```
/*      EN_REG4 */
module  EN_REG4 ( CLR_B, D, CLK, EN, Q );
input   CLR_B, CLK, EN;
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 if ( EN )
                        Q <= D;
endmodule
```

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

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

```
/*      EN_CNT10        */
module  EN_CNT10        ( RESET_B, CLK, EN, Q );
input   RESET_B, CLK, EN;
output  [3:0] Q;
reg     [3:0] Q;
        always  @( posedge CLK or negedge RESET_B  )
                if      ( !RESET_B )
                        Q <= 0;
                else if ( EN )
                        if ( Q == 9 )
                                Q <= 0;
                        else
                                Q <= Q + 1;
endmodule
```

```
/*      EN_UDCNT10      */
module  EN_UDCNT10      ( RESET_B, CLK, EN, UP, Q );
input   RESET_B, CLK, EN, UP;
output  [3:0] Q;
reg     [3:0] Q;
        always  @( posedge CLK or negedge RESET_B  )
                if      ( !RESET_B )
                        Q <= 0;
                else if ( EN )
                        if ( UP )                       // COUNT UP
                                if ( Q == 9 )
                                        Q <= 0;
                                else
                                        Q <= Q + 1;
                        else                            // COUNT DOWN
                                if ( Q == 0 )
                                        Q <= 9;
                                else
                                        Q <= Q - 1;
endmodule
```

```
/*      LD_EN_UDCNT10   */
module  LD_EN_UDCNT10   (  RESET_B, CLK, LOAD, EN, UP, IN, Q );
input   RESET_B, CLK, LOAD, EN, UP;
input   [3:0] IN;
output  [3:0] Q;
reg     [3:0] Q;
        always  @( posedge CLK or negedge RESET_B  )
                if      ( !RESET_B )
                        Q <= 0;
                else if ( LOAD )
                        Q <= IN;
                else if ( EN )
                        if ( UP )                       // COUNT UP
                                if ( Q == 9 )
                                        Q <= 0;
                                else
                                        Q <= Q + 1;
                        else                            // COUNT DOWN
                                if ( Q == 0 )
                                        Q <= 9;
                                else
                                        Q <= Q - 1;
endmodule
```

```
/*      CNT100  */
module  CNT100  (  RESET_B, CLK, LOAD, EN, UP, IN, Q, CNT_99 );
input   RESET_B, CLK, LOAD, EN, UP;
input   [7:0] IN;
output  CNT_99;
output  [7:0] Q;
wire    LOW_CNT_9, HIGH_CNT_9;
        LD_EN_UDCNT10   LOW_CNT ( RESET_B, CLK,LOAD, EN,
                                  UP, IN[3:0], Q[3:0], LOW_CNT_9 ),
                        HIGH_CNT ( RESET_B, CLK, LOAD, EN & LOW_CNT_9 ,
                                  UP, IN[7:4], Q[7:4], HIGH_CNT_9 );
        assign  CNT_99 = LOW_CNT_9 & HIGH_CNT_9;
endmodule

/*      LD_EN_UDCNT10   */
module  LD_EN_UDCNT10   ( RESET_B, CLK, LOAD, EN, UP, IN, Q, RC );
input   RESET_B, CLK, LOAD, EN, UP;
input   [3:0] IN;
output  RC;
output  [3:0] Q;
reg     [3:0] Q;
        assign  RC = UP & ( Q == 9 ) | ~UP & ( Q == 0 );
        always  @( posedge CLK or negedge RESET_B  )
                if      ( !RESET_B )
                        Q <= 0;
                else if ( LOAD )
                        Q <= IN;
                else if ( EN )
                        if ( UP )                       // COUNT UP
                                if ( Q == 9 )
                                        Q <= 0;
                                else
                                        Q <= Q + 1;
                        else                            // COUNT DOWN
                                if ( Q == 0 )
                                        Q <= 9;
                                else
                                        Q <= Q - 1;
endmodule
```

```
/*      CNTFF   */
module  CNTFF   (  RESET_B, CLK, LOAD, EN, UP, IN, Q, CNT_FF );
input   RESET_B, CLK, LOAD, EN, UP;
input   [7:0] IN;
output  CNT_FF;
output  [7:0] Q;
wire    LOW_CNT_F, HIGH_CNT_F;
        LD_EN_UDCNTF    LOW_CNT ( RESET_B, CLK,LOAD, EN,
                                  UP, IN[3:0], Q[3:0], LOW_CNT_F ),
                        HIGH_CNT ( RESET_B, CLK, LOAD, EN & LOW_CNT_F,
                                  UP, IN[7:4], Q[7:4], HIGH_CNT_F );
        assign  CNT_FF = LOW_CNT_F & HIGH_CNT_F;
endmodule

/*      LD_EN_UDCNTF    */
module  LD_EN_UDCNTF    ( RESET_B, CLK, LOAD, EN, UP, IN, Q, RC );
input   RESET_B, CLK, LOAD, EN, UP;
input   [3:0] IN;
output  RC;
output  [3:0] Q;
reg     [3:0] Q;
        assign  RC = UP & ( Q == 15 ) | ~UP & ( Q == 0 );
        always  @( posedge CLK or negedge RESET_B  )
                if      ( !RESET_B )
                        Q <= 0;
                else if ( LOAD )
                        Q <= IN;
                else if ( EN )
                        if ( UP )                       // COUNT UP
                                Q <= Q + 1;
                        else                            // COUNT DOWN
                                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 if ( Q == 9 )
                        Q <= 0;
                else
                        Q <= Q + 1;
endmodule
```

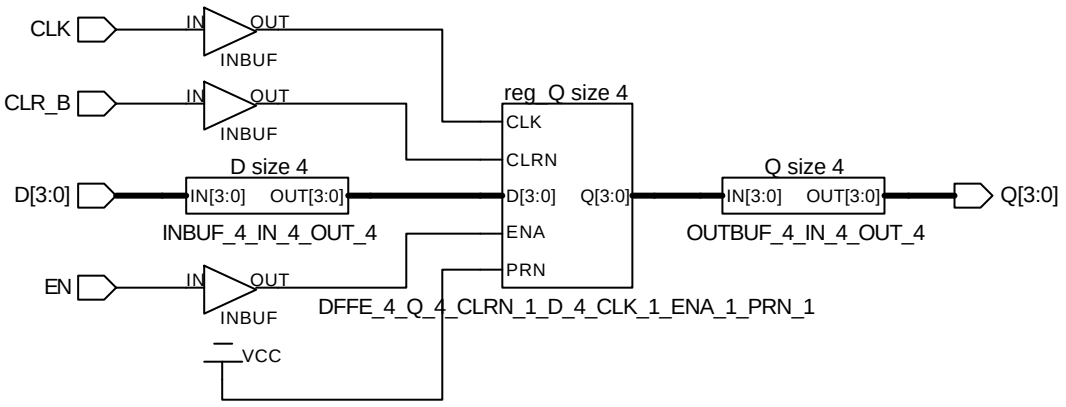
```
/*      TRAP_CNT10      */
module  TRAP_CNT10      ( RESET_B, CLK, TEST, Q );
input   RESET_B, CLK, TEST;
output  [3:0] Q;
reg     [3:0] Q;
wire    [3:0] OUT;
        assign  OUT = { TEST, 3'b000} | INC_OUT ( Q );
        always  @( posedge CLK or negedge RESET_B )
                if ( !RESET_B )
                        Q <= 0;
                else
                        Q <= OUT;

function        [3:0] INC_OUT;
input   [3:0] Q;
        if ( Q == 9 )
                INC_OUT = 0;
        else
                INC_OUT = Q + 1;
endfunction

endmodule
```

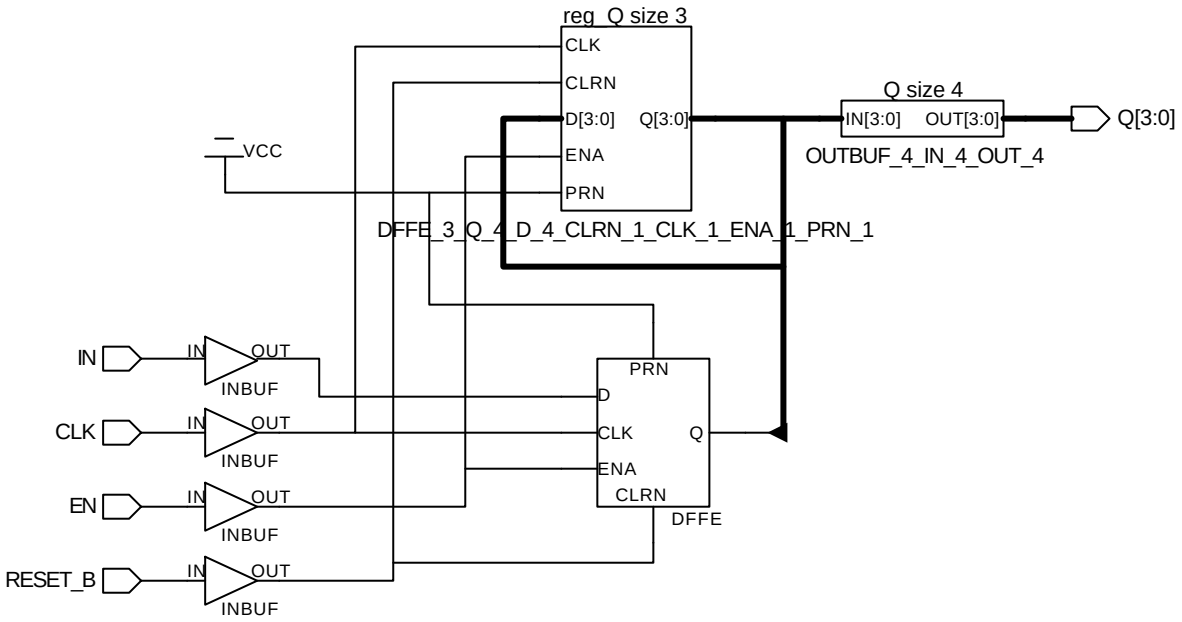
```
/*      SW_DEF  */
module  SW_DEF  ( RESET_B, CLK, SW_INPUT, Q );
input   RESET_B, CLK, SW_INPUT;
output  [3:0] Q;
reg     QA, QB;
wire    ALFA, BETA;
        always  @( posedge CLK or negedge RESET_B  )
                if      ( !RESET_B )
                        begin
                                QA <= 0;
                                QB <= 0;
                        end
                else
                        begin
                                QA <= SW_INPUT;
                                QB <= QA;
                        end
        assign  ALFA  = ~QB &  QA;
        assign  BETA =  QB & ~QA;
        EN_CNT10        EN_CNT10    ( RESET_B, CLK, ALFA | BETA, Q );
endmodule

/*      EN_CNT10        */
module  EN_CNT10        ( RESET_B, CLK, EN, Q );
input   RESET_B, CLK, EN;
output  [3:0] Q;
reg     [3:0] Q;
        always  @( posedge CLK or negedge RESET_B  )
                if      ( !RESET_B )
                        Q <= 0;
                else if ( EN )
                        if ( Q == 9 )
                                Q <= 0;
                        else
                                Q <= Q + 1;
endmodule
```

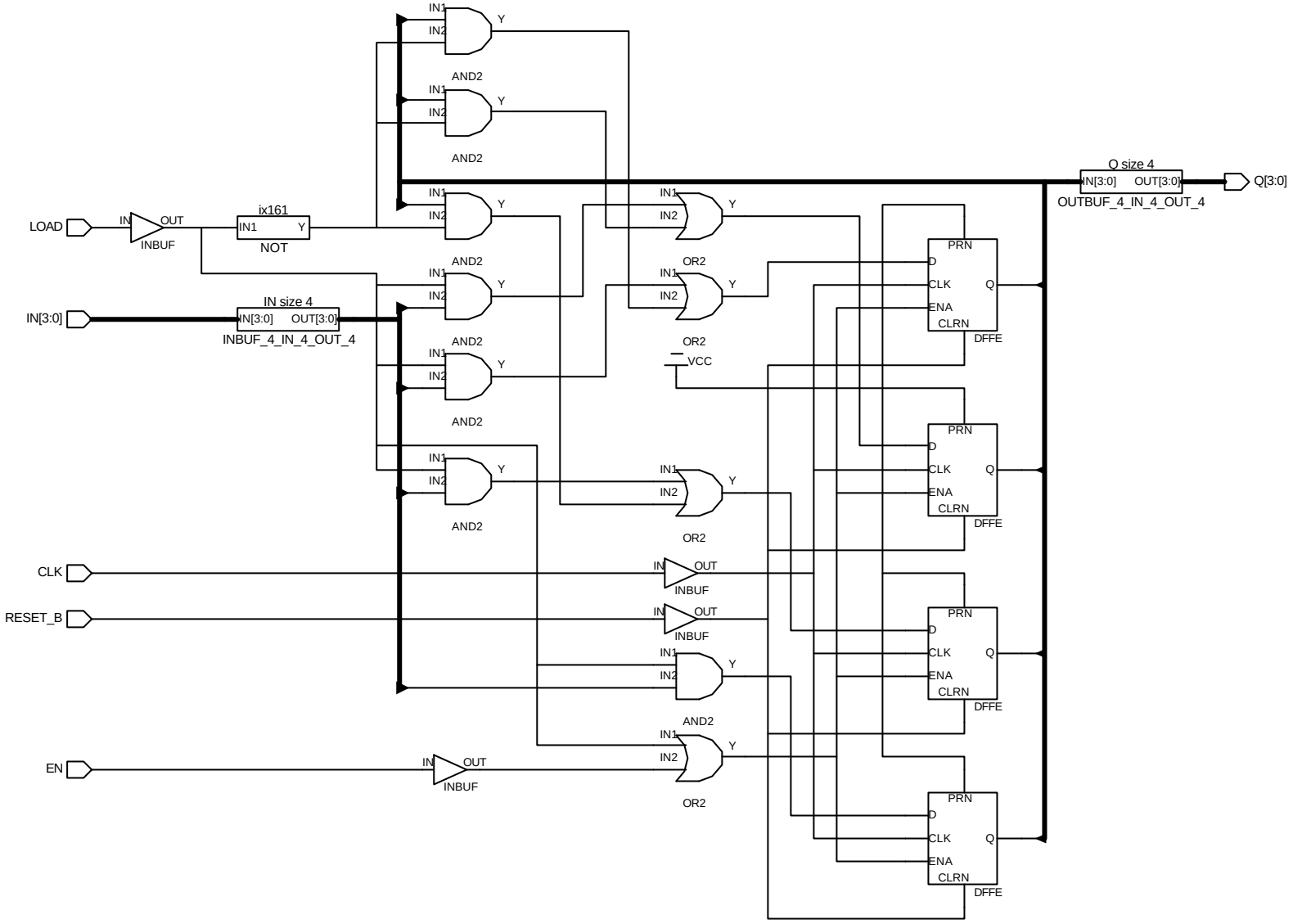
```
/*      DEC_3TO8        */
module  DEC_3TO8        ( IN, OUT );
input   [2:0] IN;
output  [7:0] OUT;
reg     [7:0] OUT;
        always  @( IN )
                case( IN )
                        0: OUT = 8'b0000_0001;
                        1: OUT = 8'b0000_0010;
                        2: OUT = 8'b0000_0100;
                        3: OUT = 8'b0000_1000;
                        4: OUT = 8'b0001_0000;
                        5: OUT = 8'b0010_0000;
                        6: OUT = 8'b0100_0000;
                        7: OUT = 8'b1000_0000;
                endcase
endmodule
```

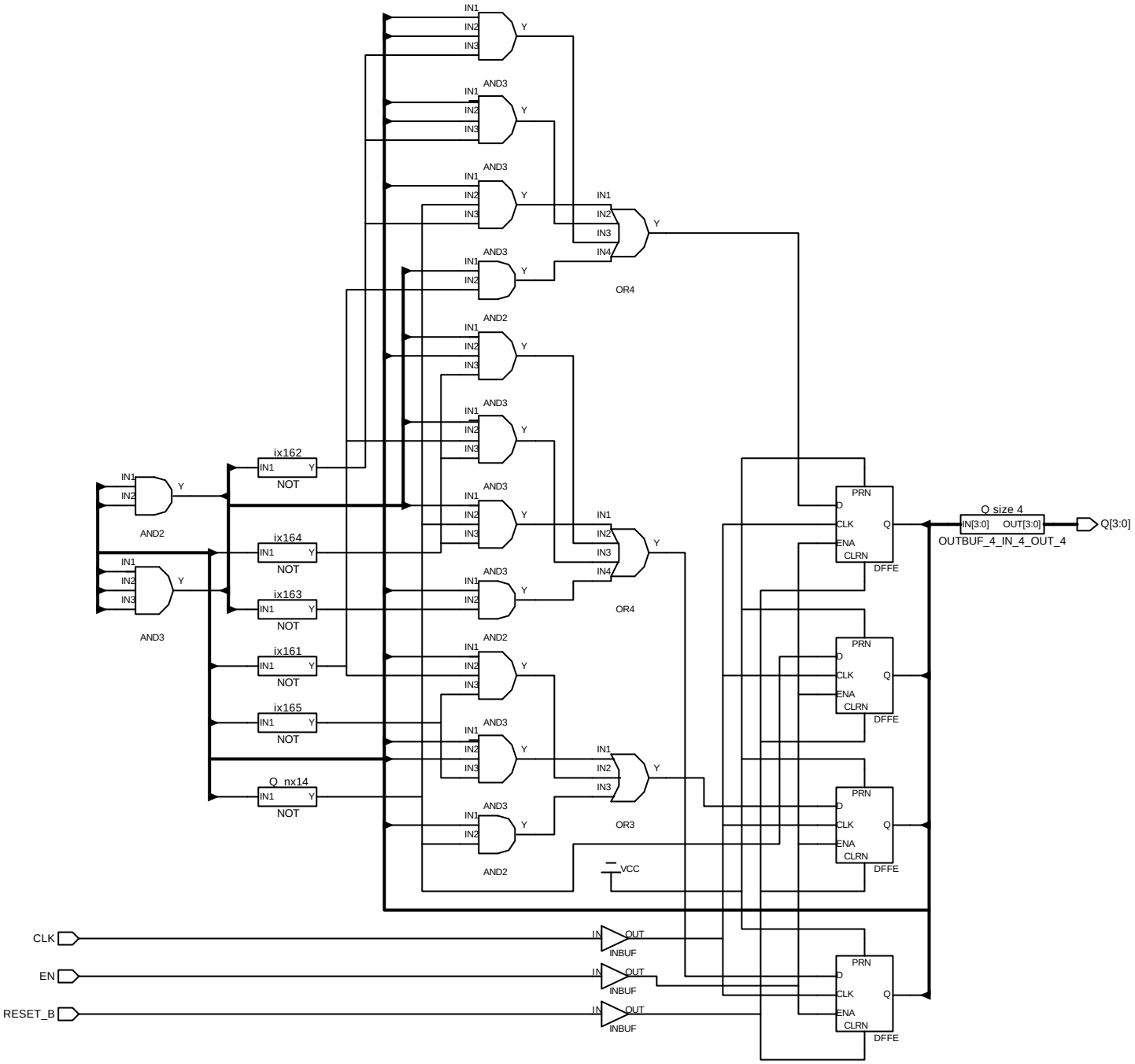
```
/*      CNT10   */
module  CNT10   ( RESET_B, CLK, TEST, Q );
input   RESET_B, CLK;
input   [3:0] TEST;
output  [3:0] Q;
reg     [3:0] Q;
wire    [3:0] OUT;
        assign  OUT = TEST ^ INC_OUT ( Q );
        always  @( posedge CLK or negedge RESET_B )
                if ( !RESET_B )
                        Q <= 0;
                else
                        Q <= OUT;

function        [3:0] INC_OUT;
input   [3:0] Q;
        if ( Q >= 9 )
                INC_OUT = 0;
        else
                INC_OUT = Q + 1;
endfunction

endmodule
```

