87 學年度大學校院積體電路設計比賽

研究所 Cell-based 組競賽試題

(for VERILOG)

一、試題說明：

請以 CIC Compass Design Kit V4.x 所提供的 High-Performance 版本 Standard Cell (CB60HP231D) 設計一個 Sign-Magnitude Postfix Expression Co-processor (後置數學式協同處理器，其功能及規格將詳述於後)。參賽者在規定時間內須完成 Front-end 及 Back-end Design，並繳交各項設計相關資料以供主辦單位評分。

1-1 功能需求

假設有一資料處理系統其方塊圖如 Figure 1. 所示，此系統中的四個模組的功能簡述如下：

Module Name	Description
HOST	Host Computer Module, **given**
POSTFIX	Postfix Expression Co-processor Module
MEM	Shared Memory Module, **given**
CLKGEN	Clock Generator Module, **given**

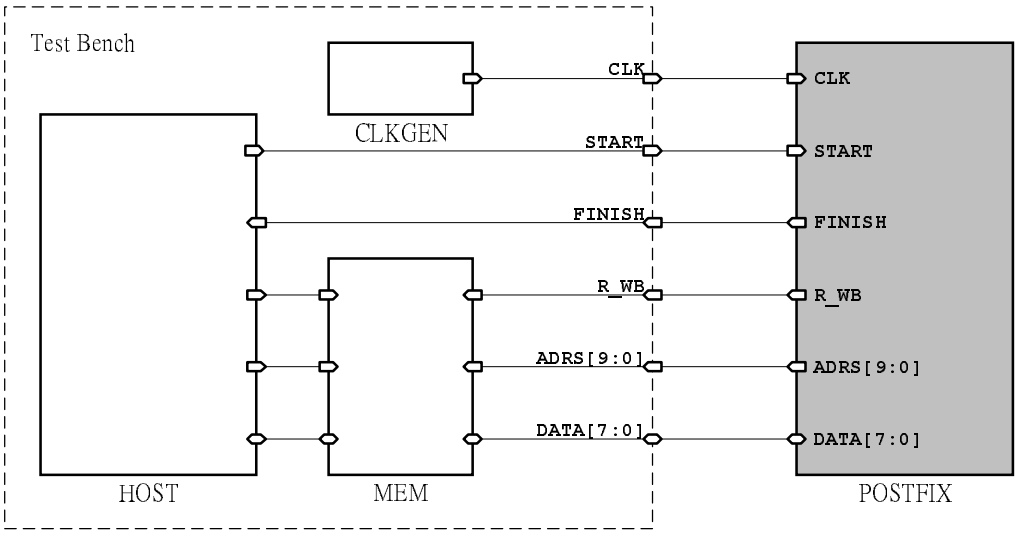


Figure 1.

在資料處理過程中 HOST 會陸續將 Postfix Expression 寫入 MEM，當 MEM 中累積了一定數量的 Expression 後，HOST 透過 START 訊號要求 POSTFIX 開始進行計算 (HOST 將 START 由 Low 設定為 High，維持 2 個 clock cycle 後再將其設定為 Low)。POSTFIX 在接到 START 訊號要求後開始由 MEM 讀出 Postfix Expression 並進行計算，計算完成後再將結果寫回 MEM。當 MEM 中的所有 Postfix Expression 都處理完畢後，POSTFIX 必須將 FINISH 訊號由 Low 設定為 High，維持至少一個 clock cycle 後再將其設定為 Low 以回覆 HOST 計算已完成 (FINISH 訊號在 START 訊號被設定為 HIGH 之後到所有運算結束前，應一直維持為 LOW)。

對 POSTFIX 而言，其由 MEM 讀入的資料為一連串的 Postfix Expression。 所謂的 Postfix Expression 是由 Operand 及 Operator 構成。Operand 代表數學式中之運算元，Operator 代表數學式中之運算子(如: +, -, * 等)。其特徵是 Operator 放在 Operand 之後，而非放 Operand 之間，故稱之為 Postfix Expression。例如：a + b 的 postfix expression 為 a b +，而 a * b - c * d 的 Postfix Expression 為 a b * c d * - ，但 a * (b - c) * d 的 Postfix Expression 則為 a b c - * d *。

