```
/*      NOR_RSFF        */
module  NOR_RSFF        ( R, S, Q, Q_B );
input   R, S;
output  Q, Q_B;
        nor     ( Q   , R, Q_B );
        nor     ( Q_B, S, Q   );
endmodule
```

```
/*      RSFF    */
module  RSFF    ( R, S, Q, Q_B );
input   R, S;
output  Q, Q_B;
reg     Q, Q_B;
        always  @( R or S )
                case ({ R , S })
                        1:begin Q <= 1; Q_B <= 0;       end
                        2:begin Q <= 0; Q_B <= 1;       end
                        3:begin Q <= 0; Q_B <= 0;       end
                endcase
endmodule
```

```
/*      TFF     */
module  TFF     ( R, T, Q, Q_B );
input   R, T;
output  Q, Q_B;
reg     Q;

        assign  Q_B = ~Q;
        always  @( posedge R or posedge T )
                if ( R )
                        Q <= 0;
                else
                        Q <= ~Q;
endmodule
```

```
/*      SY_RSFF */
module  SY_RSFF ( R, S, CLK, Q, Q_B );
input   R, S, CLK;
output  Q, Q_B;
reg     Q;
        assign  Q_B = ~Q;
        always  @( posedge CLK )
                case ({ R ,S })
                        1:Q <= 1;
                        2:Q <= 0;
                        3:Q <= 1'bx;
                endcase
endmodule
```

```
/*      SY_TFF  */
module  SY_TFF  ( R, T, CLK, Q, Q_B );
input   R, T, CLK;
output  Q, Q_B;
reg     Q;
        assign  Q_B = ~Q;
        always  @( posedge CLK or posedge R )
                if ( R )
                        Q <= 0;
                else if ( T )
                        Q <= ~Q;
endmodule
```

```
/*      SY_DFF  */
module  SY_DFF  ( D, CLK, Q, Q_B );
input   D, CLK;
output  Q, Q_B;
reg     Q;
        assign  Q_B = ~Q;
        always  @( posedge CLK )
                        Q <= D;
endmodule
```

```
/*      SY_JKFF          */
module  SY_JKFF ( J, K, CLK, Q, Q_B );
input   J, K, CLK;
output  Q, Q_B;
reg     Q;
        assign  Q_B = ~Q;
        always  @( posedge CLK )
                case ({ J , K })
                        1:Q <= 0;
                        2:Q <= 1;
                        3:Q <= ~Q;
                endcase
endmodule
```

```
/*      SY_RSFF */
module  SY_RSFF ( R, S, CLK, Q, Q_B );
input   R, S, CLK;
output  Q, Q_B;
reg     Q;
        assign  Q_B = ~Q;
        always  @( posedge CLK )
                Q <= S | ( ~R & Q );
endmodule
```

```
/*      SY_TFF  */
module  SY_TFF  ( R, T, CLK, Q, Q_B );
input   R, T, CLK;
output  Q, Q_B;
reg     Q;
        assign  Q_B = ~Q;
        always  @( posedge CLK or posedge R )
                if ( R )
                        Q <= 0;
                else
                        Q <= Q ^ T;
endmodule
```

```
/*      SY_JKFF          */
module  SY_JKFF ( J, K, CLK, Q, Q_B );
input   J, K, CLK;
output  Q, Q_B;
reg     Q;
        assign  Q_B = ~Q;
        always  @( posedge CLK )
                Q <= ( J & ~Q ) | ( ~K & Q );
endmodule
```

```
/*      R_SYDFF */
module  R_SYDFF ( R_B, D, CLK, Q, Q_B );
input   R_B, D, CLK;
output  Q, Q_B;
reg     Q;
        assign  Q_B = ~Q;
        always  @( posedge CLK or negedge R_B )
                if ( !R_B )
                        Q <= 0;
                else
                        Q <= D;
endmodule
```

```
/*      LAT     */
module  LAT     ( D, G, Q, Q_B );
input   D, G;
output  Q, Q_B;
reg     Q;
        assign  Q_B = ~Q;
        always  @( D or G )
                if ( G )
                        Q <= D;
endmodule
```

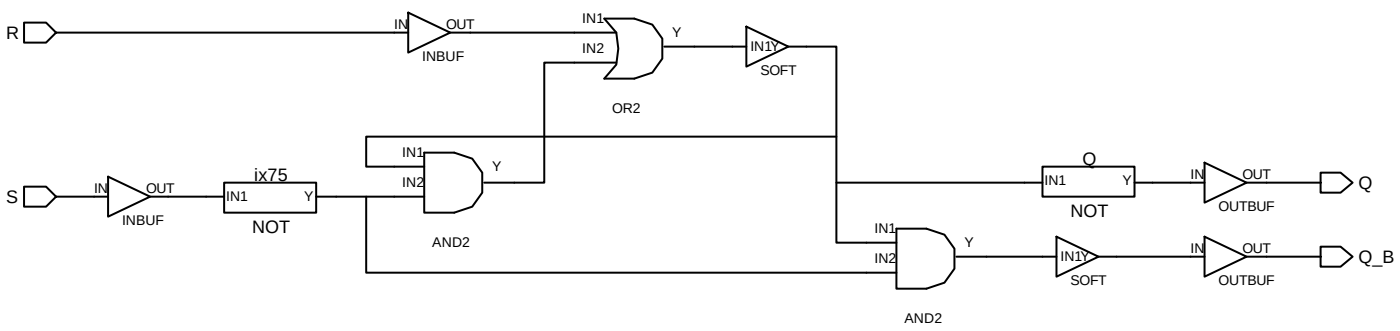
```
/*      FF_DELAY        */
module  FF_DELAY        ( R_B, D, CLK, Q, Q_B );
input   R_B, D, CLK;
output  Q, Q_B;
reg     Q;
parameter       R_OUT   =  9;
parameter       CLK_OUT = 10.5;

        assign  Q_B = ~Q;

        always  @( posedge CLK or negedge R_B )
                if ( !R_B )
                        #R_OUT          Q <= 0;
                else
                        #CLK_OUT        Q <= D;
endmodule
```

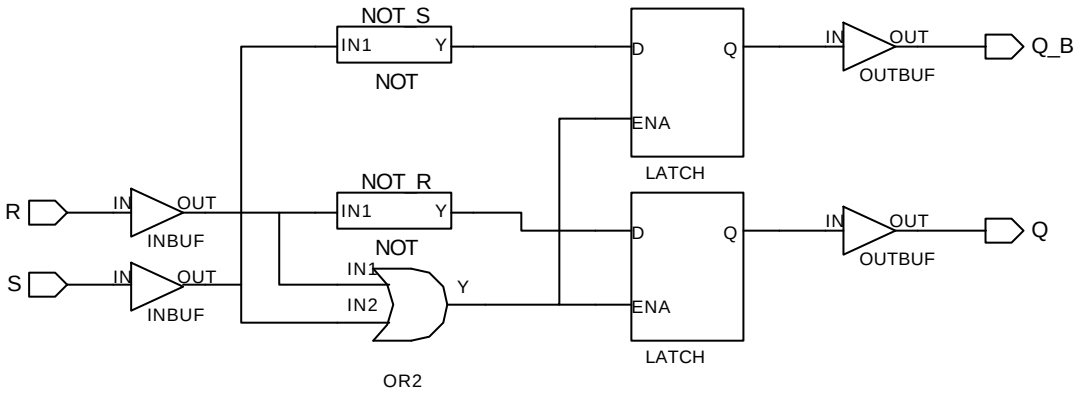
```
/*      NAND_RSFF       */
module  NAND_RSFF       ( R_B, S_B, Q, Q_B );
input   R_B, S_B;
output  Q, Q_B;
reg     Q, Q_B;
        always  @( R_B or S_B )
                case ({ R_B , S_B })
                        0:begin Q <= 1; Q_B <= 1;  end  // R_B = 0, S_B = 0
                        1:begin Q <= 0; Q_B <= 1;  end  // R_B = 0, S_B = 1
                        2:begin Q <= 1; Q_B <= 0;  end  // R_B = 1, S_B = 0
                endcase
endmodule
```