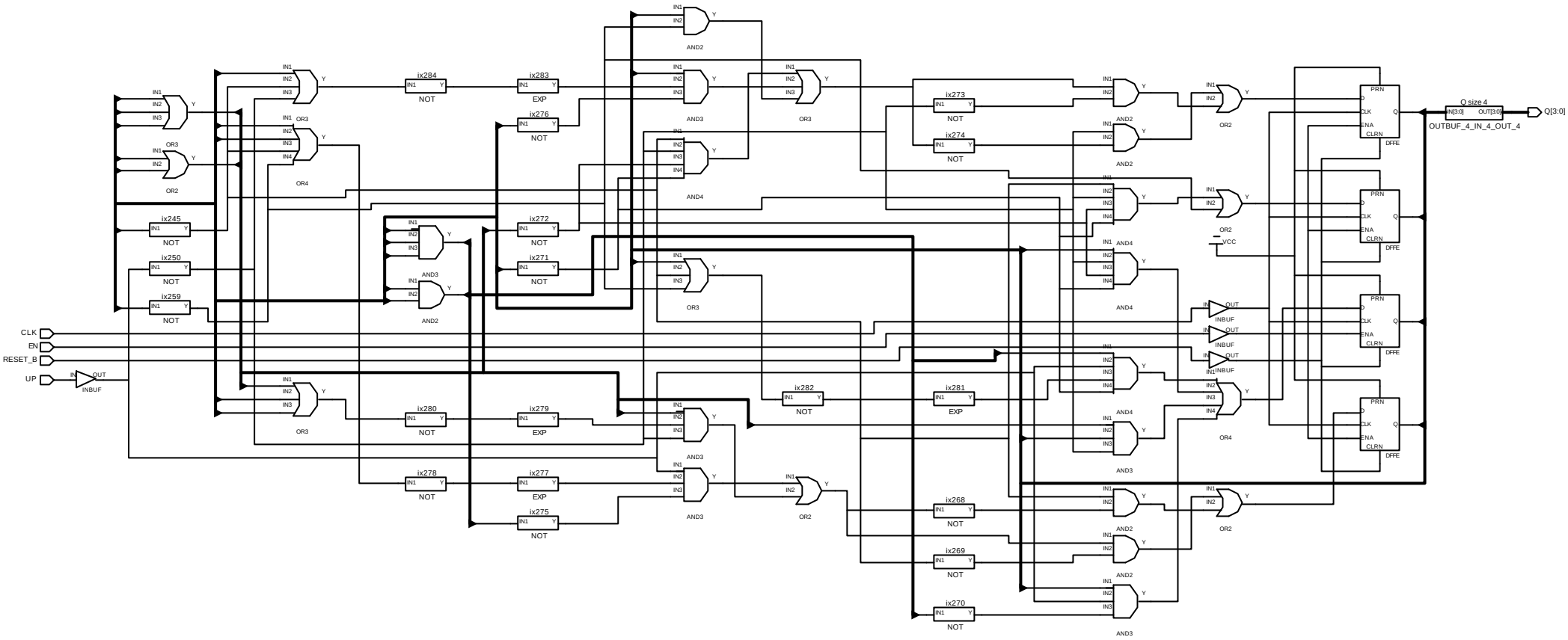
```
/*      LD_EN_UDCNT16   */
module  LD_EN_UDCNT16   (  RESET_B, CLK, LOAD, EN, UP, IN, Q );
input   RESET_B, CLK, LOAD, EN, UP;
input   [3:0] IN;
output  [3:0] Q;
reg     [3:0] Q;
        always  @( posedge CLK or negedge RESET_B  )
                if      ( !RESET_B )
                        Q <= 0;
                else if ( LOAD )
                        Q <= IN;
                else if ( EN )
                        if ( UP )                       // COUNT UP
                                Q <= Q + 1;
                        else                            // COUNT DOWN
                                Q <= Q - 1;
endmodule
```

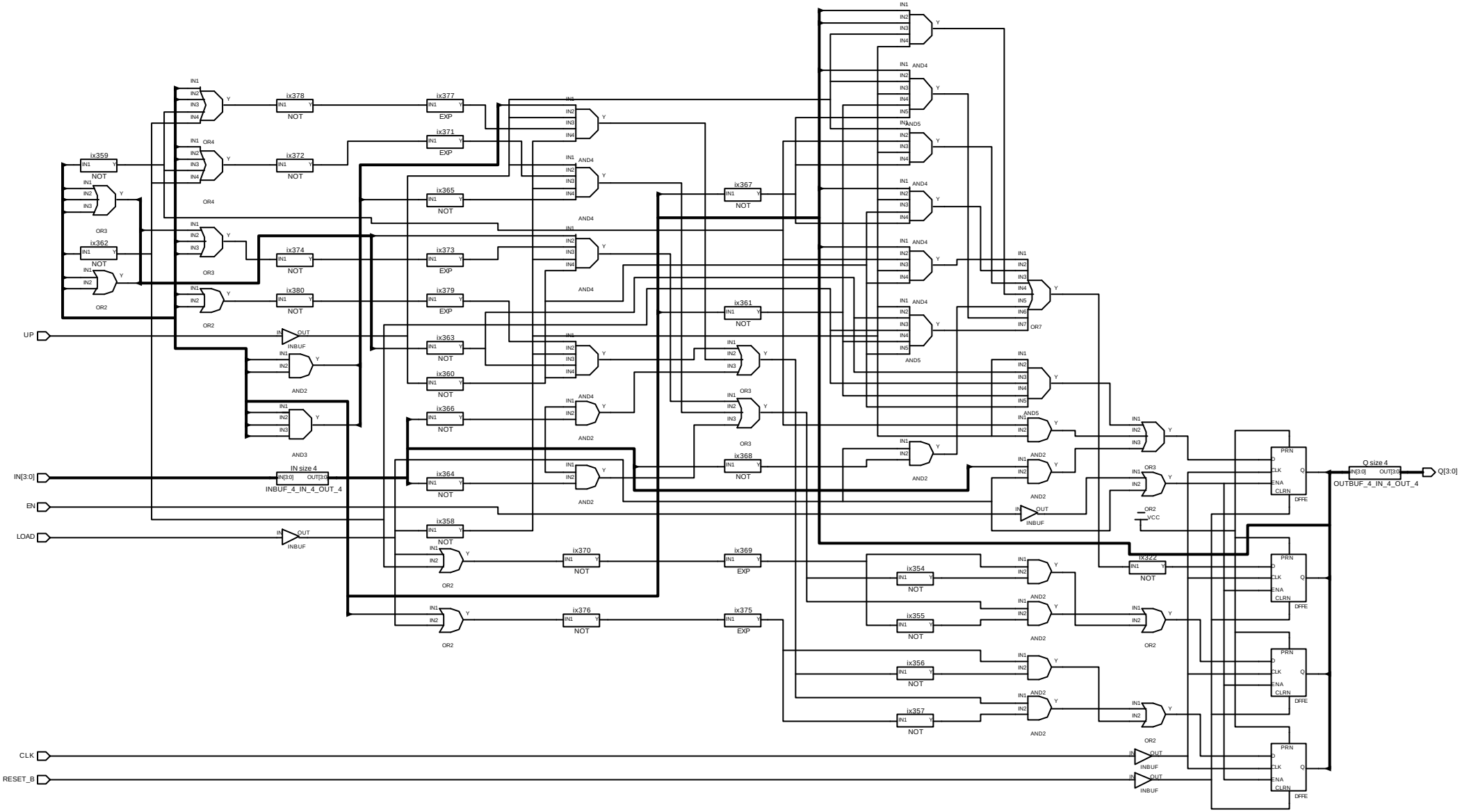

CLK
IN
OUT
INBUF
CLR_B
IN
OUT
INBUF
reg_Q size 4
CLK
CLRN
D size 4
D[3:0]
IN[3:0]
OUT[3:0]
INBUF_4_IN_4_OUT_4
D[3:0]
Q[3:0]
ENA
PRN
Q size 4
IN[3:0]
OUT[3:0]
OUTBUF_4_IN_4_OUT_4
Q[3:0]
EN
IN
OUT
INBUF
VCC
DFFE_4_Q_4_CLRN_1_D_4_CLK_1_ENA_1_PRN_1

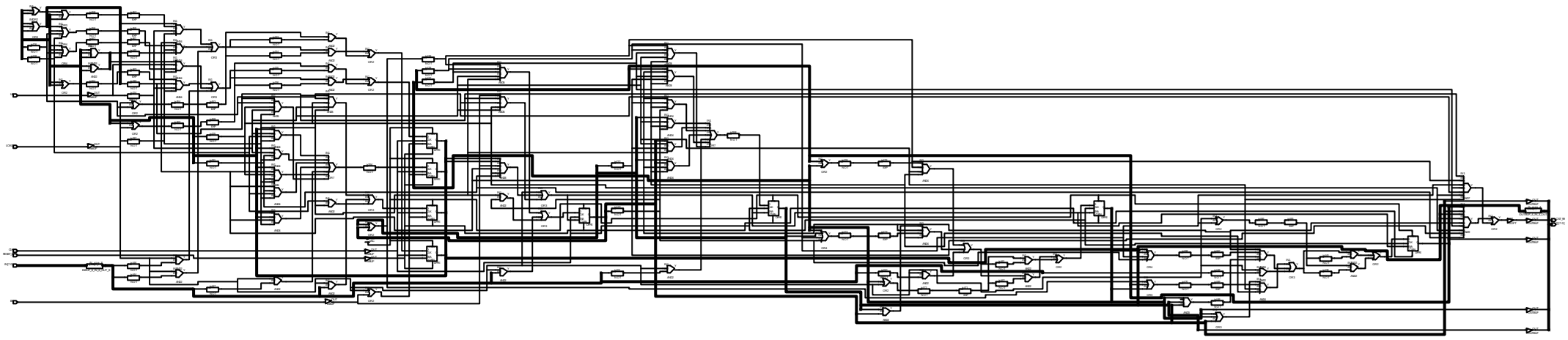

reg_Q size 3
CLK
CLRN
D[3:0]
Q[3:0]
ENA
PRN
DFFE_3_Q_4_D_4_CLRN_1_CLK_1_ENA_1_PRN_1
Q size 4
IN[3:0]
OUT[3:0]
OUTBUF_4_IN_4_OUT_4
Q[3:0]
VCC
IN
CLK
EN
RESET_B
IN
OUT
INBUF
PRN
D
CLK
Q
ENA
CLRN
DFFE

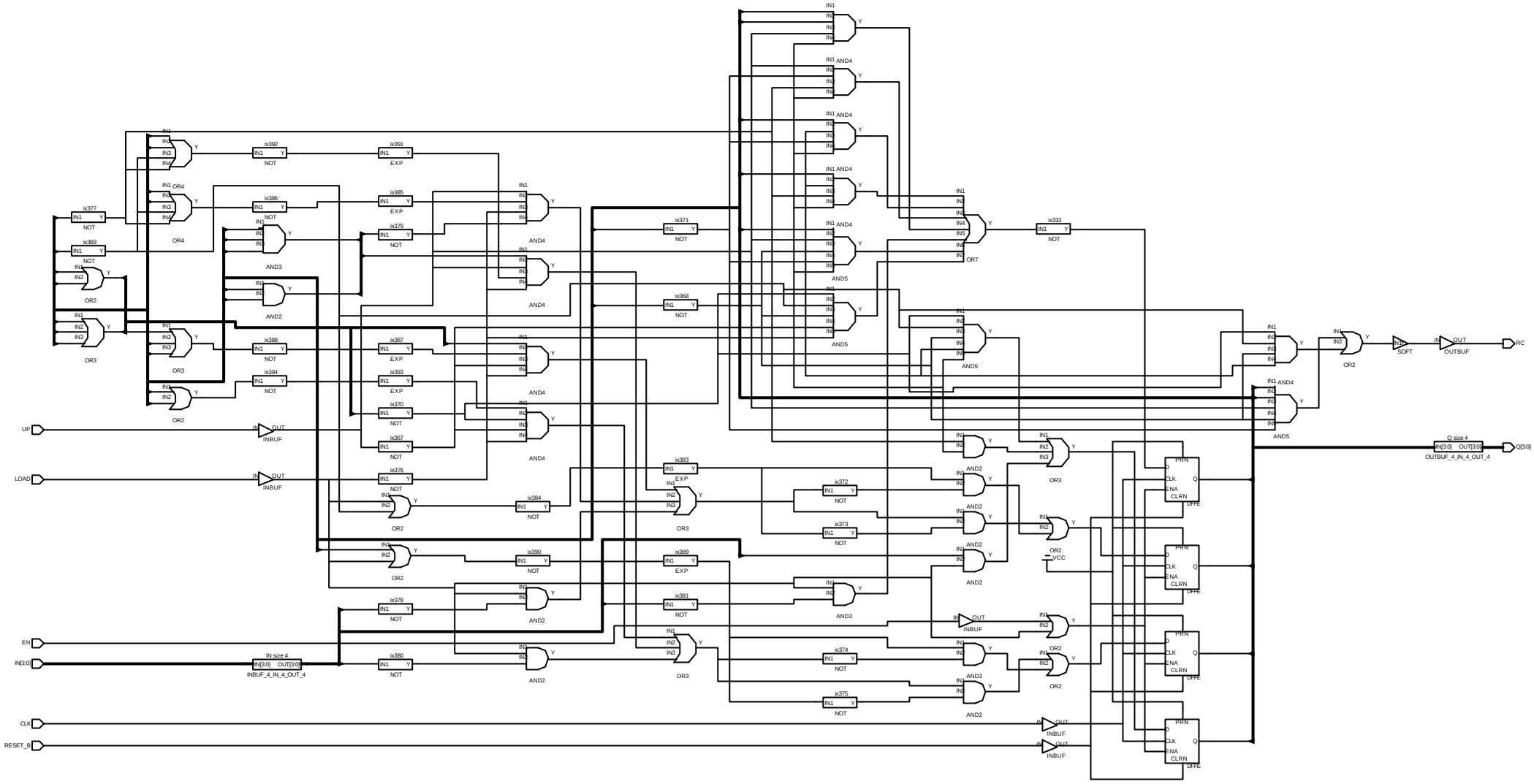

LOAD
IN
OUT
INBUF
ix161
IN1
Y
NOT
IN[3:0]
IN size 4
IN[3:0]
OUT[3:0]
INBUF_4_IN_4_OUT_4
CLK
RESET_B
EN
IN1
IN2
Y
AND2
OR2
VCC
PRN
D
CLK
Q
ENA
CLRN
DFFE
Q size 4
OUTBUF_4_IN_4_OUT_4
Q[3:0]

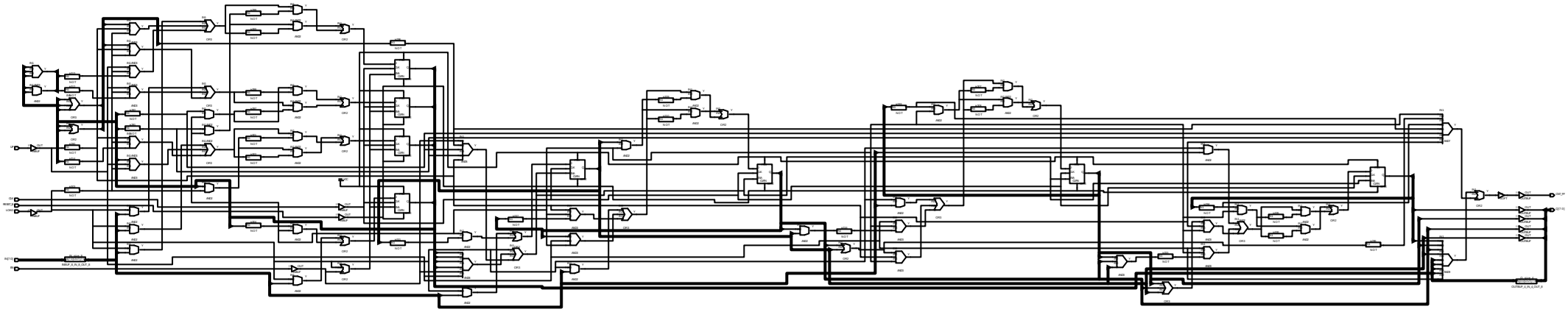

IN1
IN2
IN3
Y
AND3
AND2
OR4
OR3
NOT
ix162
ix164
ix163
ix161
ix165
Q_nx14
IN4
PRN
D
CLK
Q
ENA
CLRN
DFFE
Q size 4
IN[3:0]
OUT[3:0]
OUTBUF_4_IN_4_OUT_4
Q[3:0]
VCC
CLK
EN
RESET_B
INBUF
OUT

