power supply voltage for TTL circuits is 5 volts, and the two logic levels are approximately 0 and 3.5 volts.

ECL

The emitter-coupled logic (ECL) family provides the highest-speed digital circuits in integrated form. ECL is used in systems such as supercomputers and signal processors where high speed is essential. The transistors in ECL gates operate in a nonsaturated state, a condition that allows the achievement of propagation delays of 1 to 2 nanoseconds.

MOS

The metal-oxide semiconductor (MOS) is a unipolar transistor that depends on the flow of only one type of carrier, which may be electrons (n-channel) or holes (p-channel). This is in contrast to the bipolar transistor used in TTL and ECL gates, where both carriers exist during normal operation. A p-channel MOS is referred to as PMOS and an n-channel as NMOS. NMOS is the one that is commonly used in circuits with only one type of MOS transistor.

CMOS

The complementary MOS (CMOS) technology uses PMOS and NMOS transistors connected in a complementary fashion in all circuits. The most important advantages of CMOS over bipolar are the high packing density of circuits, a simpler processing technique during fabrication, and a more economical operation because of low power consumption.

Because of their many advantages, integrated circuits are used exclusively to provide various digital components needed in the design of computer systems. To understand the organization and design of digital computers it is very important to be familiar with the various components encountered in integrated circuits. For this reason, the most basic components are introduced in this chapter with an explanation of their logical properties. These components provide a catalog of elementary digital functional units commonly used as basic building blocks in the design of digital computers.

2-2 Decoders

Discrete quantities of information are represented in digital computers with binary codes. A binary code of n bits is capable of representing up to 2^n distinct elements of the coded information. A decoder is a combinational circuit that converts binary information from the n coded inputs to a maximum of 2^n unique outputs. If the n-bit coded information has unused bit combinations, the decoder may have less than 2^n outputs.

decoder

The decoders presented in this section are called n-to-m-line decoders, where $m \leq 2^n$. Their purpose is to generate the 2^n (or fewer) binary combinations of the n input variables. A decoder has n inputs and m outputs and is also referred to as an $n \times m$ decoder.

The logic diagram of a 3-to-8-line decoder is shown in Fig. 2-1. The three data inputs, A_0, A_1, and A_2, are decoded into eight outputs, each output

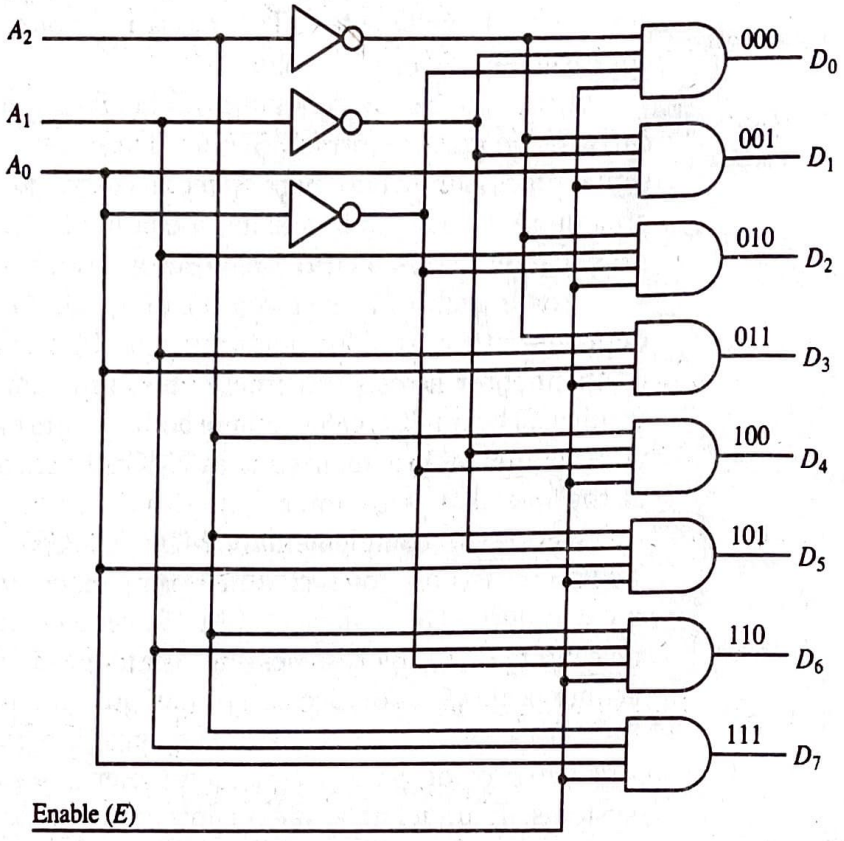


Figure 2-1 3-to-8-line decoder.

representing one of the combinations of the three binary input variables. The three inverters provide the complement of the inputs, and each of the eight AND gates generates one of the binary combination. A particular application of this decoder is a binary-to-octal conversion. The input variables represent a binary number and the outputs represent the eight digits of the octal number system. However, a 3-to-8-line decoder can be used for decoding any 3-bit code to provide eight outputs, one for each combination of the binary code.

Enable input

Commercial decoders include one or more enable inputs to control the operation of the circuit. The decoder of Fig. 2-1 has one enable input, E. The decoder is enabled when E is equal to 1 and disabled when E is equal to 0.

