Logic design — 13

Flags and flag sorters

by B. Holdsworth* and D. Zissos† *Chelsea College, University of London

†Dept. of Computing Science, University of Calgary, Canada.

When a student in a classroom environment wishes to ask a question, he raises his hand, and waits for the teacher to ask him to speak. On being told to go ahead, the student lowers his hand and proceeds to ask his question. Similarly when a device wishes to communicate with another device in the same system, it raises a flag in order to attract attention.

A SIMPLE communication system is shown in Fig. 1 where f represents a flag signal, which is, in this case, equivalent to the student raising his hand in the classroom. If and when the called device is able to communicate with the calling device, it sends back, as in the case of the teacher, a 'go-ahead' signal. The calling device then clears its flag and the two devices communicate.

A typical example of the use of flags occurs when a peripheral device such as a paper tape reader is ready to transfer data to another device, such as the c.p.u. of a digital computer. Firstly, it makes the data available for transfer and secondly, it generates a signal — the flag — to inform the c.p.u. that the data is available for transfer.

If device 2 in Fig. 1 is dealing with a situation during which it must not be interrupted, the flag of device 1 is *disabled*, that is, it is prevented from being raised. When the restriction is removed, the circuit is enabled. In the classroom analogy this corresponds to the teacher first not allowing his students to interrupt him to ask him a question and then at some later time removing this restriction.

Summarising, a flag is a signal generated and used by a device to inform some other device that it wishes to communicate with it. A disable signal prevents the flag from being raised and an enable signal allows it to be raised. A clear signal turns the flag off without disabling it.

Flag circuits

The block diagram of a flag circuit with turn-on, clear, enable, and disable facilities is shown in Fig. 2. A signal on terminal *e* enables the circuit, whereas a signal on terminal *d* disables the circuit. Clearly, the two signals must not be applied simultaneously. When the circuit is enabled, a signal on terminal *p* generates a flag, which is cleared by a signal on terminal c. When the circuit is disabled, the presence of signal *p* does not turn the flag on. If enable and disable facilities are not needed, terminals *e* and *d* are omitted.

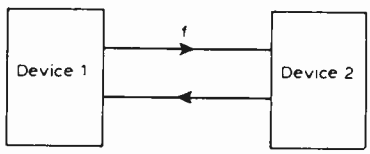


Fig. 1. Two communicating devices, with a flag signal.

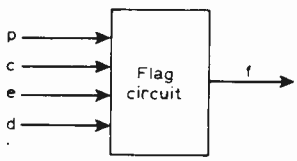


Fig. 2. Elements of a flag circuit.

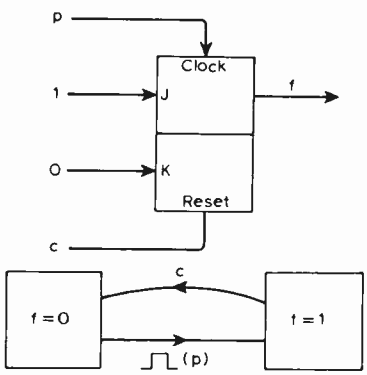


Fig. 3. Simplest flag circuit (a) and its state diagram (b).

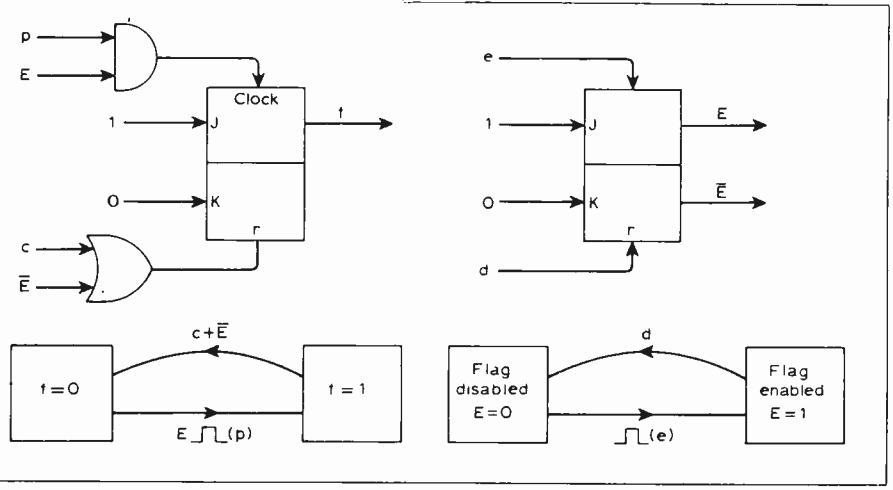
Flag circuit 1. The simplest circuit consists of a single JK flip-flop, which uses signal *p* as its clock pulse and signal c as the reset signal, as shown in Fig. 3(a). Since J = 1 and K = 0 a pulse on terminal *p* will set the flip-flop, whilst a signal c on the reset terminal resets the flip-flop unconditionally. The state diagram for this circuit is shown in Fig. 3(b).

