SoC in Real

VLSI Design/CAD 2007

Call for Papers

第18屆 超大型積體電路設計暨計算機輔助設計技術研討會

The 18th VLSI Design/CAD Symposium

2007年8月7日至2007年8月10日 花蓮美侖大飯店

超大型積體電路設計暨計算機輔助設計技術研討會是國內IC設計相關領域最具代表性的技術研討會，提供產、學、研各界發表、觀摩與討論最新科技發展之機會。本屆研討會主題為『*SoC in Real*』，會中將邀請多組國內外隊伍與公司進行系統動態展示，歡迎各位踴躍參加！ 另本次會議將頒發『最佳論文獎』，敬請踴躍投稿！

論文範圍 徵文範圍包括VLSI Design及CAD兩大領域，論文主題包含(但不限於)：

A. VLSI Design

- VLSI System and SoC Design
- Memory/CPU/DSP Cores
- Digital Design
- FPGA/CPLD Rapid Prototyping
- Analog/Mixed-Signal/RF ICs
- MEMS/Interface Design
- Signal Processing ICs
- Communication ICs
- Consumer Electronics

B. CAD

- SoC Design Methodology
- Modeling
- Simulation and Verification
- Logic and High-Level Synthesis
- Physical Design
- HW/SW Co-Design
- Testing/DFT/DFM/BIST
- Embedded Systems and Software
- Nano/Giga Scale CAD Tools

論文格式 限以英文書寫，請用A4大小，double-column 10-pt格式，最多四頁。論文書寫請參照IEEE標準格式，參考範例請見網頁，格式不符恕不接受投稿！

投稿方式 一律電子投稿，限用PDF檔案格式。投稿若獲接受，均以原稿刊登，恕不再接受修改。

重要日期

- 投稿截止日期：2007年5月27日
- 論文接受通知：2007年6月27日
- 註冊截止日期：2007年7月15日

指導委員會

主　　席：周景揚

委　　員：王進賢、何建明、吳誠文、李昆忠、李鎮宜、汪重光、林永隆、徐爵民、張世杰、陳良基、劉深淵、劉濱達、魏慶隆

<依姓氏筆劃順序排列>

大會組織

榮譽主席：張宗仁

大會主席：王朝欽

議程主席：蕭勝夫、鄭獻榮

總務主席：黃英哲、郭可驥、李淑敏

出版主席：陳儒雅

財務主席：張雲南

公關主席：R. Rieger

註冊主席：邱日清

展示主席：李志鵬

議程委員

王行健、王廷基、王益文、王惠貞、江正雄、吳文慶、吳安宇、呂良鴻、吳啓豐、呂學坤、李長龍、李泰成、李健雄、林榮彬、邱進峰、邱瀝毅、邱瀞德、姚嘉瑜、柯明道、柏振球、洪子聖、紀新洲、唐經洲、馬金溝、張耀文、梁伯嵩、許孟烈、許明華、郭建男、陳中和、陳竹一、陳伯奇、陳春僥、陳美麗、陳培殷、陳添福、陳調斌、陳巍仁、曾奕龍、馮武雄、黃俊郎、黃俊達、黃錫瑜、葉經緯、蔡宗漢、鄭國興、鄭福炯、賴永康、謝明得、簡韶逸、蘇培陞

<依姓氏筆劃順序排列>

大會網址 http://www.ee.nsysu.edu.tw/vlsi2007

主辦單位：國立中山大學電機工程系、國立中山大學資訊工程系、國立中山大學通訊工程研究所、台灣積體電路設計學會

協辦單位：教育部顧問室SOC總聯盟、國科會工程處工程科技推展中心、國家晶片系統設計中心(CIC)

聯 絡 人：趙芳頡小姐 Tel：07-5252000 ext. 4149 vlsi2007@ee.nsysu.edu.tw

戈麗安小姐 Tel：07-5254337

使用Synopsys軟體達成功率最佳化與分析之技術

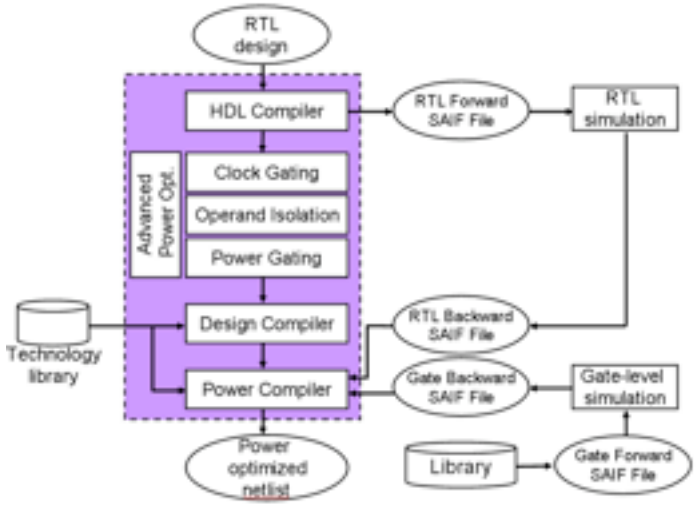
王旭昇 andy@cic.org.tw

摘要

近年來隨著製程不斷進步與IC設計度複雜度提昇，關於功率消耗問題已成為產品好壞重要關鍵之一。低消耗功率之設計其優點如節省包裝費用(Packaging Cost)、延長電池壽命(Battery Life)、降低能源損耗(Energy Saving)，及增加晶片可靠性(Reliability)等。因此對於如何達成低功率設計(Low Power Design)技術與準確地功率消耗分析(Power Analysis)已成為當前最重要課題!欲達成Low Power設計，本文將針對CIC現有的Synopsys Power Compiler軟體來達到Power Optimization。功率消耗主要可分為兩大類：動態功率(Dynamic Power)消耗與靜態功率(Leakage Power)消耗。本文將針對這兩大類之功率消耗問題，如何使用Power Compiler所具備之Power Optimization技術來達到Low Power Design及其實驗步驟作說明，並列舉幾個小範例，依據所述之步驟予以實現，觀察其效能與改善幅度。最後再使用PrimePower針對平均功率(Average Power)消耗、瞬間功率(Peak Power)消耗等項目做細部分析，達成Power sign-off之目的。

一、簡介

圖一為Synopsys目前所提供之Power Optimization技術與設計流程，其可分為兩大類。



圖一 - Power Optimization Methodology and Design Flow

1. 傳統功率最佳化流程

本流程可分為RTL Power Optimization、Gate-level Power Optimization兩種方法。前者是提供設計者對於自己的電路作初步的功率評估之用，當評估出之功率未達到預定目標，可提早建議設計者更換電路架構或設計。後者則是將電路初步合成完結果，以實際的Pattern作Gate-level Simulation得到整個電晶體電路on/off頻率紀錄於SAIF檔案中，提供Power Compiler

