TABLE 18-2
Operation of a Multiplier Using a Counter

Time	State	Counter	Product Register	*St*	*M*	*K*	Load	*Ad*	*Sh*	Done
t_0	S_0	00	000000000	0	0	0	0	0	0	0
t_1	S_0	00	000000000	1	0	0	1	0	0	0
t_2	S_1	00	000001011	0	1	0	0	1	0	0
t_3	S_2	00	011011011	0	1	0	0	0	1	0
t_4	S_1	01	001101101	0	1	0	0	1	0	0
t_5	S_2	01	100111101	0	1	0	0	0	1	0
t_6	S_1	10	010011110	0	0	0	0	0	1	0
t_7	S_1	11	001001111	0	1	1	0	1	0	0
t_8	S_2	11	100011111	0	1	1	0	0	1	0
t_9	S_3	00	010001111	0	1	0	0	0	0	1

At time t_0 the control is reset and waiting for a start signal. At time t_1, the start signal $St = 1$, and a Load signal is generated. At time t_2, $M = 1$, so an Ad signal is generated. When the next clock occurs, the output of the adder is loaded into the accumulator and the control goes to S_2. At t_3, an Sh signal is generated, so, shifting occurs and the counter is incremented at the next clock. At t_4, $M = 1$, so $Ad = 1$, and the adder output is loaded into the accumulator at the next clock. At t_5 and t_6, shifting and counting occurs. At t_7, three shifts have occurred and the counter state is 11, so $K = 1$. Because $M = 1$, addition occurs, and the control goes to S_2. At t_8, $Sh = K = 1$, so at the next clock the final shift occurs, and the counter is incremented back to state 00. At t_9, a Done signal is generated.

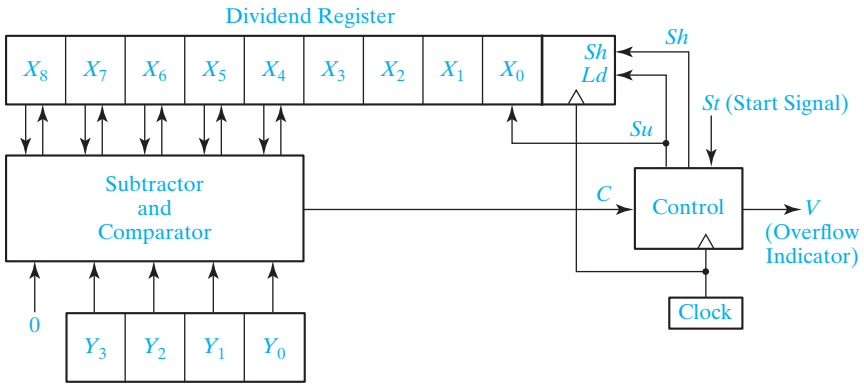
The multiplier design given here can easily be expanded to 8, 16, or more bits simply by increasing the register size and the number of bits in the counter. The add-shift control would remain unchanged.

18.3 Design of a Binary Divider

We will consider the design of a divider for positive binary numbers. As an example, we will design a circuit to divide an 8-bit dividend by a 4-bit divisor to obtain a 4-bit quotient. The following example illustrates the division process:

```
                          1010   quotient
divisor          1101 ) 10000111  dividend
                        1101
                        0111
                        0000
                         1111
(135 ÷ 13 = 10 with      1101
a remainder of 5)         0101
                          0000
                          0101   remainder
```

FIGURE 18-10
Block Diagram for Binary Divider



Just as binary multiplication can be carried out as a series of add and shift operations, division can be carried out by a series of subtraction and shift operations. To construct the divider, we will use a 9-bit dividend register and a 4-bit divisor register, as shown in Figure 18-10. During the division process, instead of shifting the divisor to the right before each subtraction as shown in the preceding example, we will shift the dividend to the left. Note that an extra bit is required on the left end of the dividend register so that a bit is not lost when the dividend is shifted left. Instead of using a separate register to store the quotient, we will enter the quotient bit-by-bit into the right end of the dividend register as the dividend is shifted left. Circuits for initially loading the dividend into the register will be added later.

