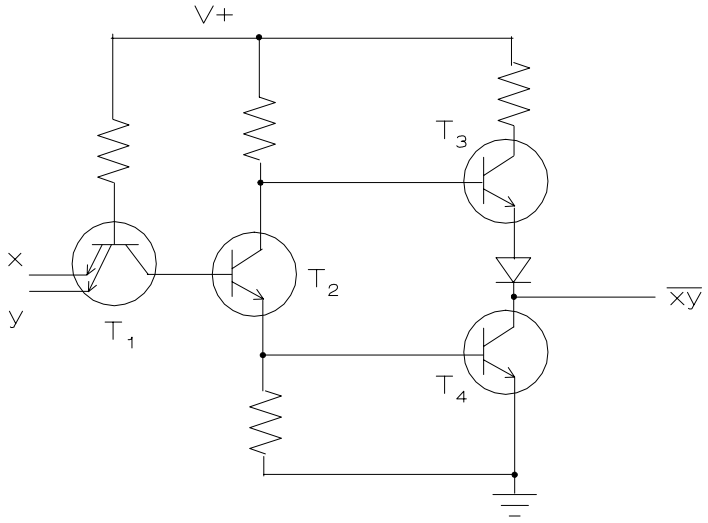
5.Basic Gate Combinations

5.1 TTL NAND Gate



In logic circuits transistors play the role of switches. For those in the TTL gate the conducting state (on) occurs when the base-emmiter signal is high, and the non-conducting (off) when it is low. Signals corresponding to logical variables x and y are applied to the emitters of the input transistor T_1 where the actual logic is essentially performed. Transistor T_2 plays the role of a phase splitter. When it is off, the collector voltage is at essentially V and the emitter is at 0. When it is conducting current I_2 flows through the series resistors. The collector voltage drops to $V-I_2R$ and the emitter voltage rises to I_2R, where R is the appropriate resistor.
The collector and emitter voltages then vary in opposite phase. Moreover each phase leads to only one of the two output transistors being conducting. Note that if either x or y is low (ie =0) then T_1 is on and the base voltage is at its lowest value $V-I_1R$. Here again R is used generically, each element having distinct values in practice. The collector of T_1 and the base of T_2 are low so that T_2 is off. In this situation the base of T_3 is high turning this transistor on while the base of T_4 is 0 and this transistor is off. In this situation the output is connected to the high voltage line V and goes high. When connected to a load it will deliver current.

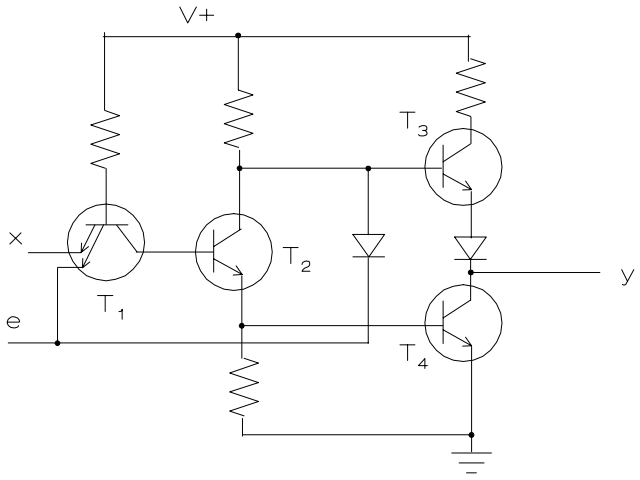
In the case when both x and y are high then T_1 is off, turning T_2 on. This reduces the voltage on the base of T_3 below its emitter voltage so that it becomes non-conducting. The base of T_4 on the other hand is driven up by an amount I_2R and becomes conducting. In this case the output is connected to ground through T_4 and the output is low. Note that in this state, when connected to a load, current will flow into the gate.

The operation is summarized in the following table.

x	y	T_1	T_2	T_3	T_4	output
0	0	on	off	on	off	1
0	1	on	off	on	off	1
1	0	on	off	on	off	1
1	1	off	on	off	on	0

The pair of transistors T_3 and T_4 constitute what is known as a totem pole output. They are arranged so that only one of the two is conducting in a steady state condition.

