1

開始學習 SKILL

1. 簡介

多年以來，CADENCE 公司的 CAD tool 一直是世界上使用最廣泛，功能最強大的積體電路(Integrated Circuit)設計工具。而爲了因應 IC 的複雜度越來越高，設計的困難度也越來越高的情況，CADENCE 的 CAD 整合開發環境也越來越龐大，所提供的功能也日益強大，造成使用者在維護及管理上的一大負擔。再則，每一家的設計公司的在設計的流程中多多少少都會有一些小步驟，無法用 CADENCE tool 提供的基本做法來達成； 或者是不同公司的 tools 之間資料轉換的問題。 工程師遇到此類問題可能需要透過人工的方式去完成連接設計流程中相連的兩個步驟； 或是乾脆去開發一些小軟體來完成這些特定的工作，而此時使用者可能會面臨如何將自己開發的軟體的 I/O 與 CADENCE tool 的整合環境相連結的問題。 一般的做法是產生一些資料檔來做資料交換的中介，這些資料檔的格式可能是 CADENCE 支援的一標準資料交換格式，也可以是使用者自訂的資料格式。這樣子是一種間接的做法，因爲使用者無法直接去存取 CADENCE 環境的內部資料，所以在處理上的彈性會小很多，也較不方便。

爲了方便使用者使用整個 CADENCE tools 的整合開發環境，以解決上述的困擾，CADENCE 公司遂發展了 SKILL 語言。 SKILL 是一種高階的、交談式的語言，是用於 CADENCE tool 的整合開發環境內的命令語言（command language）。 SKILL 採用人工智慧語言 LISP 的語法爲藍本，再加上常用的 C 語言的部份語法設計而成。SKILL 語言提供許多的介面函式，能讓使用者可以撰

寫程式直接去存取 CADENCE 整合環境內的電路資料內容； 也可以讓使用者去開發將自己開發的應用程式併入 CADENCE tool 的整合環境裡。有了 SKILL 言，使用者可以讓 CADENCE tool 更充份地融入整個設計流程之中，減少瑣碎的人工轉換時間，提升公司的生產力。

1.1 LISP 式的語法

在 SKILL 裡面，使用函式的呼叫方式可以有兩種方式：

（一）Algebraic 表示形式，也就是

Func（arg1 arg2 …）

（二）前置表示形式，此爲 LISP 型式的語法

（Func arg1 arg2 …）

程式是由敘述來組成的，正如在 LISP 語言裡面一樣，SKILL 的敘述是以串列（list） 的形式來表示。 如此的設計方式使得程式可以和資料用同樣的方式來處理。 使用者可以動態地建立、修改、或計算函式或表示式的值。

另外 ，在 SKILL 中不像一般的程式語言一樣有提供字元這種資料形態，字元就是用符號本身來表示 ，例如字元 “A”就是用”A”這個符號（變數）來代表。

2. 快速瀏覽 SKILL

2.1 解釋名詞

本節將介紹一些專有名詞：

名詞	意義
OUTPUT	SKILL 程式執行結果可以顯示到 xterm、設計視窗、檔案、或是命令解譯視窗 CIW（command interpreter window）
CIW	很多 CADENCE 的應用程式之起始工作視窗，包含有一行命令輸入列、一個輸出區域、一個功能選單列。
SKILL expression	呼叫一個 SKILL function 的程式敘述
SKILL function	一個有命名的、參數化的程式段

要啓動一個 SKILL 的 function 有幾種方式，在不同的 CADENCE 的應用程式裡，使用者可以透過 Bindkey, Form, Menu, 或 SKILL process 等來啓動，其意義如下：

名詞	意義
Bindkeys	將一個 function 與一個鍵盤的事件關聯起來，當使用者引發一個鍵盤事件時，CADENCE 軟體便會計算其相關聯的 function
Form	有些函式需要使用者提供一些資料輸入，通常是透過一個跳出式表格（pop-up form）來輸入資料
Menu	當使用者選擇 menu 上的一個項目時 ，系統便會執行相關之 SKILL 函式的計算
CIW	使用者可以直接在 CIW 輸入一個 SKILL function 來得到一個立即的計算結果
SKILL process	使用者也可以透過一個 UNIX 環境底下的行程，來啓動 CADENCE 裡的 interpreter，以便直譯某些 SKILL 程式

