**CAD/VLSI Circuit Design期末報告**

**7109064300　　范文軒　　電機丁組(數位)**

**(一)、動機**

搜尋演算法是近些年科技中很重要的基石，而字串比對演算法(string matching algorithm)也是其中之一。像這樣的比對方式會被使用至DNA序列的比對、資料庫的查詢(query)等。而如今資料庫系統中的資料量越來越大。為了某個特徵比對，可能會使用到多次的比對查詢。因此加速這個過程，就顯得重要的。若使用暴力破解的方式，必須一個字元依次比較，而因為沒有辦法加速，時間複雜度會為O(m\*n) (in worst case)。如今不同新的演算法也不斷推出，像是近年最多人使用的演算法之一：KMP演算法[1]，優化了最原始的方式，使得無意義的比較可以跳過，帶來更多速度上的優勢，而時間複雜度減至O(m+n) ( in worst case)。而本次構想希望參考可平行化運算KMP演算法[2]，並且利用硬體亦可平行化的優勢，實做出有平行化的KMP演算法的SME(string matching engine)。

**(二)、主要概念與預想[2]**

1. **構想概要**

**(1). KMP Algorithm[1]**

KMP演算法相較於暴力破解，多增加了前處理(preprocessing)的部分，在此演算法即所謂的Failure Function。增加了一個關於Pattern的表格去紀錄若判別失敗該跳去哪一個位置。

**(2). 可平行化KMP Algorithm[2]**

而在參考[2]中，使用了平行化技巧，將比對時分切成4個特殊大小的字串同時比對，進而減少時間的損耗。

參考演算法概念後，在設計中設計以下分割之作業：Idle、String與Pattern輸入、Failure Function建立、比對環節以及輸出結果。並打算在比對環節中設計4的PE(Process Engine)同時運作於比對環節中進而實驗平行化。

1. **KMP演算法**

**(1). Failure Function**

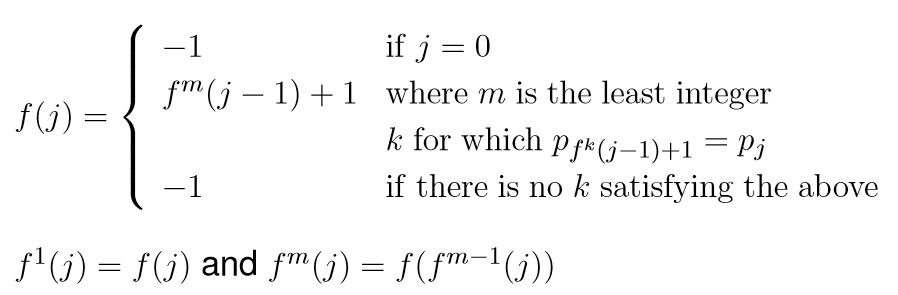
Failure Function是用來計算Pattern比對失敗時可以跳去哪一個Pattern位置繼續比較。算法主要概念為「前綴(suffix)要位移幾位可以和目前的字串相同」。

**（舉例）**



**圖一、Failure Function概念舉例**

但通常如果用此定義算通常會計算非常久，所以後來大多都使用Dynamic Programming的方式加速Failure Function的算法。

****

**圖二、Failure Function with Dynamic Programming**

**(2). KMP Matching Operation**

由上述的Failure Function完成計算後，生成的Failure Table可以紀錄比對失敗時Pattern該從Failure Function的值加一的位置，與String比對失敗的地方繼續比較下去，依此類推。





**(三)、主要架構**



**圖一、SME系統架構圖**

1. **SME概要**

　　此次設計的SME使用多模組的方式建構（可參考上圖），主要有Shared Memory、Control Unit以及KMP Compare PE(Process Engine)。而Shared Memory內部為了要計算KMP演算法中的前處理部分，多設計了專門計算Failure Function的模組 — Failure Function Calculator一併放在Shared Memory內部，來使用特殊方式將Shared Memory實作出來(將在三、c部分說明)。

　　資料將從外部傳入Shared Memory，Shared Memory將接收並且儲存在內部。等待資料全部載入完成，將會啟動Failure Function Calculator計算該Pattern的Failure Function後一併將輸入的資料分享到每一個PE中，而Control Unit將會把需要做的長度告訴各個PE，而各個PE會依照KMP演算法的方式計算出各分段結果並傳給Control Unit，最後，Control Unit會將結果輸出。

