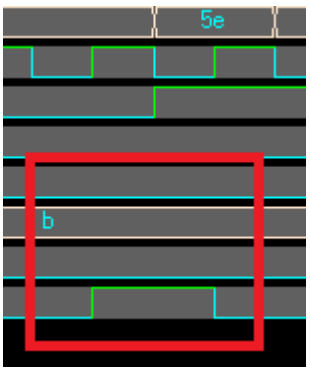
此時當 valid 拉起時，match 會是 0，如右圖所示。



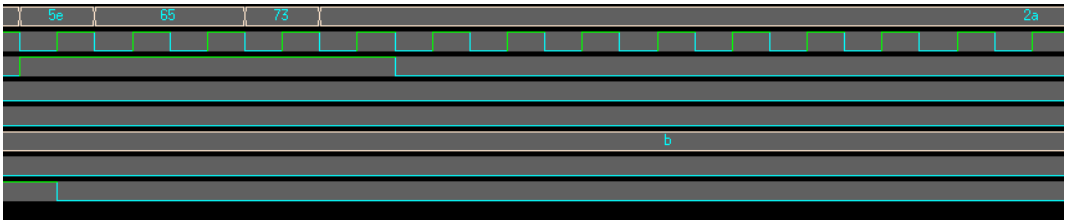
- 第四組 pattern : **e..ee$**



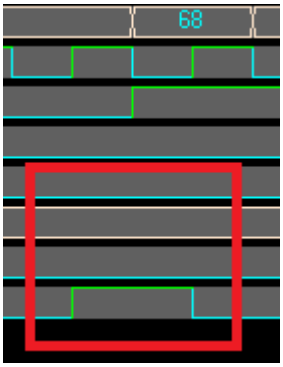
此時當 valid 拉起時，match 會是 0，如右圖所示。



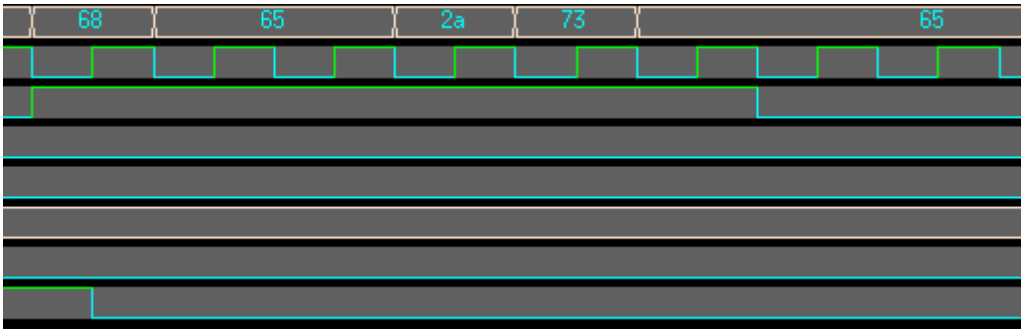
- 第五組 pattern : ^ees*



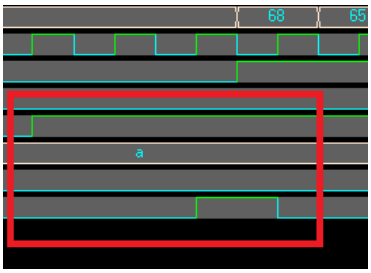
此時當 valid 拉起時，match 會是 0，如右圖所示。



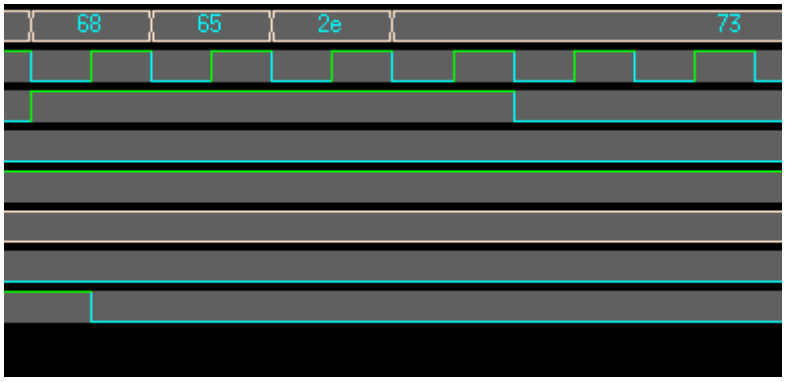
- 第六組 pattern : **hee*se**



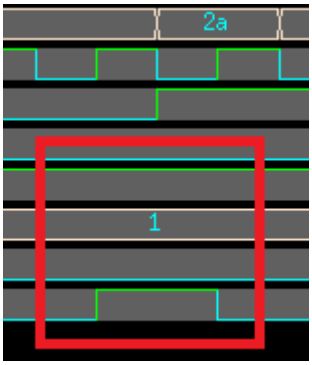
此時當 valid 拉起時，match 會是 1 且 match_index 為 a，如下圖所示。



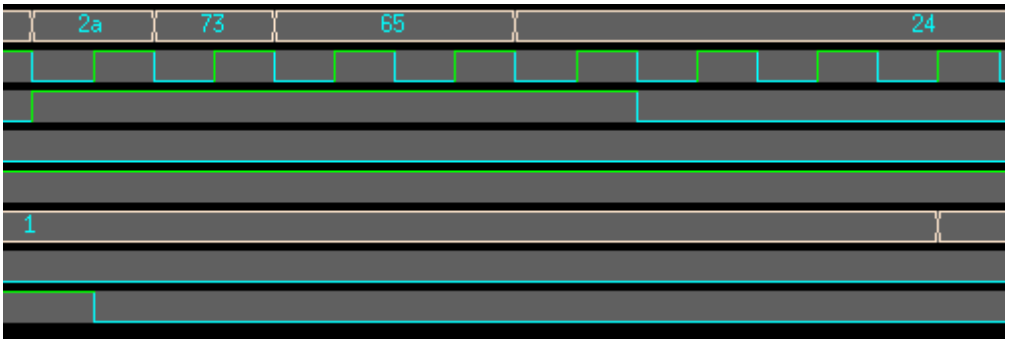
- 第七組 pattern : **he.s**



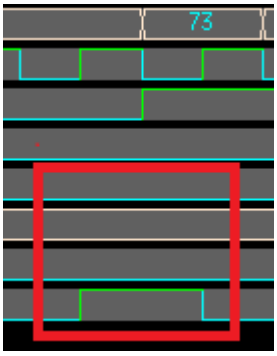
此時當 valid 拉起時，match 會是 1 且 match_index 為 1，如下圖所示。



