Module 4

www.learnabout-electronics.org

Digital Electronics

4.0 Introduction to Combinational Logic

Binary Arithmetic

What you'll learn in Module 4

Section 4.0 Introduction.

Section 4.1 Binary Arithmetic Circuits.

- Half Adder.
- Full Adder.
- Parallel Adders.
- Logic Circuit Simulation.
- Twos Complement Overflow.
- Carry Look Ahead Adders.

Section 4.2 Data-Select & Multiplexing

- Basic data−select and multiplexing circuits.
- Multiplexing.
- Demultiplexing.
- Multi−bit multiplexers.
- Address Decoding.
- Multiplexer Simulation.

Section 4.3 Binary Comparators.

- Equality comparators.
- Magnitude comparators.
- 4−Bit magnitude comparators.
- Cascaded Comparators.

Section 4.4 Encoders and Decoders.

- Priority encoders.
- Diode matrix encoders.
- Logic decoders.
- 74 Series decoders.
- Address decoders.
- Simulating Encoders & Decoders.

Section 4.5 Combinational Logic Quiz

- Test your knowledge of Combinational Logic.

Combinational logic has many uses in electronic systems. It is used to carry out the essential arithmetic, not only in computers and calculators, but also in navigation systems, robots and many other types of automatic machinery.

However complex such calculations need to be, they all depend on some basic combinational logic circuits to carry out binary addition and subtraction. This arithmetic is discussed in Digital Electronics Module 1 'Number Systems', and a study of Module 1 will be a great help in understanding the techniques discussed here in Module 4.

Logical Decisions

Mathematics and logical decision making also uses combinational logic in the form of Comparators, described in Module 4.3. These circuits decide whether one value is the same, larger or smaller than another value.

Data Routing

After making a logical decision, data may need to be routed to different parts of the electronic system, this routing is controlled by more combinational logic circuits such as data selectors, multiplexers and demultiplexers, which are described in Module 4.2.

Encoding and Decoding

When connecting a logic circuit with the outside world, incoming information from a keyboard or other input device must be changed (encoded) into an appropriate binary form. Also before binary data produced by the digital system can be used by an output device, such as a display, it must be decoded into a form that can be used by the display. Encoders and decoders used for such jobs are also combinational logic circuits, and are described in Module 4.4

4.1 Binary Arithmetic Circuits

What you'll learn in Module 4.1

After studying this section, you should be able to:

Understand the operation of Binary Adder Circuits.

- Half Adder
- Full Adder.
- 4 Bit Parallel Adders.
- 8 Bit Adder/Subtractors.
- Twos Complement Overflow.
- Carry Look Ahead Adders.

Use Free Software to Simulate Logic Circuit operation.

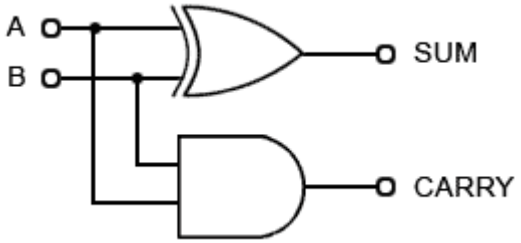


Fig.4.1.1 The Half Adder

Table 4.1.1			
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

The Half Adder

Binary arithmetic is carried out by combinational logic circuits, the simplest of which is the half adder, shown in Fig. 4.1.1. This circuit consists, in its most basic form of two gates, an XOR gate that produces a logic 1 output whenever A is 1 and B is 0 or when B is 1 and A is 0. The AND gate produces a logic 1 at the carry output when both A and B are 1. The half adder truth table is shown in Table 4.1.1 and describes the result of binary addition.

$$1 \text{ plus } 0 = 1_2 \quad (1_{10})$$

and

$$1 \text{ plus } 1 = 10_2 \quad (2_{10})$$

The half adder is fine for adding two 1-bit numbers together, but for binary numbers containing several bits, a carry may be produced at some time (as a result of adding 1 and 1) that must be added to the next column. As the half adder has only two inputs it cannot add in a carry bit from a previous column, so it is not practical for anything other than 1-bit additions.

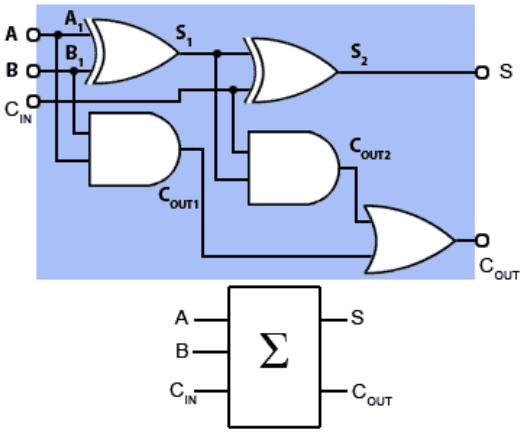


Fig. 4.1.2 The Full Adder Circuit.

The Full Adder

When 2 or more bits are to be added, the circuit used is the Full Adder, shown in Fig 4.1.2, (blue background) together with its simplified block diagram symbol. This circuit simply comprises two half adders, the sum of A and B from the first half adder is used as input S_1 on the second half adder, which now produces a sum of the first half adder sum (S_1) plus any 'carry in' from the C_{IN} terminal. Any carries produced by the two half adders are then 'ORed' together to produce a single C_{OUT} output. The truth table for the circuit is given in Table 4.1.2.

Table 4.1.2							
Half Adder$_1$					Half Adder$_2$		
A_1	B_1	C_{IN}	C_{OUT1}	S_1	C_{OUT2}	S_2	C_{OUT}
			$A \cdot B$	$A \oplus B$	$A_2 \cdot B_2$	$A_2 \oplus C_{IN}$	$C_{OUT1}+C_{OUT2}$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	1	0	1	0
0	1	1	0	1	1	0	1
1	0	0	0	1	0	1	0
1	0	1	0	1	1	0	1
1	1	0	1	0	0	0	1
1	1	1	1	0	0	1	1

Parallel Adders

Even the full adder is only adding two single bit binary numbers, but full adders may be combined to form parallel adders, which will add two multi bit numbers. Parallel adders can be built in several forms to add multi-bit binary numbers, each bit of the parallel adder using a single full adder circuit. As parallel adder circuits would look quite complex if drawn showing all the individual gates, it is common to replace the full adder schematic diagram with a simplified block diagram version.

4 Bit Parallel Adder

Fig 4.1.3 illustrates how a number of full adders can be combined to make a parallel adder, also called a 'Ripple Carry Adder' because of the way that any carry appearing at the carry in input (C_{IN}) or produced when adding any of the 4-bit inputs, 'ripples' along the adder stages until a final carry out appears at the carry out output (C_{OUT}) of full adder for bit A_3+B_3.

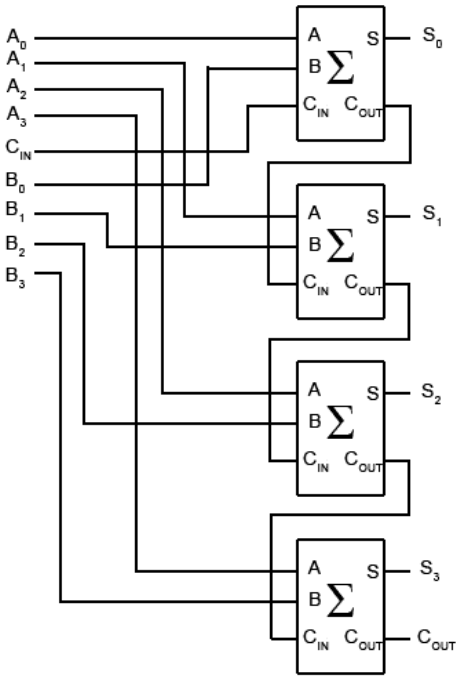


Fig 4.1.3 4-Bit Parallel Adder

8 Bit Parallel Adder/Subtractor

To carry out arithmetic however, it is also necessary to be able to subtract. A further development of the parallel adder is shown in Fig.4.1.5. This is an 8-bit parallel adder/subtractor. This circuit adds in the same way as the adder in Fig. 4.1.3 but subtracts using the twos complement method described in Digital Electronics Module 1.5 (Ones and Twos Complement).

When subtraction is required, the control input is set to logic 1, which causes the bit at any particular B input to be complemented by an XOR gate before being fed to input B of the full adder circuit.

Twos complement subtraction in an 8-bit adder/subtractor requires that the 8-bit number at input B is complemented (inverted) and has 1 added to it, before being added to the 8-bit number at input A. The result of this will be an 8-bit number in twos complement format, i.e. with its value represented by the lower 7 bits (bit 0 to bit 6) and the sign represented by the most significant bit (bit 7). The logic 1 on the control input is therefore also fed to the first carry input of the adder to be included in the addition, which for subtraction is therefore:

$$\mathbf{A + \overline{B} + 1}$$

(Here + signifies addition rather than OR)

Alternatively, if addition of A and B is required, then the control input is at logic 0 and number B is fed to the adder without complementing.

How an XOR gate is used here to change the adder into a subtractor by inverting the B inputs can be seen from the truth table for an XOR gate, shown in Table 4.1.3 in Fig. 4.1.6. Notice that if input A, (used as the CONTROL input) of the XOR gate is at logic 0, then the XOR gate selects input B, but if input A is logic 1, then it selects the inverse of input B (i.e. $\overline{B}$).

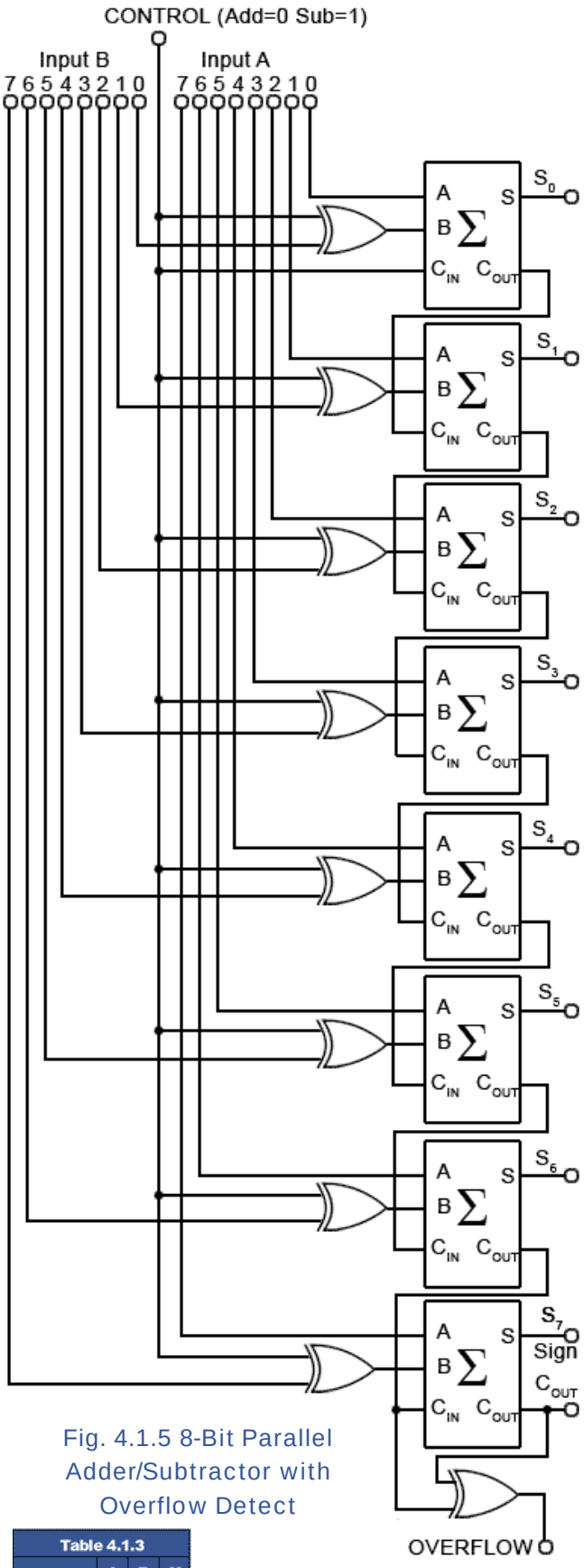


Fig. 4.1.5 8-Bit Parallel Adder/Subtractor with Overflow Detect

Table 4.1.3	A	B	X
X = B	0	0	0
	0	1	1
X = $\overline{B}$	1	0	1
	1	1	0

Fig. 4.1.6 XOR Gate as Data Selector

Twos Complement Overflow

The 8-bit adder/subtractor illustrated in Fig. 4.1.5 is designed to add or subtract 8-bit binary numbers using twos complement notation. In this system the most significant bit (bit 7) is not used as part of the number's value, it is used to indicate the sign of the number (0 = positive and 1 = negative).