The operation of the decoder can be clarified using the truth table listed in Table 2-1. When the enable input E is equal to 0, all the outputs are equal to 0 regardless of the values of the other three data inputs. The three $\times$'s in the table designate don't-care conditions. When the enable input is equal to 1, the decoder operates in a normal fashion. For each possible input combination, there are seven outputs that are equal to 0 and only one that is equal to 1. The output variable whose value is equal to 1 represents the octal number equivalent of the binary number that is available in the input data lines.

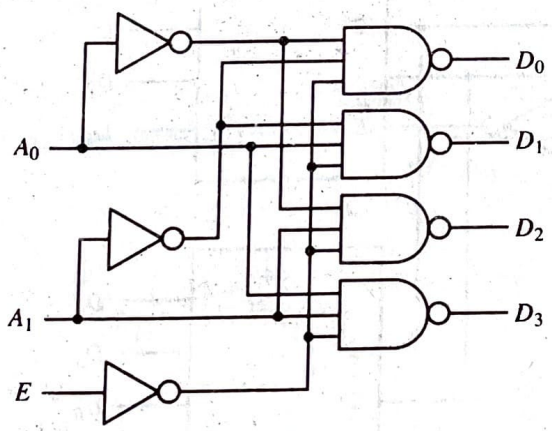
TABLE 2-1 Truth Table for 3-to-8-Line Decoder

Enable	Inputs			Outputs							
E	A_2	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	×	×	×	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

NAND Gate Decoder

Some decoders are constructed with NAND instead of AND gates. Since a NAND gate produces the AND operation with an inverted output, it becomes more economical to generate the decoder outputs in their complement form. A 2-to-4-line decoder with an enable input constructed with NAND gates is shown in Fig. 2-2. The circuit operates with complemented outputs and a complemented enable input E. The decoder is enabled when E is equal to 0. As indicated by the truth table, only one output is equal to 0 at any given time; the other three outputs are equal to 1. The output whose value is equal to 0 represents the equivalent binary number in inputs A_1 and A_0. The circuit is disabled when E is equal to 1, regardless of the values of the other two inputs.

Figure 2-2 2-to-4-line decoder with NAND gates.



(a) Logic diagram

E	A_1	A_0	D_0	D_1	D_2	D_3
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	×	×	1	1	1	1

(b) Truth table

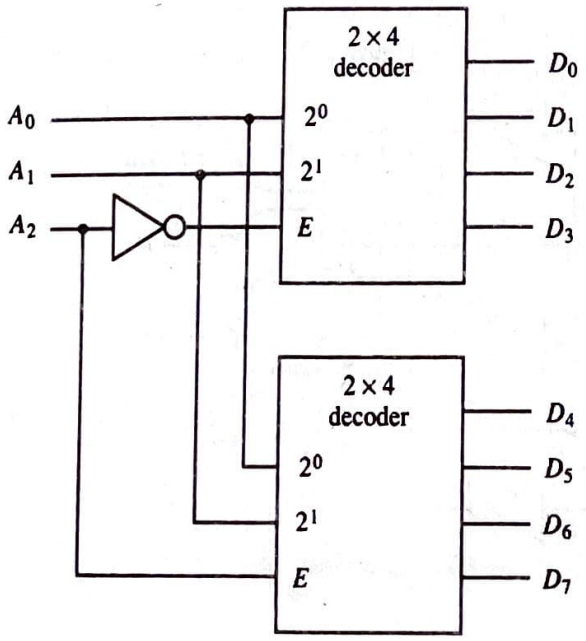
When the circuit is disabled, none of the outputs are selected and all outputs are equal to 1. In general, a decoder may operate with complemented or uncomplemented outputs. The enable input may be activated with a 0 or with a 1 signal level. Some decoders have two or more enable inputs that must satisfy a given logic condition in order to enable the circuit.

Decoder Expansion

There are occasions when a certain-size decoder is needed but only smaller sizes are available. When this occurs it is possible to combine two or more decoders with enable inputs to form a larger decoder. Thus if a 6-to-64-line decoder is needed, it is possible to construct it with four 4-to-16-line decoders.

Figure 2-3 shows how decoders with enable inputs can be connected to form a larger decoder. Two 2-to-4-line decoders are combined to achieve a 3-to-8-line decoder. The two least significant bits of the input are connected to both decoders. The most significant bit is connected to the enable input of one decoder and through an inverter to the enable input of the other decoder. It is assumed that each decoder is enabled when its E input is equal to 1. When E is equal to 0, the decoder is disabled and all its outputs are in the 0 level. When $A_2 = 0$, the upper decoder is enabled and the lower is disabled. The lower decoder outputs become inactive with all outputs at 0. The outputs of the upper decoder generate outputs D_0 through D_3, depending on the values of A_1 and A_0 (while $A_2 = 0$). When $A_2 = 1$, the lower decoder is enabled and the upper is disabled. The lower decoder output generates the binary equivalent D_4 through D_7 since these binary numbers have a 1 in the A_2 position.

Figure 2-3 A 3 × 8 decoder constructed with two 2 × 4 decoders.



The example demonstrates the usefulness of the enable input in decoders or any other combinational logic component. Enable inputs are a convenient feature for interconnecting two or more circuits for the purpose of expanding the digital component into a similar function but with more inputs and outputs.

