CHAPTER -1 LOGIC FAMILIES

1.) Logic families classification

a.) TTL…Transistor-transistor logic (TTL) is a digital logic design in which bipolar transistor s act on direct-current pulses. Many TTL logic gate s are typically fabricated onto a singleintegrated circuit (IC). TTL ICs usually have four-digit numbers beginning with 74 or 54.

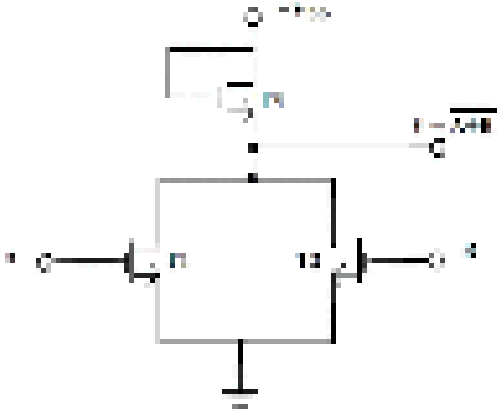
A TTL device employs transistor s with multiple emitters in gates having more than one input. TTL is characterized by high switching speed (in some cases upwards of 125 MHz), and relative immunity to noise . Its principle drawback is the fact that circuits using TTL draw more current than equivalent circuits using metal oxide semiconductor (MOS) logic. Low-current TTL devices are available, but the reduced current demand comes at the expense of some operating speed.

b.) ECL…In electronics, **emitter-coupled logic** (**ECL**) is a high-speed integrated circuit bipolar transistor **logic** family.**ECL** uses an overdriven BJT differential amplifier with single-ended input and limited **emitter** current to avoid the saturated (fully on) region of operation and its slow turn-off behavior

c.) CMOS…The term CMOS stands for "Complementary Metal Oxide Semiconductor". CMOS technology is one of the most popular technology in the computer chip design industry and broadly used today to form integrated circuits in numerous and varied applications. Today's computer memories, CPUs and cell phones make use of this technology due to several key advantages. This technology makes use of both P channel and N channel semiconductor devices.

The main advantage of CMOS over NMOS and BIPOLAR technology is the much smaller power dissipation. Unlike NMOS or BIPOLAR circuits, a Complementary MOS circuit has almost no static power dissipation. Power is only dissipated in case the circuit actually switches. This allows integrating more CMOS gates on an IC than in NMOS or bipolar technology, resulting in much better performance. Complementary Metal Oxide Semiconductor transistor consists P-channel MOS (PMOS) and N-channel MOS (NMOS)

d.) MOS…***MOS logic family implements the logic gates using MOSFET devices. MOSFETs are high density devices which can easily and economically fabricated on ICs. MOS logic gates can be fabricated using either only NMOS or only PMOS devices.***



Types of integration

a.) SSI (Small scale integration)…The first integrated circuits contained only a few transistors and so were called "Small-Scale Integration **(SSI)**. They used circuits containing transistors numbering in the tens. They were very crucial in development of early computers. SSI-small scale integration contains less than 100 components (about 10 gates)

b.) MSI (Medium scale integration)…. SSI was followed by introduction of the devices which contained hundreds of transistors on each chip, and so were called "Medium-Scale Integration (**MSI**).MSI were attractive economically because which they cost little more systems to be produced using smaller circuit boards, less assembly work, and a number of other advantages.

c.) LSI (large scale integration)… Next development was of Large Scale Integration (LSI). The development of LSI was driven by economic factors and each chip comprised tens of thousands of transistors. It was in 1970s, when LSI started getting manufactured in huge quantities.MSI contains less than 500 components (more than 10 but less than 100 gates)

d.) VLSI (very large scale integration)…Microprocessor chips produced in 1994 contained more than three million transistors. ULSI refer to "Ultra-Large Scale Integration" and correspond to more than 1 million of transistors. However there is no qualitative leap between VLSI and ULSI, hence normally in technical texts the "VLSI" term cover ULSI.

2.)

- Characteristics of TTl…Power dissipation is usually 10 mW per gate.
- Propagation delays are 10 nS when driving a 15 pF/400 ohm load.
- Voltage levels range from 0 to Vcc where Vcc is typically 4.75V - 5.25V. Voltage range 0V - 0.8V creates logic level 0. Voltage range 2V - Vcc creates logic level 1.
- Characteristics of CMOS… Dissipates low power: The power dissipation is dependent on the power supply voltage, frequency, output load, and input rise time. At 1 MHz and 50 pF load, the power dissipation is typically 10 nW per gate.
- Short propagation delays: Depending on the power supply, the propagation delays are usually around 25 nS to 50 nS.
- Rise and fall times are controlled: The rise and falls are usually ramps instead of step functions, and they are 20 - 40% longer than the propagation delays.
- Noise immunity approaches 50% or 45% of the full logic swing.
- Levels of the logic signal will be essentially equal to the power supplied since the input impedance is so high.
- Voltage levels range from 0 to VDD where VDD is the supply voltage. A low level is anywhere between 0 and 1/3 VDD while a high level is between 2/3 VDD and VDD.

Comparison of CMOS ans TTL

- CMOS circuits do not draw as much power as TTL circuits while at rest. However, CMOS power consumption increases faster with higher clock speeds than TTL does. Lower current draw requires less power supply distribution, therefore causing a simpler and cheaper design.
- Due to longer rise and fall times, the transmission of digital signals becomes simpler and less expensive with CMOS chips.
- CMOS components are more susceptible to damage from electrostatic discharge than TTL components.

Propagation delay…

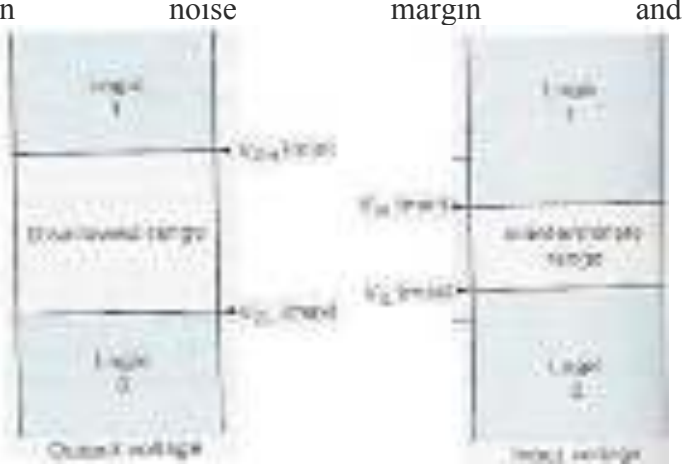
1) Propagation delay, symbolized t_{pd}, is the time required for a digital signal to travel from the input(s) of a logic gate to the output. It is measured in microseconds (µs),nanoseconds (ns), or picoseconds (ps), where 1 µs = 10^{-6} s, 1 ns = 10^{-9} s, and 1 ps = 10^{-12}s.

2) In electronics, digital circuits and digital electronics, the propagation delay, or **gate delay**, is the length of time which starts when the input to a logic gate becomes stable and valid to change, to the time that the output of that logic gate is stable and valid to change. Often on manufacturers' datasheets this refers to the time required for the output to reach 50% of its final output level when the input changes to 50% of its final input level. Reducing gate delays in digital circuits allows them to process data at a faster rate and improve overall performance. The determination of the propagation delay of a combined circuit requires

identifying the longest path of propagation delays from input to output and by adding each tpd time along this path.

Noise margin…(1)Noise margin is the amount of noise that a CMOS circuit could withstand without compromising the operation of circuit. Noise margin does makes sure that any signal which is logic '1' with finite noise added to it, is still recognised as logic '1' and not logic '0'. It is basically the difference between signal value and the nosie value. Refer to the diagram below.

(2) Definition: Ability of the gate to tolerate fluctuations of the voltage levels.The input and output voltage levels defined above point. Stray electric and magnetic fields may induce unwanted voltages, known as noise, on the connecting wires between logic circuits. This may cause the voltage at the input to a logic circuit to drop below VIH or rise above VIL and may produce undesired operation. The circuit's ability to tolerate noise signals is referred to as the noise immunity, a quantitative measure of which is called noise margin.
The noise margins defined above are referred to as dc noise margins. Strictly speaking, thenoise is generally thought of as an a.c. signal with amplitude and pulse width. For high speed ICs, a pulse width of a few microseconds is extremely long in comparison to the propagation delay time of the circuit and therefore, treated as d.c. as far as the response of the logic circuit is concerned. As the noise pulse width decreases and approaches the propagation delay time of the circuit, the pulse duration is too short for the circuit to respond. Under this condition, a large pulse amplitude would be required to produce a change in the circuit output. This means that a logic circuit can effectively tolerate a large noise amplitude if the noise is of a very short duration. This is referred to as ac noise margin and is substantially greater than the dc noise margin. It is generally supplied by the manufacturers in the form of a curve between noise margin and noise pulse width.



VNH	=	HIGH-state	noise	margin
VNL	=	LOW-state	noise	margin
VIL	=	LOW-state	input	voltage
VIH	=	HIGH-state	input	voltage
VOL	=	LOW-state	output	voltage
VOH	=	HIGh-state	output	voltage

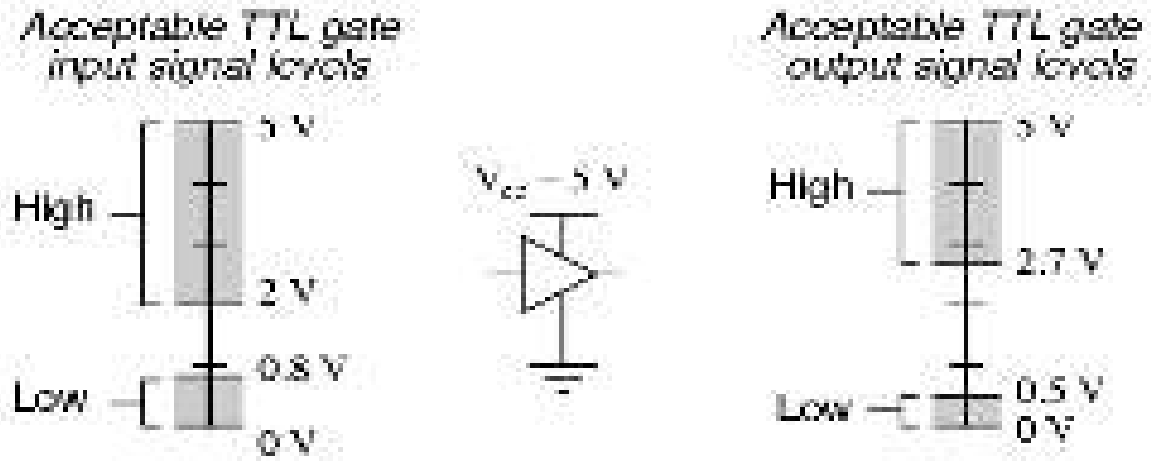
Where,

VNH = VOH -VIH

VNL = VIL – VOL

Manufacturers specify voltage limits to represent the logical 0 or 1. These limits are not the same at the input and output sides. For example, a particular gate A may output a voltage of 4.8V when it is supposed to output a HIGH but, at its input side, it can take a voltage of 3V as HIGH. In this way, if any noise should corrupt the signal, there is some margin for error.

Logic levels…In digital circuits, a **logic level** is one of a finite number of states that a digital signal can inhabit. **Logic levels**are usually represented by the voltage difference between the signal and ground, although other standards exist.



Power dissipation…**Power dissipation** is usually 10 mW per gate. Propagation delays are 10 nS when driving a 15 pF/400 ohm load. Voltage levels range from 0 to Vcc where Vcc is typically 4.75V - 5.25V. Voltage range 0V - 0.8V creates logic level 0

ECL **logic families** requires large currents therefore **power dissipation** is 3 to 10 times higher than that of TTL **logic families**. Because of its large **power** consumption and high requirement of silicon area, CMOS **logic** gates are preferred over ECL **logic families** in large scale integrated circuits

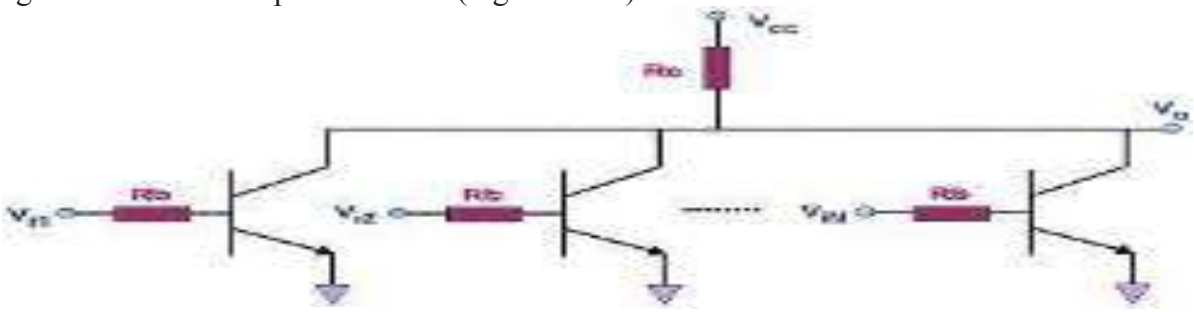
Fan in and Fan out…

Fan In: The fan-in defined as the maximum number of inputs that a logic gate can accept. If number of input exceeds, the output will be undefined or incorrect. It is specified by manufacturer and is provided in the data sheet.

Fan Out: The fan-out is defined as the maximum number of inputs (load) that can be connected to the output of a gate without degrading the normal operation. Fan Out is calculated from the amount of current available in the output of a gate and the amount of current needed in each input of the connecting gate. It is specified by manufacturer and is provided in the data sheet. Exceeding the specified maximum load may cause a malfunction because the circuit will not be able supply the demanded power.

The difference between these two characteristics of a digital IC is significant from the definitions above.

RTL(resistor transistor logic)… A bipolar transistor switch is the simplest RTL gate (inverter or NOT gate) implementing logical negation.[2] It consists of a common-emitter stage with a base resistor connected between the base and the input voltage source. The role of the base resistor is to expand the very small transistor input voltage range (about 0.7 V) to the logical "1" level (about 3.5 V) by converting the input voltage into current. Its resistance is settled by a compromise: it is chosen low enough to saturate the transistor and high enough to obtain high input resistance. The role of the collector resistor is to convert the collector current into voltage; its resistance is chosen high enough to saturate the transistor and low enough to obtain low output resistance (high fan-out).



Direct couples transistor logic…**Direct-coupled transistor logic** (**DCTL**) is similar to resistor–transistor logic (RTL) but the input transistor bases are connected directly to the collector outputs without any baseresistors. Consequently, DCTL gates have fewer components, are more economical, and are simpler to fabricate onto integrated circuits than RTL gates. Unfortunately, DCTL has much smaller signal levels, has more susceptibility to ground noise, and requires matched transistor characteristics. The transistors are also heavily overdriven; that is a good feature in that it reduces the saturation voltage of the output transistors, but it also slows the circuit down due to a high stored charge in the base.[1] Gate fan-out is limited due to "current hogging": if the transistor base-emitter voltages (V_{BE}) are not well matched, then the base-emitter junction of one transistor may conduct most of the input drive current at such a low base-emitter voltage that other input transistors fail to turn on.[2]

DCTL is close to the simplest possible digital logic family, using close to fewest possible components per logical element.[3]

A similar logic family, direct-coupled transistor-transistor logic, is faster than ECL.[4]

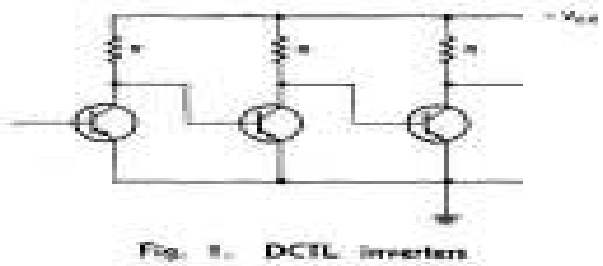
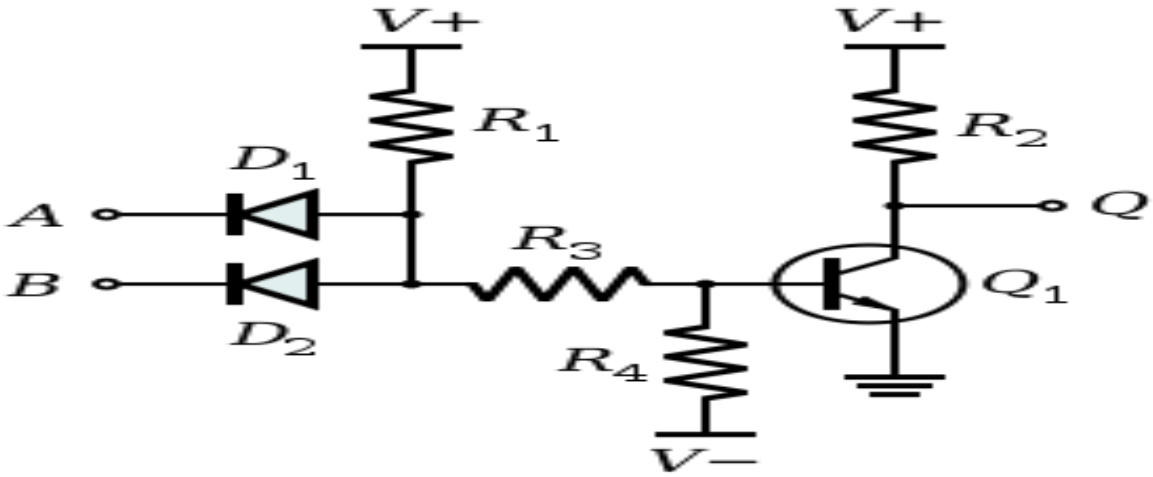
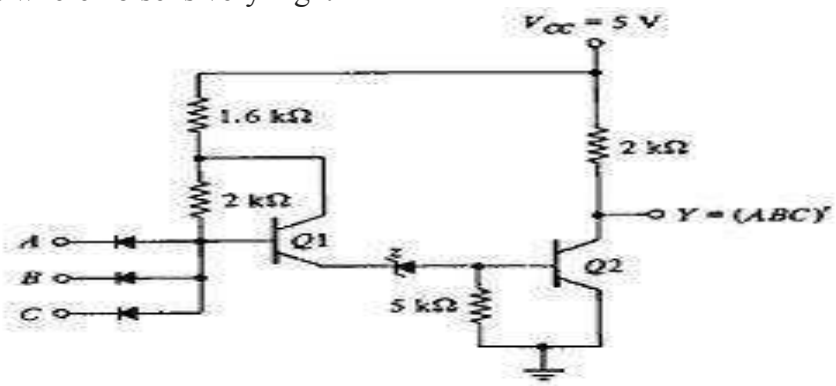


Fig. 1. DCTL inverters

Diode transistor logic…**Diode–transistor logic** (**DTL**) is a class of digital circuits that is the direct ancestor of transistor–transistor logic. It is called so because the logic gating function (e.g., AND) is performed by a diode network and the amplifying function is performed by a transistor (in contrast with RTL and TTL).