5.2 Tristate Drivers

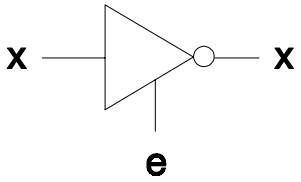


A variation on the NAND gate in which a diode connects the base of T_3 to one input, labelled e, profoundly effects the operation. In the normal gate the condition with one input low leads to the state with T_3 on and T_4 off. In this case however if e is low, the base of T_3 is held low and both of these transistors are off. If one thinks of the arrangement as a series of two switches then the output is now connected to the centre of two disconnected switches, so in essence the circuit is completely removed from the load. This state is not one of the two logic conditions and is referred to as the high Z state, since it represents a high impedance output point. The system behaves normally when e is high, having a 0 output for x=1, and a 1 output for x=0. Thus the system may be thought of as having three states and is referred to as a tristate device. The behaviour is summarized in the table. The input e acts as a control or more specifically enable signal. When the gate is not enabled, the output is in the high Z state no matter what the value of x. The entry X stands for either 0 or 1, referred to as "don't care". When the gate is enabled the device behaves as a NOT

gate or logic inverter with respect to the x input.

e	x	T_1	T_2	T_3	T_4	output
0	X	on	off	off	off	high Z
1	0	on	off	on	off	1
1	1	off	on	off	on	0

The circuit symbol for the tristate is shown in the diagram. This circuit is designed solely for the purpose of connection to one of the conducting paths of a bus. Since this conducting path, or line, is shared by several devices it is necessary to require that only one device access the line at any one time. This is achieved by connecting the device output through a tristate driver to the line. Only one of the set of devices has a tristate driver to which it is connected enabled at any time. In this way all other device outputs are in effect disconnected.



5.3 Truth tables and Karnaugh maps

An arrangement of gates which accepts a set of logical variables or in other words a logical word as inputs and produces a set of logical variables which are prescribed logical functions of the input variables (constituting the output word) is a combinational circuit. Since the properties of logical functions are completely defined by the truth table, the table plays an important role in the development of the circuit. A logical function of N variables can be written f(X) where X is the N-bit word consisting of the variables. Thus if the function is dependent on the states of three variables and is thus written f(u,v,w), X is the three bit word uvw. The domain of a function of N variables is the register set $\mathbb{R}_N$. The truth table is constructed by entering the value assumed by f for each word X in $\mathbb{R}_N$. The words are ordered according to the value of $\mathbb{I}_{10}(X)$. It is convenient to indicate this by writing the set as $X_j, j=0,2^N-1$, so that the subscript of X_j is the value $j=\mathbb{I}_{10}(X_j)$. Special logical functions called minterms $m_j(X)$ are defined by the relation

$$m_j(X_k)=\delta_{jk} \qquad \textbf{(5.1)}$$

where δ_{jk} is the Kronecker delta. The explicit form is the AND of each variable in the 1 state and the complement of each variable in the 0 state.

For convenience a general OR of several terms will be designated as Σ'. It then follows that

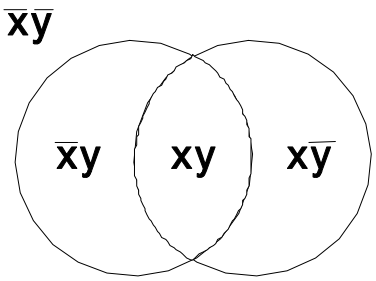
$$f(X)=\sum{}'f(X_j)m_j(X) \qquad \textbf{(5.2)}$$

This is the logical analogue of the statement

$$f(x)=\int_{-\infty}^{\infty} f(x')\delta(x-x')dx' \qquad \textbf{(5.3)}$$

for continuous real functions.