The preceding division example (135 divided by 13) is now reworked, showing the location of the bits in the registers at each clock time. Initially, the dividend and divisor are entered as follows:

0	1	0	0	0	0	1	1	1

1	1	0	1

Subtraction cannot be carried out without a negative result, so we will shift before we subtract. Instead of shifting the divisor one place to the right, we will shift the dividend one place to the left:

1 0 0 0 0 1 1 1 | 0 ← Dividing line between dividend and quotient
1 1 0 1

Note that after the shift, the rightmost position in the dividend register is "empty".

Subtraction is now carried out, and the first quotient digit of 1 is stored in the unused position of the dividend register:

0 0 0 1 1 1 1 1 | 1 ← first quotient digit

Next, we shift the dividend one place to the left:

0 0 1 1 1 1 1 | 1 0
1 1 0 1

Because subtraction would yield a negative result, we shift the dividend to the left again, and the second quotient bit remains 0:

$$\begin{array}{r} 0\ 1\ 1\ 1\ 1\ 1 \\ \quad 1\ 1\ 0\ 1 \end{array} \Big\vdots \begin{array}{l} 1\ 0\ 0 \\ \\ \end{array}$$

Subtraction is now carried out, and the third quotient digit of 1 is stored in the unused position of the dividend register:

$$0\ 0\ 0\ 1\ 0\ 1 \Big\vdots 1\ 0\ 1 \longleftarrow \text{third quotient digit}$$

A final shift is carried out and the fourth quotient bit is set to 0:

$$\underbrace{0\ 0\ 1\ 0\ 1}_{\text{remainder}} \Big\vdots \underbrace{1\ 0\ 1\ 0}_{\text{quotient}}$$

The final result agrees with that obtained in the first example. Note that in the first step the leftmost 1 in the dividend is shifted left into the leftmost position (X_8) in the X register. If we did not have a place for this bit, the division operation would have failed at this step because $0000 < 1101$. However, by keeping the leftmost bit in X_8, $10000 \geq 1101$, and subtraction can occur.

If as a result of a division operation, the quotient would contain more bits than are available for storing the quotient, we say that an overflow has occurred. For the divider of Figure 18-10 an overflow would occur if the quotient is greater than 15, because only 4 bits are provided to store the quotient. It is not actually necessary to carry out the division to determine if an overflow condition exists, because an initial comparison of the dividend and divisor will tell if the quotient will be too large. For example, if we attempt to divide 135 by 7, the initial contents of the registers would be:

$$\begin{array}{r} 0\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1 \\ \quad 0\ 1\ 1\ 1 \end{array}$$

Because subtraction can be carried out with a nonnegative result, we should subtract the divisor from the dividend and enter a quotient bit of 1 in the rightmost place in the dividend register. However, we cannot do this because the rightmost place contains the least significant bit of the dividend, and entering a quotient bit here would destroy that dividend bit. Therefore, the quotient would be too large to store in the 4 bits we have allocated for it, and we have detected an overflow condition. In general, for Figure 18-10, if initially $X_8X_7X_6X_5X_4 \geq Y_3Y_2Y_1Y_0$ (i.e., if the left five bits of the dividend register exceed or equal the divisor), the quotient will be greater than 15 and an overflow occurs. Note that if $X_8X_7X_6X_5X_4 \geq Y_3Y_2Y_1Y_0$, the quotient is

$$\frac{X_8X_7X_6X_5X_4X_3X_2X_1X_0}{Y_3Y_2Y_1Y_0} \geq \frac{X_8X_7X_6X_5X_4 0000}{Y_3Y_2Y_1Y_0} = \frac{X_8X_7X_6X_5X_4 \times 16}{Y_3Y_2Y_1Y_0} \geq 16$$

