嵌入式系統設計方法論
A Methodology for Embedded System Design

陳慶瀚
中央大學資訊工程學系
pierre@csie.ncu.edu.tw
2006年11月23日

大綱

PART 1 :為什麼需要方法論？

PART 2 :MIAT設計方法論

PART 3 :方法論實務

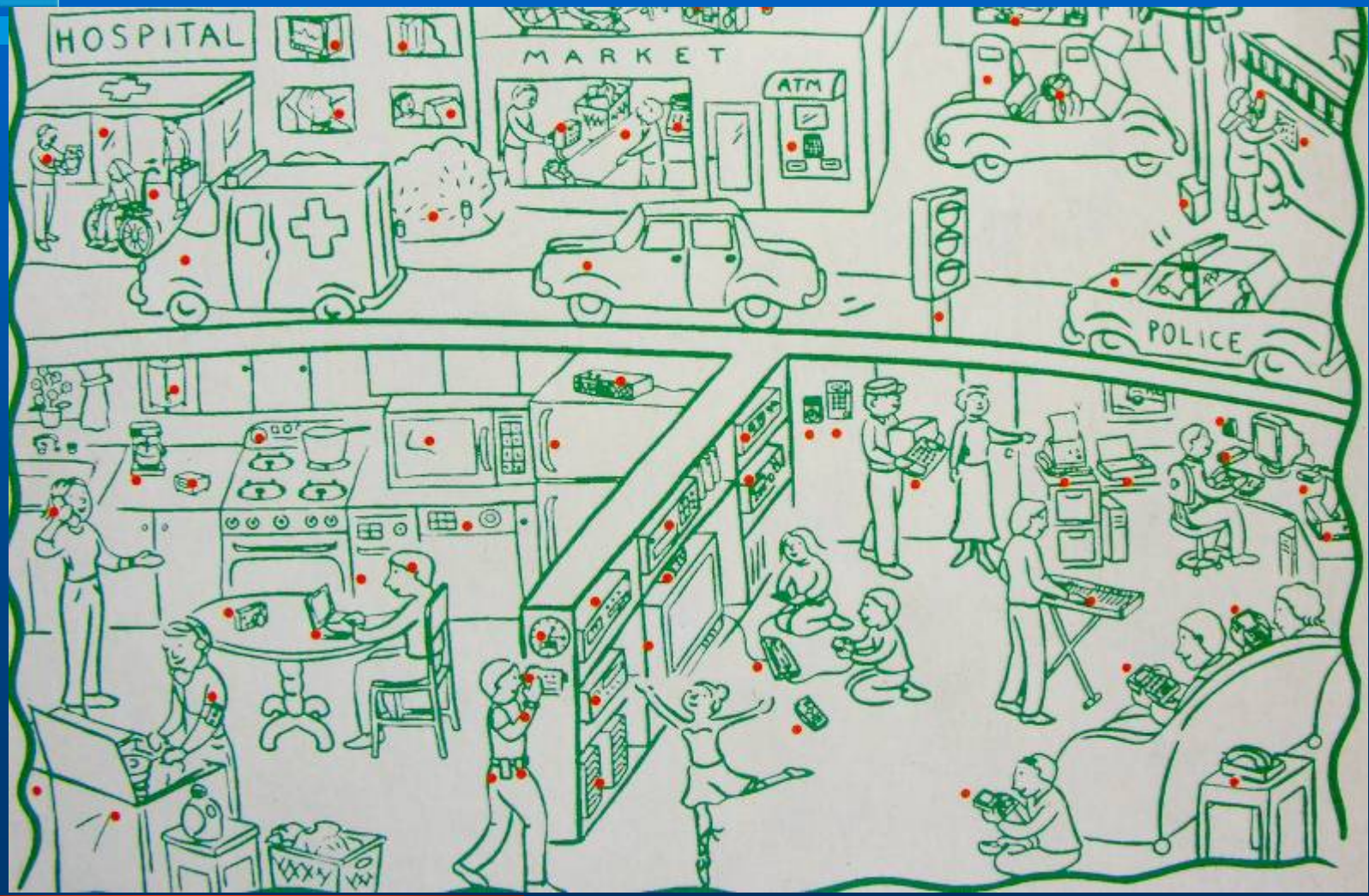
PART 4 : 複雜系統晶片設計

PART 5 : 結語

PART 1

為什麼需要方法論？

無所不在的嵌入式系統



嵌入式系統的產品趨勢

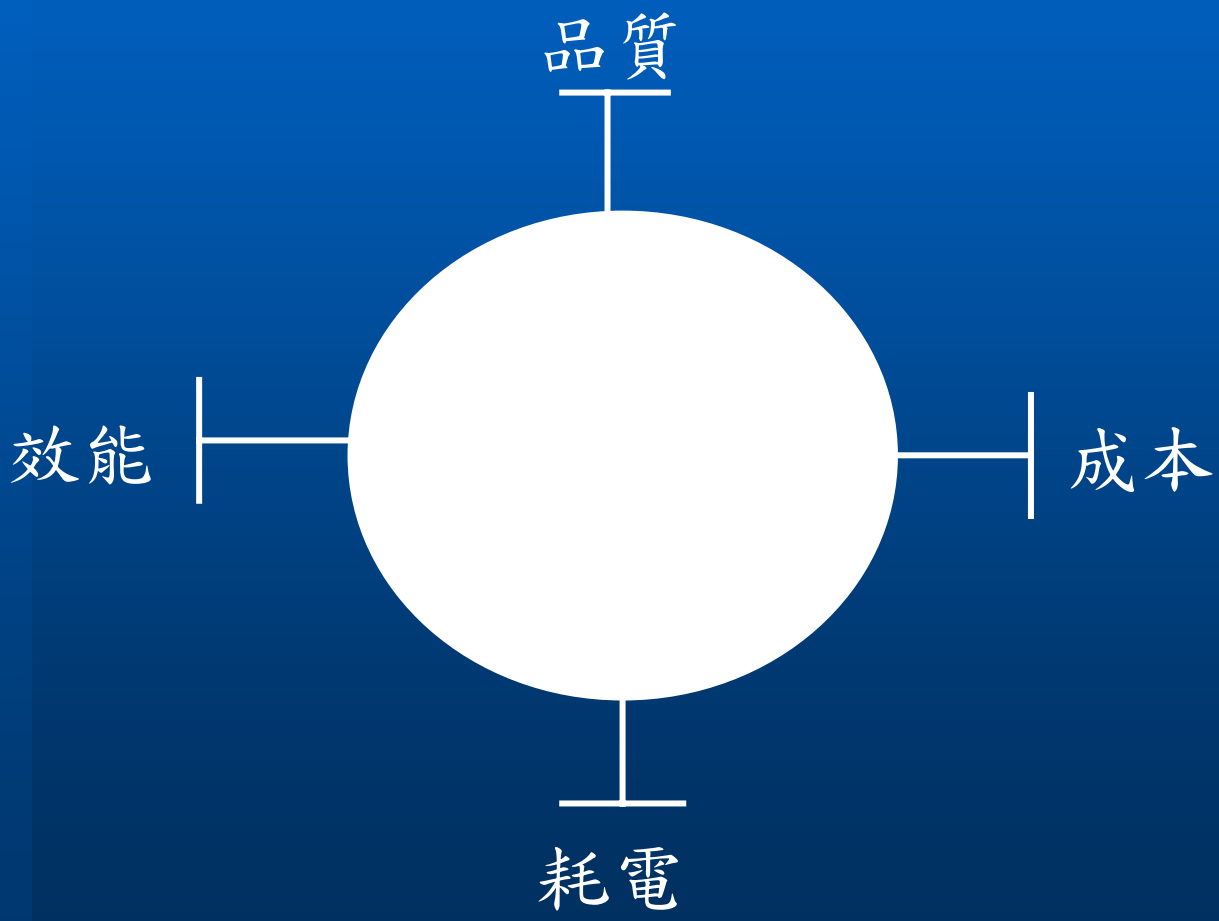
- 技術推陳出新
- 多樣化生產形式
- 產品生命週期短
- 單價低
- 內建一個以上的微處理器

>>Evolving to Complexity!

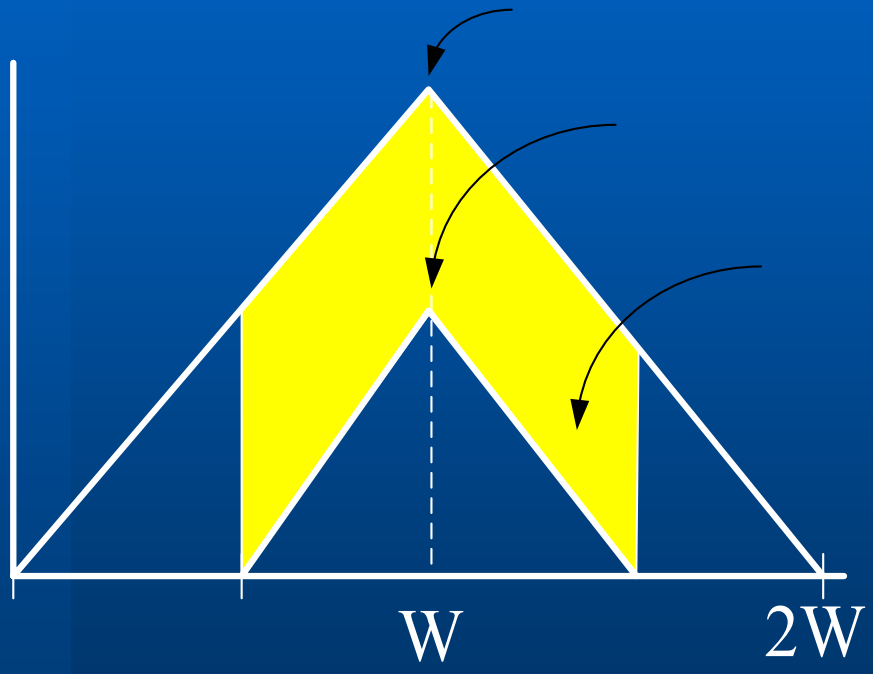
嵌入式系統設計挑戰

- 功能正確性
- 單位製造成本
- NRE(Non-Recurring Engineering)費用
- 體積
- 效能
- 耗電
- 彈性
- 原型產品產出時間(Time-to-prototype)
- 上市時間(Time-to-market)
- 維修性
- 安全性

嵌入式系統設計挑戰



Time-to-market



- 產品生命週期2W=52周, 延遲進入4周D=4，損失= 22%
- 產品生命週期2W=52周, 延遲進入10周D=4，損失= 50%

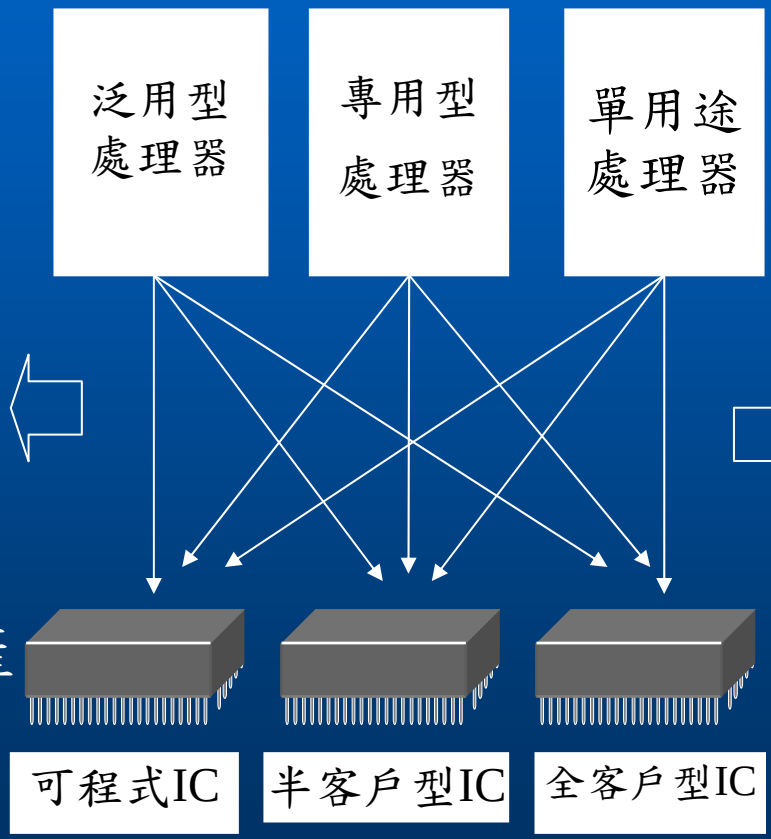
嵌入式系統設計主要技術

- 微處理器(Microprocessor): 泛用型處理器, 專用型處理器
- 晶片(IC): 全客戶型IC,半客戶型IC, 可程式IC
- 設計方法論(Design Methodology): 設計、編譯、電路合成、測試、驗證的工具、技術和方法

設計Trade-Off

泛用型

彈性
可維修性
NRE成本
快速上市
少量多樣生產



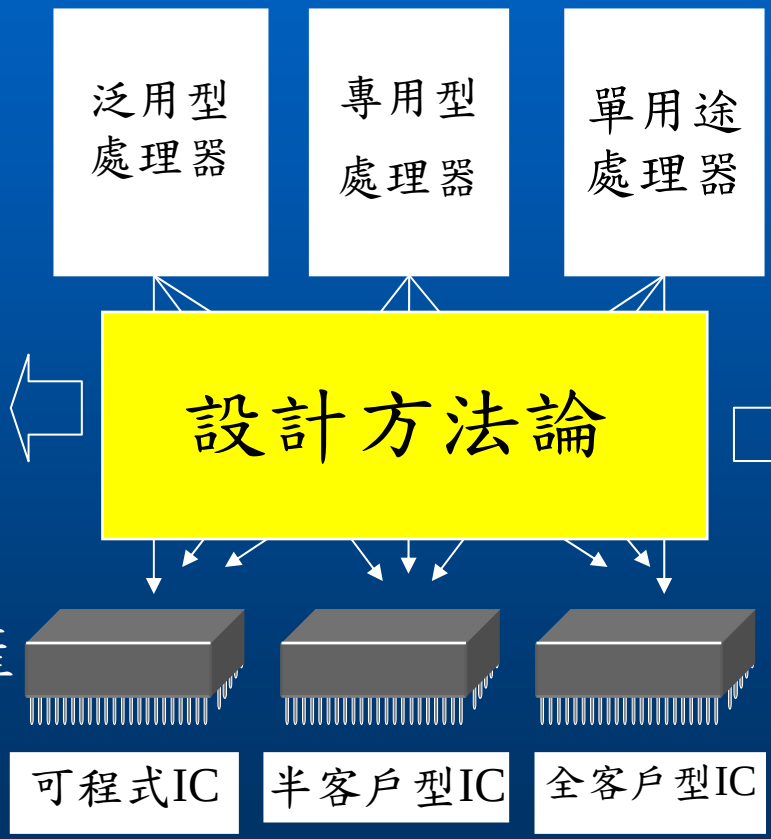
客制型

低耗電
高效能
體積輕薄短小
大量生產

設計Trade-Off

泛用型

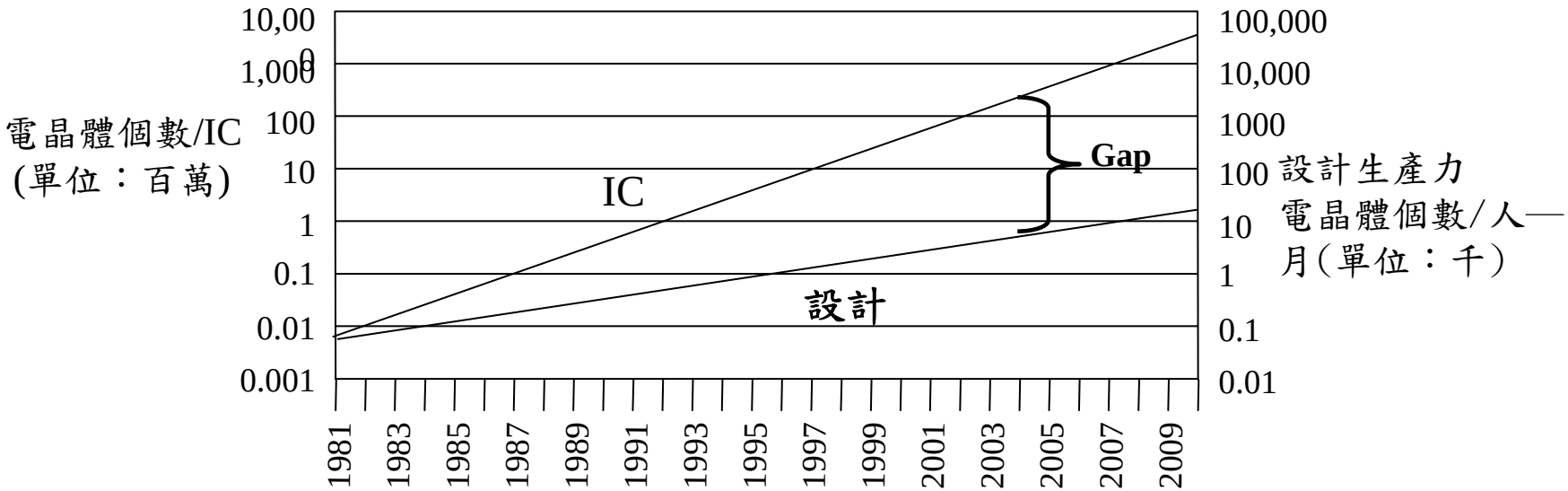
彈性
可維修性
NRE成本
快速上市
少量多樣生產



客制型

低耗電
高效能
體積輕薄短小
大量生產

為何需要設計方法論？


電晶體個數/IC
(單位：百萬)
10,00
0
1,000
100
10
1
0.1
0.01
0.001
IC
設計
Gap
100,000
10,000
1000
100
10
1
0.1
0.01
設計生產力
電晶體個數/人—月(單位：千)
1981
1983
1985
1987
1989
1991
1993
1995
1997
1999
2001
2003
2005
2007
2009

