```
/*      REG4    */
module  REG4    ( CLR_B, D, CLK, Q );
input   CLR_B, CLK;
input   [3:0] D;
output  [3:0] Q;
reg     [3:0] Q;
        always  @( posedge CLK or negedge CLR_B  )
                if ( !CLR_B )
                        Q <= 0;
                else
                        Q <= D;
endmodule
```

```
/*      REG4    */
module  REG4    ( CLR_B, D, CLK, Q );
input   CLR_B, CLK;
input   [3:0] D;
output  [3:0] Q;
wire    [3:0] Q_B;
        R_SYDFF         R_SYDFF0        ( CLR_B, D[0], CLK, Q[0], Q_B[0] ),
                        R_SYDFF1        ( CLR_B, D[1], CLK, Q[1], Q_B[1] ),
                        R_SYDFF2        ( CLR_B, D[2], CLK, Q[2], Q_B[2] ),
                        R_SYDFF3        ( CLR_B, D[3], CLK, Q[3], Q_B[3] );
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
```

```
/*      SIN_POUT_SHIFT  */
module  SIN_POUT_SHIFT  ( RESET_B, IN, CLK, Q );
input   RESET_B, CLK, IN;
output  [3:0] Q;
reg     [3:0] Q;
        always  @( posedge CLK or negedge RESET_B  )
                if ( !RESET_B )
                        Q <= 0;
                else
                        Q <= {Q,IN};
endmodule
```

```
/*      SIN_POUT_SHIFT  */
module  SIN_POUT_SHIFT  ( RESET_B, IN, CLK, Q );
input   RESET_B, CLK, IN;
output  [3:0] Q;
wire    [3:0] Q_B;
        R_SYDFF         R_SYDFF0        ( RESET_B,   IN, CLK, Q[0], Q_B[0] ),
                        R_SYDFF1        ( RESET_B, Q[0], CLK, Q[1], Q_B[1] ),
                        R_SYDFF2        ( RESET_B, Q[1], CLK, Q[2], Q_B[2] ),
                        R_SYDFF3        ( RESET_B, Q[2], CLK, Q[3], Q_B[3] );
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
```

```
/*      PIN_SOUT_SHIFT  */
module  PIN_SOUT_SHIFT  ( LOAD, IN, CLK, Q );
input   LOAD, CLK;
input   [3:0] IN;
output  [3:0] Q;
reg     [3:0] Q;
        always  @( posedge CLK or posedge LOAD  )
                if      ( LOAD )
                        Q <= IN;
                else
                        Q <= Q << 1;    // Q <= { Q, 0 }でも可
endmodule
```

```
/*      CNT16   */
module  CNT16   ( RESET_B, CLK, Q );
input   RESET_B, CLK;
output  [3:0] Q;
wire    [3:0] Q_B;
        R_SYDFF  R_SYDFF0  ( RESET_B, Q_B[0],    CLK, Q[0], Q_B[0] ),
                 R_SYDFF1  ( RESET_B, Q_B[1], Q_B[0], Q[1], Q_B[1] ),
                 R_SYDFF2  ( RESET_B, Q_B[2], Q_B[1], Q[2], Q_B[2] ),
                 R_SYDFF3  ( RESET_B, Q_B[3], Q_B[2], Q[3], Q_B[3] );
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
```

```
/*      CNT4    */
module  CNT4    ( RESET_B, CLK, Q );
input   RESET_B, CLK;
output  [1:0] Q;
wire    [1:0] Q_B;
        R_SYDFF  R_SYDFF0  ( RESET_B, Q_B[0],    CLK, Q[0], Q_B[0] ),
                 R_SYDFF1  ( RESET_B, Q_B[1], Q_B[0], Q[1], Q_B[1] );
endmodule

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

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

```
/*      CNT10   */
module  CNT10   ( RESET_B, CLK, Q );
input   RESET_B, CLK;
output  [3:0] Q;
wire    [3:0] Q_B;
wire    ALFA ,BETA;
parameter       DELTA_G = 5;
        assign  #DELTA_G        ALFA  = ~( Q[3] & Q[1] );
        assign  #DELTA_G        BETA = RESET_B & ALFA;

        R_SYDFF  R_SYDFF0  ( BETA, Q_B[0],    CLK, Q[0], Q_B[0] ),
                 R_SYDFF1  ( BETA, Q_B[1], Q_B[0], Q[1], Q_B[1] ),
                 R_SYDFF2  ( BETA, Q_B[2], Q_B[1], Q[2], Q_B[2] ),
                 R_SYDFF3  ( BETA, Q_B[3], Q_B[2], Q[3], Q_B[3] );
endmodule

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

```
/*      CNT4    */
module  CNT4    ( RESET_B, CLK, Q );
input   RESET_B, CLK;
output  [1:0] Q;
wire    [1:0] Q_B;

        R_SYDFF  R_SYDFF0  ( RESET_B, Q_B[0]      , CLK, Q[0], Q_B[0] ),
                 R_SYDFF1  ( RESET_B, Q[1] ^ Q[0], CLK, Q[1], Q_B[1] );
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
```

```
/*      CNT4    */
module  CNT4    ( RESET_B, CLK, Q );
input   RESET_B, CLK;
output  [1:0] Q;
reg     [1:0] Q;
        always  @( posedge CLK or negedge RESET_B )
                if ( !RESET_B )
                        Q <= 0;
                else
                        begin
                                Q[1] <= Q[1] ^ Q[0];
                                Q[0] <= ~Q[0];
                        end
endmodule
```

```
/*      CNT4    */
module  CNT4    ( RESET_B, CLK, Q );
input   RESET_B, CLK;
output  [1:0] Q;
reg     [1:0] Q;
        always  @( posedge CLK or negedge RESET_B )
                if ( !RESET_B )
                        Q <= 0;
                else
                        Q <= Q + 1;
endmodule
```

```
/*      CNT10   */
module  CNT10   ( RESET_B, CLK, Q );
input   RESET_B, CLK;
output  [3:0] Q;
reg     [3:0] Q;
        always  @( posedge CLK or negedge RESET_B )
                if ( !RESET_B )
                        Q <= 0;
                else
                        begin
                                Q[3] <= Q[2] & Q[1] & Q[0]
                                        | Q[3] & ~Q[0];
                                Q[2] <= Q[2] & ~Q[1]
                                        | Q[2] & ~Q[0]
                                        | ~Q[2] & Q[1] & Q[0];
                                Q[1] <= Q[1] & ~Q[0]
                                        | ~Q[3] & ~Q[1] & Q[0];
                                Q[0] <= ~Q[0];
                        end
endmodule
```

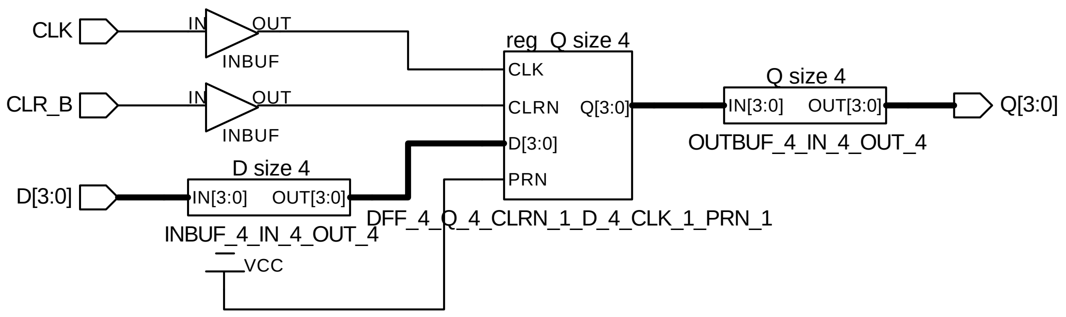
```
/*      CNT10   */
module  CNT10   ( RESET_B, CLK, Q );
input   RESET_B, CLK;
output  [3:0] Q;
reg     [3:0] Q;
        always  @( posedge CLK or negedge RESET_B )
                if ( !RESET_B )
                        Q <= 0;
                else if ( Q == 9 )
                        Q <= 0;
                else
                        Q <= Q + 1;
endmodule
```

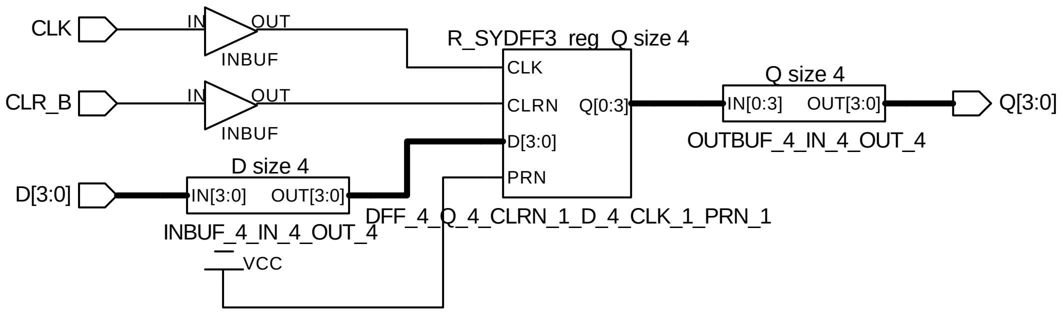
```
/*      CNT10   */
module  CNT10   ( RESET_B, CLK, Q );
input   RESET_B, CLK;
output  [3:0] Q;
reg     [3:0] Q;
        always  @( posedge CLK )
                if ( !RESET_B )
                        Q <= 0;
                else if ( Q == 9 )
                        Q <= 0;
                else
                        Q <= Q + 1;
endmodule
```

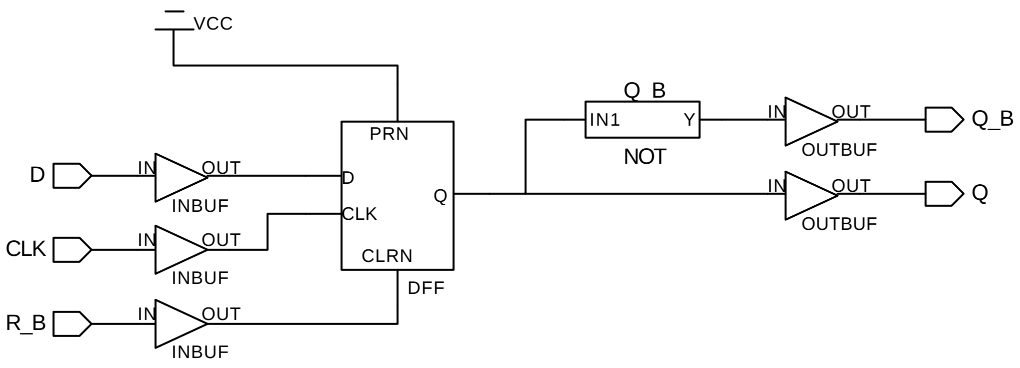
```
/*      CNT256  */
module  CNT256  ( RESET_B, CLK, Q );
input   RESET_B, CLK;
output  [7:0] Q;
reg     [7:0] Q;
        always  @( posedge CLK )
                if ( !RESET_B )
                        Q <= 0;
                else
                        Q <= Q + 1;
endmodule
```

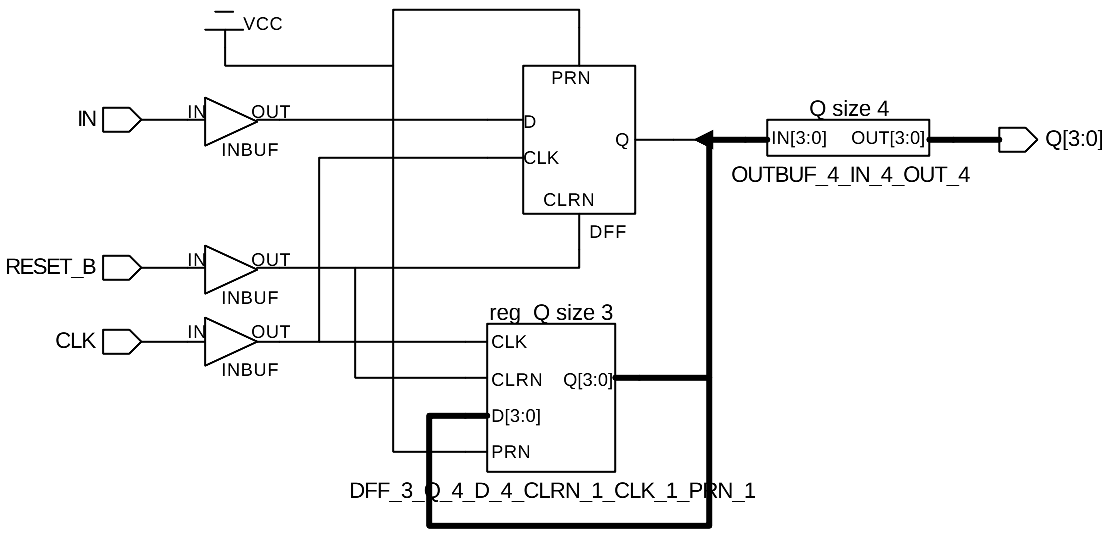
```
/*      CNT_X   */
module  CNT_X   ( RESET_B, CLK, Q );
input   RESET_B, CLK;
output  [2:0] Q;
reg     [2:0] Q;
        always  @( posedge CLK or negedge RESET_B )
                if ( !RESET_B )
                        Q <= 0;
                else
                        begin
                                Q[0] <= ~Q[0];
                                Q[1] <= ~Q[2] & ~Q[1] & Q[0]
                                         | Q[1] & ~Q[0];
                                Q[2] <= Q[1] & Q[0] | Q[2] & ~Q[0];
                        end
endmodule
```

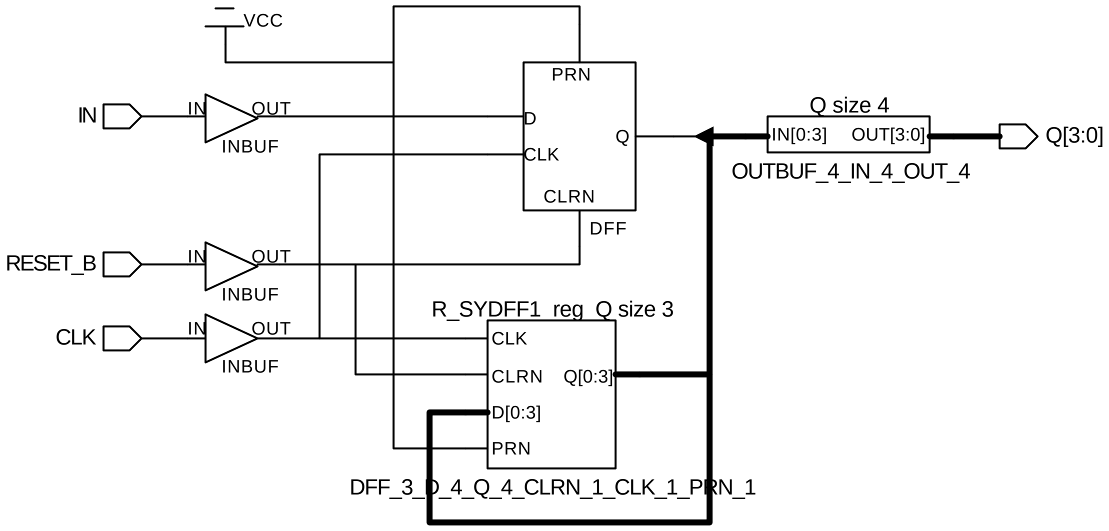
```
/*      CNT12   */
module  CNT12   ( RESET_B, CLK, Q );
input   RESET_B, CLK;
output  [3:0] Q;
wire    [3:0] J, K, Q_B;

        assign  J[0] = 1;
        assign  K[0] = 1;
        assign  J[1] = Q[0];
        assign  K[1] = Q[0];
        assign  J[2] = ~Q[3] & Q[1] & Q[0];
        assign  K[2] = Q[1] & Q[0];
        assign  J[3] = Q[2] & Q[1] & Q[0];
        assign  K[3] = Q[1] & Q[0];

        R_SYJKFF JKFF0  ( RESET_B, J[0], K[0], CLK, Q[0], Q_B[0] ),
                 JKFF1  ( RESET_B, J[1], K[1], CLK, Q[1], Q_B[1] ),
                 JKFF2  ( RESET_B, J[2], K[2], CLK, Q[2], Q_B[2] ),
                 JKFF3  ( RESET_B, J[3], K[3], CLK, Q[3], Q_B[3] );
endmodule

/*      R_SYJKFF        */
/*      FIG7-58 参照    */
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
```