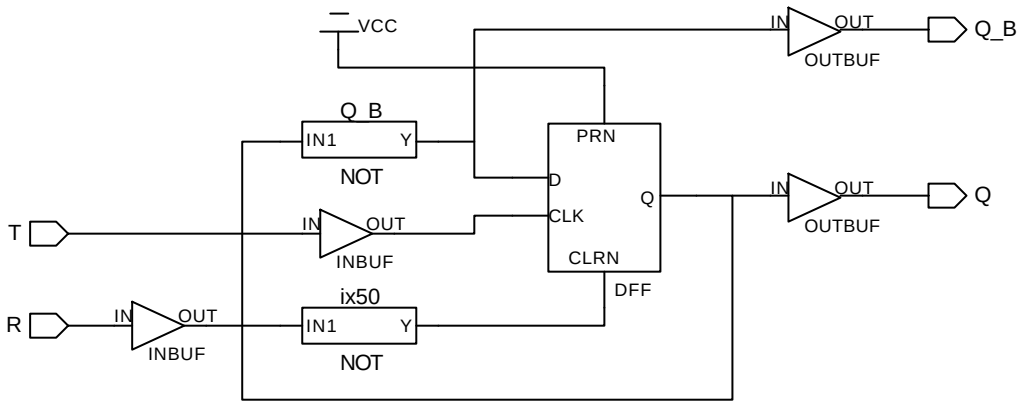
```
/*      SY_RSFF */
module  SY_RSFF ( R, S, CLK, Q, Q_B );
input   R, S, CLK;
output  Q, Q_B;
reg     Q;
        assign  Q_B = ~Q;
        always  @( negedge CLK )
                case ({ R ,S })
                        1:Q <= 1;
                        2:Q <= 0;
                        3:Q <= 1'bx;
                endcase
endmodule
```

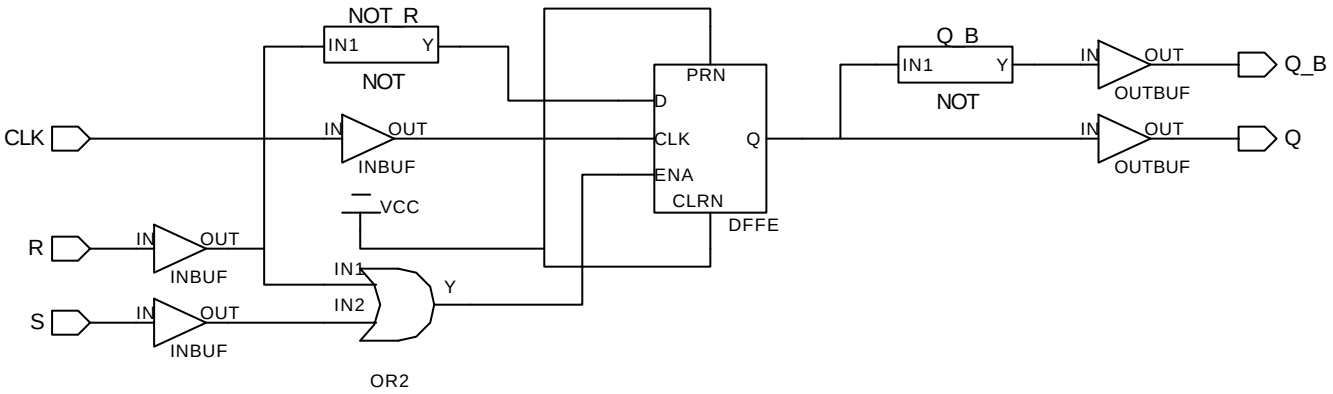
```
/*      R_SYJKFF        */
module  R_SYJKFF        ( R_B, J, K, CLK, Q, Q_B );
input   R_B, J, K, CLK;
output  Q, Q_B;
reg     Q;
        assign  Q_B = ~Q;
        always  @( posedge CLK or negedge R_B )
                if ( !R_B )
                        Q <= 0;
                else
                        case ({ J , K })
                                1:Q <= 0;
                                2:Q <= 1;
                                3:Q <= ~Q;
                        endcase
endmodule
```

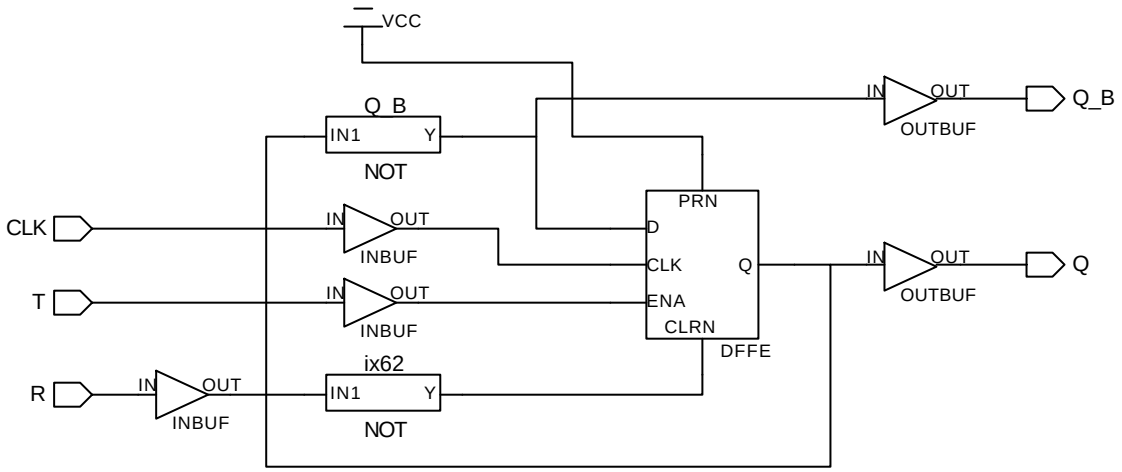
```
/*      LAT     */
module  LAT     ( D, G, Q, Q_B );
input   D, G;
output  Q, Q_B;
reg     Q;
wire    D_DELAY, G_DELAY;
parameter       G_OUT = 11.5;
parameter       D_OUT =  6;

        assign  #D_OUT  D_DELAY = D;
        assign  #G_OUT  G_DELAY = G;
        assign  Q_B = ~Q;
        always  @( D_DELAY or G_DELAY )
                if ( G_DELAY )
                        Q <= D_DELAY;
endmodule
```

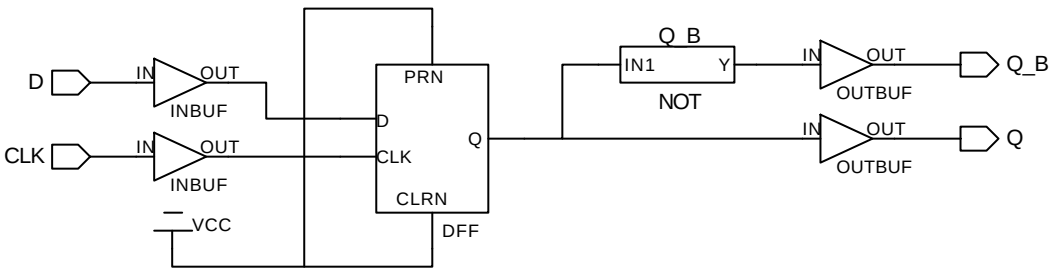

R
IN
OUT
INBUF
IN1
IN2
Y
OR2
IN1Y
SOFT
S
IN
OUT
INBUF
ix75
IN1
Y
NOT
IN1
IN2
Y
AND2
Q
IN1
Y
NOT
IN
OUT
OUTBUF
Q
IN1
IN2
Y
AND2
IN1Y
SOFT
IN
OUT
OUTBUF
Q_B

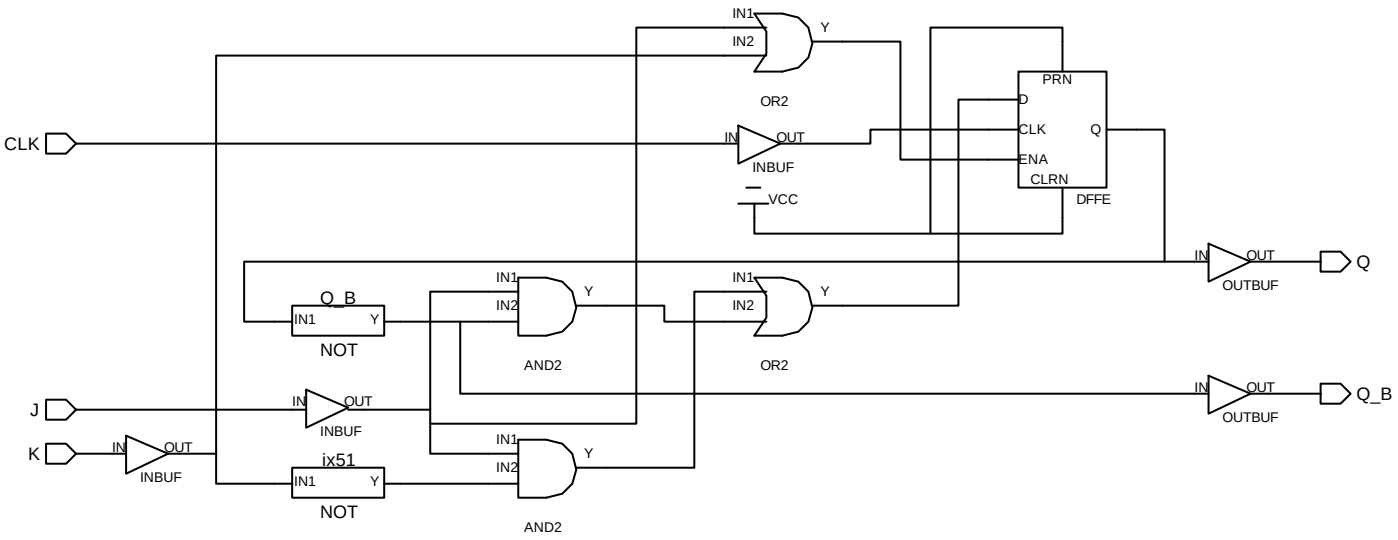

NOT_S
IN1
Y
NOT
D
Q
ENA
LATCH
IN
OUT
OUTBUF
Q_B
NOT_R
IN1
Y
NOT
D
Q
ENA
LATCH
IN
OUT
OUTBUF
Q
R
IN
OUT
INBUF
S
IN
OUT
INBUF
IN1
IN2
Y
OR2