The operation of the divider can be explained in terms of the block diagram of Figure 18-10. A shift signal (Sh) will shift the dividend one place to the left on the next rising clock edge. Because the subtracter is a combinational circuit, it computes

$X_8X_7X_6X_5X_4 - Y_3Y_2Y_1Y_0$, and this difference appears at the subtracter output after a propagation delay. A subtract signal (Su) will load the subtracter output into $X_8X_7X_6X_5X_4$ and set the quotient bit (the rightmost bit in the dividend register) to 1 on the next rising clock edge. To accomplish this, Su is connected to both the Ld input on the shift register and the data input on flip-flop X_0. If the divisor is greater than the five leftmost dividend bits, the comparator output is $C = 0$; otherwise, $C = 1$. The control circuit generates the required sequence of shift and subtract signals. Whenever $C = 0$, subtraction cannot occur without a negative result, so a shift signal is generated. Whenever $C = 1$, a subtract signal is generated, and the quotient bit is set to one.

Figure 18-11 shows the state graph for the control circuit. When a start signal (St) occurs, the 8-bit dividend and 4-bit divisor are loaded into the appropriate registers. Note that this assumes the divisor and dividend are available and stable during the clock cycle when $Load = 1$; they may be available before and after this clock cycle, but they must be available at least during this clock cycle. If C is 1 after the load, the upper half of the dividend is larger than the divisor and the quotient would require five or more bits. Because space is only provided for a 4-bit quotient, this condition constitutes an overflow, so the divider is stopped, and the overflow indicator is set by the V output. Normally, the initial value of C is 0, so a shift will occur first, and the control circuit will go to state S_2. Then, if $C = 1$, subtraction occurs. After the subtraction is completed, C will always be 0, so the next active clock edge will produce a shift. This process continues until four shifts have occurred, and the control is in state S_5. Then, a final subtraction occurs if $C = 1$, and no subtraction occurs if $C = 0$. No further shifting is required, and the control goes to the stop state. For this example, we will assume that when the start signal (St) occurs, it will be 1 for one clock time, and, then, it will remain 0 until the control circuit is back in state S_0. Therefore, St will always be 0 in states S_1 through S_5.

We will now design the control circuit using a one-hot assignment (see Section 15.9) to implement the state graph. One flip-flop is used for each state with $Q_0 = 1$ in S_0, $Q_1 = 1$ in S_1, $Q_2 = 1$ in S_2, etc. By inspection, the next-state and output equations are

$$
\begin{aligned}
&Q_0^+ = St'Q_0 + CQ_1 + Q_5 \qquad && Q_1^+ = StQ_0 \\
&Q_2^+ = C'Q_1 + CQ_2 && Q_3^+ = C'Q_2 + CQ_3 \\
&Q_4^+ = C'Q_3 + CQ_4 && Q_5^+ = C'Q_4 \\
&Load = St\,Q_0 && V = CQ_1 \\
&Sh = C'(Q_1 + Q_2 + Q_3 + Q_4) = C'(Q_0 + Q_5)' \\
&Su = C(Q_2 + Q_3 + Q_4 + Q_5) = C(Q_0 + Q_1)'
\end{aligned}
\tag{18-1}
$$

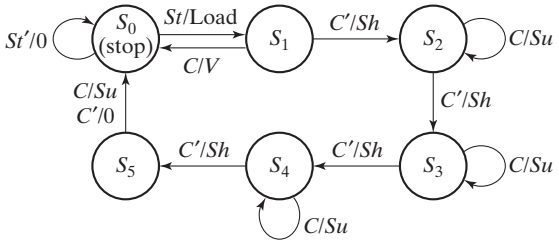
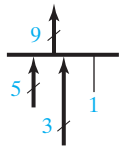


FIGURE 18-11
State Graph for Divider Control Circuit

Because there are three arrows leading into S_0, Q_0^+ has three terms. The equation for *Sh* has been simplified by noting that if the circuit is in state S_1 or S_2 or S_3 or S_4, it is not in state S_0 or S_5.

