

Module 5 – Flip flops, counters and shift registers

Overview

Combinational logic systems, no matter how complex, produce outputs that are strict functions of the inputs. Flip-flops, counters and shift registers, on the other hand, respond not only to the inputs but also the current state of the device. That is, the system has memory. So the output is dependent on the 'state' of the system. These systems are classified as sequential logic systems, as shown in the following concept map.

Sequential logic systems (concept map)

Objectives

At the completion of this module you will be able to:

- describe the use of basic logic gates in developing the set/reset (SR) bistable or flip flop
- explain the use of graphical timing diagrams to describe the operation of basic logic systems
- recognise and describe the operation of typical flip flops including D, MS, T and JK systems
- outline the principles of digital counters
- develop and design typical serial and parallel counting systems
- list the problems associated with propagation delay and stability in counting systems
- describe the principles of feedback in counting systems and its use in shift register development.

5.1 The bistable or flip flop

In counting and sequential networks it is necessary to provide some form of **memory element** to record the state of a problem at any moment in time.

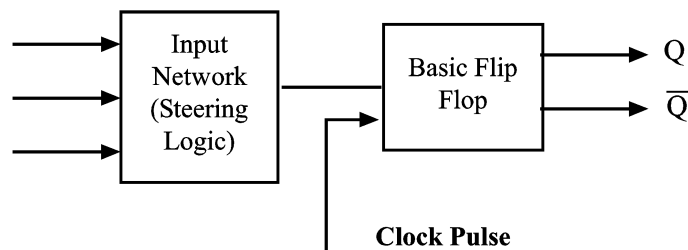
The most commonly used device is known as the **flip flop** or **bistable** element as it has 2 stable states. These correspond to logic levels 0 and 1, the output of the element changing from one state to the other on demand.

A flip flop therefore may be defined as a device which stores binary information in the form of a 0 or a 1 state and can be maintained indefinitely in either of those states provided it is continually supplied with power. It can also be easily changed from one state to the other.

There are a large number of different types of flip flop. The flip flops most commonly used have names like: SR flip flop, (set-reset); D flip flop; T flip flop and JK flip flop. We will look at the characteristics of these flip flops.

5.2 General principles of flip flops

A block diagram of a typical flip flop is shown:

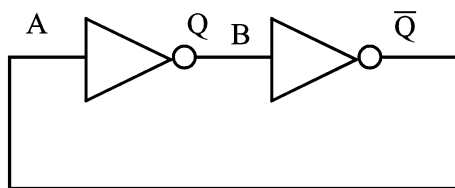


The input signals are combined in the input logic network to give one or more outputs to the flip flop. There may also be another input to the FF known as a **clock pulse**. This is sometimes used to ensure the operation of the FF occurs at a particular time. The outputs from the FF are Q and its complement \bar{Q} .

Where a clock pulse is **not** used, the system is said to be **asynchronous** since the input to the FF may be dependant on the output of some previous network or even flip flop. Where a clock pulse is used, the system is usually called **synchronous**.

5.3 The basic bistable construction

A basic bistable could be constructed from two inverters as shown:



Let us **assume** some initial conditions and mark them on the diagram.

Let $A = 1$, then $Q = 0 = B$, so $\bar{Q} = 1$ which is correct for A .

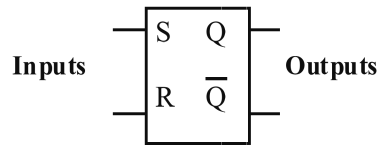
Similarly, if $A = 0$ the other state holds.

- This basic system has no way of changing from one state to the other, but it does have 2 **stable** conditions or states.
- A practical circuit is made from transistors used as switches and signals can be injected into their bases to force a switch to change from one state to the other.

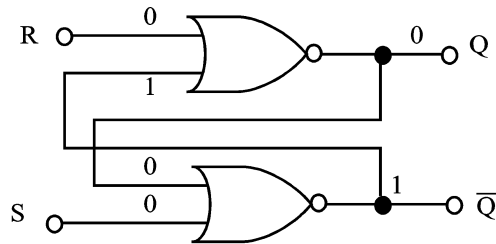
These inputs are called the **set** (S) input and the **reset** (R) input.

The basic system is then called a **(set-reset)** or **(S-R)** flip flop and is manufactured commercially.

Its symbol is:



- This system can be made from a pair of cross-connected NOR gates.



