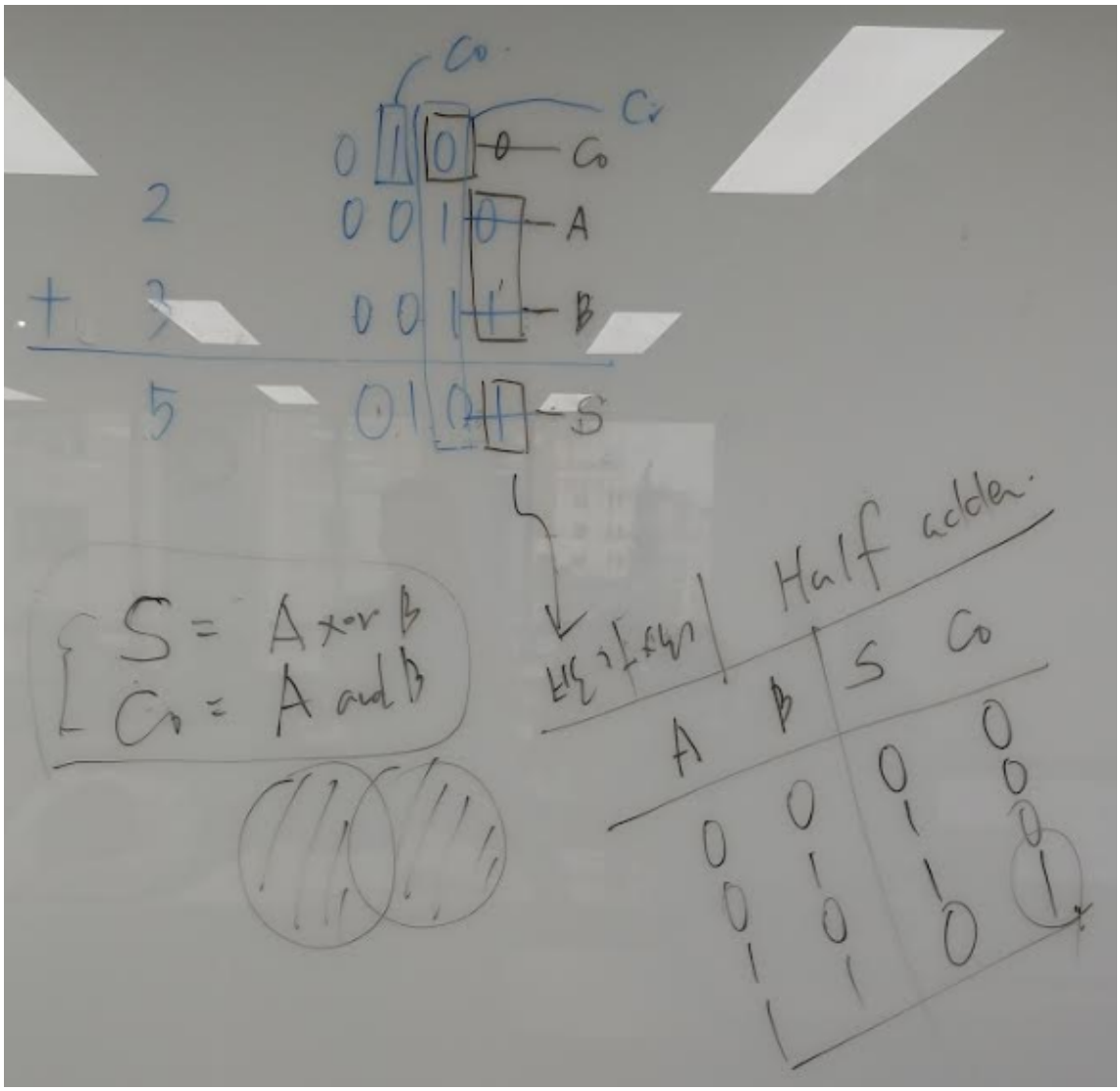
논리회로

논리회로 = 조합회로(combinational logic)
+ 순차회로(sequential logic)

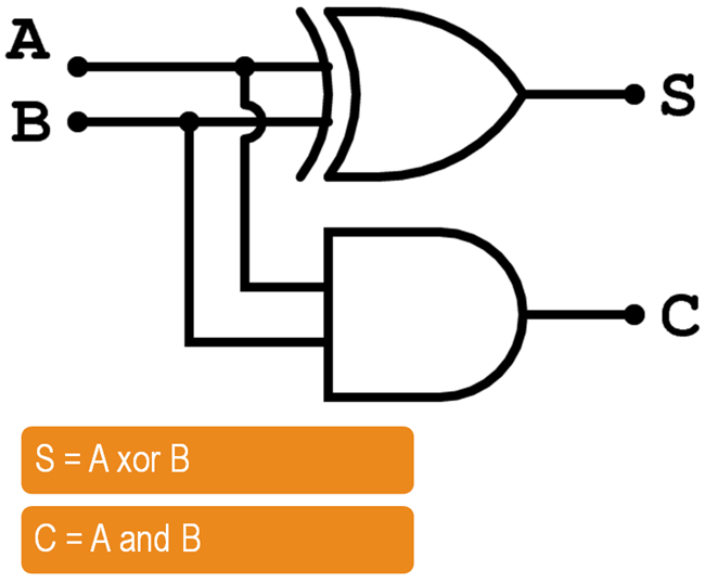
&, |, ~, ^, +, -, *, /, >, >=, <, <=, ==, !=, <<, >> : ALU

덧셈기 구현

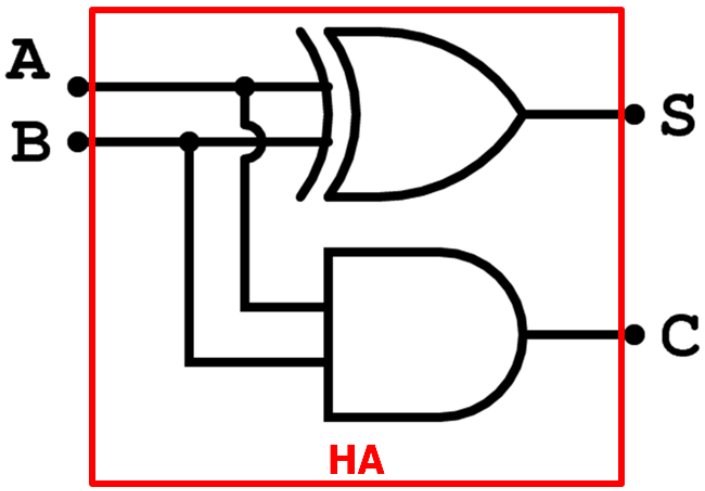
반가산기 알고리즘



반가산기 회로



module HA



module HA 구현 (게이트 수준 구현, 모듈 정의하기)