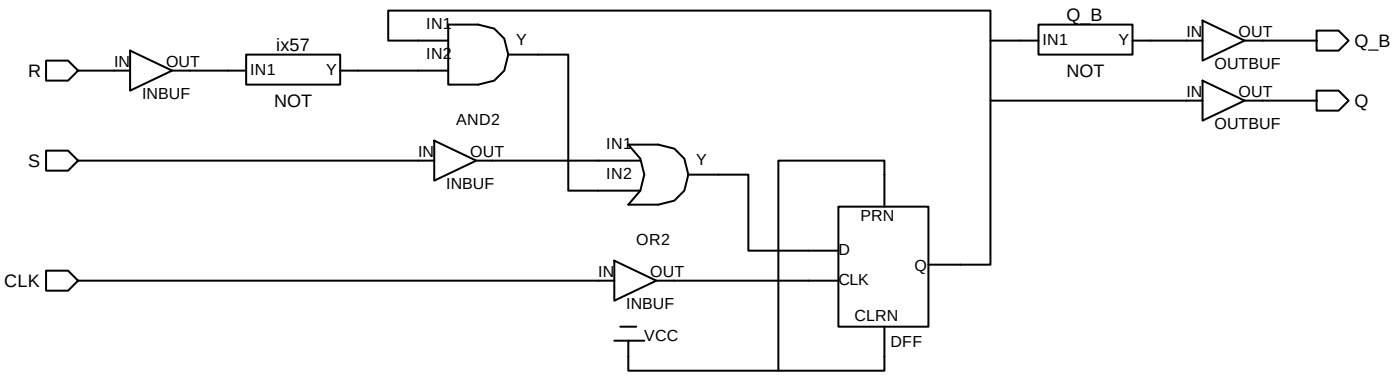

VCC
IN
OUT
Q_B
OUTBUF
Q_B
IN1
Y
NOT
PRN
D
Q
CLK
CLRN
DFF
IN
OUT
Q
OUTBUF
T
IN
OUT
INBUF
ix50
R
IN
OUT
IN1
Y
INBUF
NOT

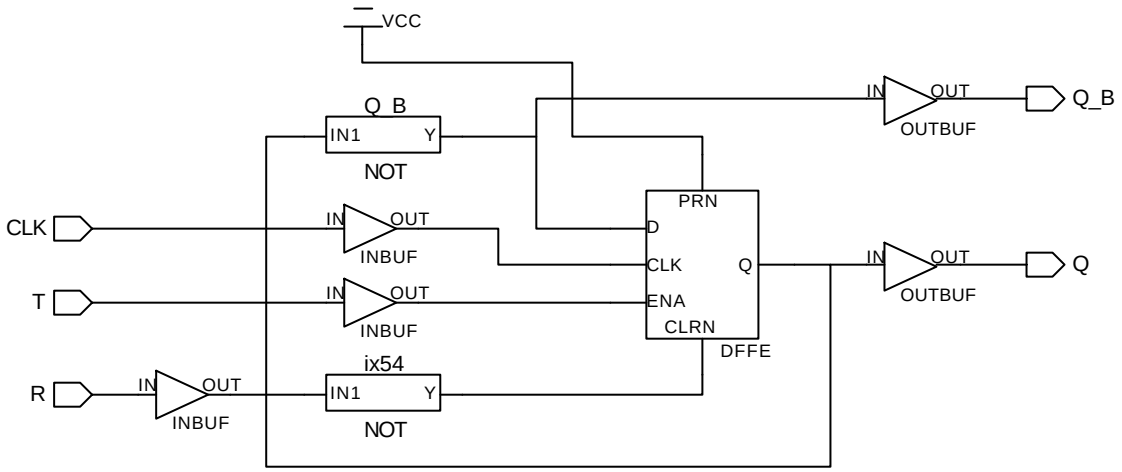

NOT_R
IN1
Y
NOT
PRN
D
CLK
Q
ENA
CLRN
DFFE
Q_B
IN1
Y
NOT
IN
OUT
OUTBUF
Q_B
IN
OUT
OUTBUF
Q
CLK
IN
OUT
INBUF
VCC
R
IN
OUT
INBUF
S
IN
OUT
INBUF
IN1
IN2
Y
OR2

