명지 공학인 윤리 서약서

보고서 및 논문 윤리 서약

1. 나는 보고서 및 논문의 내용을 조작하지 않겠습니다.
2. 나는 다른 사람의 보고서 및 논문의 내용을 내 것처럼 무단으로 복사하지 않겠습니다.
3. 나는 다른 사람의 보고서 및 논문의 내용을 참고하거나 인용할 시 참고 및 인용 형식을 갖추고 출처를 반드시 밝히겠습니다.
4. 나는 보고서 및 논문을 대신하여 작성하도록 청탁하지도 청탁받지도 않겠습니다.

나는 보고서 및 논문 작성 시 위법 행위를 하지 않고, 명지인으로서 또한 공학인으로서 나의 양심과 명예를 지킬 것을 약속합니다.

시험 윤리 서약

1. 나는 대리시험을 청탁하거나 청탁받지 않겠습니다.
2. 나는 허용되지 않은 교과서, 노트 및 타학생의 답안지 등을 보고 답안지를 작성하지 않겠습니다.
3. 나는 타인에게 답안지를 보여주지 않겠습니다.
4. 나는 감독관의 지시와 명령에 따라 시험 과정에 참여하겠습니다.

나는 시험에 위법 행위를 하지 않고, 명지인으로서 또한 공학인으로서 나의 양심과 명예를 지킬 것을 약속합니다.

2022년 10월 01일

서약자

(학번) 60171878

(성명) 허무혁 **(인)**

1. Verilog 소스코드

숙제 1. pfa.v 코드

```
module pfa(input a, input b, input ci, output p, output g, output s);



assign p = a ^ b;
assign g = a & b;
assign s = a ^ b ^ ci;

endmodule
```

숙제 1. cla_logic.v 코드

```
module cla_logic(input ci, input [3:0] p, input [3:0] g, output [3:1] C, output PG, output GG, output Co);



assign C[1] = g[0] | (p[0] & ci);
assign C[2] = g[1] | (p[1] & g[0]) | (p[1] & p[0] & ci);
assign C[3] = g[2] | (p[2] & g[1]) | (p[2] & p[1] & g[0]) | (p[2] & p[1] & p[0] & ci);
assign Co = 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] & ci);

assign PG = p[3] & p[2] & p[1] & p[0];
assign GG = g[3] | (p[3] & g[2]) | (p[3] & p[2] & g[1]) | (p[3] & p[2] & p[1] & g[0]);

endmodule
```

숙제 1. cla.v 코드

```
module cla4(input [3:0] a, input [3:0] b, input ci, output [3:0] S, output Co, output GG, output PG);
wire [3:0] g;
wire [3:0] p;
wire [3:1] C;

cla_logic cla0 (ci, p, g, C, PG, GG, Co);
pfa f0 (a[0], b[0], ci, p[0], g[0], S[0]);
pfa f1 (a[1], b[1], C[1], p[1], g[1], S[1]);
pfa f2 (a[2], b[2], C[2], p[2], g[2], S[2]);
pfa f3 (a[3], b[3], C[3], p[3], g[3], S[3]);

endmodule
```

숙제 1. tb_cla4.v 테스트벤치 코드

```
module tb_cla4;
reg [3:0] a;
```

```
reg [3:0] b;
reg ci;
wire [3:0] S;
wire Co;
wire PG;
wire GG;

initial begin
a = 4'b0000;
b = 4'b0000;
ci = 0;
#200;
a = 4'b1100;
b = 4'b0100;
#200;
a = 4'b1100;
b = 4'b0011;
#200;
a = 4'b1011;
b = 4'b0001;
$stop;
end

cla4 dut(a, b, ci, S, Co, GG, PG);
endmodule
```

숙제 2. cla16.v 코드

```
module cla16(input [15:0] a, input [15:0] b, input ci, output [15:0] S, output Co, output GGo, output PGo);
wire [3:0] GG;
wire [3:0] PG;
wire [3:1] C;



cla_logic cla1 (ci, PG[3:0], GG[3:0], C[3:1], PGo, GGo, Co);
cla4 u0 (a[3:0], b[3:0], ci, S[3:0], PG[0], GG[0]);
cla4 u1 (a[7:4], b[7:4], C[1],  S[7:4], PG[1], GG[1]);
cla4 u2 (a[11:8], b[11:8], C[2], S[11:8], PG[2], GG[2]);
cla4 u3 (a[15:12], b[15:12], C[3], S[15:12], PG[3], GG[3]);

endmodule
```

