Contents

Now that we know how individual gates work and how they are constructed, let's continue with examining how we combine gates to form circuits.

Circuits Recap:

Categories of Circuits

With the two general categories of circuits being:

1. Combinational Circuits where the input values explicitly determine the output
2. Sequential Circuits where the output is a function of the input values as well as the existing stage of the circuit
    a. Thus, sequential circuits usually involve the storage of information.

Description of the operations of circuits

The operation of these circuits can be described using three notations:

1. Boolean expressions
2. Logic diagrams and
3. Truth tables.

These notations are different but equally powerful, representative techniques

Boolean Algebra

When we reduce algebraic expressions using Boolean Algebra to decrease the number of gates needed to implement a circuit, we are decreasing the cost, latency issues and so on.

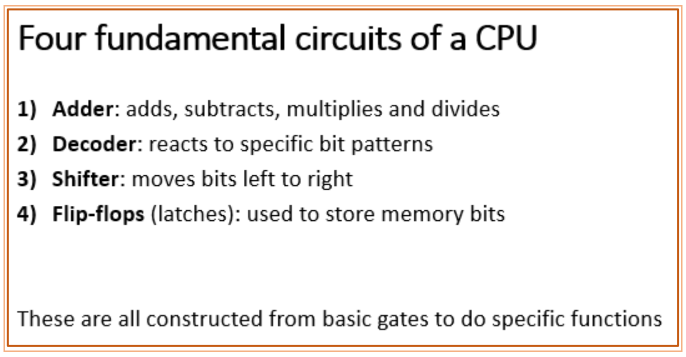
Expressions should be minimised by using:

1. De Morgans Law and other Boolean algebra properties, which allows us to apply provable mathematical principles to design logic circuits.
2. Alternatively the step-by-step approach of Karnaugh

The final result/output in the truth table of the original expression when compared to the truth table after reducing an expression should be ***identical***.

This demonstrated ***circuit equivalence***. *That is, both circuits produce exactly the same output for each input value combination.*

Four Fundamental Circuits of a CPU



Four fundamental circuits of a CPU

1) **Adder**: adds, subtracts, multiplies and divides
2) **Decoder**: reacts to specific bit patterns
3) **Shifter**: moves bits left to right
4) **Flip-flops** (latches): used to store memory bits

These are all constructed from basic gates to do specific functions

1) Adder

Perhaps the most basic operation a computer can perform is to add two numbers together through binary and these types of addition operations are carried out by a special kind of **combinational circuit** called **adders** as described previously

2) Decoder:

A ***decoder*** is a circuit that has one input and more than one output and can be used to change a code into a set of signals e.g. a memory address decoder which can ensure that data is read from the correct chip at the correct time by first determining which chip to use & then find the actual address on that specific chip

3) Shift Registers

This sequential device loads the data present on its inputs and then moves or “shifts” it to its output once every clock cycle, and is used for the storage or the transfer of binary data

End of recap of previous notes

4. Flip-flop (latches)

- A *latch* is an electronic logic *circuit* that has two inputs and one output. One of the inputs is called the SET input; the other is called the RESET input

Flip flops are an application of logic gates. They are a special form of *latch*. With the help of Boolean logic you can create memory with them. Flip flops can also be considered as the most basic idea of a Random Access Memory (RAM). When a certain input value is given to them, they will be remembered and executed, if the logic gates are designed correctly.

Many people use the terms *latch* and *flip-flop* interchangeably – technically a latch is level triggered and a flip-flop is edge triggered (on the clock cycle).

The first flip-flop was deigned (using vacuum tubes) in 1918 and were originally called *Eccles-Jordan trigger circuits*.

Flip-flops

- Are the basic technology behind computer memory chips, which is *volatile*.
- Are a sequential logic circuit in which their output is dependent on their input state

Unlike Combinational Logic circuits that change state depending upon the actual signals being applied to their inputs at that time, ***Sequential Logic circuits*** have some form of inherent "Memory" built in.

The output state of a sequential logic circuit is a function of the following three states:

1. the "present input",
2. the "past input" and/or
3. the "past output".

Sequential Logic circuits remember these conditions and stay fixed in their current state until the next clock signal changes one of the states, giving sequential logic circuits "Memory".