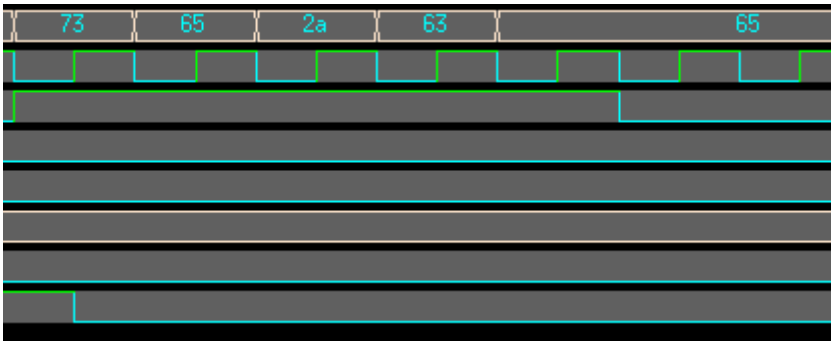
- 第八組 pattern : ***see$**



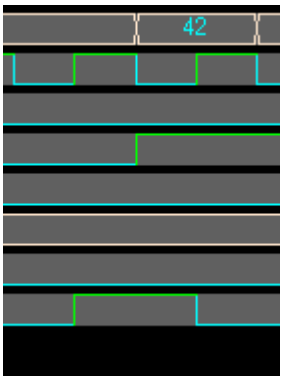
此時當 valid 拉起時，match 會是 0，如右圖所示。



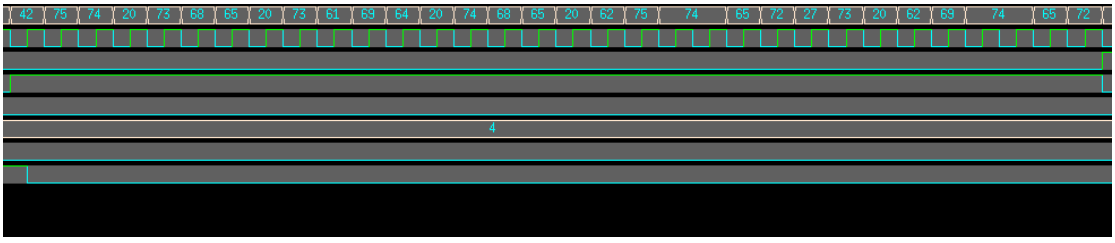
- 第九組 pattern : **se*ce**



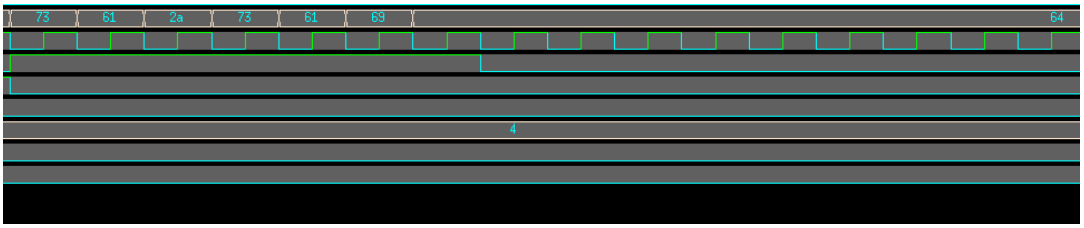
此時當 valid 拉起時，match 會是 0，如右圖所示。



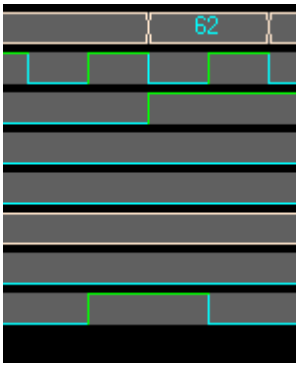
(4) 第四組 string : but she said the butter’s bitter



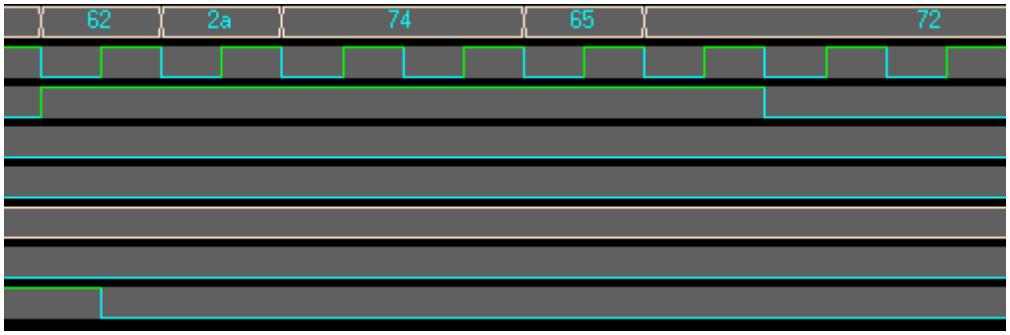
- 第一個 pattern : **sa*said**



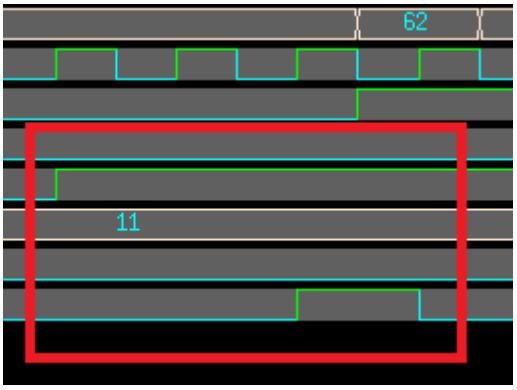
此時當 valid 拉起時，match 會是 0，如右圖。



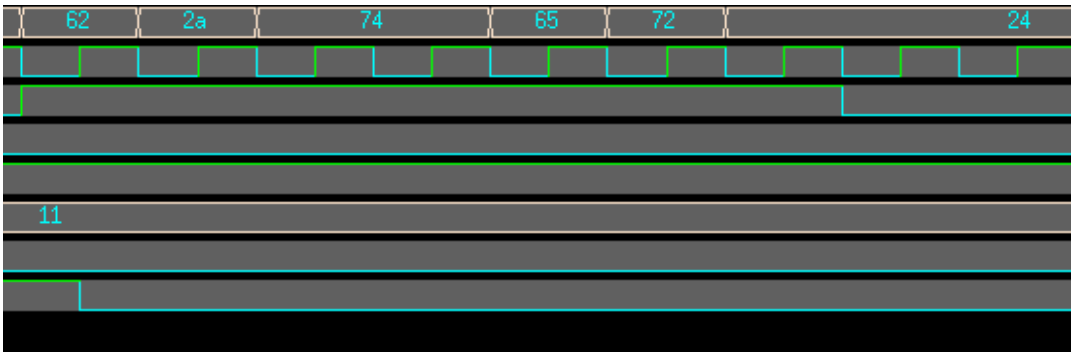
- 第二個 pattern : **b*tter**



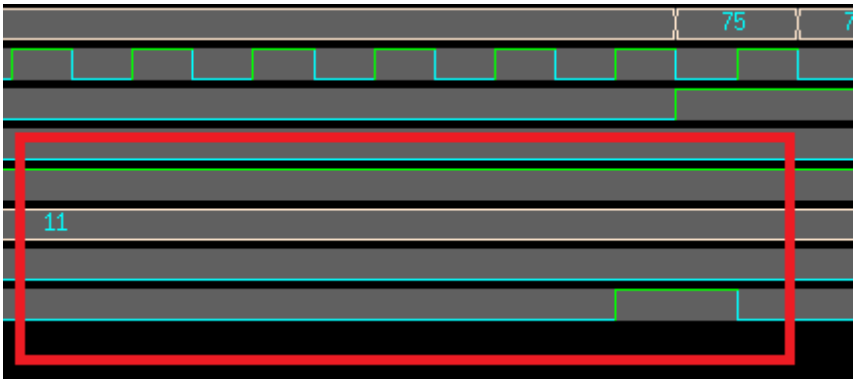
此時當 valid 拉起時，match 會是 1 且 match_index 為 11，如下圖所示。



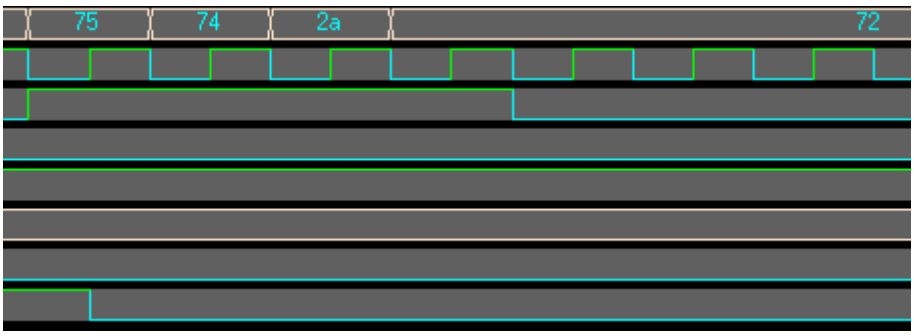
- 第三個 pattern : **b*tter$**



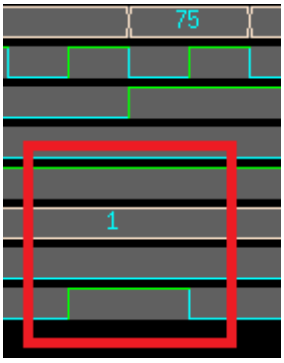
此時當 valid 拉起時，match 會是 1 且 match_index 為 11，如下圖所示。