A rearrangement of the presentation of the truth table is the Karnaugh map, useful in simplifying expressions. The evolution of this arrangement may be understood by examining the Venn diagram for two sets x and y. Note that in terms of logical variables a natural order is equivalent to 00, 01, 11, 10 in contrast to the order based upon the integer code, that is 00, 01, 10, 11. Here it is understood that, for example 01 represents (NOT x) AND y. The reason for the usefulness of this ordering is that it makes obvious the presence of terms in which a common factor is in the AND condition with a variable OR its complement. Since the OR of a variable and its complement is always 1 then this variable may be removed from the expression. Thus one has



$$\bar{x}y\vee xy=(\bar{x}\vee x)y=y \qquad \textbf{(5.4)}$$

The Karnaugh map presentation highlights the existence of such redundancies allowing simplification based upon the truth table rather than algebraic manipulation. This becomes more important for more than two variables. As an example consider the von Neumann function of three variables which is 1 when at least two of the three variables is 1. The truth table is given below. From Eqn(5.2) the function could be written

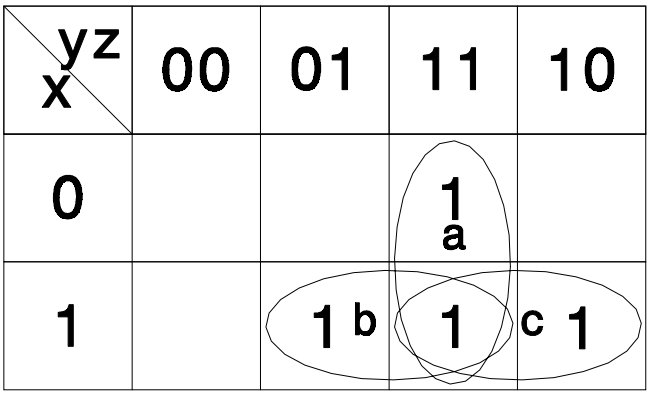
$$\begin{aligned} f(X)&=m_3(X)\vee m_5(X)\vee m_6(X)\vee m_7(X)\\ &=\bar{x}yz\vee x\bar{y}z\vee xy\bar{z}\vee xyz \end{aligned} \qquad \textbf{(5.5)}$$

In the second line of the equation the explicit functions for each minterm, ie each table entry X_j, is given. Because these are ANDed

with the corresponding fnction value only those entries appear for which the function is 1.

x	y	z	f(X)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

The Karnaugh map is arranged in a 2x4 matrix form with the 4 combinations of the variables y and z arranged in the order mentioned from the Venn diagram and x as a single variable having



only the two possibilities.
Only those combinations, 011,101,111 and 110 for which f(X)=1 are entered as is standard practice. Because of the arrangement adjacent pairs represent simplification. Thus the pair labelled a represent the combination yz since x may be 0 OR 1, the pair labelled b correspond to xz since these are only 1 while y may be 0 OR 1 and c produces xy since for this pair x and y are 1 but z is 0 OR 1.

The function may be thus written

$$f(X)=yz\vee xz\vee xy \qquad \textbf{(5.6)}$$

Clearly it must be possible to simplify Eqn(5.5) to the above form since both expressions are based upon the same truth table. The four terms in Eqn(5.5) correspond to the four entries in the map. If one were to attempt to simplify the expression as given, one could factor the last two terms according to

$$\begin{aligned} f(X)&=\bar{x}yz\vee x\bar{y}z\vee xy(\bar{z}\vee z)\\ &=\bar{x}yz\vee x\bar{y}z\vee xy \end{aligned} \qquad \textbf{(5.7)}$$