為何需要設計方法論？

—建構一個知識管理平台，使個體的經驗和知識得以累積和分享；

—保護組織的研發知識成果；

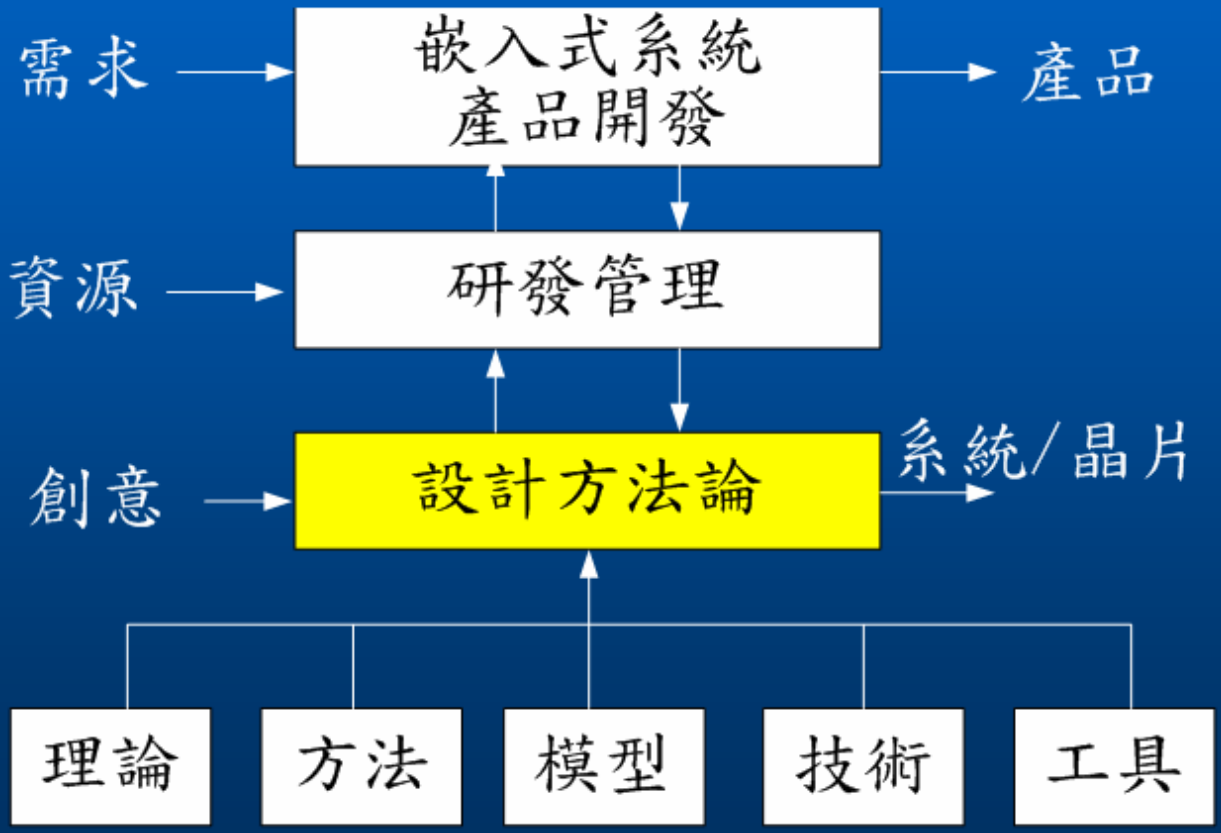
—運用群體知識，避免個人知識解題的侷限性；

—避免重蹈錯誤和資源重複投資。

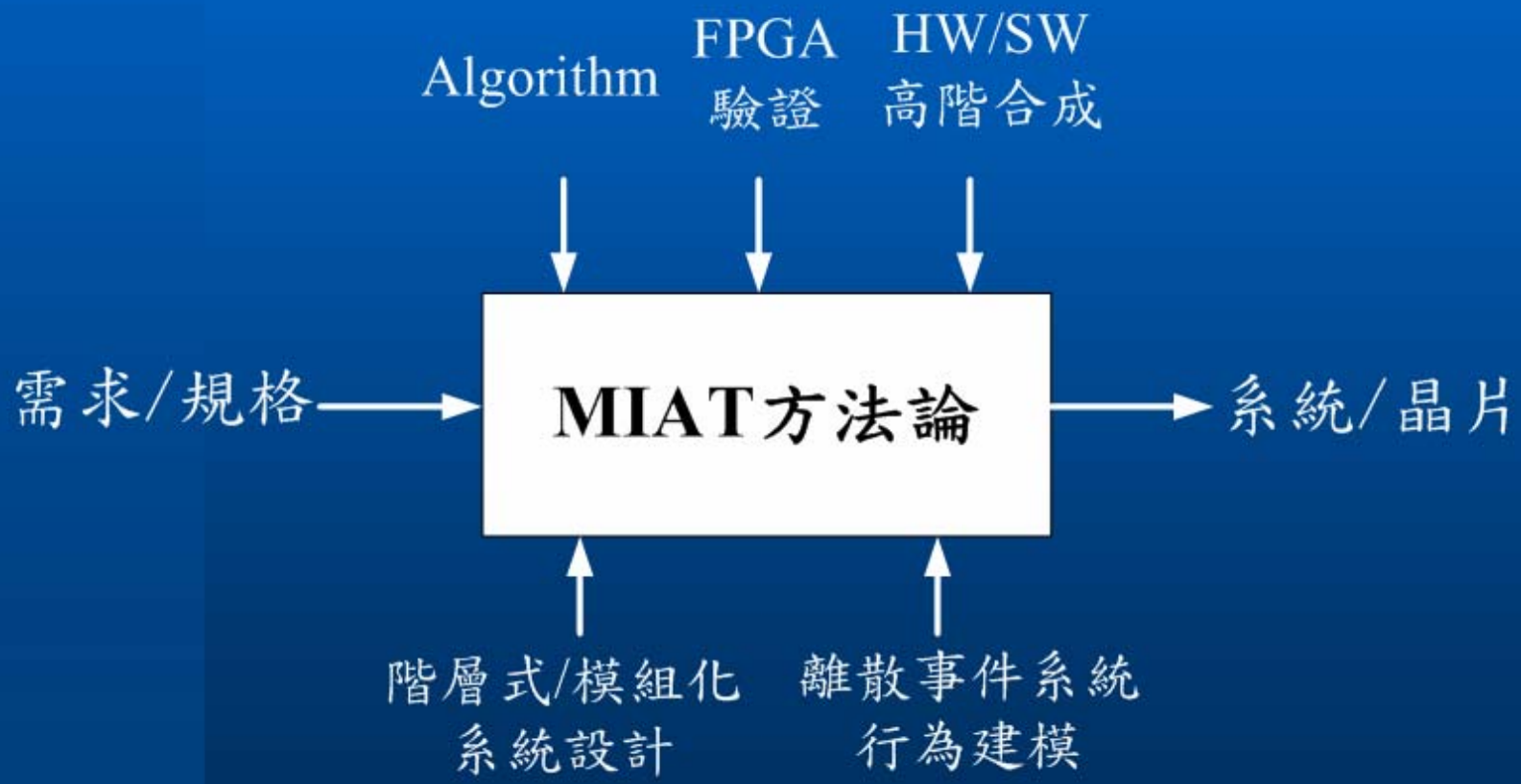
PART 2

MIAT設計方法論

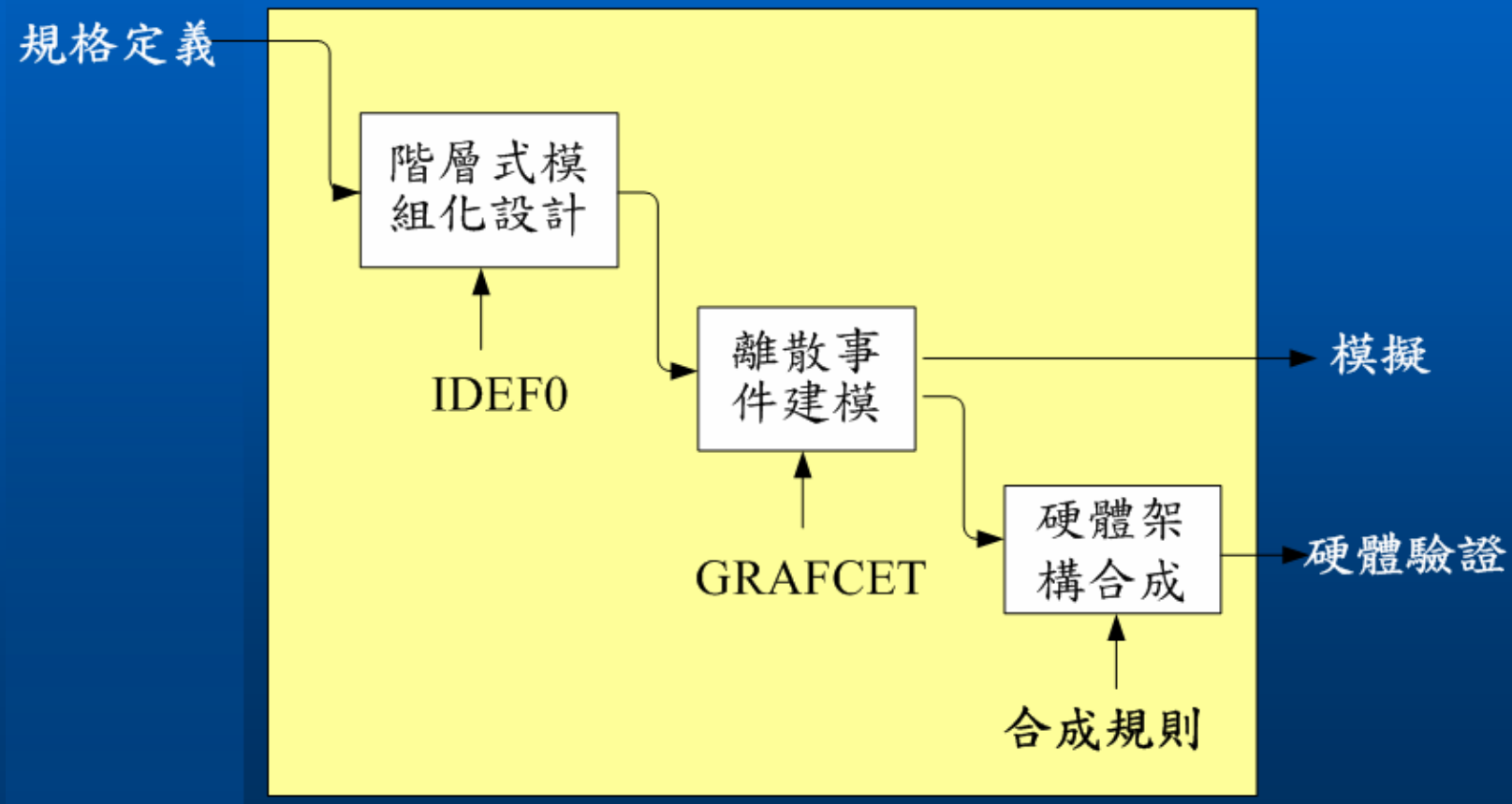
開發鏈的新角色—設計方法論



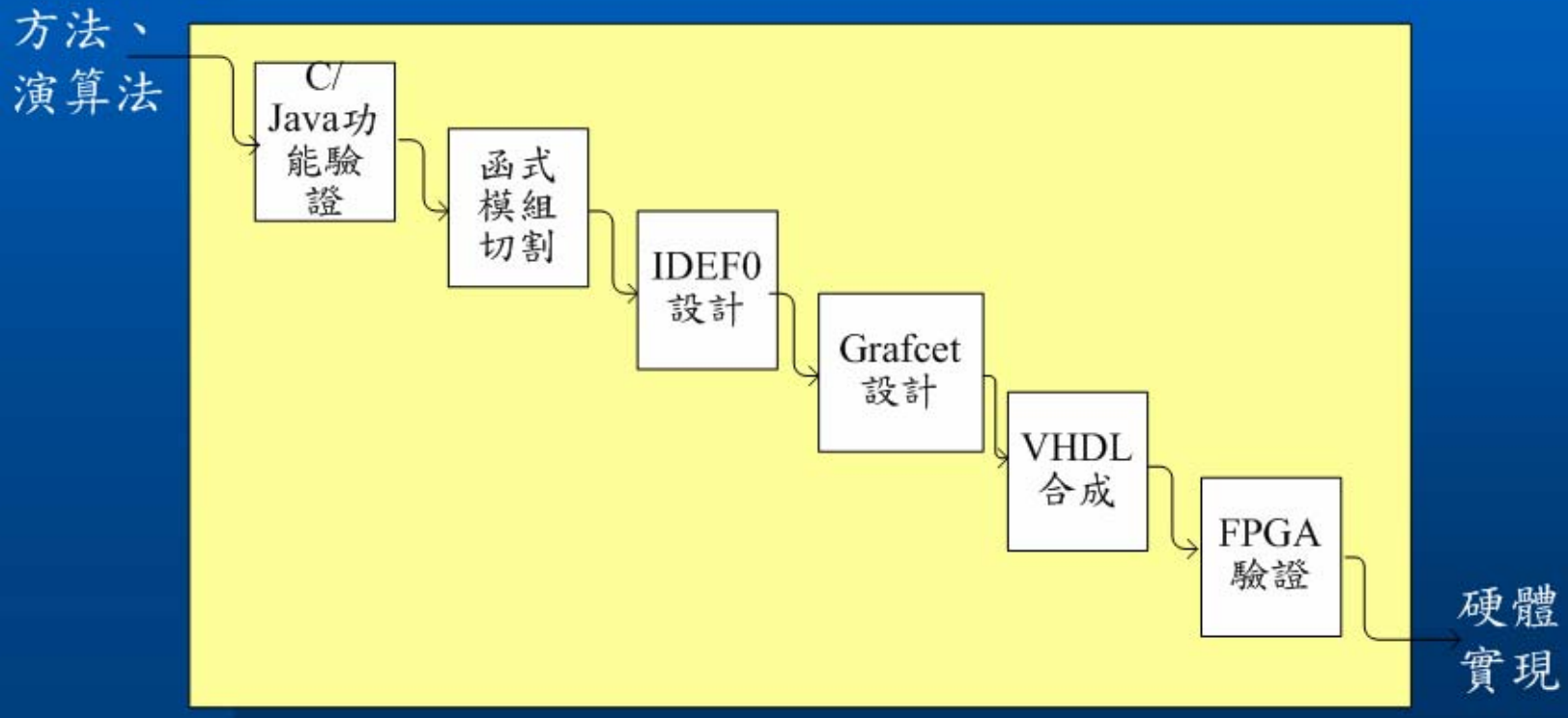
MIAT複雜系統設計方法論



MIAT方法論上層架構

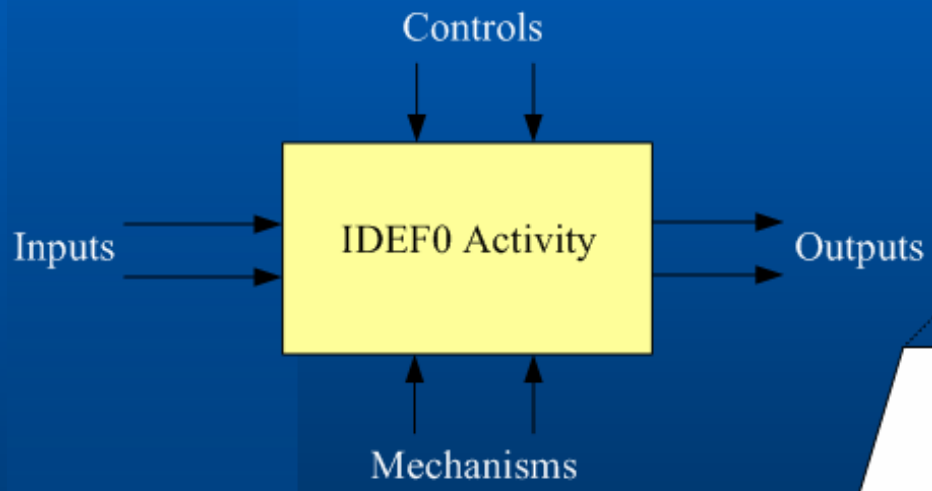


MIAT方法論的底層架構

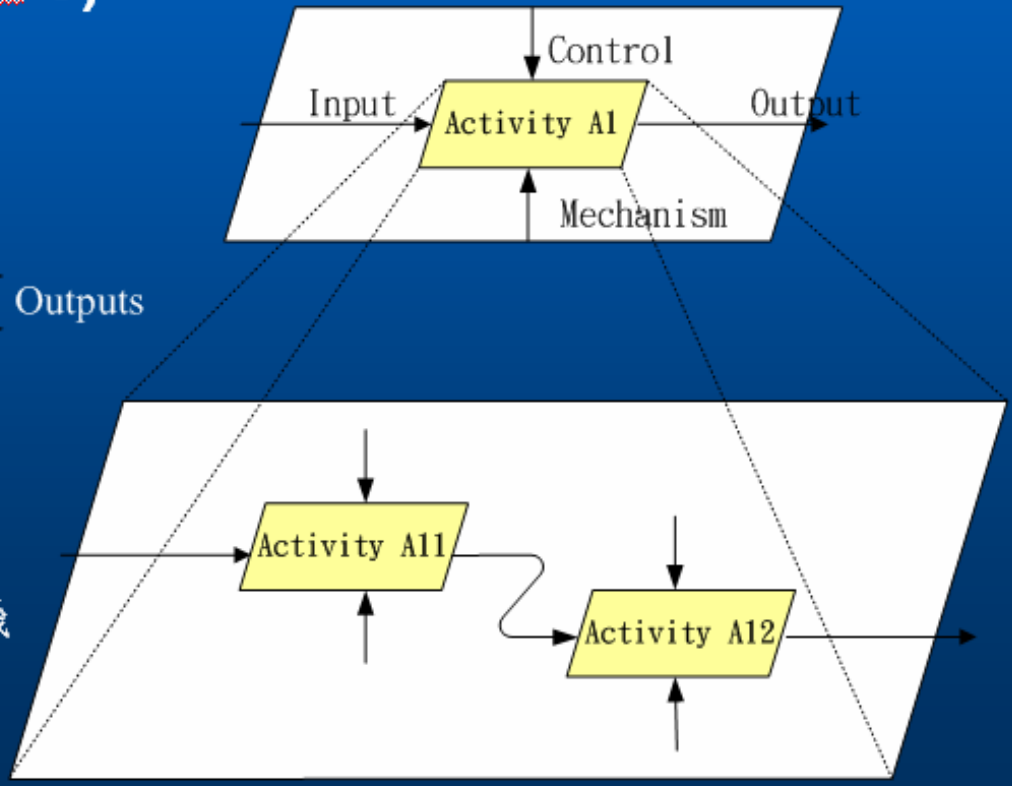


階層式與模組化設計

- IDEF0(Integrated Computer-Aided Manufacturing (ICAM) DEFinition 0)

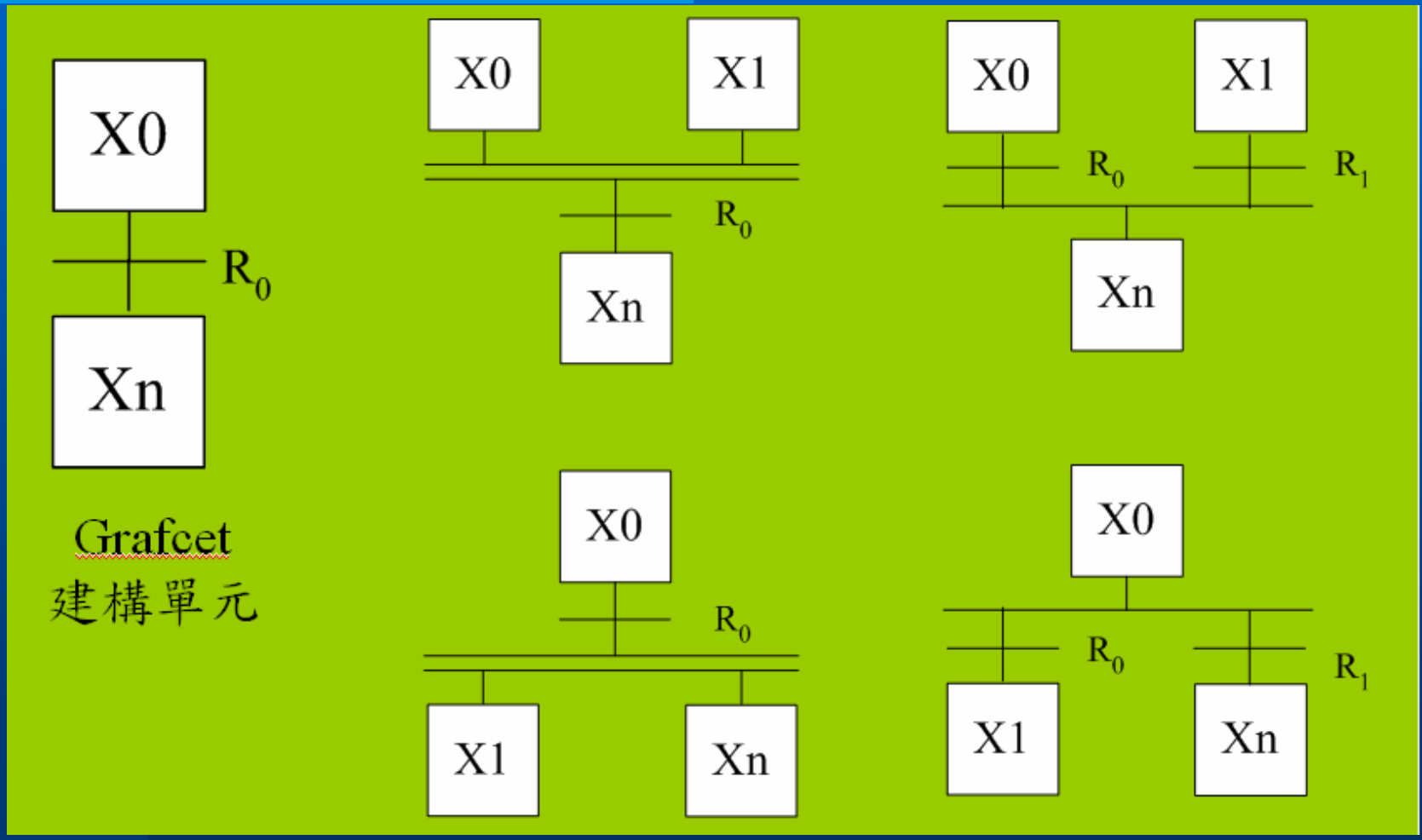


IDEF0基本模組功能方塊與箭頭意義

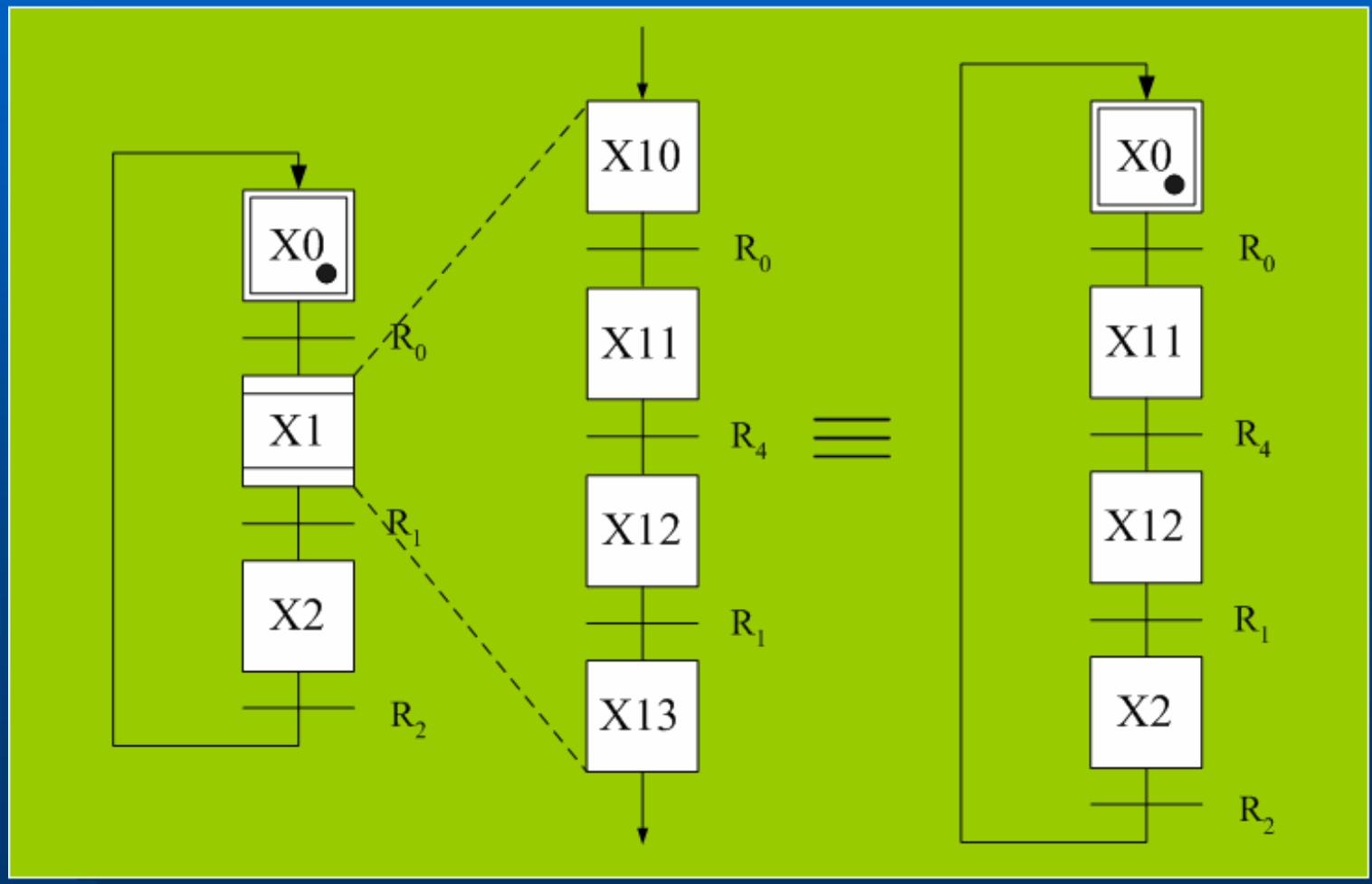


IDEF0 階層化架構

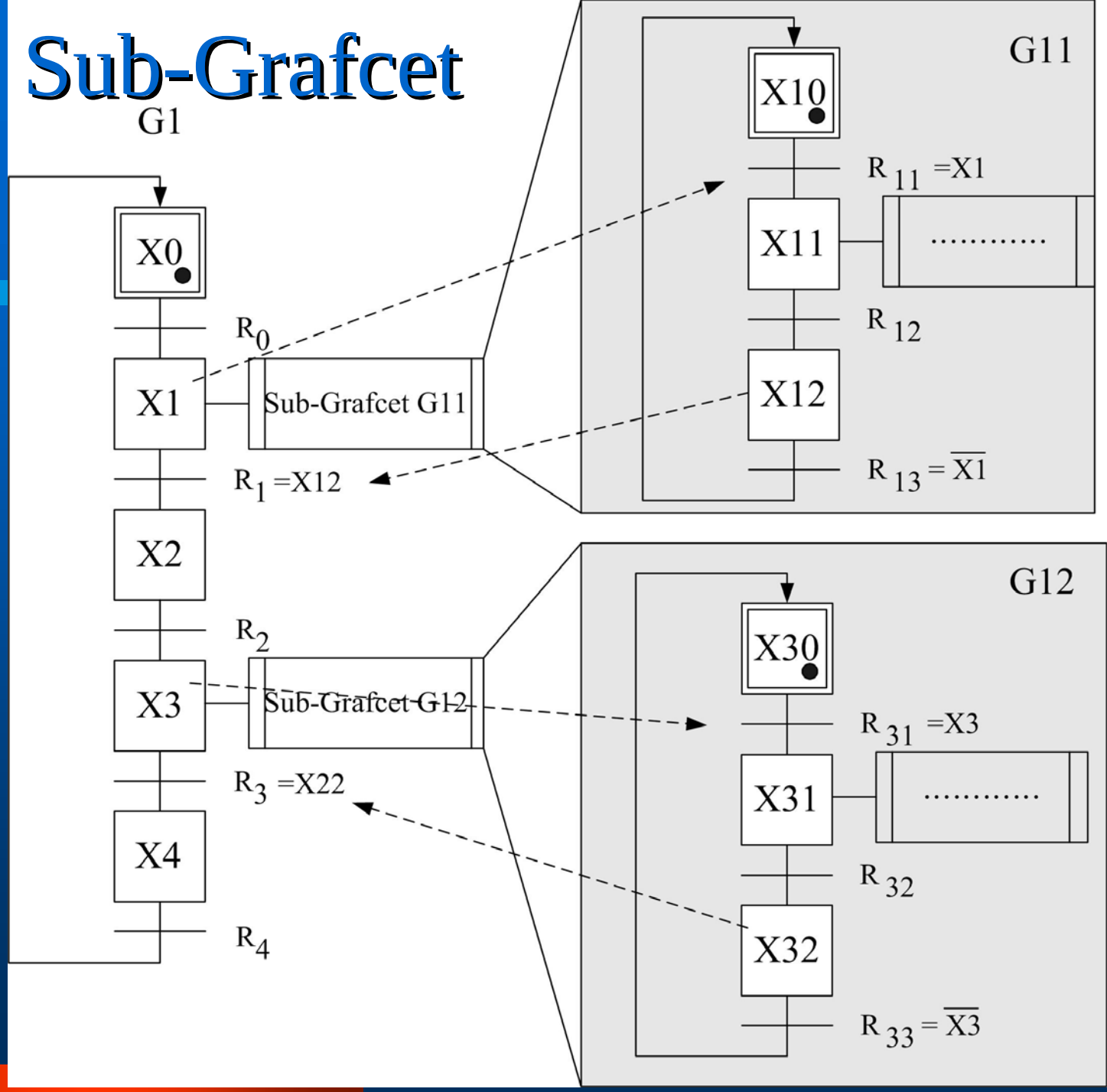
離散事件建模—GRAFCET



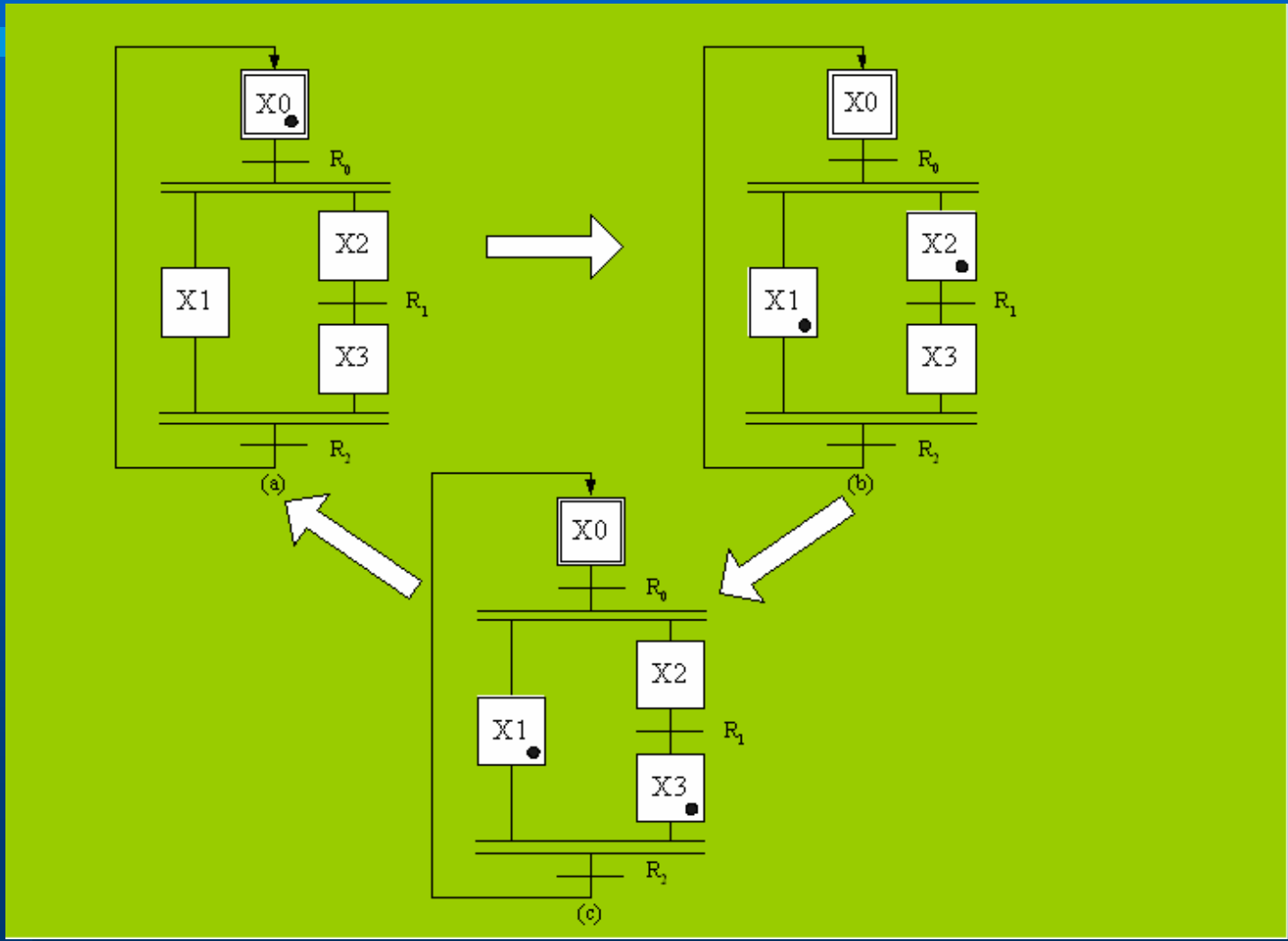
Macrostep



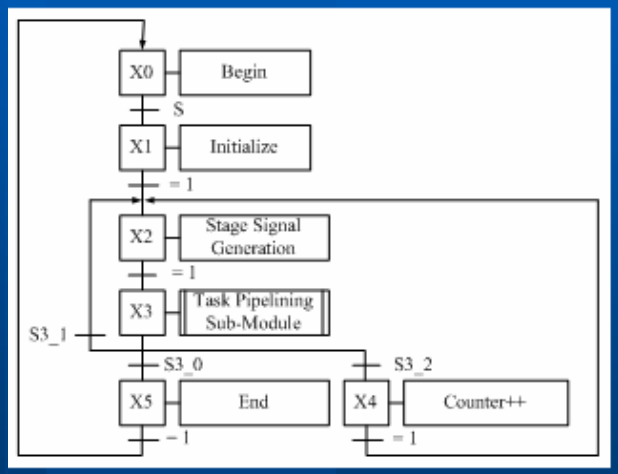
Sub-Grafcet

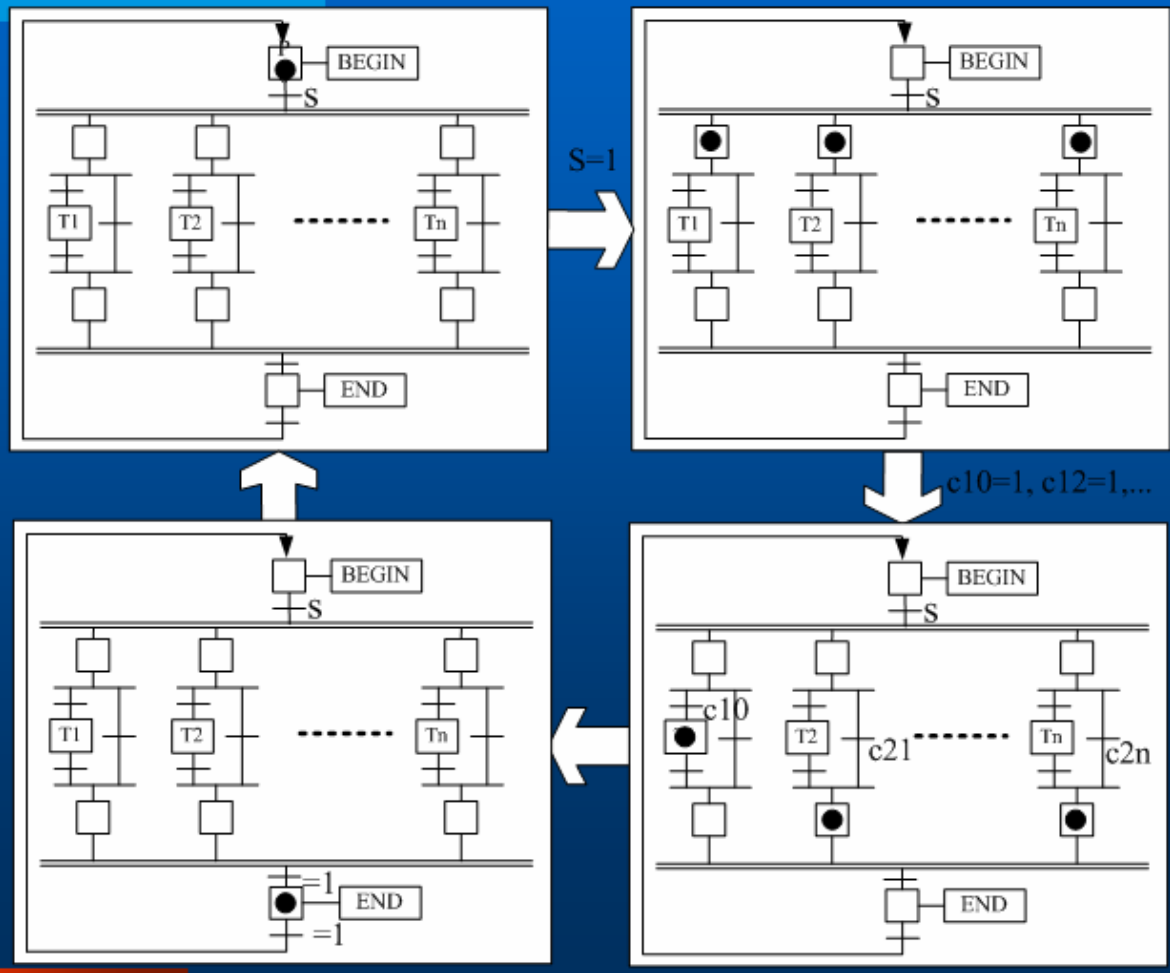

G1
X0
R0
X1
Sub-Grafcet G11
R1 =X12
X2
R2
X3
Sub-Grafcet G12
R3 =X22
X4
R4
G11
X10
R11 =X1
X11
.....
R12
X12
R13 = X1
G12
X30
R31 =X3
X31
.....
R32
X32
R33 = X3