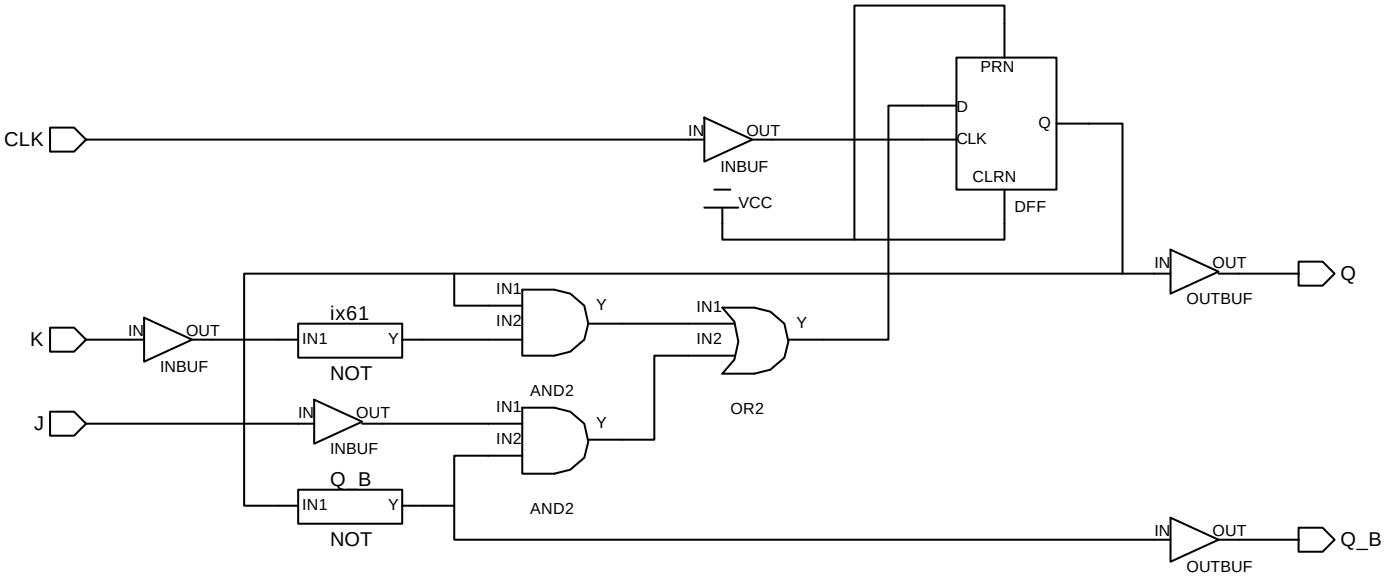

VCC
Q_B
IN1
Y
NOT
IN
OUT
OUTBUF
Q_B
CLK
INBUF
PRN
D
CLK
Q
ENA
CLRN
DFFE
T
INBUF
R
INBUF
ix62
IN1
Y
NOT
IN
OUT
OUTBUF
Q

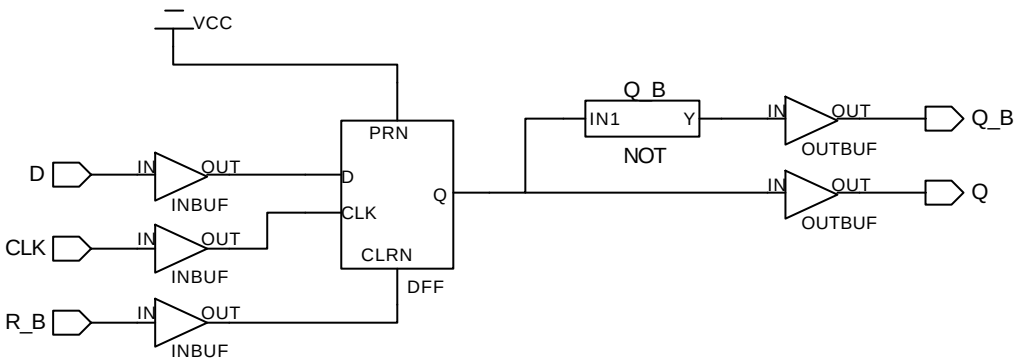

D
IN
OUT
INBUF
CLK
IN
OUT
INBUF
VCC
PRN
D
CLK
Q
CLRN
DFF
Q_B
IN1
Y
NOT
IN
OUT
OUTBUF
Q_B
IN
OUT
OUTBUF
Q

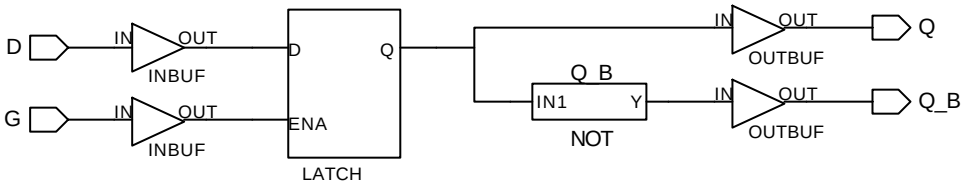

IN1
IN2
Y
OR2
PRN
D
CLK
Q
ENA
CLRN
DFFE
CLK
IN
OUT
INBUF
VCC
IN
OUT
Q
OUTBUF
Q_B
IN1
Y
NOT
IN1
IN2
Y
AND2
IN1
IN2
Y
OR2
IN
OUT
Q_B
OUTBUF
J
IN
OUT
INBUF
K
IN
OUT
INBUF
ix51
IN1
Y
NOT
IN1
IN2
Y
AND2

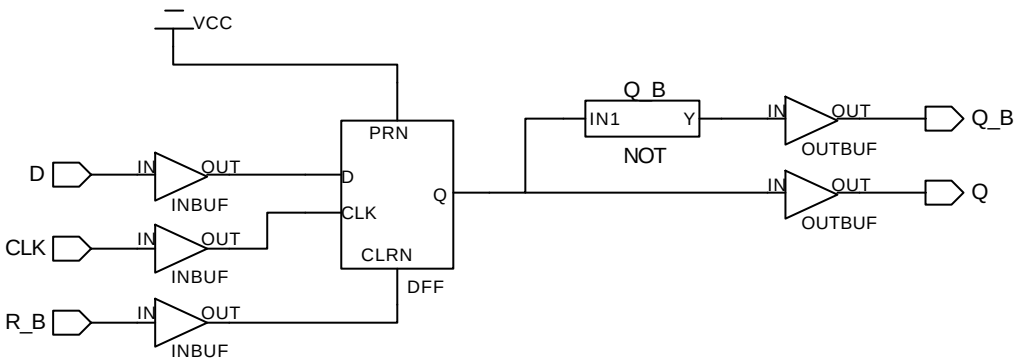

R
IN
OUT
INBUF
ix57
IN1
Y
NOT
IN1
IN2
Y
AND2
S
IN
OUT
INBUF
IN1
IN2
Y
OR2
CLK
IN
OUT
INBUF
VCC
PRN
D
CLK
Q
CLRN
DFF
Q_B
IN1
Y
NOT
IN
OUT
OUTBUF
Q_B
IN
OUT
OUTBUF
Q

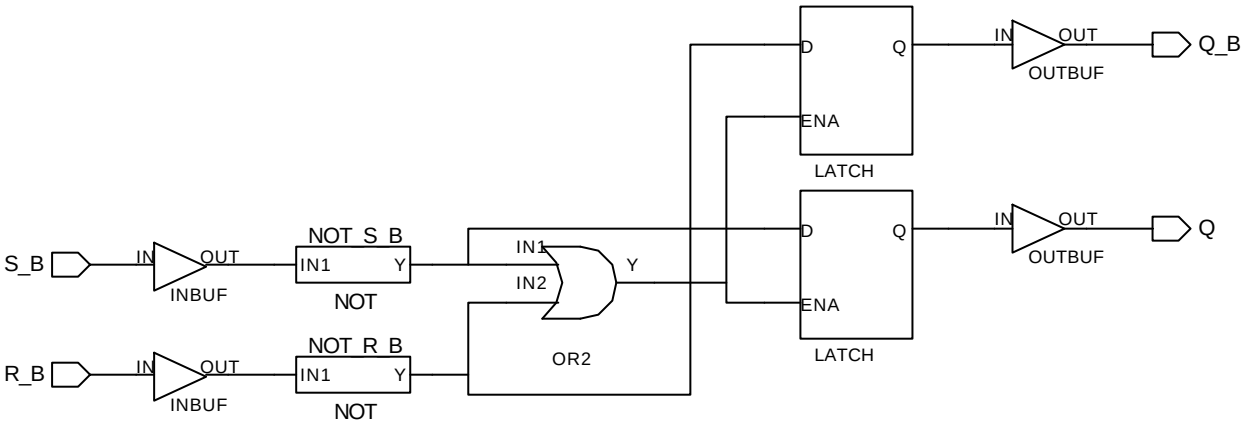

VCC
Q_B
IN1
Y
NOT
IN
OUT
OUTBUF
Q_B
CLK
INBUF
PRN
D
CLK
Q
ENA
CLRN
DFFE
T
INBUF
R
INBUF
ix54
IN1
Y
NOT
IN
OUT
OUTBUF
Q

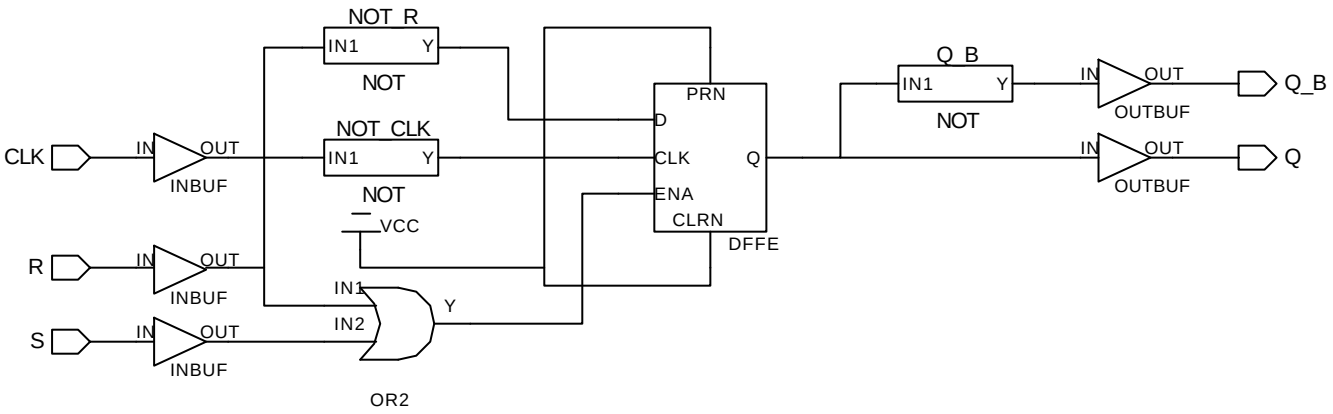

CLK
IN
OUT
INBUF
VCC
PRN
D
Q
CLK
CLRN
DFF
IN
OUT
OUTBUF
Q
K
IN
OUT
INBUF
ix61
IN1
Y
NOT
IN1
IN2
Y
AND2
IN1
IN2
Y
OR2
J
IN
OUT
INBUF
Q_B
IN1
Y
NOT
IN1
IN2
Y
AND2
IN
OUT
OUTBUF
Q_B