In general the flip flop are now constructed using bipolar junction transistors (if these BJTs were analysed you would notice that they give the circuit equivalence of the D type flip-flop detailed below)

> Recall, a junction transistor is "off" when there is no base current and switches to "on" when the base current flows. That means it takes an electric current to switch the transistor on or off.

To “remember” a past state, sequential circuits rely on a concept called ***feedback***.

Transistors like the BJT can be hooked up with logic gates so their output connections ***feed back into*** their inputs.

- The transistor then stays **on** even when the base current is removed. Each time a new base current flows, the transistor "flips" on or off.
- It remains in one of those stable states (either on or off) until another current comes along and flips it the other way.

This kind of arrangement is known as a **flip-flop** and it turns a transistor into a simple memory device that stores a **0** (when it's off) or a **1** (when it's on).

Flip-flops are the basic technology behind computer memory chips.

I. SR Flip-Flop

The **SR flip-flop**, also known as a *SR Latch*, can be considered as one of the most basic sequential logic circuit possible. This simple flip-flop is basically a one-bit memory bistable device that has two inputs, one which will "SET" the device (meaning the output = "1"), and is labelled **S** and one which will "RESET" the device (meaning the output = "0"), labelled **R**.

Then the SR description stands for "Set-Reset". The reset input resets the flip-flop back to its original state with an output Q that will be either at a logic level "1" or logic "0" depending upon this set/reset condition.

A basic NAND gate SR flip-flop circuit provides feedback from both of its outputs back to its opposing inputs and is commonly used in memory circuits to store a single data bit. Then the SR flip-flop actually has three inputs, Set, Reset and its current output Q relating to its current state or history.

The term "Flip-flop" relates to the actual operation of the device, as it can be "flipped" into one logic Set state or "flopped" back into the opposing logic Reset state.

The NAND Gate SR Flip-Flop

The simplest way to make any basic single bit set-reset SR flip-flop is to connect together a pair of cross-coupled 2-input NAND gates as shown, to form a Set-Reset bistable also known as an active LOW SR NAND Gate Latch, so that there is feedback from each output to one of the other NAND gate inputs. This device consists of two inputs, one called the *Set*, S and the other called the *Reset*, R with two corresponding outputs: Q and its inverse or complement $\overline{Q}$

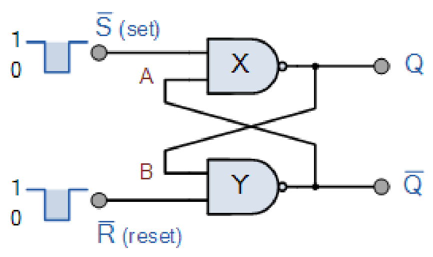
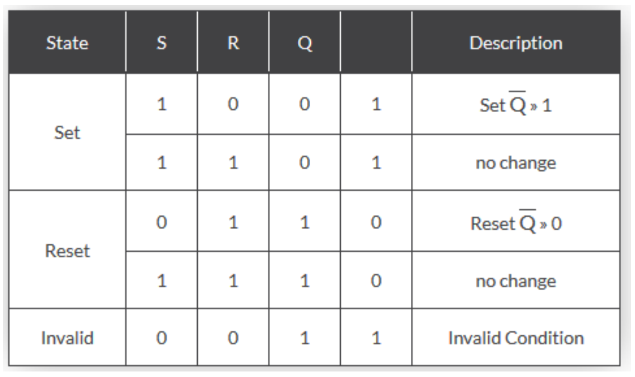


Figure 2 Circuit Diagram for an SR flip-flop



State	S	R	Q		Description
Set	1	0	0	1	Set $\overline{Q}$ » 1
	1	1	0	1	no change
Reset	0	1	1	0	Reset $\overline{Q}$ » 0
	1	1	1	0	no change
Invalid	0	0	1	1	Invalid Condition

Figure 1 Truth Table for the SR function

Consider the circuit shown above.

If the input R is at logic level “0” (R = 0) and input S is at logic level “1” (S = 1), the NAND gate *Y* has at least one of its inputs at logic “0” therefore, its output Q must be at a logic level “1” (NAND Gate principles). Output Q is also fed back to input “A” and so both inputs to NAND gate *X* are at logic level “1”, and therefore its output Q must be at logic level “0”.

