3e5.4 6.4 READY′ is an expression, with ′ being a unary operator. Use a name like READY_L or /READY instead.

3e5.8 6.9 Both LOW-to-HIGH and HIGH-to-LOW transitions cause positive transitions on the outputs of three gates (every second gate), and negative transitions on the other three. Thus, the total delay in either case is

$$\begin{aligned} t_p &= 3t_{\mathrm{pLH(LS00)}} + 3t_{\mathrm{pHL(LS00)}} \\ &= 3\cdot 15 + 3\cdot 15 \\ &= 90 \text{ ns} \end{aligned}$$

Since t_{pLH} and t_{pHL} for a 74LS00 are identical, the same result is obtained using a single worst-case delay of 15 ns.

3e5.12 6.13 The smallest typical delay through one ’LS86 for any set of conditions is 10 ns. Use the rule of thumb, “minimum equals one-fourth to one-third of typical,” we estimate 3 ns as the minimum delay through one gate. Therefore, the minimum delay through the four gates is estimated at 12 ns.

The above estimate is conservative, as it does not take into account the actual transitions in the conditions shown. For a LOW-to-HIGH input transition, the four gates have typical delays of 13, 10, 10, and 20 ns, a total of 53 ns, so the minimum is estimated at one-fourth of this or 13 ns. For a HIGH-to-LOW input transition, the four gates have typical delays of 20, 12, 12, and 13 ns, a total of 57 ns, so the minimum is estimated at 14 ns.

3e5.15 6.16 A decoder with active-low outputs ought to be faster, considering that this decoder structure can be implemented directly with inverting gates (which are faster than noninverting) as shown in Figures 6–35 and 6–37.

3e5.16 6.17 The worst-case ’138 output will have a transition in the same direction as the worst-case ’139 output, so we use t_{pHL} numbers for both, which is the worst combination. The delay through the ’139 is 38 ns, and from the active-low enable input of the ’138 is 32 ns, for a total delay of 70 ns. Using “worst-case” numbers for the parts and ignoring the structure of the circuit, an overly pessimistic result of 79 ns is obtained.

We can also work the problem with 74HCT parts. Worst-case delay through the ’139 is 43 ns, and from the active-low enable input of the ’138 is 42 ns, for a total delay of 85 ns. Ignoring the structure of the circuit, an overly pessimistic result of 88 ns is obtained.

We can also work the problem with 74FCT parts. Worst-case delay through the ’139 is 9 ns, and from the active-low enable input of the ’138 is 8 ns, for a total delay of 17 ns. Ignoring the structure of the circuit, a slightly pessimistic result of 18 ns is obtained.

Finally, we can work the problem with 74AHCT parts. Worst-case delay through the ’139 is 10.5 ns, and from the active-low enable input of the ’138 is 12 ns, for a total delay of 22.5 ns. Ignoring the structure of the circuit, a slightly pessimistic result of 23.5 ns is obtained.

3e5.21 6.21 Both halves of the ’139 are enabled simultaneously when EN_L is asserted. Therefore, two three-state drivers will be enabled to drive SDATA at the same time. Perhaps the designer forgot to put an extra inverter on the signal going to 1G or 2G, which would ensure that exactly one source drives SDATA at all times.

3e5.22 6.22 The total delay is the sum of the decoding delay through the 74LS139, enabling delay of a 74LS151, and delay through a 74LS20: $38 + 30 + 15 = 83$ ns.