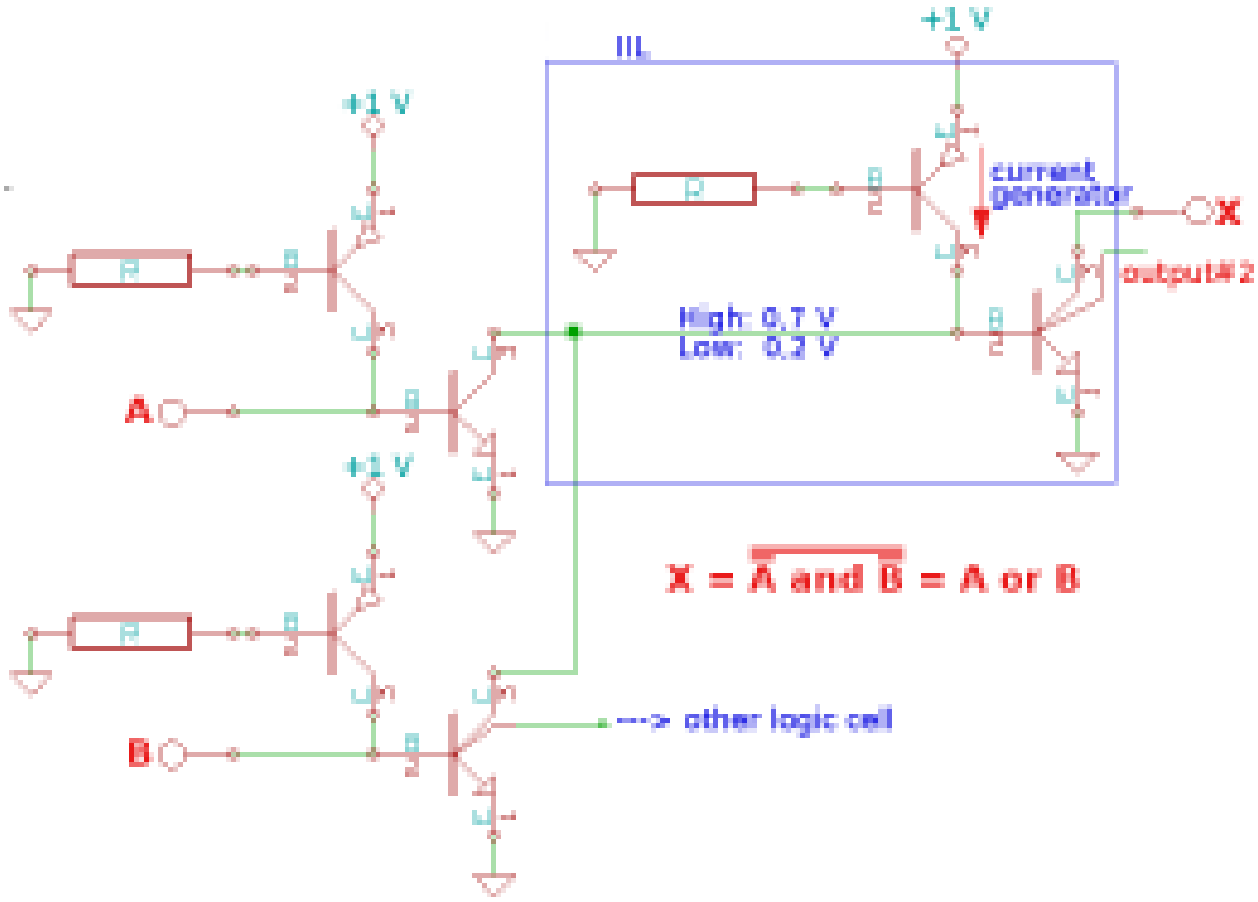
The DTL circuit shown in the picture consists of three stages: an input diode logic stage (D1, D2 and R1), an intermediate level shifting stage (R3 and R4), and an output common-emitter amplifier stage (Q1 and R2). If both inputs A and B are high (logic 1; near V+), then the diodes D1 and D2 are reverse biased. Resistors R1 and R3 will then supply enough current to turn on Q1 (drive Q1 into saturation) and also supply the current needed by R4. There will be a small positive voltage on the base of Q1 (V_{BE}, about 0.3 V for germanium and 0.6 V for silicon). The turned on transistor's collector current will then pull the output Q low (logic 0; $V_{CE(sat)}$, usually less than 1 volt). If either or both inputs are low, then at least one of the input diodes conducts and pulls the voltage at the anodes to a value less than about 2 volts. R3 and R4 then act as a voltage divider that makes Q1's base voltage negative and consequently turns off Q1. Q1's collector current will be essentially zero, so R2 will pull the output voltage Q high (logic 1; near V+).
High threshold logic… (**HTL**) is a variant of Diode–transistor logic which is used in such environments where noise is very high.



Working…The threshold values at the input to a logic gate determine whether a particular input is interpreted as a logic 0 or a logic 1.(e.g. anything less than 1 V is a logic 0 and anything above 3 V is a logic 1. In this example, the threshold values are 1V and 3V). HTL incorporates Zener diodes to create a large offset between logic 1 and logic 0 voltage levels. These devices usually ran off a 15 V power supply and were found in industrial control, where the high differential was intended to minimize the effect of noise.

Integrated injection logic… (**IIL**, **I^2L**, or **I2L**) is a class of digital circuits built with multiple collector bipolar junction transistors (BJT).[1] When introduced it had speed comparable to TTL yet was almost as low power as CMOS, making it ideal for use in VLSI (and larger) integrated circuits. Although the logic voltage levels are very close

(High: 0.7V, Low: 0.2V), I2L has high noise immunity because it operates by current instead of voltage. It is sometimes also known as **merged transistor logic**.



Working…The heart of an I2L circuit is the common emitter open collector inverter. Typically, an inverter consists of an NPN transistor with the emitter connected to ground and the base biased with a forward current. The input is supplied to the base as either a current sink (low logic level) or as a high-z floating condition (high logic level). The output of an inverter is at the collector. Likewise, it is either a current sink (low logic level) or a high-z floating condition (high logic level).

Like direct-coupled transistor logic, there is no resistor between the output (collector) of one NPN transistor and the input (base) of the following transistor.

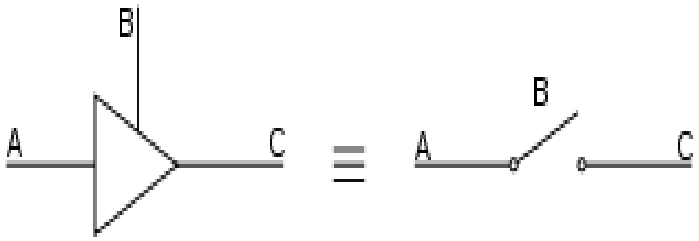
To understand how the inverter operates, it is necessary to understand the current flow. If the bias current is shunted to ground (low logic level), the transistor turns off and the collector floats (high logic level). If the bias current is not shunted to ground because the input is high-z (high logic level), the bias current flows through the transistor to the emitter, switching on the transistor, and allowing the collector to sink current (low logic level). Because the output of the inverter can sink current but cannot source current, it is safe to connect the outputs of multiple inverters together to form a wired AND gate. When the outputs of two inverters are wired together, the result is a two-input NOR gate because the configuration (NOT A) AND (NOT B) is equivalent to NOT (A OR B) (per De Morgan's Theorem).

PMOS logic family…**P-type metal-oxide-semiconductor logic** uses p-channel metal-oxide-semiconductor field effect transistors (MOSFETs) to implementlogic gates and other digital circuits. PMOS transistors operate by creating an inversion layer in an n-type transistor body. This inversion layer, called the p-channel, can conduct holes between p-type "source" and "drain" terminals.

The p-channel is created by applying voltage to the third terminal, called the gate. Like other MOSFETs, PMOS transistors have four modes of operation: cut-off (or subthreshold), triode, saturation (sometimes called active), and velocity saturation.
While PMOS logic is easy to design and manufacture (a MOSFET can be made to operate as a resistor, so the whole circuit can be made with PMOS FETs), it has several shortcomings as well. The worst problem is that there is a direct current (DC) through a PMOS logic gate when the PUN is active, that is, whenever the output is high, which leads to static power dissipation even when the circuit sits idle.
Also, PMOS circuits are slow to transition from high to low. When transitioning from low to high, the transistors provide low resistance, and the capacitive charge at the output accumulates very quickly (similar to charging a capacitor through a very low resistance). But the resistance between the output and the negative supply rail is much greater, so the high-to-low transition takes longer (similar to discharge of a capacitor through a high resistance). Using a resistor of lower value will speed up the process but also increases static power dissipation.
Additionally, the asymmetric input logic levels make PMOS circuits susceptible to noise.[1]
Most P-MOS integrated circuits require a power supply of 17-24 volt DC.[2] The Intel 4004 PMOS microprocessor, however, uses PMOS logic with polysilicon rather than metal gates allowing a smaller voltage differential. For compatibility with TTL signals, the 4004 uses positive supply voltage V_{SS}=+5V and negative supply voltage V_{DD} = -10V.[3]
Though initially easier to manufacture,[4] PMOS logic was later supplanted by NMOS logic using n-channel field-effect transistors. NMOS is faster than PMOS. Modern integrated circuits are CMOS logic, which uses both p-channel and n-channel transistors.

NMOS logic family…The NMOS logic family uses N-channel MOSFETS. N-channel MOS devices require a smaller chip area per transistor compared with P-channel devices, with the result that NMOS logic offers a higher density. Also, owing to the greater mobility of the charge carriers in N-channel devices, the NMOS logic family offers higher speed too. It is for this reason that most of the MOS memory devices and microprocessors employ NMOS logic or some variation of it such as VMOS, DMOS and HMOS. VMOS, DMOS and HMOS are only structural variations of NMOS, aimed at further reducing the propagation delay. Figures (a), (b) and (c) respectively show an inverter, a two-input NOR and a two-input NAND using NMOS logic. The logic circuits are self-explanatory.

TRI state logic…In digital electronics **three-state**, **tri-state**, or **3-state** logic allows an output port to assume a high impedance state, effectively removing the output from the circuit, in addition to the 0 and 1 logic levels.
This allows multiple circuits to share the same output line or lines (such as a bus which cannot listen to more than one device at a time).
Three-state outputs are implemented in many registers, bus drivers, and flip-flops in the 7400 and 4000 series as well as in other types, but also internally in many integrated circuits. Other typical uses are internal and external buses in microprocessors, computer memory, and peripherals. Many devices are controlled by an active-low input called OE(Output Enable) which dictates whether the outputs should be held in a high-impedance state or drive their respective loads (to either 0- or 1-level).
The term *tri-state*[1] should not be confused with ternary logic



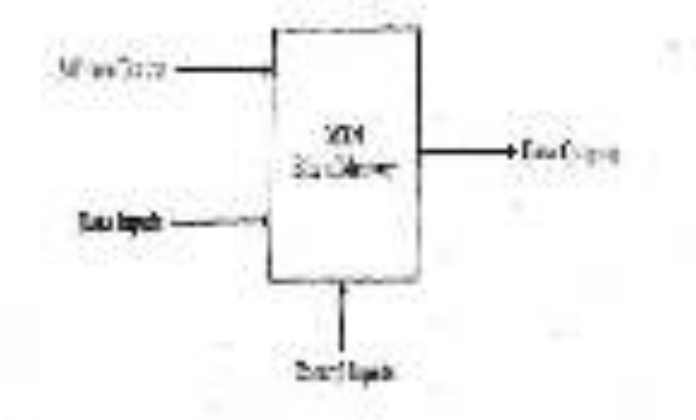
INPUT		OUTPUT
A	B	C
0	1	0
1		1
X	0	Z (high impedance

Tri state buffer…A Tri-state Buffer is another type of buffer circuit which can be used to control the passage of a logic signal from its input to its output. The tri-state buffer is a combinational device whose output can be electronically turned "ON" or "OFF" by means of an external "Control" or "Enable" (EN) signal input allowing them to be used in bus-orientated systems.
As their name implies, the output at "Q" for a **Tri-state Buffer** can take on one of three possible states, logic "0", logic "1", and High-Z (high impedance), that is, an open circuit, rather than the standard "0" and "1" states.
The buffers enable or control signal can be either a logic "0" or a logic "1" level signal with the output being inverting and non-inverting as the digital signal passes through it. The two most commonly used tri-state buffer IC's being the TTL 74LS125 and the TTL 74LS126.

UNI directional and Bi directional Bus…**Address** bus is Unidirectional because the microprocessor is addressing a specific memory location. No outside devices can not write into Microprocessor. Data bus is**Bidirectional** because the Microprocessor can read data from memory or write data to the memory.

Surface mount technology…**Surface-mount technology** (**SMT**) is a method for producing electronic circuits in which the components are mounted or placed directly onto the surface of printed circuit boards (PCBs). An electronic device so made is called a **surface-mount device** (**SMD**). In industry it has largely replaced the through-hole technology construction method of fitting components with wire leads into holes in the circuit board. Both technologies can be used on the same board, with the through-hole technology used for components not suitable for surface mounting such as large transformers and heat-sinked power semiconductors.
By employing SMT, the production process speeds up, but the risk of defects also increase due to component miniaturization and to the denser packing of boards. In those conditions, detection of failures has become critical for any SMT manufacturing process.

An SMT component is usually smaller than its through-hole counterpart because it has either smaller leads or no leads at all. It may have short pins or leads of various styles, flat contacts, a matrix of solder balls (BGAs), or terminations on the body of the component.

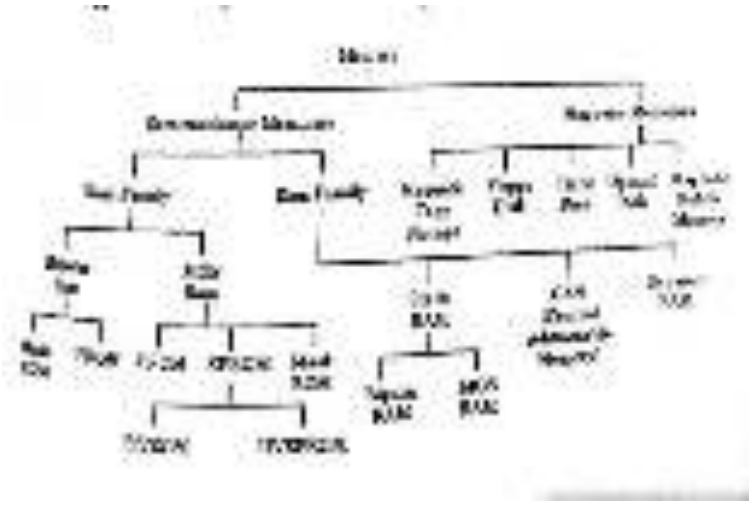


CHAPTER 3 MEMORIES

Memories

Memory is the faculty of the mind by which information is encoded, stored, and retrieved. Memory is vital to experiences and related to limbic systems, it is the retention of information over time for the purpose of influencing future action.Classification of Memory--

Memory organization-- The basic memory element in semi conductor memories are flip flop. The information in the flip flop is stored in the binary form(either 0or1).A memory chip have large number of locations and each location is used for storage of one word of digital information. The number of locations and number of bits comprising the word vary from memory to memory.The memory chip is identified by two number M and N bits.Here the number M tells us about the number of locations available in the memory and N-tells us about the number of bits at each location. The most commonly used number of word per chip are 64,256,512,1024,2048 etc and the values for word size are 1,4,8,16etc.The block diagram of a memory is shown below -



Each of the number of locations of the memory are identified by the unique address. So, for accessing any location of the memory we require address inputs. Any memory location can be addressed by using the following relation

2*=M

Where X reperesents the address input.

The address inout is applied to a decoder circuit which activates one of its M outputs deoending upon the applied address input and the desired memory location is selected.In this way,we can say that the memory location selection depends upon the applied address input. The control input is used to tell the memory whether a read or a write operation is desired.

Classification of Semiconductor Memories

The semiconductor conductor memories can be classified according to their principle of operation,physical characteristics, mode of access and the technology used for fabrication.

Semiconductor memories on the basis of principle of operation----

The most commonly used semiconductor memories on the basis of principle of operation are listed below

1) Sequential memories
2)Random access memories
3)Read only memories
4)Content addressable memories

1) **Sequential memories-- In computing, sequential access memory (SAM) is a class of data storage devices that read stored data in a sequence. This is in contrast to random access memory (RAM) where data can be accessed in any order. Sequential access devices are usually a form of magnetic storage. While sequential access memory is read in sequence, accesses can still be made to arbitrary locations by "seeking" to the requested location. This operation, however, is often relatively inefficient (see seek time, rotational latency).**

Magnetic sequential access memory is typically used for secondary storage in general-purpose computers due to their higher density at lower cost compared to RAM, as well as resistance to wear and non-volatility. Examples of SAM devices still in use include hard disks, CD-ROMs and magnetic tapes. Historically, drum memory has also been used.