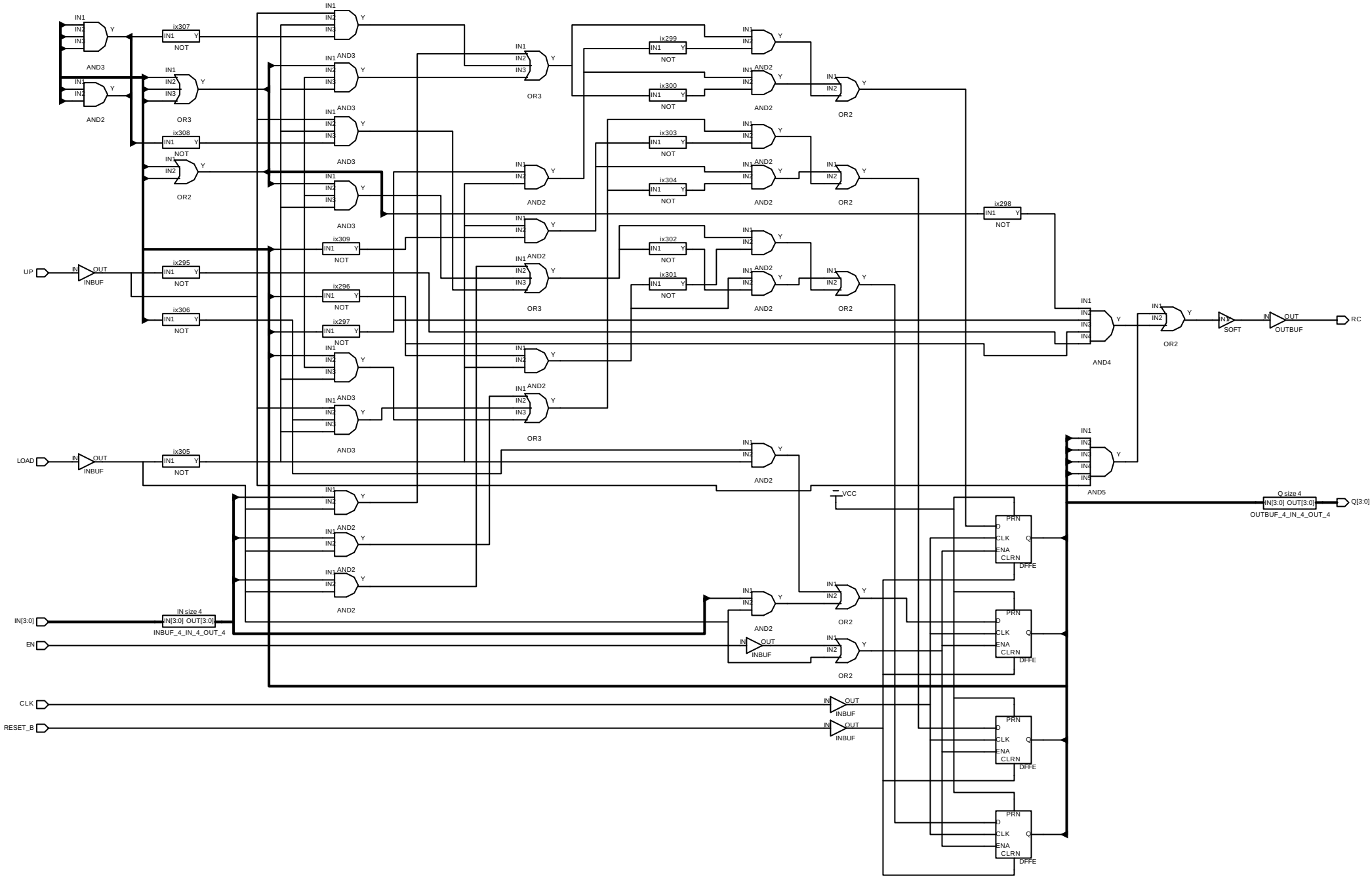

CLK
EN
RESET_B
UP
INBUF
OUT
IN
OR2
OR3
OR4
AND2
AND3
AND4
NOT
EXP
ix284
ix283
ix276
ix245
ix250
ix259
ix272
ix271
ix280
ix279
ix278
ix277
ix275
ix282
ix281
ix273
ix274
ix268
ix269
ix270
VCC
DFFE
PRN
CLK
ENA
CLRN
Q size 4
OUTBUF_4_IN_4_OUT_4
Q[3:0]

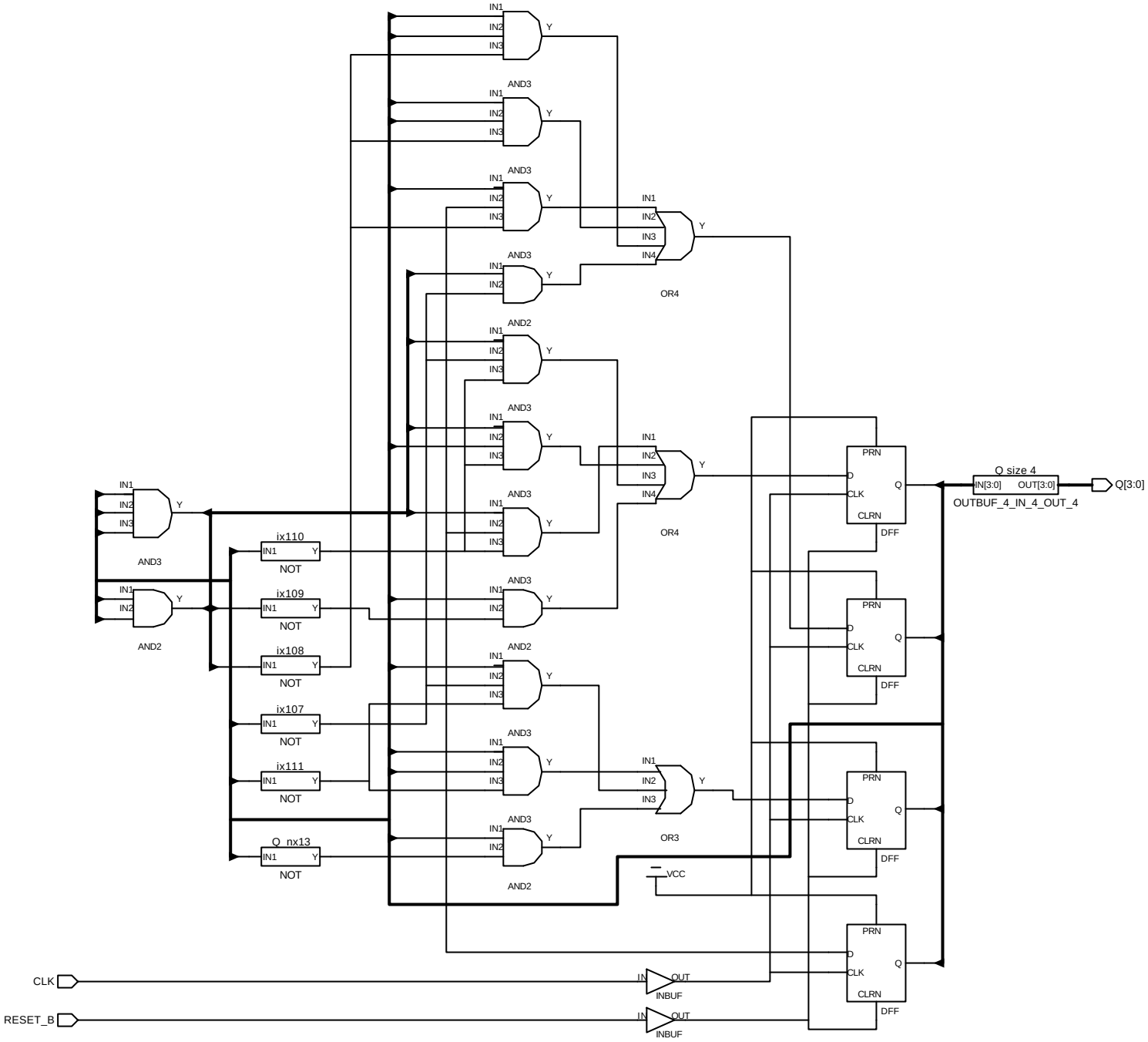

UP
IN[3:0]
EN
LOAD
CLK
RESET_B
IN size 4
INBUF_4_IN_4_OUT_4
Q size 4
OUTBUF_4_IN_4_OUT_4
Q[3:0]
INBUF
OUTBUF
VCC
NOT
EXP
AND2
AND3
AND4
AND5
OR2
OR3
OR4
OR7
DFRE
ix358
ix359
ix360
ix361
ix362
ix363
ix364
ix365
ix366
ix367
ix368
ix369
ix370
ix371
ix372
ix373
ix374
ix375
ix376
ix377
ix378
ix379
ix380
ix322
ix354
ix355
ix356
ix357

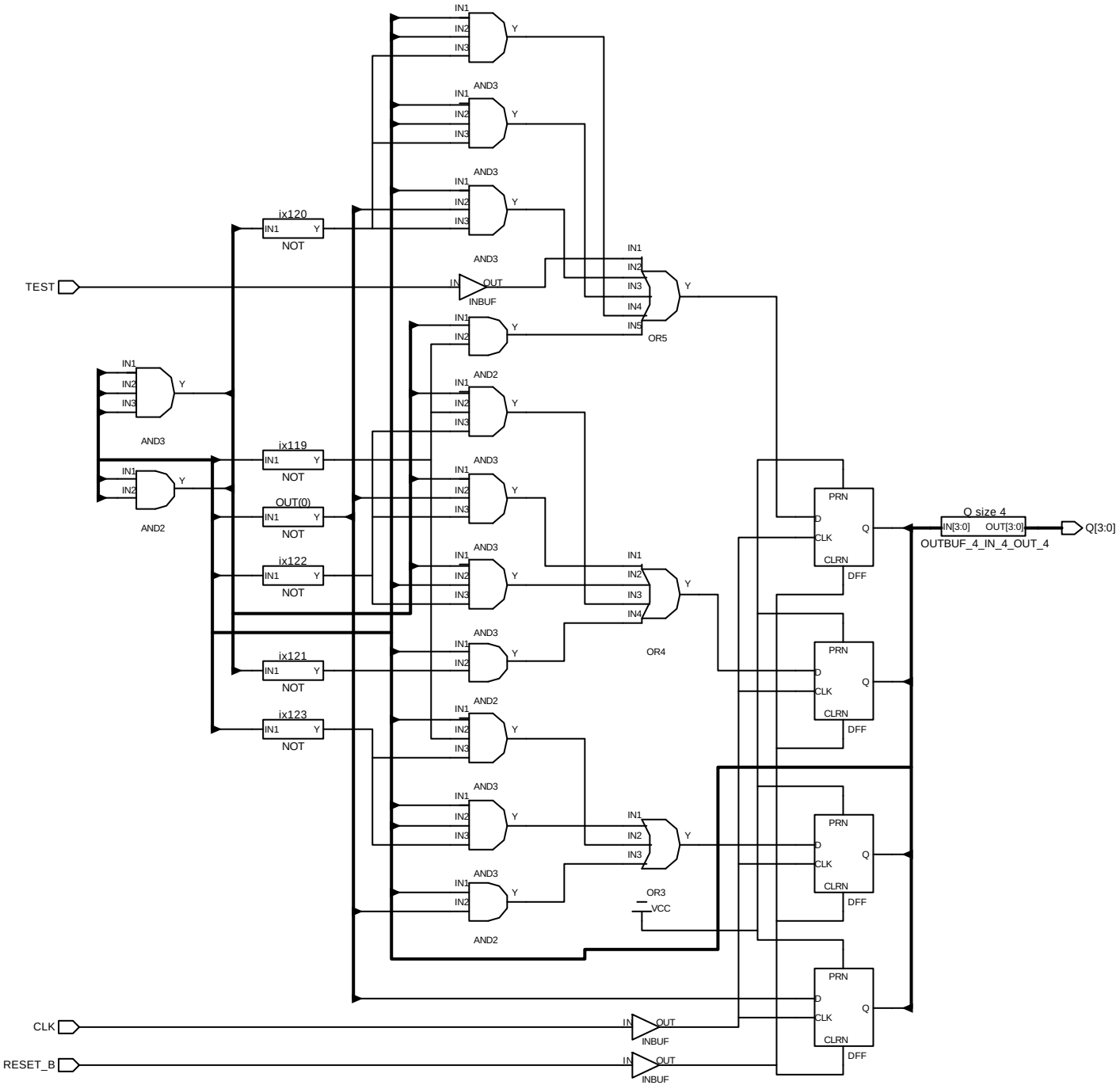


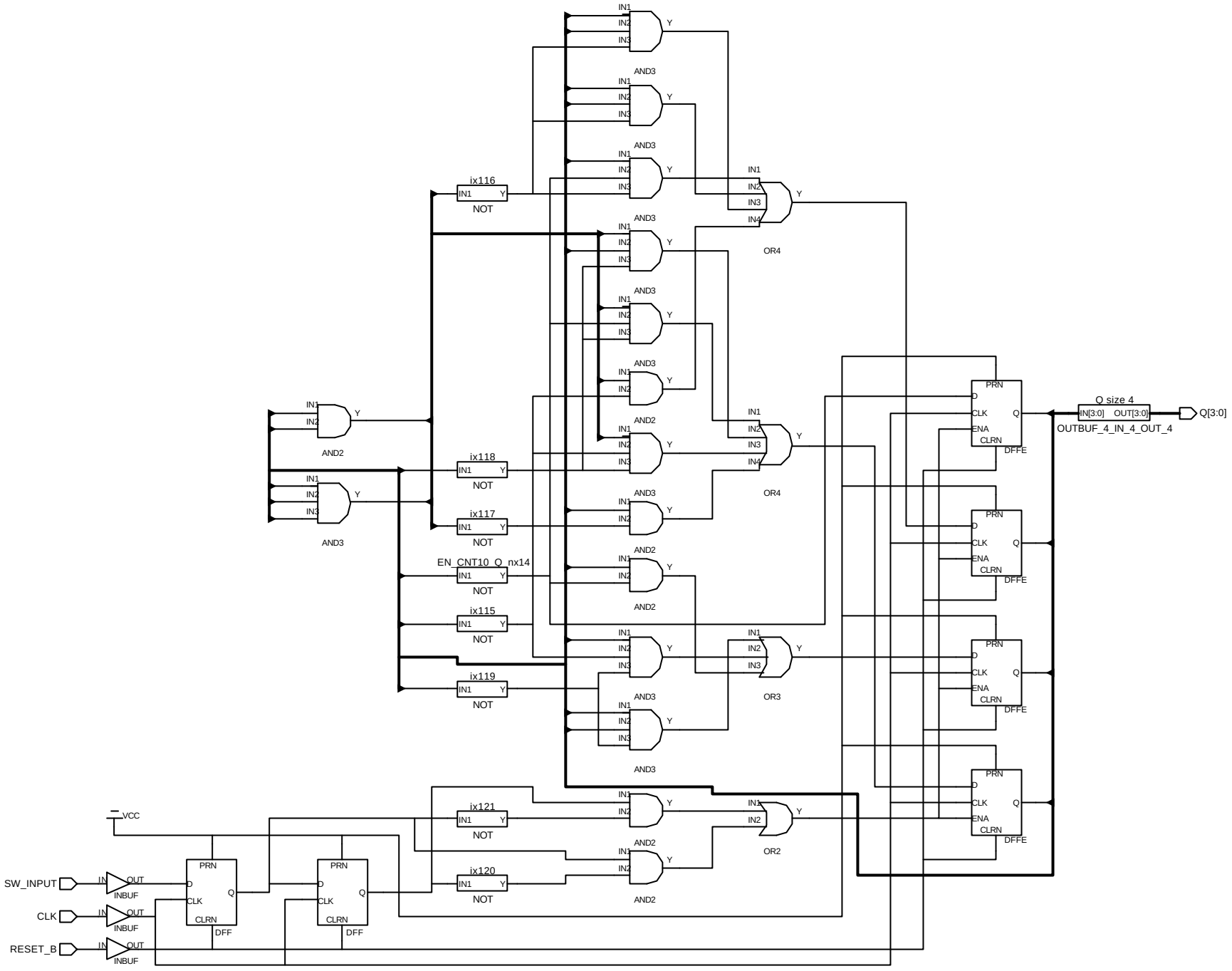

UP
LOAD
EN
IN[3:0]
CLK
RESET_B
INBUF
INBUF_4_IN_4_OUT_4
IN size 4
OR4
OR3
OR2
AND5
AND4
AND3
AND2
NOT
EXP
OR7
VCC
SOFT
OUTBUF
OUTBUF_4_IN_4_OUT_4
Q size 4
Q[3:0]
RC
DFFE