Again NAND gate principals. If the reset input R changes state, and goes HIGH to logic “1” with S remaining HIGH also at logic level “1”, NAND gate *Y* inputs are now R = “1” and B = “0”. Since one of its inputs is still at logic level “0” the output at Q still remains HIGH at logic level “1” and there is no change of state. Therefore, the flip-flop circuit is said to be “Latched” or “Set” with Q = “1” and Q = “0”.

Reset State

In this second stable state, $\overline{\mathbf{Q}}$ is at logic level “0”, its inverse output at Q is at logic level “1”, (Q = “1”), and is given by R = “1” and S = “0”. As gate *X* has one of its inputs at logic “0” its output Q must equal logic level “1” (again NAND gate principles). Output Q is fed back to input “B”, so both inputs to NAND gate *Y* are at logic “1”, therefore, $\overline{\mathbf{Q}}$ = “0”.

If the set input, S now changes state to logic “1” with input R remaining at logic “1”, output Q still remains LOW at logic level “0” and there is no change of state. Therefore, the flip-flop circuits “Reset” state has also been latched and we can define this “set/reset” action in the following truth table.

Application of Flip-Flops in Computer Memory Chips

Random Access Memory (RAM) is volatile memory that sits next to the CPU. (Volatile meaning that it is wiped out when the computer is switched off). It is used to store instructions and data *currently* used by the CPU.

Main memory consists of a number of storage locations, each of which is identified by a unique address

The ability of the CPU to identify each location is known as its addressability

Each location stores a word i.e. the number of bits that can be processed by the CPU in a single operation. Word length may be typically 16, 24, 32, 64 bits….

A large word length improves system performance, though may be less efficient on occasions when the full word length is not used

Memory circuits (RAM) function by storing the voltage present on an input signal whenever they are triggered by a control signal, and they retain that stored voltage until the next assertion of the control (or trigger) signal and this is how they are therefore suitable for use to implement volatile memory

RAM consists of billions of **Data Cells**, each data cell being able to store one **bit** of information. For instance a 2GB RAM can store 2,000,000,000 Bytes of information = 16,000,000,000 bits of information and hence consists of 16,000,000,000 data cells.

Calculating the maximum capacity of memory in a computer

Max capacity = no. of addresses * capacity of each location

E.g. What is the max. amount of memory in a system which has a a 32 bit address bus?

ANSWER:

2^{32} [2^{32} refers to the number of available memory "Address" locations, each of which have 8 bits, so we can 'address' or 'access' 2^{32} different bytes]

=4,294,967,296/1024 [what are we referring to here? Bits/bytes/MB/KB?]

=4194304/1024 _B [why is it dividing by 1024?]

=4096/1024 _B

=4GB

This is the reason why 32-bit programs (or OS's) can only ever (and always will) be able to 'see' 4 GB of memory.

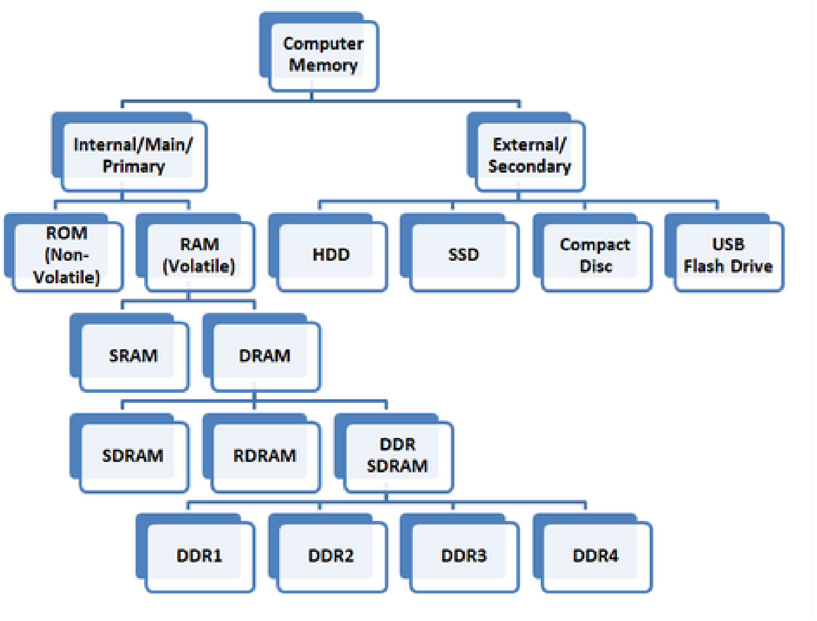
So, the total capacity of RAM in a 32-bit OS is 4GB

The width of the address bus determines the amount of memory a system can address. For example, a system with a **32-bit** address bus can address $\mathbf{2^{32}}$ (4,294,967,296) memory locations. If each memory location holds one byte, the addressable memory space is 4 GB.