Flag circuit 2. The flag circuit in Fig. 4 is basically the same as circuit 1, with the addition of a second flip-flop, the function of which is to enable and disable the flag flip-flop. If the second or E flip-flop is set, the circuit reduces functionally to that in Fig. 3. When E = 0 the flag flip-flop is unconditionally reset and cannot be turned on. The E flip-flop is set by the enable pulse on its clock line and reset by a disable signal *d* on its unconditional reset terminal *r*.

Flag circuit 3. Alternatively a flag circuit can be described by the two state diagrams shown in Figs. 5(a) and 5(b). From the state diagrams, the following equations are obtained.

turn-on set of E = e,
turn-off set of E = d,
hence E = e + $E\bar{d}$.
Turn-on set of A = BE,
turn-off set of A = $\bar{p}c\bar{B}$,
hence A = BE + A $\overline{\bar{p}c\bar{B}}$,
= BE + A (p + c + B).

Fig. 4. Circuit 2, with an additional flip-flop to enable or disable the flag. State diagrams are at (b).



Logic design 13 *continued from page 82*

group number is that specified by the output of the group selector.

The operation of this flag sorter will be explained by direct reference to Fig. 14. Since more than one group signal may be present at any one time, an eight-flag sorter is used to lock out all but one of the group signals. This flag sorter is called the group selector. It will be assumed that two flags in group 1 (say f_3 and f_5) and one flag (say f_6) in group 4 are raised, that is $g_1 = 1$ and $g_4 = 1$. By direct reference to Fig. 13 it can be seen that the group selector leaves the homing state S_0 and assumes state S_3. When in state S_3 the group selector's output is 100. This output is decoded, causing signal e_4 to assume 1, thus enabling the group 4 flag sorter. The output of flag sorter 4(110) is connected to lines A, B and C of the data bus, whilst the output of the group selector 100 are connected to lines D, E and F of the data bus. The signals on these six lines ABCDEF = 110100 will identify the flag to be served, namely flag 6 in group 4. At the same time the group selector will also generate the interrupt signal I.

References

1. "Problems and solutions in Logic Design" D. Zissos, Oxford University Press 1976.
2. "Digital Interface Design" D. Zissos and F. G. Duncan, Oxford University Press 1973. □

Turn-on set of B = $p\bar{A}$,
turn-off set of B = $\bar{A}c$,
hence B = $p\bar{A} + B\bar{A}\bar{c}$,
= $pA + B(\bar{A} + \bar{c})$.
f = AB.

Diagrams corresponding to these equations are shown in Fig. 5(c).

Identification of flags

In a computer system, each of the peripherals can communicate directly with the computer. As explained earlier, every device generates its own flag when it needs to communicate with the central equipment. These flags are ORed to generate a master flag: when the central device is a computer or a microprocessor the master flag is called an interrupt request because it is used to request the computer or the microprocessor to interrupt its current activity and service the peripheral's needs. The master flag or the interrupt request simply informs the central device that one of the peripherals in the system wishes to communicate with it. It is the function of the called device to identify which of the peripherals wishes to communicate with it.

There exists two basic methods for identifying flags – the polling method and the vectored method.

Polling method. In this method, also known as the rest and skip method, the central device, when it receives an interrupt, sequences through the peripherals looking for the individual device that needs servicing, as illustrated in Fig. 6. When it finds such a device, it stops sequencing and calls the corresponding service routine, at the end of which the polling of the devices continues until they have all been polled. At this point the main programme is resumed.

The flow chart of a polling routine for *n* devices is shown in Fig. 6(b). The counter in the flow chart is an internal counter in the central device. In this system no hardware assistance is provided for determining which device is requesting service. Whilst this method saves in hardware cost, considerable processor time is used in test loops and system performance is degraded.

Vectored method. In this method, the presence of a flag is automatically detected and identified by means of a circuit known as a flag sorter. The basic arrangement of the system incorporating the flag sorter is shown in Fig. 7.