所有的 SKILL 函式都會傳回一值。在本書中我們將用“⇨”來表示函式的回傳值。 在 SKILL 裡面，大小寫是不同的。 要呼叫一個 SKILL 的函式的方式如下：

```
strcat( ”How” ”are” ”you” )
⇨”How are you”
```

或者用：

```
( strcat “How” “are” “you” )
⇨”How are you”
```

注意的是，在函式名稱與左括弧之間不可以留空白。 函式的內容可以分成幾行來寫，不一定要在同一行才可以。同樣地，幾個函式也可以放在同一行上，但此時只有最後一個函式值會回傳到螢幕上。

2.2 基本資料型態

在 SKILL 裡面資料也是大小寫不一的（case-sensitive），最簡單的資料型態有整數、浮點數、與字串。字串用雙引號 “” 來括起來。

2.3 運算子

SKILL 提供了一組運算子，每一個運算子對應到一個 SKILL 的函式。下表是一些常用的運算子：

運算子（依優先順序由大到小排列）	對應之 SKILL 函式	運算
**	expt	Arithmetic
*	times	“
/	quotient	“
+	plus	“
-	difference	“
++S, S++	preincrement, postincrement	“
= =	equal	tests for equality
! =	nequal	tests for equality
=	setq	Assignment

以下是一個應用運算子的例子：

x =5 y =6x+y

⇨11

2.4 變數

在 SKILL 的程式中你不需要去宣告變數， 一個變數的名稱可以包含英文字母、數字、底線（__）、問號（？）。 變數名稱的第一個字不可是數字。你可以用= 運算去指定一個變數的值。 你也可以直接鍵入一個變數名稱來執行以取得其值。你也可以用 type 這個函式來取得變數的資料型態。

3. SKILL 串列

SKILL 串列 簡單地說就是一些依序排列的 SKILL 資料物件。一個串列內可以包含任何 SKILL 資料型態的資料，一個串列的資料表示式要用括弧括起來。SKILL 允許有空的串列，表示成“（）”或 *nil*。 串列 裡面的元素也可以是另一個串列。以下是一些串列的例子：

串列	說明
（1 2 3）	一個包含１, 2, 3 三個整數的串列
（1）	一個包含 一個整數 1 的串列
（）	一個空的串列
（1（2 3）4）	一個串列 裡面還包含另一個串列

至如何指定一個串列變數的值呢？舉例如下：

S1= ‘(”A” ”B” “C”)

請注意在左括號之前須加上一個單引號。

3.1 建立串列

有幾個方法可以建立一個串列，舉例如下：

1. 直接使用單引號，例如：
   ‘(２４６) ⇨(２４６)
2. 呼叫使用串列 函式，例如：
   x=2 y=4
   list(x y 6) ⇨(２４６)
3. 使用 cons 函式，例如：
   r = ‘(4 6)
   r =cons(2 r) ⇨ (２４６)
4. 用 append 函式來合併兩個串列：
   alist = ‘(１2 3)
   blist = ‘(4 5 6)
   clist =append（alist blist） ⇨(１２3４5６)

3.2 串列與座標表示

由串列衍生出來的實際資料表示有兩個最主要的是座標（coordinates）與界限方塊（bounding box）。 對佈局圖（layout）而言， 吾人常用到的是這兩樣東西。在 SKILL 中 xy 座標可用兩個元素的串列來表示，例如

xVal=100

yVal=200
pCoordinate= xVal:yVal ⇨（100 200）

而 bounding box 是用左下及右上兩點座標來表示，例如

bBox=list (300:400:500:600)

