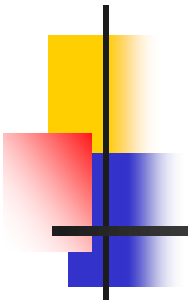
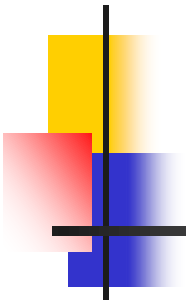
# Cell Based Design Flow Overview



# Outline

- Overview
- Behavior Simulation (RTL)
- Logic Synthesis
- Gate Level Simulation
- Physical Design
- Layout Verification
- Post-Layout simulation
- HSPICE Simulation



# Our Goal

- 使修課學生透過友善的圖文解說,從而了解整個Cell-Based Flow的流程.

晶片設計製作中心
晶片及系統
設計流程

DSP系統設計
SPW / HDS

功能模擬/驗証
Verilog,VHDL

邏輯合成/驗証
Synopsys Design Analyzer

邏輯閘層模擬/驗証
Verilog,VHDL,Debussy

電路模擬/驗証
HSPICE / TimeMill

佈局/驗証
Layout Edit/
DRC,ERC,LVS,LPE

佈局及繞線/驗証
SE / Apollo
DRC,ERC,LVS,LPE

分割,佈局,繞線
Xilinx,Altera

Full-Custom

Cell-based ASIC

FPGA

CPLD

佈局後模擬/驗証
HSPICE,Timemill,
Verilog,VHDL

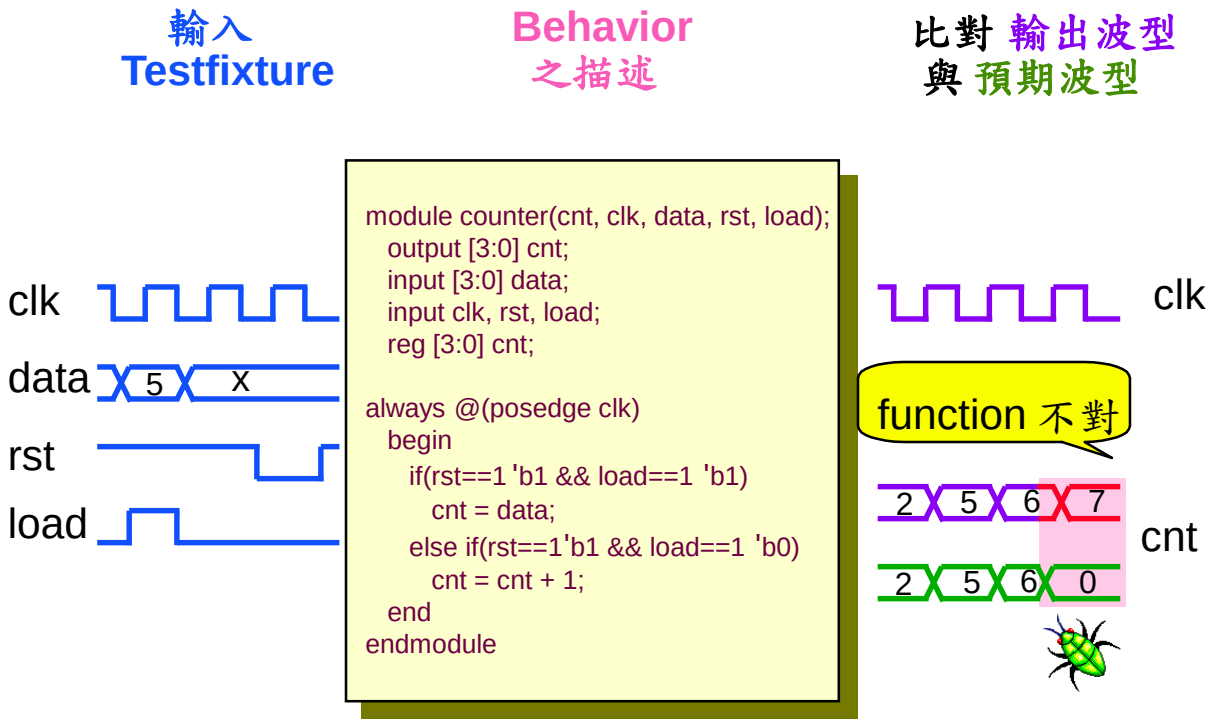
佈局後模擬/驗証
Viewlogic,Altera
Verilog,VHDL

佈局
GDSII

可程式邏輯元件/
印刷電路板

# Behavior(Function) Simulation

使用 Hardware Description Language (Verilog or VHDL) 描述電路的行為, 為了驗證所描述的 Behavior (function) 之正確性, 必須做 Behavior Simulation.
當Simulation 的結果產生之後, 我們拿它與所希望的結果相比較, 看看是否一致.
如果對結果不滿意, 就需修改 Behavior Description, 直到 Simulation 的結果符合所需為止.

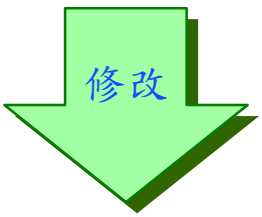


# 修改 Description

找出Behavior Description 中錯誤的地方並修改它.

```
module counter(cnt, clk, data, rst, load);
  output [3:0] cnt;
  input [3:0] data;
  input clk, rst, load;
  reg [3:0] cnt;

  always @(posedge clk)
  begin
    if(rst==1'b1 && load==1'b1)
      cnt = data;
    else if(rst==1' b1 && load==1'b0)
      cnt = cnt + 1;
  end
endmodule
```




修正 function

```
module counter(cnt, clk, data, rst, load);
  output [3:0] cnt;
  input [3:0] data;
  input clk, rst, load;
  reg [3:0] cnt;
  always @(rst)
  begin
    if(!rst)
      assign cnt =4 ' b0;
    else
      deassign cnt;
  end
  always @(posedge clk)
  begin
    if(rst==1 'b1 && load==1 'b1)
      cnt = data;
    else if(rst==1' b1 && load==1 'b0)
      cnt = cnt + 1;
  end
endmodule
```

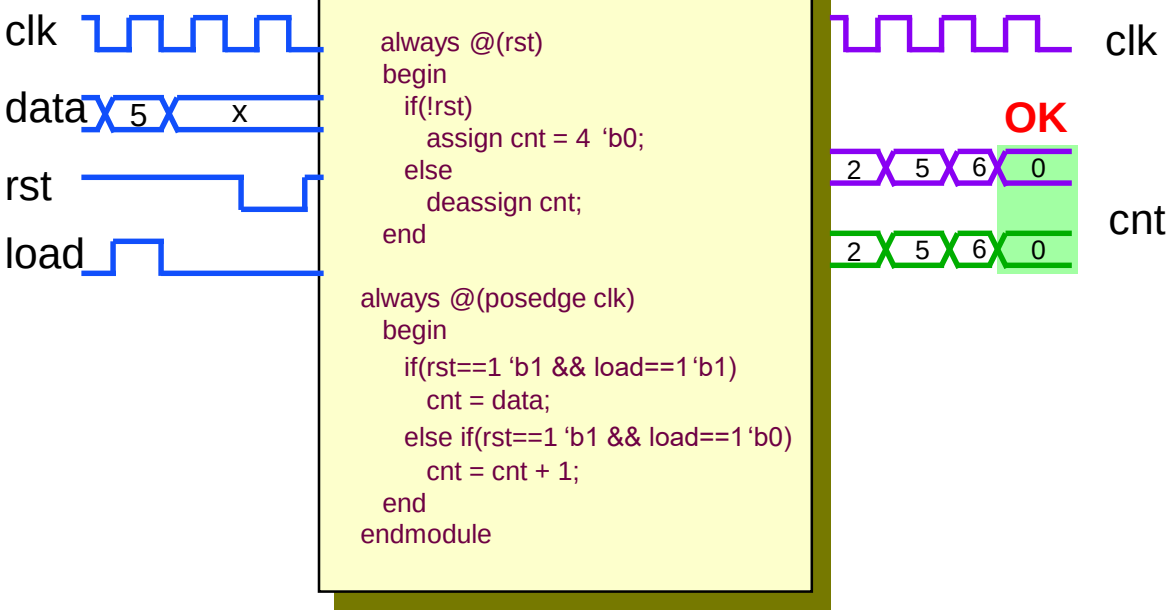
# Re-Simulation

修改完之 Behavior Description 要再做 Simulation,
檢查其輸出波型是否符合要求.