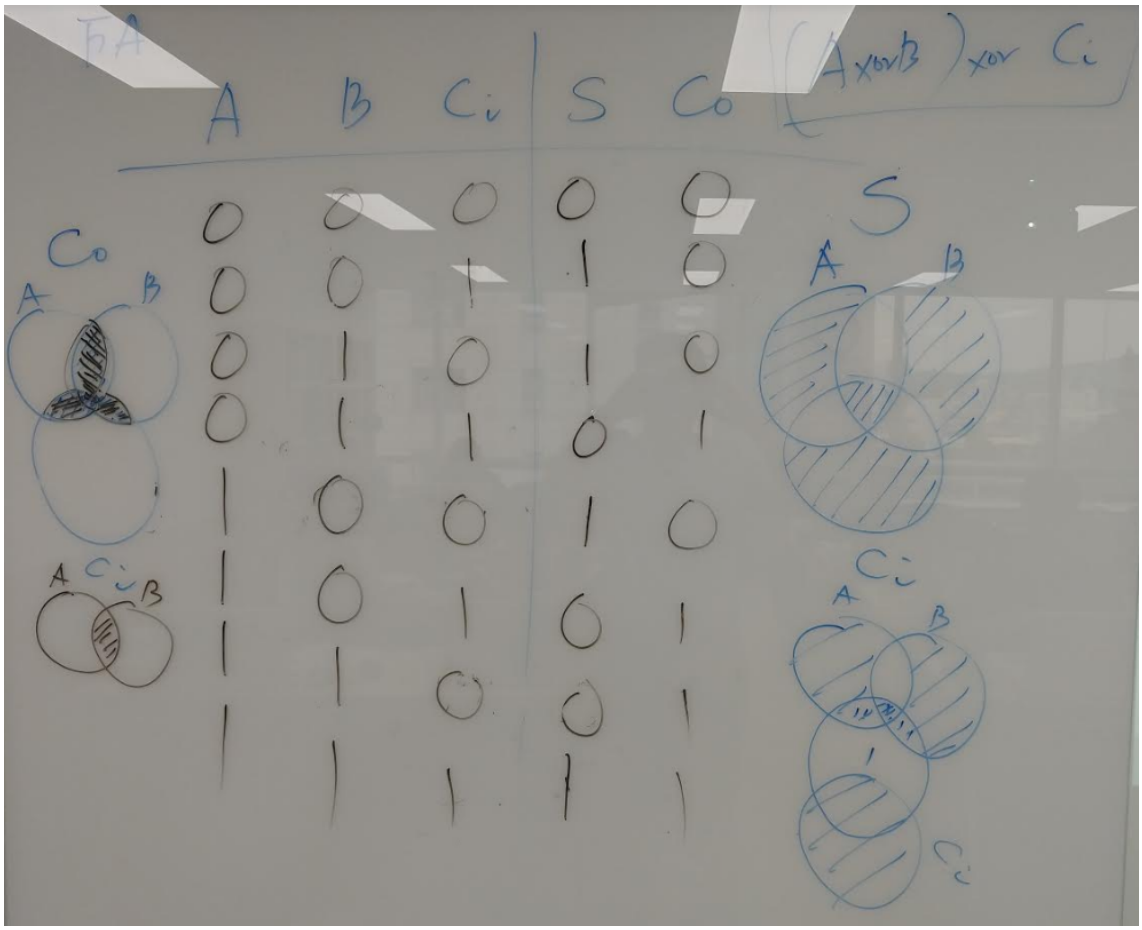
HA.v

```
module HA(
    input A,
    input B,
    output S,
    output C
    );

    assign S = A ^ B;
    assign C = A & B;

endmodule
```

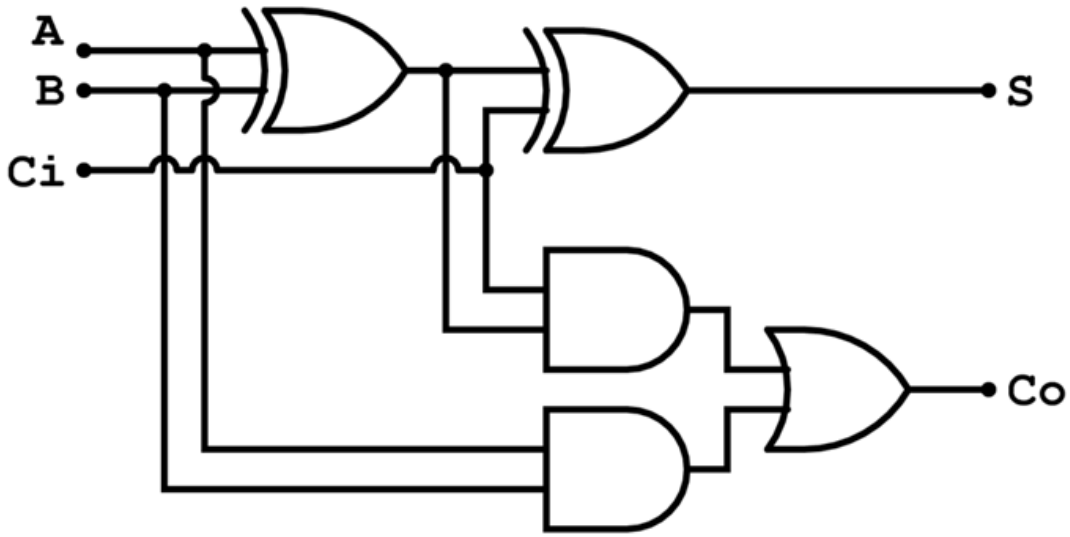
다음 작업을 수행합니다.

1. Comb_logic 프로젝트에 [Add Sources] 메뉴를 이용하여 HA.v 파일 추가
2. 모듈 이름은 HA, 포트는 A, B, S, C로 구성하며 A, B는 input, S, C는 output으로 설정
3. HA.v 파일을 마우스 오른쪽 버튼을 누른 후, [Set as Top] 모듈로 설정합니다.
4. 합성을 진행합니다.
5. [Layout]--[I/O Planning] 메뉴를 선택합니다. 핀을 다음과 같이 할당합니다.
   A(V16)
   B(V17)
   S(U16)
   C(E19)
6. 전압은 3.3V로 합니다.
7. 구현을 합니다.
8. 비트스트림을 생성한 후, 보드에 다운로드 받아 테스트합니다.