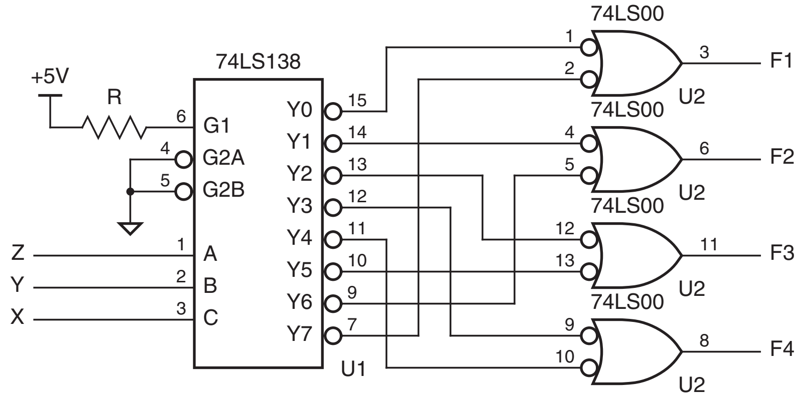
3e5.25 6.25 The worst-case delay is the sum of the delays through an ’LS280, select-to-output through an ’LS138, and through an ’LS86: $50 + 41 + 30 = 121$ ns.

3e5.30 6.30 The worst-case delay is the sum of four numbers:

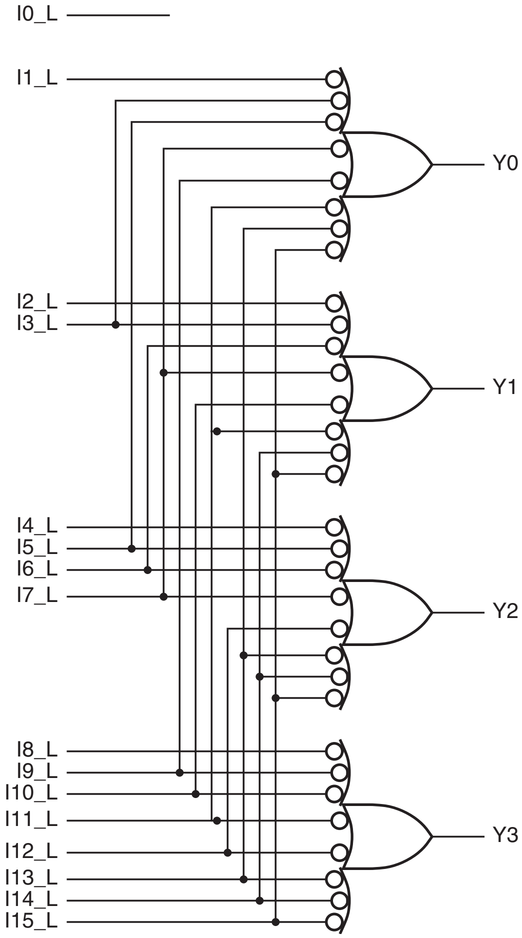
- In U1, the worst-case delay from any input to C4 (22 ns).
- In U2, the worst-case delay from C0 to C4 (22 ns).
- In U3, the worst-case delay from C0 to C4 (22 ns).
- In U4, the worst-case delay from C0 to any sum output (24 ns).

Thus, the total worst-case delay is 90 ns.

