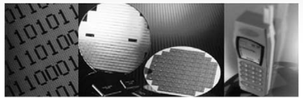
Digital IC Design
數位積體電路設計

VLSI Technology



資工系學生的硬體設計能力培養

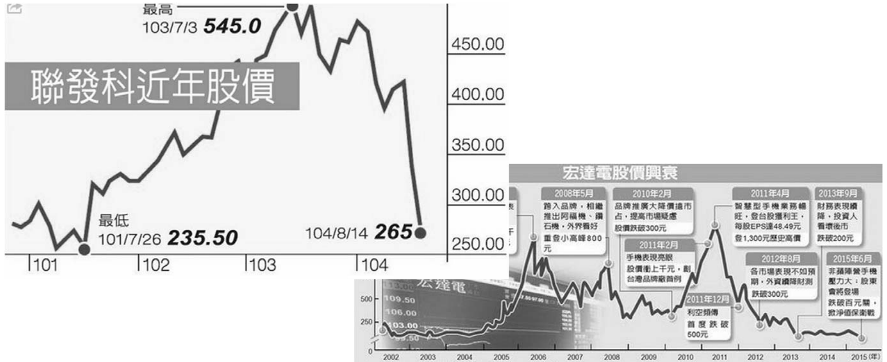
Lecture: Dr. Ching-Che Chung 鍾菁哲
Dept. of CSIE, National Chung Cheng University,
EA406, No. 168, University Rd., Min-Hsiung, Chia-Yi, Taiwan,
Email: wildwolf@cs.ccu.edu.tw; URL: http://www.s3lab.org

資工系學生培養硬體設計能力?

- 資工系學生，會寫程式應該的，但是：
  - ➢資工系不是學習軟體設計的科系嗎? 為什麼要培養與學習硬體設計能力？硬體設計讓電機系去學就好。
  - ➢學習硬體設計，倒底能對我帶來什麼幫助？立即的幫助 or 遠程的幫助 or 或者是沒有幫助
  - ➢那到底要從何培養硬體設計(數位IC設計)能力呢？

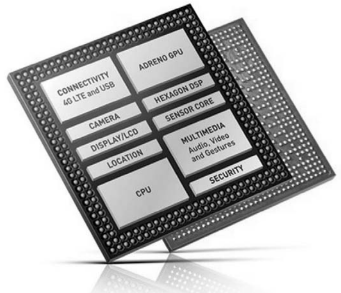
台灣硬體相關產業最近狀況不好

- **聯發科市值 60天縮水 2,000 億**
  - ➢高階 Helio X10 處理器晶片被小米新款手機打成低階手機晶片，及紅色供應鍊威脅。



世界硬體相關產業最近狀況也不好

- **高通(Qualcomm)營運慘淡**
  - ➢行動晶片大廠高通，將大砍 4,000 員工，並預計將公司一分為二，讓專利授權部門和晶片部門各自獨立。目前2/3年度獲利來自專利授權。



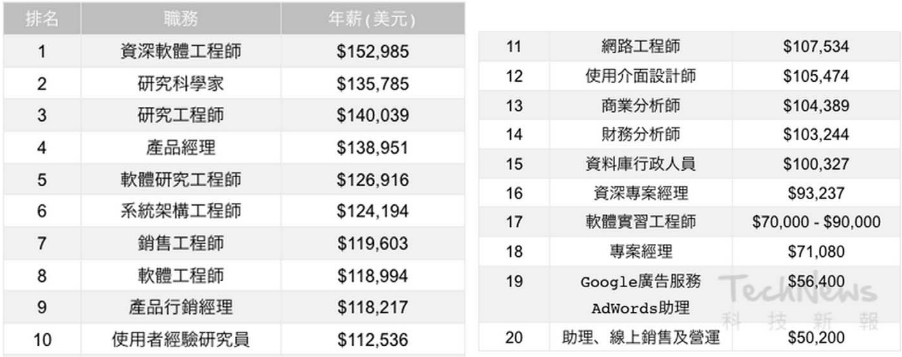
驍龍 810 的發熱問題

硬體大廠併購潮

- 高通24億美元收購 CSR
  - 因為這兩年手機市場萎縮，死守手機領域無法維持獲利，物聯網和車用電子為手機之後的藍海市場。

Google 各職位年薪大解密

- Google 以軟體起家，資深工程師年薪 456萬台幣



排名	職務	年薪(美元)
1	資深軟體工程師	$152,985
2	研究科學家	$135,785
3	研究工程師	$140,039
4	產品經理	$138,951
5	軟體研究工程師	$126,916
6	系統架構工程師	$124,194
7	銷售工程師	$119,603
8	軟體工程師	$118,994
9	產品行銷經理	$118,217
10	使用者經驗研究員	$112,536
11	網路工程師	$107,534
12	使用介面設計師	$105,474
13	商業分析師	$104,389
14	財務分析師	$103,244
15	資料庫行政人員	$100,327
16	資深專案經理	$93,237
17	軟體實習工程師	$70,000 - $90,000
18	專案經理	$71,080
19	Google廣告服務 AdWords助理	$56,400
20	助理、線上銷售及營運	$50,200

TechNews 科技新報

Google 台灣　　醒醒吧，你是賺台幣還是賺美金!!

http://goo.gl/1GVPjX　2014/08/22

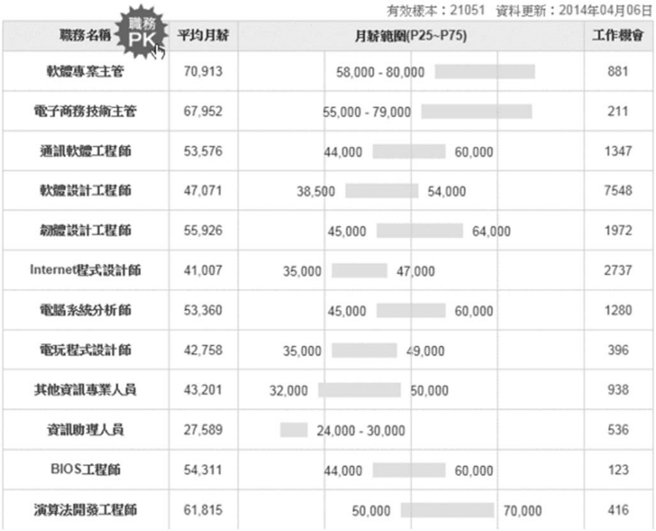
Facebook 各職位起薪令人羨慕

- Facebook 是軟體工程師嚮往的指標性公司
  - Software engineering intern（軟體工程師實習生）
    - 年收入：74,700美元（台幣 2,230,400元）
  - Site reliability engineer（SRE 網站可靠性工程師）
    - 年收入：80,413美元（台幣 2,401,500元）
  - Introductory software engineer（初級軟體工程師）
    - 年收入：100,000美元（台幣 2,987,000元）
  - Software engineer（軟體工程師）
    - 年收入：111,562美元（新台幣 3,332,600元）
  - Senior software engineer（高級軟體工程師）
    - 年收入：132,503美元（新台幣 3,957,100元）

醒醒吧，有多少台灣人擠的
進去這間!!

http://goo.gl/BQFkeo　2012/06/08

台灣軟體工程師的薪水-現實面



有效樣本：21051　資料更新：2014年04月06日

職務名稱 職務PK	平均月薪	月薪範圍(P25~P75)	工作機會
軟體專案主管	70,913	58,000 - 80,000	881
電子商務技術主管	67,952	55,000 - 79,000	211
通訊軟體工程師	53,576	44,000 60,000	1347
軟體設計工程師	47,071	38,500 54,000	7548
韌體設計工程師	55,926	45,000 64,000	1972
Internet程式設計師	41,007	35,000 47,000	2737
電腦系統分析師	53,360	45,000 60,000	1280
電玩程式設計師	42,758	35,000 49,000	396
其他資訊專業人員	43,201	32,000 50,000	938
資訊助理人員	27,589	24,000 - 30,000	536
BIOS工程師	54,311	44,000 60,000	123
演算法開發工程師	61,815	50,000 70,000	416

http://goo.gl/IfPrwu　2014/04/06

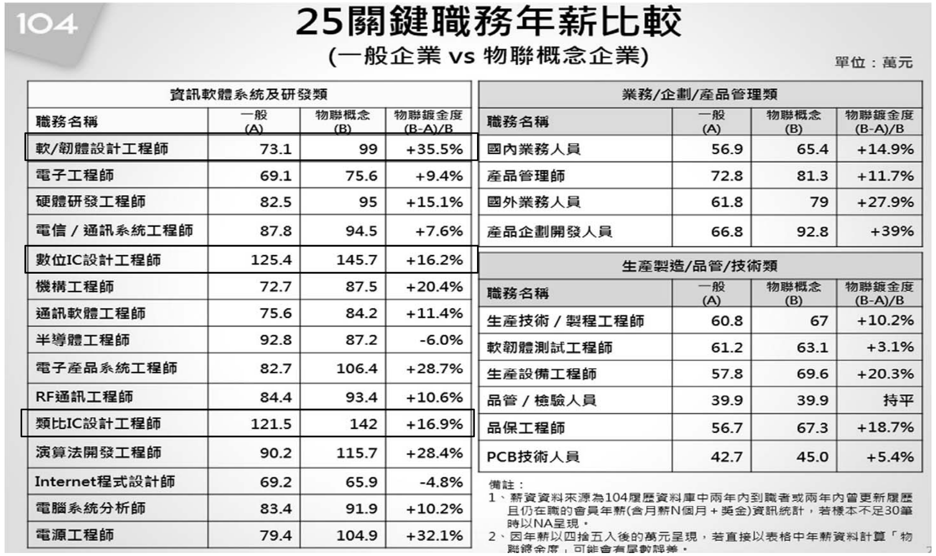
資工系畢業生前景之好，前所未有

- **物聯網、機器人、大數據運用與軟硬整合，資工背景人才需求孔急**
  - ➢聯發科給資工碩班畢業生起薪七萬，已經不是新聞。

Google台灣區總經理簡立峰表示「資工系畢業生前景之好，前所未有」。 報系資料照

http://goo.gl/mqM6pi

104人力銀行 物聯網關鍵 25大職務



104

25關鍵職務年薪比較

(一般企業 vs 物聯概念企業)