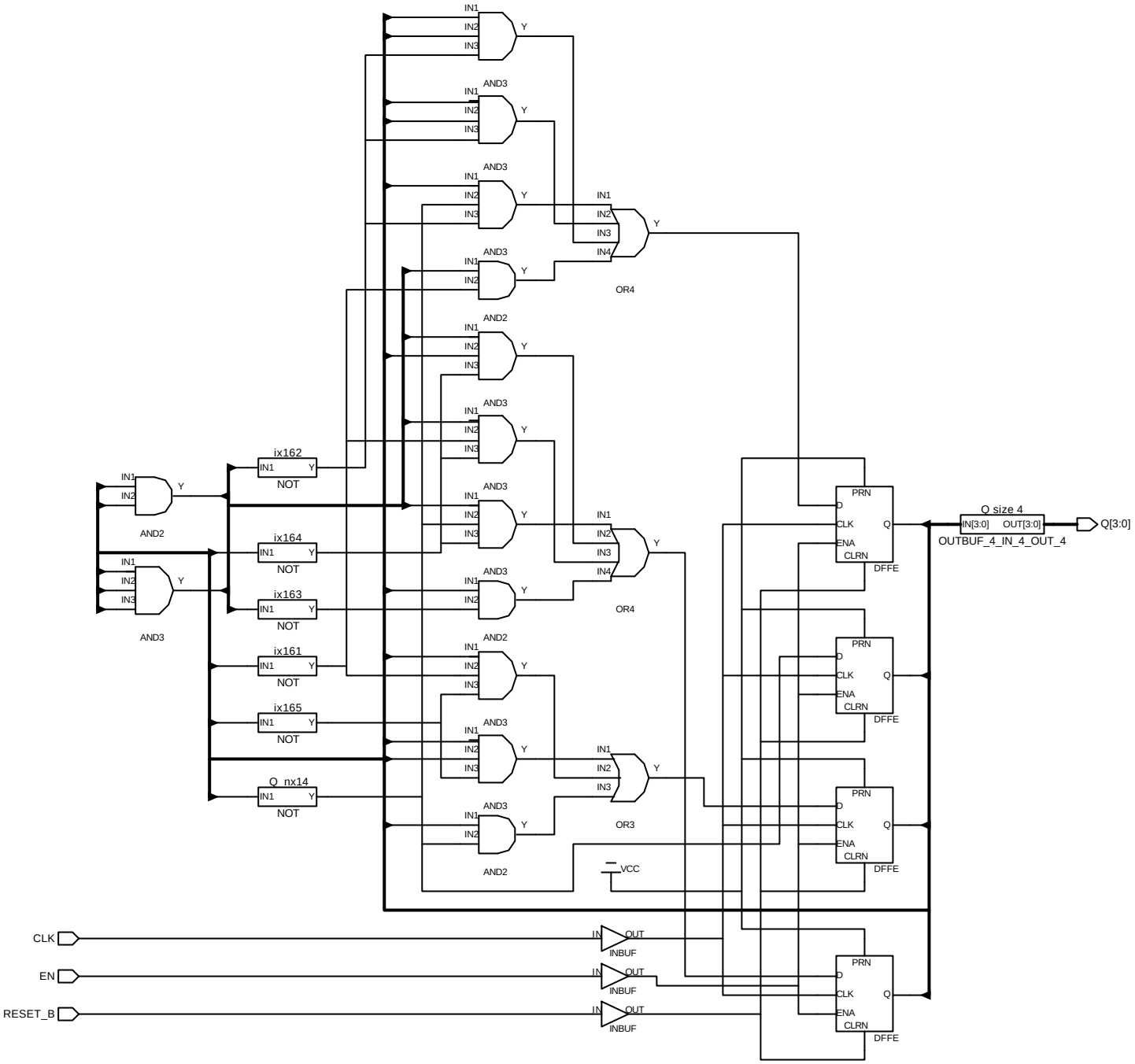


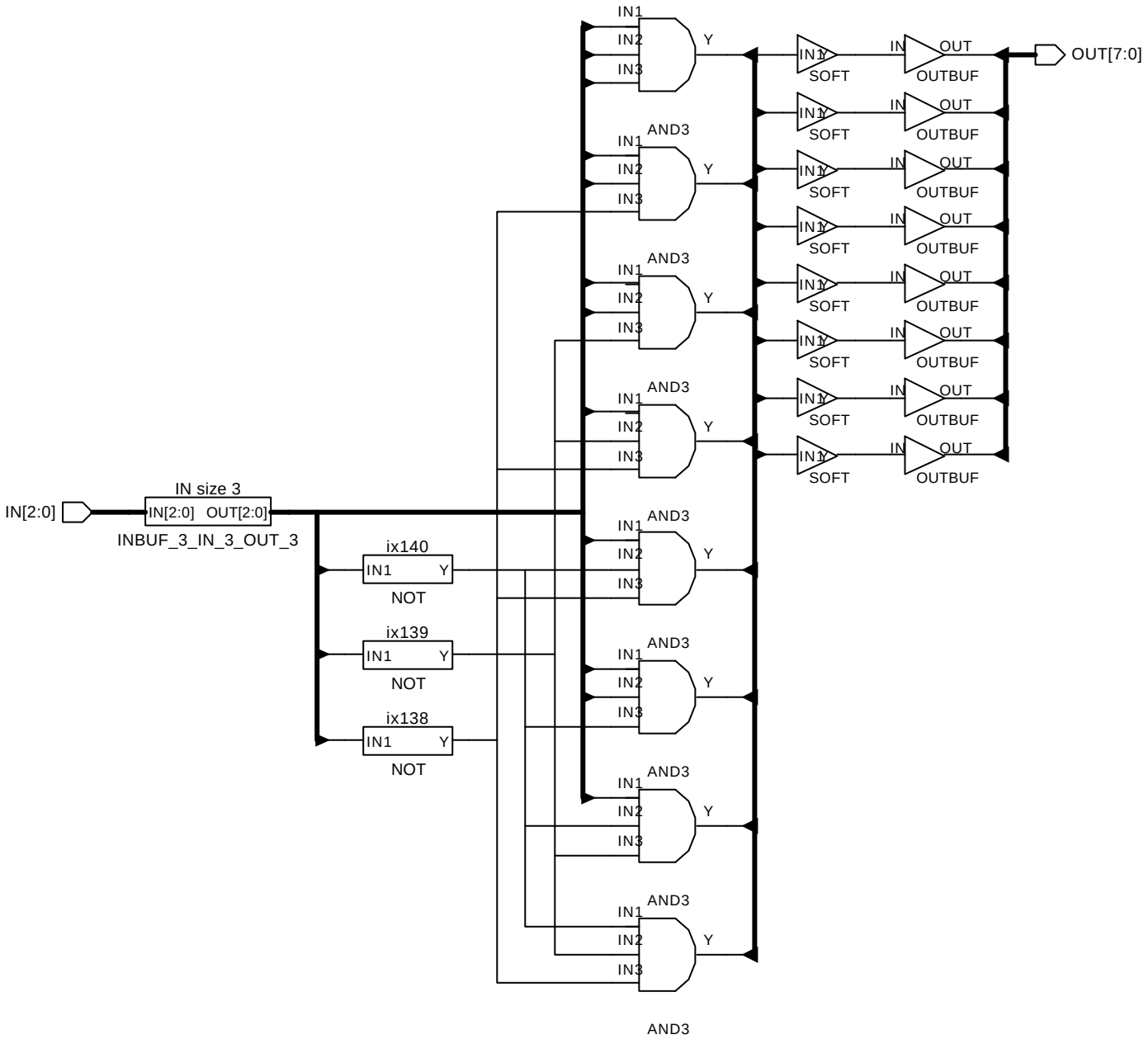

UP
LOAD
IN[3:0]
EN
CLK
RESET_B
INBUF
SOFT
OUTBUF
RC
VCC
AND2
AND3
AND4
AND5
OR2
OR3
NOT
DFFE
PRN
CLRN
ENA
IN size 4
INBUF_4_IN_4_OUT_4
Q size 4
OUTBUF_4_IN_4_OUT_4
Q[3:0]

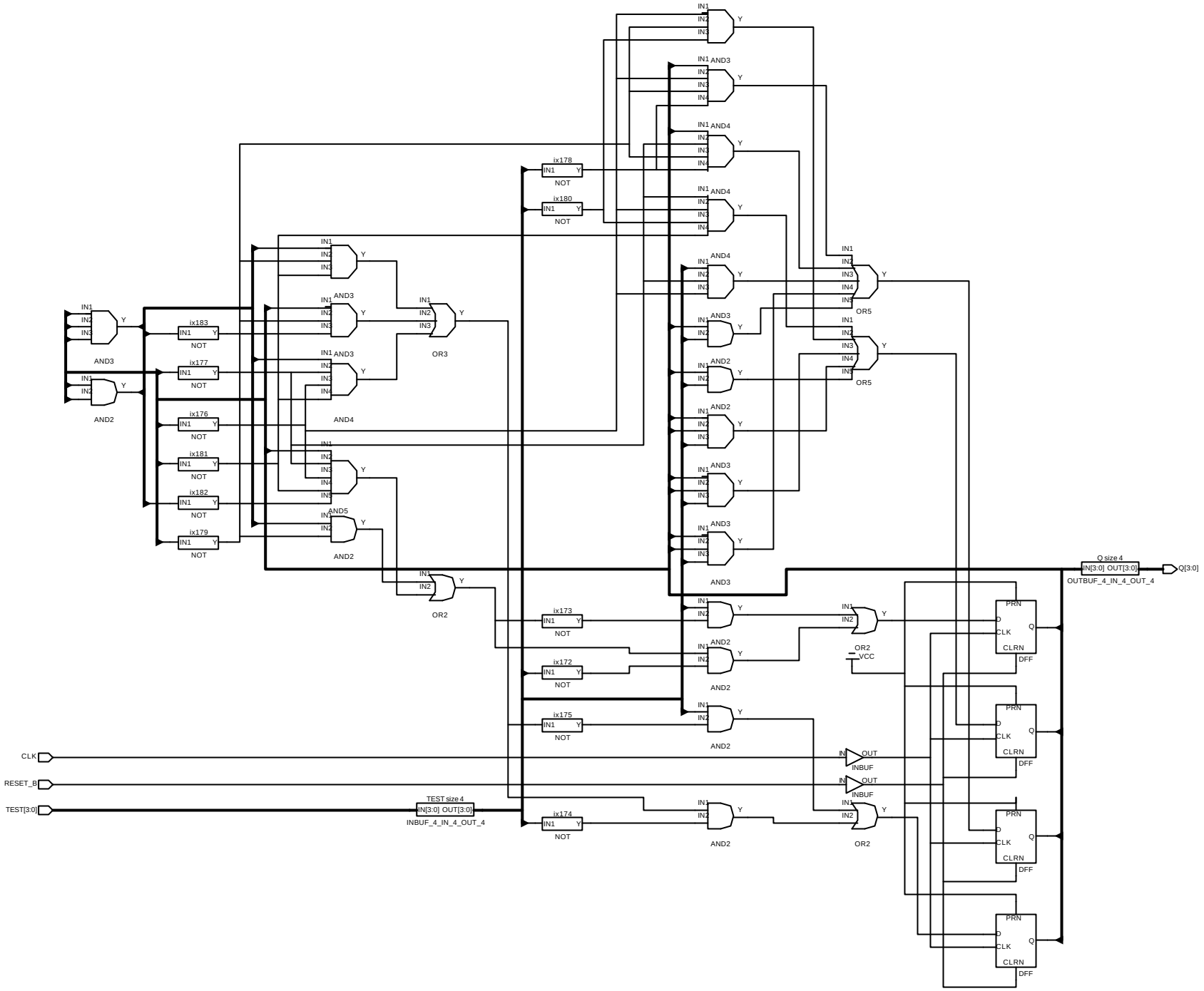

IN1
IN2
IN3
Y
AND3
AND2
IN4
OR4
OR3
NOT
ix110
ix109
ix108
ix107
ix111
Q_nx13
VCC
PRN
D
CLK
CLRN
Q
DFF
Q_size 4
IN[3:0]
OUT[3:0]
OUTBUF_4_IN_4_OUT_4
Q[3:0]
CLK
RESET_B
IN
OUT
INBUF

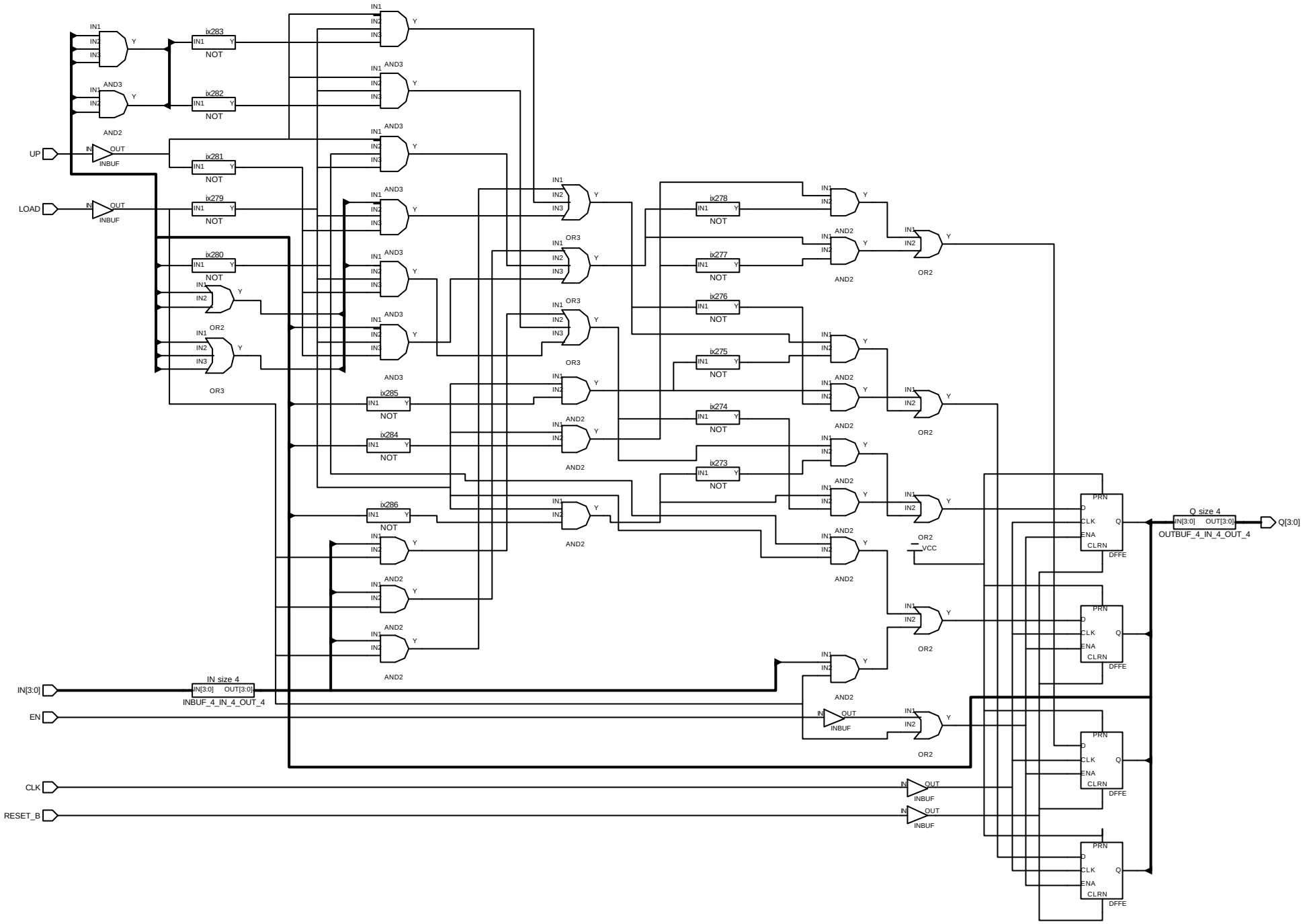

TEST
CLK
RESET_B
ix120
ix119
OUT(0)
ix122
ix121
ix123
NOT
AND3
AND2
OR5
OR4
OR3
VCC
INBUF
DFF
PRN
CLK
CLRN
D
Q
IN1
IN2
IN3
IN4
IN5
Y
IN
OUT
O size 4
IN[3:0]
OUT[3:0]
OUTBUF_4_IN_4_OUT_4
Q[3:0]

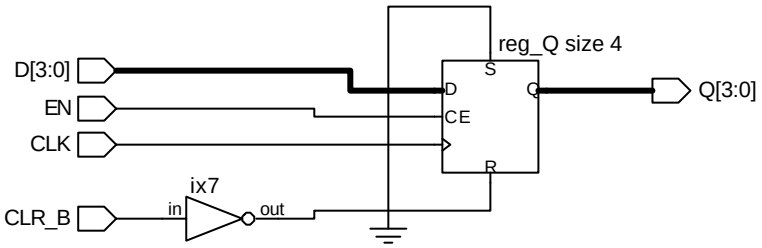

SW_INPUT
CLK
RESET_B
VCC
INBUF
DFF
PRN
CLRN
ix116
ix117
ix118
ix119
ix120
ix121
EN_CNT10_Q_nx14
NOT
AND2
AND3
OR2
OR3
OR4
DFFE
Q size 4
IN[3:0] OUT[3:0]
OUTBUF_4_IN_4_OUT_4
Q[3:0]

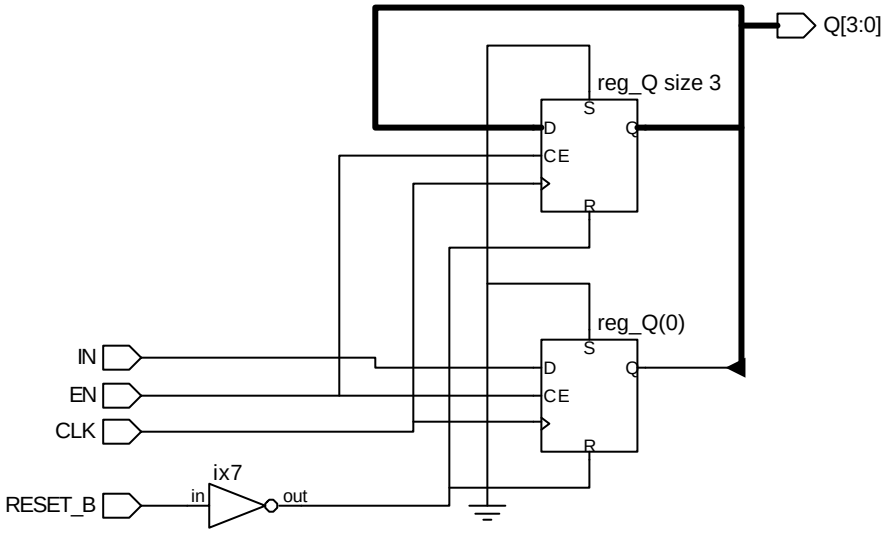

IN1
IN2
IN3
Y
AND3
AND2
OR4
OR3
NOT
ix162
ix164
ix163
ix161
ix165
Q_nx14
IN4
PRN
D
CLK
Q
ENA
CLRN
DFFE
Q size 4
IN[3:0]
OUT[3:0]
OUTBUF_4_IN_4_OUT_4
Q[3:0]
VCC
CLK
EN
RESET_B
INBUF
OUT