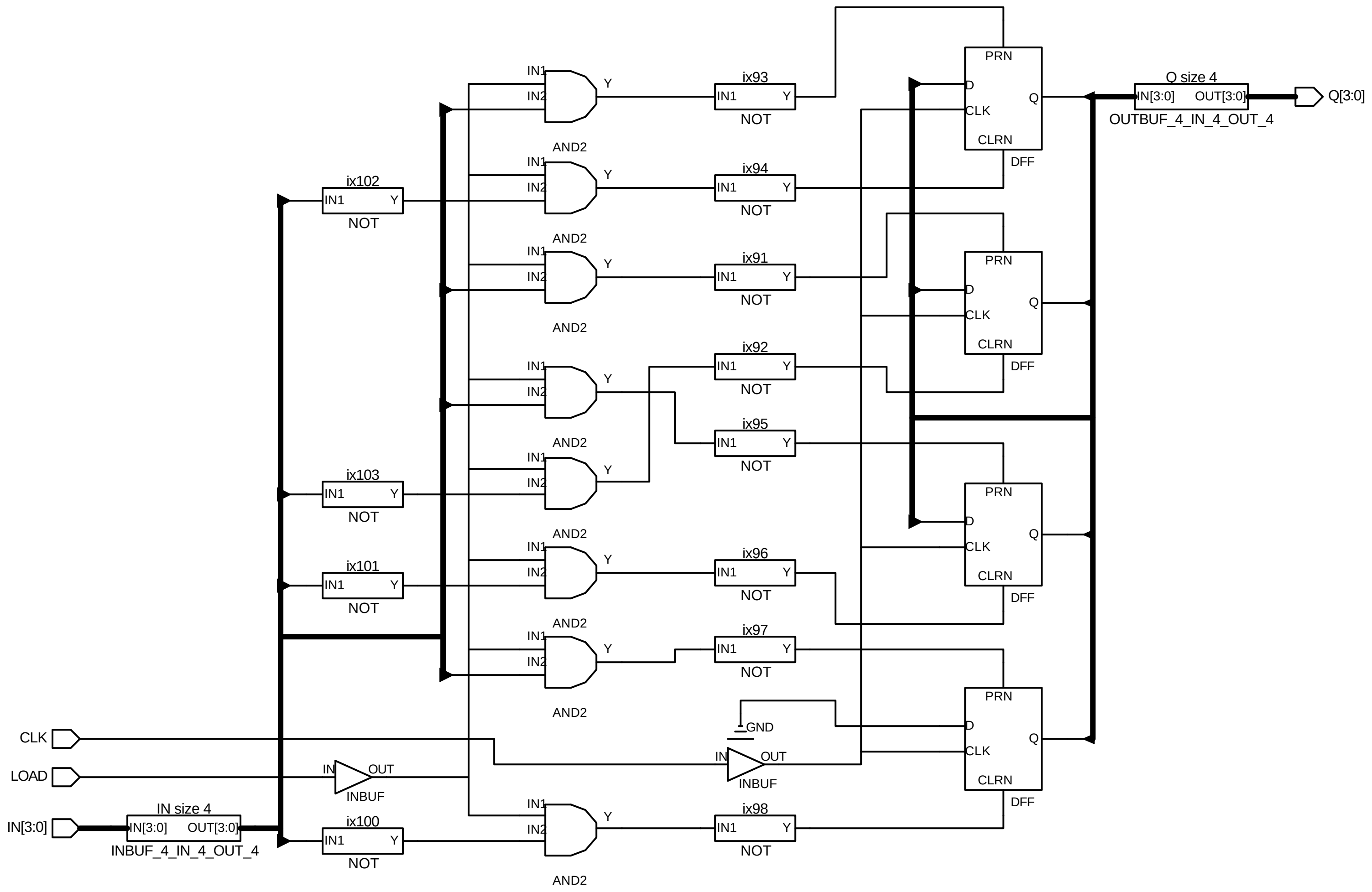

CLK
IN
OUT
INBUF
CLR_B
IN
OUT
INBUF
D[3:0]
D size 4
IN[3:0]
OUT[3:0]
INBUF_4_IN_4_OUT_4
VCC
reg_Q size 4
CLK
CLRN
D[3:0]
PRN
Q[3:0]
DFF_4_Q_4_CLRN_1_D_4_CLK_1_PRN_1
Q size 4
IN[3:0]
OUT[3:0]
OUTBUF_4_IN_4_OUT_4
Q[3:0]

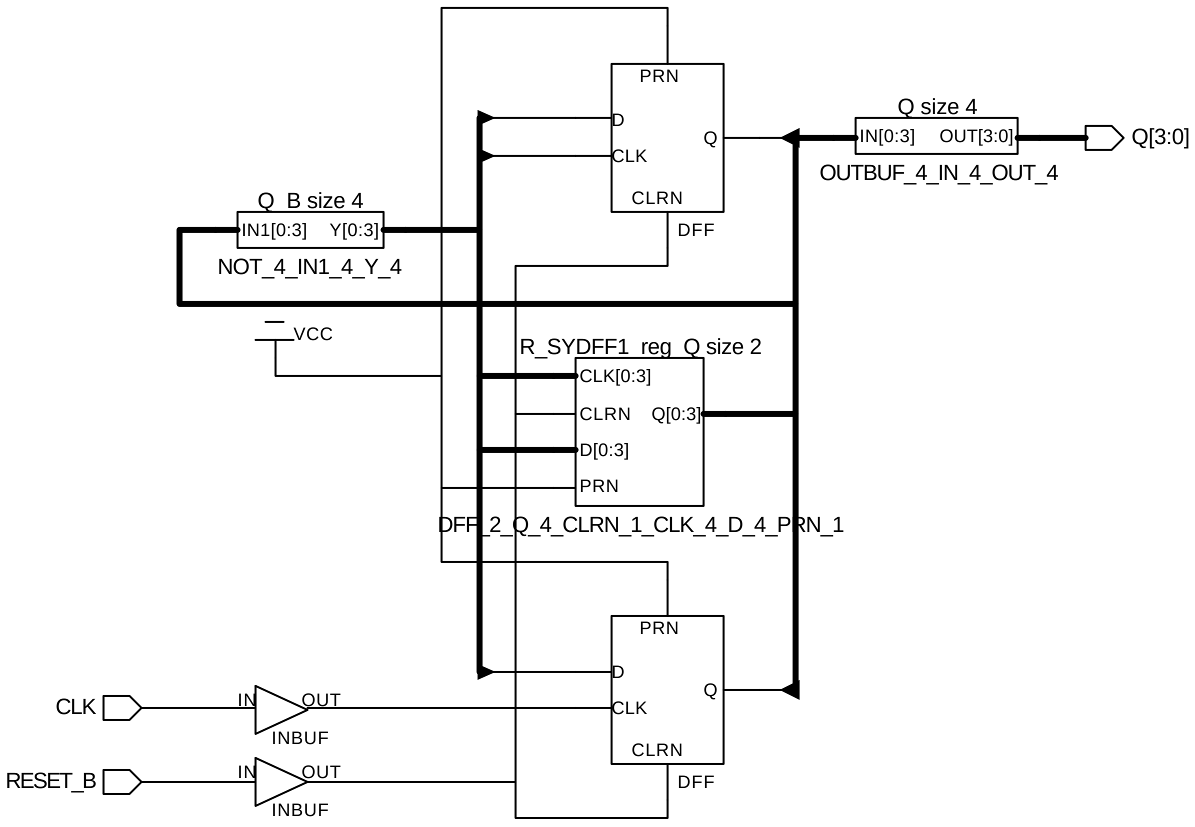

CLK
IN
OUT
INBUF
CLR_B
IN
OUT
INBUF
D[3:0]
D size 4
IN[3:0]
OUT[3:0]
INBUF_4_IN_4_OUT_4
VCC
R_SYDFF3_reg_Q size 4
CLK
CLRN
D[3:0]
PRN
Q[0:3]
DFF_4_Q_4_CLRN_1_D_4_CLK_1_PRN_1
Q size 4
IN[0:3]
OUT[3:0]
OUTBUF_4_IN_4_OUT_4
Q[3:0]


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

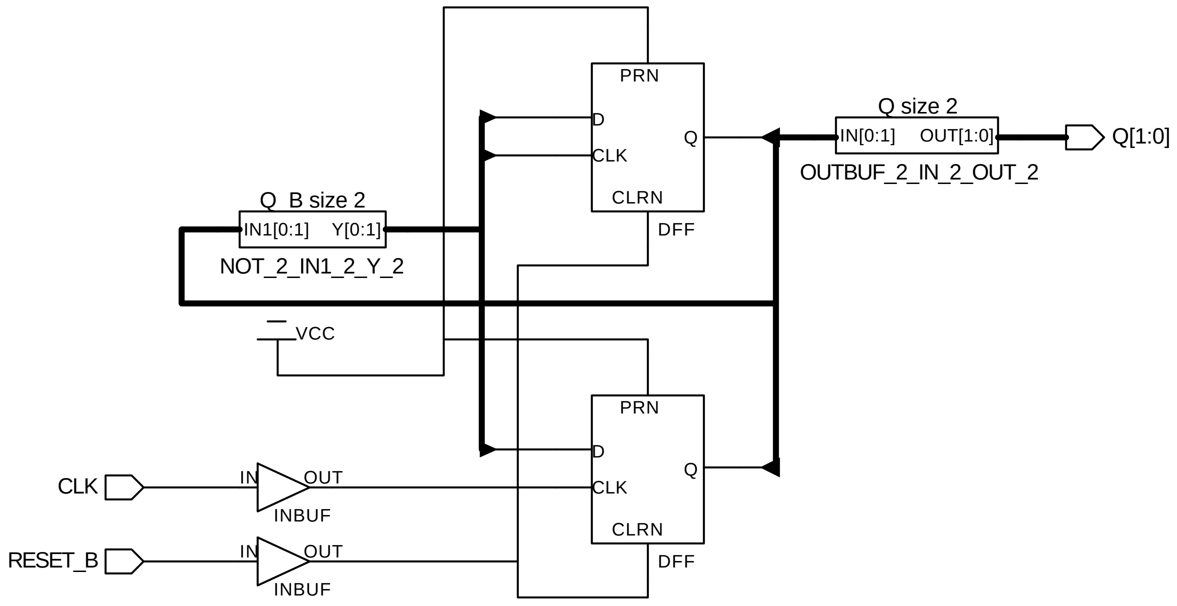

VCC
IN
IN
OUT
INBUF
RESET_B
IN
OUT
INBUF
CLK
IN
OUT
INBUF
PRN
D
CLK
Q
CLRN
DFF
Q size 4
IN[3:0]
OUT[3:0]
OUTBUF_4_IN_4_OUT_4
Q[3:0]
reg_Q size 3
CLK
CLRN
Q[3:0]
D[3:0]
PRN
DFF_3_Q_4_D_4_CLRN_1_CLK_1_PRN_1

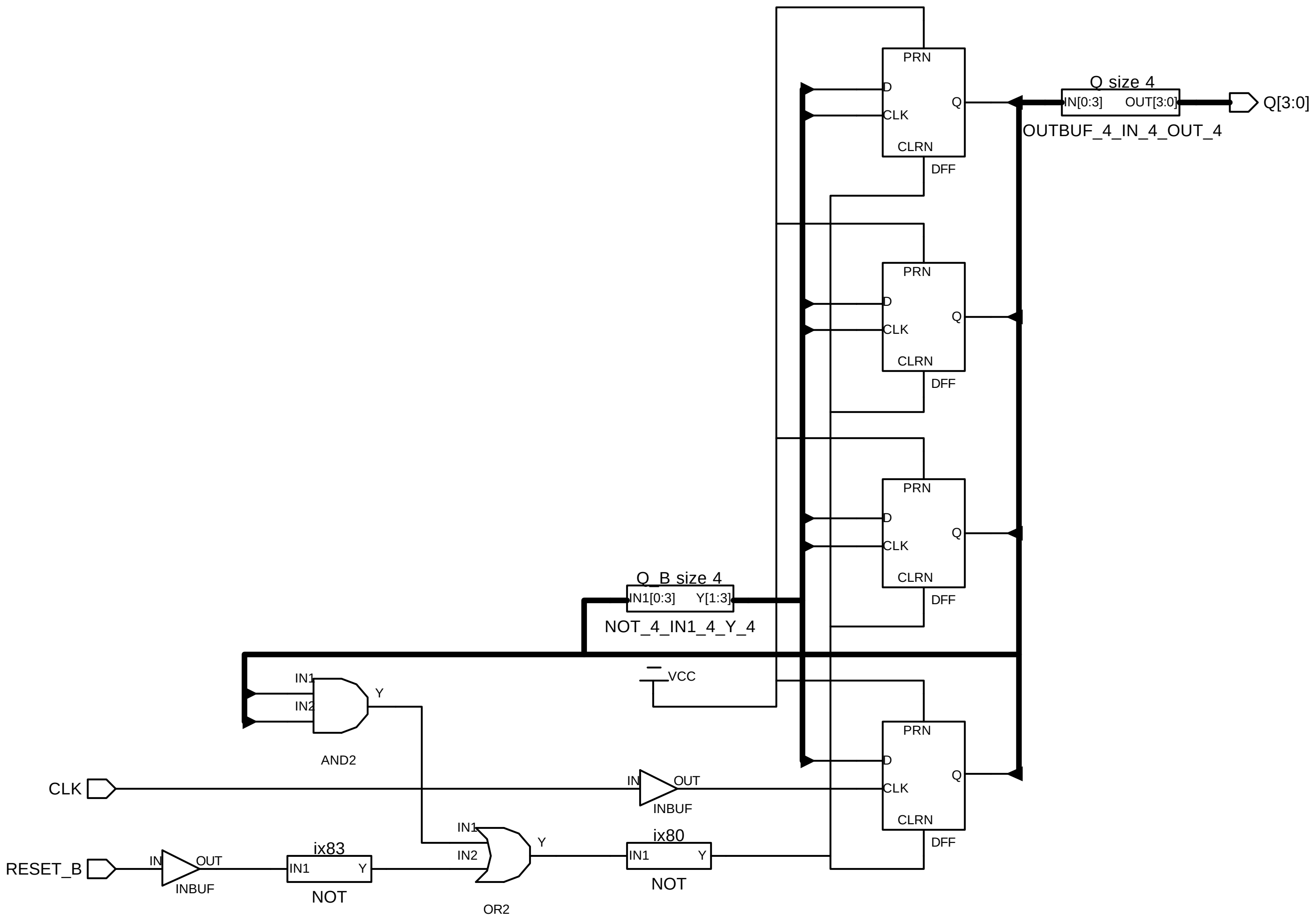

VCC
IN
IN
OUT
INBUF
RESET_B
IN
OUT
INBUF
CLK
IN
OUT
INBUF
PRN
D
CLK
Q
CLRN
DFF
R_SYDFF1_reg_Q size 3
CLK
CLRN
Q[0:3]
D[0:3]
PRN
DFF_3_D_4_Q_4_CLRN_1_CLK_1_PRN_1
Q size 4
IN[0:3]
OUT[3:0]
OUTBUF_4_IN_4_OUT_4
Q[3:0]