전가산기 알고리즘

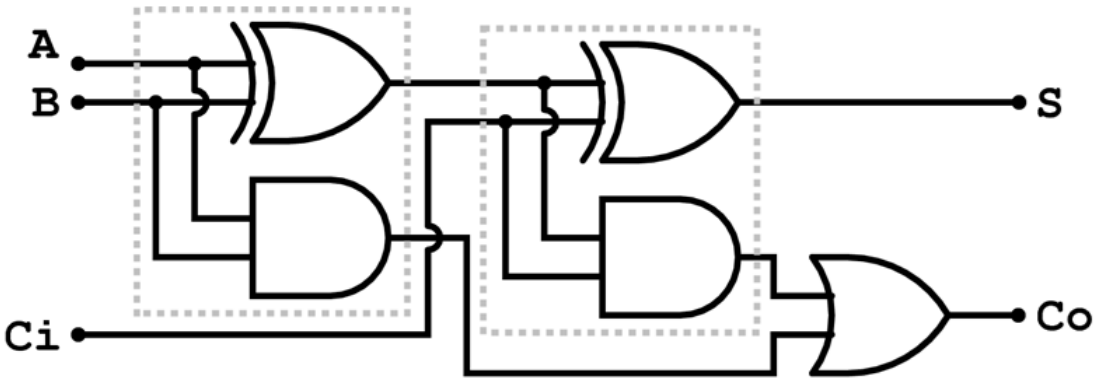


전가산기 회로



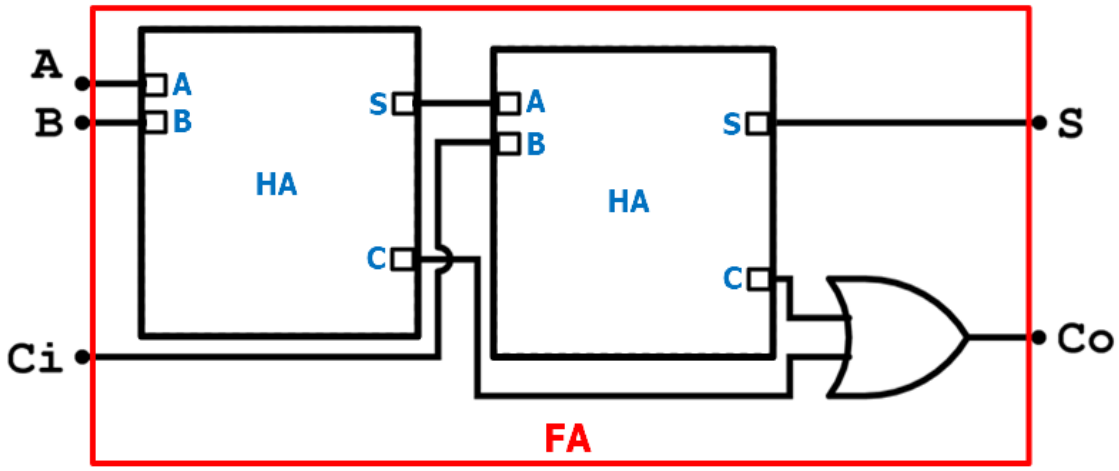
S = (A xor B) xor Ci

Co = (A and B) or (Ci and (A xor B))

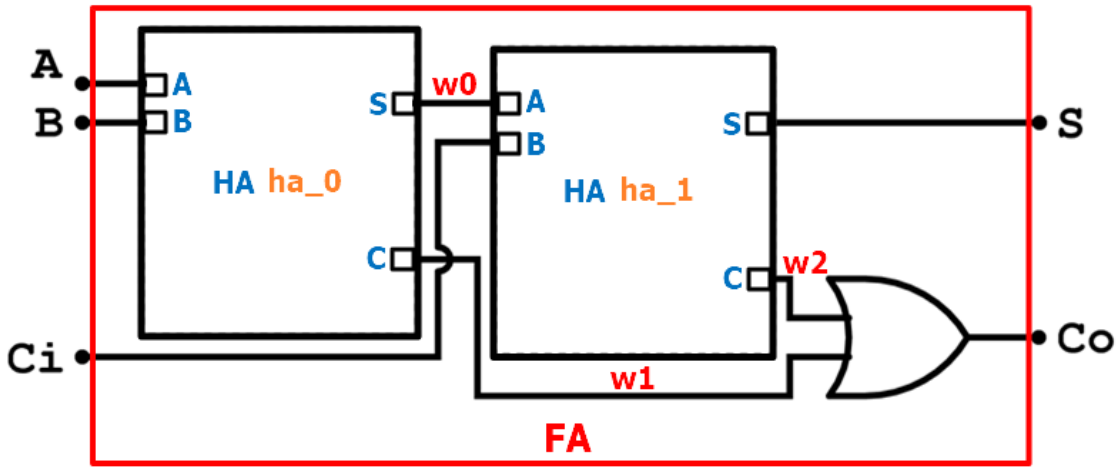


전가산기 구현

모듈 경계 정하기



전선 이름 정하기



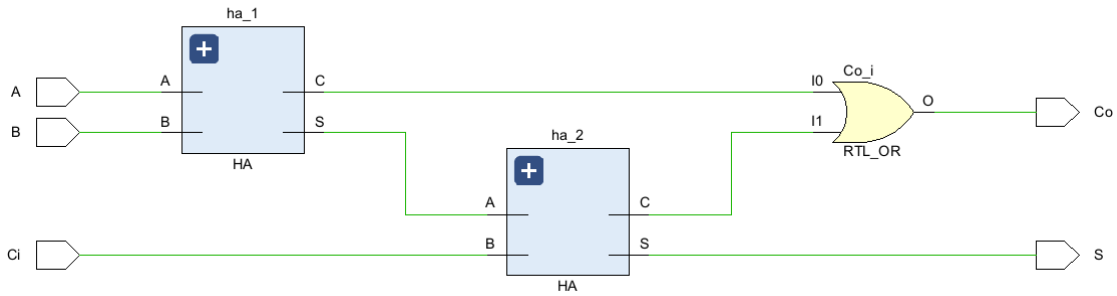
FA 구현하기(구조적 구현, 모듈 사용하기)
내부 와이어 선언, 와이어 연결