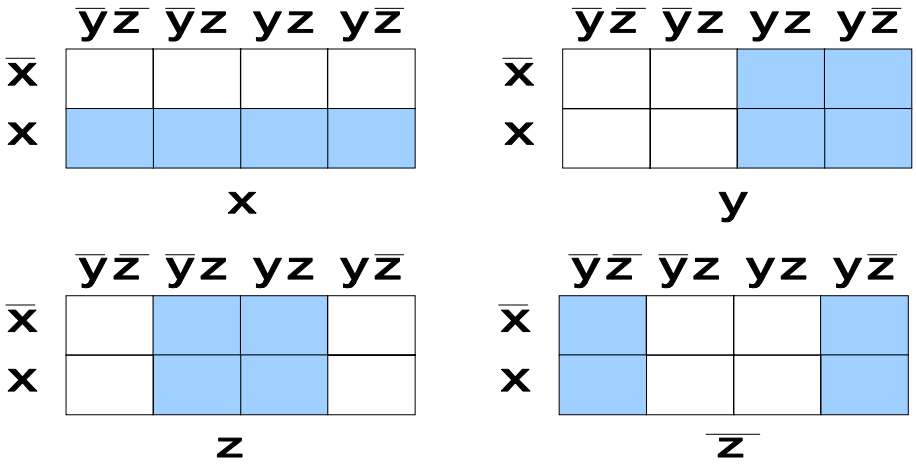
While this expression is reduced to three terms it is still not as simple as possible and further simplification does not appear possible. Note that the grouping leading to the last term is the pairing labelled c in the map, and makes use of the term xyz. Also notice that in the map reduction the term xyz is used in all three pairs. This points to the resolution of the dilemma.

The method of algebraic simplification in this case relies on the obvious fact that any logical variable a satisfies a$\vee$a=a. Then it is possible to augment the expression in Eqn(5.5) by two more xyz terms giving

$$\begin{aligned} f(X)&=\bar{x}yz\vee x\bar{y}z\vee xy\bar{z}\vee xyz\vee xyz\vee xyz\\ &=yz(x\vee\bar{x})\vee xz(y\vee\bar{y})\vee xy(z\vee\bar{z})\\ &=yz\vee xz\vee xy \end{aligned} \qquad \textbf{(5.8)}$$

The terms in the second and third lines of Eqn(5.8) correspond to the pairings indicated as a,b and c respectively in the Karnaugh map of f(X). This example illustrates the power of the Karnaugh map approach in simplification of expressions. Algebraic simplification in this case would only have been possible if the required trick of augmenting the expression with redundant terms had been employed. This procedure was only made evident in this context by prior reduction using the map approach, and it is doubtful that it would otherwise have been recognized.

The major groupings are exemplified in the figure following. The structure of the map is indicated in terms of the corresponding minterm explicit functions rather than the binary words which they select. The shaded areas represent those input words for which the output is one. Each single variable occupies 4 contiguous positions. Note from the map for the complement of z that the end regions are contiguous so that one must treat the map as if it wrapped around.



A four variable map is formed as a symmetrical 4x4 array, with the 4 rows generated by two variables and the 4 columns the remaining two. This approach becomes difficult for more than 4 variables.

5.4 Adders

The fundamental arithmetic operation is addition of two bits corresponding to two logical variables. In this case the outcome must in general be a two-bit word. Consider 1+1=10. The resultant word is 10. The 0 is referred to as the sum bit and will be designated here by s and the resultant 1 is referred to as the carry bit,c. When adding words of more than one bit, the operation commences with the two least significant bits and proceeds in sequence to higher order bits. For all but the LSB this requires that the sum of the two bits together with the carry from the operation performed on the bits of one less order be added. An adder element is thus a device with a three-bit input word and a two-bit output word. Consider the addition of two N-bit words $X=\{x_i,i=1,N\}$ and $Y=\{y_i,i=1,N\}$. Then N adder elements will be required. For the i^{th} element, the input word is $x_iy_ic_{i+1}$ and the output word is s_ic_i. The following truth table defines the output functions. In principle Karnaugh maps may be constructed for each of the outputs, or the functions may be obtained directly from the minterms. In either case scrutiny of the behaviour of c_i reveals that it is true when at least any two inputs are high and hence is the von Neumann function discussed in the previous section. Similarly the sum bit is the parity of the input word. This is of course realized as the XOR of the input variables. Hence the solution may be obtained by inspection and a detailed analysis is

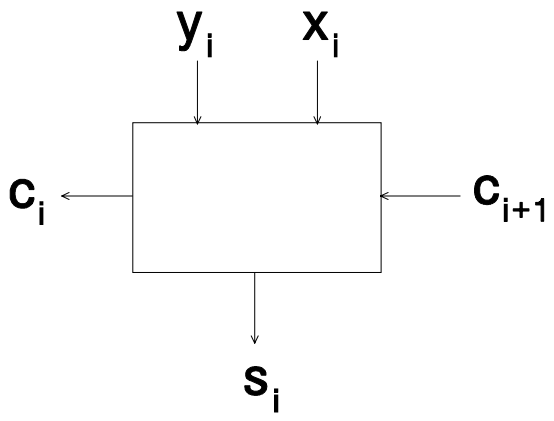
not really necessary.