Grafcet離散事件模擬

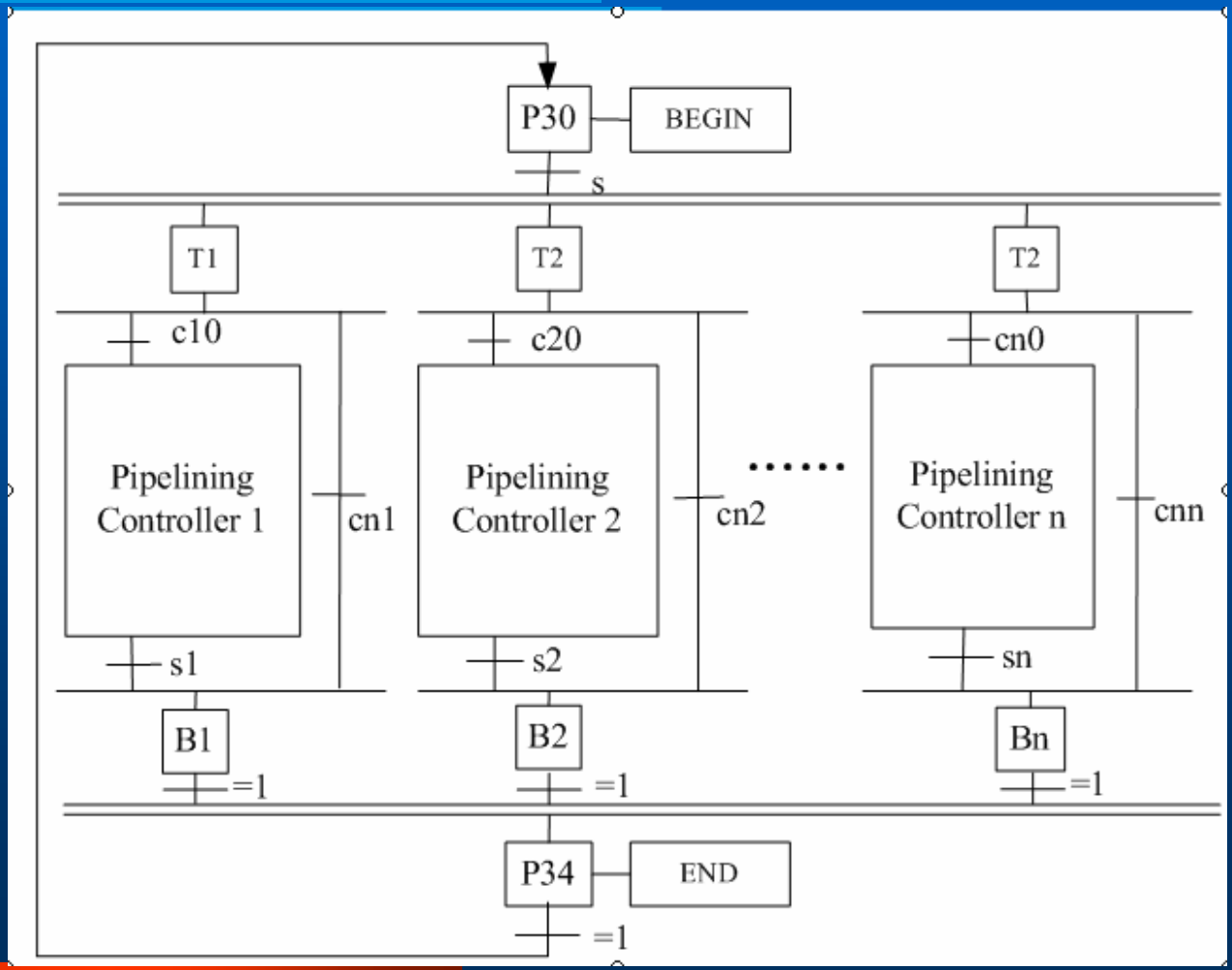


Pipelining離散事件建模

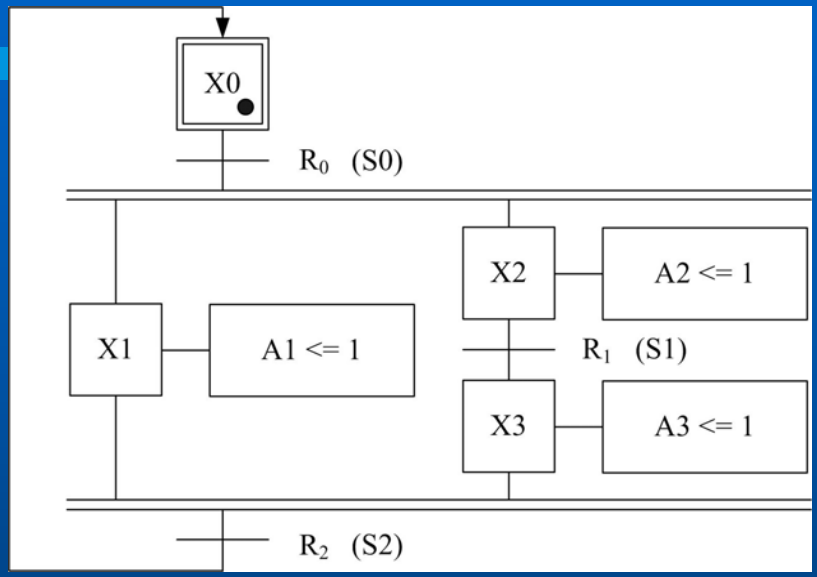


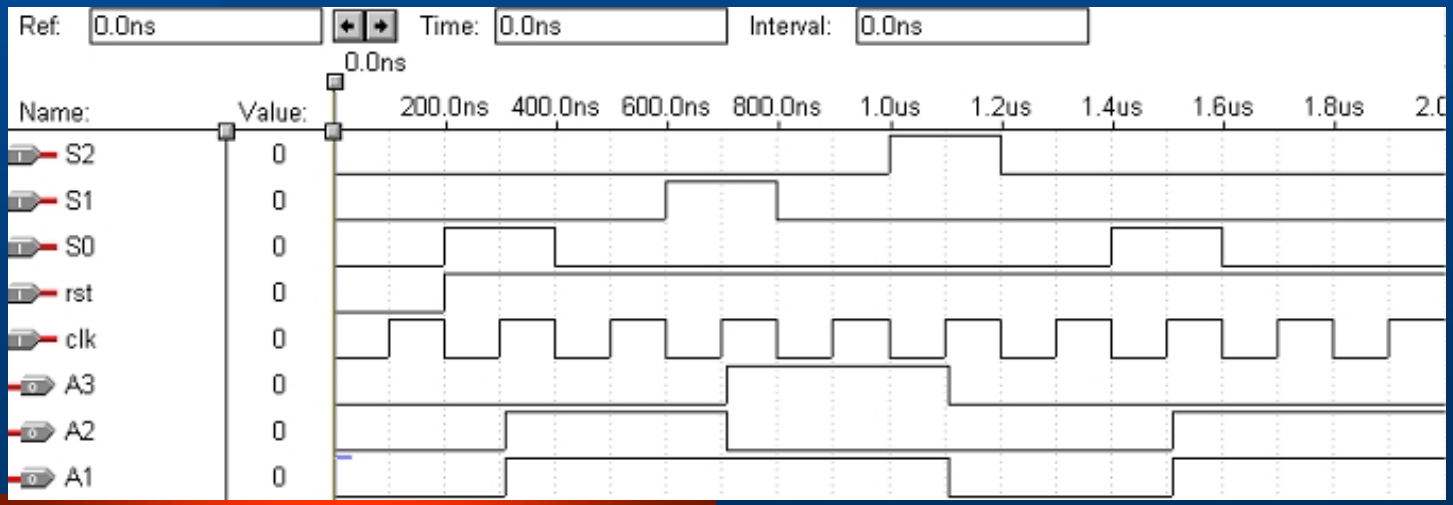


階層式Pipelining

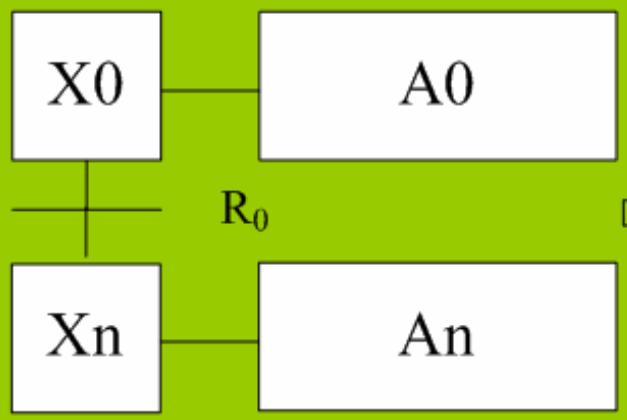


Grafcet離散事件模擬與驗證





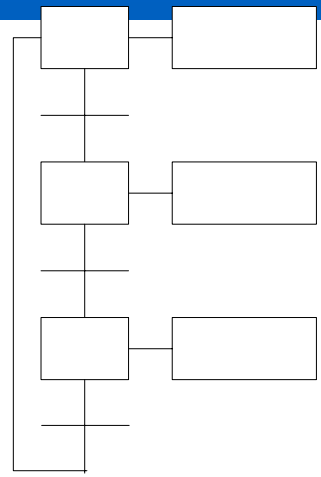
VHDL電路合成

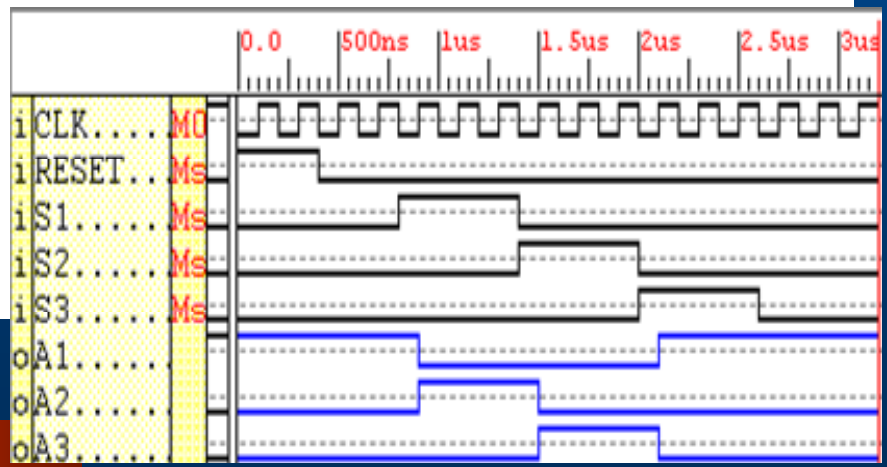



```
if  X0='1' and R0='1' then X0<='0'; Xn<='1';
end if;
A0 <= X0 ;
An <= Xn ;
```

VHDL硬體合成

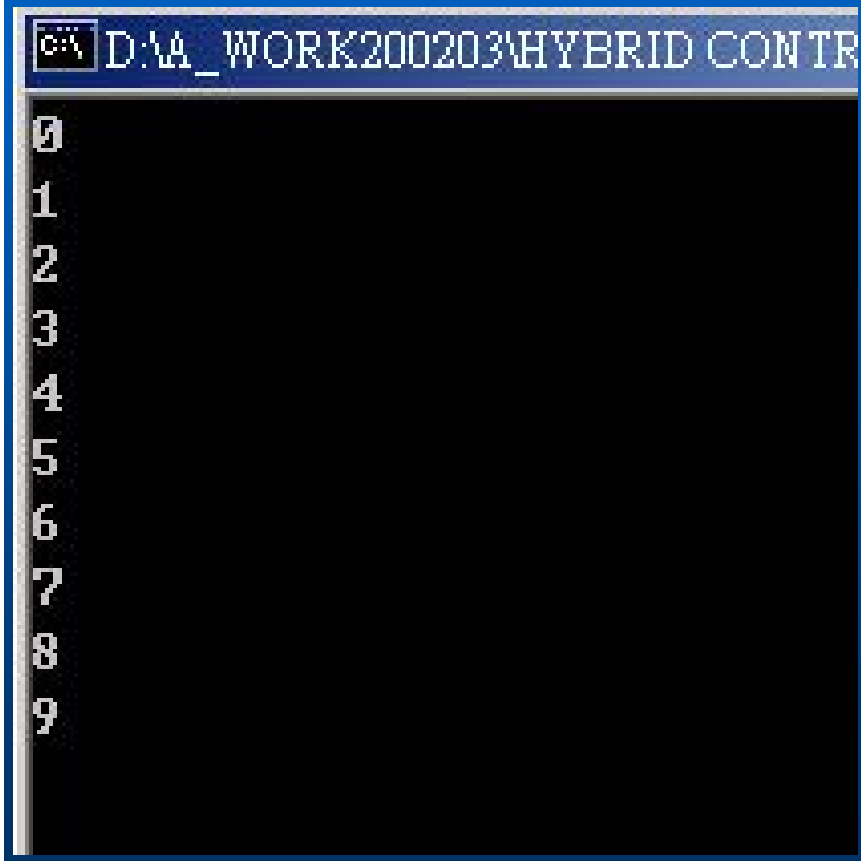
```
architecture DEMO1_arch of DEMO1 is
        signal X1,X2,X3:STD_LOGIC;
begin
    process(CLK)
        begin
                if RESET='1' then

                        X1<='1';
                        X2<='0';
                        x3<='0';

                elsif CLK'EVENT and CLK='1' then

                        if  X1='1' and S1='1'  then   X1<='0';X2<='1'; end if;
                        if  X2='1' and S2='1'  then   X2<='0';X3<='1'; end if;
                        If  X3='1' and S3='1'  then   X3<='0';X1<='1'; end if;

                end if;

                A1<=X1;
                A2<=x2;
                A3<=X3;

        end process;
end DEMO1_arch;
```



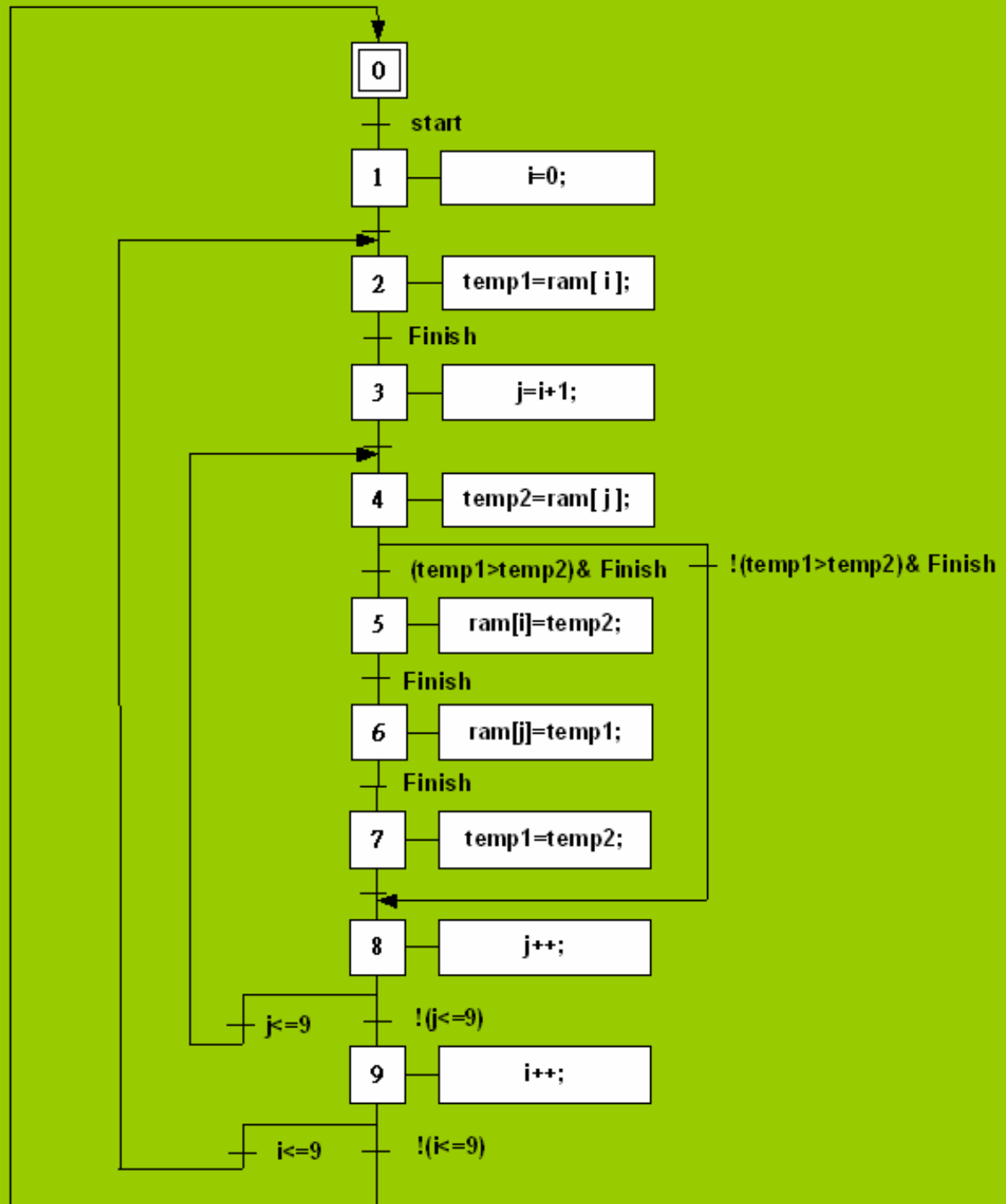


演算法：資料排序

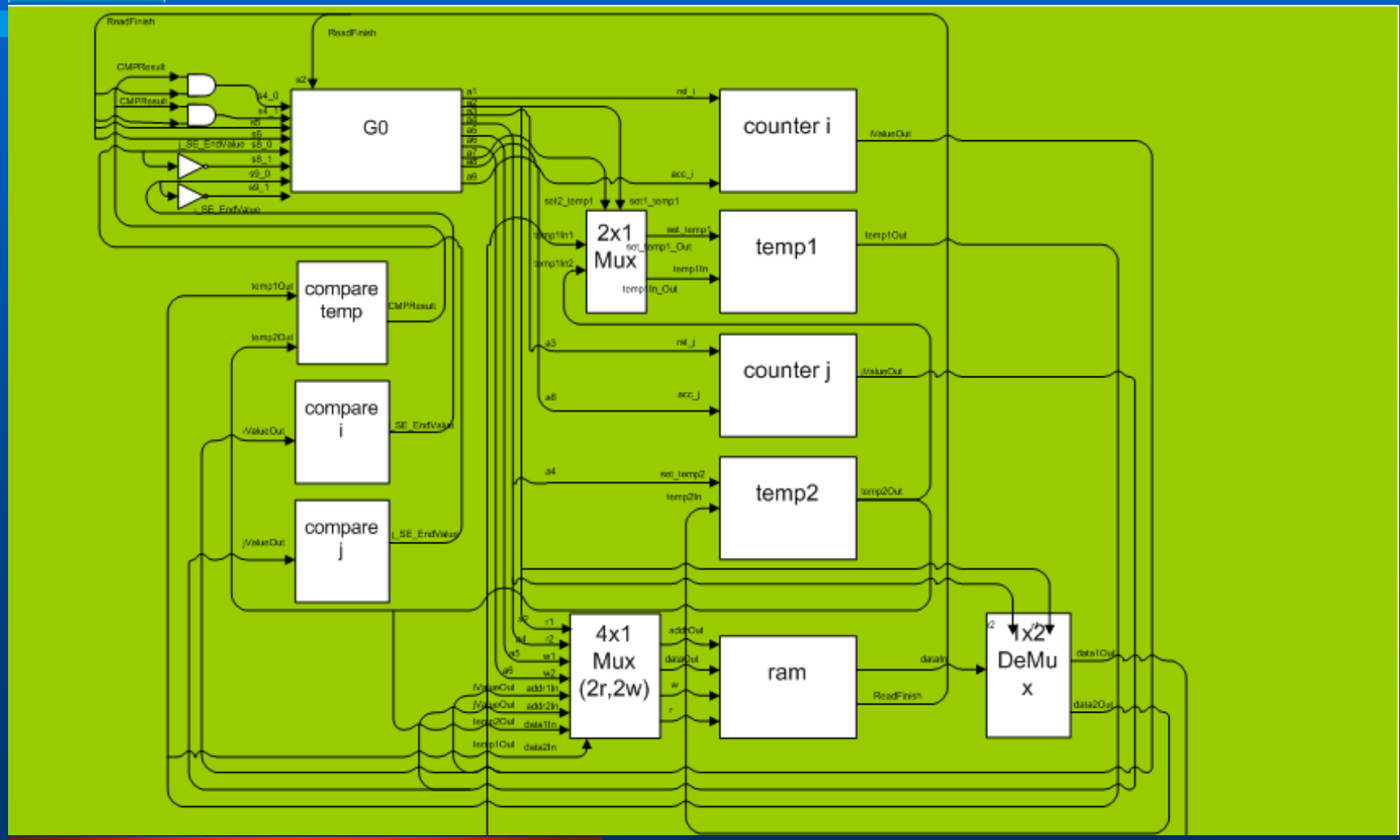
```
unsigned char raru[10]={1,3,5,7,9,2,4,6,8,0};
unsigned char terup1;
unsigned char terup2;

for(int i=0;i<10;i++)
{
    terup1=raru[i];
    for(int j=i+1;j<10;j++)
    {
        terup2=raru[j];
        if(terup1>terup2)
        {
            raru[i]=terup2;
            raru[j]=terup1;
            terup1=terup2;
        }
    }
}
```



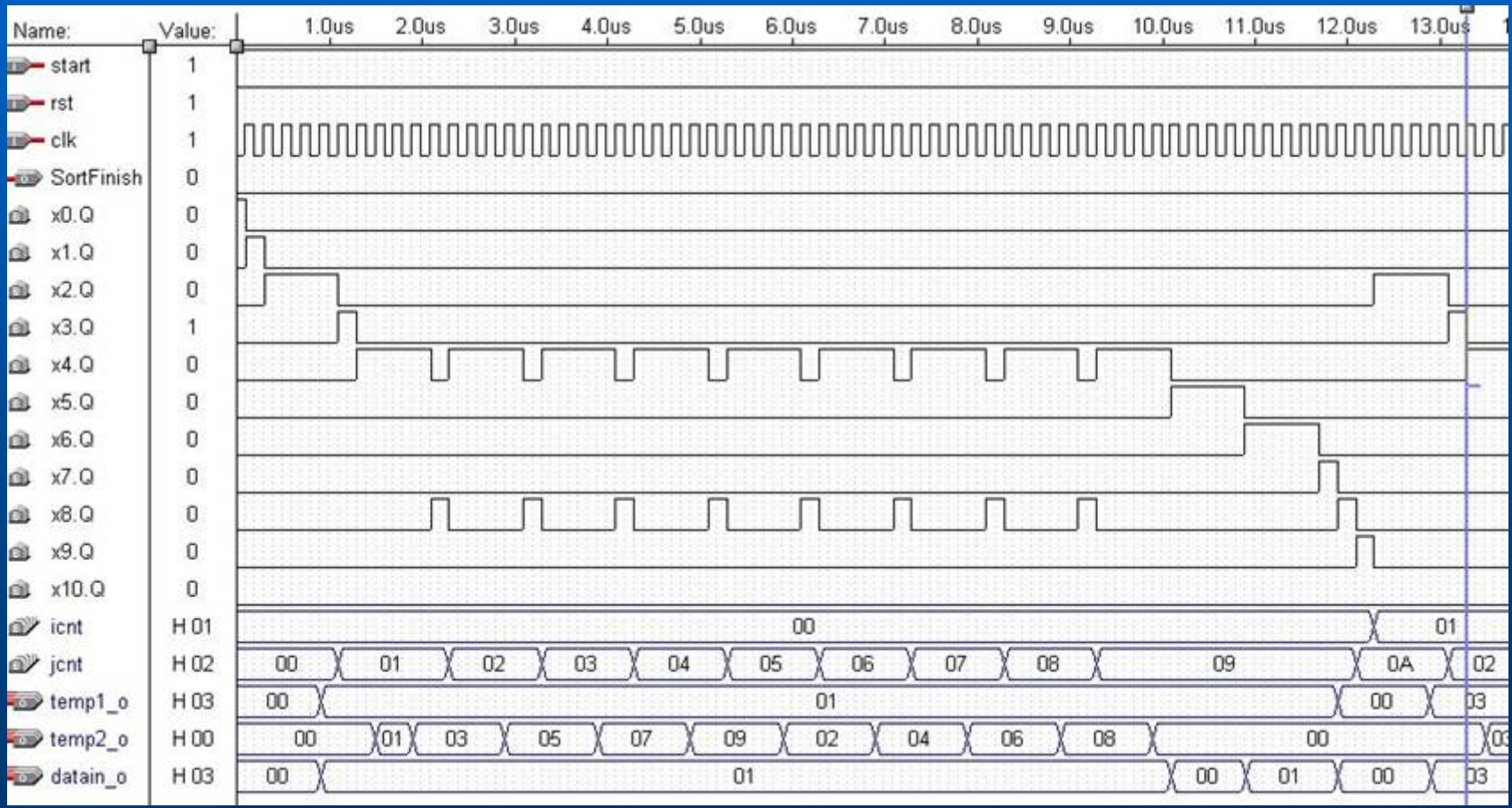
離散事件建模


0
start
1
i=0;
2
temp1=ram[i];
Finish
3
j=i+1;
4
temp2=ram[j];
(temp1>temp2)& Finish
!(temp1>temp2)& Finish
5
ram[i]=temp2;
Finish
6
ram[j]=temp1;
Finish
7
temp1=temp2;
8
j++;
j<=9
!(j<=9)
9
i++;
i<=9
!(i<=9)