單位：萬元

資訊軟體系統及研發類			
職務名稱	一般 (A)	物聯概念 (B)	物聯鍍金度 (B-A)/B
軟/韌體設計工程師	73.1	99	+35.5%
電子工程師	69.1	75.6	+9.4%
硬體研發工程師	82.5	95	+15.1%
電信 / 通訊系統工程師	87.8	94.5	+7.6%
數位IC設計工程師	125.4	145.7	+16.2%
機構工程師	72.7	87.5	+20.4%
通訊軟體工程師	75.6	84.2	+11.4%
半導體工程師	92.8	87.2	-6.0%
電子產品系統工程師	82.7	106.4	+28.7%
RF通訊工程師	84.4	93.4	+10.6%
類比IC設計工程師	121.5	142	+16.9%
演算法開發工程師	90.2	115.7	+28.4%
Internet程式設計師	69.2	65.9	-4.8%
電腦系統分析師	83.4	91.9	+10.2%
電源工程師	79.4	104.9	+32.1%

業務/企劃/產品管理類			
職務名稱	一般 (A)	物聯概念 (B)	物聯鍍金度 (B-A)/B
國內業務人員	56.9	65.4	+14.9%
產品管理師	72.8	81.3	+11.7%
國外業務人員	61.8	79	+27.9%
產品企劃開發人員	66.8	92.8	+39%

生產製造/品管/技術類			
職務名稱	一般 (A)	物聯概念 (B)	物聯鍍金度 (B-A)/B
生產技術 / 製程工程師	60.8	67	+10.2%
軟韌體測試工程師	61.2	63.1	+3.1%
生產設備工程師	57.8	69.6	+20.3%
品管 / 檢驗人員	39.9	39.9	持平
品保工程師	56.7	67.3	+18.7%
PCB技術人員	42.7	45.0	+5.4%

備註：

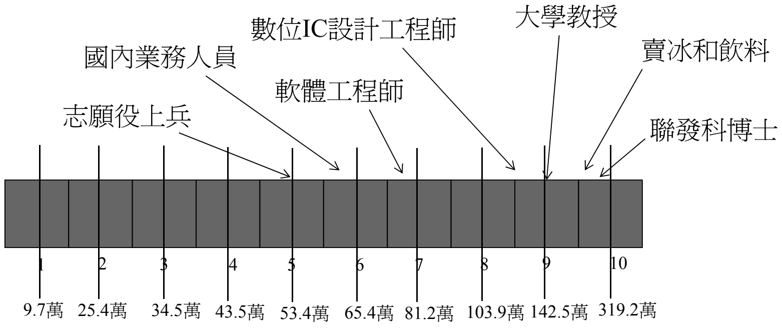
1、薪資資料來源為104履歷資料庫中兩年內到職者或兩年內曾更新履歷且仍在職的會員年薪(含月薪N個月＋獎金)資訊統計，若樣本不足30筆時以NA呈現。

2、因年薪以四捨五入後的萬元呈現，若直接以表格中年薪資料計算「物聯鍍金度，可能會有尾數誤差。

http://goo.gl/9iujbW 2014/11/04

最新綜合所得稅申報統計數據

- **102年度共 598.9萬戶申報家戶所得十等分數據**
  - ➢綜合所得稅未計入移轉性收入、攤販、地下經濟所得、炒房、炒股獲利。



http://www.fia.gov.tw/ct.asp?xItem=3119&ctNode=730&mp=4

Software/Hardware Engineer

$$F = \sum_{1}^{32} a_i \times X_i$$ **Timing, Area, Power**

where a_i is constant, X_i is input

Software Engineer:

1. We can use write a program with C/C++ and write a for-loop to implement this algorithm.
2. There is no way to speed up the processing time of the algorithm unless we have a faster computer.

Hardware Engineer:

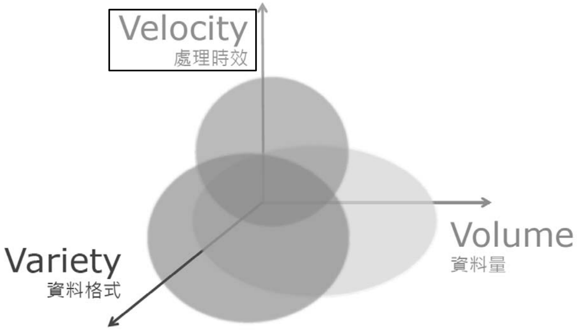
1. Can we simplify this algorithm ? Maybe we can lose some quality.
2. If we use 32 multipliers to implement this algorithm, this design can be computed in one cycle. However, what is the area cost and power cost for this architecture.
3. What if we use 4 multipliers or 8 multipliers, is it acceptable for computing time?

為何要硬體與軟體都學習

- 軟體於 CPU 之上執行
  - ➢當 CPU 來不及即時處理(Real Time)所需之運算時，需要增加 CPU 核心數或是客制化硬體加速。 C[n] = a[n] * b[n], n=1 to 1000
  - ➢使用 multi-thread 方式，增加運算使用的CPU核心數，會因為作業系統的負載，容易讓加速達到飽和，此時增加核心數無用。
  - ➢客制化硬體可以有效加速運算的進行，然而硬體運作的彈性比不上軟體，軟硬體需協同運作，以達系統最佳效能。

Big Data 時代，處理時效很重要

- **The growing popularity of web systems, mobile devices, surveillance videos, and wireless sensors generate large amounts of data from different sources.**



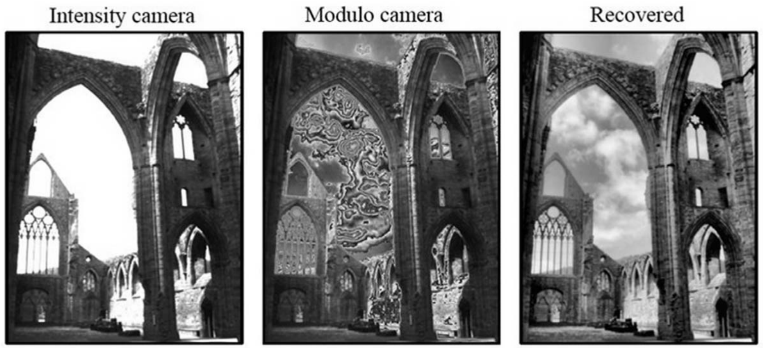
政府推展物聯網(IoT)與大數據分析

- 阿姆斯特丹免費提供感測裝置讓市民帶回家
  - ➢只要將感測裝置裝在陽台或窗戶，感測裝置就會自動偵測空氣與環境品質資料上傳雲端。另外提供城市 SDK，讓新的應用程式容易開發。

http://www.ithome.com.tw/news/97980

神奇的餘數相機(Modulo Camera)

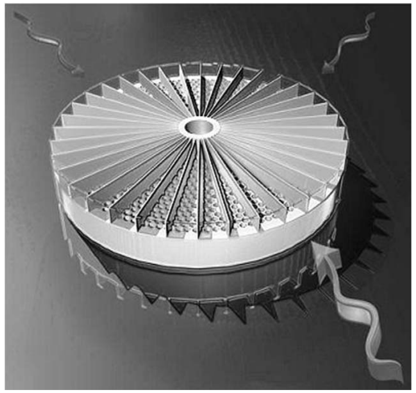
- 當感光元件超過其可吸收之電荷容量後，會自動重置歸零重新累積，例如只有10容量的像素，吸收31份量的光線，就會重置三次。
  - ➢從演算法到即時可用的相機，需要硬體加速。



http://web.media.mit.edu/~hangzhao/modulo.html

360度定位遠方聲音

- 美國 Duke University 改進麥克風結構
  - ➢不同方向收入的訊號進行不同編碼調變，讓嵌入式電路區別聲音傳來的方向後，可從一群吵雜的人聲中，辨識出某人的聲音。
  - ➢從演算法到即時反應，需要硬體加速。



http://goo.gl/z4QoaM

齧齒動物大腦建構研究

- IBM 使用 48 塊 TrueNorth 試驗晶片建構動物大腦：
  - ➢每個 TrueNorth 包含 4096 hardware cores，這個系統可以類比 4800 萬個神經細胞。這個系統加上 Deep Learning Algorithm，可以提供面部辨識、語言即時翻譯與圖片分類。

http://technews.tw/2015/08/20/ibms-rodent-brain-chip-could-make-our-phones-hyper-smart/