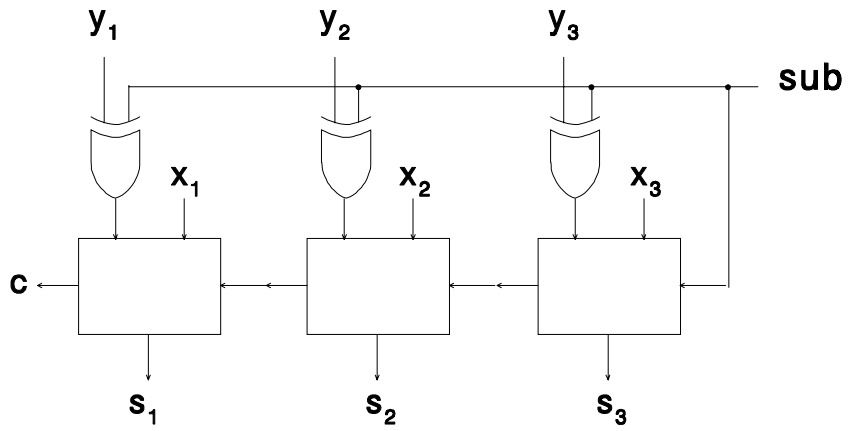
x_i	y_i	c_{i+1}	c_i	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

In summary the input-output relations for an adder element are

$$s_i = x_i \oplus y_1 \oplus c_{i+1}$$
$$c_i = x_i y_i \vee x_i c_{i+1} \vee y_i c_{i+1} \qquad \textbf{(5.9)}$$

The relationships in Eqn(5.9) define the gate arrangements required. The sum bit is the output of a 3-input XOR gate, the inputs of which are the circuit inputs. The carry bit is the output of a 3-input OR gate the inputs of which are in turn outputs of three, 2-input AND gates. The inputs to the AND gates are the three possible pairs of the three device input variables. A short form symbol for an element is shown in the figure. One such element is required for each bit of the words to be added. The elements are interconnected at the carry inputs. The device is often referred to as a full adder since a half adder is defined as one which does not take into account the carry.





A circuit which can add or subtract three-bit numbers using the twos complement code is shown above. Recall that the XOR gate behaves as a controlled inverter. When sub=0, the outputs of the three XOR gates equal the inputs so the device forms a normal addition. When sub=1 however, the output of each XOR gate is the complement of the input. This accomplishes the addition of X and the complement of Y. The sub signal is also coupled to the input carry of the lowest order element so one is also added.

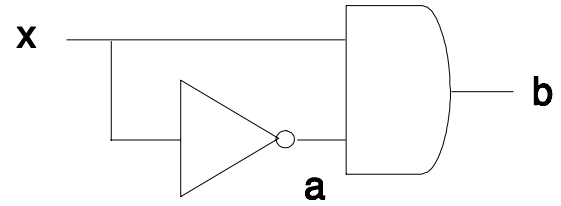
5.5 Propagation delay

The objective of a combinational circuit is to produce an output word from an input word according to a defined logical function. This is achieved in the steady state. Of course the systems in which the circuit operates are dynamic and the input word changes. The transition from the old word to the new word, or from the initial to the final state is somewhat problematic in several respects.

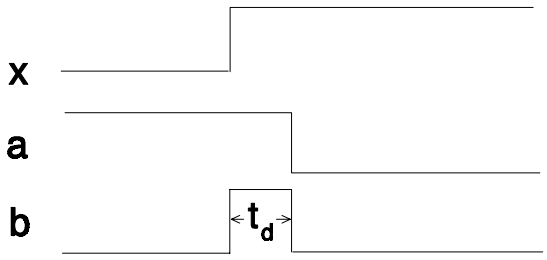
First consider a single bit of the word. In changing say from a one to a zero, there is a time period during which the voltage level is

not consistent with either of the two states. During this period the value is indeterminate.

