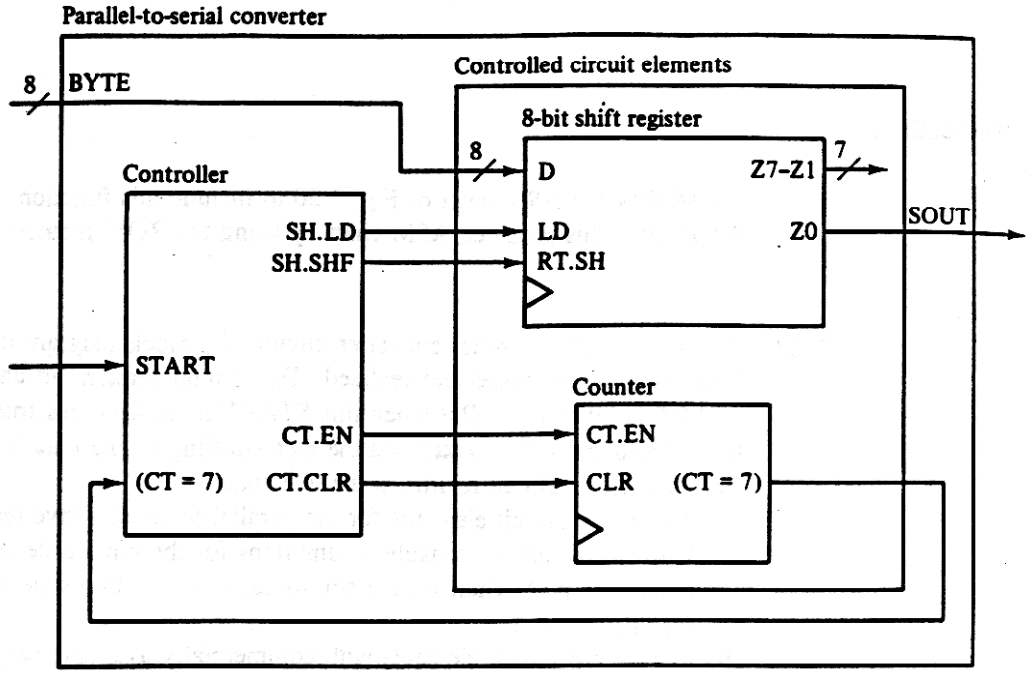
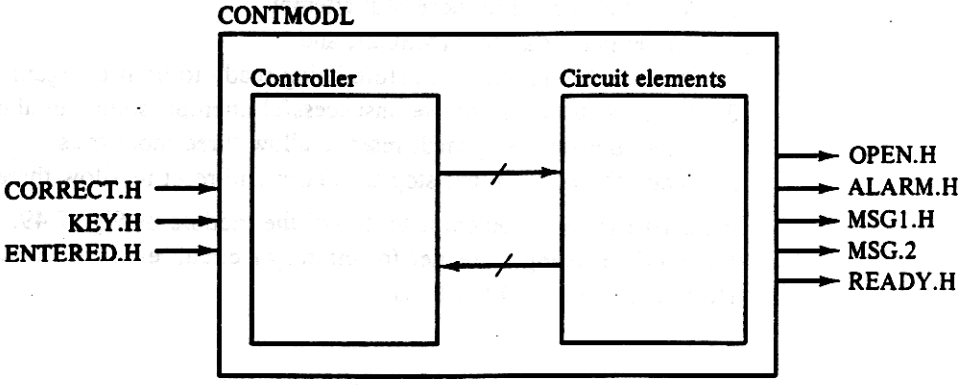


Figure 7.48 Parallel-to-serial converter circuit for Problem 7.21.



