MULTIPLEXER

Advanced Digital Electronics

ألألكترونيات الرقمية المتقدمة

(stage 4^{th} ***Branch Electronic***)

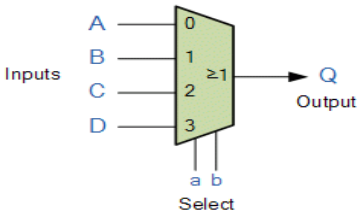
A U C Almaaref University Collage

Department of Computer Engineering Techniques

K.DAWAH .ABBAS

Department of Computer Engineering Techniques, College of Engineering, University of Al Maarif, Al Anbar,31001, Iraq

MULTIPLEXER



The Multiplexer

The multiplexer is a combinational logic circuit designed to switch one of several input lines to a single common output line

DAWAH

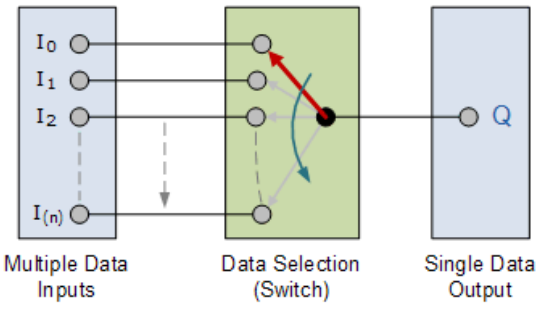
Multiplexing is the generic term used to describe the operation of sending one or more analogue or digital signals over a common transmission line at different times or speeds and as such, the device we use to do just that is called a **Multiplexer**.

The *multiplexer*, shortened to "MUX" or "MPX", is a combinational logic circuit designed to switch one of several input lines through to a single common output line by the application of a control signal. Multiplexers operate like very fast acting multiple position rotary switches connecting or controlling multiple input lines called "channels" one at a time to the output.

Multiplexers, or MUX's, can be either digital circuits made from high speed logic gates used to switch digital or binary data or they can be analogue types using transistors, MOSFET's or relays to switch one of the voltage or current inputs through to a single output.

The most basic type of multiplexer device is that of a one-way rotary switch as shown.

Basic Multiplexing Switch



The rotary switch, also called a wafer switch as each layer of the switch is known as a wafer, is a mechanical device whose input is selected by rotating a shaft. In other words the rotary switch is a manual switch that you can use to select individual data or signal lines simply by turning its inputs "ON" or "OFF". So how can we select each data input automatically using a digital device.

MULTIPLEXER

In digital electronics, multiplexers are also known as data selectors because they can "select" each input line, are constructed from individual Analogue Switches encased in a single IC package as opposed to the "mechanical" type selectors such as normal conventional switches and relays.

They are used as one method of reducing the number of logic gates required in a circuit design or when a single data line or data bus is required to carry two or more different digital signals. For example, a single 8-channel multiplexer.

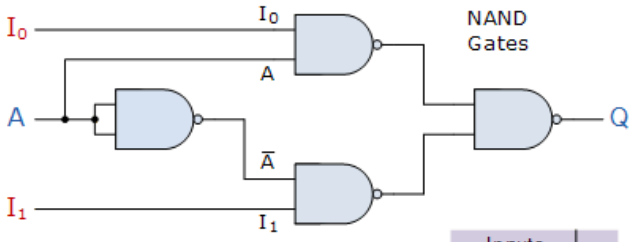
Generally, the selection of each input line in a multiplexer is controlled by an additional set of inputs called *control lines* and according to the binary condition of these control inputs, either "HIGH" or "LOW" the appropriate data input is connected directly to the output. Normally, a multiplexer has an even number of 2^n data input lines and a number of "control" inputs that correspond with the number of data inputs.

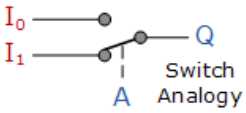
Note that multiplexers are different in operation to *Encoders*. Encoders are able to switch an n-bit input pattern to multiple output lines that represent the binary coded (BCD) output equivalent of the active input.

We can build a simple 2-line to 1-line (2-to-1) multiplexer from basic logic NAND gates as shown.

MULTIPLEXER

2-input Multiplexer Design





Truth Table

Inputs			
A	I_1	I_0	Q
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

The input A of this simple 2-1 line multiplexer circuit constructed from standard NAND gates acts to control which input (I_0 or I_1) gets passed to the output at Q.

From the truth table above, we can see that when the data select input, A is LOW at logic 0, input I_1 passes its data through the NAND gate multiplexer circuit to the output, while input I_0 is blocked. When the data select A is HIGH at logic 1, the reverse happens and

MULTIPLEXER

now input I_0 passes data to the output Q while input I_1 is blocked.

So by the application of either a logic "0" or a logic "1" at A we can select the appropriate input, I_0 or I_1 with the circuit acting a bit like a single pole double throw (SPDT) switch.

As we only have one control line, (A) then we can only switch 2^1 inputs and in this simple example, the 2-input multiplexer connects one of two 1-bit sources to a common output, producing a 2-to-1-line multiplexer. We can confirm this in the following Boolean expression.

$$Q = \overline{\overline{A}.I_{\overline{0}}}.I_1 + \overline{A}.I_0.I_1 + A.I_0.\overline{I_{\overline{1}}} + A.I_0.I_1$$

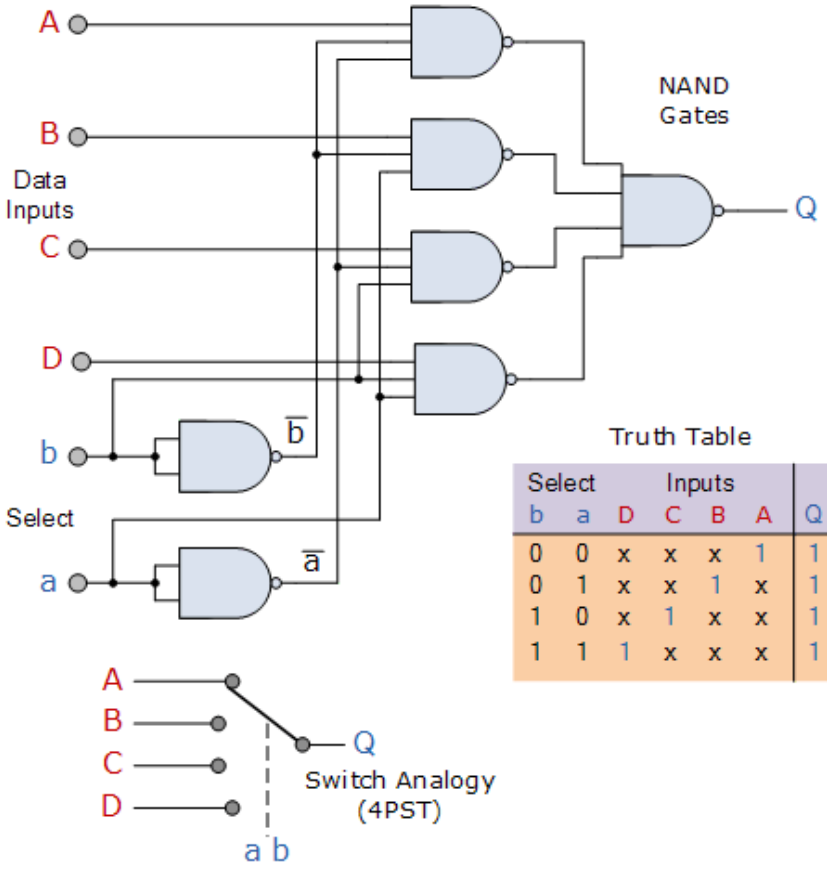
and for our 2-input multiplexer circuit above, this can be simplified too:

$$Q = \overline{A}.I_1 + A.I_0$$

We can increase the number of data inputs to be selected further simply by following the same procedure and larger multiplexer circuits can be implemented using smaller 2-to-1 multiplexers as their basic building blocks. So for a 4-input multiplexer we would therefore require two data select lines as 4-inputs represents 2^2 data control lines give a circuit with four inputs, I_0, I_1, I_2, I_3 and two data select lines A and B as shown.