Second suppose the bits undergo the transitions at different times. There will then be a series of incorrect transient words. Clearly synchronization of the transition across bits is desirable.



Finally no system can respond instantaneously to a stimulus. In the case of a logic gate the response is characterized by a time delay between the application of a new level at the input and the formation of that level at the output. This is referred to as the propogation delay.



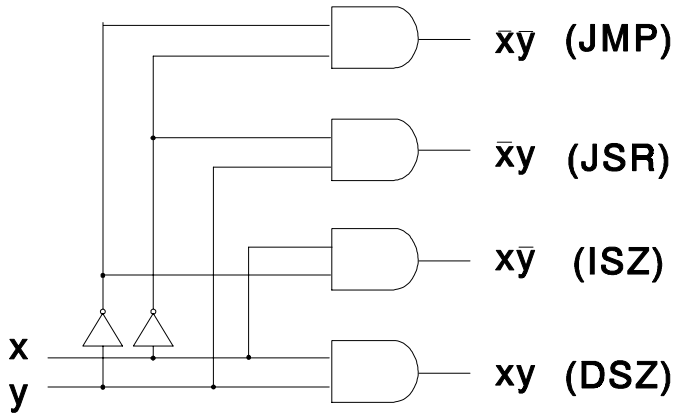
The effect of such a delay may be illustrated by the simple circuit shown for which the output logic function for input x is x ANDed with NOT x and hence should be 0 at all times. As shown in the figure however, the output of the inverter, labelled a requires a time t_d before the final state level is established. Thus if x undergoes a transition from 0 to 1 at t=0, the output remains at 1 for $0 \le t \le t_d$. During this period the inverter input-output relation is not satisfied, and since both inputs to the AND gate are high, the output b is 1. After the propogation delay, the inverter output transition to the correct level, 0, is completed and the AND gate output returns to the expected value. Thus there is a transient period during which the truth table for the circuit is not in accord with its actual performance. The aberrant output pulse at b is called a glitch. Clearly it is necessary to allow for such delays in using the outputs of circuit elements in a digital system.

Consider the case of the adder discussed above. The correct sum output will not occur until the carry has propogated through all higher order bits. This can take up to Nt_d for an N-bit adder, and thus this time imposes a limitation on the speed at which the addition operation may be performed.

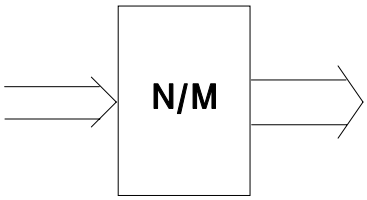
5.6 Decoders

A decoder is a device with N inputs which may have up to 2^N outputs. A given output becomes high when the input word is a specific word of $\mathbb{R}_N$. Thus each output is a minterm. A two input four output decoder is shown in the following diagram. The outputs,

in order of top to bottom correspond to $m_0(X)$, $m_1(x)$, $m_2(X)$ and $m_3(X)$ where X=xy. The most common application of a decoder is in decoding



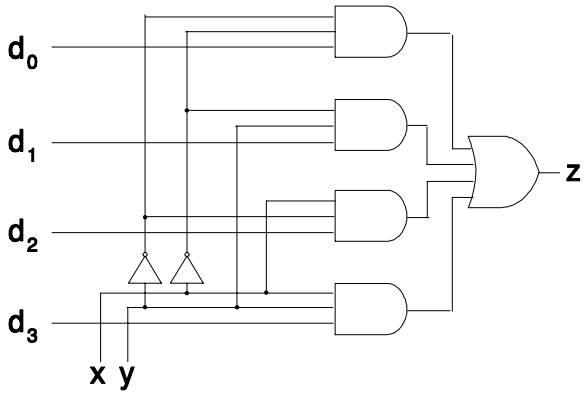
instructions. The example indicated in the diagram corresponds to the situation in which the input is taken from the fourth and fifth bits of a NOVA memory reference instruction without accumulator. A 00 input coding for an unconditional jump would then activate the m_0 line. This signal would be used to load the program counter with the address of the next instruction to be performed. Similarly, for each of the 4 possible instructions the corresponding output line would be arranged to initiate the appropriate action when in the 1 state. As another example, when outputting ASCII code words to a device such as a printer each output line of the decoder initiates the required action to form the corresponding character.