The semiconductor sequential memories are classified into two types--

1) Shift Registers
2) Charge coupled devices

2) Random access memories--RAM (pronounced ramm) is an acronym for random access memory, a type of computer memory that can be accessed randomly; that is, any byte of memory can be accessed without touching the preceding bytes. RAM is found in servers, PCs, tablets, smartphones and other devices, such as printers.It can be classified into two types-

1) Static RAM
2) Dynamic RAM

3) Read only memory-- Read-only memory is a type of non-volatile memory used in computers and other electronic devices. Data stored in ROM can only be modified slowly, with difficulty, or not at all, so it is mainly used to store information which is fixed. The read only memory is further classified into different types such as PROM,EPROM and E²ROM.

4) Content addressable memories--The content addressable memories perform the association operation in addition to read/write operations.

According to physical characteristics--

The semiconductor memories are classified into two types accordingto their physical characteristics.

1) Erasable or non-erasable. 2) Volatile or non volatile

In a erasable memory,we can erase the contents of the memory and new information can be stored in those contents of the memory. But in the non-erasable memory, we cannot erase the contents of the memory. The best example of a non-erasable memory is ROM.

In case of a volatile memory, when the electrical power is switched off then the information stored in the memory is lost. The example of a volatile memory is RAM. In case of a non volatile memory, the information stored remains in the same position even when the power is switched off. The example of a non volatile memory is ROM.

According to Mode of Access--

The semiconductor memories are classified into two types according to their mode pf access.

1) Sequential access

2) Random access

In case of sequential access memories, the access time is different for different location. But in case of random access memories, the access time is same for different locations.

According to fabrication technology

The semiconductor memories are classified ino two types according to fabrication technology.

1) Bipolar fabrication technology

2) Unipolar or Mos fabrication technology

Both these technologies are discussed in detail in the later section of this chapter. The static RAM, ROM can be fabricated by using bipolar and unipolar fabrication technology. But the dynamic RAM, EPROM can be fabricated only by using the MOS technologies.

ROM---

Read-Only Memory, ROM is a storage medium that is used with computers and other electronic devices. As the name indicates, data stored in ROM may only be read. It is either modified with extreme difficulty or not at all. ROM is mostly used for firmware updates. A simple example of ROM is the cartridge used with video game consoles, which allows one system to run multiple games. Another example of ROM is EEPROM, which is a programmable ROM used for the computer BIOS, as shown in the picture below.

Note: Unlike Random Access Memory (RAM), ROM is non-volatile, which means it keeps its contents regardless of whether or not it has power.

ROM can be classified into various type depending upon the programming process employed.

1) Mask programmable ROM

2) Programmable ROM (PROM)

3) Erasable programmable ROM (EPROM)

Block diagram of a ROM

A Read Only Memory (ROM) is a device that includes both the decoder and the OR gates within a single IC package. The Fig. 3.82 shows the block diagram of ROM. It consists of n input lines and m output lines. Each bit combination of the input variables is called an address. Each bit combination that comes out of the output lines is called a word. The number of bits per word is equal to the number of output lines, m. The address specified in binary number denotes one of the minterms of n variables. The number of distinct addresses possible with n input variables is 2n. An output word can be selected by a unique address, and since there are 2n distinct addresses in a ROM, there are 2n distinct words in the ROM. The word available on the output lines at any given time depends on the address value applied to the input lines.

Let us consider 64 x 4 ROM. The Read Only Memory consists of 64 words of 4 bits each. This means that there are four output lines and particular word from 64 words presently available on the output lines is determined from the six input lines. There are only six inputs in a 64 x 4 ROM because 26 = 64, and with six variables, we can specify 64 addresses or minterms. For each address input, there is a unique selected word. Thus, if the input address is 000000, word number 0 is selected and applied to the output lines. If the input address is 111111, word number 63 is selected and applied to the output lines.

PROM--

Programmable read-only memory (PROM) is read-only memory (ROM) that can be modified once by a user. PROM is a way of allowing a user to tailor a microcode program using a special machine called a PROM machine programmer . This current supplies an electrical ROM to specific cells in the fuse in that effectively blows a known as them. The process is Since this burning the PROM . for error, process leaves no margin designed most ROM chips use to be modified by users read- erasable programmable only memory (EPROM) or electrically erasable programmable read-only memory (EEPROM).

Fig. 3.82 Block diagram of ROM

EPROM--EPROM (erasable programmable read-only memory) is programmable read-only memory (programmable ROM) that can be erased and re-used. Erasure is caused by shining an intense ultraviolet light through a window that is designed into the memory chip. (Although ordinary room lighting does not contain enough ultraviolet light to cause erasure, bright sunlight can cause erasure. For this reason, the window is usually covered with a label when not installed in the computer.)

A different approach to a modifiable ROM is electrically erasable programmable read-only memory (EEPROM).

E²PROM or EEPROM--

It is abbreviated or electrically erasable programmable read only memory, we apply electrical voltage of proper polarity and amplitude. In case of EPROM, for erasing the contents of the memory, we use ultraviolet light but in case of E²PROM, we use electrical voltage of proper polarity and amplitude.

Application of ROM--

ROM (Read Only Memory): ROM is a device used for storage purpose. Once the data is stored within the memory, then it cannot be altered. ROM keeps the program needed to boot the computer initially.

The below table shows the ROM IC with their characteristics.

Random access memory (RAM) --

RAM (pronounced ramm) is an acronym for random access memory, a type of computer memory that can be accessed randomly; that is, any byte of memory can be accessed without touching the preceding bytes. RAM is found in servers, PCs, tablets, smartphones and other devices, such as printers.

Main Types of RAM

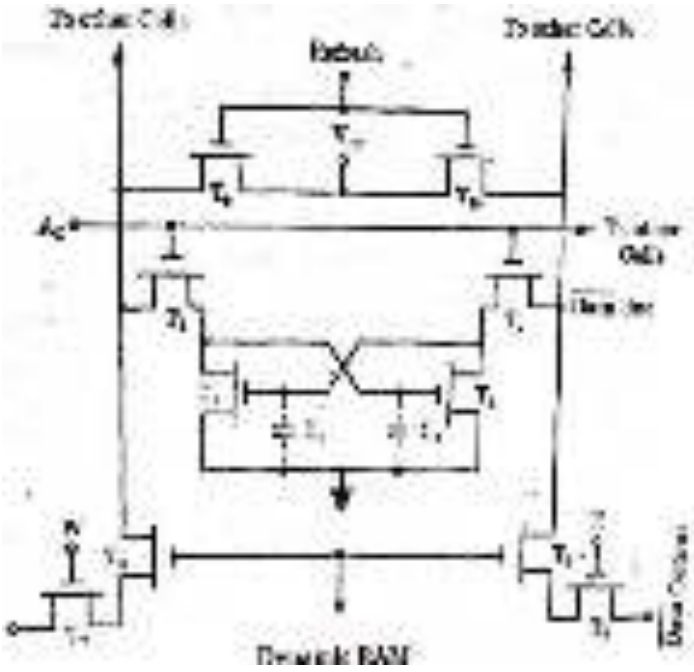
There are two main types of RAM:

1) DRAM (Dynamic Random Access Memory)

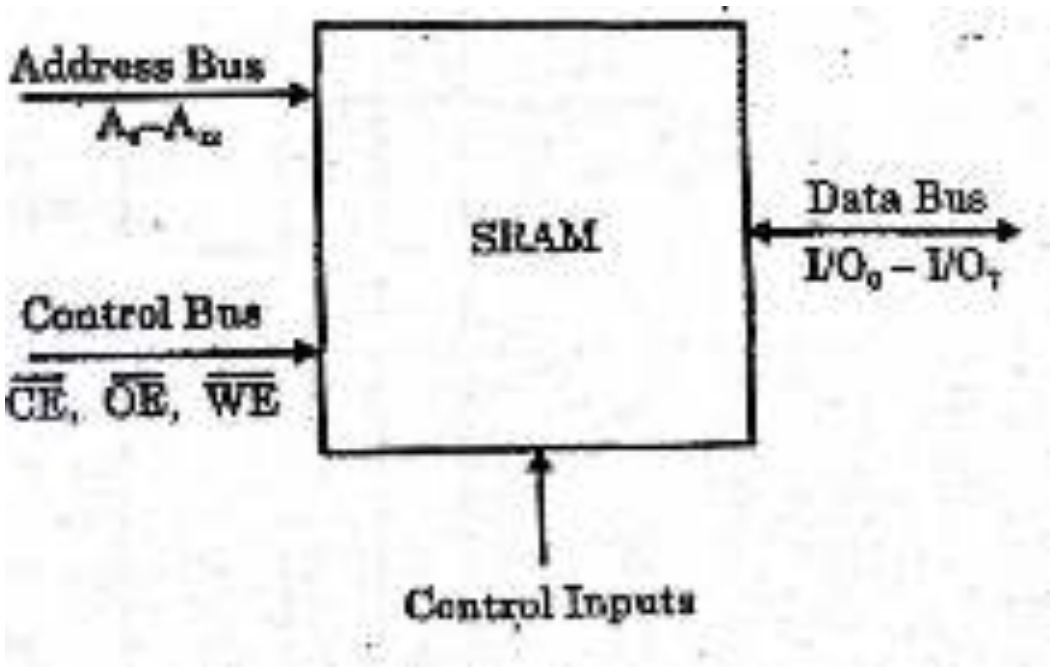
2) SRAM (Static Random Access Memory)

DRAM (Dynamic Random Access Memory) – The term dynamic indicates that the memory must be constantly refreshed or it will lose its contents. DRAM is typically used for the main memory in computing devices. If a PC or smartphone is advertised as having 4-GB RAM or 16-GB RAM, those numbers refer to the DRAM, or main memory, in the device.

More specifically, most of the DRAM used in modern systems is synchronous DRAM, or SDRAM. Manufacturers also sometimes use the



acronym DDR (or DDR2, DDR3, DDR4, etc.) to describe the type of SDRAM used by a PC or server. DDR stands for double data rate, and it refers to how much data the memory can transfer in one clock cycle.
In general, the more RAM a device has, the faster it will perform.
SRAM (Static Random Access Memory) – While DRAM is typically used for main memory, today SRAM is more often used for system cache. SRAM is said to be static because it doesn't need to be refreshed, unlike dynamic RAM, which needs to be refreshed thousands of times per second. As a result, SRAM is faster than DRAM. However, both types of RAM are volatile, meaning that they lose their contents when the power is turned off.



Advantages of DRAM over SRAM-- SRAMs are used for specific applications within the PC, where their strengths outweigh their weaknesses compared to DRAM:

• Simplicity: SRAMs don't require external refresh circuitry or other work in order for them to keep their data intact.
• Speed: SRAM is faster than DRAM. In contrast, SRAMs have the following weaknesses, compared
to DRAMs:
• Cost: SRAM is, byte for byte, several times more expensive than DRAM.
• Size: SRAMs take up much more space than DRAMs (which is part of why the cost is higher).

Expansion of Memory-- In many applications the micro computer system requirements for memory are greater than what is available in a single device. There are two basic reasons for expanding the memory capacity.
1) The byte wide length is not larger enough.
2) Total storage capacity is not enough bytes.
Both of these expansion needs can be satisfied by interconnecting a number of memory IC's. So,the expansion of memory consist of two types.
1) Expanding word length

2) Expanding word capacity

1) Expanding word length-- If a word length m is required for a memory and the available word length of the memory IC's is M where m>M, then a number of similar IC's can be connected together to get the desired word size. The following way isadopted when the similar IC's are connected--

1) The address lines of each chip are connected individually. This means the address line A1 of the overall memory. In the same manner, connect all the address lines.

2) Connect the read input of each IC together and it becomes the read inout for the overall memory. In the same way, connect the CS and WR inputs. The following example shows how we can expand the word length of a memory.

2) Expanding word capacity--

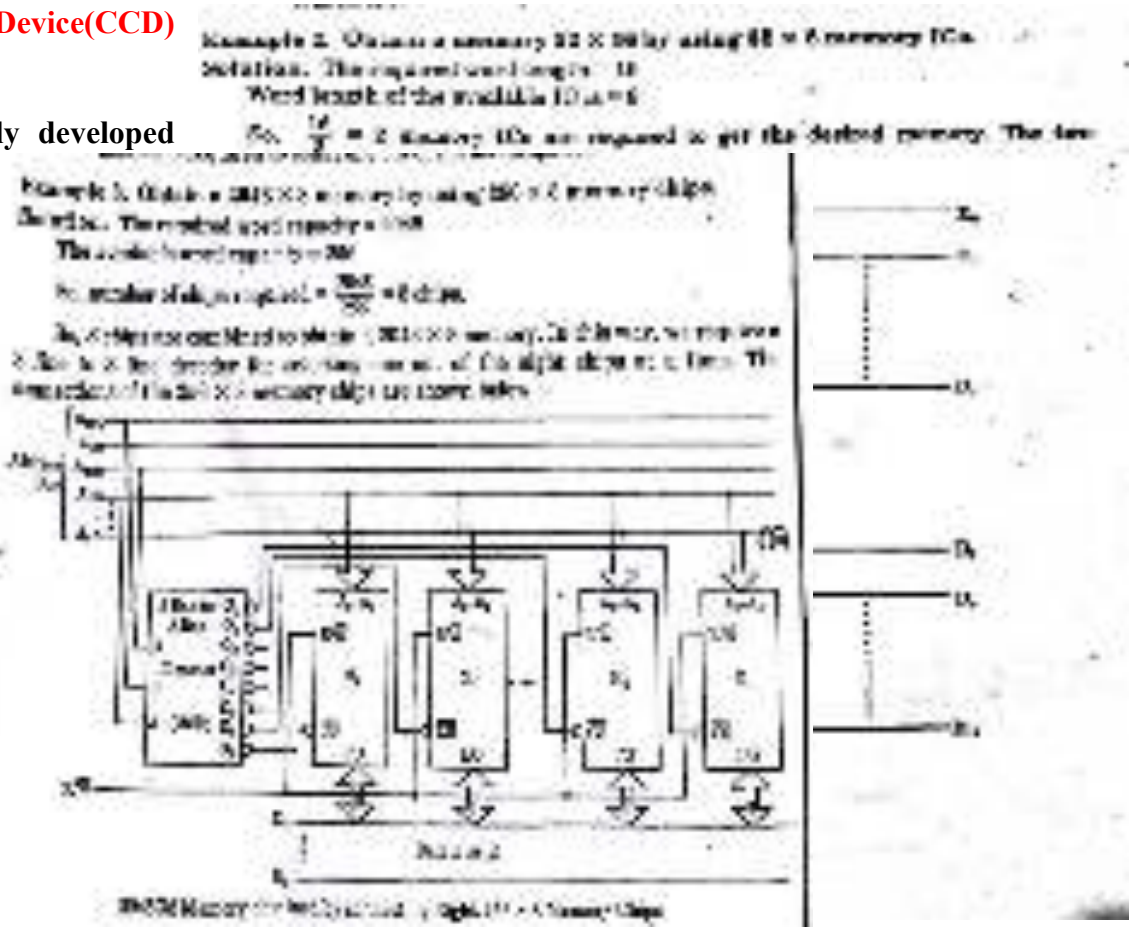
The memory IC chips can be combined together for exoanding the word capacity or memory. For obtaining a memory of word capacity n words, if the available memory chips has N words each. The number of memory IC chips require are given by n/N.The following way adopted when the similar IC's are connected -:

1) The address lines of each memory chip are connected individually. This means the address line A0 of each chip are connected together and it becomes address line A0 of the overall memory. In the same way, connect all the address lines.

2) Connect the RD and WR inputs of each chip together.

3) In this case, we also required a decoder of proper size and connect each of the decoder outputs to one of the CS terminals of the memory chips. For example if four memory IC chips are to be connected, then a 2 line to 4 line decoder is required to select one out of the four chips at a time

Charge Coupled Device(CCD) Memory--

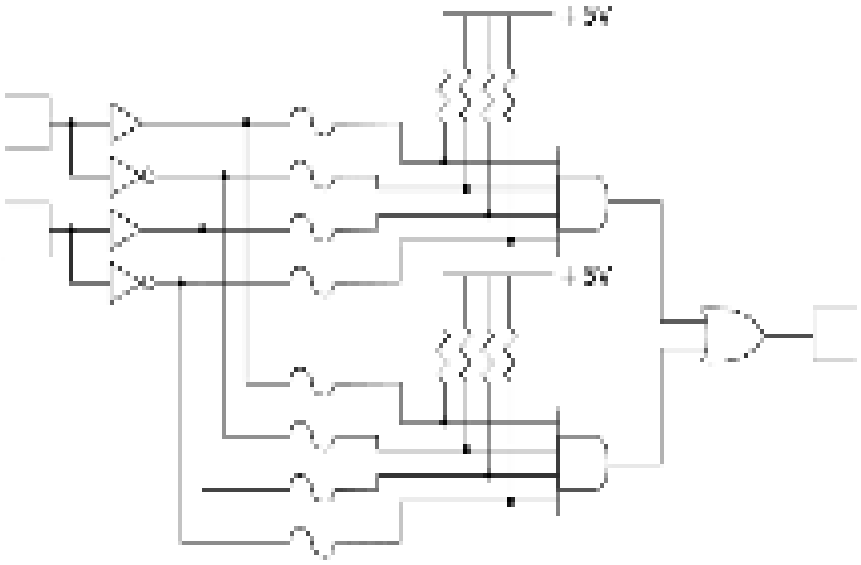


A specially developed CCD used for ultraviolet imaging in a wire-bonded package

A charge-coupled device (CCD) is a device for the movement of electrical charge, usually from within the device to an area where the charge can be

manipulated, for example conversion into a digital value. This is achieved by "shifting" the signals between stages within the device one at a time. CCDs move charge between capacitive bins in the device, with the shift allowing for the transfer of charge between bins.

In recent years CCD has become a major technology for digital imaging. In a CCD image sensor, pixels are represented by p-doped metal-oxide-semiconductors (MOS) capacitors. These capacitors are biased above the threshold for inversion when image acquisition begins, allowing the conversion of incoming photons into electron charges at the semiconductor-oxide interface; the CCD is then used to read out these charges. Although CCDs are not the only technology to allow for light detection, CCD image sensors are widely used in professional, medical, and scientific applications where high-quality image data are required. In applications with less exacting quality demands, such as consumer and professional digital cameras, active pixel sensors, also known as complementary metal-oxide-semiconductors (CMOS) are generally used; the large quality advantage CCDs enjoyed early on has narrowed over time.