32bit CPUs can address 2^{32} bytes

There are *many* types of RAM, two main types are:

1. Static RAM (SRAM)
2. Dynamic RAM (DRAM)



The memory cell is the fundamental building block of computer memory. The memory cell is an electronic circuit that stores one bit of binary information and it must be set to store a logic 1 (high voltage level) and reset to store a logic 0 (low voltage level). Its value is maintained/stored until it is changed by the set/reset process.

Memory operations require the following:

- Data – data written to/read from, memory as required by the operation
- Address – specifies the memory location to operate on

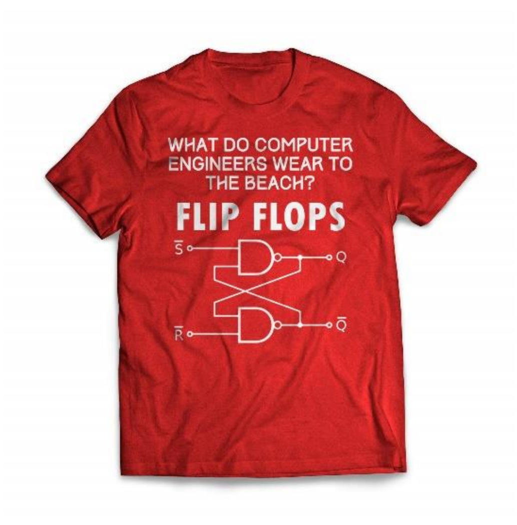
The *address bus* carry this info into memory (*n* bits specify locations of 2^n words)

In **SRAM**, the memory cell is a type of flip-flop circuit, usually implemented using FETs (Field Effect Transistor) whose output fields are controlled by a variable electric field). This means that SRAM requires very low power when not being accessed, it is expensive, has low storage density, several transistors required for each bit cell.

DRAM, is based around a capacitor. Charging and discharging this capacitor can store a "1" or a "0" in the cell. However, the charge in this capacitor slowly leaks away, and must be refreshed periodically. Because of this refresh process, DRAM uses more power, but it can achieve greater storage densities and lower unit costs compared to SRAM. With DRAM there is one transistor and one capacitor per cell.

II. D-Type Flip-Flop Circuits

The D-Type Flip-Flop (DFF) Circuits is also called D-Type Latch (as it's made from latches)



The DFF is widely used in all sorts of modern digital circuits.

A DFF uses a basic cell for a memory element, but it only allows the value stored in memory to be changed (or "programmed") when a timing control input is asserted.

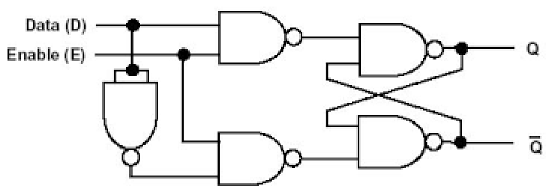
Thus, a DFF has two inputs

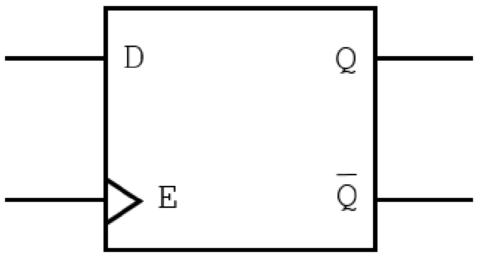
1. the timing control input and
2. a data input.

The timing control input, commonly called "gate", or "clock", or "latch enable", is used to coordinate when new data can be written into the memory element, and conversely, when data cannot be written (see Clock Signal and Delaying Effect below).