Flag sorters

A flag sorter (alternatively called a priority encoder) is defined as a device that automatically detects the presence of a signal at its input and identifies it, producing an interrupt signal I, and the identity of the signal being automatically generated on address lines A and B. The interrupt signal I is the OR function of all the input signals,

$$I = f_0 + f_1 + f_2 + f_3$$

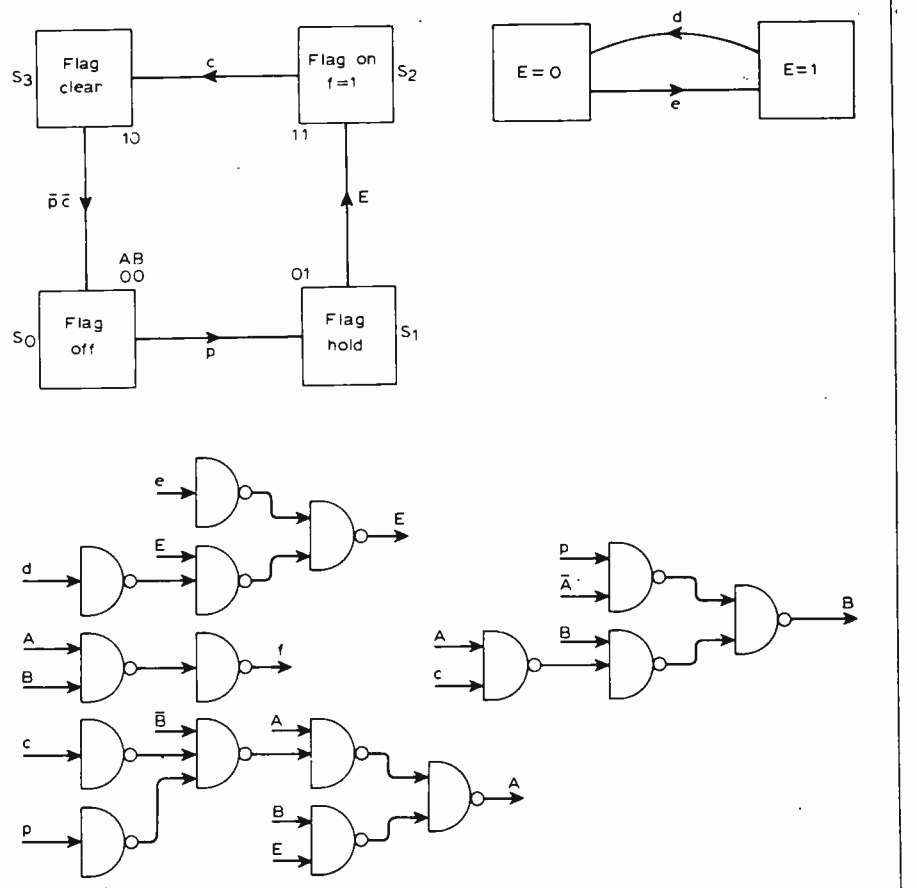


Fig. 5. State diagram for circuit 3 enable (a) and flags (b). Circuit diagram is at (c).

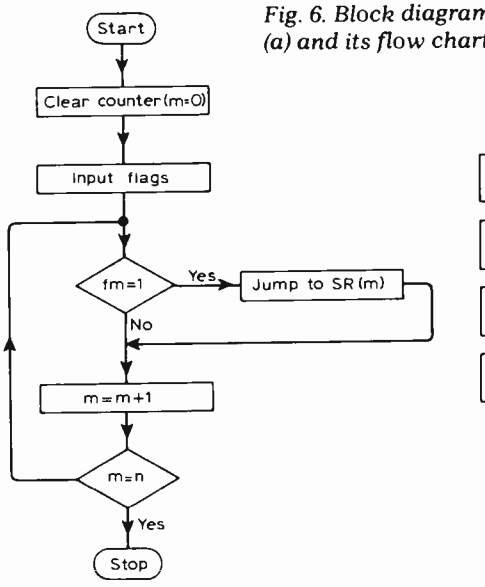


Fig. 6. Block diagram of p flag identifier (a) and its flow chart (b).

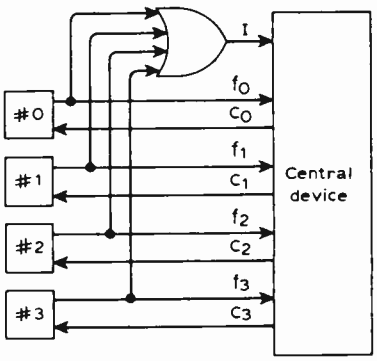
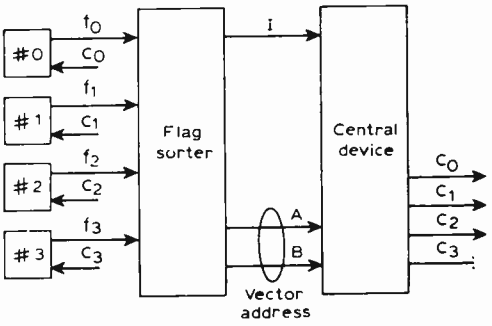
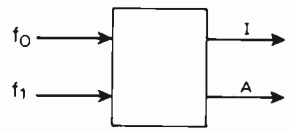


Fig. 7. Use of a flag sorter.