Simplified programmable logic device

Content Addressable Memory (CAM)

Content-addressable memory (CAM) is a special type of computer memory used in certain very-high-speed searching applications. It is also known as associative memory, associative storage, or associative array, although the last term is more often used for a programming data structure.[1] It compares input search data (tag) against a table of stored data, and returns the address of matching data (or in the case of associative memory, the matching data).[2] Several custom computers, like the Goodyear STARAN, were built to implement CAM, and were designated associative computers.

Programmable Logic Devices(PLD)-

A programmable logic device (PLD) is an electronic component used to build reconfigurable digital circuits. Unlike a logic gate, which has a fixed function, a PLD has an undefined function at the time of manufacture. Before the PLD can be used in a circuit it must be programmed, that is, reconfigured.

Advantages of PLDs:

Problems of using standard ICs:
Problems of using standard ICs in logic design are that they require hundreds or thousands of these ICs, considerable amount of circuit board space, a great deal of time and cost in inserting, soldering, and testing. Also require keeping a significant inventory of ICs.

Advantages of using PLDs:

Advantages of using PLDs are less board space, faster, lower power requirements (i.e., smaller power supplies), less costly assembly processes, higher reliability (fewer ICs and circuit connections means easier troubleshooting), and availability of design software.

There are three fundamental types of standard PLDs: PROM, PAL, and PLA.

A fourth type of PLD, which is discussed later, is the Complex Programmable Logic Device (CPLD), e.g., Field Programmable Gate Array (FPGA).
A typical PLD may have hundreds to millions of gates.
In order to show the internal logic diagram for such technologies in a concise form, it is necessary to have special symbols for array logic.

Programmable logic Array (PLA) --

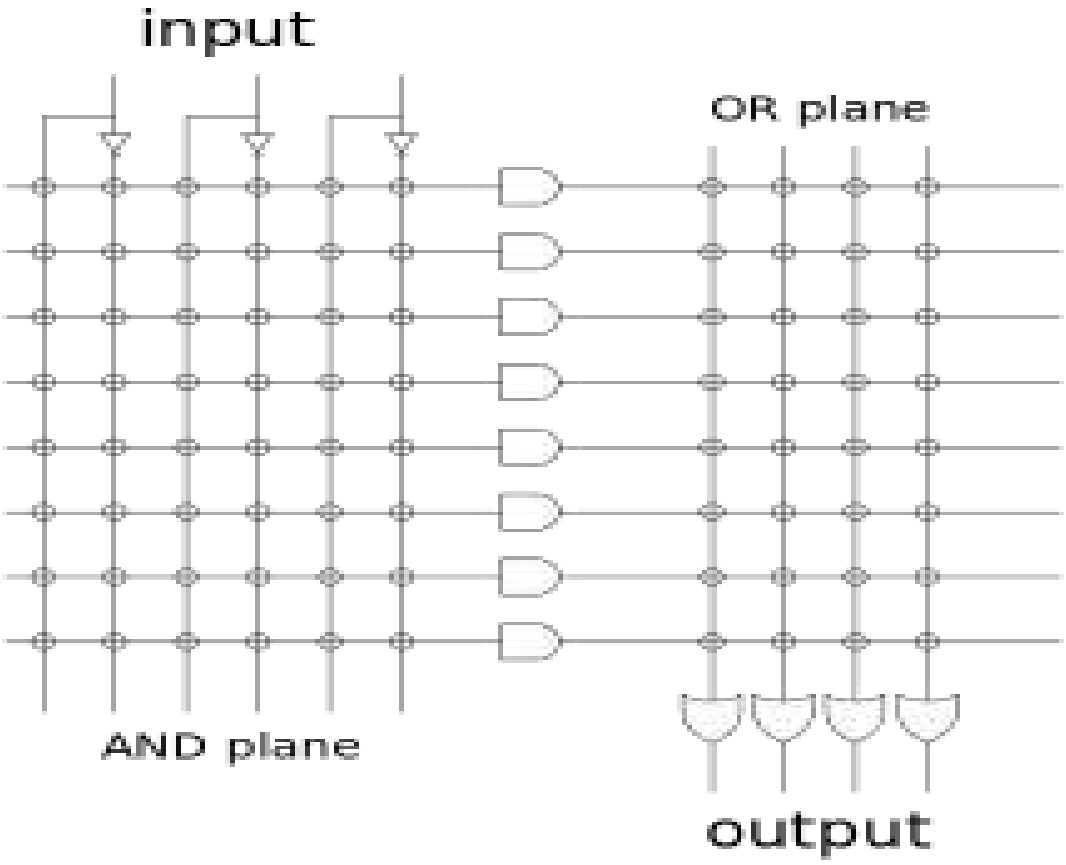
A programmable logic array (PLA) is a kind of programmable logic device used to implement combinational logic circuits. The PLA has a set of programmable AND gate planes, which link to a set of programmable OR gate planes, which can then be conditionally complemented to produce an output. It has 2^N AND Gates for N input variables and for M outputs from PLA, there should be M OR Gates, each with programmable inputs from all of the AND gates. This layout allows for a large number of logic functions to be synthesized in the sum of products canonical forms.

PLAs differ from Programmable Array Logic devices (PALs and GALs) in that both the AND and OR gate planes are programmable.

Programmable Array Logic (PAL)- PAL is a family of programmable logic device semiconductors used to implement logic functions in digital circuits introduced by Monolithic Memories, Inc. (MMI) in March 1978.[1] MMI obtained a registered trademark on the term PAL for use in "Programmable Semiconductor Logic Circuits". The trademark is currently held by Lattice Semiconductor.[2]

PAL devices consisted of a small PROM (programmable read-only memory) core and additional output logic used to implement particular desired logic functions with few components.

Using specialized machines, PAL devices were "field-programmable". PALs were available in several variants:



"One-time programmable" (OTP) devices could not be updated and reused after initial programming (MMI also offered a similar family called HAL, or "hard array logic", which were like PAL devices except that they were mask-programmed at the factory.).
UV erasable versions (e.g.: PALCxxxxx e.g.: PALC22V10) had a quartz window over the chip die and could be erased for re-use with an ultraviolet light source just like an EPROM.
Later versions (PALCExxx e.g.: PALCE22V10) were flash erasable devices.
In most applications, electrically-erasable GALs are now deployed as pin-compatible direct replacements for one-time programmable PALs.

Applications of PLAs

Considering the above advantages and disadvantages, PLAs have numerous unique applications. A micro- processor chip uses many PLAs because of easy of design change and check. In particular, PLAs are used in its control logic, which is complex and requires many changes, even during its design. Also, PLAs are used for code

conversions, microprogram address conversions, decision tables, bus priority resolvers, and memory overlay.

When a new product is to be manufactured in small volume or test-marketed, PLAs is a choice. When the new product is well received in the market and does not need further changes, PLAs can be replaced by random-logic gate networks for low cost for high volume production and high speed. Also, a full- custom design approach is very time-consuming, probably taking months or years, but if PLAs are used in the control logic, a number of different custom-design chips with high performance can be made quickly by changing only one connection mask for the PLAs, although these chips cannot have drastically different performance and functions.

Programmable Array Logic--

Programmable Array Logic (PAL) is a family of programmable logic device semiconductors used to implement logic functions in digital circuits introduced by Monolithic Memories, Inc. (MMI) in March 1978.[1] MMI obtained a registered trademark on the term PAL for use in "Programmable Semiconductor Logic Circuits". The trademark is currently held by Lattice Semiconductor.[2]

PAL devices consisted of a small PROM (programmable read-only memory) core and additional output logic used to implement particular desired logic functions with few components.

Using specialized machines, PAL devices were "field-programmable". PALs were available in several variants:

The figure shown below shows a programmable array logic with five inputs,ten programmable AND gates and five fixed OR gates.

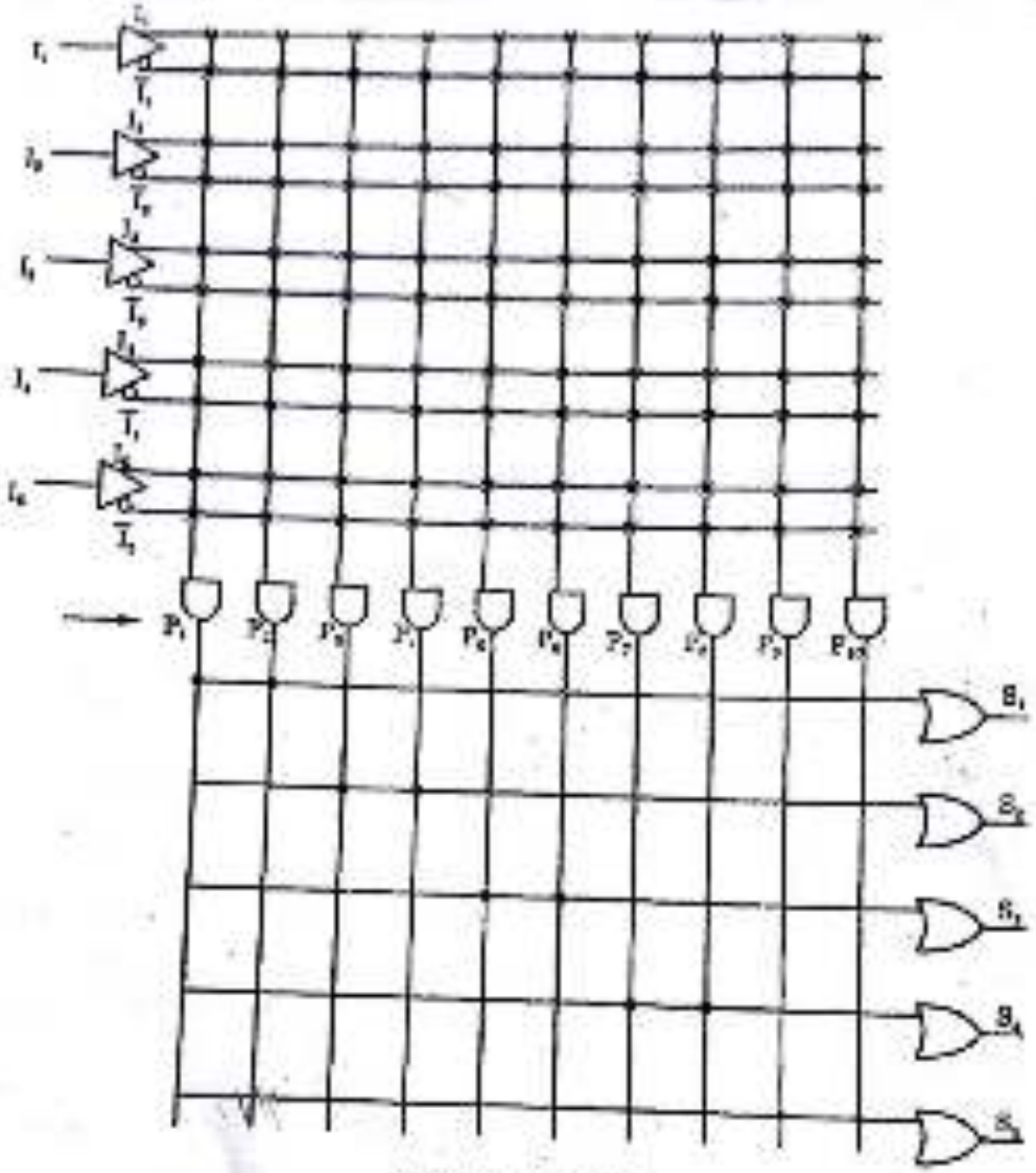


Diagram of a PAL

In the above diagram each AND gate is connected to all the ten inputs which are in complemented & uncomplemented form with fusible links. Each OR gate gets inputs from the outputs of the only two AND gates which is shown by • in the above diagram. The output & input circuits of PAL is similar to PLA. The number of fusible links in a PAL is given by the product of 2 M & n where n is the number of product terms & M represents the number of input variables.

Field Programmable Gate Array(FPGA)-

A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by a customer or a designer after manufacturing – hence "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC). (Circuit diagrams were previously used to specify the configuration, as they were for ASICs, but this is increasingly rare.)

FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together", like many logic gates that can be inter-wired in different configurations. Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates like AND and XOR. In most FPGAs, logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory.

CHAPTER 4 COMBINATIONAL CIRCUITS

COMBINATIONAL CIRCUITS

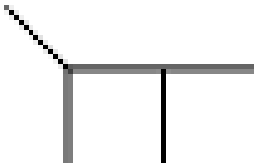
*Minimization of Boolean expression using k-map method

***Karnaugh Map (K-map)** (pronounced *car-no map*) is a graphical TOol that provides a simple and straightforward method of minimizing Boolean expressions. The K-map method was introduced in 1953 by Maurice Karnaugh as an enhancement TO Veitch diagram.

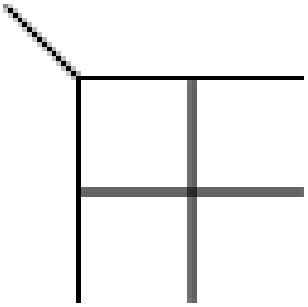
***Map Formats**[edit]

A K-map is a square or rectangle divided inTO a number of smaller squares called **cells**. Each cell on the K-Map corresponds directly TO a line in a truth table. There are always cells in a K-Map where is the number of variables in the function. Below are the usual formats for 1-4 variable k-maps (larges k-maps are discussed later on).

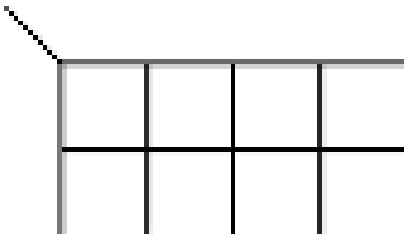
1-Variable K-map



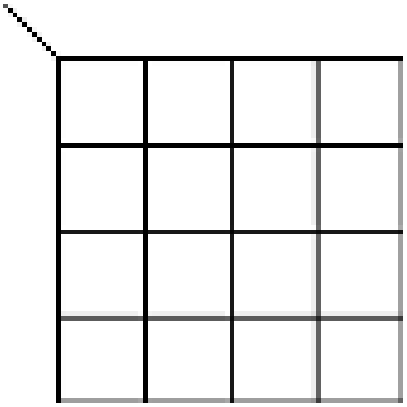
2-Variables K-map



3-Variables K-map



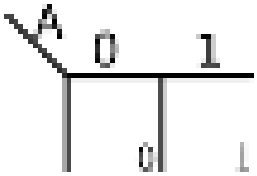
4-Variables K-map



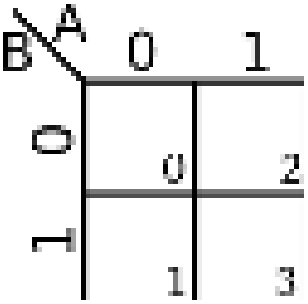
***Map Cell Numbering**

sometimes the individual cells are numbered in accordance with their minterm and maxtermindices. Strictly speaking this is unnecessary, but it may be useful in various situations when working with minterms and maxterms. Cell numbering are usually written in one of the cell corners.

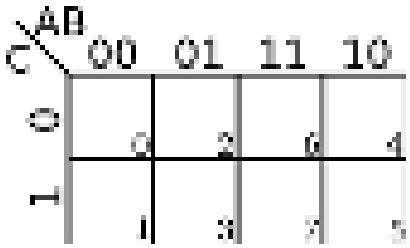
1-Variable K-map



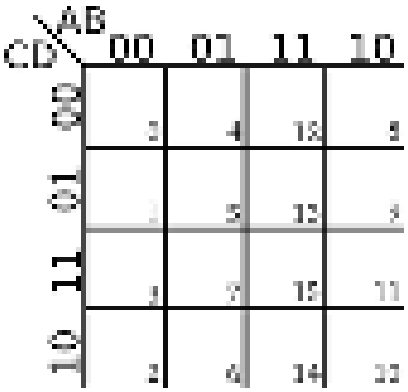
2-Variables K-map



3-Variables K-map



4-Variables K-map



***from Boolean expression**[edit]

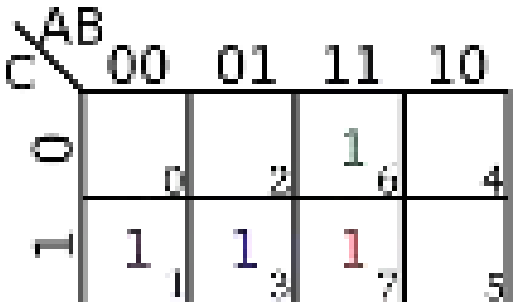
Because each cell on the K-map represents a particular minterm (or maxterm). Converting the desired Boolean function into sum of minterms form can help considerably.

Consider the following Boolean function.

To make it easier to transfer the data to a K-map, the equation can be manipulated a bit so that it's in sum of minterms canonical form.