- A clock determines ***when*** computational activities occur
- Signals determines ***what*** computational activities occur

Each data cell consists of a **D-Type Flip-Flop circuit** that is usually built using **NAND logic gates** connected as follows (***recall*** the use of a NAND gate as an inverter):





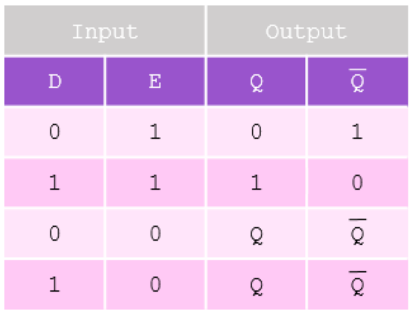
A D-Type Flip-Flop Circuit is used to store ***1 bit*** of information.

It has two input pins **D** (Data) and **E** (Enabler) and two output pins (Q and $\overline{Q}$).

Q represents the current state of the circuit.

When enabler **E**=**1**, output **Q** can be set to Data input **D**.

If a 1 is asserted on D & the clock is pulsed, output Q=1



Input		Output	
D	E	Q	$\overline{Q}$
0	1	0	1
1	1	1	0
0	0	Q	$\overline{Q}$
1	0	Q	$\overline{Q}$

When enabler **E**=**0**, the output **Q** cannot be changed.

If a 0 is asserted on D & the clock is pulsed, output Q=0

Therefore an output of 1 means the circuit is currently “storing” a value of 1

This flip-flop is sometimes called a transparent latch, because while Enable is high(E=1), the data outputs follow the data input(Q=D).

≡When Clock pulse is Low (not high) then there is no change in output.

Because the latch holds the data forever while the enable input is low, it can be used to store information, for example, in a computer memory.

Lots of latches plus a big decoder equals one computer memory.

And so, this is why this circuit is used to create memory cells (e.g in the RAM).

A circuit may use *many* flip-flops; together they define the circuit state

This D-type flip flop is a variant of the more basic S-R flip-flop (**S**et-**R**eset) which has the following truth table, with it’s obvious problem being when S=1 and R=1:

S = 1 & R = 0: set
S = 0 & R = 0: hold
S = 0 & R = 1: reset
S = 1 & R = 1: not allowed

“not allowed” because when S=1 and R=1 (maybe when a user presses two buttons together) simultaneously, we don’t know what value Q will take. The D-latch eliminates this undefined state – it is active only at clock transitions.

Clock Signal and Delaying Effect

The enabler input E is often connected to another circuit called the **clock** (e.g. CPU clock). The clock signal constantly and regularly alternate between two states: 0 and 1, similar to a heartbeat. Inside the CPU the clock signal controls the execution of the Fetch-Execute cycle.

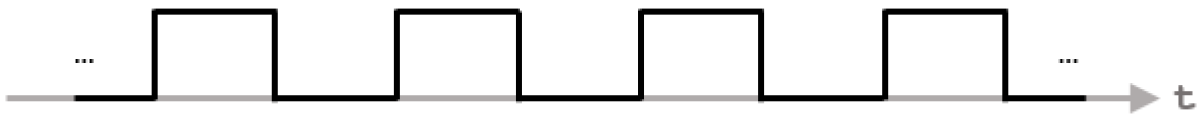
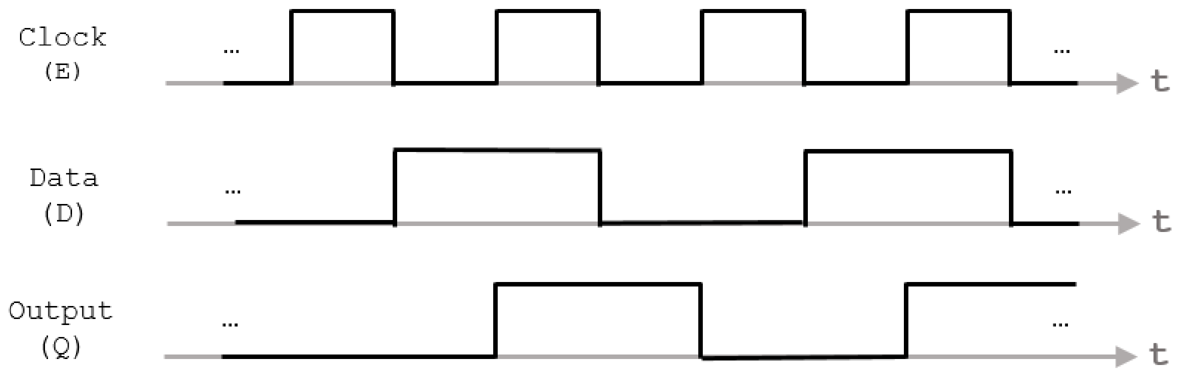


Figure 3 The Clock Signal

When the clock signal is applied to the Enabler input (in this case also called the clock input), the flip-flop output Q can only change values when triggered by the clock signal. The value of the flip-flop is held or **delayed** until the next clock signal. This delaying effect is also called a **latch**.