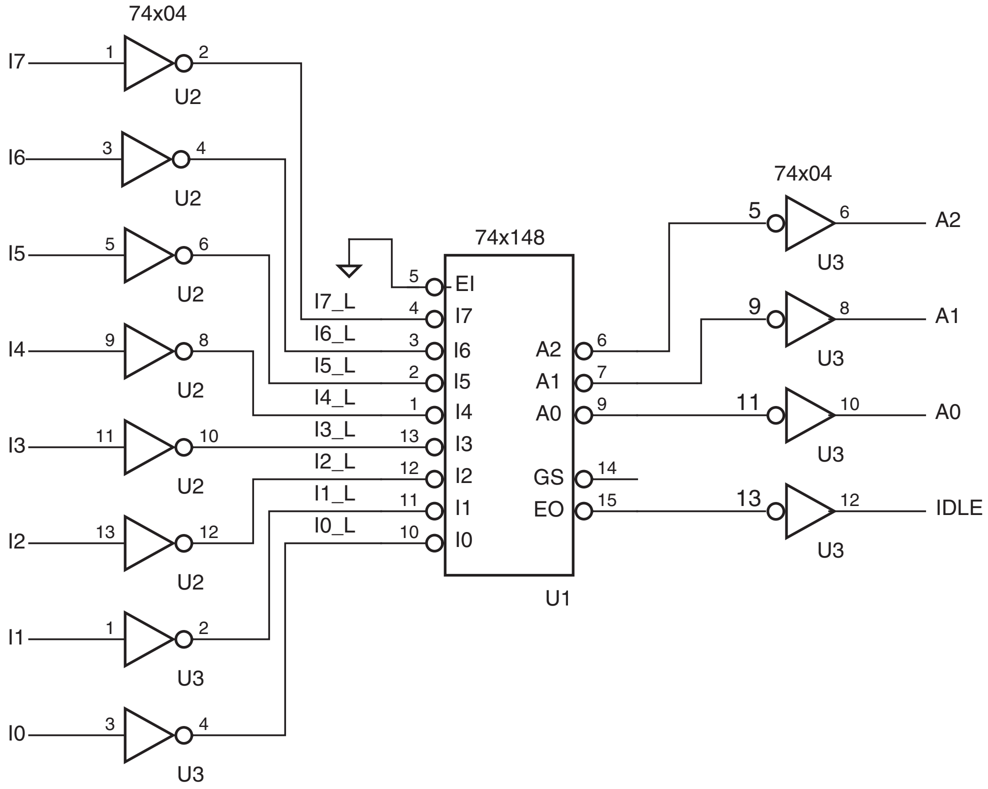
3e5.82 6.43



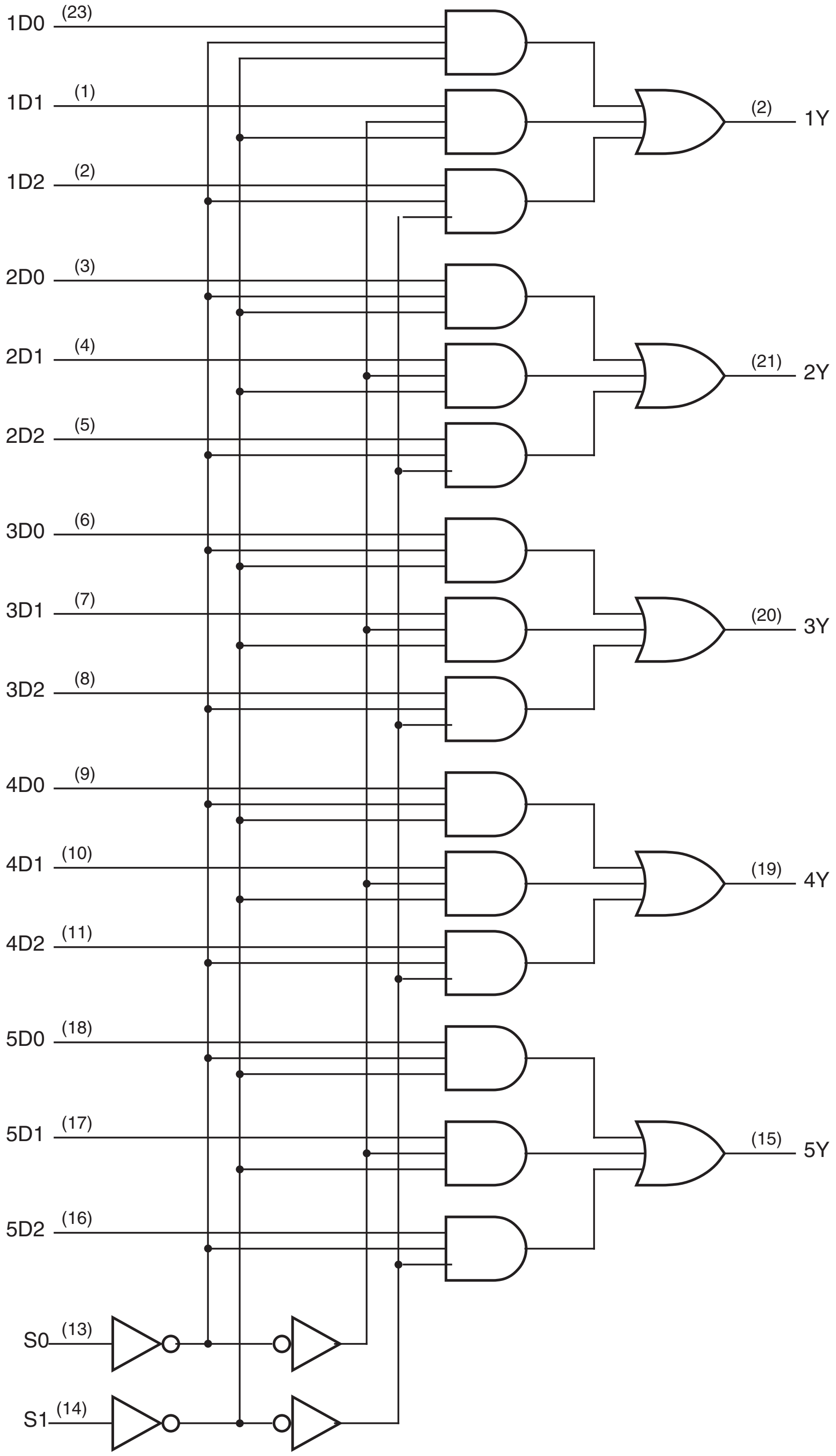
3e5.46 6.51 The inputs are active low and the outputs are active high in this design.



