has been to present and illustrate the general principles of digital design and to consider various tools that are useful in the design process. These design principles and tools form a solid foundation on which to build experience and to exercise creativity.

SUPPLEMENTARY READING (see Bibliography)

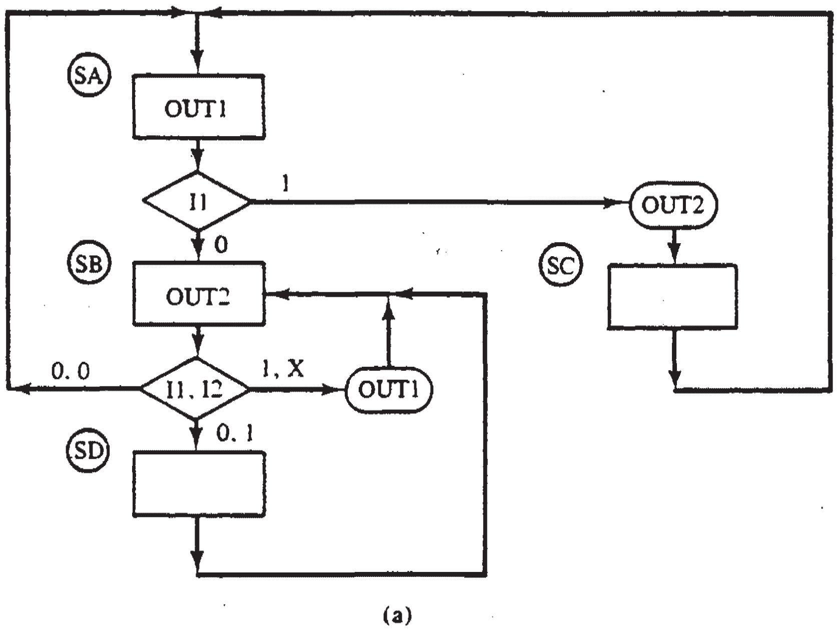
[Clare 73], [Fletcher 80], [Kline 83], [Mano 79], [Mano 84], [Peatman 80], [Prosser 87], [Wiatrowski 80]

PROBLEMS

7.1. Explain the general model for a digital circuit design shown in Fig. 7.1.

7.2. The digital design process can be divided into three major phases: the preliminary design phase, the refinement phase, and the realization phase. At the conclusion of each phase, what are the expected end products in terms of the controller and the controlled circuit elements?

7.3. Given the ASM chart of Fig. 7.39(a), complete the corresponding timing diagram of Fig. 7.39(b).



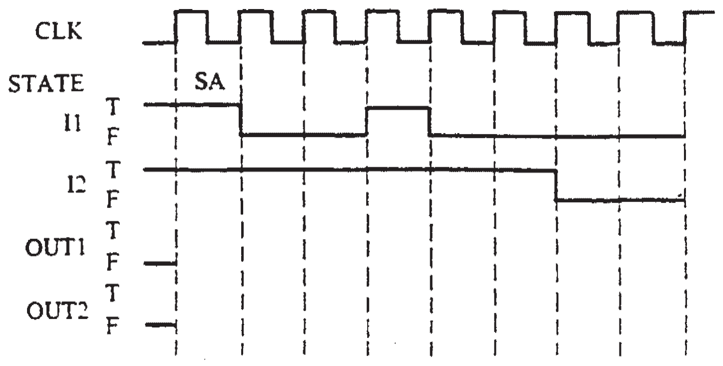
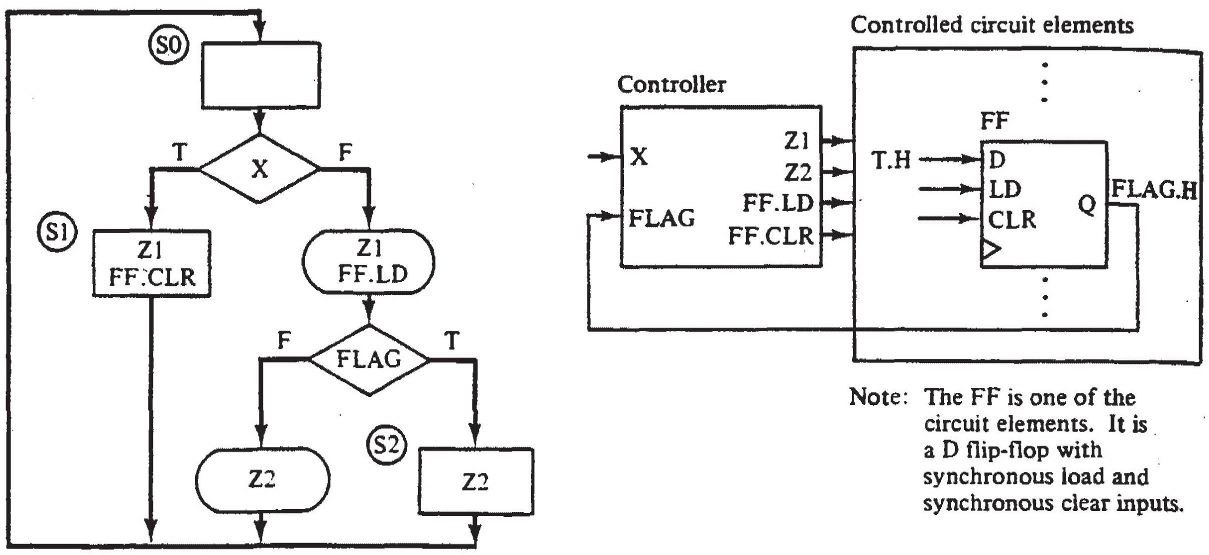


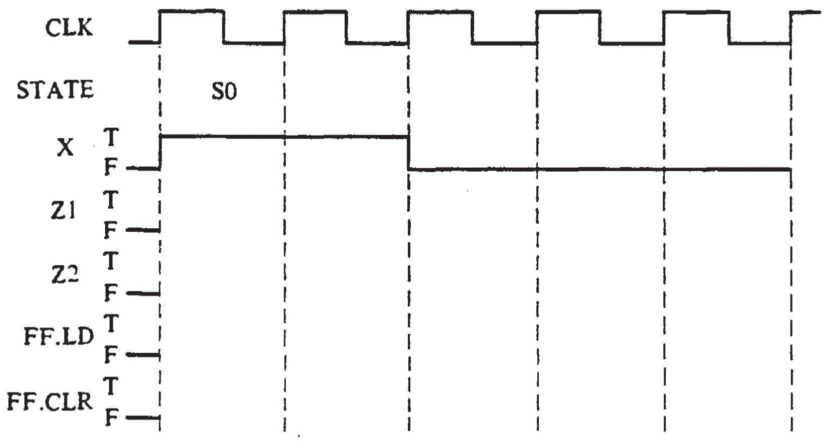
Figure 7.39 ASM chart and timing diagram for Problem 7.3.

