```
/*      NOT_LS  */
module  NOT_LS  ( IN, OUT );
input   IN;
output  OUT;
        not     #( 15, 15 )     U1      ( OUT , IN );
endmodule
```

```
/*      NOT_F   */
module  NOT_F   ( IN, OUT );
input   IN;
output  OUT;
        not     #( 6, 5.3 )     U2      ( OUT , IN );
endmodule
```

```
/*      NOT_LS  */
module  NOT_LS  ( IN, OUT );
input   IN;
output  OUT;
parameter       LS04_TPLH = 15;
parameter       LS04_TPHL = 15;

        assign  #( LS04_TPLH, LS04_TPHL )       OUT = ~IN;
endmodule
```

```
/*      NOT_F   */
module  NOT_F   ( IN, OUT );
input   IN;
output  OUT;
parameter       F04_TPLH = 6;
parameter       F04_TPHL = 5.3;
        assign  #( F04_TPLH, F04_TPHL )         OUT = ~IN;
endmodule
```

```
/*      AND_AS  */
module  AND_AS  ( IN_1, IN_2, OUT );
input   IN_1, IN_2;
output  OUT;
parameter       AS08_TPLH = 5.5;
parameter       AS08_TPHL = 5.5;

        assign  #( AS08_TPLH, AS08_TPHL )       OUT = IN_1 & IN_2;
endmodule
```

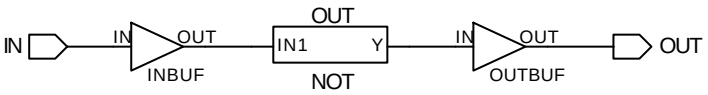
```
/*      AND     */
module  AND     ( IN, OUT );
input   IN;
output  OUT;
tri1    IN_PUP;
        assign  OUT = IN & IN_PUP;
endmodule
```

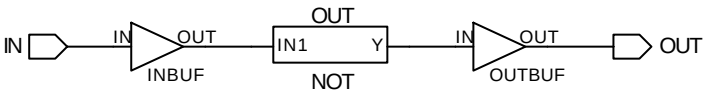
```
/*      OR      */
module  OR      ( IN, OUT );
input   IN;
output  OUT;
tri0    IN_PD0;
        assign  OUT = IN | IN_PD0;
endmodule
```

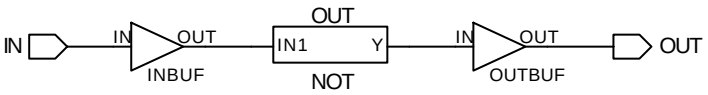
```
/*      TRI_BUF */
module  TRI_BUF ( IN_A, IN_B, IN_C, EN_A, EN_B, EN_C, OUT );
input   IN_A, IN_B, IN_C;
input   EN_A, EN_B, EN_C;
output  OUT;
tri     TRI_OR;
        bufif1  U1      ( TRI_OR, IN_A, EN_A );
        bufif1  U2      ( TRI_OR, IN_B, EN_B );
        bufif1  U3      ( TRI_OR, IN_C, EN_C );
        buf     U4      ( OUT, TRI_OR );
endmodule
```

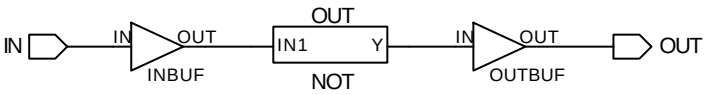
```
/*      TRI_BUF */
module  TRI_BUF ( IN_A, IN_B, IN_C, EN_A, EN_B, EN_C, OUT );
input   IN_A, IN_B, IN_C;
input   EN_A, EN_B, EN_C;
output  OUT;
tri     TRI_OR;
        assign  TRI_OR = TRI_BUF_G ( IN_A, EN_A );
        assign  TRI_OR = TRI_BUF_G ( IN_B, EN_B );
        assign  TRI_OR = TRI_BUF_G ( IN_C, EN_C );
        assign  OUT    = TRI_OR;

        function        TRI_BUF_G;
        input   IN, EN;

                TRI_BUF_G = ( EN )? IN: 1'bz;

        endfunction
endmodule
```