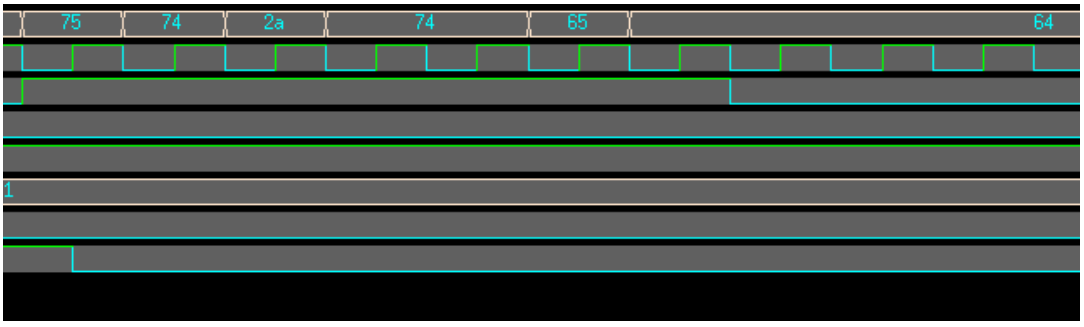
- 第四個 pattern : **ut*r**



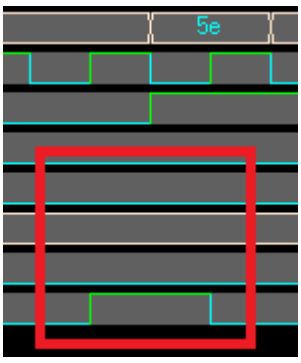
此時當 valid 拉起時，match 會是 1 且 match_index 為 1，如下圖所示。



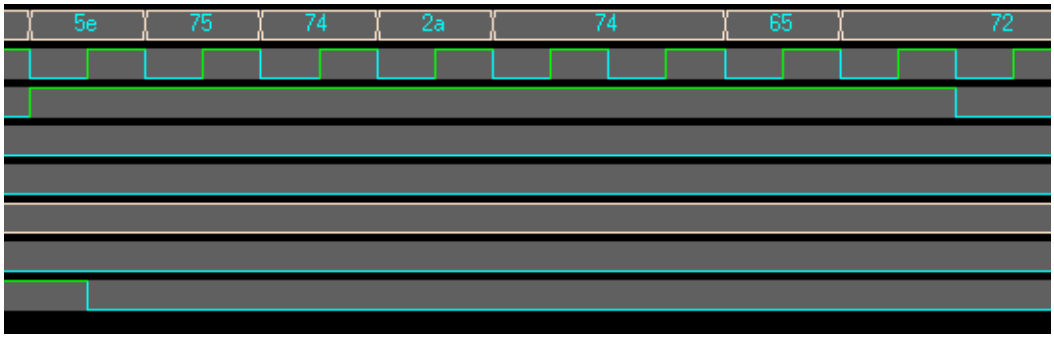
- 第五個 pattern : **ut*tted**



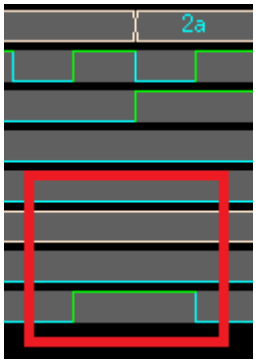
此時當 valid 拉起時，match 會是 0，如右圖所示。



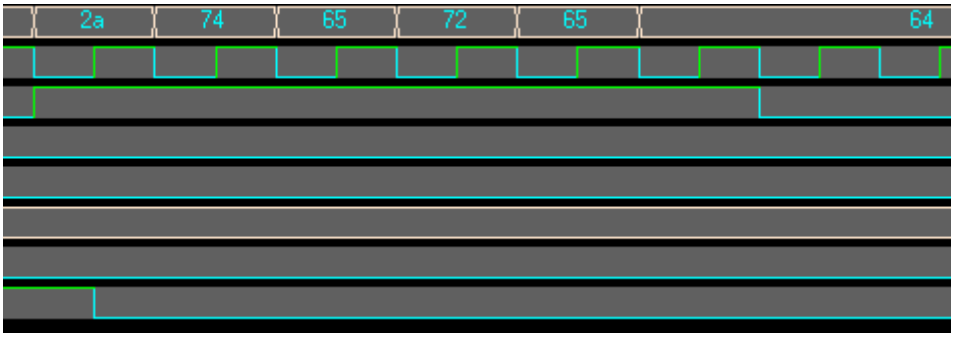
- 第六個 pattern : **^ut*tter**



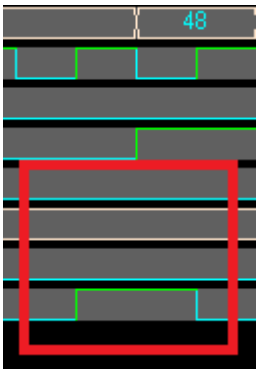
此時當 valid 拉起時，match 會是 0，如右圖所示。



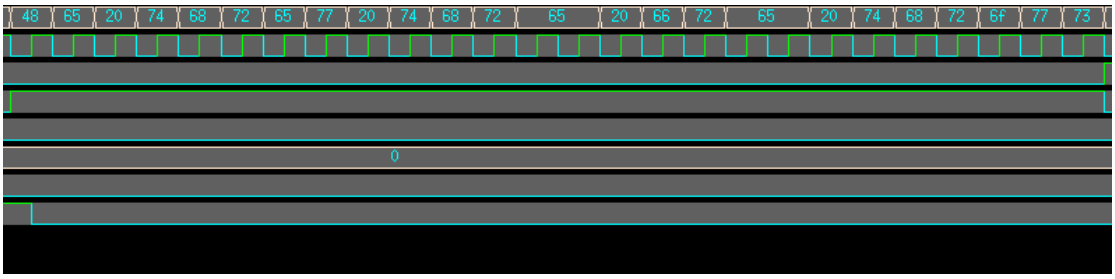
- 第七個 pattern : ***tered**



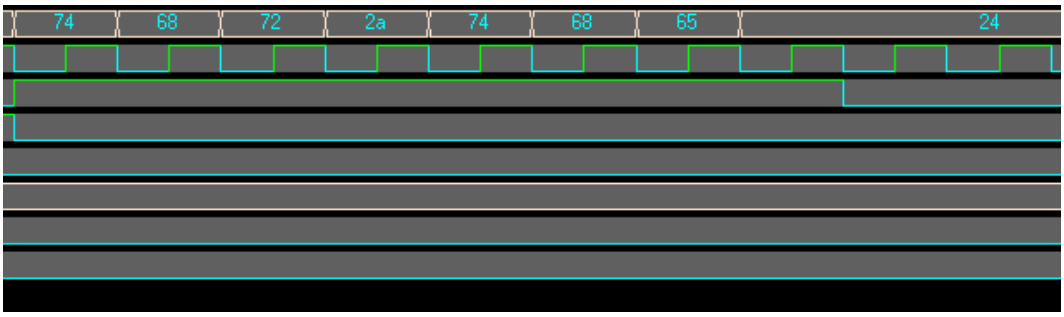
此時當 valid 拉起時，match 會是 0，如右圖所示。



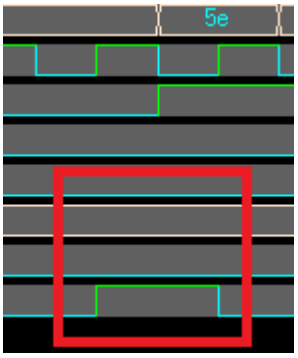
(5)　第五組 string : he threw three free throws



