## Study and Prepare document on following topics:

## Logic Gates

A logic gate is a physical device implementing a Boolean function, it performs a logical operation on one or more logical inputs, and produces a single logical output.

Depending on the context, the term may refer to an ideal logic gate, one that has for instance zero rise time and unlimited fan-out, or it may refer to a non-ideal physical device[1] (see Ideal and real op-amps for comparison).

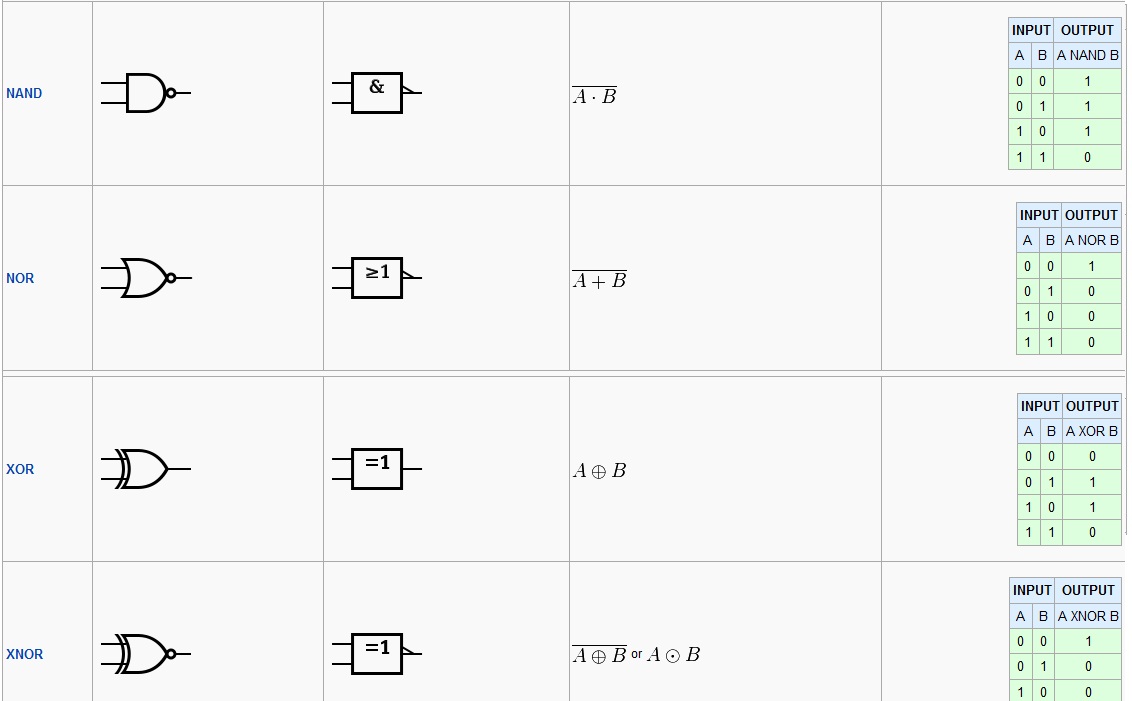
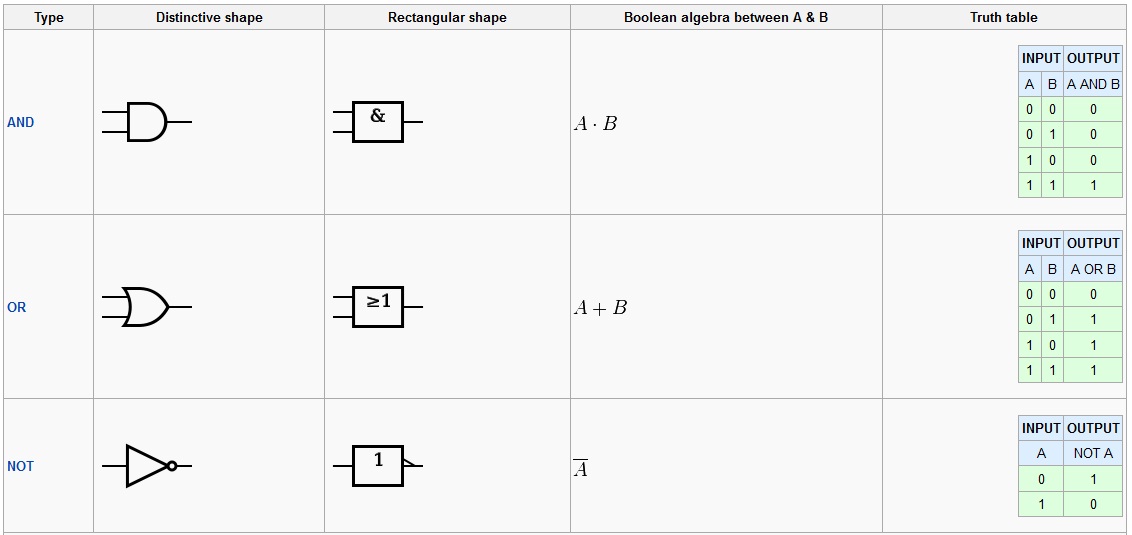
Logic gates are primarily implemented using diodes or transistors acting as electronic switches, but can also be constructed using electromagnetic relays (relay logic), fluidic logic, pneumatic logic, optics, molecules, or even mechanical elements. With amplification, logic gates can be cascaded in the same way that Boolean functions can be composed, allowing the construction of a physical model of all of Boolean logic, and therefore, all of the algorithms and mathematics that can be described with Boolean logic.

Logic circuits include such devices as multiplexers, registers, arithmetic logic units (ALUs), and computer memory, all the way up through complete microprocessors, which may contain more than 100 million gates. In practice, the gates are made from field-effect transistors (FETs), particularly MOSFETs (metal–oxide–semiconductor field-effect transistors).

Compound logic gates AND-OR-Invert (AOI) and OR-AND-Invert (OAI) are often employed in circuit design because their construction using MOSFETs is simpler and more efficient than the sum of the individual gates.[2]

Since the 1990s, most logic gates are made in CMOS technology (i.e. NMOS and PMOS transistors are used). Often millions of logic gates are packaged in a single integrated circuit.

There are several logic families with different characteristics (power consumption, speed, cost, size) such as: RDL (resistor-diode logic), RTL (resistor-transistor logic), DTL (diode-transistor logic), TTL (transistor-transistor logic) and CMOS (complementary metal oxide semiconductor). There are also sub-variants, e.g. standard CMOS logic vs. advanced types using still CMOS technology, but with some optimizations for avoiding loss of speed due to slower PMOS transistors.