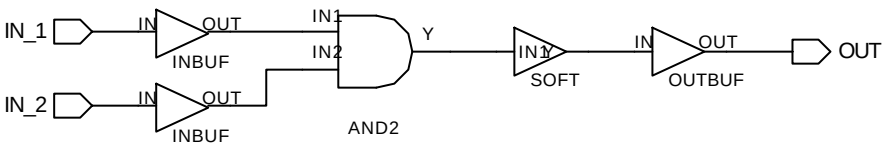
```
/*      PROB_2  */
module  PROB_2  ( A, B, C, OUT );
input   A, B, C;
output  OUT;
wire    ASTR_1, ASTR_2, ASTR_3;
tri1    IN_PUP;
parameter       AS00_TPLH = 4.5;
parameter       AS00_TPHL = 4;

        assign  #( AS00_TPLH, AS00_TPHL )       ASTR_3 = ~( C & IN_PUP );
        assign  #( AS00_TPLH, AS00_TPHL )       ASTR_1 = ~( A & C );
        assign  #( AS00_TPLH, AS00_TPHL )       ASTR_2 = ~( B & ASTR_3 );
        assign  #( AS00_TPLH, AS00_TPHL )       OUT    = ~( ASTR_1 & ASTR_2 );
endmodule
```

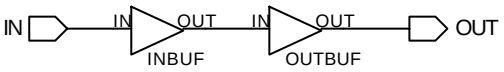

IN
IN
OUT
INBUF
OUT
IN1
Y
NOT
IN
OUT
OUTBUF
OUT

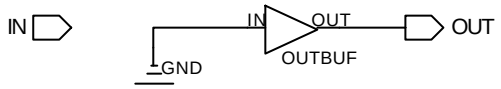

IN
IN
OUT
INBUF
OUT
IN1
Y
NOT
IN
OUT
OUTBUF
OUT