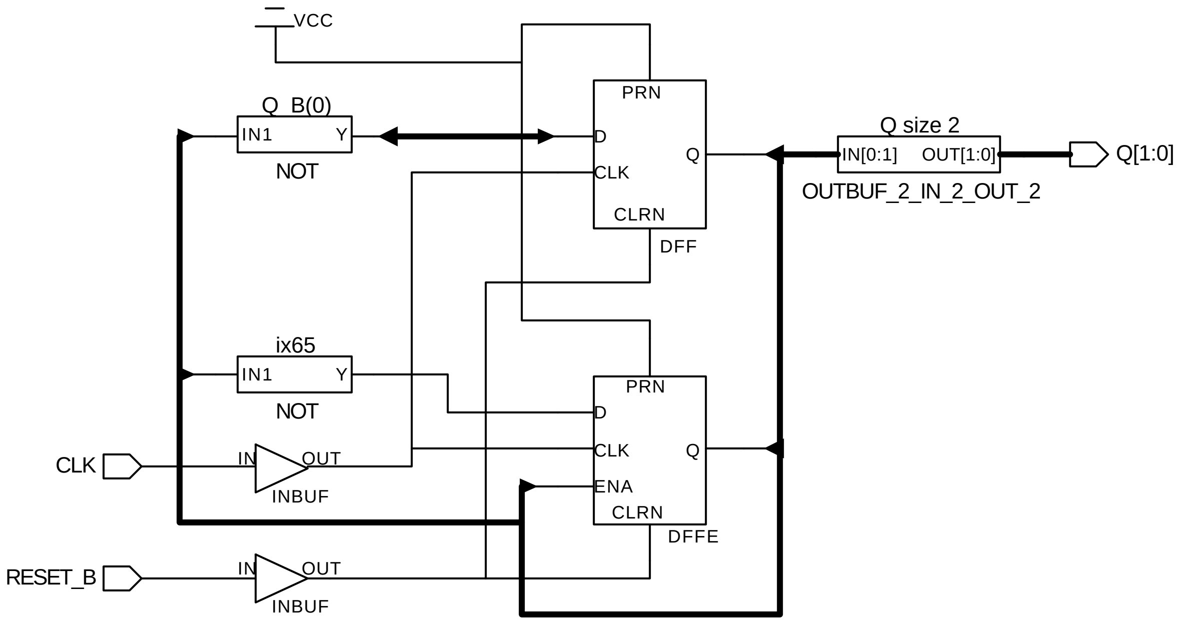

CLK
LOAD
IN[3:0]
IN size 4
IN[3:0]
OUT[3:0]
INBUF_4_IN_4_OUT_4
IN
OUT
INBUF
ix102
ix103
ix101
ix100
NOT
AND2
IN1
IN2
Y
ix93
ix94
ix91
ix92
ix95
ix96
ix97
ix98
GND
PRN
D
CLK
CLRN
Q
DFF
Q size 4
IN[3:0]
OUT[3:0]
OUTBUF_4_IN_4_OUT_4
Q[3:0]

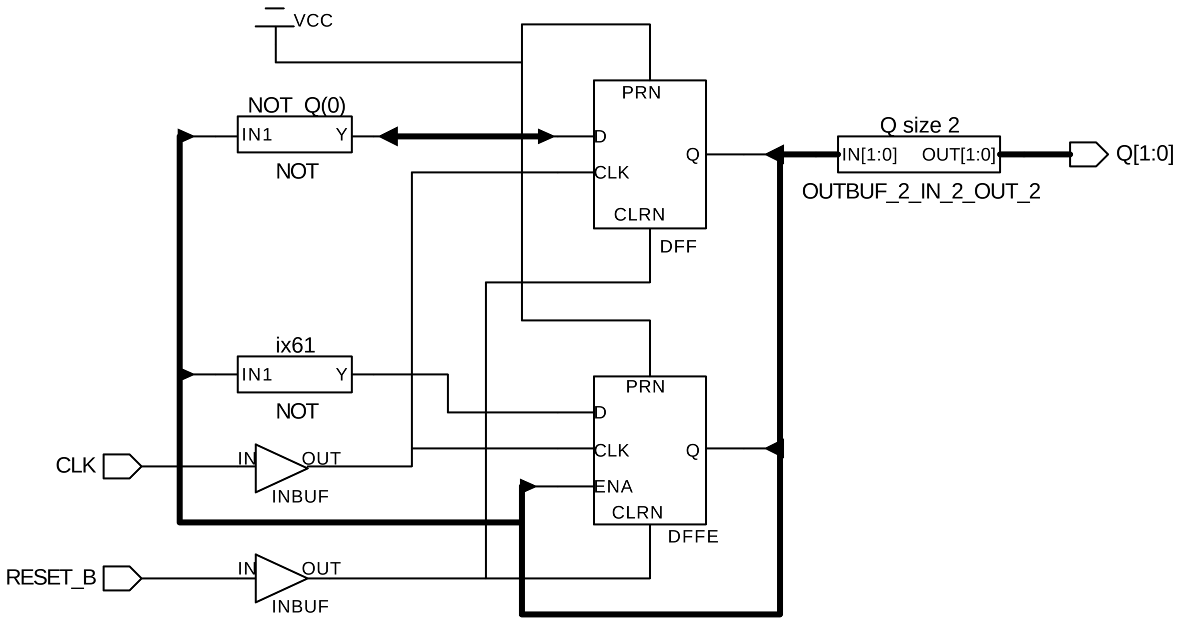


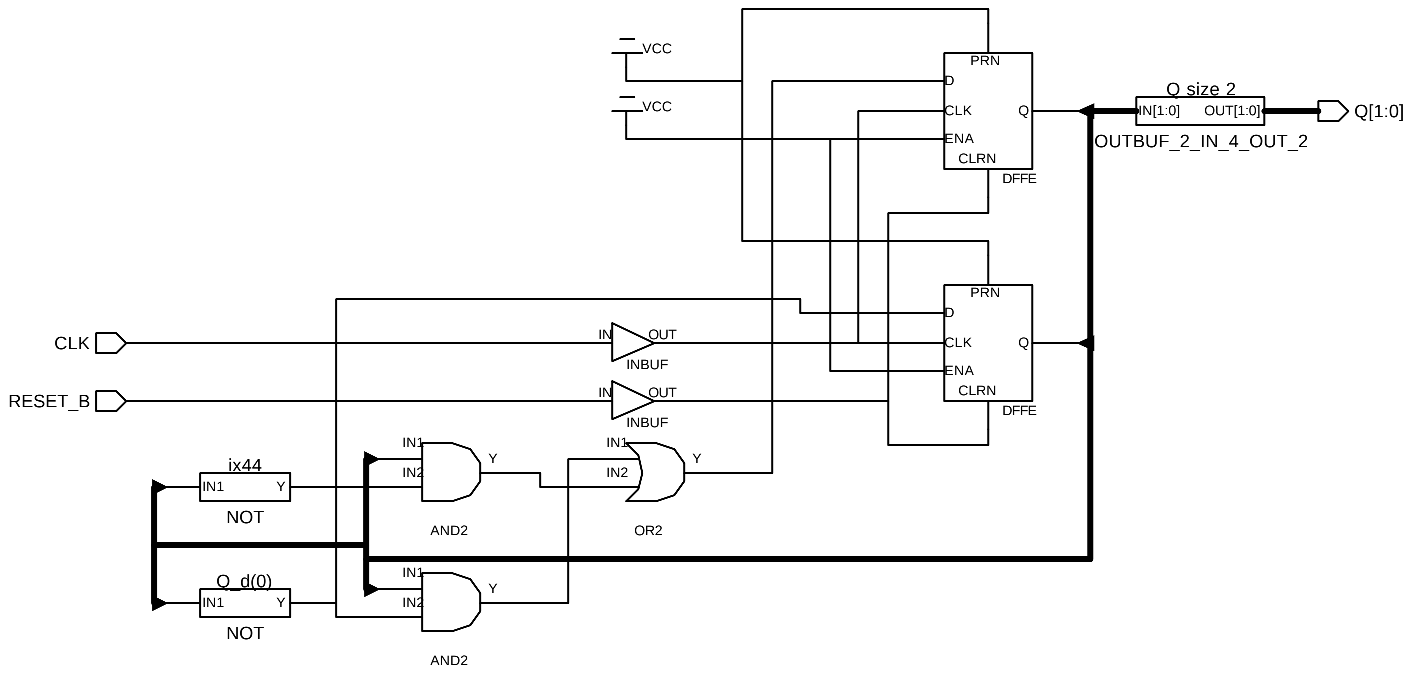
PRN
D
Q
CLK
CLRN
DFF
Q size 4
IN[0:3]
OUT[3:0]
Q[3:0]
OUTBUF_4_IN_4_OUT_4
Q_B size 4
IN1[0:3]
Y[0:3]
NOT_4_IN1_4_Y_4
VCC
R_SYDFF1_reg_Q size 2
CLK[0:3]
CLRN
Q[0:3]
D[0:3]
PRN
DFF_2_Q_4_CLRN_1_CLK_4_D_4_PRN_1
PRN
D
Q
CLK
CLRN
DFF
CLK
IN
OUT
INBUF
RESET_B
IN
OUT
INBUF

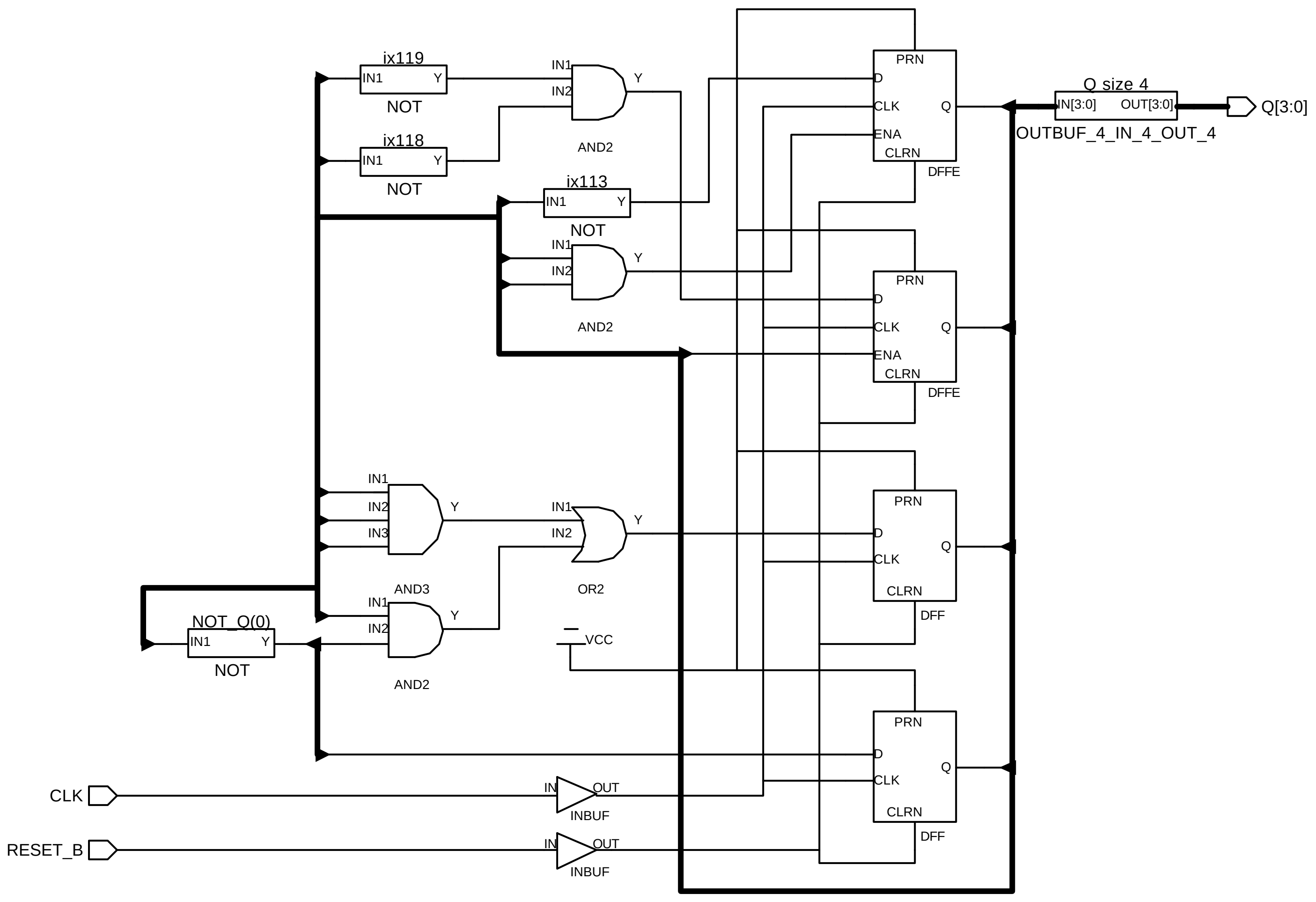

PRN
D
CLK
Q
CLRN
DFF
Q size 2
IN[0:1]
OUT[1:0]
Q[1:0]
OUTBUF_2_IN_2_OUT_2
Q_B size 2
IN1[0:1]
Y[0:1]
NOT_2_IN1_2_Y_2
VCC
PRN
D
CLK
Q
CLRN
DFF
CLK
IN
OUT
INBUF
RESET_B
IN
OUT
INBUF


PRN
D
CLK
Q
CLRN
DFF
Q size 4
IN[0:3]
OUT[3:0]
Q[3:0]
OUTBUF_4_IN_4_OUT_4
Q_B size 4
IN1[0:3]
Y[1:3]
NOT_4_IN1_4_Y_4
VCC
IN1
IN2
Y
AND2
CLK
IN
OUT
INBUF
RESET_B
ix83
NOT
OR2
ix80
NOT

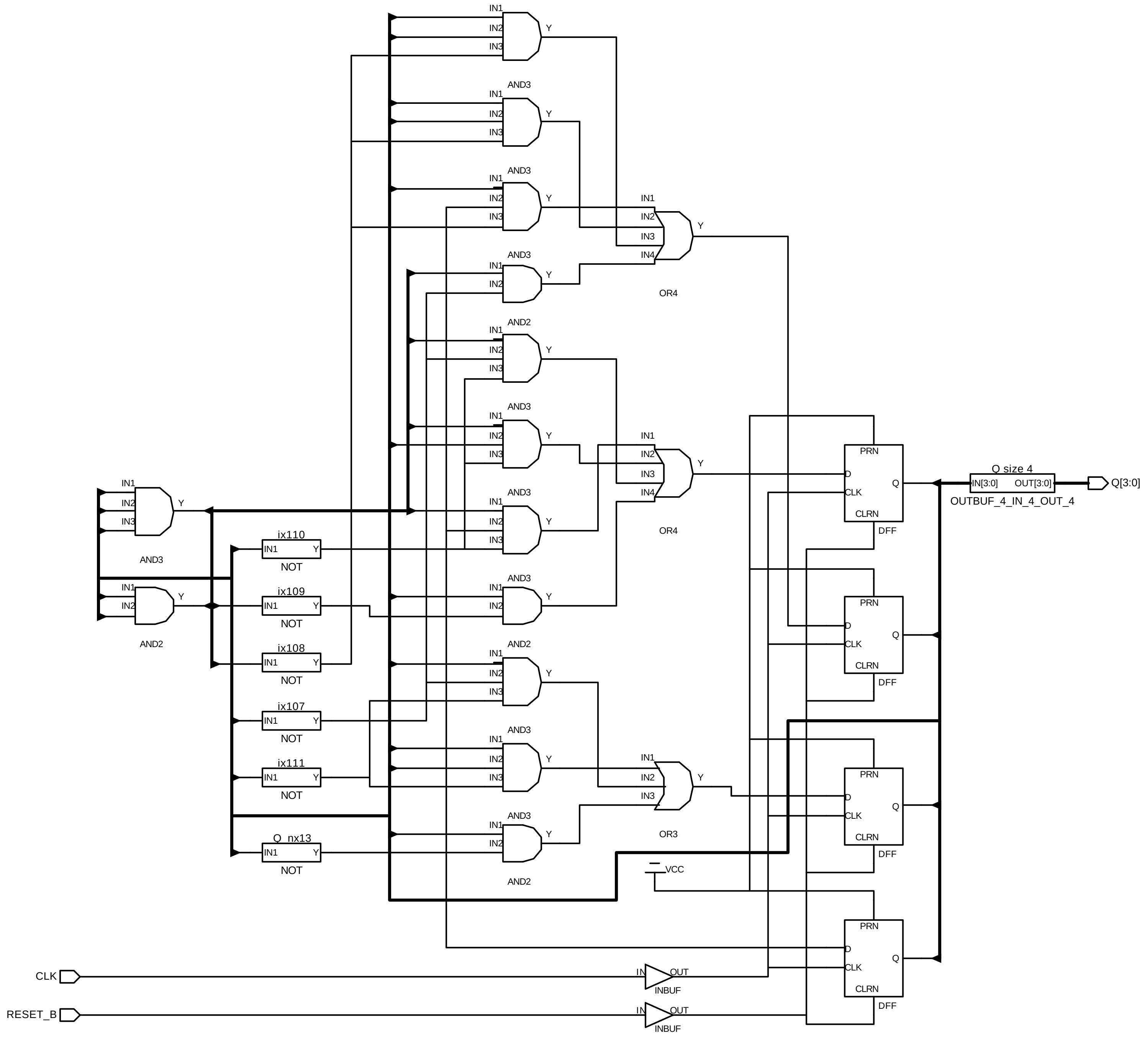

VCC
Q_B(0)
IN1
Y
NOT
PRN
D
CLK
Q
CLRN
DFF
Q size 2
IN[0:1]
OUT[1:0]
Q[1:0]
OUTBUF_2_IN_2_OUT_2
ix65
IN1
Y
NOT
PRN
D
CLK
Q
ENA
CLRN
DFFE
CLK
IN
OUT
INBUF
RESET_B
IN
OUT
INBUF