輸入
Testfixture

Behavior
之描述

比對 輸出波型
與 預期波型



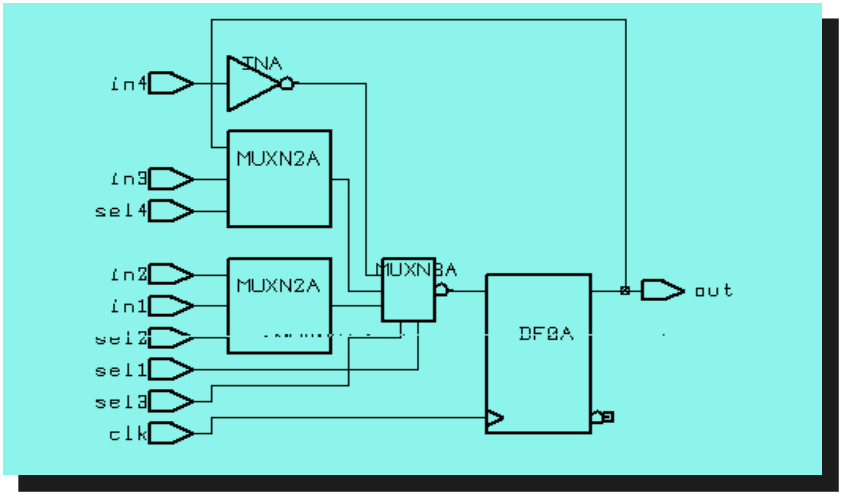
# 邏輯合成
# Logic Synthesis

- Logic Synthesis ***translates*** RTL design to gate-level design

```
                    :
always @(posedge clk) begin
  if (sel1) begin
    if (sel2)
      out=in1;
    else
      out=in2;
  end
  else if (sel3) begin
    if (sel4)
       out=in3;
  end
  else out=in4;
end
                    :
```

