컴퓨터 구조

2013210061

채윤병

Lab Session 5

**adder1bit.v**

module adder1bit (a, b, c, s, p, g);

input a, b, c;

output s, p, g;

/\* to do \*/

xor(s,a,b,c);

and(g,a,b);

xor(p,a,b);

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;

/\* to do \*/

assign c[1]=g[0] | (p[0] & c0);

assign c[2]=g[1] | (p[1] & (g[0] | (c0 & p[0])));

assign c[3]=g[2] | (p[2] & (g[1] | (p[1] & (g[0] | (p[0] & c0)))));

assign c[4]=g[3] | (p[3] & g[2]) | (p[2] & p[2] & g[1]) | (p[3] & p[2] & p[1] & g[0]) | (p[3] & p[2] & p[1] & p[0] & c0);

assign P = p[3] & p[2] & p[1] & p[0];

assign G = g[3] | (p[3] & g[2]) | (p[3] & p[2] & g[1]) | (p[3] & p[2] & p[1] & g[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;

/\* to do \*/

wire [3:0] p,g;

wire [4:1] c;

adder1bit a00(a[0], b[0], cin , sum[0], p[0], g[0]);

adder1bit a01(a[1], b[1], c[1], sum[1], p[1], g[1]);

adder1bit a02(a[2], b[2], c[2], sum[2], p[2], g[2]);

adder1bit a03(a[3], b[3], c[3], sum[3], p[3], g[3]);

carrygu carrygu0(p, g, cin, P, G, c);

assign cout = c[4];

endmodule

**CLA.v**

`include "adder4bit.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 [3:0] c;

wire [3:0] p, g;

wire [3:0] c\_temp;

wire P, G;

/\* to do \*/

adder4bit a1(a[3:0] , b[3:0] , cin , s[3:0] , p[0], g[0], c\_temp[0]);

adder4bit a2(a[7:4] , b[7:4] , c[0], s[7:4] , p[1], g[1], c\_temp[1]);

adder4bit a3(a[11:8] , b[11:8] , c[1], s[11:8] , p[2], g[2], c\_temp[2]);

adder4bit a4(a[15:12], b[15:12], c[2], s[15:12], p[3], g[3], c\_temp[3]);

carrygu carrygu22(p, g, cin, P, G, c);

assign cout = c[3];

endmodule

**tbCLA.v**

module tb4CLA;

wire C\_OUT;

wire [15:0] SUM;

reg C\_IN;

reg [15:0] A, B;

CLA CLA00 (A, B, C\_IN, SUM, C\_OUT);

initial

begin

C\_IN = 1'd1;

A = 16'd21;

B = 16'd56;

#5

C\_IN = 1'd0;

A = 16'd23;

B = 16'd198;

#5

C\_IN = 1'd1;

A = 16'd423;

B = 16'd345;

#5

C\_IN = 1'd0;

A = 16'd999;

B = 16'd0;

#5

C\_IN = 1'd1;

A = 16'd5435;

B = 16'd3456;

#5

C\_IN = 1'd1;

A = 16'd2454;

B = 16'd9867;

#5

C\_IN = 1'd1;

A = 16'd62551;

B = 16'd15456;

#5

C\_IN = 1'd1;

A = 16'd2315;

B = 16'd4548;

#5

C\_IN = 1'd1;

A = 16'd54544;

B = 16'd64545;

#5

C\_IN = 1'd1;

A = 16'd15478;

B = 16'd63265;

#5

C\_IN = 1'd1;

A = 16'd6152;

B = 16'd65210;

#5

C\_IN = 1'd1;

A = 16'd1;

B = 16'd1;

#5

C\_IN = 1'd1;

A = 16'd45944;

B = 16'd9545;

#5

C\_IN = 1'd1;

A = 16'd1234;

B = 16'd65531;

#5

C\_IN = 1'd1;

A = 16'd17832;

B = 16'd64512;

#5

C\_IN = 1'd1;

A = 16'd54581;

B = 16'd64259;

#5

C\_IN = 1'd1;

A = 16'd2;

B = 16'd6;

#5

C\_IN = 1'd1;

A = 16'd4;

B = 16'd6;

#5

C\_IN = 1'd1;

A = 16'd12345;

B = 16'd1;

#5

C\_IN = 1'd1;

A = 16'd5;

B = 16'd65537;

#5

C\_IN = 1'd1;

A = 16'd54;