創意製造商機：革命性的嬰兒車

- 大眾汽車(Volkswagen)的嬰兒車
  - ➢更貼心的考慮到使用嬰兒車意外預防

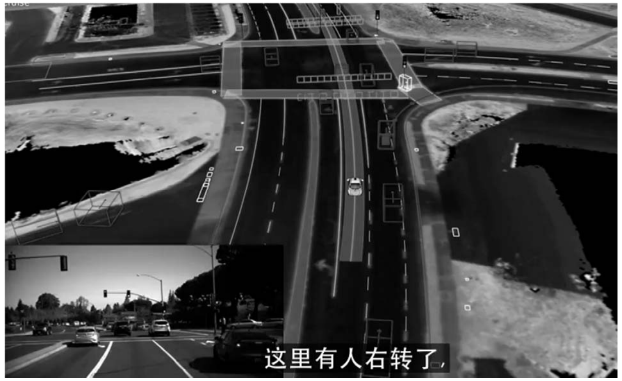
創意製造商機：取代郵差與送貨員

- 使用到自動駕駛技術和人臉辨識技術的機器人
  - ➢可以幫忙做小區域與街道送貨

https://www.facebook.com/NowThisNews/videos/892444064179052/?pnref=story

自動車是如何看到路況的

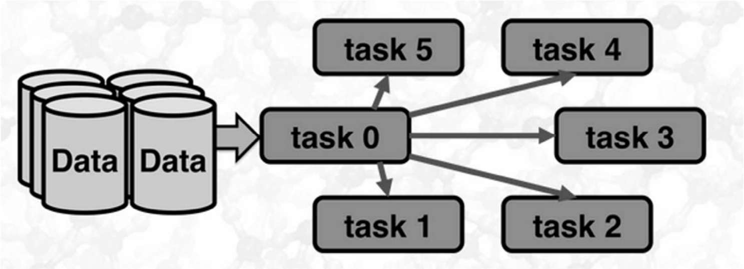
- **Google Driverless Car Program**：
  - ➢大量的機械學習與感測器使用，運算量大。



http://www.ted.com/talks/chris_urmson_how_a_driverless_car_sees_the_road

Traditional Parallel Applications

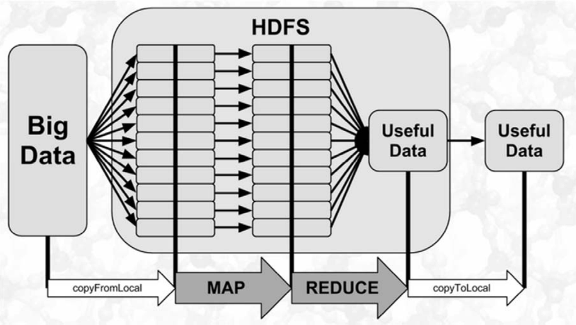
- **The input data are stored on remote storage device serving files over NFS.**
  - ➢ The master worker divides up the input data and sends to each of the other works.
  - ➢ Disk I/O in the master worker has fundamental limitation.



http://www.glennklockwood.com/data-intensive/hadoop/overview.html

Hadoop : Software Framework for Big Data

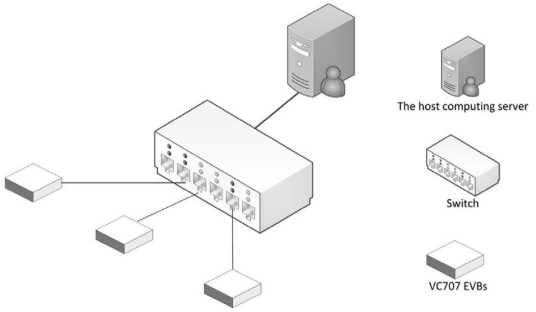
- **HDFS provides data distribution, replication, and automatic recovery.**
  - ➢ The MapReduce framework enables the automatic paralleling and distribution of large-scale computation applications on the large cluster of computing servers.

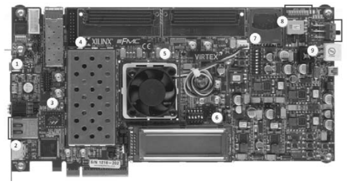


http://www.glennklockwood.com/data-intensive/hadoop/overview.html

Hardware Accelerator - 1

- **The proposed accelerator platform is composed of many VC707 FPGA evaluation boards (EVBs).**
  - ➢ The computing server communicates with FPGA EVBs with Gigabit Ethernet switch. Then, the workloads of the computing server can be shared in FPGA EVBs.

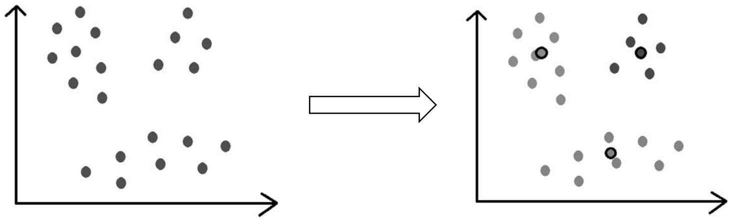




VC707

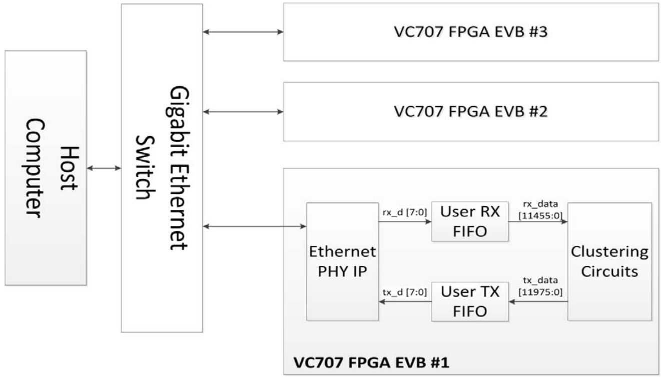
Hardware Accelerator - 2

- **We use K-means algorithm to demonstrate the speedup of the hardware accelerator.**
  - The goal of K-means algorithm is to separate the input data into the number *k* of clusters.
  - There are many floating point operations in K-means algorithm. (Calculating Euclidean distance)



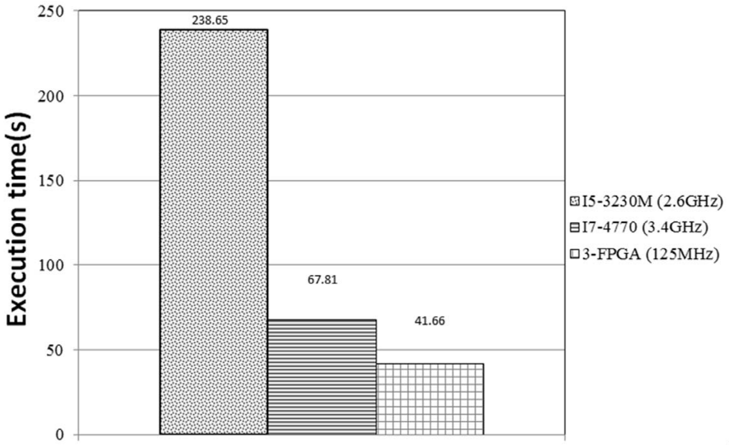
Hardware Accelerator - 3

- **Circuits in FPGA**
  - The single VC707 EVB can handle 115 data nodes from the host computer transmission packet simultaneously. Then it returns cluster number of each node to the host computer by one packet.



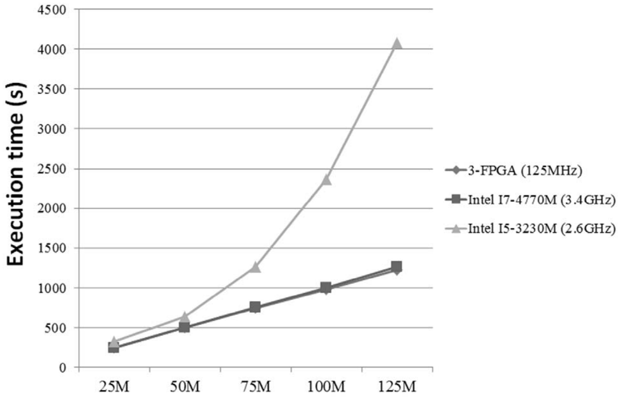
Hardware Accelerator - 4

- **Execution time without disk I/O latency**
  - The input dataset is 125 million three-dimensional nodes.



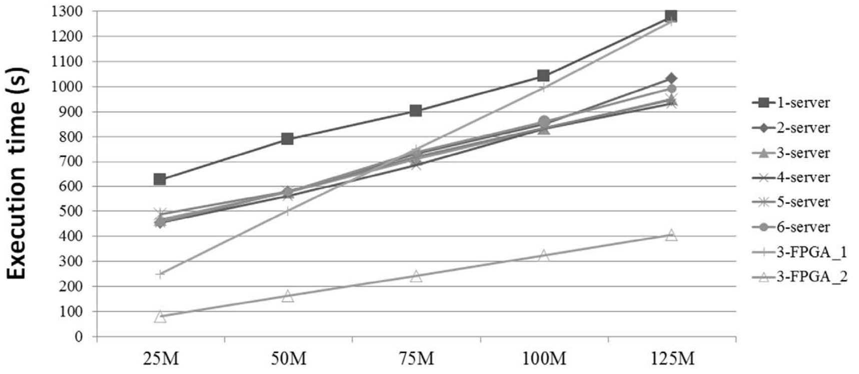
Hardware Accelerator - 5

- **Execution time with disk I/O latency**
  - Since the disk I/O latency dominates the execution time, the speedup of the hardware accelerator is not important.



Hardware Accelerator - 6

- **Execution time with disk I/O latency**
  - We compare the hardware accelerator with **Hadoop server** with n-servers. Here 3-FPGA_2 shows the case when data are already separated in three computers.