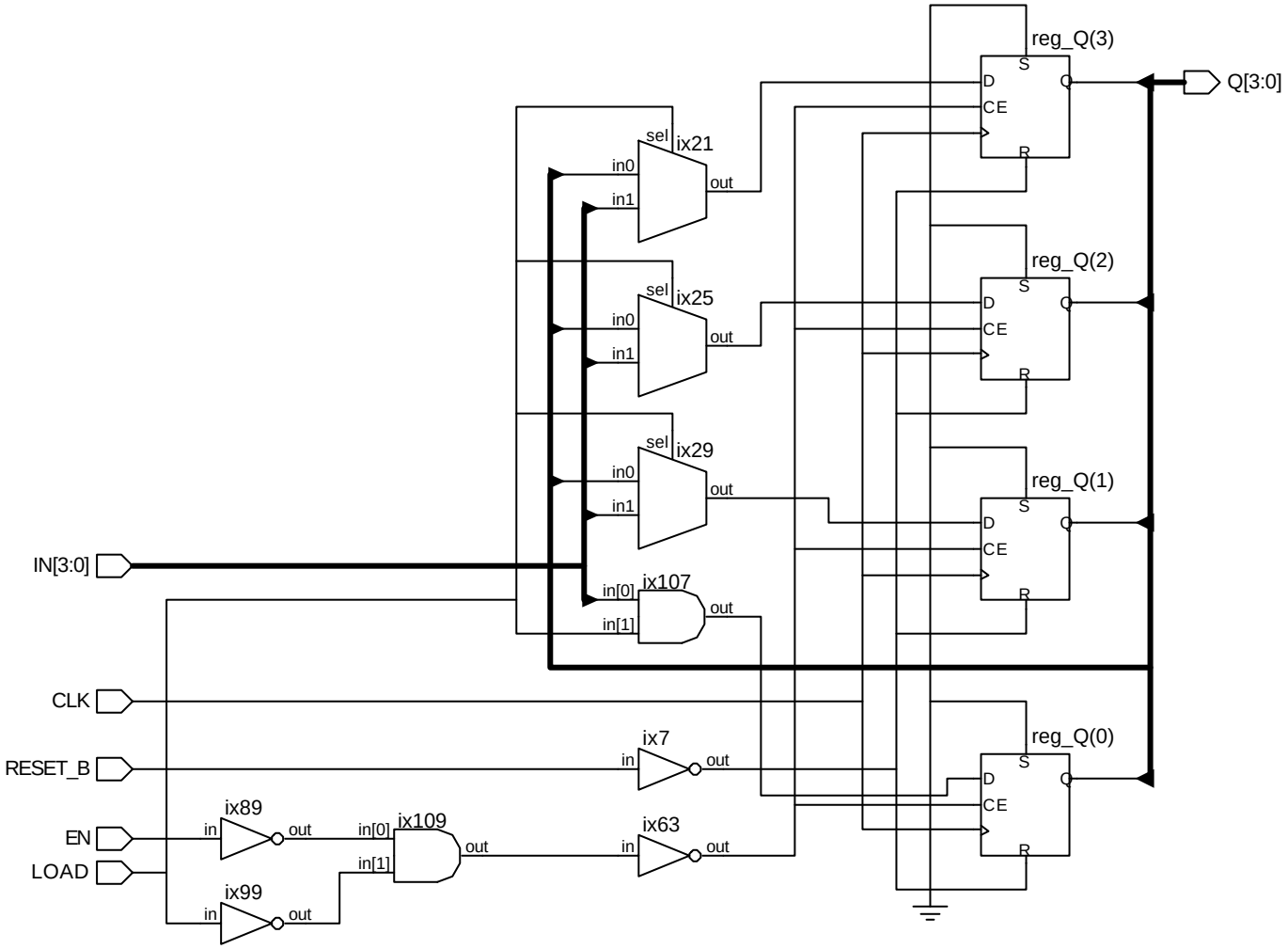

IN[2:0]
IN size 3
IN[2:0] OUT[2:0]
INBUF_3_IN_3_OUT_3
ix140
IN1 Y
NOT
ix139
IN1 Y
NOT
ix138
IN1 Y
NOT
AND3
IN1
IN2
IN3
Y
AND3
AND3
AND3
AND3
AND3
AND3
AND3
IN Y
SOFT
IN OUT
OUTBUF
OUT[7:0]

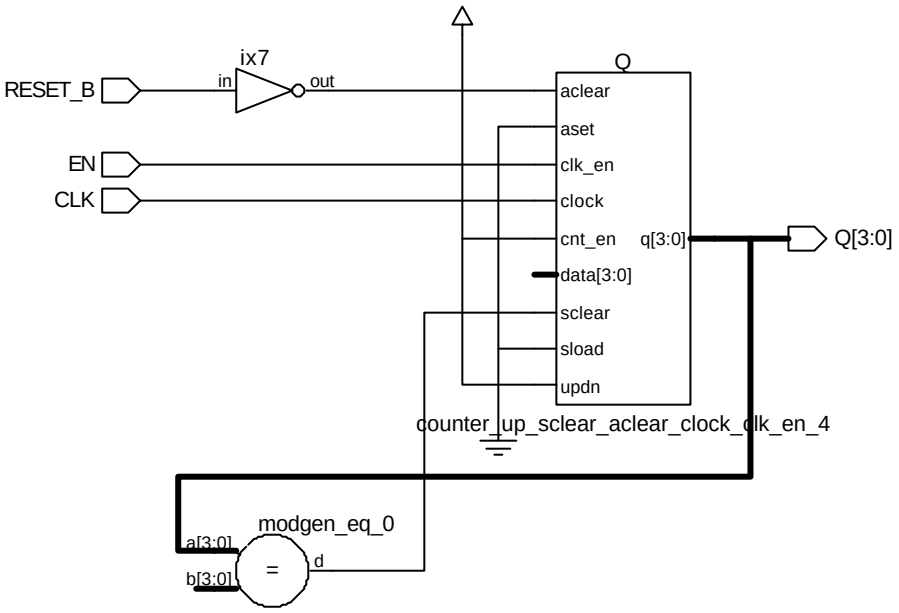

CLK
RESET_B
TEST[3:0]
TEST size 4
IN[3:0] OUT[3:0]
INBUF_4_IN_4_OUT_4
Q size 4
IN[3:0] OUT[3:0]
OUTBUF_4_IN_4_OUT_4
Q[3:0]
AND2
AND3
AND4
AND5
OR2
OR3
OR5
NOT
ix172
ix173
ix174
ix175
ix176
ix177
ix178
ix179
ix180
ix181
ix182
ix183
INBUF
VCC
DFF
PRN
CLRN
D
CLK
Q
IN1
IN2
IN3
IN4
IN5
Y

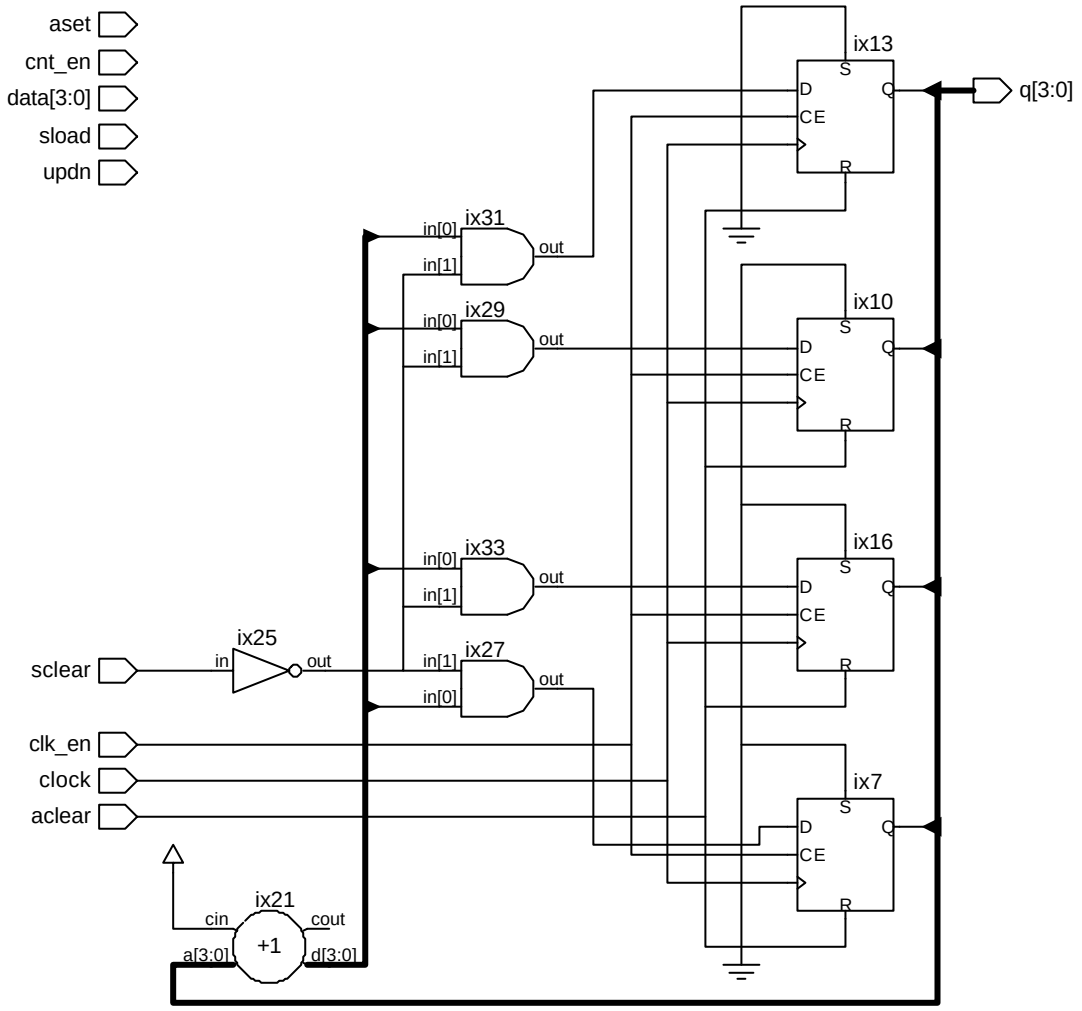

UP
LOAD
IN[3:0]
EN
CLK
RESET_B
INBUF
IN size 4
INBUF_4_IN_4_OUT_4
Q size 4
OUTBUF_4_IN_4_OUT_4
Q[3:0]
NOT
AND2
AND3
OR2
OR3
VCC
DFFE
PRN
CLK
ENA
CLRN
ix273
ix274
ix275
ix276
ix277
ix278
ix279
ix280
ix281
ix282
ix283
ix284
ix285
ix286

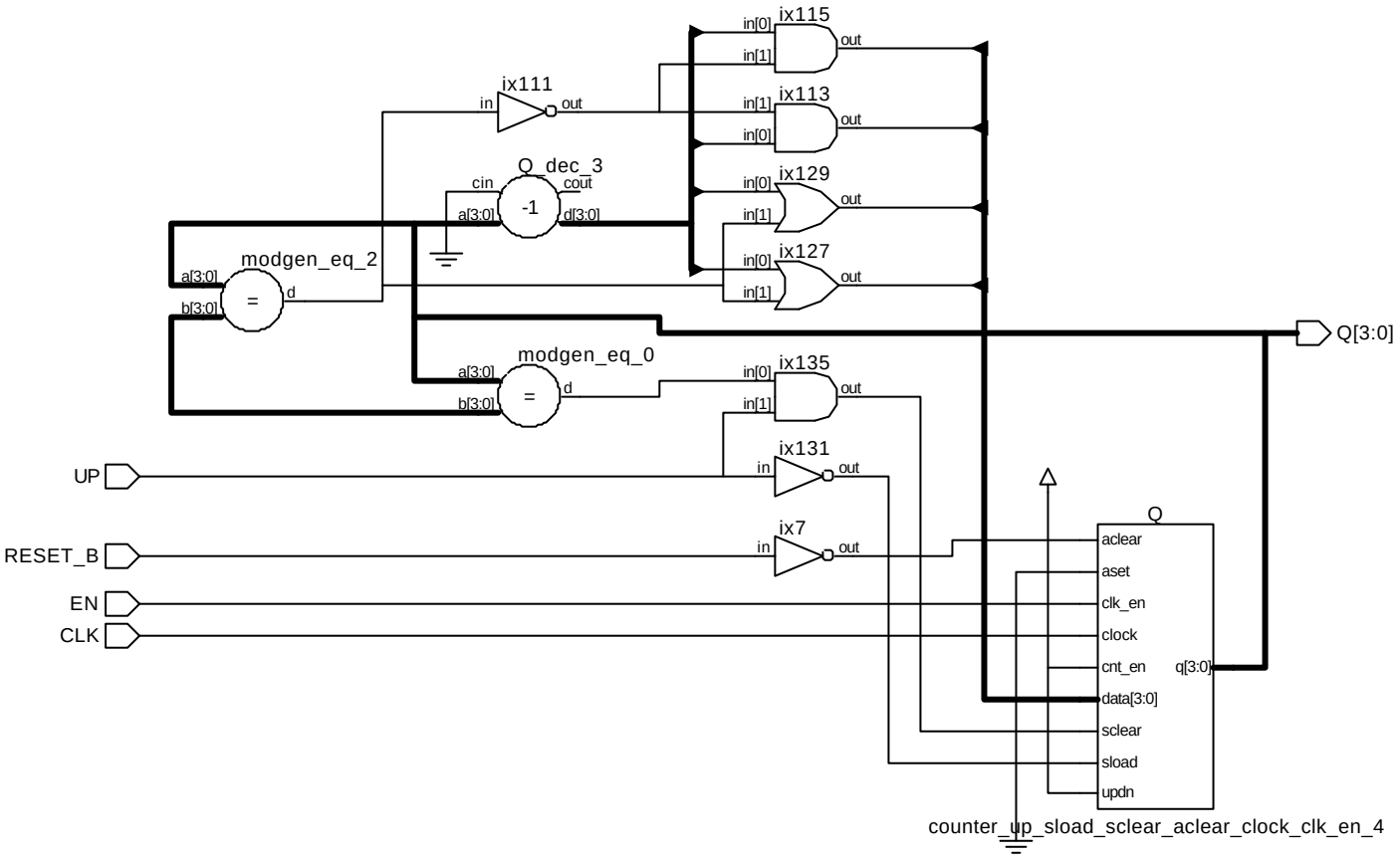

reg_Q size 4
D[3:0]
EN
CLK
CLR_B
ix7
in
out
S
D
CE
R
Q
Q[3:0]