作Power Optimization與精確的功率評估。此外Power Compiler還可搭配Multi-Vt的Library來針對Leakage Power做最佳化。根據官方統計數據表示，使用Gate-level Power Optimization技術，可降低Dynamic Power消耗約1-5%，Leakage Power消耗約80%-95%。

2. 進階功率最佳化(Advanced Power Optimization)技術

設計者根據Power Compiler初步評估功率結果，若未能達成預期目標亦不想重新設計電路架構或改寫RTL Code，此時可以考量使用Synopsys現有之Advanced Power Optimization技術來改善。其Advanced Power Optimization技術有三種，茲分述如下。

2.1 Clock-Gating

當設計者RTL Code中，含有if enable active時才動作反之則不動作，類似這樣的coding style時，就可以透過Power Compiler的幾個簡易指令，可自動做出clock gating電路，根據官方數據可降低Dynamic Power消耗約15%-40%。

2.2 Operand Isolation

當電路中含有多個Component(如：加、減、乘、除運算元)，而這些Component並非一直執行運算，當不需計算時希望他能停下來休息，此時可以使用Operand Isolation技術，Power Compile會自動插入幾個AND或OR 閘(Gate)讓這些Component停下來休息，以降低Dynamic Power消耗，根據官方數據可降低Dynamic Power消耗約5%-20%。

2.3 Power Gating

當整個電路都停止運作時，此時仍有Leakage Power會消耗，消耗量之多寡完全影響到待機時間，因此對於Leakage Power消耗的問題不得不重視。Power Gating有Save/Restore兩種狀態。當進入sleep mode時，會將目前資料用high Vt的Cell Save起來，達到Low Leakage Power消耗；反之則將Save資料再Restore回Register，繼續執行運算。因此可達到Low Leakage Power消耗，且又不影響到效能。

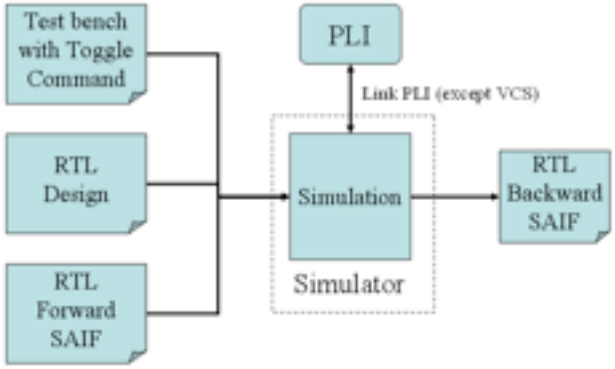
以上為Synopsys Power Compiler來達到Power Optimization功能簡介，本文將針對這些功能分成不同章節來做細部說明與其實作步驟。第二節中，我們將介紹RTL & Gate-level Power Optimization流程，並說明如何使用Multi-Vt Library搭配Gate-level Power Optimization來大幅度地改善Leakage Power之消耗。第三節即將介紹的進階功能，包括Clock Gating、Operand Isolation、Power Gating等技術來大幅度地降低功率消耗。緊接其下的第四節中，我們將針對搭配Power Compiler功能來做合成後之結果，使用PrimePower來做Power的Average Power、Peak Power分析，並產生simulation過程中功率消耗分布圖(waveform)來做細部分析。第五節，我們將針對實作步驟中用到的小範例，依二、三節所敘述之步驟將其製作出來產生實驗數據，並依此數據觀察Power Compiler各種最佳化技術所表現出之效能與改善幅度。最後是我們的結論。

二、傳統功率最佳化方法與設計流程

圖一中扣除Advanced Power Optimization技術的部份，即為Power Compiler基本流程。設計者欲達成Low Power電路設計，關於合成步驟是完全相同的，只需再搭配SAIF檔案來提供Power Compiler設計電路之on/off頻率資訊，即可精準地藉由Power Compiler的Power Optimization Techniques來降低Power的消耗量。從不同的SAIF產生階段而衍生出三種不同流程，包括RTL Power Optimization、Gate-level Dynamic Power Optimization、Gate-level multi-Vt Leakage Power Optimization，茲分述如下。

1. RTL Power Optimization Methodology

RTL Power Optimization顧名思義就是藉由RTL simulation來獲得hierarchy boundary及registers這些點的switch activity資訊。Power Compiler便可依此資訊做Power Optimization。但合成前所獲得之switch activity資訊僅限於boundary而並非所有點，因此該流程一般是作為功率評估之用。以下將針對該流程作細部說明與其實作步驟。



圖二 - Generate RTL Backward SAIF Flow

<1> 產生RTL Backward SAIF File

欲產生RTL Backward SAIF File其需具備之相關檔案與流程如圖二所示。

<1.1> RTL Fordward SAIF

<1.1.1> 保留原始RTL設計的Hierarchy名稱

dc_shell-t% set power_preserve_rtl_hier_names true

<1.1.2> 讀取RTL Verilog Netlist

dc_shell-t% read_verilog your_design.v

<1.1.3> 產生Fordward SAIF File

dc_shell-t% rtl2saif -output design_rtl.fsaif

<1.2> TestBench with Toggle Command

於testbench中新增下面幾行來計算switch activity資訊。

```
================================================================
    `ifdef RTL_SAIF
        initial begin
            $read_rtl_saif("design_rtl.fsaif ");
            $set_toggle_region(test.core);
            $toggle_start;
            #`Duration;
            $toggle_stop;
            $toggle_report("design_rtl.bsaif",1.0e-9,"test");
        end
    `endif
================================================================
```

