數位系統導論實驗

Lab7 Structural Modeling of Carry-Lookahead Adder(CLA)

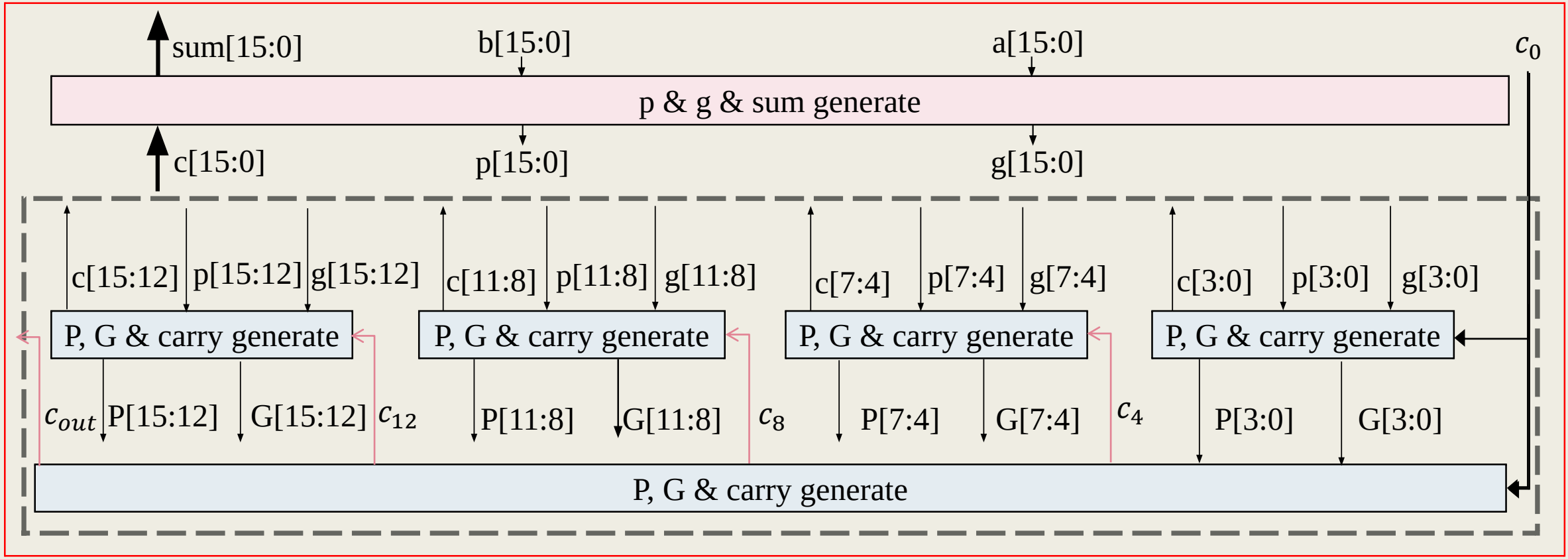
Outline

- 課程目的
- 範例練習
- 作業說明及評分方式

課程目的

- 練習使用structural modeling 描述數位電路，並練習完成課程教授之CLA細部設計

16bit CLA



16 bit CLA

範例練習 – PG & c generator module

- 模組的部分與 4bit CLA 大同小異，但需在原本生成 c 的模組加入生成 group G 與 group P 的描述

```
//submodule for generating P, G & carry
module PG_carry_generator( p, g, cin, gG, gP, c);

  input [3:0] p, g;
  input cin;
  output gG, gP;
  output [4:1] c;

  assign gG = (g[0] & p[1] & p[2] & p[3]) | (g[1] & p[2] & p[3]) | (g[2] & p[3]) | g[3];
  assign gP = p[0] & p[1] & p[2] & p[3];

  assign c[1] = g[0] | (p[0] & cin);
  assign c[2] = g[1] | (p[1] & g[0]) | (p[1] & p[0] & cin);
  assign c[3] = g[2] | (p[2] & g[1]) | (p[2] & p[1] & g[0]) | (p[2] & p[1] & p[0] & cin);
  assign c[4] = g[3] | (p[3] & g[2]) | (p[3] & p[2] & g[1]) | (p[3] & p[2] & p[1] & g[0]) | (p[3] & p[2] & p[1] & p[0] & cin);

endmodule
```

範例練習 – pg & sum generator module

- 調整各個模組的輸入埠、輸出埠與接線，使其硬體描述符合架構圖

```
//submodule for generating p & g
module pg_generator( a, b, p, g);

   input [15:0] a, b;
   output [15:0] p, g;

   assign p = a | b ;
   assign g = a & b ;

endmodule
```

```
//submodule for generating sum
module sum_generator(a, b, cin, c, sum);

   input [15:0] a, b;
   input cin;
   input [15:1] c;
   output [15:0] sum;

   assign sum = a ^ b ^ {c,cin} ;

endmodule
```

範例練習 – Top module

```
//top module
module cla_16bit( a, b, cin, sum, cout);

  //宣告區段
  input [15:0] a, b;
  input cin;
  output [15:0] sum;
  output cout;

  wire [3:0] G, P;
  wire [15:0] p, g;
  wire [16:0] c;
  wire [4:1] c4x;
```

```
  //邏輯區段
  //generating  p & g
  pg_generator pg1(a[15:0], b[15:0], p[15:0], g[15:0]);
  //generating group p & g
  PG_carry_generator PG1( p[3:0], g[3:0], , G[0], P[0], );
  PG_carry_generator PG2( p[7:4], g[7:4], , G[1], P[1], );
  PG_carry_generator PG3( p[11:8], g[11:8], , G[2], P[2], );
  PG_carry_generator PG4( p[15:12], g[15:12], , G[3], P[3], );
  //generating c4, c8, c12 & cout
  PG_carry_generator carry_c4x(P[3:0], G[3:0], cin, , , c4x[4:1]);
  assign cout = c4x[4];
  //generating carry
  PG_carry_generator carry1(p[3:0], g[3:0], cin, , , c[4:1]);
  PG_carry_generator carry2(p[7:4], g[7:4], c4x[1], , , c[8:5]);
  PG_carry_generator carry3(p[11:8], g[11:8], c4x[2], , , c[12:9]);
  PG_carry_generator carry4(p[15:12], g[15:12], c4x[3], , , c[16:13]);
  //generating sum
  sum_generator sum1(a[15:0], b[15:0], cin, c[15:1], sum[15:0]);

endmodule
```

範例練習 – 驗證16bit CLA

1. 將課程提供的 testbench 與完成的 16bit CLA 放在同一資料夾
2. 輸入指令 “iverilog –o test testbench_16bit.v”與 “vvp test”
3. 觀察 testbench 的20筆測試是否皆正確，藉此驗證 16 bit CLA
4. 若出現錯誤，可利用波形圖觀察出錯環節

作業說明及課程評分

- 作業Part1：參考範例以 Structural Modeling 完成 64bit CLA 且 Fan-in ≤ 4，並用課程提供的 testbench 驗證完全正確
- 作業Part2：參考範例以 Structural Modeling 完成 16bit CLA 且 Fan-in ≤ 3，並用課程提供的 testbench 驗證完全正確
- Demo 時間：測驗時間共分四梯次，分別為 19:20、19:40、 20:00 與 20:20
- Demo 梯次：與 Lab 1 相同
- Demo 地點：工程一館101A
- 評分方式：part1 70%，part2 30%