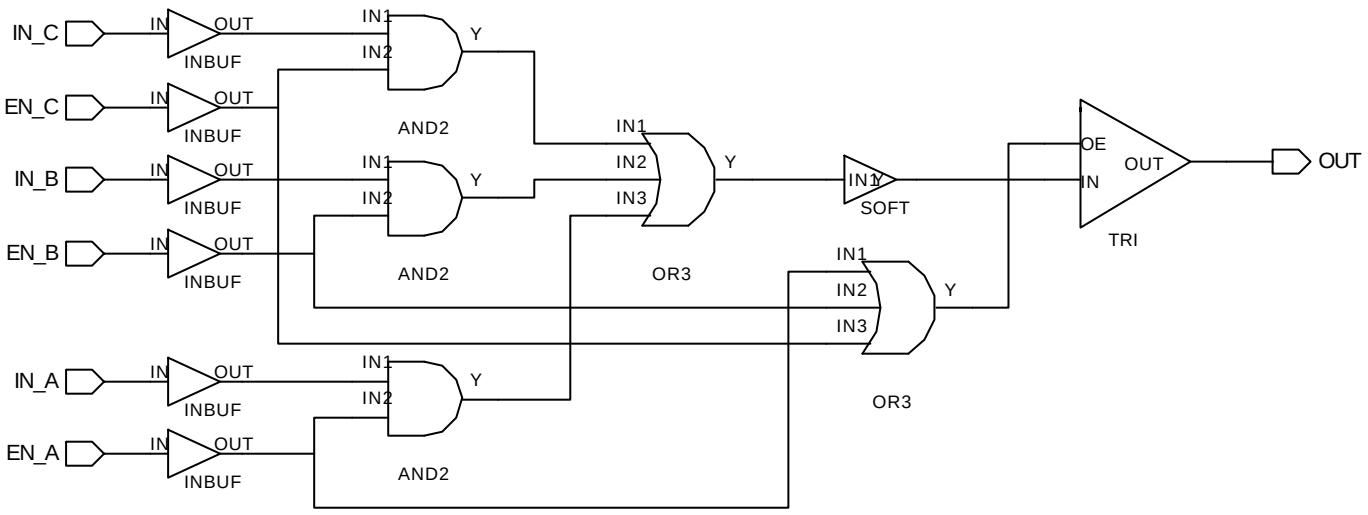

IN
IN
OUT
INBUF
OUT
IN1
Y
NOT
IN
OUT
OUTBUF
OUT

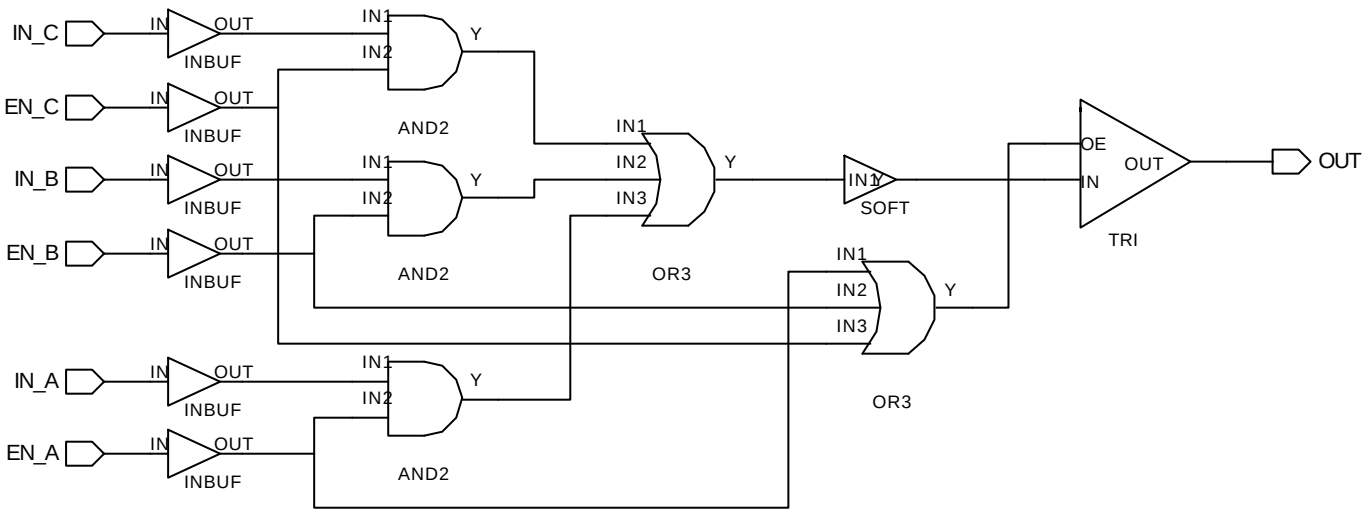

IN
IN
OUT
INBUF
OUT
IN1
Y
NOT
IN
OUT
OUTBUF
OUT

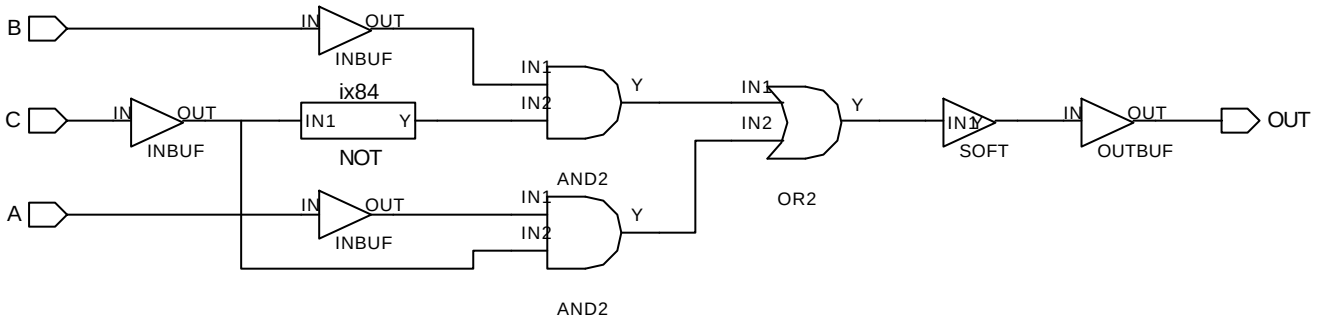

IN_1
IN
OUT
INBUF
IN_2
IN
OUT
INBUF
IN1
IN2
Y
AND2
IN
Y
SOFT
IN
OUT
OUTBUF
OUT

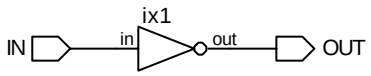

IN
IN
OUT
INBUF
IN
OUT
OUTBUF
OUT

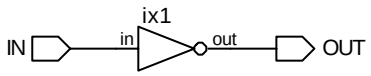

IN
IN
OUT
OUT
GND
OUTBUF

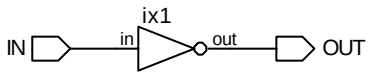

IN_C
IN_B
EN_C
EN_B
IN_A
EN_A
IN
OUT
INBUF
IN1
IN2
Y
AND2
IN3
OR3
SOFT
OE
TRI
OUT

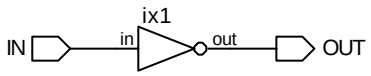

IN_C
IN_B
EN_C
EN_B
IN_A
EN_A
IN
OUT
INBUF
IN1
IN2
Y
AND2
IN3
OR3
SOFT
OE
TRI
OUT

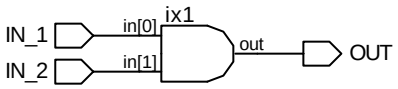