7.4. Given the ASM chart and block diagram of Fig. 7.40(a), complete the corresponding timing diagram of Fig. 7.40(b).



S0
X
T
F
S1
Z1
FF.CLR
Z1
FF.LD
FLAG
F
T
S2
Z2
Z2
Controlled circuit elements
Controller
X
FLAG
Z1
Z2
FF.LD
FF.CLR
FF
T.H
D
LD
CLR
Q
FLAG.H
Note: The FF is one of the circuit elements. It is a D flip-flop with synchronous load and synchronous clear inputs.

(a)



(b)

Figure 7.40 ASM chart, block diagram, and timing diagram for Problem 7.4.

7.5. Given the timing diagram of Fig. 7.41, reconstruct the ASM chart that corresponds to it.

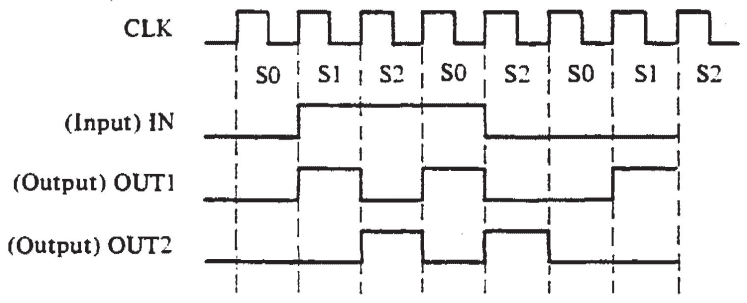


Figure 7.41 Timing diagram for Problem 7.5.

7.6. Given the timing diagram of Fig. 7.42, reconstruct the ASM chart that corresponds to it.

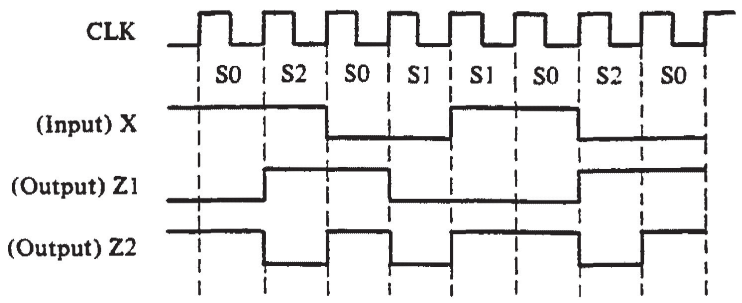


Figure 7.42 Timing diagram for Problem 7.6.

7.7. For the ASM chart specified in Fig. 7.40(a),

(a) Make a block diagram design (similar to the one shown in Fig. 7.3) of the corresponding state generator, specifying the inputs, outputs, and the state flip-flops (D type).

(b) Construct the next-state table for the state generator, given the following state code assignment:

	C1	C0
S0	1	0
S1	0	0
S2	0	1

(c) Realize the controller by the traditional method. Use D flip-flops, and draw a detailed circuit diagram of the controller.

7.8. For the ASM chart of Fig. 7.43,

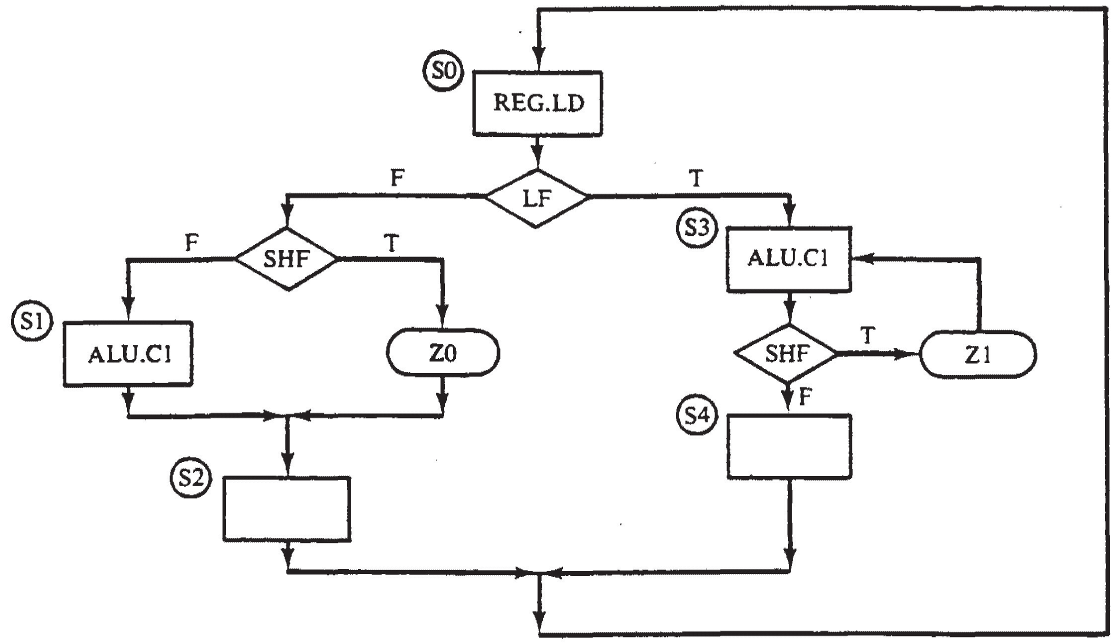


Figure 7.43 ASM chart for Problem 7.8.

(a) Make a block diagram of the corresponding controller, specifying the controller inputs and outputs.

(b) Make a block diagram design (similar to the one shown in Fig. 7.3) of the corresponding state generator, specifying the inputs, outputs, and the state flip-flops (D type).

(c) Determine the next-state table for the state generator, given the following state code assignment:

	C2	C1	C0
S0	0	0	0
S1	0	0	1
S2	0	1	0
S3	0	1	1
S4	1	0	0

(d) Realize the controller by the traditional method. Use D flip-flops, and draw a detailed circuit diagram of the controller.