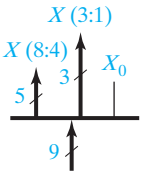
The subtracter in Figure 18-10 can be constructed using five full subtracters, as shown in Figure 18-12. Because the subtracter is a combinational circuit, whenever the numbers in the divisor and dividend registers change, these changes will propagate to the subtracter outputs. The borrow signal will propagate through the full subtracters before the subtracter output is transferred to the dividend register. If the last borrow signal (b_9) is 1, this means that the result is negative. Hence, if b_9 is 1, the divisor ($Y_3Y_2Y_1Y_0$) is greater than $X_8X_7X_6X_5X_4$, and $C = 0$. Therefore, $C = b_9'$, and a separate comparator circuit is unnecessary. Under normal operating conditions (no overflow) for this divider, we can also show that $C = d_8'$. At any subtraction step, because the divisor is only four bits, $d_8 = 1$ would allow a second subtraction without shifting. However, this can never occur because the quotient digit cannot be greater than 1. Therefore, if subtraction is possible, d_8 will always be 0 after the subtraction, so $d_8 = 0$ implies $X_8X_7X_6X_5X_4$ is greater than $Y_3Y_2Y_1Y_0$ and $C = d_8'$.

The block diagram of Figure 18-10 does not show how the dividend is initially loaded into the X register. This can be accomplished by adding a MUX at the X register inputs, as shown in Figure 18-13. This diagram uses bus notation to avoid drawing multiple wires. When several busses are merged together to form a single bus, a *bus merger* is used. For example, the symbol



means that the 5-bit subtracter output is merged with bits $X_3X_2X_1$ and a logic 1 to form a 9-bit bus. Thus, the MUX output will be $d_8d_7d_6d_5d_4X_3X_2X_1$1 when Load = 0.

