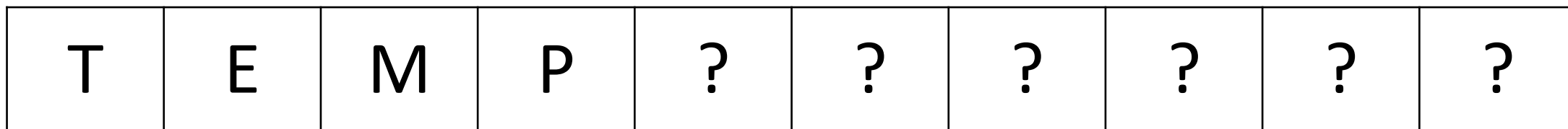


SME

# 基礎想法

把要算的pattern 存進  
register裡，假設patter是  
TEMP



然後把最後一個  
有pattern的  
address存下來

# 基礎想法

- 把文本照順序讀進來
  - 假設是文本是"TEMP"

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| T | E | M | P | ? | ? | ? | ? | ? | ? |
|---|---|---|---|---|---|---|---|---|---|

用一組flag來記錄是否match，詳見後續例子

[illegible]



# 基礎想法

文本輸入:E

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| T | E | M | P | ? | ? | ? | ? | ? | ? |
|---|---|---|---|---|---|---|---|---|---|

如果輸入和儲存的pattern相符，而且前一組flag是1，就把flag拉1

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|

T!=E，所以"flag 0"現在要拉回0

# 基礎想法

文本輸入:M

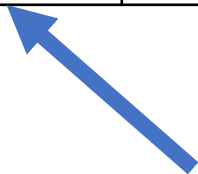
|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| T | E | M | P | ? | ? | ? | ? | ? | ? |
|---|---|---|---|---|---|---|---|---|---|

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|

# 基礎想法

文本輸入:P

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| T | E | M | P | ? | ? | ? | ? | ? | ? |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |



如果最後一個有pattern的地址的flag是1，就把valid拉成1，並算出對應的輸出address

# 特殊規則處理

- Case insensitive
  - T\_data, P\_data輸入先擋一層flip-flop，如果Case insensitive是1，就把存進去的大寫英文全部改成小寫



# 特殊規則處理

- 特殊符號“.”
  - 在flag比對時，如果儲存的pattern是“.”，而且輸入不是換行符號的話，也算pattern match

# 特殊規則處理

- 特殊符號“\$”
  - 在存pattern時，改成存對應的換行符號“0A”

# 特殊規則處理

- 特殊符號“^”
  - 在存**pattern**時，改成存對應的換行符號“0A”
  - 因為規則上換行符號不算**pattern**的開頭，如果**pattern**第一個符號是“^”的話，之後輸出**address**要加1

# 特殊規則處理

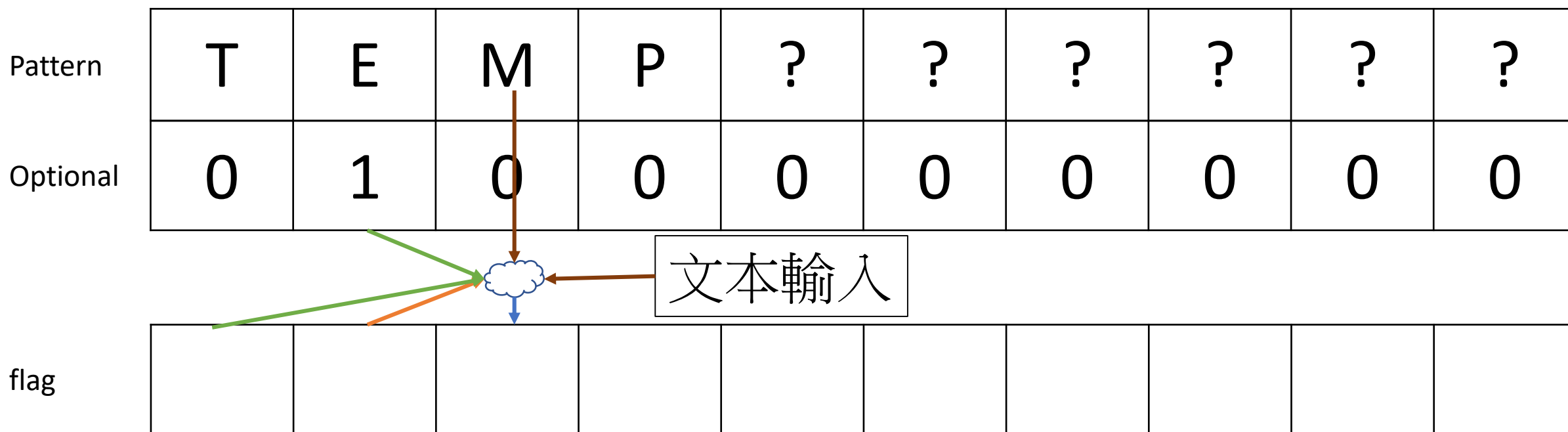
- 特殊符號“?”
  - 並不把“?”存成一個pattern，而是多開一組1bit的register來記錄前一個pattern是不是optional
  - 假設pattern是TE?MP

|          |   |   |   |   |   |   |   |   |   |   |
|----------|---|---|---|---|---|---|---|---|---|---|
| Pattern  | T | E | M | P | ? | ? | ? | ? | ? | ? |
| Optional | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# 特殊規則處理

- 特殊符號“?”

- 在flag比對時，如果前一個pattern是optional，會看再前一個pattern是否相符
- 最多往前比三層，因為題目說?最多只會連續兩個



# 特殊規則處理

- 特殊符號“?”
  - 如果因為optional，中間pattern有跳號的話，輸出address也要有對應的改變
  - 直觀的想法是多開一組register，把初始address按照flag傳遞
    - 第0格裡面會存potential match在文本裡的位置
    - 跳兩格的時候，就把前三格存的address拉過來
    - 跳一格的時候，就把前兩格存的address拉過來
    - 沒有跳的時候，就把前一格存的address拉過來
  - 但是這樣需要多開 $12 \times 16 = 192$ bit的register
    - 我的design最大的array只有 $8 \times 16$ 的pattern array
    - 比例上來說，我覺得sequential circuit的面積會變大很多，所以沒有採用這個做法

# 特殊規則處理

- 特殊符號“?”
  - 題目有提到最多只會有四個“?”
    - 所以輸出位置跟沒有跳號的狀況最多只會差4
    - 如果改成紀錄差距的話，就只需要 $3 \times 16 = 48$  bit
  - 用一個**shift array**來傳遞差距，有跳號的時候就把存的值往上加
    - 傳遞方法和前一頁的**address**傳遞是類似的
    - 最後在輸出前，把這個差值加上

# 特殊規則處理

- 還有部分**corner case**需要處理，不過因為比較細枝末微，就不詳細紀錄了
  - ?出現在第一個或最後一個**pattern**需要特別處理
  - **Case sensitive**多加的輸入**flip-flop**會讓**pipeline**亂掉，需要多加一個**buffer state**
  - 很多的**counter**會有多加一，少加一的狀況需要修正
  - ....