7.9. Given the ASM chart of Fig. 7.4,

(a) Specify the ASM outputs as a function of the state code and the ASM inputs. In other words, complete the following truth table:

C1	C0	IN. BIT	BUF. FULL	COUNT. EN	REG. LD	OUT. FLAG
0	0	0	0			
0	0	0	1			
0	0	1	0			
		·				
		·				
		·				
1	1	1	1			

(b) Using gates, realize the ASM outputs directly as a function of the state code and the ASM inputs. Compare your realization with the one shown in Fig. 7.6, which has a circuit for decoding the state code.

7.10. Given the ASM chart of Fig. 7.40(a) and the code assignment of Problem 7.7(b),

(a) Specify, in the form of a truth table, the ASM outputs (Z1, Z2, FF.LD, and FF.CLR) as a function of the state code (C1, C0) and the ASM inputs (X, FLAG).

(b) Using gates, realize the ASM outputs directly as a function of the state code and the ASM inputs. Compare this realization with your solution to Problem 7.7.

7.11. For the ASM chart of Fig. 7.40(a) and the code assignment of Problem 7.7(b), realize the corresponding controller by using a PAL16R4. Compare it with your solution to Problem 7.7.

7.12. For the ASM chart of Fig. 7.43 and the code assignment of Problem 7.8, realize the corresponding controller by using a PAL16R4. Compare it with your solution to Problem 7.8.

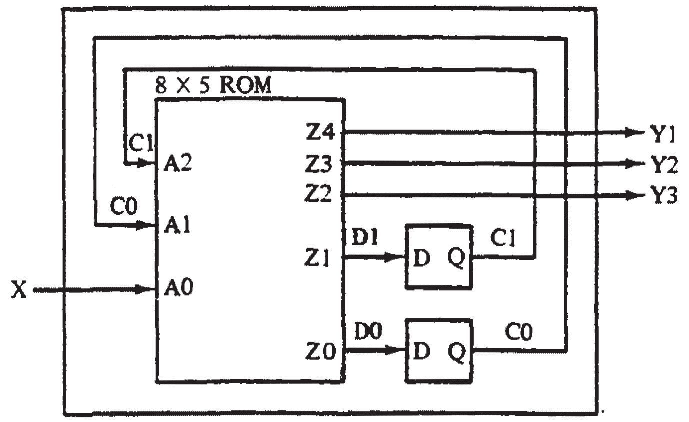
7.13. For the ASM chart of Fig. 7.40(a) and the code assignment of Problem 7.7(b), realize the corresponding controller by using the ROM method and D flip-flops. That is,

(a) Make a block diagram design of the controller, specifying the state flip-flops and all the appropriate connections to the inputs and outputs of the ROM. In other words, make a block diagram design similar to the one shown in Fig. 7.8, but without the actual ROM contents.

(b) Specify the ROM contents in hexadecimal.

7.14. Repeat Problem 7.13 for the ASM chart of Fig. 7.43 and the code assignment of Problem 7.8(c).

7.15. Figure 7.44 shows a block diagram design of a controller based on the ROM method and with D flip-flops. The ROM contents are also given. Derive the corresponding ASM chart.



Contents of the ROM

Location	Contents
0	11H
1	11H
2	16H
3	0FH
4	00H
5	00H
6	04H
7	04H

Figure 7.44 Controller block diagram and ROM contents for Problem 7.15.

7.16. For the ASM chart of Fig. 7.39(a),

(a) Draw a block diagram of the corresponding controller, specifying the controller inputs and outputs.

(b) Make a block diagram design (similar to the one of Fig. 7.10) for the corresponding state generator, specifying the inputs, outputs, and the state flip-flops (J-K type).

(c) Determine the next-state table for the state generator, given the following state code assignment:

	C1	C0
SA	0	0
SB	0	1
SC	1	0
SD	1	1

(d) Realize the controller by the traditional method. Use J-K flip-flops, and draw a detailed circuit diagram of the controller.

7.17. For the ASM chart of Fig. 7.39(a) and the state code assignment of Problem 7.16(c), realize the corresponding controller by using the ROM method and J-K flip-flops: that is,

(a) Make a block diagram design of the controller, specifying the state flip-flops and all the appropriate connections to the ROM inputs and outputs.

(b) Specify the ROM contents in hexadecimal.

7.18. Convert the Mealy state graph of Fig. 7.45 into an equivalent ASM chart. The inputs are X1 and X2, and the outputs are Z1 and Z2.

7.19. Convert the Moore state graph of Fig. 7.46 into an equivalent ASM chart. The inputs are X1 and X2, and the outputs are Z1, Z2, and Z3.

7.20. Assume that the design of the dynamic RAM controller circuit described in Sec. 7.8.1 is to be modified. In addition to the already specified functions, if the RD and WR signals are both true, then the function of the circuit is to be described by the timing diagram of Fig. 7.47. Specifically,

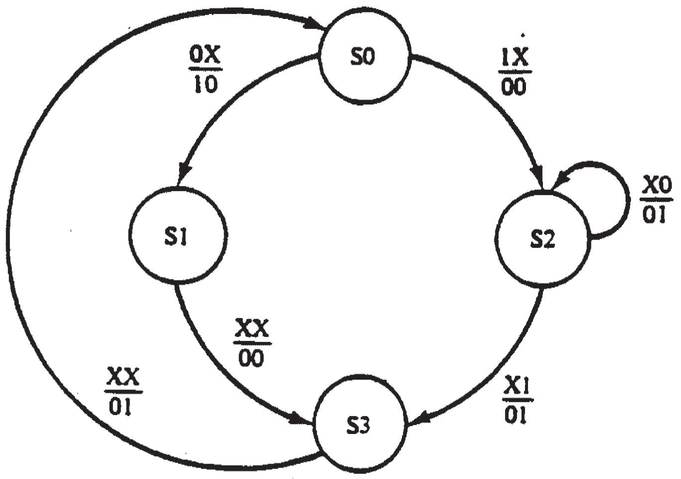


Figure 7.45 Mealy state graph for Problem 7.18.