MULTIPLEXER

4-to-1 Channel Multiplexer



Truth Table

Select		Inputs				
b	a	D	C	B	A	Q
0	0	x	x	x	1	1
0	1	x	x	1	x	1
1	0	x	1	x	x	1
1	1	1	x	x	x	1

MULTIPLEXER

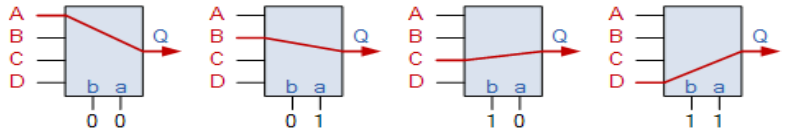
The Boolean expression for this 4-to-1 **Multiplexer** above with inputs A to D and data select lines a, b is given as:

$$Q = \overline{ab}A + a\bar{b}B + \bar{a}bC + abD$$

In this example at any one instant in time only ONE of the four analogue switches is closed, connecting only one of the input lines A to D to the single output at Q. As to which switch is closed depends upon the addressing input code on lines "a" and "b".

So for this example to select input B to the output at Q, the binary input address would need to be "a" = logic "1" and "b" = logic "0". Thus we can show the selection of the data through the multiplexer as a function of the data select bits as shown.

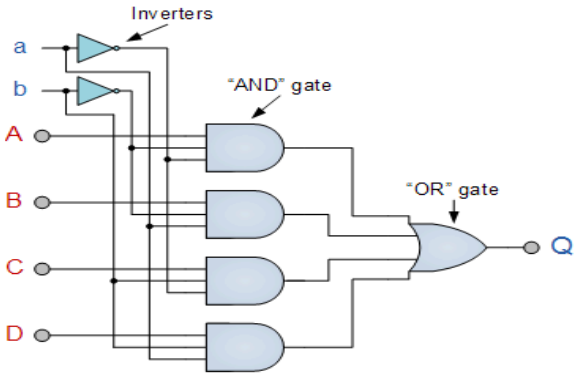
Multiplexer Input Line Selection



Adding more control address lines, (n) will allow the multiplexer to control more inputs as it can switch 2^n inputs but each control line configuration will connect only ONE input to the output.

Then the implementation of the Boolean expression above using individual logic gates would require the use of seven individual gates consisting of AND, OR and NOT gates as shown.

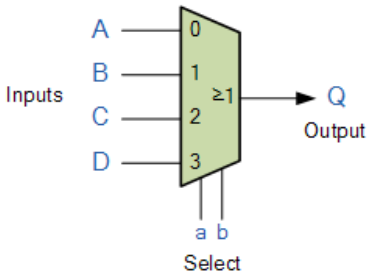
4 Channel Multiplexer using Logic Gates



The symbol used in logic diagrams to identify a multiplexer is as follows:

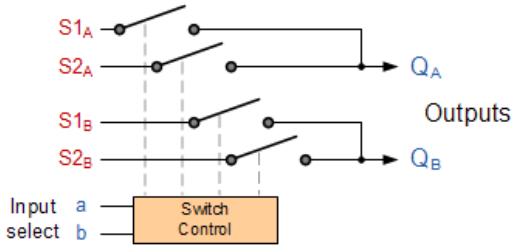
MULTIPLEXER

Multiplexer Symbol



Multiplexers are not limited to just switching a number of different input lines or channels to one common single output. There are also types that can switch their inputs to multiple outputs and have arrangements or 4-to-2, 8-to-3 or even 16-to-4 etc configurations and an example of a simple Dual channel 4 input multiplexer (4-to-2) is given below:

4-to-2 Channel Multiplexer



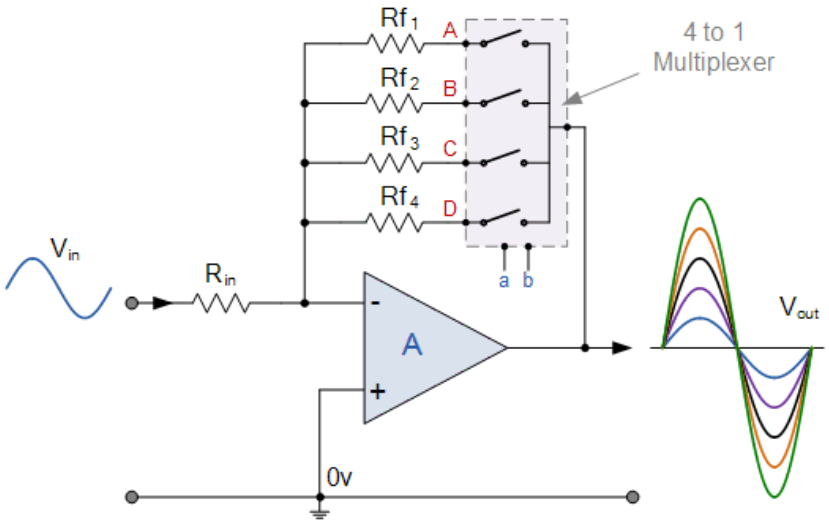
Here in this example the 4 input channels are switched to 2 individual output lines but larger arrangements are also possible. This simple 4-to-2 configuration could be used for example, to switch audio signals for stereo pre-amplifiers or mixers.

Adjustable Amplifier Gain

As well as sending parallel data in a serial format down a single transmission line or connection, another possible use of multi-channel multiplexers is in digital audio applications as mixers or where the gain of an analogue amplifier can be controlled digitally, for example.