숙제 2. tb_cla16.v 테스트벤치 코드

```
module tb_cla16;
```

```
reg [15:0] a;
reg [15:0] b;
reg ci;
wire [15:0] S;
wire Co;
wire GGo;
wire PGo;

initial begin
a = 16'b0000_1100_0010_1000;
b = 16'b0100_1100_1011_1010;
ci = 0;
#200;
a = 16'b0000_0000_1010_1000;
b = 16'b0100_1100_1010_1010;
#200;
a = 16'b1100_1100_0010_1000;
b = 16'b0100_1100_1010_1000;
#200;
a = 16'b0110_0100_0110_1111;
b = 16'b0100_1100_1000_1000;
#200;
a = 16'b1101_1100_0010_1000;
b = 16'b1100_1100_1010_1000;
$stop;
end

cla16 dut(a, b, ci, S, Co, GGo, PGo);
endmodule
```

숙제 3. fa.v 코드

```
module fa(
    input a,
    input b,
    input ci,
    output s,
    output co
    );

    assign s = a ^ b ^ ci;
    assign co = ci & (a ^ b) | (a & b);



endmodule
```

숙제 3. add4.v 코드

```
module add4(input [3:0] a, input [3:0] b, input ci, output [3:0] s, output co);
```

```
    wire [3:1]    c;
    fa u0(a[0], b[0], ci, s[0], c[1]);
    fa u1(a[1], b[1], c[1], s[1], c[2]);
    fa u2(a[2], b[2], c[2], s[2], c[3]);
    fa u3(a[3], b[3], c[3], s[3], co);

endmodule
```

숙제 3. add16.v 코드

```
module add16(input [15:0] a, input [15:0] b, input ci, output [15:0] s, output co);
    wire [3:1]    c;

    add4 u0(a[3:0], b[3:0], ci, s[3:0], c[1]);
    add4 u1(a[7:4], b[7:4], c[1], s[7:4], c[2]);
    add4 u2(a[11:8], b[11:8], c[2], s[11:8], c[3]);
    add4 u3(a[15:12], b[15:12], c[3], s[15:12], co);

endmodule
```

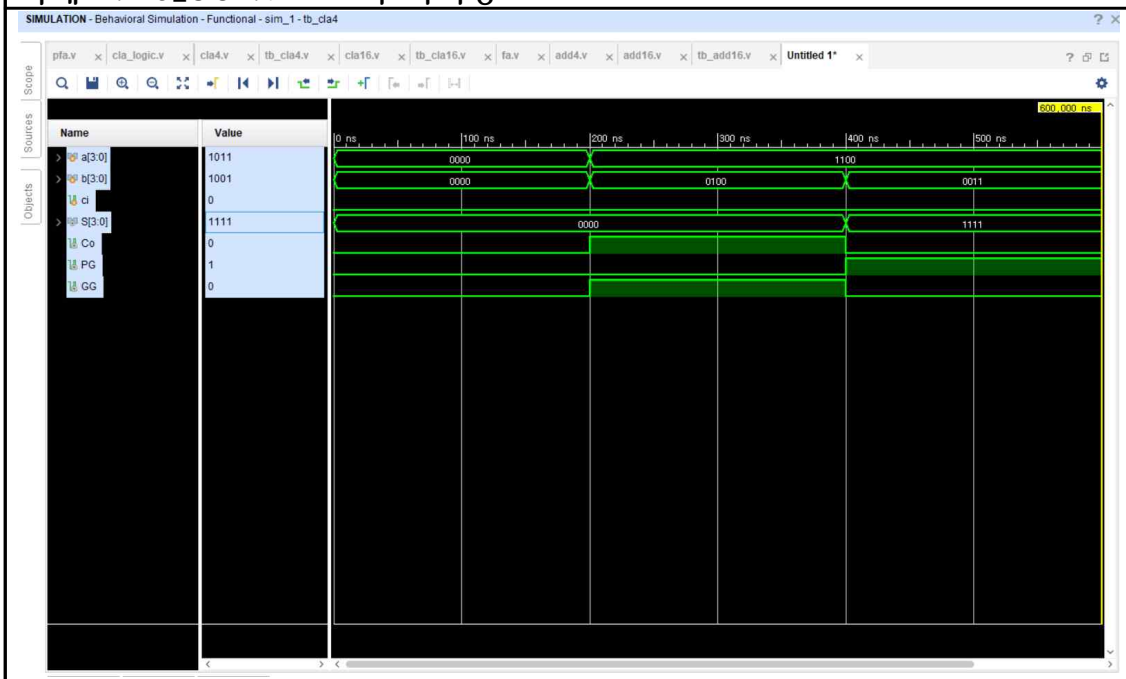
숙제 3. tb_add16.v 코드

```
module tb_add16;
reg [15:0] a;
reg [15:0] b;
reg ci = 0;
wire [15:0] S, s;
wire Co;
wire co;
wire GGo;
wire PGo;

initial begin
a = 16'b0000_1100_0010_1000;
b = 16'b0100_1100_1011_1010;
#200;
a = 16'b0000_0000_1010_1000;
b = 16'b0100_1100_1010_1010;
#200;
a = 16'b1100_1100_0010_1000;
b = 16'b0100_1100_1010_1000;
#200;
a = 16'b0110_0100_0110_1111;
b = 16'b0100_1100_1000_1000;
#200;
```