或者可以用下列方式來產生：

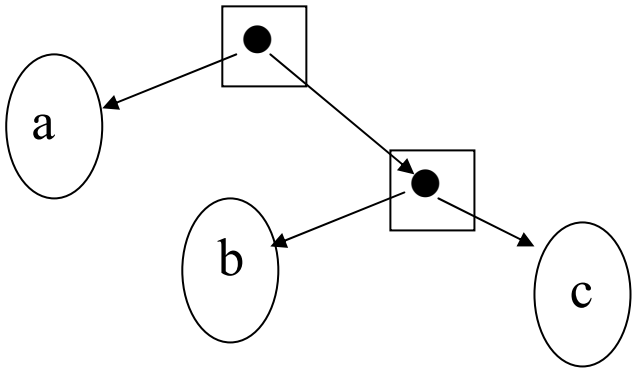
lowL = 300:400
upperR=500:600
bBox=list(lowerL upperR)

或者直接用雙層串列

bBox= ‘((300 400) (500 600))

3.3 使用串列

在 SKILL 中， 一個串列的內部儲存方式相當於是二元樹（binary tree）的結構，每一個內點都相當於一個分支決定點（branch decision point），例如：有一個 list(a b c)，此一 list 在內部儲存的方式可以下圖表示



在 SKILL 語言中，提供了很多的 list 相關的函式，以下是較基本的：

函式名稱	作用	範例	
1Chapter 3 Car	取出 list 中第一個元素。也可取出 bBox 的左下點座標	A='(1 2 3 4) car（A）	⇨（1 2 3 4） ⇨ 1
Cdr	去掉 list 第一個元素之後的子 list	A='(1 2 3 4) car（A）	⇨（1 2 3 4） ⇨（2 3 4）
Nth	取出 list 中的第 n 個元素	A='(1 2 3 4) nth（3A）	⇨（1 2 3 4） ⇨ 3
member	判斷某個資料項目是否在串列中	A='(1 2 3 4) member（3A）	⇨（1 2 3 4） ⇨（3 4）
length	計算一個串列的元素個數	A='(1 2 3 4) member（3A）	⇨（1 2 3 4） ⇨（3 4）
xcoord	取出 x 座標值	pCoor=100:200 xCoord（pCoor）	⇨（100 200） ⇨ 100
ycoord	取出 y 座標值	pCoor=100:200 yCoord（pCoor）	⇨（100 200） ⇨ 200
cadr	右上點座標	pBox='((100 150)（200 500）) upperR=cadr（pBox）	⇨（200 500）
caar	左下點之 x 座標	lowLx=caar（pBox）	⇨ 100
cadar	左下點之 y 座標	lowLy=caar（pBox）	⇨ 150
caadr	右上點之 x 座標	upperRx=caadr（pBox）	⇨ 200
cadadr	右上點之 y 座標	upperRx=caadr（pBox）	⇨ 250

4.檔案輸出／輸入

本節介紹如何在 SKILL 程式裡面去存取一個 UNIX 底下的檔案， 並且介紹如何去產生一個特定格式的輸出檔案。

4.1 顯示資料

在 SKILL 的程式裡面可以使用 *print* 和 *println* 這兩個函式來顯示資料到螢幕上。其中 *println* 的函式會在輸出資料之後再加上一個“換行”的控制字元。例如：

print (“hello”) print(“hello”)

的結果是

“hello” ”hello”

而執行

println(”hello”) println(”hello”)

的結果是

“hello”

“hello”

另外可以用 *printf* 這個函式來產生格式化的輸出結果，其用法類似在 C 語言裡面的 *printf* 函式。我們用以下的例子來說明：

printf（”%-15s %-10d” layerName rectCount）

其中括號內的第一個引數是一個字串，稱爲控制字串。 在控制字串裡面的每一個%開頭到空白爲止的子字串，稱爲一個“指引”（directive），用以指示在控制字之後對應的資料將如何被顯示出來。

控制字串內的“指引”的文法型式如下

%【-】【寬度】【精確度】轉換碼

其中由中括號包含者表示是可有可無的。現在說明其意義：

代號	意義	
【-】	代表向左對齊	
【寬度】	資料顯示出來至少佔據的格數	
【精確度】	要顯示出來的資料字數	
轉換碼	整數:	d
	浮點數:	f
	字串／符號:	s
	字元:	c
	數值:	n
	串列:	L
	point 串列:	P
	bounding box 串列:	B