其中

PS1：test為testfixture module name，core為待測之instance name。

PS2：Duration為總模擬時間，不可比模擬時間長，否則將導致SAIF檔案產生失敗。

PS3：1.0e-9為時間單位ns，此單位可從.lib file得知。

<1.3> RTL Simulation產生Backward SAIF File

unix% vcs -R testfixture.v design_rtl.v +define+RTL_SAIF

<2> RTL Power Optimization

<2.1> 讀取RTL Verilog Netlist

<2.2> 將RTL Design給予適當的Constraints並做合成

<2.3> 讀取RTL的Backward SAIF File

dc_shell-t> read_saif -input design_rtl.bsaif -inst test/core

<2.4> 設定Power Constraints

dc_shell-t> set_max_total_power 0 uw

<2.5> 開始進行Power Optimization

dc_shell-t> set compile_power_opto_only true

dc_shell-t> compile -inc

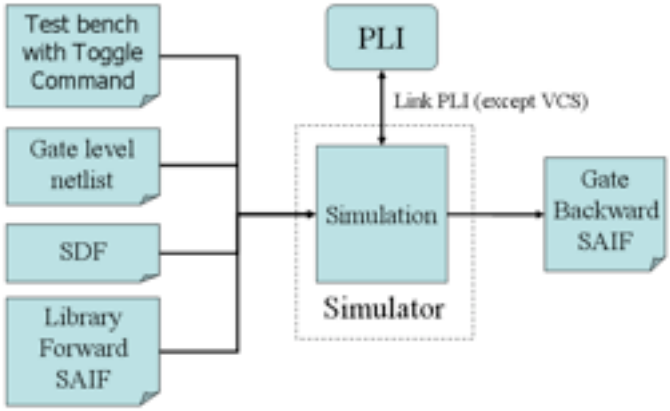
<2.6> 功率分析

dc_shell-t> report_power

<2.7> 將合成完結果存檔

三、Gate-level Dynamic Power Optimization Methodology

本流程為Power Compiler降低Dynamic Power之最基本功能，將初步合成完的Gate-level Netlist作Gate-level simulation讓電路中的每個Instance都有switch activity資訊，使得準確度較高。因此讀者可以直接使用Gate-level Power Optimization流程，而無需用到RTL Power Optimization。以下將針對該流程作細部說明與其實作步驟。



圖三 - Generate Gate-level Backward SAIF Flow

<1> 產生Gate-level Backward SAIF File

欲產生Gate-level Backward SAIF File其需具備之相關檔案與流程如圖三所示。

<1.1> Library Fordward SAIF

dc_shell-t% lib2saif -output design_gate.fsaif fast.db

PS：一般在合成時基於setup/hold time問題，會用到兩種環境所模擬出的元件庫，然而在Power部份，只需考量worst case的元件庫即可。

<1.2> 將Design做到初步合成結束產生Gate-level Netlist與其SDF檔案

<1.3> TestBench with Toggle Command

於testbench中新增下面幾行來計算switch activity資訊。

```
================================================================
    `ifdef GATE_SAIF
        initial begin
            $read_lib_saif("design_gate.fsaif");
            $set_toggle_region(test.core);
            $toggle_start;
            #`Duration;
            $toggle_stop;
            $toggle_report("design_gate.bsaif",1.0e-9,"test");
        end
```

```
    `endif
    `ifdef SDF

        initial $sdf_annotate(design.sdf, core);
    `endif
================================================================
```

其中

PS1：test為testfixture module name，core為待測之instance name。

PS2：Duration為總模擬時間，不可比模擬時間長，否則將導致SAIF檔案產生失敗。

PS3：1.0e-9為時間單位ns，此單位可從.lib file得知。

PS4：Gate-level Simulation需多加入Timing資訊.sdf檔案。

<1.4> Gate-level Simulation產生Backward SAIF File

unix% vcs -R testfixture.v design_gate.v －v tsmc13_neg.v +define+SDF +define+GATE_SAIF

<2> Gate-level Dynamic Power Optimization

<2.1> 讀取已初步合成完成檔案(db或ddc格式)

<2.2> 讀取Gate-level的Backward SAIF File

dc_shell-t> read_saif -input design_gate.bsaif -inst test/core

<2.3> 觀察是否所有點都含有Switch Activity資訊

dc_shell-t> report_saif

Object type	User Annotated (%)	Default Activity (%)	Propagated Activity (%)	Total
Nets	219(100.00%)	0(0.00%)	0(0.00%)	219
Ports	52(100.00%)	0(0.00%)	0(0.00%)	52
Pins	405(100.00%)	0(0.00%)	0(0.00%)	405

<2.4> 執行Gate-level Dynamic Power Optimization前之功率分析

dc_shell-t> report_power

<2.5> 設定Power Constraints

dc_shell-t> set_max_dynamic_power 0 uw -effort high

<2.6> 開始進行Power Optimization

dc_shell-t> set compile_power_opto_only true

dc_shell-t> compile -inc

<2.7> 執行Gate-level Dynamic Power Optimization後之功率分析

dc_shell-t> report_power

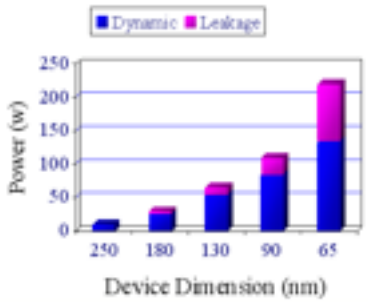
```
  Cell Internal Power  = 350.1141 uW   (26%)
  Net Switching Power  = 997.0524 uW   (74%)
                         ---------
Total Dynamic Power    =   1.3472 mW  (100%)

Cell Leakage Power     =  18.6784 uW
```

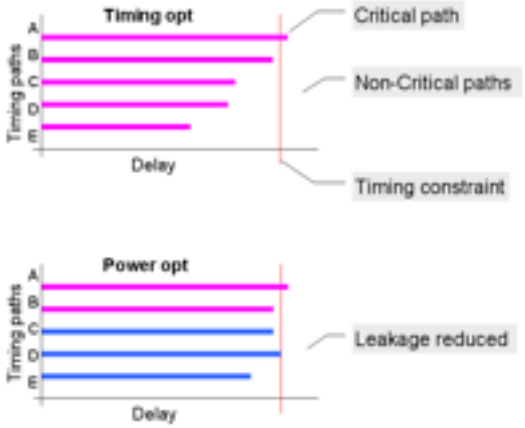
<2.8> 將合成完結果存檔

3. Gate-level Multi-Vt Leakage Power Optimization Methodology

隨著製程不斷演進，電晶體的臨界電壓(Threshold Voltage)越來越低，因而造成Leakage Power消耗量呈現指數型的上升，如圖四所示，因此我們不得不重視Leakage Power消耗問題。欲降低Leakage Power消耗量，首先需具備Multi-Vt的Library。高臨界電壓(Higher Vt)的Cell擁有Lower Leakage Power及較長的延遲時間(Longer delay)，反之低臨界電壓(Lower Vt)的Cell擁有Higher Leakage Power及較短的延遲時間(Shorter delay)的特性。因此有了這兩種Library，Power Compiler便可以利用這與生俱來的特性，將電路中的關鍵路徑(Critical Path)置換成速度快的Low Vt Cell，其餘Non-Critical Path則置換成Lower Leakage Power的High Vt Cell，如圖五所示。接下來將說明如何降低Leakage Power消耗之操作步驟。



圖四 - Device Scaling Down and Power Consumption



圖五 - Saving Leakage Power on Non-critical Paths (資料來源Synopsys)

<1> 設定Multi-Vt Library (元件庫來源Synopsys)

dc_shell-t% set link_library " lvt.db hvt.db dw_foundation.sldb"

dc_shell-t% set target_library " lvt.db hvt.db"

<2> 讀取已初步合成完成檔案(db或ddc格式)