MULTIPLEXER

Digitally Adjustable Amplifier Gain



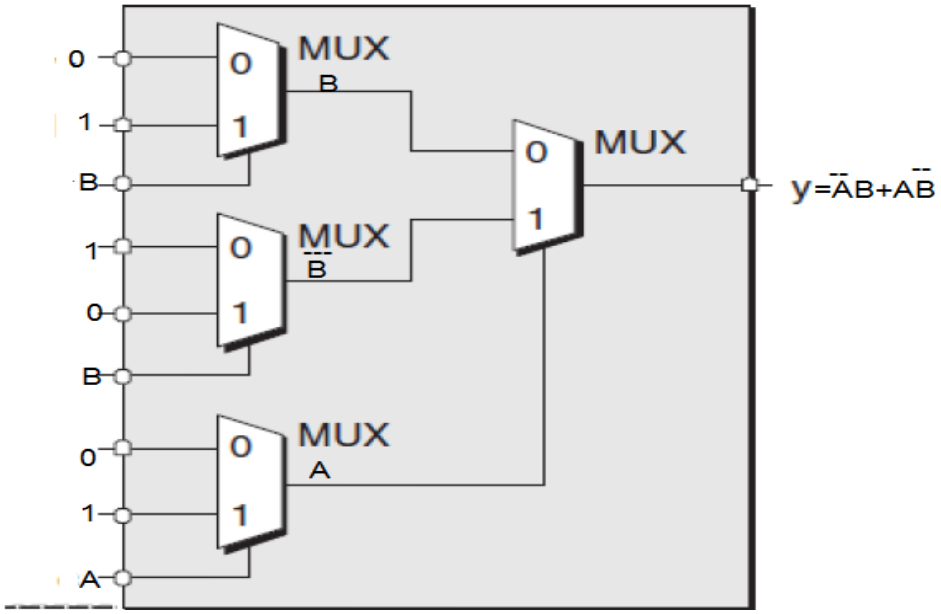
Here, the voltage gain of the inverting operational amplifier is dependent upon the ratio between the input resistor, R_{IN} and its feedback resistor, Rf as determined in the Op-amp

EXAMPLE

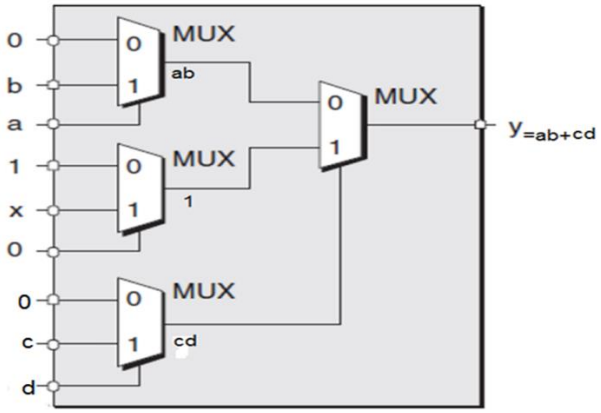
Implement the following functions using MUX-based Logic Block:

1- Y=A⊕ B

MULTIPLEXER



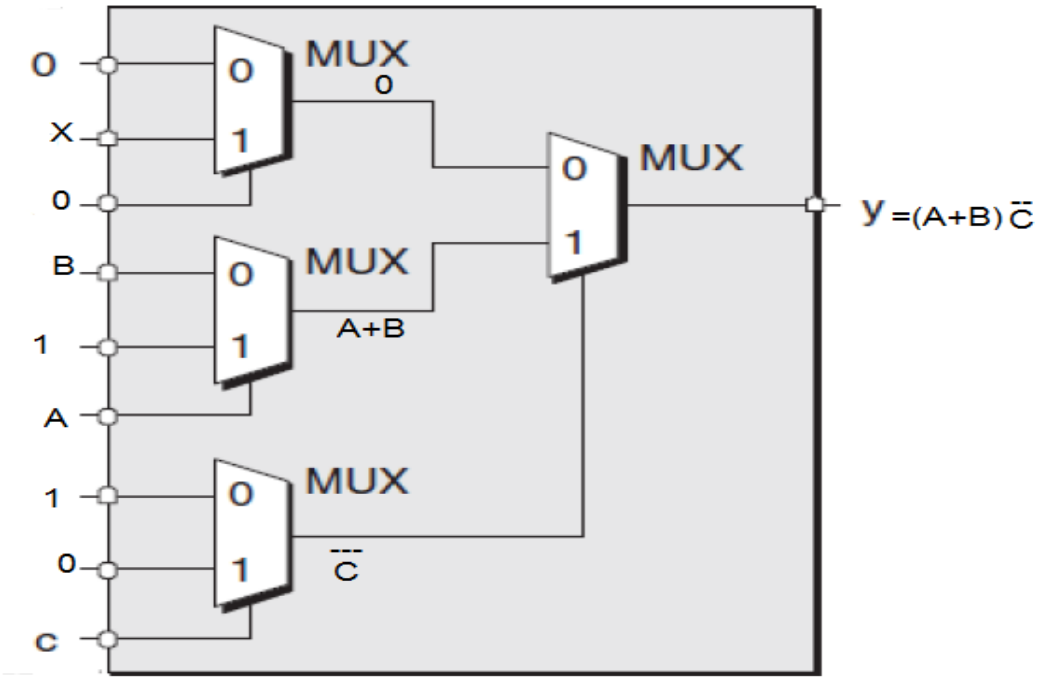
2- y=ab+cd



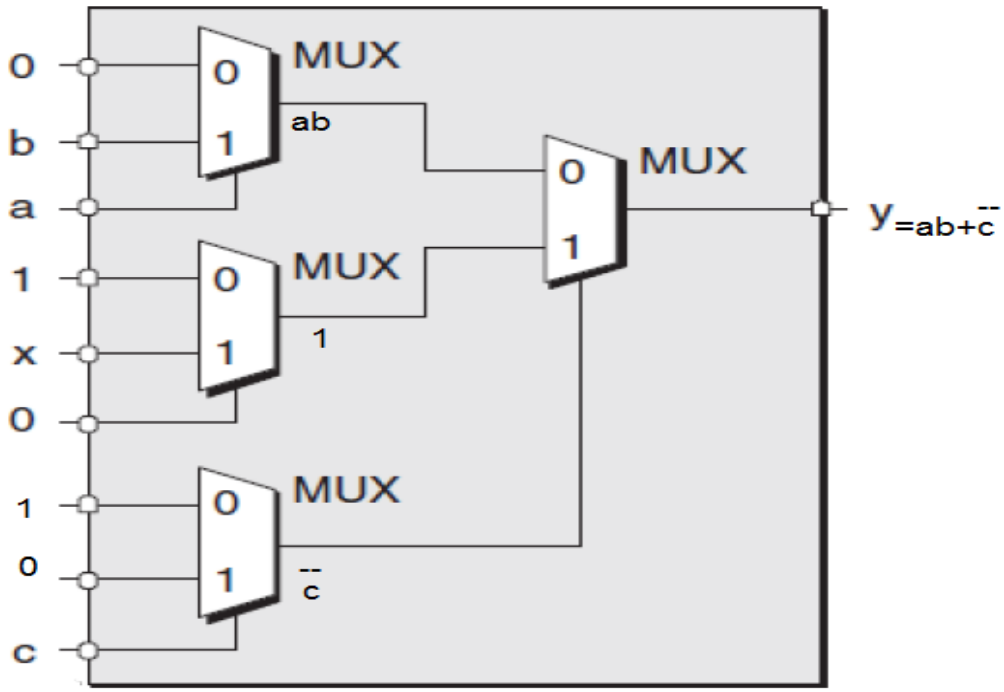
3-Y=(A+B)$\overline{C}$

Department of Computer Engineering Techniques, College of Engineering, University of Al Maarif, Al Anbar,31001, Iraq