MIT 打造的省錢節能大數據運算架構

- **MIT 研究員使用 NAND Flash 取代 DRAM**
  - 即使 NAND Flash 速度只有 DRAM 的 1/10，如果分佈式運算整體運算時間中的 5%需要寫入硬碟，此架構可提升伺服器效能。
  - 每台伺服器連接至一個 FPGA，每個 FPGA 連接 500GB NAND Flash，FPGA 可程式化，並且個別 FPGA 間可以彼此互通。

http://goo.gl/lniJCR

競爭的社會

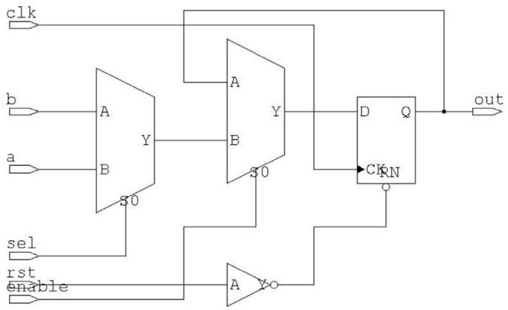
- 出社會後，誰會停下來等你呢?
  - 2015/08/25 北京田徑世錦賽女子一萬公尺決賽

跑了9999米後　她提早1米慶祝就丟了獎牌

簡易的數位 IC 設計流程介紹 -1

- **Digital IC Design is also a programming process**
  - Write C-Like **Verilog** program to design your digital IC.
  - Use **logic synthesizer** to implement circuit
  - Use **automatic placement and routing tool** to layout circuit

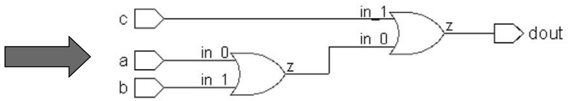
```
always@(posedge clk or posedge rst)
begin
    if(rst)
        out <= 1'b0;
    else if (enable) begin
        if (sel)
            out <= a;
        else
            out <= b;
    end
end
```



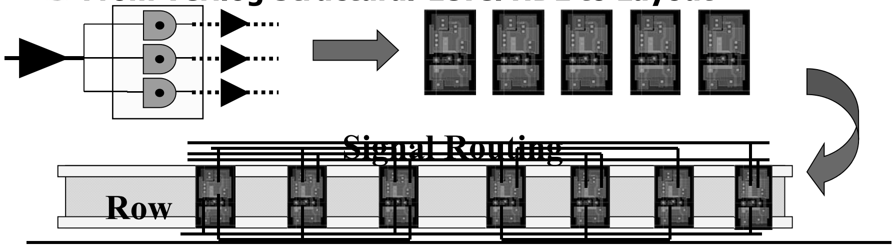
簡易的數位 IC 設計流程介紹 -2

- **From Verilog Behavior-Level HDL to Logic Gates (netlist)**

```
module comb_or(dout,a,b,c);
input a,b,c;
output dout;

assign dout = a | b | c;

endmodule
```

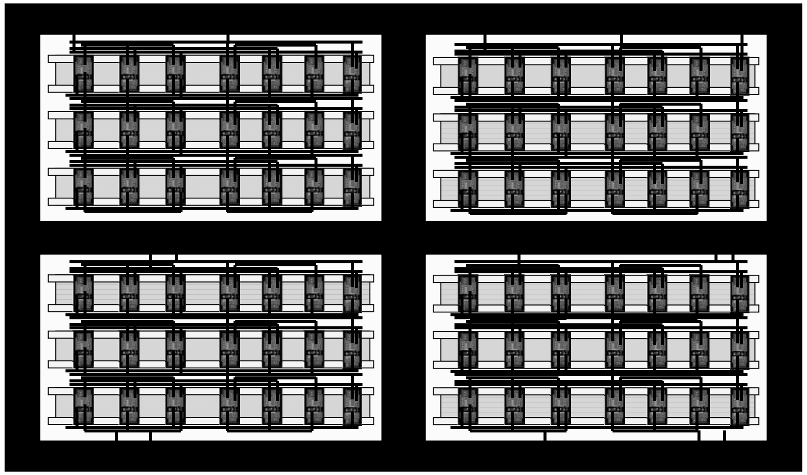


- **From Verilog Structural-Level HDL to Layout**



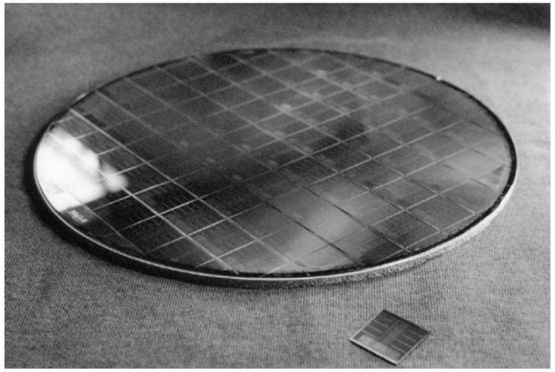
簡易的數位 IC 設計流程介紹 -3

- **Chip Floorplanning and Detail Routing**



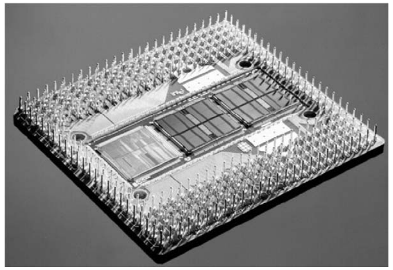
簡易的數位 IC 設計流程介紹 -4

- **Each die is a piece of a full wafer.**

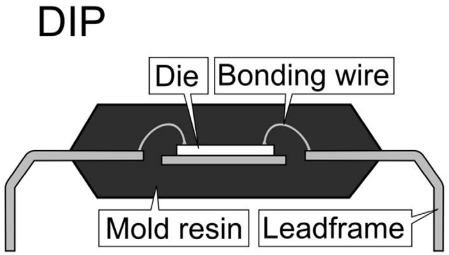


簡易的數位 IC 設計流程介紹 -5

- **Package Functions**
  - ➢ Electric connection of signals and power from chip to board.
  - ➢ Mechanical connection of chip to board
  - ➢ Remove heat produced on chip
  - ➢ Protects chip from mechanical damage

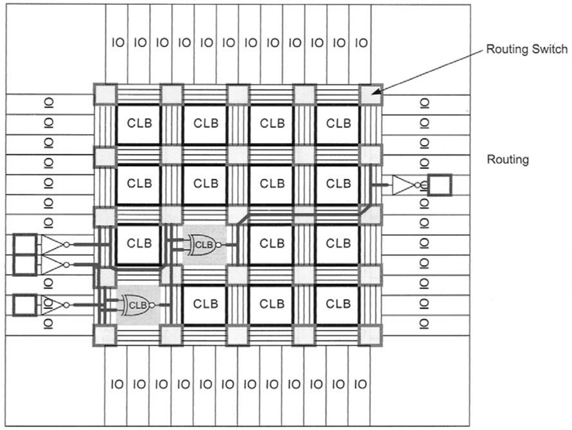


Pentium Pro



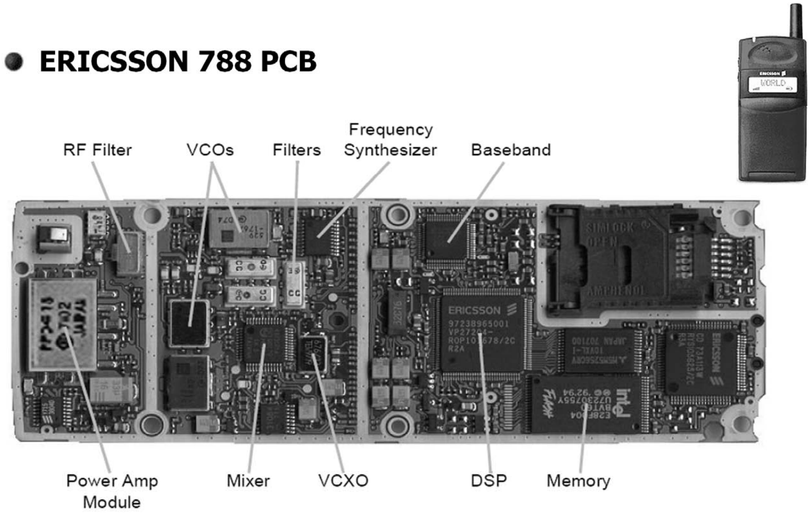
簡易的數位 IC 設計流程介紹 -6

- **Field Programmable Gate Array (FPGA)**
  - ➢ Each logic block can be programmable to different functions.



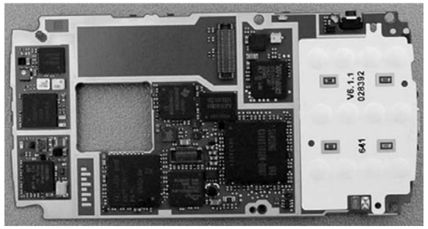
簡易的數位 IC 設計流程介紹 - 7

- **ERICSSON 788 PCB**

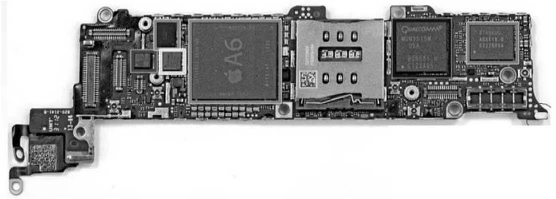


簡易的數位 IC 設計流程介紹 -8

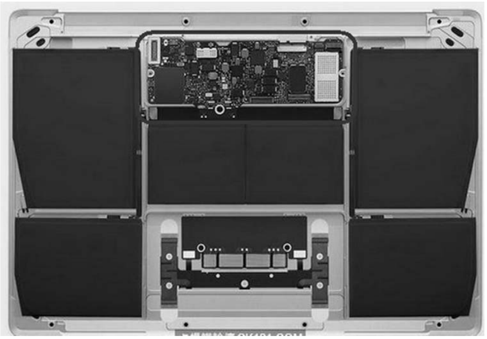
- **Nokia N95**



- **iPhone 5**



簡易的數位 IC 設計流程介紹 -9



MacBook

- **Portable Systems & Hand Held Equipment**
  - ➢ lower power consumption (Battery operated)
- **Real-time System & High-Speed Computer**
  - ➢ noise control & heat removal (High-Speed Bus, Package Technology)
- **Semiconductor Vendors**
  - ➢ limitations of sub-micro process (Reliability issues)

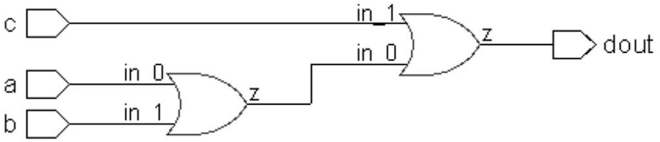
硬體描述語言

- 使用硬體描述語言 (Verilog/VHDL) ，來設計不同的電路架構。
  - ➢ Verilog 可用於規劃同時使用多少套運算電路來進行運算。
  - ➢ Verilog 可以規劃每個 Clock 週期，各個電路模組要處裡哪些動作，因此可以安排電路 Pipeline 運作流程。
  - ➢ Verilog 使用高階敘述(if, else, +-*/)，可以使用類似C語言的語言結構，描述所設計的電路的功能。
  - ➢ Verilog 只負責描述你所希望的電路架構，但是不保證你所規劃出來的電路架構，符合所希望的運算速度。
  - ➢ 常見的 Verilog Simulator: NC-Verilog, VCS