Cross-connected NOR Gates

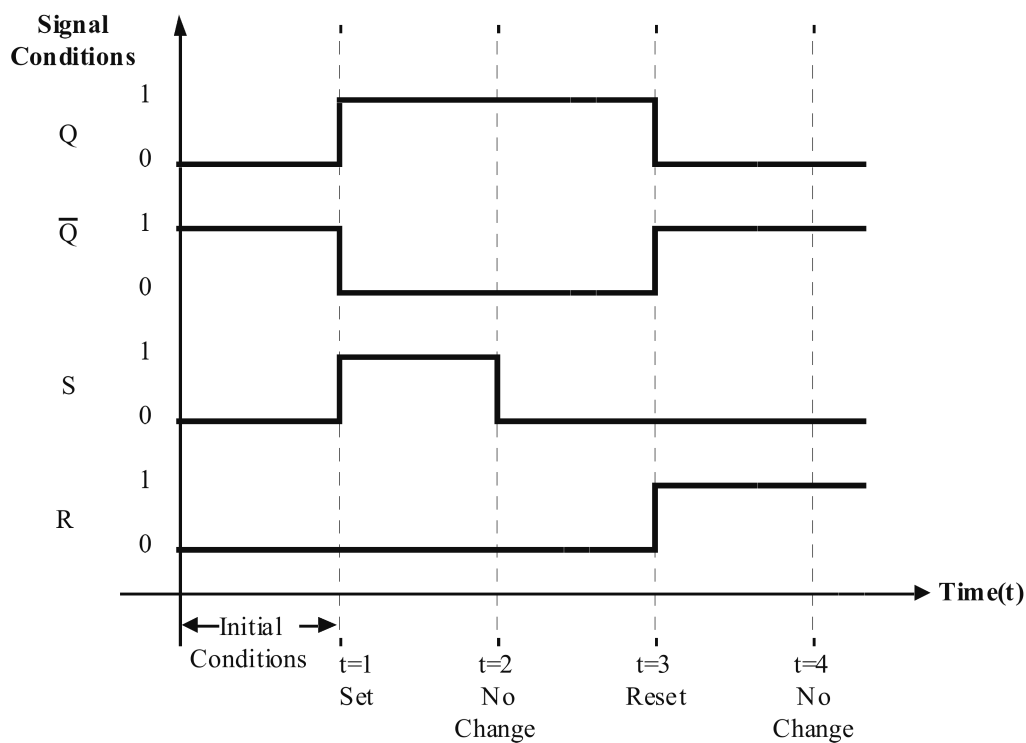
| A | B | F |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

Truth Table for a NOR Gate

To consider the operation of this system it is again necessary to assume some initial conditions.

Let $S = 0$, $R = 0$, $Q = 0$, and $\bar{Q} = 1$.

Quite often in analysing the operation of logic systems a **timing diagram** is used. This is a graphical representation of the systems operation with time. It is a waveform display similar to that which is viewed on the screen of a cathode ray oscilloscope.



Timing Diagram

When drawing this diagram, start with the initial conditions and change one input only at any one time, such as $t = 1$ on the diagram. It is not normal to change the input conditions simultaneously nor is it normal to set and reset at the same time. Therefore we avoid the condition $S = R = 1$.

It is often helpful to mark the chosen conditions on the flip flop diagram to ensure that such **assumed** conditions can exist. Consult the appropriate truth table to verify this.

Basic NOR gate S-R flip-flop truth table.

| S | R | Q | \bar{Q} | Comment |
|---|---|---|-----------|----------------------------------|
| 1 | 0 | 1 | 0 | Q is Set to logic 1 |
| 0 | 0 | 1 | 0 | Q remains Set (after S=1, R=0) |
| 0 | 1 | 0 | 1 | Q is Reset to logic 0 |
| 0 | 0 | 0 | 1 | Q remains Reset (after S=0, R=1) |
| 1 | 1 | 0 | 0 | undefined, thus avoided |

Examination of the timing diagram shows us that this flip flop will only change state **whenever S or R goes from a logic zero to a logic one condition**. It ignores one to zero transitions.



Activity 5.1

The same flip flop can be constructed of NAND gates. As an exercise you should now draw the logic diagram, assume initial conditions and sketch the timing diagram for this system.

You should discover that this flip flop will only change state whenever S or R goes from a one to a zero condition. The solution to this exercise is given at the end of this module.