No matter what the word size of a digital system (8-bits 16-bits 32-bits etc.), a given number of bits can only process numbers up to a maximum value that can be held in its designed word length.

During arithmetical operations it is possible that adding two numbers (with either positive or negative values) that are both within the system's limit, can produce a result that is too large for the system's word length to hold.

For example, in a twos complement adder such as shown in Fig. 4.1.5, when adding either positive or negative 7-bit values, the result could be larger than 7 bits can accommodate. Therefore the result will need to occupy one extra bit, which means that the calculated value will 'overflow' into bit eight, losing a major part (128_{10}) of the value and changing the sign of the result.

To overcome this problem, it is necessary first to detect that an overflow problem has occurred, and then to solve it either by using additional circuits or, in computing, by implementing a corrective routine in software.

Fortunately there is a quite simple method for detecting when an overflow occurs. As shown in Fig. 4.1.5 the overflow detection system consists of a single exclusive or (XOR) gate that takes its inputs from the carry in and carry out connections of the bit 7 (sign bit) adder.

When the carry in (C_{IN}) and carry out (C_{OUT}) bits of this adder are examined, it can be seen that if an overflow has occurred C_{IN} and C_{OUT} will be different, but if no overflow has occurred they will be identical.

Adding Two Positive (In Range) Numbers

Table 4.1.4 shows the effect of adding two positive values where the sum is within the range that can be held in 7 bits ($<=127_{10}$). The result of adding two positive numbers has produced a correct positive result with no carry and no overflow.

Table 4.1.4

	C_{OUT}	7	6	5	4	3	2	1	0
Carry >	0	0	0	0	1	0	0	0	
$+73_{10}$	A	0	1	0	0	1	0	0	1
$+42_{10}$	B	0	0	1	0	1	0	1	0
$+115_{10}$	Sum	0	1	1	1	0	0	1	1

Twos Complement Subtraction

Table 4.1.5 shows a twos complement subtraction performed by adding a negative number to a positive number. The result is 31_{10} (within the range 0 to $+127_{10}$), the sign bit is 0 indicating positive result, C_{IN} and C_{OUT} are both 1, so no overflow is detected and the carry bit will be discarded.

Table 4.1.5

	C_{OUT}	7	6	5	4	3	2	1	0
Carry >	1	1	0	0	0	0	0	0	
73_{10}	A	0	1	0	0	1	0	0	1
-42_{10}	B	1	1	0	1	0	1	1	0
31_{10}	Sum	0	0	0	1	1	1	1	1

Adding Twos Complement Negative Numbers

Table 4.1.6 shows the effect of adding two negative values where the sum is less than $+127_{10}$ therefore a correct negative result of -73_{10} (in twos complement notation) has been obtained. Both C_{IN} and C_{OUT} are logic 1 and no overflow will be signalled. As only 8-bit calculations are being considered, the carry will be discarded.

Table 4.1.6

	C_{OUT}	7	6	5	4	3	2	1	0
Carry >	1	1	0	0	0	0	0	0	
-63_{10}	A	1	1	0	0	0	0	0	1
-10_{10}	B	1	1	1	1	0	1	1	0
-73_{10}	Sum	1	0	1	1	0	1	1	1

Out of Range Result Causes Overflow

When the addition of two positive numbers shown in Table. 4.1.7 results in a sum greater than $+127_{10}$ the sign bit is changed from 0 to 1, incorrectly signifying a negative result. As the 'carry in' from bit 6 to bit 7 is 1 and the 'carry out' from bit 7 into the Carry bit is 0 an overflow is detected indicating an incorrect answer.

Table 4.1.7

	C_{OUT}	7	6	5	4	3	2	1	0
Carry >	0	1	1	0	0	0	1	0	
115_{10}	A	0	1	1	1	0	0	1	1
42_{10}	B	0	0	1	0	1	0	1	0
157_{10} ~~-99_{10}~~	Sum	1	0	0	1	1	1	0	1

Notice that if the result of 10011101_2 were to be considered as an unsigned binary value, the addition in Table 4.1.7 would be correct (157_{10}). However as the calculation is using twos complement notation, the answer of -99_{10} must be considered as wrong.

Out of Range Addition of Negative Values

Table 4.1.8 shows that adding two negative values can also produce a change in sign and a wrong twos complement result if it is greater than -128_{10}. In this case adding -63_{10} and -73_{10} should have produced a negative result of -136_{10} and not $+120_{10}$. To check this, the correct answer (although still with the wrong sign) could be obtained if, noting that an overflow had occurred, the answer was complemented and 1 added, giving an unsigned binary result of 10001000_2 which converts to $128 + 8 = 136_{10}$. Overflow errors can be corrected, but this would require either some additional electronics or a software action in response to the overflow signal.

Table 4.1.8

	C_{OUT}	7	6	5	4	3	2	1	0
Carry >	1	0	0	0	0	1	1	1	
-63_{10}	A	1	1	0	0	0	0	0	1
-73_{10}	B	1	0	1	1	0	1	1	1
-136_{10} ~~$+120_{10}$~~	Sum	0	1	1	1	1	0	0	0

Carry Look Ahead Adders

The adders described in this module are generally called Ripple Carry Adders because of the way that the carry bit is propagated from one stage of the adder to the next, rippling through the chain of full adders until the carry out is produced at the carry out pin of the final stage.

This process takes some time, which is proportional to the number of bits added. Although this may be a minor problem in small adders, with an increase in the number of bits in the binary words to be added, the time delay before the final carry out is produced becomes unacceptable.

To overcome this problem, IC manufacturers offer a range of 'Carry Look Ahead Adders' in which the addition and carry out are produced simultaneously. The system uses complex combinational logic to assess whether, at each individual adder a carry will be produced, based on the state of the A and B inputs to that stage, and the logic state of the carry in bit to the first stage.

Fig. 4.1.7 shows an arrangement for producing a carry out by splitting the full adder into a partial full adder (grey block), which has two additional outputs, a propagate (P) output that takes a logic 1 output whenever inputs A and B are 1,0 or 0,1 and a generate (G) output that will be logic 1 whenever the A and B inputs are at 1,1. Using this information it is possible to decide on the logic state of the carry out depending on a combination of the C_{IN} state and the A and B states.

In the carry generator (blue block), the P input is ANDed with the C_{IN} and ORed with the G input to produce a carry out. The carry out is fed to the successive adders in the normal way, but the C_{IN}, P and G signals are fed in parallel to the other adder stages, where the state of the carry out for each adder stage can be ascertained from the shared C_{IN} signal and the A and B states for the successive stages, depending on the input states at each stage, rather than

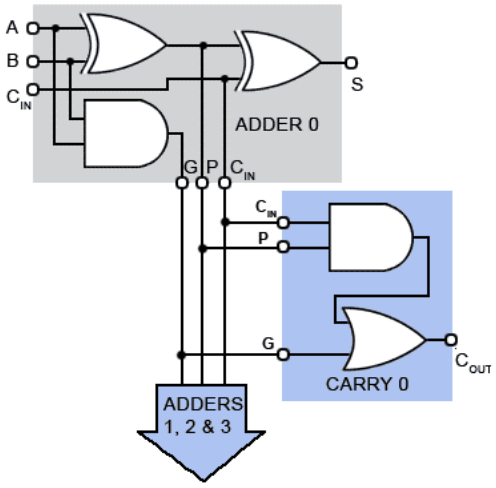


Fig. 4.1.7 First Stage of a Carry Look Ahead Adder

waiting for the calculations to complete at all the stages.

A generalised arrangement in block diagram form (Fig.4.1.8) shows the carry out (C_{OUT}) being produced by the parallel carry generator from the A and B input signals and the C_{IN} signal, rather than from the carry out of the final adder stage as in the ripple adders.

The use of look ahead adders is important in practical circuits, not only to speed up operation but because to have an adder that produces part of its answer (the sum) at one time, and another part of its answer (the carry out) at another time, would cause timing problems in other parts of the circuit.

A typical example of a Carry Look Ahead Adder is the MC14008B from ON Semiconductor.

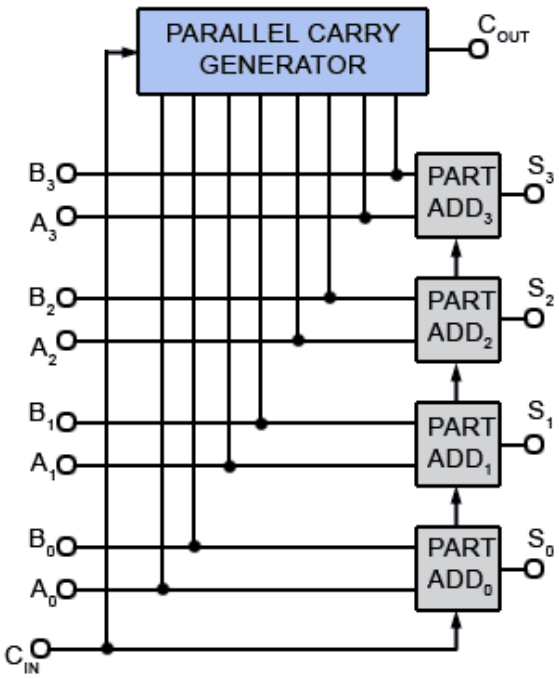


Fig. 4.1.8 Carry Look Ahead Adder Block Diagram

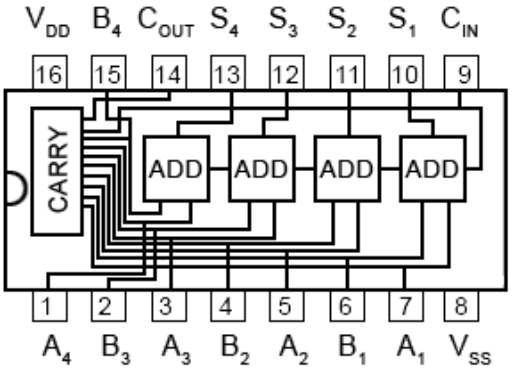


Fig 4.1.9 MC14008B Carry Look Ahead Adder

4.2 Data Selectors and Multiplexers

What you'll learn in Module 4.2

After studying this section, you should be able to:

Recognise uses for Data Select and Multiplexer Circuits.