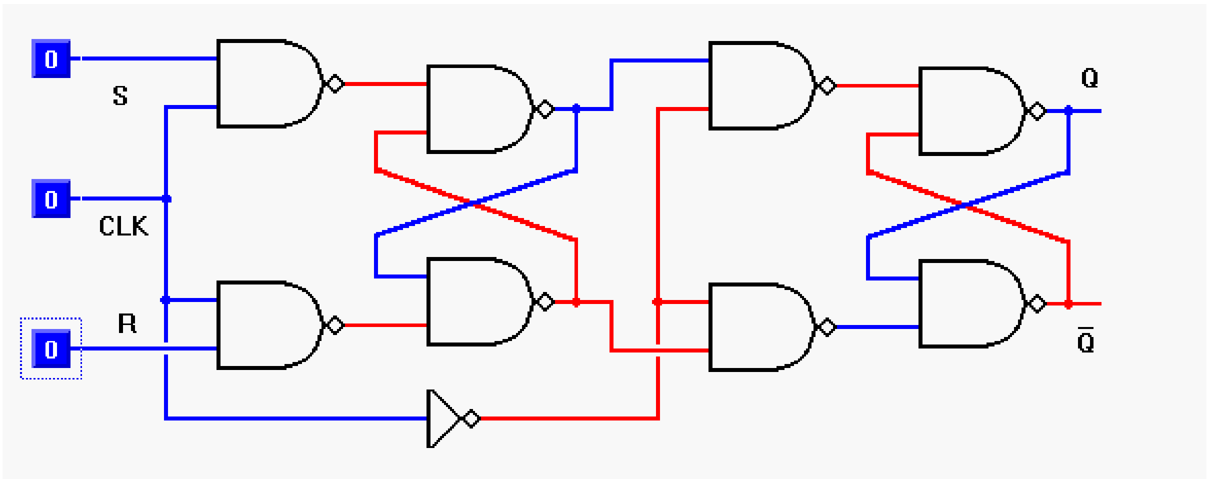
In other words, the change of input (D) is ***not applied immediately*** to the output Q but is applied at the next "tick of the clock". There are many applications to this delay such as the ability to create a frequency divider (divide the clock frequency by a multiple of 2).



The D Flip Flop is by far the most important of the clocked flip-flops as it ensures that ensures that inputs S and R are never equal to one at the same time

III. Edge-triggered latch (master-slave structure)

This in effect is two identical clocked D-type circuits connected together, with each one operating on opposite halves of the clock signal.



the inverter connected between the two CLK inputs ensures that the two sections will be enabled during opposite half-cycles of the clock signal.

This is the key to the operation of this circuit.

There is still one problem left to solve: the possible race condition which may occur if both the S and R inputs are at logic 1 when CLK falls from logic 1 to logic 0. In the example above, we assume that the race will end with the master latch in a random state, but this will not be certain with real components. Often, one gate in the output latch will react slightly faster than the other, but we can never be sure which one. Therefore, we need to have a way to prevent race conditions from occurring at all. That way we won't have to figure out which gate in the circuit won the race on any particular occasion.

The solution is to add some additional feedback from the slave latch to the master latch. The resulting circuit is called a *JK flip-flop*.

The only time new data can be stored in the circuit, is during the brief moment when the enable input goes from low to high. This transition is referred to as a rising clock edge, and so this tandem latch configuration is called a rising edge triggered latch

The most important use of the edge triggered flip-flop is in the data register. The characteristic of loading data only at the rising edge of the clock pulse is critical in creating the computer machine cycle - A register is simply a collection of edge triggered flip-flops

The computer clock provides a continuous stream of pulses to the computer processor. With each upward swing of the voltage on the clock output, the register loads whatever values are present on its inputs.

As soon as the data is loaded it appears on the outputs, and stays there steady throughout the rest of the clock cycle. The data on the register outputs is used as input for the computer circuits that carry out the operations desired.