- 第一個 pattern : **thr*the$**

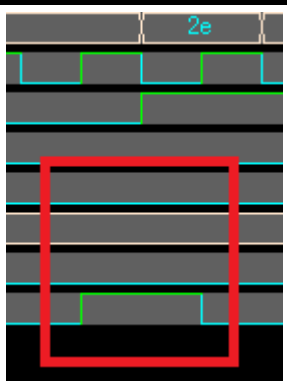


此時當 valid 拉起時，match 會是 0，如右圖所示。

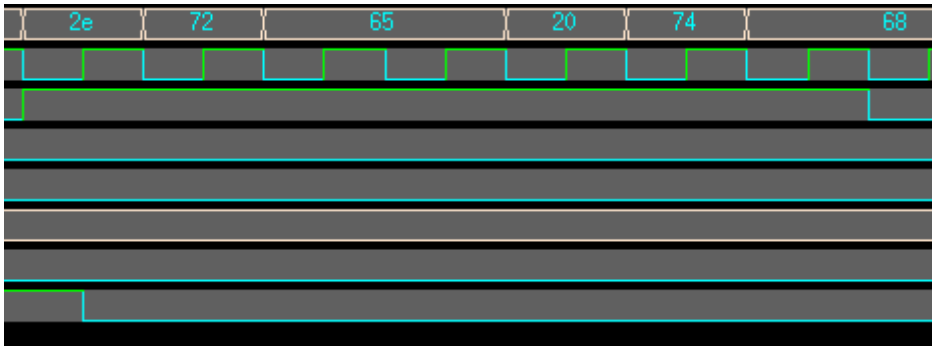


- 第二個 pattern : **^thro*th**

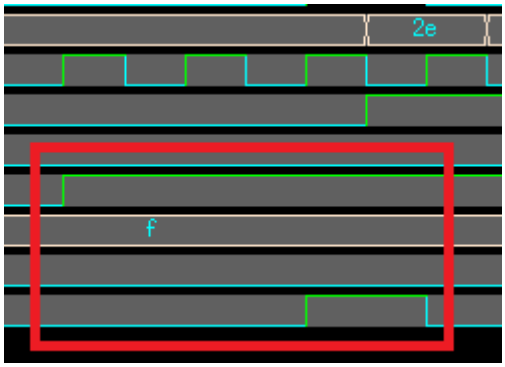
此時當 valid 拉起時，match 會是 0，如右圖所示。



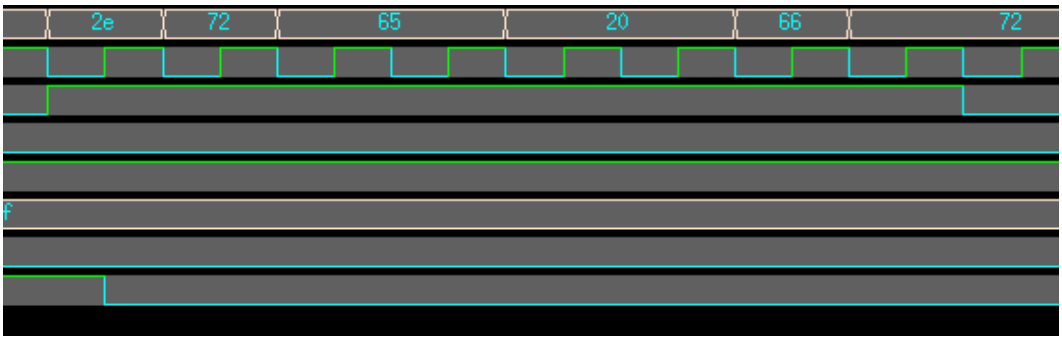
- 第三個 pattern : **.ree th**



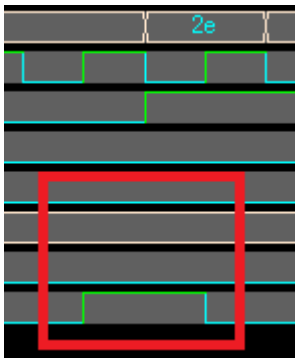
此時當 valid 拉起時，match 會是 1 且 match_index 為 f，如下圖所示。



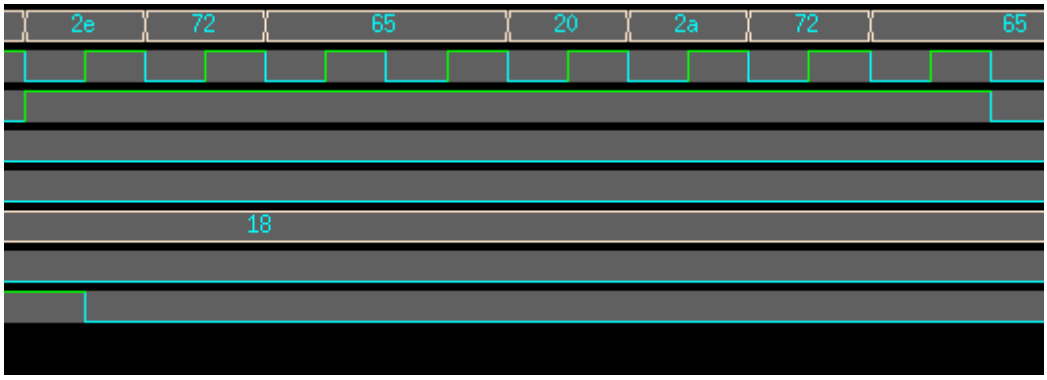
- 第四個 pattern : **.ree fr**



此時當 valid 拉起時，match 會是 0，如右圖所示。

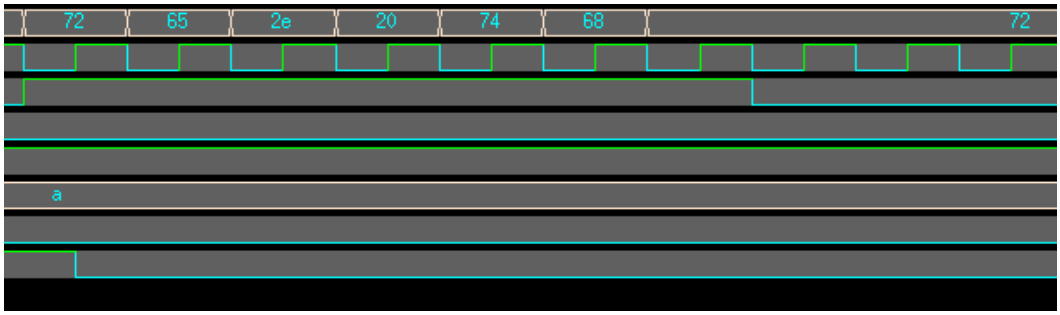


- 第五個 pattern : **.ree *re**

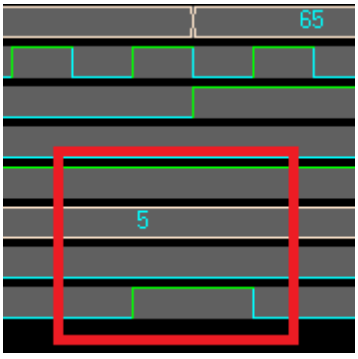


此時當 valid 拉起時，match 會是 1 且 match_index 為 a，如下圖所示。

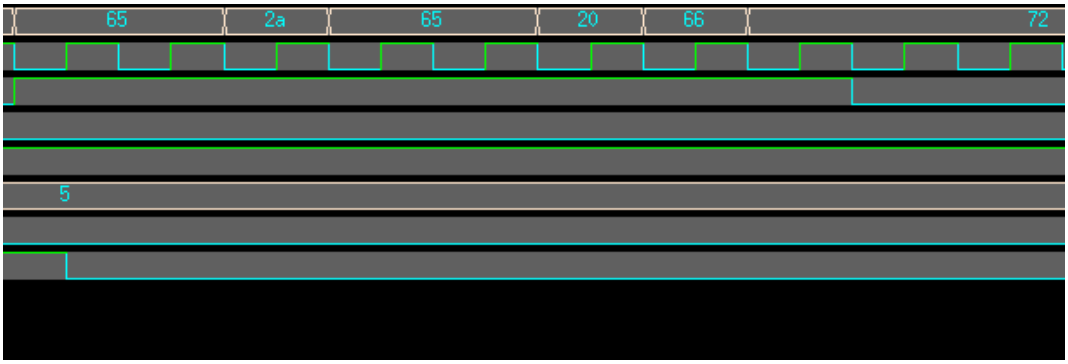
- 第六個 pattern : **re. thr**



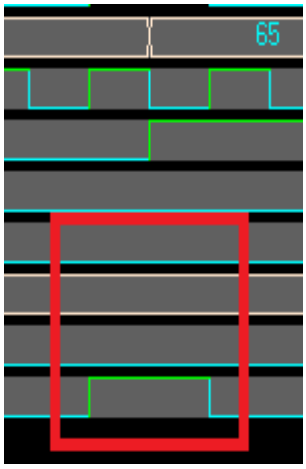
此時當 valid 拉起時，match 會是 1 且 match_index 為 5，如下圖所示。