Understand the operation of Data Select and Multiplexer Circuits.

- Basic Data Select (Multiplexer) Circuits.
- Multiplexing
- De-multiplexing

Understand the operation of Multi-Bit Multiplexers.

- Addressing
- Multiplexer Simulation.

Access Multiplexer IC Datasheets.

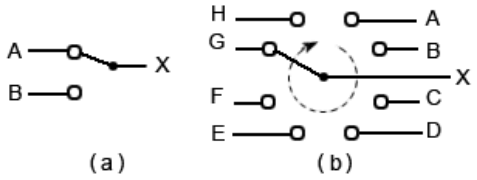


Fig. 4.2.1 Mechanical Selector switches

A simple way to connect multiple sources of information in analogue electronic systems is by using mechanical switches, such as those illustrated in Fig. 4.2.1. In example (a) a single pole double throw switch is used to select either input A or input B to be connected to output X. Example (b) shows a rotary selector switch that can multiplex any one of eight inputs to a single output.

In digital electronics, selecting multiple data sources can be performed by combinational logic circuits. Logic signals applied to one or more data select inputs initiate the selection of data, which may be steady logic levels or whole streams of digital information. Switching digital signals in this way is much faster and more reliable than using mechanical switch contacts. Digital data selectors and multiplexers are therefore a vital part of many digital systems. The names 'data selector' and 'multiplexer' are commonly interchanged, with multiplexers called data selectors and vice versa. If there is any difference, a circuit selecting between two inputs may be called a data selector, and more complex circuits combining multiple inputs into a single output, using various methods and existing in both digital and analogue forms, would be called multiplexers.

Basic Data Select (Multiplexer) Circuits

A simple data selector consisting of a single XOR gate was used in the 8 Bit Adder/Subtractor circuit shown in Figs. 4.1.5 and 4.1.6 in Module 4.1 to change the function of the circuit from addition to subtraction, but this was only required to select data B or its inverse $\overline{B}$. The circuit shown in Fig. 4.2.2 however can select either of two completely independent data inputs.

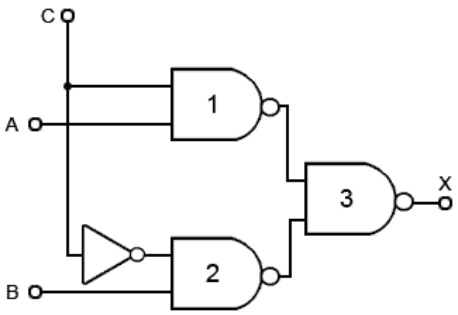


Fig. 4.2.2 Data Selector

The operation of Fig. 4.2.2 is quite straightforward and relies on 'enabling' either of a pair of NAND gates (1 and 2), but not both.

From the truth table for a NAND gate shown in Table 4.2.1 it can be seen that if one of the inputs (e.g. input A) is kept at logic 1, then the output will be the inverse of the other input. The gate is said to be enabled. If however input A is kept at logic 0, then the output will always be logic 1 whatever the state of the second input. The gate is therefore disabled, and the input cannot reach the output, even in inverted form. Gate 3 in Fig. 4.2.2 is simply combining the inputs from the other two gates. Table 4.2.2 illustrates the operation of Fig.4.2.2.

Table 4.2.1

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

Table 4.2.2

C	B	A	X
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Multiplexing

The control input (C) to the circuit in Fig.4.2.2 is fed directly to gate 1, but inverted to gate 2. This ensures that whatever the logic state of C, one gate is enabled, whilst the other is disabled.

Therefore when C = 1, NAND gate 1 will be enabled and its output will be the inverse of its data input (i.e. $\overline{A}$), and because $\overline{C}$ (in this case logic 0) is applied to the control input of gate 2 its output will be logic 1.

Applying logic 0 to input C will cause gate 1 to be disabled, making its output logic 1, and gate 2 will be enabled making its output $\overline{B}$.

Gate 3 will therefore always have one of its inputs held at logic 1, because either gate 1 or gate 2 is disabled, whilst the other input to gate 3 will be either $\overline{A}$ or $\overline{B}$. Gate 3 output will be the inverse of this input, so the result at the output X will be either A or B depending on the state of the control line, as can be seen from Table 4.2.2.

Note that this arrangement of three NAND gates (or four if an additional NAND gate is used in place of the inverter) works just the same as having an inverter (NOT gate) select either of two AND gates whose outputs are combined by an OR gate (De Morgan's theorem) but uses only one Quad 2-input gate IC instead of the three required by the NOT/AND/OR solution.

De-multiplexing

Having combined, or multiplexed two data sources into one output line, it will usually be necessary at some point to separate or de-multiplex the combined data into separate outputs once more. To do this for the circuit in Fig. 4.2.2, a circuit such as that shown in Fig 4.2.3 will be required.

Two connections from the data select circuit are required to connect the data to this simple de-multiplexer, one to connect the data from output X of the data select circuit in Fig. 4.2.2 to the combined A/B input of Fig.4.2.3 and another connection from C on Fig. 4.2.2 to C on Fig. 4.2.3 to share the control signal.

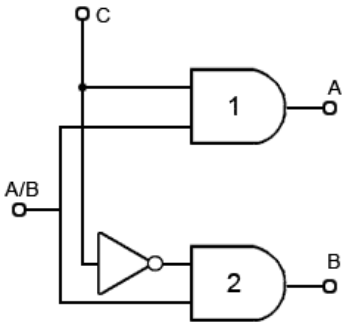


Fig. 4.2.3 A Simple Demultiplexer

Having to use two connecting lines to connect the multiplexer to the de-multiplexer to carry two signals does not apparently justify using these two extra circuits, however the principle of multiplexing demonstrated in Figs. 4.2.2 and 4.2.3 can be extended to multiplex a greater number of data inputs, and the more lines that are multiplexed in this way, the more efficient the system becomes. Also there are additional ways to use these techniques other than transferring data from one place to another, as explained in Digital Electronics Module 4.4 (Encoders and Decoders).

Multi - Bit Multiplexers

There are many uses for multiplexers. Wherever a number of signals, or logic states, need to be passed down a single communication channel such as a wire, a radio channel, or a telephone, some form of multiplexing is used. Sometimes the multiplexing and de-multiplexing can be very complex, much more so than the circuit in Figs 4.2.2 and 4.2.3. In some systems, data is transferred over very long distances, in others such as transferring data within computers, the distances may be very short. Fig. 4.2.4 shows a 4 to 1 line multiplexer, which enables a 4-bit binary number to be passed over 3 lines, one for data and two for control.

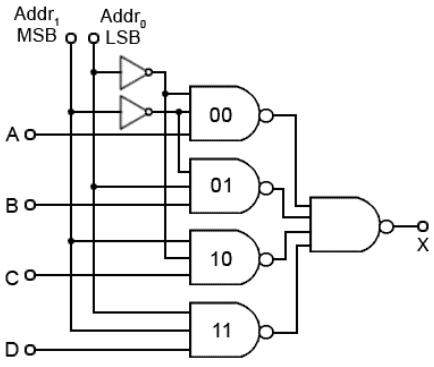


Fig 4.2.4 A 4 to 1 Line Multiplexer

Addressing

Larger multiplexers, such as 4, 8 or 16 bit types, which are readily available in IC form, use a method of 'addressing' a particular data gate using a binary code. Fig 4.2.4 shows a 4 to 1 multiplexer where, in order to output data from a particular input, one of the four 3-input NAND gates must be enabled by a logic 1 on two of its inputs, leaving the third input for data. To achieve this two address lines are used, giving four possible combinations of 1 and 0.

Look carefully at the address lines. When both are at logic 0 the two inverters (NOT gates) produce logic 1s at two of the inputs to NAND gate 00. None of the other NAND gates addressed by these lines has both its address inputs at logic 1. If the least significant bit (lsb) of the address is 1 and the most significant bit (msb) is 0 then NAND gate 01 is enabled. Because two address lines can give four possible binary combinations, you should find that, counting from the top, gate 00 is enabled by the address inputs 00_2, gate 01 by address 01_2, gate 10 by 10_2 and gate 11 by 11_2.

Multiplexer IC Datasheets

There are many commercially available multiplexer ICs available with a variety of extra features. The following is a list of datasheets for some basic multiplexers similar to those described in this article.

74HC151 8 to 1 Multiplexer from NXP

74HC153 4 to 1 Multiplexer from Texas Instruments

74HC257 Quad 2 input Multiplexer from Philips Semiconductor (NXP)

74HC352 Dual 4 to 1 Multiplexer from Texas Instruments

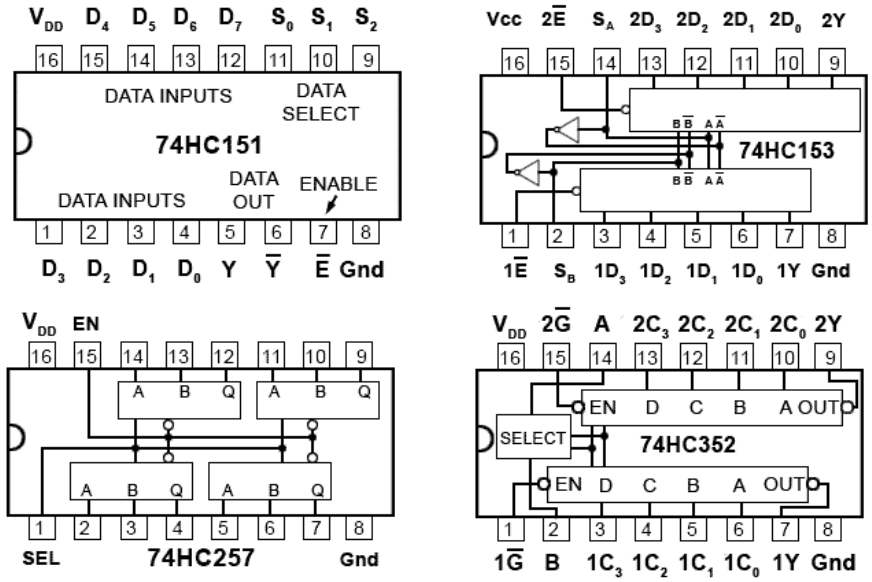


Fig. 4.2.6 Multiplexer ICs

4.3 Binary Comparators

What you'll learn in Module 4.3

After studying this section, you should be able to:

Recognise uses for Binary Comparator Circuits.