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

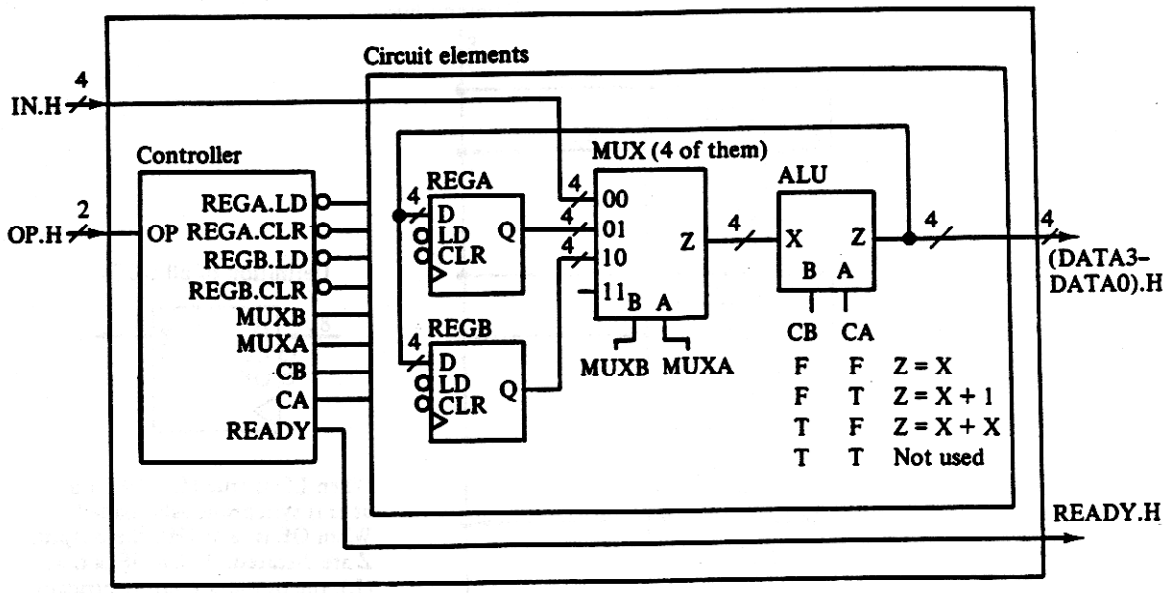
ALARM: an output signal for another module to sound the alarm

MSG1: an output signal for another module to output message 1

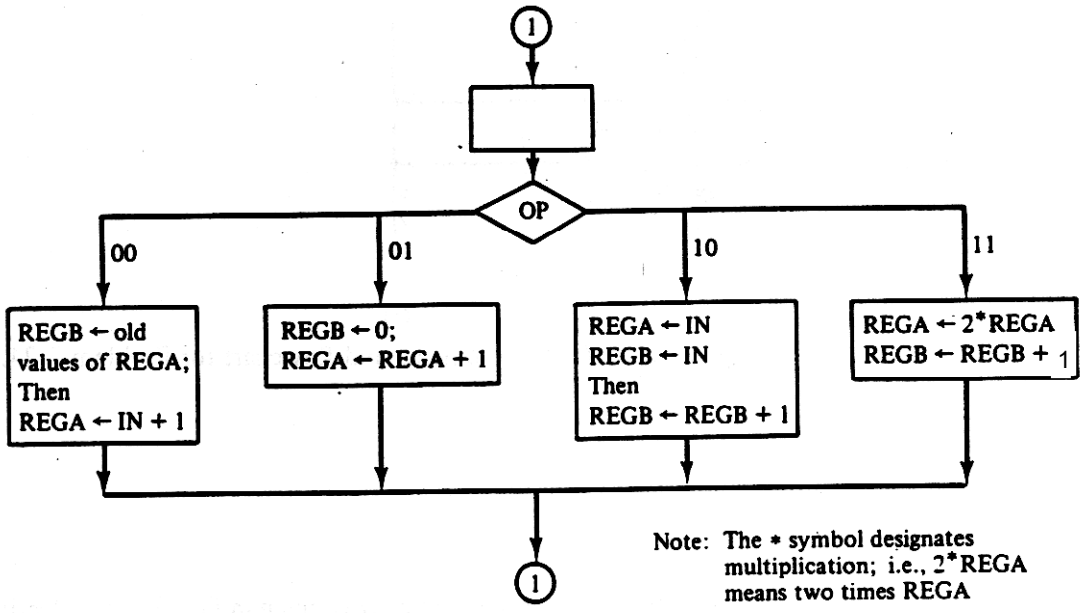
MSG2: an output signal for another module to output message 2

READY: an output signal for another module to allow another attempt

Figure 7.49 Module for Problem 7.22.