Non-electronic implementations are varied, though few of them are used in practical applications. Many early electromechanical digital computers, such as the Harvard Mark I, were built from relay logic gates, using electro-mechanical relays. Logic gates can be made using pneumatic devices, such as the Sorteberg relay or mechanical logic gates, including on a molecular scale.[8] Logic gates have been made out of DNA (see DNA nanotechnology)[9] and used to create a computer called MAYA (see MAYA II). Logic gates can be made from quantum mechanical effects (though quantum computing usually diverges from boolean design). Photonic logic gates use non-linear optical effects.

A digital circuit which one or more input signals but only one output signal is called **Logical gates.**

AND - True if A and B are both True

OR - True if either A or B are True

NOT - Inverts value: True if input is False; False if input is True

XOR - True if either A or B are True, but False if both are True

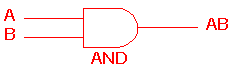
NAND - AND followed by NOT: False only if A and B are both True

NOR - OR followed by NOT: True only if A and B are both False

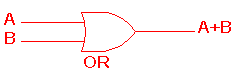
XNOR - XOR followed by NOT: True if A and B are both True or both False

**Basic Logic Gates**

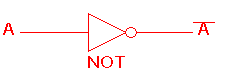
All digital systems can be constructed by only three basic logic gates. These basic gates are called the AND gate, the OR gate, and the NOT gate. Some textbooks also include the NAND gate, the NOR gate and the EOR gate as the members of the family of basic logic gates. The description of the operations of these gates are listed below [[Ref.2]](http://www.ied.edu.hk/has/phys/de/de-ref.htm):

**AND gate** 

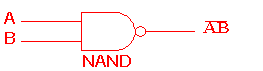
The AND gate is a circuit which gives a high output (logic 1) if all its inputs are high. A dot (.) is used to indicate the AND operation. In practice, however, the dot is *usually omitted*.

**OR gate** 

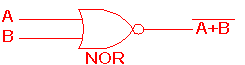
The OR gate is a circuit which gives a high output if one or more of its inputs are high. A plus sign (+) is used to indicate the OR operation.

**NOT gate** 

The NOT gate is a circuit which produces at its output the negated (inverted) version of its input logic. The circuit is also known as an *inverter*. If the input variable is A, the inverted output is written as .

**NAND gate** 

The NAND gate is a NOT-AND circuit which is equivalent to an AND circuit followed by a NOT circuit. The output of the NAND gate is high if any of its inputs is low.

**NOR gate** 

The NOR gate is a NOT-OR circuit which is equivalent to an OR circuit followed by a NOT circuit. The output of the NOR gate is low if any of its inputs is high.

**EOR gate** 

The Exclusive-OR gate is a circuit which gives a high output if either of its two inputs is high, but not both. A encircled plus sign () is used to indicate the EOR operation.

## 2. Adders

In electronics, an adder or summer is a digital circuit that performs addition of numbers. In many computers and other kinds of processors, adders are used not only in the arithmetic logic unit(s), but also in other parts of the processor, where they are used to calculate addresses, table indices, and similar operations.

Although adders can be constructed for many numerical representations, such as binary-coded decimal or excess-3, the most common adders operate on binary numbers. In cases where two's complement or ones' complement is being used to represent negative numbers, it is trivial to modify an adder into an adder–subtractor. Other signed number representations require a more complex adder.

**Half adder**

The half adder adds two single binary digits A and B. It has two outputs, sum (S) and carry (C). The carry signal represents an overflow into the next digit of a multi-digit addition. The value of the sum is 2C + S. The simplest half-adder design, pictured on the right, incorporates an XOR gate for S and an AND gate for C. With the addition of an OR gate to combine their carry outputs, two half adders can be combined to make a full adder.[1]

The half-adder adds two input bits and generate carry and sum which are the two outputs of half-adder.

**Full adder**

Schematic symbol for a 1-bit full adder with Cin and Cout drawn on sides of block to emphasize their use in a multi-bit adder

A full adder adds binary numbers and accounts for values carried in as well as out. A one-bit full adder adds three one-bit numbers, often written as A, B, and Cin; A and B are the operands, and Cin is a bit carried in from the next less significant stage.[2] The full-adder is usually a component in a cascade of adders, which add 8, 16, 32, etc. bit wide binary numbers. The circuit produces a two-bit output, output carry and sum typically represented by the signals Cout and S, where \mathrm{sum} = 2 \times C\_{out} + S. The one-bit full adder's truth table is:

Full-adder logic diagram

Inputs Outputs

A B Cin Cout S

0 0 0 0 0

1 0 0 0 1

0 1 0 0 1

1 1 0 1 0

0 0 1 0 1

1 0 1 1 0

0 1 1 1 0

1 1 1 1 1

A full adder can be implemented in many different ways such as with a custom transistor-level circuit or composed of other gates. One example implementation is with S = A \oplus B \oplus C\_{in} and C\_{out} = (A \cdot B) + (C\_{in} \cdot (A \oplus B)).

In this implementation, the final OR gate before the carry-out output may be replaced by an XOR gate without altering the resulting logic. Using only two types of gates is convenient if the circuit is being implemented using simple IC chips which contain only one gate type per chip. In this light, Cout can be implemented as C\_{out} = (A \cdot B) \oplus (C\_{in} \cdot (A \oplus B)).

A full adder can be constructed from two half adders by connecting A and B to the input of one half adder, connecting the sum from that to an input to the second adder, connecting Ci to the other input and OR the two carry outputs. Equivalently, S could be made the three-bit XOR of A, B, and Ci, and Cout could be made the three-bit majority function of A, B, and Cin.

