

LAB 05 (Carry Look-ahead Adder)

2013210111 남세현

1. 코드

[adder1bit.v]

```
module adder1bit (a, b, c, s, p, g);  
    input a, b, c;  
    output s, p, g;  
  
    assign p = a ^ b;  
    assign g = a & b;  
    assign s = p ^ c;  
endmodule
```

[carrygu.v]

```
module carrygu (p, g, c0, P, G, c);  
    input [3:0] p, g;  
    input c0;  
    output P, G;  
    output [4:1]c;  
  
    wire [3:0] p, g;  
  
    assign c[1] = g[0] | ( p[0] & c0 );  
    assign c[2] = g[1] | ( p[1] & c[1] );  
    assign c[3] = g[2] | ( p[2] & c[2] );  
    assign c[4] = g[3] | ( p[3] & c[3] );  
  
    assign G = g[3] | ( p[3] & g[2] ) | ( p[3] & p[2] & g[1] ) | ( p[3] & p[2] & p[1] & g[0] );  
    assign P = p[3] & p[2] & p[1] & p[0];  
endmodule
```

[adder4bit.v]

```
`include "adder1bit.v"
```

```
`include "carrygu.v"
```

```
module adder4bit(a, b, cin, sum, P, G, cout);
```

```
    input [3:0] a, b;
```

```
    input cin;
```

```
    output [3:0] sum;
```

```
    output P, G, cout;
```

```
    wire [4:1] carry;
```

```
    wire [3:0] p;
```

```
    wire [3:0] g;
```

```
    adder1bit adder1bit0 ( a[0], b[0], cin, sum[0], p[0], g[0] );
```

```
    adder1bit adder1bit1 ( a[1], b[1], carry[1], sum[1], p[1], g[1] );
```

```
    adder1bit adder1bit2 ( a[2], b[2], carry[2], sum[2], p[2], g[2] );
```

```
    adder1bit adder1bit3 ( a[3], b[3], carry[3], sum[3], p[3], g[3] );
```

```
    carrygu carrygu_ (p, g, cin, P, G, carry);
```

```
    assign cout = carry[4];
```

```
endmodule
```

[CLA.v]

```
`include "adder4bit.v"
```

```
`include "carrygu.v"
```

```
module CLA (a, b, cin, s, cout);
```

```
    input [15:0] a, b;
```

```
    input cin;
```

```
    output [15:0] s;
```

```
    output cout;
```

```
    wire [15:0] a, b;
```

```
    wire [15:0] s;
```

```
    wire [2:0] c;
```

```
    wire [3:0] p, g;
```

```
    wire [3:0] c_temp;
```

```
    wire P, G;
```

```
    adder4bit adder4bit0 ( a[3:0], b[3:0], cin,    s[3:0], p[0], g[0], c[0] );
```

```
    adder4bit adder4bit1 ( a[7:4], b[7:4], c[0],    s[7:4], p[1], g[1], c[1] );
```

```
    adder4bit adder4bit2 ( a[11:8], b[11:8], c[1],    s[11:8], p[2], g[2], c[2] );
```






```
    adder4bit adder4bit3 ( a[15:12], b[15:12], c[2], s[15:12], p[3], g[3], cout );
```

```
    carrygu carrygu_ ( p, g, cin, P, G, c_temp);
```

```
    assign cout = c_temp[3];
```

```
endmodule
```

2. 결과

 /tb4CLA/C_OUT	1'd1										
 + /tb4CLA/SUM	-16'd11982	16'd78	16'd221	16'd769	16'd999	16'd8892	16'd12322	16'd12472	16'd6864	-16'd11982	16'd13208
 /tb4CLA/C_IN	-1'd1										
 + /tb4CLA/A	-16'd10992	16'd21	16'd23	16'd423	16'd999	16'd5435	16'd2454	-16'd2985	16'd2315	-16'd10992	16'd15478
 + /tb4CLA/B	-16'd991	16'd56	16'd198	16'd345	16'd0	16'd3456	16'd9867	16'd15456	16'd4548	-16'd991	-16'd2271

16'd5827	16'd3	-16'd10046	16'd1230	16'd16809	-16'd12231	16'd9	16'd11	16'd12347	16'd7
16'd6152	16'd1	-16'd19592	16'd1234	16'd17832	-16'd10955	16'd2	16'd4	16'd12345	16'd5
-16'd326	16'd1	16'd9545	-16'd5	-16'd1024	-16'd1277	16'd6		16'd1	

3. 결과 분석

RCA처럼 한번에 두 비트씩 연산을 하는게 아니라, 일종의 트리구조의 형태로 덧셈을 수행하는 CLA다. 입력값 16bit A, B, 그리고 C_IN의 덧셈값 연산 결과인 SUM, C_OUT 모두 정상적으로 잘 출력되었다.

4. 구현시 어려웠던 점

Verilog 코드 중 wire a[3:0]과 wire [3:0]a는 다른 것이라고 한다. 이 둘의 차이점을 몰라서 컴파일 오류를 해결하지 못했었는데, 구글링 결과로 문제를 해결하였다.