<3> 讀取Gate-level的Backward SAIF File

<4> 執行Gate-level Leakage Power Optimization前之功率分析

```
  Cell Internal Power  = 189.4462 uW   (59%)
  Net Switching Power  = 131.0786 uW   (41%)
                         ---------
Total Dynamic Power    = 320.5248 uW  (100%)

Cell Leakage Power     =  12.0397 uW
```

<5> 設定Power Constraints

dc_shell-t> set_max_leakage_power 0 uw -effort high

<6> 開始進行Power Optimization

dc_shell-t> set compile_power_opto_only true

dc_shell-t> compile -inc

<7> 執行Gate-level Leakage Power Optimization後之功率分析

dc_shell-t> report_power

<8> 將合成完結果存檔

四、進階功率最佳化技術

欲達到更低的功率消耗，本節將介紹三種功率最佳化進階技法，包括降低Dynamic Power的Clock Gating、Operand Isolation及Leakage Power的Power Gating。這些技法共同特色就是在RTL Level時就要預先設定處理的。第一次合成後對於功率消耗問題的改善，可立即達到一定水準，再搭配前一節所提的Gate-level Power Optimization技法，便可再度提升Power Optimization的品質。關於這些技法原理與實作步驟茲分述如下。

1. Clock-Gating Technique

使用Clock-Gating(CG)技術降低Dynamic Power，效果相當好，但並非所有的設計都可以使用的，有其語法上的限制。使用Clock-Gating必要條件為設計中要含enable訊號，當enable訊號active時可接受新的一筆資料輸入，反之維持之前的訊號。圖六我們舉了幾個符合Clock-Gating Coding Style的標準範例，這些Coding Style看似不同，其實電路架構都是相似的如圖七(a)所示。若設計中擁有圖七(a)電路架構者，方可利用Power Compiler的簡單幾道指令便可自動做出Clock-Gating電路架構，如圖七(b)所示。以下將針對本技術作細部說明與其實作步驟。

If statement

```
always@ (posedge clk)
    if (enable) Q <= D_in;
```

Conditional Assignment

```
always@ (posedge clk)
    Q <=(enable)? D_in : Q;
```

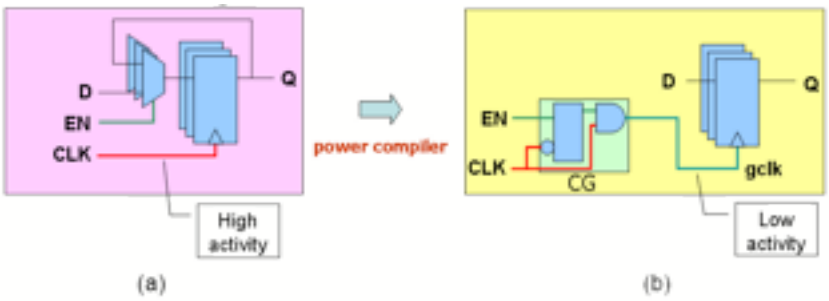
If + Loop statement

```
always @ (posedge clk)
  if (enable)
    for (i=0; i<8; i=i+1)
      s[i] = a[i] ^ b[i];
```

Case statement

```
always@ (posedge clk)
    case (enable)
        1'b1: Q  <=  D_in;
        1'b0: Q  <=  Q;
    endcase
```

圖六 - Standard Clock-Gating Coding Style



圖七 - (a) No Clock Gating (b) Automatic Generated CG Structure by Power Compiler

<1> 讀取RTL Verilog Netlist並給予合成時所需之適當的Constraints

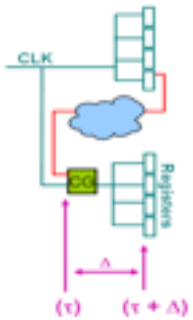
<2> 使用Power Compiler的語法自動插入Clock-Gating電路

<2.1> 設定Clock-Gating Style

dc_shell-t> set_clock_gating_style -max_fanout 16 -num_stage 1
-sequential_cell latch -setup 1.3
-control_point before -control_signal se

<2.1.1> -sequential_cell

在預設值中CG Cell的形式就是自動使用latch作為glitch prevention之用，目的是讓CLK為high，且當EN訊號同時也觸發時，透過latch的保護防止EN訊號在這週期內訊號發生擾動情形，而造成gclk訊號錯誤，如圖七(b)所示。另外此參數後面也可以接上”none”，但必須注意到，EN訊號必須是由一個暫存器(Register)所送出，且該Register必須與Gated Clock之緣觸發方向要相同，才能予以使用。



圖八 Clock Tree Synthesis and CG

<2.1.2> -setup (ideal_setup+△)

如圖八所示，在合成時clock訊號到達所有Register會假設是同步到達，雖可透過set_clock_latency指令來估測此時間，然而如果將CG Cell問題考量進來，該時間只估測到CG之前，CG之後則必須透過此參數來設定。在本指令中設定-setup 1.3意為 τ =1 (clock latency time)，△=0.3。另外設定完成後，需要再設定如下，△=0.3之時間才會計算進去。

dc_shell-t> propagate_constraints -gate_clock

<2.1.3> -max_fanout 16

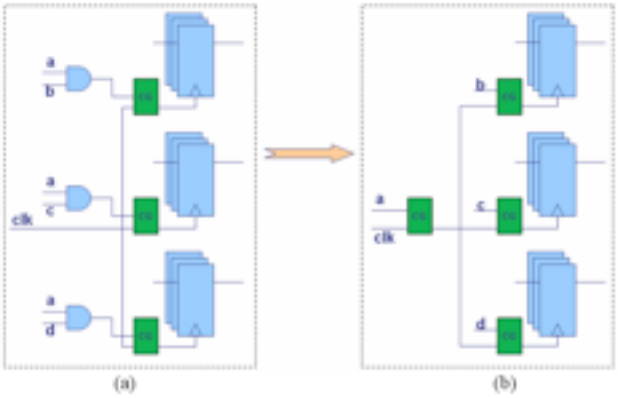
意為一個CG Cell後面所連接之Register個數限制在16個以內。

<2.1.4> -num_stage

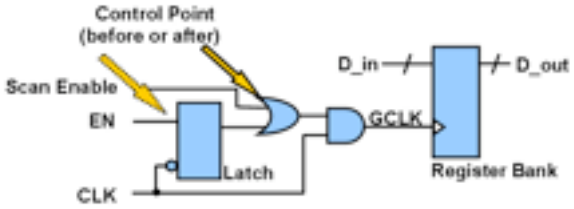
用來設定clk訊號到達Register共拆成幾個CG Cell。本功能主要是使用於當enable有多個時，設定不同個數的-num_stage，Power Compiler會將共同項的部份做分解其結果如圖九所示。