Inputs		Outputs	
f_0	f_1	I	A
0	0	0	0
X	1	1	1
1	0	1	0

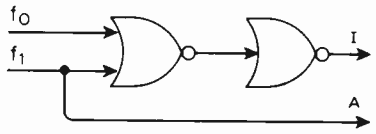
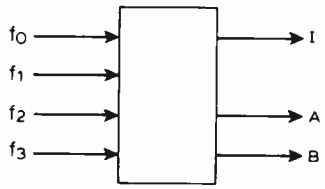


Fig. 8. Block diagram (a), truth table (b) and circuit (c) for a two-flag sorter.



Inputs				Outputs		
f_0	f_1	f_2	f_3	I	A	B
0	0	0	0	0	0	0
X	X	X	1	1	1	1
X	X	1	0	1	0	1
X	1	0	0	1	1	0
1	0	0	0	1	0	0

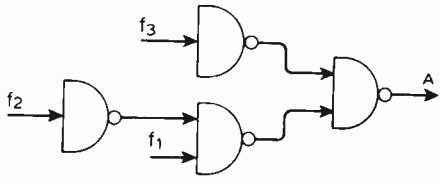
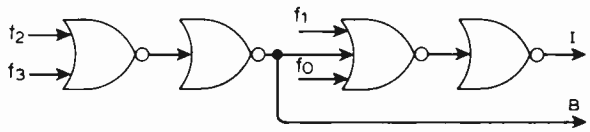


Fig. 9. Four-flag sorter.



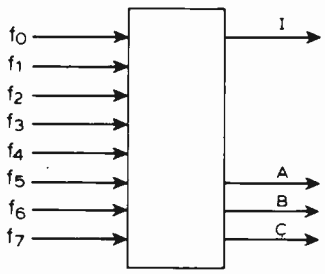
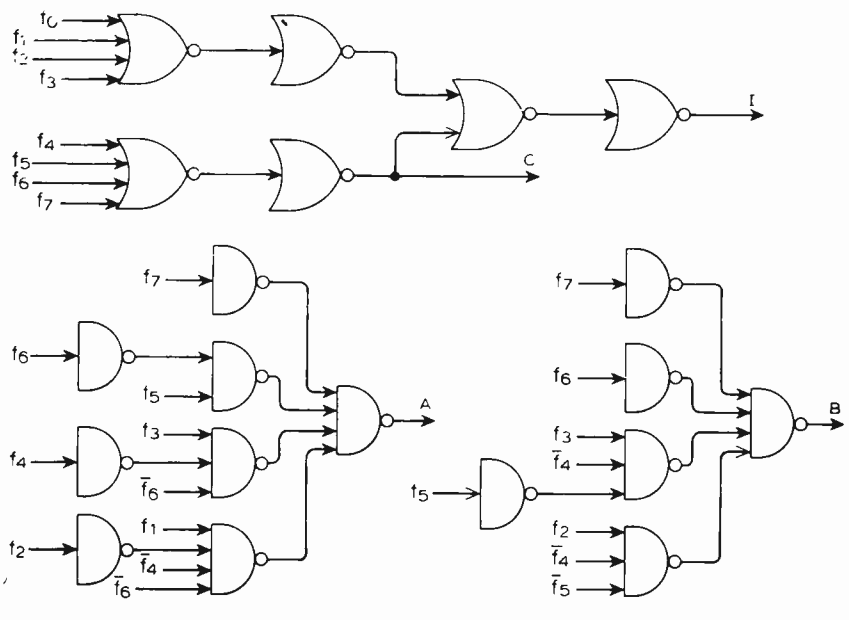


Fig. 10. Eight-flag sorter.

Inputs								Outputs			
f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	I	A	B	C
0	0	0	0	0	0	0	0	0	0	0	0
X	X	X	X	X	X	X	1	1	1	1	1
X	X	X	X	X	X	1	0	1	0	1	1
X	X	X	X	X	1	0	0	1	1	0	1
X	X	X	X	1	0	0	0	1	0	0	1
X	X	X	1	0	0	0	0	1	1	1	0
X	X	1	0	0	0	0	0	1	0	1	0
X	1	0	0	0	0	0	0	1	1	0	0
1	0	0	0	0	0	0	0	1	0	0	0



in the case of the four-flag sorter shown in Fig. 7. For convenience, the address signals are generated in the binary (8-4-2-1) code, although any other code may equally well be used.