B
C
A
IN
OUT
INBUF
IN
OUT
INBUF
IN
OUT
INBUF
ix84
IN1
Y
NOT
IN1
IN2
Y
AND2
IN1
IN2
Y
AND2
IN1
IN2
Y
OR2
IN1
Y
SOFT
IN
OUT
OUTBUF
OUT

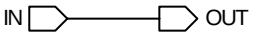

ix1
IN
in
out
OUT


ix1
IN
in
out
OUT

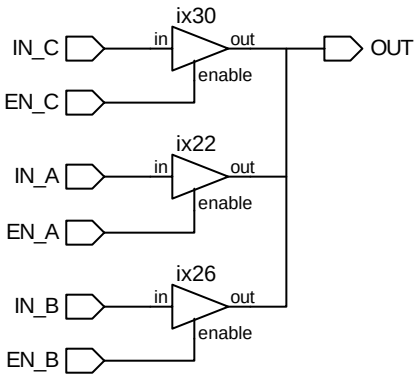

ix1
IN
in
out
OUT

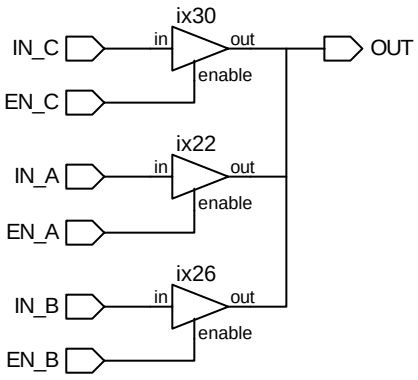

ix1
IN
in
out
OUT

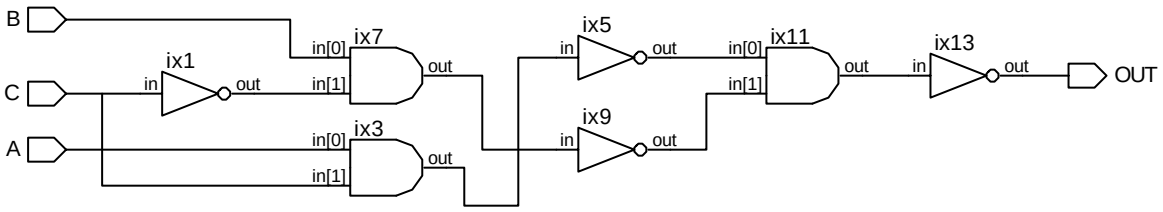

IN_1
IN_2
in[0]
in[1]
ix1
out
OUT


IN
OUT

IN
OUT


ix30
IN_C
in
out
OUT
enable
EN_C
ix22
IN_A
in
out
enable
EN_A
ix26
IN_B
in
out
enable
EN_B


ix30
IN_C
in
out
OUT
enable
EN_C
ix22
IN_A
in
out
enable
EN_A
ix26
IN_B
in
out
enable
EN_B


B
C
A
ix1
in
out
ix7
in[0]
in[1]
out
ix3
in[0]
in[1]
out
ix5
in
out
ix9
in
out
ix11
in[0]
in[1]
out
ix13
in
out
OUT