<2.1.5> -Control Point and －Control Signal

當電路中同時含有測試電路與CG電路時，CG Cell內涵latch電路會導致Scan Chain之生成有極大問題。透過-control point before (or after)，Power Compiler會自動在latch前或後方加上一個OR Gate，OR Gate的一端接上EN訊號，使用-control signal參數告知Scan Enable訊號名稱好讓另一端Tool自動接上Scan Enable訊號，如圖十所示。當進入Test Mode時，Scan Enable訊號為high，將導致GCLK與CLK訊號是相同的，CG Cell動作不會影響GCLK。



圖九 - Multi Stage Clock Gating (a) num_stage=1 (b) num_stage=2



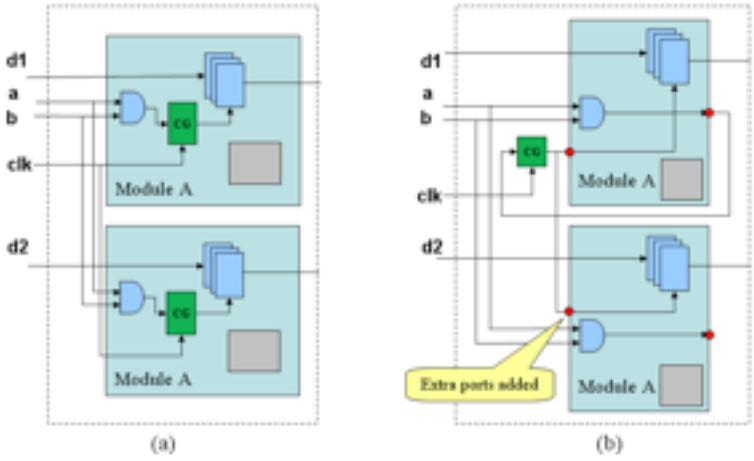
圖十 - Control Point For Testability

<2.2> 插入CG Cell

dc_shell-xg-t> insert_clock_gating －global －module

<2.2.1> -global

關於Clock-Gating可分為Centralize與Distribution架構，若使用-global參數，可讓CG架構成為Centralize形式，反之則是Distribution架構，如圖十一所示。此功能僅限於XG Mode方可使用。



圖十一 - (a) Distribution Structure (b) Centralize Structure (use －global parameter)

```
assign  gclk = Clock & Enable;

always @(posedge gclk or negedge Reset)
begin
    if (!Reset)
        Data_Out <= 8'b0;
    else
        Data_Out <= Data_Out + 8'b1;
end
```

圖十二 - Manual CG Coding Style

<2.2.2> -module

若設計者的Gated Clock是用RTL撰寫而成的，其形式如圖十二所示。此時若希望Power Compiler將此Code用CG予以置換，只需多加個-module參數即可達成。

<2.2.3> insert_clock_gating report

```
Flip-Flops                          Banks                 Bit-Width
                                 number  percentage      number  percentage
Clock gated (total):                2          66          32          66
Clock not gated because
   Bank was excluded:               0           0           0           0
   Bank width too small:            0           0           0           0
   Activity too low:                0           0           0           0
   Enable condition not met:        1          33          16          33
   Setup condition violated:        0           0           0           0
Total:                              3         100          48         100

Clock gates in design                                 number  percentage
   Replaced clock gates:                                 0           0
   Inserted clock gates:                                 2         100
Total:                                                   2         100
```

<3> 進行合成後存檔，之後接續Gate-level Power Optimization流程

2. Operand Isolation Technique

Operand Isolation(O.I.)顧名思義就是將電路中的所有算術運算(Arithmetic Operator)單元或階層式組合邏輯元件(Combinational Hierarchy Cell)當無必要做運算時，可使用O.I.方法自動插入Isolation Cell，讓這些無需計算的元件(Componet)暫時停止計算，達到降低Dynamic Power消耗之目的。O.I.技術並非所有電路皆可使用，必須擁有Observability Don't Care Condition (ODC)電路。如圖十三(a)所示，加法器Add_0做有效的(valid)運算唯獨當SEL_0、SEL_1分別選到1、0。倘偌該加法器沒被SEL訊號選到，該Component則無計算之必要，此時可以透過Power Compiler自動插入Isolation Cell，即用簡單的AND、OR邏輯閘來擋掉DATA輸入，讓該Component之電晶體電路不會發生充放電情形，自然就可以少去不必要的Dynamic Power消耗，如圖十三(b)所示。以下將針對本技術作細部說明與其實作步驟。

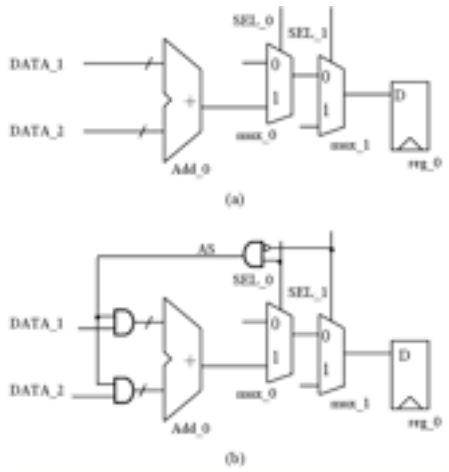
<1> 開啟OI功能

dc_shell-t> set do_operand_isolation true

<2> 讀取RTL Verilog Netlist並給予合成時所需之適當的Constraints

<3> 設定欲使用Operand Isolation之邏輯閘形式

dc_shell-t> set_operand_isolation_style -logic adaptive -verbose



圖十三 - (a) Original Circuit Example (b) Insert Isolation Logic to Apply Operand Isolation

<3.1> -logic [AND | OR | adaptive]

透過此參數可以設定O.I. style指定使用AND或OR閘。在此強烈建議使用adaptive選項由Power Compiler自動決定該使用哪一個邏輯閘最為妥當。Power Compiler取決最適當之邏輯閘方式為當輸入訊號為0的機率低於50%，則自動挑選AND閘，反之若高於50%，則自動挑選OR閘，目的為了降低O.I. Cell功率消耗。

<3.2> -user_directives

本參數不使用表示電路中的所有Component皆需使用O.I.技術。若使用表示告知Power Compiler手動指定需使用O.I.技術的Component。手動指定方法有二：

<3.2.1> 使用Power Compiler指令指定

dc_shell-t> set_operand_isolation_cell {“U1”}

<3.2.2> 使用RTL Pragma指定

Verilog => P=a+b; //synopsys isolate_operand

VHDL => P=a+b; --pragma isolate_operand

<4> 設定Operand Isolation 可接受之Slack基準值

dc_shell-t> set_operand_isolation_slack 0.6 – weight 1

<4.1> slack = 0.6

Slack=0.6意為當Power Compiler加入Operand Isolation Cell後，會導致時序增加一些，若加入的O.I. Cell累積時脈多增加0.6ns以上，針對該Component將不採用Operand Isolation。