在本試題中僅使用到三種算術的 Operator，加、減及乘(+, -, *)。因此，在所有的 Postfix Expression 中，每一個 Operator 前面都有其運算所需的兩個 Operand (當然這些 Operand 可能是前面其他 Operator 的運算結果)。對 POSTFIX co-processor 而言，若由 MEM 讀入的資料為 Operand，則先將其暫存。若讀入的資料為 Operator，則將先前暫存的 Operand 取出來做運算，其運算結果也需暫存以供後續的 Operator 使用。如此重複，一直到 Expression 結束，再將最後的運算結果寫入 MEM。

以 a b c - * d * 為例，其運算動作如下：(以下各動作純粹為解釋 Postfix Expression 的求值方法，參賽者可自行設計 POSTFIX co-processor 之動作)

1. Read a , Store a
2. Read b , Store b
3. Read c , Store c
4. Read - , Execute b-c , Store ***(b-c)***
5. Read * , Execute a****(b-c)*** , Store ***a*(b-c)***
6. Read d , Store d
7. Read * , Execute ***a*(b-c)***d , Store ***a*(b-c)*d***
8. Read End of Expression , Write ***a*(b-c)*d*** to **MEM**

➢ (斜粗體字表示為前面 Operator 的運算結果)：

1-2 MEM 之空間分配

主辦單位所提供之 MEM 為 1024 byte，每 byte 為 8 個 bit 之 Asynchronous RAM，其中位址 000_{16} ~ $1FF_{16}$ 用來放置由 HOST 寫入的 Postfix Expressions，由 000_{16} 開始放起。位址 200_{16} ~ $3FF_{16}$ 用來放置 POSTFIX co-processor 運算的結果，由 200_{16} 開始放起。參賽者設計 POSTFIX co-processor 時若需要用到記憶體做為暫存空間

時，可使用此 MEM 的空間，其面積不列入計分。 MEM 之 timing 如附件二。

1-3 輸入及輸出格式定義

在本試題中，由 HOST 寫入 MEM 中的 Operand 均為 **16-bit *Sign-Magnitude* Integer** 的正數，其值域為 0_{10} ~ 32767_{10} (即 Operand 的 MSB 恆等於 0)，在 MEM 中以 Big-Endian 方式儲存(如附件一所示)。而運算子 Operator 也是以 16-bit 方式表示，但是其 MSB 恆等於 1，本系統中一共定義了 5 個 Operator，其編碼如下表所示：(除定義的五個 CODE 外，其餘 MSB 為 1 的 CODE 均不會被使用及出現)

Operator	Code	Description
ADD	$80FF_{16}$	***Sign-Magnitude*** Saturation Adding
SUB	$81FF_{16}$	***Sign-Magnitude*** Saturation Subtracting
MUL	$82FF_{16}$	***Sign-Magnitude*** Saturation Multiplying
EOE	$83FF_{16}$	End of Expression, put the result to MEM
END	$FFFF_{16}$	End of Expression Sequence

註: $80FF_{16}$ 代表以十六進制表示法表示之 80FF，相當於十進制之 33023_{10}，本試題中其餘數值資料皆以此類推。

在 POSTFIX 中的數學運算均採用 ***Sign-Magnitude*** 方式，其輸出資料格式為 16-bit ***Sign-Magnitude*** Integer，其值域為 -32767_{10} ~ 32767_{10}。其中 MSB 為 Sign Bit，MSB = 0 表正數，MSB = 1 表負數，其餘 15 個 bit 則代表其 Magnitude(0000_{16}及 8000_{16} 均代表 0_{10})，輸出資料在 MEM 中以 Big-Endian 方式儲存。