3e5.47 6.52

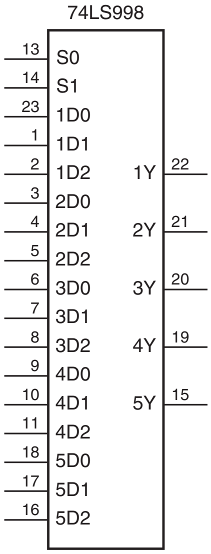

74x04
I7
1
2
U2
I6
3
4
U2
I5
5
6
U2
I4
9
8
U2
I3
11
10
U2
I2
13
12
U2
I1
1
2
U3
I0
3
4
U3
74x148
5
EI
I7_L
4
I7
I6_L
3
I6
I5_L
2
I5
I4_L
1
I4
I3_L
13
I3
I2_L
12
I2
I1_L
11
I1
I0_L
10
I0
A2
6
A1
7
A0
9
GS
14
EO
15
U1
74x04
5
6
A2
U3
9
8
A1
U3
11
10
A0
U3
13
12
IDLE
U3

3e5.54 6.63 An internal logic diagram for the multiplexer is shown below.



A truth table and pin assignment for the mux are shown below.

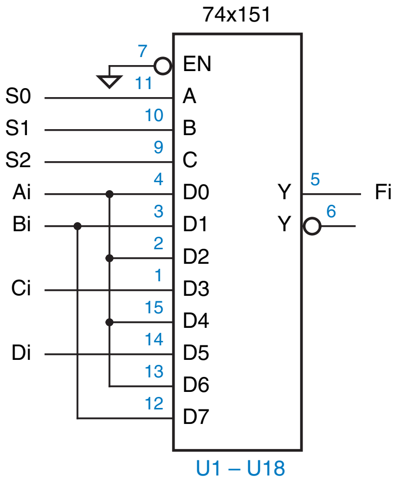
Inputs		Outputs				
S1	S0	1Y	2Y	3Y	4Y	5Y
0	0	1D0	2D0	3D0	4D0	5D0
0	1	1D1	2D1	3D1	4D1	5D1
1	0	1D2	2D2	3D2	4D2	5D2
1	1	0	0	0	0	0



The mux can be built using a single PLD, a PAL20L8 or GAL20V8; the pin assignment shown above is based on the PLD. The corresponding ABEL program, `MUX3BY5.ABL`, is shown below.