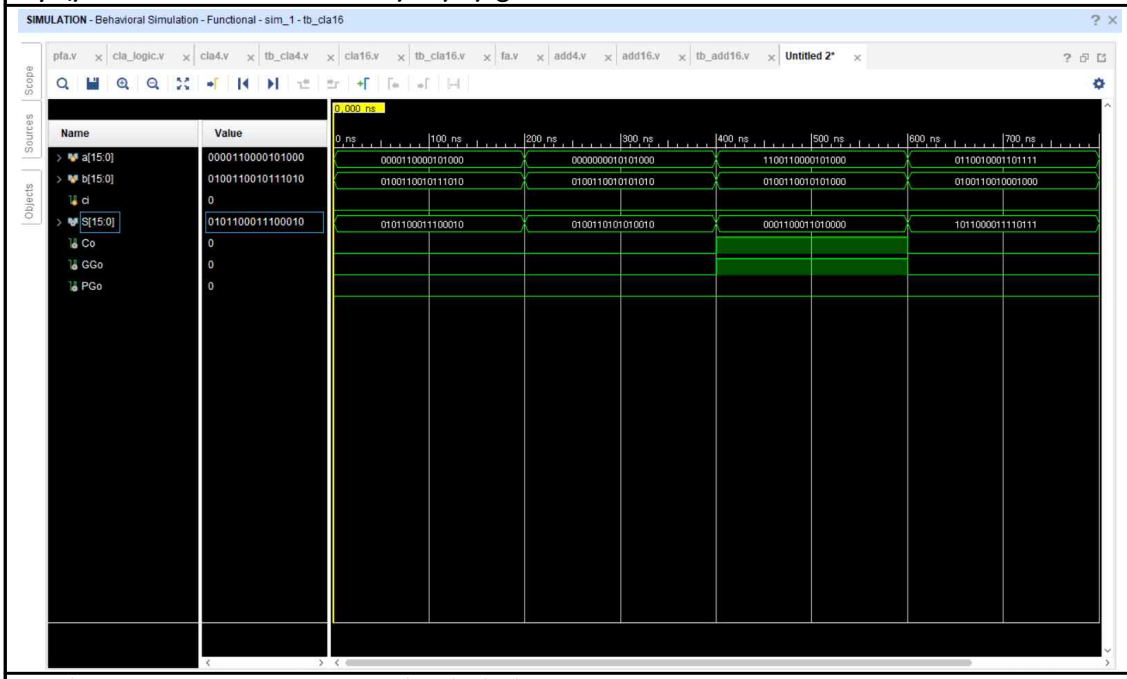
```
a = 16'b1101_1100_0010_1000;
b = 16'b1100_1100_1010_1000;
$stop;
end

add16 dut1(a, b, ci, s, co);
cla16 dut2(a, b, ci, S, Co, GGo, PGo);

endmodule
```