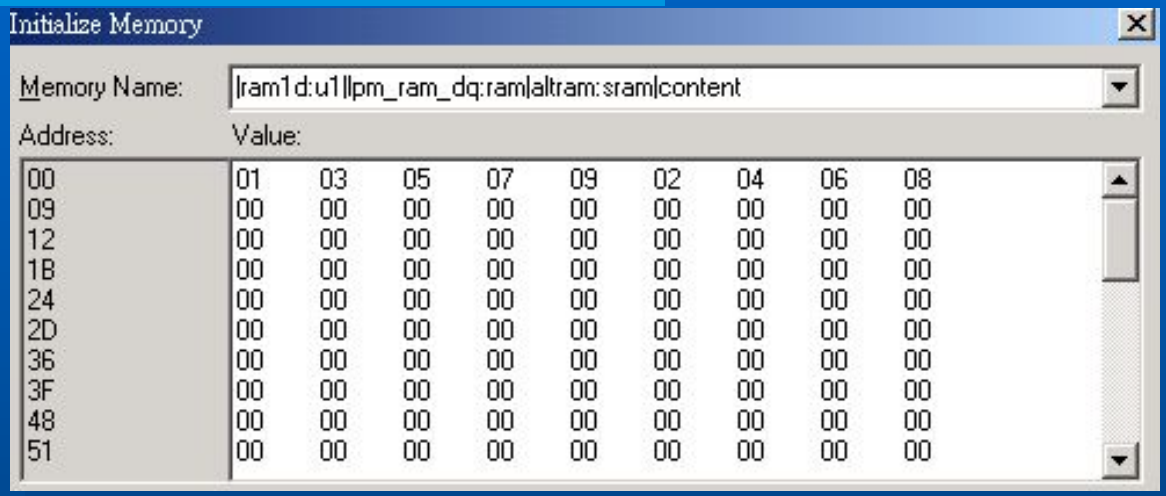
電路合成

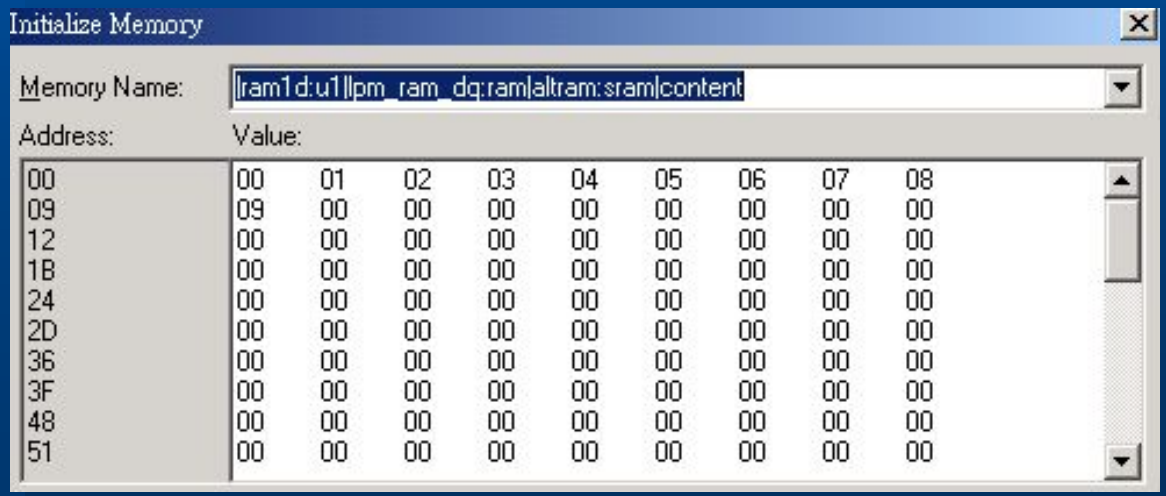

ReadFinish
CMPResult
G0
counter i
2x1
Mux
temp1
compare
temp
compare
i
compare
j
counter j
temp2
4x1
Mux
(2r,2w)
ram
1x2
DeMu
x

功能驗證：時序模擬


Name:
Value:
1.0us
2.0us
3.0us
4.0us
5.0us
6.0us
7.0us
8.0us
9.0us
10.0us
11.0us
12.0us
13.0us
start 1
rst 1
clk 1
SortFinish 0
x0.Q 0
x1.Q 0
x2.Q 0
x3.Q 1
x4.Q 0
x5.Q 0
x6.Q 0
x7.Q 0
x8.Q 0
x9.Q 0
x10.Q 0
icnt H 01
jcnt H 02
temp1_o H 03
temp2_o H 00
datain_o H 03

FPGA硬體驗證


Initialize Memory
Memory Name:
|ram1d:u1|lpm_ram_dq:ram|altram:sram|content
Address:
Value:
00 01 03 05 07 09 02 04 06 08
09 00 00 00 00 00 00 00 00 00
12 00 00 00 00 00 00 00 00 00
1B 00 00 00 00 00 00 00 00 00
24 00 00 00 00 00 00 00 00 00
2D 00 00 00 00 00 00 00 00 00
36 00 00 00 00 00 00 00 00 00
3F 00 00 00 00 00 00 00 00 00
48 00 00 00 00 00 00 00 00 00
51 00 00 00 00 00 00 00 00 00


Initialize Memory
Memory Name:
|ram1d:u1|lpm_ram_dq:ram|altram:sram|content
Address:
Value:
00 00 01 02 03 04 05 06 07 08
09 09 00 00 00 00 00 00 00 00
12 00 00 00 00 00 00 00 00 00
1B 00 00 00 00 00 00 00 00 00
24 00 00 00 00 00 00 00 00 00
2D 00 00 00 00 00 00 00 00 00
36 00 00 00 00 00 00 00 00 00
3F 00 00 00 00 00 00 00 00 00
48 00 00 00 00 00 00 00 00 00
51 00 00 00 00 00 00 00 00 00

方法論工作流程

1.C++演算法驗證

C++程式碼, 驗證資料檔，離散版本演算法

2.IDEFO階層式模組架構圖設計

參考”指紋晶片系統階層架構圖.vsd”

3.GRAFCET離散事件模型建立

參考”Grafcet G2 Point Maching.vsd”

VHDL電路合成

4.硬體驗證

時序模擬波形, 電路方塊圖, 效能/使用資源檔

5.文件撰寫

彙整上述資料在單一文件檔

方法論工作流程

1.C++演算法驗證

C++程式碼, 驗證資料檔，離散版本演算法

2.IDEFO階層式模組架構圖設計

參考”指紋晶片系統階層架構圖.vsd”

3.GRAFCET離散事件模型建立

參考”Grafcet G2 Point Maching.vsd”

VHDL電路合成

4.硬體驗證

時序模擬波形, 電路方塊圖, 效能/使用資源檔

5.命名規範

6.文件撰寫

彙整上述資料在單一文件檔

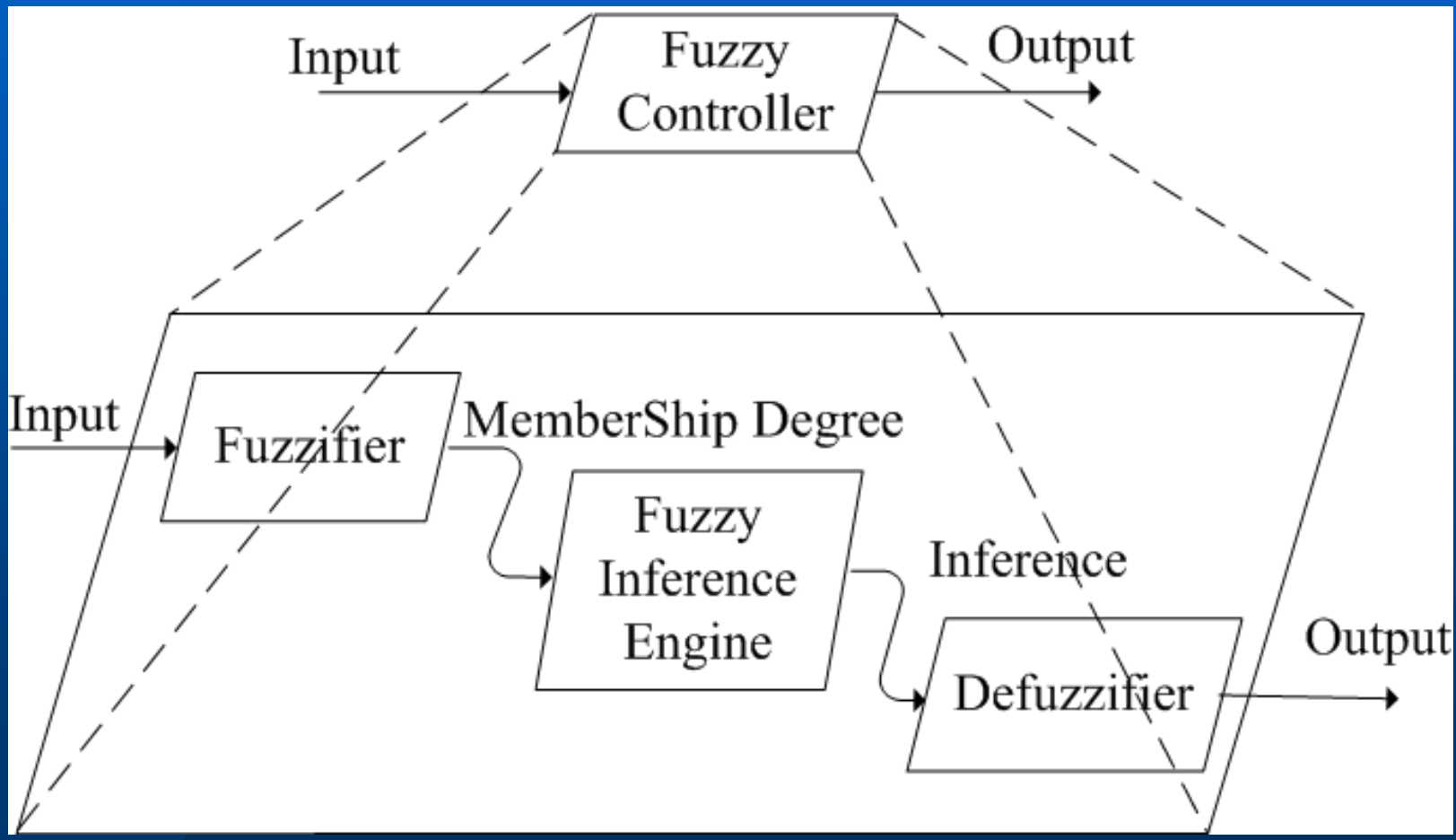
PART 3

方法論的實務和案例

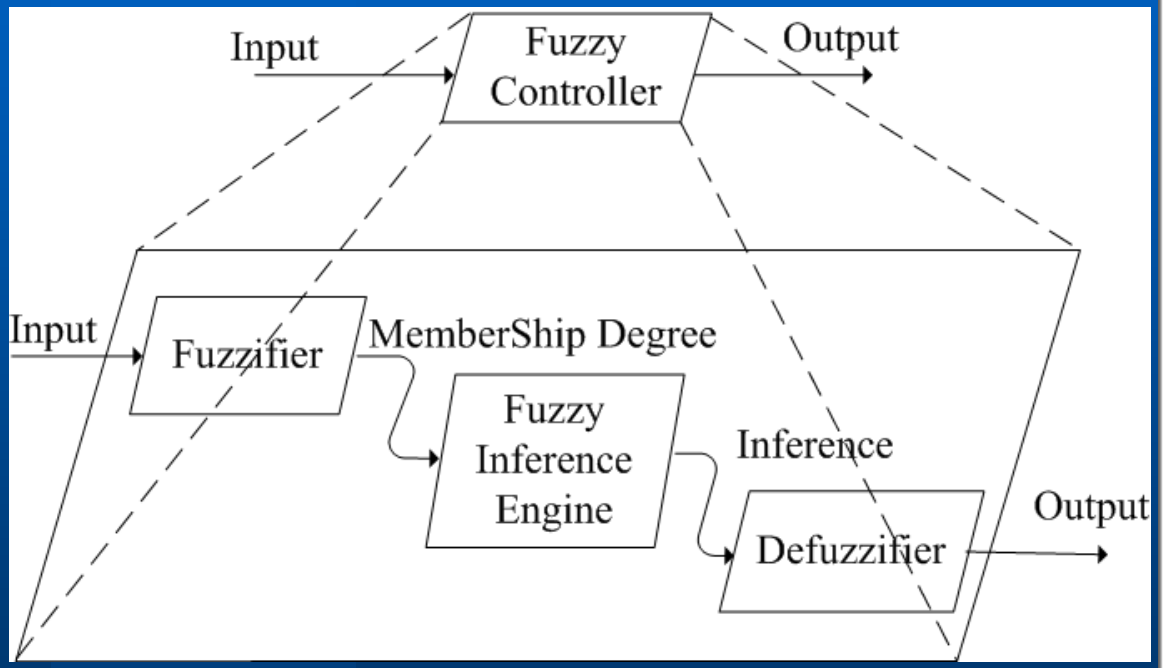
系統設計案例

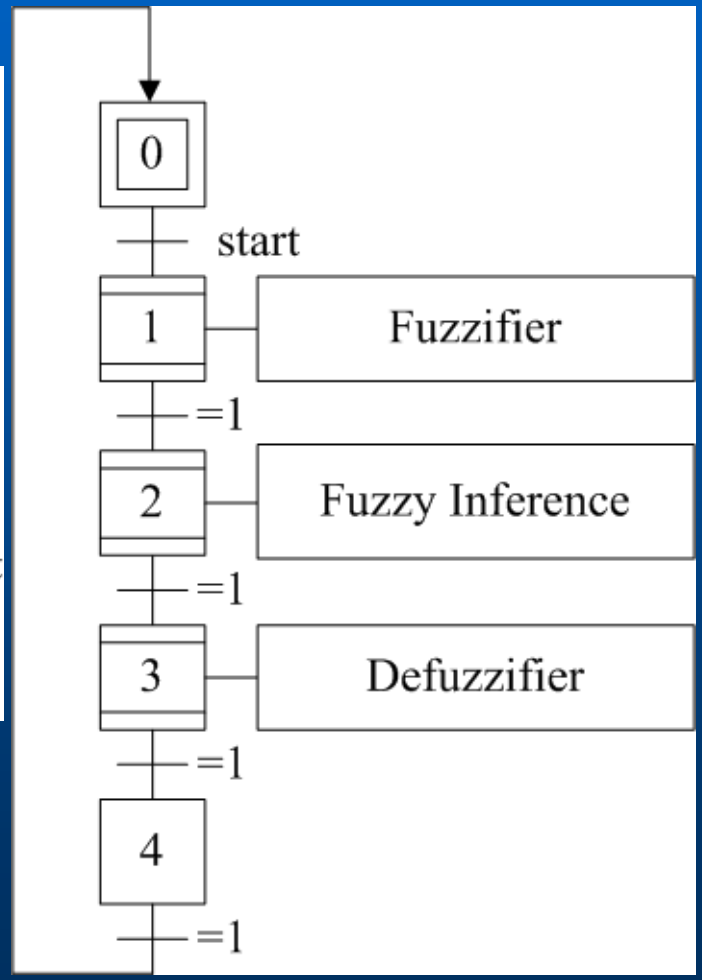
- Fuzzy Controller
- JPEG2000 Codec
- Biometric Authentication Processor