MULTIPLEXER



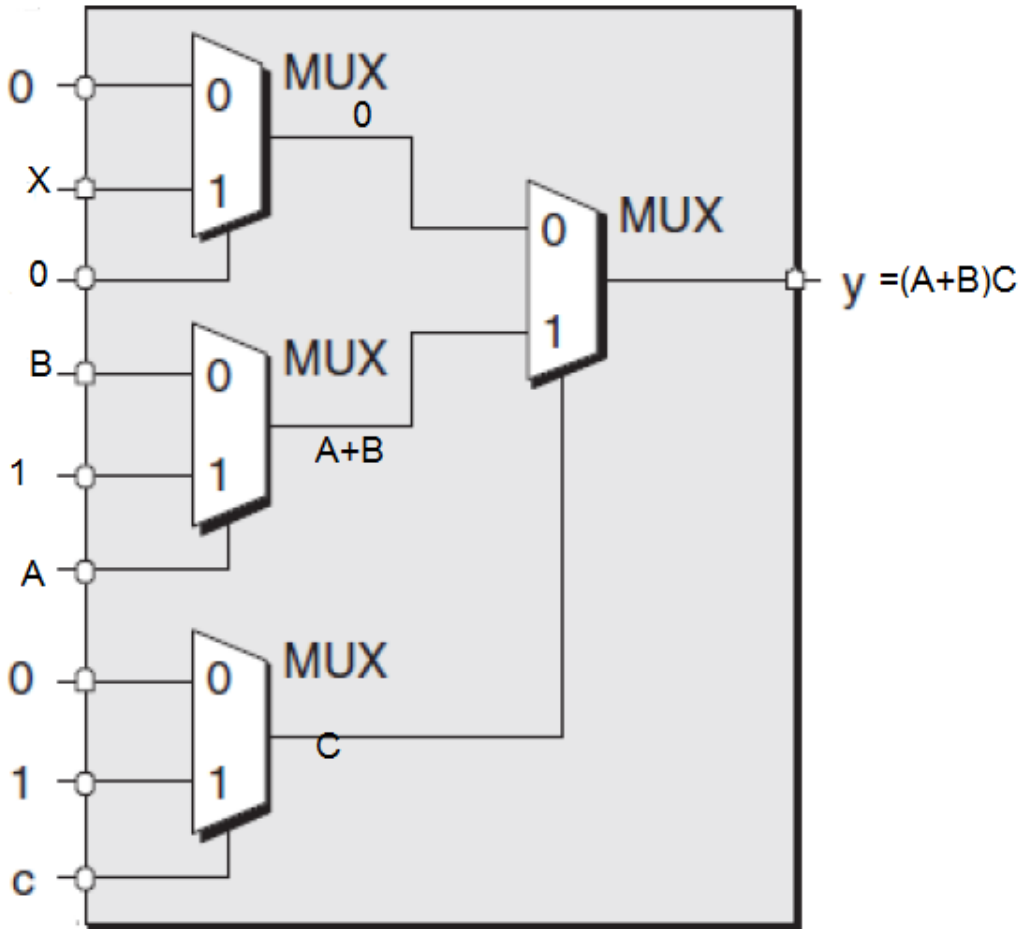
4-Y=AB+$\overline{C}$



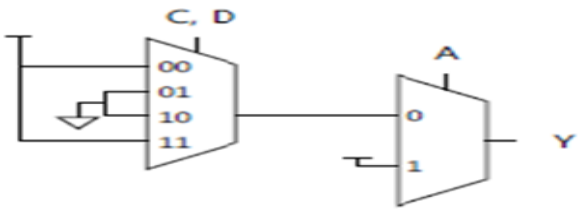
Department of Computer Engineering Techniques, College of Engineering, University of Al Maarif, Al Anbar,31001, Iraq

MULTIPLEXER

5- Y=(A+B)C



Example: Write a minimized Boolean equation for the function performed by the circuit below?



C	D	A	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

MULTIPLEXER

$$Y=\overline{C}\,\overline{D}\,\overline{A}+\overline{C}\,\overline{D}\,A+\overline{C}\,DA+C\overline{D}\,A+CD\overline{A}+CDA$$

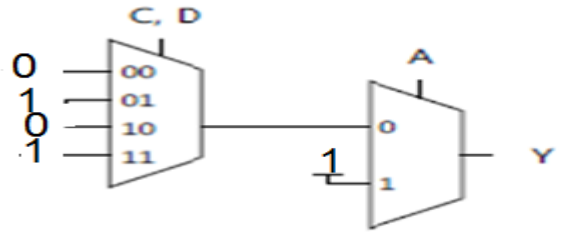
$$Y=\overline{C}\,\overline{D}(\overline{A}+A)+\overline{C}\,DA+C\overline{D}\,A+CD(\overline{A}+A)$$

$$Y=\overline{C}\,\overline{D}+\overline{C}\,DA+C\overline{D}\,A+CD$$

C	D	Y
0	0	1
0	1	A
1	0	A
1	1	1

$Y=\overline{C}\,\overline{D}+\overline{C}DA+C\overline{D}A+CD$

Example: Write a minimized Boolean equation for the function performed by the circuit below?



C	D	A	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$Y=\overline{C}\,\overline{D}\,A+\overline{C}\,D\,\overline{A}+\overline{C}\,DA+C\overline{D}\,A+CD\overline{A}+CDA$$

Department of Computer Engineering Techniques, College of Engineering, University of Al Maarif, Al Anbar,31001, Iraq

MULTIPLEXER

$Y=\overline{C}\,\overline{D}\,A+\overline{C}\,D(\overline{A}+A)+C\,\overline{D}A+CD(\overline{A}+A)$

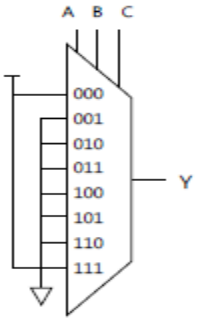
$Y=\overline{C}\,\overline{D}A+\overline{C}\,D+C\overline{D}\,A+CD$

$Y=\overline{D}A(\overline{C}+C)+D(\overline{C}+C)=A\,\overline{D}+D$

Example: Implement from the table .a)8x1MUX . b) 4x1 Mux

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

(a) an 8:1 multiplexer

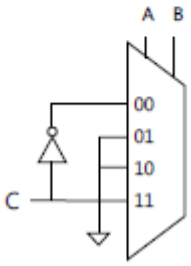


b)4x1MUX

MULTIPLEXER

4:1 multiplexer and one inverter

A	B	Y
0	0	$\overline{C}$
0	1	0
1	0	0
1	1	C



we use an 8 to 1 Mux. If we choose to connect *A*, *B*, and *C* to the inputs of the Multiplexer, then for each combination of *A*, *B* and *C*, although only one Mux input is selected, we need to realize two potentially different values of F. If they are different, they will be functions of D (either D or $\overline{D}$). To see this, we partition the truth table into sections where A, B and C have the same value. This is shown in Figure

A	B	C	D	F	
0	0	0	0	0	0
0	0	0	1	0	
0	0	1	0	1	1
0	0	1	1	1	
0	1	0	0	0	*D*
0	1	0	1	1	
0	1	1	0	0	*D*
0	1	1	1	1	
1	0	0	0	0	0
1	0	0	1	0	
1	0	1	0	1	1
1	0	1	1	1	
1	1	0	0	0	0
1	1	0	1	0	
1	1	1	0	0	*D*
1	1	1	1	1	