The design and implementation of flag sorters is straightforward, using conventional methods, either combinationally or sequentially. There are practical situations where a given flag signal must be given priority over the others. If it is present with other flag signals, its address must be given at the output, regardless of the state of the flag sorter: this is particularly the case in automatic plants, where certain alarm signals are given priority over other alarms. Unless it is otherwise specified, the higher the suffix of a flag the higher its priority.

Combinational flag sorters. The block diagram of a simple two-flag sorter is shown in Fig. 8(a). This circuit is required to generate an interrupt signal I to indicate one or more flags are present and an address signal A which can have the value 0 or 1 and is able to identify the two flag signals f_0 and f_1, where f_1 is the flag signal with the highest priority.

A truth table for the circuit is shown in Fig. 8(b). In this table the X entry is used to denote a 0 or a 1, and its use in the second row indicates that A = 1 if $f_1 = 1$ irrespective of whether flag signal f_0 is present or not.

From the truth table the logic equations for A and I can be derived. They are:

$$\begin{aligned} I &= \bar{f}_0 f_1 + f_0 f_1 + f_0 \\ &= f_1 + f_0 \\ A &= \bar{f}_0 f_1 + f_0 f_1 \\ &= f_1 \end{aligned}$$

The implementation of these equations is shown in Fig. 8(c).

The block diagram of a combinational four-flag sorter is shown in Fig. 9(a). In

this case, to identify the four flags, two address signals are required. A truth table for the circuit is shown in Fig. 9(b), and the logic equations for I, A and B are derived directly from the entries in this table. The equations are:

$$I = f_0 + f_1 + f_2 + f_3$$
$$A = f_3 + \bar{f}_3\bar{f}_2 f_1 = f_3 + \bar{f}_2 f_1$$
$$B = f_3 + \bar{f}_3 f_2 = f_3 + f_2$$

As the number of flags to be sorted increases so does the complexity of the logic equations for the address signals. In the case of an eight-flag sorter three address signals A, B and C are required. The block diagram of the sorter is shown in Fig. 10(a) and the corresponding truth table is shown in Fig.10(b). The logic equations for I, A, B and C derived from the truth table are:

$$I = f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7$$
$$A = f_7 + f_5\bar{f}_6\bar{f}_7 + f_3\bar{f}_4\bar{f}_5\bar{f}_6\bar{f}_7 + f_1\bar{f}_2\bar{f}_3\bar{f}_4\bar{f}_5\bar{f}_6\bar{f}_7$$
$$= f_7 + f_5\bar{f}_6 + f_3\bar{f}_4\bar{f}_6 + f_1\bar{f}_2\bar{f}_4\bar{f}_6$$
$$B = f_7 + f_6\bar{f}_7 + f_3\bar{f}_4\bar{f}_5\bar{f}_6\bar{f}_7 + f_2\bar{f}_3\bar{f}_4\bar{f}_5\bar{f}_6\bar{f}_7$$
$$= f_7 + f_6 + f_3\bar{f}_4\bar{f}_5 + f_2\bar{f}_4\bar{f}_5$$
$$C = f_7 + f_6\bar{f}_7 + f_5\bar{f}_6\bar{f}_7 + f_4\bar{f}_5\bar{f}_6\bar{f}_7$$
$$= f_7 + f_6 + f_5 + f_4$$

Sequential flag sorters. The main disadvantage of the combinational flag sorter is that the address signals may change whilst being read by the central device. For example, in the case of the eight-flag sorter, if the interrupt signal is raised by flag 3, and flag 4 subsequently arrives, the output of the flag sorter will be changing from ABC = 110 to ABC = 001. During the change signal C has to turnon whilst signals A and B have to turnoff. If C changes more rapidly than A and B the output is momentarily ABC = 111 and, assuming this occurs when the central device is reading the address of the flag sorter, address ABC = 111 will be recorded in error, resulting in circuit misoperation. It is for this reason that sequential flag sorters have been introduced.

The sequential circuit, whose block diagram is shown in Fig. 11(a), has to generate an interrupt signal if any of the three flags f_1, f_2, or f_3 are raised and in this case it will be arranged that the flag signals are serviced in cyclic order. The flag signals may be regarded as if they are arranged in a circle which is scanned whenever a flag signal is raised. When there is no flag signal present, the circuit rests in a homing state.