Fuzzy Controller模組架構

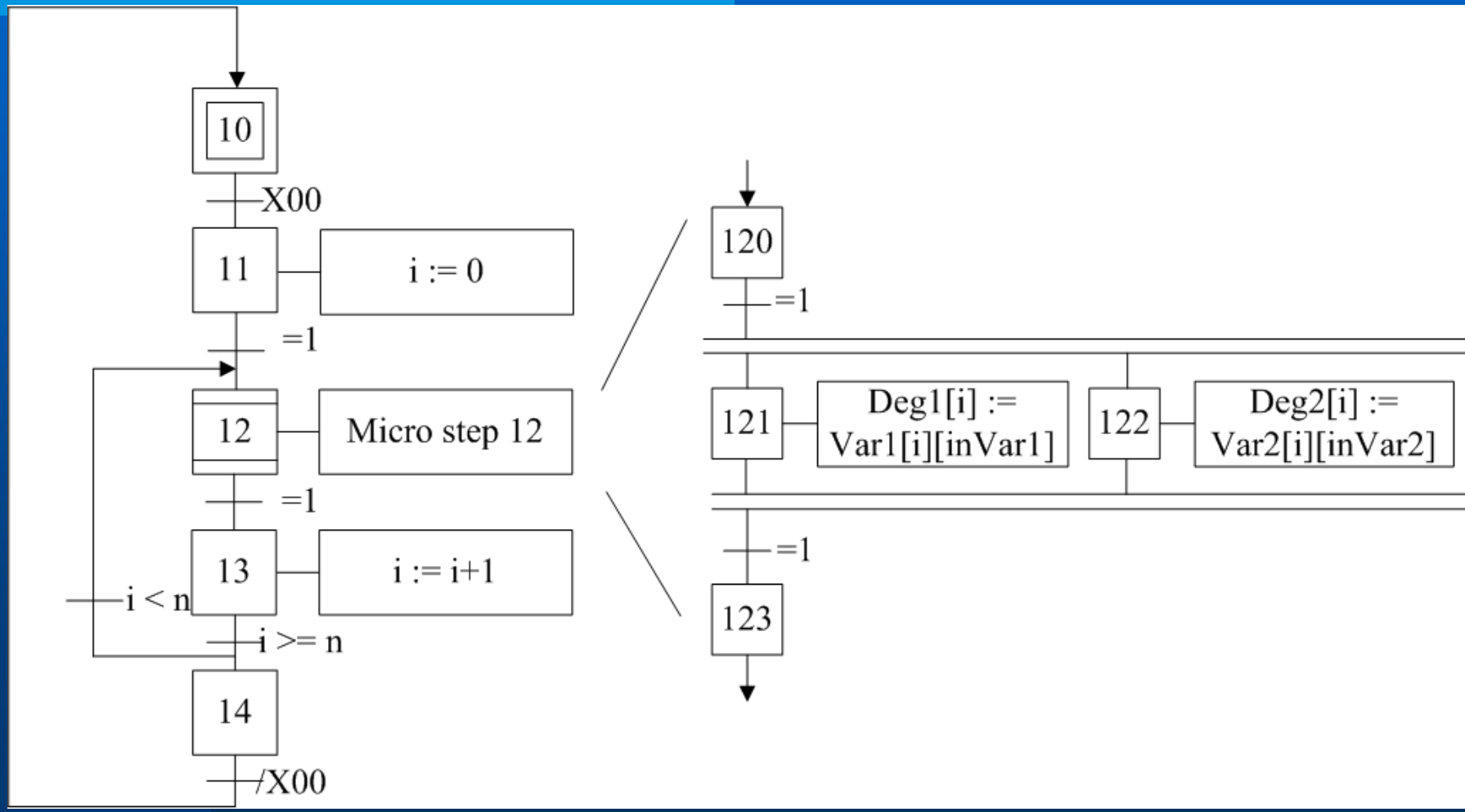


Fuzzy Controller的離散事件模擬

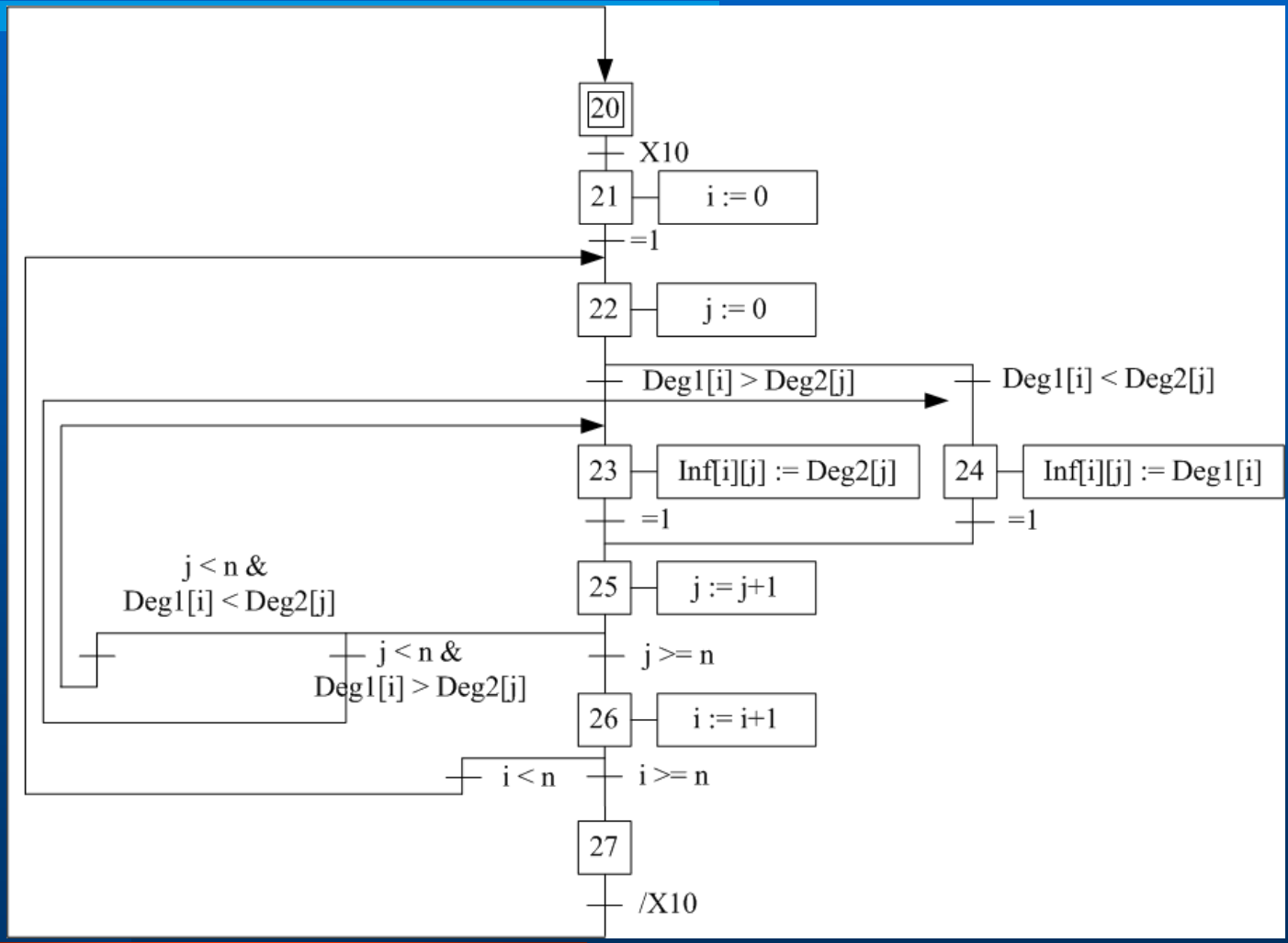




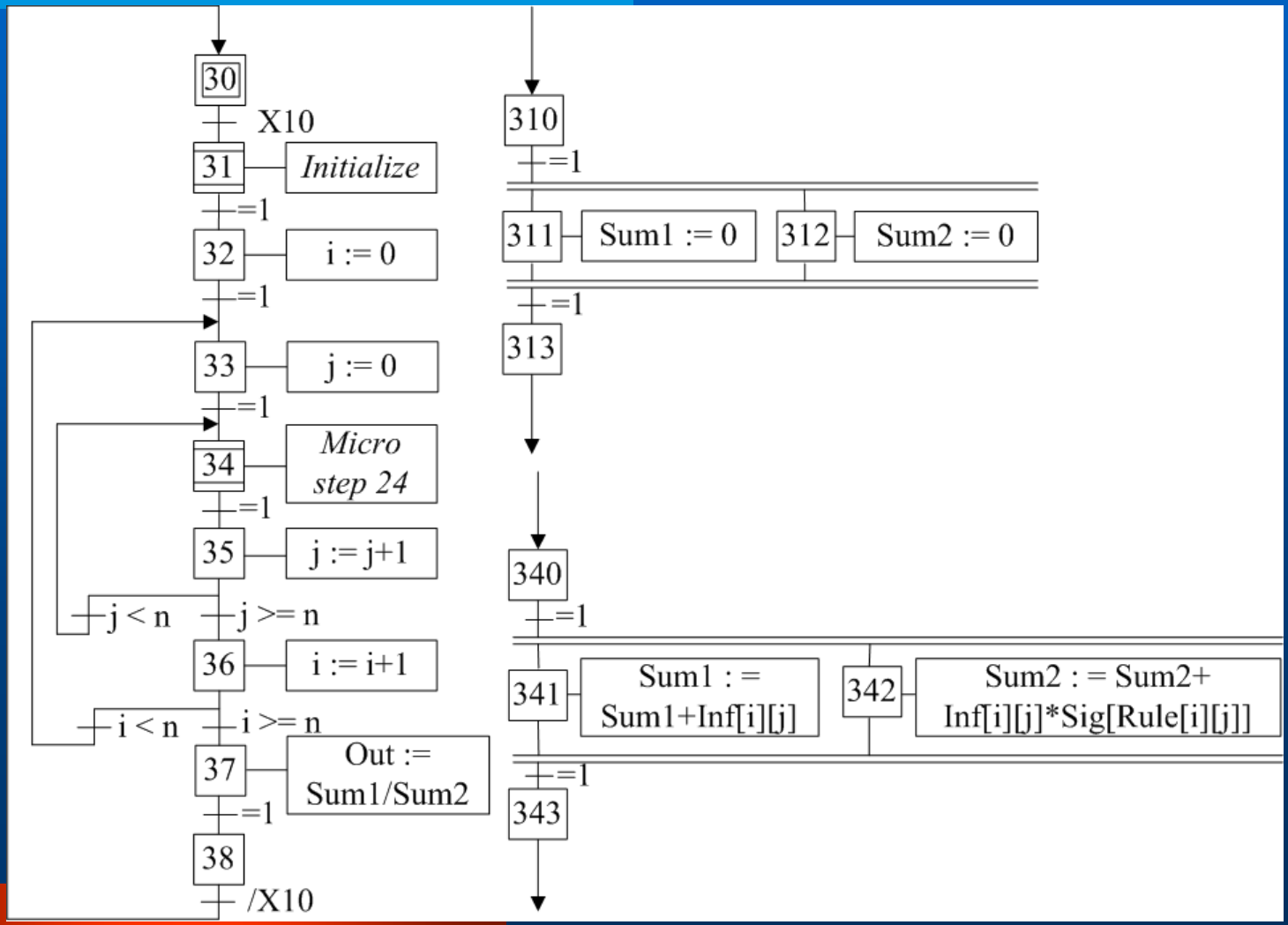
Fuzzifier的離散事件模擬



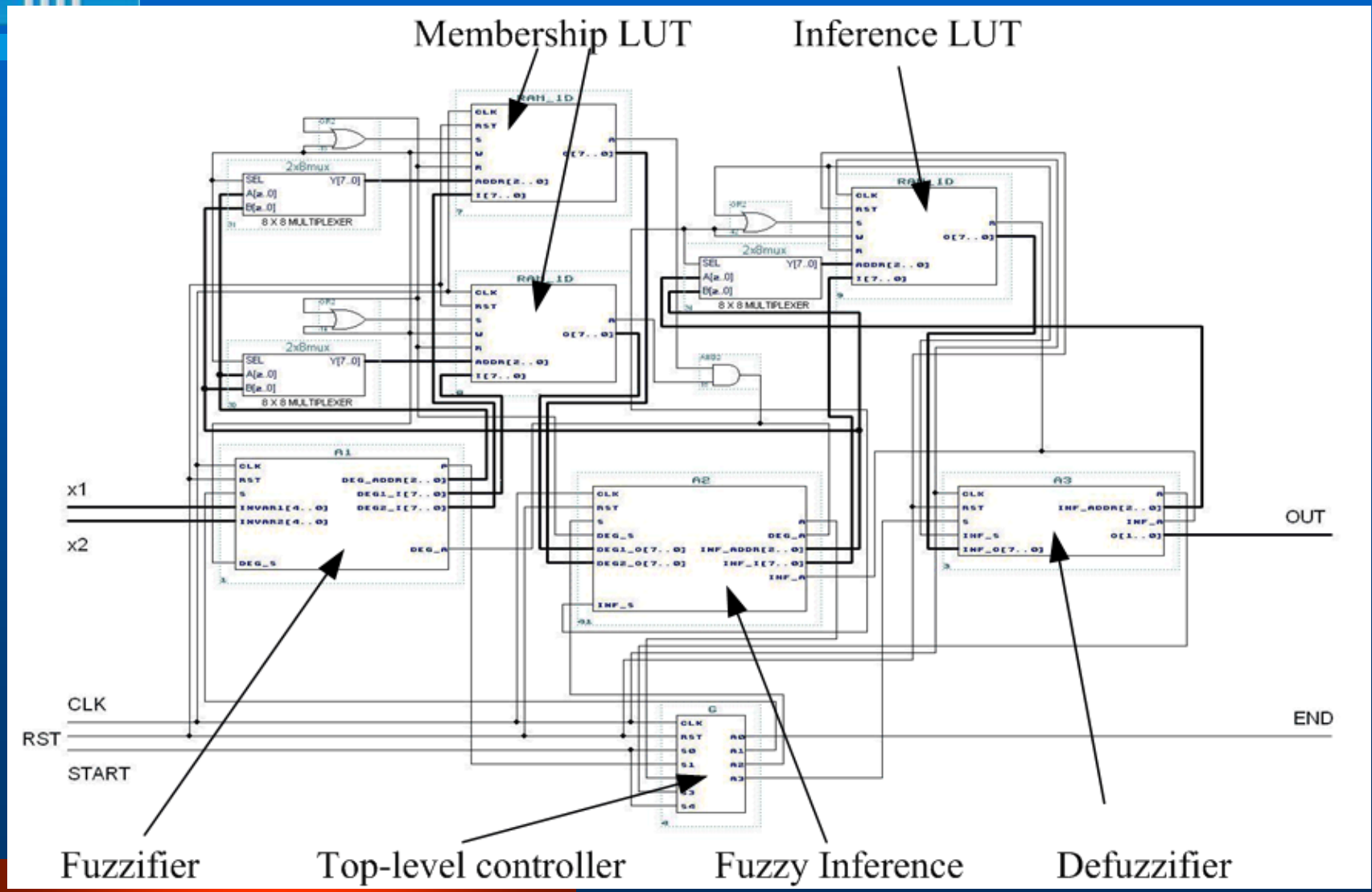
Fuzzy Inference模組
離散事件建模



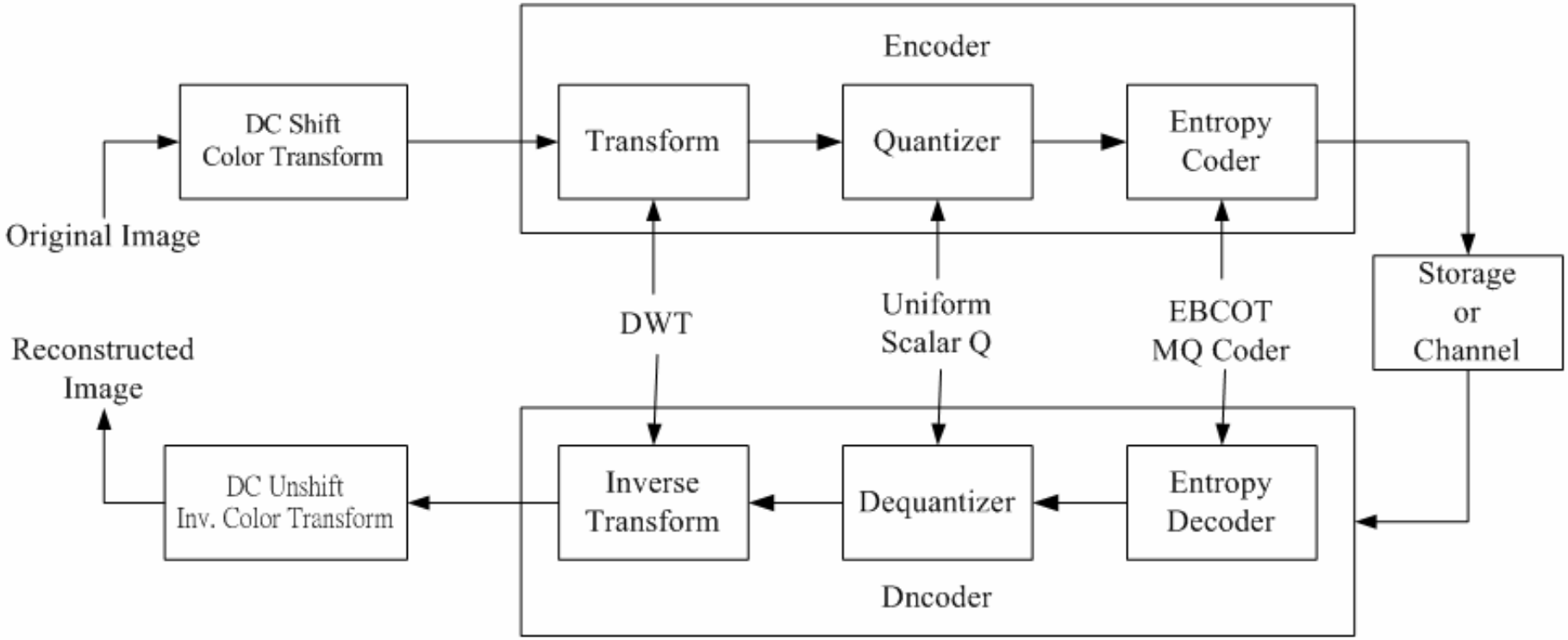
Defuzzifier模組離散事件建模



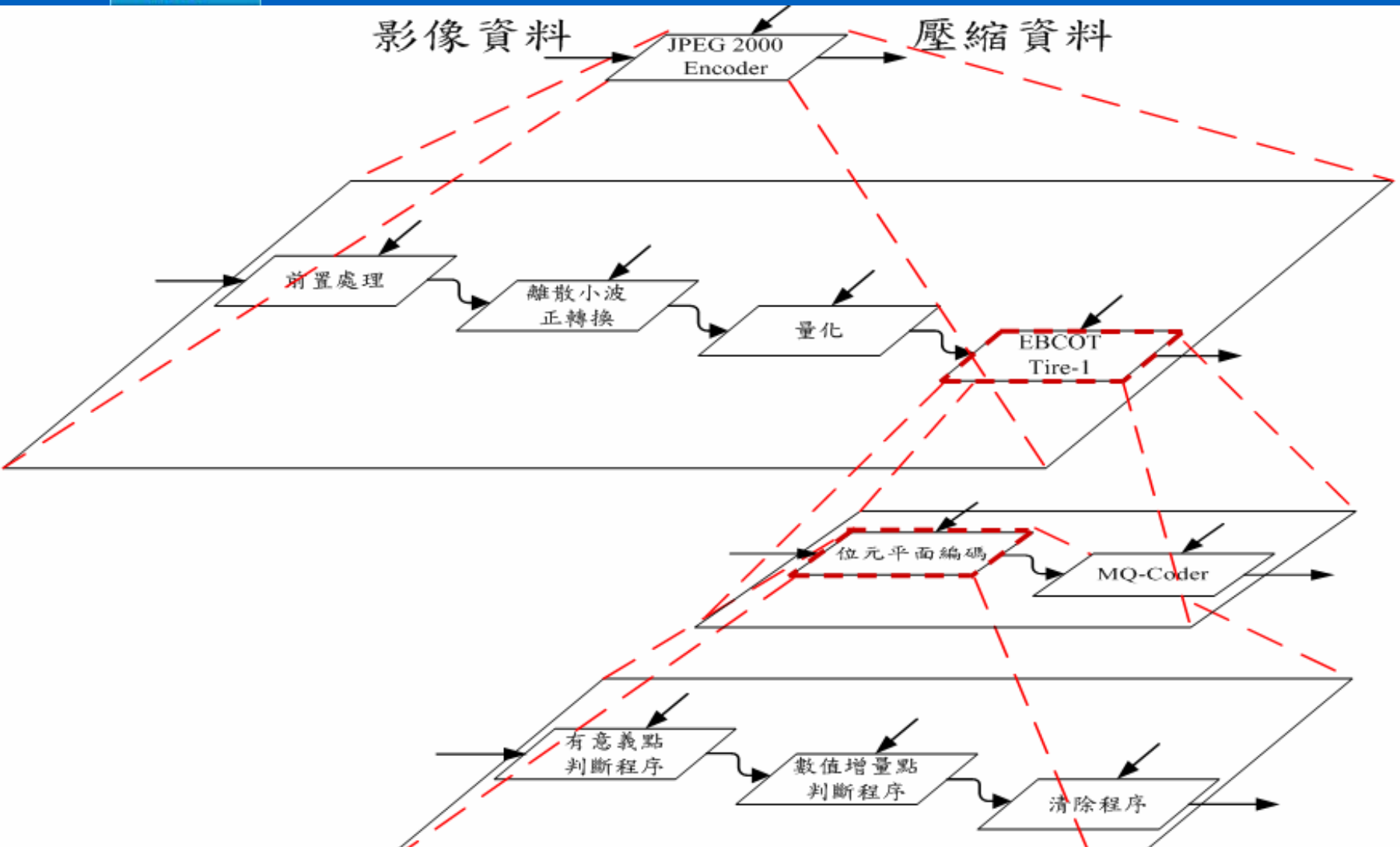
Fuzzy Controller電路合成



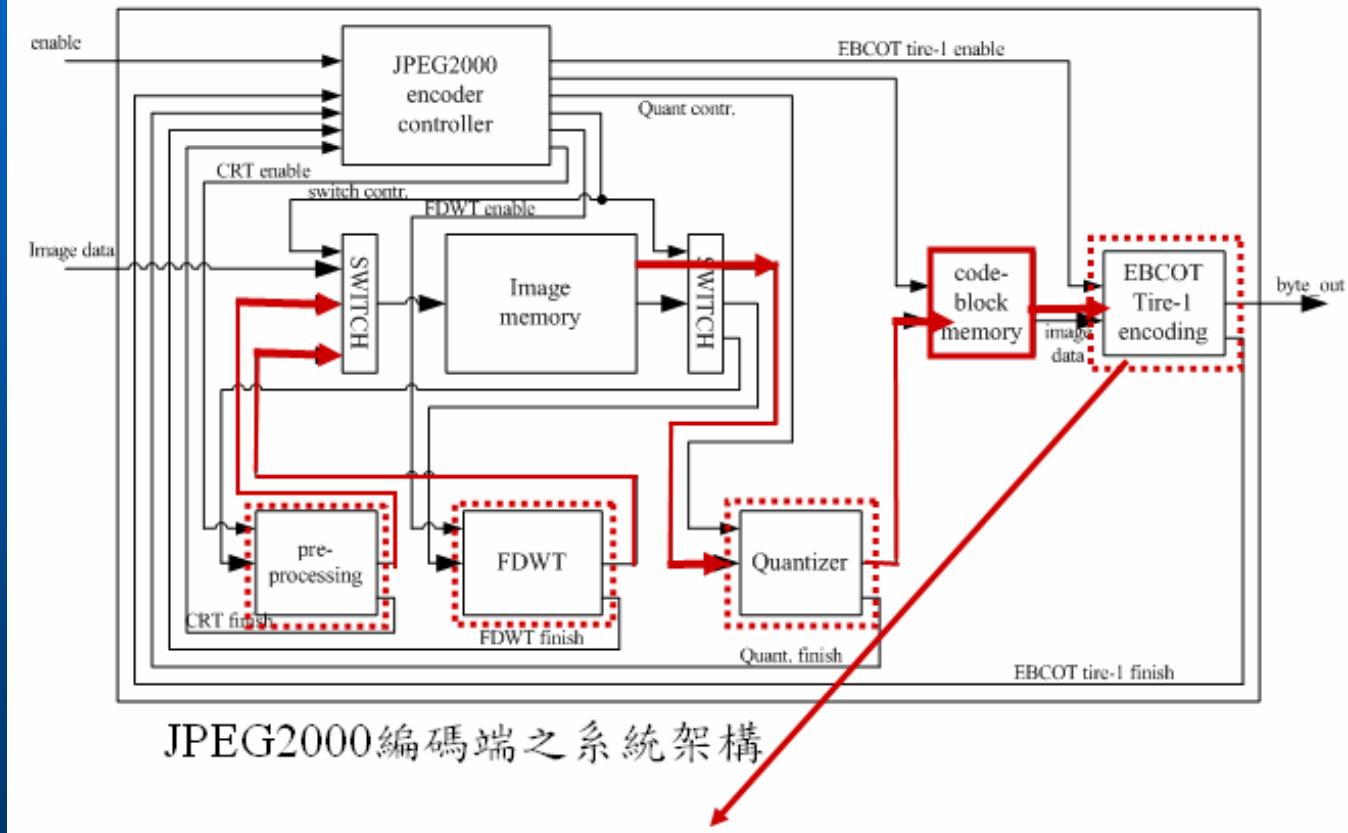
JPEG2000 Codec



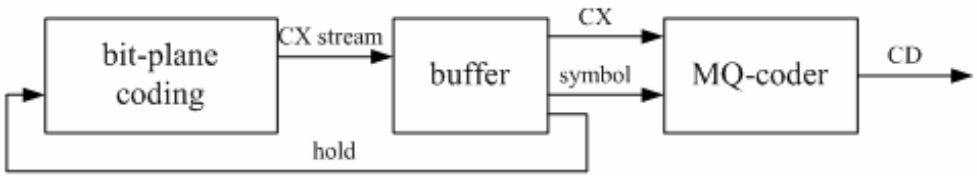
JPEG2000 – IDEF0架構



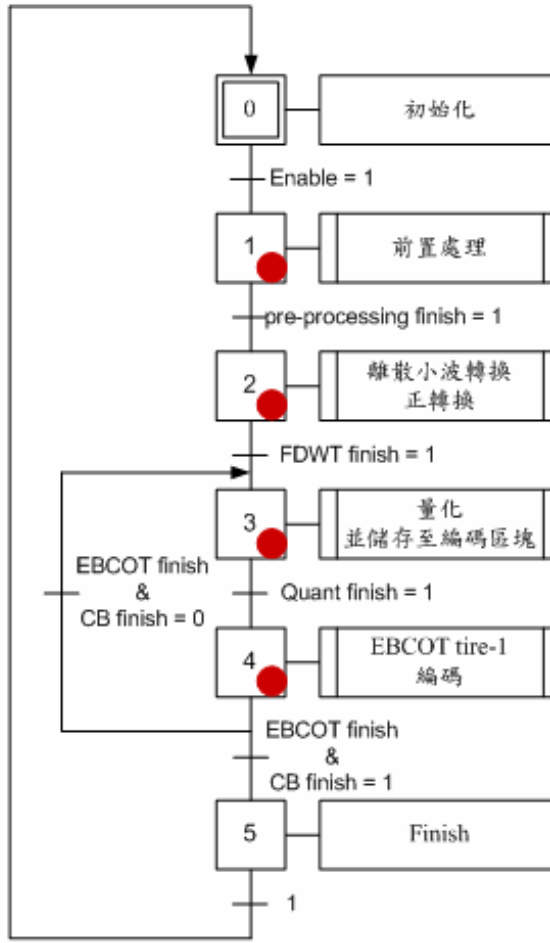
JPEG2000 – GRAFCET模型



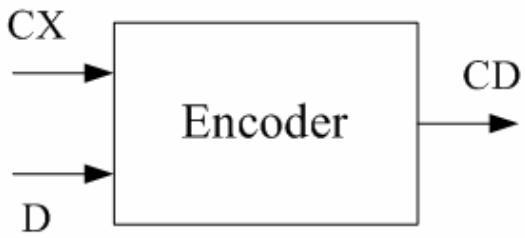
JPEG2000編碼端之系統架構

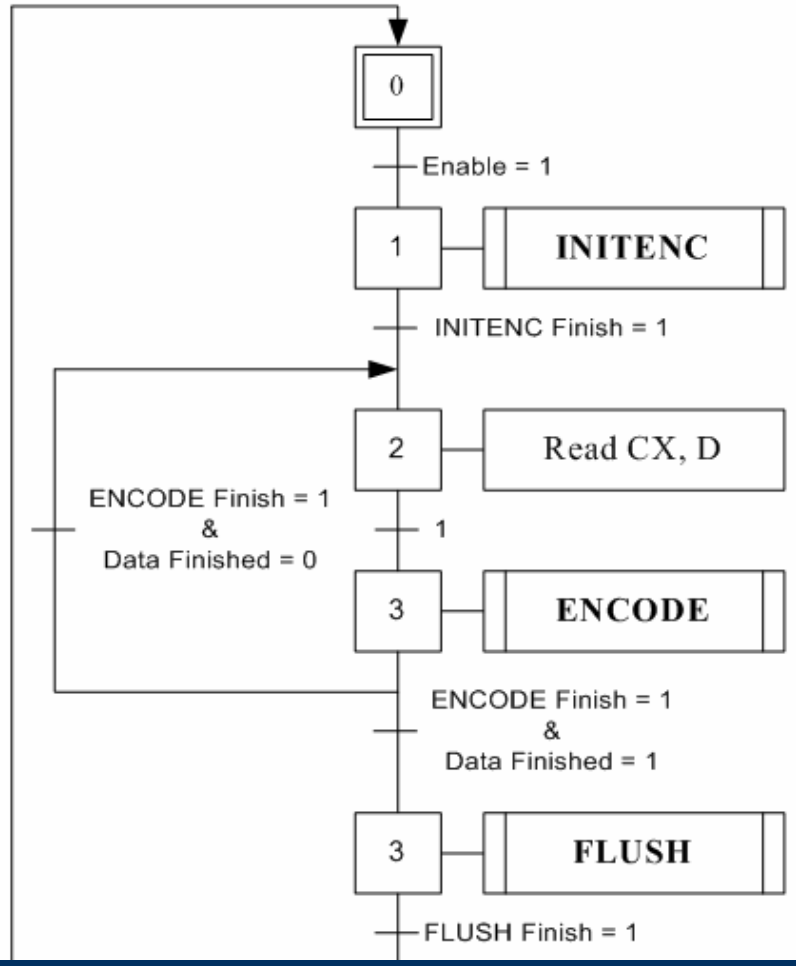


EBCOT tire-1編碼端架構圖

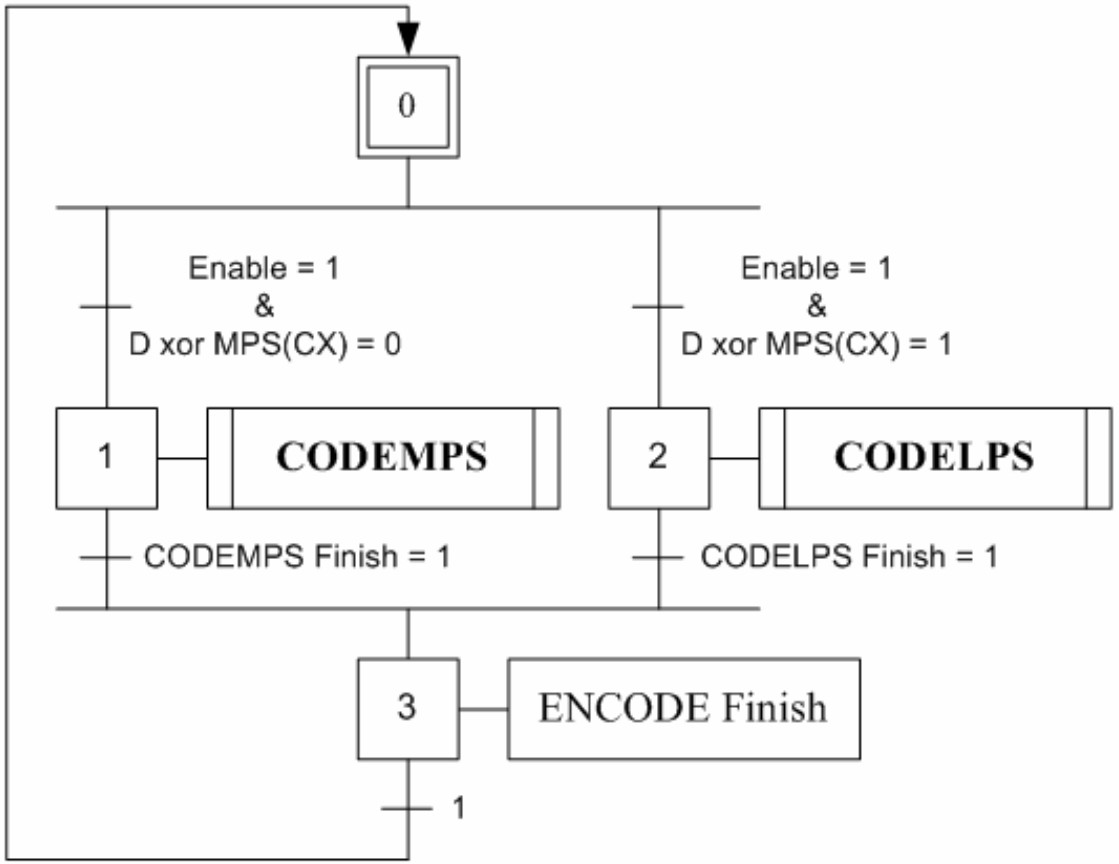


MQ-coder

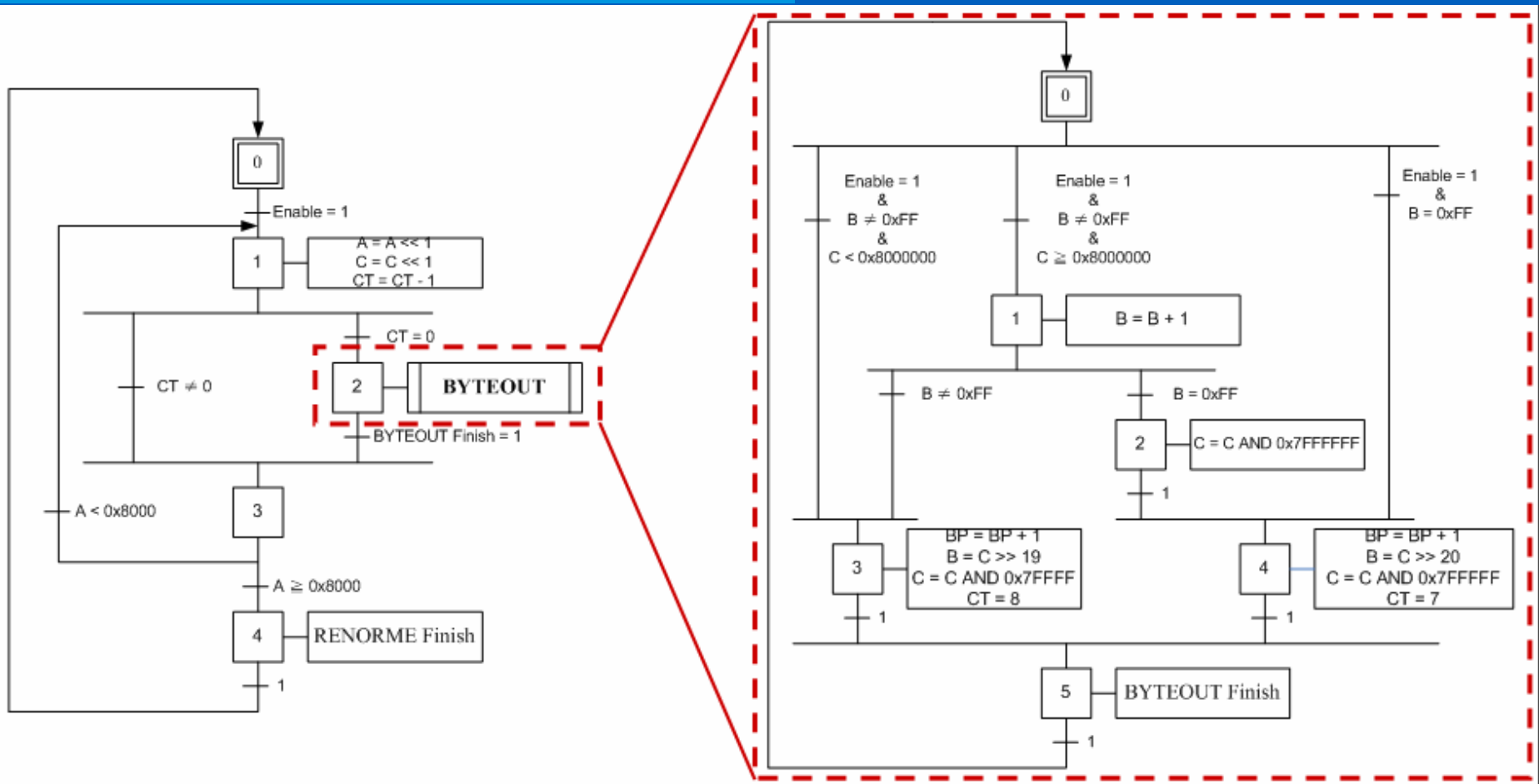

CX
D
Encoder
CD


0
Enable = 1
1
INITENC
INITENC Finish = 1
2
Read CX, D
ENCODE Finish = 1
&
Data Finished = 0
1
3
ENCODE
ENCODE Finish = 1
&
Data Finished = 1
3
FLUSH
FLUSH Finish = 1