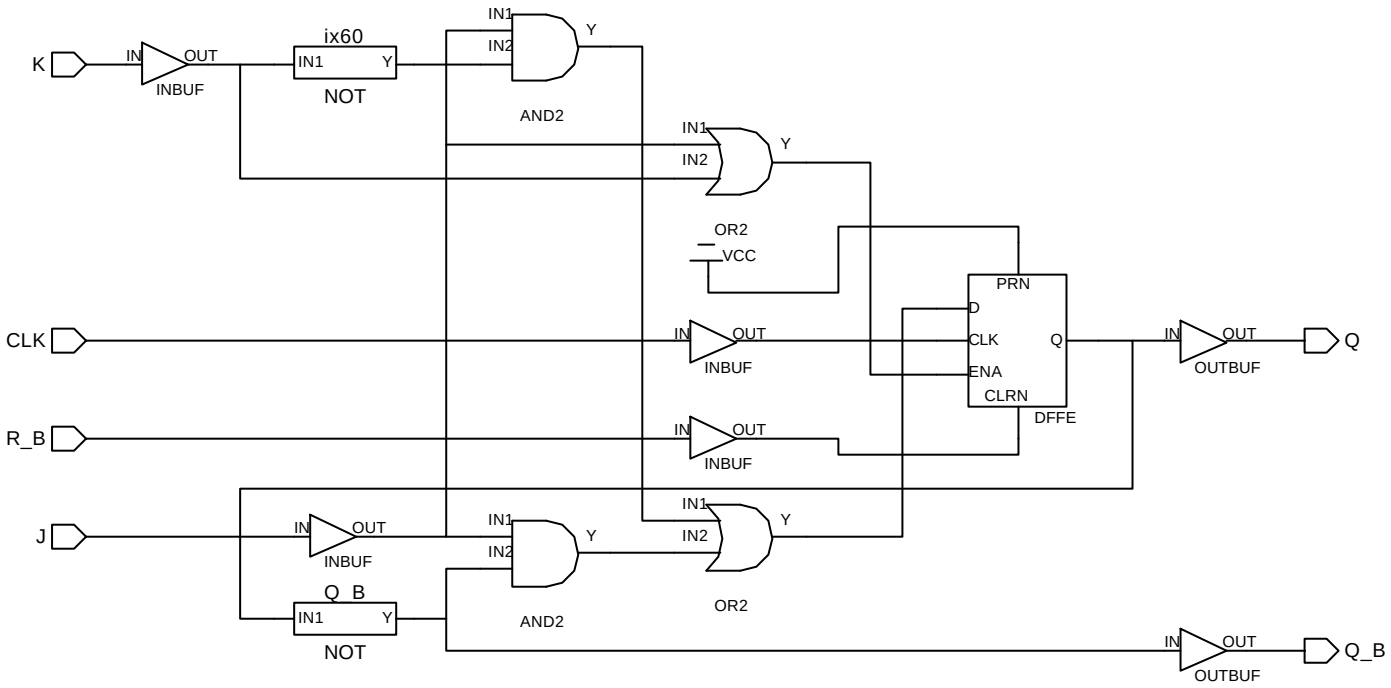

VCC
PRN
D
CLK
CLRN
Q
DFF
D
IN
OUT
INBUF
CLK
IN
OUT
INBUF
R_B
IN
OUT
INBUF
Q_B
IN1
Y
NOT
IN
OUT
OUTBUF
Q_B
IN
OUT
OUTBUF
Q

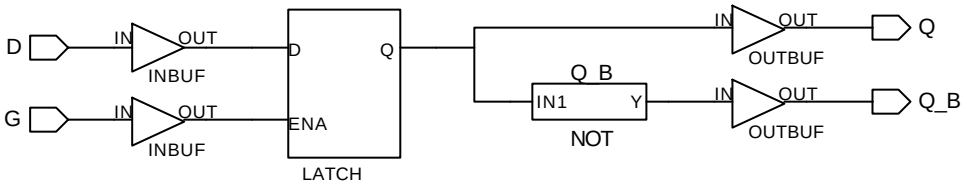

D
G
IN
OUT
INBUF
IN
OUT
INBUF
D
Q
ENA
LATCH
Q_B
IN1
Y
NOT
IN
OUT
OUTBUF
Q
IN
OUT
OUTBUF
Q_B

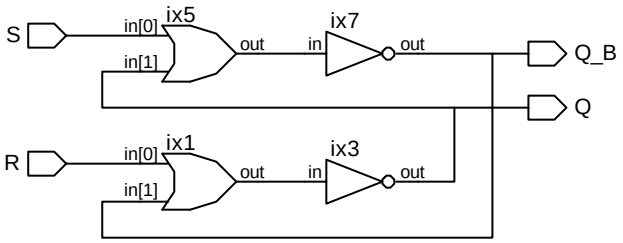

VCC
PRN
D
CLK
CLRN
Q
DFF
D
IN
OUT
INBUF
CLK
IN
OUT
INBUF
R_B
IN
OUT
INBUF
Q_B
IN1
Y
NOT
IN
OUT
OUTBUF
Q_B
IN
OUT
OUTBUF
Q

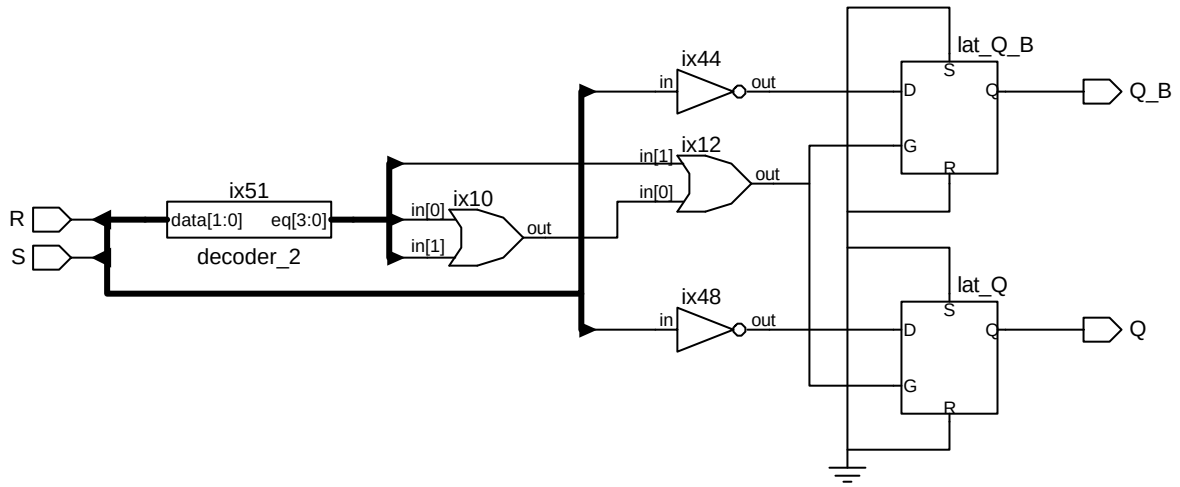

S_B
IN
OUT
INBUF
NOT_S_B
IN1
Y
NOT
R_B
IN
OUT
INBUF
NOT_R_B
IN1
Y
NOT
IN1
IN2
Y
OR2
D
Q
ENA
LATCH
D
Q
ENA
LATCH
IN
OUT
OUTBUF
Q_B
IN
OUT
OUTBUF
Q

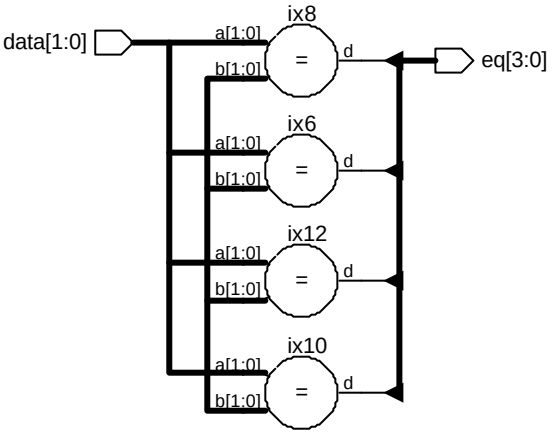

NOT_R
IN1
Y
NOT
NOT_CLK
IN1
Y
NOT
CLK
IN
OUT
INBUF
R
IN
OUT
INBUF
S
IN
OUT
INBUF
VCC
IN1
IN2
Y
OR2
PRN
D
CLK
Q
ENA
CLRN
DFFE
Q_B
IN1
Y
NOT
IN
OUT
OUTBUF
Q_B
IN
OUT
OUTBUF
Q

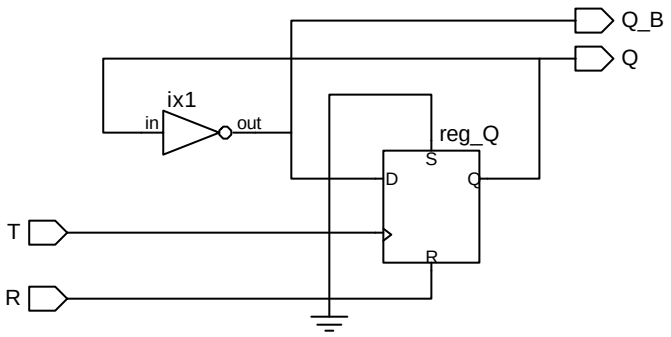

K
IN
OUT
INBUF
ix60
IN1
Y
NOT
IN1
IN2
Y
AND2
IN1
IN2
Y
OR2
VCC
PRN
D
CLK
Q
ENA
CLRN
DFFE
CLK
IN
OUT
INBUF
R_B
IN
OUT
INBUF
J
IN
OUT
INBUF
Q_B
IN1
Y
NOT
IN1
IN2
Y
AND2
IN1
IN2
Y
OR2
IN
OUT
OUTBUF
Q
IN
OUT
OUTBUF
Q_B