以下是另外一些例子：

```
aList = ’(5 10 15)
printf( ”\n A list: ”%L” aList)        ⇨t
A list: (5 10 15)
```

4.2 資料寫入檔案

要寫入資料到一個檔案有幾個步驟。首先，用 *outfile* 函式去取得一個檔案的輸出埠（output port）；其次，使用 *print*，*println*，或 *fprintf* 指令來寫入資料；最後再用 close 函式將輸出埠關閉。

以下是應的範例：

```
port1 = outfile( ”/temp/file1”)
println( list( ”a” ”b”) port1)
println( list( ”c” ”d”) port1)
close( port1)
```

結果在檔案 /temp/file1 內看到

（”a” ”b”）
（”c” ”d”）

不像 print 或 println 函式，fprintf 函式的輸出埠參數是放在第一個引數的位置。 其使用類似 printf 函式，舉例如下：

```
port1=outfile(”/temp/file1”)
I=10
fprintf(port1 ”A number:%d”  I)
close(port1)
```

結果在/temp/file1 裡面可以看到

```
A number:10
```

4.3 讀檔

要從檔案去讀取資料也有幾個步驟。首先，用 *inflie* 函式去取得一個檔案的輸入埠（input port）；其次，使用 *gets* 函式去一次讀取一行資料，或使用 fscanf 來讀取格式化的資料；最後再用 close 函式將輸入埠關閉。

以下是應的範例：

```
inport=infile (”/temp/file1”)
gets (nextLine inport)
println (nextLine)
close (inport）
```

注意，inport 是放在 gets 函數的第二個引數位置。接下來是 fscanf 的應用：

```
inport=infile(”/temp/file”)
fscanf ( inport”%s”  w)
println(w)
close(inport)
```

fscanf 的控制字串原理與前面介紹 printf 的類似。常用的轉換碼有 %d、 %f、與 %s，分別代表整數、浮點數、與字串。

5. 控制流程

就像所有的程式語言一樣， SKILL 也提供了一些流程控制的指令，以及關係運算子。在以下的小節中將會逐一介紹。

5.1 關係與邏輯運算子

在 SKILL 中的關係運算子如下表所列：

運算子	引數型態	對應之函式	範例	回傳值
<	numeric	lessp	1<2	t
			4<2	nil
<=	numeric	leqp	4<=6	t
>	numeric	greaterp	10 > 5	t
>=	numeric	geqp	3>=1	t
= =	numeric	equal	2.0= =2	t
	string		“xyz”= =”XYZ”	nil
	list			
!=	numeric	nequal	“xyz”!=”XYZ”	t
	string			
	list			

邏輯運算子主要有兩個：

運算子	引數型態	對應之函式	範例	回傳值
&&	General	and	1 && 2	2
			4 && 6	6
			a && nil	nil
			nil && t	nil
‖	General	or	3 ‖ 2	3
			5 ‖ 4	5
			a ‖ nil	t
			nil‖ a	t

5.2 if 函式

使用 if 指令大概是讀者最熟悉的控制命令了，幾乎所有程式語言都有。在 SKILL 裡面 if 是一個函式，其使用方法可由下面的例子得知

```
if（Shape= =”rect”
        then   println（”Rectangle”）
               count= count +1
        else
               println（”Not rectangle”）
               miscount= miscount + 1
）
```

注意！在 if 與左括弧之間不可以有空白，否則會出現錯誤訊息。

5.3 when 與 unless 函式

when 的用法如下例：

```
when(Shape= =”rect”
          println(”Rectangle”)
          ++count
        )；when
```

當 when 迴圈內的判斷式爲真時，則繼續執得迴圈內的敘述。而 unless 的用法也是當迴圈內的條件判斷式爲真時執行迴圈內的敘述。如下例：

```
unless(Shape= =”rect”)
          println(”Rectangle”)
          ++miscount
        )；unless
```