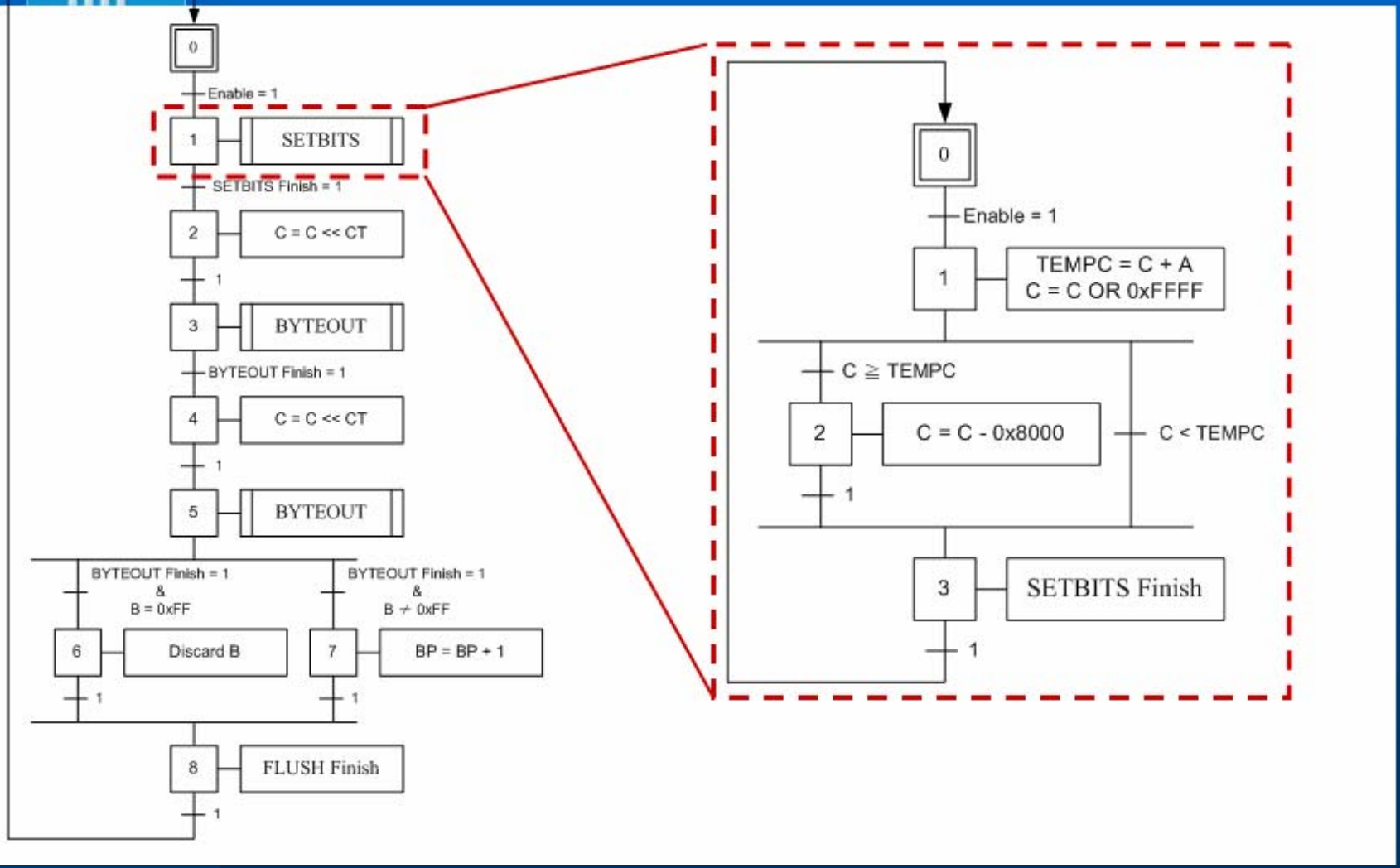
MQ-coder/ENCODE模組



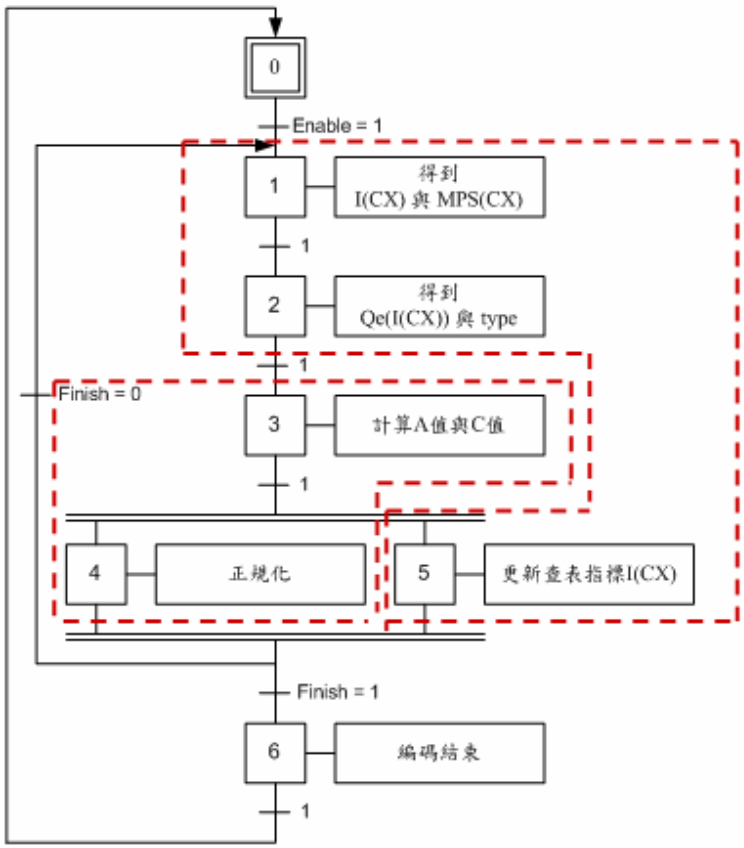
MQ-coder/RENORME模組

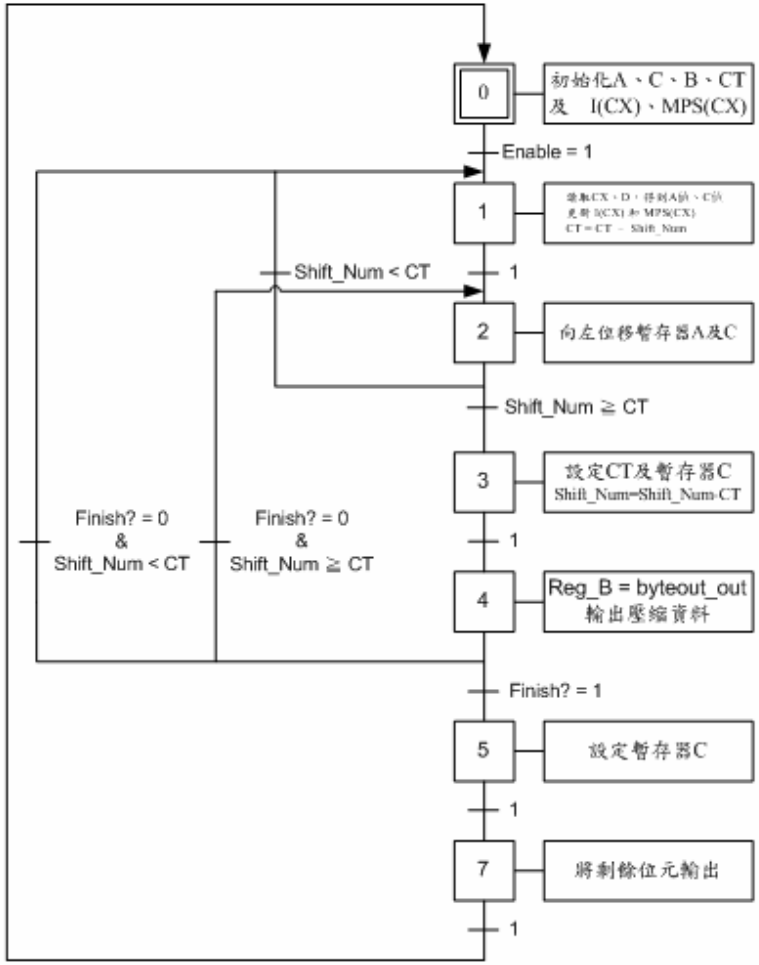


MQ-coder/FLUSH模組

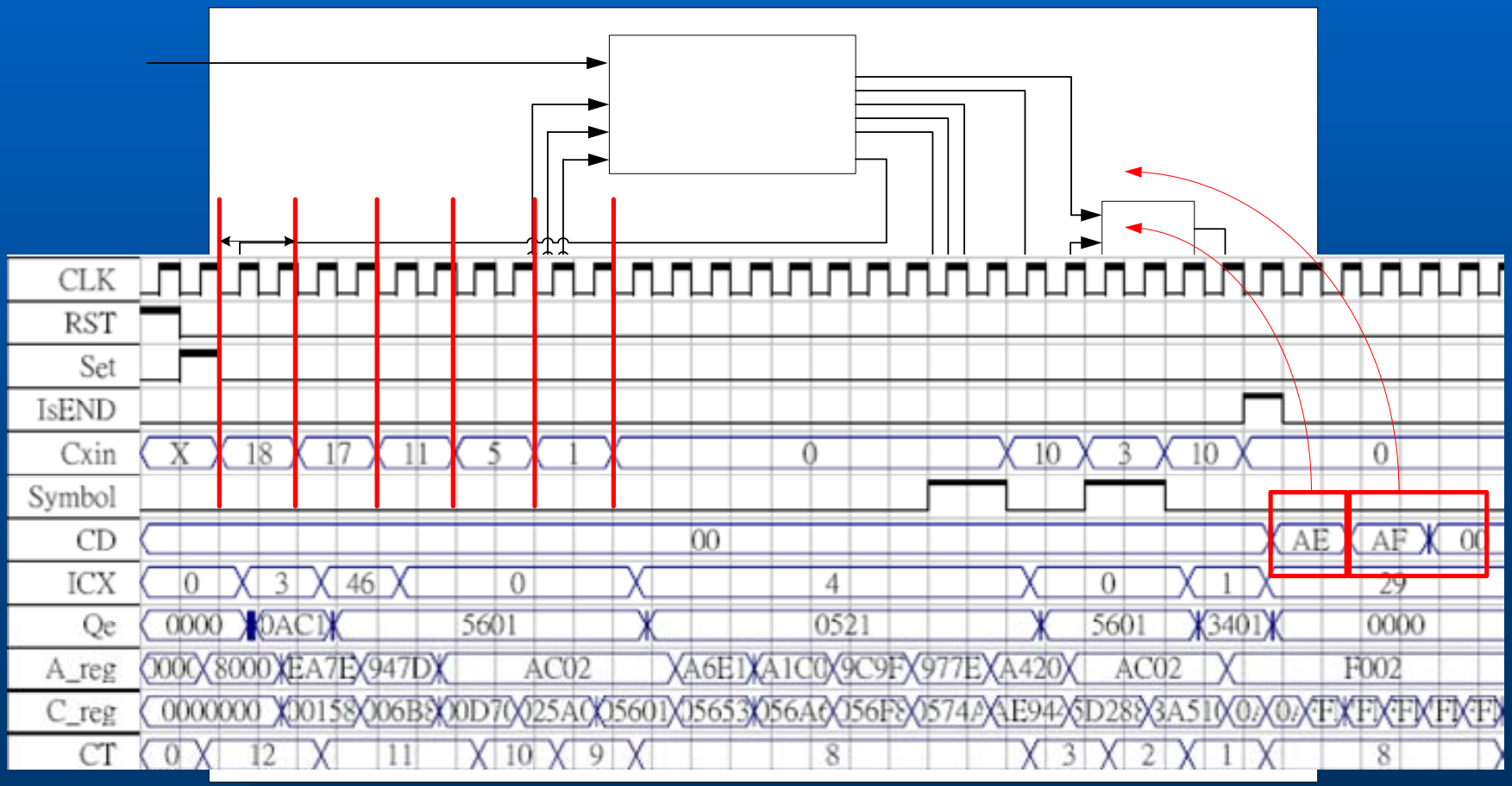


MQ-coder GRAFCET modeling





MQ-coder時序驗證

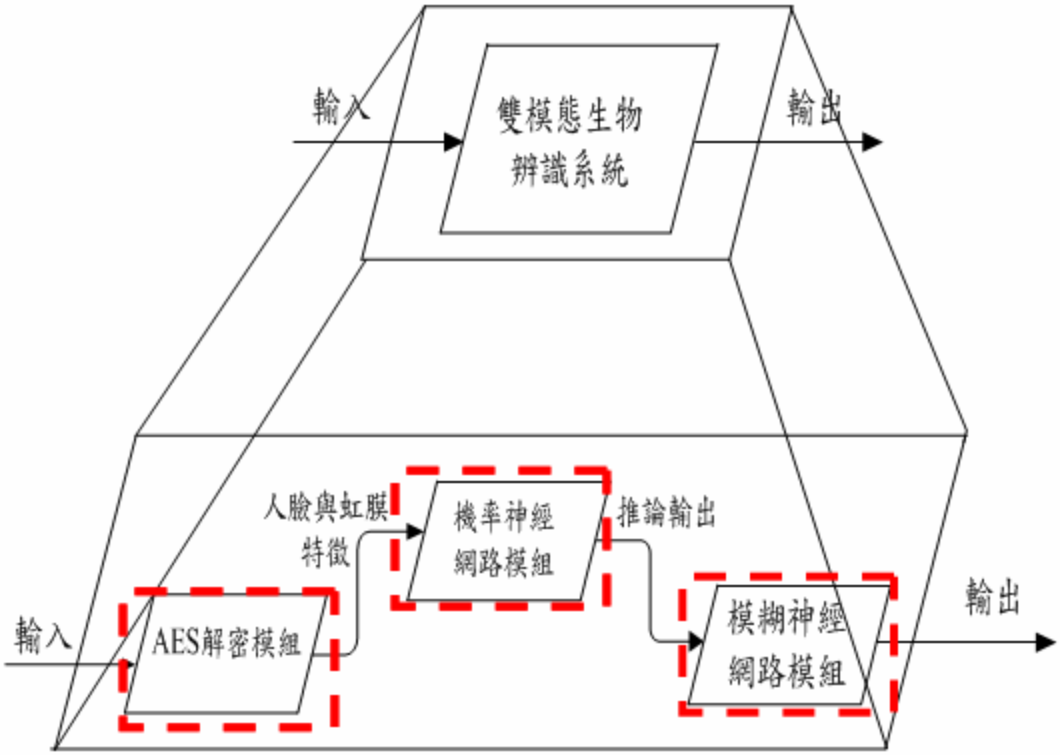


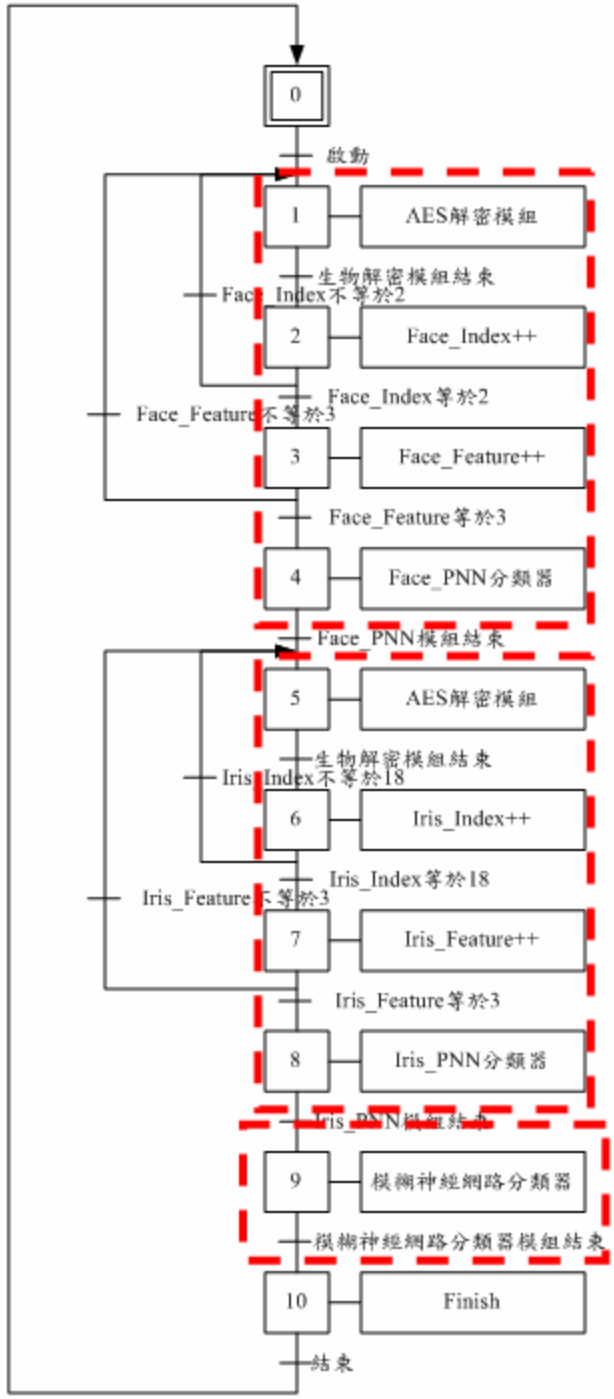
JPEG2000-FPGA驗證

	編碼		解碼	
Test images	Performance (cycles)	Execution time (ms)	Performance (cycles)	Execution time (ms)
Lena	1275901	51.036	1191047	47.642
Baboo	1307470	52.299	1222115	48.884
Peppers	1300757	52.030	1215508	48.620
Boats	1286449	51.218	1198677	47.947
Goldhill	1273402	50.936	1188588	47.543

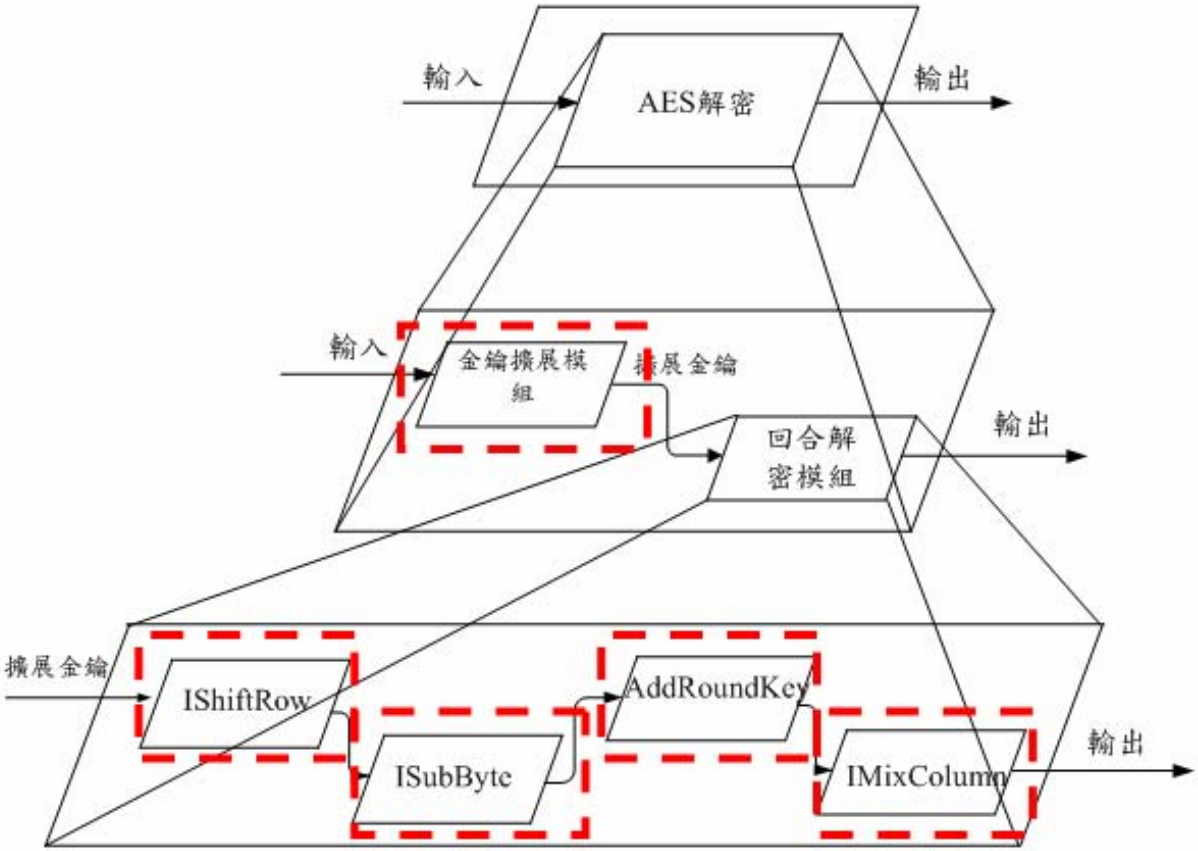
硬體資源(Altera Cyclone EP1C12Q240C8 FPGA)		
	編碼子系統	解碼子系統
Total Logic Elements	3, 539/ 12, 060	3, 006/ 12, 060
Memory	10.883 Kbytes	
Max Frequency	28.57 MHz	25 MHz

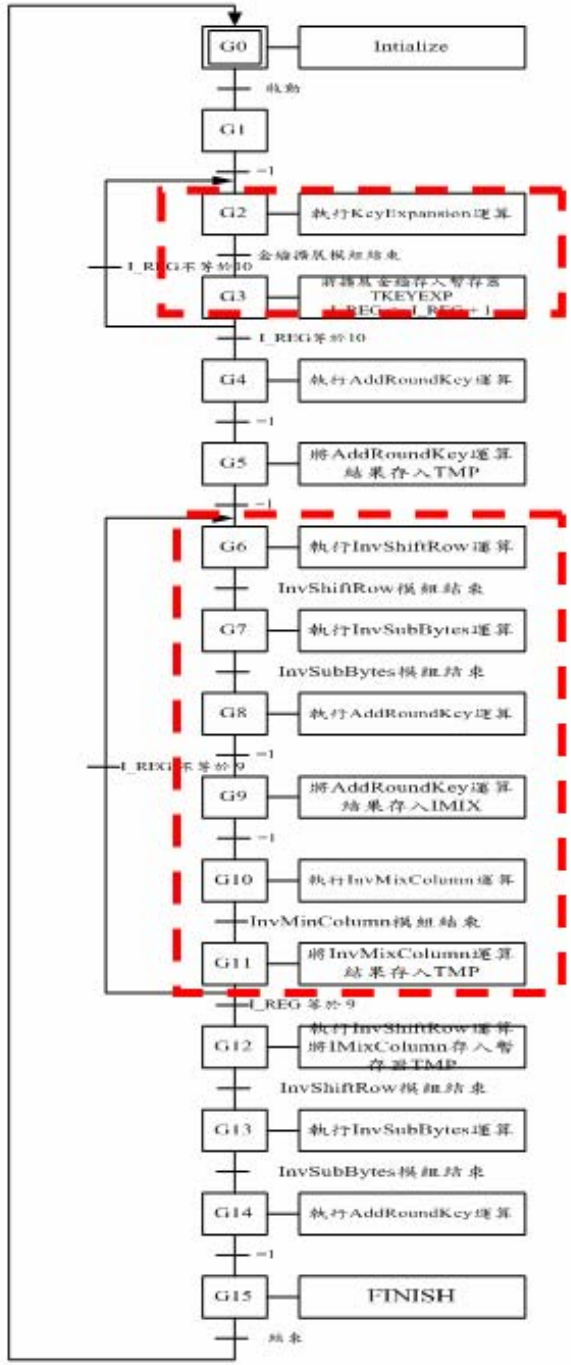
Biometric Authentication Processor(BAP)