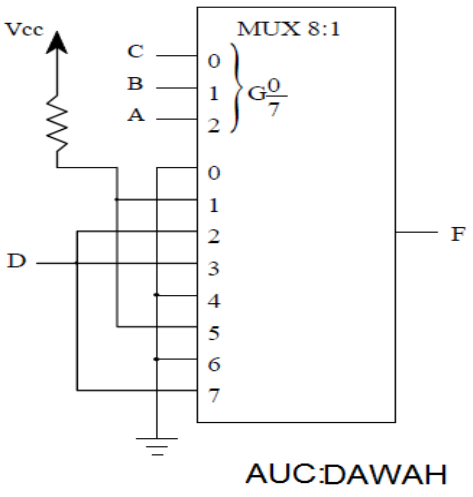
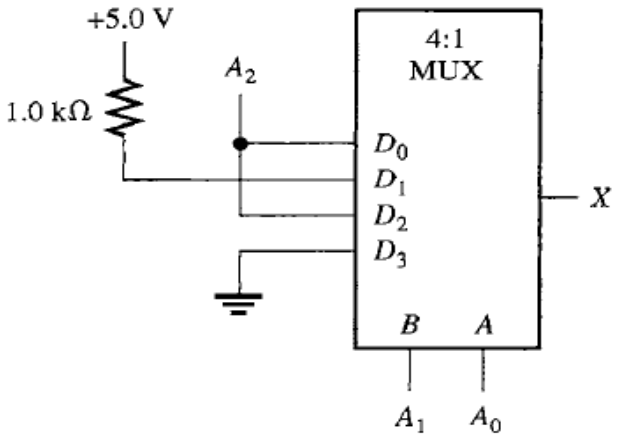


Figure : An 8 to 1 Multiplexer Implementation

MULTIPLEXER

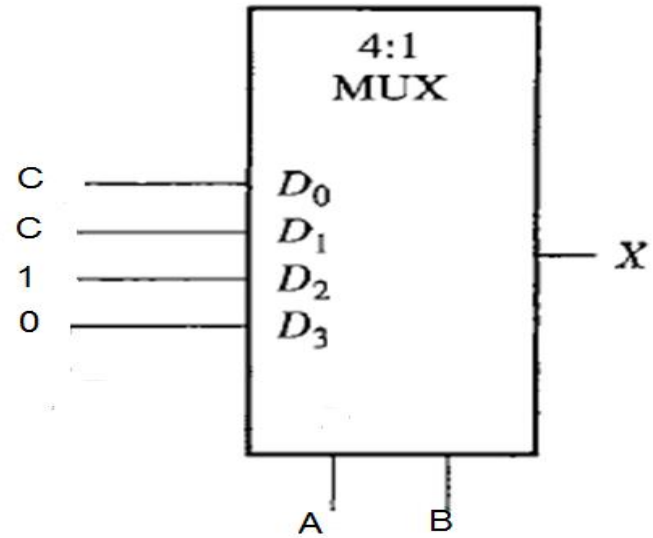
Example: find the Boolean function for the circuit below



Solution: $= \overline{A_2}\,\overline{A_1}A_0 + A_2\overline{A_1}\,\overline{A_0} + A_2\overline{A_1}\,A_0 + +A_2A_1\overline{A_0}$

$=\overline{A_1}A_0\left(\overline{A_2} + A_2\right) + A_2\overline{A_0}\left(\overline{A_1} + A_1\right) = \boldsymbol{\overline{A_1}A_0 + A_2\overline{A_0}}$

EXAMPLE



MULTIPLEXER

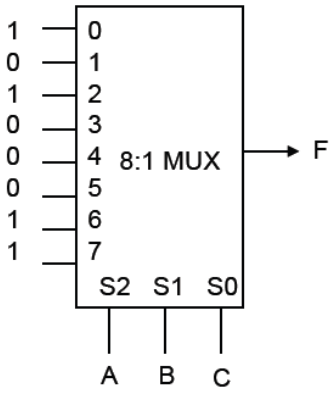
	A	B	X
D0	0	0	C
D1	0	1	C
D2	1	0	1
D3	1	1	0

$$X=\overline{A}\,\overline{B}\,C+\overline{A}\,BC+A\overline{B}+AB0 = \overline{A}C+A\overline{B}$$

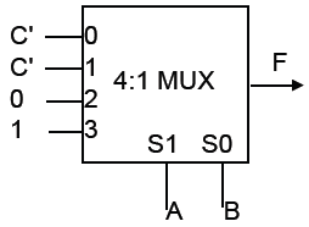
EXAMPLE: F(A,B,C) = m0 + m2 + m6 + m7

= A'B'C' + A'BC' + ABC' + ABC

= A'B'(C') + A'B(C') + AB'(0) + AB(1)



	A	B	C	F	
m0	0	0	0	1	C'
m1	0	0	1	0	
m2	0	1	0	1	C'
m3	0	1	1	0	
m4	1	0	0	0	0
m5	1	0	1	0	
m6	1	1	0	1	1
m7	1	1	1	1	



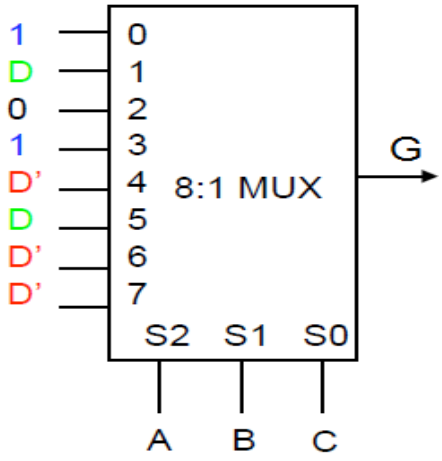
DAWAH

1

Example: G(A,B,C,D) can be realized by an 8:1 MUX choose A,B,C as control variables

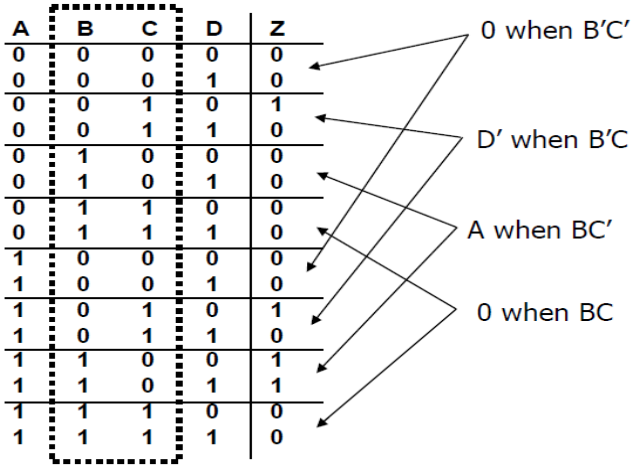
MULTIPLEXER

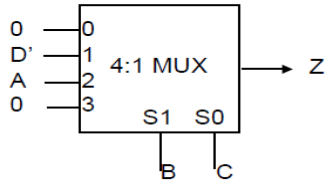
A	B	C	D	G	
0	0	0	0	1	1
0	0	0	1	1	
0	0	1	0	0	D
0	0	1	1	1	
0	1	0	0	0	0
0	1	0	1	0	
0	1	1	0	1	1
0	1	1	1	1	
1	0	0	0	1	D'
1	0	0	1	0	
1	0	1	0	0	D
1	0	1	1	1	
1	1	0	0	1	D'
1	1	0	1	0	
1	1	1	0	1	D'
1	1	1	1	0	