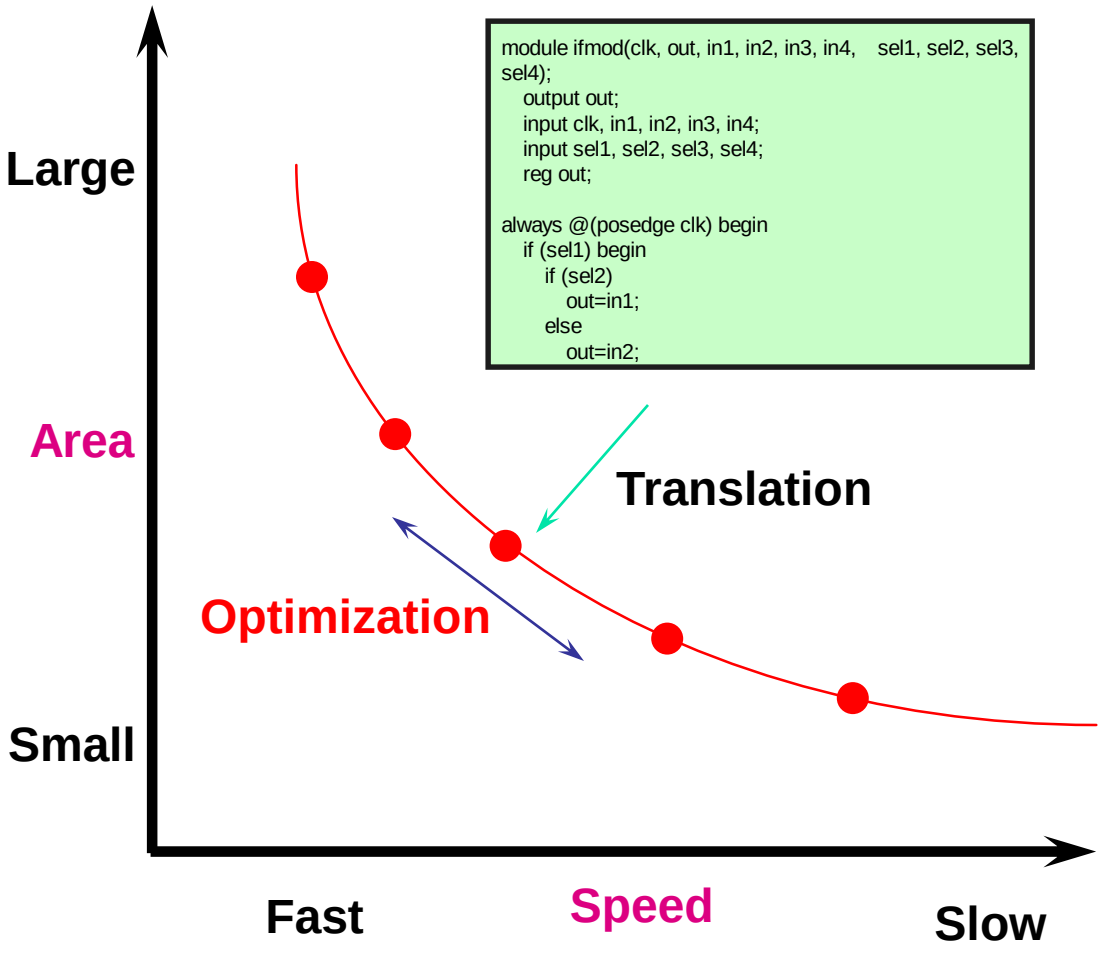
RTL

gate-level design



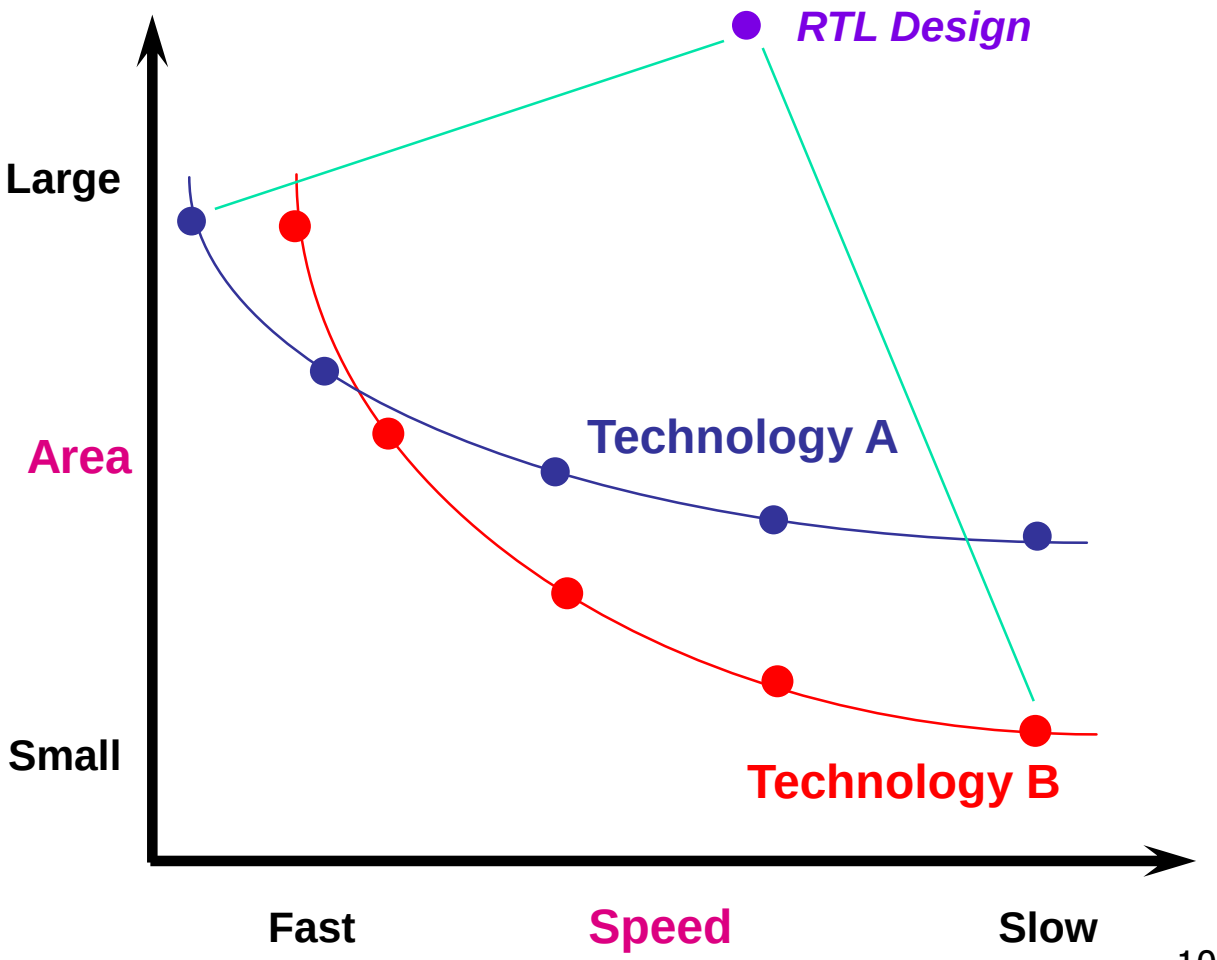
# Synthesis is Constraint Driven

- Logic Synthesizer first translates RTL design to an intermediate gate-level design, then ***optimize*** according to the **area** and **timing** constraint



# Synthesis Handles Technology Independent Design

A *RTL design* can be mapped to several different *technology*, designers can choose that meets their requirement



# Synthesis Issue (1/3)

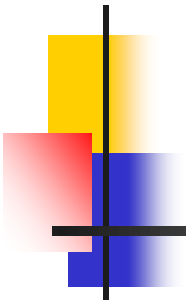
- 在合成的過程中,你只能決定 Input Signal delay, Output Signal 的 Load 還有 clock , timing 等等 constraint.

- 邏輯合成器只會根據你的條件限制幫你合成出符合Timing跟Delay要求的 Gate Level電路.

# Synthesis Issue (2/3)

- 有些實驗室會自行製作 Cell-Base Library,以直接撰寫 Gate-Level Code的方式避開這塊自己無法掌握的部分.

- 但假如你的電路gate count過大,你還是得要依賴synthesis的tool幫助你合成出 Gate Level的 Net_list.



# Synthesis Issue (3/3)

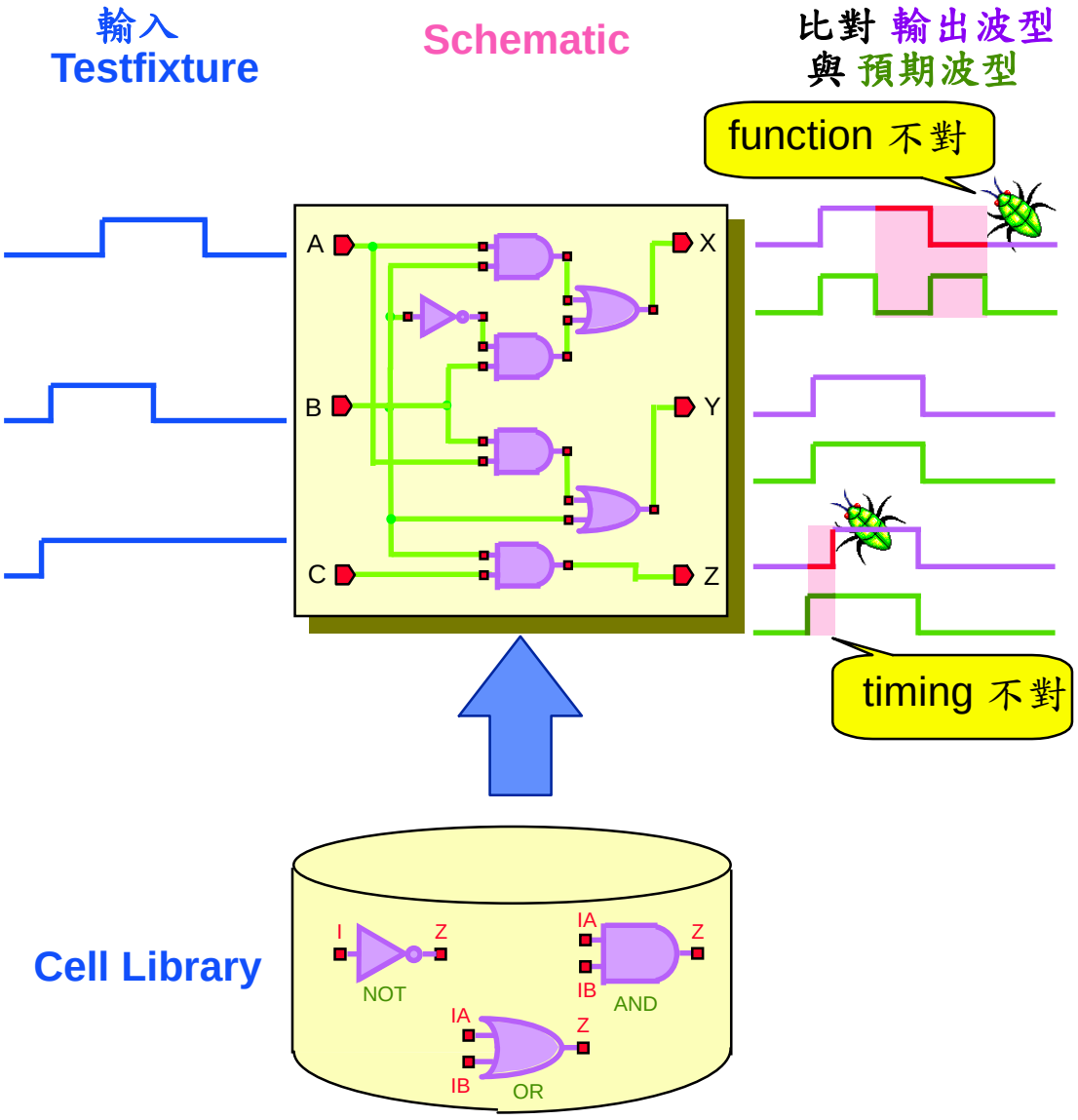
- 有些RTL Code, Synopsys 的 Design Analyzer無法幫你 Synthesis. (不可合成的Code)

- 有些RTL Code你誤以為可以合成 Filp-Flop,結果卻會合成出 Latch. (將會對整個電路的Timing與穩定性造成嚴重的影響).