- Saturation 運算： 因為輸出資料格式為 16-bit ***Sign-Magnitude*** Integer，所以在計算過程中若發生 Overflow(運算結果小於-32767 或大於 32767)則需做 saturation 動作。若運算結果大於 32767 則令其為 32767，若小於-32767 則令其為-32767。
- 假設由 HOST 寫入 MEM 中的 Postfix Expression 均為正確之格式，設計時不需考慮輸入格式錯誤之處理。另外每個 Postfix Expression 中所需暫存的 Operand 數目不會超過 8 個。

1-4 佈局注意事項

1. 規定使用 block layout 的方式 (即不含 I/O pad，只含 core cell 的方式)。
2. Power Ring 及 Power Pin 的寬度：vdd! 和 gnd! 各 40um。
3. 一律不加 power strip，也不做 clock tree synthesis，也不加 core filler。
4. Signal Pin 的位置由參賽者自行決定。
5. 其餘注意事項可參閱 http://www.cic.edu.tw/~chtsai/icc99_note.html的說明。

二、使用軟體注意事項

1. 主辦單位提供：

00.README	說明檔
postfix.v	The deceleration of the postfix template module
test.v	testfixture file
rtl.f	RTL simulation option file
INPUT.DAT	input data
EXPECT.DAT	expected result
in_file	The initial file of the Compass Mercury Delay Calculator.
dlc.init	The initial file of the Cadence CDC.

- 請用postfix.v中的宣稱及template module來建立 POSTFIX module。
- test.v包含了Fig.1中虛線部分的模組(HOST, MEM and CLKGEN)。test.v會將INPUT.DAT的data讀入MEM中，並發出START訊號。等到收到FINISH訊號，test.v會將MEM中的運算結果與EXPECT.DAT中的data做比較。若結果正確，則會顯示PASS訊息。若有錯誤，則會顯示發生錯誤的address與data。此外，test.v尚會顯示START訊號設定為HIGH開始到FINISH被設定為HIGH所花的時間。
- 若要改變clock period，可以修改test.v中的"HALF_CYCLE_TIME"參數，**clock period = HALF_CYCLE_TIME * 2**。（HALF_CYCLE_TIME 預設值為25ns，即clock period為50ns）
- 執行gate-level模擬時，可使用**Compass linear model**(cb60hp231d.vmd)搭配 **Mercury Delay Calculator**(只能在Solaris執行)或是使用 **Table model**(hp_io_time.v, hp_io_typ.ctlf) 搭配**Cadence CDC**(可在SunOS 4.1.x與Solaris執行)。 但評分時統一以**Mercury Delay Calculator**計算的timing為準。
- Mercury Delay Calculator 之呼叫可參考 test.v
  // $VTOOLS_decal("in_file", "product.xch");
- Cadence CDC 之呼叫可參考 test.v
  // $cdc(top, "dlc.init");
  請自行修改dlc.init內的 CTLF_FILE path。
  CTLF_FILE "Your_Lib_Path/hp_io_typ.ctlf"
- 若需儲存SHM waveform 可參考 test.v
  // $shm_open("waves.shm");
  // $shm_probe("AS");

2. 請利用 CIC 提供的 COMPASS cell library(**4.x 版之 High-performance cell**)完成設計。

3. 設計者可以從 RTL(VHDL or Verilog)著手，經由 Synopsys synthesize 產生

gate-level netlist，再將 netlist 轉入 OPUS 中以 Cadence Silicon Ensemble (97A) 進行 layout placement and routing。也可以直接在 OPUS(97A)中進行 schematic entry、simulation 及 layout placement and routing

4. 設計的 module 不需包含 I/O Pads.
   在 Physical Design 做 routing 前請將 block 的 i/o pin 調到 grid 上。(在 Place&Route選單=>Block=>Adjust Pins, 按OK)。

5. 佈局驗證(DRC, ERC 及 LVS)請使用 Dracula(Cadence 97A)。跑 Dracula 需在 layout 加上 text label,請使用 layout editor 以(text2 drawing) layer 當作 text 打在 i/o pin 的位置。