Example: Realize F = B'CD' + ABC' with a 4:1 multiplexer and a minimum of other gates:

A	B	C	D	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0





DAWAH

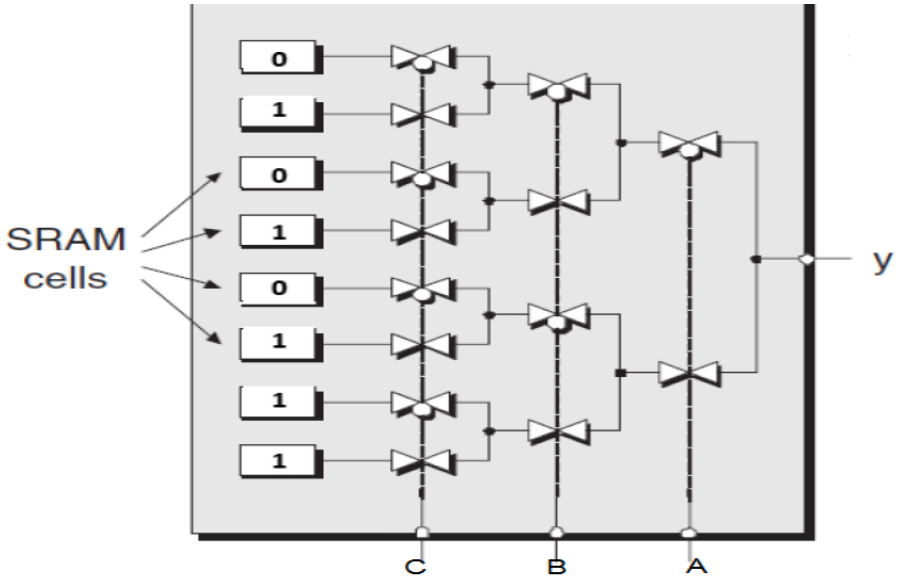
SOLUTION: Z = B'C'(0) + B'C(D') + BC'(A) + BC(0)

MULTIPLEXER

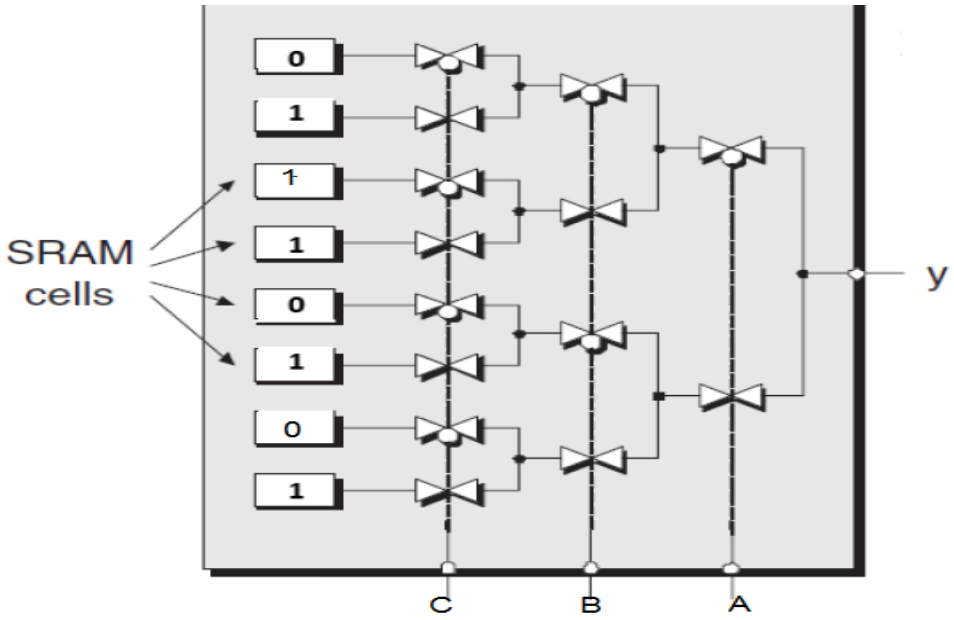
EXAMPLE: Implement the function using LUT- based Logic Block