2. 타이밍도 캡쳐

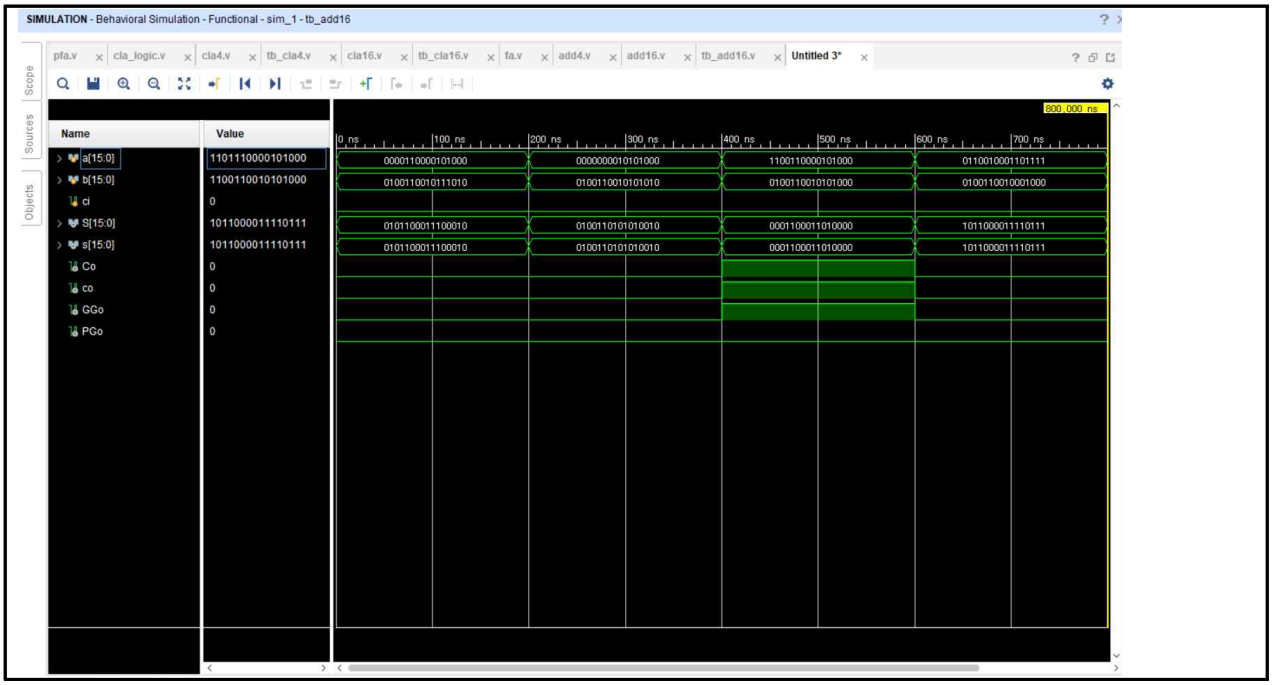
숙제 1. tb_cla4.v 코드의 타이밍도



숙제 2. tb_cla16.v 코드의 타이밍도



숙제 3. tb_add16.v 코드의 타이밍도



3. Discussion

숙제 1. Discussion (설계 방법)

숙제 1은 3개의 모듈을 사용하여 hierarchical 설계를 하는 것이 목표이다. 따라서 cla4.v을 top module로 삼아 pfa.v, cla_logic.v을 포함 시켰다. pfa.v 코드는 partial full adder의 기능을 수행한다. cla_logic.v 코드는 carry lookahead logic 기능을 수행한다.