Encoders

An encoder is a digital circuit that performs the inverse operation of a decoder. An encoder has 2^n (or less) input lines and n output lines. The output lines generate the binary code corresponding to the input value. An example of an encoder is the octal-to-binary encoder, whose truth table is given in Table 2-2. It has eight inputs, one for each of the octal digits, and three outputs that generate the corresponding binary number. It is assumed that only one input has a value of 1 at any given time; otherwise, the circuit has no meaning.

TABLE 2-2 Truth Table for Octal-to-Binary Encoder

Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

The encoder can be implemented with OR gates whose inputs are determined directly from the truth table. Output $A_0 = 1$ if the input octal digit is 1 or 3 or 5 or 7. Similar conditions apply for the other two outputs. These conditions can be expressed by the following Boolean functions:

$$A_0 = D_1 + D_3 + D_5 + D_7$$
$$A_1 = D_2 + D_3 + D_6 + D_7$$
$$A_2 = D_4 + D_5 + D_6 + D_7$$

The encoder can be implemented with three OR gates.

2-3 Multiplexers

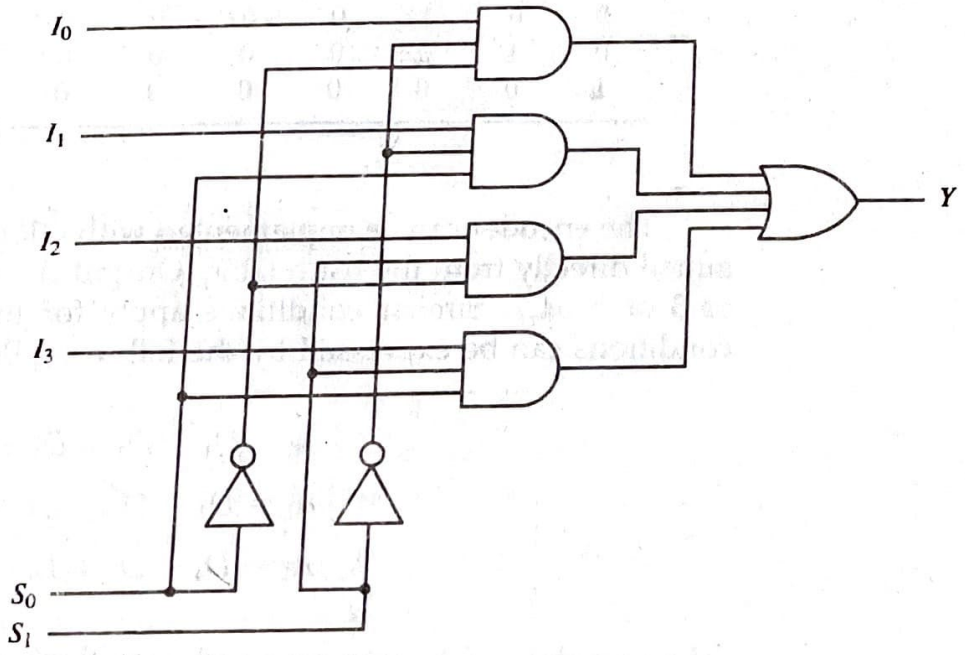
multiplexer

A multiplexer is a combinational circuit that receives binary information from one of 2^n input data lines and directs it to a single output line. The selection of a particular input data line for the output is determined by a set of selection inputs. A 2^n-to-1 multiplexer has 2^n input data lines and n input selection lines whose bit combinations determine which input data are selected for the output.

A 4-to-1-line multiplexer is shown in Fig. 2-4. Each of the four data inputs I_0 through I_3 is applied to one input of an AND gate. The two selection inputs S_1 and S_0 are decoded to select a particular AND gate. The outputs of the AND gates are applied to a single OR gate to provide the single output. To demonstrate the circuit operation, consider the case when $S_1S_0 = 10$. The AND gate associated with input I_2 has two of its inputs equal to 1. The third input of the gate is connected to I_2. The other three AND gates have at least one input equal to 0, which makes their outputs equal to 0. The OR gate output is now equal to the value of I_2, thus providing a path from the selected input to the output.

The 4-to-1 line multiplexer of Fig. 2-4 has six inputs and one output. A truth table describing the circuit needs 64 rows since six input variables can have 2^6 binary combinations. This is an excessively long table and will not be shown here. A more convenient way to describe the operation of multiplexers is by means of a function table. The function table for the multiplexer is shown in Table 2-3. The table demonstrates the relationship between the four data inputs and the single output as a function of the selection inputs S_1 and S_0.

Figure 2-4 4-to-1-line multiplexer.



data selector

When the selection inputs are equal to 00, output Y is equal to input I_0. When the selection inputs are equal to 01, input I_1 has a path to output Y, and similarly for the other two combinations. The multiplexer is also called a *data selector*, since it selects one of many data inputs and steers the binary information to the output.