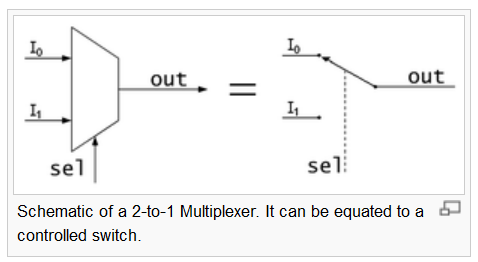
## 3. MUX and D-MUX

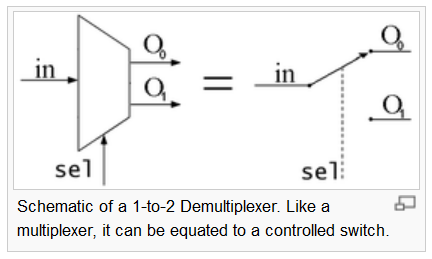
In electronics, a multiplexer (or mux) is a device that selects one of several analog or digital input signals and forwards the selected input into a single line.[1] A multiplexer of 2n inputs has n select lines, which are used to select which input line to send to the output.[2] Multiplexers are mainly used to increase the amount of data that can be sent over the network within a certain amount of time and bandwidth.[1] A multiplexer is also called a data selector.

An electronic multiplexer makes it possible for several signals to share one device or resource, for example one A/D converter or one communication line, instead of having one device per input signal.

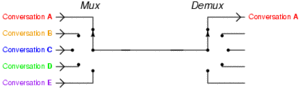
Conversely, a demultiplexer (or demux) is a device taking a single input signal and selecting one of many data-output-lines, which is connected to the single input. A multiplexer is often used with a complementary demultiplexer on the receiving end.[1]

An electronic multiplexer can be considered as a multiple-input, single-output switch, and a demultiplexer as a single-input, multiple-output switch.[3] The schematic symbol for a multiplexer is an isosceles trapezoid with the longer parallel side containing the input pins and the short parallel side containing the output pin.[4] The schematic on the right shows a 2-to-1 multiplexer on the left and an equivalent switch on the right. The sel wire connects the desired input to the output.





In analog circuit design, a multiplexer is a special type of analog switch that connects one signal selected from several inputs to a single output.



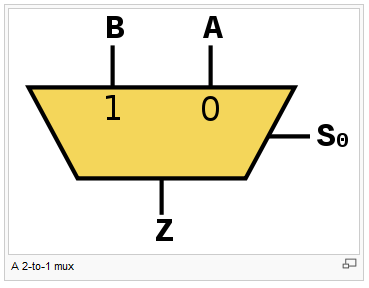
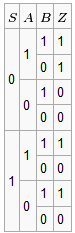
Digital multiplexers

In digital circuit design, the selector wires are of digital value. In the case of a 2-to-1 multiplexer, a logic value of 0 would connect I0 to the output while a logic value of 1 would connect I1 to the output. In larger multiplexers, the number of selector pins is equal to log 2 (n) where  n is the number of inputs.

For example, 9 to 16 inputs would require no fewer than 4 selector pins and 17 to 32 inputs would require no fewer than 5 selector pins. The binary value expressed on these selector pins determines the selected input pin.

A 2-to-1 multiplexer has a boolean equation where A and B are the two inputs, S is the selector input, and Z is the output:

Z=(A.)+(B.S)

useful in situations when cost is a factor and for modularity.

## 4. Encoders and Decoders

Decoder expansion

Combine two or more small decoders with enable inputs to form a larger decoder e.g. 3-to-8-line decoder constructed from two 2-to-4-line decoders.

Decoder with enable input can function as demultiplexer.

Decoder

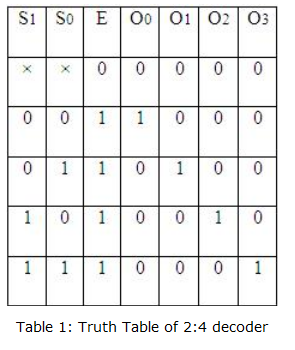
In digital electronics, a decoder can take the form of a multiple-input, multiple-output logic circuit that converts coded inputs into coded outputs, where the input and output codes are different e.g. n-to-2n , binary-coded decimal decoders. Decoding is necessary in applications such as data multiplexing, 7 segment display and memory address decoding.

The example decoder circuit would be an AND gate because the output of an AND gate is "High" (1) only when all its inputs are "High." Such output is called as "active High output". If instead of AND gate, the NAND gate is connected the output will be "Low" (0) only when all its inputs are "High". Such output is called as "active low output".

A slightly more complex decoder would be the n-to-2n type binary decoders. These types of decoders are combinational circuits that convert binary information from 'n' coded inputs to a maximum of 2n unique outputs. In case the 'n' bit coded information has unused bit combinations, the decoder may have less than 2n outputs. 2-to-4 decoder, 3-to-8 decoder or 4-to-16 decoder are other examples.

The input to a decoder is parallel binary number and it is used to detect the presence of a particular binary number at the input. The output indicates presence or absence of specific number at the decoder input.

Let us suppose that a logic network has 2 inputs A and B. They will give rise to 4 states A, A’, B, B’ . The truth table for this decoder is shown below:



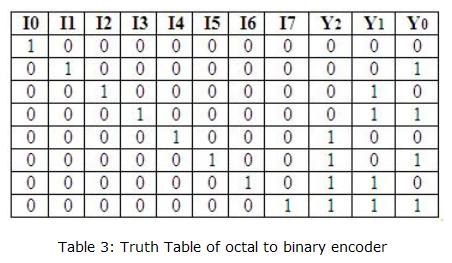
Encoder

An encoder is a device, circuit, transducer, software program, algorithm or person that converts information from one format or code to another. The purpose of encoder is standardization, speed, secrecy, security, or saving space by shrinking size. Encoders are combinational logic circuits and they are exactly opposite of decoders. They accept one or more inputs and generate a multibit output code.

Encoders perform exactly reverse operation than decoder. An encoder has M input and N output lines. Out of M input lines only one is activated at a time and produces equivalent code on output N lines. If a device output code has fewer bits than the input code has, the device is usually called an encoder.

Octal to binary encoder

Octal-to-Binary take 8 inputs and provides 3 outputs, thus doing the opposite of what the 3-to-8 decoder does. At any one time, only one input line has a value of 1. The figure below shows the truth table of an Octal-to-binary encoder.