<4.2> -weight <floating>

本參數用來控制該指令執行之強弱程度，當weight=0表示該Component Timing Path不管slack值是否有超過設定值一概插入O.I. Cell，當weight=1表示大於slack設定值就不使用O.I.技術。

<5> 進行合成

<6> Report Operand Isolation

dc_shell-t> report_operand_isolation

該範例中有四個ALU，一個MUX，因此Power Compiler只有針對四個Component插入Operand Isolation Cell，共計花費112個Cell。

```
                Operand Isolation Summary
-----------------------------------------------------
|    Isolation Style                 |     adaptive     |
|    Isolation Method                |     automatic    |
|    Number of Isolation gates       |        112       |
|    Number of Isolated objects      |     5 (100.00%)  |
|               operators            |     4 (80.00%)   |
|               hierarchical cells   |     0 (0.00%)    |
|               ungrouped objects    |     1 (20.00%)   |
|    Number of Unisolated objects    |     0 (0.00%)    |
|               operators            |     0 (0.00%)    |
|               hierarchical cells   |     0 (0.00%)    |
-----------------------------------------------------
```

<7> Report Power w/o Operand Isolation

```
  Cell Internal Power  = 411.8463 uW   (31%)
  Net Switching Power  = 914.5862 uW   (69%)
                         ---------
Total Dynamic Power    =   1.3264 mW  (100%)

Cell Leakage Power     =  19.3570 uW
```

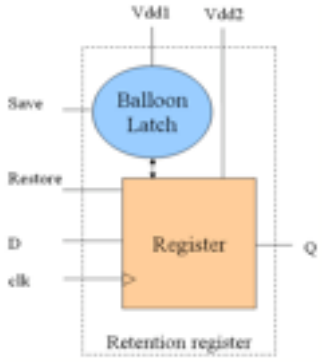
<8> Report Power w/ Operand Isolation

```
  Cell Internal Power  = 140.1330 uW   (30%)
  Net Switching Power  = 334.1541 uW   (70%)
                         ---------
Total Dynamic Power    = 474.2872 uW  (100%)

Cell Leakage Power     =  20.7070 uW
```

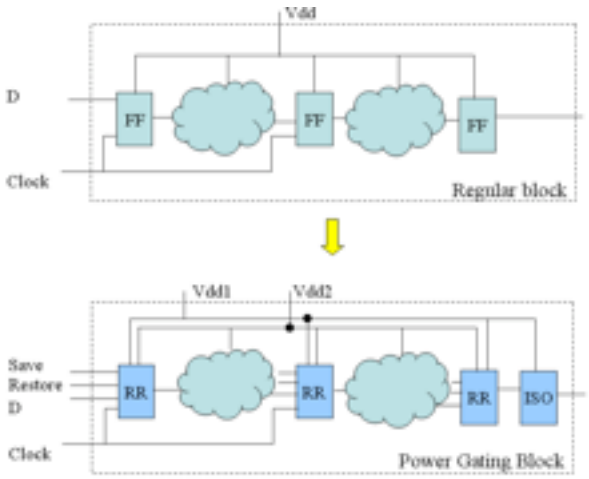
3. Power Gating Technique

Power Gating是一種降低Leakage Power的技術，使用此技術時必須具備兩種特殊Cell：Retention Register、Isolation Cell。Retention Register又可以稱作是MTCMOS(Multi-Threshold CMOS)，其結構如圖十四所示，內含兩主體架構：Balloon Latch、Regular Register。Balloon Latch其為High-Vth MOS Latch Circuit，當進入Sleep Mode時，會將目前Regular Register的內容值儲存到Balloon Latch中，此時VDD2電源整個會被cut-off掉，且Balloon Latch為High-Vth，因此可降低Leakage Power消耗；當進入Active Mode，Balloon Latch的內容值會自動再存回Regular



圖十四 - Retention Register Struct

Register，Regular Register是Low-Vth MOS，因此將Regular Register置換成Retention Register並不影響原始的效能表現。一使用Power Gating Technique之簡單範例如圖十五所示，原始電路中只含有一組Power及Regular Register，當使用Synopsys Power Gating Technique時會將Regular Register置換成Retention Register，並使用到兩組Power，一組提供Balloon Latch進入Sleep Mode時儲存值使用，另一組提供Regular Register使用，當Sleep Mode啟動時會將VDD2電源cut-off掉。此外，Power Compiler會在電路中的每個Block輸出前加上一個Isolation Cell，其目的是為了當該Block進入Sleep Mode時，能繼續Hold住之前輸出值，讓其他未進入Sleep Mode之Block能正常運作用。由於CIC尚未擁有Power Gating Cell，因此關於使用Power Gating技術之操作步驟概述如下。



圖十五 - One Example using Power Gating Technique

<1> 將含有Retention Register library加入至target_library裡

dc_shell-t> set target_library “retention.db slow.db fast.db"

<2> 讀取RTL Verilog Netlist並給予合成時所需之適當的Constraints

<3> 開啟Power Gating功能

dc_shell-t> set power_enable_power_gating true

<4> 設定Power Gating Cell Type及設定欲置換Retention Register之區域

dc_shell-t> set_power_gating_style －type ret1 －hdl_block reg16c_proc

<4.1> －type

跟Power Compiler指定要使用哪一個Retention Register，可透過此參數直接設定。如圖十六(a)所示，欲使用DFF_RT，就在該參數後面打上”ret1”即可。

<4.2> －hdl_block

該參數若不使用，表示電路中之所有Regular Register全部置換成Retention Register。假若欲設定局部區域，可使用該參數並搭配RTL named block方式做指定，如圖十六(b)所示，named block為reg16c_proc，則在該參數之後打上該名稱，即可完成置換區域設定。

```
Cell (DFF_RT)
---
power_gating_cell : "ret1";
---
pin (sleep){
---
power_gating_pin (power_pin1,
"0");}
pin (wake){
---
power_gating_pin (power_pin2,
"0";}
```

(a)

```
module reg16c (regin, regout, clk, gate, clr);
input gate;
input clr;
input clk;
input [15:0] regin;
output [15:0] regout;
reg [15:0] regout;

always @(posedge clk) begin: reg16c_proc
if (clr) regout = #1 16'b0;
else if (gate) regout = #1 regin;
end

endmodule
```

(b)