(a)



(b)

Figure 7.50 Circuit elements and flowchart for Problem 7.23.

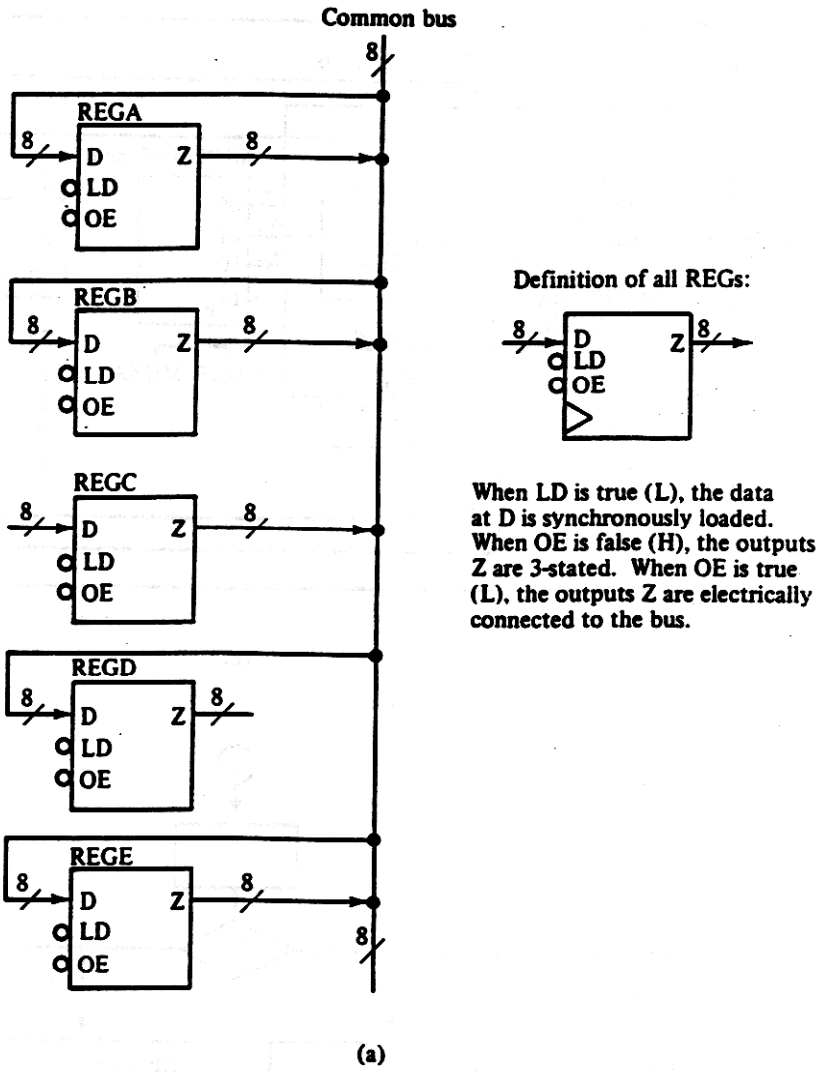
7.24. Figure 7.51(a) shows circuit elements for a digital circuit that is to be designed. They are organized in a common bus structure.

(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?

REGA ← REGD and REGC ← REGA

Explain your answer.