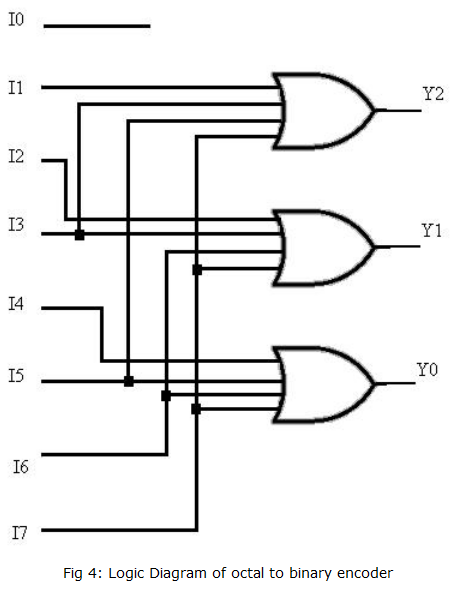


For an 8-to-3 binary encoder with inputs I0-I7 the logic expressions of the outputs Y0-Y2 are:

Y0 = I1 + I3 + I5 + I7

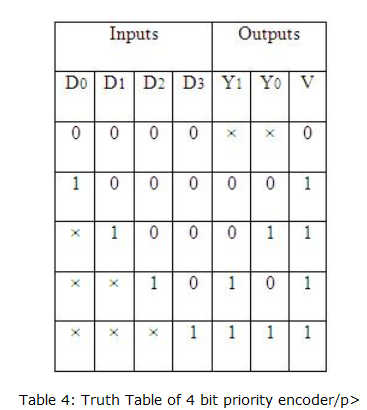
Y1= I2 + I3 + I6 + I7

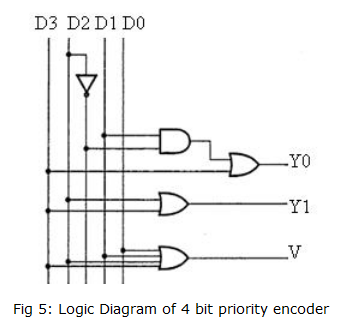
Y2 = I4 + I5 + I6 +I7



**Priority encoder**

A priority encoder is a circuit or algorithm that compresses multiple binary inputs into a smaller number of outputs. The output of a priority encoder is the binary representation of the ordinal number starting from zero of the most significant input bit. They are often used to control interrupt requests by acting on the highest priority request. It includes priority function. If 2 or more inputs are equal to 1 at the same time, the input having the highest priority will take precedence. Internal hardware will check this condition and priority is set.





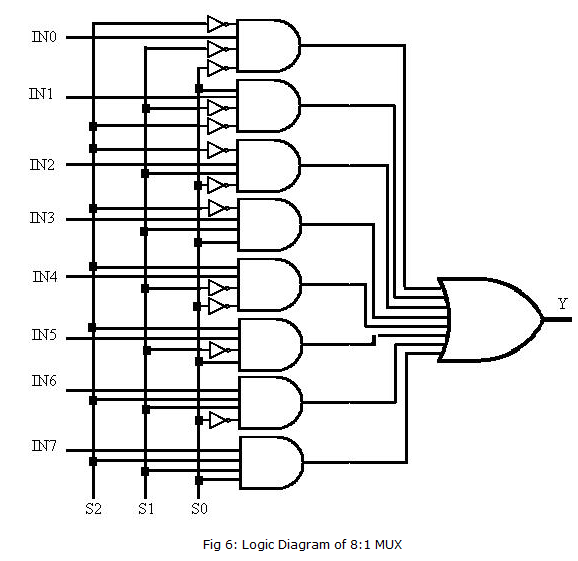
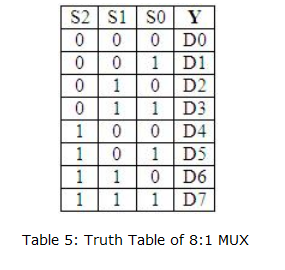
IC 74148 is an 8-input priority encoder. 74147 is 10:4 priority encoder

Multiplexer

In electronics, a multiplexer or mux is a device that selects one of several analog or digital input signals and forwards the selected input into a single line. A multiplexer of 2n inputs has n select lines, which are used to select which input line to send to the output. An electronic multiplexer can be considered as a multiple-input, single-output switch i.e. digitally controlled multi-position switch. The digital code applied at the select inputs determines which data inputs will be switched to output.

A common example of multiplexing or sharing occurs when several peripheral devices share a single transmission line or bus to communicate with computer. Each device in succession is allocated a brief time to send and receive data. At any given time, one and only one device is using the line. This is an example of time multiplexing since each device is given a specific time interval to use the line.

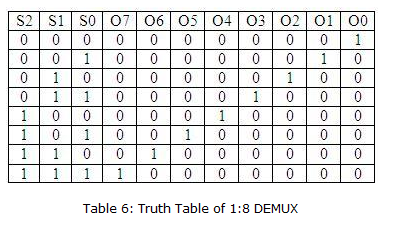
In frequency multiplexing, several devices share a common line by transmitting at different frequencies.

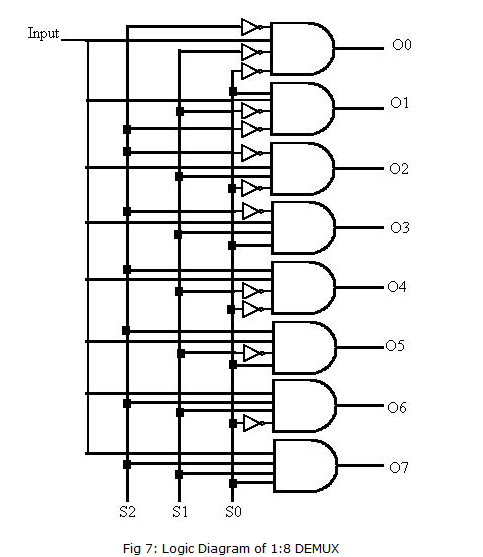


**Demultiplexer**

A demultiplexer (or demux) is a device taking a single input signal and selecting one of many data-output-lines, which is connected to the single input. A multiplexer is often used with a complementary demultiplexer on the receiving end. A demultiplexer is a single-input, multiple-output switch. Demultiplexers take one data input and a number of selection inputs, and they have several outputs. They forward the data input to one of the outputs depending on the values of the selection inputs.

Demultiplexers are sometimes convenient for designing general purpose logic, because if the demultiplexer's input is always true, the demultiplexer acts as a decoder. This means that any function of the selection bits can be constructed by logically OR-ing the correct set of outputs. Demultiplexer is called as a ‘distributro’, since it transmits the same data to different destinations.



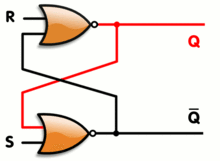


## 5. Flip Flops

In electronics, a flip-flop or latch is a circuit that has two stable states and can be used to store state information. A flip-flop is a bistable multivibrator. The circuit can be made to change state by signals applied to one or more control inputs and will have one or two outputs. It is the basic storage element in sequential logic. Flip-flops and latches are a fundamental building block of digital electronics systems used in computers, communications, and many other types of systems.