SOLUTION

1-Y=AB+C

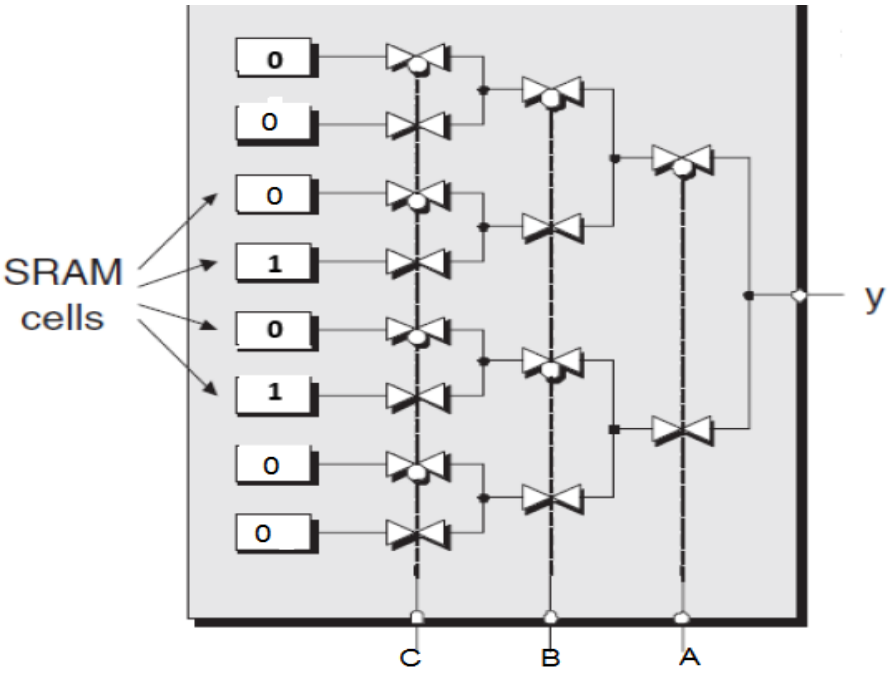


2- Y=$\overline{A}$B+C



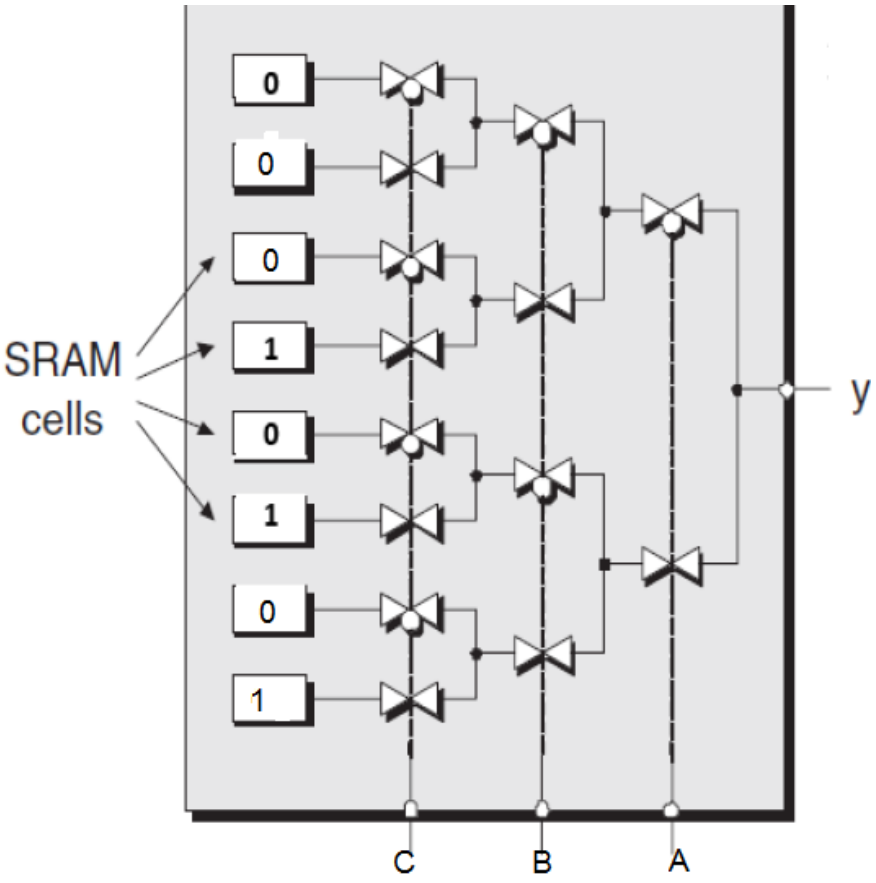
MULTIPLEXER

3-Y=(A⊕B)C



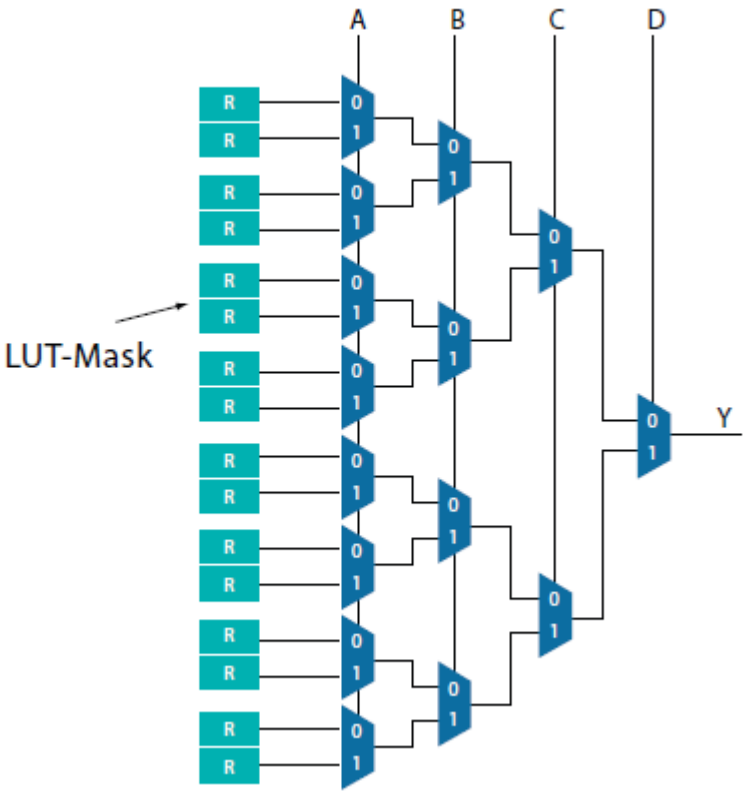
4- Y=(A+B)C

MULTIPLEXER

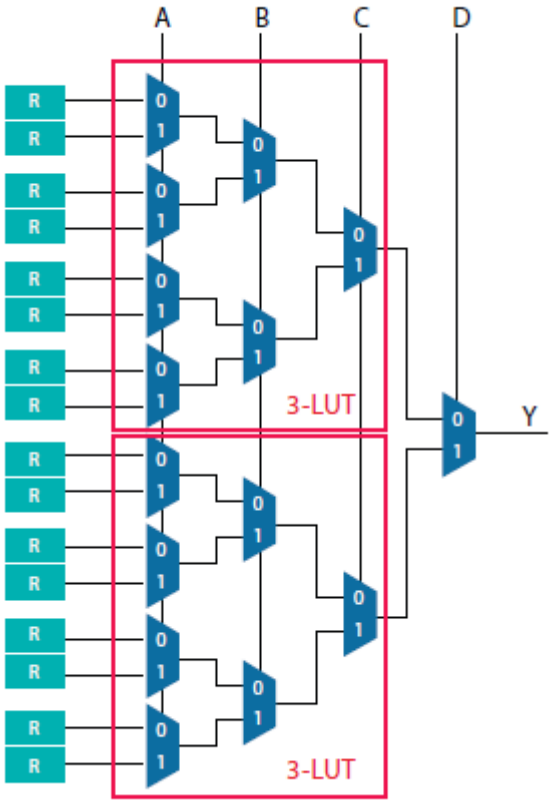


MULTIPLEXER

(Building Look-up Tables (LUTs
An overview of how LUTs are built helps describe the key innovations in the ALM. A LUT is typically built out of
SRAM bits to hold the configuration memory (CRAM) LUT-mask and a set of multiplexers to select the bit of
CRAM that is to drive the output. To implement a k-input LUT (k-LUT)—a LUT that can implement any function of
k inputs—2k SRAM bits and a 2k:1 multiplexer are needed. Figure 2 shows a 4-LUT, which consists of 16 bits of
SRAM and a 16:1 multiplexer implemented as a tree of 2:1 multiplexers. The 4-LUT can implement any function of
inputs (A, B, C, D) by setting the appropriate value in the LUT-mask. To simplify the 4-LUT ٤
in Figure 2, it can also
.be built from two 3-LUTs connected by a 2:1 multiplexer
Figure 2. Building a LUT



MULTIPLEXER

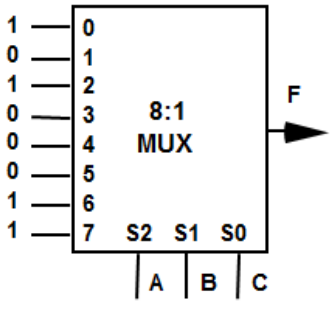


Example:

F(A,B,C) = m0 + m2 + m6 + m7

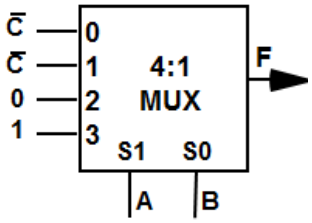
= A' B' C' + A' B C' + A B C' + A B C

= A' B' (C') + A' B (C') + A B' (0) + A B (1)



"Lookup Table"

A	B	C	F	
0	0	0	1	$\bar{C}$
0	0	1	0	
0	1	0	1	$\bar{C}$
0	1	1	0	
1	0	0	0	0
1	0	1	0	
1	1	0	1	1
1	1	1	1	



Department of Computer Engineering Techniques, College of Engineering, University of Al Maarif, Al Anbar,31001, Iraq