A suitable internal state diagram is shown in Fig. 11(b) where the addresses corresponding to flag signals f_1, f_2, and f_3 are AB = 01, AB = 10 and AB = 11. In order to avoid races between secondary signals, it is convenient to take these addresses in an order in which only one secondary signal changes at a time, allowing the use of the secondary signals directly as the address signals, and stipulating the cyclic order of the flags as f_1, f_3 and f_2.

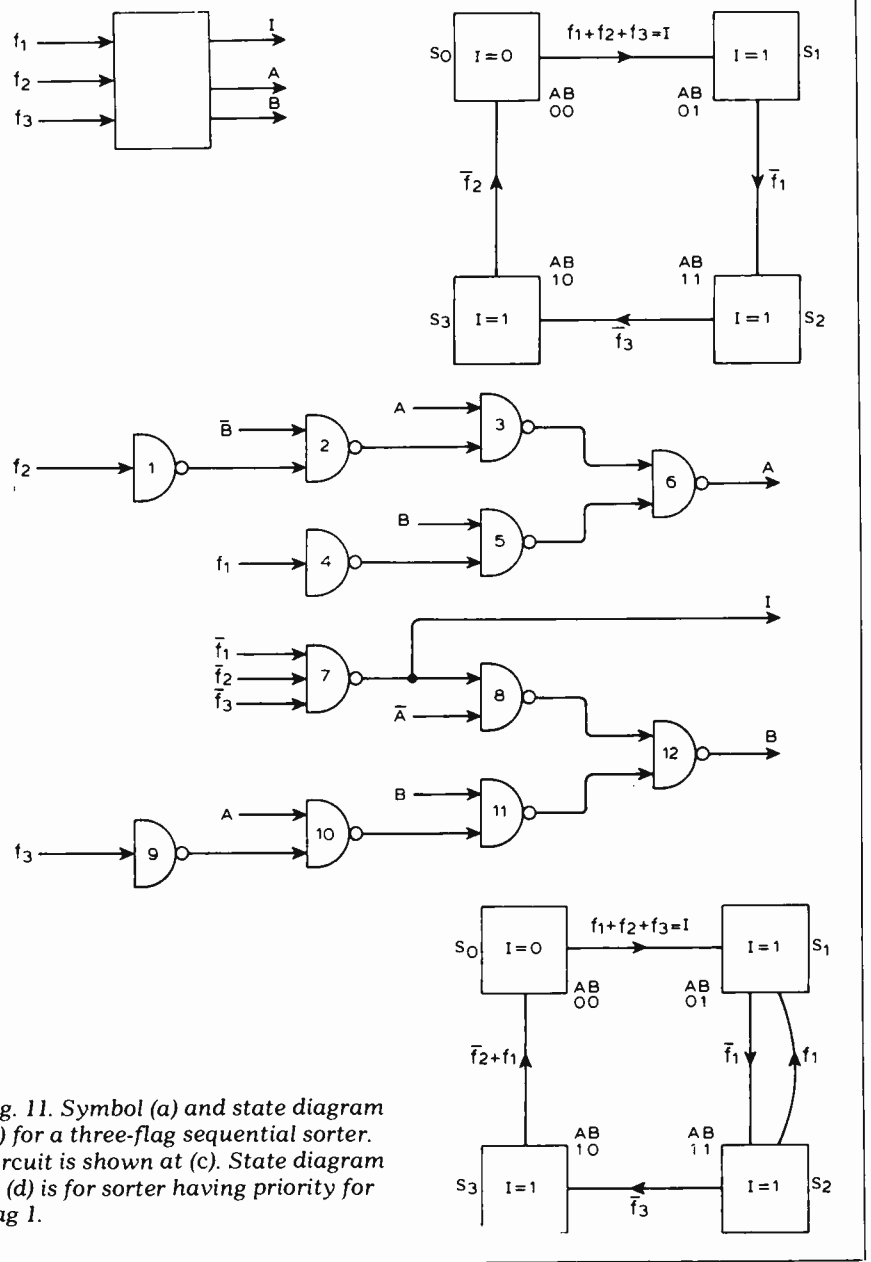
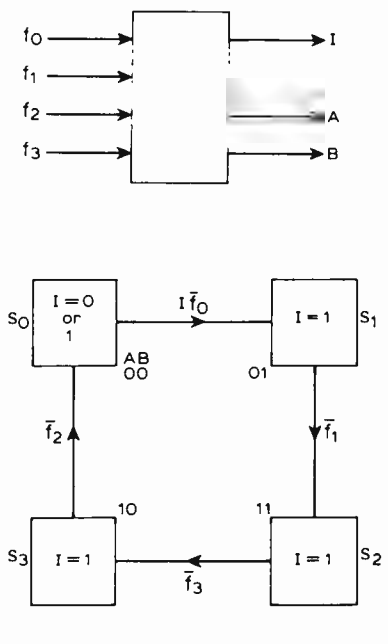


Fig. 11. Symbol (a) and state diagram (b) for a three-flag sequential sorter. Circuit is shown at (c). State diagram at (d) is for sorter having priority for flag 1.