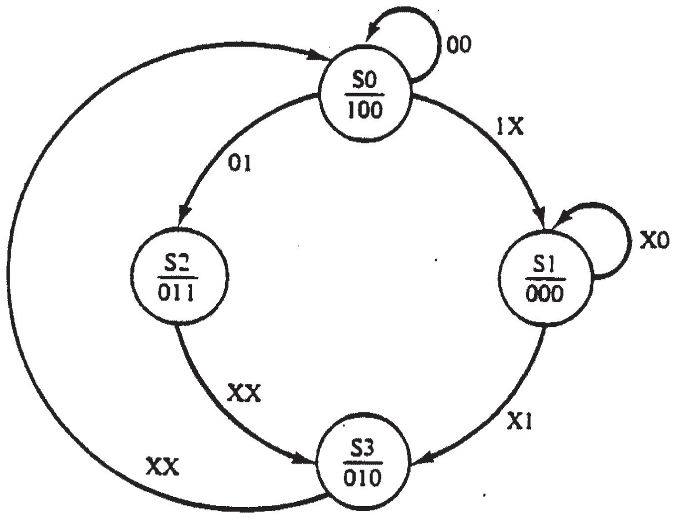


Figure 7.46 Moore state graph for Problem 7.19.

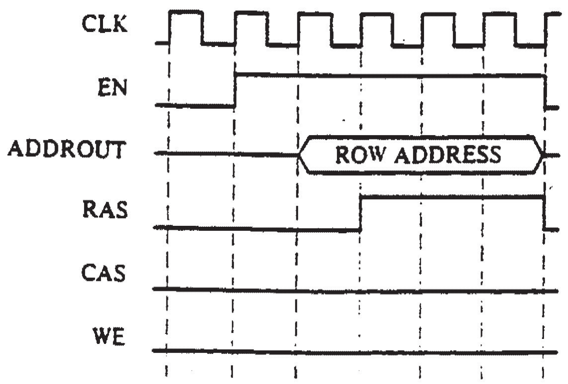


Figure 7.47 Timing diagram for Problem 7.20.

(a) Modify the ASM chart of Fig. 7.20 to include this function.
(b) Realize this modified ASM chart by using the ROM method and D flip-flops.

7.21. An 8-bit parallel-to-serial converter circuit, the block diagram of which is shown in Fig. 7.48, is to be designed and realized. This circuit remains in an idle state as long as the START input is false. But when this START input becomes true, the 8-bit data BYTE is loaded into the shift register and the right-shifting of the data begins. After the 8 bits are shifted out, the circuit returns to the idle state.

(a) The major circuit elements for the parallel-to-serial converter circuit are given in Fig. 7.48. Make any reasonable assumptions for the circuit elements that are required and derive the ASM chart for the controller. (*Hint:* A two-state ASM chart can be derived for this circuit.)
(b) Realize the circuit elements with commercially available chips.
(c) Realize the ASM chart by using any of the methods presented in this chapter.

7.22. A lock to a certain safe can be opened with a correct combination or with a key. If the combination feature is used, then the lock must be supplied with the correct combination within three attempts. Otherwise an alarm will sound. Specifications for the lock circuit are as follows:

1. After the first unsuccessful attempt, output message 1 (MSG1).
2. After the second unsuccessful attempt,
   (a) output message 2 (MSG2), and
   (b) wait for 10 seconds before being ready to be tried again (READY).
3. After the third successive unsuccessful attempt, sound an alarm.
4. When the safe is opened, reset to allow three more tries.
5. When the key is used, stop the alarm and reset to allow three more tries.

Your part in this problem is to design the module of Fig. 7.49. Make a detailed design of this module, using a counter for the major circuit element. Realize the controller with the ROM method and D flip-flops.

7.23. Shown in Fig. 7.50(a) are the circuit elements for a digital circuit that functions as follows:

If OP = 00, then REGB ← REGA; followed by REGA ← IN + 1.
If OP = 01, then REGB ← 0; REGA ← REGA + 1.
If OP = 10, REGA ← IN; REGB ← IN; followed by REGB ← REGB + 1.
If OP = 11, then the contents of REGA are doubled; REGB ← REGB + 1.

Notation: An arrow (←) designates to load the data at the next active clock transition. For example, REGA ← IN + 1 designates to add 1 to the value of IN and load the sum into REGA at the next active clock transition. In the function statements, more than one operation can be performed within a single state unless otherwise stated, as designated by "followed by."

The flowchart for the controller is given in Fig. 7.50(b). Derive the corresponding ASM chart for it. Optimize it by using the least number of states. In other words, if more than one operation can be performed within a single state, then do so. Also, make use of conditional outputs.

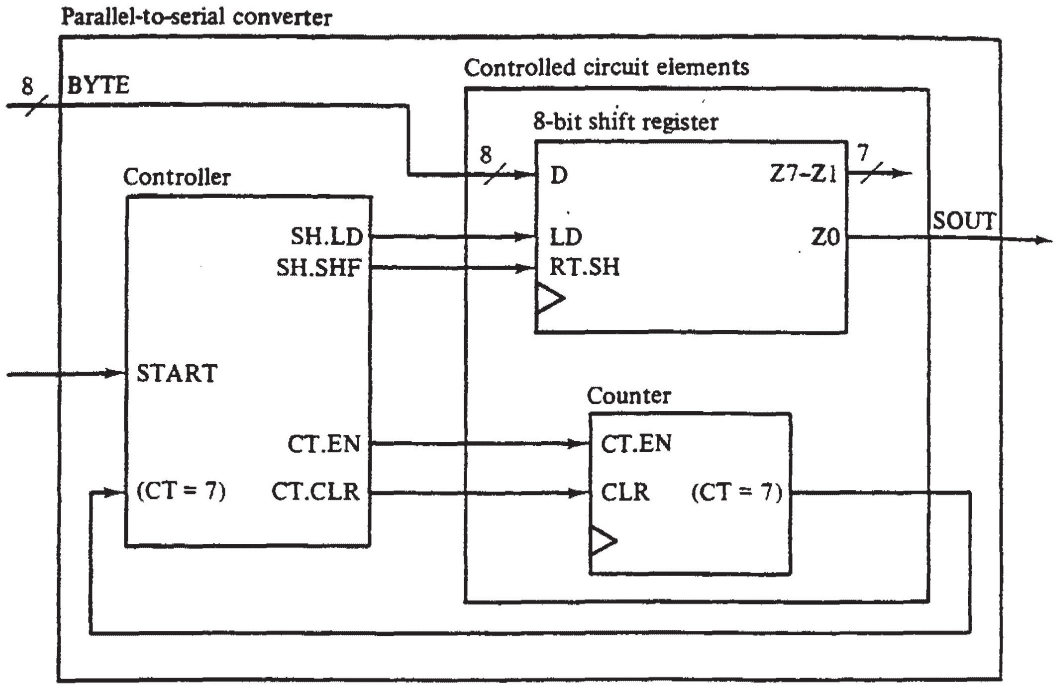
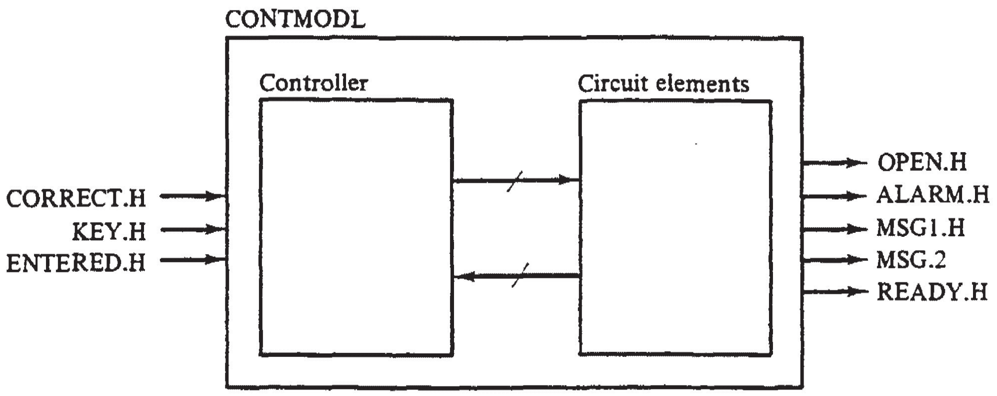