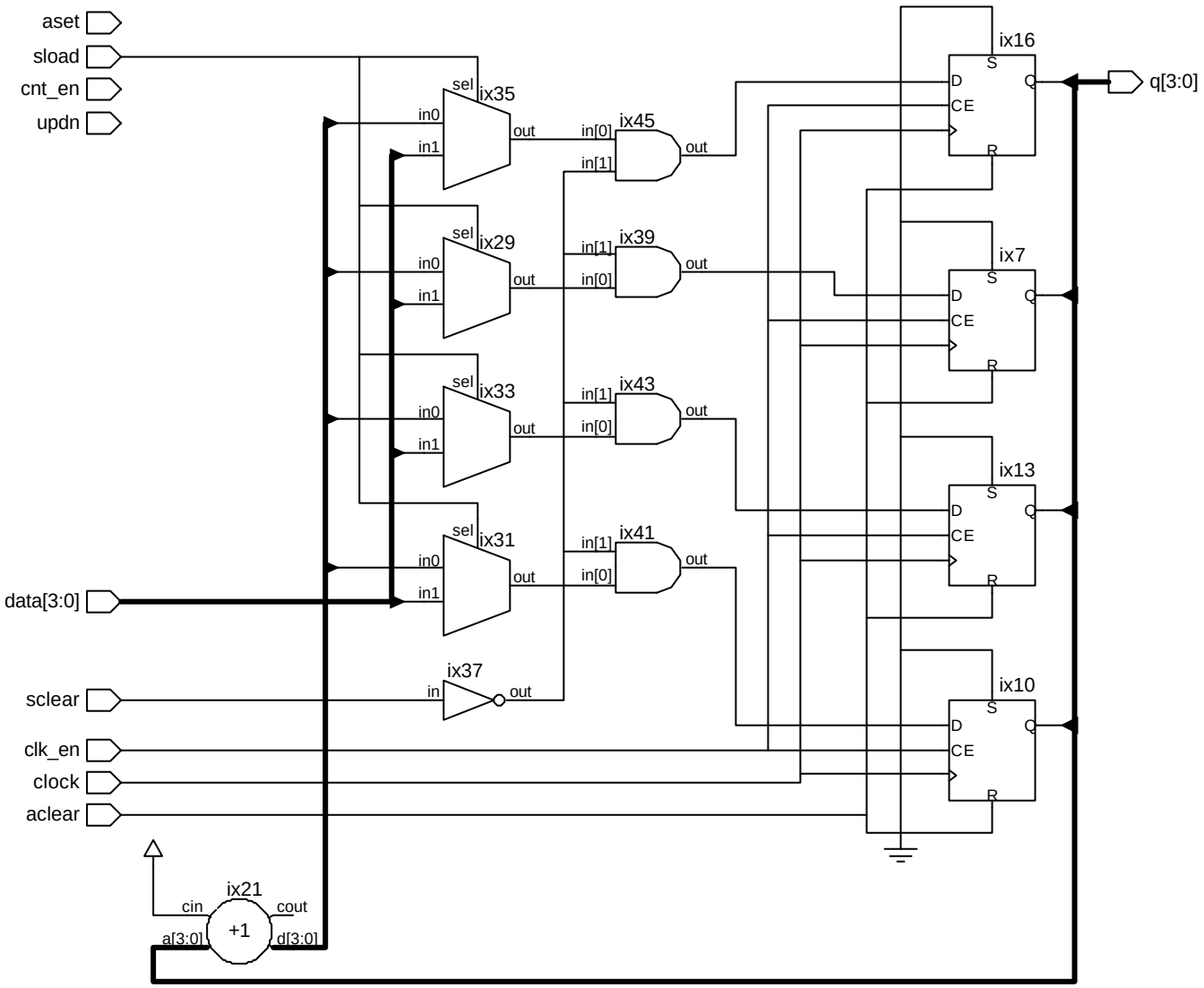

Q[3:0]
reg_Q size 3
S
D
Q
CE
R
reg_Q(0)
S
D
Q
CE
R
IN
EN
CLK
RESET_B
ix7
in
out

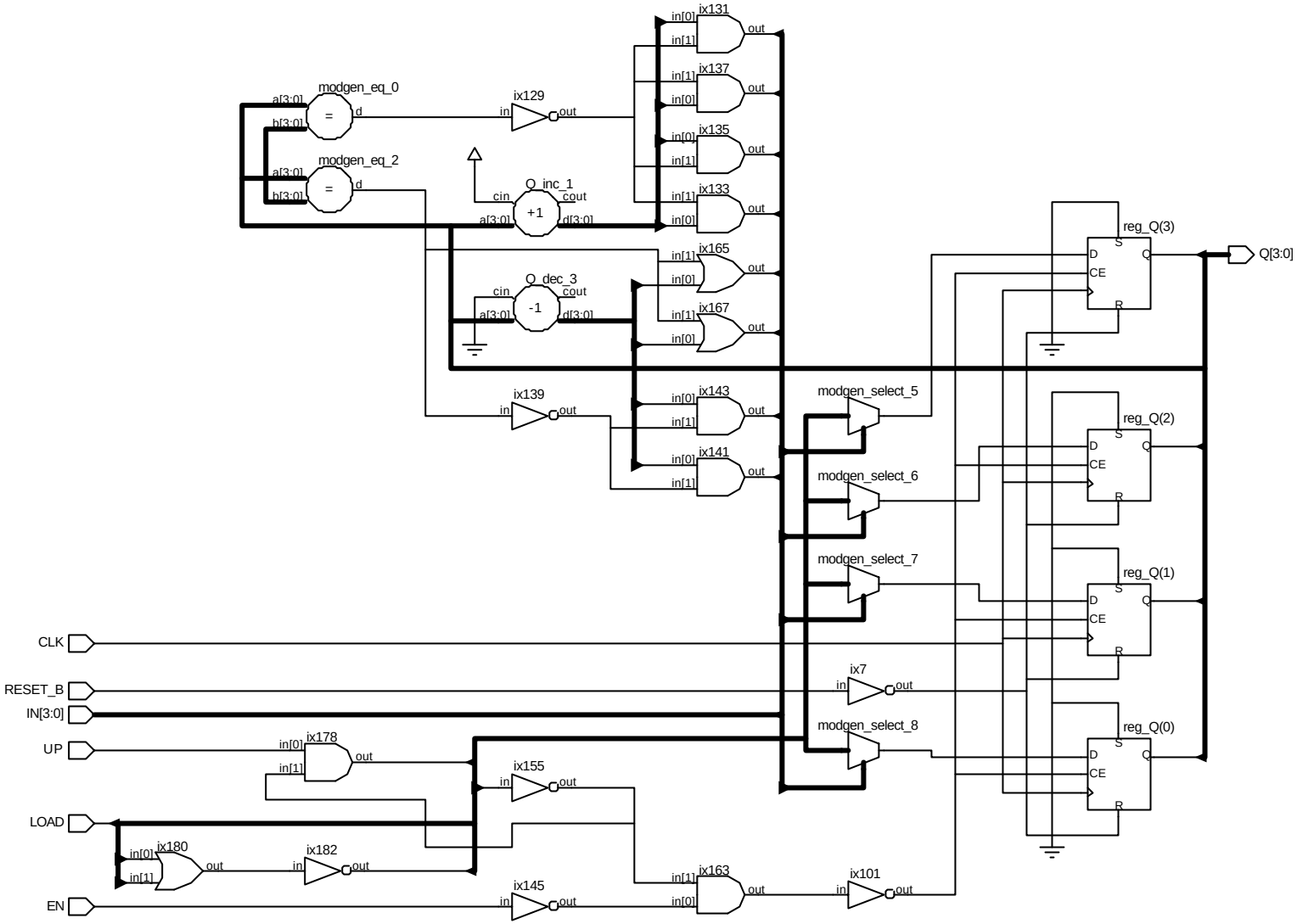

reg_Q(3)
reg_Q(2)
reg_Q(1)
reg_Q(0)
S
D
CE
R
Q
Q[3:0]
ix21
ix25
ix29
ix107
sel
in0
in1
in[0]
in[1]
out
IN[3:0]
CLK
RESET_B
EN
LOAD
ix7
ix89
ix99
ix109
ix63
in

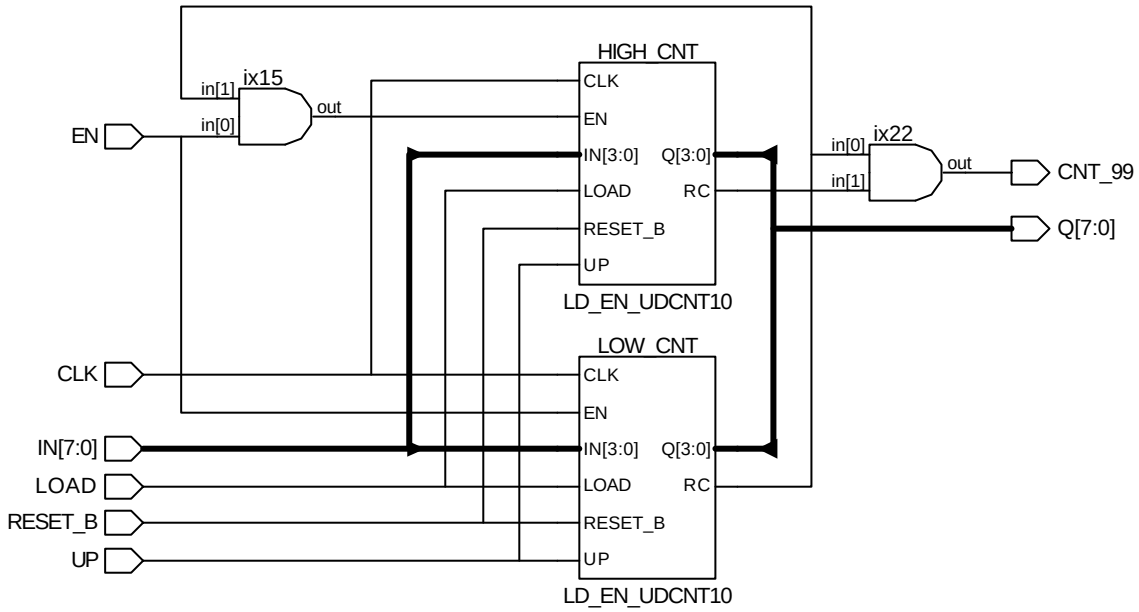

ix7
in
out
RESET_B
EN
CLK
Q
aclear
aset
clk_en
clock
cnt_en
q[3:0]
data[3:0]
sclear
sload
updn
Q[3:0]
counter_up_sclear_aclear_clock_clk_en_4
modgen_eq_0
a[3:0]
b[3:0]
=
d

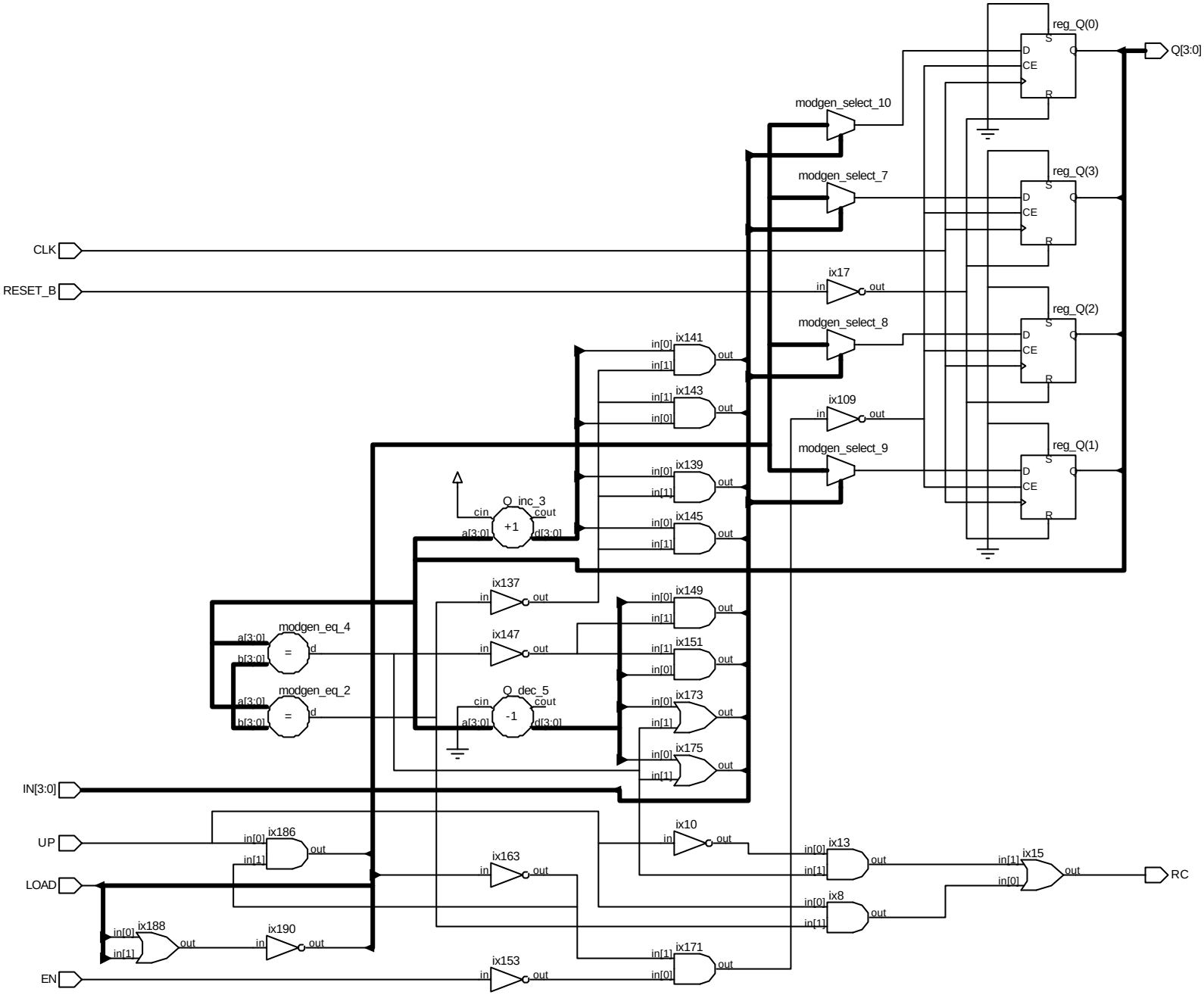

aset
cnt_en
data[3:0]
sload
updn
sclear
clk_en
clock
aclear
q[3:0]
ix13
ix10
ix16
ix7
ix31
ix29
ix33
ix27
ix25
ix21
S
D
CE
R
Q
in[0]
in[1]
out
in
cin
cout
a[3:0]
d[3:0]
+1

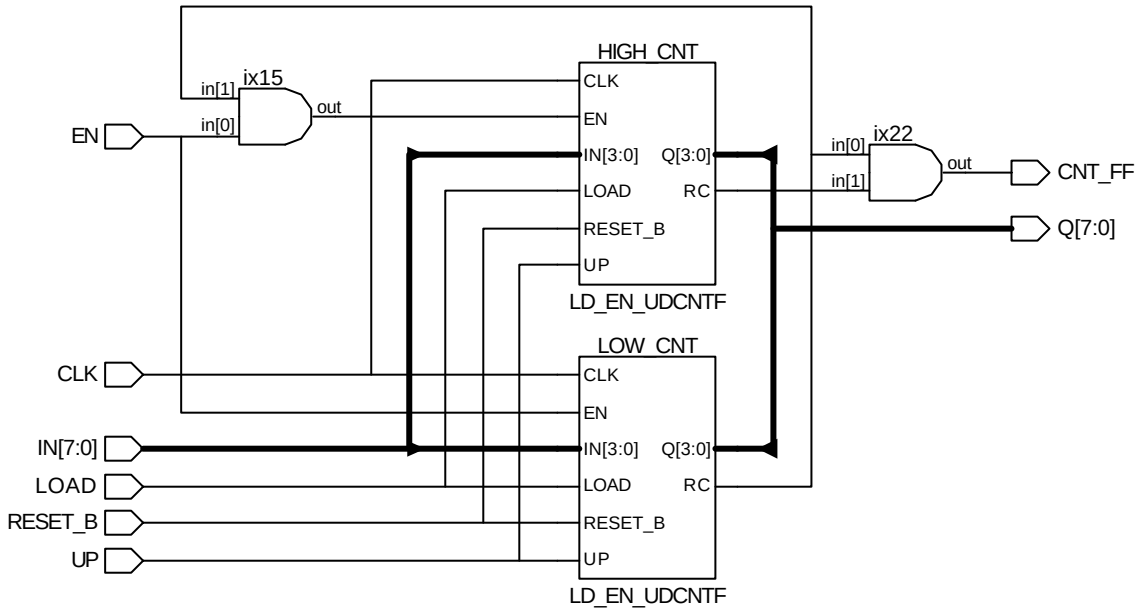

ix115
in[0]
in[1]
out
ix111
in
out
ix113
in[1]
in[0]
out
Q_dec_3
cin
cout
a[3:0]
-1
d[3:0]
ix129
in[0]
in[1]
out
modgen_eq_2
a[3:0]
b[3:0]
=
d
ix127
in[0]
in[1]
out
Q[3:0]
modgen_eq_0
a[3:0]
b[3:0]
=
d
ix135
in[0]
in[1]
out
ix131
in
out
UP
ix7
in
out
RESET_B
EN
CLK
Q
aclear
aset
clk_en
clock
cnt_en
q[3:0]
data[3:0]
sclear
sload
updn
counter_up_sload_sclear_aclear_clock_clk_en_4

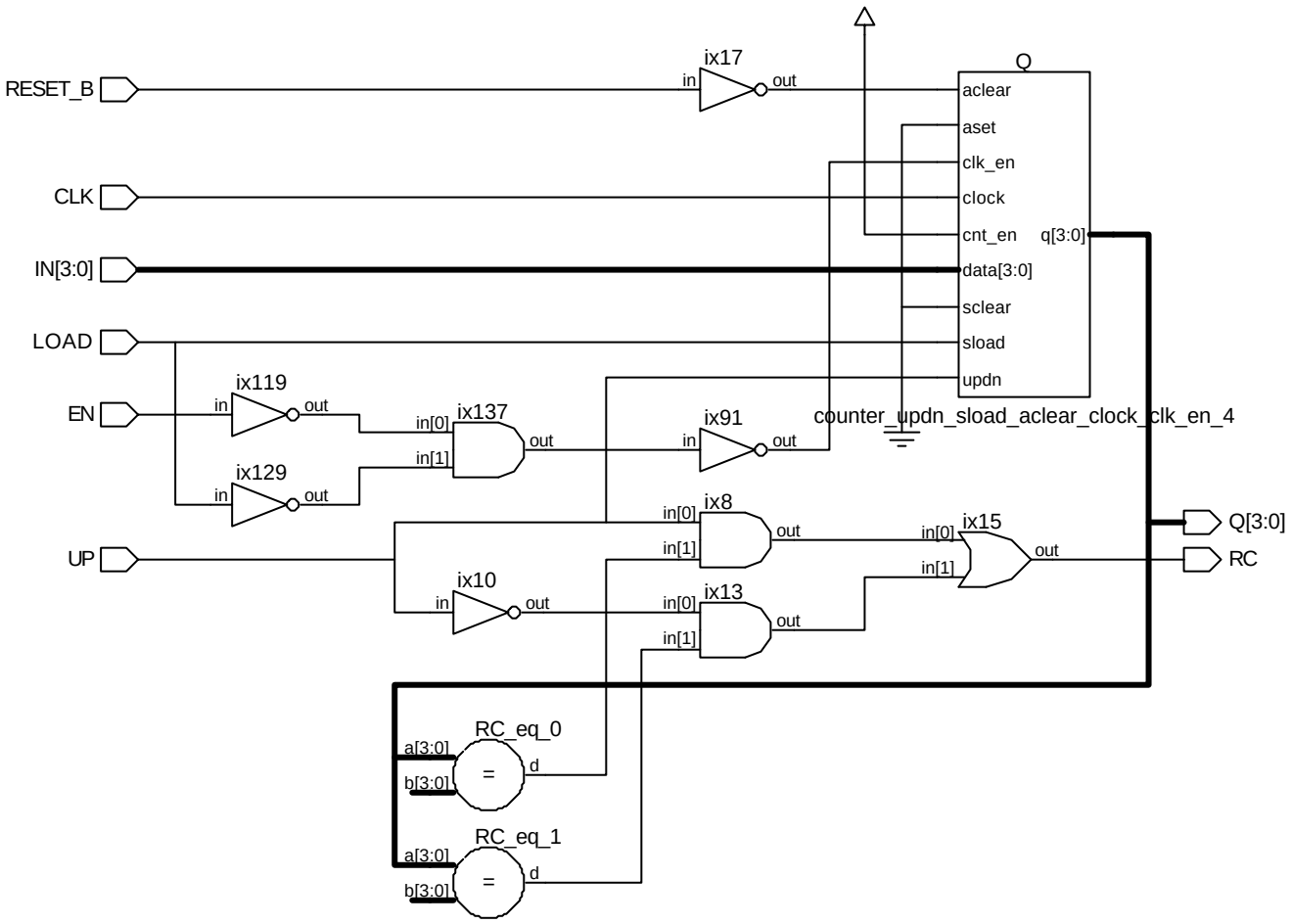

aset
sload
cnt_en
updn
data[3:0]
sclear
clk_en
clock
aclear
ix35
ix29
ix33
ix31
ix37
ix45
ix39
ix43
ix41
ix16
ix7
ix13
ix10
ix21
sel
in0
in1
out
in
in[0]
in[1]
D
CE
S
R
Q
q[3:0]
cin
cout
a[3:0]
d[3:0]
+1