硬體描述語言用於組合電路

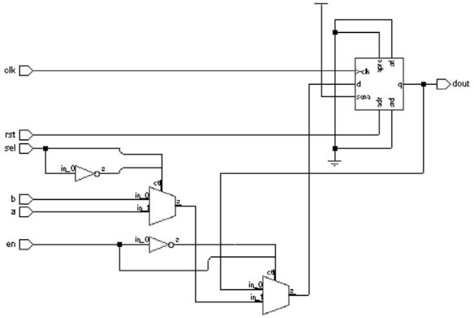
- 學習硬體描述語言，並了解你所撰寫的程序會實現怎樣的電路。
  - ➢ Verilog Example: Combinational Circuit

```
module comb_or(dout,a,b,c);
input a,b,c;
output dout;

assign dout = a | b | c;

endmodule
```



硬體描述語言用於序列電路 - 1

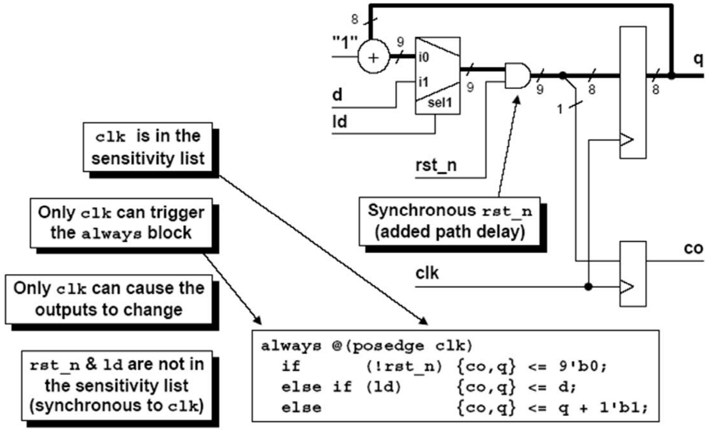
- Sequential Circuit with Asynchronous RESET



```
module ff_ar(dout,clk,rst,en,sel,a,b);

input clk,rst,en,sel,a,b;
output dout;

reg dout;

always @(posedge clk or posedge rst)
begin
   if (rst)
      dout <= 1'b0;
   else if (en) begin
      if (sel)
        dout <= a;
      else
        dout <= b;
      end
   end
end
```

硬體描述語言用於序列電路 - 2

- Sequential Circuit with Synchronous RESET



clk is in the sensitivity list

Only clk can trigger the always block

Only clk can cause the outputs to change

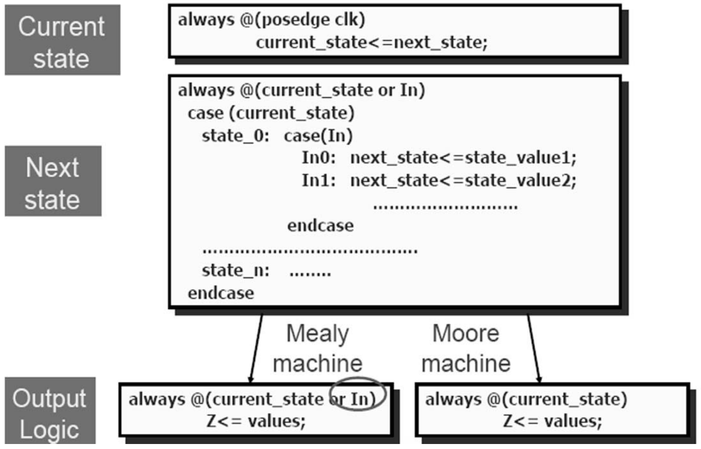
rst_n & ld are not in the sensitivity list (synchronous to clk)

Synchronous rst_n (added path delay)

```
always @(posedge clk)
  if       (!rst_n) {co,q} <= 9'b0;
  else if  (ld)     {co,q} <= d;
  else              {co,q} <= q + 1'b1;
```

硬體描述語言用於序列電路 - 3

- Modeling Finite State Machine



```
always @(posedge clk)
          current_state<=next_state;
```

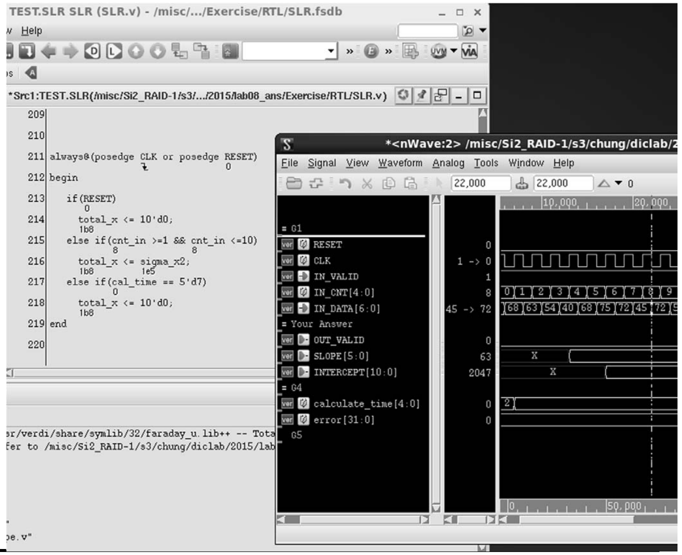
```
always @(current_state or In)
  case (current_state)
    state_0:  case(In)
                In0:  next_state<=state_value1;
                In1:  next_state<=state_value2;
                      ..........................
              endcase
    .......................................
    state_n:   ........
  endcase
```

```
always @(current_state or In)
      Z<= values;
```

```
always @(current_state)
      Z<= values;
```

Debug 硬體描述語言的工具

- Verdi


TEST.SLR SLR (SLR.v) - /misc/.../Exercise/RTL/SLR.fsdb
Help
*Src1:TEST.SLR(/misc/Si2_RAID-1/s3/.../2015/lab08_ans/Exercise/RTL/SLR.v)
*<nWave:2> /misc/Si2_RAID-1/s3/chung/diclab/2
File Signal View Waveform Analog Tools Window Help
22,000
22,000
10,000
20,000
0
50,000

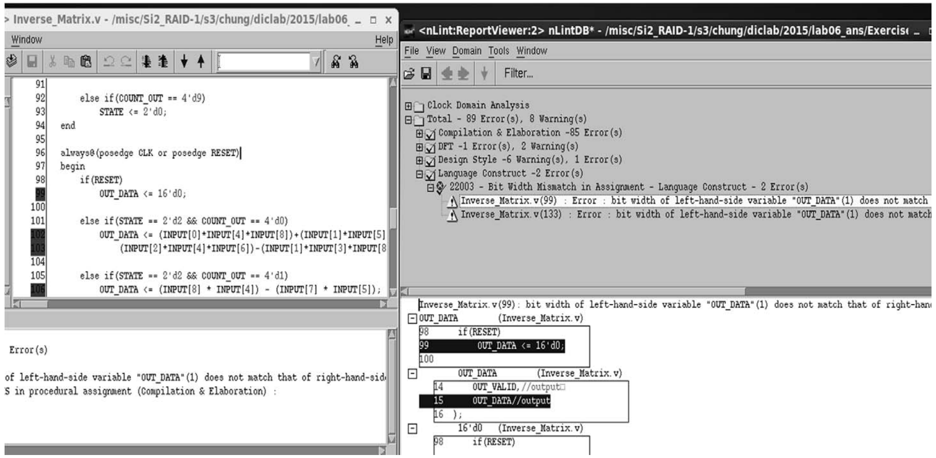
```
always@(posedge CLK or posedge RESET)
                                  0
begin
  if(RESET)
    0
    total_x <= 10'd0;
    1b8
  else if(cnt_in >=1 && cnt_in <=10)
          8              8
    total_x <= sigma_x2;
    1b8        1e5
  else if(cal_time == 5'd7)
          0
    total_x <= 10'd0;
    1b8
end
```

```
r/verdi/share/symlib/32/faraday_u.lib++ -- Tota
fer to /misc/Si2_RAID-1/s3/chung/diclab/2015/lab
e.v"
```

```
= G1
RESET                 0
CLK              1 -> 0
IN_VALID              1
IN_CNT[4:0]           8
IN_DATA[6:0]    45 -> 72
= Your Answer
OUT_VALID             0
SLOPE[5:0]           63
INTERCEPT[10:0]    2047
= G4
calculate_time[4:0]   0
error[31:0]           0
G5
```

檢查 Verilog 程式碼的 Quality

- nLint: Check the quality of your Verilog code


> Inverse_Matrix.v - /misc/Si2_RAID-1/s3/chung/diclab/2015/lab06
Window
Help
<nLint:ReportViewer:2> nLintDB* - /misc/Si2_RAID-1/s3/chung/diclab/2015/lab06_ans/Exercis
File View Domain Tools Window
Filter...

```
 91
 92        else if(COUNT_OUT == 4'd9)
 93            STATE <= 2'd0;
 94    end
 95
 96    always@(posedge CLK or posedge RESET)
 97    begin
 98        if(RESET)
 99            OUT_DATA <= 16'd0;
100
101        else if(STATE == 2'd2 && COUNT_OUT == 4'd0)
102            OUT_DATA <= (INPUT[0]*INPUT[4]*INPUT[8])+(INPUT[1]*INPUT[5]
103                (INPUT[2]*INPUT[4]*INPUT[6])-(INPUT[1]*INPUT[3]*INPUT[8
104
105        else if(STATE == 2'd2 && COUNT_OUT == 4'd1)
106            OUT_DATA <= (INPUT[8] * INPUT[4]) - (INPUT[7] * INPUT[5]);
```