圖十六 - (a) Retention Register Library Example (b) RTL Example with a named block

<5> 建立新的Port作為Power Gating控制Sleep Mode使用

dc_shell-t> create_port －dir in SAVE

dc_shell-t> create_port －dir in RESTORE

<6> 開始進行合成

<7> 將Sleep Mode控制訊號與Retention Register做連接

dc_shell-t> hookup_power_gating_ports -type ret1 －port {SAVE RESTORE}

<8> 觀察Power Gating之連接情形使用電路圖或Report Power Gating

dc_shell-t> report_power_gating

```
--------------------------------------------------------------------------------------------------------
                                        Power Gating Cell Report
--------------------------------------------------------------------------------------------------------
Cell Name (Library Cell Name)           |Power Gating Style     |Power Gating Pin       |Signal
--------------------------------------------------------------------------------------------------------
y_reg_0_ (TSC10RT_CF)                   |ret1                   |REST (2)               |RESTOREn
                                        |                       |SAVE (1)               |SAVEn
y_reg_1_ (TSC10RT_CF)                   |ret1                   |REST (2)               |RESTOREn
                                        |                       |SAVE (1)               |SAVEn
y_reg_2_ (TSC10RT_CF)                   |ret1                   |REST (2)               |RESTOREn
                                        |                       |SAVE (1)               |SAVEn

                                        ...
```

五、功率分析

為何要用PrimePower做Power Analysis而不直接使用Power Compiler做分析? Power Compiler如前兩節所述，主要作為Power Optimization工具，Power Report功能陽春，只能秀出Average Power消耗，無法再做更進一步地Power Analysis。Average Power消耗定義為總能量消耗(Total Energy)除以總模擬時間(Total Simulation Time)，因此每套軟體都有這功能。PrimePower是以事件發生為基準(event based)做功率消耗分析，Small Time Interval精確度可達10-18去記載該時間內所消耗之功率，因此可以觀察出電路之Peak Power值，此外PrimePower也可以針對每個Hierarchy Block做細部的Power Analysis。關於PrimePower之Power Analysis與操作步驟說明如下。

<1> 於.synopsys_pp.setup檔案中設定含Power資訊之Library

pp_shell> set search_path ". ../syn_dc/model"

pp_shell> set link_library "* fast.db "

<2> 開啟PrimePower軟體

unix% primepower -64 -gui

<3> 讀取合成完或佈局(layout)完之Netlist

<4> 設定Primary Input Transition與Output Loading

設定此兩參數提供PrimePower查Power Model表計算Power之用。

<4.1> Input Transition

pp_shell> set_input_transition 0.2 [all_inputs]

<4.2> Loading

<4.2.1> Wire Load Model

前端設計之際尚未含有實際繞線(Routing)資訊，可用Wire Load Model取代之。

<4.2.2> Parasitics Annotation from Physical Compiler or APR Tool

pp_shell> read_parasitics design.spec

<5> 讀取SDF資訊提供PrimePower Timing 資訊

pp_shell> read_sdf design.sdf

<6> 讀取Switch Activity資訊

pp_shell> read_vcd design.vcd -strip_path test/top

PS1：test為testfixture module name，top為待測之instance name

PS2：PrimePower可以接受Event Based的VCD格式，也可以接受Toggle Rate Based的SAIF格式。

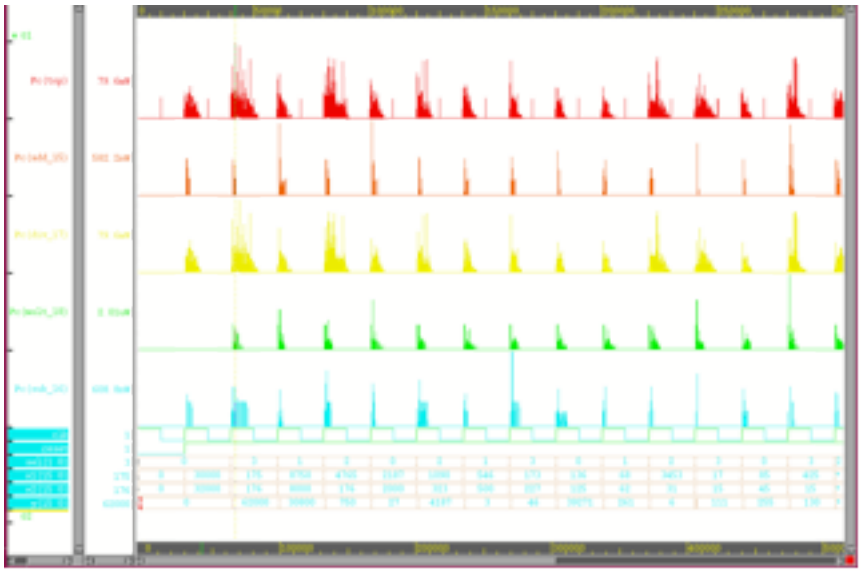
PS3：當Design中有Hierarchy架構就必須使用strip_path參數。

<7> 設定Power Waveform輸出格式

pp_shell> set_waveform_options -interval 0.01 -format fsdb – file pwr1

<7.1> -interval Value(Value >= 10-18)

PrimePower可將計算完之功率消耗在每隔interval所設定值紀錄一筆該時間內功率消耗資訊，紀錄至整個模擬時間結束為止，儲存成waveform格式。設計者可以搭配Function Simulation之波形與Power Waveform並列作功率消耗上的觀察，例如:本範例中在所有輸入Pattern當中，若輸入176除以175時，可以找到瞬間最大功率消耗高達79.6mw，發生時間點為43ns時，如圖十七所示。



圖十七 - Power Waveform and Function Waveform

<7.2> -format fsdb

該格式可以使用SpringSoft nWave開啟。

<7.3> -file pwr1

指定Power Waveform的輸出檔名。

<8> 計算功率

pp_shell> calculate_power -waveform –statistics

<8.1> -waveform

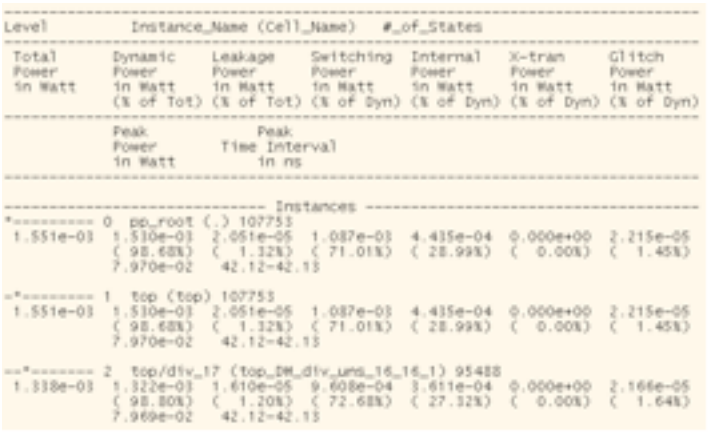
計算功率時同時產生fsdb格式的Power Waveform檔案。該參數需搭配set_waveform_options一起使用。