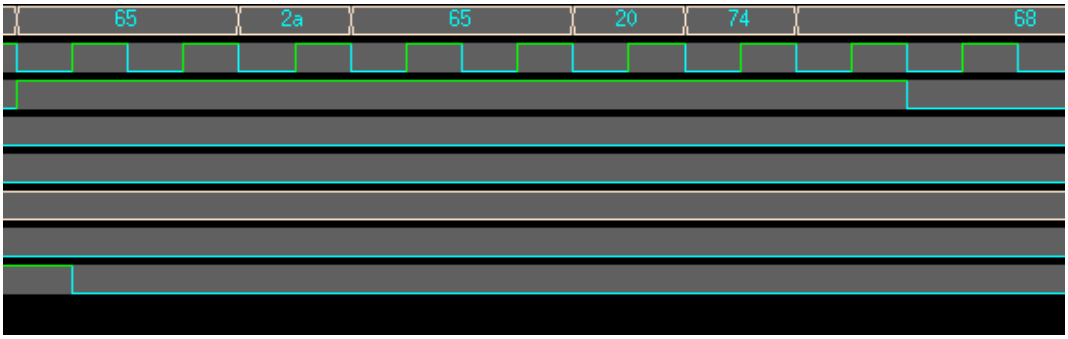
- 第七個 pattern : **ee*ee fr**



此時當 valid 拉起時，match 會是 0，如右圖所示。

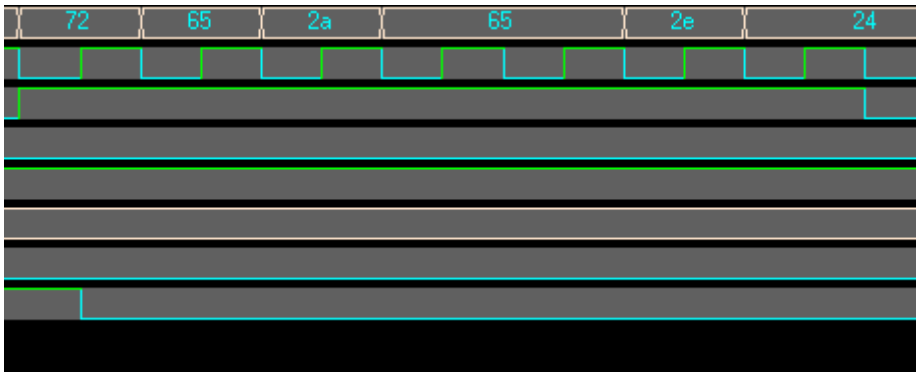


- 第八個 pattern : **ee*ee th**

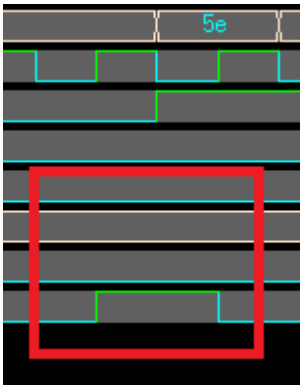


此時當 valid 拉起時，match 會是 1 且 match_index 為 c，如下圖所示。

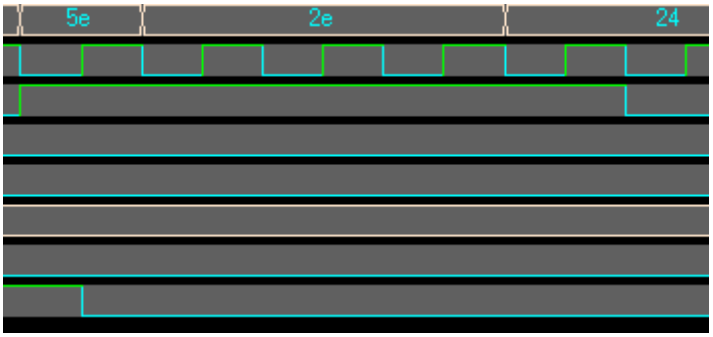
- 第九個 pattern : **re*ee.$**



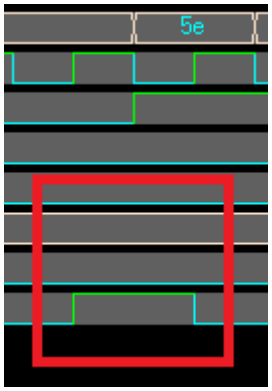
此時當 valid 拉起時，match 會是 0，如右圖所示。



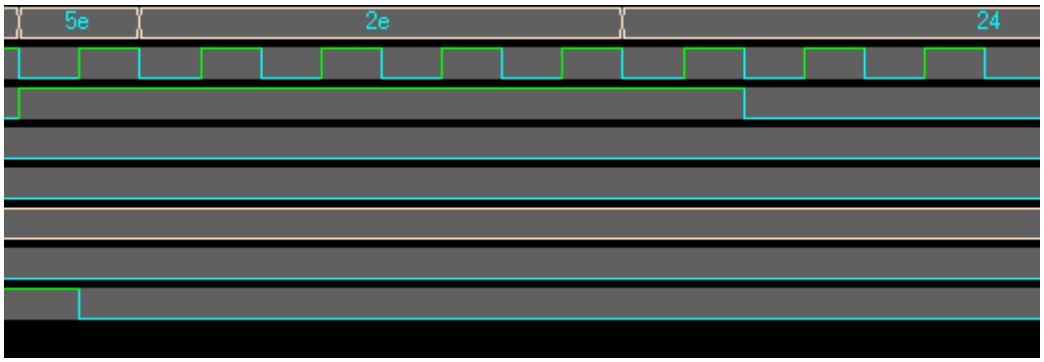
- 第十個 pattern : **^…$**



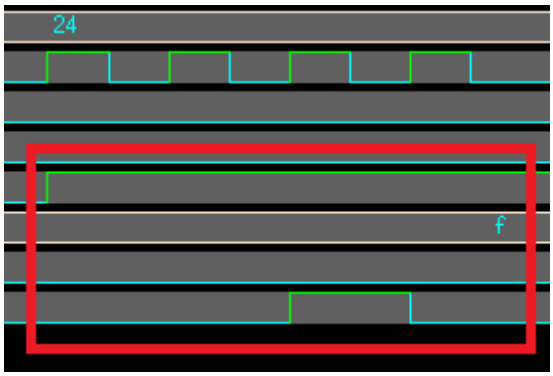
此時當 valid 拉起時，match 會是 0，
如右圖所示。



- 第十一個 pattern : ^....$



此時當 valid 拉起時，match 會是 1 且 match_index 為 f，如下圖所示。



四、 邏輯合成流程說明與合成參數(以最佳結果為例)

在 Xshell 輸入 dv &即可打開 Design Compiler，將 clock 設定為自己訂的週期 15ns，接著在 Design Compiler 指令欄中輸入 set_fix_multiple_port_nets -all -buffer_constants 避免合成出現可能的問題，即可完成電路合成(如下圖)，出現 Optimization Complete 代表合成完成，須注意是否有 latch，若有則代表 code 要修改為 flipflop。

Register Name	Type	Width	Bus	MB	AR	AS	SR	SS	ST
cs1_reg	Flip-flop	3	Y	N	Y	N	N	N	N
cs2_reg	Flip-flop	3	Y	N	Y	N	N	N	N

Register Name	Type	Width	Bus	MB	AR	AS	SR	SS	ST
match_index_reg	Flip-flop	5	Y	N	Y	N	N	N	N
cnt_m_temp_reg	Flip-flop	5	Y	N	Y	N	N	N	N
index_p_temp_reg	Flip-flop	5	Y	N	Y	N	N	N	N
finish_reg	Flip-flop	1	N	N	Y	N	N	N	N
index_s_reg	Flip-flop	6	Y	N	Y	N	N	N	N
pat_is_star_reg	Flip-flop	1	N	N	Y	N	N	N	N
index_p_reg	Flip-flop	5	Y	N	Y	N	N	N	N
cnt_m_reg	Flip-flop	5	Y	N	Y	N	N	N	N

Register Name	Type	Width	Bus	MB	AR	AS	SR	SS	ST
match_reg	Flip-flop	1	N	N	Y	N	N	N	N

Register Name	Type	Width	Bus	MB	AR	AS	SR	SS	ST
valid_reg	Flip-flop	1	N	N	Y	N	N	N	N

Register Name	Type	Width	Bus	MB	AR	AS	SR	SS	ST
str_reg_reg	Flip-flop	256	Y	N	Y	N	N	N	N

Register Name	Type	Width	Bus	MB	AR	AS	SR	SS	ST
cnt_s_reg_reg	Flip-flop	6	Y	N	Y	N	N	N	N

Register Name	Type	Width	Bus	MB	AR	AS	SR	SS	ST
pat_reg_reg	Flip-flop	64	Y	N	Y	N	N	N	N