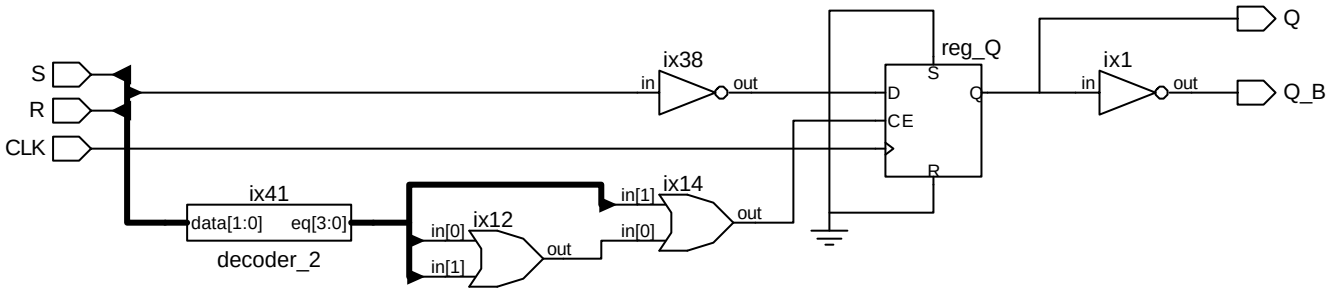

D
G
IN
OUT
INBUF
IN
OUT
INBUF
D
Q
ENA
LATCH
Q_B
IN1
Y
NOT
IN
OUT
OUTBUF
Q
IN
OUT
OUTBUF
Q_B

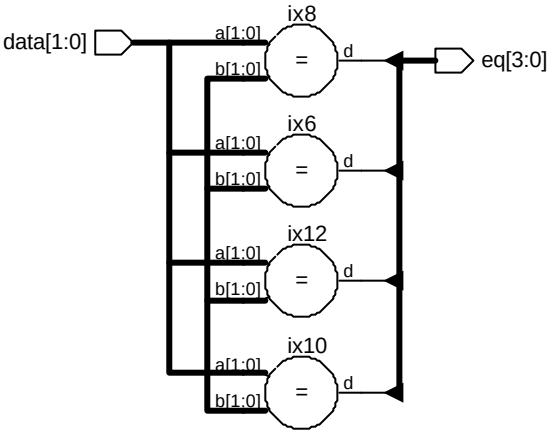

S
R
ix5
ix1
ix7
ix3
in[0]
in[1]
out
in
Q_B
Q

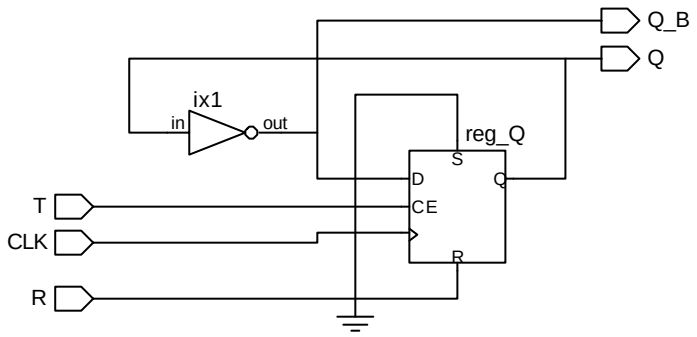

R
S
ix51
data[1:0]
eq[3:0]
decoder_2
ix10
in[0]
in[1]
out
ix44
in
out
ix12
in[1]
in[0]
out
ix48
in
out
lat_Q_B
S
D
Q
G
R
Q_B
lat_Q
S
D
Q
G
R
Q

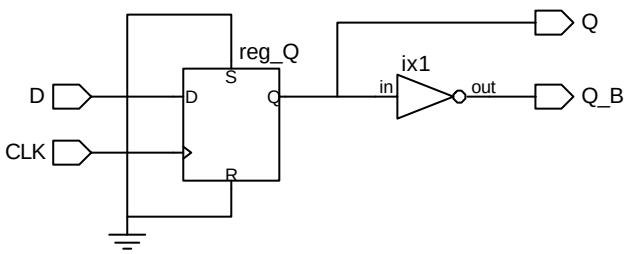

data[1:0]
ix8
a[1:0]
b[1:0]
=
d
eq[3:0]
ix6
a[1:0]
b[1:0]
=
d
ix12
a[1:0]
b[1:0]
=
d
ix10
a[1:0]
b[1:0]
=
d

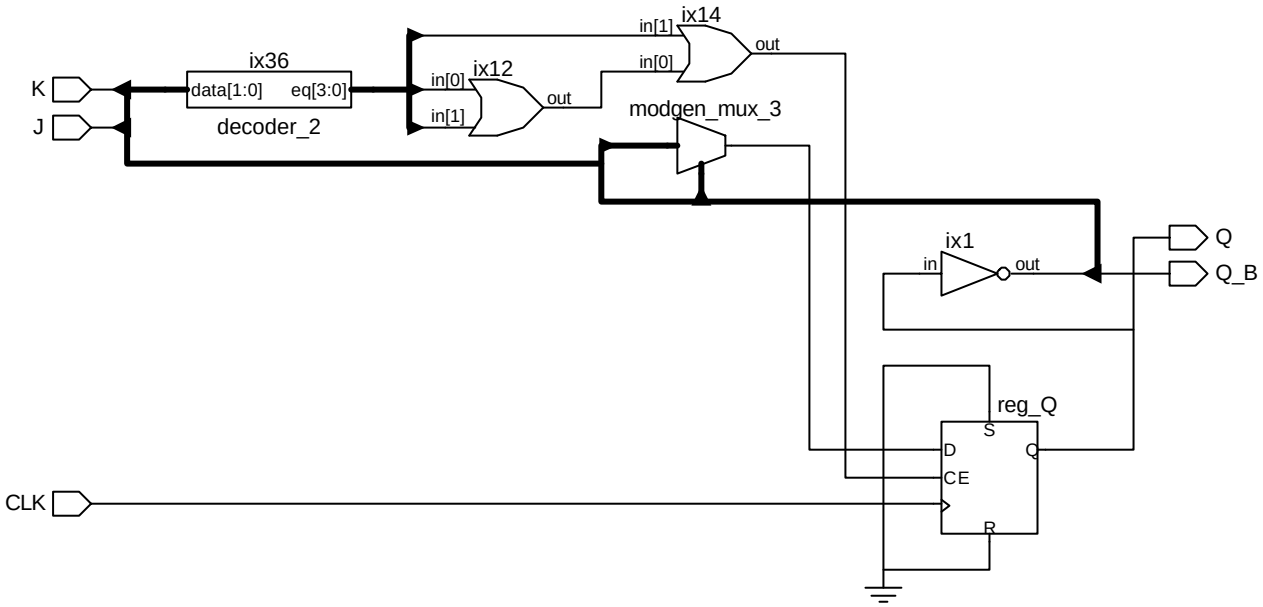

Q_B
Q
ix1
in
out
reg_Q
S
D
Q
T
R
R

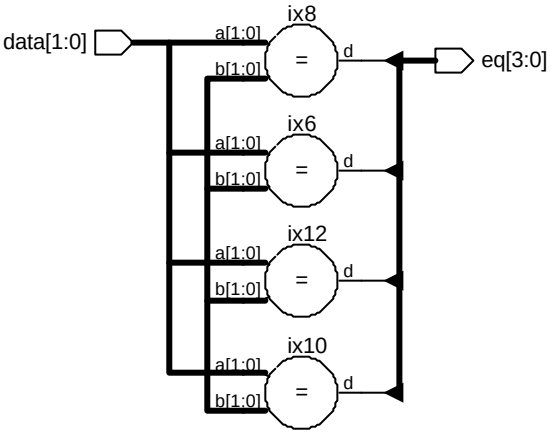

S
R
CLK
ix38
in
out
reg_Q
S
D
Q
CE
R
ix1
in
out
Q
Q_B
ix41
data[1:0]
eq[3:0]
decoder_2
ix12
in[0]
in[1]
out
ix14
in[1]
in[0]
out