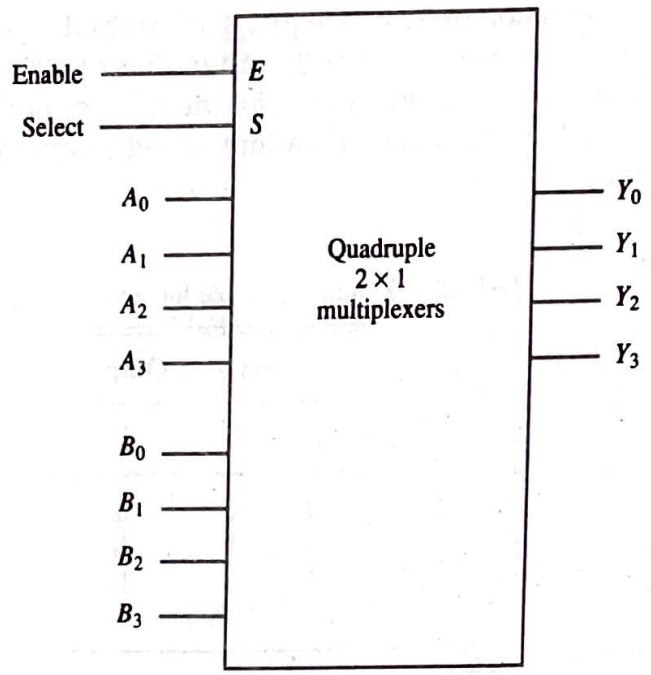
TABLE 2-3 Function Table for 4-to-1-Line Multiplexer

Select		Output
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

The AND gates and inverters in the multiplexer resemble a decoder circuit, and indeed they decode the input selection lines. In general, a 2^n-to-1-line multiplexer is constructed from an n-to-2^n decoder by adding to it 2^n input lines, one from each data input. The size of the multiplexer is specified by the number 2^n of its data inputs and the single output. It is then implied that it also contains n input selection lines. The multiplexer is often abbreviated as MUX.

As in decoders, multiplexers may have an enable input to control the operation of the unit. When the enable input is in the inactive state, the outputs are disabled, and when it is in the active state, the circuit functions as a normal multiplexer. The enable input is useful for expanding two or more multiplexers to a multiplexer with a larger number of inputs.

In some cases two or more multiplexers are enclosed within a single integrated circuit package. The selection and the enable inputs in multiple-unit construction are usually common to all multiplexers. As an illustration, the block diagram of a quadruple 2-to-1-line multiplexer is shown in Fig. 2-5. The circuit has four multiplexers, each capable of selecting one of two input lines. Output Y_0 can be selected to come from either input A_0 or B_0. Similarly, output Y_1 may have the value of A_1 or B_1, and so on. One input selection line S selects one of the lines in each of the four multiplexers. The enable input E must be active for normal operation. Although the circuit contains four multiplexers, we can also think of it as a circuit that selects one of two 4-bit data lines. As shown in the function table, the unit is enabled when $E = 1$. Then, if $S = 0$, the four A inputs have a path to the four outputs. On the other hand, if $S = 1$, the four B inputs are applied to the outputs. The outputs have all 0's when $E = 0$, regardless of the values of S.



(a) Block diagram

E	S	Y
0	×	All 0's
1	0	A
1	1	B

(b) Function table

Figure 2-5 Quadruple 2-to-1 line multiplexers.

2-4 Registers

A register is a group of flip-flops with each flip-flop capable of storing one bit of information. An n-bit register has a group of n flip-flops and is capable of storing any binary information of n bits. In addition to the flip-flops, a register may have combinational gates that perform certain data-processing tasks. In its broadest definition, a register consists of a group of flip-flops and gates that effect their transition. The flip-flops hold the binary information and the gates control when and how new information is transferred into the register.

Various types of registers are available commercially. The simplest register is one that consists only of flip-flops, with no external gates. Figure 2-6 shows such a register constructed with four D flip-flops. The common clock input triggers all flip-flops on the rising edge of each pulse, and the binary data available at the four inputs are transferred into the 4-bit register. The four outputs can be sampled at any time to obtain the binary information stored in the register. The *clear* input goes to a special terminal in each flip-flop. When this input goes to 0, all flip-flops are reset asynchronously. The clear input is useful for clearing the register to all 0's prior to its clocked operation. The clear input must be maintained at logic 1 during normal clocked operation. Note that the clock signal enables the D input but that the clear input is independent of the clock.

register load

The transfer of new information into a register is referred to as *loading* the register. If all the bits of the register are loaded simultaneously with a common