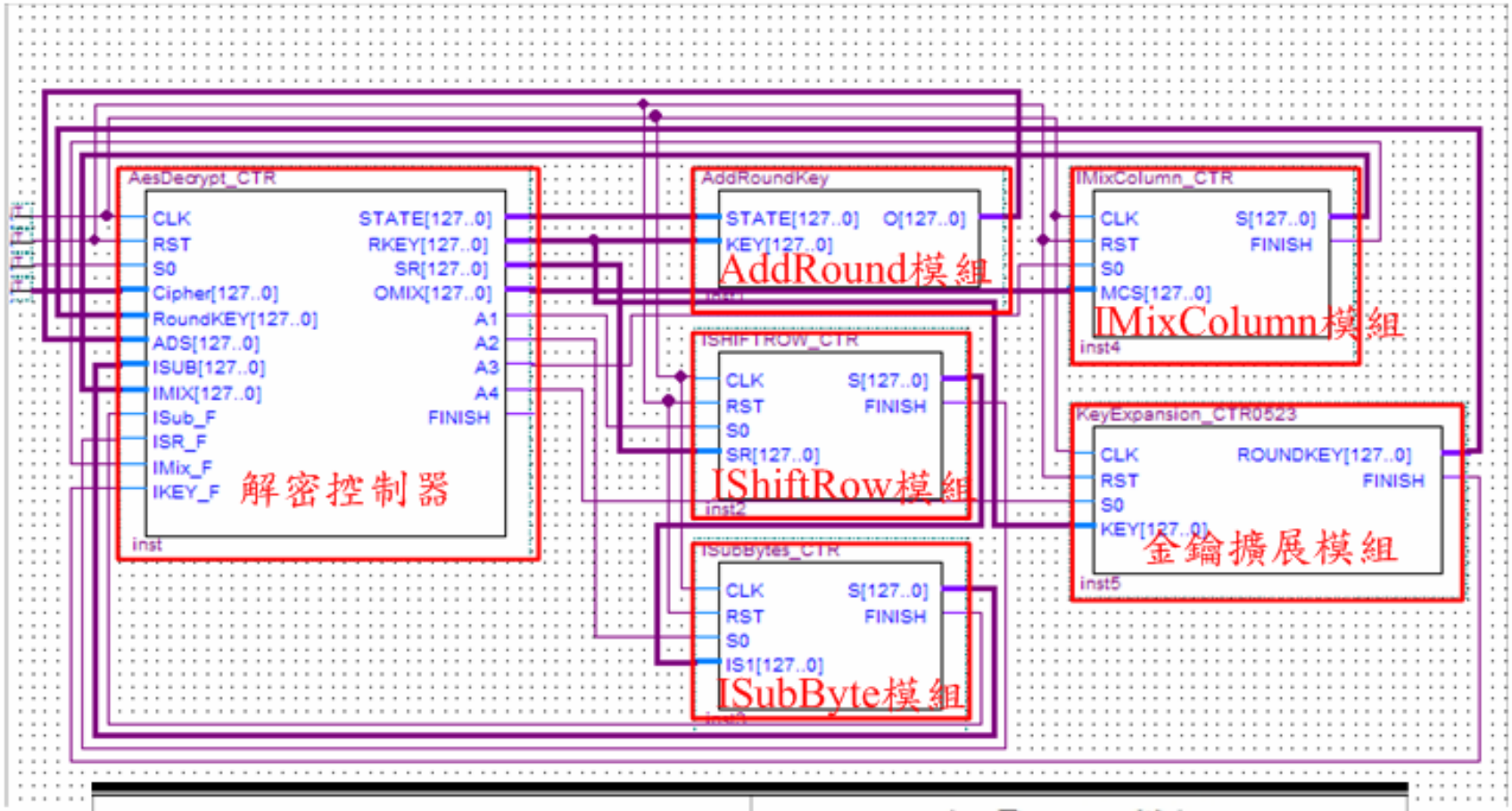


BAP/AES特徵解密模組



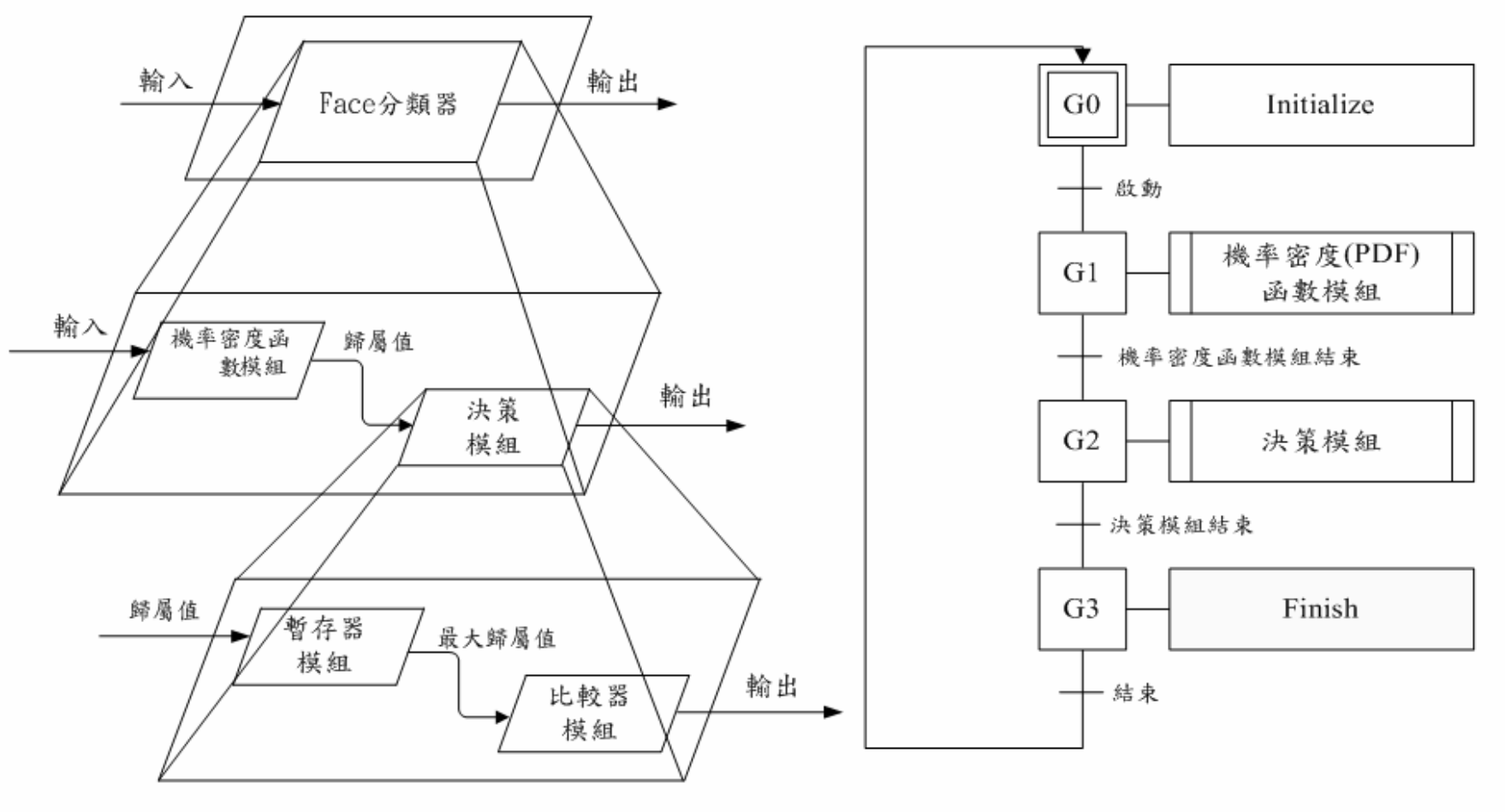


BAP- AES解密模組硬體合成

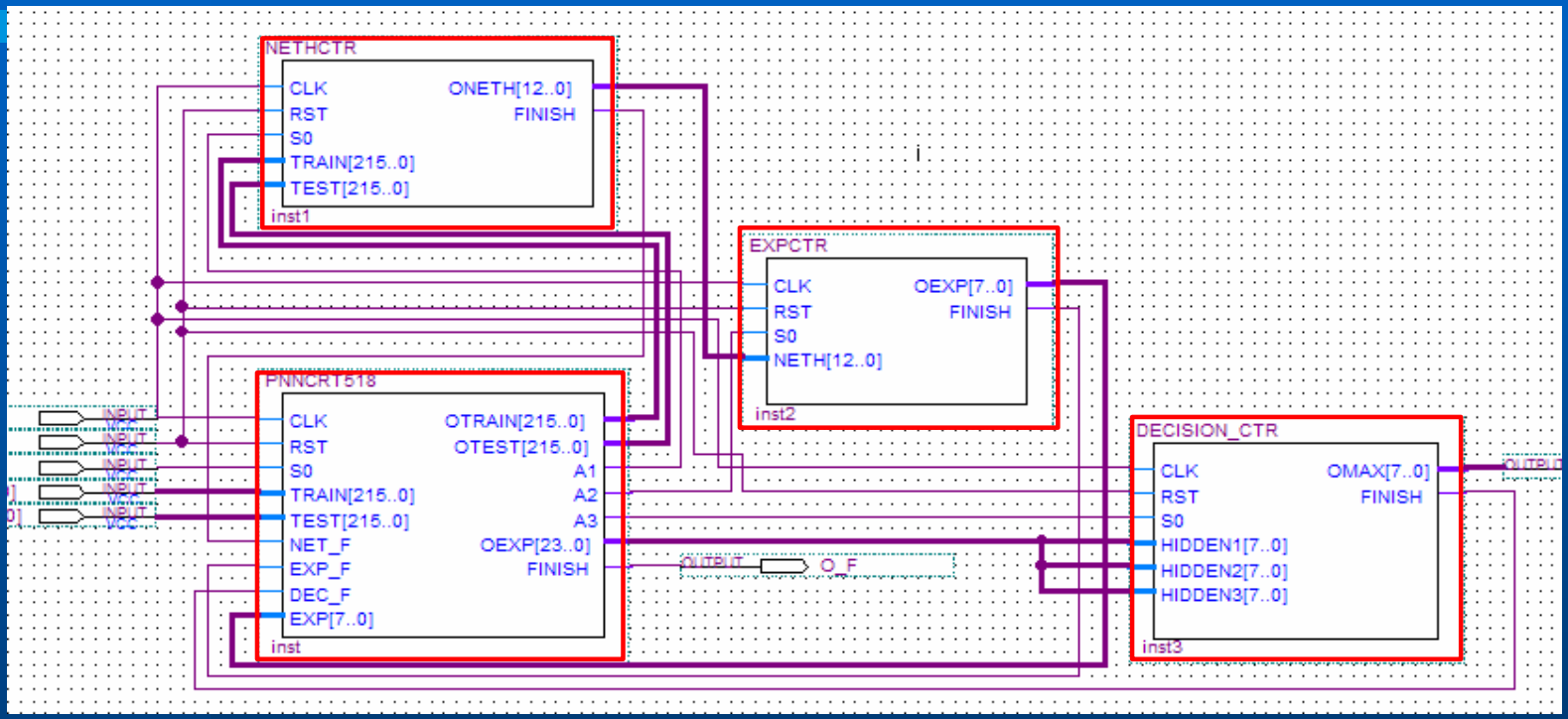


	AesDecrypt 模組
Total Logic Elements	3,731/ 33,216
Cycles/per Text	1300
Most Critical Path Delay	11.909 ns
Performance	83.97 MHz

BAP/PNN神經網路模組

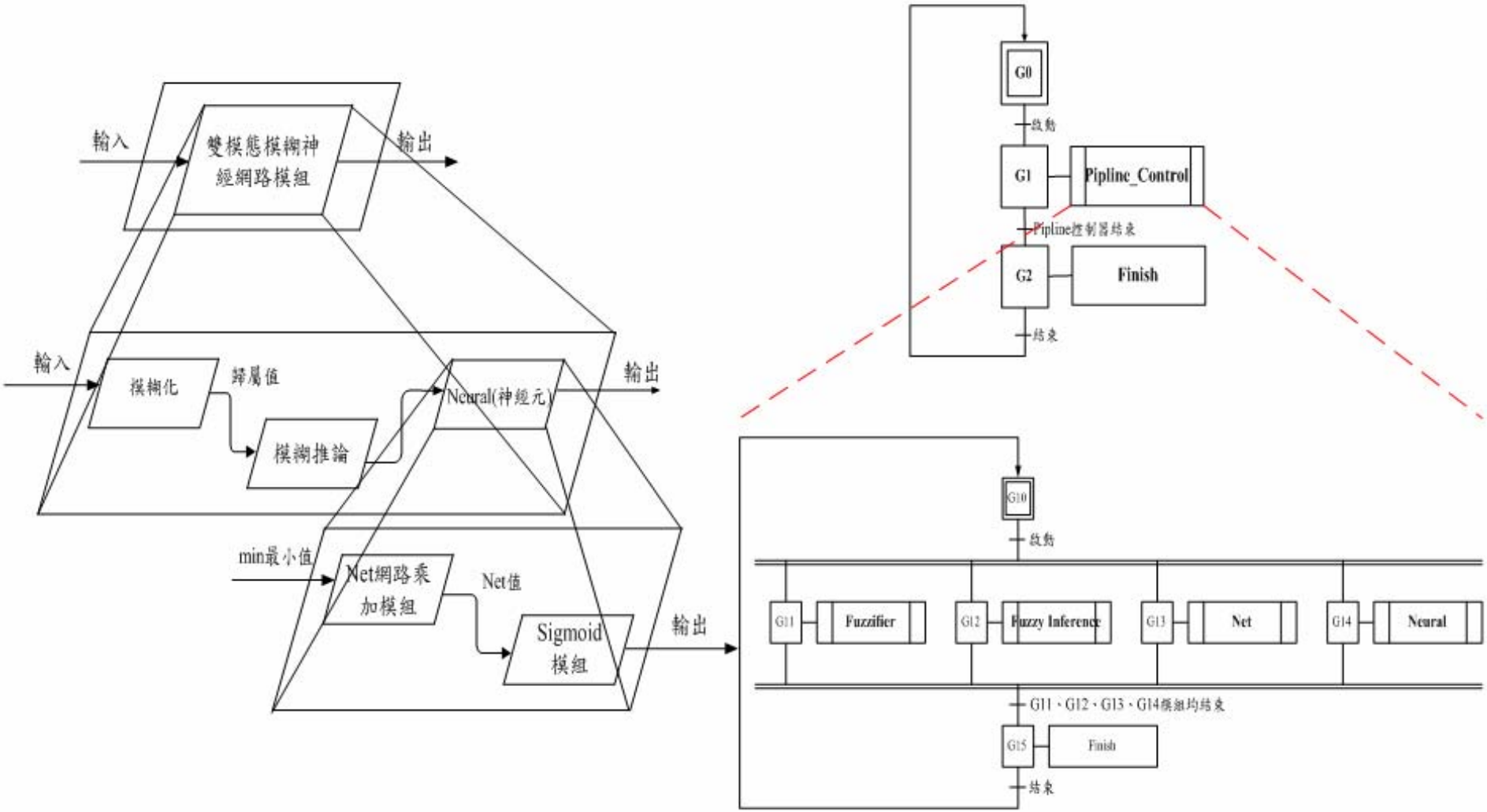


BAP/PNN神經網路硬體合成

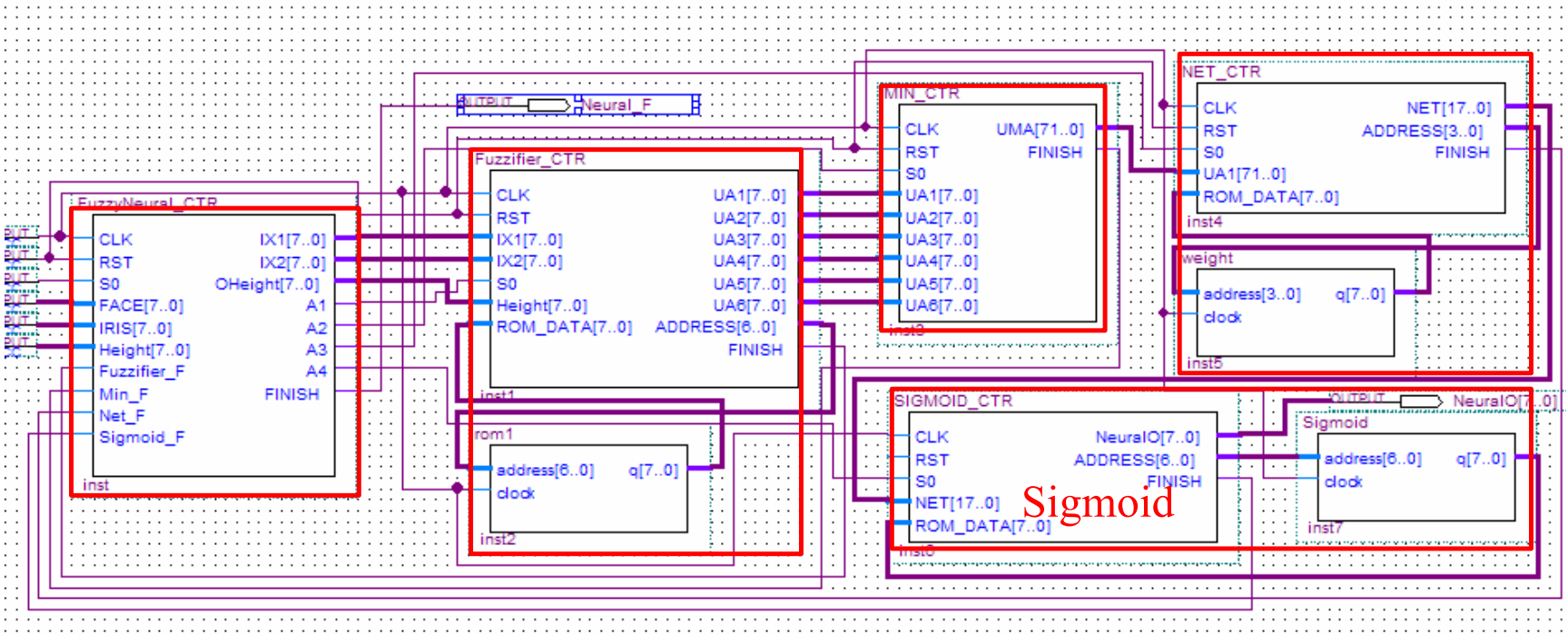


	系統整合模組
Total Logic Elements	1,100 / 33,216
Cycles/per Decision	50
Most Critical Path Delay	12.764 ns
Performance	78.35 MHz

BAP/模糊神經網路建模

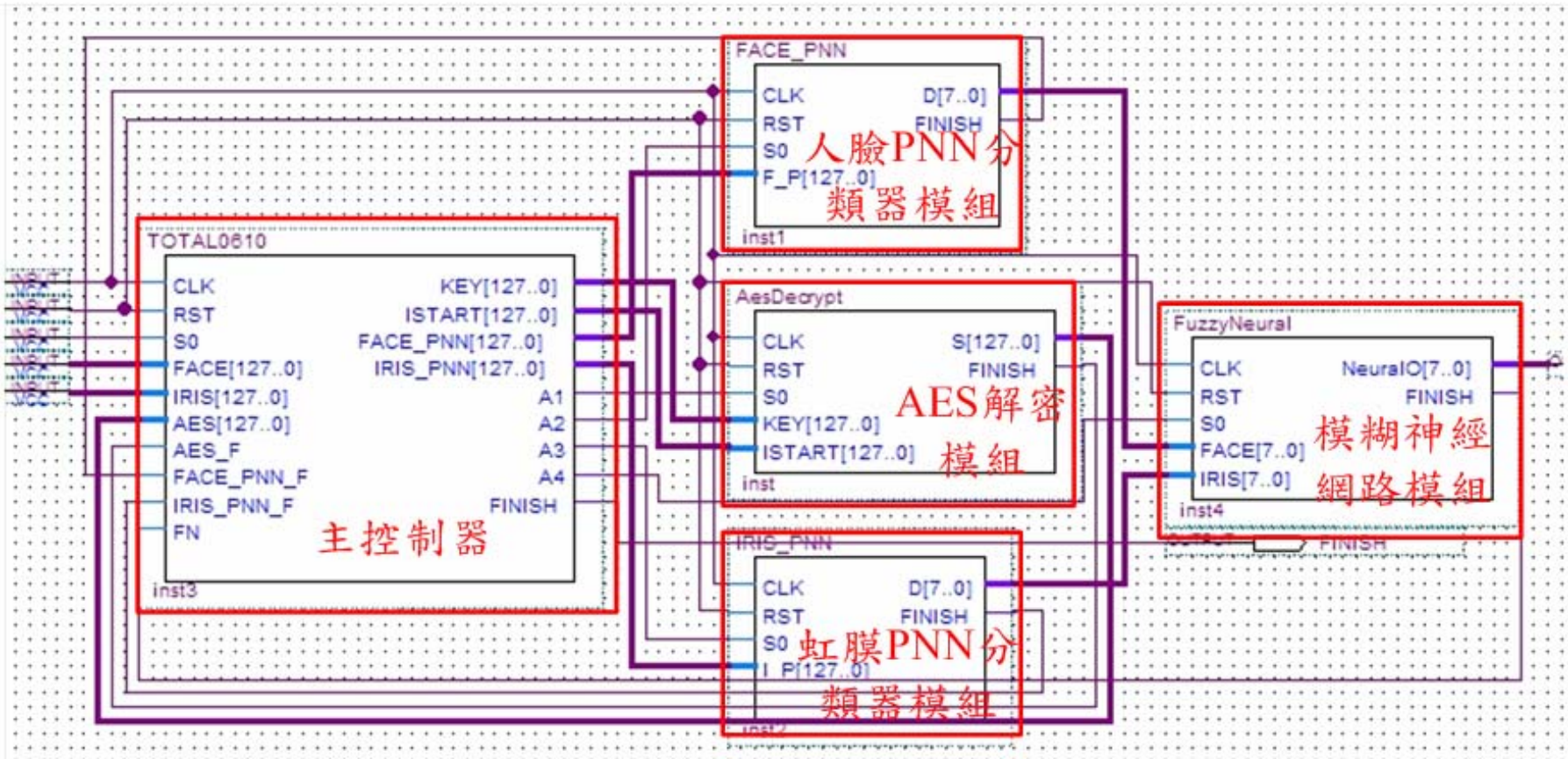


BAP/模糊神經網路硬體合成



	循序架構	管線化架構
Total Logic Elements	310 / 33,216	399 / 33,216
Cycles/per FuzzyNeural	30	16
Most Critical Path Delay	13.789 ns	9.740ns
Performance	72.52 MHz	102.67 MHz

BAP系統合成架構


FACE_PNN
CLK
RST
S0
F_P[127..0]
D[7..0]
FINISH
人臉PNN分類器模組
inst1
TOTAL0610
CLK
RST
S0
FACE[127..0]
IRIS[127..0]
AES[127..0]
AES_F
FACE_PNN_F
IRIS_PNN_F
FN
KEY[127..0]
ISTART[127..0]
FACE_PNN[127..0]
IRIS_PNN[127..0]
A1
A2
A3
A4
FINISH
主控制器
inst3
AesDecrypt
CLK
RST
S0
KEY[127..0]
ISTART[127..0]
S[127..0]
FINISH
AES解密模組
inst
IRIS_PNN
CLK
RST
S0
I_P[127..0]
D[7..0]
FINISH
虹膜PNN分類器模組
FuzzyNeural
CLK
RST
S0
FACE[7..0]
IRIS[7..0]
NeuralO[7..0]
FINISH
模糊神經網路模組
inst4

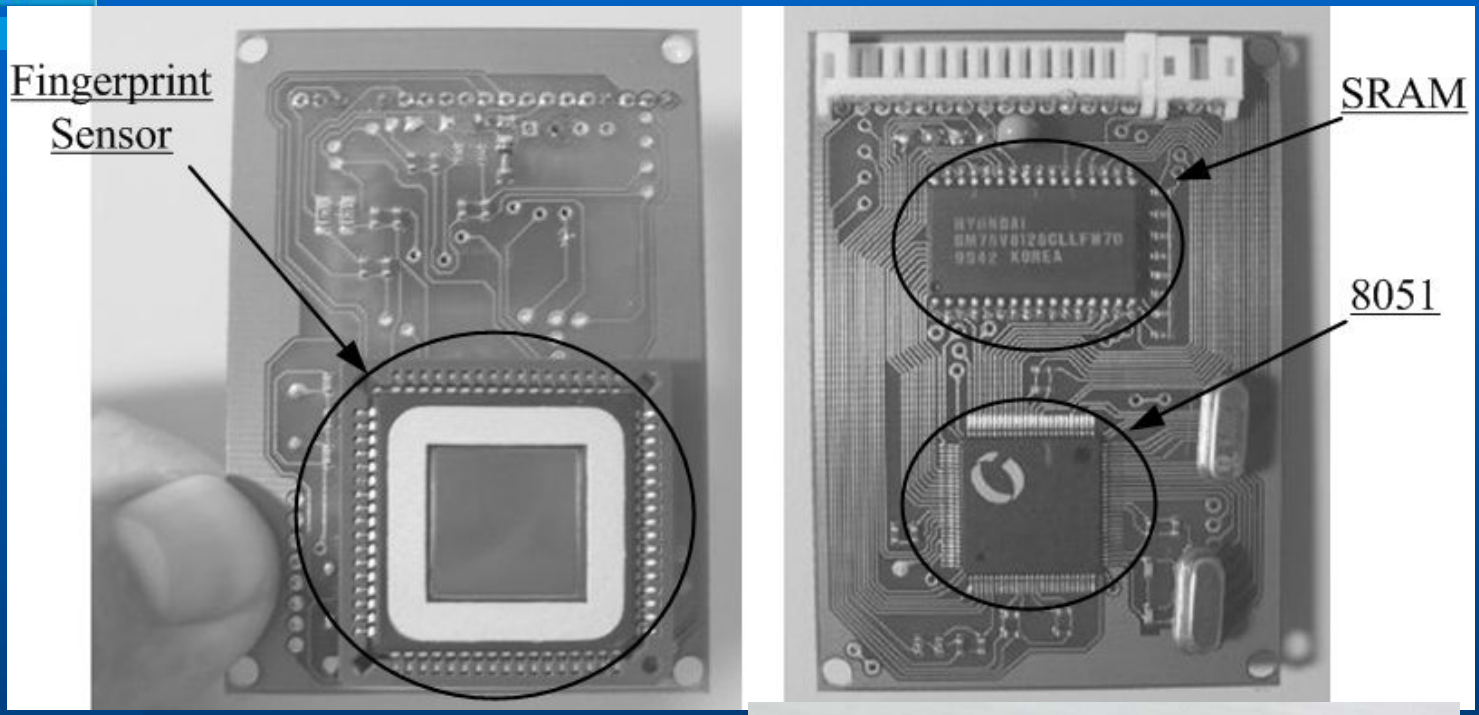
BAP生物辨識系統晶片性能

	系統整合模組
Total Logic Elements	5,703/ 33,216 (17%)
Cycles/per verification	11667
Verification Speed	0.175ms
Most Critical Path Delay	13.038 ns
Max Frequency	76.70 MHz

複雜系統晶片設計demo

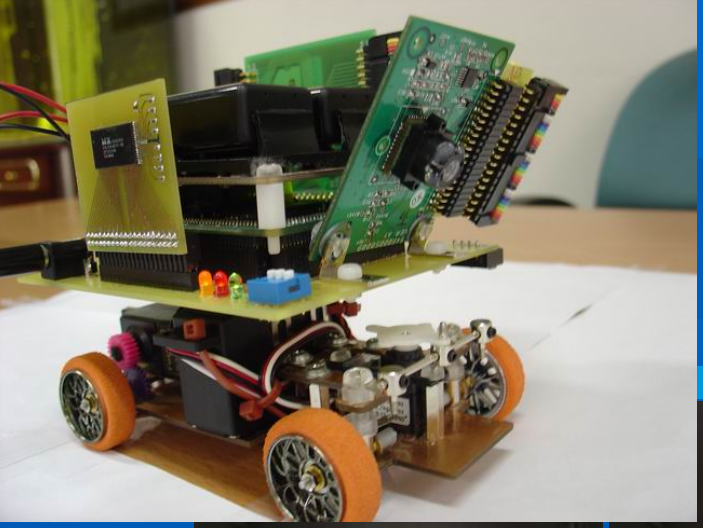
- 精簡嵌入式指紋辨識系統
- 視覺伺服自動倒車入庫系統
- 智慧型影像插補器ASIC
- 渾沌加密與保密通訊ASIC
- 被動式自動對焦系統
- 具保密通信功能的指紋遙控器
- DSP嵌入式指紋辨識系統
- 仿昆蟲智慧型六腳機器人

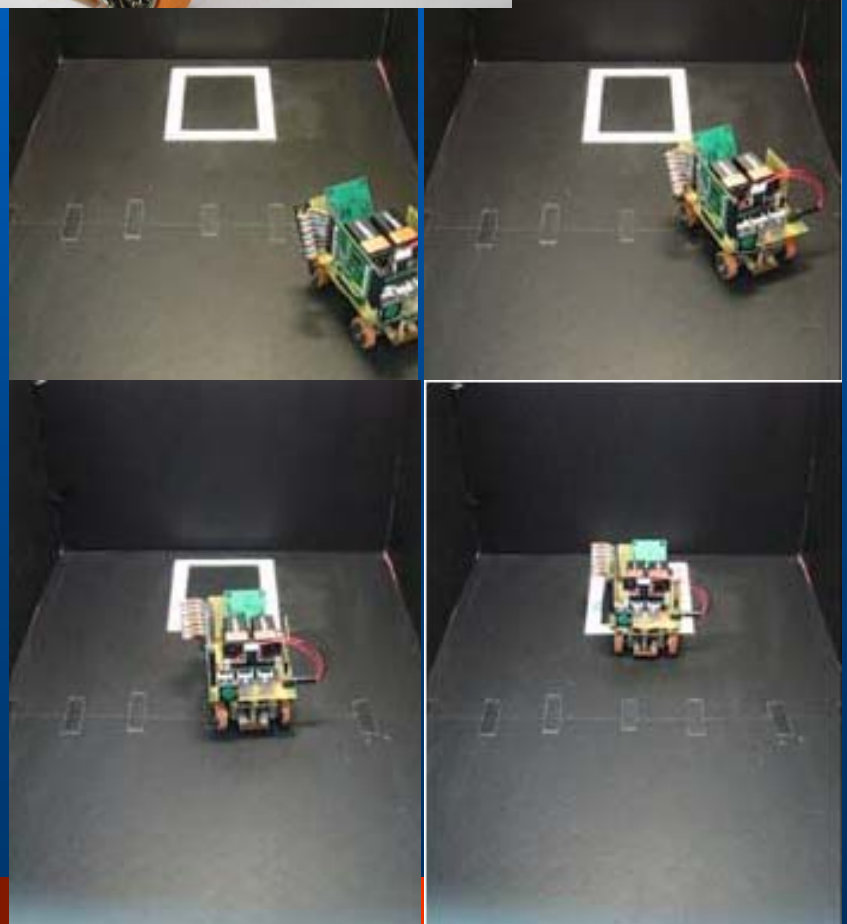
精簡嵌入式指紋身份識別系統



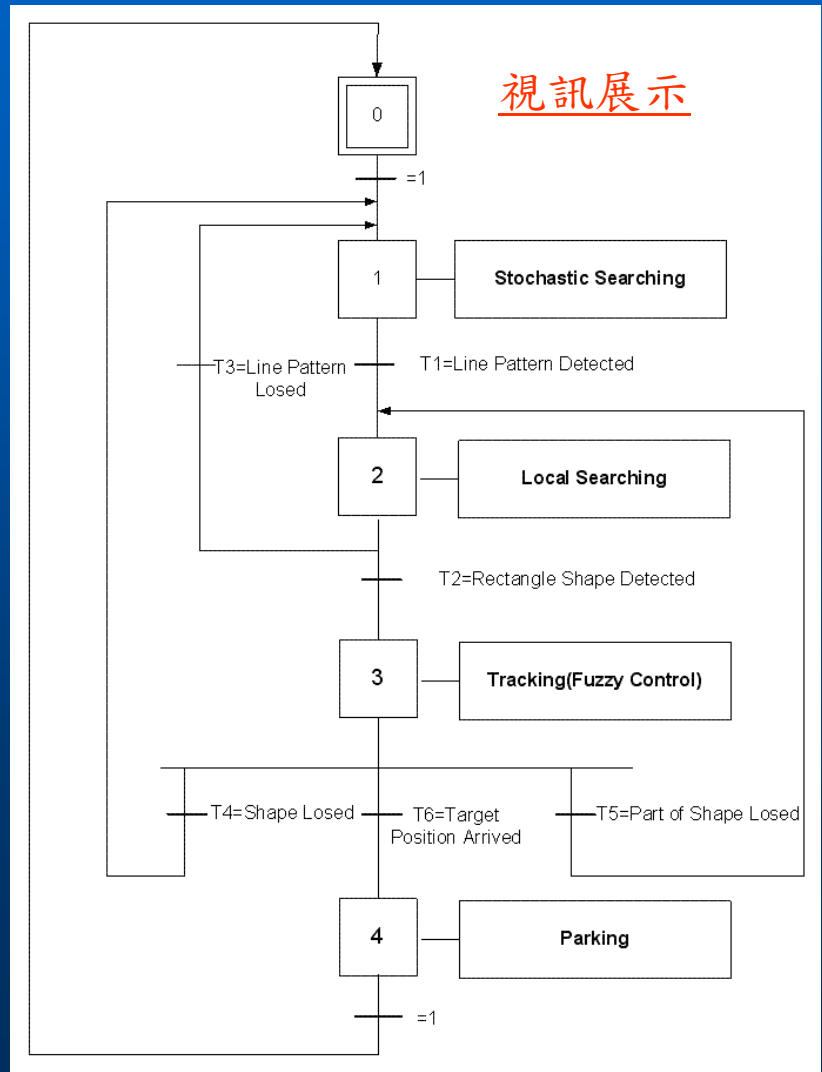
25MHz 8051
12 Kbytes ROM
35 Kbytes RAM
EER=4.5%