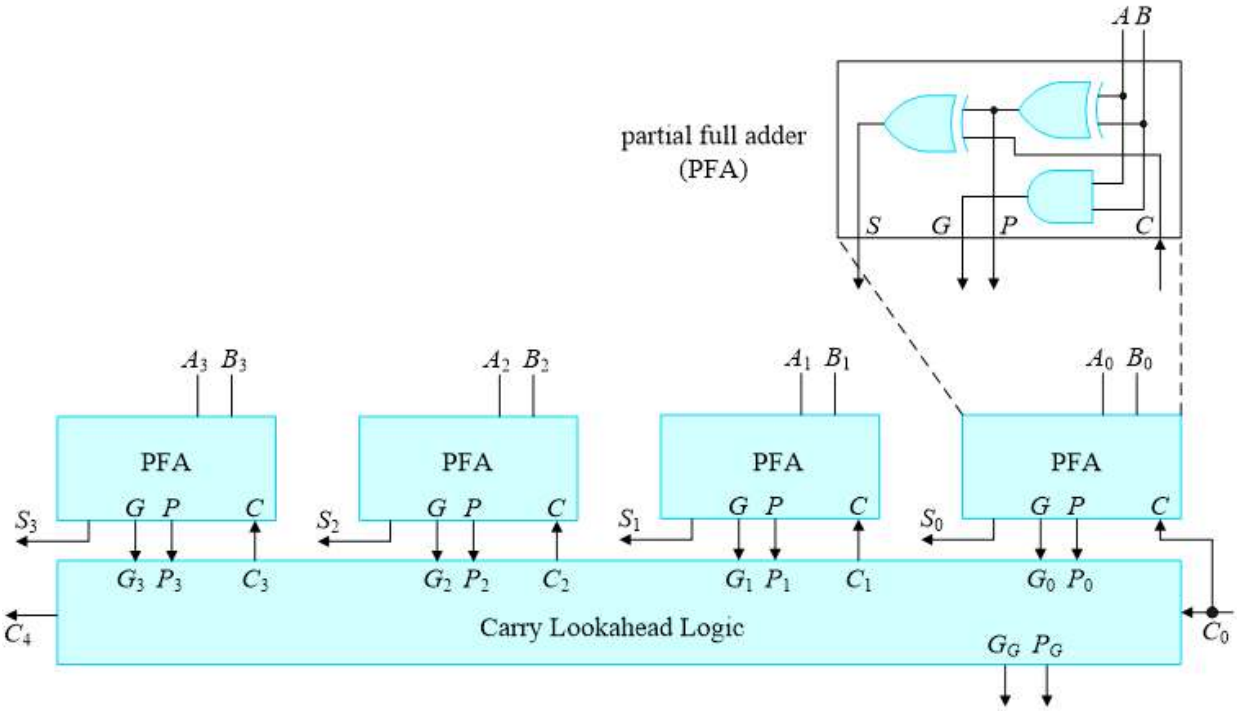


그림 1 출처. 디지털시스템 HW-CLA.ppt 파일

ppt의 사진을 참고하여 pfa를 설계하였다. 입력은 a, b, ci를 통하여 받아 게이트를 걸쳐 출력 p, g, s를 내보낸다. cla_logic의 경우 pf와 동일한 입력 ci와 pfa의 p, g, C의 출력을 입력으로 받는다. 이 때 p, g는 4bit, c는 3bit이다. cla_logic의 내부 논리회로를 거쳐 출력 PG, GG, Co를 통해 내보낸다.

$$C_1 = G_0 + P_0 C_0$$

$$C_2 = G_1 + P_1 C_1 = G_1 + P_1(G_0 + P_0 C_0) = G_1 + P_1 G_0 + P_1 P_0 C_0$$

$$C_3 = G_2 + P_2 C_2 = G_2 + P_2(G_1 + P_1 G_0 + P_1 P_0 C_0) = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$$

$$C_4 = G_3 + P_3 C_3 = G_3 + P_3(G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0)$$
$$= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$$

$$S_i = A_i \oplus B_i \oplus C_i = P_i \oplus C_i$$

그림 2. 출처 디지털 논리회로 한빛아카데미 282p

$$P_G = P_3 P_2 P_1 P_0 \,, \quad G_G = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0$$