# Gate Level Simulation

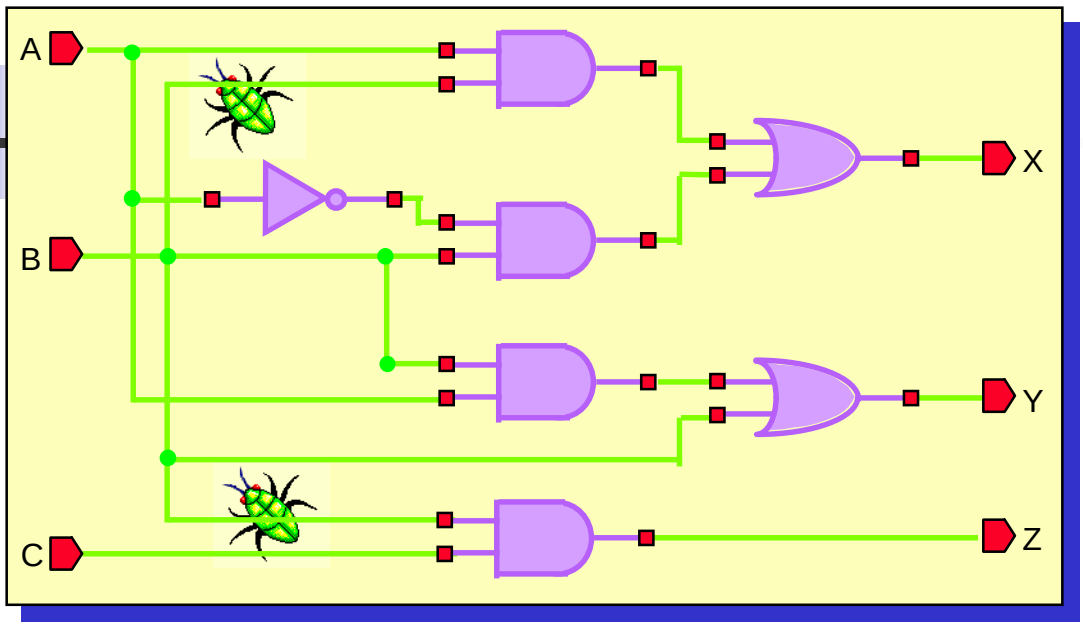
在產生**schematic**之後，必須做 Gate Level Simulation. 其目的在驗證電路的 function 之正確性及 timing 是否符合設計時的要求.

當 Simulation 的結果產生之後，我們拿它與所希望的結果相比較,看看是否一致. 如果對結果不滿意, 就需修改原先設計,直到 Simulation 的結果符合所需為止.



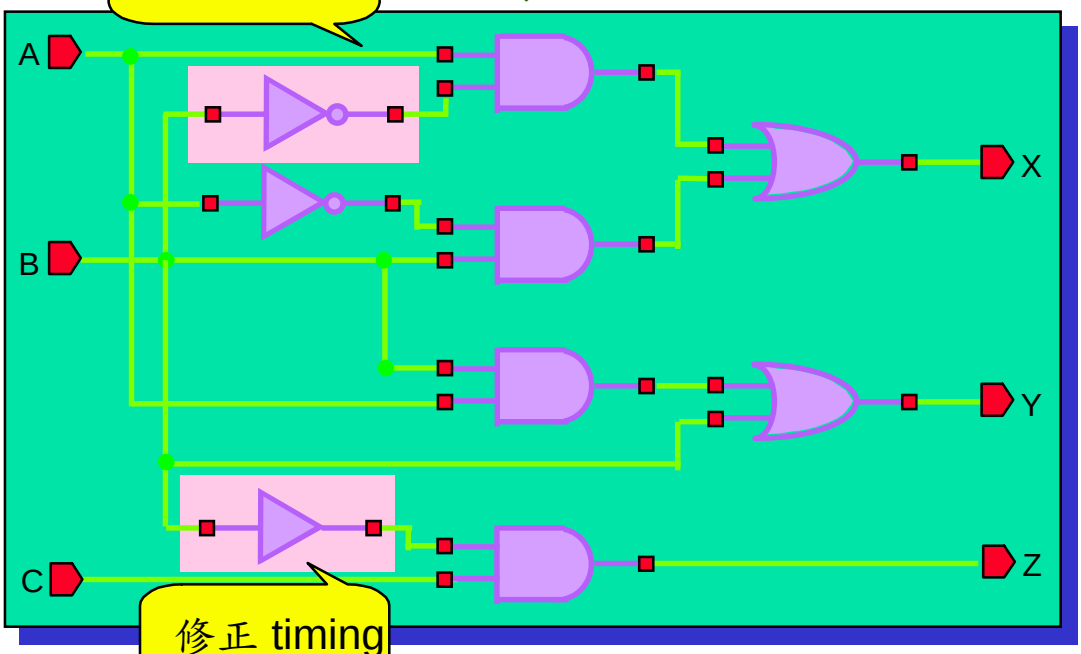
# 修改 Schematic

找出在 Schematic 中造成輸出波型錯誤的地方並修改它.




修改

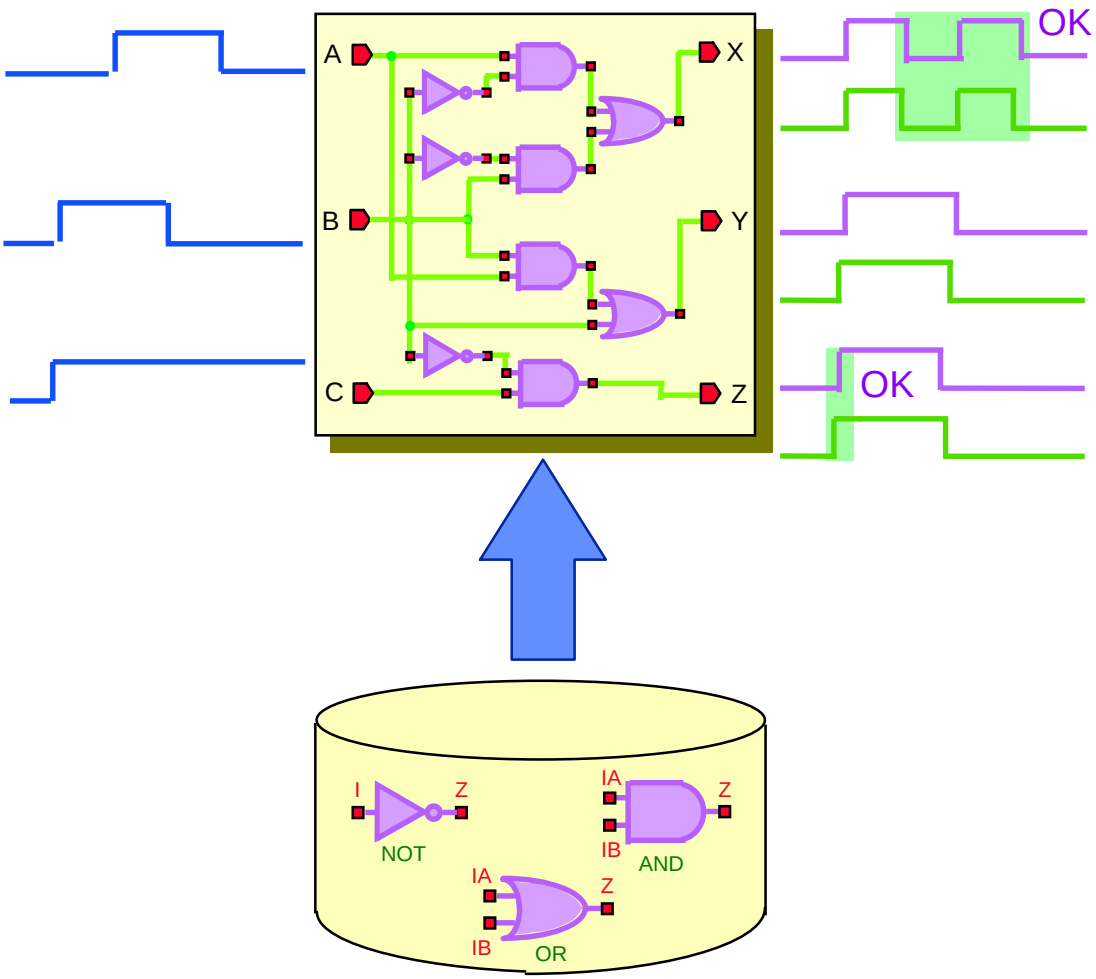
修正 function




修正 timing

# Re-Simulation

修改完之 Schematic 要再做 Simulation,
檢查其輸出波型是否符合要求.