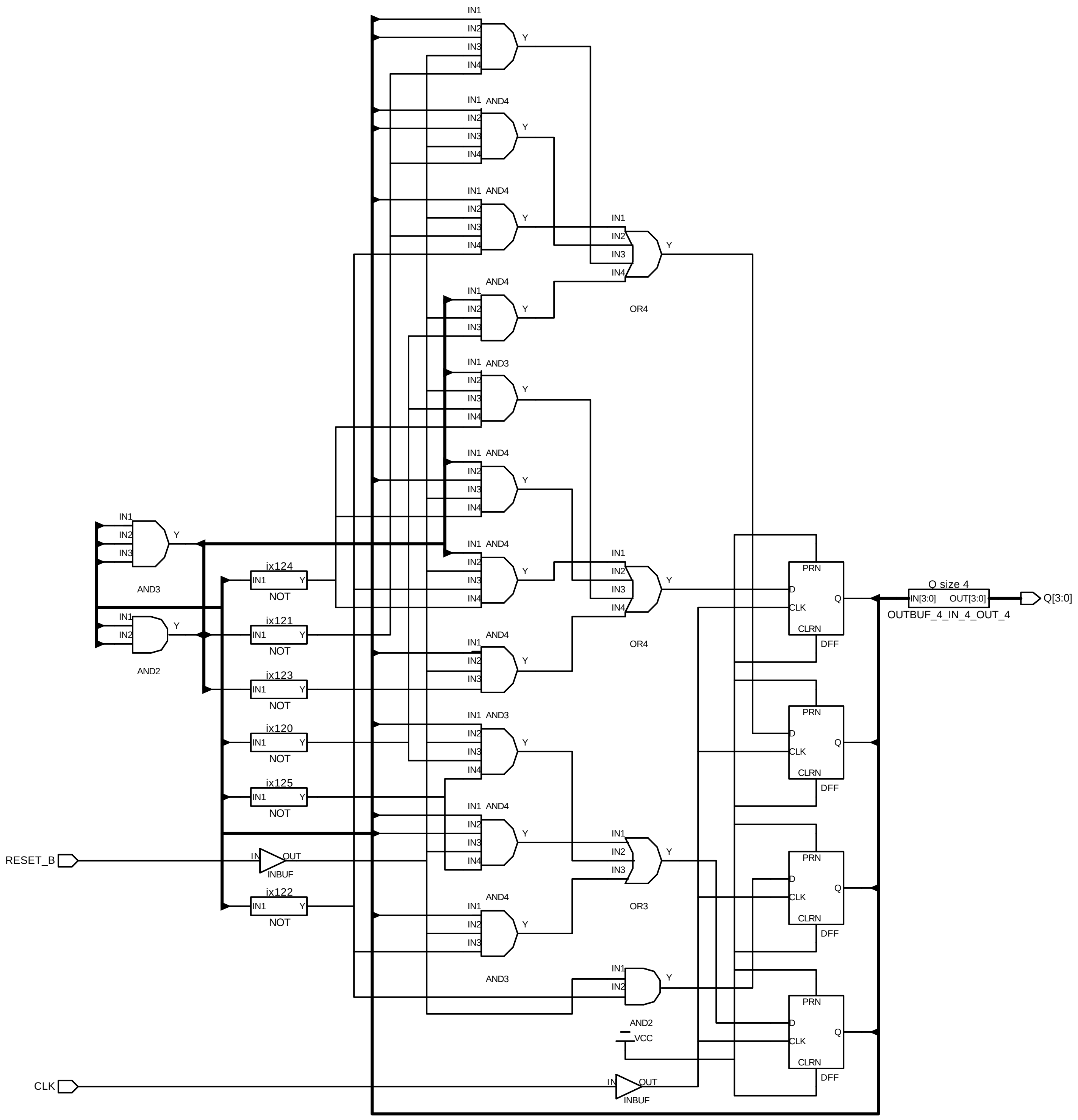

VCC
NOT_Q(0)
IN1
Y
NOT
PRN
D
CLK
Q
CLRN
DFF
Q size 2
IN[1:0]
OUT[1:0]
Q[1:0]
OUTBUF_2_IN_2_OUT_2
ix61
IN1
Y
NOT
PRN
D
CLK
Q
ENA
CLRN
DFFE
CLK
IN
OUT
INBUF
RESET_B
IN
OUT
INBUF


VCC
VCC
PRN
D
CLK
ENA
CLRN
Q
DFFE
Q_size_2
IN[1:0]
OUT[1:0]
OUTBUF_2_IN_4_OUT_2
Q[1:0]
PRN
D
CLK
ENA
CLRN
Q
DFFE
CLK
IN
OUT
INBUF
RESET_B
IN
OUT
INBUF
ix44
IN1
Y
NOT
IN1
IN2
Y
AND2
IN1
IN2
Y
OR2
Q_d(0)
IN1
Y
NOT
IN1
IN2
Y
AND2

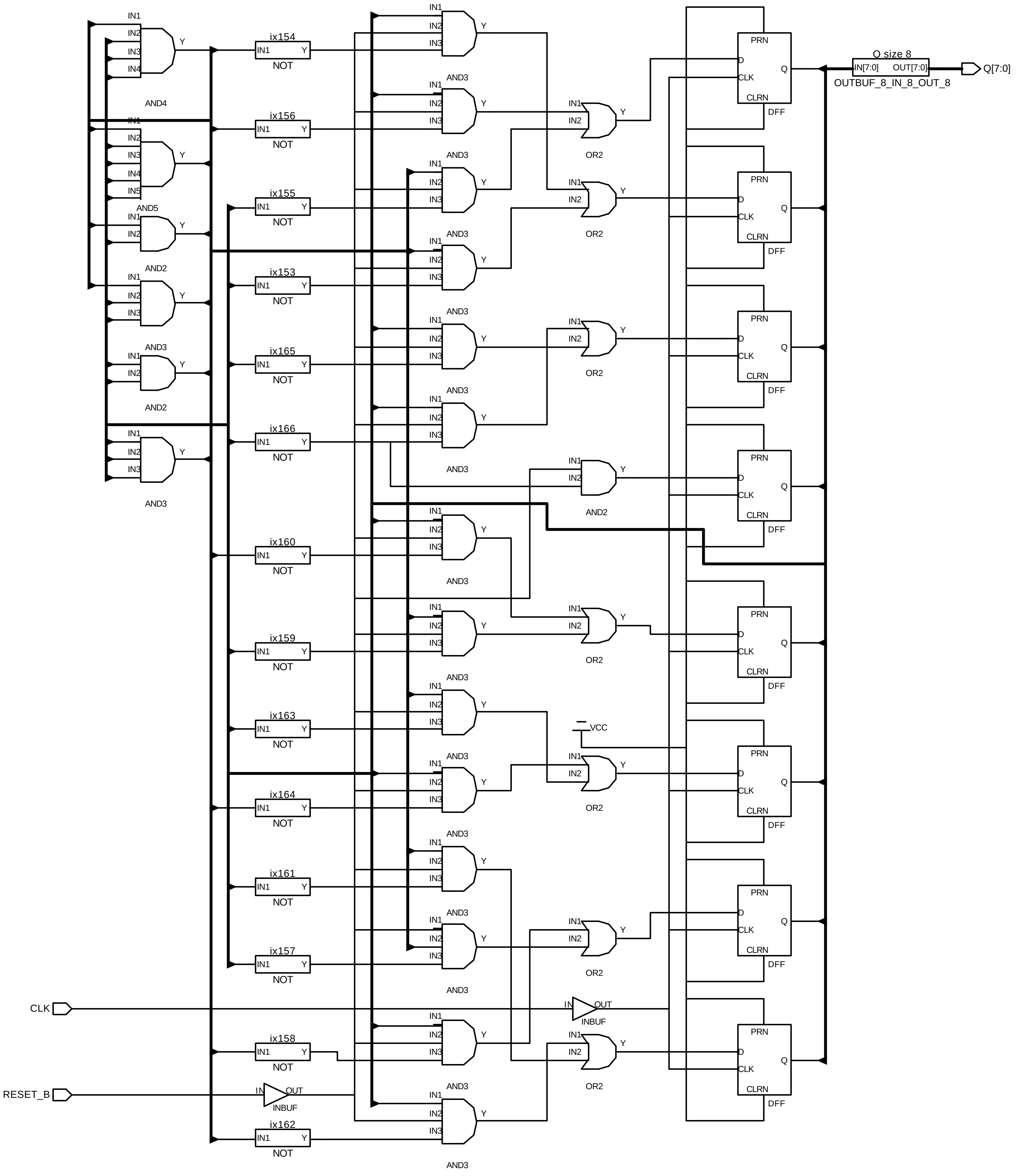


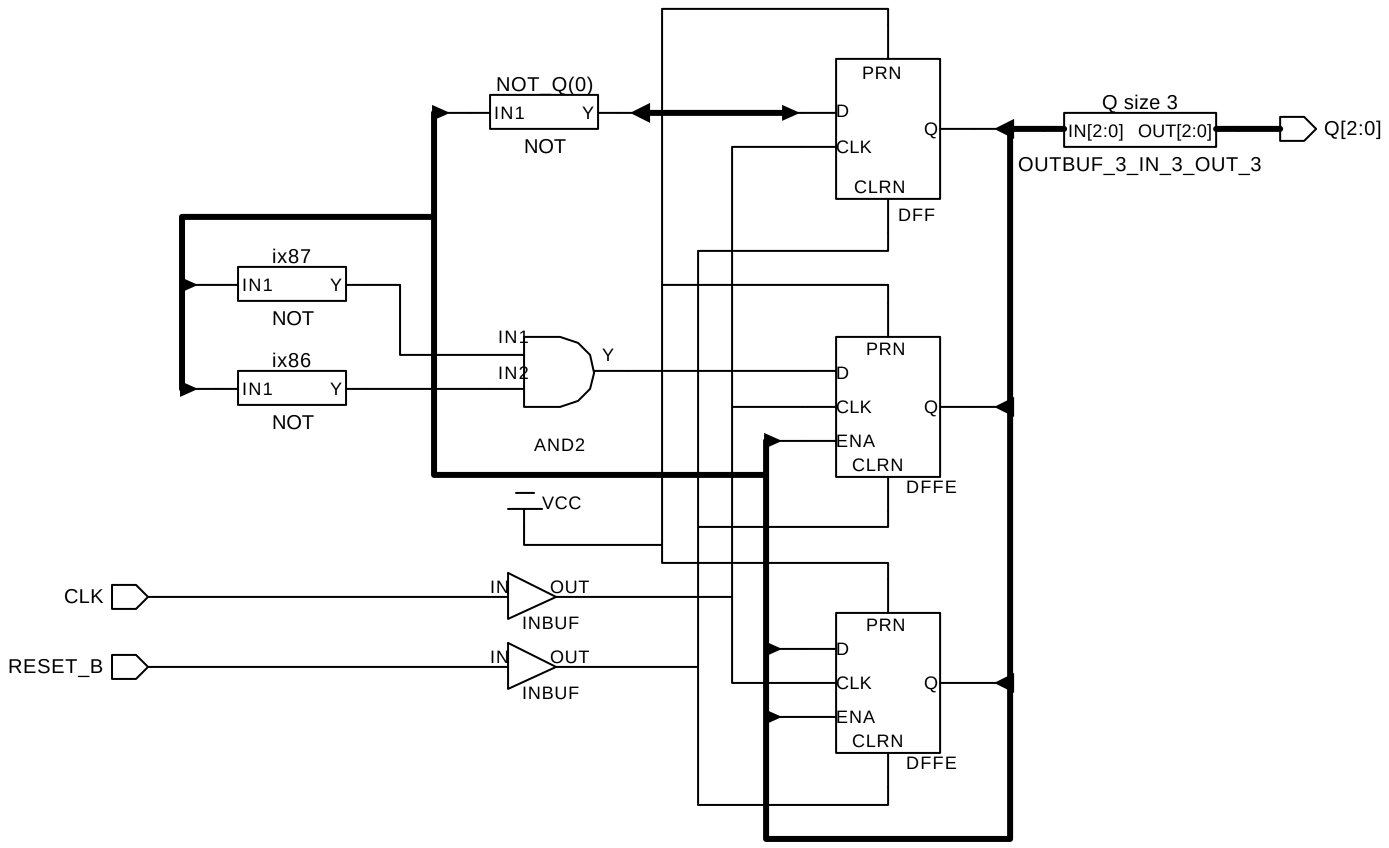
ix119
IN1
Y
NOT
ix118
IN1
Y
NOT
IN1
IN2
Y
AND2
ix113
IN1
Y
NOT
IN1
IN2
Y
AND2
PRN
D
CLK
Q
ENA
CLRN
DFFE
Q size 4
IN[3:0]
OUT[3:0]
OUTBUF_4_IN_4_OUT_4
Q[3:0]
PRN
D
CLK
Q
ENA
CLRN
DFFE
IN1
IN2
IN3
Y
AND3
IN1
IN2
Y
OR2
PRN
D
CLK
Q
CLRN
DFF
NOT_Q(0)
IN1
Y
NOT
IN1
IN2
Y
AND2
VCC
PRN
D
CLK
Q
CLRN
DFF
CLK
IN
OUT
INBUF
RESET_B
IN
OUT
INBUF