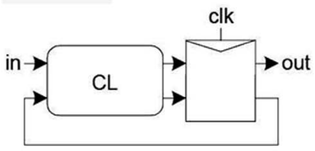
```
Error(s)

of left-hand-side variable "OUT_DATA"(1) does not match that of right-hand-sid
S in procedural assignment (Compilation & Elaboration) :
```

```
⊞ Clock Domain Analysis
⊟ Total - 89 Error(s), 8 Warning(s)
  ⊞ ☑ Compilation & Elaboration -85 Error(s)
  ⊞ ☑ DFT -1 Error(s), 2 Warning(s)
  ⊞ ☑ Design Style -6 Warning(s), 1 Error(s)
  ⊟ ☑ Language Construct -2 Error(s)
    ⊟ 22003 - Bit Width Mismatch in Assignment - Language Construct - 2 Error(s)
        Inverse_Matrix.v(99) : Error : bit width of left-hand-side variable "OUT_DATA"(1) does not match
        Inverse_Matrix.v(133) : Error : bit width of left-hand-side variable "OUT_DATA"(1) does not match
```

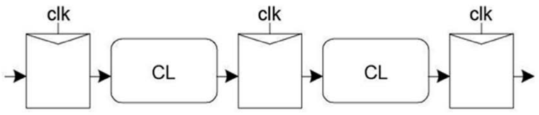
```
Inverse_Matrix.v(99): bit width of left-hand-side variable "OUT_DATA"(1) does not match that of right-hand
⊟ OUT_DATA        (Inverse_Matrix.v)
  98       if(RESET)
  99           OUT_DATA <= 16'd0;
  100
⊟          OUT_DATA        (Inverse_Matrix.v)
    14       OUT_VALID, //output
    15       OUT_DATA//output
    16   );
⊟          16'd0    (Inverse_Matrix.v)
    98       if(RESET)
```

Sequential Machine

- **A sequential machine is a machine**
  - for which the output values depend **not only on the present input values** but also the history of previous input.
  - Memory elements are added to combinational logic to build sequential machines.



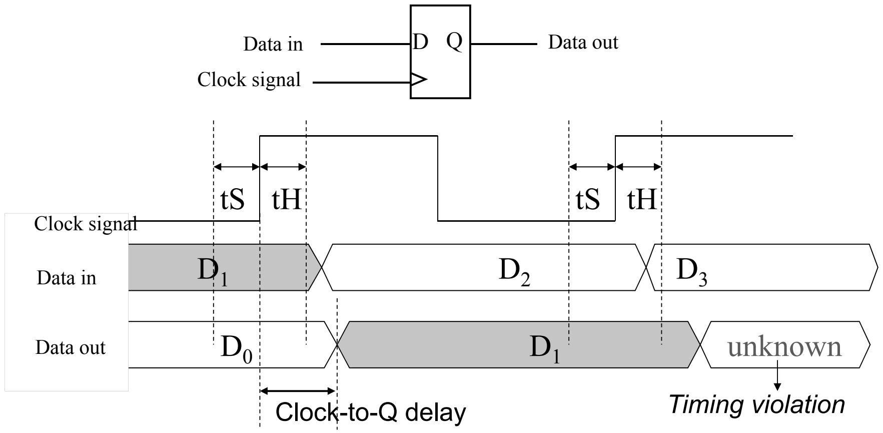
Finite State Machine



Pipeline

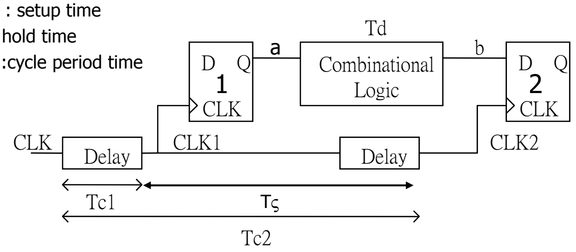
Sequential Element: D-Flip/Flop

- **D-Flip/Flop Timing Violations**
  - Setup time and hold time



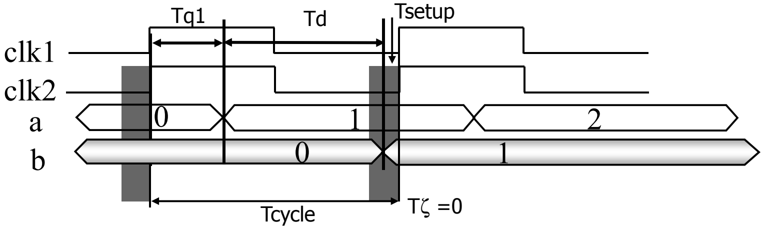
Setup/Hold Time Criterion

- **Hold Time Margin**
  - **(Tq1 + Td) > T_{hold} +(Tζ)**
- **Setup Time Margin**
  - **(T_{cycle} + Tζ) > T_{setup}+(Tq1 + Td)**
    - Td :combinational logic delay time
    - Tc : clock delay time
    - Tς: clock skew between Tc1 and Tc2
    - Tq: clock to Q delay time
    - Tsetup : setup time
    - Thold :hold time
    - Tcycle :cycle period time

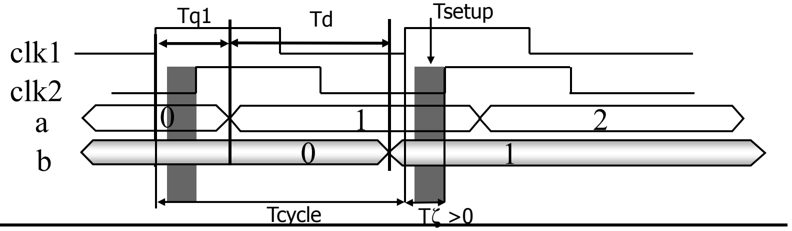


Setup/Hold Time Criterion

- **Setup Time Margin:** Tcycle > Tsetup+(Tq1 + Td) , without clock skew

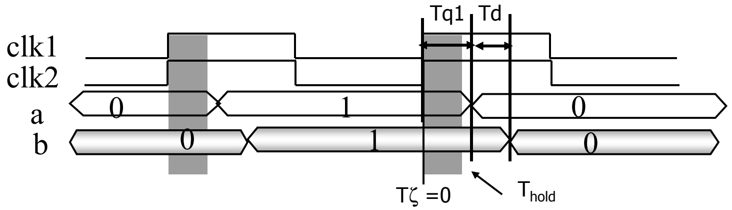


- **Setup Time Margin:** (Tcycle + Tζ) > Tsetup+(Tq1+Td) , with clock skew

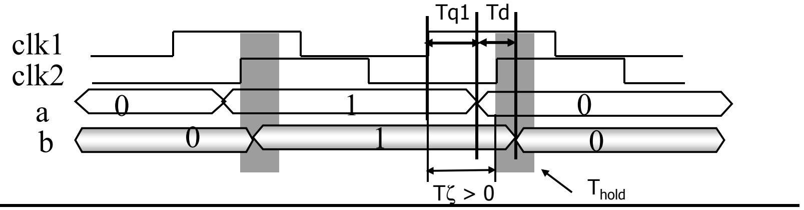


Setup/Hold Time Criterion

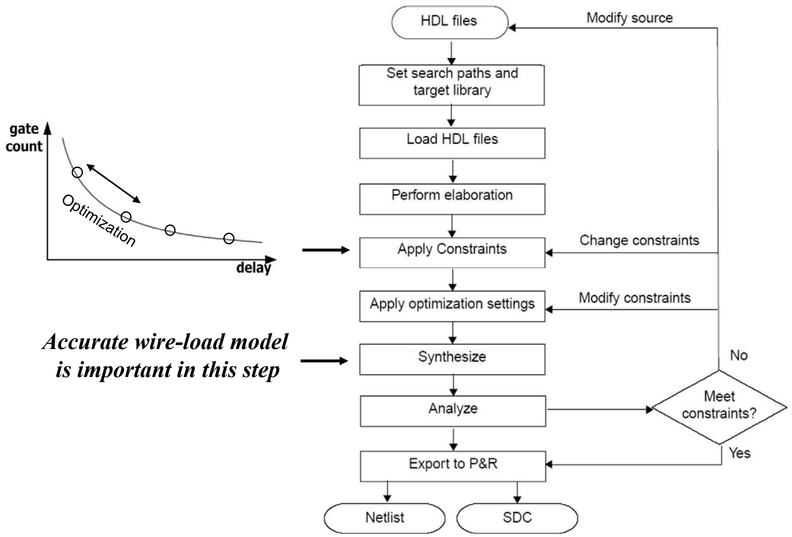
- **Hold Time Margin:** (Tq1 + Td)> T_{hold} , without clock skew



- **Hold Time Margin:** (Tq1 + Td)> T_{hold} +(T ζ), with clock skew



使用電路合成軟體產生 Gate-Level Circuit

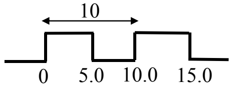


Accurate wire-load model is important in this step

Clock Constraints

- **Define the clocks**
  - For synchronous designs, the clock period constrains all register-to-register paths in the design.
  - Defines the period and waveform for the clock
    - Syntax: ***create_clock*** -name clock_name source_objects –period period_value -waveform edge_list
    - source_objects : a list of pins or ports on which to apply this clock.
    - -period period_value : the period of the clock waveform
    - -waveform edge_list : Specifies the rise and fall edge times of the clock waveforms
    - Ex: The clock applied on port CLK with a period of 10.0, rise at 0.0 and fall at 5.0.