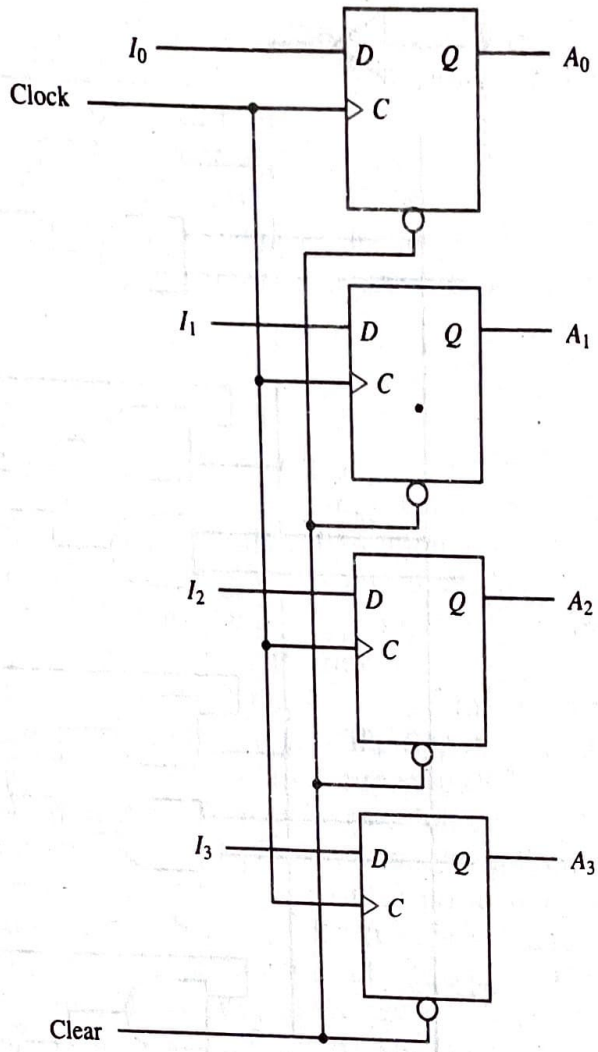


Figure 2-6 4-bit register.

clock pulse transition, we say that the loading is done in parallel. A clock transition applied to the C inputs of the register of Fig. 2-6 will load all four inputs I_0 through I_3 in parallel. In this configuration, the clock must be inhibited from the circuit if the content of the register must be left unchanged.

Register with Parallel Load

Most digital systems have a master clock generator that supplies a continuous train of clock pulses. The clock pulses are applied to all flip-flops and registers in the system. The master clock acts like a pump that supplies a constant beat to all parts of the system. A separate control signal must be used to decide which specific clock pulse will have an effect on a particular register.

A 4-bit register with a load control input that is directed through gates and into the *D* inputs is shown in Fig. 2-7. The C inputs receive clock pulses at all times. The buffer gate in the clock input reduces the power requirement

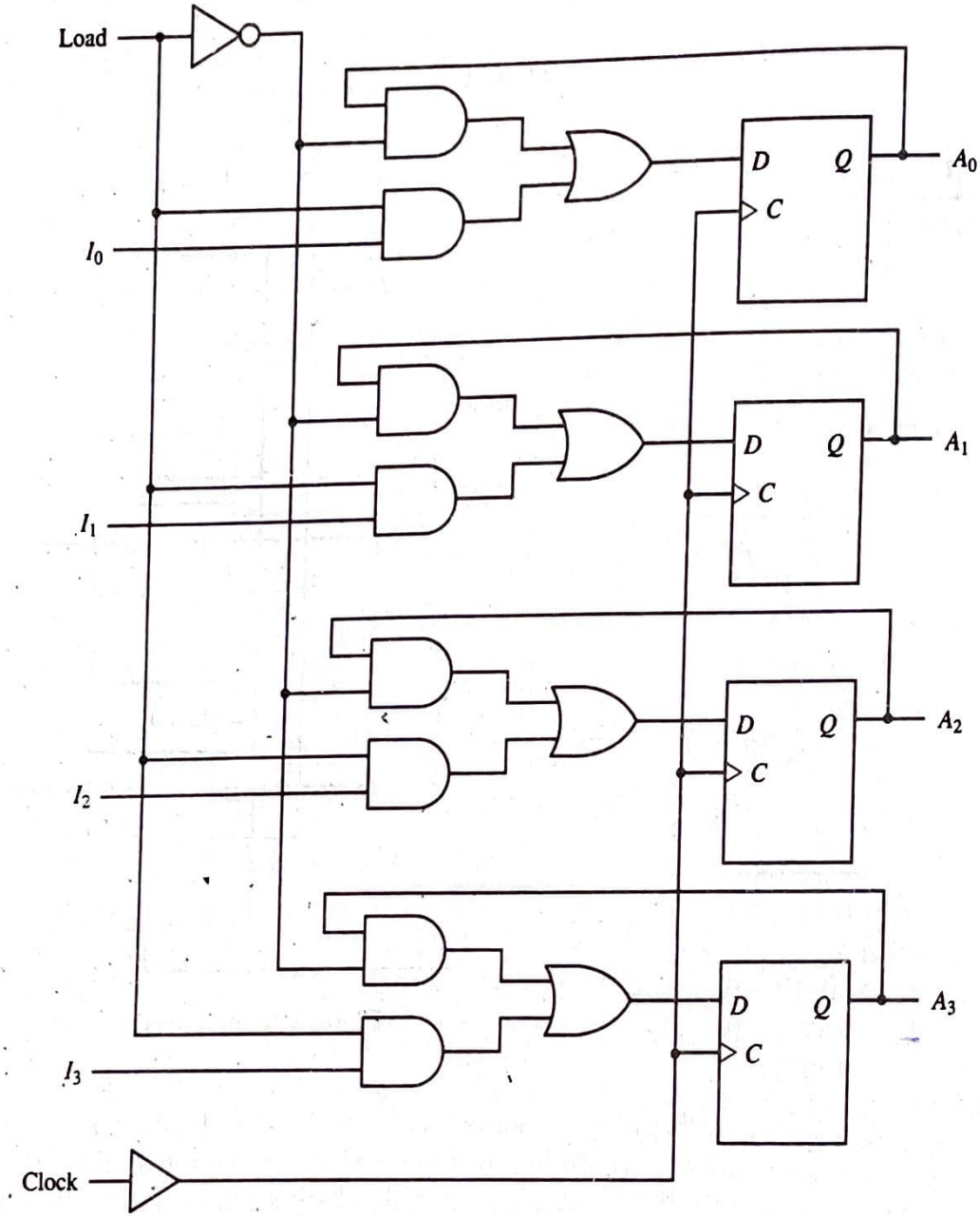


Figure 2-7 4-bit register with parallel load.

from the clock generator. Less power is required when the clock is connected to only one input gate instead of the power consumption that four inputs would have required if the buffer were not used.

load input

The load input in the register determines the action to be taken with each clock pulse. When the load input is 1, the data in the four inputs are transferred into the register with the next positive transition of a clock pulse. When the load input is 0, the data inputs are inhibited and the *D* inputs of the flip-flops are connected to their outputs. The feedback connection from output to input is necessary because the *D* flip-flop does not have a "no change" condition.