Flip-flops and latches are used as data storage elements. Such data storage can be used for storage of state, and such a circuit is described as sequential logic. When used in a finite-state machine, the output and next state depend not only on its current input, but also on its current state (and hence, previous inputs). It can also be used for counting of pulses, and for synchronizing variably-timed input signals to some reference timing signal.

Flip-flops can be either simple (transparent or opaque) or clocked (synchronous or edge-triggered); the simple ones are commonly called latches.[1] The word latch is mainly used for storage elements, while clocked devices are described as flip-flops.[2] A latch is level-sensitive, whereas a flip-flop is edge-sensitive. That is, when a latch is enabled it becomes transparent, while a flip flop's output only changes on a single type (positive going or negative going) of clock edge.



History

The first electronic flip-flop was invented in 1918 by William Eccles and F. W. Jordan.[3][4] It was initially called the Eccles–Jordan trigger circuit and consisted of two active elements (vacuum tubes).[5] Such circuits and their transistorized versions were common in computers even after the introduction of integrated circuits, though flip-flops made from logic gates are also common now.[6][7] Early flip-flops were known variously as trigger circuits or multivibrators.

According to P. L. Lindley, a JPL engineer, the flip-flop types discussed below (RS, D, T, JK) were first discussed in a 1954 UCLA course on computer design by Montgomery Phister, and then appeared in his book Logical Design of Digital Computers.[8][9] Lindley was at the time working at Hughes Aircraft under Dr. Eldred Nelson, who had coined the term JK for a flip-flop which changed states when both inputs were on. The other names were coined by Phister. They differ slightly from some of the definitions given below. Lindley explains that he heard the story of the JK flip-flop from Dr. Eldred Nelson, who is responsible for coining the term while working at Hughes Aircraft. Flip-flops in use at Hughes at the time were all of the type that came to be known as J-K. In designing a logical system, Dr. Nelson assigned letters to flip-flop inputs as follows: #1: A & B, #2: C & D, #3: E & F, #4: G & H, #5: J & K. Nelson used the notations "j-input" and "k-input" in a patent application filed in 1953.[10]

Implementation

Flip-flops can be either simple (transparent or asynchronous) or clocked (synchronous); the transparent ones are commonly called latches.[1] The word latch is mainly used for storage elements, while clocked devices are described as flip-flops.[2]

Simple flip-flops can be built around a pair of cross-coupled inverting elements: vacuum tubes, bipolar transistors, field effect transistors, inverters, and inverting logic gates have all been used in practical circuits. Clocked devices are specially designed for synchronous systems; such devices ignore their inputs except at the transition of a dedicated clock signal (known as clocking, pulsing, or strobing). Clocking causes the flip-flop to either change or retain its output signal based upon the values of the input signals at the transition. Some flip-flops change output on the rising edge of the clock, others on the falling edge.

Since the elementary amplifying stages are inverting, two stages can be connected in succession (as a cascade) to form the needed non-inverting amplifier. In this configuration, each amplifier may be considered as an active inverting feedback network for the other inverting amplifier. Thus the two stages are connected in a non-inverting loop although the circuit diagram is usually drawn as a symmetric cross-coupled pair (both the drawings are initially introduced in the Eccles–Jordan patent).

Flip-flop types

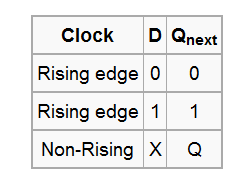
Flip-flops can be divided into common types: the **SR** ("set-reset"), **D** ("data" or "delay"[[11]](http://en.wikipedia.org/wiki/Flip-flop_%28electronics%29#cite_note-11)), **T** ("toggle"), and **JK** types are the common ones. The behavior of a particular type can be described by what is termed the characteristic equation, which derives the "next" (i.e., after the next clock pulse) output, Qnext in terms of the input signal(s) and/or the current output, Q

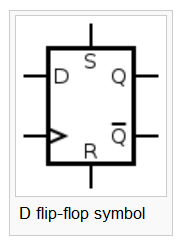
### D flip-flop

The D ﬂip-ﬂop is widely used. It is also known as a data or delay flip-flop.

The D flip-flop captures the value of the D-input at a definite portion of the clock cycle (such as the rising edge of the clock). That captured value becomes the Q output. At other times, the output Q does not change.[20][21] The D flip-flop can be viewed as a memory cell, a zero-order hold, or a delay line.[citation needed]

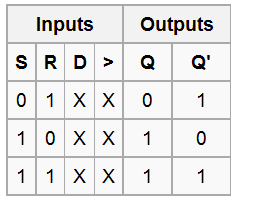
Truth table:





('X' denotes a Don't care condition, meaning the signal is irrelevant)

Most D-type flip-flops in ICs have the capability to be forced to the set or reset state (which ignores the D and clock inputs), much like an SR flip-flop. Usually, the illegal S = R = 1 condition is resolved in D-type flip-flops. By setting S = R = 0, the flip-flop can be used as described above. Here is the truth table for the others S and R possible configurations:



These flip-flops are very useful, as they form the basis for [shift registers](http://en.wikipedia.org/wiki/Shift_registers), which are an essential part of many electronic devices. The advantage of the D flip-flop over the D-type "transparent latch" is that the signal on the D input pin is captured the moment the flip-flop is clocked, and subsequent changes on the D input will be ignored until the next clock event. An exception is that some flip-flops have a "reset" signal input, which will reset Q (to zero), and may be either asynchronous or synchronous with the clock.

The above circuit shifts the contents of the register to the right, one bit position on each active transition of the clock. The input X is shifted into the leftmost bit position.