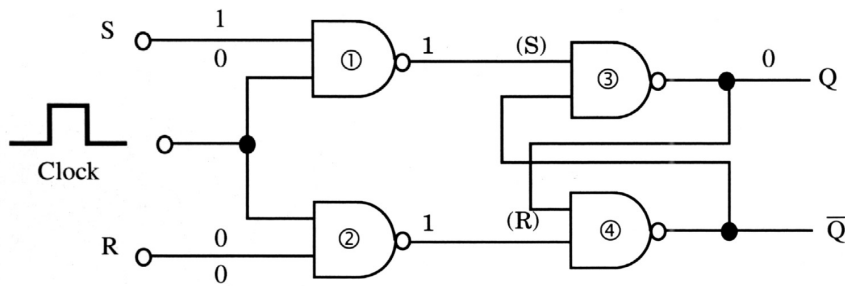
5.4 Clocked flip flops

The flip flop is often used as a temporary storage element. That is, it stores a logic 1 at its output. In many applications a stream of data may be being generated during a process, but it is desirable to only store the final result.

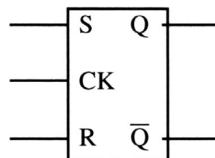
A typical example would be in computer multiplication or a digital counter where only the final result is needed, **not** all the intermediate steps.

The storage element then is **isolated** from the output during the calculating phase and finally **enabled** by a timing pulse at the end of calculations in time to receive the result.

This timing pulse is often called a '**clock pulse**' or '**strobe**' or '**gate pulse**'. It is usually a pulse of short duration (in time) as shown in the diagram. We can draw a **clocked SR-NAND flip flop**.



The symbol is:

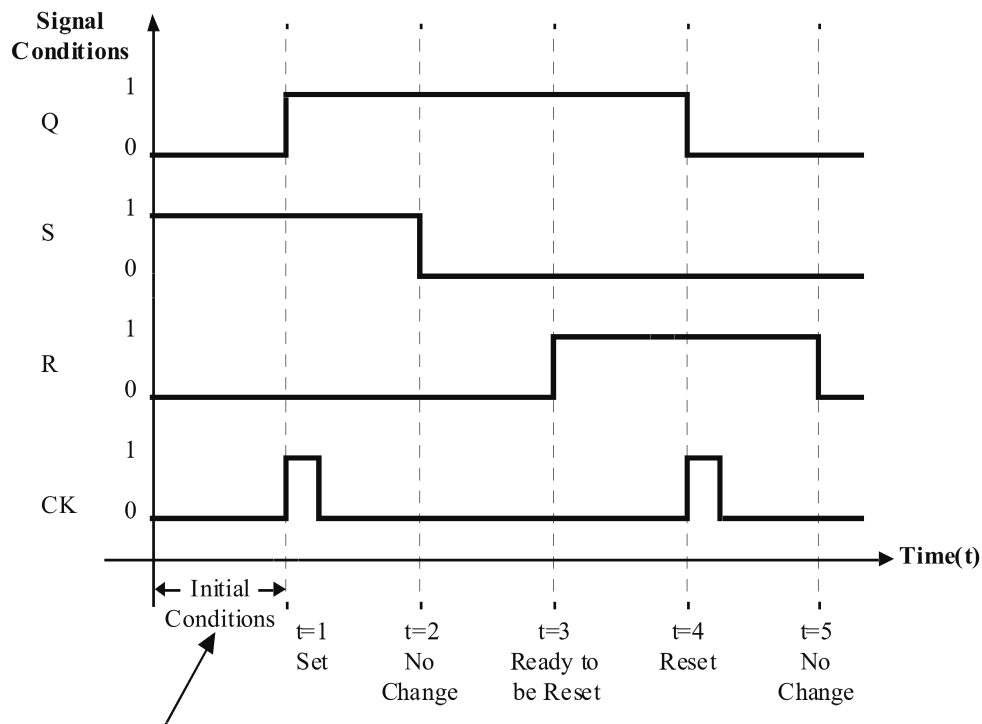


To study its operation using a timing diagram,

Let $Q = 0$, $S = 1$, $R = 0$, $CK = 0$

Note: The necessary set or reset conditions are established on gates ① and ② **but** are only

transmitted to gates and on receipt of the clock pulse. The clock pulse is said to enable gates ① and ②.



Initially $S = 1$ and the FF is ready to be set but prevented from doing so by the clock.

When the clock is applied at $t = 1$, the FF will set and $Q = 1$.

At $t = 2$, the set line is returned to zero and no change will result at $t = 3$, the reset line is set to a 1. Again nothing can happen until the clock is present as well. This is known as the 'ready to be reset' condition.