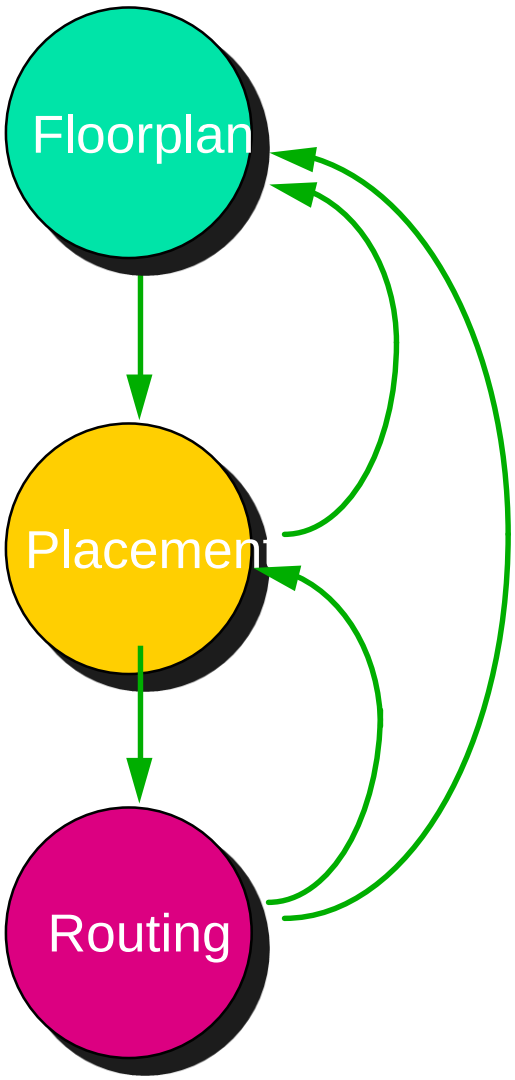
# Physical Design

Physical Design 有三個步驟 -- 首先產生 Floorplan , 然後作Placement, 再作Routing.

1. Floorplan:規畫各Block 的型狀, 大小, 位置及方向等.

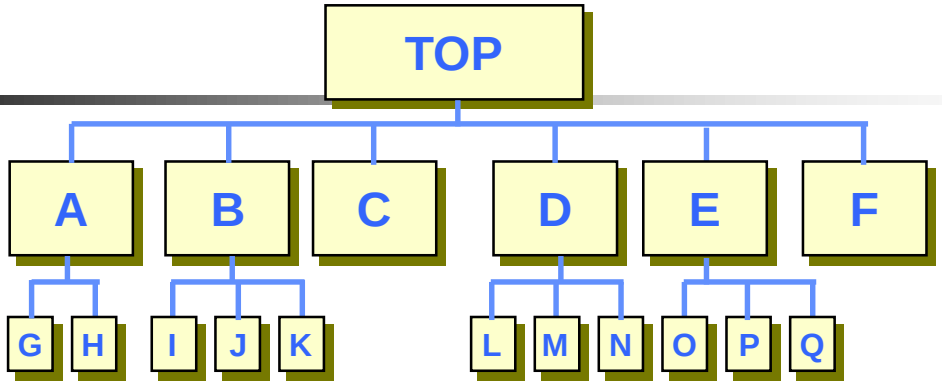
2. Placement:依 Block 實際的 size 將其固定在適當位置.