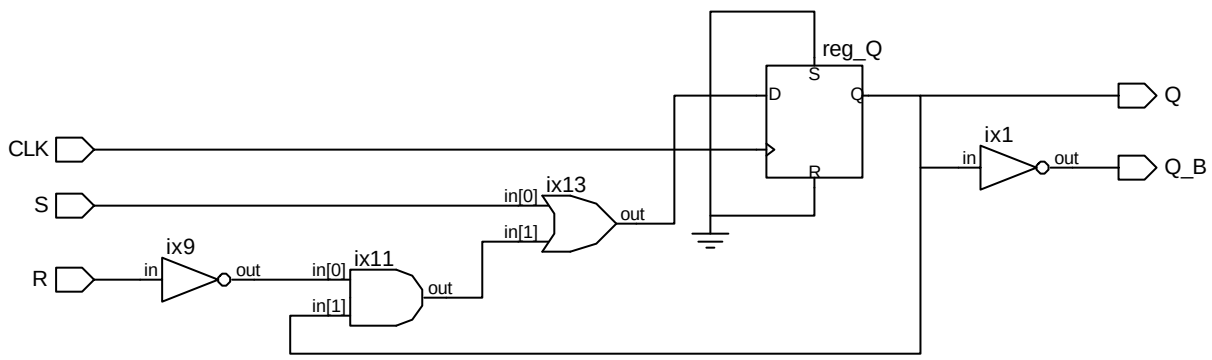

data[1:0]
ix8
a[1:0]
b[1:0]
=
d
eq[3:0]
ix6
a[1:0]
b[1:0]
=
d
ix12
a[1:0]
b[1:0]
=
d
ix10
a[1:0]
b[1:0]
=
d

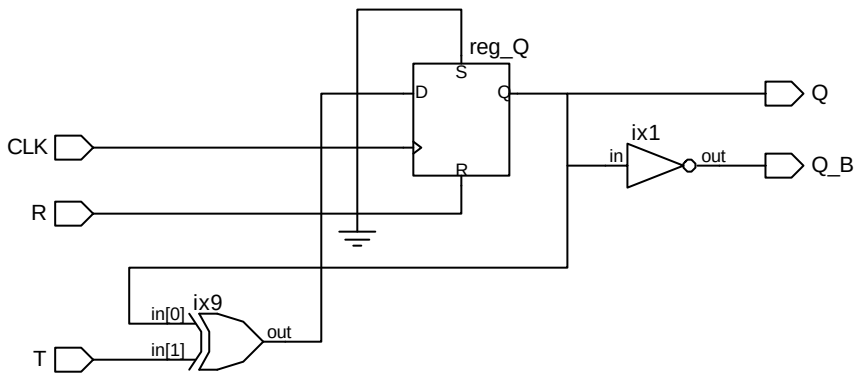

Q_B
Q
ix1
in
out
reg_Q
S
D
Q
T
CE
CLK
R
R

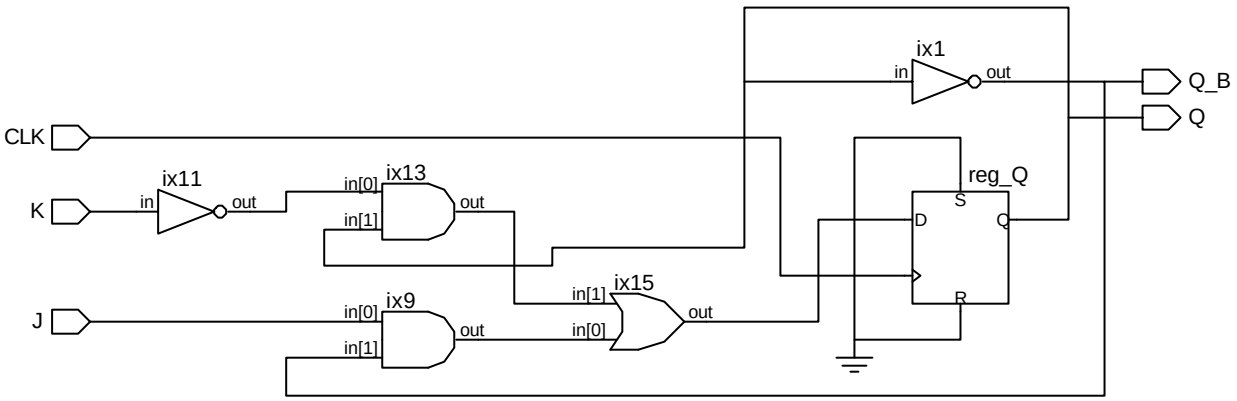

Q
reg_Q
S
ix1
D
D
Q
in
out
Q_B
CLK
R

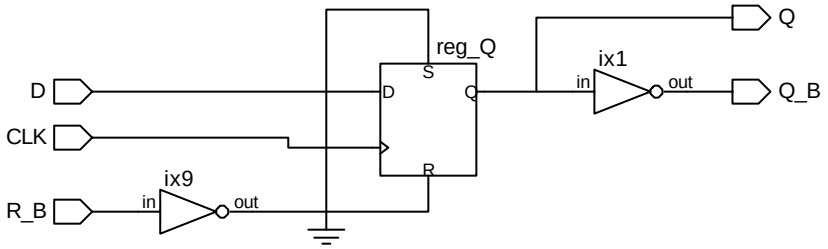

K
J
ix36
data[1:0]
eq[3:0]
decoder_2
ix12
in[0]
in[1]
out
ix14
in[1]
in[0]
out
modgen_mux_3
ix1
in
out
Q
Q_B
reg_Q
S
D
Q
CE
R
CLK

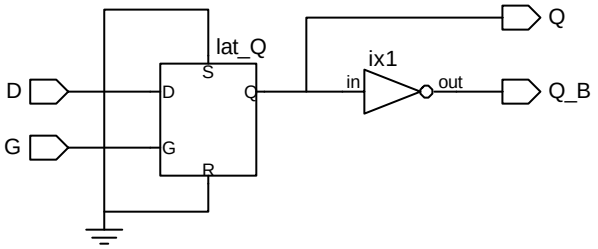

data[1:0]
ix8
a[1:0]
b[1:0]
=
d
eq[3:0]
ix6
a[1:0]
b[1:0]
=
d
ix12
a[1:0]
b[1:0]
=
d
ix10
a[1:0]
b[1:0]
=
d

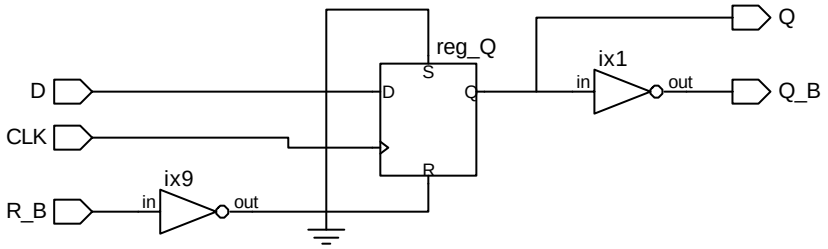

reg_Q
S
D
Q
R
CLK
S
R
Q
ix1
in
out
Q_B
ix13
in[0]
in[1]
out
ix9
in
out
ix11
in[0]
in[1]
out

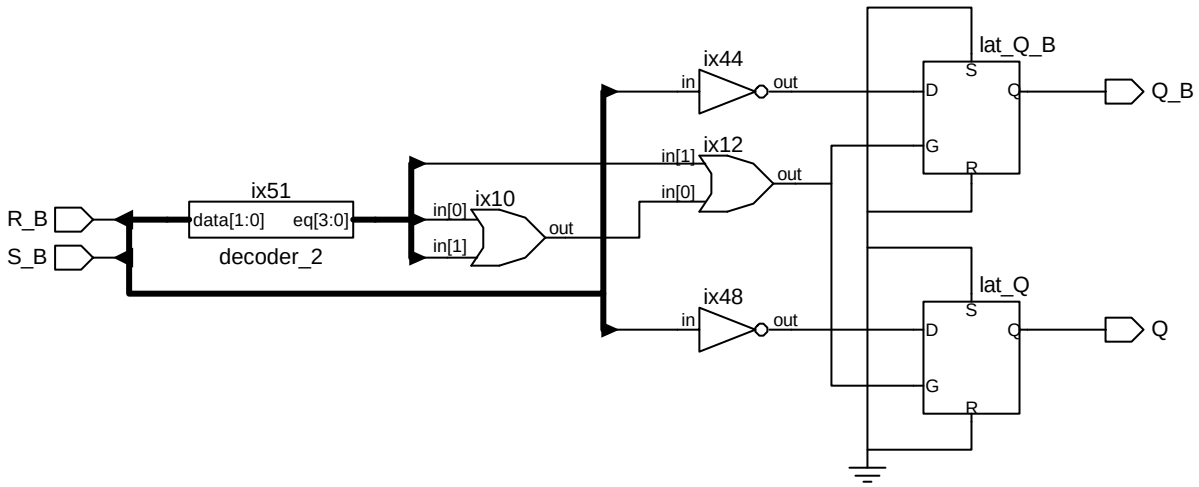

reg_Q
S
D
Q
R
CLK
R
Q
ix1
in
out
Q_B
ix9
in[0]
in[1]
out
T

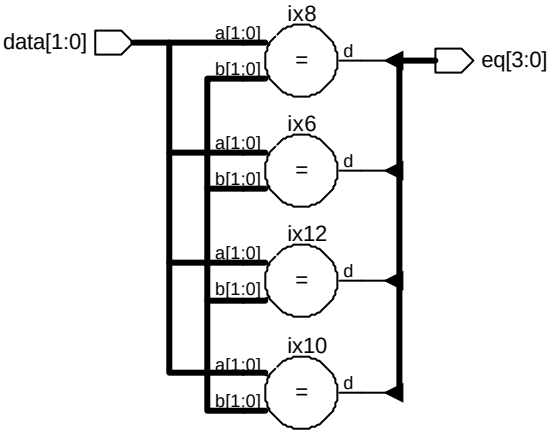

ix1
in
out
Q_B
Q
CLK
K
ix11
in
out
ix13
in[0]
in[1]
out
reg_Q
S
D
Q
R
J
ix9
in[0]
in[1]
out
ix15
in[1]
in[0]
out