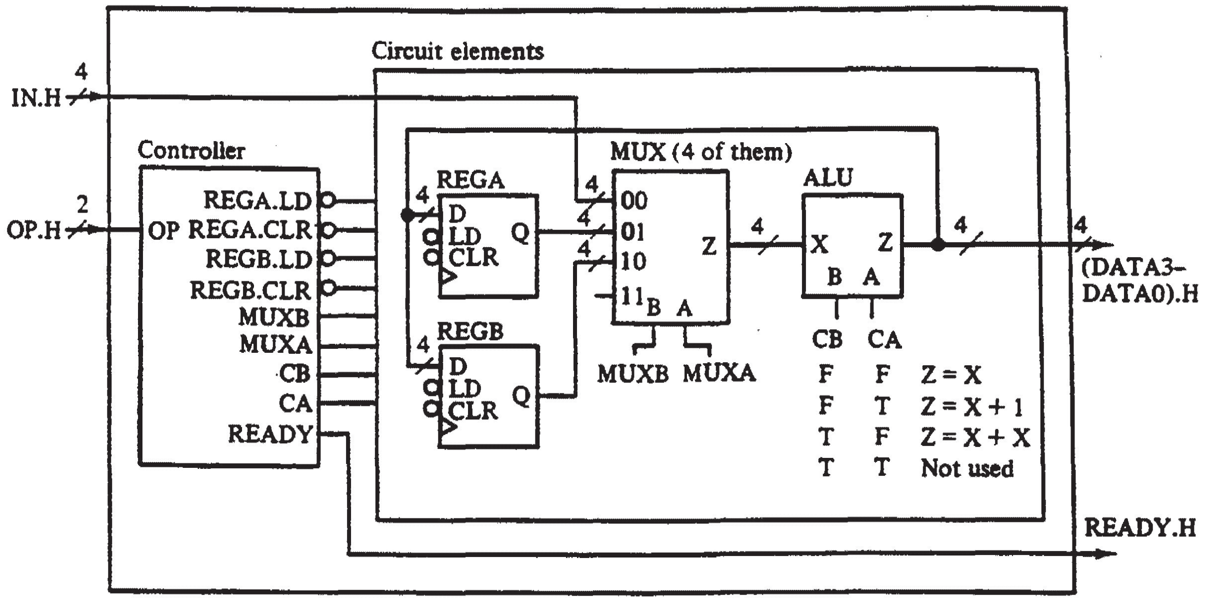
Figure 7.48 Parallel-to-serial converter circuit for Problem 7.21.



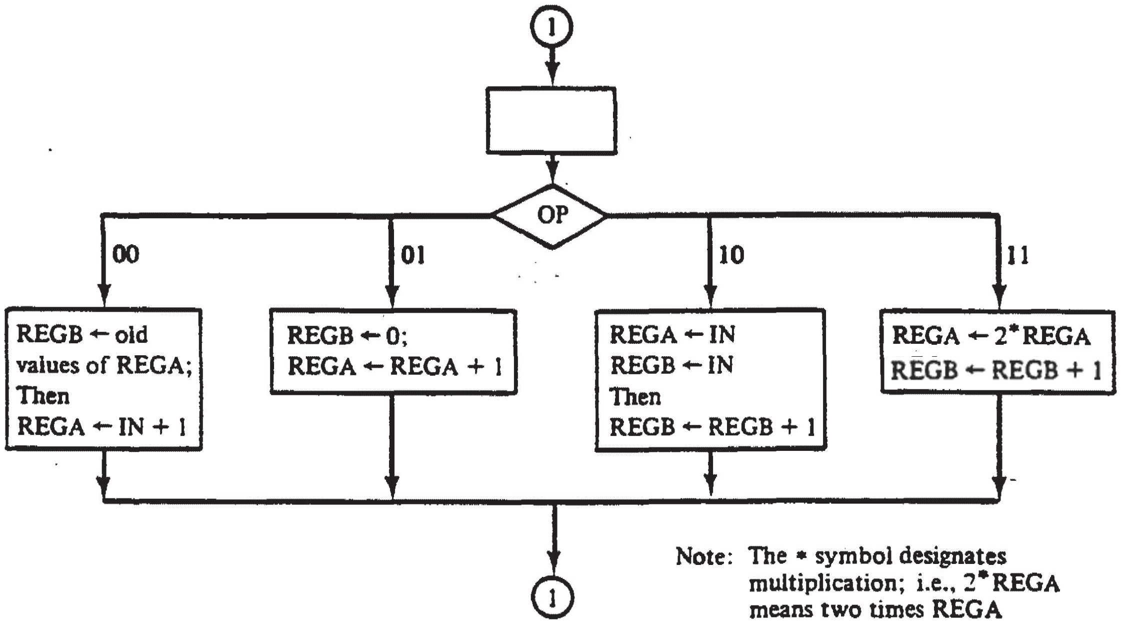
Signal specifications

CORRECT (from another comparator module): T = correct combination;
F = incorrect combination
KEY: T = a key is used to open the safe; F = no key is used
ENTERED: T = another combination is entered;
F = another attempt has not been made
OPEN: an output signal for another module to open the safe
ALARM: an output signal for another module to sound the alarm
MSG1: an output signal for another module to output message 1
MSG2: an output signal for another module to output message 2
READY: an output signal for another module to allow another attempt

Figure 7.49 Module for Problem 7.22.