3. Routing: 做Bolck 之間及 Block 與 I/O pin 間的繞線



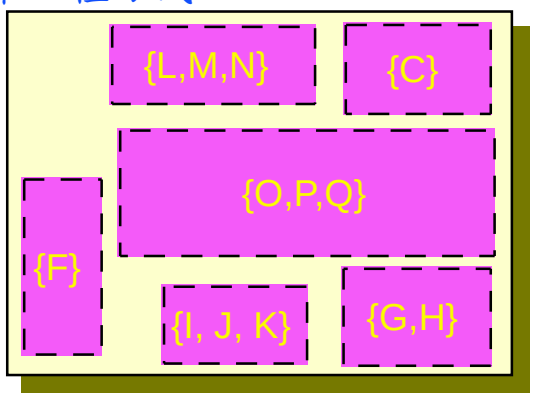
# Floorplan

## Design Hierarchy

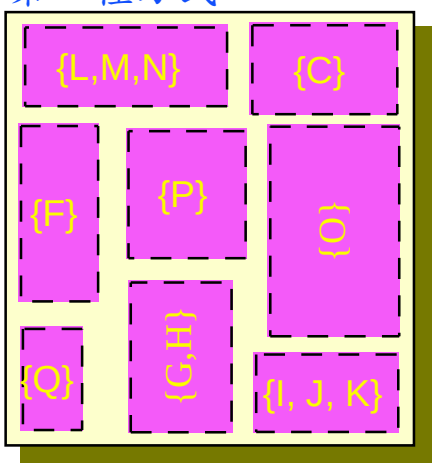



**Floorplanning**

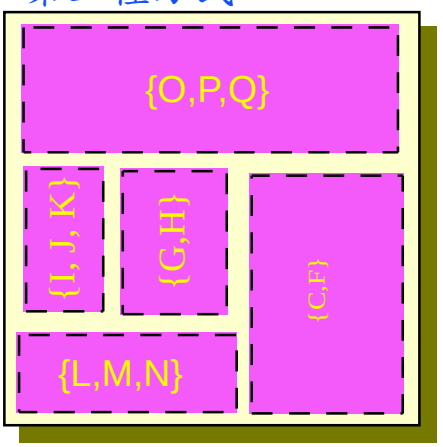
第一種方式




第二種方式

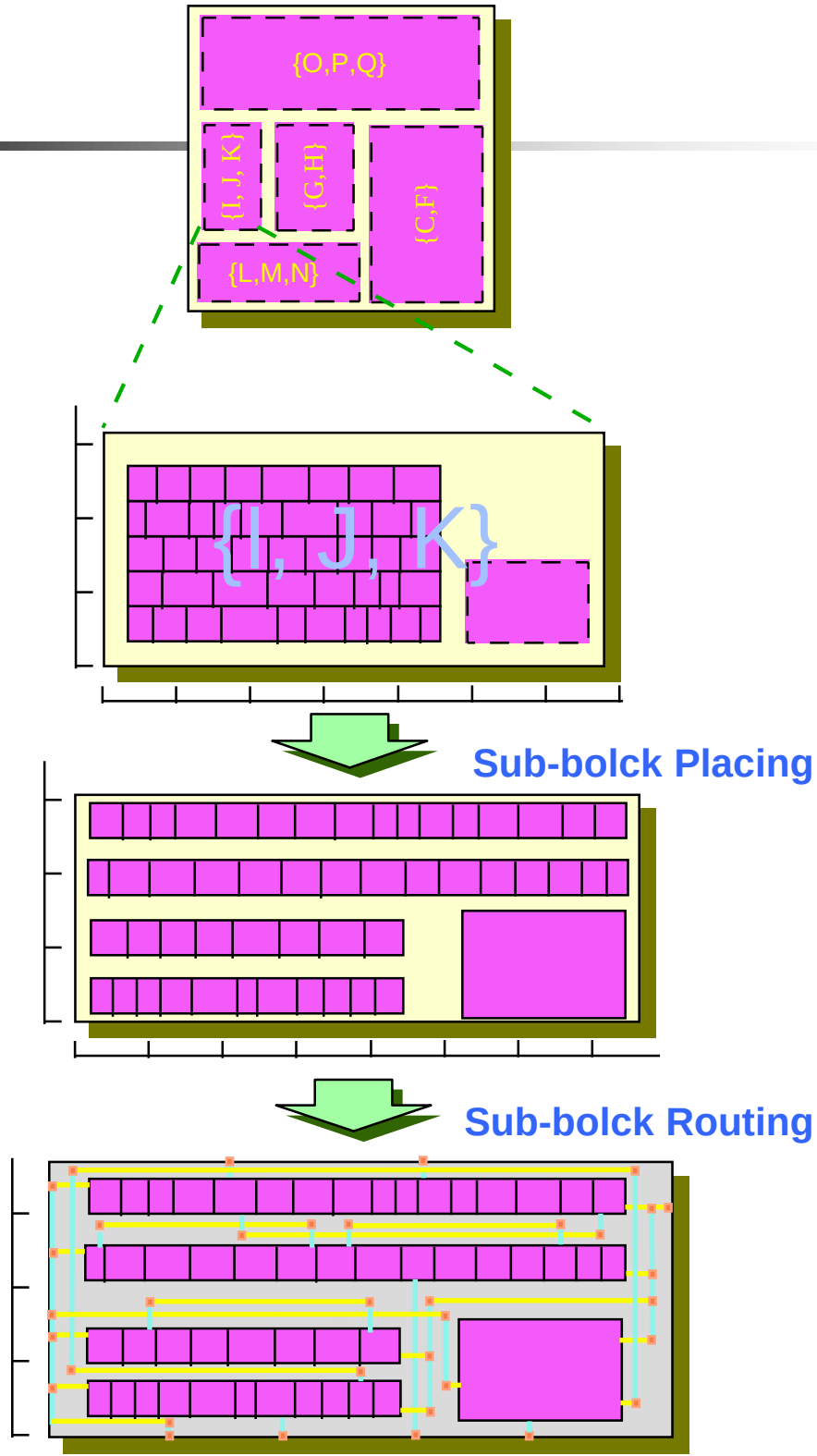



第三種方式



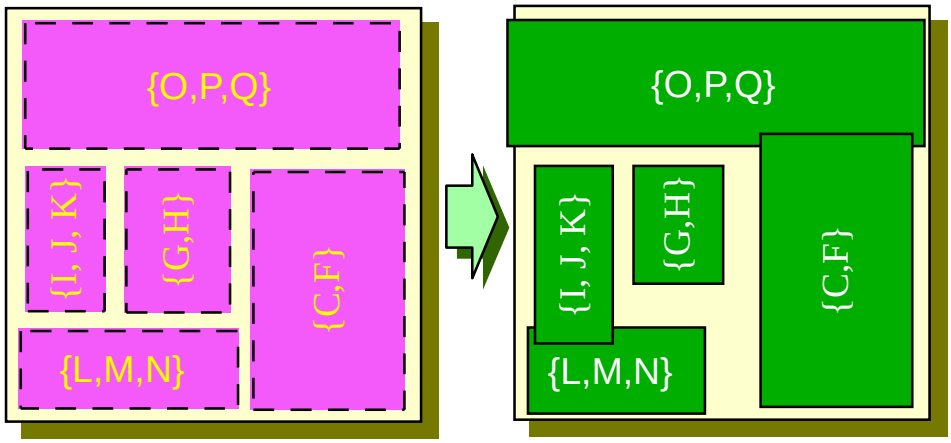
# Sub-block Floorplan, Place, Route

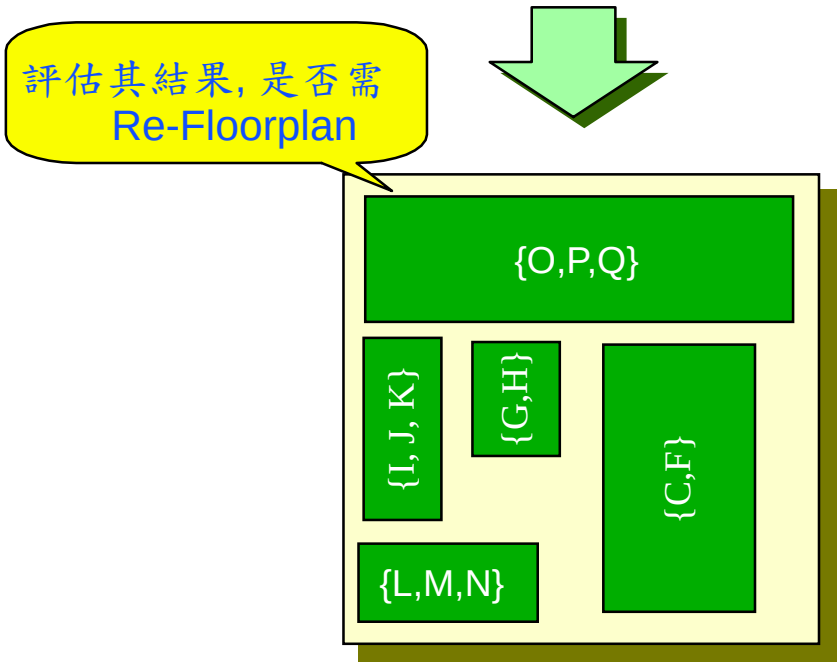
經過 Floorplan 之後, 決定 Partition 方式及各 Block 的形狀, 大小, 放置方式等. 再進行 Placement 將其固定.

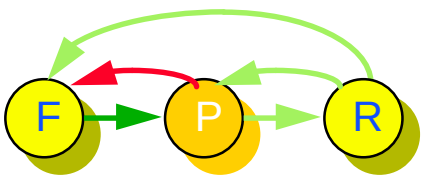


# Placement

經過 Placement 之後, 可能由於 Block Size 改變, 使結果變差. 此時就需重做 Floorplan, 然後再一次 Placement.

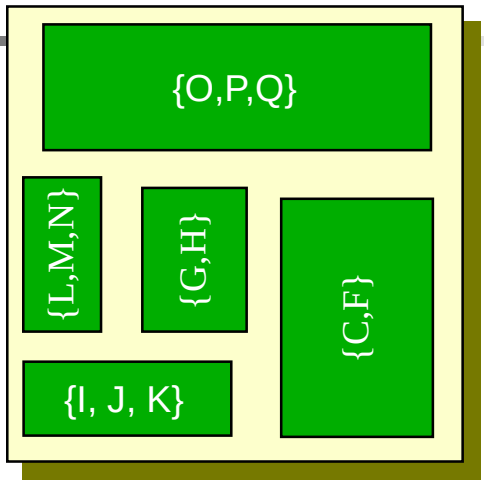


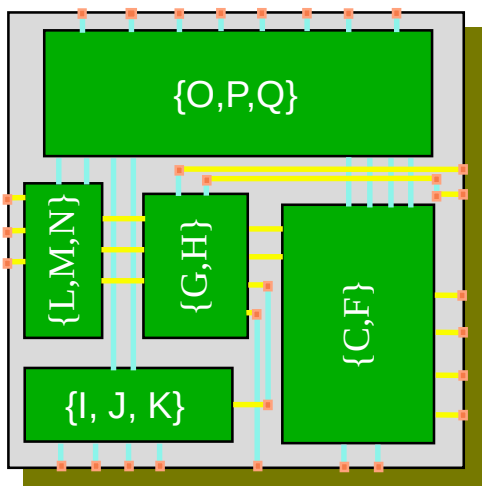



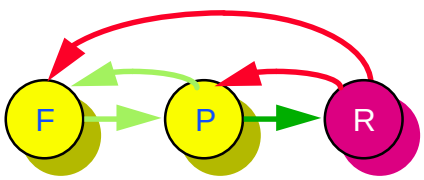
# Routing

Place 完成後就可以進行 Routing, 連接各 Block 以及 I/O Pin.