Similarly, the symbol



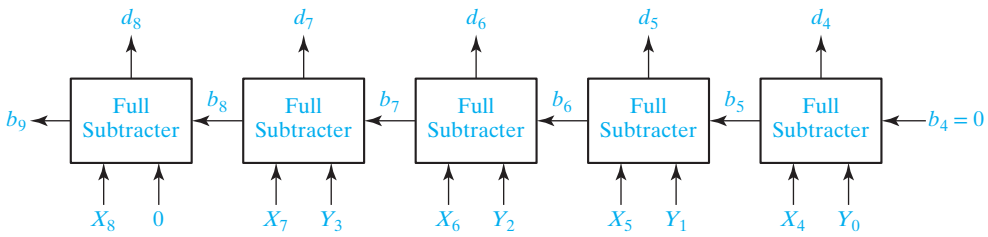


FIGURE 18-12
Logic Diagram for 5-Bit Subtracter

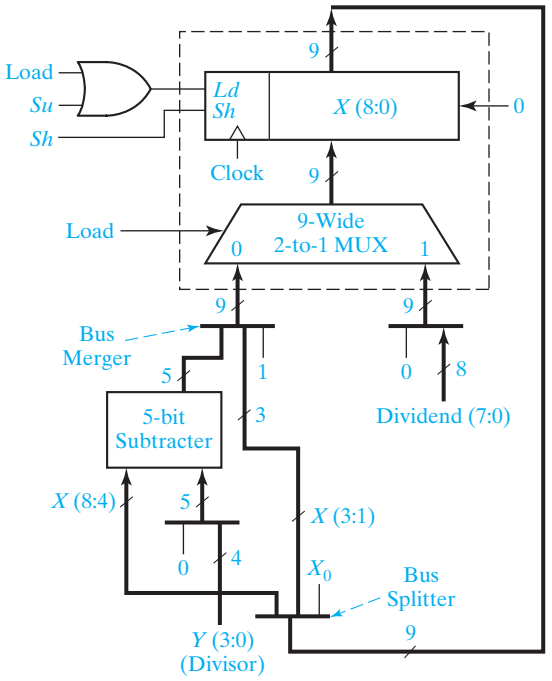


FIGURE 18-13
Block Diagram for Divider Using Bus Notation

represents a *bus splitter* that splits the 9 bits from the X register into $X_8X_7X_6X_5X_4$ and $X_3X_2X_1$; X_0 is not used. Bus mergers and splitters do not require any actual hardware; they are just a symbolic way of showing bus connections.

The X register is a left-shift register with parallel load capability, similar to the register in Figure 12-10. On the rising clock edge, it is loaded when Ld = 1 and shifted left when $Sh = 1$. Because the register must be loaded with the dividend when Load = 1 and with the subtracter output when $Su = 1$, Load and Su are ORed together and connected to the Ld input. The MUX selects the dividend (preceded by a 0) when Load = 1. When Load = 0, it selects the bus merger output which consists of the subtracter output, $X_3X_2X_1$, and a logic 1. When $Su = 1$ and the clock rises, this MUX output is loaded into X. The net result is that $X_8X_7X_6X_5X_4$ gets the subtracter output, $X_3X_2X_1$ is unchanged, and X_0 is set to 1.

Figure 18-14 shows an alternative version of the divider. The primary difference is the use of a 4-bit subtracter rather than a 5-bit subtracter. The 4-bit subtracter is shown in Figure 18-12 with the leftmost full subtracter deleted. It can be shown that the 4 least significant output bits from the 5-bit subtracter of Figure 18-13 do not depend upon X_8. (See Problem 18.32.) Since the most significant bit of the 5-bit subtracter is discarded by the following shift of the X register, this bit is not needed. However, now the borrow from the 4-bit subtracter, b_8, is not sufficient to determine whether a subtract operation should be done. The state graph of Figure 18-11 still applies, but now C depends on both X_8 and b_8.

FIGURE 18-14
Alternative Divider

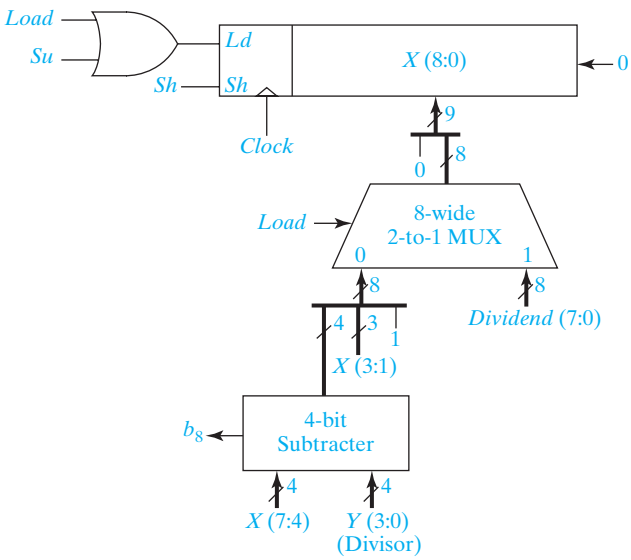
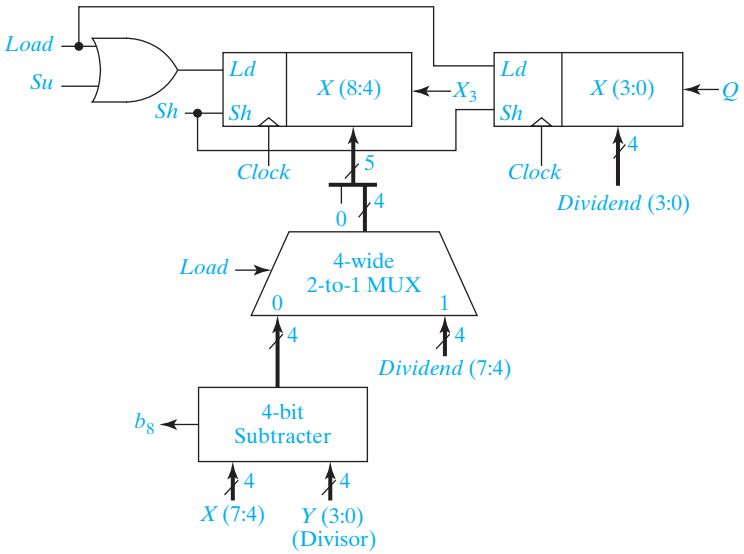