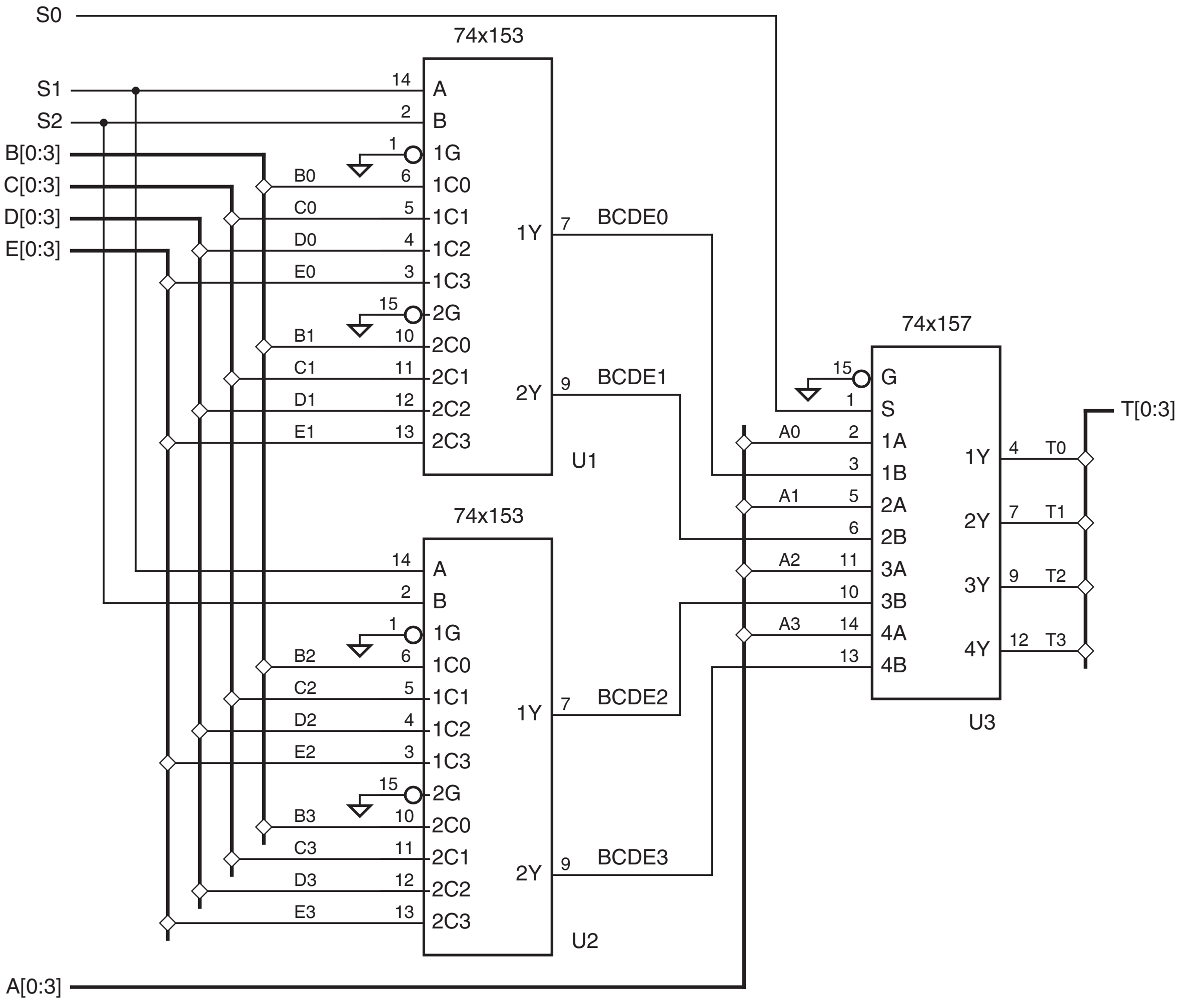
```
module Mux_3x5
title '5-Bit, 3-Input Multiplexer
J. Wakerly, Marquette University'
MUX3BY5 device 'P20L8';

" Input pins
I1D0, I1D1, I1D2                  pin 23, 1, 2;
I2D0, I2D1, I2D2                  pin 3, 4, 5;
I3D0, I3D1, I3D2                  pin 6, 7, 8;
I4D0, I4D1, I4D2                  pin 9, 10, 11;
I5D0, I5D1, I5D2                  pin 18, 17, 16;
S0, S1                            pin 13, 14;
" Output pins
Y1, Y2, Y3, Y4, Y5                pin 22, 21, 20, 19, 15;

" Set definitions
BUS0 = [I1D0,I2D0,I3D0,I4D0,I5D0];
BUS1 = [I1D1,I2D1,I3D1,I4D1,I5D1];
BUS2 = [I1D2,I2D2,I3D2,I4D2,I5D2];
OUT  = [Y1,  Y2,  Y3,  Y4,  Y5  ];

" Constants
SEL0 = ([S1,S0]==[0,0]);
SEL1 = ([S1,S0]==[0,1]);
SEL2 = ([S1,S0]==[1,0]);
IDLE = ([S1,S0]==[1,1]);

equations
OUT = SEL0 & BUS0 # SEL1 & BUS1 # SEL2 & BUS2 # IDLE & 0;

end Mux_3x5
```