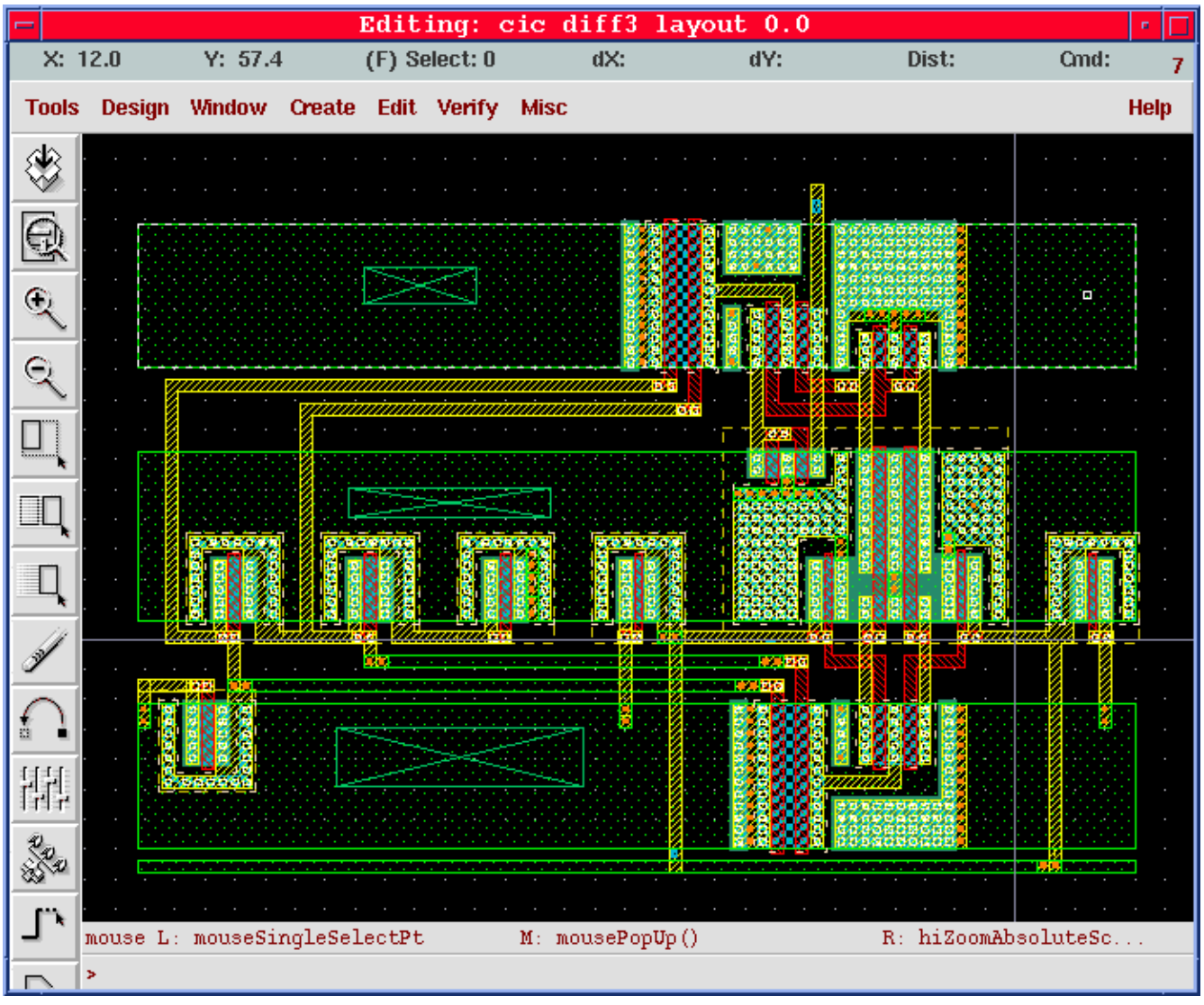


# Layout Editor
# 光罩佈局編輯軟體

電路模擬合乎規格後，就開始根據線路繪製佈局，Layout Editor即用以繪製實體電路的佈局，繪製的佈局是製做光罩的依據，而光罩為積體電路製做時必備品。

下圖即光罩佈局實例。

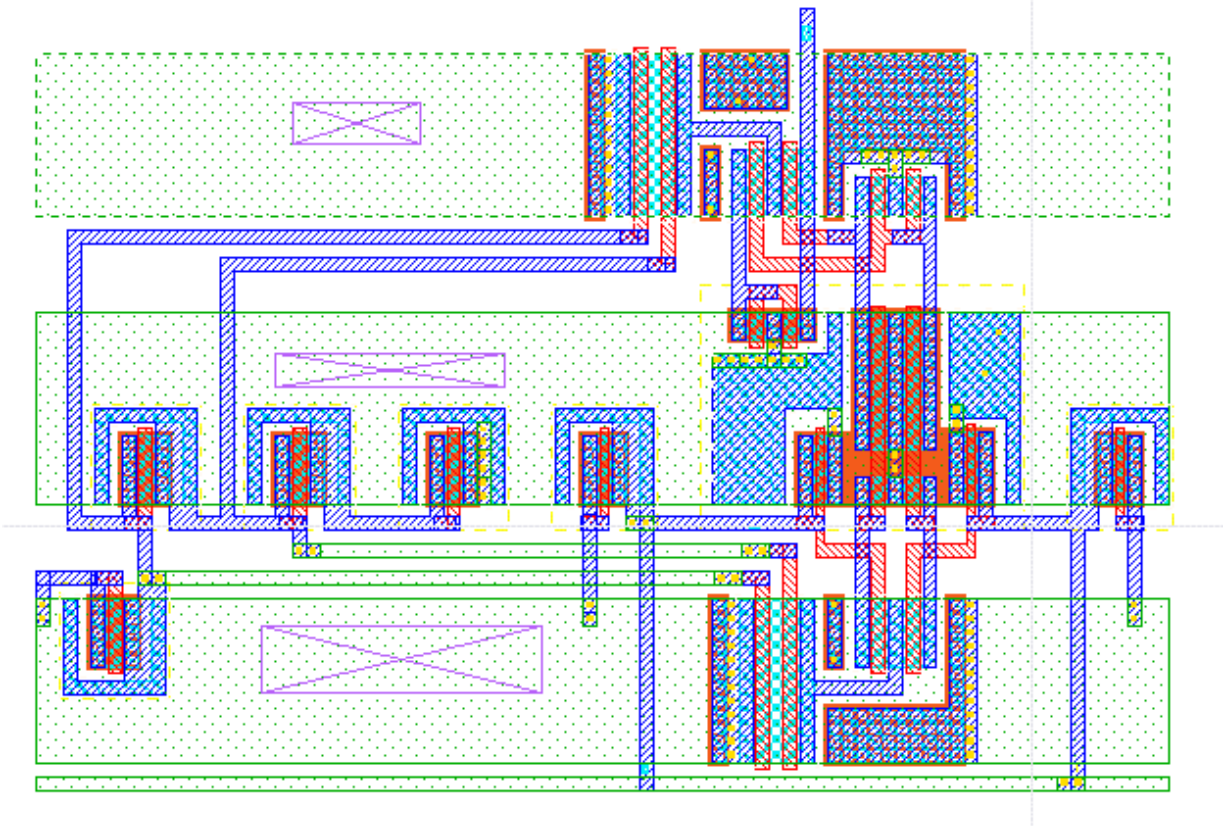


# CALIBRE -佈局驗證軟體

**Calibre用以檢查光罩佈局正確與否，包括:**
**Design Rule Check, Electrical Rule Check, Layout versus Schematic, Layout Parameter Extraction.**

**(1) DRC >> 檢查佈局是否合乎工廠所要求的佈局幾何空間規定(design rule)。**

**(2) ERC >> 檢查佈局是否有不正常的電氣特性。如短路或浮接等。**
**(3) LVS >> 檢查佈局是否與原先的設計線路一致。**
**(4) LPE >> 由佈局抽取出電晶體及寄生電路元件如寄生電容等，以netlist形式表示。**