FA.v
module FA(input A, input B, input Ci, output S, output Co

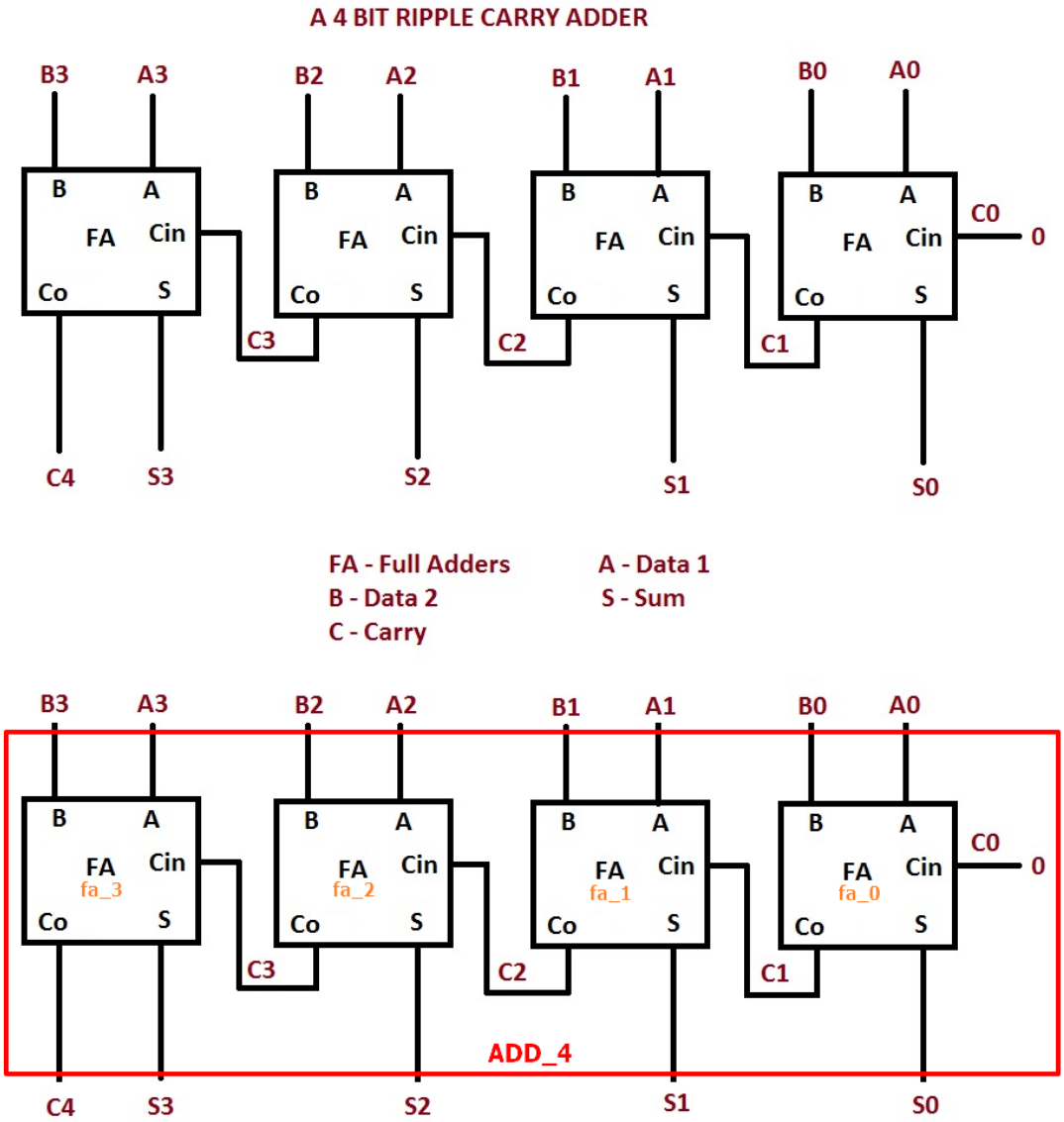
```
    );

    wire w1, w2, w3;

    HA ha_1(.A(A), .B(B), .S(w1), .C(w2));
    HA ha_2(.A(w1), .B(Ci), .S(S), .C(w3));
    assign Co = w2 | w3;

endmodule
```

다음 작업을 수행합니다.

1. Comb_logic 프로젝트에 [Add Sources] 메뉴를 이용하여 FA.v 파일 추가
2. 모듈 이름은 FA, 포트는 A, B, Ci, S, Co로 구성하며 A, B, Ci는 input, S, Co는 output으로 설정
3. FA.v 파일을 마우스 오른쪽 버튼을 누른 후, [Set as Top] 모듈로 설정합니다.
4. 합성을 진행합니다.
5. [Layout]--[I/O Planning] 메뉴를 선택합니다. 핀을 다음과 같이 할당합니다.
   A(V16)
   B(V17)
   Ci(R12)
   S(U16)
   Co(E19)
6. 구현을 합니다.
7. 비트스트림을 생성한 후, 보드에 다운로드 받아 테스트합니다.



덧셈기 4 비트(버스 선언과 사용)

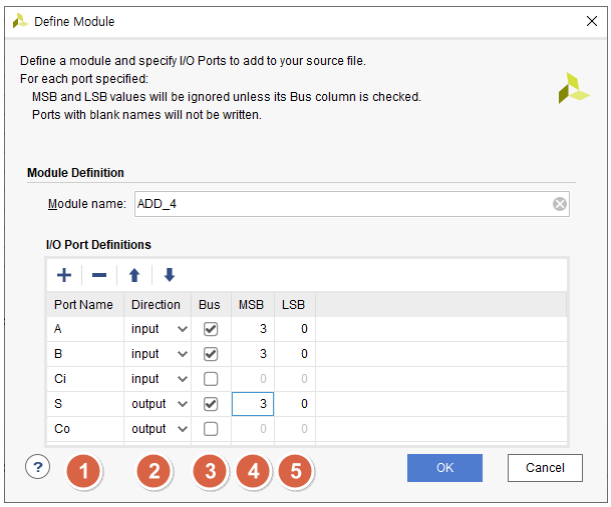


ADD_4.v
module ADD_4(input [3:0] A, input [3:0] B, input Ci, output [3:0] S, output Co); wire [2:0] w;

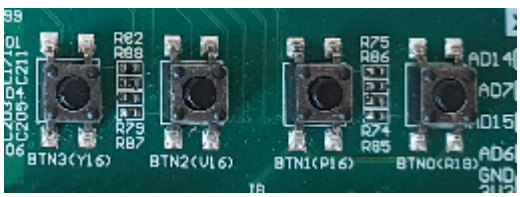
```
    FA fa_3(.A(A[3]), .B(B[3]), .Ci(w[2]), .S(S[3]), .Co(Co));
    FA fa_2(A[2], B[2], w[1], S[2], w[2]);
    FA fa_1(A[1], B[1], w[0], S[1], w[1]);
    FA fa_0(A[0], B[0], Ci, S[0], w[0]);

endmodule
```