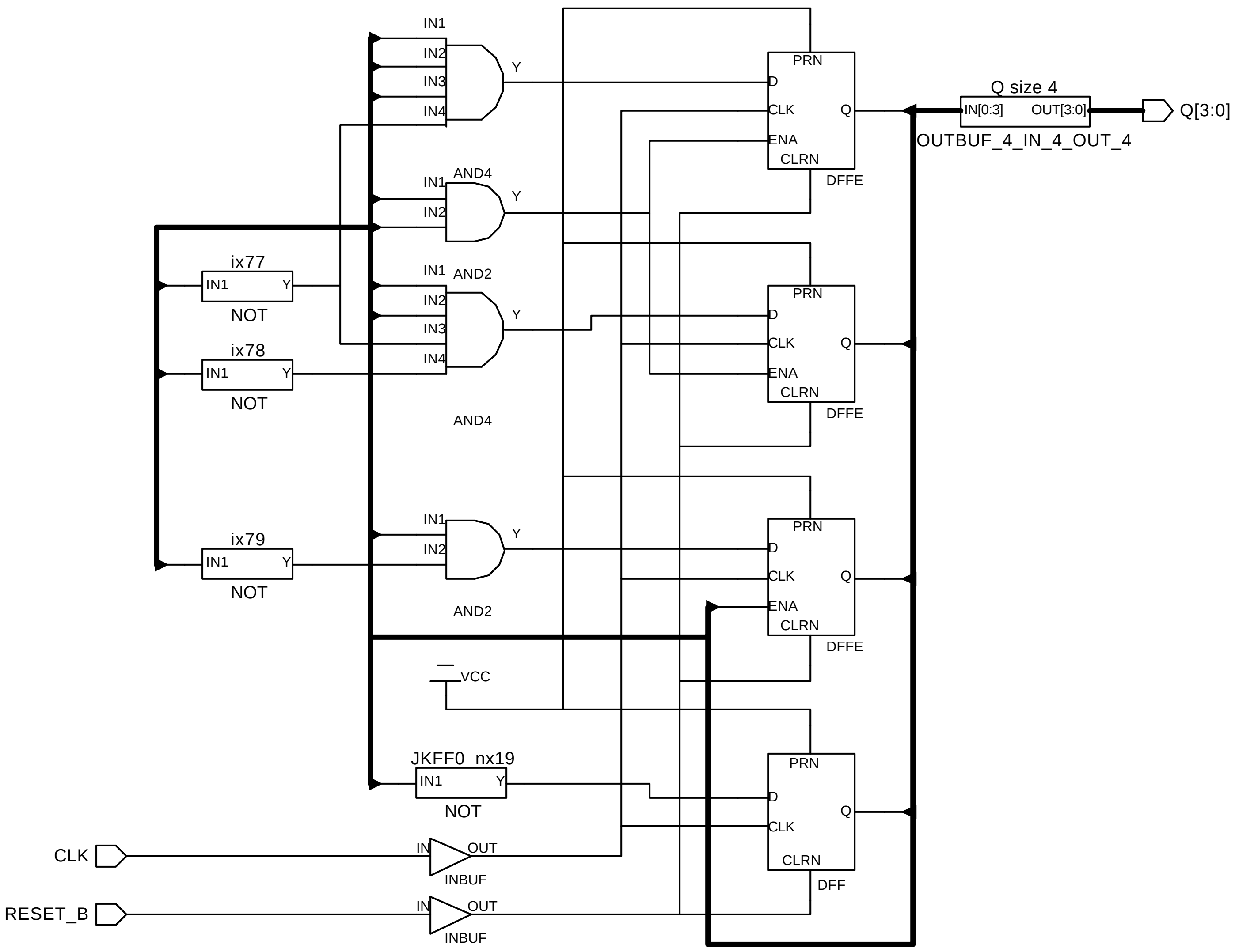

AND3
AND3
AND3
AND2
AND3
AND3
AND3
AND2
AND3
AND3
AND2
OR4
OR4
OR3
IN1
IN2
IN3
IN4
Y
ix110
ix109
ix108
ix107
ix111
Q_nx13
NOT
NOT
NOT
NOT
NOT
NOT
VCC
PRN
D
CLK
CLRN
Q
DFF
DFF
DFF
DFF
Q size 4
IN[3:0]
OUT[3:0]
OUTBUF_4_IN_4_OUT_4
Q[3:0]
CLK
RESET_B
IN
OUT
INBUF
INBUF

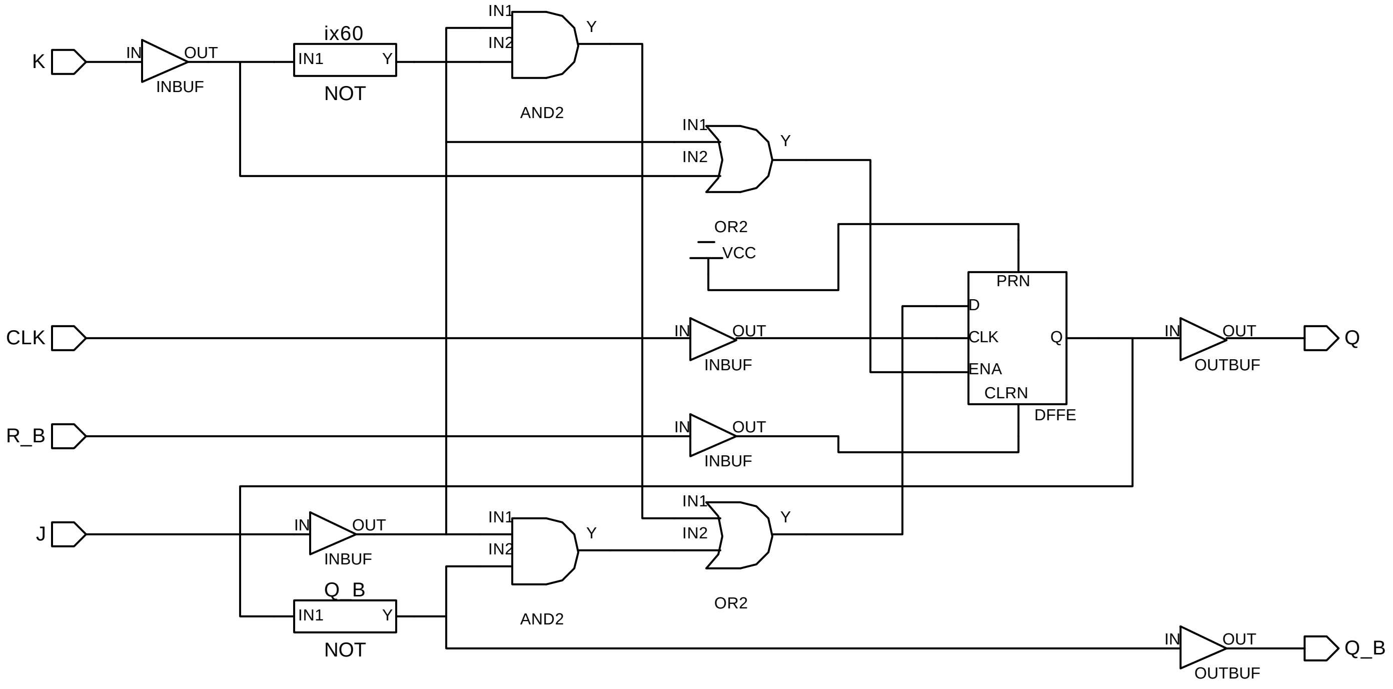


RESET_B
CLK
Q[3:0]
AND3
AND2
AND4
OR4
OR3
NOT
ix124
ix121
ix123
ix120
ix125
ix122
INBUF
VCC
DFF
PRN
CLRN
Q size 4
OUTBUF_4_IN_4_OUT_4
IN[3:0]
OUT[3:0]

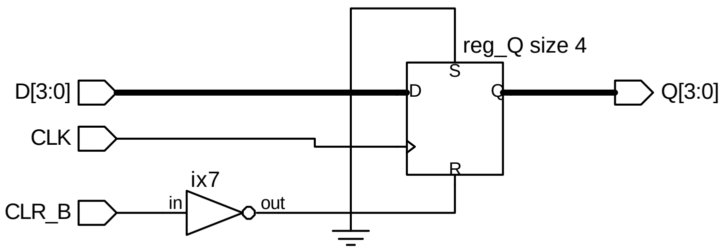

AND4
AND5
AND2
AND3
AND2
AND3
ix154
NOT
ix156
NOT
ix155
NOT
ix153
NOT
ix165
NOT
ix166
NOT
ix160
NOT
ix159
NOT
ix163
NOT
ix164
NOT
ix161
NOT
ix157
NOT
ix158
NOT
ix162
NOT
IN1
IN2
IN3
IN4
IN5
Y
AND3
OR2
AND2
VCC
CLK
RESET_B
INBUF
IN
OUT
PRN
D
CLK
CLRN
Q
DFF
Q size 8
IN[7:0]
OUT[7:0]
OUTBUF_8_IN_8_OUT_8
Q[7:0]

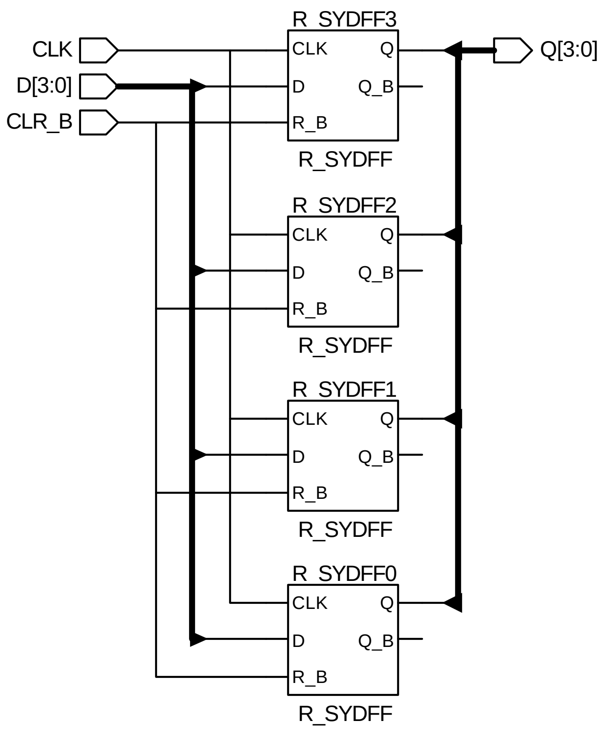

NOT_Q(0)
IN1
Y
NOT
PRN
D
CLK
Q
CLRN
DFF
Q size 3
IN[2:0]
OUT[2:0]
Q[2:0]
OUTBUF_3_IN_3_OUT_3
ix87
IN1
Y
NOT
ix86
IN1
Y
NOT
IN1
IN2
Y
AND2
PRN
D
CLK
Q
ENA
CLRN
DFFE
VCC
CLK
IN
OUT
INBUF
RESET_B
IN
OUT
INBUF
PRN
D
CLK
Q
ENA
CLRN
DFFE

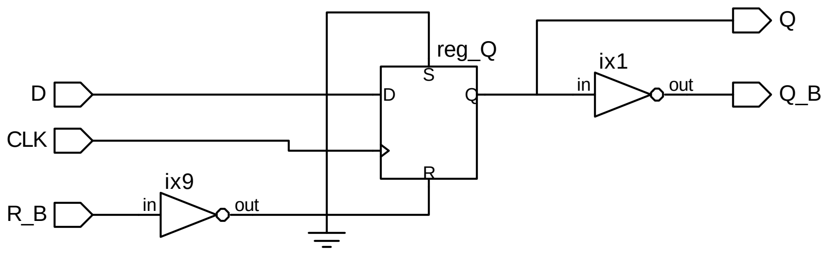

IN1
IN2
IN3
IN4
Y
AND4
IN1
IN2
Y
AND2
IN1
IN2
IN3
IN4
Y
AND4
IN1
IN2
Y
AND2
ix77
IN1
Y
NOT
ix78
IN1
Y
NOT
ix79
IN1
Y
NOT
VCC
JKFF0_nx19
IN1
Y
NOT
CLK
IN
OUT
INBUF
RESET_B
IN
OUT
INBUF
PRN
D
CLK
ENA
CLRN
Q
DFFE
PRN
D
CLK
ENA
CLRN
Q
DFFE
PRN
D
CLK
ENA
CLRN
Q
DFFE
PRN
D
CLK
CLRN
Q
DFF
Q size 4
IN[0:3]
OUT[3:0]
OUTBUF_4_IN_4_OUT_4
Q[3:0]

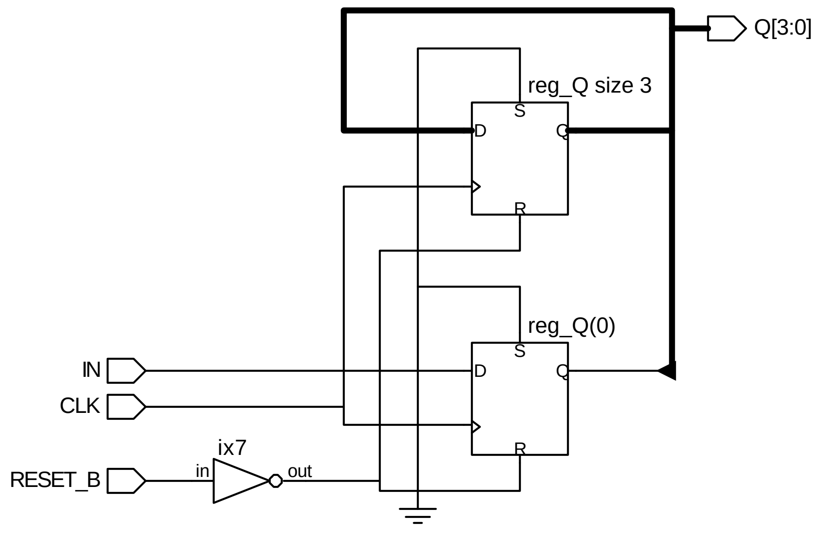


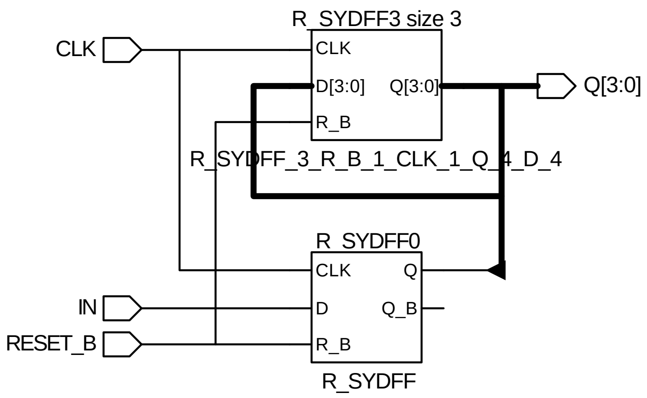
K
IN
OUT
INBUF
ix60
IN1
Y
NOT
IN1
IN2
Y
AND2
IN1
IN2
Y
OR2
VCC
CLK
IN
OUT
INBUF
PRN
D
CLK
ENA
CLRN
Q
DFFE
IN
OUT
OUTBUF
Q
R_B
IN
OUT
INBUF
J
IN
OUT
INBUF
Q_B
IN1
Y
NOT
IN1
IN2
Y
AND2
IN1
IN2
Y
OR2
IN
OUT
OUTBUF
Q_B