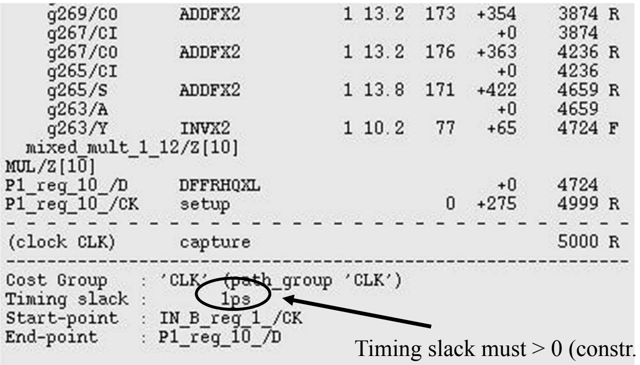
      create_clock -name CLK1 CLK -period 10 -waveform { 0.0 5.0 }



Duty-cycle of CLK is
(5.0 - 0.0)/(10.0 - 0.0)=50%

Report Analysis: Timing

- **Report Design Timing:**
  - Generates a timing report that provides information about the various paths in the design.
  - Syntax: ***report timing*** > report_filename

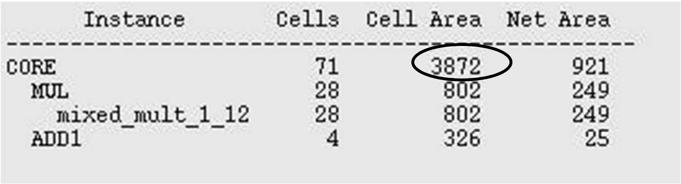


```
    g269/CO       ADDFX2           1 13.2  173  +354    3874 R
    g267/CI                                      +0     3874
    g267/CO       ADDFX2           1 13.2  176  +363    4236 R
    g265/CI                                      +0     4236
    g265/S        ADDFX2           1 13.8  171  +422    4659 R
    g263/A                                       +0     4659
    g263/Y        INVX2            1 10.2   77   +65    4724 F
  mixed_mult_1_12/Z[10]
MUL/Z[10]
P1_reg_10_/D    DFFRHQXL                         +0     4724
P1_reg_10_/CK   setup                       0  +275     4999 R
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(clock CLK)     capture                                 5000 R
-------------------------------------------------------------
Cost Group    : 'CLK' (path_group 'CLK')
Timing slack  :       1ps
Start-point   : IN_B_reg_1_/CK
End-point     : P1_reg_10_/D
```

Timing slack must > 0 (constr. met)

Report Analysis: Area

- **Report Design Area:**
  - Generates an area report that provides information about area of each modules in the design.
  - Syntax: ***report area*** > report_filename

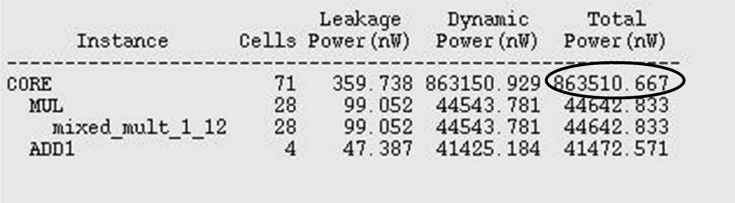


```
      Instance         Cells  Cell Area  Net Area
-------------------------------------------------
CORE                      71       3872       921
  MUL                     28        802       249
    mixed_mult_1_12       28        802       249
  ADD1                     4        326        25
```

  - In UMC 0.18um cell-library, area (um^2)/9.97 is gate count.
    - 9.97 is the area of min. 2-input NAND gate in the cell-library.
    - Ex: 3872/9.97 = 388 (gate count)
  - The same RTL design synthesized in different process always get almost the same gate count.

Report Analysis: Power

- **Report Design Power:**
  - Generates an power report that provides information about power consumption of each modules in the design.
  - Syntax: ***report power*** > report_filename

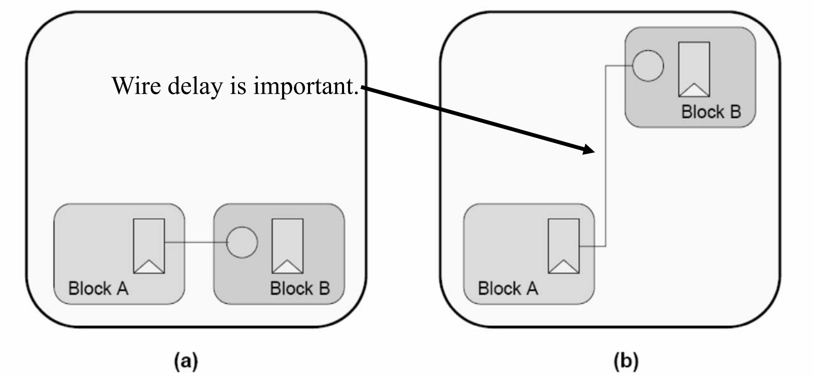


```
                          Leakage    Dynamic      Total
     Instance       Cells Power(nW)  Power(nW)    Power(nW)
--------------------------------------------------------------
CORE                  71   359.738 863150.929 863510.667
  MUL                 28    99.052  44543.781  44642.833
    mixed_mult_1_12   28    99.052  44543.781  44642.833
  ADD1                 4    47.387  41425.184  41472.571
```

  - Total Power = Dynamic Power + Leakage Power

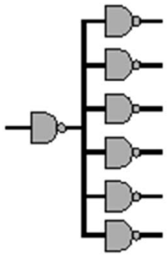
Wiring Problem - 1

- **Floorplan affects timing budget and wire-load model.**
  - Wire delay can only be estimated in synthesis level.
  - Wire delay can be calculated after APR and RC extraction.



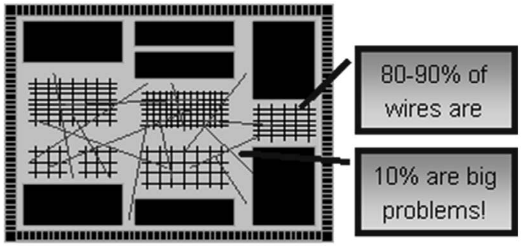
Wiring Problem - 2

- **Difference in Synthesis View and Physical View**



Synthesis View

All wires of fan-out=n are the same.

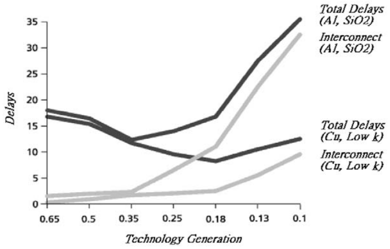


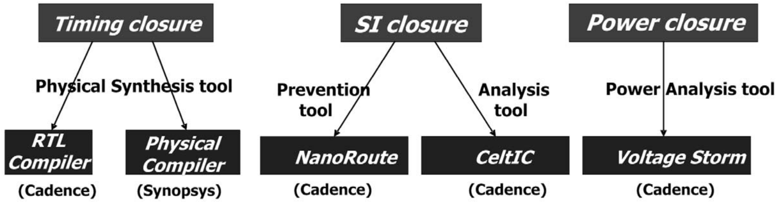
Physical View

Each wire is unique.

Wiring Problem - 3

- **Wiring delay dominates overall delay**
- **New problems due to large wire resistance**
  - Timing closure
  - Signal Integrity closure (crosstalk, …)
  - Power closure (IR drop, …)



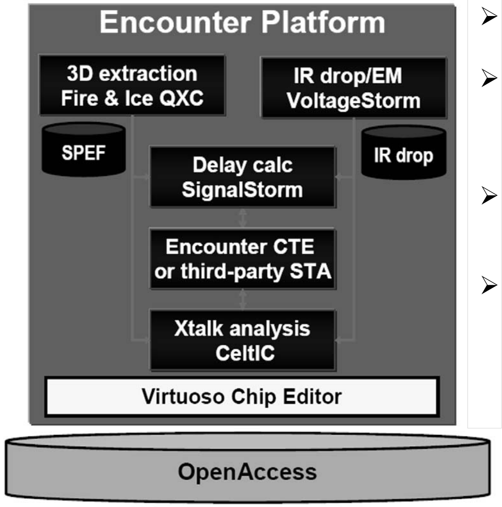


Timing closure
Physical Synthesis tool
RTL Compiler (Cadence)
Physical Compiler (Synopsys)

SI closure
Prevention tool
NanoRoute (Cadence)
Analysis tool
CeltIC (Cadence)

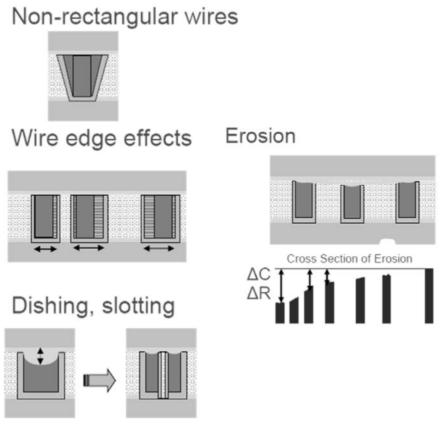
Power closure
Power Analysis tool
Voltage Storm (Cadence)

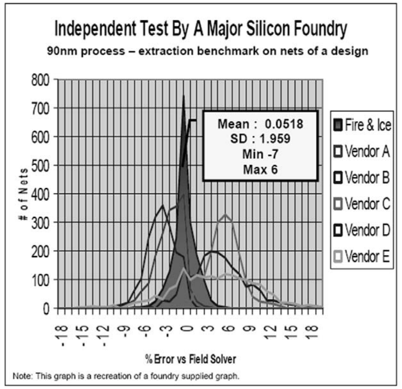
Nanometer Timing/SI Sign-Off



- Prototyping and implementation coupled with sign-off capabilities
- Integrated silicon-proven flow covering 3D extraction, IR drop,timing, and crosstalk analysis
- Analysis of interaction of IR drop and xtalk on both noise immunity and performance
- Chip finishing and wire editing using OpenAccess