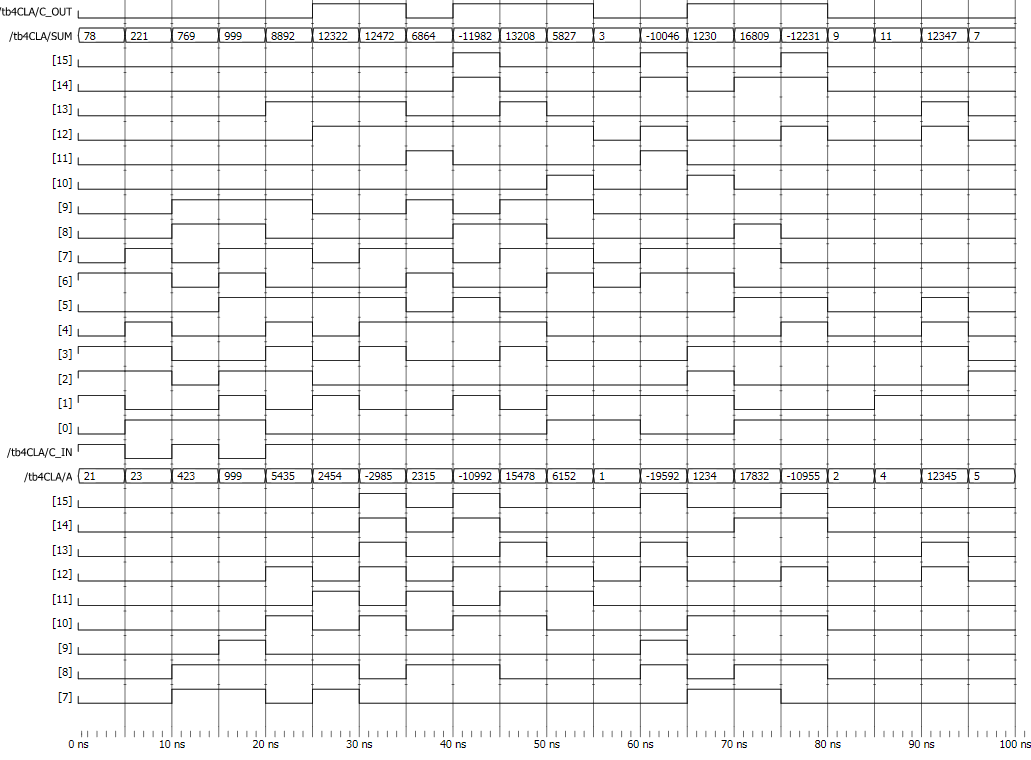
B = 16'd45;

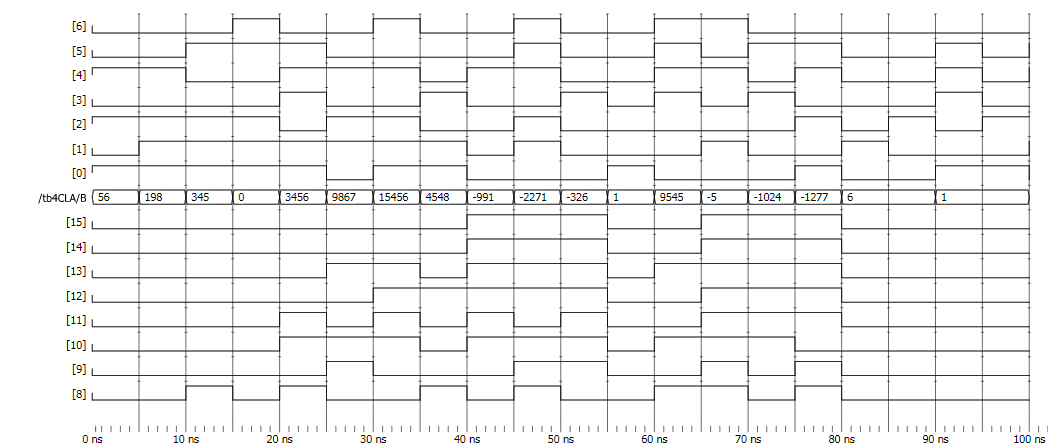
#5

$stop;

end

endmodule





캐리 룩 어헤드 가산기는 carrygu와 같은 특별한 모듈이 존재한다. 캐리 룩 어헤드 가산기는 리플 캐리 가산기의 시간에 대한 한계를 극복하는 모델이며 캐리는 순차적으로 전달받는 것이 아니라 carrygu에 의해 같은 시간에 한꺼번에 계산되어 출력된다. 구조를 정확히 보자면 carrygu는 두 번 쓰였는데 한 번은 adder4bit를 구현하기 위해서 쓰이고 또 다른 한 번은 CLA를 구현하기 위해 쓰였다. 4비트 가산기는 1비트 가산기 4개와 carrygu로 이루어져있고, CLA는 4비트 가산기 4개와 carrygu로 이루어져있다. 즉 같은 구조가 되풀이 되면서 1\*4\*4 구조를 구성했다. carrygu를 구현하기 위해서는 가산기들이 리플 캐리 가산기처럼 합과 캐리를 출력하는 것이 아니라 p와 g와 합을 출력한다. g = a & b 이고 p = a ^ b 를 각각 나타내며 g와 p는 c를 계산하기 위해 carrygu에서 쓰인다. C[n]=g[n-1] | (p[n-1] & c[n-1));로 표현되며 carrygu에서 cin과 g[0]~g[n], p[0]~p[n] 입력을 받는다면 c[1]~c[n+1]을 출력할 수 있다. 따라서 가산기가 아무리 커져도 모든 캐리값이 한번에 전달되며 계산은 carrygu가 계산을 끝내는 시간에서 한 단계가 지나면 완성된다.

위의 표를 보면 알 수 있듯이 음수의 덧셈도 구현할 수 있고 CLA에서 캐리가 출력되는 것도 표로 감지할 수 있었다. 그런데 그래프를 분석하던중 25~30ns 구간에서 입력값이 2454, 9867인데 캐리값이 1로 출력된 것을 확인할 수 있었는데 내 예상과 달라서 분석하기가 어려웠다. 분명 입력된 값의 14~15비트가 0이고 SUM도 마찬가지인데 어째서 캐리가 출력됐을까? 연산 자체가 틀리진 않아서 쉽게 발견할 수 없었고 값이 틀리지 않아서 왜 이런 경우가 발생했는지도 알 수 없었다. 아마 CLA 구간에서 c\_temp 값을 출력으로 하지 않고 c 값을 출력하면서 차이가 발생했던 것일까?