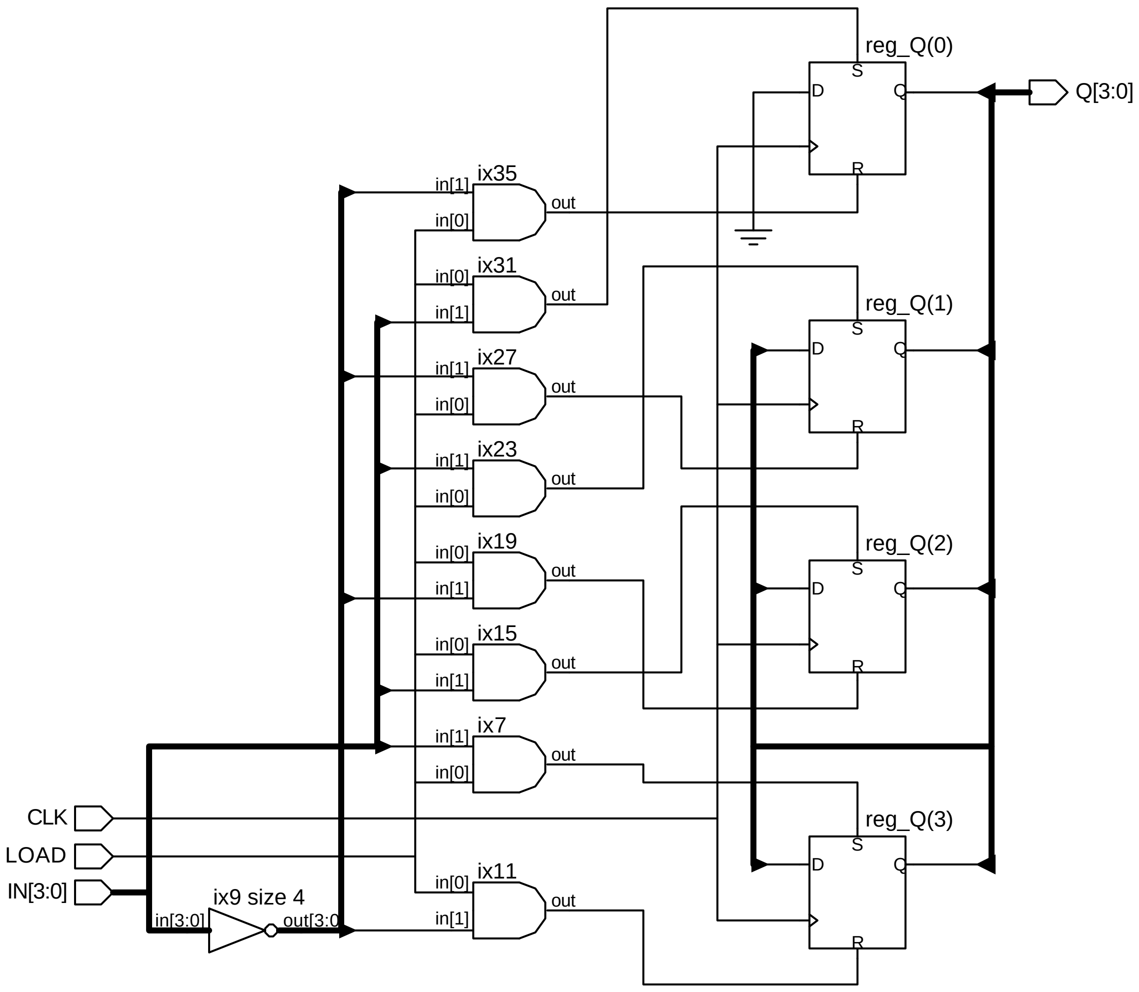

reg_Q size 4
S
D
Q
R
D[3:0]
CLK
CLR_B
ix7
in
out
Q[3:0]


CLK
D[3:0]
CLR_B
R_SYDFF3
CLK
Q
D
Q_B
R_B
R_SYDFF
R_SYDFF2
CLK
Q
D
Q_B
R_B
R_SYDFF
R_SYDFF1
CLK
Q
D
Q_B
R_B
R_SYDFF
R_SYDFF0
CLK
Q
D
Q_B
R_B
R_SYDFF
Q[3:0]

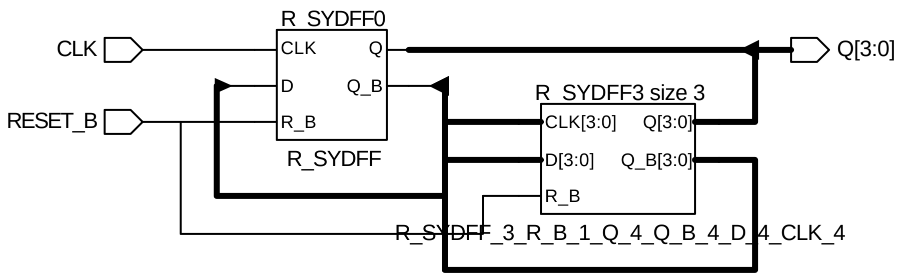

reg_Q
D
CLK
R_B
ix9
in
out
S
D
Q
R
Q
ix1
in
out
Q_B

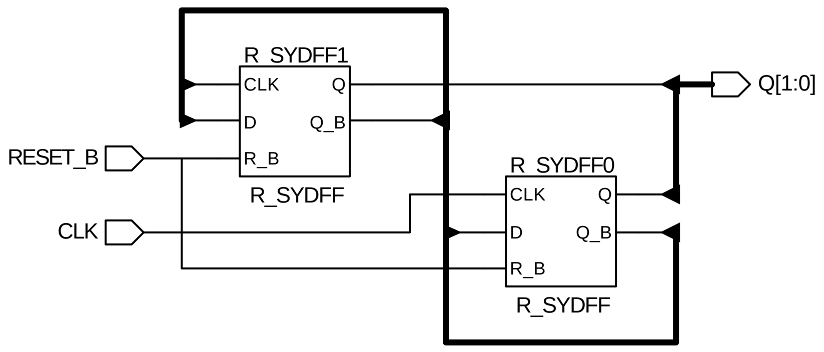

Q[3:0]
reg_Q size 3
S
D
Q
R
reg_Q(0)
S
D
Q
R
IN
CLK
RESET_B
ix7
in
out

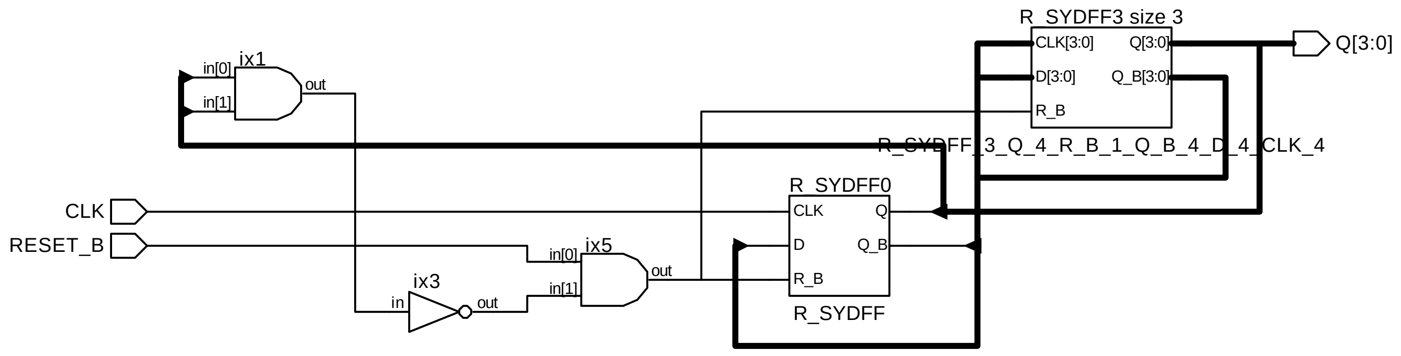

R_SYDFF3 size 3
CLK
CLK
D[3:0]
Q[3:0]
R_B
Q[3:0]
R_SYDFF_3_R_B_1_CLK_1_Q_4_D_4
R_SYDFF0
CLK
Q
IN
D
Q_B
RESET_B
R_B
R_SYDFF

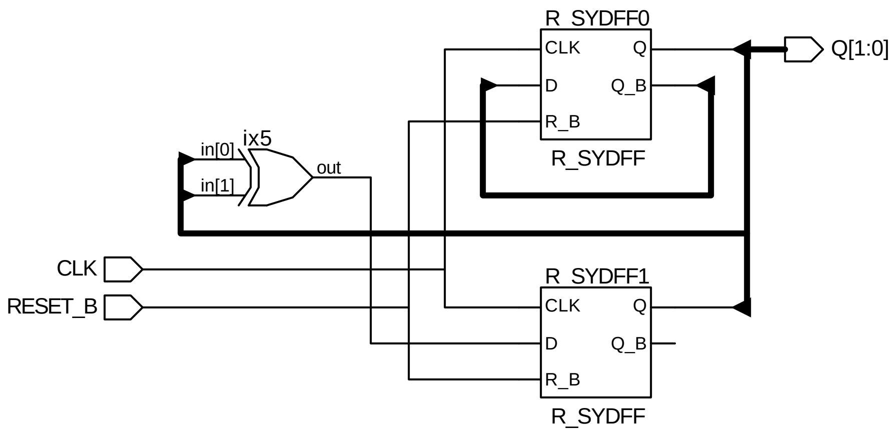


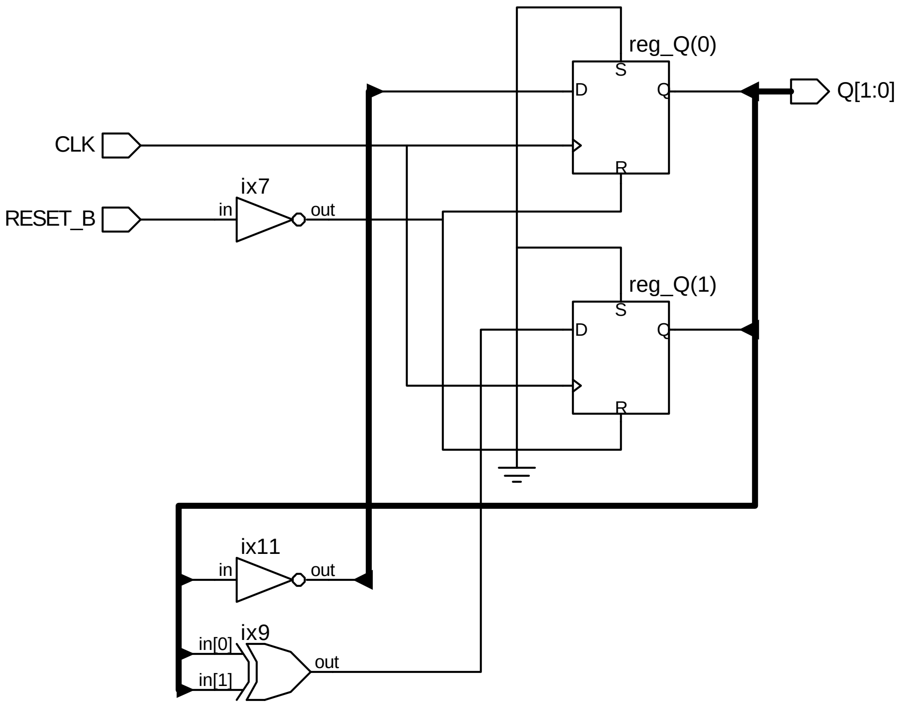
reg_Q(0)
reg_Q(1)
reg_Q(2)
reg_Q(3)
Q[3:0]
ix35
ix31
ix27
ix23
ix19
ix15
ix7
ix11
ix9 size 4
in[3:0]
out[3:0]
in[0]
in[1]
out
D
Q
S
R
CLK
LOAD
IN[3:0]

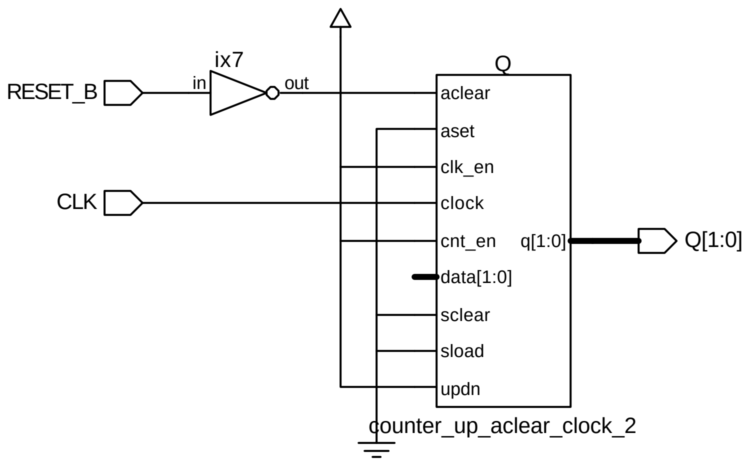

CLK
RESET_B
R_SYDFF0
CLK
D
R_B
Q
Q_B
R_SYDFF
R_SYDFF3 size 3
CLK[3:0]
D[3:0]
R_B
Q[3:0]
Q_B[3:0]
R_SYDFF_3_R_B_1_Q_4_Q_B_4_D_4_CLK_4
Q[3:0]

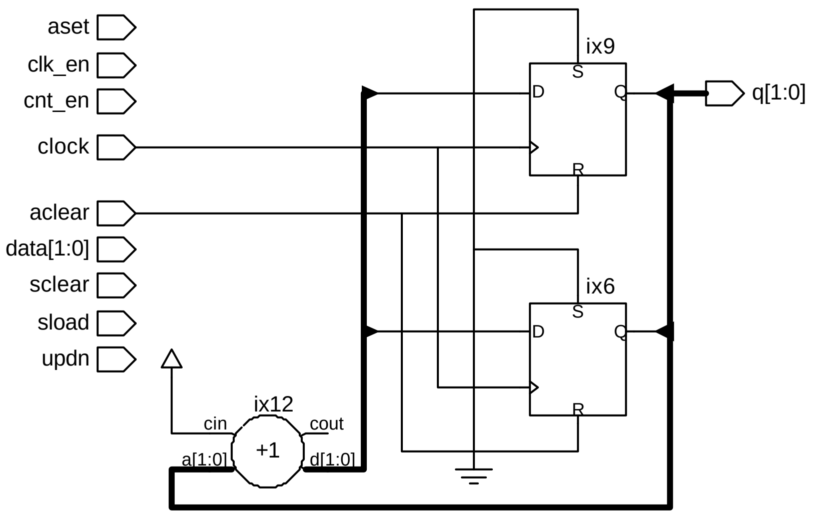

R_SYDFF1
CLK
Q
D
Q_B
R_B
R_SYDFF
RESET_B
CLK
R_SYDFF0
CLK
Q
D
Q_B
R_B
R_SYDFF
Q[1:0]

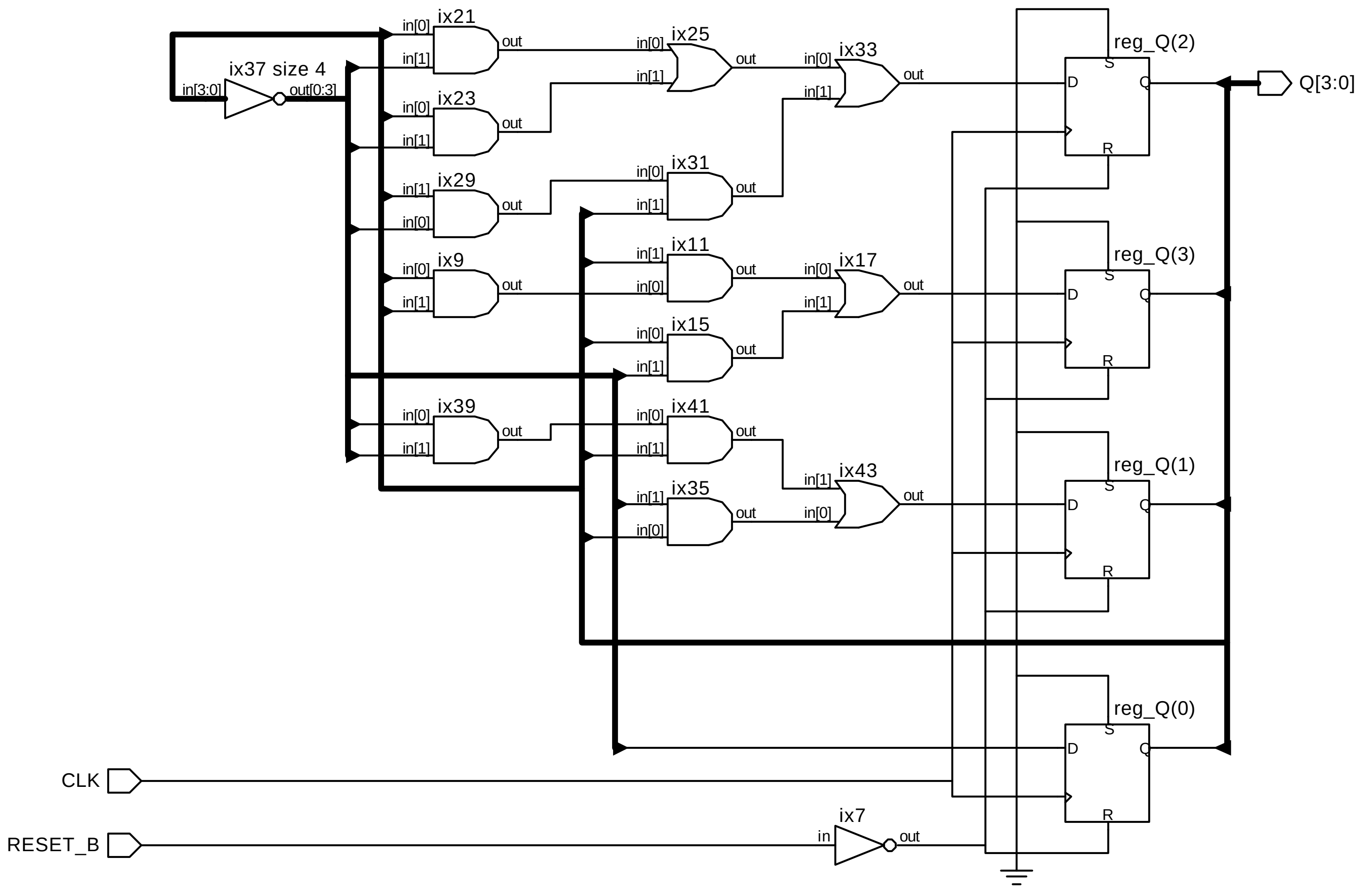

R_SYDFF3 size 3
CLK[3:0]
Q[3:0]
Q[3:0]
D[3:0]
Q_B[3:0]
R_B
ix1
in[0]
in[1]
out
R_SYDFF_3_Q_4_R_B_1_Q_B_4_D_4_CLK_4
R_SYDFF0
CLK
Q
CLK
RESET_B
ix5
in[0]
in[1]
out
D
Q_B
R_B
ix3
in
out
R_SYDFF