Register Name	Type	Width	Bus	MB	AR	AS	SR	SS	ST
cnt_p_reg	Flip-flop	5	Y	N	Y	N	N	N	N

```
    0:00:06   24588.5      0.00       0.0
    0:00:06   24588.5      0.00       0.0
    0:00:06   24588.5      0.00       0.0
Warning: The set_dont_touch_network command
Loading db file '/mnt3/CBDK_IC_Contest_v2.1

Note: Symbol # after min delay cost means e

 Optimization Complete
 ---------------------
```

(1) Power

Dynamic power = **499.5753 uW**

Static power = **153.7063 uW**

```
****************************************
Report : power
        -analysis_effort low
Design : SME
Version: P-2019.03
Date   : Sun Jun 11 20:46:57 2023
****************************************


Library(s) Used:

    slow (File: /mnt3/CBDK_IC_Contest_v2.1/SynopsysDC/db/slow.db)


Operating Conditions: slow   Library: slow
Wire Load Model Mode: top


Global Operating Voltage = 1.08
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1pW


  Cell Internal Power  = 499.5753 uW   (76%)
  Net Switching Power  = 153.7063 uW   (24%)
                         ---------
Total Dynamic Power    = 653.2817 uW  (100%)

Cell Leakage Power     =  16.8070 uW
```

(2) Area

電路面積為 **23449.580629 um^2**， 約使用了 **4690** 個邏輯閘。

```
****************************************
Report : area
Design : SME
Version: P-2019.03
Date   : Thu Jun  8 16:02:05 2023
****************************************

Information: Updating design information... (UID-85)
Library(s) Used:

    slow (File: /mnt3/CBDK_IC_Contest_v2.1/SynopsysDC/db/slow.db)

Number of ports:                         19
Number of nets:                        1841
Number of cells:                       1661
Number of combinational cells:         1289
Number of sequential cells:             372
Number of macros/black boxes:             0
Number of buf/inv:                      205
Number of references:                    59

Combinational area:             11445.568222
Buf/Inv area:                    1286.629179
Noncombinational area:          12004.012407
Macro/Black Box area:               0.000000
Net Interconnect area:         250041.269440

Total cell area:                23449.580629
Total area:                    273490.850069

***** End Of Report *****
```

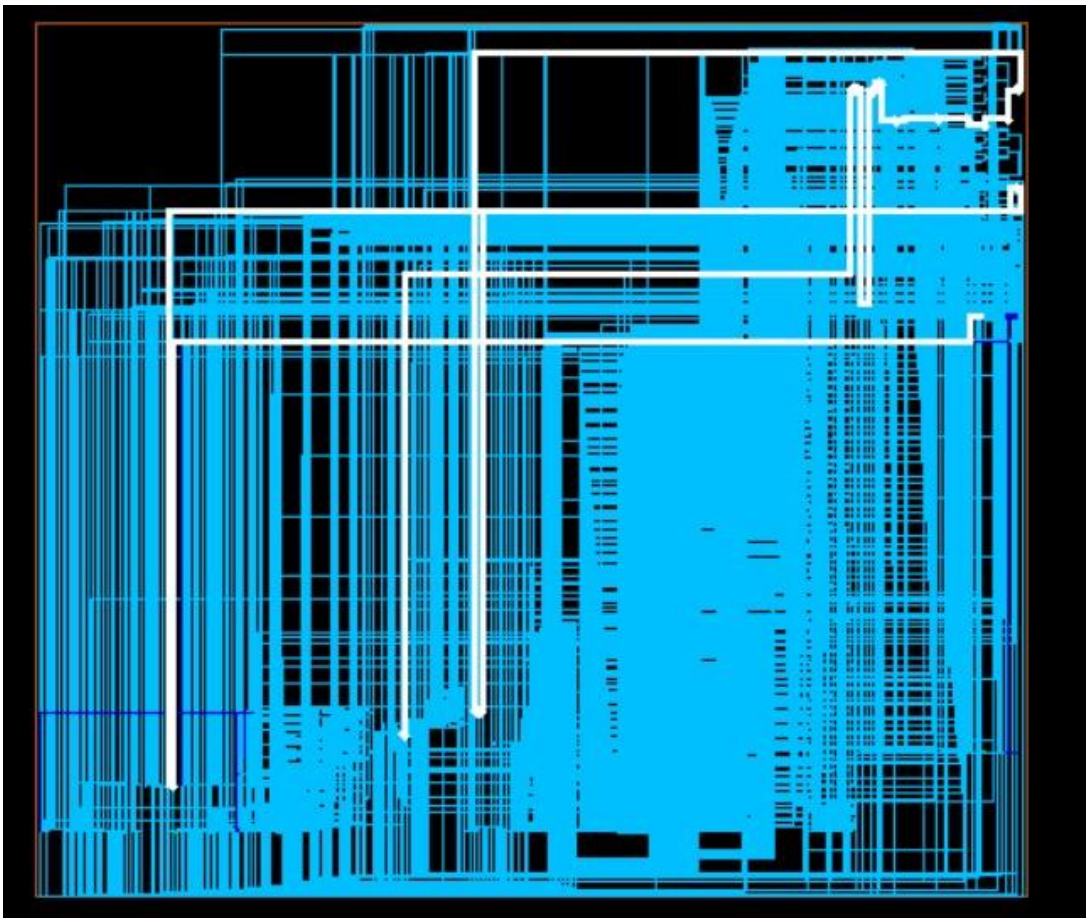
(3) Timing

以週期為 20ns 的 clk 為例，資料約在 **16.94ns** 時抵達，slack 約為 **3.17**。

由於此製程的 setup time 為 0.29，故最小 clk 可設置為 **6.94+2.09=7.23ns**。

Point	Incr	Path
clock clk (fall edge)	10.00	10.00
clock network delay (ideal)	0.50	10.50
input external delay	0.00	10.50 f
isstring (in)	0.01	10.51 f
U2609/Y (NAND2X1)	0.35	10.86 r
U2226/Y (CLKINVX1)	0.54	11.41 f
U1908/Y (OAI2BB2X2)	0.51	11.92 f
U1920/Y (CLKINVX1)	0.47	12.39 r
U2620/Y (XOR2X1)	0.33	12.72 f
U2617/Y (NOR3X1)	0.35	13.06 r
U2612/Y (NAND4X1)	0.34	13.40 f
U2225/Y (AOI211X1)	0.40	13.80 r
U2223/Y (NAND4X1)	0.34	14.14 f
U2608/Y (NOR3BXL)	0.37	14.52 f
U1945/Y (OAI21XL)	0.84	15.36 r
U2212/Y (AOI32X1)	0.51	15.87 f
U2102/Y (CLKINVX1)	0.53	16.40 r
U2637/Y (AOI22X1)	0.24	16.64 f
U2636/Y (OAI21XL)	0.30	16.94 r
index_s_reg[3]/D (DFFRX1)	0.00	16.94 r
data arrival time		16.94
clock clk (rise edge)	20.00	20.00
clock network delay (ideal)	0.50	20.50
clock uncertainty	-0.10	20.40
index_s_reg[3]/CK (DFFRX1)	0.00	20.40 r
library setup time	-0.29	20.11
data required time		20.11
data required time		20.11
data arrival time		-16.94
slack (MET)		3.17

(4) Critical Path



五、 邏輯合成與驗證結果

(1) 電路面積

```
****************************************
Report : area
Design : SME
Version: P-2019.03
Date   : Thu Jun  8 16:02:05 2023
****************************************

Information: Updating design information... (UID-85)
Library(s) Used:

    slow (File: /mnt3/CBDK_IC_Contest_v2.1/SynopsysDC/db/slow.db)

Number of ports:                           19
Number of nets:                          1841
Number of cells:                         1661
Number of combinational cells:           1289
Number of sequential cells:               372
Number of macros/black boxes:               0
Number of buf/inv:                        205
Number of references:                      59

Combinational area:              11445.568222
Buf/Inv area:                     1286.629179
Noncombinational area:           12004.012407
Macro/Black Box area:                0.000000
Net Interconnect area:          250041.269440

Total cell area:                 23449.580629
Total area:                     273490.850069

***** End Of Report *****
```

(2) Cycle 數目與 score

```
-------------------------------------
-- Simulation finish, ALL PASS  --
-- cycle =1860 , Score =100
-------------------------------------
```

(3) Score^3 / (電路面積*總 cycle 數目)