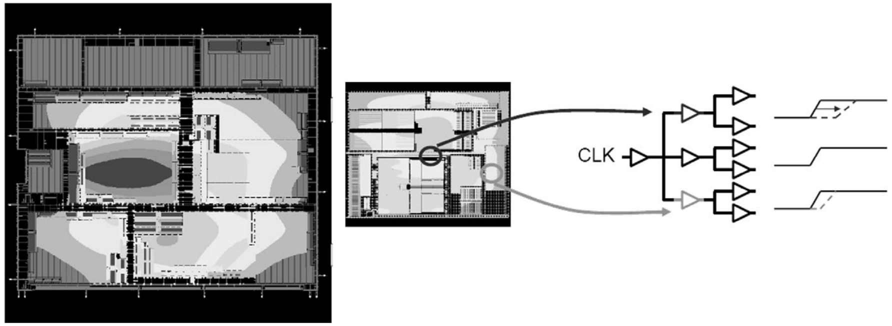
Gate-Level RC Extractor





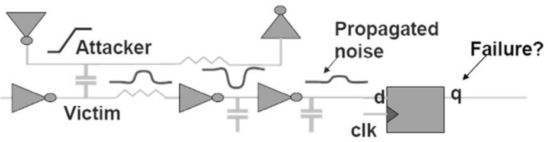
- Fire&Ice for for 3D parasitic extraction

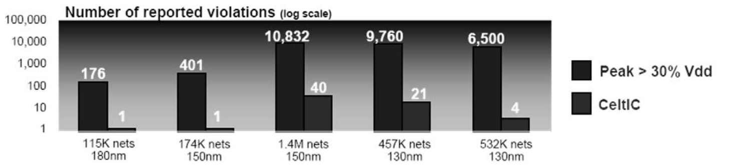
Power Grid Analysis



- IR drop and Ground bounce combine to impact silicon performance (setup/hold time violations)
- Electromigration (EM) caused by current flow and current density may make chip failure

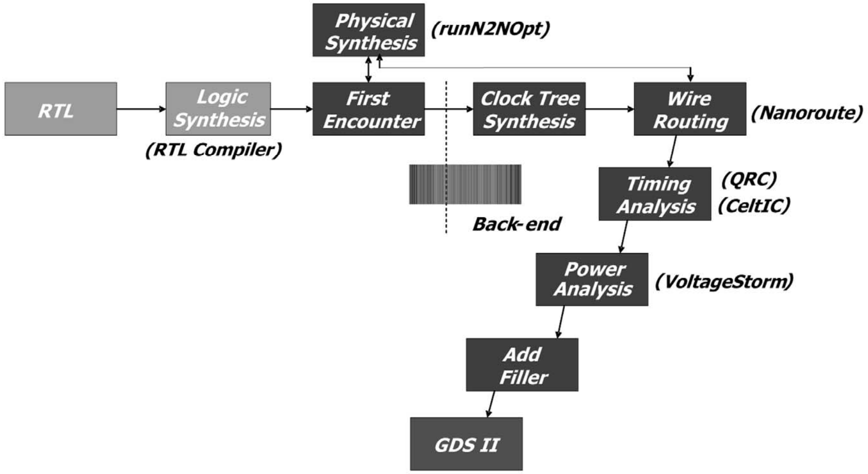
Glitch Analysis





- Crosstalk glitches propagate to latches to create functional failures.
- Glitch Analysis is needed to detect crosstalk induced functional and timing failures
- Spice-like accuracy using mixed cell and transistor modeling

SoC Encounter Design Flow



Physical Synthesis (runN2NOpt)

RTL

Logic Synthesis (RTL Compiler)

First Encounter

Clock Tree Synthesis

Wire Routing (Nanoroute)

Back-end

Timing Analysis (QRC) (CeltIC)

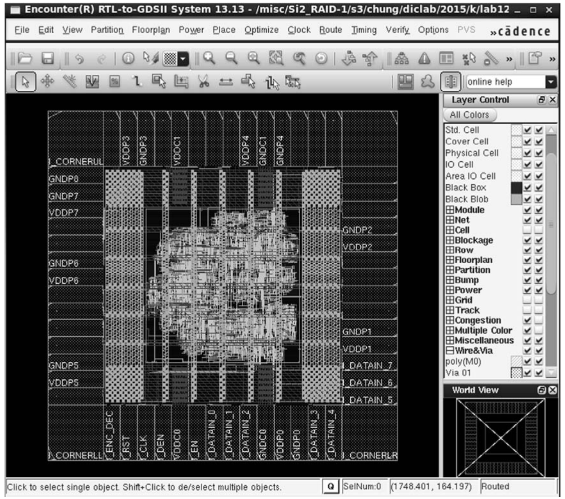
Power Analysis (VoltageStorm)

Add Filler

GDS II

NanoRoute – Detail Routing (cont'd)

- **Result of NanoRoute**

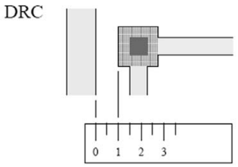


Encounter(R) RTL-to-GDSII System 13.13 - /misc/Si2_RAID-1/s3/chung/diclab/2015/k/lab12

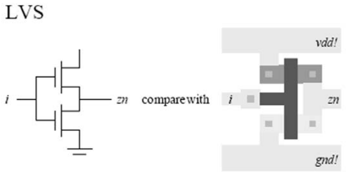
Std. Cell ☑ ☑
Cover Cell ☑ ☑
Physical Cell ☑ ☑
IO Cell ☑ ☑
Area IO Cell ☑ ☑
Black Box ☑ ☑
Black Blob ☑ ☑
⊞Module ☑ ☑
⊞Net ☑ ☑
⊞Cell □ □
⊞Blockage ☑ ☑
⊞Row ☑ ☑
⊞Floorplan ☑ ☑
⊞Partition ☑ ☑
⊞Bump ☑ ☑
⊞Power ☑ ☑
⊞Grid □ □
⊞Track □ □
⊞Congestion ☑ □
⊞Multiple Color ☑ □
⊞Miscellaneous ☑ ☑
⊟Wire&Via ☑ ☑
poly(M0) ☑ ☑
Via 01 ☑ ☑

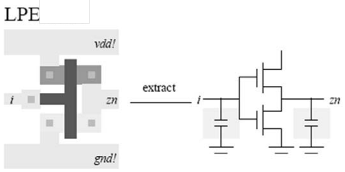
Click to select single object. Shift+Click to de/select multiple objects. | SelNum:0 | (1748.401, 164.197) | Routed

Layout Verifications



- **DRC : Design Rule Checks**
  - ➢ The operation checks for design rule violations.
- **LVS : Layout Versus Schematic**
  - ➢ The operation checks for inconsistencies between the schematic and the physical layout.
- **LPE/PEX : Layout Parameter Extraction**
  - ➢ Extract device parameters from layout, including transistors, parasitic capacitors, and resistors.
  - ➢ Extracted netlist can be used in post-layout transistor-level simulation (by UltraSim, HSIM, or NanoSim)





Post-layout Gate-Level Simulation

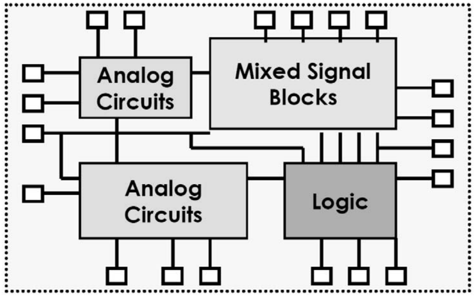
- **Post layout timing simulation**
  - ➢ The post-layout design must be simulated with extracted RC. A SDF files generated from QRC and Encounter Delay Calculator are used for this propose.
- **Modify your verilog test file**
  - ➢ $sdf_annotate("the_SDF_file_name", the_instance_name);
  - ➢ E.g. : $sdf_annotate("CHIP.sdf", I_DAG);
- **Run post-layout gate-level simulation**
  - ➢ Using NC-Verilog to run post-layout gate-level simulation
  - ➢ UNIX%ncverilog -f run.f
    - ■ Simulate the worse case delay to check there is no setup time violation.
    - ■ You can also write out best case delay SDF file for hold time verification.

Post-Layout Transistor-Level Simulation

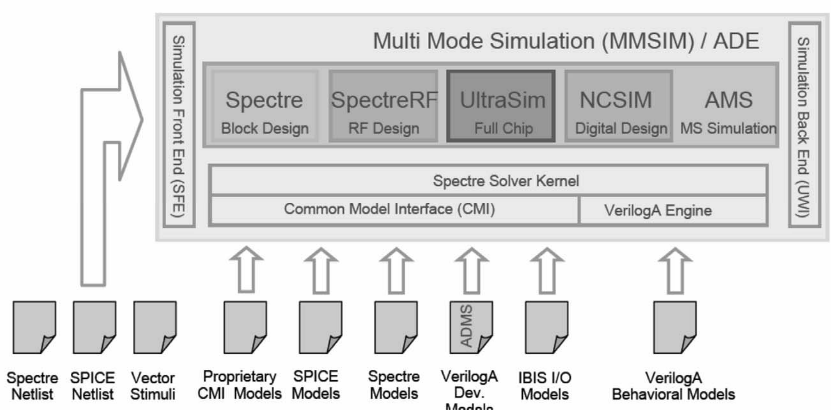
- **UltraSIM, HSIM, NanoSim are used for this purpose.**
- **In LPE extracted netlist, since it contains so many resistors and capacitors, thus the simulator must have the capability for large circuit and have following features:**
  - ➢ Automatic RC reduction
  - ➢ Simulation time and simulation accuracy is adjustable.
  - ➢ Special simulation algorithm for speed-up circuit simulation for regular circuits (ex: RAM blocks)
  - ➢ Mixed-mode simulation ability (ex: co-sim with Verilog)

Full-Chip Simulation and Verification Strategy

- **What simulations must be run to know the device will operate to specifications ?**
- **What is the highest level of abstraction I can use to achieve my accuracy goals ?**
- **How can I best apply the tools at my disposal ?**



Multi-Mode Simulator (MMSIM)



- The AMS simulator which is a single executable combining the Spectre circuit solver and Incisive-NCSIM digital solver. In addition the UltraSim FastSpice simulator is integrated with the AMS simulator for high speed transistor level verification.

下一代的學習能力更強

- 你再不多學點東西就落伍囉
  - ➢從小學習，長大也要繼續學，畢業了也要繼續學習。

HTML=Hyper Text Markup Language
<h1>hello beautiful wonderful baby</h1>