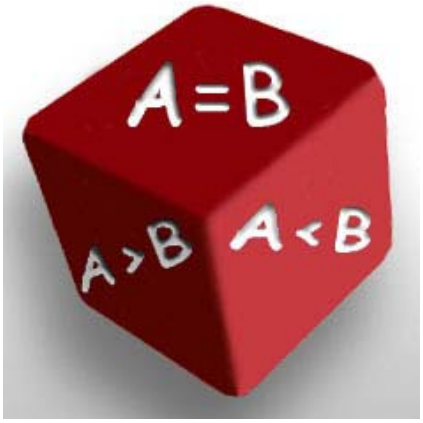
Understand the operation of Binary Comparators.

- Equality Comparators.
- Magnitude Comparators.

Simulate the operation of Multi-Bit Comparators using software.

- 4 Bit Magnitude Comparators
- Cascaded 4 bit Comparators.

Access Comparator IC Datasheets.



Binary comparators, also called digital comparators or logic comparators, are combinational logic circuits that are used for testing whether the value represented by one binary word is greater than, less than, or equal to the value represented by another binary word. Two basic types of comparator can be used.

- An equality comparator.
- A magnitude comparator.

Equality Comparators

An equality comparator, such as that illustrated in Fig 4.3.1 is the simplest multi-bit logic comparator, and can be used for such circuits as electronic locks and security devices where a binary password consisting of multiple bits is input to the comparator to be compared with another preset word.

In Fig.4.3.1, a logic 1 will be present at the output if the two input words match, otherwise the output remains at 0. Therefore there is only one input combination that is correct, and the more bits the input words possesses, the more possible wrong combinations there are. With extra circuitry for counting, additional security may be provided by limiting the number of tries before the input is inhibited.

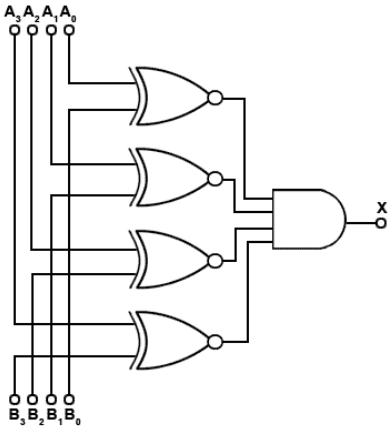


Fig. 4.3.1 Four Bit Equality Comparator

The circuit of the equality comparator consists of an exclusive NOR (XNOR) gate per pair of input bits. If the two inputs are identical (both 1s or both 0s) an output of logic 1 is obtained.

The outputs of the XNOR gates are then combined in an AND gate, the output of which will be 1, only when all the XNOR gates indicate matched inputs.

Magnitude Comparators

The magnitude comparator can also be used to indicate equality, but has a further two outputs, one that is logic 1 when word A is greater than word B, and another that is logic 1 when word A is less than word B. Magnitude comparators therefore form the basis of decision making in logic circuits.

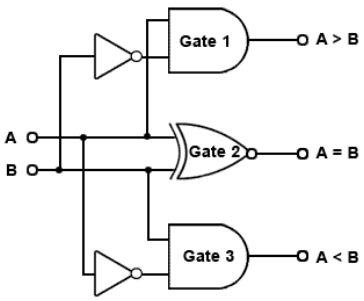


Fig. 4.3.2 One Bit Magnitude Comparator

Any logical problem can be reduced to one or more (sometimes many) yes/no decisions based on a pair of compared values.

A simple 1-bit magnitude comparator is shown in Fig 4.3.2. Gate 1 produces the function A>B and gate 3 gives A<B while gate 2 is an XNOR gate giving an equality output.

This basic circuit for a magnitude comparator may be extended for any number of bits but the more bits the circuit has to compare, the more complex the circuit becomes. Integrated circuit magnitude comparators are available that can be used to provide comparisons between multi-bit words. One such IC is the 74HC85 CMOS 4-bit magnitude comparator shown in Fig 4.3.3. This IC compares two 4-bit words and provides an output on pins 5, 6 and 7 that indicate whether the input words are equal, or if not, whether A or B has the higher numerical value.

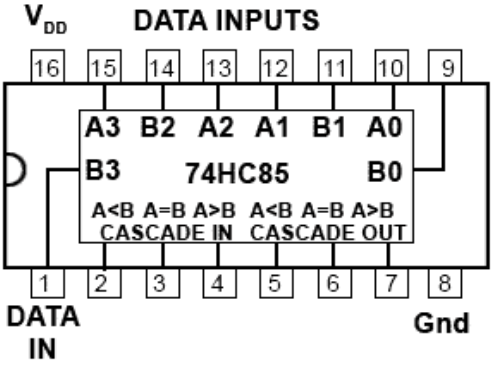


Fig. 4.3.3. 74HC85 4-Bit Magnitude Comparator

Medium Scale Integrated (MSI) Devices

Fig. 4.3.4 shows a simplified circuit of a typical four-bit comparator, based on the 74HC85 IC with input and output buffers omitted. If you have been studying previous digital electronics modules with learnabout-electronics, you may notice that the level of complexity in Fig 4.3.4 is much greater than in previous circuits. In Module 2.1 it was stated that any digital circuit relies on just a few types of logic gate (AND, OR, NAND, NOR, NOT, XOR and XNOR) and even this list can be reduced by utilising just AND OR and NOT to obtain the other logic functions. Therefore more complex logic circuits still use combinations of these basic functions but it is the connections between them, and the rapid increase in the number of gates used that adds to the complexity.

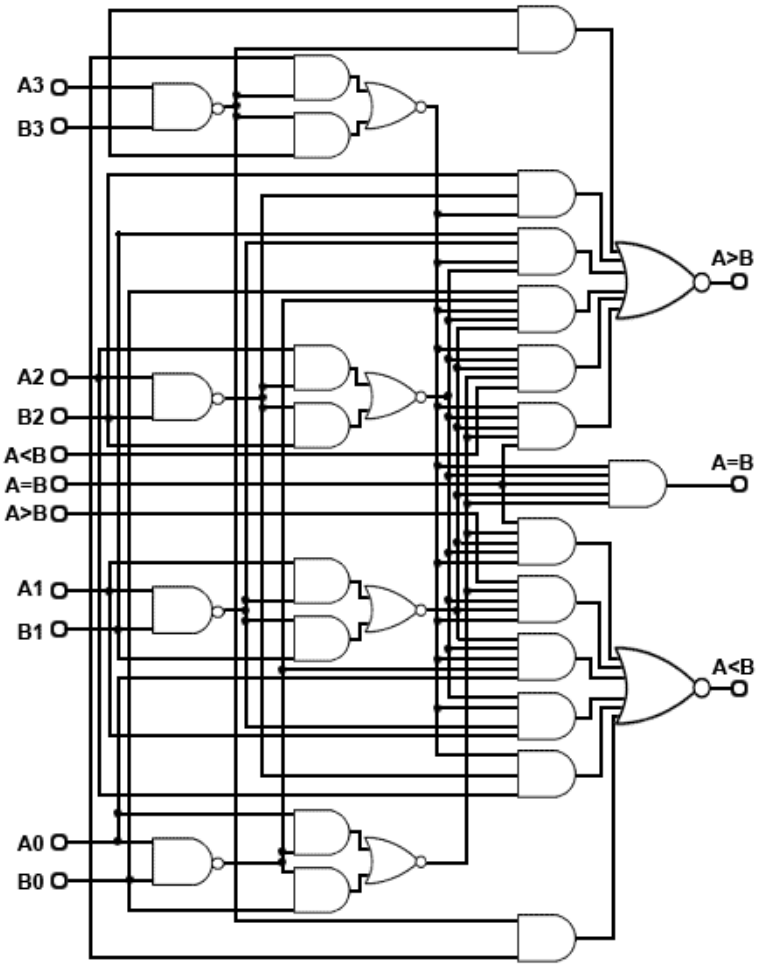


Fig. 4.3.4 Simplified Circuit of the 74HC85

ICs like the 74HC85 are called 'Medium Scale Integrated' or MSI devices to distinguish them from SSI (Small Scale Integrated) devices such as the basic logic gate ICs studied in Modules 2.1 and 2.2.

Although these devices seem (and are!) complex is interesting to compare the number of individual transistors in this circuit with those used in the circuits described in earlier modules. Fig 4.3.4

shows 31 gates (not including the omitted input and output buffer gates), and each gate comprises about 4 transistors per gate giving a total transistor count for this typical MSI chip of well over 124 transistors, so it is not surprising that the circuit looks complex!

This one small IC then, contains more transistors than would be found for example in many analogue colour TV receivers, however this circuit does much less that would be required of the same number of transistors in a TV, and its operation is much easier to understand, especially if you already understand the operation of basic logic gates.

Note the outputs in Fig. 4.3.3, for A<B, A=B and A>B on pins 5, 6 and 7, and similar inputs on pins 2, 3 and 4, which enable a number of 74HC85 chips to be connected together to provide magnitude comparators for any word length.

Comparators in Cascade

When two or more ICs are cascaded together, as shown in Fig. 4.3.5, the outputs of the first IC (representing the least significant 4 bits) are connected to the cascade inputs of the second IC and so on. The final result of the comparison appears on the three cascade outputs of the most significant 4-bit comparator.

To ensure a correct comparison, the cascade inputs of the first (least significant) comparator should be connected as follows:

A<B (pin 2) and A>B (pin 4) = logic 0.

A=B (pin 3) = logic 1.

This also applies to a single IC if only two 4-bit words are being compared.

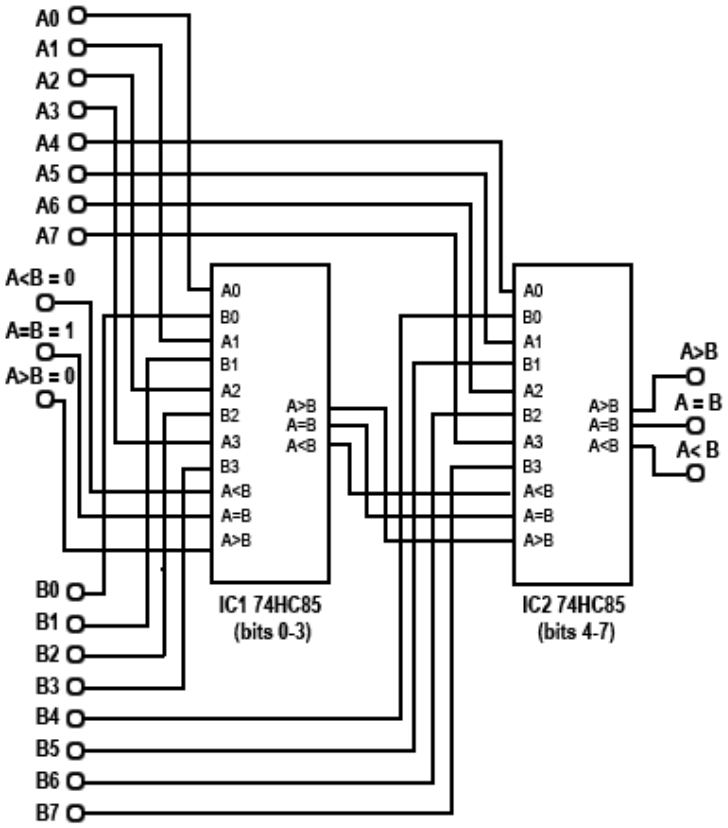


Fig. 4.3.5 Eight-Bit Magnitude Comparator Using Two 74HC85 ICs

4.4 Encoders and Decoders

Binary Encoders

What you'll learn in Module 4.4

After studying this section, you should be able to:

Recognise the need for Code Converters

Understand the operation of Binary Encoders.