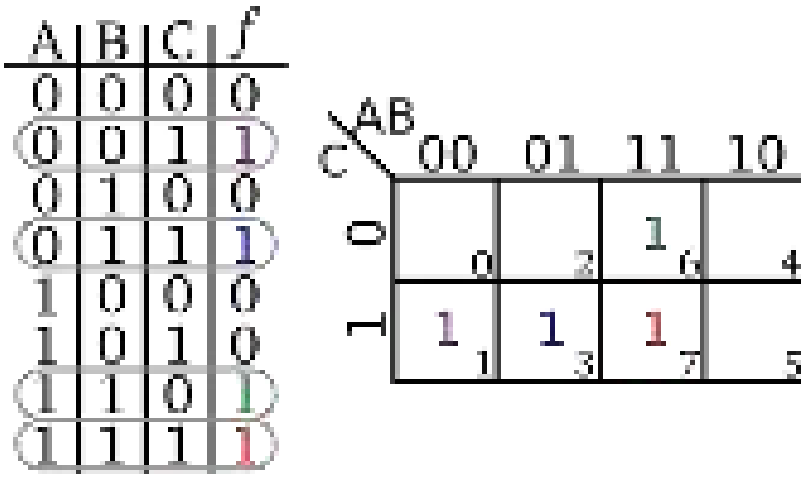
Each minterm in the equation is than transferred into the K-map where each variable in the minterm represents a 1 and each complemented variable represents a 0.

$$f(a,b,c) = \underset{111}{ABC} + \underset{110}{AB\overline{C}} + \underset{011}{\overline{A}BC} + \underset{001}{\overline{A}\overline{B}C}$$



***from truth table**[edit]

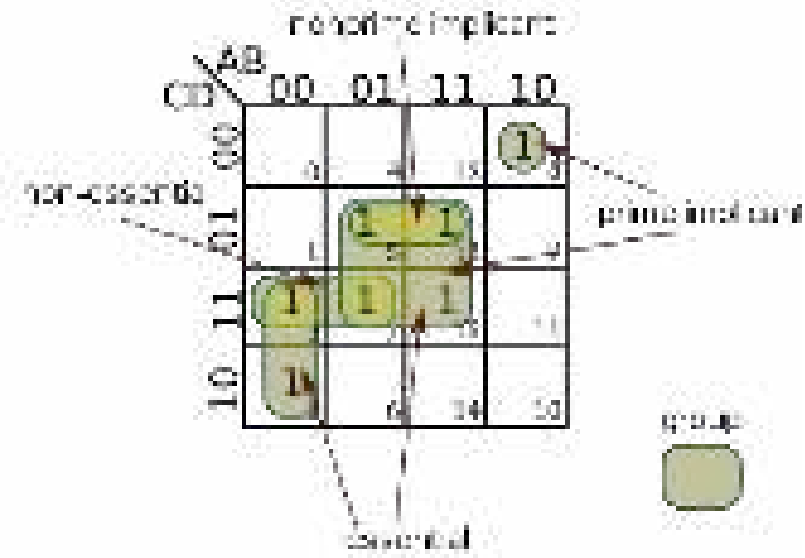
Transferring the data from a truth table to a K-map is slightly more straightforward since each cell corresponds directly to each row in the table. A cell on the K-map is labeled 1 when the row they represent in the truth table results in a 1; ot
herwise the cell is labeled 0. Often times if the cell is 0, the 0 itself is simply omitted and is understood to mean that.



Simplification

Generating simplified equations for a Karnaugh map involves two simple steps:

1. finding largest groups of 1s
2. generating an equation from the identified groups



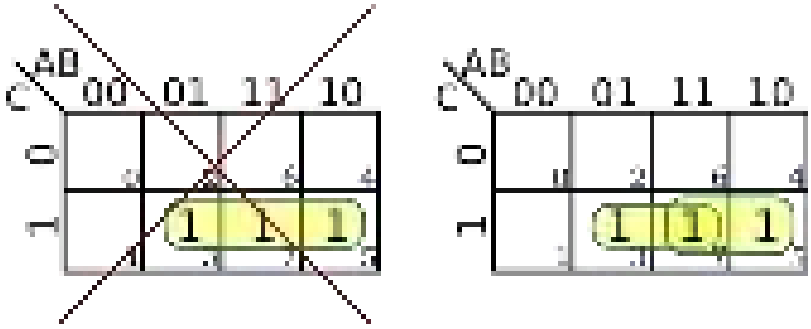
Groups

An **implicant** is the individual product term in the sum of product expression. On a K-map implicants are represented as one or more adjacent cells of 1s. A **group** is a loose term for the enclosure containing adjacent squares of 1s. When a group contains the most adjacent 1 cells it possibly can, it is called a **prime implicant**. When a group encloses cells of 1s that are not shared with any other group, it is called an **essential prime implicant**. Likewise, when a group encloses cells of 1s that are all shared with other groups, it is called a **non-essential prime implicant**.

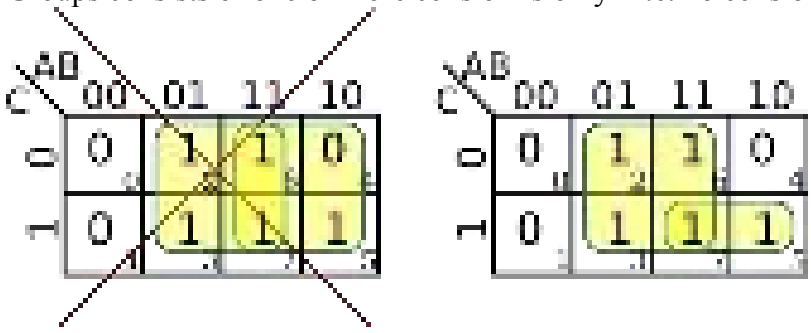
Rules

When grouping implicants together, there is a set of rules that must be followed:

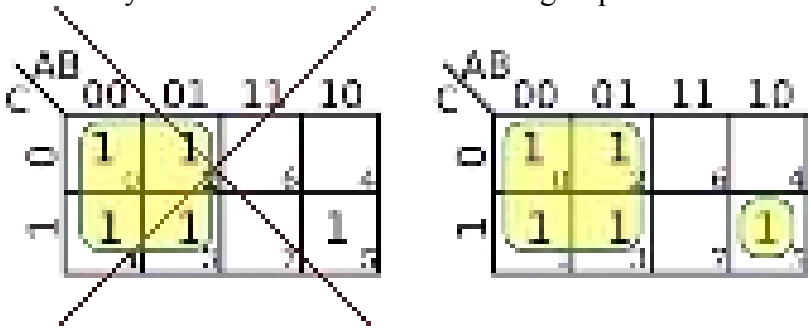
- Groups are made of power of 2 number of cells (e.g. 1, 2, 4, 8, 16)



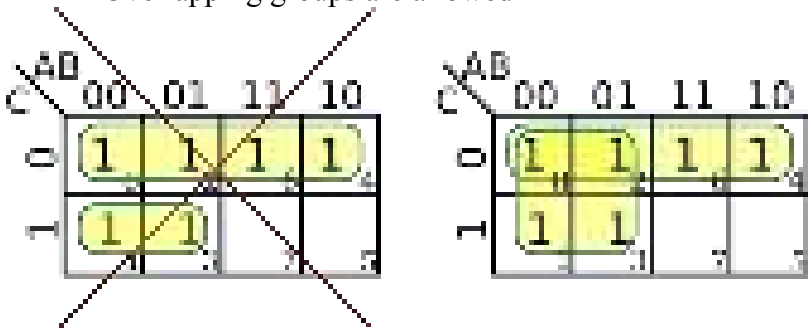
- Groups consists of one or more cells of 1s only - i.e. no cells of 0s



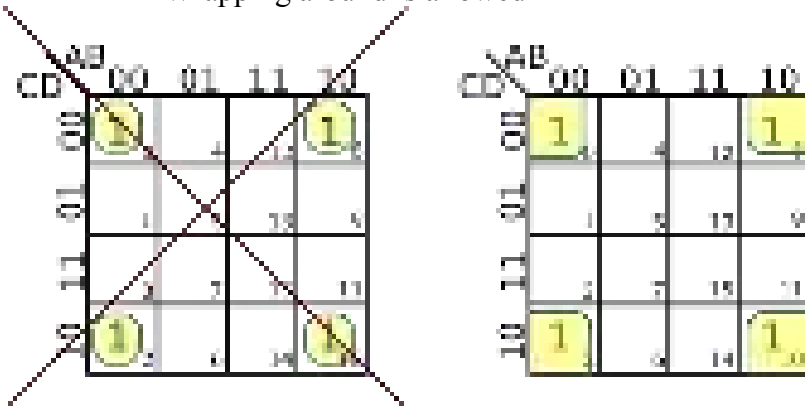
- Every cell of 1 must be in at lest one group



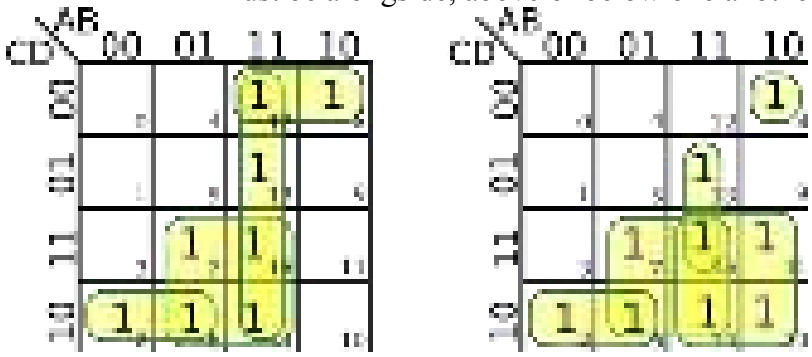
- Overlapping groups are allowed



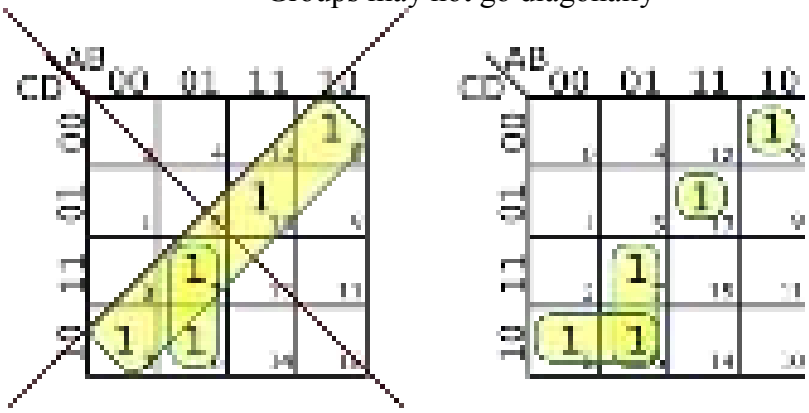
- Wrapping around is allowed



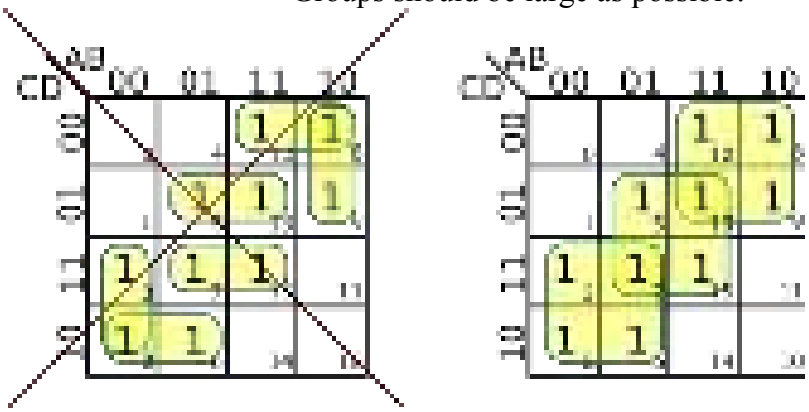
- Groups can be made of cells that are adjacent to one another. I.e. cells must be alongside, above or below one another



- Groups may not go diagonally

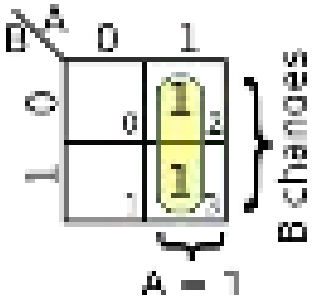


- Groups should be large as possible.

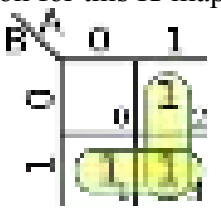


Simplified Equation

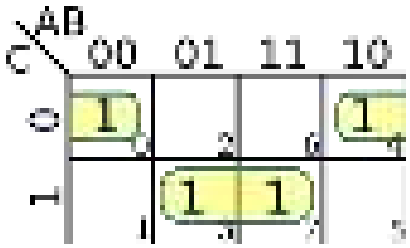
After all the maximum groups have been marked in the K-map, a simplified Boolean expression may be obtained by ORing together the individual expressions for each of the groups. The expression for a group is the variables or the complement of the variables that do not change between cells.



For example, consider the example to the left. This Karnaugh map has a single group that covers both and . Because changes, it is dropped from our expression, living us with just . Therefore the Boolean expression for this K-map is simply .



The Karnaugh map on the left on the other hand has two groups. One group spans both and and another that spans and . In the expression for the group that spans vertically, changes yielding the expression . Likewise in the expression that spans horizontally, changes, yielding the expression . The simplified equation for this K-map is the ORing of all the individual term - .



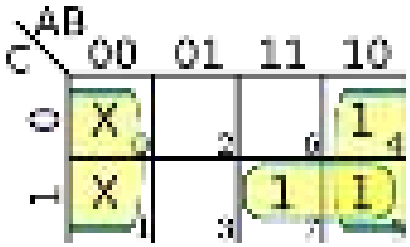
In this K-map, cells 0 and 4 are considered adjacent as well as cells 3 and 7. For the group involving cells 0 and 4, changes, therefore it is dropped from the expression. Because is always 0 and is always 0 as well, the equation for that group is . For the second group involving cells 3 and 7, changes once again. In this group is always 1 and is always 1 as well. The equation for this group is . The final simplified equation for this K-map is the ORing of all the terms - .

Don't cares

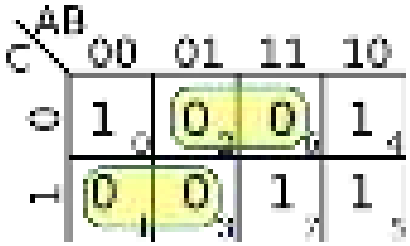
Further information

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

Incompletely specified function are functions with combination of inputs that should never occur. Those unspecified minterms are called don't care values. Don't care values open opportunities for further simplification of the Boolean expression. When it comes to Karnaugh maps, don't care values, which are represented with *X*'s are considered either a 0 or 1, whichever results in the biggest group - i.e. the simplest expression.



The truth table to the Karnaugh map below is on the right and represents an incompletely specified function. Note the don't care values for two of the outputs. I.e. and should never happen. In this example, the two don't care values can be used to make the 2-cell group be a 4-cell group allowing us to eliminate a whole variable: . The Boolean expression is therefore .Product of Sum (PoS)



While usually used to generate sum of products, Karnaugh maps can be used to generate product of sum just as easily by applying the same rules we designed above, but for 0 cells instead. Remember that when working with maxterms instead of minterms, when a variable is 1, it is complemented instead of when it's 0.

For example consider the K-map on the right. In this case the 0 cells are grouped. The Boolean function for this Karnaugh map.

*Tabular method of function minimization

Introduction

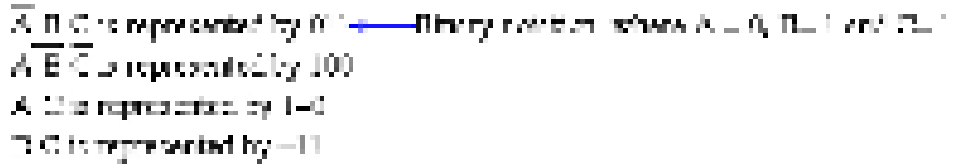
In order to understand the tabular method of minimisation, it is best you understand the numerical assignment of Karnaugh map cells and the incompletely specified functions also known as the **can't happen conditions**. This is because the tabular method is based on these principles.

The tabular method which is also known as the Quine-McCluskey method is particularly useful when minimising functions having a large number of variables, e.g. The six-variable functions. Computer programs have been developed employing this algorithm. The method reduces a function in standard sum of products form to a set of prime implicants from which as many variables are eliminated as possible. These prime implicants are then examined to see if some are redundant.

The tabular method makes repeated use of the law A + = 1. Note that Binary notation is used for the function, although decimal notation is also used for the functions. As usual a variable in true form is denoted by 1, in inverted form by 0, and the abscence of a variable by a dash (-).

Rules of Tabular Method

Consider a function of three variables f(A, B, C):



Consider the function:

Listing the two minterms shows they can be combined

Now consider the following:

Note that these variables cannot be combined

This is because the *FIRST RULE* of the Tabular method for two terms to combine, and thus eliminate one variable, is that they must differ in only **one digit** position.

Bear in mind that when two terms are combined, one of the combined terms has one digit more at logic 1 than the other combined term. This indicates that the number of 1's in a term is significant and is referred to as its index.

For example: f(A, B, C, D)

0000.....Index 0

0010, 1000.....Index 1

1010, 0011, 1001.....Index 2

1110, 1011.....Index 3

1111.....Index 4

The necessary condition for combining two terms is that the indices of the two terms must differ by one logic variable which must also be the same.

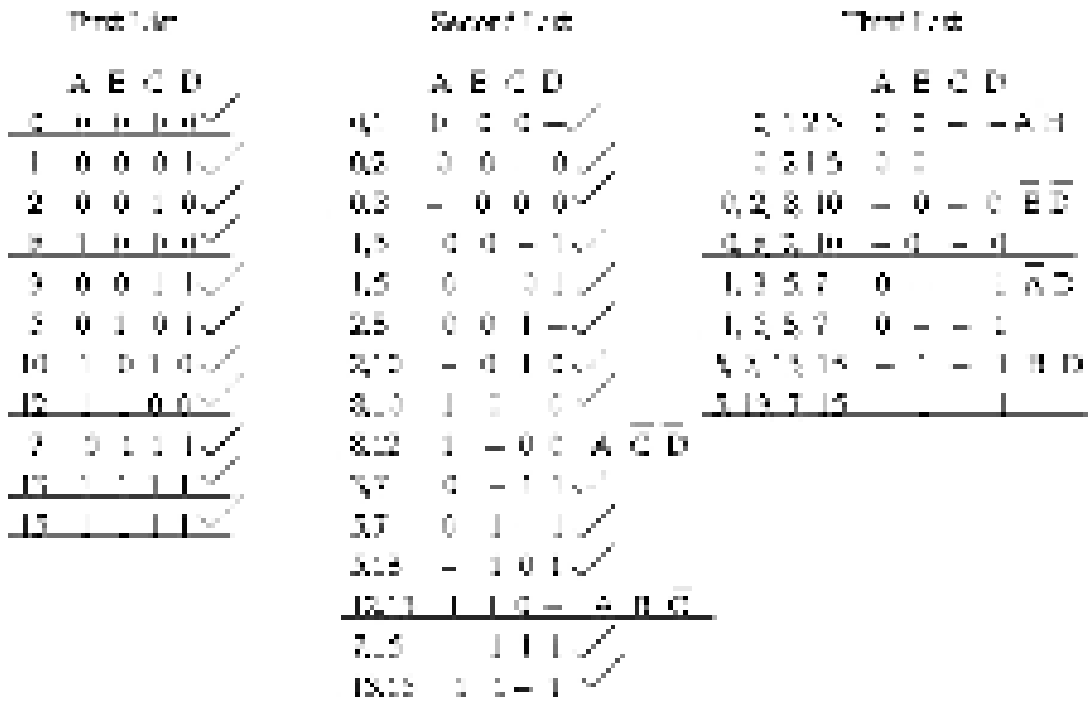
Examples

Example

Consider the function f(A, B, C, D) = $\sum$ (0,1,2,3,5,7,8,10,12,13,15), note that this is in decimal form.