Fig. 12. Four-flag sequential sorter.



Assuming there are no flag signals present, the circuit takes up the homing state S_0 and the circuit outputs are I = 0 and AB = 00. If one or more flag signal is present at the input, $I = f_1 + f_2 + f_3 = 1$ and the circuit makes a transition from state S_0. For $f_1 = 1$ the circuit assumes the state S_1 where I = 1 and AB = 01. On the other hand, if $f_1 = 0$, the circuit assumes state S_2, I = 1 and AB = 11. Similarly if $f_3 = 0$ the circuit makes a transition to state S_3, AB = 10 and I = 1. The circuit will then return to the homing state when $f_2 = 0$.

Turn-on set of $A = B\bar{f}_1$,
Turn-off set of $A = \bar{B}f_2$,
Turn-on set of $B = \bar{A}(f_1 + f_2 + f_3)$,
Turn-off set of $B = A\bar{f}_3$.
$A = B\bar{f}_1 + A(B + f_2)$.
$B = \bar{A}(f_1 + f_2 + f_3) + B(\bar{A} + f_3)$.
$= \bar{A}I + B(\bar{A} + f_3)$.
$I = f_1 + f_2 + f_3$.

The implementation of these equations is shown in Fig. 11(c).

If it is stipulated that the flag f_1 must always be given priority, then to satisfy this requirement, the internal state diagram must be modified as shown in Fig. 11 (d). An examination of this state diagram indicates that the modifications only change the turn-off conditions of the secondary signal A which now becomes:

$$A = B\bar{f}_1 + A(\overline{\bar{B}(\bar{f}_2 + f_1) + Bf_1})$$
$$= B\bar{f}_1 + A(B + f_2)\bar{f}_1.$$

The implementation of this equation only requires an additional signal $\bar{f}_1$ at the input of gate 3 in Fig. 11(c).

A suitable arrangement for the detection of four flags is shown in Fig. 12(a) and the corresponding internal state diagram in Fig. 12(b). The presence of any incoming flag signal causes signal I to be generated. If flag f_0 is raised, the circuit remains in state S_0 and the flag sorter outputs are I = 1, A = 0 and B = 0.

State S_0 is a homing state, because the flag sorter automatically assumes this state when the incoming signals f_0, f_1, f_2 and f_3 are cleared. The order in which the signal addresses are generated depends on the current state of the sorter. For example, when address AB = 00 is at the output, with I = 1, the incoming signals have the following priorities: priority No $1 - f_1$, priority No $2 - f_3$, priority No $3 - f_2$.

The flag sorter logic equations which can be developed from the internal state diagram are;

$$I = f_0 + f_1 + f_2 + f_3$$
$$A = B\bar{f}_1 + A(B + f_2)$$
$$B = \bar{A}\bar{f}_0 I + B(\bar{A} + f_3),$$

and the implementation of these equations is left to the reader.

Clearly, the state diagram and implementation of an eight-flag sorter can be developed in a similar manner, the state diagram being shown in Fig. 13.

Modular expansion

In the case of flag sorters for more than eight flags, it is more convenient to use a modular construction, which allows any number of flag signals to be identified. The design principle will be demonstrated for the case of 64 flags.

The incoming flags are arranged into eight groups of eight flags each, as shown in Fig. 14, the flags in each group being connected to an eight-flag sorter. Each flag sorter generates a signal g independently of the other sorters. Thus $g_0 = 1$ if any of the flags in group 0 are present, similarly for group flags $g_1, g_2, g_3, g_4, g_5, g_6$ and g_7. Individual flags are indentified by six binary suffices, the first three indicating the flag number and the second three the group number, for example, f_{010111}, refers to flag 2 in group 7. The second set of three digits is generated directly by the group selector, while the first three digits are generated by the flag sorter, whose