- Priority Encoders.
-

Understand the operation of Binary Decoders.

- Logic Decoders.
- 74 Series Decoders.

Simulate the operation of Encoders and Decoders using software.

- 8 to 3 Line Encoders.
- Decoders.
  - 2 to 4 line
  - BCD to decimal
  - BCD to 7 segment
  - Address decoders

Access Encoder and Decoder IC Datasheets.

Digital Electronics Module 1 (Number Systems) described a number of different binary codes that are used to perform a range of functions in digital circuits. Mathematics, graphics, data manipulation and physical control systems are among many of the functions that are carried out using binary data, and each of these uses may require binary data arranged in various forms of binary codes. For example text may be represented by an ASCII code (American standard Code for Information Interchange), in which each letter, number or symbol is represented by a 7-bit binary code. Decimal numbers in a calculator may be sent to a numeric display using BCD (Binary Coded Decimal). Notice that the word 'code' appears in each of these titles, and a binary code differs from normal binary because it is arranged in a particular way to suit a given purpose.

Priority Encoders

Binary Encoders generally have a number of inputs that must be mutually exclusive, i.e. only one of the inputs can be active at any one time. The encoder then produces a binary code on the output pins, which changes in response to the input that has been activated.

Depending on the encoding purpose, each each different IC has its own particular method for solving encoding problems. For example, a simple decimal to BCD (or 10-to-4 line) encoder would be expected to have ten input pins, but in fact the 74HC147 has only 9. The tenth condition (zero) is assumed to be present because when none of the 1 to 9 input pins is active, this must indicate zero. The input pins may be used to connect to switches on a decimal keypad, and the encoder would output a 4-bit BCD code, (0000_2 to 1001_2) depending on which key has been pressed, or simply to identify which one of ten input lines in a circuit is active, by outputting an appropriate number in four bit BCD code.

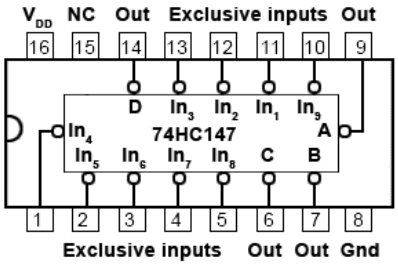


Fig. 4.4.1 74HC147 10-to-4-Line Priority Encoder

Priority Encoding

Because it is always possible when using input switches that more than one input may be active at a single time, most encoders of this type feature 'priority encoding' where, if more than one input is made active at the same time, the output will select only the most significant active input. For example, if 6 and 7 are pressed together the BCD output will indicate 7.

The Pinout diagram for the 74HC147 10-to-4-line priority encoder from NXP (Philips Semiconductor), is illustrated in Fig.4.4.1.

Chip Enable Inputs

Some other encoder ICs also feature extra inputs and outputs that allow several ICs to be connected together to achieve more flexibility in the numbers of input and output lines available. These

typically include $\overline{\text{ENABLE}}$ inputs, which may consist of one or more input pins that need to have a particular logic level applied (usually logic 0) in order to activate the encoding action. In the absence of a correct $\overline{\text{ENABLE}}$ signal the output pins of the IC will remain in their inactive state.

Switch Bounce

One problem with combinational logic circuits is that unintended changes in output data can occur during the times when the outputs of the IC are changing. This can be due to problems such as switch contacts 'bouncing' as they close, creating rapid and unpredictable changes in logic levels for a very short time, however logic IC operate at high speed and will respond to these very fast changes.

Race Hazards

Problems can also occur due to 'race hazards' where different paths that digital signals take through a logic circuit may have different numbers of gates. For example two logic signals that change simultaneously at two circuit inputs may take different routes through the circuit before being applied to some common gate later in the circuit. However, if one signal passes through six gates for example, while the other signal passes through seven gates, each of the signals will have encountered a different total propagation delay due to the different number of gates they encountered. Therefore they will each arrive at the common gate at slightly different times, and so for a very short time an unexpected logic level may occur at that gate output.

In using combinational logic ICs such as an encoder, problems like switch bounce and race hazards must be allowed for, and one (though not necessarily the best) solution can be to temporarily make the $\overline{\text{ENABLE}}$ pin high during times when data is likely to change. This disables the encoder for a short time until the signal data has settled at its new state, so that there is no chance of errors at the output during changes of input signals.

74HC148 8-to-3-Line Encoder

The 74HC148 also uses priority encoding and features eight active low inputs and a three-bit active low binary (Octal) output. The internal logic of the 74HC148 is shown in Fig. 4.4.2

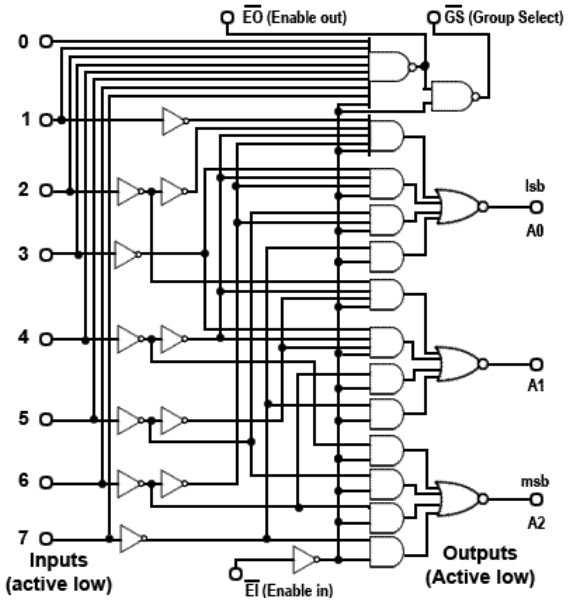


Fig.4.4.2 74HC148 8-to-3-Line Encoder

The IC is enabled by an active low Enable In ($\overline{\text{EI}}$), and an active low Enable Out (EO) is provided so that several ICs can be connected in cascade, allowing the encoding of more inputs, for example a 16-to-6-line encoder using two 8-to-3 encoders. The CMOS 74HC148 also uses active low inputs and outputs. The operation of the 74HC148 can be seen from its truth table shown in Table 4.4.1.

Notice from Table 4.4.1 that the IC is only active when $\overline{\text{EI}}$ is low, and also that for each input selected by a low logic level (L), all lower value inputs indicate 'Don't Care', typical of priority encoding.

Table 4.4.1

Inputs									Outputs				
$\overline{\text{EI}}$	0	1	2	3	4	5	6	7	A2	A1	A0	$\overline{\text{GS}}$	$\overline{\text{EO}}$
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L	X	X	X	X	X	X	L	H	L	L	H	L	H
L	X	X	X	X	X	L	H	H	L	H	L	L	H
L	X	X	X	X	L	H	H	H	L	H	H	L	H
L	X	X	X	L	H	H	H	H	H	L	L	L	H
L	X	X	L	H	H	H	H	H	H	L	H	L	H
L	X	L	H	H	H	H	H	H	H	H	L	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H

The Group Select ($\overline{GS}$) output is used together with $\overline{EO}$ for connecting additional 74HC148 ICs in cascade.

The $\overline{EI}$ input is normally used on the most significant IC and whenever an input on this IC is selected, the $\overline{EO}$ output goes high (disabling any less significant ICs), and the Group Select ($\overline{GS}$) output goes low indicating that the group of outputs of this IC are active.

16-to-4-Line Encoder

Fig 4.4.3 shows a simulation created in Logisim, which demonstrates how two 74HC148 ICs can be connected in cascade to make a 16-to-4-line encoder. Notice how $\overline{EI}$ is used to enable the most significant encoder, and how $\overline{EO}$ and $\overline{EI}$ in the centre of the diagram are used to cascade the ICs. As the output (0000_{16} to $FFFF_{16}$) will now require 4 bits. The $\overline{GS}$ (Group Select) pin, which changes to its low logic state when any input on the most significant IC is active, is used to create the fourth output bit, (2^3) for any output value above 7.

In this simulation, the active low outputs of the encoder have been inverted to provide active high inputs to the hexadecimal display.

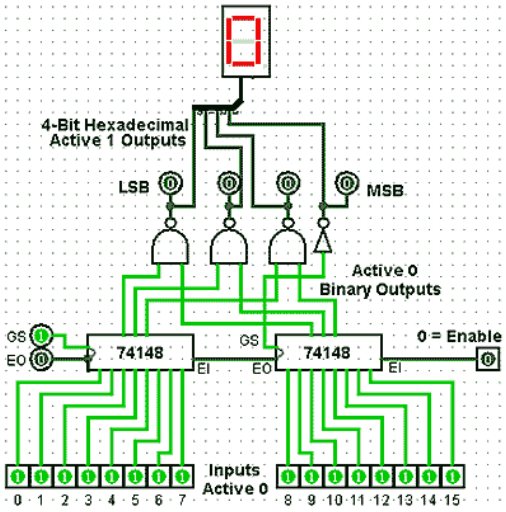


Fig. 4.4.3 16-to-4-Line Encoder (Logisim Simulation)

Diode Matrix Encoders

Where encoders are needed for non-standard applications, they can also be implemented using a diode matrix, such as the decimal-to-BCD_{2421} encoder shown in Fig 4.4.4.

In this example, as any one of the ten switches is closed +5V is applied to just one of the horizontal lines. Any diode that has its anode connected to that horizontal line, and its cathode connected to a vertical line (that is held at zero volts by a resistor connected to Gnd) will conduct.

When current flows through any of the resistors, the top of that resistor will be at +4.4V (i.e. +5V minus a 0.6V drop in across the diode), which will be seen by the output as logic 1.

For example if switch 6 is closed, the two diodes connected between line 6 and columns X_2 and X_1 will conduct, making outputs X_2 and X_1 logic 1 and giving a $binary_{2421}$ output word of 0110_2 (6_{10}).

This particular diode matrix will therefore give an output in BCD_{2421} code from 0000_{2421} to 1111_{2421} for closure of switches 0 to 9.

Many other output sequences are possible therefore, by using different arrangements of the diode positions.

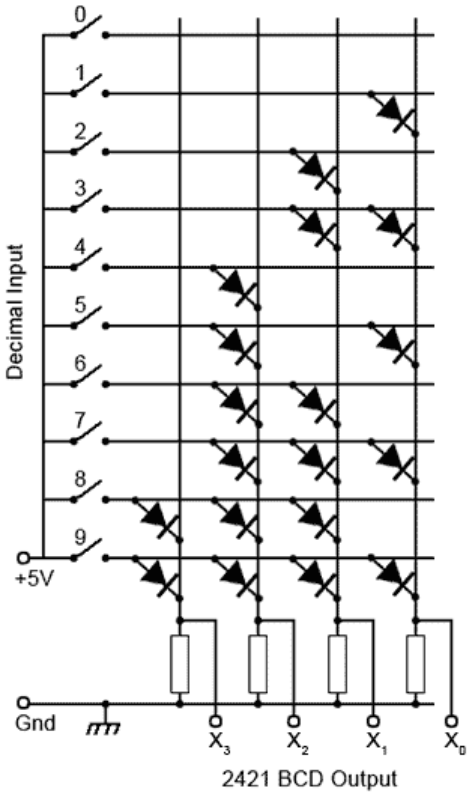


Fig 4.4.4 Decimal-to-2421 BCD Diode Matrix Encoder

Although the encoder circuits described in this module may be used in a number of useful encoding situations, they have some features that limit their use for realistic keyboard encoding.

