1

开始学习 SKILL

1. 简介

多年以来，Cadence 公司的 CAD tool 一直是世界上使用最广泛，功能最强大的集成电路(Integrated Circuit)设计工具。而为了符合 IC 的复杂度越来越高，设计的困难度也越来越高的情况，Cadence 的 CAD 整合开发环境也越来越庞大，所提供的功能也日益强大，造成使用者在维护及管理上的一大负担。再则，每一家的设计公司的在设计的流程中多多少少都会有一些小步骤，无法用 Cadence tool 提供的基本做法来达成；或者是不同公司的 tools 之间资料转换的问题。工程师遇到此类问题可能需要透过人工的方式去完成連接设计流程中相連的两个步骤；或是干脆去开发一些小软件来完成这些特定的工作，而此时使用者可能会面臨如何将自己开发的软件的 I/O 与 Cadence tool 的整合环境相連结的问题。一般的做法是产生一些资料档来做资料交换的中介，这些资料档的格式可能是 Cadence 支持的一标准资料交换格式，也可以是使用者自定义的资料格式。这样子是一种间接的做法，因为使用者无法直接去存取 Cadence 环境的内部资料，所以在处理上的弹性会小很多，也较不方便。

为了方便使用者使用整个 Cadence tools 的整合开发环境，以解决上述的困扰，Cadence 公司遂发展了 SKILL 语言。SKILL 是一种高阶的、交谈式的语言，是用于 Cadence tool 的整合开发环境内的命令语言（command language）。SKILL 采用人工智能语言 LISP 的语法为藍本，再加上常用的 C 语言的部份语法设计而成。SKILL 语言提供许多的接口函数，能让使用者可以撰

写程序直接去存取 Cadence 整合环境内的电路资料内容； 也可以让使用者 去开发将自己开发的应用程序并入 Cadence tool 的整合环境里。有了 SKILL 言，使用者可以让 Cadence tool 更充份地融入整个设计流程之中，减少琐碎 的人工转换时间，提升公司的生产力。

1.1 LISP 式的语法

在 SKILL 里面，使用函数的呼叫方式可以有两种方式：

（一）Algebraic 表示形式，也就是

Func（arg1 arg2 ...）

（二）前置表示形式，此为 LISP 型式的语法

（Func arg1 arg2 ...）

程序是由叙述来组成的，正如在 LISP 语言里面一样，SKILL 的叙述是以串列（list） 的形式来表示。 如此的设计方式使得程序可以和资料用同样的方式来处理。 用户可以动态地建立、修改、或计算函数或表示式的值。

另外 ，在 SKILL 中不像一般的程序语言一样有提供字符这种资料形态，字符就是用符号本身来表示 ，例如字符 “A”就是用”A”这个符号（变数）来代表。

2. 快速浏览 SKILL

2.1 解释名词

本节将介绍一些专有名词：

名词	意义
OUTPUT	SKILL 程序执行结果可以显示到 xterm、设计窗口、档案、或是命令解译窗口 CIW（command interpreter window）
CIW	很多 Cadence 的应用程序之起始工作窗口，包含有一行 命令输入列、一个输出区域、一个功能选单列。
SKILL expression	呼叫一个 SKILL function 的程序叙述
SKILL function	一个有命名的、参数化的程序段

要启动一个 SKILL 的 function 有几种方式，在不同的 Cadence 的应用程序里， 用户可以透过 Bindkey, Form, Menu, 或 SKILL process 等来启动，其意义如下：

名词	意义
Bindkeys	将一个 function 与一个键盘的事件关联起来，当使用者引发一个键盘事件时，Cadence 软件便会计算其相閞联的 function
Form	有些函数需要使用者提供一些资料输入，通常是透过一个跳出式表格（pop-up form）来输入资料
Menu	当用户选择 menu 上的一个项目时 ，系统便会执行相关 之 SKILL 函数的计算
CIW	使用者可以直接在 CIW 输入一个 SKILL function 来得到一个立即的计算结果
SKILL process	使用者也可以透过一个 UNIX 环境底下的行程，来启动 Cadence 里的 interpreter，以便直译某些 SKILL 程序