**b、Shared Memory**

　　　因字串的長度需要相對應大小的暫存器，因此利用Shared Memory來增大暫存器的使用率，同時間也減少不必要PE裡的暫存器。

**c、Failure Function Calculator**

　　　Failure Function部分會將Shared Memory的Pattern資料傳入，進而計算。而此次設計計算Failure Function的方式是利用，Dynamic Programming的演算法實作，利用上一筆的結果來繼續計算下一筆的，使得比起重新計算更加有效率。

**d、KMP Compare Process Engine(PE)**

　　利用Sharing Memory的輸出得到資料、Failure Function進而比對，與先前的KMP演算法無異，唯有不同之處為可以彈性調整字串比較的長短來做比較，也是為了做到平行化的前提

。

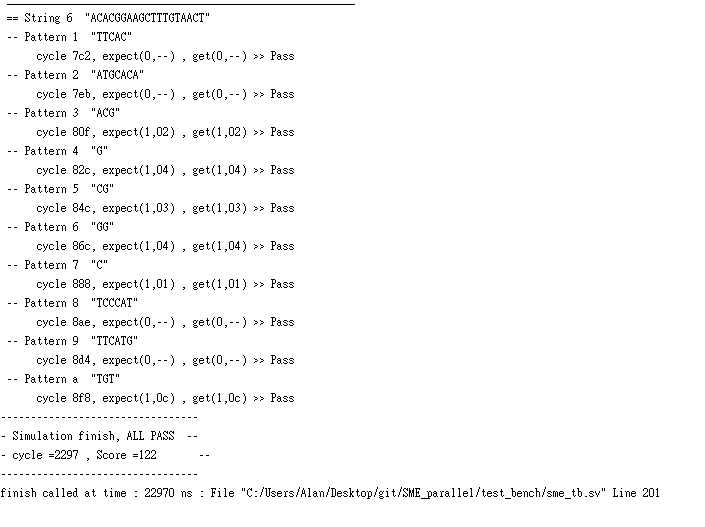
**e、資料與Pattern**

圖一為此次設計的架構，左方的Host為Testbench的部分，會將需要比對的資料傳送至右方的設計中（SME），為了模擬DNA的比對情況，將會傳送內容為DNA的序列（Ａ、Ｔ、Ｃ、Ｇ）的字串序列。

**e、彈性化**

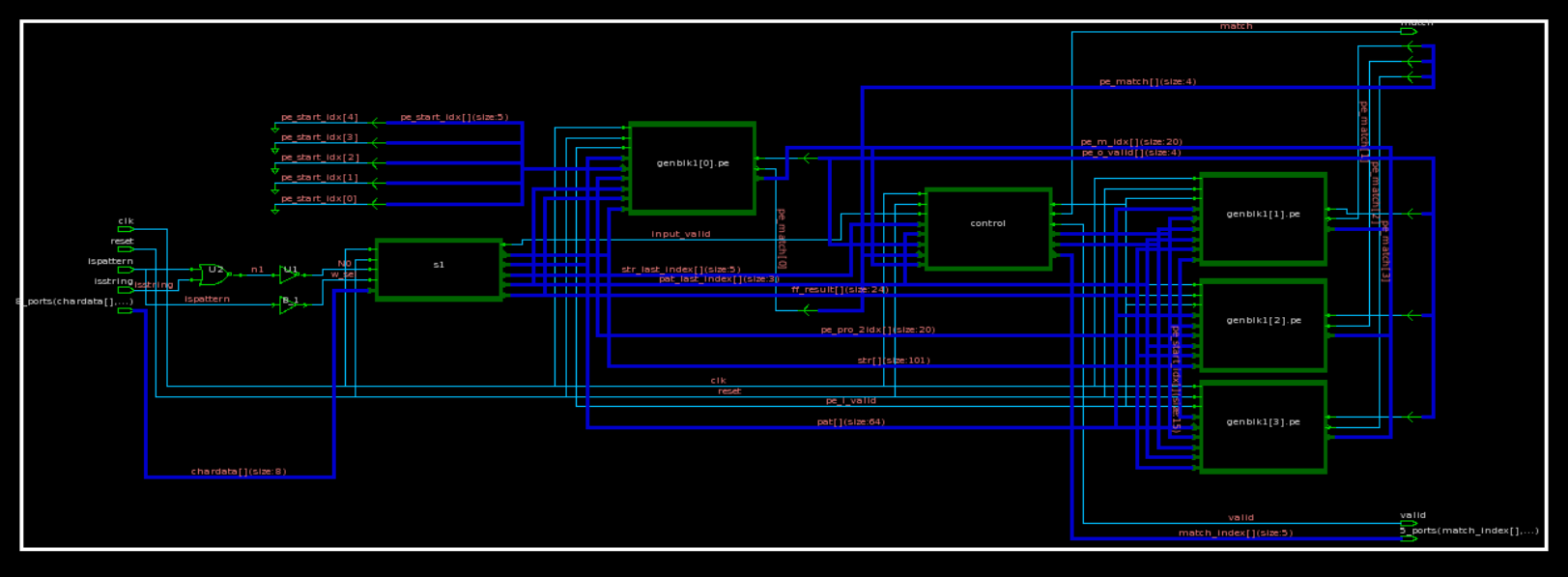
因為為了使此SME更加有彈性，設置了多個參數可供調整，可自行決定String與Pattern的最大長度，同時也可以自由擴大PE數目。