Figure 18-15 shows a second alternative divider. In this version only the upper part of the X register is loaded when a subtract operation is required. Then, on the following shift operation, a quotient bit of 1, $Q = 1$, is shifted into X_0. If no subtract operation is required, a quotient bit of 0, $Q = 0$, is shifted into X_0. This simplifies the circuit since only a 4-wide multiplexer is needed. This circuit is analyzed in Problem 18.33.

FIGURE 18-15
Another
Alternative Divider

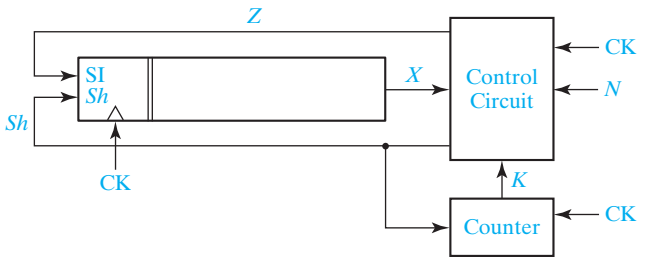


Programmed Exercise 18.1

Cover the answers with a sheet of paper and slide it down as you check your answers. Write your answer in the space provided before looking at the correct answers.

This exercise concerns the design of a circuit which forms the 2's complement of a 16-bit binary number. The circuit consists of three main components—a 16-bit shift register which initially holds the number to be complemented, a control circuit, and a counter which counts the number of shifts. The control circuit processes the number in the shift register one bit at a time and stores the 2's complement back in the shift register. Draw a block diagram of the circuit. Show the necessary inputs and outputs for the control circuit including a start signal (N) which is used to initiate the 2's complement operation.

Answer

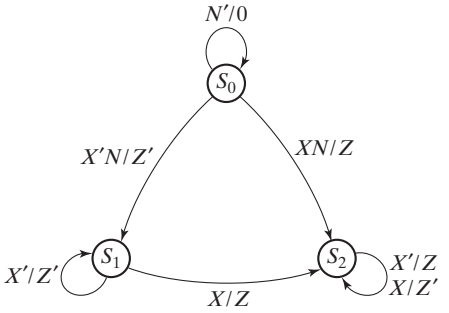


State a rule for forming the 2's complement which is appropriate for use with the preceding block diagram.

Answer Starting with the least significant bit, complement all of the bits to the left of the first 1.

Draw a state graph for the control circuit (three states) which implements the preceding rule. The 2's complement operation should be initiated when $N = 1$. (Assume that N will be 1 for only one clock time.) When drawing your graph, do not include any provision for stopping the circuit. (In the next step you will be asked to add the signal K to your state graph so that the circuit will stop after 16 shifts.) Explain the meaning of each state in your graph.