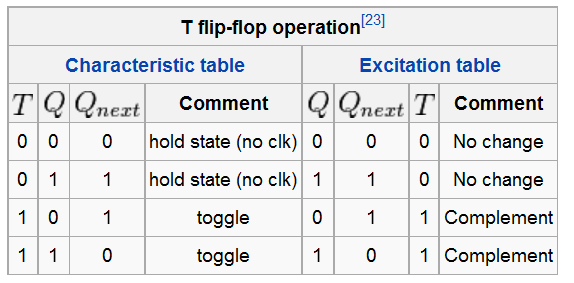
#### 

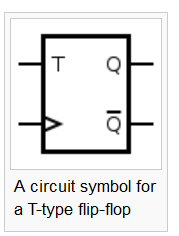
### T flip-flop

If the T input is high, the T flip-flop changes state ("toggles") whenever the clock input is strobed. If the T input is low, the flip-flop holds the previous value. This behavior is described by the characteristic equation:

 (expanding the [XOR](http://en.wikipedia.org/wiki/XOR_gate) operator)

and can be described in a [truth table](http://en.wikipedia.org/wiki/Truth_table):



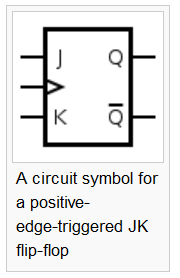


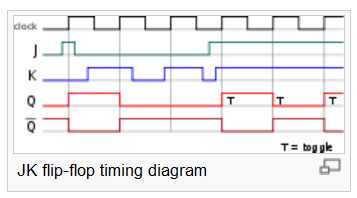
When T is held high, the toggle flip-flop divides the clock frequency by two; that is, if clock frequency is 4 MHz, the output frequency obtained from the flip-flop will be 2 MHz. This "divide by" feature has application in various types of digital [counters](http://en.wikipedia.org/wiki/Counter). A T flip-flop can also be built using a JK flip-flop (J & K pins are connected together and act as T) or D flip-flop (T input and Qprevious is connected to the D input through an XOR gate).

### JK flip-flop

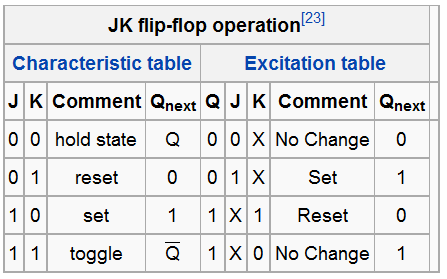
The JK flip-flop augments the behavior of the SR flip-flop (J=Set, K=Reset) by interpreting the S = R = 1 condition as a "flip" or toggle command. Specifically, the combination J = 1, K = 0 is a command to set the flip-flop; the combination J = 0, K = 1 is a command to reset the flip-flop; and the combination J = K = 1 is a command to toggle the flip-flop, i.e., change its output to the logical complement of its current value. Setting J = K = 0 does NOT result in a D flip-flop, but rather, will hold the current state. To synthesize a D flip-flop, simply set K equal to the complement of J. Similarly, to synthesize a T flip-flop, set K equal to J. The JK flip-flop is therefore a universal flip-flop, because it can be configured to work as an SR flip-flop, a D flip-flop, or a T flip-flop.

The characteristic equation of the JK flip-flop is: 



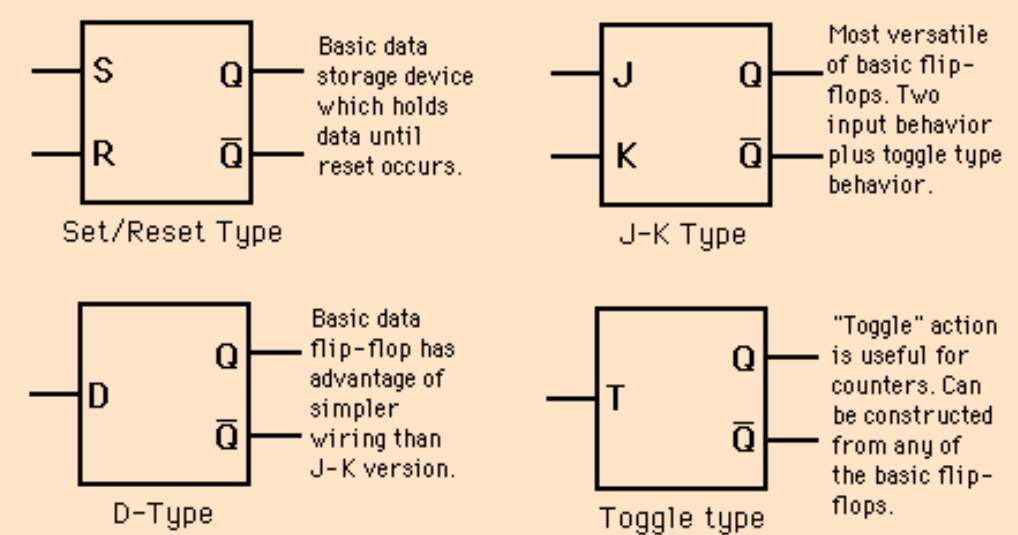


and the corresponding truth table is:



**Flip-Flops**

"Flip-flop" is the common name given to two-state devices which offer basic memory for sequential logic operations. Flip-flops are heavily used for digital data storage and transfer and are commonly used in banks called "registers" for the storage of binary numerical data.



## 6. Counters

In digital logic and computing, a counter is a device which stores (and sometimes displays) the number of times a particular event or process has occurred, often in relationship to a clock signal.

Electronic counters

In electronics, counters can be implemented quite easily using register-type circuits such as the flip-flop, and a wide variety of classifications exist:

Asynchronous (ripple) counter – changing state bits are used as clocks to subsequent state flip-flops

Synchronous counter – all state bits change under control of a single clock

Decade counter – counts through ten states per stage

Up/down counter – counts both up and down, under command of a control inputMontgomery

Ring counter – formed by a shift register with feedback connection in a ring

Johnson counter – a twisted ring counter

Cascaded counter

modulas counter.

Each is useful for different applications. Usually, counter circuits are digital in nature, and count in natural binary. Many types of counter circuits are available as digital building blocks, for example a number of chips in the 4000 series implement different counters.

Occasionally there are advantages to using a counting sequence other than the natural binary sequence—such as the binary coded decimal counter, a linear feedback shift register counter, or a Gray-code counter.