With each clock pulse, the *D* input determines the next state of the output. To leave the output unchanged, it is necessary to make the *D* input equal to the present value of the output.

Note that the clock pulses are applied to the *C* inputs at all times. The load input determines whether the next pulse will accept new information or leave the information in the register intact. The transfer of information from the inputs into the register is done simultaneously with all four bits during a single pulse transition.

2-5 Shift Registers

A register capable of shifting its binary information in one or both directions is called a shift register. The logical configuration of a shift register consists of a chain of flip-flops in cascade, with the output of one flip-flop connected to the input of the next flip-flop. All flip-flops receive common clock pulses that initiate the shift from one stage to the next.

serial input

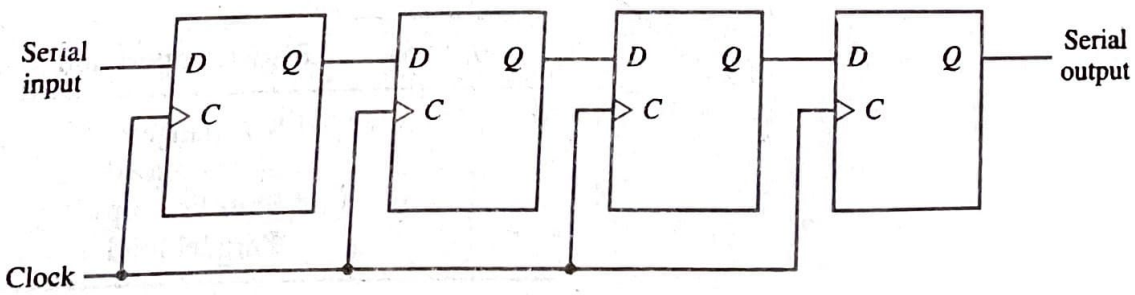
The simplest possible shift register is one that uses only flip-flops, as shown in Fig. 2-8. The output of a given flip-flop is connected to the *D* input of the flip-flop at its right. The clock is common to all flip-flops. The *serial input* determines what goes into the leftmost position during the shift. The *serial output* is taken from the output of the rightmost flip-flop.

Sometimes it is necessary to control the shift so that it occurs with certain clock pulses but not with others. This can be done by inhibiting the clock from the input of the register if we do not want it to shift. When the shift register of Fig 2-8 is used, the shift can be controlled by connecting the clock to the input of an AND gate, and a second input of the AND gate can then control the shift by inhibiting the clock. However, it is also possible to provide extra circuits to control the shift operation through the *D* inputs of the flip-flops rather than the clock input.

Bidirectional Shift Register with Parallel Load

A register capable of shifting in one direction only is called a unidirectional shift register. A register that can shift in both directions is called a bidirectional shift register. Some shift registers provide the necessary input and output terminals

Figure 2-8 4-bit shift register.



for parallel transfer. The most general shift register has all the capabilities listed below. Others may have some of these capabilities, with at least one shift operation.

1. An input for clock pulses to synchronize all operations.
2. A shift-right operation and a serial input line associated with the shift-right.
3. A shift-left operation and a serial input line associated with the shift-left.
4. A parallel load operation and n input lines associated with the parallel transfer.
5. n parallel output lines.
6. A control state that leaves the information in the register unchanged even though clock pulses are applied continuously.

A 4-bit bidirectional shift register with parallel load is shown in Fig. 2-9. Each stage consists of a D flip-flop and a 4×1 multiplexer. The two selection inputs S_1 and S_0 select one of the multiplexer data inputs for the D flip-flop. The selection lines control the mode of operation of the register according to the function table shown in Table 2-4. When the mode control $S_1S_0 = 00$, data input 0 of each multiplexer is selected. This condition forms a path from the output of each flip-flop into the input of the same flip-flop. The next clock transition transfers into each flip-flop the binary value it held previously, and no change of state occurs. When $S_1S_0 = 01$, the terminal marked 1 in each multiplexer has a path to the D input of the corresponding flip-flop. This causes a shift-right operation, with the serial input data transferred into flip-flop A_0 and the content of each flip-flop A_{i-1} transferred into flip-flop A_i for $i = 1, 2, 3$. When $S_1S_0 = 10$ a shift-left operation results, with the other serial input data going into flip-flop A_3 and the content of flip-flop A_{i+1} transferred into flip-flop A_i for $i = 0, 1, 2$. When $S_1S_0 = 11$, the binary information from each input I_0 through I_3 is transferred into the corresponding flip-flop, resulting in a parallel load operation. Note that the way the diagram is drawn, the shift-right operation shifts the contents of the register in the down direction while the shift left operation causes the contents of the register to shift in the upward direction.