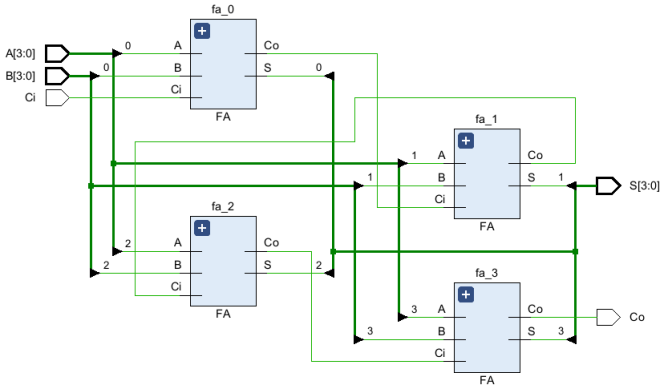
다음 작업을 수행합니다.

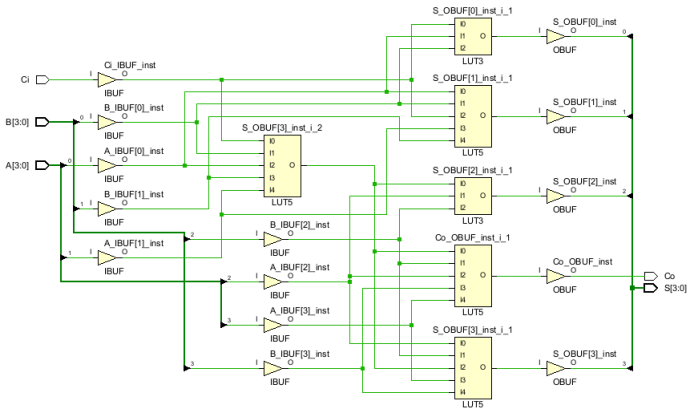
1. Comb_logic 프로젝트에 [Add Sources] 메뉴를 이용하여 ADD_4.v 파일 추가
2. 모듈 이름은 ADD_4, 포트는 A, B, Ci, S, Co로 구성하며 A, B, Ci는 input, S, Co는 output으로 설정, A, B, S은 4비트 버스로 설정



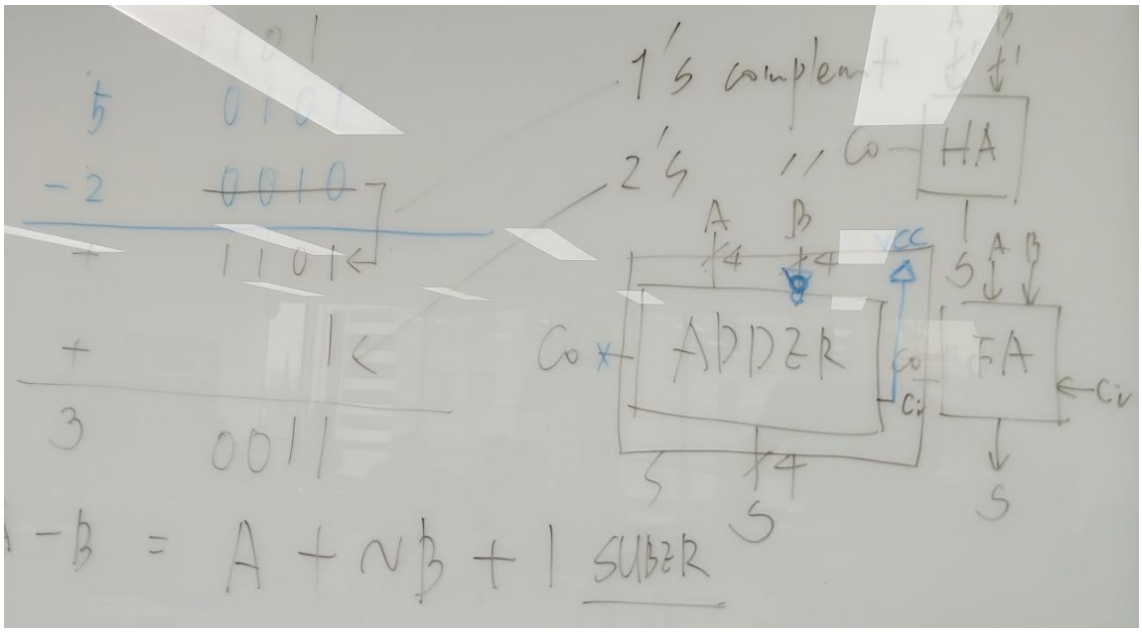
3. ADD_4.v 파일을 마우스 오른쪽 버튼을 누른 후, [Set as Top] 모듈로 설정합니다.
4. 합성을 진행합니다.
5. 핀을 다음과 같이 할당합니다.
   A3~A0(W17,W16,V16,V17)
   B3~B0(W13,W14,V15,W15)
   Ci(R2)
   S3~S0(V19,U19,E19,U16)
   Co(W18)
6. 구현을 합니다.
7. 비트스트림을 생성한 후, 보드에 다운로드 받아 테스트합니다.