Counters are useful for digital clocks and timers, and in oven timers, VCR clocks, etc.[1]

Asynchronous (ripple) counter

An asynchronous (ripple) counter is a single [JK-type flip-flop](http://en.wikipedia.org/wiki/Flip-flop_%28electronics%29#JK_flip-flop), with its J (data) input fed from its own inverted output. This circuit can store one bit, and hence can count from zero to one before it overflows (starts over from 0). This counter will increment once for every clock cycle and takes two clock cycles to overflow, so every cycle it will alternate between a transition from 0 to 1 and a transition from 1 to 0. Notice that this creates a new clock with a 50% [duty cycle](http://en.wikipedia.org/wiki/Duty_cycle) at exactly half the frequency of the input clock. If this output is then used as the clock signal for a similarly arranged D flip-flop (remembering to invert the output to the input), one will get another 1 bit counter that counts half as fast. Putting them together yields a two-bit counter:

You can continue to add additional flip-flops, always inverting the output to its own input, and using the output from the previous flip-flop as the clock signal. The result is called a ripple counter, which can count to 2n − 1 where n is the number of bits (flip-flop stages) in the counter. Ripple counters suffer from unstable outputs as the overflows "Ripple" from stage to stage, but they do find frequent application as dividers for clock signals, where the instantaneous count is unimportant, but the division ratio overall is (to clarify this, a 1-bit counter is exactly equivalent to a divide by two circuit; the output frequency is exactly half that of the input when fed with a regular train of clock pulses).

The use of flip-flop outputs as clocks leads to timing skew between the count data bits, making this ripple technique incompatible with normal synchronous circuit design styles.

### Synchronous counter

A simple way of implementing the logic for each bit of an ascending counter (which is what is depicted in the image to the right) is for each bit to toggle when all of the less significant bits are at a logic high state. For example, bit 1 toggles when bit 0 is logic high; bit 2 toggles when both bit 1 and bit 0 are logic high; bit 3 toggles when bit 2, bit 1 and bit 0 are all high; and so on.

Synchronous counters can also be implemented with hardware [finite state machines](http://en.wikipedia.org/wiki/Finite_state_machine), which are more complex but allow for smoother, more stable transitions.

Hardware-based counters are of this type.

### Decade counter

A decade counter is one that counts in decimal digits, rather than binary. A decade counter may have each digit binary encoded (that is, it may count in binary-coded decimal, as the 7490 integrated circuit did) or other binary encodings (such as the bi-quinary encoding of the 7490 integrated circuit). Alternatively, it may have a "fully decoded" or one-hot output code in which each output goes high in turn (the 4017 is such a circuit). The latter type of circuit finds applications in multiplexers and demultiplexers, or wherever a scanning type of behavior is useful. Similar counters with different numbers of outputs are also common.

The decade counter is also known as a mod-counter when it counts to ten (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). A Mod Counter that counts to 64 stops at 63 because 0 counts as a valid digit.

### Up/down counter

A counter that can change state in either direction, under the control of an up or down selector input, is known as an up/down counter. When the selector is in the up state, the counter increments its value. When the selector is in the down state, the counter decrements the count.

### Ring counter

A ring counter is a circular shift register which is initiated such that only one of its flip-flops is the state one while others are in their zero states.

A ring counter is a Shift Register (a cascade connection of flip-flops) with the output of the last one connected to the input of the first, that is, in a ring. Typically, a pattern consisting of a single bit is circulated so the state repeats every n clock cycles if n flip-flops are used. It can be used as a cycle counter of n states.

### Johnson counter

A Johnson counter (or switchtail ring counter, twisted-ring counter, walking-ring counter, or Moebius counter) is a modified ring counter, where the output from the last stage is inverted and fed back as input to the first stage.[2][3][4] The register cycles through a sequence of bit-patterns, whose length is equal to twice the length of the shift register, continuing indefinitely. These counters find specialist applications, including those similar to the decade counter, digital-to-analog conversion, etc. They can be implemented easily using D- or JK-type flip-flops.

Computer science counters

In computability theory, a counter is considered a type of memory. A counter stores a single natural number (initially zero) and can be arbitrarily long. A counter is usually considered in conjunction with a finite-state machine (FSM), which can perform the following operations on the counter:

Check whether the counter is zero

Increment the counter by one.

Decrement the counter by one (if it's already zero, this leaves it unchanged).

The following machines are listed in order of power, with each one being strictly more powerful than the one below it:

Deterministic or non-deterministic FSM plus two counters

Non-deterministic FSM plus one stack

Non-deterministic FSM plus one counter

Deterministic FSM plus one counter

Deterministic or non-deterministic FSM.

For the first and last, it doesn't matter whether the FSM is a deterministic finite automaton or a nondeterministic finite automaton. They have power. The first two and the last one are levels of the Chomsky hierarchy.

The first machine, an FSM plus two counters, is equivalent in power to a Turing machine. See the article on counter machines for a proof.

Mechanical counters

Long before electronics became common, mechanical devices were used to count events. These are known as tally counters. They typically consist of a series of disks mounted on an axle, with the digits 0 through 9 marked on their edge. The right most disk moves one increment with each event. Each disk except the left-most has a protrusion that, after the completion of one revolution, moves the next disk to the left one increment. Such counters were originally used to control manufacturing processes, but were later used as odometers for bicycles and cars and in tape recorders and fuel dispensers. One of the largest manufacturers was the Veeder-Root company, and their name was often used for this type of counter.[5]

## 7. Registers

Registers

Registers are groups of flip-flops, where each flip-flop is capable of storing one bit of information. An n-bit register is a group of n flip-flops. The basic function of a register is to hold information in a digital system and make it available to the logic elements for the computing process. Registers consist of a finite number of flip-flops. Since each flip-flop is capable of storing either a "0" or a "1", there is a finite number of 0-1 combinations that can be stored into a register. Each of those combinations is known as state or content of the register. With flip-flops we can store data bitwise but usually data does not appear as single bits. Instead it is common to store data words of n bit with typical word lengths of 4, 8, 16, 32 or 64 bit. Thus, several flip-flops are combined to form a register to store whole data words. Registers are synchronous circuits thus all flip-flops are controlled by a common clock line. As registers are often used to collect serial data they are also called accumulators. There exist several types of registers as there are -

Shift Registers

Information often comes bitwise i.e. one bit at every clock pulse. Shift registers are used to store such data.

A shift register has one input. Every clock pulse one bit is

loaded into the first flip-flop of the register while all the actual flip-flop contents are shifted and the “oldest” bit got dropped. If the output of all flip-flops (and therefore the register’s complete content) are read from the lines Q1 to Qn the register is used as Serial In – Parallel Out (SIPO). A typical purpose for such a SIPO register is to collect data that is delivered bitwise and that is needed in n-bit data words (e.g. to convert the signals from serial ports of a computer: the line transports 1 bit a time, the computer uses 8, 16 or 32 bit data words). Shifting bits are important for mathematical operations: if the output of the whole register is interpreted as a binary number, shifting by one bit corresponds to multiplying or dividing by 2 (depends on which flip-flop is interpreted as MSB).