- Priority encoders do not sense information from two or more keys that are pressed at the same time.
- Switches on keypads normally contact for only a brief time, these basic encoders are not able to store and remember the data input from a pressed key once it is released.
- When a switch is closed the contacts may 'bounce' giving several brief 1 and 0 logic states, when ideally there should be only one change in state for each key press.

To overcome common problems such as these, a more complex circuit (or IC) is required. These will typically have features such as key bounce elimination, built in data memory, timing control using a clock (oscillator) circuit and some ability to differentiate between two or more keys pressed at the same time. Another important feature is the ability to signal to the system that the keyboard is controlling, when a key has been pressed and new data needs to be read.

For small keypads having less than 20 keys the processing has typically been carried out by an ASIC (Application Specific Integrated Circuit) such as the MM74C922 Keyboard Encoder, although this IC is now being listed as obsolete by some manufacturers as many modern circuits, especially those with more keys, use a dedicated microprocessor or micro-controller (MCU)to carry out keyboard decoding.

Binary Decoders

These circuits in IC form are often called Decoders/Demultiplexers and perform the opposite function to an encoder (or multiplexer).

Binary data is used in digital circuits in the form of one or another binary code, which is an arrangement of the binary bits in a particular order to represent 'real' quantities such as a set of decimal numbers (BCD code) or text (ASCII). In a complete digital system therefore it is often necessary to convert one code to another, or to convert a binary code to drive some user interface such as a LED display.

A decoder is a combinational logic circuit that takes a binary input, usually in a coded form, and produces a one-bit output, on each of a number of output lines. The logic state (1 or 0) on any of the output lines depends on a particular code appearing on the input lines.

2-to-4-Line Decoder

For example, a 2-to-4-line decoder is shown in Fig. 4.4.5, in this circuit the two input lines can be set to any one of four binary values, 00, 01, 10 or 11. Resulting from this input, and provided that the (active high) Enable input is set to logic 1, the output line corresponding to the binary value at inputs A and B changes to logic 1. The other output lines remain at logic 0.

When the binary value at inputs A and B changes, the logic 1 on the output changes to a different line as appropriate. If the enable input is set to logic 0, all the outputs remain at logic 0 whatever values appear at inputs A and B.

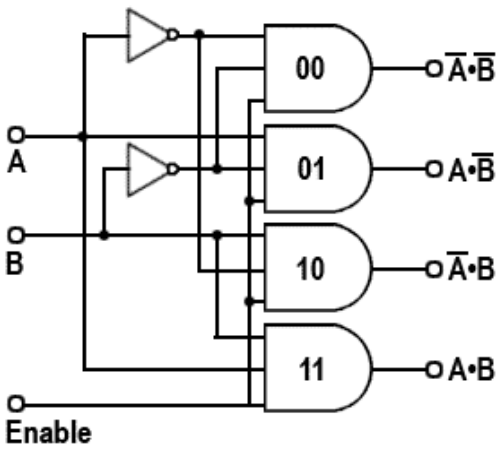


Fig 4.4.5 2-to-4-Line Decoder

To obtain a logic 1 at any of the four outputs, the appropriate 3 input AND gate must have all of its inputs at logic 1. Provided that the Enable input is at logic 1, the output is controlled by using NOT gates to invert the logic applied from inputs A and B as required.

For example if inputs A and B are both at logic 0, the NOT gates at the inputs to the top (00) AND gate, invert both 0 inputs to logic 1, and therefore logic 1 appears at the 00 output. The 01 and 10 AND gates each have one input directly connected to the A or B input, whilst the other input is inverted.

The 11 gate has both A and B inputs directly connected to the AND gate so that 11_2 applied to A and B results in logic 1 at the 11 output.

Notice the similarity between Fig 4.4.5 and the 4-to-1-Line Multiplexer shown in Fig 4.2.4. In fact Fig. 4.4.5 could act as a demultiplexer for Fig 4.2.4 if the A and B inputs are used as control lines, and the enable input of Fig 4.4.5 used as the single data input. This example of dual use explains why decoders are often called Decoder/Demultiplexers. The circuit operation of Fig. 4.4.5 is shown in truth table form in Table 4.4.2.

Table 4.4.2

Inputs			Outputs			
EN	B	A	00	01	10	11
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

74 Series Decoder ICs

2-to-4-line decoders (also called 1 of 4 decoders) are commercially available in both HC and HCT types in a number of versions from different manufacturers. These are typically dual packages such as the 74HC139 from NXP with two decoders per chip. One difference, (commonly used) from the basic example shown in Fig. 4.4.5 is that the outputs, and sometimes also the inputs, on such ICs may be 'active low' meaning that the active or logic 1 state is at the lower voltage of the two possible logic states, so that the output is sinking current when it is 'logic 1'. This provides a greater drive capability than would be available if logic 1 was at its high voltage, and sourcing current. Also, decoder ICs are very often used to activate the Enable or Chip Select ($\overline{CS}$) inputs of other ICs, which are usually active low, so having a decoder with an active low output saves using extra inverter gates.

Another feature found in 74 series ICs is the common presence of buffer gates (which may be inverting or non-inverting) at the IC inputs and outputs to give improved input and output capabilities Clamp diodes and current limiting resistors are also often incuded at the inputs and outputs to give improved protection from high electrostatic external voltages.

BCD-to-Decimal Decoder

The 74HC42 BCD-to-Decimal decoder IC from Philips Semiconductors contains a more complex circuit, as illustrated in both block and logic schematic form in Fig. 4.4.6.

The input is in 4-bit BCD_{8421} format, and each of the ten outputs, labelled $\overline{Y0}$ to $\overline{Y9}$ produce a logic 0 for an appropriate BCD_{8421} input of 0000_{8421} to 0101_{8421}. Any input value greater than 0101_{8421} results in all of the output pins remaining at their high level, as shown in pale blue in Table 4.4.3.

Note that the truth table (Table 4.4.3) shows the appropriate high and low logic levels as 1 and 0 respectively to match the logic levels shown in the downloadable Logisim simulation.

On most data sheets for ICs the levels are shown as H (the higher voltage) and L (the lower voltage) to avoid confusion in cases where negative logic is used.

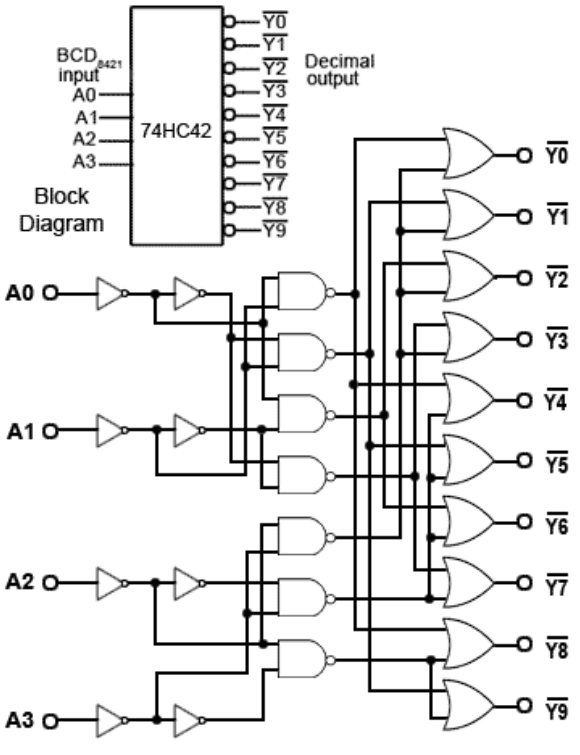


Fig. 4.4.6 The 74HC42 BCD-to-Decimal Decoder

BCD to decimal decoders were originally used for driving cold cathode numerical displays (Nixie tubes), which are neon filled glass plug-in tubes with ten anodes in the shape of numbers 0 to 9 that glow when activated by a high voltage.

However, decimal decoders are also useful for a variety of other uses. Remember that decoders are often also called demultiplexers, as they can be used for many demultiplexing tasks and for driving devices such as lamps, motors and relays in control systems.

Fig 4.4.7 Cold Cathode Display

Table 4.4.3

INPUTS				OUTPUTS									
A3	A2	A1	A0	$\overline{Y0}$	$\overline{Y1}$	$\overline{Y2}$	$\overline{Y3}$	$\overline{Y4}$	$\overline{Y5}$	$\overline{Y6}$	$\overline{Y7}$	$\overline{Y8}$	$\overline{Y9}$
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	0	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1

BCD to 7 Segment Decoders

Because cold cathode displays require a high voltage drive, they have mostly been replaced by low voltage LED or LCD displays using 7 segment displays, therefore the BCD-to-7-segment decoder has become one of the most commonly available decoders.

As shown in block diagram format in Fig. 4.4.8, this type of decoder has 4 inputs for binary coded decimal and an output for each of the 7 LEDs that make up the 7-segment display. The eighth LED (labelled dp or sometimes h) will normally be controlled by some extra logic outside the decoder. 7-segment displays may have a common cathode connection, needing to be driven by logic 1 outputs, or common anode connection requiring logic 0 outputs from the decoder.

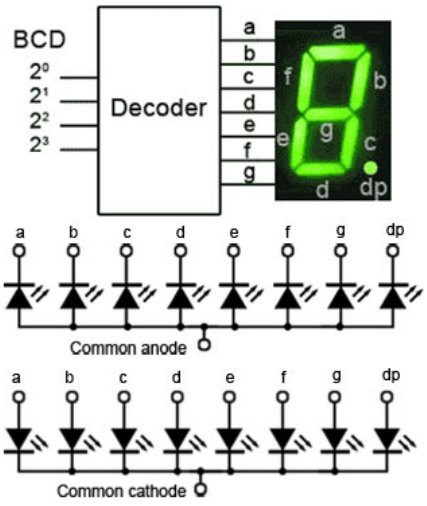


Fig. 4.4.8 Driving a 7 Segment Display

Decoder/Drivers

Depending on the decoder IC and display type used, it may sometimes be necessary to use a transistor amplifier to drive each segment of the display, however there are decoder/driver ICs available, such as the 74LS46, 47, 48 and 49 that have sufficient output current and a choice of output designs such as open collector, internal pull-up resistors and active high or active low output levels that allow direct connection to both led and filament lamp displays.

7-Segment Fonts

When illuminated by the correct logic levels, the seven-segment display will show all the decimal numbers from 0 to 9. Depending on the logic design of the IC, some decoders will automatically blank the display for any value greater than 9, while others display a unique (non-numeric) pattern for each value from 10 to 15 as shown in Fig. 4.4.9, and may display 6 and 9 with or without a 'tail'.

Fig. 4.4.9 Typical 7 Segment Font

For displaying Hexadecimal numbers, the letters A b C d E and F are used to avoid confusion between capital B and 8, and capital D and 0.

Basic BCD to 7-Segment Decoder

Fig. 4.4.10 is a screen grab from Logisim, showing a working simulation of a basic BCD to 7-segment decoder (based on the 74LS49 in the TTL range of 7-segment decoders from Texas Instruments). This IC uses the font illustrated in Fig. 4.4.9 and a single active low $\overline{BI}$ pin for use as a blanking input.