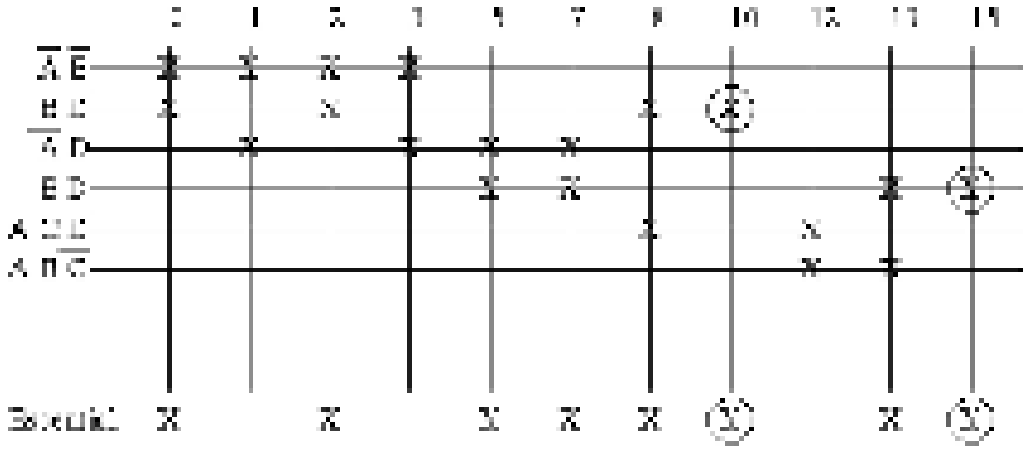
$\sum$ (0000,0001,0010,0011,0101,0111,1000,1010,1100,1101,1111) in binary form.

(0,1,1,2,2,3,1,2,2,3,4) in the index form.



The prime implicants are: $\bar{A}\bar{B} + \bar{B}\bar{C} + \bar{A}D + BD + A\bar{C}\bar{D} + AB\bar{C}$

The chart is used to remove redundant prime implicants. A grid is prepared having all the prime implicants listed at the left and all the minterms of the function along the top. Each minterm covered by a given prime implicant is marked in the appropriate position.



From the above chart, BD is an essential prime implicant. It is the only prime implicant that covers the minterm decimal 15 and it also includes 5, 7 and 13. $\bar{B}\bar{C}$ is also an essential prime implicant. It is the only prime implicant that covers the minterm denoted by decimal 10 and it also includes the terms 0, 2 and 8. The other minterms of the function are 1, 3 and 12. Minterm 1 is present in $\bar{A}\bar{B}$ and $\bar{A}D$. Similarly for minterm 3. We can therefore use either of these prime implicants for these minterms. Minterm 12 is present in $A\bar{C}\bar{D}$ and $AB\bar{C}$, so again either can be used.

Thus, one minimal solution is: $Z = \bar{B}\bar{C} + BD + \bar{A}\bar{B} + A\bar{C}\bar{D}$

*Quine mcclaaskey method

From Wikipedia, the free encyclopedia

The **Quine–McCluskey algorithm** (or **the method of prime implicants**) is a method used for minimization of Boolean functions that was developed by Willard V. Quine and extended

by Edward J. McCluskey.[1][2][3] It is functionally identical to Karnaugh mapping, but the tabular form makes it more efficient for use in computer algorithms, and it also gives a deterministic way to check that the minimal form of a Boolean function has been reached. It is sometimes referred to as the tabulation method.

The method involves two steps:

1. Finding all prime implicants of the function.
2. Use those prime implicants in a *prime implicant chart* to find the essential prime implicants of the function, as well as other prime implicants that are necessary to cover the function.

- Complexity

Although more practical than Karnaugh mapping when dealing with more than four variables, the Quine–McCluskey algorithm also has a limited range of use since the problem it solves is NP-hard: the runtime of the Quine–McCluskey algorithm grows exponentially with the number of variables. It can be shown that for a function of n variables the upper bound on the number of prime implicants is $3^n/\ln(n)$. If $n = 32$ there may be over $6.5 * 10^{15}$ prime implicants. Functions with a large number of variables have to be minimized with potentially non-optimal heuristic methods, of which the Espresso heuristic logic minimizer is the de-facto standard in 1995.[*needs update*][4]

Example[edit]

Step 1: finding prime implicants[edit]

Minimizing an arbitrary function:

This expression says that the output function f will be 1 for the minterms 4,8,10,11,12 and 15 (denoted by the 'm' term). But it also says that we don't care about the output for 9 and 14 combinations (denoted by the 'd' term). ('x' stands for don't care).

	A	B	C	D		f
m0	0	0	0	0		0
m1	0	0	0	1		0
m2	0	0	1	0		0
m3	0	0	1	1		0
m4	0	1	0	0		1
m5	0	1	0	1		0
m6	0	1	1	0		0
m7	0	1	1	1		0
m8	1	0	0	0		1
m9	1	0	0	1		x
m10	1	0	1	0		1
m11	1	0	1	1		1
m12	1	1	0	0		1
m13	1	1	0	1		0
m14	1	1	1	0		x
m15	1	1	1	1		1

One can easily form the canonical sum of products expression from this table, simply by summing the minterms (leaving out don't-care terms) where the function evaluates to one:

which is not minimal. So to optimize, all minterms that evaluate to one are first placed in a minterm table. Don't-care terms are also added into this table, so they can be combined with minterms:

Number of 1s	Minterm	Binary Representation
1	m4	0100
	m8	1000
2	m9	1001
	m10	1010
	m12	1100
3	m11	1011
	m14	1110
4	m15	1111

At this point, one can start combining minterms with other minterms. If two terms vary by only a single digit changing, that digit can be replaced with a dash indicating that the digit doesn't matter. Terms that can't be combined any more are marked with an asterisk (*). When going from Size 2 to Size 4, treat '-' as a third bit value. For instance, -110 and -100 can be combined, as well as -110 and -11-, but -110 and 011- cannot. (Trick: Match up the '-' first.)

Number of 1s	Minterm	0-Cube	Size 2 Implicants		Size 4 Implicants	
1	m4	0100	m(4,12)	-100*	m(8,9,10,11)	10--*
	m8	1000	m(8,9)	100-	m(8,10,12,14)	1--0*
	—	—	m(8,10)	10-0	—	
	—	—	m(8,12)	1-00	—	
2	m9	1001	m(9,11)	10-1	m(10,11,14,15)	1-1-*
	m10	1010	m(10,11)	101-	—	
	—	—	m(10,14)	1-10	—	
	m12	1100	m(12,14)	11-0	—	
3	m11	1011	m(11,15)	1-11	—	
	m14	1110	m(14,15)	111-	—	
4	m15	1111		—	—	

Note: In this example, none of the terms in the size 4 implicants table can be combined any further. Be aware that this processing should be continued otherwise (size 8 etc.).

Step 2: prime implicant chart[edit]

None of the terms can be combined any further than this, so at this point we construct an essential prime implicant table. Along the side goes the prime implicants that have just been generated, and along the top go the minterms specified earlier. The don't

care terms are not placed on top—they are omitted from this section because they are not necessary inputs.

	4	8	10	11	12	15	⇒	A	B	C	D
m(4,12)*	X				X		⇒	—	1	0	0
m(8,9,10,11)		X	X	X			⇒	1	0	—	—
m(8,10,12,14)		X	X		X		⇒	1	—	—	0
m(10,11,14,15)*			X	X		X	⇒	1	—	1	—

To find the essential prime implicants, we run along the top row. We have to look for columns with only 1 "X". If a column has only 1 "X", this means that the minterm can only be covered by 1 prime implicant. This prime implicant is *essential*.
For example: in the first column, with minterm 4, there is only 1 "X". This means that m(4,12) is essential. So we place a star next to it. Minterm 15 also has only 1 "X", so m(10,11,14,15) is also essential. Now all columns with 1 "X" are covered.
The second prime implicant can be 'covered' by the third and fourth, and the third prime implicant can be 'covered' by the second and first, and neither is thus essential. If a prime implicant is essential then, as would be expected, it is necessary to include it in the minimized boolean equation. In some cases, the essential prime implicants do not cover all minterms, in which case additional procedures for chart reduction can be employed. The simplest "additional procedure" is trial and error, but a more systematic way is Petrick's method. In the current example, the essential prime implicants do not handle all of the minterms, so, in this case, one can combine the essential implicants with one of the two non-essential ones to yield one equation:

[5]

Both of those final equations are functionally equivalent to the original, verbose equation:

CHAPTER 6 *ALU*

DIGITAL ELECTRONICS

ALU

- **After studying this section, you should be able to:**
- Understand Stuff.
- Understand the operation of a basic ALU circuit.
- • Stuff.
- Understand the relationship between Binary Arithmetic and digital circuits.
- • Twos complement arithmetic.
- • Adders & error flags.
- • Logic operations with shift registers.
- Use Free Stuff. .

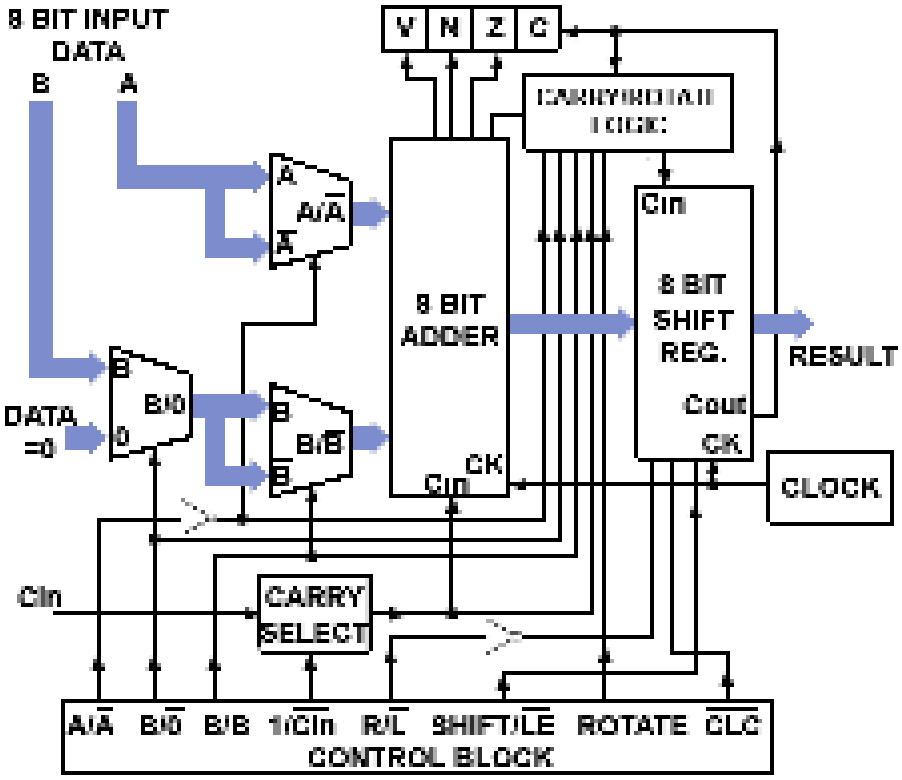


Fig. 5.8.1 ALU Block Diagram

Connecting Digital Circuits Together.

Digital Electronics Modules 2 to 5 have described how basic logic gates may be combined, not only to perform standard logic functions, but to build circuits that can perform complex logic tasks. Both small scale integrated (SSI) and medium scale integrated (MSI) chips are available in many forms, that can be directly connected together to make very complex circuits. It is this inter-connectivity that makes digital electronics so powerful and so versatile.

The standard circuits described in modules 2 to 5, both combinational and sequential, can be used to perform arithmetic operations such as addition, subtraction and counting, as well as logical operations such as combining data sources (multiplexing) and shifting bits left or right within a binary word.

As explained in Module 1, binary arithmetic is normally carried out electronically by using twos complement notation. The most common and versatile method of carrying out such operations is in an Arithmetic and Logic Unit (ALU), a circuit that forms the heart of any calculating or computing system.

The Arithmetic and Logic Unit

A simplified ALU is illustrated in Fig 5.8.1, which uses an arrangement of both combinational and sequential circuits from those described in modules 2 to 5. Their purpose is to perform the basic (though still complex) binary arithmetic described in Module 1.

Data passing through the ALU circuit does so on a system of buses, shown by the broad arrows in Fig. 5.8.1. These buses consist of groups of wires (usually as 8 parallel bits in simple systems) each carrying a single byte of binary data. In this system, data word A is the primary data source, and data word B is the secondary data source that may be added to, or subtracted from word A.

The ALU can also perform other operations. It can increment, add 1 to word A, or decrement, subtract 1 from it. By complementing (inverting) the logic value of individual bits of the data word A and adding 1 to the result, it is possible to use twos complement arithmetic to perform subtractions.

The shift register at the ALU output can also perform a 'logical shift-left' on word A by shifting the 8 bits consecutively into the carry bit, alternatively the shift register can create a rotating pattern of bits, rotating left, and using the carry bit as a ninth bit in the sequence, or rotate the 8 bits right ignoring the carry bit. Any of these functions can be selected by the control block, using various combinations of the eight control lines shown in Fig. 5.8.1.

Putting the correct pattern of 1s and 0s (the control word) on the control lines will cause the ALU to perform the required arithmetic or logical operation on the data being input at A and B. With a control word of 8-bits, this could potentially allow up to 256 different combinations, or control words, which would be more than ample, even for very complex microprocessors or micro controllers. However this basic ALU needs only eight control words to control the different operations available.

To see the ALU operate as described below, you can download our free, fully interactive LogisimALU circuit (assuming you have the free Logisim Digital Simulator installed on your desktop or laptop computer), see our extra Logisim page for details.

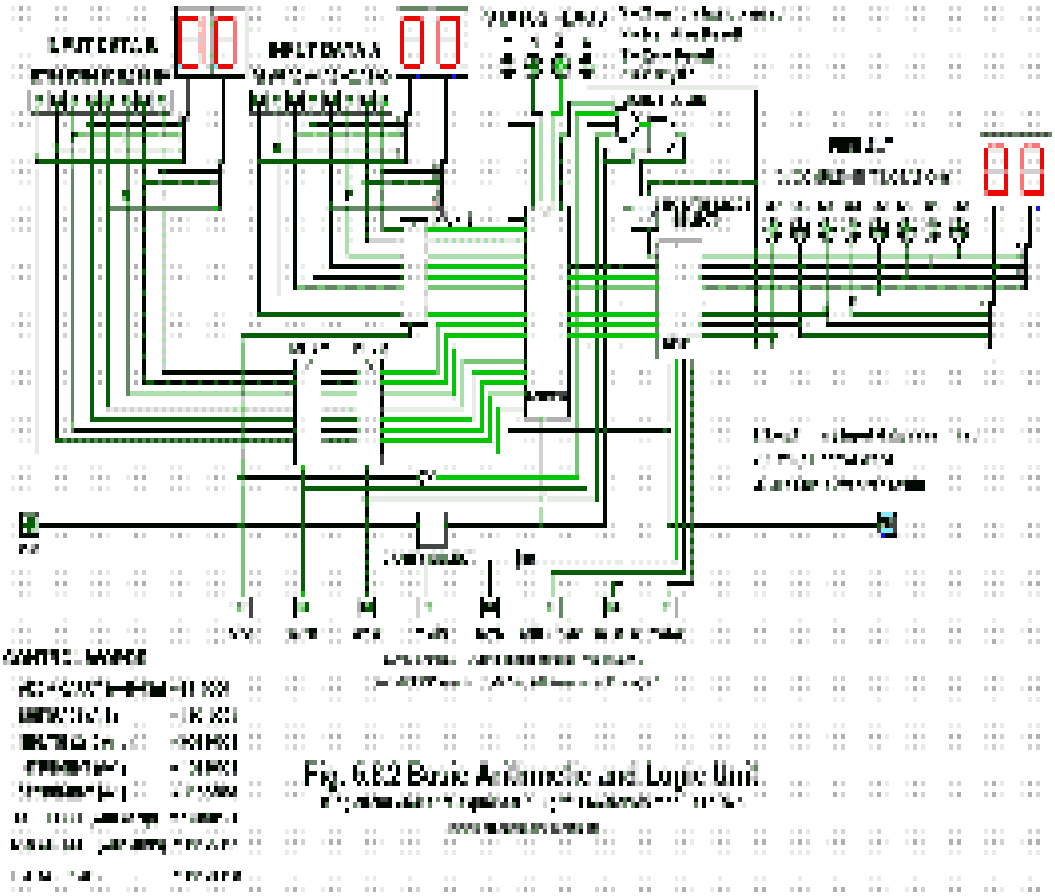


Fig. 5.8.2 Basic Arithmetic and Logic Unit - Logisim Simulation

The Component Parts

Any of the component parts of the Logisim design can be examined in detail by double clicking on the component (in simulation mode). To return to the main document, click 'main' in the component menu at the left of the screen.

Note: In this section the tilde character ~ is used where necessary to indicate NOT (e.g. ~LE = LE) to match the usage in the Logisim simulations.