Cyclic Registers

Sometimes it is necessary to “recycle” the same values again and again. Thus the bit that usually would get dropped is fed to the register input again to receive a cyclic serial register

Parallel In-Serial Out

As there is a need for serial – parallel conversion the inverse operation is equally required. It is done by a Parallel In – Serial Out register (PISO) that allows loading data as whole data words and serial shifting. For this operation it needs two control lines: one to trigger the shifting and one to control when a new data word is loaded to the register . it is prepared by clearing allthe status of the flipflops output by using a clear function i.e. each flipflop is equals to 0 then shifting it by taking a single bit of the given input that input is shifts the present bits in the flipflop to the next flipflop

Barrel Shifters

A barrel shifter is a digital circuit that can shift a data word by a specified number of bits in one clock cycle. It can be implemented as a sequence of multiplexers (mux.), and in such an implementation the output of one mux is connected to the input of the next mux in a way that depends on the shift distance. For example, take a 4-bit barrel shifter, with inputs A, B, C and D. The shifter can cycle the order of the bits ABCD as DABC, CDAB, or BCDA; in this case, no bits are lost. That is, it can shift all of the outputs up to three positions to the right (and thus make any cyclic combination of A, B, C and D). The barrel shifter has a variety of applications, including being a useful component in microprocessors (alongside the ALU).

A common usage of a barrel shifter is in the hardware implementation of floating-point arithmetic. For a floating-point add or subtract operation, the significand of the two numbers must be aligned, which requires shifting the smaller number to the right, increasing its exponent, until it matches the exponent of the larger number. This is done by subtracting the exponents, and using the barrel shifter to shift the smaller number to the right by the difference, in one cycle. If a simple shifter were used, shifting by n bit positions would require n clock cycles.

Cascade Shifters

ISO standard

The ISO has defined standard ISO 12620 on Data Category Registry.[1] This is a registry for registering linguistic terms used in various fields of translation, computational linguistics and natural language processing and defining mappings both between different terms and the same terms used in different systems. The registers identified are:

bench-level register

dialect register

facetious register

formal register

in house register

ironic register

neutral register

slang register

taboo register

technical register

vulgar register

Diatype

The term diatype is sometimes used to describe language variation which is determined by its social purpose (Gregory 1967). In this formulation, language variation can be divided into two categories: dialect, for variation according to user, and diatype for variation according to use (e.g. the specialised language of an academic journal). This definition of diatype is very similar to those of register.

The distinction between dialect and diatype is not always clear; in some cases a language variety may be understood as both a dialect and a diatype.

Diatype is usually analysed in terms of field, the subject matter or setting; tenor, the participants and their relationships; and mode, the channel of communication, such as spoken, written or signed.

In computer architecture, a processor register is a small amount of storage available as part of a CPU or other digital processor. Such registers are (typically) addressed by mechanisms other than main memory and can be accessed more quickly. Almost all computers, load-store architecture or not, load data from a larger memory into registers where it is used for arithmetic, manipulated, or tested, by some machine instruction. Manipulated data is then often stored back in main memory, either by the same instruction or a subsequent one. Modern processors use either static or dynamic RAM as main memory, the latter often being implicitly accessed via one or more cache levels. A common property of computer programs is locality of reference: the same values are often accessed repeatedly and frequently used values held in registers improves performance. This is what makes fast registers (and caches) meaningful.

Processor registers are normally at the top of the memory hierarchy, and provide the fastest way to access data. The term normally refers only to the group of registers that are directly encoded as part of an instruction, as defined by the instruction set. However, modern high performance CPUs often have duplicates of these "architectural registers" in order to improve performance via register renaming, allowing parallel and speculative execution. Modern x86 is perhaps the most well known example of this technique.[1]

Allocating frequently used variables to registers can be critical to a program's performance. This register allocation is either performed by a compiler, in the code generation phase, or manually, by an assembly language programmer.

Categories of registers

Registers are normally measured by the number of bits they can hold, for example, an "8-bit register" or a "32-bit register". A processor often contains several kinds of registers, that can be classified accordingly to their content or instructions that operate on them:

User-accessible registers – The most common division of user-accessible registers is into data registers and address registers.

Data registers can hold numeric values such as integer and floating-point values, as well as characters, small bit arrays and other data. In some older and low end CPUs, a special data register, known as the accumulator, is used implicitly for many operations.

Address registers hold addresses and are used by instructions that indirectly access primary memory.

Some processors contain registers that may only be used to hold an address or only to hold numeric values (in some cases used as an index register whose value is added as an offset from some address); others allow registers to hold either kind of quantity. A wide variety of possible addressing modes, used to specify the effective address of an operand, exist.

The stack pointer is used to manage the run-time stack. Rarely, other data stacks are addressed by dedicated address registers, see stack machine.

Conditional registers hold truth values often used to determine whether some instruction should or should not be executed.

General purpose registers (GPRs) can store both data and addresses, i.e., they are combined Data/Address registers and rarely the register file is unified to include floating point as well.

Floating point registers (FPRs) store floating point numbers in many architectures.

Constant registers hold read-only values such as zero, one, or pi.

Vector registers hold data for vector processing done by SIMD instructions (Single Instruction, Multiple Data).

Special purpose registers (SPRs) hold program state; they usually include the program counter (aka instruction pointer) and status register (aka processor status word). The aforementioned stack pointer is sometimes also included in this group. Embedded microprocessors can also have registers corresponding to specialized hardware elements.

Instruction registers store the instruction currently being executed.

In some architectures, model-specific registers (also called machine-specific registers) store data and settings related to the processor itself. Because their meanings are attached to the design of a specific processor, they cannot be expected to remain standard between processor generations.

Control and status registers – There are three types: program counter, instruction registers and program status word (PSW).

Registers related to fetching information from RAM, a collection of storage registers located on separate chips from the CPU (unlike most of the above, these are generally not architectural registers):

Memory buffer register

Memory data register

Memory address register

Memory Type Range Registers (MTRR)

Hardware registers are similar, but occur outside CPUs.

Some examples

The table shows the number of registers of several mainstream architectures. Note that in x86-compatible processors the stack pointer (ESP) is counted as an integer register, even though there are a limited number of instructions that may be used to operate on its contents. Similar caveats apply to most architectures.

x86 FPUs have 8 80-bit stack levels in legacy mode, and at least 8 128-bit XMM registers in SSE modes.