그림 3. 출처 디지털 논리회로 한빛아카데미 283p

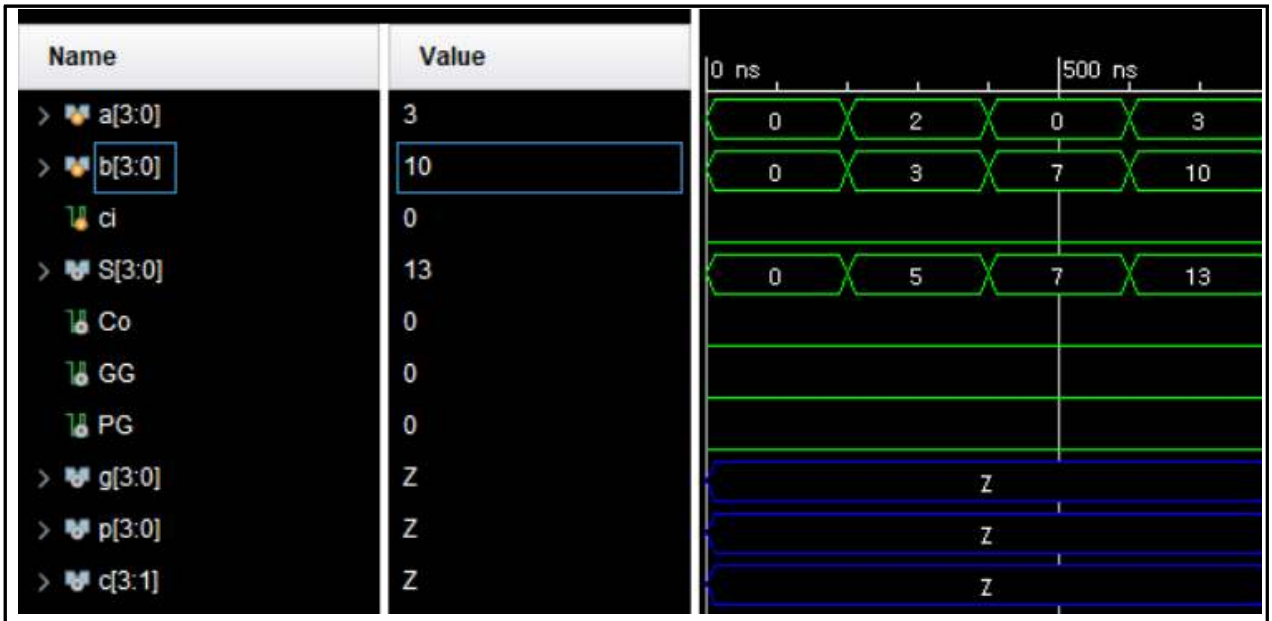
cla_logic의 회로구성에 필요한 수식검산은 한빛아카데미의 디지털 논리회로 교재를 통해 도움을 받았다. cla4의 top module은 1개의 cla_logic과 4개의 pfa를 사용한다. 이때 입력은 4bit a, b와 1bit ci 이며 출력은 4bit ci와 S, Co를 사용했다. 4개의 pfa와 1개의 cla의 연결은 4bit의 wire g, p 와 wire C를 사용하였다.

```
cla_logic cla0 (ci, p, g, C, PG, GG, Co);
pfa f0 (a[0], b[0], ci, p[0], g[0], S[0]);
pfa f1 (a[1], b[1], C[1], p[1], g[1], S[1]);
pfa f2 (a[2], b[2], C[2], p[2], g[2], S[2]);
pfa f3 (a[3], b[3], C[3], p[3], g[3], S[3]);
```

다음과 같이 선언하여 모듈들을 엮어 최종회로를 구성하였다.

숙제 1. Discussion (설계시 발생한 문제들)

cla4를 구성하는데 여러 문제들이 발생했다. 처음 발생한 문제는 pfa의 출력 p, g와 cla_logic의 입력 p, g의 변수를 같은 이름을 쓰지 않아 회로 구성에 문제가 발생했다. 해당 문제는 지난 수업때 add4의 예제 코드를 보며 맞게 수정하였다. 두 번째 발생한 문제는 cla_logic의 출력 PG, GG와 cla4의 출력 PG, GG를 같은 이름을 쓰지 않아 회로가 꼬였다. 해당 문제는 ppt파일의 회로도를 보고 맞게 수정하였다. 세 번째 발생한 문제는 tb_cla4의 타이밍도에 Z값이 찍히는 변수들이 있었다.



여기서 g, p, c 값에 Z값이 찍힌 것은 cla4 dut(a, b, ci, S, Co, GG, PG); 명령을 통하여 dut 파일을 불러왔지만 tb_cla4의 코드상에 wire [3:0] g, p, c; 선언이 되어있었기 때문이다. 해당 g, p, c는 cla4 dut(a, b, ci, S, Co, GG, PG); 선언 안에 들어가 있지 않아 끊어진 선 처리가 되었던 것으로 추측한다. 따라서 코드상에서 지워 최종 타이밍도에는 표시되지 않게 처리하였다.

숙제 2. Discussion (설계 방법)