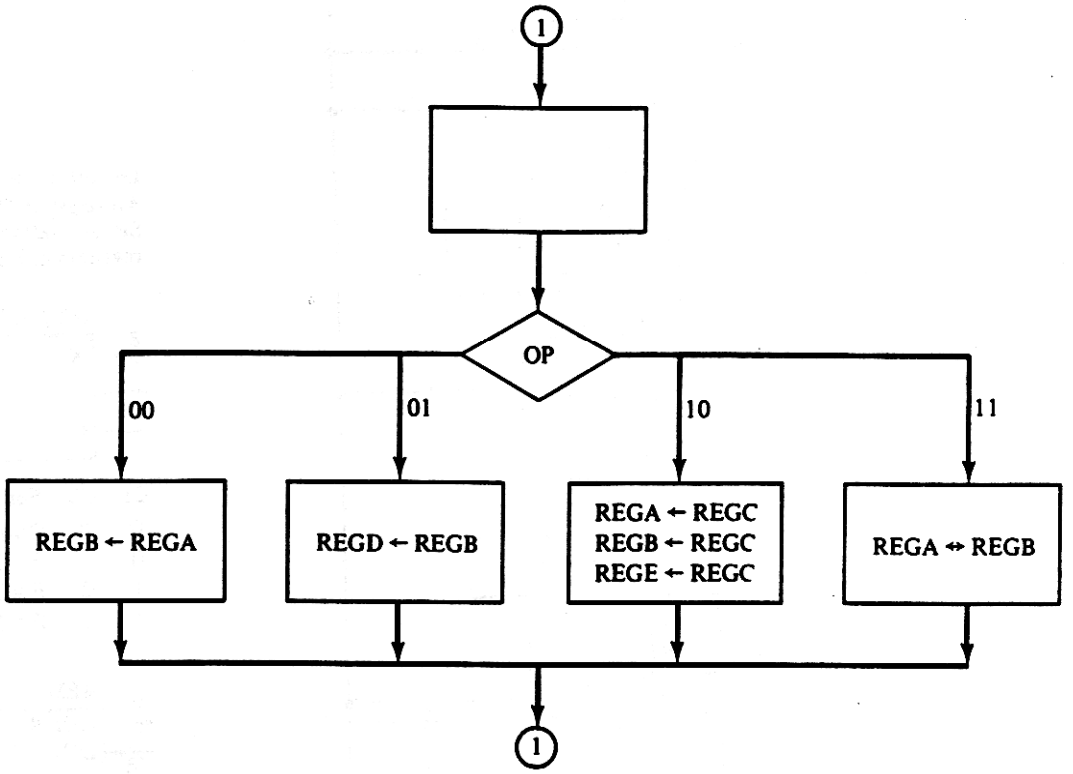
(a)

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 more than one operation can be performed in a single state, then do so. Also, make use of 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, how 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, make use of conditional outputs.



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

(b)

Figure 7.51 *(cont.)*

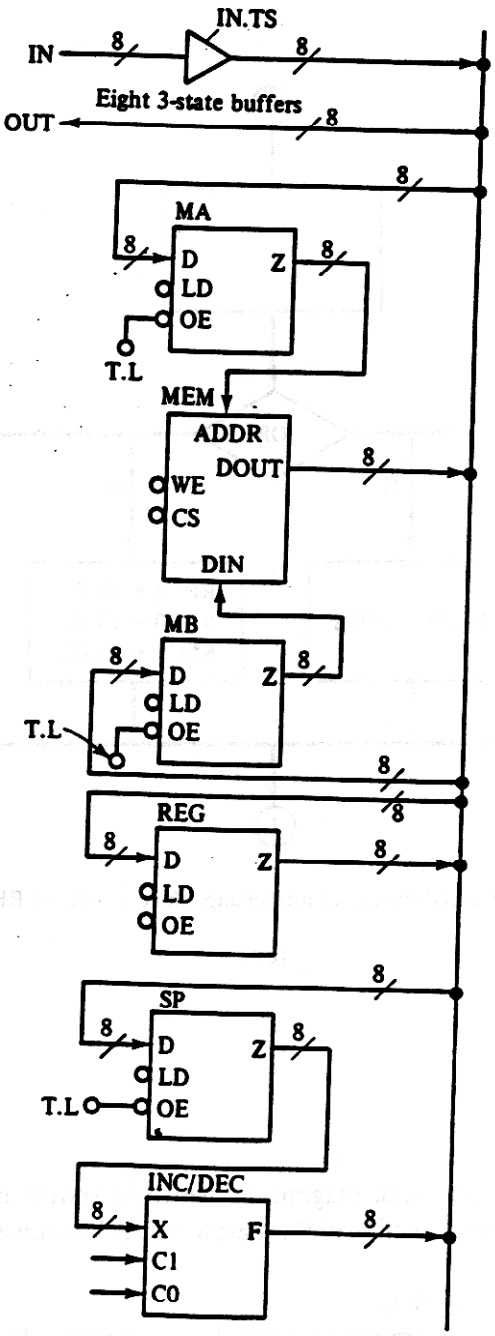
7.26. A hardware stack module, the block diagram of which is shown in Fig. 7.53, is to be designed and implemented. The function of this hardware stack module is defined as follows:

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.