TABLE 2-4 Function Table for Register of Fig. 2-9

Mode control		
S_1	S_0	Register operation
0	0	No change
0	1	Shift right (down)
1	0	Shift left (up)
1	1	Parallel load

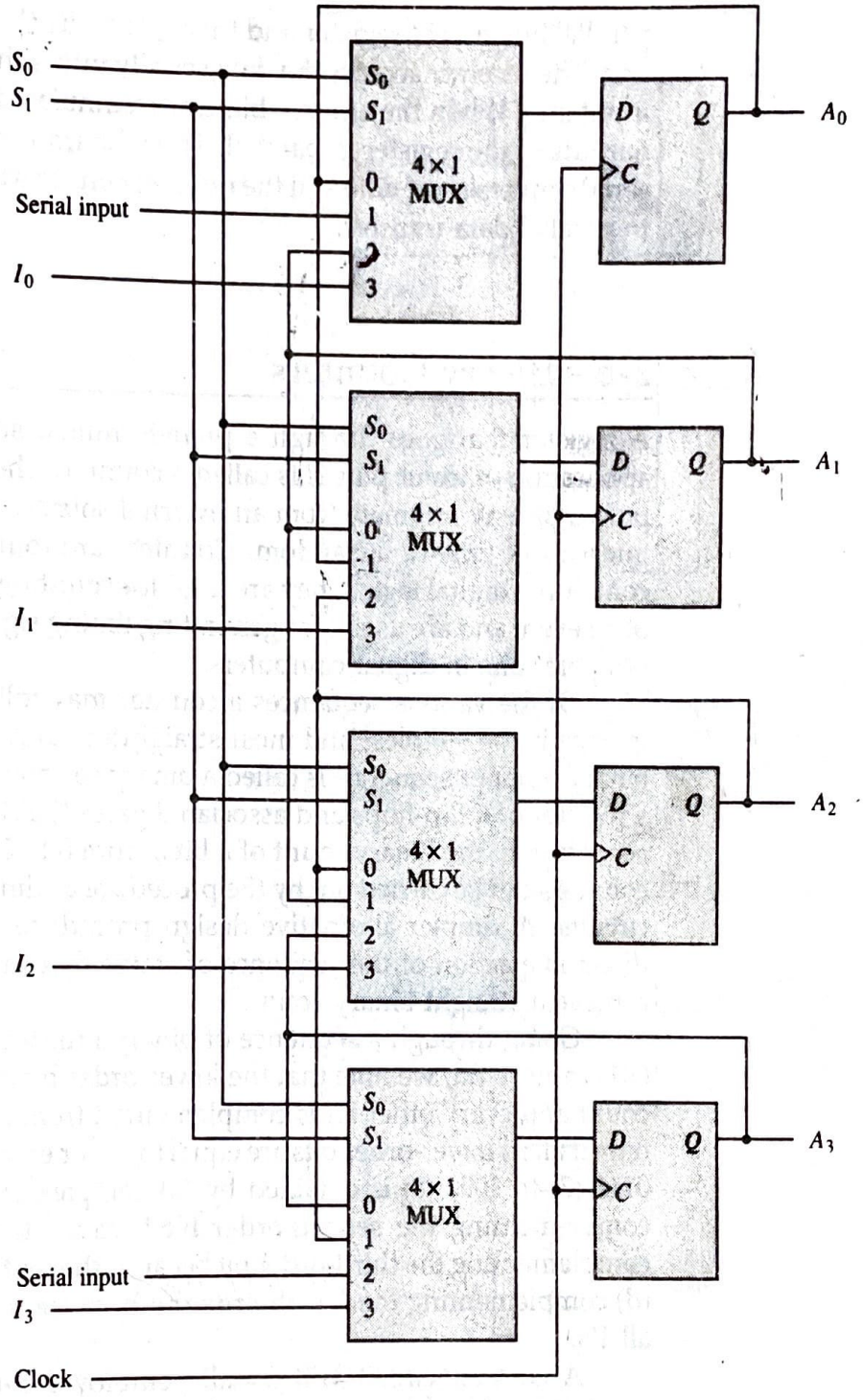


Figure 2-9 Bidirectional shift register with parallel load.

Shift registers are often used to interface digital systems situated remotely from each other. For example, suppose that it is necessary to transmit an n-bit quantity between two points. If the distance between the source and the destination is too far, it will be expensive to use n lines to transmit the n bits in parallel. It may be more economical to use a single line and transmit the information serially one bit at a time. The transmitter loads the n-bit data in

parallel into a shift register and then transmits the data from the serial output line. The receiver accepts the data serially into a shift register through its serial input line. When the entire n bits are accumulated they can be taken from the outputs of the register in parallel. Thus the transmitter performs a parallel-to-serial conversion of data and the receiver converts the incoming serial data back to parallel data transfer.

2-6 Binary Counters

A register that goes through a predetermined sequence of states upon the application of input pulses is called a counter. The input pulses may be clock pulses or may originate from an external source. They may occur at uniform intervals of time or at random. Counters are found in almost all equipment containing digital logic. They are used for counting the number of occurrences of an event and are useful for generating timing signals to control the sequence of operations in digital computers.