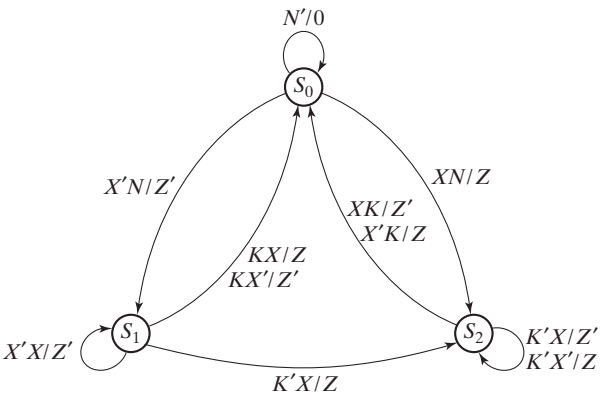
Answer



The counter will generate a completion signal (K) when it reaches state 15. Modify your state graph so that when $K = 1$, the circuit will complete the 2's complement operation and return to the initial state. Also, add the *Sh* output in the appropriate places.

Answer Check the input labels on all arrows leaving each state of your graph. Make sure that two of the labels on arrows leaving a given state cannot have the value 1 at the same time. Make any necessary corrections to your graph, and then check your final answer.

Final Answer

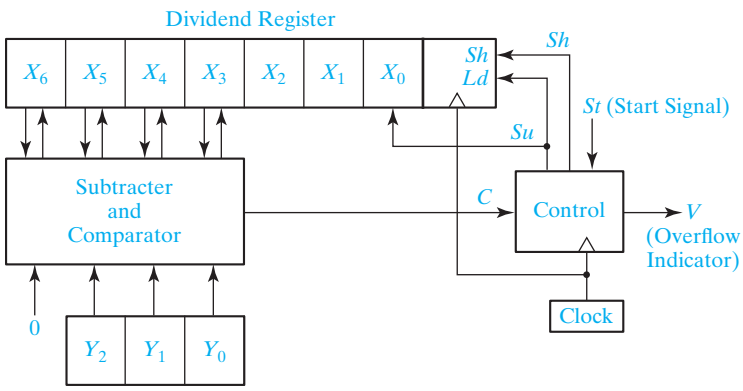


(*Note: Sh* should be added to the graph everywhere Z or Z' appears.)

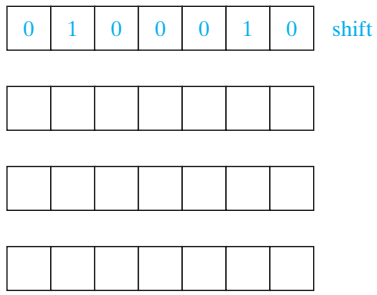
Programmed Exercise 18.2

This exercise concerns the design of a binary divider to divide a 6-bit number by a 3-bit number to find a 3-bit quotient. The right 3 bits of the dividend register should be used to store the quotient. Draw a block diagram for the divider. Omit the signals required to initially load the dividend register and assume the dividend is already loaded.

Answer



If the contents of the dividend register is initially 0100010 and the divisor is 110, show the contents of the dividend register after each of the first three rising clock edges. Also, indicate whether a shift or a subtraction should occur next.



Answer

0	1	0	0	0	1	0	shift
1	0	0	0	1	0	0	subtract
0	0	1	0	1	0	1	shift
0	1	0	1	0	1	0	shift

Now, show the remaining steps in the computation and check your answer by converting to decimal.

Answer

1	0	1	0	1	0	0	subtract
0	1	0	0	1	0	1	(finished)

If the dividend register initially contained 0011001 and the divisor is 010, can division take place? Explain.

Answer No. Because 011 > 010, subtraction should occur first, but there is no place to store the quotient bit. In other words, the quotient would be greater than three bits, so an overflow would occur.

Draw a state graph for the divider which will produce the necessary sequence of *Su* and *Sh* signals. Assume that the comparator output is $C = 1$ if the upper four bits of the dividend register is greater than the divisor. Include a stop state in your graph which is different than the reset state. Assume that the start signal (*St*) will remain 1 until the division is completed. The circuit should go to the stop state when division is complete or when an overflow is detected. The circuit should then reset when $St = 0$.