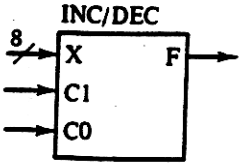
Notes:

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.

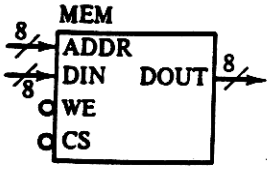
(a) Using the circuit elements shown in Fig. 7.52(a), derive the ASM chart for the controller for the hardware stack module. Optimize it by using the minimum number of states.
(b) Using any commercially available chips, realize your design.



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.

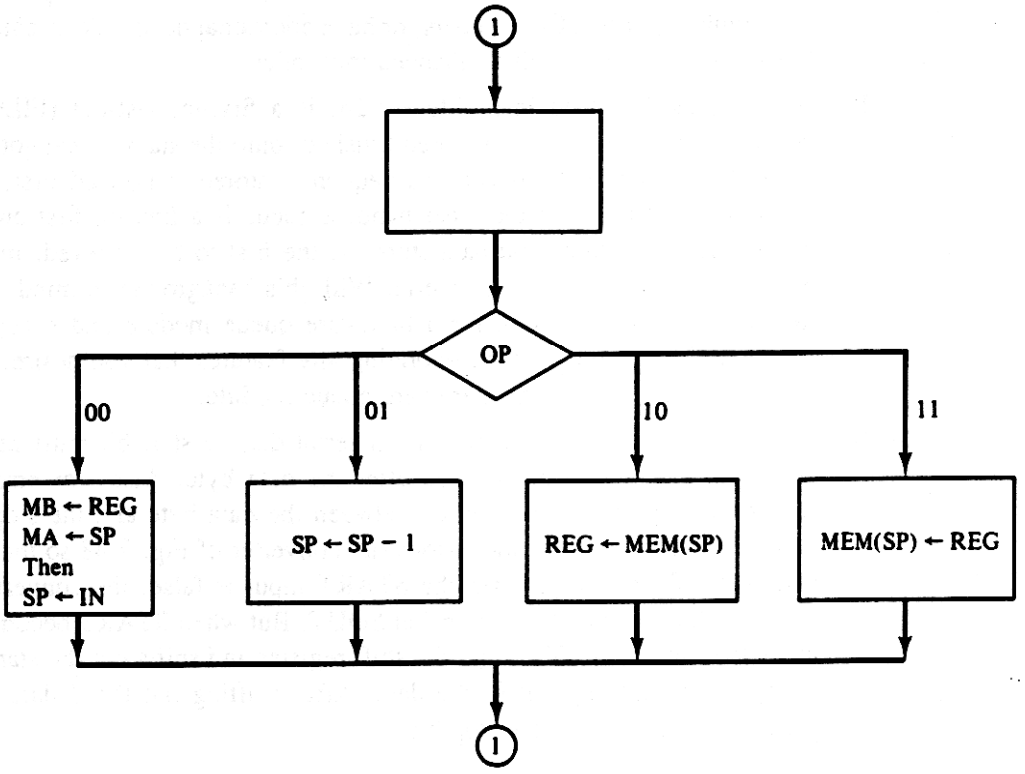
WE	CS	Function
X	H	Outputs DOUT are 3-stated.
L	L	Data applied at DIN is written to the RAM location specified by ADDR. Also, DOUT outputs are 3-stated.
H	L	Contents of RAM location specified by ADDR are outputted at DOUT.

(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



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

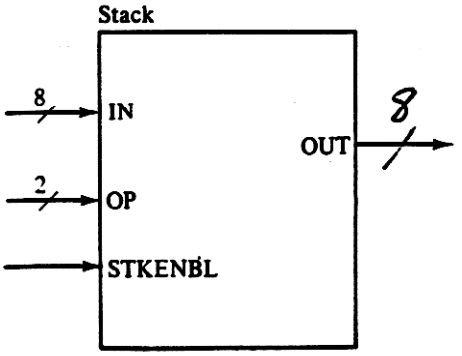


Figure 7.53 Hardware stack module for Problem 7.26.