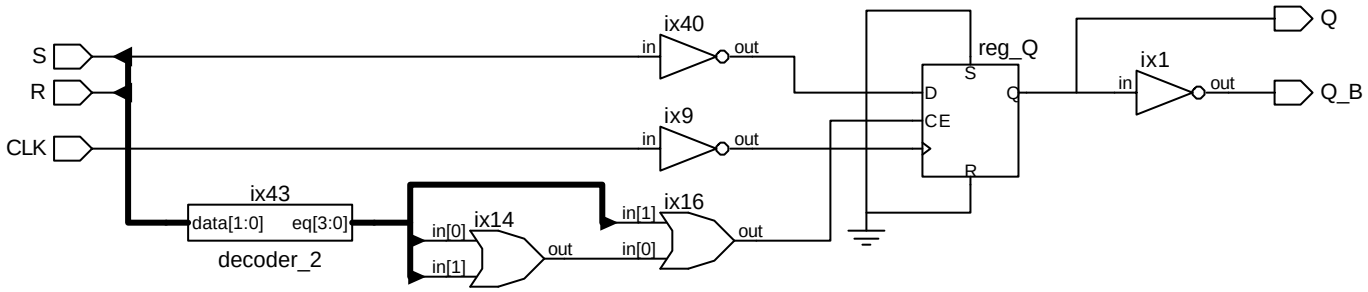

Q
reg_Q
S
ix1
in
out
D
D
Q
Q_B
CLK
R
ix9
in
out
R_B

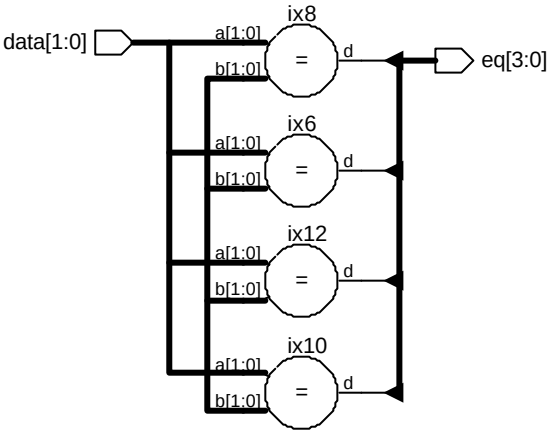

Q
lat_Q
S
ix1
D
D
Q
in
out
Q_B
G
G
R

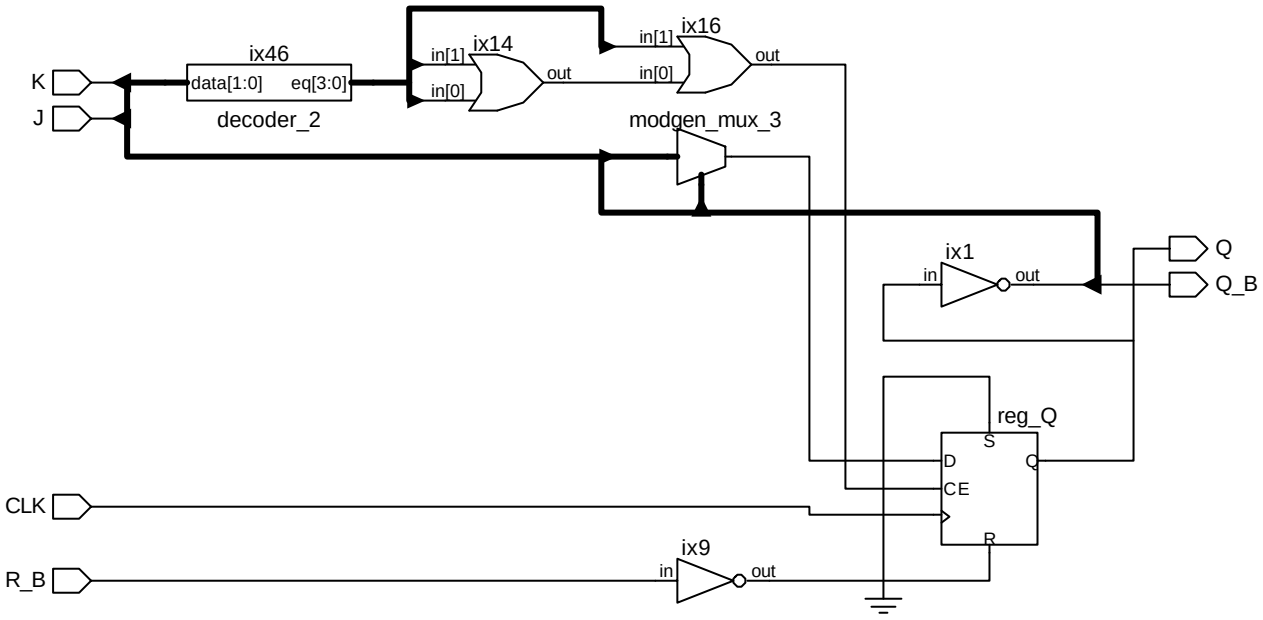

Q
reg_Q
S
ix1
in
out
D
D
Q
Q_B
CLK
R
ix9
in
out
R_B

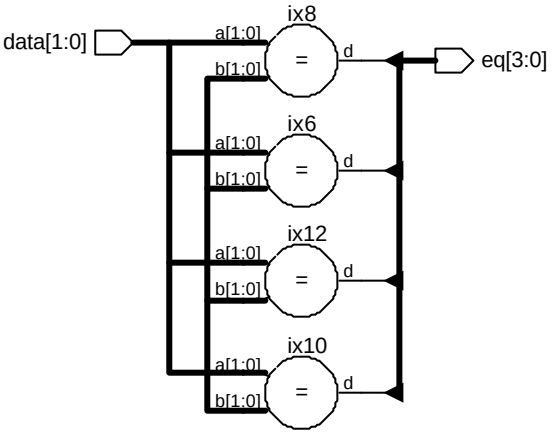

R_B
S_B
ix51
data[1:0]
eq[3:0]
decoder_2
ix10
in[0]
in[1]
out
ix44
in
out
ix12
in[1]
in[0]
out
ix48
in
out
lat_Q_B
S
D
Q
G
R
Q_B
lat_Q
S
D
Q
G
R
Q

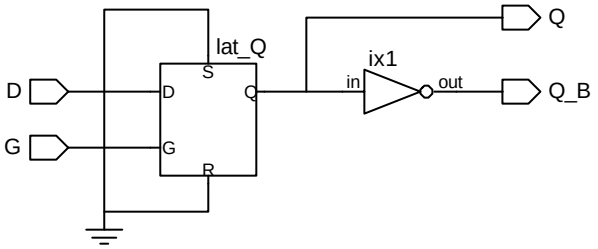

data[1:0]
ix8
a[1:0]
b[1:0]
=
d
eq[3:0]
ix6
a[1:0]
b[1:0]
=
d
ix12
a[1:0]
b[1:0]
=
d
ix10
a[1:0]
b[1:0]
=
d


S
R
CLK
ix40
in
out
ix9
in
out
ix43
data[1:0]
eq[3:0]
decoder_2
ix14
in[0]
in[1]
out
ix16
in[1]
in[0]
out
reg_Q
S
D
CE
Q
R
ix1
in
out
Q
Q_B


data[1:0]
ix8
a[1:0]
b[1:0]
=
d
eq[3:0]
ix6
a[1:0]
b[1:0]
=
d
ix12
a[1:0]
b[1:0]
=
d
ix10
a[1:0]
b[1:0]
=
d


K
J
ix46
data[1:0]
eq[3:0]
decoder_2
ix14
in[1]
in[0]
out
ix16
in[1]
in[0]
out
modgen_mux_3
ix1
in
out
Q
Q_B
reg_Q
S
D
Q
CE
R
CLK
R_B
ix9
in
out


data[1:0]
ix8
a[1:0]
b[1:0]
=
d
eq[3:0]
ix6
a[1:0]
b[1:0]
=
d
ix12
a[1:0]
b[1:0]
=
d
ix10
a[1:0]
b[1:0]
=
d


Q
lat_Q
S
ix1
D
D
Q
in
out
Q_B
G
G
R