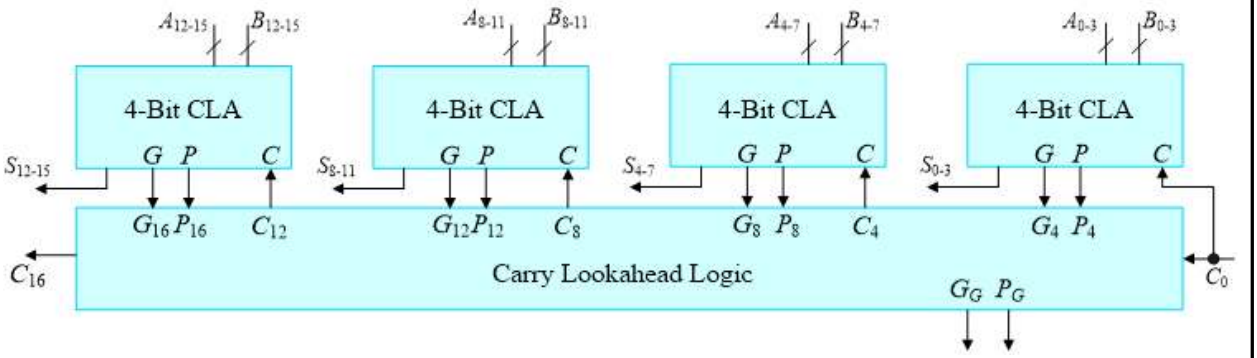


그림 4. 출처. 디지털시스템 HW-CLA.ppt 파일

숙제 2는 숙제 1에서 설계한 4bit cla와 cal_logic을 사용하여 16bit cla를 설계하는 것이다. 해당 문제는 지난 수업때 addsub4를 사용하여 addsub8을 설계한 방식과 매우 유사했다. 먼저 16bit의 입력 a, b와 1bit의 ci를 선언한 뒤 16bit의 출력 s와 1bit의 Co, GGo, PGo를 선언하였다. 4개의 cla4와 1개의 cla_logic은 4bit의 wire GG, PG와 3bit의 C를 사용하여 엮었다.

cla_logic cla1 (ci, PG[3:0], GG[3:0], C[3:1], PGo, GGo, Co);
cla4 u0 (a[3:0], b[3:0], ci, S[3:0], PG[0], GG[0]);
cla4 u1 (a[7:4], b[7:4], C[1], S[7:4], PG[1], GG[1]);
cla4 u2 (a[11:8], b[11:8], C[2], S[11:8], PG[2], GG[2]);
cla4 u3 (a[15:12], b[15:12], C[3], S[15:12], PG[3], GG[3]);

다음과 같이 선언하여 모듈들을 엮어 최종회로를 구성하였다.

숙제 2. Discussion (설계시 발생한 문제들)

숙제 2에서 발생한 문제는 cla4의 출력 PG, GG와 다른 변수를 cla16에서 선언해주어야 한다는 것이다. 이유는 모듈들을 엮을시 최종 출력에서 cla4에서 빠져나오는 출력과 cla_logic에서 빠져나오는 출력이 합선될 문제가 있는 것으로 추측한다. 따라서 cla16에서는 PGo와 GGo를 선언하여 겹치지 않도록 설계하였다. 출력 Co의 경우 cla4에서도 같은 Co 변수를 쓰지만 1bit의 값이고 최종회로도에서

Co는 합선될 가능성이 없기에 그렇다고 추측한다. 이 이외에 숙제2에서 특별하게 발생한 문제는 없다.

숙제 3. Discussion (설계 방법)