6. 為避免主辦單位驗證結果時的困擾，所有電路之輸出、輸入及電源等接腳須與題目指定之名稱相同。

三、繳交資料

1. Verilog RTL-level synthesizable code 檔案。
2. Verilog Gate-level code 檔案。
3. 完整的設計資料庫(Cadence design library)
   請利用 tar 整合成一個 .tar 檔，假設你的 library name 為 **your_lib**。經由 tar 指令

   >tar cvf *yourname*.tar your_lib

   將會得到 *yourname*.tar 檔名的檔案。
4. 佈局之 GDSII 檔案及 DRC 與 LVS 驗證檔案(*.sum 及 *.lvs 檔)。

✣ 請另建一個新目錄，並將以上各項需要繳交的檔案複製到此新目錄下。在此目錄下執行以下指令將所有檔案整合並壓縮為一個檔案：

> tar cvf xxxxxxx.tar *
↑
檔案名稱自取
> compress xxxxxxx.tar

經由以上指令可得到 **xxxxxxx.tar.Z** 的檔案。

✣ 文件說明檔(**report.xxx**):

本項檔案係用來說明參賽者繳交的各相關檔案之檔案名稱、使用軟體項目、相關規格及其他說明事項。本項檔案格式由主辦單位提供，請參照隨題目所取回的檔案中的 **report.000** 檔，將相關資料名稱填入檔案中。

✣ 請將最後壓縮整合的檔案(**xxxxxxx.tar.Z**)以及文件說明檔(**report.xxx**)使用 binary 模式利用 **icresult** 的帳號及密碼傳送至以下四個傳送網站之一即可。

（請先上傳 xxxxxxx.tar.Z，再上傳 report.xxx）

1. 台灣大學　　: video3.ee.ntu.edu.tw (140.112.41.223)
2. 晶片中心　　: ftp.cic.edu.tw (140.126.24.62)
3. 雲林科技大學 : 140.125.35.10
4. 成功大學　　: cad7.ee.ncku.edu.tw (140.116.156.157)

傳送的目錄為 inst_cell/參賽隊號 。

- 各項設計資料檔如需更新時，請重複以上步驟，並另取新的檔名傳送，注意務必更改文件說明檔 **report.xxx** 的相關內容。
- 文件說明檔的檔名須以 **report** 為檔名開頭，副檔名請以數字依序命名，如 **report.000** 代表原始檔名，**report.001** 代表第一個更新版本，如另有更新，請依此類推。
- 其他相關事項請參考參賽手冊。

四、評分方式：

1. 經主辦單位驗證其電路功能正確及各項資料完備者始予以計分。
2. 分數計算公式為: Score=1/(Time*Area)，Score 最高者為第一名。若有 Score 相同及接近者，由評審委員依照各設計之創意排定名次。
   - Time：由 START 訊號設定為 HIGH 開始到 FINISH 被設定為 HIGH 所花的時間。
   - Area：僅計算 design boundary 的面積，不含 IO pad。在 Silicon Ensemble 視窗中，圍繞在晶片周圍的一個紫色框即為 design boundary。

附件一:範例

I. Sign-Magnitude 運算

$4422_{16}+2211_{16}=6633_{16}$

```
   0100 0100 0010 0010
 + 0010 0010 0001 0001
-------------------------
   0110 0110 0011 0011
```

$4422_{16}+(-2211_{16})=2211_{16}$

```
   0100 0100 0010 0010
 + 1010 0010 0001 0001
-------------------------
   0010 0010 0001 0001
```

$(-4422_{16})+2211_{16}=-2211_{16}$

```
   1100 0100 0010 0010
 + 0010 0010 0001 0001
-------------------------
   1010 0010 0001 0001
```

$(-4422_{16})+(-2211_{16})=-6633_{16}$

```
   1100 0100 0010 0010
 + 1010 0010 0001 0001
-------------------------
   1110 0110 0011 0011
```

$4422_{16}+4422_{16}=8844_{16} > 7FFF_{16}$ ==saturation=> $7FFF_{16}$

```
   0100 0100 0010 0010
 + 0100 0100 0010 0010
-------------------------
   0111 1111 1111 1111
```