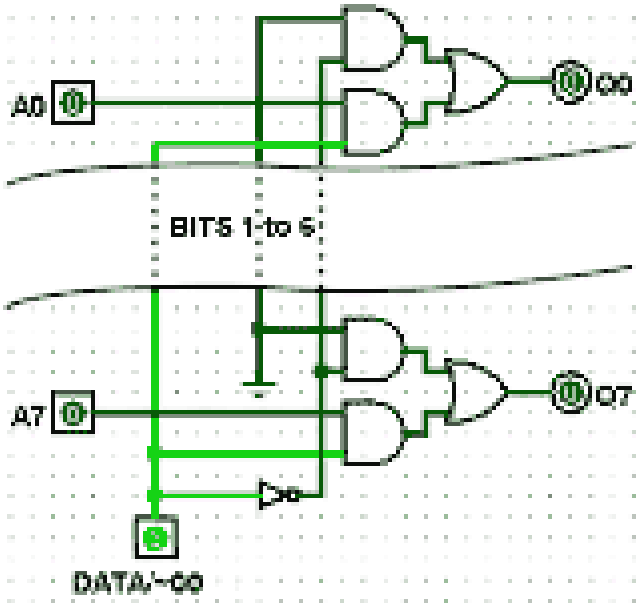


Fig. 5.8.4 MUX 2

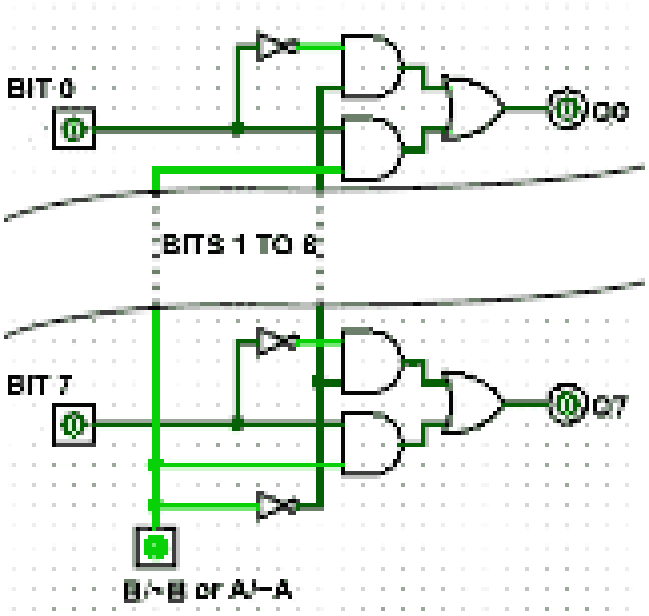


Fig. 5.8.3 MUX 1 and MUX 3

Multiplexers

MUX 1 and MUX 3 are identical 8 bit multiplexers that select either the input data word A (MUX 1) or data word B (MUX 3) or their internally generated complement, as shown in Fig. 5.8.3.

MUX 2 is a similar design but selects either the data word B or the zero value 00_{HEX} , as shown in Fig. 5.8.4.

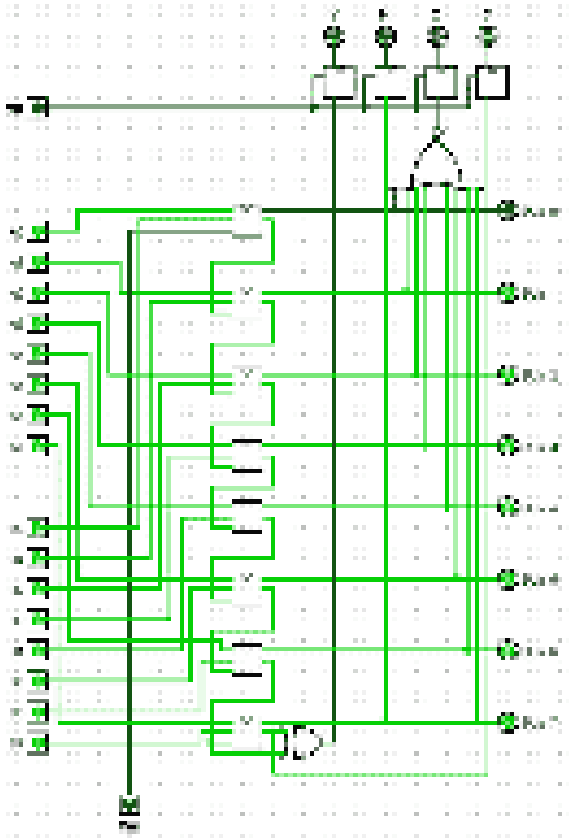


Fig. 5.8.5 The ALU Adder Component

8-Bit Adder

The adder component is an 8-bit ripple carry adder; real ALUs would normally feature a 'carry-look-ahead' adder, allowing for high-speed operation. However for this example the much simpler ripple carry adder is adequate, as the operation is totally manual.
The adder component is illustrated in Fig. 5.8.5 and consists of eight full-adder circuits with additional logic consisting of an XOR gate to detect overflow errors, and an 8-input NOR gate to detect a zero result.
Negative results are indicated by sampling the most significant bit of the 'sum' output, and a 'carry' is indicated by sampling the carry output of the most significant full adder.
Four D type flip-flops are used as 'flag' outputs to indicate the current state of the ALU after each operation.

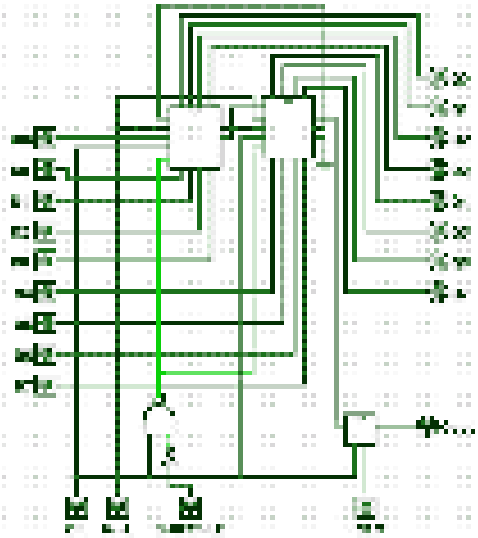


Fig. 5.8.6 The ALU Shift Register Component

The Shift Register

This component uses two 4-bit shift registers (from Module 5.7) connected in cascade as shown in Fig. 5.8.6. Inputs are provided for clock pulses, (CK), a right/left shift control (R/~L) and an input to control whether the shift register is in shift, or load-enable modes (SHIFT/~LE).
If ~LE is chosen temporarily during shift operations, the shift register can be reloaded from the data placed on the 8-bit 'Data A' and 'carry-in' (C_{IN}) inputs. This action is synchronised to the CK pulse by the external NAND and NOT gates connecting the SHIFT/~LE input to the two ~LOAD inputs of the 4-bit shift registers.
An additional JK flip-flop (mimicking a D type flip-flop) is placed between the 'serial-right' output of the shift register and C_{OUT} to allow the 'clear carry' input (~CLC) to clear the carry flag.

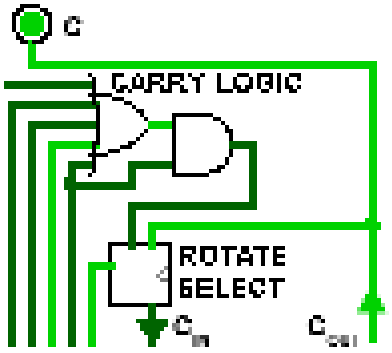


Fig.5.8.7 ALU Carry Logic

Carry Logic and Rotate Select

The carry logic circuit shown in Fig. 5.8.7 prevents the carry flag being set in rotate right mode, as bits rotate from bit 0 and re-enter the shift register at bit 7, therefore allowing correct carry flag operation in both left and right rotate modes.
When the ROTATE input is at logic 1, the Rotate Select circuit in Fig 5.8.7 allows C_{OUT} from the shift register to be fed back to the shift register C_{IN} input for continuous bit rotation.

ALU Operation

Addition

To perform an addition, input data B is added to A. This is achieved by putting logic 1 on the control inputs of multiplexers 1, 2 and 3. This causes data A and B to be applied to the adder inputs. Also, to allow any carry bit from the C_{IN} input to be included in the addition, the 1 bit carry multiplexer must have logic 0 on its control input. The shift register is only used as a PIPO register in addition mode, so its input lines R/~L and ROTATE must be at logic 0. SHIFT/~LE must also be at logic 0 to enable parallel loading of the shift register, which will hold the result of the addition (A plus B) after the application of a single CK pulse.

The Status Flags

The Flag flip-flops are special outputs from the adder circuit. They consist of four separate D type flip-flops, each of which can be set to 1 or cleared to 0. They are set or cleared by the result in the adder. They signal, or 'flag' to the user, that a particular event has occurred.

The Carry flag (C)

The carry flag will be set if the result of any arithmetic or logic event causes a logic 1 to be carried over from bit 7 into the 'carry bit', (which is the carry flag). The carry flag can be cleared at any time by making the 'clear carry' input (~CLC) logic 0.

The Overflow flag (V)

When carrying out twos complement arithmetic, errors can occur if large numbers are involved. For example if two positive numbers less than 127_{10} are added and produce a negative result (any value greater than 127_{10}). This would cause the sign of the result (indicated by bit 7) to be wrong. The overflow flag gives an indication that an error has occurred by being set to 1 to indicate an 'overflow error'. An error is sensed and the overflow flag is set when either of two conditions occurs.
There is a carry of logic 1 from bit 6 to bit 7 of the result, but the carry flip-flop is not at logic 1.
There is no carry from bit 6 to bit 7 of the result, but the carry flip-flop is at logic 1.
By using the carry-out from bit 6 and the carry-out from bit 7 of the result as inputs to an XOR gate, the output of the gate will be set to logic 1 for either of the above error conditions, signalling an overflow error at the overflow (V) flag

The Zero flag (Z)

This flip-flop is set when every bit of the result is zero.

The Negative flag (N)

A negative result, i.e. bit 7 = 1 sets this flip-flop to logic 1.

The Flag Register

The status flags are individual bits of a register called the Flag Register, and are operative not only when the ALU is in addition mode, but also in all other arithmetic modes, the C flag is also operative in shift and rotate left modes. In microprocessors the flag register not only indicates ALU results, but can also be used in decision-making. For example the ALU can be used to compare (by subtracting) two values and take various actions depending on the state of particular flags; e.g. after comparing two values, A and B, an action may be taken if A = B, indicated by the zero flag being set to 1, otherwise (if the zero flag is set to 0) take no action.

Subtraction.

Subtraction is performed using twos complement arithmetic. That is, to subtract B from A, input B is complemented and 1 added to the complemented value to form the twos complement. Then the twos complement of B is added to A in the adder to find the result. To achieve this action with data A and data B present at the inputs, logic 1 is applied to the control inputs of MUX 1 and MUX 2. MUX 3 has logic 0 applied to its control input to complement data B, while the CARRY MUX has a logic 1 applied to its control line so that the carry-in (C_{IN}) to the adder is forced to logic 1. This adds 1 to the result so that the twos complement of data B is added to data A. The result at the adder output is a twos complement number representing A - B. The flags are again set by the result as in the addition operation.

Counting with the ALU

Although the ALU does not include a binary counter circuit, it can also be used to count, by INCREMENTING or DECREMENTING, i.e. to add 1 to data A (incrementing), or subtract 1 from data A (decrementing). To count using this method would normally be carried out using (machine code or assembly language)software. A typical use could be to initiate a time delay by loading the ALU with some number, and then execute a looping routine to count down to zero by repeatedly decrementing data A. The zero result would be detected from the zero flag being set. However this would not be a common method, as the ALU (and therefore the CPU) would be occupied during the delay, and therefore not usable for other purposes. Most computer systems would also have dedicated counters for implementing similar time delays.

Incrementing.

Data A can be incremented if logic 1 is applied to the control inputs of MUX 1 and MUX 3. This will add B to A, with data B made zero by applying logic 0 to the control input of MUX 2. The 1 that must be added to data A is supplied by making the control input of the CARRY SELECT block logic 1, causing the carry input to the adder to be logic 1. The result at the adder output is therefore A + 1, again the flags are set by the result.

Decrementing.

To decrement data A, 1 must be subtracted from A. Because the ALU uses twos complement arithmetic, the twos complement of 1 added to A will in effect subtract 1 from A.

The twos complement of 1 is minus 1, which in 8-bit twos complement notation is 11111111_2. Therefore to subtract 1 from data A, data B must equal minus 1 (all bits = 1). To do this, and to make sure that the correct result is not changed by any data appearing on the data B input, logic 0 is applied to the control input of MUX 2 to make sure all data B bits = 0. Logic 0 is also applied to the control input of MUX 3. This inverts data B, (which is 00000000_2) to give 11111111_2 at the adder input.

MUX 1 must have logic 1 on its control line, to apply data A to the other adder input. The adder's carry input is set to 1 by applying logic 0 to the control line of the CARRY MUX. This ensures that, provided there is no carry-in on the C_{IN} input, the correct result at the adder output will be A − 1.

Negation

Negation is simply the inverse of a value; therefore any value and its inverse will add to produce zero. In binary arithmetic the additive inverse of a value is its twos complement. The ALU can be used to negate (find the twos complement of) data A by complementing data A and then adding 1. This involves a similar process to decrementing, except that data B is treated differently, as follows:

The control input of MUX 1 is set to logic 0, which complements data A, also data B is made zero by putting logic 0 on MUX 2 control, and logic 1 on MUX 3. The Carry Select control input is set at logic 1, to add 1 to data A in the adder.

The shift register is used as a simple PIPO register by applying logic 0 to the three shift controls and logic 1 to the ~CLC input to make sure the carry is not cleared. This gives a final result of A+1, which is the twos complement of A.

The Shift Operations

Shift operations are controlled by the four lower order control lines, R/~L controls the direction of shift or rotation, SHIFT/~LE has the dual purpose of enabling the shift operations if logic 1 is applied, or acting as a LOAD ENABLE when at logic 0, allowing the shift register to be loaded or reloaded with appropriate data. Each action of the shift register (shift, rotate or load) is actuated by a single CK pulse. Also note that the shift register in this design does not affect the V, N or Z flags.

Shift Left (with Carry)

In this mode (with control word 10100101) input data B is kept at zero and, after the shift register is loaded by temporarily making SHIFT/~LE logic 0 to move data from input A into the shift register, shift is enabled by returning SHIFT/~LE to logic 1, and both ROTATE and ~CLC are disabled. The data in the shift register will now shift one bit to the left with each CK pulse applied. This appears to multiply the value of the data by two for each shift left, but it is a very limited multiplication operation, because the result is reduced each time the left most bit is lost as it passes through the carry bit. This action is therefore considered a logical, rather than an arithmetic shift.

Rotate Left (with Carry)

If rotate is activated by applying logic 1 to the ROTATE control input with SHIFT/~LE and ~CLC also at logic 1, the data being shifted left from bit 7 and through the carry flag, is returned via the C_{IN} input of the shift register to re-enter at bit 0 by the action of the ROTATE MODE SELECT data selector.

Rotate Right

When data in the shift register is rotated right, it leaves the register via bit 0 and is returned directly to bit 7 via an internal link, without passing through the carry flag.

There are a number of other operations, such as performing 8 bit logic functions, commonly found on microprocessors that this ALU is not designed to do. The purpose of this design is to illustrate how the circuits described in Digital Electronics Modules 1 to 5 are really just part of a bigger picture, they can be inter-connected in many ways to make many different circuits. This ALU design is one example, but how you use what you learn from the pages of learnabout-electronics and how you fit that knowledge into your own imagination is up to you.

Top of Page

Binary Multiplication & Division

•Today's topics: Addition/Subtraction Multiplication Division

•Reminder: get started early on assignment 3

2

2's Complement –Signed Numbers

0000 0000 0000 0000 0000 0000 0000 0000two= 0ten 0000 0000 0000 0000 0000 0000 0000 0001two= 1ten … 0111 1111 1111 1111 1111 1111 1111 1111two= 231-1

1000 0000 0000 0000 0000 0000 0000 0000two= -231 1000 0000 0000 0000 0000 0000 0000 0001two= -(231–1) 1000 0000 0000 0000 0000 0000 0000 0010two= -(231–2) … 1111 1111 1111 1111 1111 1111 1111 1110two= -2 1111 1111 1111 1111 1111 1111 1111 1111two= -1

Why is this representation favorable? Consider the sum of 1 and -2 …. we get -1 Consider the sum of 2 and -1 …. we get +1
This format can directly undergo addition without any conversions!
Each number represents the quantity x31-231+ x30230+ x29229+ … + x121+ x020
3
Alternative Representations
•The following two (intuitive) representations were discarded because they required additional conversion steps before arithmetic could be performed on the numbers sign-and-magnitude: the most significant bit represents +/-and the remaining bits express the magnitude one's complement: -x is represented by inverting all the bits of x
Both representations above suffer from two zeroes
4
Addition and Subtraction
•Addition is similar to decimal arithmetic
•For subtraction, simply add the negative number –hence, subtract A-B involves negating B's bits, adding 1 and A
5
Overflows
•For an unsigned number, overflow happens when the last carry (1) cannot be accommodated
•For a signed number, overflow happens when the most significantbit is not the same as every bit to its left

when the sum of two positive numbers is a negative result

when the sum of two negative numbers is a positive result

The sum of a positive and negative number will never overflow
•MIPS allows adduandsubuinstructions that work with unsigned integers and never flag an overflow –to detect the overflow, other instructions will have to be executed
6
Multiplication Example
Multiplicand1000ten Multiplierx 1001ten ----------1000 0000 0000 1000 ---------- Product1001000ten
In every step •multiplicand is shifted •next bit of multiplier is examined (also a shifting step) •if this bit is 1, shifted multiplicand is added to the product
7
HW Algorithm 1
In every step •multiplicand is shifted •next bit of multiplier is examined (also a shifting step) •if this bit is 1, shifted multiplicand is added to the product
8
HW Algorithm 2
•32-bit ALU and multiplicand is untouched •the sum keeps shifting right •at every step, number of bits in product + multiplier = 64, hence, they share a single 64-bit register
9
Notes
•The previous algorithm also works for signed numbers (negative numbers in 2's complement form)
•We can also convert negative numbers to positive, multiply the magnitudes, and convert to negative if signs disagree

•The product of two 32-bit numbers can be a 64-bit number --hence, in MIPS, the product is saved in two 32-bit registers

10

MIPS Instructions

mult $s2, $s3 computes the product and stores it in two “internal” registers that can be referred to as hiand lo

mfhi $s0 moves the value in hiinto $s0 mflo $s1 moves the value in lointo $s1

Similarly for multu

11

Fast Algorithm

•The previous algorithm requires a clock to ensure that the earlier addition has completed before shifting

•This algorithm can quickly set up most inputs –it then has to wait for the result of each add to propagate down –faster because no clock is involved

--Note: high transistor cost

12

Division

1001tenQuotient Divisor1000ten| 1001010tenDividend -1000 10 101 1010 -1000 10tenRemainder

At every step, •shift divisor right and compare it with current dividend •if divisor is larger, shift 0 as the next bit of the quotient •if divisor is smaller, subtract to get new dividend and shift 1 as the next bit of the quotient

13

Division

1001tenQuotient Divisor1000ten| 1001010tenDividend

0001001010 0001001010 0000001010 0000001010 100000000000 000100000000001000000000001000 Quo: 0 000001 0000010 000001001

At every step, •shift divisor right and compare it with current dividend •if divisor is larger, shift 0 as the next bit of the quotient •if divisor is smaller, subtract to get new dividend and shift 1 as the next bit of the quotient

IC 74181

The SN54/74LS181 is a 4-bit Arithmetic Logic Unit (ALU) which can perform all the possible 16 logic, operations on two variables and a variety of arithmetic operations. • Provides 16 Arithmetic Operations Add, Subtract, Compare, Double, Plus Twelve Other Arithmetic Operations • Provides all 16 Logic Operations of Two Variables Exclusive — OR, Compare, AND, NAND, OR, NOR, Plus Ten other Logic Operations • Full Lookahead for High Speed Arithmetic Operation on Long Words • Input Clamp Diodes

CHAPTER -7 INTRODUCTION FUZZY LOCIC

Fuzzy logic is a form of many-valued logic in which the truth values of variables may be any real number between 0 and 1. It is employed to handle the concept of partial truth, where the truth value may range between completely true and completely false. By contrast, in Boolean logic, the truth values of variables may only be the integer values 0 or 1.
Fuzzy logic starts with and builds on a set of user-supplied human language rules. The fuzzy systems convert these rules to their mathematical equivalents. This simplifies the job of the system designer and the computer, and results in much more accurate representations of the way systems behave in the real world.