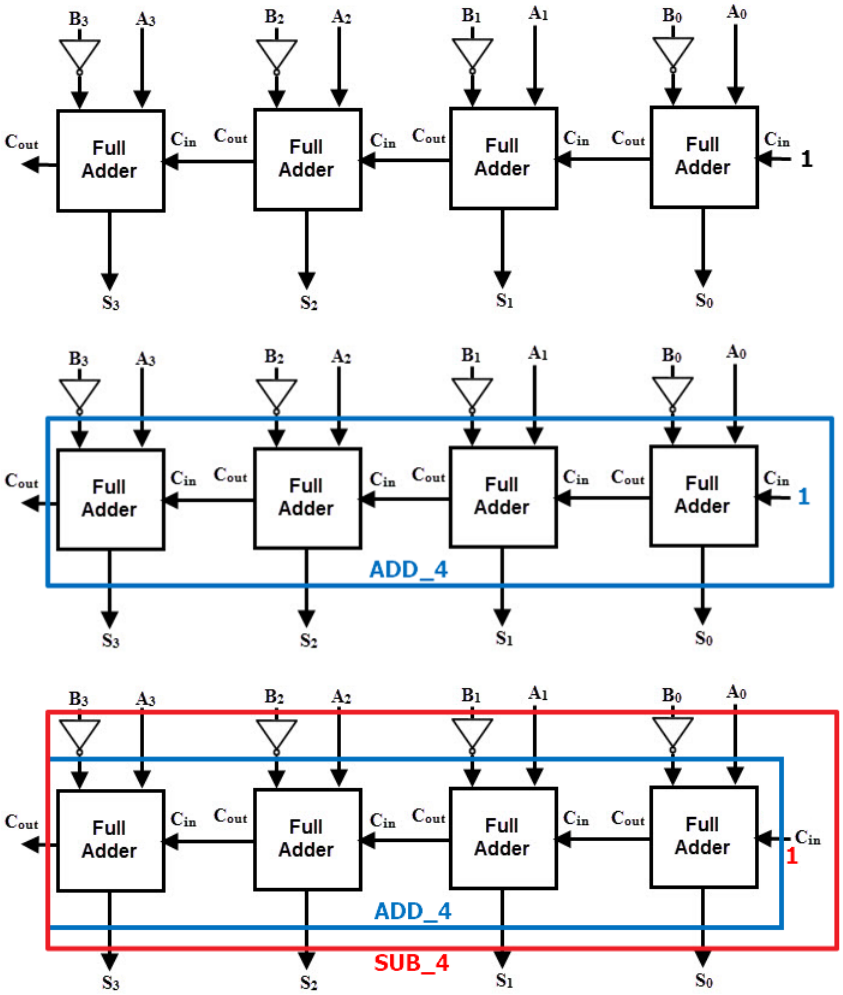

fa_0
fa_1
fa_2
fa_3
FA
A[3:0]
B[3:0]
Ci
S[3:0]
Co


Ci_IBUF_inst
B_IBUF[0]_inst
A_IBUF[0]_inst
B_IBUF[1]_inst
A_IBUF[1]_inst
B_IBUF[2]_inst
A_IBUF[2]_inst
A_IBUF[3]_inst
B_IBUF[3]_inst
IBUF
S_OBUF[3]_inst_i_2
LUT5
S_OBUF[0]_inst_i_1
S_OBUF[1]_inst_i_1
S_OBUF[2]_inst_i_1
S_OBUF[3]_inst_i_1
LUT3
Co_OBUF_inst_i_1
S_OBUF[0]_inst
S_OBUF[1]_inst
S_OBUF[2]_inst
S_OBUF[3]_inst
Co_OBUF_inst
OBUF
Ci
B[3:0]
A[3:0]
Co
S[3:0]

감산기 알고리즘


1's complent
2's
HA
ADDER
FA
SUBER

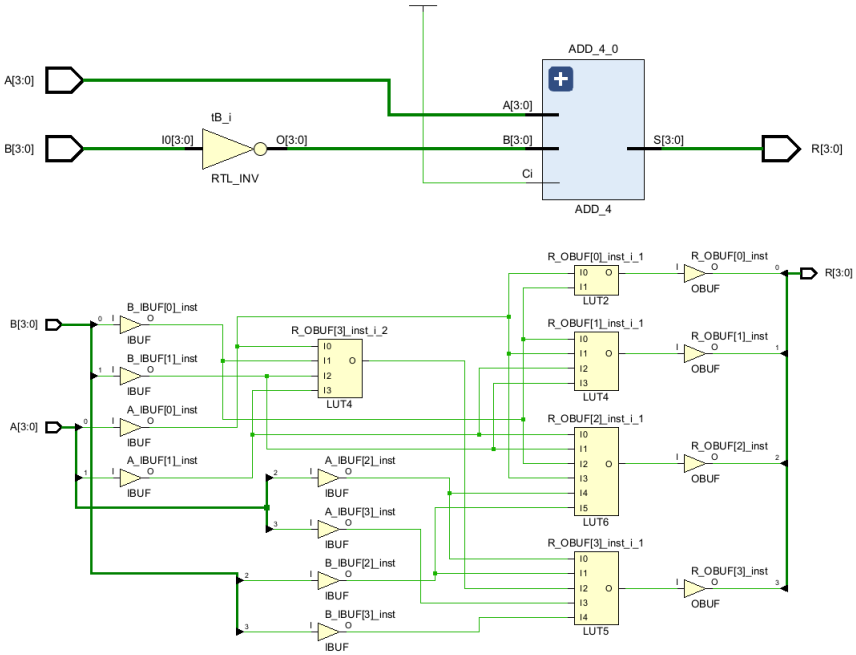
감산기 구현




```
module SUB_4(
    input [3:0] A,
    input [3:0] B,
    output [3:0] R
    );
    wire [3:0] tB;
    assign tB = ~B;

    ADD_4 ADD_4_0(.A(A), .B(tB), .S(R), .Ci(1));

endmodule
```

다음 작업을 수행합니다.

1. Comb_logic 프로젝트에 [Add Sources] 메뉴를 이용하여 SUB_4.v 파일 추가
2. 모듈 이름은 SUB_4, 포트는 A, B, R로 구성하며 A, B는 input, R은 output으로 설정, A, B, R은 4비트 버스로 설정
3. SUB_4.v 파일을 마우스 오른쪽 버튼을 누른 후, [Set as Top] 모듈로 설정합니다.
4. 합성을 진행합니다.
5. 핀을 다음과 같이 할당합니다.
   A3~A0(W17,W16,V16,V17)
   B3~B0(W13,W14,V15,W15)
   R3~R0(V19,U19,E19,U16)
6. 구현을 합니다.
7. 비트스트림을 생성한 후, 보드에 다운로드 받아 테스트합니다.