A simple symbol for a decoder is shown in the accompanying illustration. The double arrows indicate the set of lines corresponding to a given word. These would number N for an N-bit input word and up to 2^N for the M-bit output. In cases where it is necessary to avoid confusion the number of bits represented by the double

arrow can be written explicitly.

5.7 Multiplexers



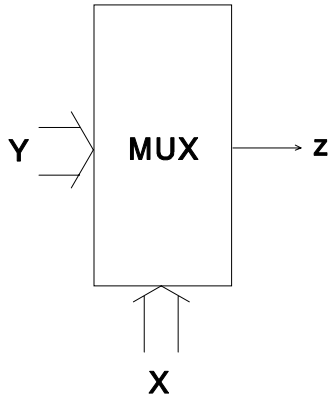
The multiplexer is named such since it is designed to perform the process known as multiplexing. In this process several input lines of information are brought to-gether to share a common single line. A separate control input determines which input line is connected to the otput at any given time. As can be seen from the figure the multiplexer is based upon the decoder function which responds to the control input by enabling the appropriate AND gate. The data input lines are labelled d_i where $i=\mathbb{I}_{10}(X)$ is the binary coded integer of the control word X=xy. Thus if X=10, $\mathbb{I}_{10}(X)=2$ and the two upper inputs to the third AND gate from the top are set to 1. This gate then acts like a connected switch passing the information contained on d_2 to the output. The connected input line may also be thought of as selected by the minterm of corresponding subscript.

It is easy to see by inspection that the general expression for the output which formally summarizes the above discussion is

$$z=\bar{x}\bar{y}d_0 \vee \bar{x}yd_1 \vee x\bar{y}d_2 \vee xyd_3 \qquad \textbf{(5.11)}$$

This equation may be made identical to Eqn (5.2) if the substitution $d_j=f(X_j)$ is made. As mentioned above the data bit is ANDed with the corresponding minterm. Of course while this example is based upon a 2/4 decoder it can be generalized to the case $N/2^N$. This is an extremely important result because it means that any logical function f(X) can be generated using a multiplexer in a

straight forward manner. The control word assumes a minterm value so that if the associated data line is set to the corresponding value of the truth table, then this value appears at the output for the appropriate X. For example XOR is realized by setting the data lines to $d_0=0$, $d_1=1$, $d_2=1$ and $d_3=0$. The use of a multiplexer to realize an arbitrary function in such a straightforward manner avoids the intricacies of function simplification, particularly important for functions of many variables. Since multiplexers are readily available function generation is easily accomplished. The symbol shown to the right makes clear that the multiplexer requires two input words and produces a one-bit word output. Note that the subscripts used for the bits of the data lines differ from the convention for the bits of the input word Y.



From the discussion above it would appear that realization of a function of N variables would require a multiplexer with an N-bit control word X. In fact however a carefule examination of the structure of the truth table indicates that the function can be generated by a multiplexer with a contol input word of length N-1.

x	y	u	z	
0	0	0	0	$d_0=0$
0	0	1	0	
0	1	0	0	$d_1=u$
0	1	1	1	
1	0	0	0	$d_2=u$
1	0	1	1	
1	1	0	1	$d_3=1$
1	1	1	1	

Suppose one wishes to realize the von Neumann function for the variables x,y,u. Consider the truth table decomposed as shown below. The N variables are decomposed into a control word of the N-1 MSBs and a single bit, the LSB. In this example N=3. The truth table is decomposed into 2^{N-1} subtables, based upon the words of $\mathbb{R}_{N-1}$. Each subtable is a truth table for the single variable comprising the LSB, designated u in this example. Since the control word connects the output to the corresponding data line the solution z(u) for each subtable, which can be obtained by inspection, defines the signal on that data line. Again in this example the data lines are connected low for d_0, to the LSB for lines d_1 and d_2 and high for d_3.