3e5.60 6.65



3e5.55 6.68 This is the actual circuit of a MUX21H 2-input multiplexer cell in LSI Logic's LCA 10000 series of CMOS gate arrays. When S is 0, the output equals A; when S is 1, the output equals B.

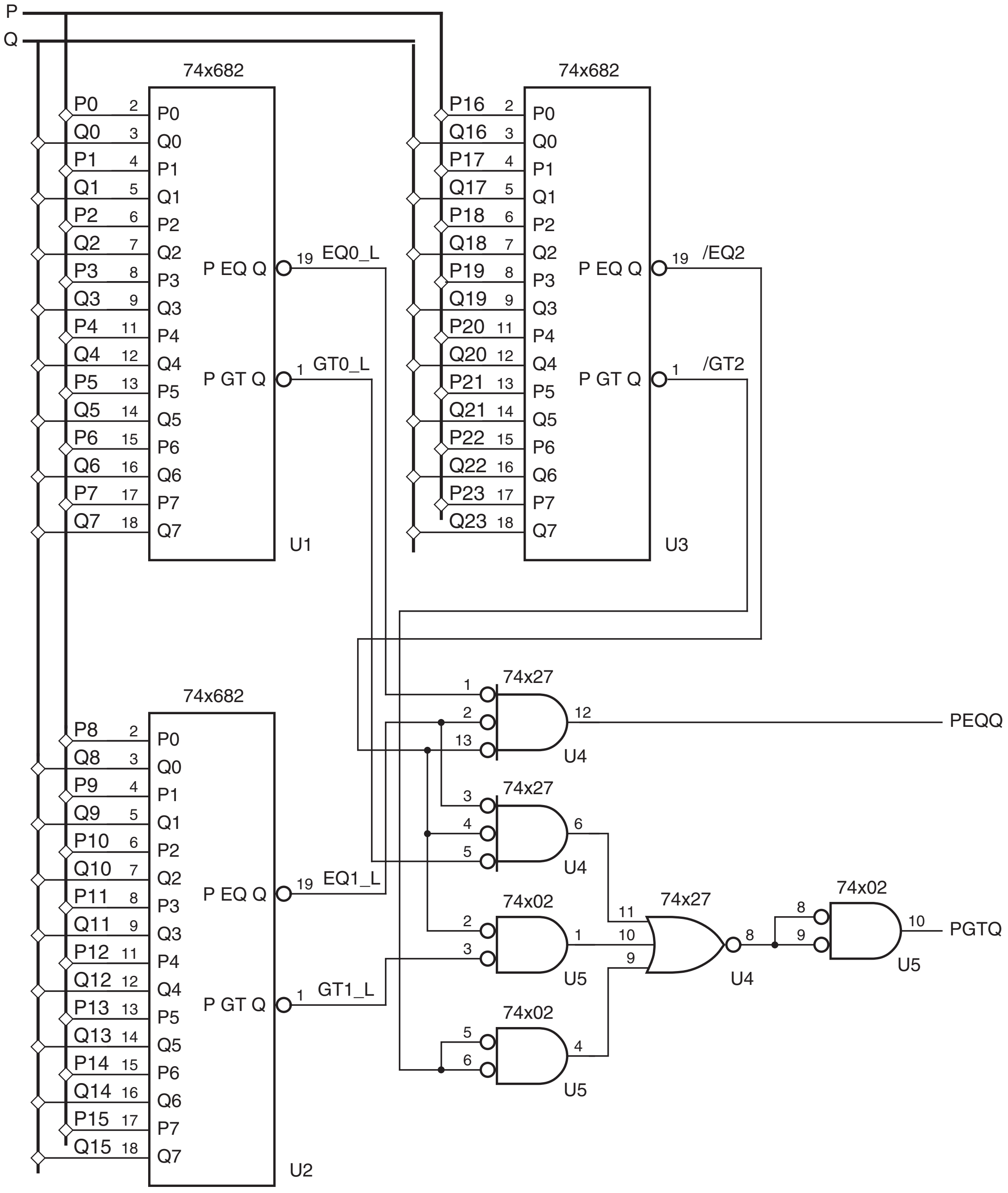
3e5.91 6.77



3e5.67 6.80 The '08 has the same pinout as the '00, but its outputs are the opposite polarity. The change in level at pin 3 of U1 is equivalent to a change at pin 4 of U2 (the input of an XOR tree), which is equivalent in turn to a change at pin 6 of U2 (the parity-generator output). Thus, the circuit simply generated and checked odd parity instead of even. The change in level at pin 6 of U1 changed the active level of the ERROR signal.

3e5.69 6.90 This problem is answered in Section 6.9.3 of the text, right before Figure 6-77, which makes it an easy answer for anyone who is paying attention.

3e5.75 6.96



3e5.93 6.97 The obvious solution is to use a 74FCT682, which has a maximum delay of 11 ns to its $\overline{\text{PEQQ}}$ output. However, there are faster parts in Table 5–3. In particular, the 74FCT151 has a delay of only 9 ns from any select input to Y or $\overline{\text{Y}}$. To take advantage of this, we use a ’138 to decode the SLOT inputs statically and apply the resulting eight signals to the data inputs of the ’151. By applying GRANT[2–0] to the select inputs of the ’151, we obtain the MATCH_L output (as well as an active-high MATCH, if we need it) in only 9 ns!

3e5.79 6.101 The function has 65 inputs, and the worst 65-input function (a 65-input parity circuit) has 2^{65-1} terms in the minimal sum-of-products expression. Our answer can't be any worse than this, but we can do better.

The expression for c_1 has 3 product terms: $c_1 = c_0 \cdot x_0 + c_0 \cdot y_0 + x_0 \cdot y_0$

The expression for c_2 is $c_2 = c_1 \cdot x_1 + c_1 \cdot y_1 + x_1 \cdot y_1$

If we substitute our previous expression for c1 in the equation above and "multiply out," we get a result with $3 + 3 + 1 = 7$ product terms. Let us assume that no further reduction is possible.

Continuing in this way, we would find that the expression for c_3 has $7 + 7 + 1 = 15$ product terms and, in general, the expression for c_i has $2^{i+1} - 1$ product terms.

Thus, the number of terms in a sum-of-products expression for c_{32} is no more than $2^{33} - 1$, fewer if minimization is possible.