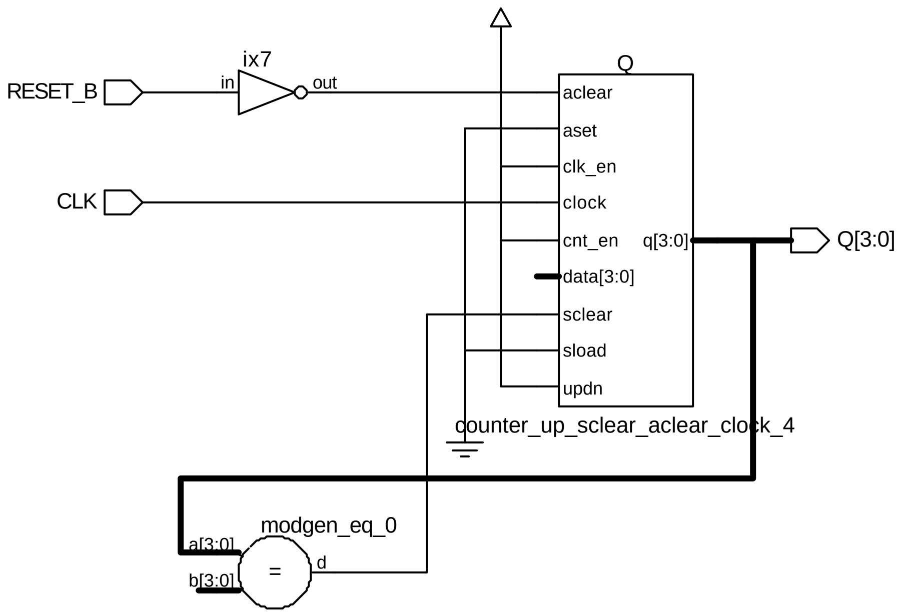

R_SYDFF0
CLK
Q
D
Q_B
R_B
R_SYDFF
Q[1:0]
ix5
in[0]
in[1]
out
CLK
RESET_B
R_SYDFF1
CLK
Q
D
Q_B
R_B
R_SYDFF


reg_Q(0)
S
D
Q
R
Q[1:0]
CLK
ix7
in
out
RESET_B
reg_Q(1)
S
D
Q
R
ix11
in
out
ix9
in[0]
in[1]
out

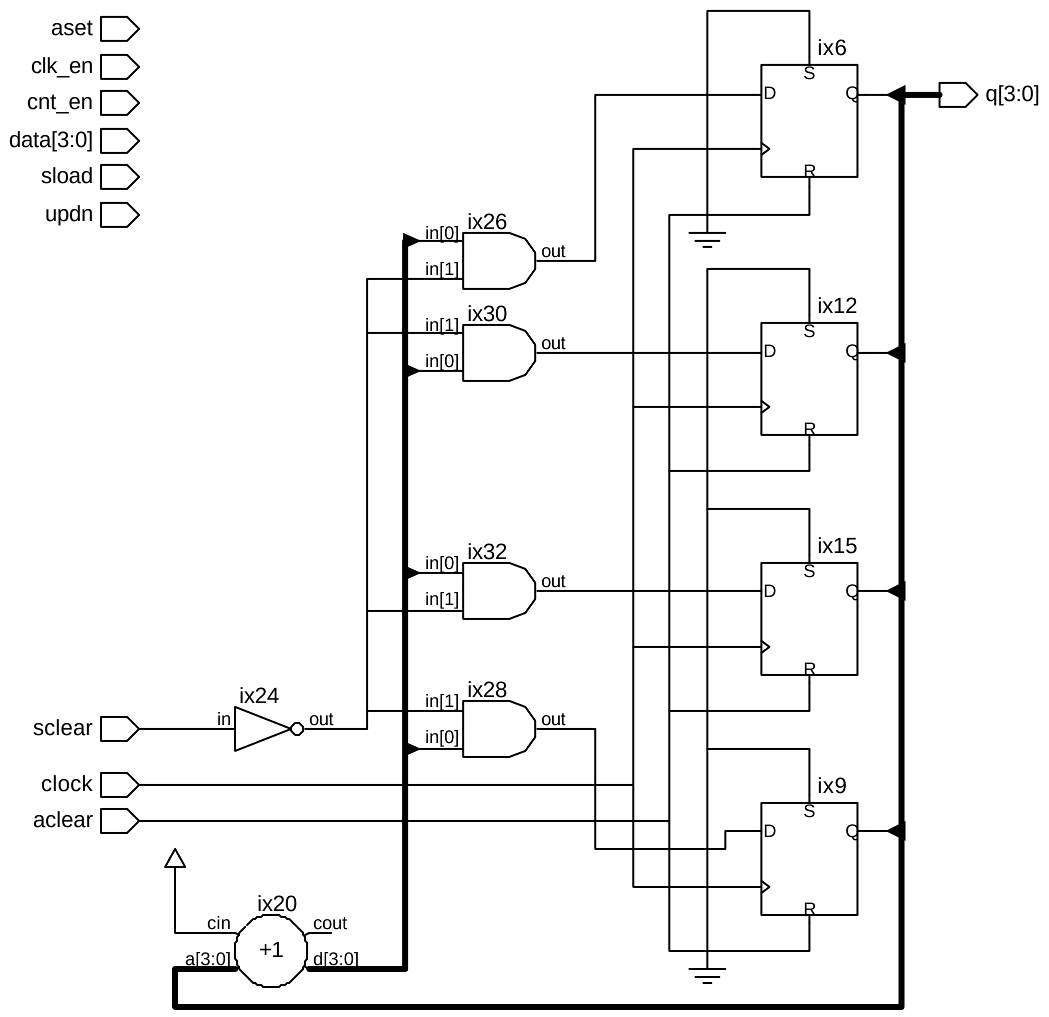

RESET_B
in
ix7
out
CLK
Q
aclear
aset
clk_en
clock
cnt_en
data[1:0]
sclear
sload
updn
q[1:0]
Q[1:0]
counter_up_aclear_clock_2

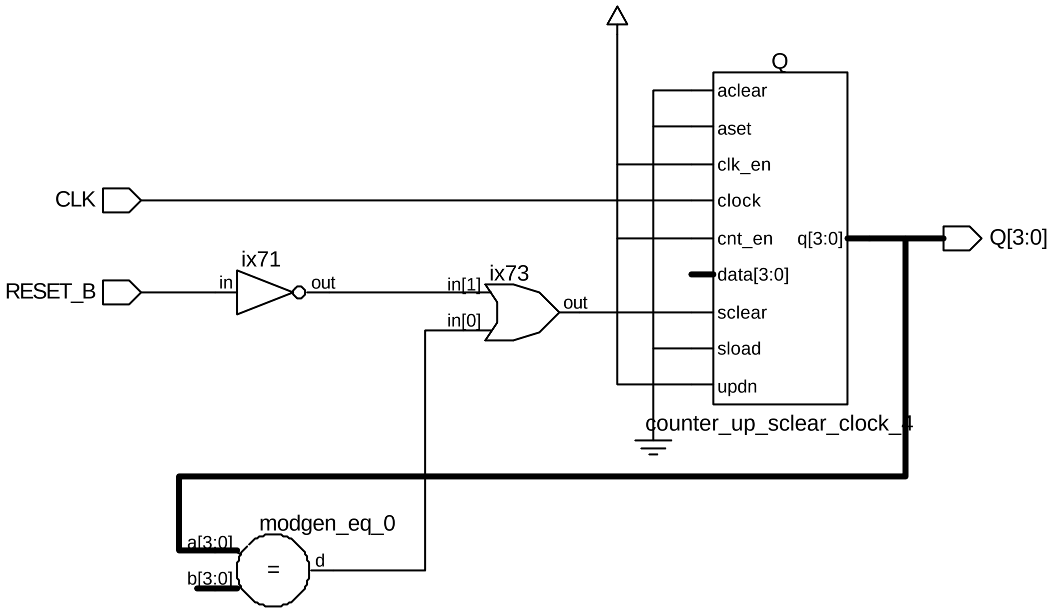

aset
clk_en
cnt_en
clock
aclear
data[1:0]
sclear
sload
updn
ix9
ix6
ix12
S
D
Q
R
cin
cout
+1
a[1:0]
d[1:0]
q[1:0]


ix37 size 4
in[3:0]
out[0:3]
ix21
ix23
ix29
ix9
ix39
ix25
ix31
ix11
ix15
ix41
ix35
ix33
ix17
ix43
ix7
in[0]
in[1]
out
in
reg_Q(2)
reg_Q(3)
reg_Q(1)
reg_Q(0)
D
S
Q
R
Q[3:0]
CLK
RESET_B

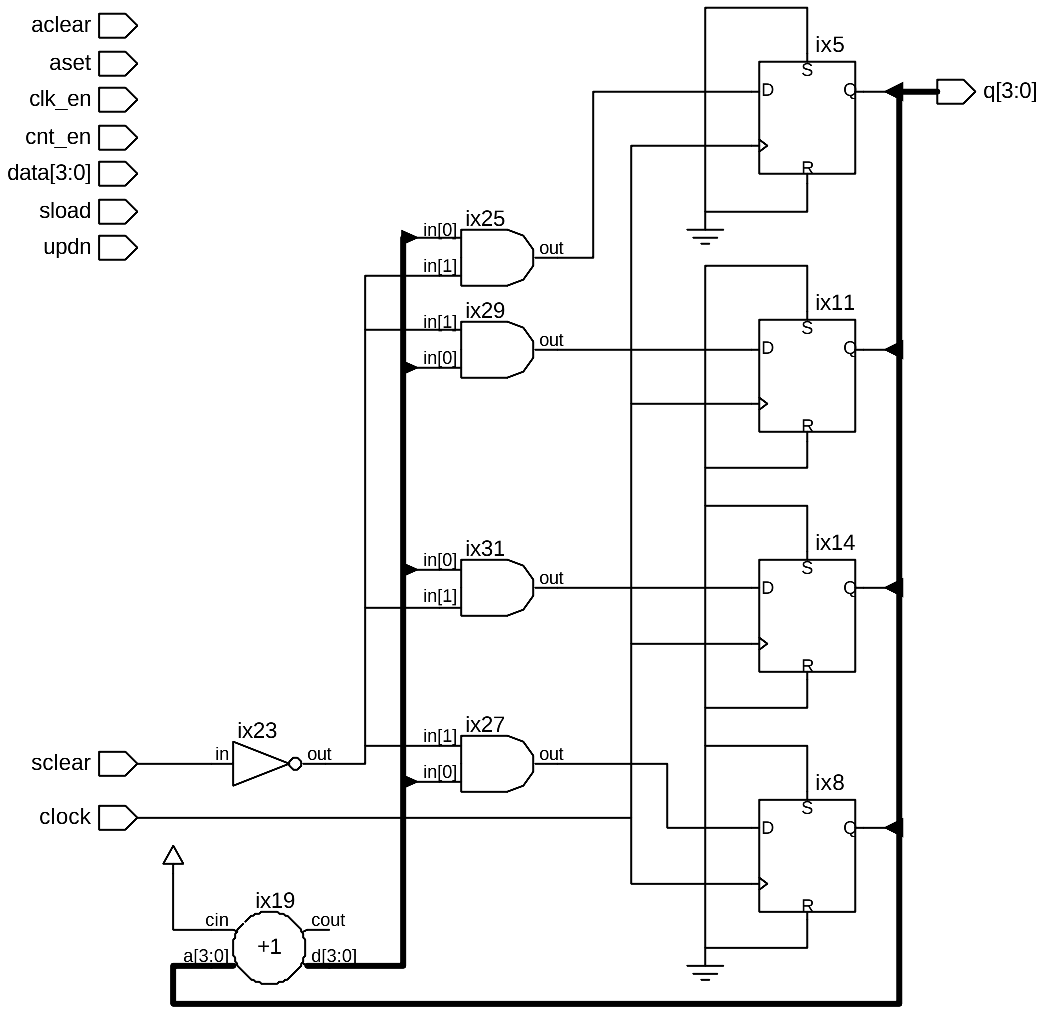


RESET_B
ix7
in
out
CLK
Q
aclear
aset
clk_en
clock
cnt_en
data[3:0]
sclear
sload
updn
q[3:0]
Q[3:0]
counter_up_sclear_aclear_clock_4
modgen_eq_0
a[3:0]
b[3:0]
=
d

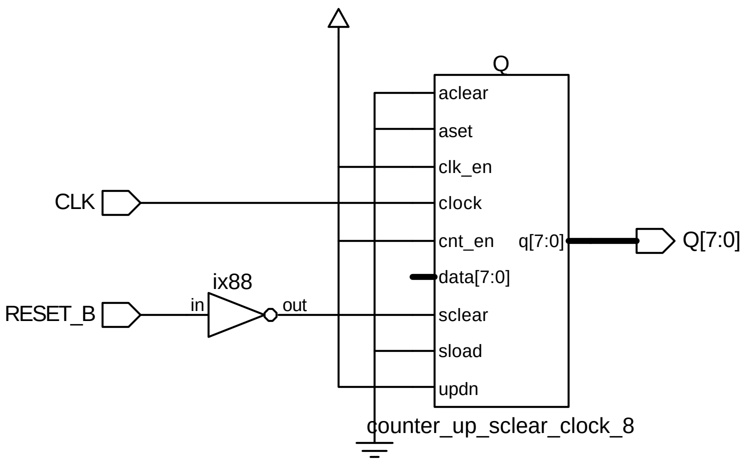

aset
clk_en
cnt_en
data[3:0]
sload
updn
sclear
clock
aclear
ix24
in
out
ix20
cin
cout
+1
a[3:0]
d[3:0]
ix26
in[0]
in[1]
out
ix30
in[1]
in[0]
out
ix32
in[0]
in[1]
out
ix28
in[1]
in[0]
out
ix6
ix12
ix15
ix9
S
D
Q
R
q[3:0]