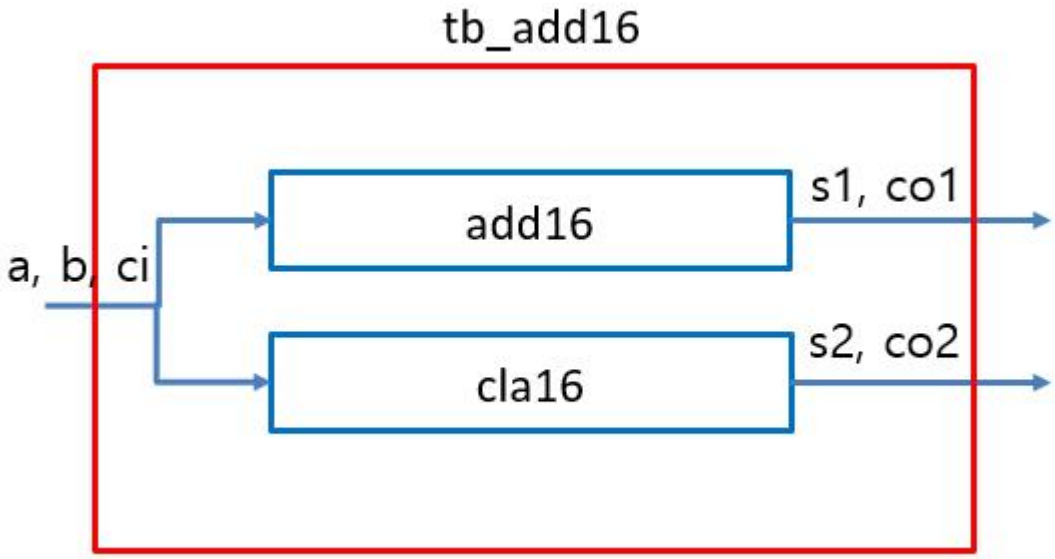


그림 5. 출처. 디지털시스템 HW-CLA.ppt 파일

숙제 3은 add4를 사용하여 add16을 만들고 tb_add16안에 add16과 cla16을 선언하여 출력을 확인하는 과제이다. add16은 16bit의 입력 a, b와 1bit의 입력 ci를 입력 받고 16bit의 출력 s, 1bit의 출력 co를 내보낸다. 내부 4개의 add4 모듈들은 3bit의 wire c를 통하여 엮어주었다.

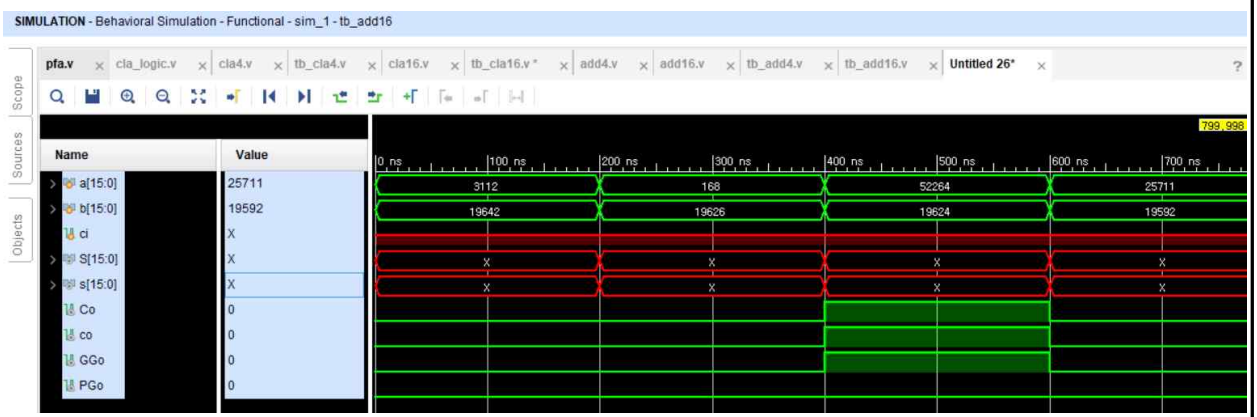
add4 u0(a[3:0], b[3:0], ci, s[3:0], c[1]);
add4 u1(a[7:4], b[7:4], c[1], s[7:4], c[2]);
add4 u2(a[11:8], b[11:8], c[2], s[11:8], c[3]);
add4 u3(a[15:12], b[15:12], c[3], s[15:12], co);

테스트벤치 tb_add16은 기존 tb_cla16 코드를 복사해 add16 dut를 추가하여 검증하였다.

add16 dut1(a, b, ci, s, co);
cla16 dut2(a, b, ci, S, Co, GGo, PGo);

숙제 3. Discussion (설계시 발생한 문제들)

add16회로를 구성하는데 발생한 문제는 없었다. 처음 작성을 시작할 때 add16과 cla16의 출력들이 합선되지 않도록 할 것을 생각했기 때문이다. 따라서 가장 걱정한 출력이 엉키는 문제는 발생하지 않았다. 그러나 테스트 벤치 상에서 X에러 unknown값이 찍히는 상황이 발생하였다.



pfa, cla_logic, cla4 등등의 코드의 출력을 여럿 점검하는 데에 많은 시간을 허비한 결과 찾은 문제는 dut를 선언해주는 과정에서

add16 dut(a, b, ci, s, co);

cla16 dut(a, b, ci, S, Co, GGo, PGo);

add16과 cla16 뒤에 같은 dut란 이름을 선언했기 때문이다. 당시에는 dut를 붙이는 것이 dut를 선언하는 것이라고 잘못알고 있었기에 이름을 바꾸어 해결한다는 방법을 모르고 있었다.

add16 dut1(a, b, ci, s, co);

cla16 dut2(a, b, ci, S, Co, GGo, PGo);

다음과 같이 dut1과 dut2라는 이름을 다르게 선언해주니 해당 문제가 해결되었다.

위 문제들 이외에 특기할 만한 문제는 발생하지 않았다.