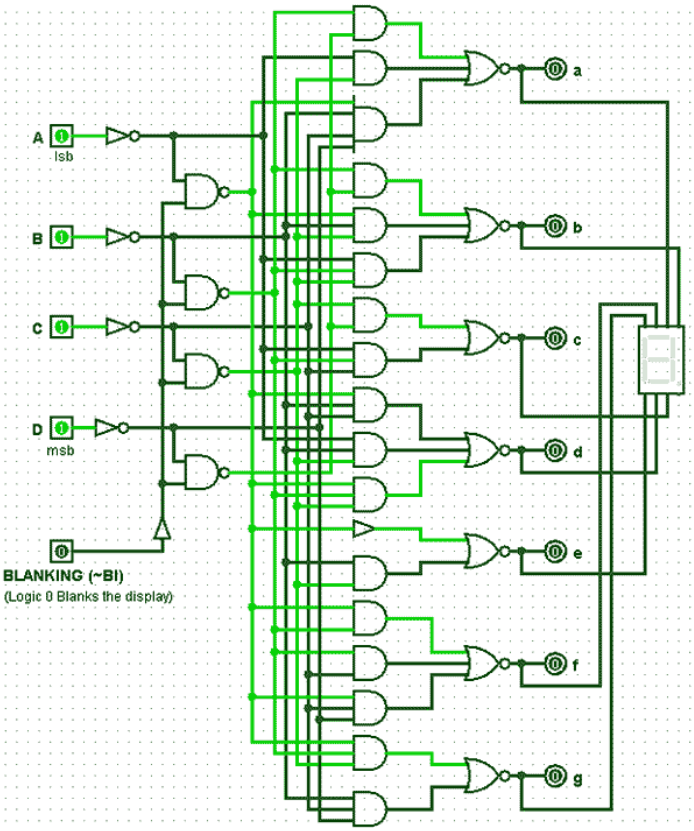


Fig. 4.4.10 Logisim simulation of a basic BCD to 7 Segment Decoder

Blanking

The blanking input pin $\overline{BI}$ can be used to turn off the display to reduce power consumption, or it can be driven with a variable width pulse waveform to rapidly switch the display on and off thereby varying the apparent brightness of the display.

Making the $\overline{BI}$ input logic 0 blanks the display whatever data is present at the decoder BCD inputs.

Ripple Blanking

As a BCD to 7 Segment decoder is designed to drive a single 7 segment display, each digit of a numeric display is driven by a separate decoder, so where multiple digits are required, a technique called Ripple Blanking is used, this allows the blanking inputs of several ICs to be connected in cascade. The Ripple Blanking Output (RBO) of the first decoder IC (controlling the most significant digit) is fed to the blanking input pin of the next most significant digit decoder and so on.

When Logic 0 is applied to the ripple blanking input (RBI) of a decoder, it blanks the display only when the BCD input to that particular decoder is 0000. A logic 0 input will therefore blank any

display digit that is 0. This allows for the suppression of any leading or trailing zeros in numbers such as 00000077 or 7.7000000.

There are a number of BCD-to-7-segment decoder ICs in the 74 series (types 7446 to 7449) each with different variations, such as active high or active low outputs, high current driver outputs, a choice of display font (whether the 6 and the 9 have a 'tail' or not), and a lamp test input to check that all LEDs are working.

Fig. 4.4.11 is a screen grab of a BCD to 7-segment decoder using these advanced features. Click the 'Simulation Available' button to download a working simulation for this circuit. Note that although the simulation works in a similar manner to a real decoder such as the 74LS48, because the $\overline{BI}$ input and $\overline{RBO}$ output on the real chip share a common pin, this creates problems for the simulator. Therefore the logic has been changed by using two tri-state buffers to separate the input and output signals.

Tri-State Logic

The simulation illustrated in Fig. 4.4.11 uses two 3-state buffers (also call Tri-state logic) to achieve isolation between a shared input and output pin. The necessary isolation was achieved by using two simple tri-state buffers, shown in Fig 4.4.12 so that the shared pin can be an input or an output, but not at the same time.

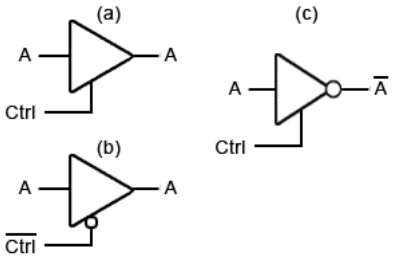


Fig. 4.4.12 3-State Buffers

The tri-state buffer (a) in Fig. 4.4.12 has an input and an output just like a normal buffer, but it also has a control (Ctrl) input. This input, when held at logic 1 enables the buffer, so whatever logic level appears at its input also appears at its output.

When logic 0 is applied to the Ctrl input however, the buffer is disabled and its output assumes a 'high impedance' state. That is, it will take up whatever logic level occurs on the line connected to its output, no matter what logic level is on its input. It is effectively open circuit, just as though making the enable input low had opened a switch between its input and output. Tri-state buffers are also available with an active low Ctrl input, that are enabled by logic 0 (b), and as inverting buffers, that invert the output when Ctrl is activated (c).

There are whole ranges of devices that have 3-state outputs. Devices such as microprocessors and memory chips, intended for use in bus systems, where many inputs and outputs share a common connection (e.g. a data bus) normally have tri-state outputs. However, apart from very large scale integrated (VLSI) devices, such as computer ICs, 3-state logic is not generally used within MSI ICs as shown in Fig 4.4.11. In these smaller scale ICs, alternatives such as open collector logic are more suitable. Discrete 3-state logic components are more often used for connections between, rather than within ICs.

Latching BCD-to-7-Segment Decoders

In common with modern practice, TTL ICs are not generally recommended for new designs and are superseded by newer HC and HCT versions. Apart from the advantage of lower power on these versions, a common feature added to these ICs is a 'Data Latch'. This is a one nibble memory (for the 4 bit BCD input) controlled by a Latch Enable ($\overline{LE}$) pin, which allows the decoder to store the 4 bit input present, when $\overline{LE}$ is logic 0 so that only the stored data is displayed. When the latch is not enabled however, it becomes 'transparent' i.e. any changes in the data appearing at the inputs are fed directly to the display. It is also common on later ranges of decoders that any input values greater than 1001_{BCD} (9_{10}) are automatically blanked. Fig. 4.4.13 shows a typical pin-out for

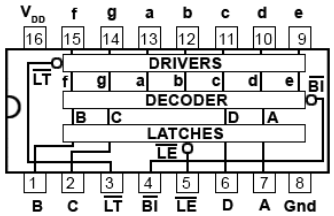


Fig. 4.4.13 Latching BCD-to-7-Segment Decoder 74HC4511

a CMOS BCD-to-7-segment decoder. Examples of these ICs are the MC14513 from ON Semiconductor and the 74HC/HCT4511 from NXP.

Address Decoders

Decoders may also be used in computer systems for address decoding. Fig. 4.4.14 illustrates a typical application where a 74HC138 3-to-8-line decoder is used to enable the microprocessor to communicate with many locations in its memory system.

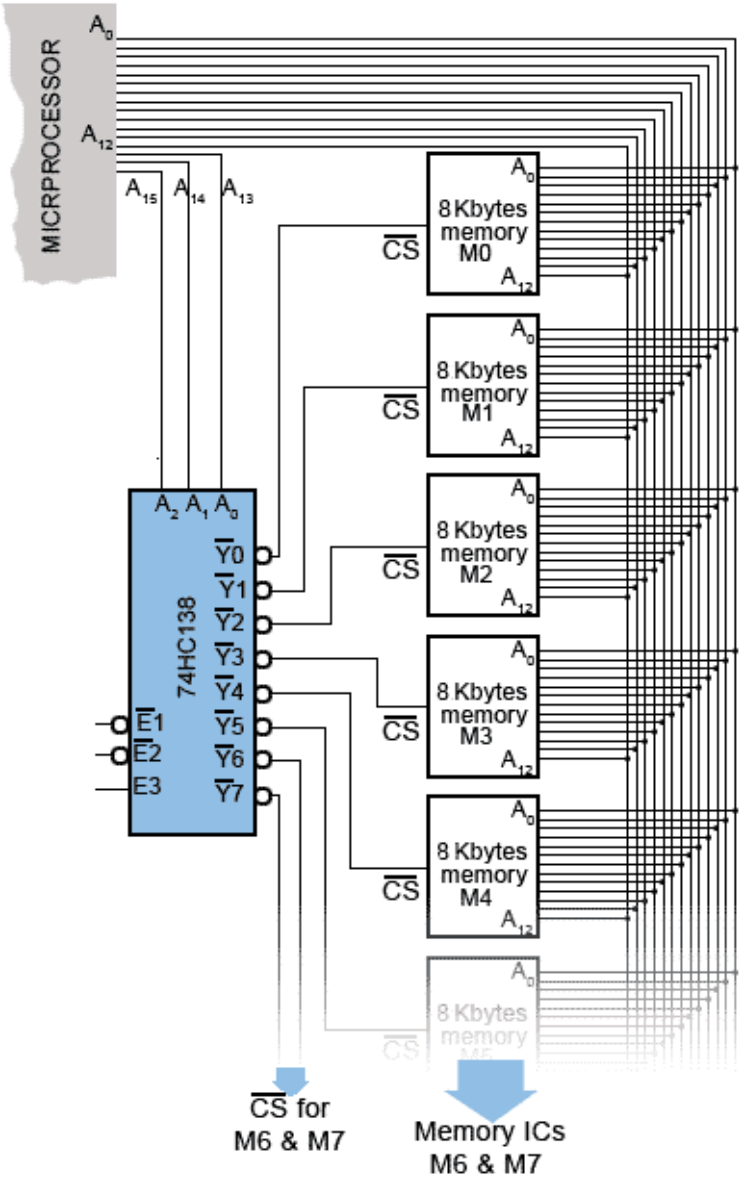


Fig. 4.4.14 Address Decoding

The memory in this example comprises 8 x 8Kbyte memory ICs, therefore each IC contains 8192 x 1 byte locations giving a total number of 8 x 8192 = 65536 locations, each having a hexadecimal location number (an address) from 0000_{16} to $FFFF_{16}$.

To contact any of these memory locations, the microprocessor outputs the address of the required memory location on the 16-bit address bus, which can hold any one of $2^{16} = 65536$ different values. However, the memory in this example is made up from 8 identical ICs, each holding 8192 locations, and as this number of locations can be addressed by $2^{13} = 13$ address lines, lines (A_0 to A_{12}) are connected from the microprocessor to the 13 address inputs of each memory IC.

This common connection means that each of the memory chips will have the same address range as all the other memory ICs, and therefore any address within the range 0000_{16} to 2000_{16} (8192_{10}) put out by the microprocessor will contact the same address in all 8 memory ICs. This obviously creates a problem, as each memory chip should have its own range of addresses with the 8 ICs forming a continuous address sequence in blocks of 8192_{10} locations.