$(-4422_{16})+(-4422_{16})=-8844_{16} < -7FFF_{16}$ ==saturation=> $-7FFF_{16}$

```
   1100 0100 0010 0010
 + 1100 0100 0010 0010
-------------------------
   1111 1111 1111 1111
```

$03C6_{16}*000F_{16}=389A_{16}$

```
   0000 0011 1100 0110
 * 0000 0000 0000 1111
-------------------------
   0011 1000 1001 1010
```

$03C6_{16}*(-000F_{16})=-389A_{16}$

```
   0000 0011 1100 0110
 * 1000 0000 0000 1111
-------------------------
   1011 1000 1001 1010
```

$(-03C6_{16})*-000F_{16}=-389A_{16}$

```
   1000 0011 1100 0110
 * 0000 0000 0000 1111
-------------------------
   1011 1000 1001 1010
```

$(-03C6_{16})*(-000F_{16})=389A_{16}$

```
   1000 0011 1100 0110
 * 1000 0000 0000 1111
-------------------------
   0011 1000 1001 1010
```

$03C6_{16}*00FF_{16}=3C23A_{16} > 7FFF_{16}$ ==saturation=> $7FFF_{16}$

```
   0000 0011 1100 0110
 * 0000 0000 1111 1111
-------------------------
   0111 1111 1111 1111
```

$(-03C6_{16})*(-00FF_{16})=-3C23A_{16} < -7FFF_{16}$ ==saturation=> $-7FFF_{16}$

```
   1000 0011 1100 0110
 * 1000 0000 1111 1111
-------------------------
   1111 1111 1111 1111
```

II. POSTFIX 運算

假設 START 訊號產生時, MEM 有三個待處理的Expression。

$7A00_{16}$-$5BC0_{16}$, $000F_{16}$*($0A00_{16}$-$0B00_{16}$), $7FFF_{16}$*$7FFF_{16}$-$6FFF_{16}$，其表示法為:

$7A00_{16}$, $5BC0_{16}$, SUB, EOE,
$000F_{16}$, $0A00_{16}$, $0B00_{16}$, SUB, MUL, EOE,
$7FFF_{16}$, $7FFF_{16}$, MUL_{16}, $6FFF_{16}$, SUB, EOE, END

MEM 的內容如下:

```
Address   Data
  000     7A 00 5B C0 81 FF 83 FF - 00 0F 0A 00 0B 00 81 FF
  010     82 FF 83 FF 7F FF 7F FF - 82 FF 6F FF 81 FF 83 FF
  020     FF FF xx xx xx xx xx xx - xx xx xx xx xx xx xx xx
```

…….

經過 POSTFIX 運算，

Read 7A00, Store 7A00
Read 5BC0, Store 5BC0

```
Read  81FF,   Execute 7A00-5BC0 => 1E40
Read  83FF,   Write 1E40 to MEM
Read  000F,   Store 000F
Read  0A00,   Store 0A00
Read  0B00,   Store 0B00
Read  81FF,   Execute 0A00-0B00 => 8100
Read  82FF,   Execute 000F*8100 => 8F00
Read  83FF,   Write 8F00 to MEM
Read  7FFF,   Store 7FFF
Read  7FFF,   Store 7FFF
Read  82FF,   Execute 7FFF*7FFF => 7FFF
Read  6FFF,   Store 6FFF
Read  81FF,   Execute 7FFF-6FFF => 1000
Read  83FF,   Write 1000 to MEM
Read  FFFF,   Set FINISH to High for one clock cycle
```

當運算結束, FINISH 訊號產生時, MEM 的內容如下:

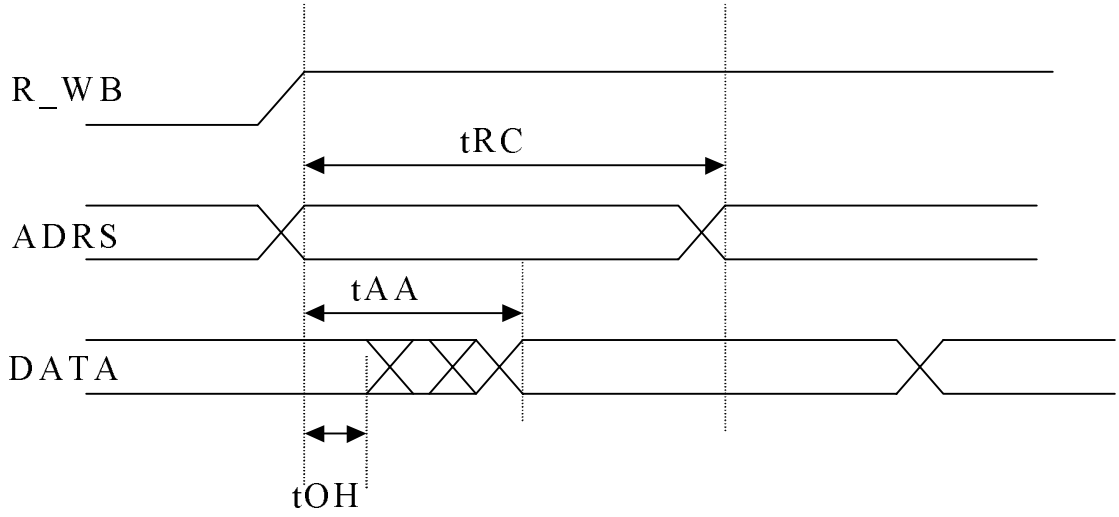
…….

```
Address   Data
200       1E 40 8F 00 10 00 xx xx - xx xx xx xx xx xx xx xx
```

…….

附件二: MEM timing diagram:

- **Read Cycle:**

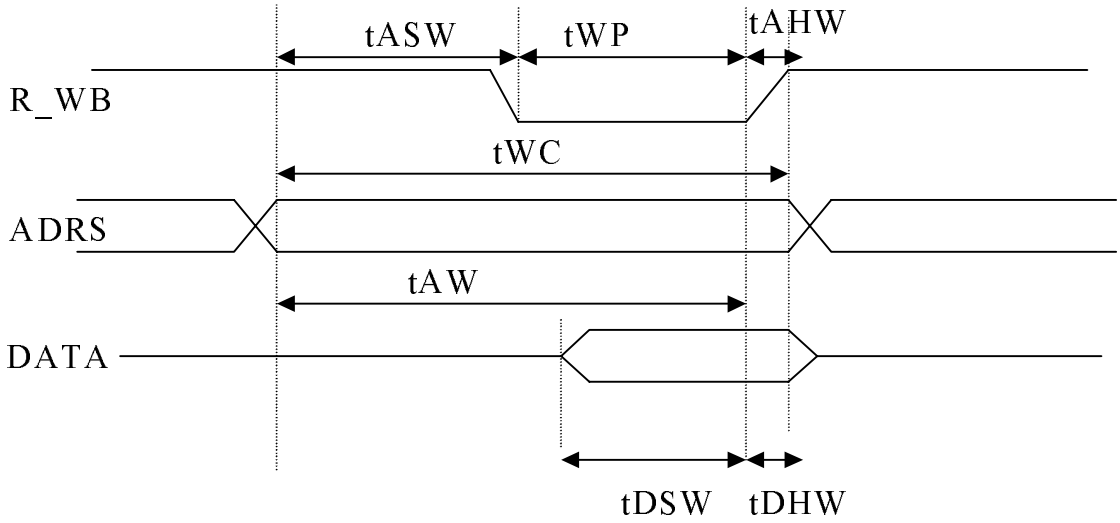


tRC: Read cycle time. (5.3 ns)

tAA: Address access time. (5.3 ns)

tOH: Output hold time from address change. (2.8 ns)

- **Write Cycle:**



tASW: Address setup to write start. (0 ns)

tAHW: Address hold time from write end. (0 ns)

tWP: R_WB pulse width. (3.7 ns)

tWC: Write cycle time. (4.9 ns)

tAW: Address setup time to write end. (4.9 ns)

tDSW: Data setup time to write end. (2.5 ns)

tDHW: Data hold time after write end. (0 ns)