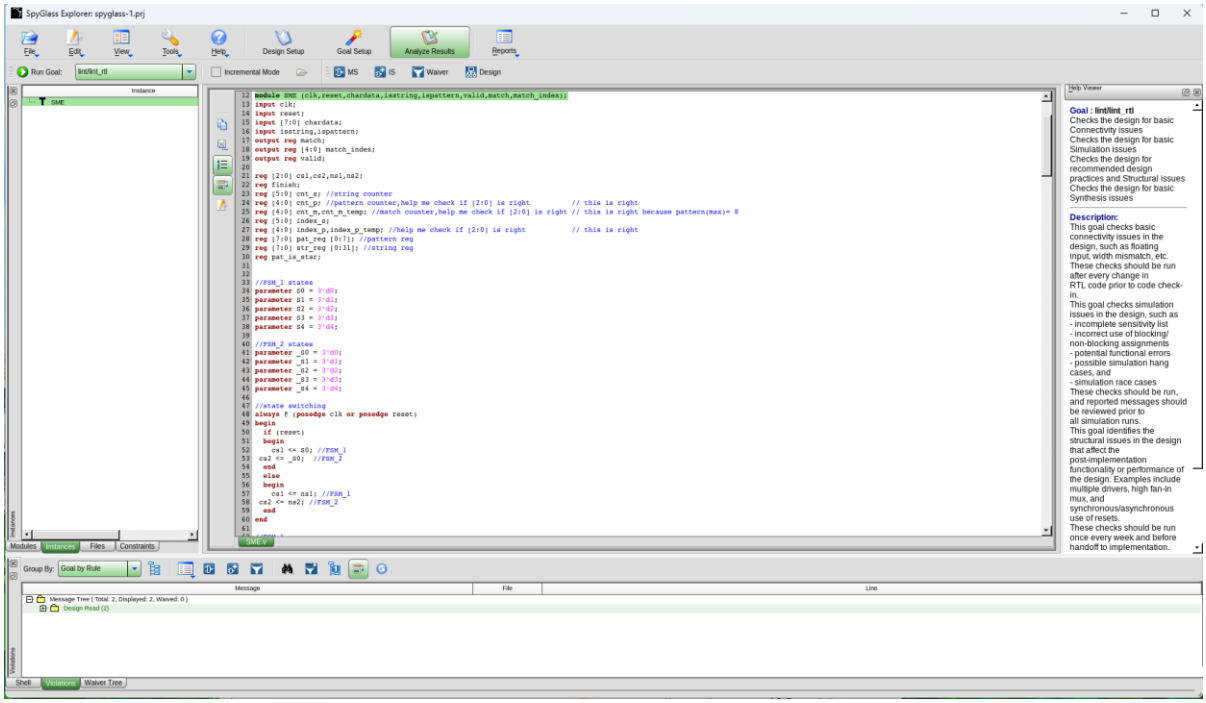
電路面積 ：23449.580629

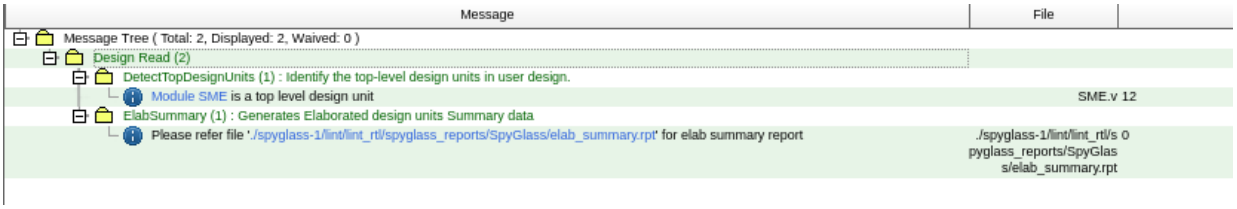
總 cycles : 1860

Score : 100

Score^3 / (area*cycles) = 100^3 / (23449.580629 * 1860) = 0.0229

六、 Spyglass 使用過程





我們第一次將我們的 code 丟進 spyglass 裡就沒有 error 或 warning 了，所以之後也沒有再針對 coding style 做調整。

七、 問題討論與反思

1. **硬體導向的設計 ：**

題目一下來我們就有到網路上查找許多實現的方法，但是我們發現到很多都是以軟體的角度去設計，因此我們這組有特別注意要用硬體的角度去設計。為了達到這個目的，我們主要是以 finite state machine 的角度來設計，雖然面積會比其他組來的大，但是這樣能確保合成後不會出問題，也是因為有特別注意 coding style 的部分，所以 spyglass 才沒有出現甚麼錯誤。

2. **Match_index :**

設計完電路並整合完成後，我們遇到的問題就是 match_index 永遠都是 0，但是其餘的功能都正確無誤，經過一些分析之後我們發現到我們在 S4 有一個初始化的動作，也就是在 match_index 會歸零，而問題在於我們也是在 S4 時做 match_index

輸出的判斷，所以無論如何輸出會是 0，最終我們將初始化的動作移至 S0，這樣問題就順利解決了。

3. **優化過程 ：**

最一開始我們是以減少 FSM 狀態的數量為優化的方向，但是經過了一番嘗試之後，我們發現減少狀態後有部分的 corner case 會出現錯誤，因此我們改變策略，朝其他方向進行優化。最初我們的面積為 25320，如下圖。

```
****************************************
Report : area
Design : SME
Version: P-2019.03
Date   : Thu May 25 13:21:48 2023
****************************************

Library(s) Used:

    slow (File: /mnt3/CBDK_IC_Contest_v2.1/SynopsysDC/db/slow.db)

Number of ports:                           19
Number of nets:                          2093
Number of cells:                         1733
Number of combinational cells:           1361
Number of sequential cells:               372
Number of macros/black boxes:               0
Number of buf/inv:                         95
Number of references:                      45

Combinational area:              13317.800149
Buf/Inv area:                      499.035591
Noncombinational area:           12002.314995
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:                 25320.115144
Total area:                 undefined

***** End Of Report *****
```

我們做了兩個部份的優化:

(1) Match_index :

一開始我們在 reset 或是狀態為 S0 的時候都會將 match_index 歸零，但是我們發現道題目有說當 valid 為 0 時，match_index 的值為多少都不用在乎，因此我們將 match_index 歸零的動作取消了。

(2) 比較器 ：

由於電路中用到許多比較器，因此我們嘗試使用邏輯閘來實現比較器的功能。最容易的就是一位元的比較器，只需要 XOR 就能完成，至於多位元的，我們嘗試寫了但是發現使用後 valid 的值會讀取不到，因此最終只有使用一位元的比較器。

做完以上優化後，再加上合成時將 area effort 設定成 high，我們的面積順利減少至 23449.58，如下圖

```
****************************************
Report : area
Design : SME
Version: P-2019.03
Date   : Thu Jun  8 16:02:05 2023
****************************************

Information: Updating design information... (UID-85)
Library(s) Used:

    slow (File: /mnt3/CBDK_IC_Contest_v2.1/SynopsysDC/db/slow.db)

Number of ports:                           19
Number of nets:                          1841
Number of cells:                         1661
Number of combinational cells:           1289
Number of sequential cells:               372
Number of macros/black boxes:               0
Number of buf/inv:                        205
Number of references:                      59

Combinational area:              11445.568222
Buf/Inv area:                     1286.629179
Noncombinational area:           12004.012407
Macro/Black Box area:                0.000000
Net Interconnect area:          250041.269440

Total cell area:                 23449.580629
Total area:                     273490.850069

***** End Of Report *****
```

4. **分工：**

我們這組一開始分工是將電路拆成不同區塊然後每人負責一個區塊，但是後來發現整合起來相當困難，每個人的訊號命名以及功能實現方法都略有不同導致我們在整合的時候花了不少時間，由這次的經驗我們深刻體會到下一次如果還有需要合作的專題，除了講好分工項目之外，更重要的其實是組員之間的溝通以及寫法上的統一。

5. **Testbench :**

我們這組有一個問題時我們缺少一個很會寫 testbench 的人，一開始看到助教給的 testbench 簡直一頭霧水，但是經過一段時間的研究我們發現到它只是引入了一個測試 pattern 的檔案，並將預期的答案與我們電路所判斷的答案最一個比較，並根據比較結果顯示出結果正確與否。除了看懂了這個 testbench 之外，這對我們來說是一個很好的自動化 testbench 的範例。