<8.2> -statistics

印出統計資訊。

<9> Power Report from PrimePower GUI

pp_shell> report_power –hier 2



六、實作結果

在第二、三節中已介紹過各種功率最佳化之技術，其中RTL Power Optimization Methodology只用來作功率估測之用、Power Gating尚未有Libraray可用，因此在本節將只針對四大項技術來評斷Power Compiler在功率改善方面成效如何。作者在此針對這四種技術分別製作了四個簡單的電路做為功率改善評估，這些數據已在二、三節Report Power中提出，在此將數據整理成如表一所示，以方便讀者針對不同最佳化技術能大致了解功率改善空間與其幅度，作為使用上的參考依據，同時再與官方促供之數據做簡易的比較。表一中不難看出所有Power Optimization技術與官方數據大約相同，而Operand Isolation其效能與設計相關(depends on design)。所舉範例整個電路都是ALU，故表現上遠比官方數據來的強眼。

表一 - Power Improve Capability with Synopsys Power Compiler Techniques

Power Opt. Technique	Before Opt.	After Opt.	Vendor Data	Improve
Gate-level Dynamic Power	1435.9 uw	1347.2 uw	1 - 5%	6.17%
Gate-level Leakage Power	12.039 uw	656.393 nw	80 – 95%	94.55%
Clock-Gating	672.668 uw	546.119 uw	15 – 40%	18.81%
Operand Isolation	1326.4 uw	474.287 uw	5 - 20%	64.24%

七、結論

實作結果顯示，Power Compiler提供相當多的Power Optimization技術，且這些最佳化技術依Design特性有時還可同時使用，更是展現出Power Optimization之優越表現。雖然工業界使用Power Compiler進行Power Optimization已經多年，但學術界使用者仍寥寥可數，希望此篇文章對於欲尋求低功率設計之研究人員，能提供相當程度的幫助。

八、參考文獻

[1] Synopsys, “Power Compiler User Guide Version 2005.09,”

[2] Synopsys, “PrimePower Manual Version 2005.09,”

[3] Synopsys, “Power Compiler Workshop Student Guide Version 2004.06,”

[4] Synopsys, “PrimePower Workshop Student Guide Version 2004.06,”

2007晶片製作成果發表會

為展示國內學術界透過本中心製作晶片之研發成果與促進產學研交流，本中心訂於96年5月3日(四)假奈米電子研究大樓舉辦晶片製作成果發表會，會中除頒獎予優良晶片設計者之外，並邀請設計者於現場作口頭報告或海報展示；此外，亦邀請國內、外相關領域之學者專家進行六場精闢演講(含Design Flow、System、Measurement等主題)及舉辦一場製程控管說明會，議程敬請詳參附件。

歡迎各界先進與朋友撥冗蒞臨指導，一起共襄盛舉！

費用：免費 (停車費用煩請自理)

報名方式：一律採用網路報名(http://www.cic.org.tw/workshop_cis)

報名日期：2007年4月2日起至4月27日止；名額有限(150人)，額滿即止。

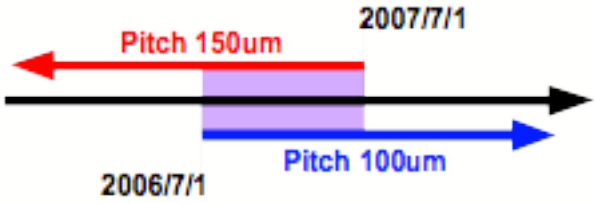
聯絡人：張淑慧小姐 (電話：03-5773693*127；傳真：03-5783372)

財團法人國家實驗研究院國家晶片系統設計中心
高頻量測服務-量測形式與佈局規則

高頻量測實驗室停止受理2007/4月之後下線梯次佈局為pitch 150μm之量測

為使晶片設計不因pad layout導致面積浪費，高頻量測實驗室2007/7/1起將停止受理2007/4月之後下線梯次(T18-96C 、T13RF-96B、T13L-96B、SiG35-96B、D35-96C、P15-96A、MEMS18-96C、MEMS35-96C)下線佈局為pitch 150μm之量測，僅提供pitch均為100μm的DC與RF探針，以預期縮小整個晶片面積。

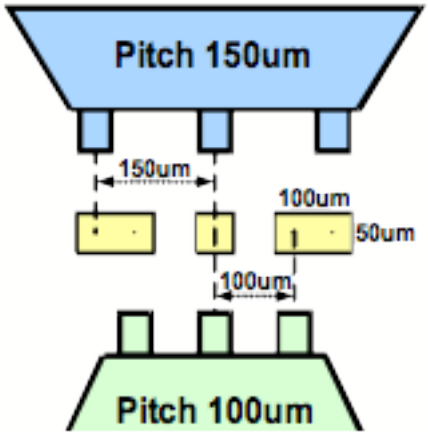
下圖列出實驗室探針之規劃時程圖。



因應探針替換，底下針對量測環境量測形式與pad layout規則作更新說明:

1. On Wafer Measurement

如果設計者晶片量測時所使用到之探針為3pin，例如RF Single-ended (GSG)或是DC 3pin (PGP)，在此建議將Ground Pad如下圖所示佈局(50um×100um)，如此在探針替換的過渡時期將不會受到探針pitch差異的影響，但注意此佈局只適用於3pin探針，大於3pin者(例如RF Differential)不適用。



- 單一Pad可允許之最小尺寸為50μm×50μm。
- RF Single-ended探針配置為GSG，Differential為GSGSG。探針方向儘可能為東西向。
- DC 探針配置為3pin(PGP)，6pin(PGPPGP)，9pin(undefined ground)。
  探針方向儘可能為南北向。
- 相鄰不同向Pad最近距離需大於150μm，以避免探針撞擊。

注意：若探針pitch不符合規則，需自行攜帶探針，填寫切結書，否則不予量測。

財團法人國家實驗研究院國家晶片系統設計中心

探針使用切結書

立書人(申請者)----------(以下簡稱甲方)委託「國家晶片系統設計中心」(以下簡稱乙方)，進行高頻探針量測。

一、委託內容：

- On Wafer Measurement
- On Wafer Measurement with PCB Bias Network

量測電路名稱：----------

二、甲方自行提供高頻探針或直流探針同意委託乙方進行量測，使用上所造成的損傷，甲方不得追究一切法律及賠償之責任。

三、高頻探針之量測必須由本中心工作人員操作。

四、本切結書內容，如有虛偽不實之情事，致使乙方遭受損害，甲方願負完全責任。

此致

財團法人國家晶片系統設計中心

立書人(申請者)：

服務機關：----------

姓　　名：----------(簽章)

地　　址：----------

聯絡電話：----------

傳真號碼：----------

E-mail　：----------

中 華 民 國　　年　　月　　日