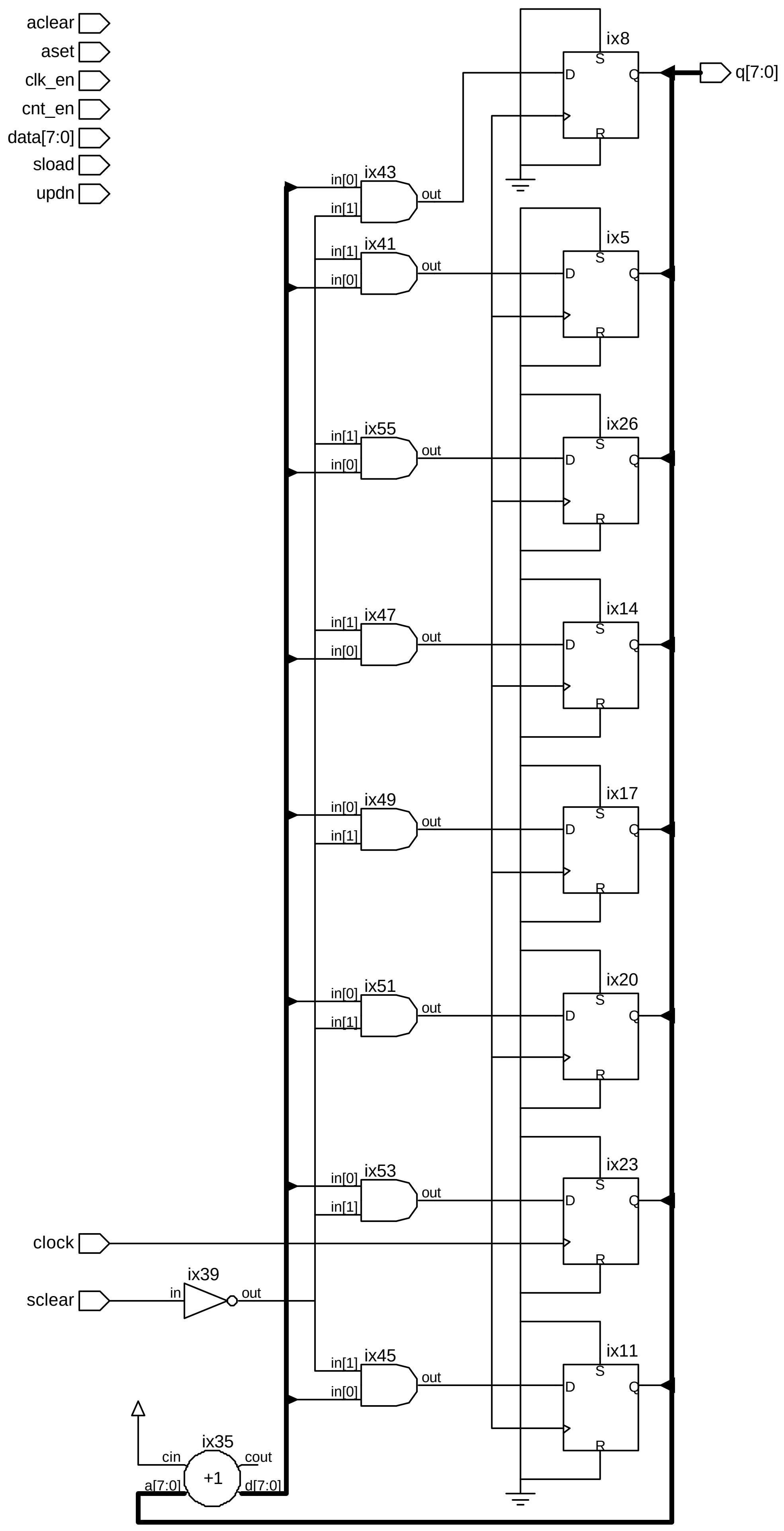
CLK
RESET_B
ix71
in
out
in[1]
in[0]
ix73
out
Q
aclear
aset
clk_en
clock
cnt_en
data[3:0]
sclear
sload
updn
q[3:0]
Q[3:0]
counter_up_sclear_clock_4
modgen_eq_0
a[3:0]
b[3:0]
=
d



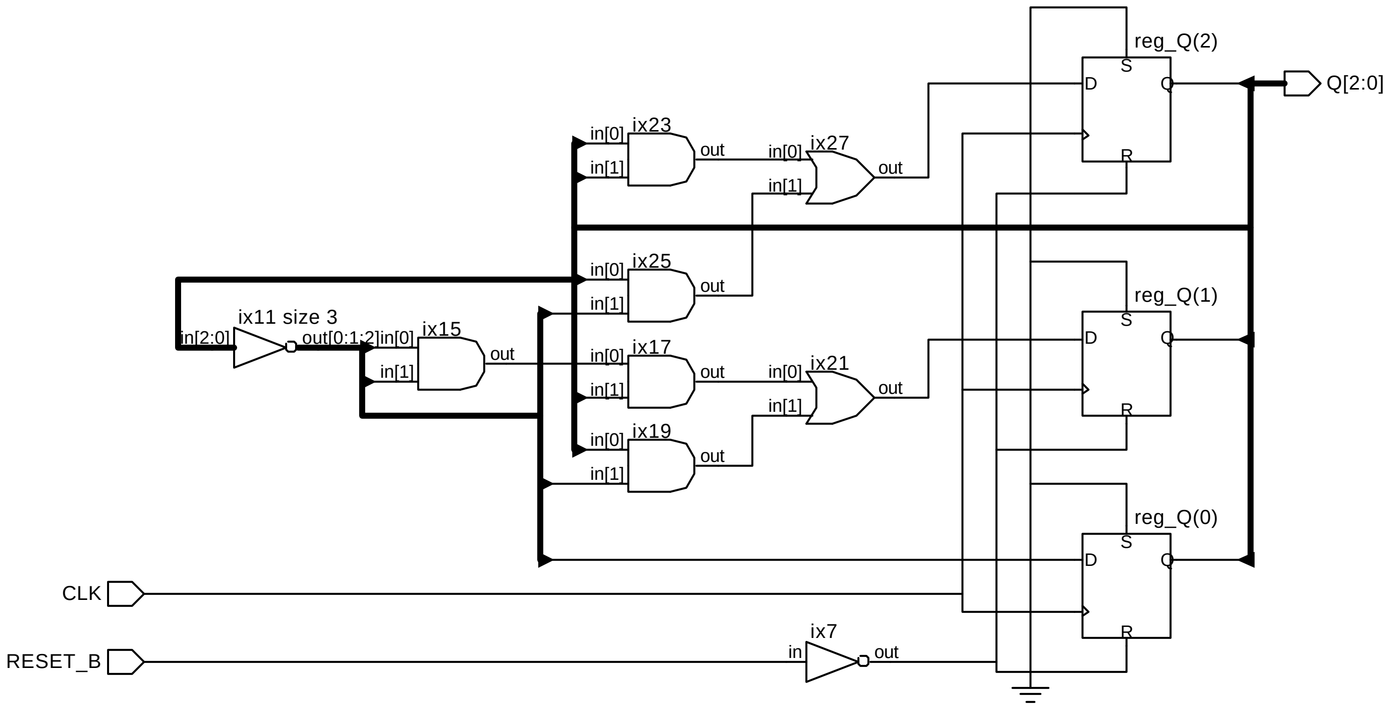
aclear
aset
clk_en
cnt_en
data[3:0]
sload
updn
sclear
clock
ix23
in
out
ix19
cin
cout
+1
a[3:0]
d[3:0]
ix25
in[0]
in[1]
out
ix29
in[1]
in[0]
out
ix31
in[0]
in[1]
out
ix27
in[1]
in[0]
out
ix5
ix11
ix14
ix8
S
D
Q
R
q[3:0]

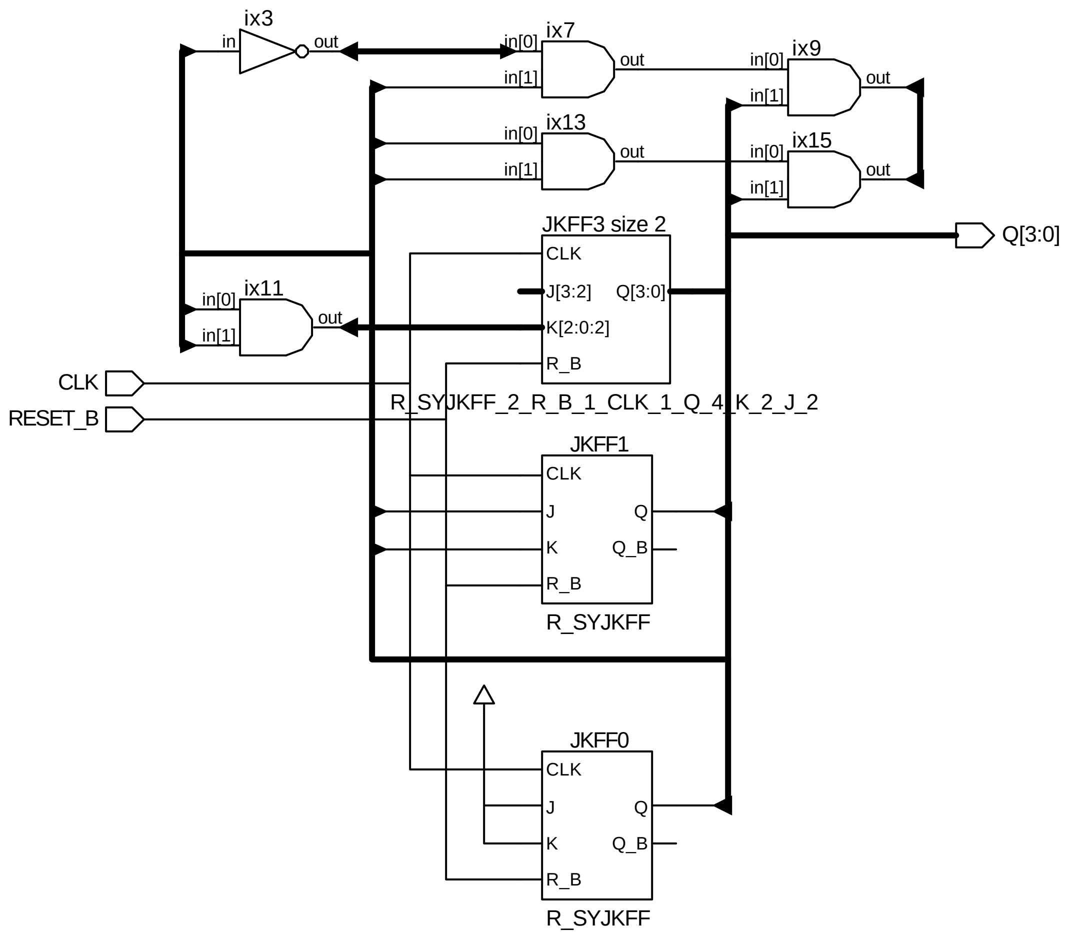


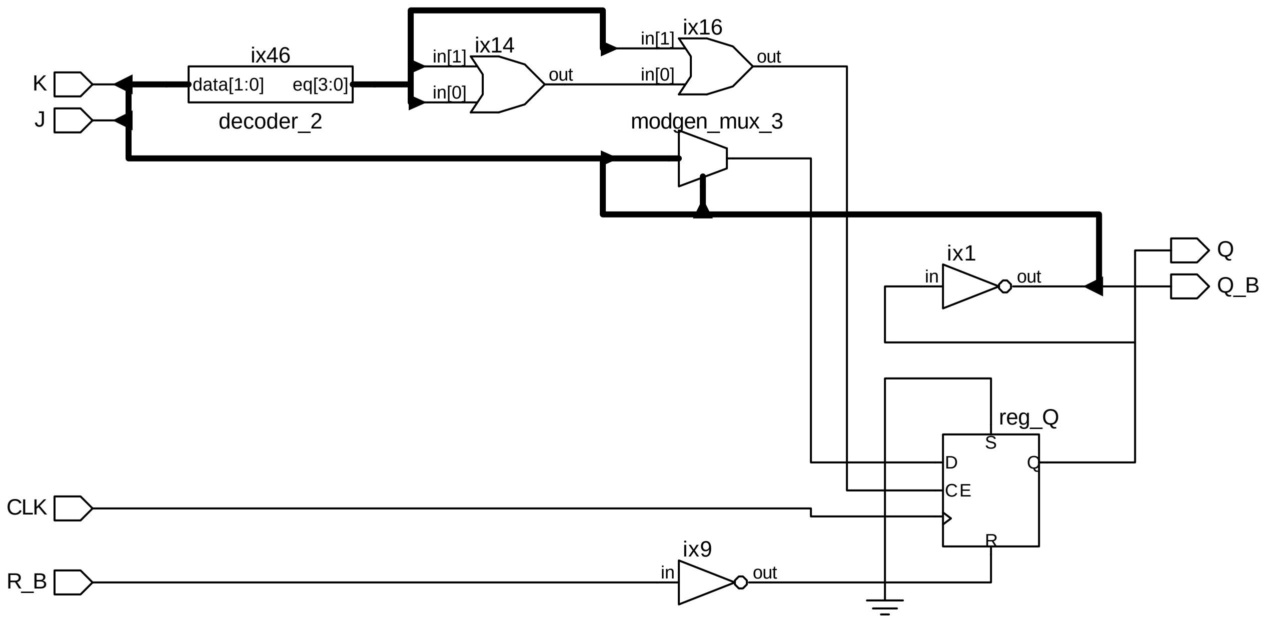
Q
aclear
aset
clk_en
clock
cnt_en
q[7:0]
data[7:0]
sclear
sload
updn
CLK
RESET_B
ix88
in
out
Q[7:0]
counter_up_sclear_clock_8

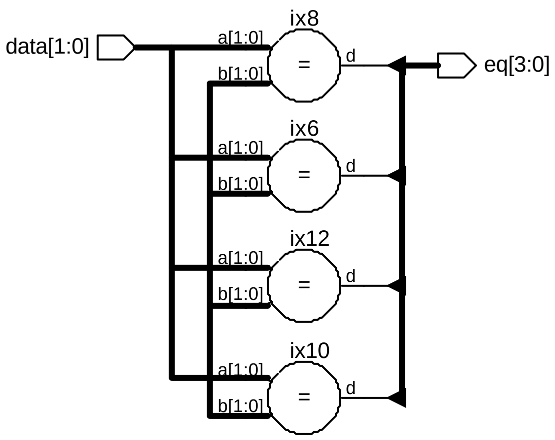


aclear
aset
clk_en
cnt_en
data[7:0]
sload
updn
clock
sclear
q[7:0]
ix8
ix5
ix26
ix14
ix17
ix20
ix23
ix11
D
S
Q
R
ix43
ix41
ix55
ix47
ix49
ix51
ix53
ix45
in[0]
in[1]
out
ix39
in
out
ix35
+1
cin
cout
a[7:0]
d[7:0]


reg_Q(2)
Q[2:0]
ix23
ix27
ix25
ix11 size 3
ix15
ix17
ix21
ix19
reg_Q(1)
reg_Q(0)
CLK
RESET_B
ix7
in[2:0]
out[0:1:2]
in[0]
in[1]
out
D
S
R
Q
in


ix3
in
out
ix7
in[0]
in[1]
out
ix9
in[0]
in[1]
out
ix13
in[0]
in[1]
out
ix15
in[0]
in[1]
out
Q[3:0]
JKFF3 size 2
CLK
J[3:2]
Q[3:0]
K[2:0:2]
R_B
ix11
in[0]
in[1]
out
CLK
RESET_B
R_SYJKFF_2_R_B_1_CLK_1_Q_4_K_2_J_2
JKFF1
CLK
J
Q
K
Q_B
R_B
R_SYJKFF
JKFF0
CLK
J
Q
K
Q_B
R_B
R_SYJKFF


ix46
K
J
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