비교기 알고리즘

if(A==B)
if(A!=B)
if(A>B)
if(A>=B)
if(A<B)
if(A<=B)

if(A==B) <=> A-B == 0
if(A!=B) <=> A-B != 0
if(A>B) <=> A-B > 0
if(A>=B) <=> A-B >= 0
if(A<B) <=> A-B < 0
if(A<=B) <=> A-B <= 0

A-B == 0 <=> A-B=R==0 <=> R[3]==0이고 R[2]==0이고 R[1]==0이고 R[0]==0
A-B != 0 <=> A-B=R!=0 <=> R[3]==1거나 R[2]==1거나 R[1]==1거나 R[0]==1
A-B > 0 <=> A-B=R>0 <=> R[3]==0이고(R[2]==1거나R[1]==1거나R[0]==1)
A-B >= 0 <=> A-B=R>=0 <=> R[3]==0
A-B < 0 <=> A-B=R<0 <=> R[3]==1
A-B <= 0 <=> A-B=R<=0 <=> R[3]==1거나(R[2]==0이고 R[1]==0이고 R[0]==0)

R[3]==0이고 R[2]==0이고 R[1]==0이고 R[0]==0 <=>
T_equ = ~R[3] & ~R[2] & ~R[1] & ~R[0] = ~(R[3] | R[2] | R[1] | R[0])

R[3]==1거나 R[2]==1거나 R[1]==1거나 R[0]==1 <=>
T_neq = R[3] | R[2] | R[1] | R[0] = ~T_equ

R[3]==0이고(R[2]==1거나R[1]==1거나R[0]==1) <=>
T_gth = ~R[3] & (R[2] | R[1] | R[0])

R[3]==0 <=>
T_gte = ~R[3]

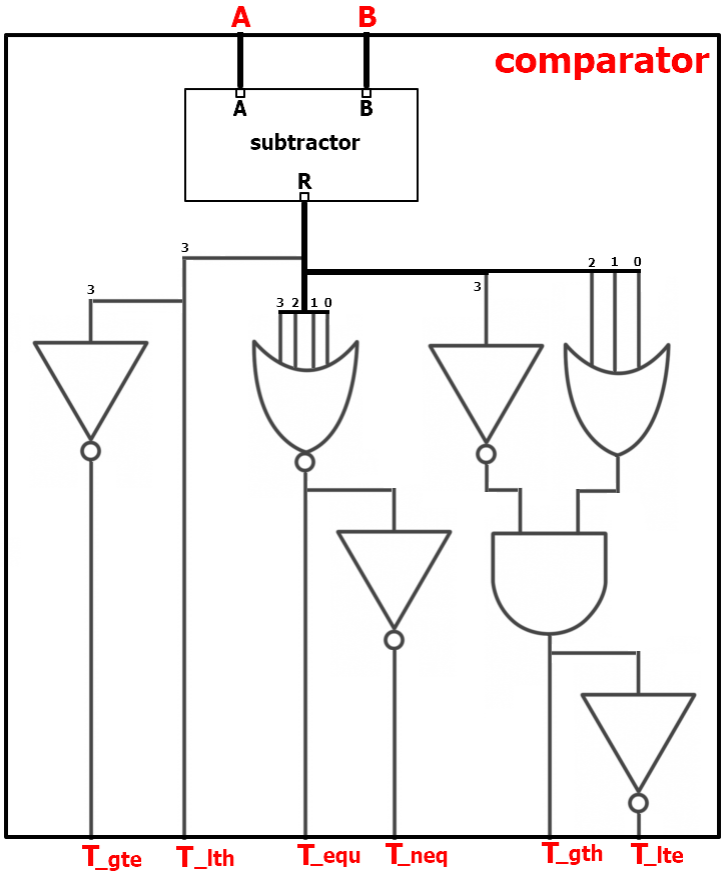
R[3]==1 <=>
T_lth = R[3] = ~T_gte

R[3]==1거나(R[2]==0이고 R[1]==0이고 R[0]==0) <=>

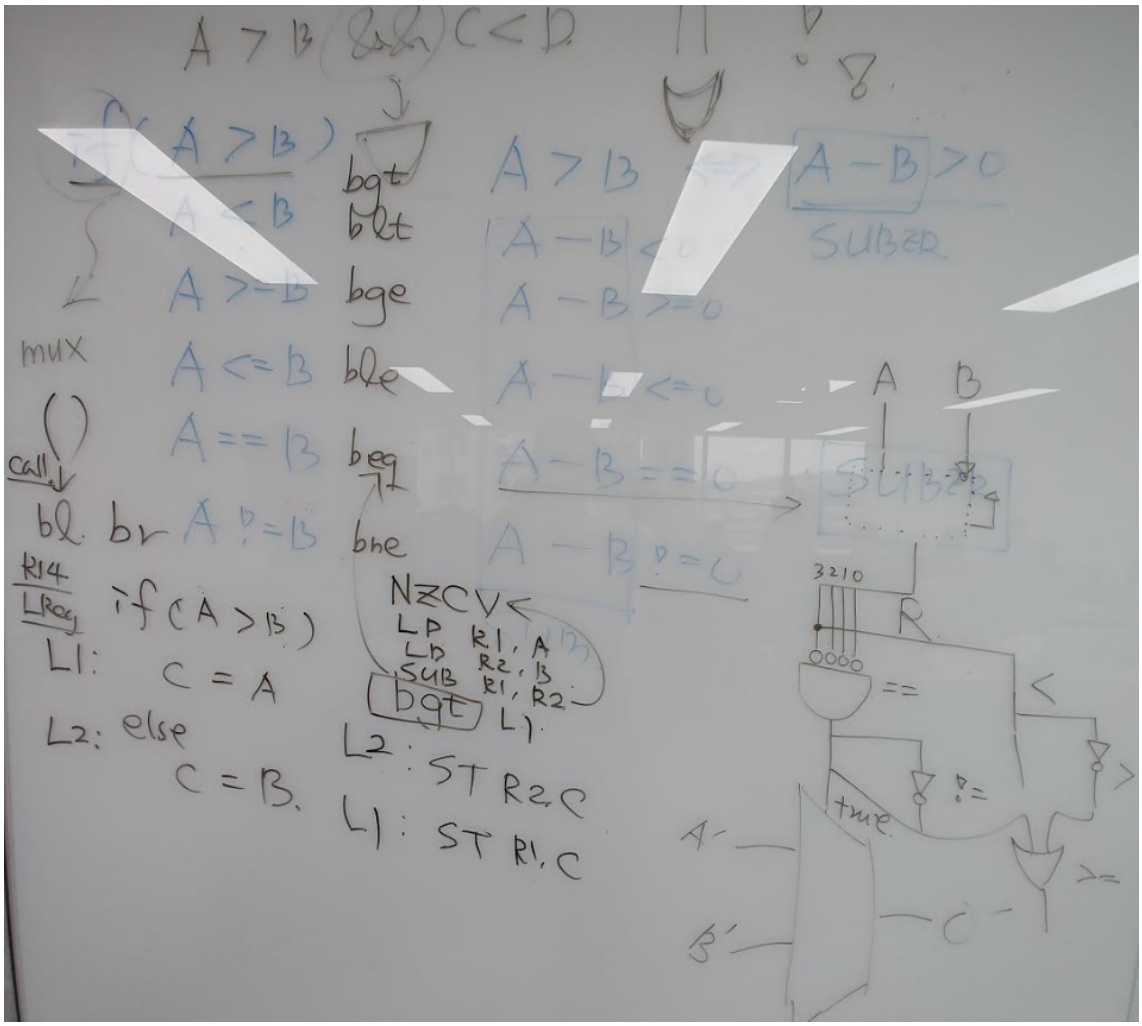
T_lte = R[3] | (~R[2] & ~R[1] & ~R[0]) = ~T_gth