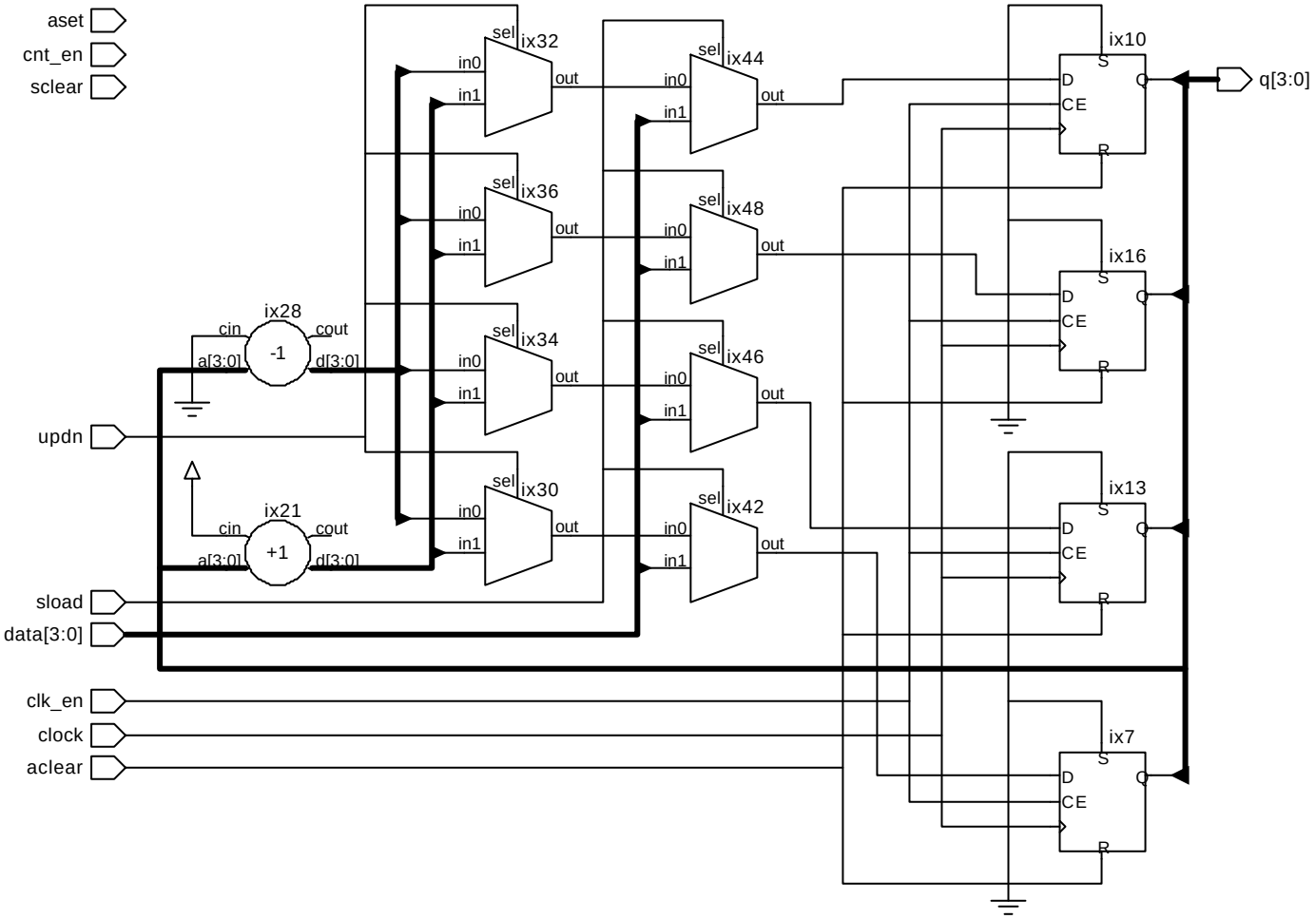

modgen_eq_0
modgen_eq_2
O_inc_1
O_dec_3
ix129
ix131
ix137
ix135
ix133
ix165
ix167
ix139
ix143
ix141
modgen_select_5
modgen_select_6
modgen_select_7
modgen_select_8
reg_Q(3)
reg_Q(2)
reg_Q(1)
reg_Q(0)
Q[3:0]
CLK
RESET_B
IN[3:0]
UP
LOAD
EN
ix7
ix178
ix155
ix180
ix182
ix145
ix163
ix101

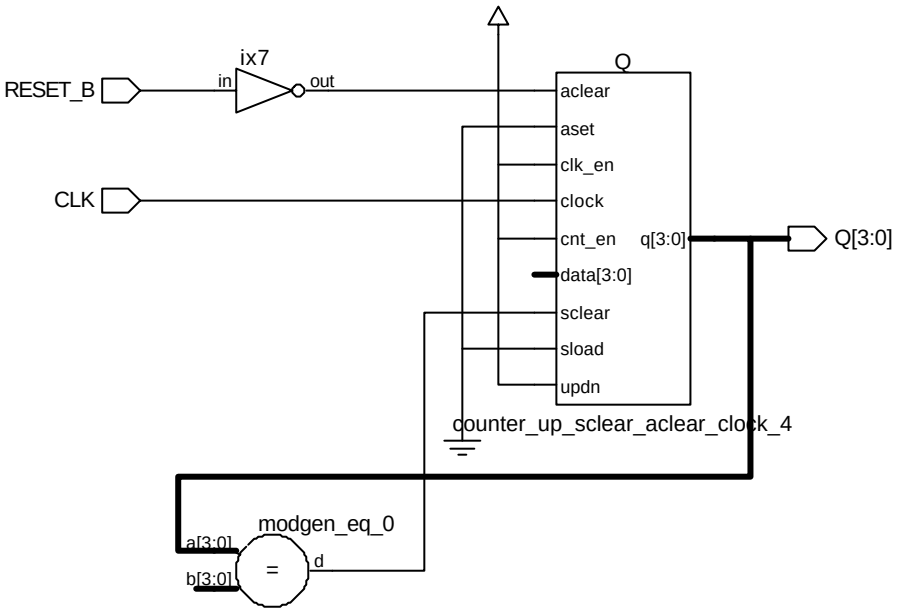

HIGH_CNT
ix15
in[1]
in[0]
out
EN
CLK
EN
IN[3:0]
Q[3:0]
LOAD
RC
RESET_B
UP
LD_EN_UDCNT10
ix22
in[0]
in[1]
out
CNT_99
Q[7:0]
LOW_CNT
CLK
IN[7:0]
LOAD
RESET_B
UP
CLK
EN
IN[3:0]
Q[3:0]
LOAD
RC
RESET_B
UP
LD_EN_UDCNT10

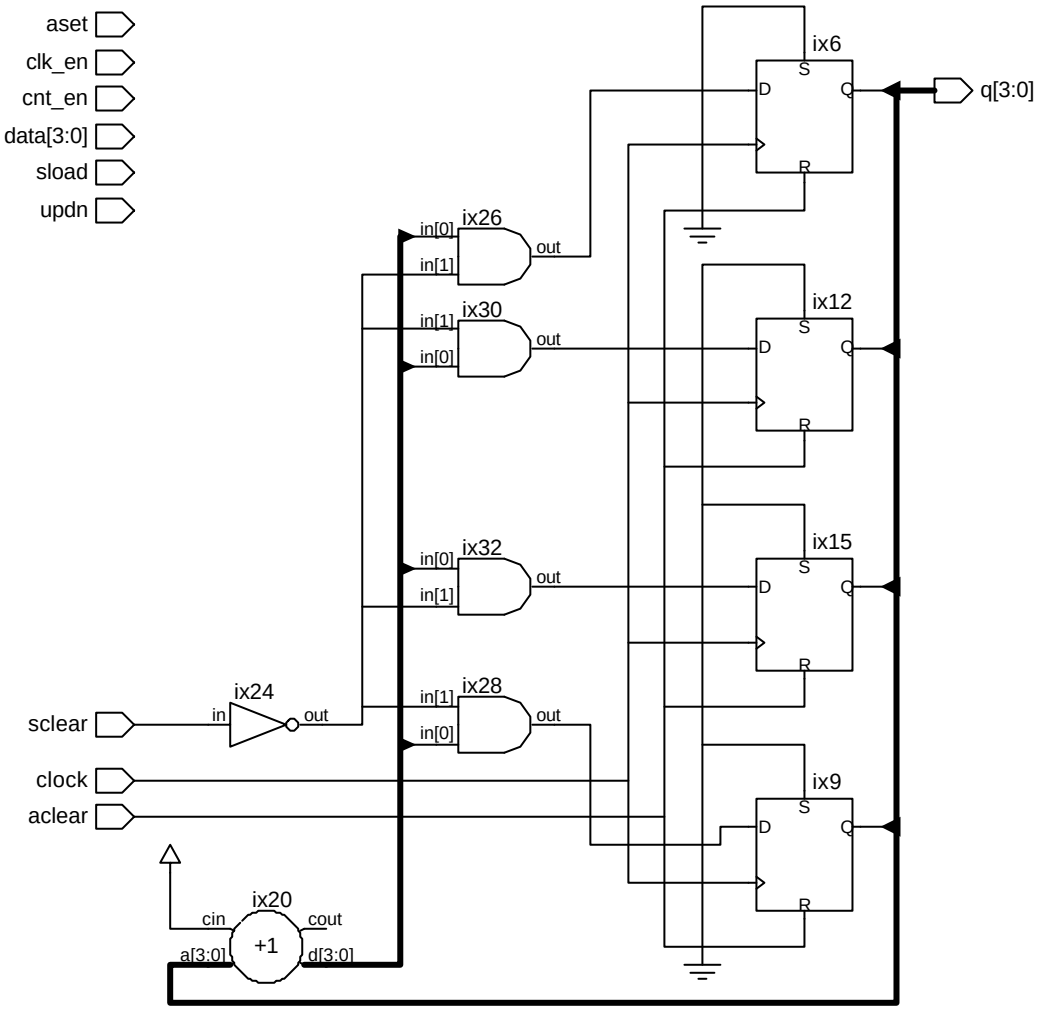

reg_Q(0)
S
D
CE
R
Q
Q[3:0]
modgen_select_10
reg_Q(3)
modgen_select_7
CLK
RESET_B
ix17
in
out
reg_Q(2)
modgen_select_8
ix141
in[0]
in[1]
out
ix143
ix109
reg_Q(1)
modgen_select_9
ix139
ix145
Q_inc_3
cin
cout
a[3:0]
+1
d[3:0]
ix137
ix149
modgen_eq_4
b[3:0]
=
d
ix147
ix151
modgen_eq_2
Q_dec_5
-1
ix173
ix175
IN[3:0]
UP
LOAD
EN
ix186
ix10
ix13
ix15
RC
ix163
ix8
ix188
ix190
ix153
ix171

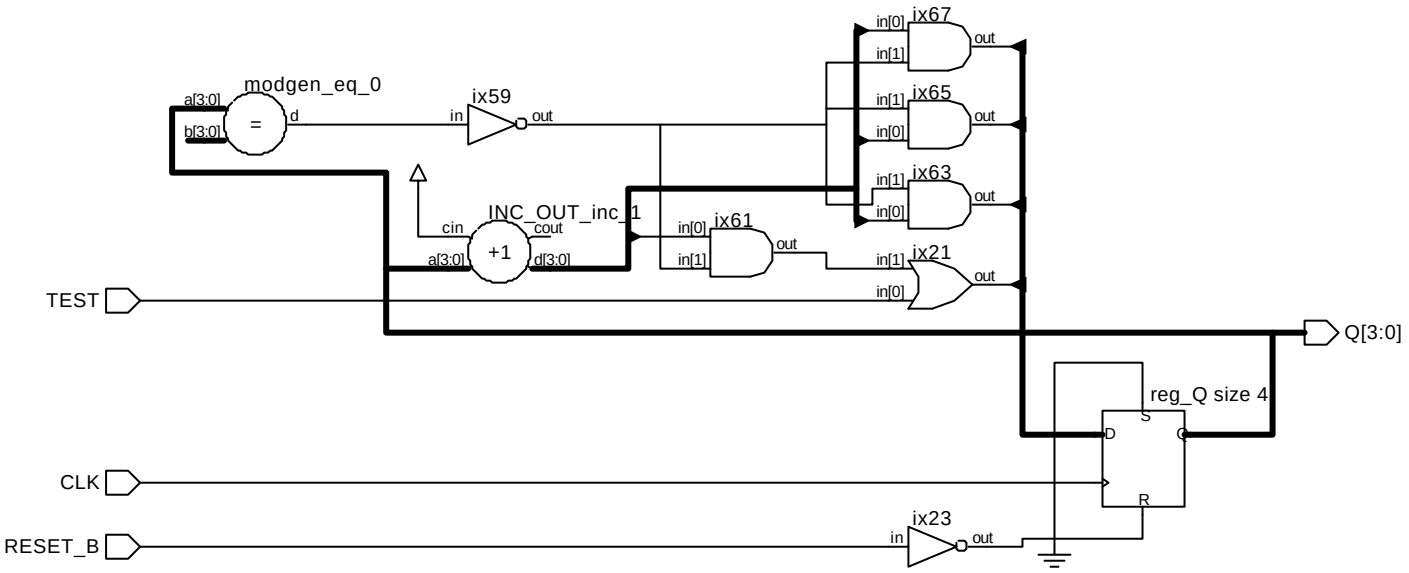

HIGH_CNT
ix15
in[1]
in[0]
out
EN
CLK
EN
IN[3:0]
Q[3:0]
LOAD
RC
RESET_B
UP
LD_EN_UDCNTF
ix22
in[0]
in[1]
out
CNT_FF
Q[7:0]
LOW_CNT
CLK
IN[7:0]
LOAD
RESET_B
UP
CLK
EN
IN[3:0]
Q[3:0]
LOAD
RC
RESET_B
UP
LD_EN_UDCNTF

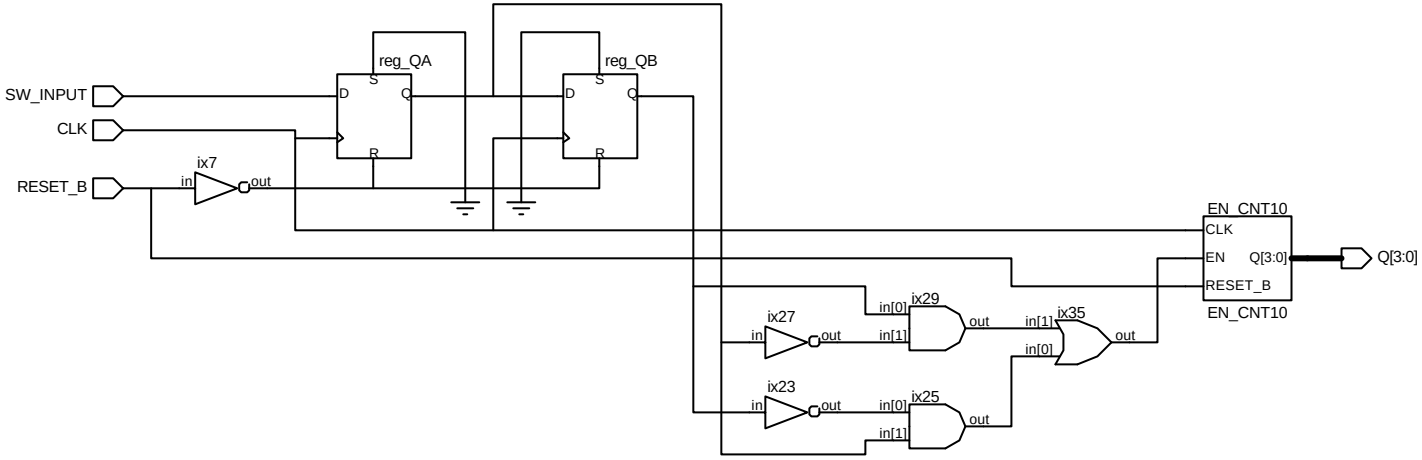

RESET_B
CLK
IN[3:0]
LOAD
EN
UP
ix17
in
out
Q
aclear
aset
clk_en
clock
cnt_en
q[3:0]
data[3:0]
sclear
sload
updn
counter_updn_sload_aclear_clock_clk_en_4
ix119
ix129
ix137
ix91
ix8
ix10
ix13
ix15
in[0]
in[1]
Q[3:0]
RC
RC_eq_0
RC_eq_1
a[3:0]
b[3:0]
=
d

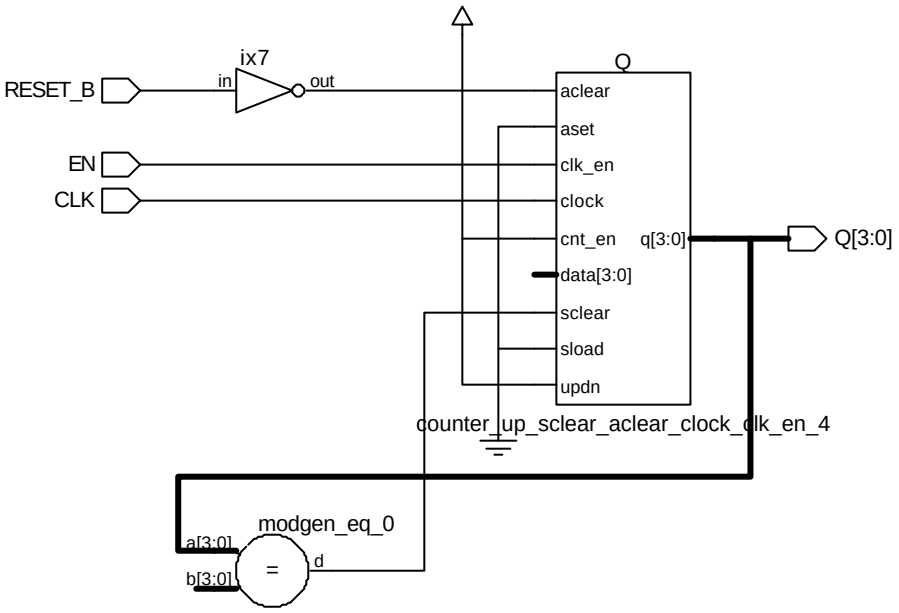

aset
cnt_en
sclear
updn
sload
data[3:0]
clk_en
clock
aclear
q[3:0]
ix32
ix36
ix34
ix30
ix44
ix48
ix46
ix42
ix28
ix21
ix10
ix16
ix13
ix7
sel
in0
in1
out
cin
cout
a[3:0]
d[3:0]
-1
+1
D
CE
S
R
Q