所有的 SKILL 函数都会传回一值。在本书中我们将用 “Ð " 来表示函数的回传值。 在 SKILL 里面，大小写是不同的。 要呼叫一个 SKILL 的函数的 方式如下：

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

或者用：

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

注意的是，在函数名称与左括号之间不可以留空白。 函数的内容可以分成几行来写，不一定要在同一行才可以。同样地，几个函数也可以放在同一行上，但此时只有最后一个函数值会回传到屏幕上。

2.2 基本资料型态

在 SKILL 里面资料也是大小写不一的（case-sensitive），最简单的资料型态有整数、浮点数、与字串。字串用双引号 “ " 来括起来。

2.3 运算符

SKILL 提供了一组运算符，每一个运算符对应到一个 SKILL 的函数。下表是一些常用的运算符：

运算符（依优先级由大到小排列）	对应之 SKILL 函数	运算
**	expt	Arithmetic
*	times	“
/	quotient	“
+	plus	“
-	difference	“
++S, S++	preincrement, postincrement	“
= =	equal	tests for equality
! =	nequal	tests for equality
=	setq	Assignment

以下是一个应用运算符的例子：

x =5 y =6x+y

Ð11

2.4 变数

在 SKILL 的程序中你不需要去宣告变数， 一个变数的名称可以包含英文字母、数字、底线（__）、问号（？）。 变数名称的第一个字不可是数字。你可以用= 运算去指定一个变数的值。 你也可以直接键入一个变数名称来执行以取得其值。你也可以用 type 这个函数来取得变数的资料型态。

3. SKILL 串列

SKILL 串列 简单地說就是一些依序排列的 SKILL 资料对象。 一个串列内可以包含任何 SKILL 资料型态的资料，一个串列的资料表示式要用括号括起来。SKILL 允许有空的串列，表示成“（） " 或 *nil*。 串列 里面的元素也可以是另一个串列。以下是一些串列的例子：

串列	说明
（1 2 3）	一个包含 1 , 2, 3 三个整数的串列
（1）	一个包含 一个整数 1 的串列
（）	一个空的串列
（1（2 3）4）	一个串列 里面还包含另一个串列

至如何指定一个串列变数的值呢？举例如下：

S1= ' ("A" "B" "C")

请注意在左括号之前须加上一个单引号。

3.1 建立串列

有几个方法可以建立一个串列，举例如下：

1. 直接使用单引号，例如：

   '(2 4 6)　　Ð(2 4 6)

2. 呼叫使用串列 函数，例如：

   x=2　y=4

   list(x y 6)　　Ð(2 4 6)

3. 使用 cons 函数，例如：

   r = '(4 6)

   r =cons(2 r)　　Ð　(2 4 6)

4. 用 append 函数来合并两个串列：

   alist = '(1 2 3)

   blist = '(4 5 6)

   clist =append（alist blist）　　Ð(1 2 3 4 5 6)

3.2 串列与坐标表示

由串列衍生出来的实际资料表示有两个最主要的是坐标（coordinates）与界限方块（bounding box）。 对布局图（layout）而言， 吾人常用到的是这两样东西。在 SKILL 中 xy 坐标可用两个元素的串列来表示，例如

xVal=100

```
yVal=200
pCoordinate= xVal:yVal          Ð（100 200）
```

而 bounding box 是用左下及右上两点坐标来表示，例如

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

或者可以用下列方式来产生：

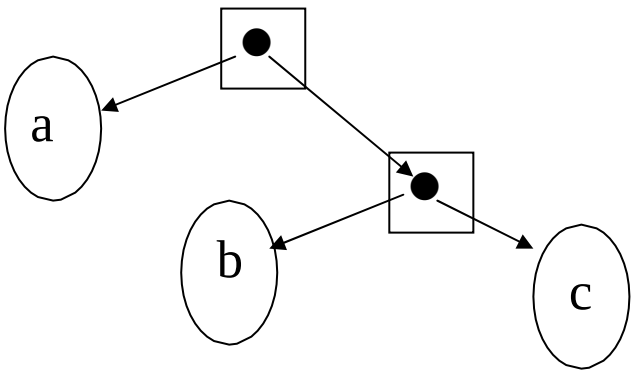
```
lowL = 300:400
upperR=500:600
bBox=list( lowerL upperR)
```