(a)


1
OP
00
01
10
11
REGB ← old values of REGA; Then REGA ← IN + 1
REGB ← 0; REGA ← REGA + 1
REGA ← IN REGB ← IN Then REGB ← REGB + 1
REGA ← 2*REGA REGB ← REGB + 1
1
Note: The * symbol designates multiplication; i.e., 2*REGA means two times REGA

(b)

Figure 7.50 Circuit elements and flowchart for Problem 7.23.

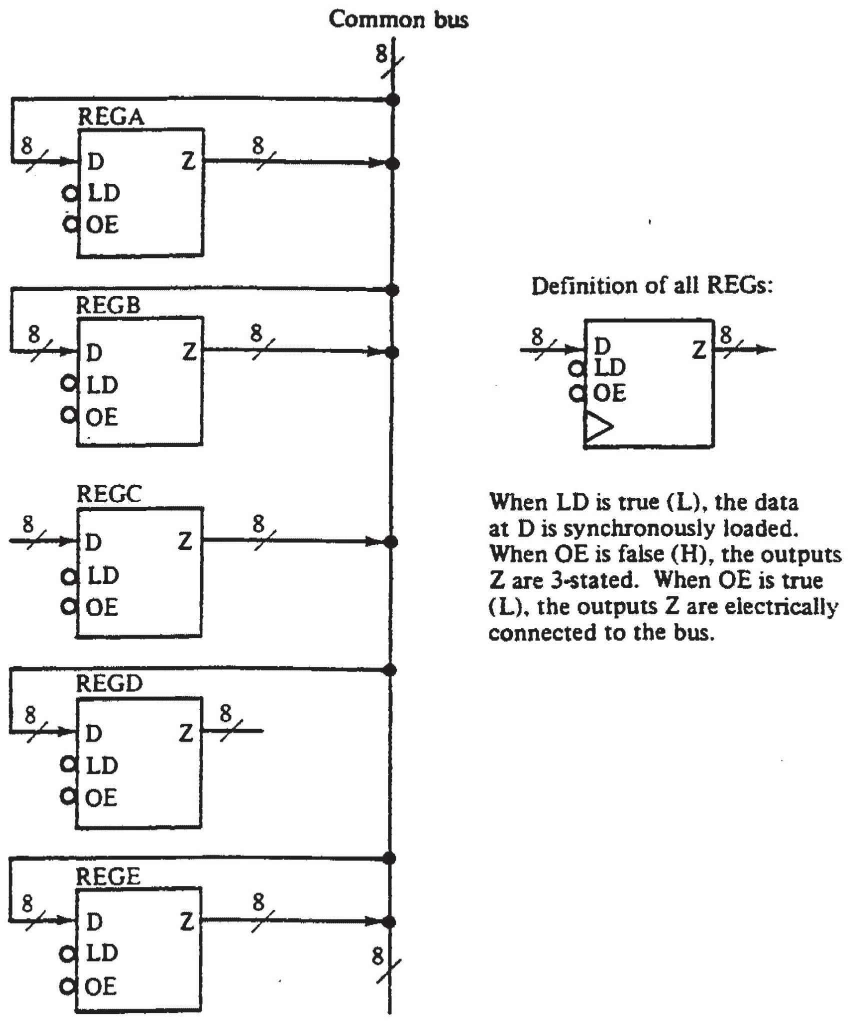
7.24. Figure 7.51(a) shows circuit elements for a digital circuit that is to be designed. They are organized in a common bus structure.

(a) It is important not to have more than one set of Z outputs connected to the common bus at any one time. Why?

(b) With the shown connections of circuit elements, can we perform the following operations within a single state?

$$\text{REGA} \leftarrow \text{REGD} \quad \text{and} \quad \text{REGC} \leftarrow \text{REGA}$$

Explain your answer.



Common bus
8
REGA
D
Z
LD
OE
REGB
REGC
REGD
REGE
Definition of all REGs:
When LD is true (L), the data at D is synchronously loaded. When OE is false (H), the outputs Z are 3-stated. When OE is true (L), the outputs Z are electrically connected to the bus.

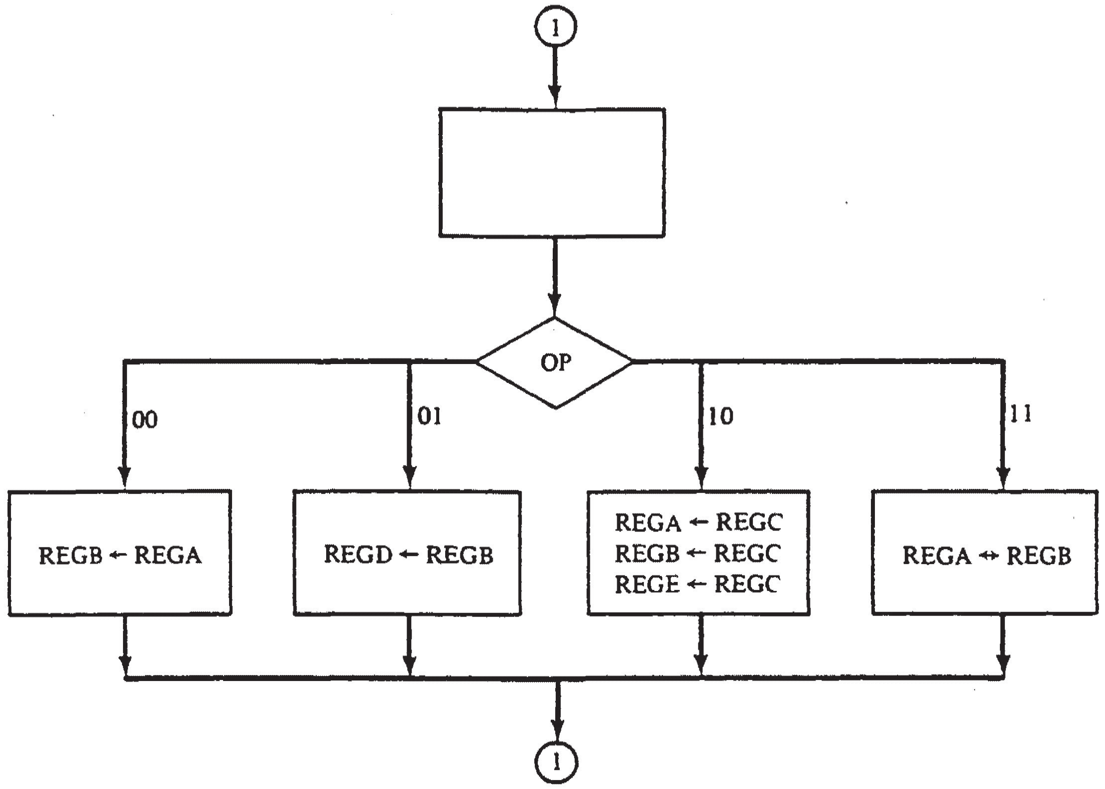
(a)