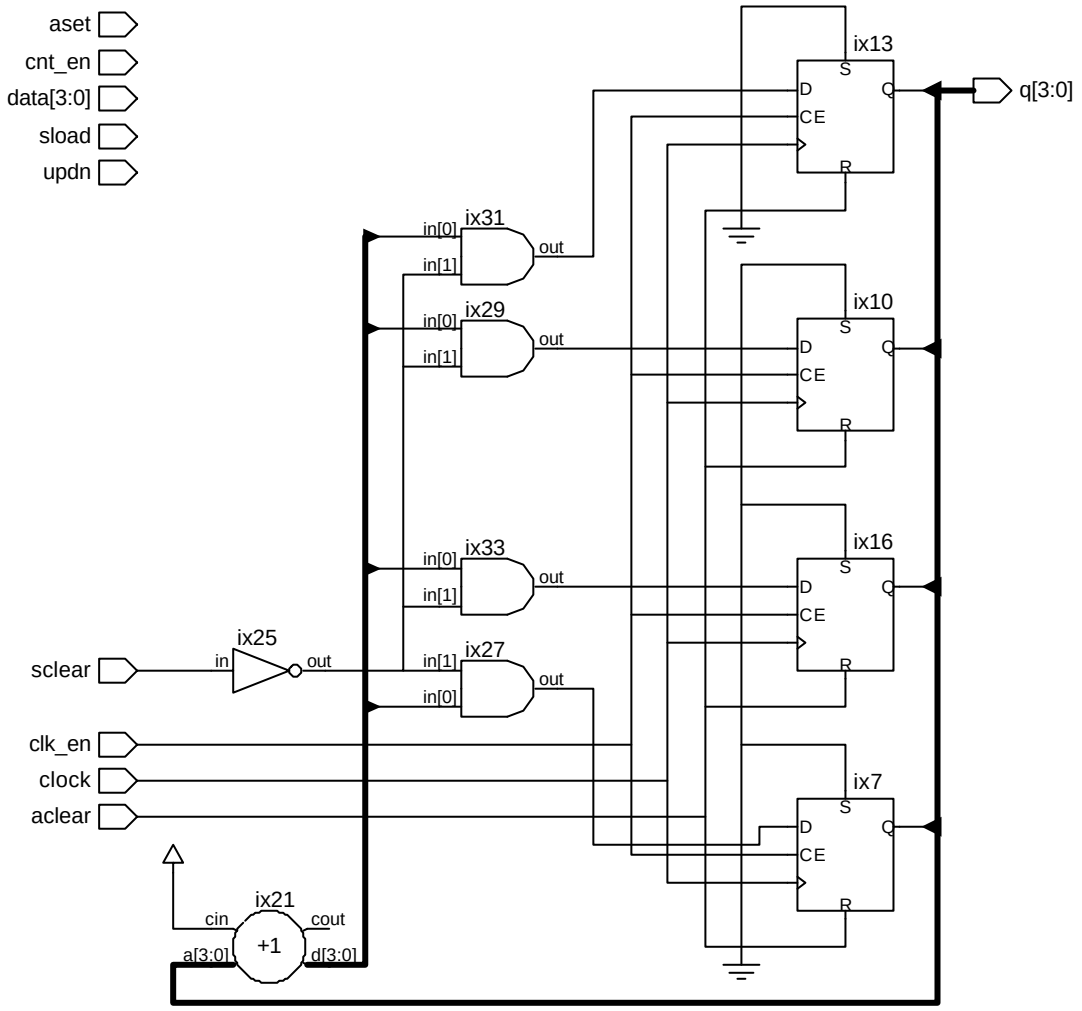

RESET_B
ix7
in
out
CLK
Q
aclear
aset
clk_en
clock
cnt_en
q[3:0]
data[3:0]
sclear
sload
updn
Q[3:0]
counter_up_sclear_aclear_clock_4
modgen_eq_0
a[3:0]
b[3:0]
=
d

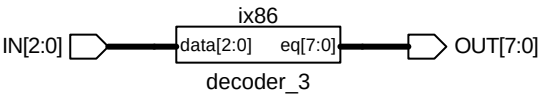

aset
clk_en
cnt_en
data[3:0]
sload
updn
sclear
clock
aclear
ix6
S
D
Q
R
q[3:0]
ix26
in[0]
in[1]
out
ix30
in[1]
in[0]
out
ix12
ix32
in[0]
in[1]
out
ix15
ix24
in
out
ix28
in[1]
in[0]
out
ix9
ix20
cin
cout
a[3:0]
+1
d[3:0]

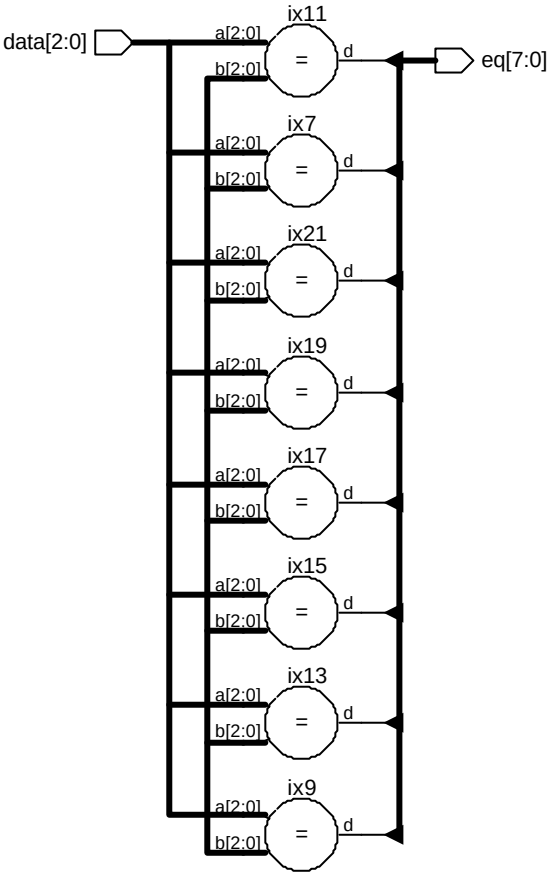

modgen_eq_0
a[3:0]
b[3:0]
=
d
ix59
in
out
INC_OUT_inc_1
cin
cout
a[3:0]
+1
d[3:0]
ix61
in[0]
in[1]
out
ix67
ix65
ix63
ix21
TEST
Q[3:0]
reg_Q size 4
S
D
Q
R
CLK
RESET_B
ix23

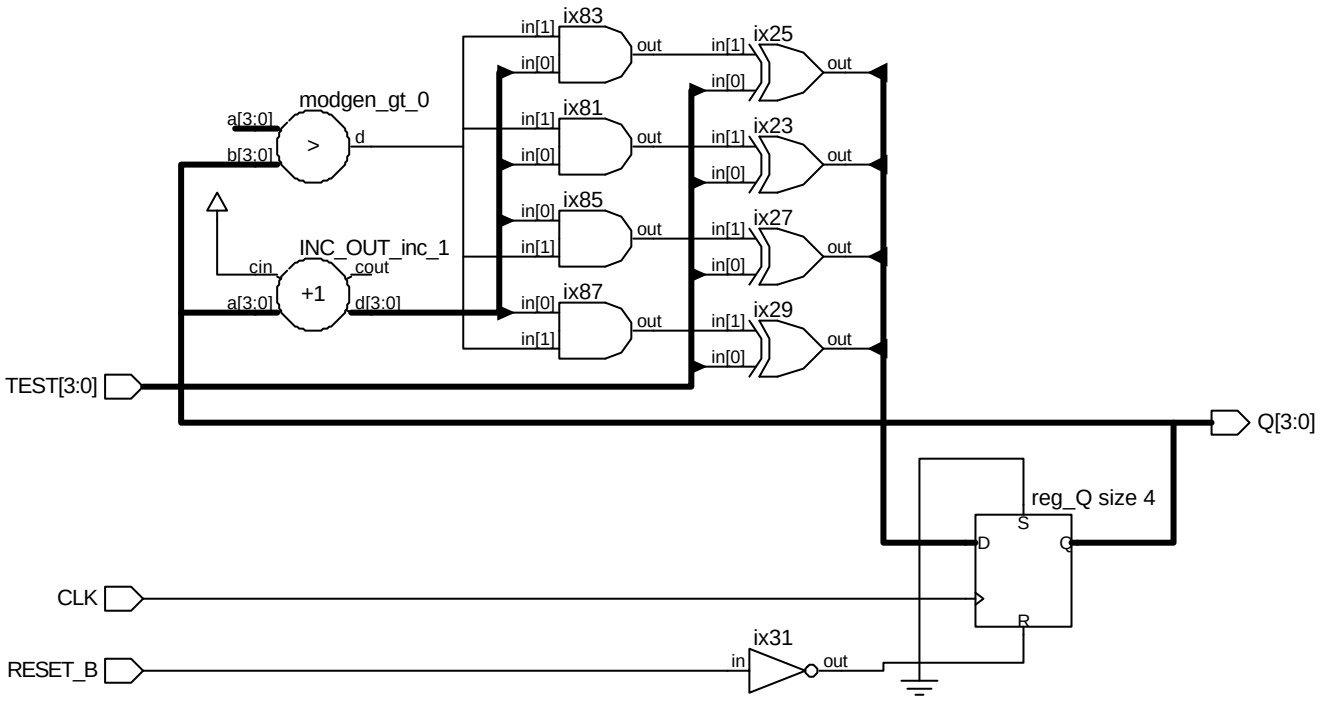

SW_INPUT
CLK
RESET_B
reg_QA
reg_QB
S
D
Q
R
ix7
in
out
ix27
ix23
ix29
ix25
ix35
in[0]
in[1]
EN_CNT10
CLK
EN
Q[3:0]
RESET_B
Q[3:0]

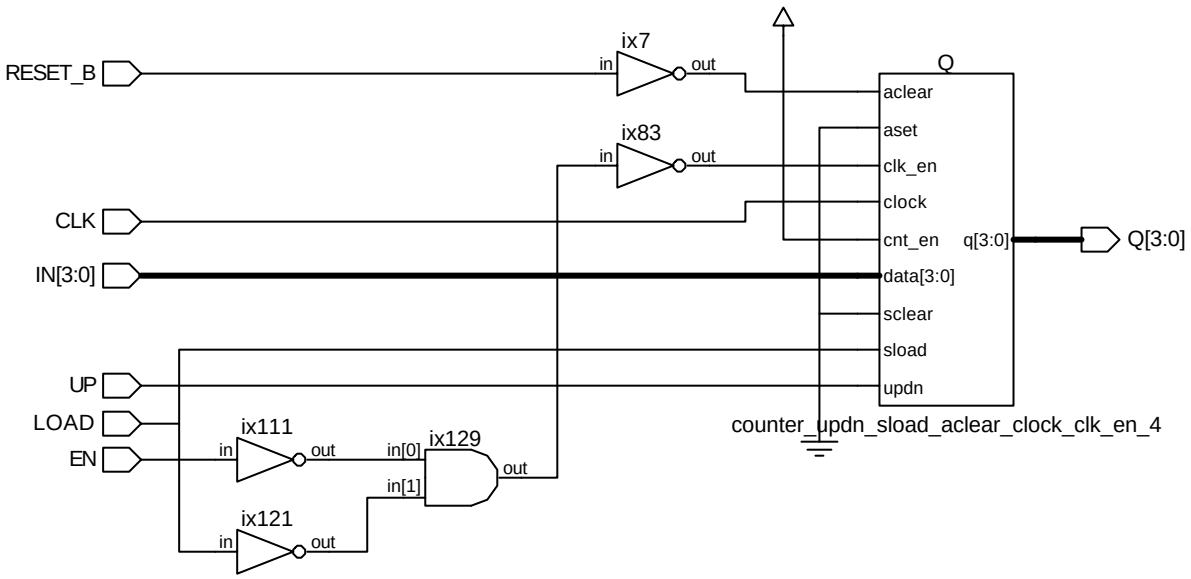

ix7
in
out
RESET_B
EN
CLK
Q
aclear
aset
clk_en
clock
cnt_en
q[3:0]
data[3:0]
sclear
sload
updn
Q[3:0]
counter_up_sclear_aclear_clock_clk_en_4
modgen_eq_0
a[3:0]
b[3:0]
=
d

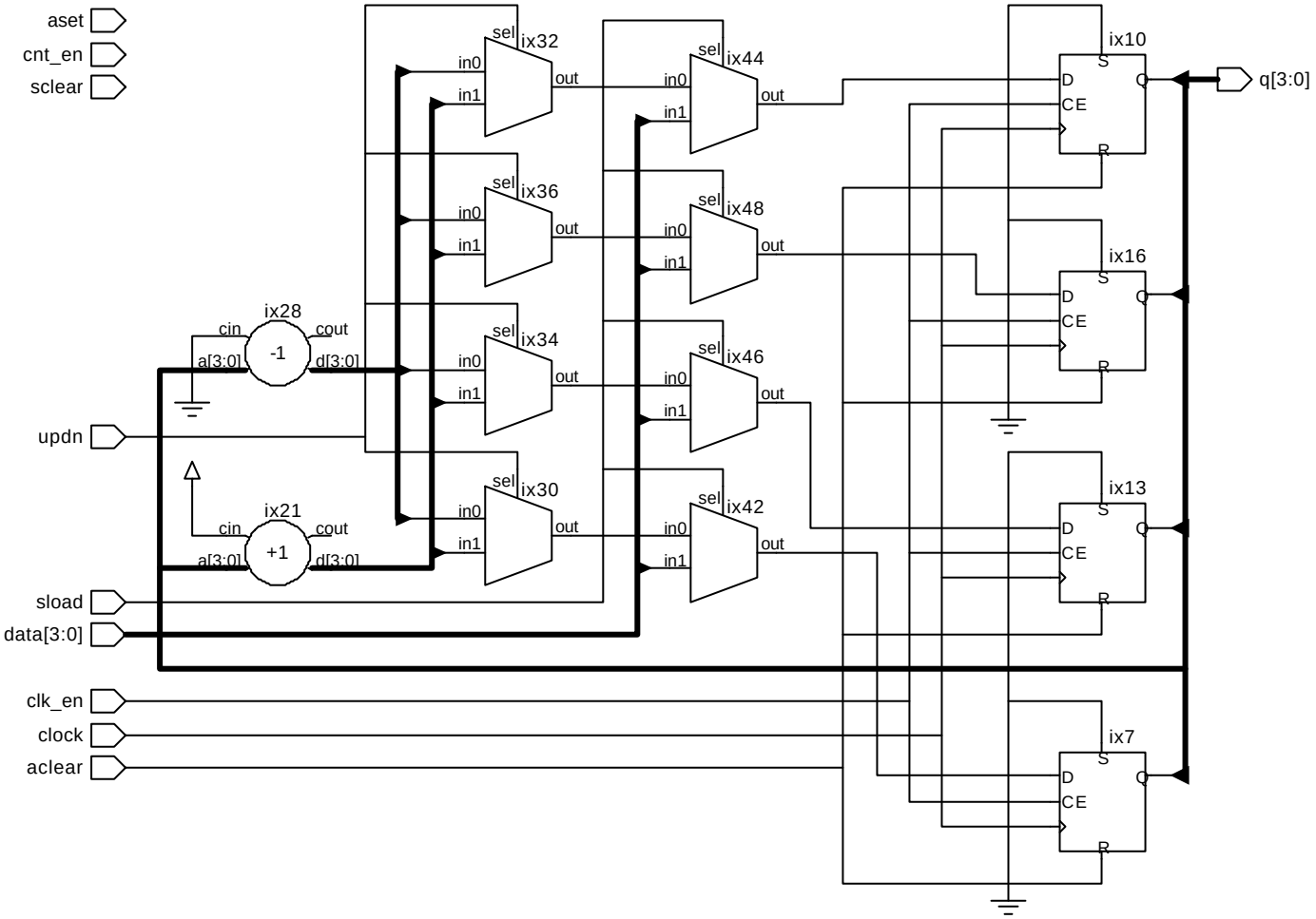

aset
cnt_en
data[3:0]
sload
updn
sclear
clk_en
clock
aclear
q[3:0]
ix13
ix10
ix16
ix7
ix31
ix29
ix33
ix27
ix25
ix21
S
D
CE
R
Q
in[0]
in[1]
out
in
cin
cout
a[3:0]
d[3:0]
+1


ix86
IN[2:0]
data[2:0]
eq[7:0]
OUT[7:0]
decoder_3


data[2:0]
ix11
a[2:0]
b[2:0]
=
d
eq[7:0]
ix7
ix21
ix19
ix17
ix15
ix13
ix9


ix83
in[1]
in[0]
out
ix25
in[1]
in[0]
out
modgen_gt_0
a[3:0]
b[3:0]
>
d
ix81
in[1]
in[0]
out
ix23
in[1]
in[0]
out
ix85
in[0]
in[1]
out
ix27
in[1]
in[0]
out
INC_OUT_inc_1
cin
cout
a[3:0]
+1
d[3:0]
ix87
in[0]
in[1]
out
ix29
in[1]
in[0]
out
TEST[3:0]
Q[3:0]
reg_Q size 4
S
D
Q
R
CLK
RESET_B
ix31
in
out


RESET_B
CLK
IN[3:0]
UP
LOAD
EN
ix7
in
out
ix83
in
out
ix111
in
out
ix121
in
out
ix129
in[0]
in[1]
out
Q
aclear
aset
clk_en
clock
cnt_en
q[3:0]
data[3:0]
sclear
sload
updn
Q[3:0]
counter_updn_sload_aclear_clock_clk_en_4


aset
cnt_en
sclear
updn
sload
data[3:0]
clk_en
clock
aclear
q[3:0]
ix32
ix36
ix34
ix30
ix44
ix48
ix46
ix42
ix28
ix21
ix10
ix16
ix13
ix7
sel
in0
in1
out
cin
cout
a[3:0]
d[3:0]
-1
+1
D
CE
S
R
Q