**(四)、合成與結果**



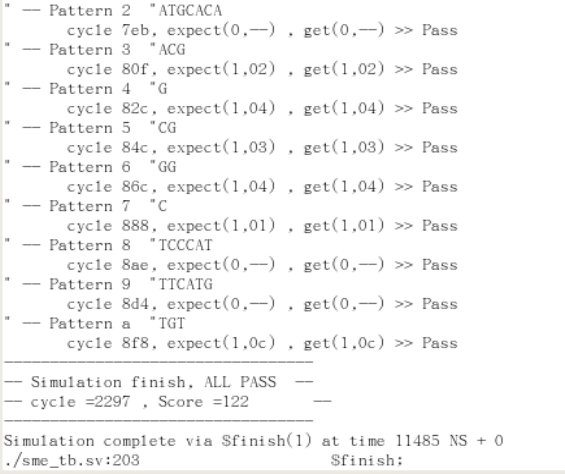
**圖二、Simulation結果(Vivado)**

**(Max String Length : 32, Max Pattern Length : 8, no.PE : 4)**

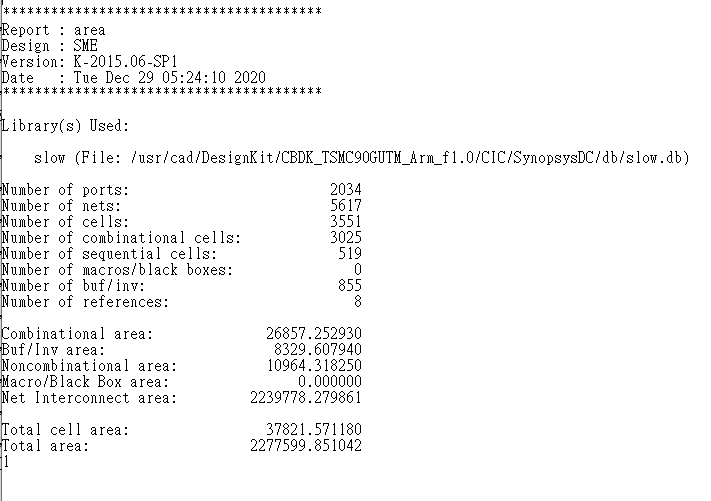


**圖三、合成結果(Design Vision)**

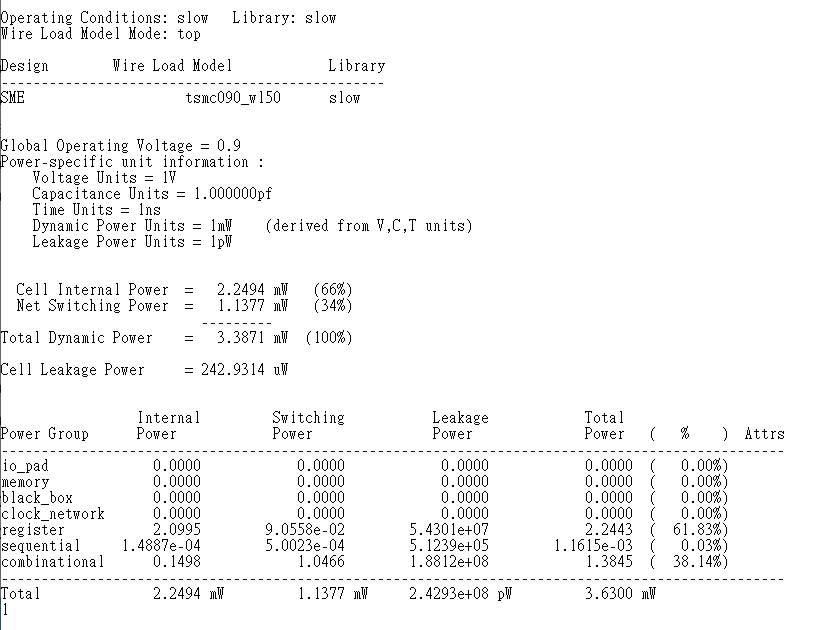
**(Process : TSMC 90nm ， Clock Constrain : 200Mhz, no.PE : 4)**

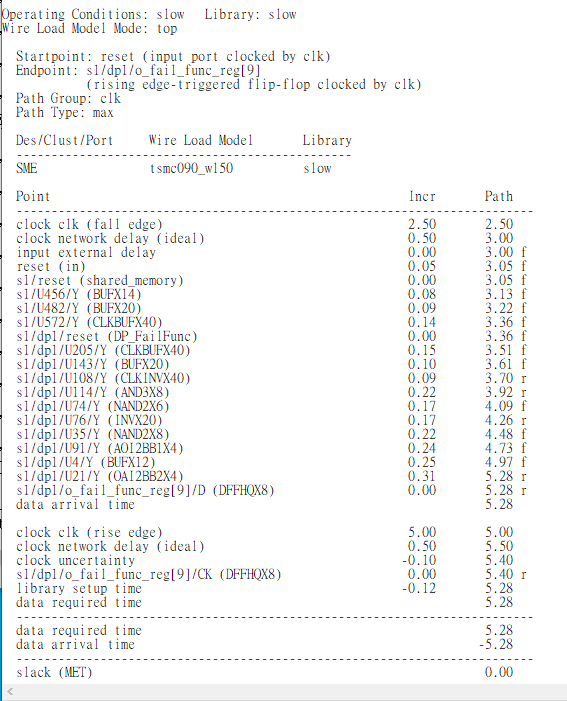
**圖四、Pre-Simulation(Ncverilog)**

**(Run Clock Rate : 200Mhz , Max String Length : 32)**

**(Max Pattern Length : 8, no.PE : 4 )**

**圖五、Area Report(Design Vision)**

**圖六、Power Report(Design Vision)**



**圖七、Timing Report(Design Vision)**

**(四)、規格**

|  |  |
| --- | --- |
| **Hardware** | |
| **Synthesis Process** | TSMC 90 nm |
| CLK Cycle | 200Mhz |
| Throughput | 5.2 M Match /Sec |

*[備註]：此Throughput之情況為原始設計目標最大String長度32Bytes，最大Pattern長度8 Bytes，SME中共有4個比對PE，於200Mhz下筆對了60筆得到2297 Cycle(即11485 ns)，由此推估Throughput可達5.2M Match/Sec。*

**(五)、參考**

**[1]** D. E. Knuth, J. H. Morris and V. R. Pratt, "Fast Pattern Matching in Strings," SIAM J. COMPUT, Vol.6, No.2, June 1977.

**[2]** U. S. Alzoabi, N. M. Alosaimi, A. S. Bedaiwi and A. M. Alabdullatif, "Parallelization of KMP String Matching Algorithm," 2013 World Congress on Computer and Information Technology (WCCIT), pp. 1-3, 2013.

[備註]所有的Code皆放在我個人的github上面：

https://github.com/vanwanTaiwan/SME\_parallel