Figure 7.51 Circuit elements and flowchart for Problem 7.24.

(c) The flowchart for the controller is given in Fig. 7.51(b). Derive the corresponding ASM chart for it. Optimize it by using the least number of states. In other words, if mor than one operation can be performed in a single state, then do so. Also, make use o conditional outputs.

7.25. (a) Given in Fig. 7.52(a) are the circuit elements for a digital circuit that is to be designed They are organized in a common bus structure. Assume that the access time of MEM is 125 ns (nanoseconds), and that the clock period for the ASM is 100 ns. Then, ho many states are required to perform a MEM read or MEM write operation?

(b) The flowchart for the controller is given in Fig. 7.52(b). Derive the corresponding ASM chart for it. Optimize the ASM chart by using the least number of states. In other words if more than one operation can be performed in a single state, then do so. Also, mak use of conditional outputs.



Note: REGA ↔ REGB means to interchange the contents of REGA and REGB

(b)

Figure 7.51 *(cont.)*

7.26. A hardware stack module, the block diagram of which is shown in Fig. 7.53, is to be designed and implemented. The function of this hardware stack module is defined as follows:

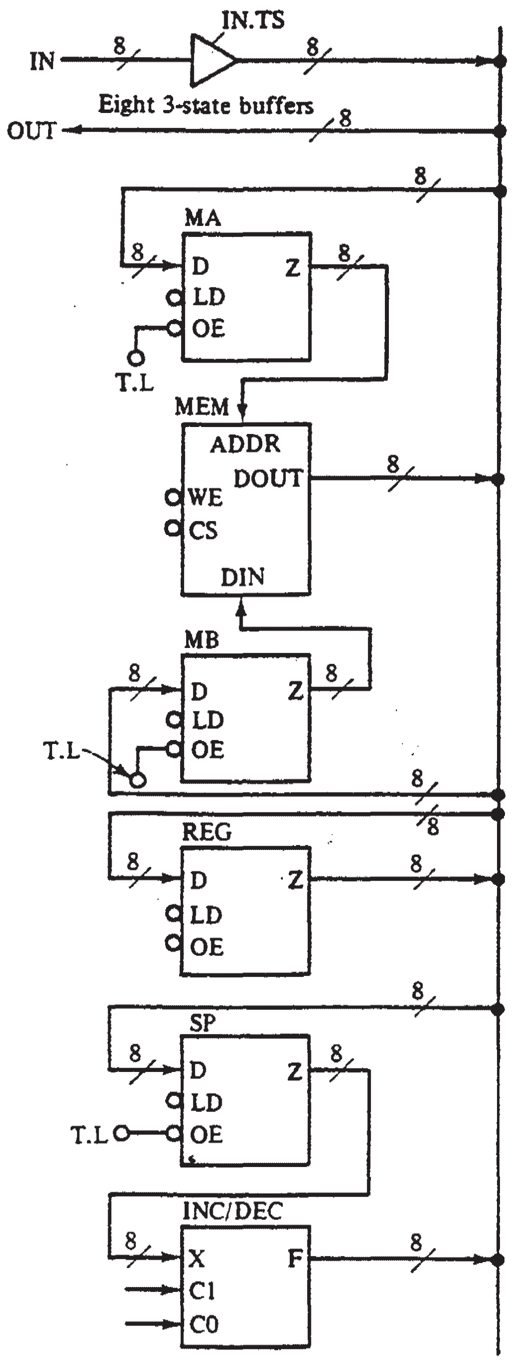
If STKENBL = 0, do nothing.
If STKENBL = 1, then there are four possible operations, depending on OP:

OP = 00	DEFINE a new top of stack; i.e., SP ← IN.
OP = 01	PUSH the stack; i.e., increment SP; MEM(SP) ← IN.
OP = 10	POP the stack; i.e., MEM(SP) is connected to OUT until STKENBL becomes 0; decrement SP.
OP = 11	READ the top of the stack; i.e., SP is connected to OUT until STKENBL becomes 0.

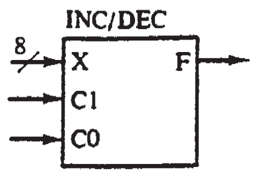
Notes:

1. SP contains the address (8 bits) of the top of the stack.
2. MEM(SP) is the memory content of that address.
3. Do not worry about the stack being empty or full.

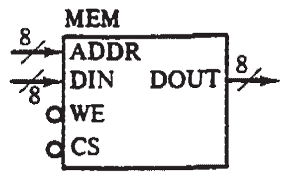
(a) Using the circuit elements shown in Fig. 7.52(a), derive the ASM chart for the controller for the hardware stack module. Optimize it by using the minimum number of states.
(b) Using any commercially available chips, realize your design.



Definition of circuit elements:
All registers (MA, MB, REG, and SP) are defined the same as the registers in Fig. 7.51(a).



C1	C0	Function
0	0	Outputs F are 3-stated
0	1	F = X
1	0	F = X + 1
1	1	F = X − 1



MEM is a 256 X 8 RAM module.