或者直接用双层串列

```
bBox= '((300 400) (500 600))
```

3.3 使用串列

在 SKILL 中， 一个串列的内部储存方式相当于是二元树（binary tree）的结构，每一个内点都相当于一个分支决定点（branch decision point），例如：有一个 list(a b c)，此一 list 在内部储存的方式可以下图表示



在 SKILL 语言中，提供了很多的 list 相关的函数，以下是较基本的：

函数名称	作用	范例	
1Chapter 3 Car	取出 list 中第一个元素。也可取出 bBox 的左下点坐标	A=’(1 2 3 4) car（A）	Ð（1 2 3 4） Ð 1
Cdr	去掉 list 第一个元素之后的子 list	A=’(1 2 3 4) car（A）	Ð（1 2 3 4） Ð（2 3 4）
Nth	取出 list 中的第 n 个元素	A=’(1 2 3 4) nth（3A）	Ð（1 2 3 4） Ð 3
member	判断某个资料项目是否在串列中	A=’(1 2 3 4) member（3A）	Ð（1 2 3 4） Ð（3 4）
length	计算一个串列的元素个数	A=’(1 2 3 4) member（3A）	Ð（1 2 3 4） Ð（3 4）
xcoord	取出 x 坐标值	pCoor=100:200 xCoord（pCoor）	Ð（100 200） Ð 100
ycoord	取出 y 坐标值	pCoor=100:200 yCoord（pCoor）	Ð（100 200） Ð 200
cadr	右上点坐标	pBox=’((100 150)（200 500）) upperR=cadr（pBox） Ð（200 500）	
caar	左下点之 x 坐标	lowLx=caar（pBox） Ð 100	
cadar	左下点之 y 坐标	lowLy=caar（pBox） Ð 150	
caadr	右上点之 x 坐标	upperRx=caadr（pBox） Ð 200	
cadadr	右上点之 y 坐标	upperRx=caadr（pBox） Ð 250	

4.档案输出／输入

本节介绍如何在 SKILL 程序里面去存取一个 UNIX 底下的档案， 并且介绍如何去产生一个特定格式的输出档案。

4.1 显示资料

在 SKILL 的程序里面可以使用 *print* 和 *println* 这两个函数来显示资料到萤幕上。其中 *println* 的函数会在输出资料之后再加上一个“换行 " 的控制字符。例如：

print (“hello”) print(“hello”)

的结果是

“hello” ”hello”

而执行

println(”hello”) println(”hello”)

的结果是

“hello”

“hello”

另外可以用 *printf* 这个函数来产生格式化的输出结果，其用法类似在 C 语言里面的 *printf* 函数。我们用以下的例子来说明：

printf（”%-15s %-10d” layerName rectCount） 其中括号内的第一个引数是一个字串，称为控制字串。在控制字串里面的每一个％开头到空白为止的子字串，称为一个“指引 "（directive），用以指示在控制 字之后对应的资料将如何被显示出来。

控制字串内的“指引 " 的文法型式如下

％【-】【宽度】【精确度】转换码

其中由中括号包含者表示是可有可无的。现在说明其意义：

代号	意义	
【-】	代表向左对齐	
【宽度】	资料显示出来至少占据的格数	
【精确度】	要显示出来的资料字数	
转换码	整数:	d
	浮点数:	f
	字串 / 符号:	s
	字符:	c
	数值:	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)        Ðt
A list: (5 10 15)
```

4.2 资料写入档案

要写入资料到一个档案有几个步骤。首先，用 *outfile* 函数去取得一个档案的输出端口（output port）；其次，使用 *print*，*println*，或 *fprintf* 指令来写入资料；最 后再用 close 函数将输出端口关闭。

以下是应的范例：

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

结果在档案 /temp/file1 内看到

（”a” ”b”）
（”c” ”d”）

不像 print 或 println 函数，fprintf 函数的输出端口参数是放在第一个引数的位置。 其使用类似 printf 函数，举例如下：

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

结果在/temp/file1 里面可以看到