Of the various sequences a counter may follow, the straight binary sequence is the simplest and most straightforward. A counter that follows the binary number sequence is called a binary counter. An n-bit binary counter is a register of n flip-flops and associated gates that follows a sequence of states according to the binary count of n bits, from 0 to $2^n - 1$. The design of binary counters can be carried out by the procedure outlined in Sec. 1-7 for sequential circuits. A simpler alternative design procedure may be carried out from a direct inspection of the sequence of states that the register must undergo to achieve a straight binary count.

Going through a sequence of binary numbers such as 0000, 0001, 0010, 0011, and so on, we note that the lower-order bit is complemented after every count and every other bit is complemented from one count to the next if and only if all its lower-order bits are equal to 1. For example, the binary count from 0111 (7) to 1000 (8) is obtained by (a) complementing the low-order bit, (b) complementing the second-order bit because the first bit of 0111 is 1, (c) complementing the third-order bit because the first two bits of 0111 are 1's, and (d) complementing the fourth-order bit because the first three bits of 0111 are all 1's.

A counter circuit will usually employ flip-flops with complementing capabilities. Both T and JK flip-flops have this property. Remember that a JK flip-flop is complemented if both its J and K inputs are 1 and the clock goes through a positive transition. The output of the flip-flop does not change if $J = K = 0$. In addition, the counter may be controlled with an enable input that turns the counter on or off without removing the clock signal from the flip-flops.

Synchronous binary counters have a regular pattern, as can be seen from the 4-bit binary counter shown in Fig. 2-10. The C inputs of all flip-flops receive the common clock. If the count enable is 0, all J and K inputs are maintained

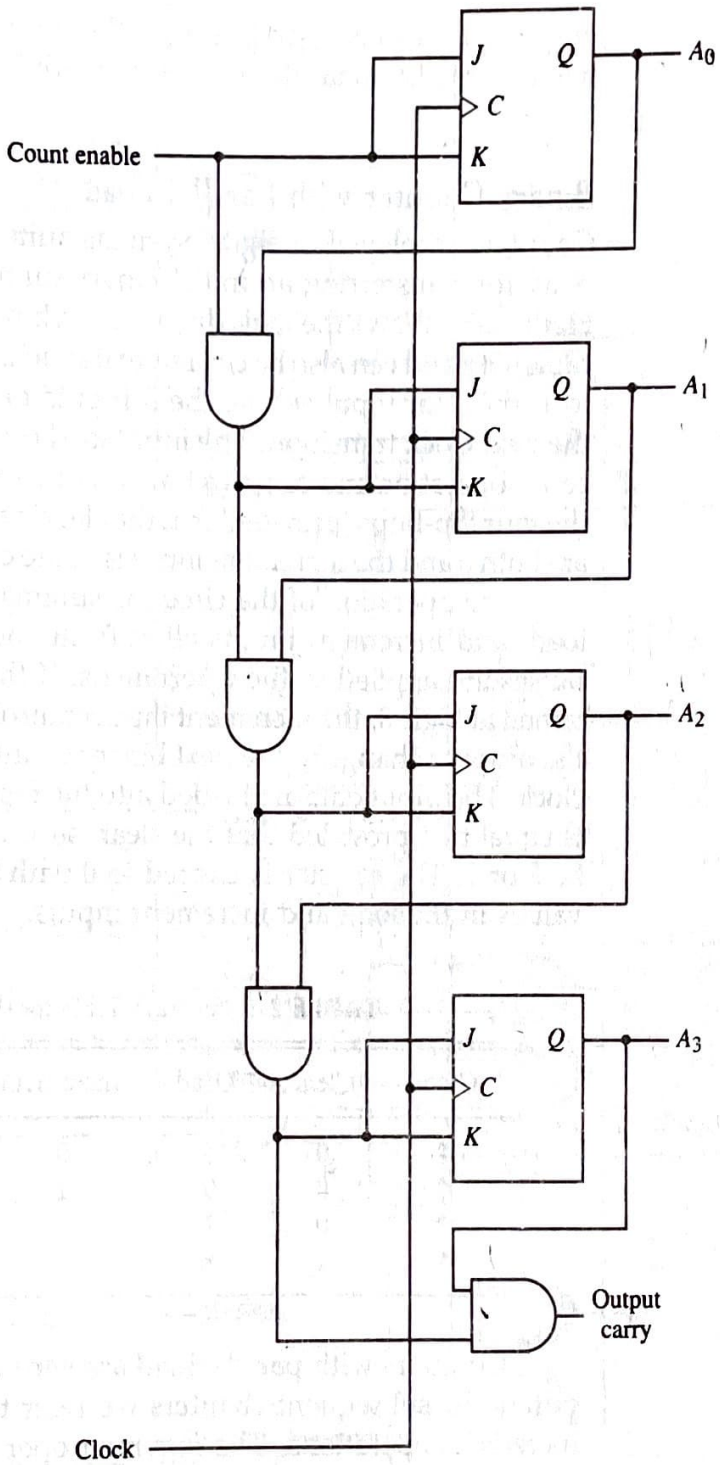


Figure 2-10 4-bit synchronous binary counter.

at 0 and the output of the counter does not change. The first stage A_0 is complemented when the counter is enabled and the clock goes through a positive transition. Each of the other three flip-flops are complemented when all previous least significant flip-flops are equal to 1 and the count is enabled. The chain of AND gates generate the required logic for the J and K inputs. The