1. FUZZY SETS

A fuzzy sets is a set with a smooth boundary. Usually the, the membership is denoted by the Greek lower case microprocessor In mathematics, **fuzzy sets** (aka **uncertain sets**) are somewhat like sets whose elements have degrees of membership. In classical set theory, the membership of elements in a set is assessed in binary terms according to a bivalent condition — an element either belongs or does not belong to the set. By contrast, fuzzy set theory permits the gradual assessment of the membership of elements in a set; this is described with the aid of a membership function valued in the real unit interval [0, 1]. The membership of elements in a set is assessed in binary terms according to a bivalent condition — an element either belongs or does not belong to the set. By contrast, fuzzy set theory permits the gradual assessment of the membership of elements in a set; this is described with the aid of a membership function valued in the real unit interval [0, 1]

2. OPERATION OF FUZZY SET

Among the basic operations which can be performed on fuzzy sets are the operations of union, intersection, complement, algebraic product and algebraic sum. In addition to these operations, new operations called "bounded-sum" and "bounded-difference" were defined by L.

3. INTERSECTION AND UNION OF FUZZY SETS

The membership function of the Intersection of two fuzzy sets A and B with membership functions and respectively is defined as the minimum of the two individual membership functions. This is called the *minimum* criterion. Or the sets operation intersection and union same as logical operation, conjunction (and) and disjunction(or).

$\mu\ (A \cap B)^{(x)} = \mu_A(x) \times \mu_B(x)$

$\mu\ (A \cap B)^{(x)} = \mu_A(x) + \mu_B(x) - \mu_A(x) \times \mu_B(x)$

There is infinite number of other choices. The choice of fuzzy conjunction operator determines the choice of fuzzy disjunction operator and vice-versa.

4. Complement of a Fuzzy set

Complement reflects negation. We can define the complement of a fuzzy set in terms of the algebraic complement of its membership function μ. Let a be a fuzzy set defined over U. then its complement, -A is defined in term of μA (μ) as

$$\mu\text{-}A\ (\mu) = 1\ \mu A\ (\mu)$$

5. SUB SET HOOD

Subset hood measure is an important concept of fuzzy set theory. In the literature, there are several well known measures of subset hood. It stands for the degree that a given fuzzy set B is a subset of another set A. Many contributions on the measure of the degree of subset hood between two fuzzy sets have already been made. This stand as

Fuzzy relations

A fuzzy relation generalized the notice of a classical black and white relation into one that allows partial membership. For example, a binary relation “friend” will classify all human relationship into either being friend or not being friend or not being friend. A fuzzy relation “friend”, in contrast can describe the degree of friendship b/w two persons.

A binary relation on variables x and y, whose domains are X and Y respectively, can be defined as se of ordered pair in X ×Y. for instance, the binary relation “less than” between real number can be define as

$$R= \{(x, y) \mid x < y, x, y \in R\}$$

Relation is a subset of X × Y. we can say it as, as n-ray relation on $x_{1,} x_{2,,} \ldots . a_n$ whose domains are X_1, X_2 …..A_n is a subset of $X_1 \times X_2 \times \ldots\ldots \times A_n$.

6. Membership functions and its properties

A membership function for a fuzzy set A on the universe of discourse X is defined as μ_A: X → [0, 1], where each element of X is mapped to a value between 0 and 1. This value, called membership value or degree of membership, quantifies the grade of membership of the element in X to the fuzzy set A.

Membership functions allow us to graphically represent a fuzzy set. The *x* axis represents the universe of discourse, whereas the *y* axis represents the degrees of membership in the [0, 1] interval. There are type membership functions

a. **TRIANGULAR MEMBERSHIP FUNCTION**

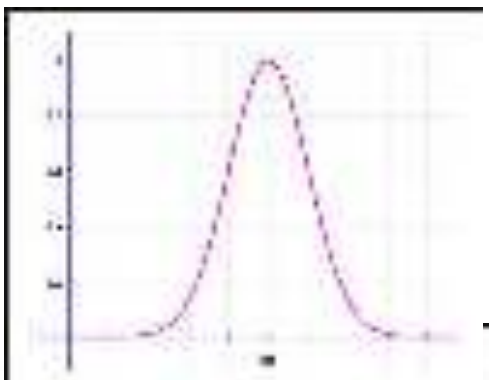
Defined by a lower limit **a**, an upper limit **b**, and a value **m**, where **a < m < b**.

Triangle

(X: a, b, c,)

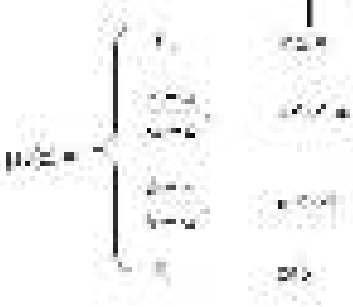
b. **GAUSSIAN MEMBERSHIP FUNCTION**:

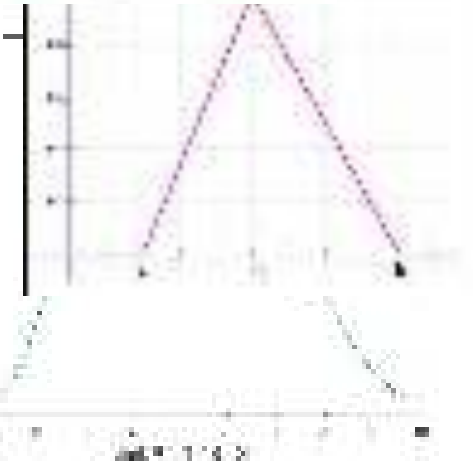
defined by a central value **m** and a standard deviation **k > 0**. The smaller k is, the narrower the “bell” is.

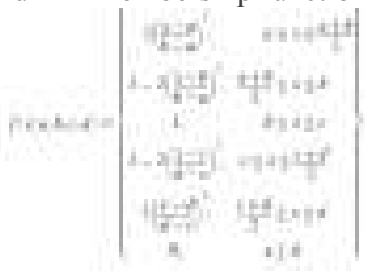


c. PI- MEMBERSHIP FUNCTION

This spline-based curve is so named because of its Π shape. The membership function is evaluated at the points determined by the vector x. The parameters *a* and *d* locate the "feet" of the curve, while *b* and *c* locate its "shoulders." The membership function is a product of smf and zmf membership functions, and is given by:





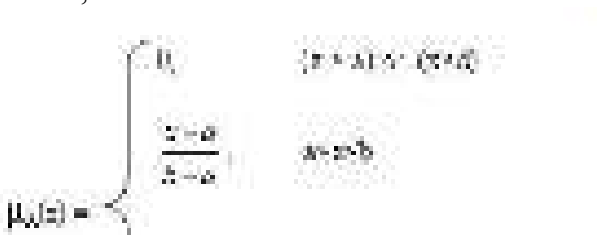


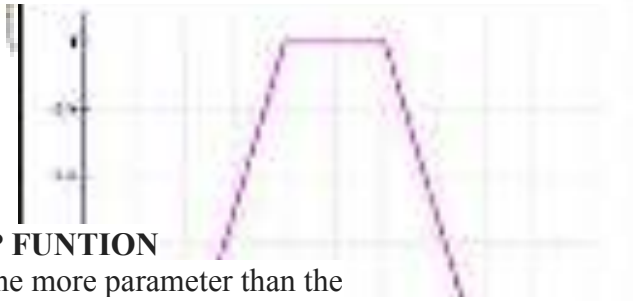
d. S SHAPED FUNCTION EQUATION

The S-shaped MF block implements an S-shaped membership function in Simulink®. Going from left to right the function increases from 0 to 1. The parameters a and b locate the left and right extremes of the sloped portion of the curve.

e. **Trapezoidal function**:

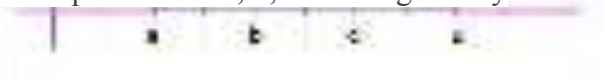
defined by a lower limit **a**, an upper limit **d**, a lower support limit **b**, and an upper support limit **c**, where **a < b < c < d**.





f. BELL SHAPED MEMBERSHIP FUNTION

The **bell membership function** has one more parameter than the Gaussian **membership function**, so it can approach a non-**fuzzy** set if the free parameter is tuned. The generalized bell function depends on three parameters *a*, *b*, and *c* as given by



7. Defuzzification

Defuzzification is the process of producing a quantifiable result in Crisp logic, given fuzzy sets and corresponding membership degrees. It is the process that maps a fuzzy set to a crisp set. It is typically needed in fuzzy control systems. These will have a number of rules that transform a number of variables into a fuzzy result, that is, the result is described in terms of membership in fuzzy sets. For example, rules designed to decide how much pressure to apply might result in "Decrease Pressure (15%), Maintain Pressure (34%), Increase Pressure (72%)". Defuzzification is interpreting the membership degrees of the fuzzy sets into a specific decision or real value.

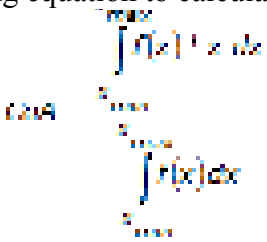
MEAN OF MAXIUMUM

Mean of Maximum. In the **Mean of Maximum** (MoM) **defuzzification** method, the fuzzy logic controller first identifies the scaled membership **function** with the greatest degree of membership.

When you use the **Mean of Maximum** (MoM) **defuzzification** method, you calculate the most plausible result. In other words, the fuzzy controller uses the typical value of the consequent term of the most valid rule as the crisp output value.

CENTER OF AREA (COA)

In the Center of Area (CoA) defuzzification method, also called the Center of Gravity (CoG) method, the fuzzy controller first calculates the area under the scaled membership functionsand within the range of the output variable. The fuzzy logic controller then uses the following equation to calculate the geometric center of this area.



8. FUZZY CONTROL SYSTEM

In a fuzzy logic controller (FLC), the dynamc bhavior of a fuzzy system is characteristized by a set of lizguistic description rules based on expert knowledge. The expert knowledge is usually of the form IF THEN.

Two I/P signal O/P (TISO) fuzzy system for example, in the case of two input – signal O/P fuzzy systems, control rules have then form.

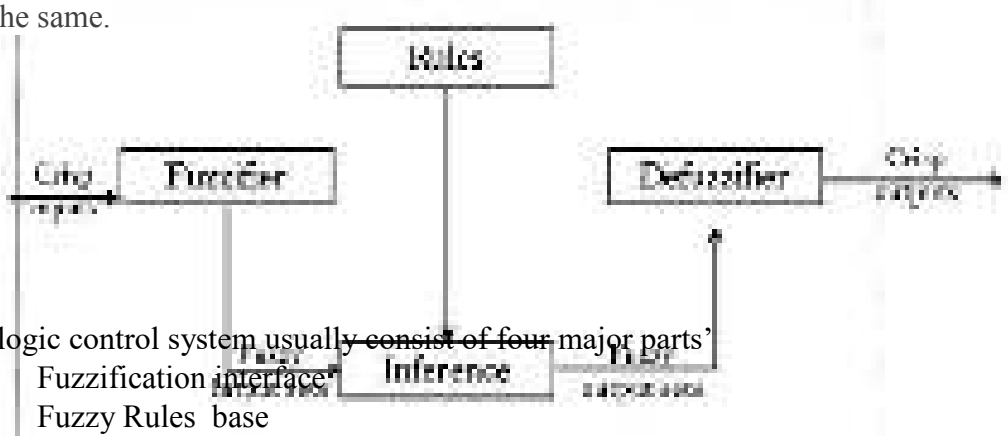
R_1 **:** if x is A_1 and y is B_1 then zis c_1
also
R_2 **:** if x is A_2 and y is B_2 then zis C_2
also
R_n **:** if x is A_n and y is B_n then zis C_n

Where x and y are the process state variables, z is control variable, A_i, B_i, and C_i are lingustic values of the lingustic varbles z,y and z in the universe of discourse U, V and W respectively.

MAMDANI TYPE OF FUZZY LOGIC CONTROL

The function is describe by a fuzzy rules base. It does not mean that FLC is a kind of transfer fucnction of different equation.This type of controller was suggested by Mamdani and Assilian in 1975 amd os called Mamdani type FLC.

Mamdani fuzzy inference is the most commonly seen fuzzy methodology and was among the first control systems built using fuzzy set theory. It was proposed in 1975 by Ebrahim Mamdani as an attempt to control a steam engine and boiler combination by synthesizing a set of linguistic control rules obtained from experienced human operators. Mamdani's effort was based on Lotfi Zadeh's 1973 paper on fuzzy algorithms for complex systems and decision processes . Although the inference process described in the next few sections differs somewhat from the methods described in the original paper, the basic idea is much the same.



Fuzzy logic control system usually ~~consist of four~~ major parts'

- Fuzzification interface
- Fuzzy Rules base
- Fuzzy inference mandine and
- Defuzification interface

9. REPRESENTIATION OF FUZZY SETS

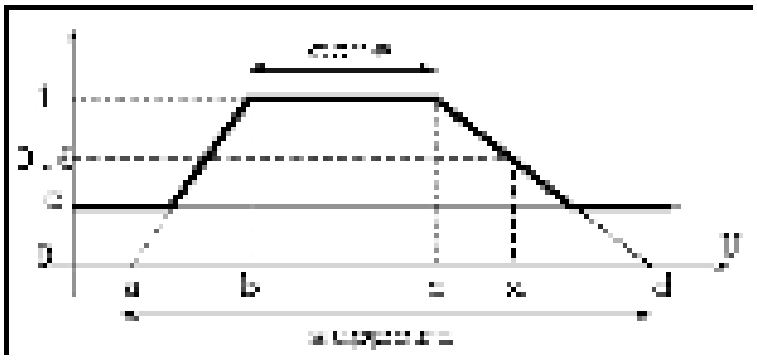
In our system, each *fuzzy set membership function*, denoted , is represented by a trapezoidal function with five parameters (*a*,*b*,*c*,*d*,*e*) corresponding to the weighted interval in the figure 2 (*e* represents the uncertainty). In the figure 2, which represents a fuzzy set *A*, implies that the element *x*does not characterize totally the fact represented by fuzzy set *A,* but with a degree of membership equals to 0.6 [8].

Moreover two characteristics of fuzzy sets are represented:

- the core of *A* defined by $Core(A) = \{x \in U / \mu_A(x) = 1\}$ and
- the support of *A* defined by $Supp(A) = \{x \in U / \mu_A(x) > 0\}$

However, in our system and algorithms, we use derived characteristics based on a *parametric definition* such as $Core^*(A) = [b, c]$ and $Supp^*(A) = [a, d]$ as shown in figure 2.

Figure



2: Representation of a fuzzy set *A*.

CLASSICAL SETS

Classical sets are sets with crisp boundaries. Usually an ordinary set (a classical or crisp set) is called a collection of objects which have some properties distinguishing them from other objects which do not possess these properties.

OPERATION OF CLASSICAL SETS

The **operations** that can be performed on the **classical sets** are dealt in detail below: Consider two **sets** A and B **defined** on the universe X. The definitions of the **operation** for **classical sets** are based on the two **sets** A and B **defined** on the universe X. A ∪ B = {x/x ∈ A or x ∈ B}.

There are various **operations** that can be performed in the **classical** or **crisp sets**. The results of the **operation** performed on the **classical sets** will be defi- nite. The**operations** that can be performed on the **classical sets** are dealt in **detail** below: Consider two **sets** A and B defined on the universe X.

There are various operations that can be performed in the classical or crisp sets. The results of the operation performed on the classical sets will be defi- nite. The operations that can be performed on the classical sets are dealt in detail below: Consider two sets A and B defined on the universe X. The definitions of the operation for classical sets are based on the two sets A and B defined on the universe X.

10. THE HEIGHT METHOD

The third defuzzification techniques is the height method, which can be viewed as a two step procedure. First, we convert the consequent membership function c_i into crisp consequent y = Ci where C_i is the center of gravity. The centroid defuzzification is then applied to the rules with crisp consequents.

Where $W_{I\ is}$ the degree to which the with matches the input data, the main benefit of the height method is its simplicity. The calculation of C_i can be performed during compilation.

The main disadvantage of this method is that it is not will justified and is offen considered an approximation to the centroid defuzzification.