視覺伺服自動倒車入庫系統

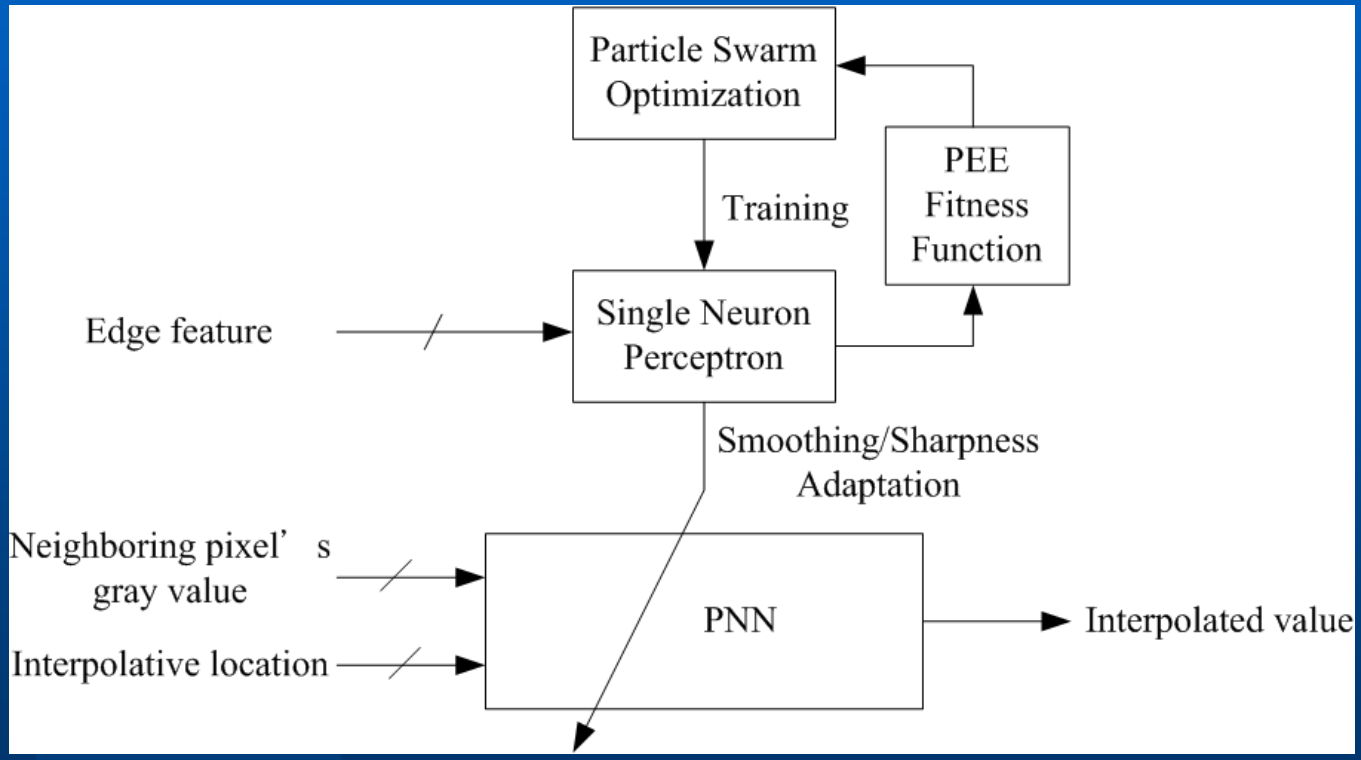


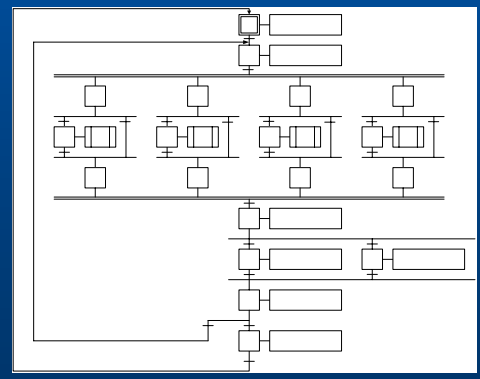


視訊展示



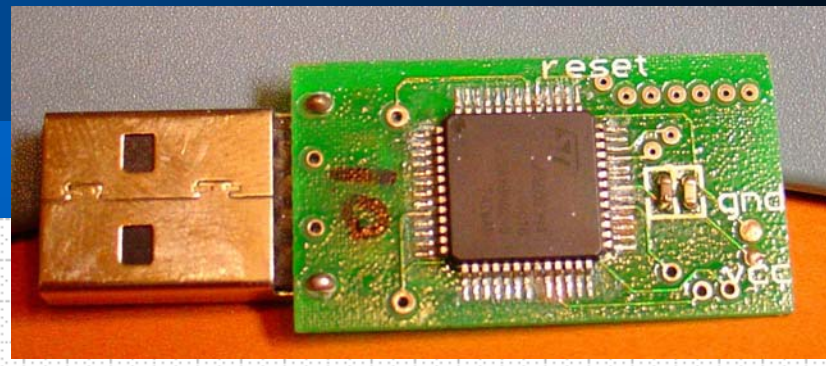
智慧型影像插補器ASIC

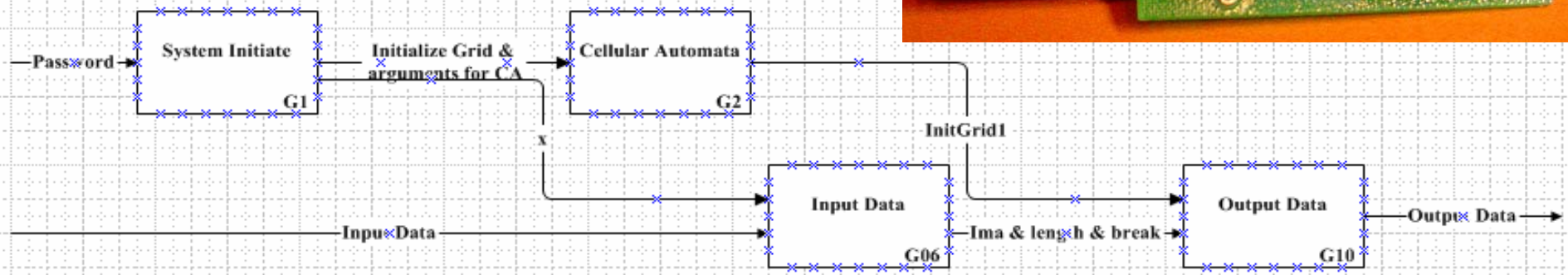




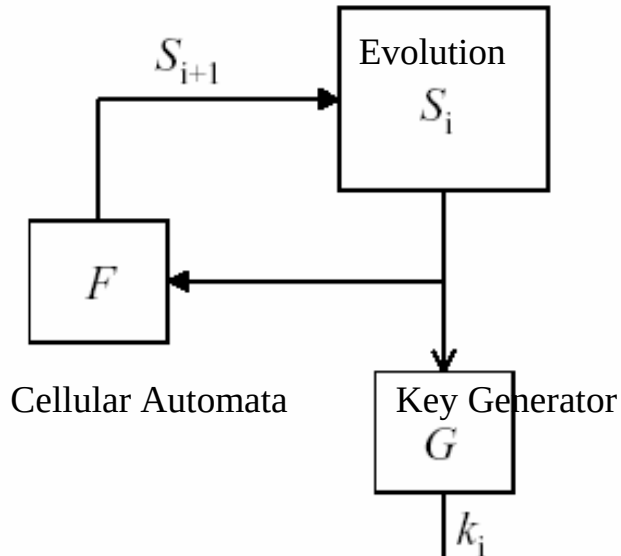
1-8X stepless zooming;
80K gates count;1 pixel/3 clocks;
20 fps at 20Mhz in XGA(1024x768)

渾沌加密晶片

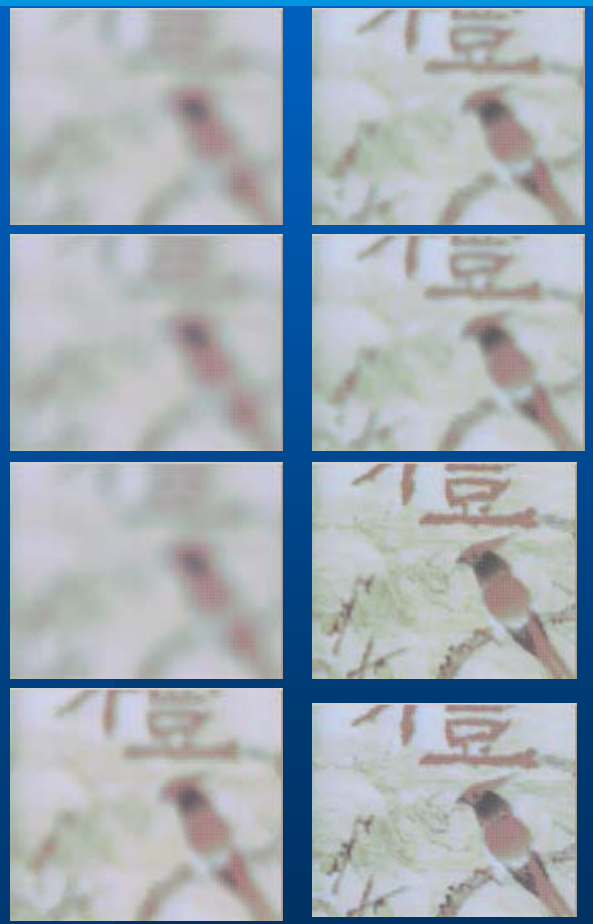


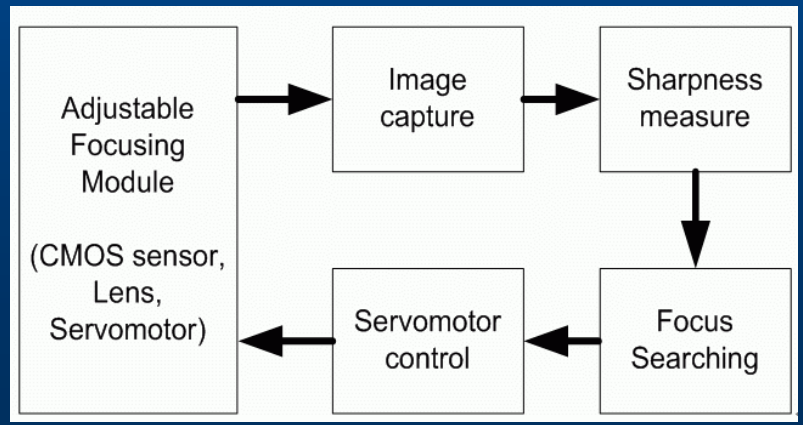


Gate Counts: ~ 70,000
Speed:41.6MB/s at 130MHZ
應用：加密影音撥放器/加密隨身碟/無線網路保密通訊/FPGA電路保密裝置

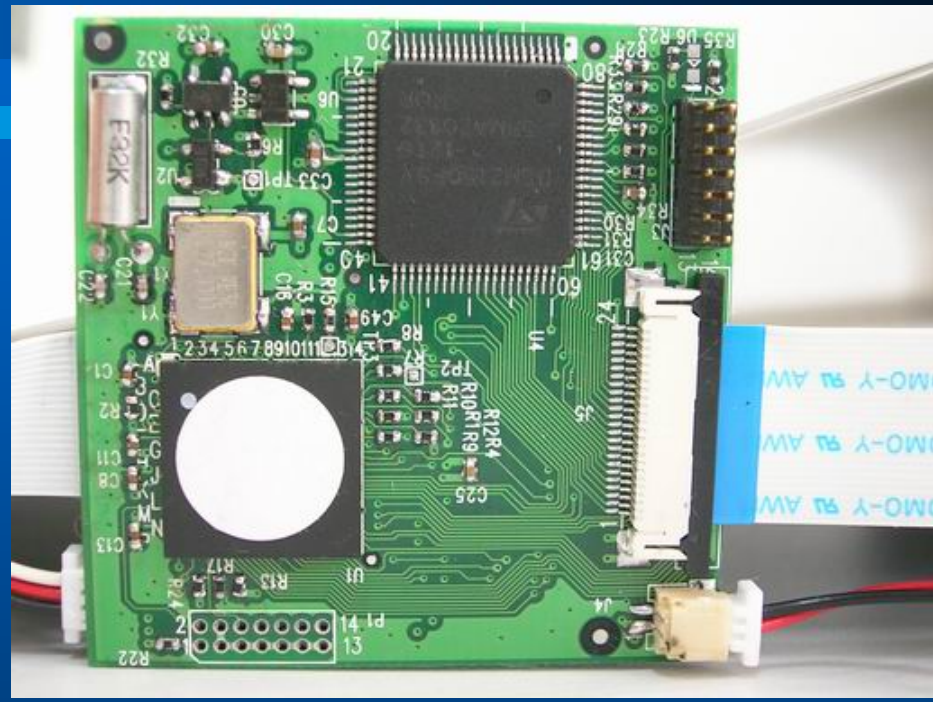


被動式自動對焦系統

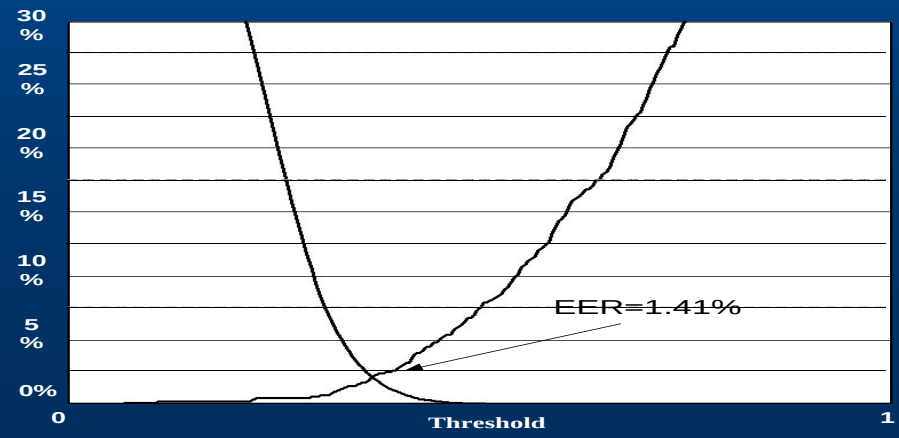



Adjustable Focusing Module
(CMOS sensor, Lens, Servomotor)
Image capture
Sharpness measure
Focus Searching
Servomotor control

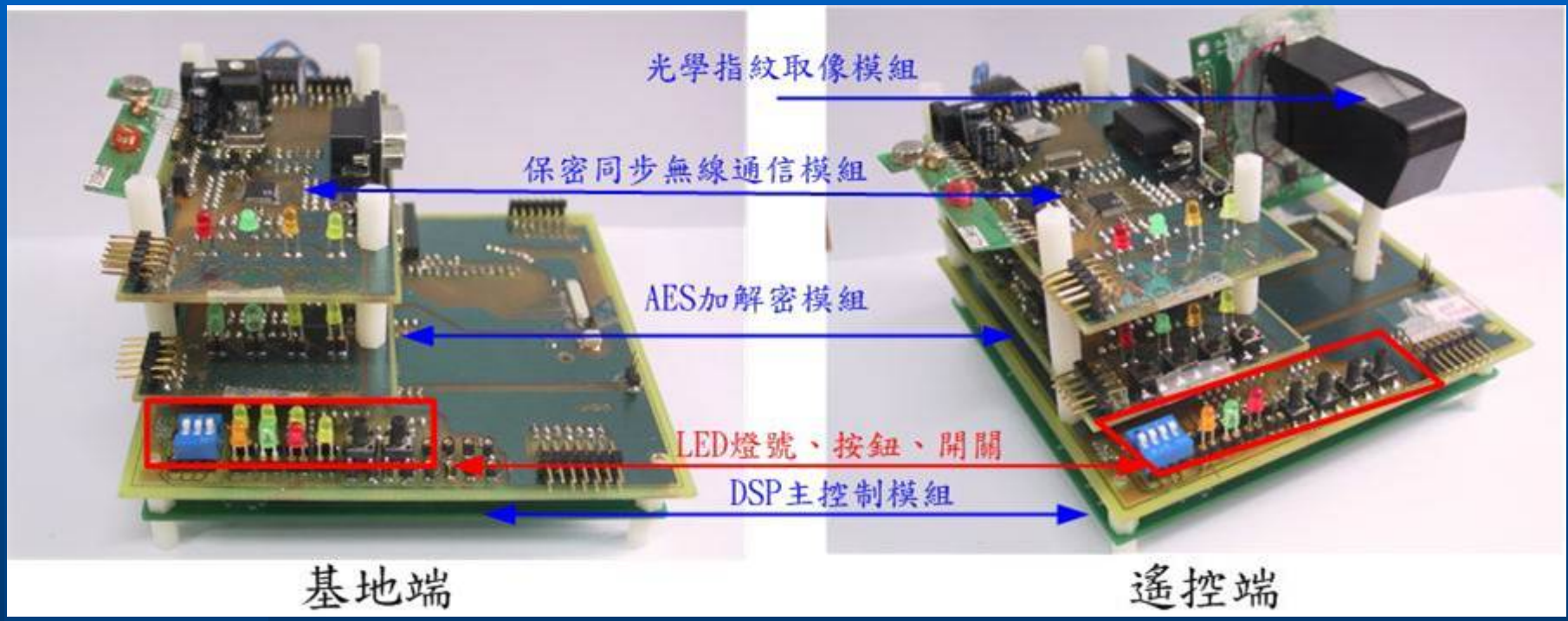
DSP嵌入式指紋辨識系統



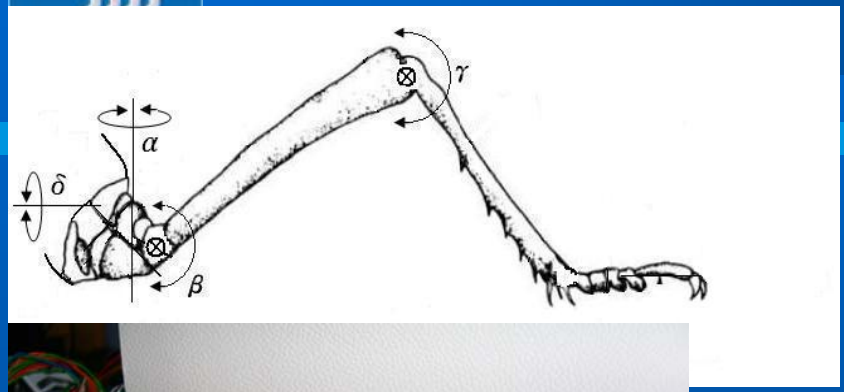
54x13=702枚指紋EER=1.41%
辨識速度<0.5秒/一枚指紋。
Code Memory:33KBytes
Data Memory:169KBytes
Flash Memory: 480 bytes/指紋特徵
指紋特徵比對速度：800枚/sec

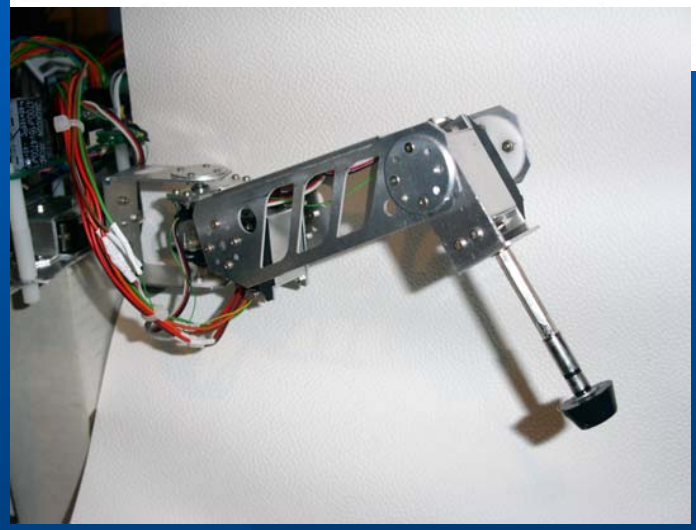


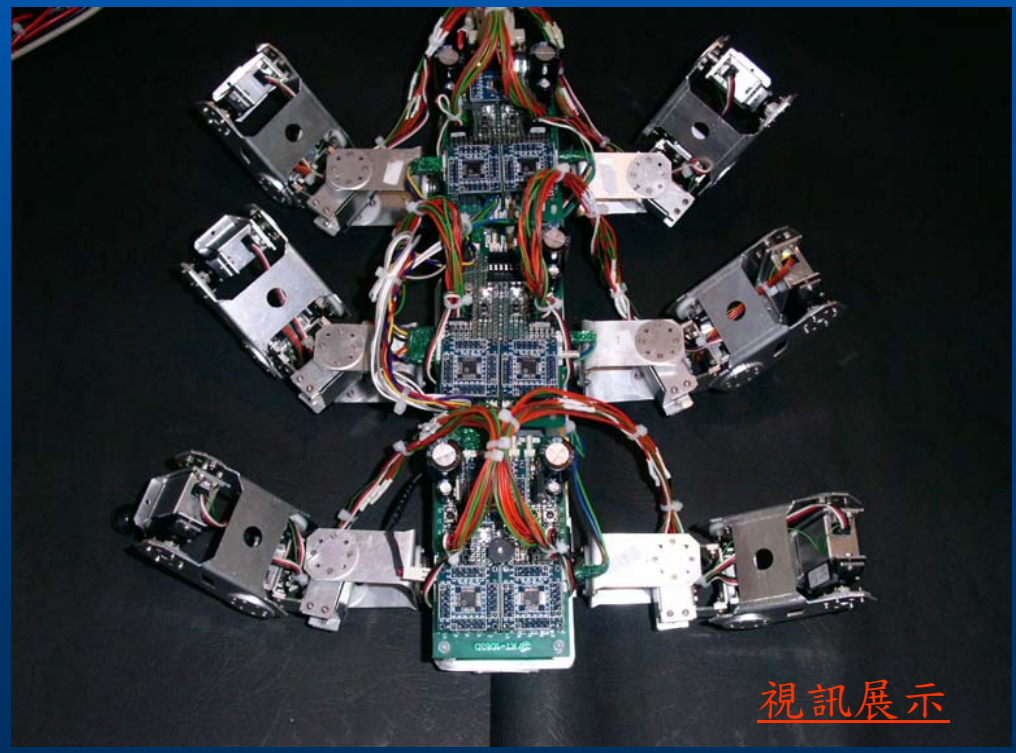
具保密通信功能的指紋遙控器

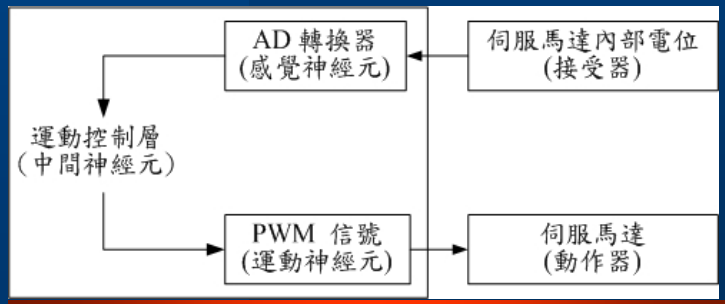


仿昆蟲智慧型六腳機器人









視訊展示

PART 4

結語

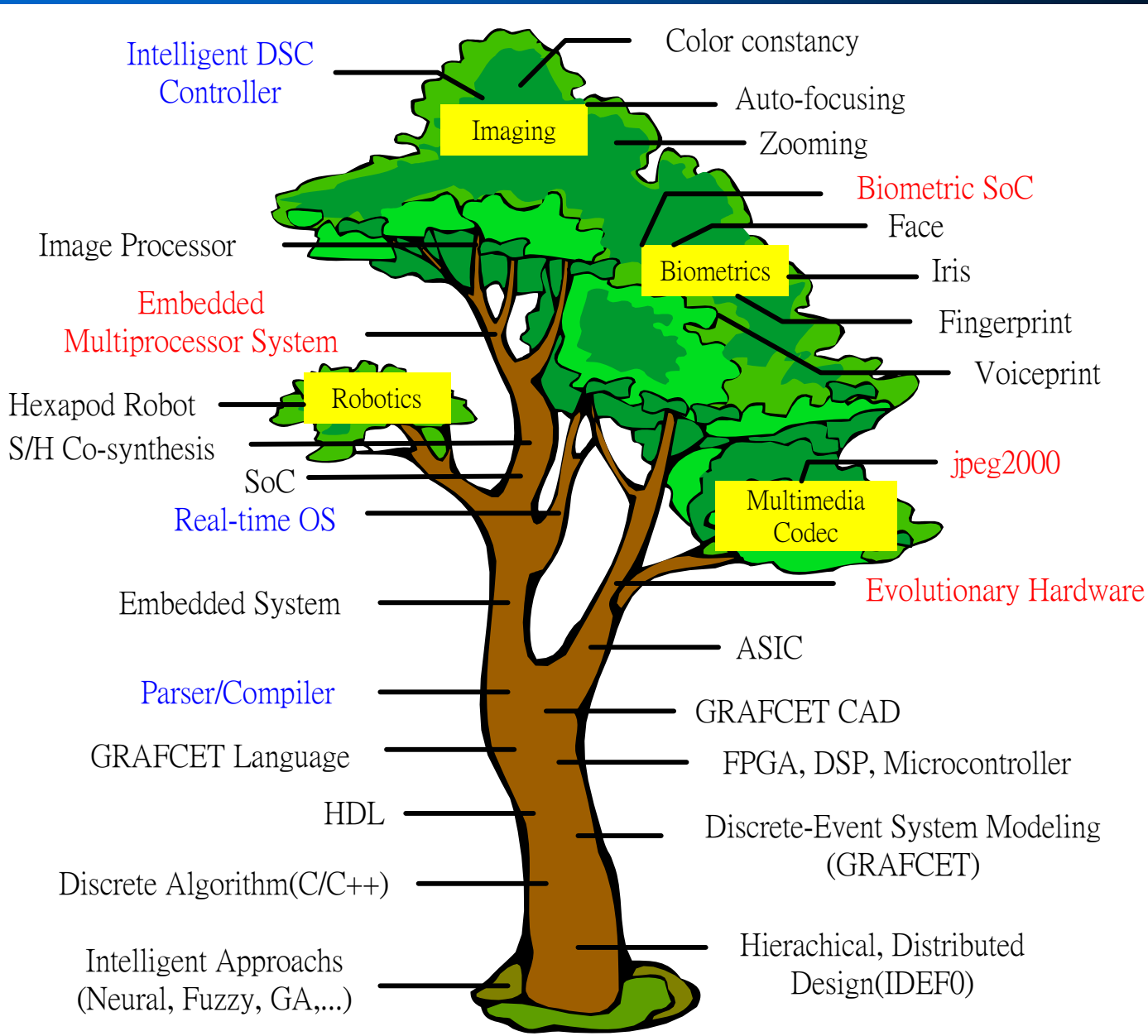
MIAT設計方法論特色

- MIAT設計方法論是基於Top-Down設計範式(paradigm)，系統設計採階層式、模組化的功能架構；
- 針對每一個功能獨立的模組，使用GRAFCET圖形化工具建立其離散事件模型；
- 根據一組合成的法則，所有GRAFCET模型得以轉譯為VHDL控制器電路，再結合一些通用的資料流(data flow)元件，便可實現系統的高階合成；
- 方法論採用均一的解決問題程序，包括問題形式化(formalization)、建模(modeling)和設計規則、工具和技術。因此利於知識的整合和團隊研發。

MIAT設計方法論的貢獻

- 設計者得以採用系統化的方式而不需過度仰賴個人技術經驗來從事複雜系統設計工作
- 利於團隊分工和協同設計
- 確保設計品質
- 縮短系統設計時間
- 高度重複使用性，易於修改和擴充原產品
- 技術和經驗得以累積，形成知識管理平台

MIAT方法論過去、現在與未來



Q&A

歡迎指教!