MULTIPLEXER

The FPGA As the name implies, a Field-Programmable Gate Array (FPGA) is an array of gates interconnected in a row-column matrix that can be programmed in the field by a computer via a USB connection. The FPGA differs from the CPLD in that, instead of solving the logic design by interconnecting logic gates, it uses a look-up table (LUT) method to resolve the particular logic requirement. This allows PLD manufacturers to form a more streamlined design, creating a much denser and faster PLD. Besides having thousands of internal logic elements, FPGAs have hundreds of I/O pins with programmable internal interconnects and storage registers. The Altera Cyclone© series is an example of an FPGA family. To see how a look-up table works, refer to Figures 4–11(a) and (b). In Figure 4–11(a), the conventional logic for the equation is implemented using 7400-series ICs. In this case, X is HIGH for three different combinations of the four inputs (X is HIGH when or 1010 or 0000). Figure 4–11(b) shows the same logic implemented in an FPGA LUT. An LUT operates similar to a truth table in that it provides for all possible input combinations and produces a HIGH when the desired combinations of 1s and 0s are provided at the inputs. In Figure 4–11(b), the routing of the logic levels is controlled by the 15 cascaded data selectors (trapezoid symbols). They

MULTIPLEXER

are actually multiplexers, which are covered in Chapter 8, but for now all we need to understand is that when the control input A, B, C, or D is HIGH, the logic level on the TRUE input is passed through from left to right. When it is LOW, the logic level on the complement input is passed through. The external A control input actually controls eight data selectors: B controls four, C controls two, and D controls one. This illustration of a LUT shows the flow of logic when the inputs are set at and In this case, since then all logic levels connected to the eight TRUE As are passed through. Therefore, by just looking at the high-lighted data path, a 1 is passed through to the B data selector. Now, since the B data selector control input is 0, then the data passes through the to the C data selector, and so on. The end result of this path is that a 1 passes through to X when To confirm that you understand this logic, follow the logic for and then for to see that these conditions are also met. As you can see, the result at X is dependent on the logic levels programmed into the SRAM (static random-access memory) memory cells (covered in Chapter 16). These memory cells are volatile and will need to be reinitialized along with the internal interconnections and registers each time the FPGA is powered on. Although CPLDs have the advantage of being non-volatile, FPGAs are much denser and faster so are used more often in middle to high-end applications. The FPGA that is on the Altera DE-2 Development board shown in Figure 4–5 is the Cyclone EP2C35F672C6N. It contains 33,216 look-up tables and has 475 pins dedicated for input/output to external circuitry.

MULTIPLEXER

According to the ordering Information at the Altera Cyclone Web site, the 672 in the part number indicates the number of pins

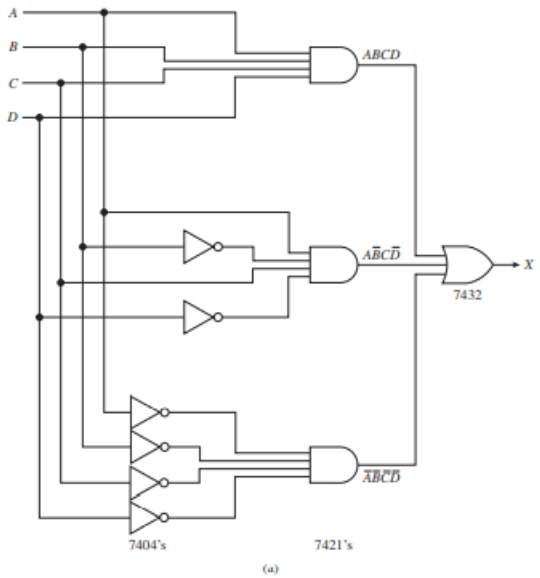
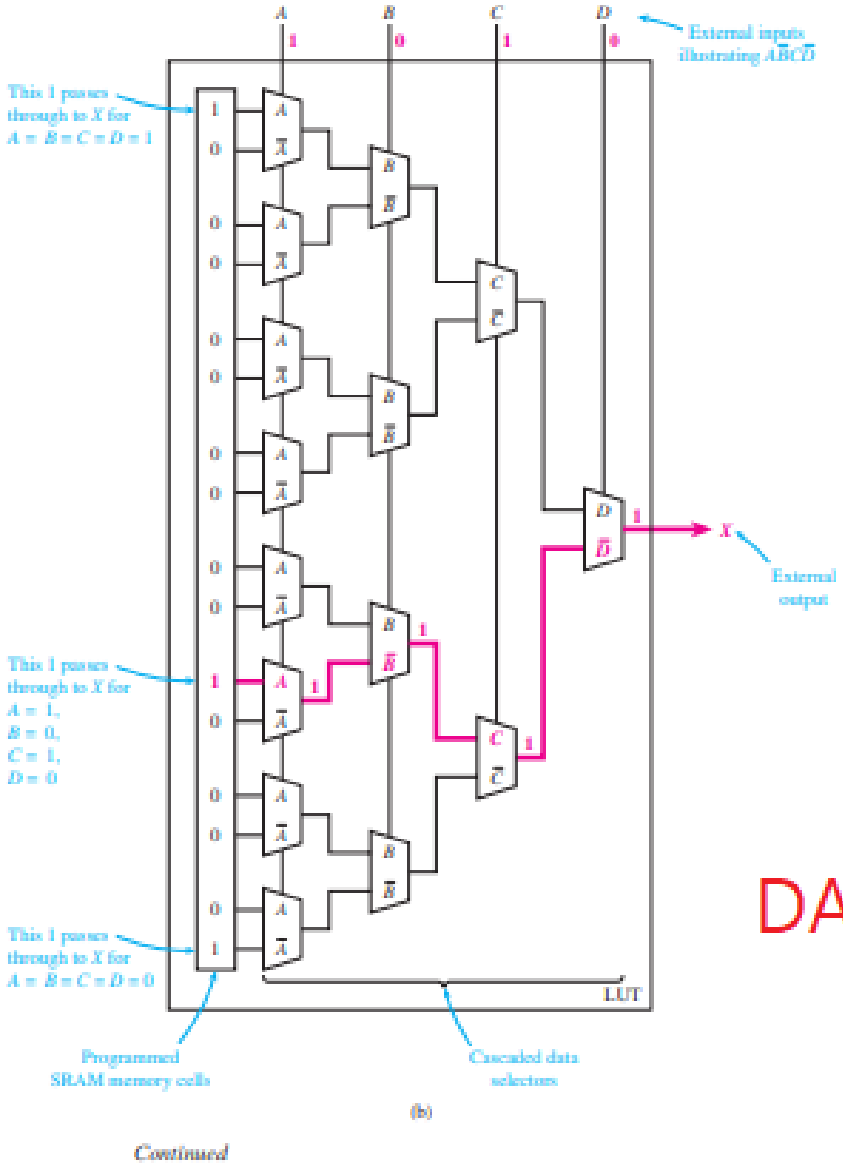


Figure $X = ABCD + A\overline{B}C\overline{D} + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$: (a) implemented using 7400-series ICs; (b) implemented within a LUT of an FPGA (showing the flow for $A\overline{B}C\overline{D}$).

MULTIPLEXER



Continued

and the letter F denotes that it is a Fine Line Ball Grid Array (BGA). In order to provide for 672 pins, the BGA pins are on the bottom of the IC setup as 26 rows by 26 columns. (The four outside corner pins are left off.)