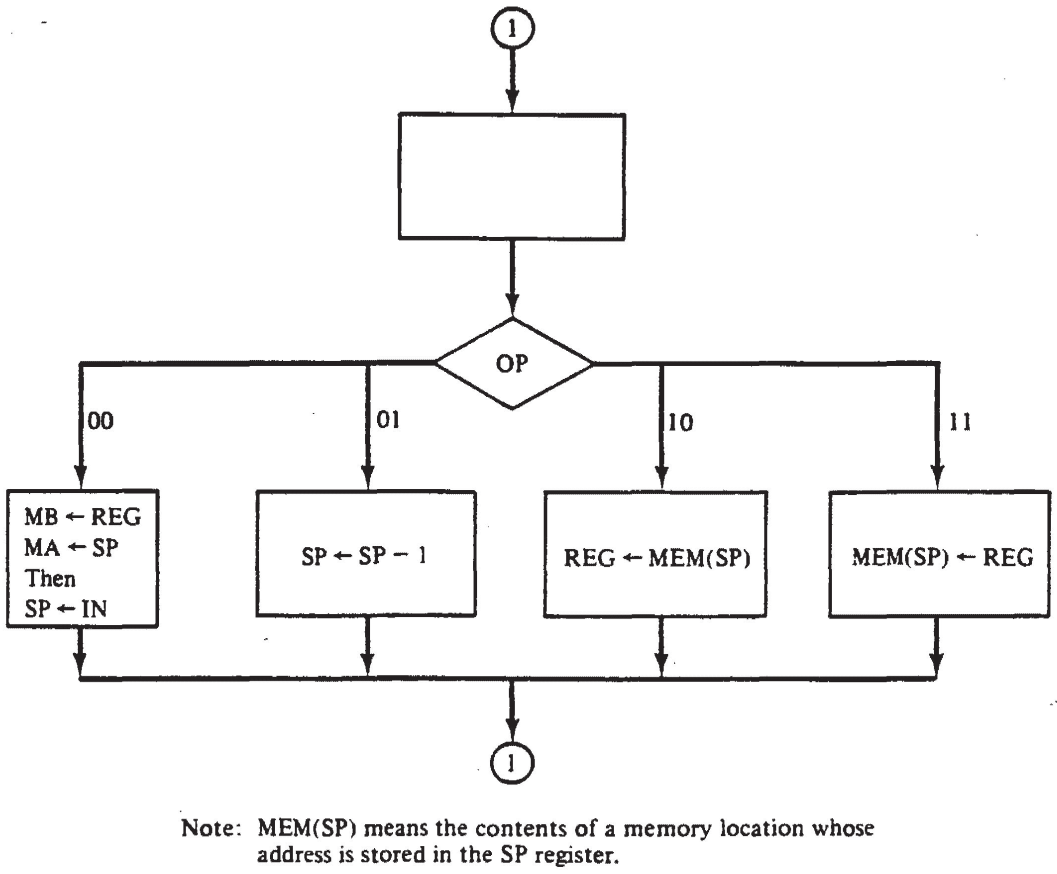
WE	CS	Function
X	H	Outputs DOUT are 3- stated.
L	L	Data applied at DIN is written to the RAM location specified by ADDR. Also, DOUT outputs are 3-stated.
H	L	Contents of RAM location specified by ADDR are outputted at DOUT.

(a)

Figure 7.52 Circuit elements and controller flowchart for Problem 7.25.

7.27. Repeat Problem 7.26 and take care of the problem of whether the stack is empty (for the POP operation) or full (for the PUSH operation).

7.28. Design and realize the hardware multiplier of Sec. 7.8.4, modified as follows: Instead of using two separate 4-bit registers (MPLIER and PRODLO) to store the multiplier and the low-nibble product, as shown in Fig. 7.36(a), use PRODLO for storing both the multiplier and low-nibble product. Specifically, use PRODLO initially for storing the multiplier since


1
OP
00
01
10
11
MB ← REG
MA ← SP
Then
SP ← IN
SP ← SP − 1
REG ← MEM(SP)
MEM(SP) ← REG
1
Note: MEM(SP) means the contents of a memory location whose address is stored in the SP register.

(b)

Figure 7.52 *(cont.)*

this register is not used initially to store the low-nibble product. As the multiplication process proceeds, have the multiplier shifted out of PRODLO one bit at a time. At the same time, have the low-nibble product shifted into PRODLO from PRODHI. This is a more elegant design that saves the use of a register.

7.29. (a) Draw a block diagram for an enhanced hardware multiplier and specify its functions. You have the flexibility of incorporating any features that you desire. For example, you

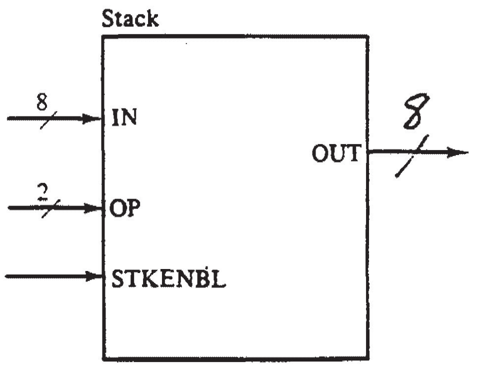


Figure 7.53 Hardware stack module for Problem 7.26.

may want your multiplier to be able to perform the multiplication of signed binary numbers, or of BCD numbers, or have additional handshaking capabilities, and so forth.

(b) Design and realize this enhanced multiplier.

7.30. A stack, as described in Problem 7.26, is a first-in, last-out (FILO) structure. In other words, when an 8-bit data is stored (pushed) onto the stack, it cannot be retrieved (popped) until all the data that has been subsequently stored is popped first, much like a stack of trays in a cafeteria. On the other hand, a queue is a first-in, first-out (FIFO) structure. In other words, the first 8-bit data stored is the first to be retrieved, much like the servicing of a line of customers in a cafeteria. With this background in mind,

(a) Draw a block diagram for a hardware queue module and specify its functions. You have the flexibility of incorporating any features that you desire.

(b) Design and realize this hardware queue module.

7.31. For the transmission of asynchronous serial data, a start bit must be inserted before each data byte and a stop bit inserted after the data byte. Also, for error-checking purposes, sometimes a parity bit is inserted between the data byte and the stop bit. In this problem you are to redesign the parallel-to-serial converter of Fig. 7.48 so that it will perform these insertions. Specifically, when the START input is false, the converter is to be in an idle state in which it outputs a "high" at SOUT. But when START becomes true, the converter loads the 8-bit data BYTE into the shift register and shifts out the start bit first, after which it begins the shifting out of the data. After shifting out the 8 data bits, it shifts out the parity bit, followed by the stop bit.

Notes:

1. The value of the start bit is L.
2. The value of the stop bit is H.
3. The value of the parity bit has to be determined as follows: the parity bit = L if the number of ones in the data byte is an even number; the parity bit = H if the number of ones in the data byte is an odd number.