This is where the address decoder is used. Notice that, in Fig. 4.4.14 the three highest order address lines (A_{13}, A_{14} and A_{15}) are connected to the 3 address inputs (A_0, A_1 and A_2) of the 74HC138 3-to-8-line decoder. Therefore, provided that the three Enable inputs ($\overline{E1}$, $\overline{E2}$ and E3) of the decoder are fed with the appropriate logic levels to enable the decoder, each of the $\overline{Y0}$ to $\overline{Y7}$ pins of the decoder will output a logic 0 for one of the 8 possible combinations of the three bit value on the address lines A_{13} to A_{15}.

Since this three bit value will only change when the 16-bit value on the address bus changes by 8192_{10} (2000_{16}) the memory chips will be selected using their chip select ($\overline{CS}$) inputs, every 8 Kbytes. The eight memory ICs will therefore provide a sequential set of memory locations covering the whole 64K of memory, addressable by the microprocessor.

The 73HC138 Decoder

The combinational logic of a typical 3-to-8-line decoder based on the 74HC138, is illustrated in Fig. 4.4.15, an IC that has many uses apart from address decoding, it is often used with a binary counter driving its inputs, when its eight outputs constantly step through a 0 to 7 sequence. Typical applications include sequence generating for lamp control, row scanning for dot matrix displays, digital operation of analogue controls and anywhere that a sequence of unique outputs is required.

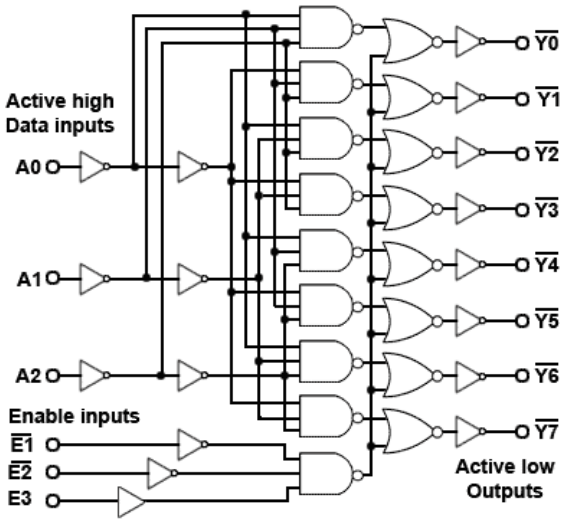


Fig. 4.4.15 3-to-8-Line Decoder Based on the 74HC138

Data sheets for the 74HC138 point out the advantages of the three Enable pins, which can be used for simply connecting the decoders together to make larger decoders. An example of this is shown in the downloadable Logisim simulation Fig. 4.4.16, where two 74HC138 ICs are connected together using only one additional NOT gate.

Note that the pin connections on the ICs in Fig. 4.4.16 are not in the consecutive 1 to 16 order of the real 74HC138 pinout, but can be identified in the downloaded simulation by hovering your mouse over any pin.

The $\overline{E1}$ (active LOW) input is used here as the fourth (2^3) data input so that for a count of 0 to 7_{10} (0000_2 to 0111_2) at the inputs, the logic 0 applied to $\overline{E1}$ enables the top IC and disables the bottom IC via the NOT gate, but for a count between 1000_2 and 1111_2 (8_{10} to 15_{10}) the fourth data input ($\overline{E1}$) becomes logic 1 and the situation is reversed, with the (active low) output continuing its (8_{10} to 15_{10}) sequence on the bottom IC.

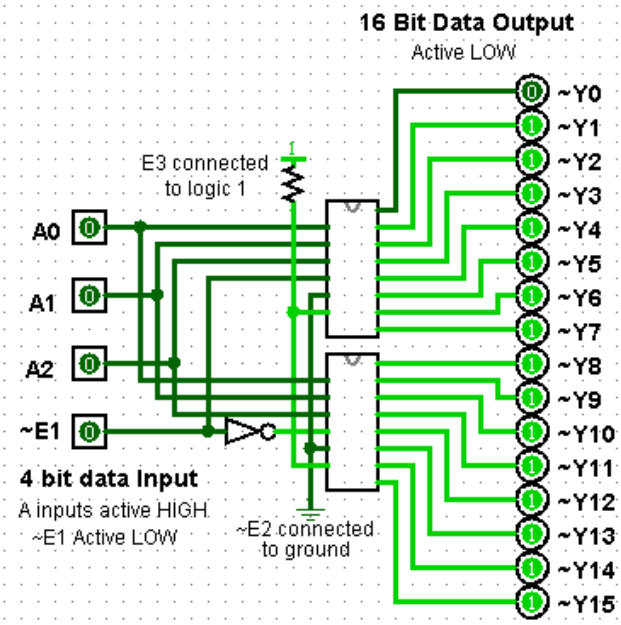


Fig. 4.4.16. Two 74HC138 ICs making a 4-to-16-Line Decoder

4.5 Combinational Logic Quiz

Try our quiz, based on the information you can find in Digital Electronics Module 4 – Combinational Logic.

Check your answers at http://www.learnabout-electronics.org/Digital/dig45.php and see how many you get right. If you get any answers wrong. Just follow the hints to find the right answer and learn about combinational logic as you go.

1.

If the inputs to a full adder are A=1, B=0, C_{IN}=1 what will be the logic states on the outputs S and C_{OUT}?

a) S=0 C_{OUT}=0

b) S=0 C_{OUT}=1

c) S=1 C_{OUT}=0

d) S=1 C_{OUT}=1

2.

In a two-gate half adder circuit, which of the listed gates would produce a Carry output?

a) AND

b) OR

c) XOR

d) NAND

3.

What type of circuit is illustrated in to Fig 4.5.1?

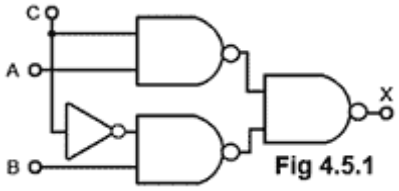


Fig 4.5.1

a) Full Adder.

b) Demultiplexer.

c) Data Selector.

d) Equality Comparator.

4.

Refer to Fig. 4.5.2, If S_1=1 and S_2= 0 what will be the logic state at the output X?

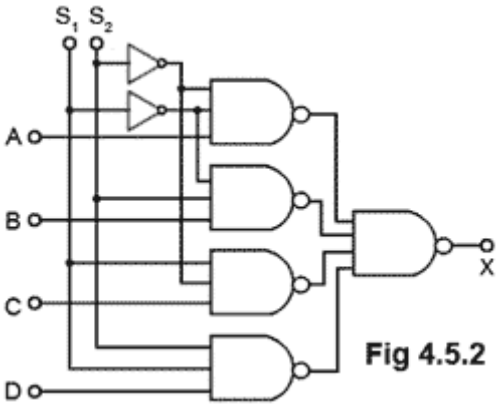


Fig 4.5.2

a) X = A

b) X = B

c) X = C

d) X = D

5.

Refer to Fig 4.5.3. If A = 0 and B = 1, what will be the logic states at X, Y and Z?

a) X=1, Y=1, Z=0

b) X=1, Y=0, Z=0

c) X=0, Y=1, Z=0

d) X=0, Y=0, Z=1

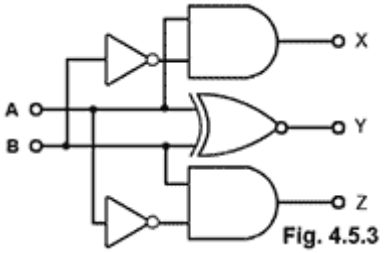


Fig. 4.5.3

6.

What is the integration scale of the IC illustrated in Fig 4.5.4?

a) SSI

b) MSI

c) LSI

d) VLSI

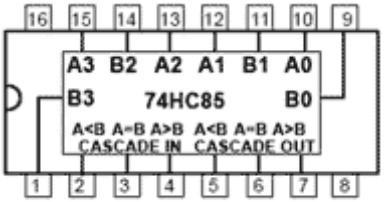


Fig. 4.5.4

7.

Refer to Fig 4.5.4. When two or more ICs of the type illustrated are connected in cascade, what should be the logic states applied to pins 2, 3 and 4 of the IC processing the least significant 4 bits?

a) Pin 2 = 0, pin 3 = 1, pin 4 = 0

b) Pin 2 = 1, pin 3 = 0, pin 4 = 0.

c) Pin 2 = 0, pin 3 = 1, pin 4 = 1.

d) Pin 2 = 0, pin 3 = 0, pin 4 = 0.

8.

What type of combinational logic circuit is illustrated in Fig. 4.4.5?

a) A Decimal to BCD_{8421} decoder.

b) A Decimal to BCD_{2421} decoder.

c) A Decimal to BCD_{8421} encoder.

d) Decimal to BCD_{2421} encoder.

9.

What is the main purpose of the Ripple Blanking facility on some decoder ICs?

a) It allows decoder ICs to be connected in cascade.

b) It prevents the display of digits greater than 9 on decimal displays.

c) It blanks leading and trailing zeros.

d) It increases the speed of decoding.

10.

What is the range of memory that can be addressed in a memory system, using a 16-bit address bus, 8Kbyte memory chips and a single 3 to 8 line address decoder?

a) 8192 memory locations.

b) 64 x 8Kbyte memory ICs.

c) 65536 memory locations.

d) 2Mb memory locations

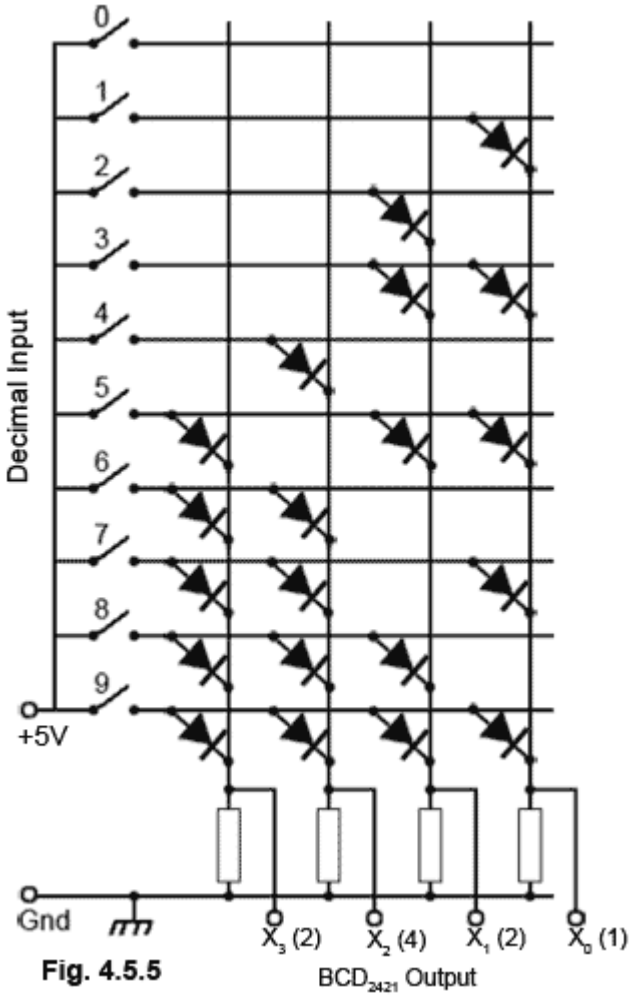


Fig. 4.5.5