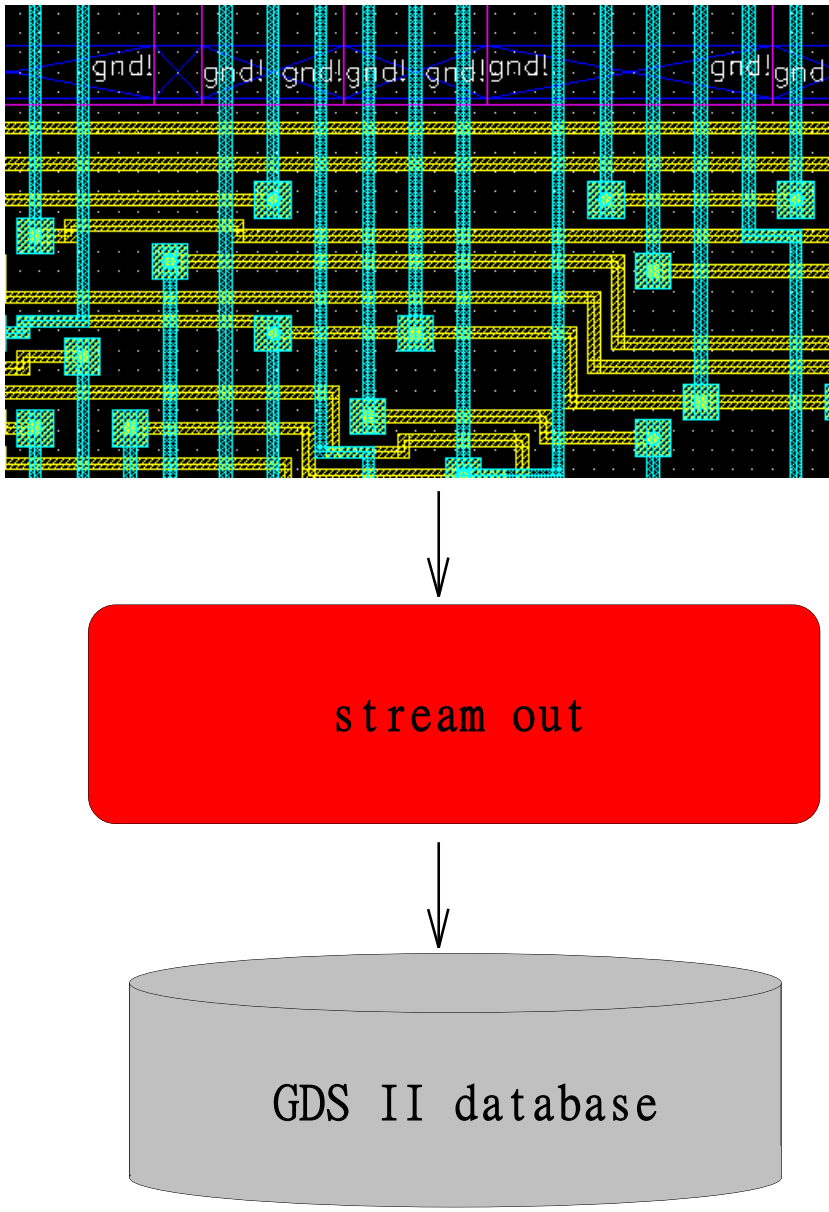


**完成(3)(4)之後，應再使用HSPICE,TimeMill或Verilog,VHDL軟體作最後的模擬驗證（post simulation)。**

# GDSII OUT

GDS II format 是一種工業界佈局資料交換的標準格式。當佈局設計完成後，必須先將佈局設計轉換成GDS II format ，以便製作光罩。

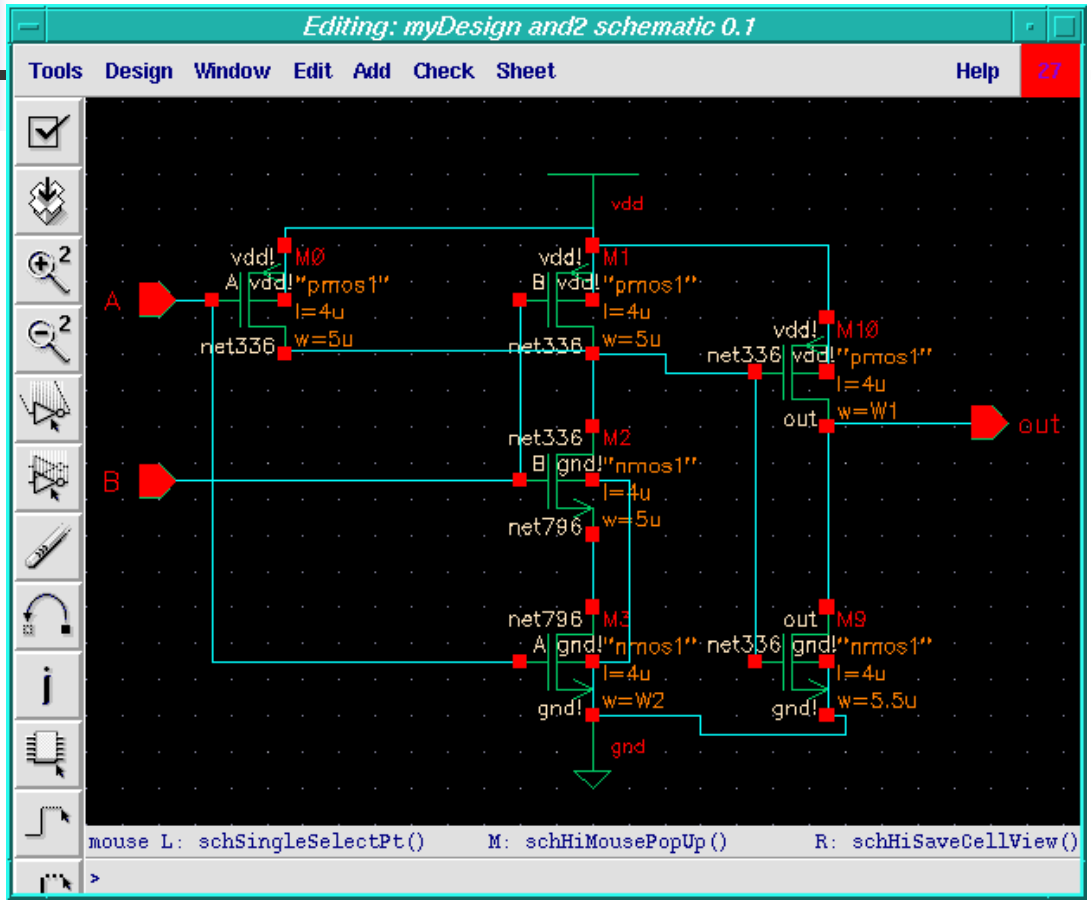
在OPUS中可利用Stream Out方式將graphic database轉換成GDS II format database 。



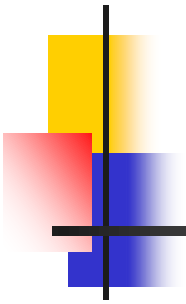
# HSPICE--電路模擬軟體

以netlist表達電路的連接，再配合電晶體模型，用HSPICE軟體可做精確的時序模擬(Timing Simulation)。

A: 將電路圖以netlist 表示




***This circuit is an impedance simulator using Op-Amp.**
**vin 1 0 ac 10mv sin(0 10mv 1khz)**
**vcc 0 7 dc 15v**
**rs 1 2 500**
**r1 7 3 47k**
**r2 3 0 5k**
**c1 4 6 10f**
**q1 4 3 5 0 qm**
**.**
**.**
**.ac dec 10 1hz 100khz**
**.op**
**.plot ac vm(6) vp(6)**
**.end**



# References


1. 清大電機 LARC實驗室博士班學生楊紹聖的授課網頁
http://larc.ee.nthu.edu.tw/~ssyang/verilog/index.htm

2. Yi-Hao Chang 與 C. P. Chen 的投影片 (2003年)

3. 國研院晶片設計中心網頁資料