R[3]==0이고 R[2]==0이고 R[1]==0이고 R[0]==0 이면 T_equ = 1

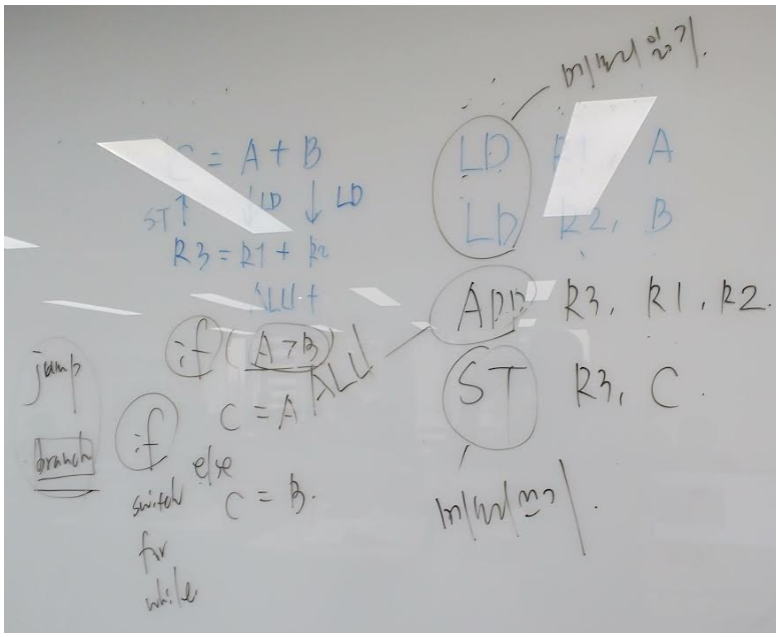
비교기 회로



비교기와 if문


A > B
C < D
bgt
blt
bge
ble
beq
bne
mux
call
bl br
R14
LReg
if (A > B)
L1: C = A
L2: else
C = B.
NZCV
LD R1, A
LD R2, B
SUB R1, R2
bgt L1
L2: ST R2, C
L1: ST R1, C
A - B > 0
SUBER.
A - B < 0
A - B >= 0
A - B == 0
A - B != 0
A
B
3210
R
==
<
!=
>
true
>=
A'
B'
C

어셈블리 언어의 종류



비교기 구현

```
module CMP_4(
    input [3:0] A, input [3:0] B,
    output Tequ, output Tneq, output Tgth,
    output Tlth, output Tgte, output Tlte);

    wire [3:0] tR;
    SUB_4 SUB_4_0(.A(A), .B(B), .R(tR));
    assign Tequ = ~(tR[3]|tR[2]|tR[1]|tR[0]);// Tequ
    assign Tneq = ~Tequ;// Tneq
    assign Tgth = ~tR[3] & (tR[2]|tR[1]|tR[0]);
    assign Tlth = tR[3];
    assign Tgte = ~tR[3];
    assign Tlte = Tlth | Tequ;

endmodule
```

다음 작업을 수행합니다.

1. Comb_logic 프로젝트에 [Add Sources] 메뉴를 이용하여 CMP_4.v 파일 추가
2. 모듈 이름은 CMP_4, 포트는 A, B, Tequ, Tneq, Tgth, Tlth, Tgte, Tlte로 구성하며 A, B는 input, Tequ, Tneq, Tgth, Tlth, Tgte, Tlte은 output으로 설정, A, B는 4비트 버스로

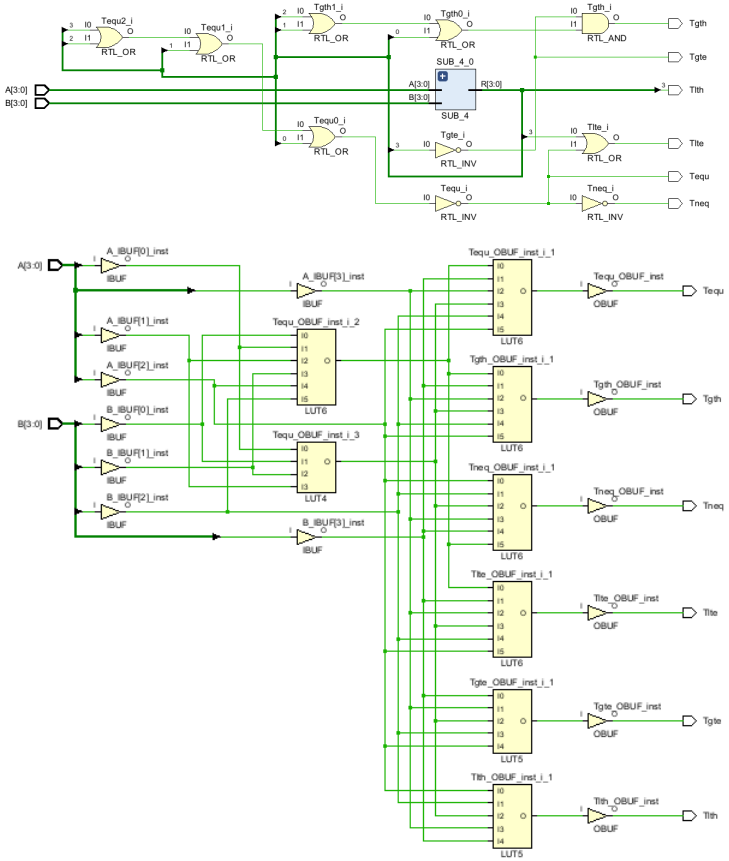
설정

3. CMP_4.v 파일을 마우스 오른쪽 버튼을 누른 후, [Set as Top] 모듈로 설정합니다.
4. 합성을 진행합니다.
5. 핀을 다음과 같이 할당합니다.

 A3~A0(W17,W16,V16,V17)

 B3~B0(W13,W14,V15,W15)

 Tequ,Tneq,Tgth,Tlth,Tgte,Tlte(U15,W18,V19,U19,E19,U16)
6. 구현을 합니다.
7. 비트스트림을 생성한 후, 보드에 다운로드 받아 테스트합니다.



BASYS 3 핀 번호