6. **自動化：**

這次助教給的相關檔案中有許多自動化的設計，像是上面提到的 testbench，以及邏輯合成時使用的 tcl 檔，都為我們的設計過程帶來很大的幫助，從這次期末專題中，除了更加熟練如何設計數位電路之外，我覺得很重要的收穫是看到了更厲害的工程師都是如何增加工作效率，讓我們這些初學者有更明確的目標。

7. **嘗試引用他組演算法**

此次的報告為小組小組的方式進行，因此會有很多的電路架構可以提供給我們參考，在眾多的演算法中，我們發現幾乎每個小組都會使用到多組的暫存器和狀態機，唯有終極夢幻屁眼派對那組不僅沒有使用到狀態機，而且使用到的暫存器組數相較大家來說減少許多，因此在完成我們這組的演算法時，我們首要就是想引用這組的演算法來試試看，但在寫完之後，發現 tb 在抓 match_index 和 match 可能有問

題，我們推想助教端的 tb 會在 valid=1 的時候抓 match_index 和 match 的值，但是在輸入 pattern 之後，必定會經過邏輯閘的運算才會有輸出產生，因此就算在 RTL_code 這邊看是沒有問題，但是在合成後就會有延遲效應的產生，一定會需要一些運算時間，我們推斷這樣的設計方式在合成後跑 TB 可能會出現錯誤，因此在最後因為時間考量就沒有再繼續研究下去。

雖然並沒有好的結果出現，但是這一組確實也提供了一個方向給大家繼續思考，而且這個演算法相較大家而言，是以水平的方式去做比較，是非常有效率的，而我們很開心能夠接觸到不同的演算法，透過接觸別組不同的演算法來累積設計電路的經驗，我們相信這是很好的共學方式。

八、 心得

B103012002 林凡皓 ：

這次期末專題算是我第一次與同學們一起合作設計一個電路，由於沒有什麼合作的經驗，所以不太清出分工、整合之類的該如何進行會比較有效率，一開始我們這組花了不少時間在討論這些事情，但是始終沒有什麼結果，最後是由組長那邊統一設計理念以及一些參數上的命名我們才能順利將這次的專題完成。

這次設計的電路比以往的作業都還要大許多，即便有分工，作業量還是比以往的作業大上不少，再加上撞到期末考週，全部事情擠在一起讓我有點手忙腳亂的，此時時間安排就顯得格外重要，將事情排開來並一件一件解決效率真的快上不少。這次期末專題，我從組員以及製作專題的過程中學習到了不少事情，像是更加熟悉數位電路設計的方法以及一些軟體的操作等，

除此之外，我覺得最大的收穫是了解到組員與組員之間的溝通以及合作的方法，還有就是有沒有一個大家公認的領導者也可以使整個團隊的運作更加順利，如果這次我們沒有選出一個公認的領導者，我們每個人的意見都不相同時根本無法進行整合以及統一，最終只會導致組員間關係惡化以及無法完成作品的悲慘結局。

由這次製作期末專題的經驗，我會記取一些失敗與不順利的教訓，還有發覺自己的不足以及應該改進的部分，不管是設計電路上的技巧與細心，還是團隊合作的方法與溝通，這些都會是我在未來要努力進步的地方。

B103011043 林祐葳 ：

這次期末專題對我來說是一個全新的挑戰，因為它完全不同於以往的作業。專題的內容非常複雜且龐大，無法獨自完成，所以需要幾個人共同合作來完成它。然而，當我們進行朋友分組時，出現了一些問題，導致我被分到了一個完全不認識的組別。起初，我感到有些害怕和不安，因為我不知道能否和這些陌生的組員順利合作。

然而，隨著我們開始實際接觸並進行專題的設計和製作，我驚喜地發現，我的組員都是非常棒和優秀的人才。他們在電路設計和相關領域都有豐富的知識和經驗，並且願意分享和協助彼此。這使我逐漸克服了最初的不安和緊張情緒，並開始積極參與團隊的討論和工

作。

然而，我也意識到自己對這門課程的理解還不夠深入，這導致我的思路經常跟不上組員的討論。在討論中，我常常感到困惑和無助，無法貢獻出有價值的意見和建議。這使得我們的討論在某種程度上缺乏實質性的進展，這也是我感到一些挫敗感的原因。

然而，我很幸運地有一位組長在團隊中發揮了領導的作用。組長有著清晰的設計理念和組織能力，他能夠有效地分配任務，統一設計方向，並確保我們在進行專題時能夠順利前進。組長的存在讓我們能夠更好地協調和整合各個成員的貢獻，進而提高了整個團隊的效率和成果。

在專題製作的過程中，我遇到了許多之前獨自完成作業時沒有遇到的挑戰。其中，我最明顯的感受是在撰寫程式碼方面，我意識到與其他組員的合作和協作是不同的。在過去，我習慣於獨立思考和撰寫程式碼，只需要考慮自己的邏輯和風格。然而，這次與組員合作的過程中，我需要適應並理解他們的程式碼風格和編碼習慣。這可能需要一些時間和調整，因為每個人都有自己獨特的寫作風格和習慣。

然而，我很幸運地發現我的組員們都具有良好的程式碼習慣和風格。他們撰寫的程式碼結構清晰、易讀且易於理解。這讓我能夠相對輕鬆地理解並與他們的程式碼進行交互和協作。儘管初期需要一些適應，但組員間的程式碼交流和協作過程相當順暢，這讓我非常感激並欣賞他們的專業態度和團隊合作能力。

因此，我想要特別感謝兩位優秀的組員，他們在專題的最後優化階段做出了巨大的貢獻。他們的專業知識和技能使我們的專題更加完善和高效。他們的努力和才智不僅在技術上提升了專題的質量，也啟發了我對於電路設計和程式開發的深入思考。

這次期末專題的經驗對我來說是一個寶貴的學習機會。它不僅加深了我對於數位電路設計方法和軟體操作的理解，也讓我認識到了團隊溝通和合作的重要性。我意識到，即使在個人技能方面還有不足之處，通過團隊合作和相互學習，我們可以彼此補充，實現共同目標。

總而言之，這次期末專題是一個難忘且寶貴的經驗，讓我學到了很多。從技術方面的數位電路設計到團隊合作和溝通能力的提升，這些都將對我未來的學習和職業發展產生重要影響。我將牢記這次經驗所帶來的教訓，並持續改進自己的能力和技巧，以成為一名更出色的團隊成員和優秀的程式開發者。

B103012001 陳胤琲

這次期末報告是一個相當龐大的電路，和以往作業的電路不同，不是單單幾個邏輯閘和狀態機就能達成，而是需要細心的將各個子電路合成成一個大電路，因此在設計、整合所花的時間，一定會相較前幾次多了許多，尤其是在整合的方面。

我第一次看到專題題目時有一點錯愕，畢竟是全新沒看過的東西，不但是之前數位系統課程沒上過的，也是硬體描述語言沒看過的，代表我接觸各個領域電路的經驗還不足，還得要多多耕耘，而第一次見到這個電路，就是要先研究字串搜尋的各個演算法，在這邊我很感謝我的組員在這邊查到各種演算法給我們討論，讓我們有一個方向能夠繼續做下去，有了演算法之後，再來就是開始設計電路，從這裡開始就可以發現電路所需要的暫存器已經變了許多，子電路也變了很多，想必在到時候整合各組員寫的電路時，一定會是一個很大的挑戰。

果不其然在整合的時候，花了我們許多時間，首先要熟悉每個人的 coding style，再者要檢查好個電路溝通的 port 有沒有出錯，對於這麼大的電路來說，都是相當花時間的，因此如同上課時說的一樣，在不同人完成不同的子電路時，大家一定要有個共識，才不會到最後搞得人仰馬翻，花了一大堆時間在 debug。

期末時我學到了分配時間的重要性，由於這次期中進度卡到期末周，代表當時在製作報告時時間是非常緊湊的，不過還好當時盡早處理完期中進度報告的內容，才在當時能完成期中進度報告又不會拖延到複習其他期末考的內容。

最後透過這次小組做專題的經驗，不論是從他組學習好的設計方法，或是檢討整合電路的不順暢，我認為都是寶貴的經驗，也希望這些經驗能夠深深的拓印在我腦海裡，在未來不論是其他課程或是到了職場，都能讓我做出正確的行動。

九、 參考資料

https://www.geeksforgeeks.org/differences-between-synchronous-and-asynchronous-counter/