```
A number:10
```

4.3 读档

要从档案去读取资料也有几个步骤。首先，用 *inflie* 函数去取得一个档案的输入端口（input port）；其次，使用 *gets* 函数去一次读取一行资料，或使用 fscanf 来读取格式化的资料；最后再用 close 函数将输入端口关闭。

以下是应的范例：

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

注意，inport 是放在 gets 函数的第二个引数位置。接下来是 fscanf 的应用：

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

fscanf 的控制字串原理与前面介绍 printf 的类似。常用的转换码有 %d、 %f、与 %s，分别代表整数、浮点数、与字串。

5. 控制流程

就像所有的程序语言一样， SKILL 也提供了一些流程控制的指令，以及关系运算符。在以下的小节中将会逐一介绍。

5.1 关系与逻辑运算符

在 SKILL 中的关系运算符如下表所列：

运算符	引数型态	对应之函数	范例	回传值
<	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			

逻辑运算符主要有两个：

运算符	引数型态	对应之函数	范例	回传值
&&	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 函数

使用 if 指令大概是读者最熟悉的控制命令了，几乎所有程序语言都有。在 SKILL 里面 if 是一个函数，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 与左括号之间不可以有空白，否则会出现错误讯息。

5.3 when 与 unless 函数

when 的用法如下例：

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

当 when 循环内的判断式为真时，则继续执得循环内的叙述。而 unless 的用法也是当循环内的条件判断式为真时执行循环内的叙述。如下例：

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

注意的是 *when* 与 *unless* 函数都会回传值，亦即最后一回循环所计算出来的值 或者是 nil。

5.4 case 函数

case 函数提供了多重分支（branching）的控制叙述，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
```

当上述 *case* 函数执行时，变数 *Shape* 的值会依序跟每一个分支部份（arm）的标记（label）来做比对（亦即”a”、”b”及”c”），如果相符则执行此一分支部份（arm）的叙述式。 如果所有的值都没有匹配成功，则最后一标记为 *t* 的分支部份的叙述会被执行。

如果有一分支部份的标记值是串列的话，SKILL 会逐一去比对串列里面的元素，只要有一个比对成功，则此一分支部的叙述便会被执行。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”，第二个分支部的叙述都会被执行。

5.5 for 函数

for 函数提供循环控制之用，其使用的方式如下例：

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

其中的索引变数 *i* 的值在进入 *for* 循环时会被存起来，待离开循环之后再回復其值。在循环执行时 *i* 的值由 0 递增到 10，每次增加 1。要注意 for 与左括 弧“（ " 之间不可以留空白。

5.6 foreach 函数

foreach 函数提供一个很方便的方法来针对一个串列里面所有的元素进行同样的处理动作。下面是一个 foreach 的应用范例：

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

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

执行时 foreach 函数时 letterList 里面的元素会依序指定给 letter 变数，在每一个循环开始时 SKILL 会指定下一个 letterList 里面的元素指定给 letter 变数

6.发展一个 SKILL 函数

要发展一个 SKILL 的函数需要几项工作，分别在下列几节中加以介绍。

6.1 群组化 SKILL 叙述

所谓将一堆 SKILL 叙述群组化的意思就是用左括号｛ 与右括号 ｝将一堆叙述包起来。 如此一来 SKILL 会将这堆叙述视为一个大的叙述，而此一叙述执行时的回传值就是里面最后一个叙述执行的回传值。 底下是一个集合叙述的例子：

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

在这个例中，最后的回传值就是最后一个叙述的值，我们将它指定给变数 *bBoxHeight*。

6.2 宣告一个 SKILL 函数

要以一个特定的名称来参考一个叙述的群组，必须先用 procedure 的宣告来给定此一群组一个名称。此一名称加上叙述群组便构成一个 SKILL 的函数。此

一名称为函数名称，而集合的叙述称为函数的本体（body），要执行一个函数必须使用函数名称再加上（ ）。 同时，为了使宣告的函数更有弹性，可以在函数名称之后的小括号内加入传递参数的宣告。 以下是一个函数声明及呼叫函数的例子：

```
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)
```