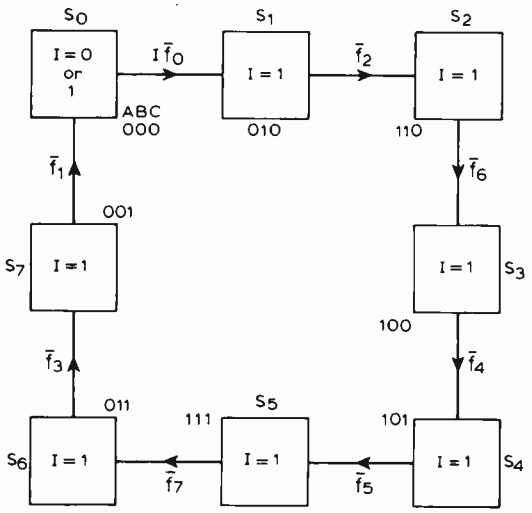


Fig. 13. State diagram for eight-flat sequential sorter.

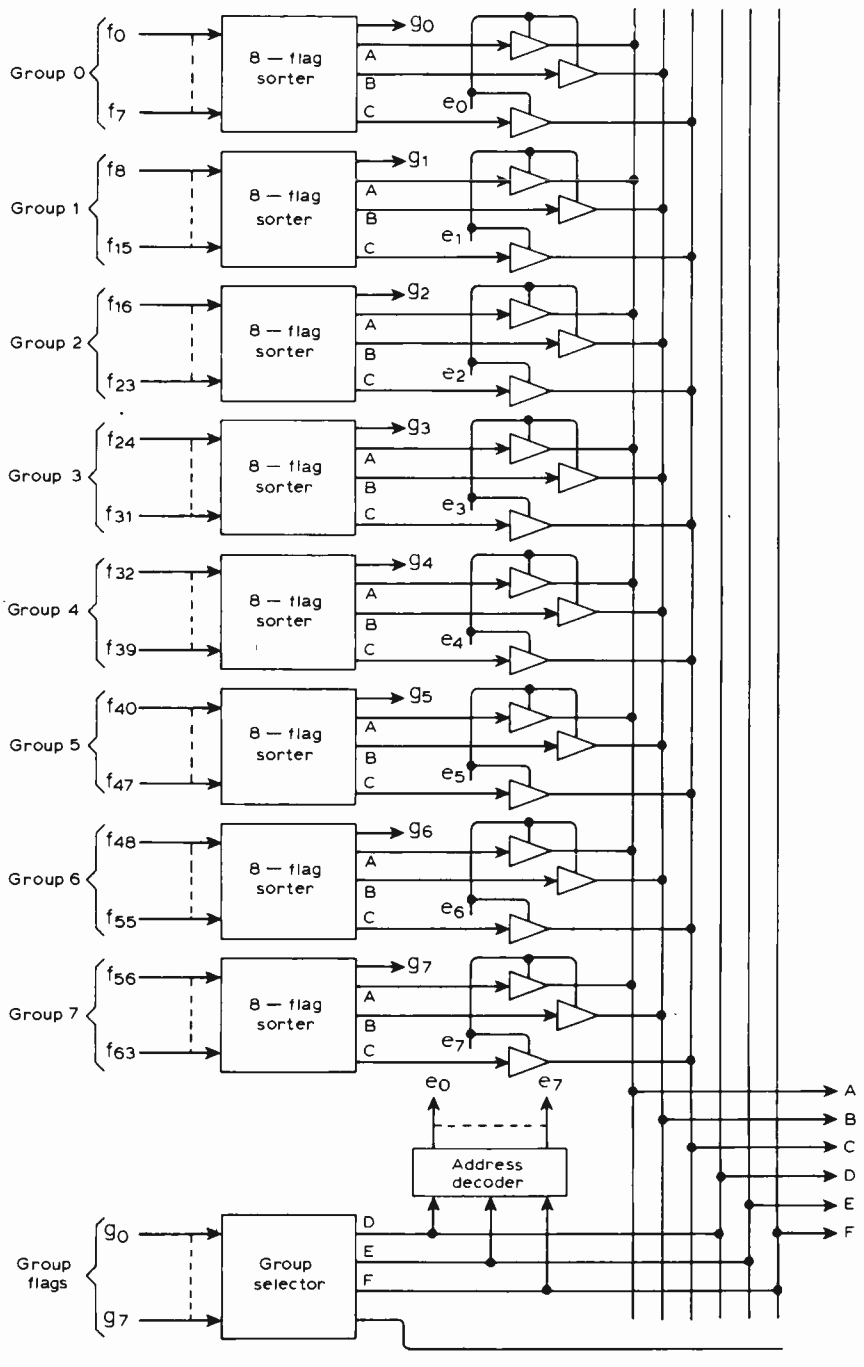


Fig. 14. 64-flag modular sorter.