注意的是 *when* 與 *unless* 函式都會回傳值，亦即最後一回迴圈所計算出來的值或者是 nil。

5.4 case 函式

case 函式提供了多重分支（branching）的控制敘述，其用法如下例：

```
case( Letter
        （”a”   ++aCount   println（”A”）   ）
```

```
        （”b”   ++bCount   println（”B”）    ）
        （”c”   ++cCount   println（”C”）    ）
        （t     ++oCount   println（”Others”）
    ）；case
```

當上述 *case* 函式執行時，變數 *Shape* 的值會依序跟每一個分支部份（arm）的標記（label）來做比對（亦即”a”、”b”及”c”），如果相符則執行此一分支部份（arm）的敘述式。 如果所有的值都沒有匹配成功，則最後一標記爲 *t* 的分支部份的敘述會被執行。

如果有一分支部份的標記值是串列的話，SKILL 會逐一去比對串列裡面的元素，只要有一個比對成功，則此一分支部的敘述便會被執行。如下例：

```
case( Letter
        （”a”   ++aCount   println(”A”)                  ）
        （（”b” “c”）  ++bcCount   println(”B or C”)     ）
        （t   ++oCount   println(”Others”)               ）
    ）；case
```

只要 Letter 的值是”b”或”c”，第二個分支部的敘述都會被執行。

5.5 for 函式

for 函式提供迴圈控制之用，其使用的方式如下例：

```
R = 0
for( i   0   10
     R =R + i
     println(”The result is:” sum)
  )
⇨ t
```

其中的索引變數 *i* 的值在進入 *for* 迴圈時會被存起來，待離開迴圈之後再回復其值。在迴圈執行時 *i* 的值由 0 遞增到 10 ，每次增加 1。要注意 for 與左括弧 “（” 之間不可以留空白。

5.6 foreach 函式

foreach 函式提供一個很方便的方法來針對一個串列裡面所有的元素進行同樣的處理動作。下面是一個 foreach 的應用範例：

```
aCount = bCount = cCount = nCount = 0
letterList = ’(”a” “a” “c” “b” “c”)
```

```
foreach(   letter   letterList
      Case( letter
            (”a”     ++aCount )
            (”b”     ++bCount )
            (”c”     ++cCount )
          ) ;case
       ) ; foreach
```

執行時 foreach 函式時 letterList 裡面的元素會依序指定給 letter 變數，在每一個迴圈開始時 SKILL 會指定下一個 letterList 裡面的元素指定給 letter 變數

6.發展一個 SKILL 函式

要發展一個 SKILL 的函式需要幾項工作，分別在下列幾節中加以介紹。

6.1 群組化 SKILL 敘述

所謂將一堆 SKILL 敘述群組化的意思就是用左括號｛ 與右括號 ｝將一堆敘述包起來。 如此一來 SKILL 會將這堆敘述視爲一個大的敘述，而此一敘述執行時的回傳值就是裡面最後一個敘述執行的回傳值。 底下是一個集合敘述的例子：

```
bBoxHieght = {
     bBox = list( 200:250 250:400)
     lowLeft = car( bBox)
     upRight = cadr( bBox)
     lowLeftY = yCoord( lowLeft)
     upRightY = yCoord( upright)
     upRightY – lowLeftY   }
```

在這個例中，最後的回傳值就是最後一個敘述的值，我們將它指定給變數 *bBoxHeight*。

6.2 宣告一個 SKILL 函式

要以一個特定的名稱來參考一個敘述的群組，必須先用 procedure 的宣告來給定此一群組一個名稱。此一名稱加上敘述群組便構成一個 SKILL 的函式。此

一名稱爲函式名稱，而集合的敘述稱爲函式的本體（body），要執行一個函式必須使用函式名稱再加上（ ）。 同時，爲了使宣告的函式更有彈性，可以在函式名稱之後的小括弧內加入傳遞參數的宣告。 以下是一個函式宣告及呼叫函式的例子：

```
procedure(CalBBoxWidth( bBox)
        ll      =car( bBox)
        ur       =cadr( bBox)
        llx     =xCoord(ll)
        urx     =xCoord(ur)
        urx – llx
  ) ; procedure

bBox = list(100:100 200:300)
bBoxWidth= CalBBoxWidth(bBox)
```