For example, the accumulator register data output may be directed by a multiplexor to one of the inputs of the ALU, where it serves as an addend. The result of the addition appears on the ALU outputs, which can be directed by multiplexors to the inputs of the same accumulator register. The result waits at the inputs of the register for the next upward swing of voltage on the CP register input, sometimes called the write input (WR). When this occurs, the ALU result is loaded into the register and displayed on the outputs. The value held in the register from the previous cycle is overwritten and discarded, having served its purpose. The cycle is repeated endlessly, the data following paths dictated the program instructions, which control the various multiplexors and registers. If the accumulator had used transparent latches, the ALU result would feed through the register, to the ALU, back to the register and through again to the computer circuits in an uncontrolled loop.

The edge-triggered register ensures that the result is written in an instant, and then the inputs close, allowing the computer to perform its operations in an orderly and stable way.

IV. The JK Flip Flop

The JK Flip Flop is used to use the **“Not Allowed”** state in basic S-R Flip Flop to become **“Toggle current state”** , while the D-Flip Flop is used to have one input only for data transfer and storage.

Arranging these together, you get “Memory” with *n Bits* where n is the number of the flip flops used, also they are used in data communication and counters for example, to store the states of data.

Flip-Flop Symbols

Illustrated shown below are some typical flip-flop symbols, using a “black-box approach”, an approach that is useful to represent specific clusters of electronic components:

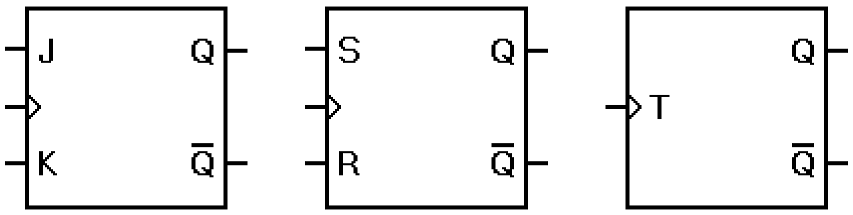


Figure 4 Flip Flop Symbols

The symbols above are nearly identical — only the inputs vary. This is typical of the "black-box" approach.

It is very seldom that a flip-flop will actually be used alone. Such circuits are far more useful when grouped together and acting in concert. There are two general ways in which flip-flops may be interconnected to perform useful functions: *counters* and *registers*

Flip-Flip Applications:

Static RAM

In static RAM, a form of flip-flop holds each bit of memory. A flip-flop for a memory cell takes 4 or 6 transistors along with some wiring, but never has to be refreshed. This makes static RAM significantly faster than dynamic RAM. However, because it has more parts, a static memory cell takes a lot more space on a chip than a dynamic memory cell. Therefore you get less memory per chip, and that makes static RAM a lot more expensive.

So static RAM is fast and expensive, and dynamic RAM is less expensive and slower. Therefore static RAM is used to create the CPU's speed-sensitive cache, while dynamic RAM forms the larger system RAM space.

Serial-to-Parallel Shift Register

Registers refer to a group of flip-flops operating as a coherent unit to hold data.

The circuit below is known as a *shift register* because data is shifted through it, from flip-flop to flip-flop.

If you apply one *byte* (8 bits) of data to the initial data input one bit at a time, and apply one clock pulse to the circuit after setting each bit of data, the entire byte present at the flip-flop outputs in parallel format.

Therefore, this circuit is known as a serial-in, parallel-out shift register. It is also known sometimes as a shift-in register, or as a serial-to-parallel shift register.

By standardised convention, the least significant bit (LSB) of the byte is shifted in first.

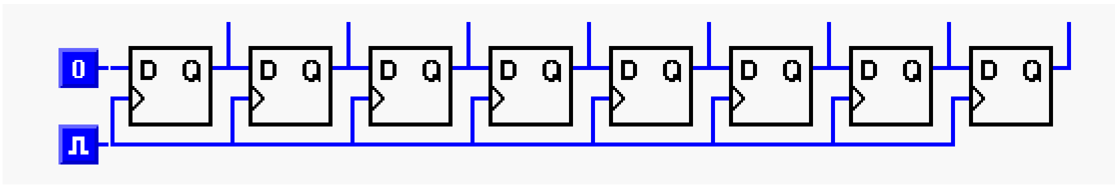


Figure 5 S-to-P Shift Register