At $t = 5$ the reset line can be set to a zero with no change.

5.4.1 The 'D' flip flop

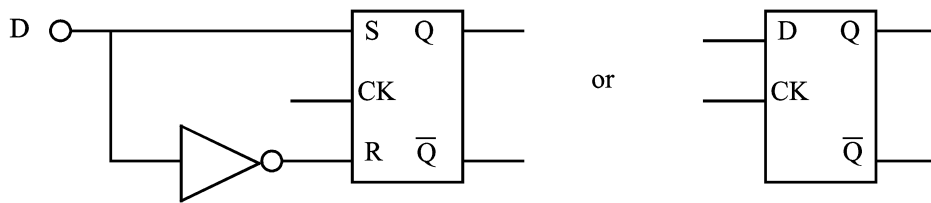
So far the flip flops we have considered require at least 2 inputs, S and R. This is called **double-rail** input data. Should S and R equal each other we have a 'not allowed' condition.

If an inverter is placed between S and R we could prevent this condition ever occurring.

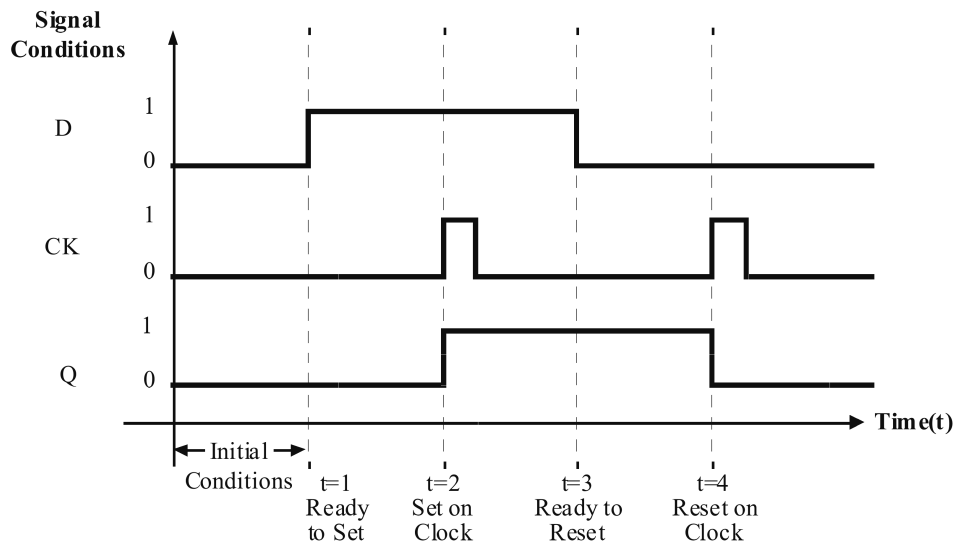
Further only a single line of input data is needed to operate the flip flop. This is called **single-rail** data.

A flip flop constructed in this fashion is called a **D latch** or **D flip flop**.

Using a clocked-NAND SR flip flop we have:



Symbol for a D Flip Flop



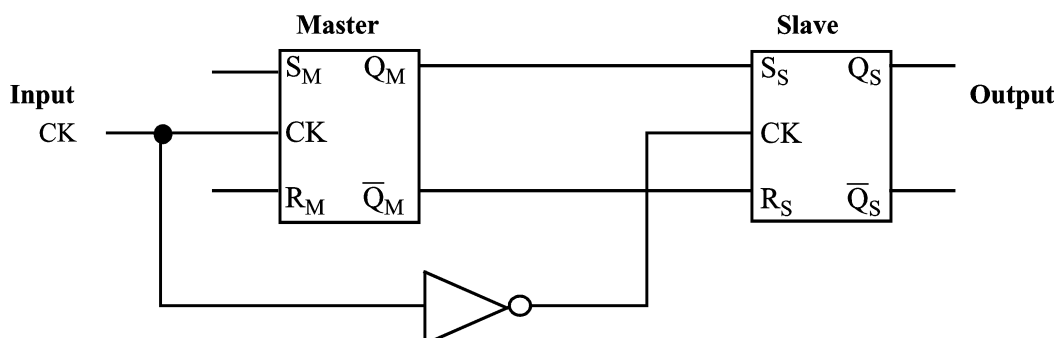
Timing Diagram for a D Flip Flop

5.4.2 The Master-Slave (MS) flip flop

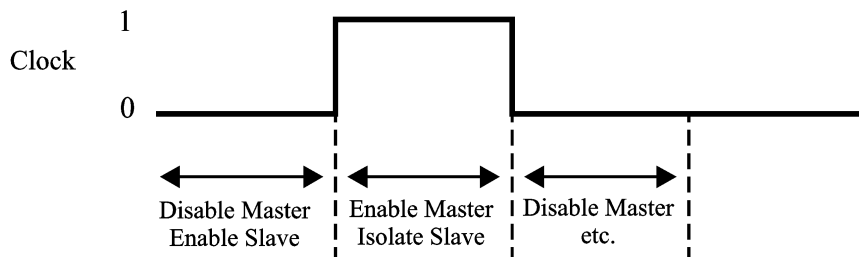
In the SR flip flop, there is sometimes inadequate isolation between **input** and **output**.

Complete isolation can be obtained by cascading 2 SR flip flops known as a **master** and a **slave** and clocking the two out of phase.

This is called a master-slave (M-S) flip flop.



In operation, when the clock is **low**, the master is disabled and the slave is enabled and vice versa. This is due to the inverter being placed in the clock input circuit to the slave FF.

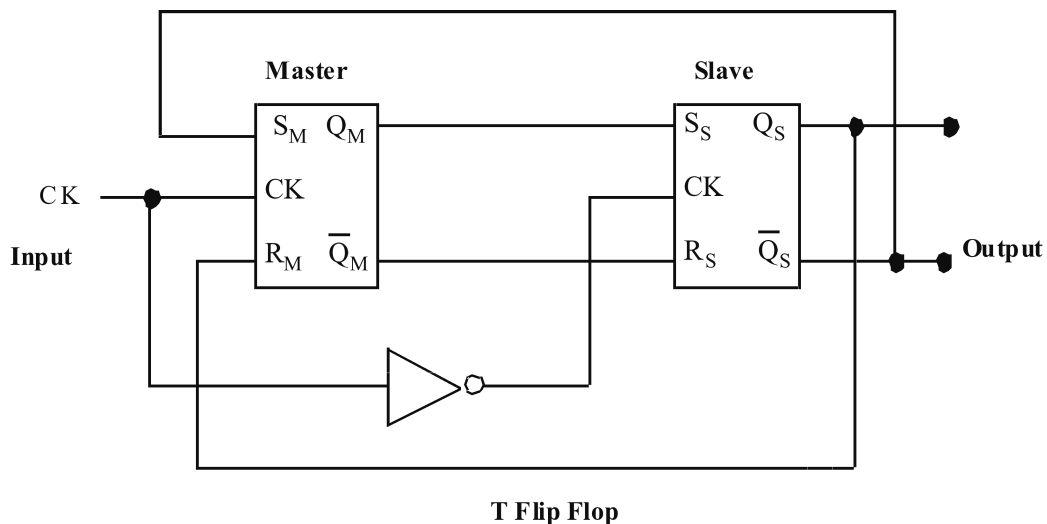


So during the time when the clock pulse is **low** the output conditions of the master flip flop are transferred to the slave which responds accordingly. When the clock is **high** the master is isolated from the slave and the master can respond to its new inputs.

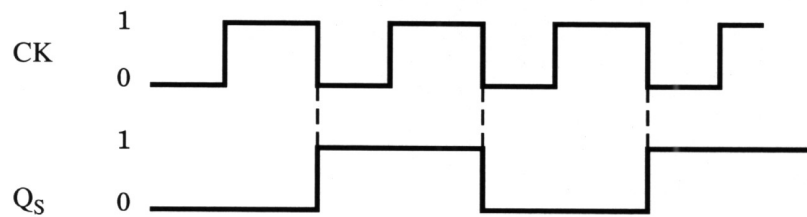
You can see that with correct timing of the clock only the desired signals may be accepted by the master and transferred to the slave outputs. Another way to interpret this is to say the Master responds to the rising edge of the input clock pulse and the Slave responds to the falling edge of the input clock pulse or the output of the master-slave flip flop changes on the falling edge of the input clock pulse.

5.4.3 The 'T' flip flop

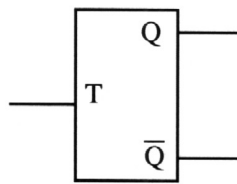
If we now apply feedback to a master-slave flip flop and feed the slave outputs cross-connected back to the master inputs we create a system which changes state on the falling edge of each clock pulse. By changing state on each clock pulse, we say the flip flop 'toggles' on each clock. This is why it is known as a 'T' flip flop.



The timing diagram is:

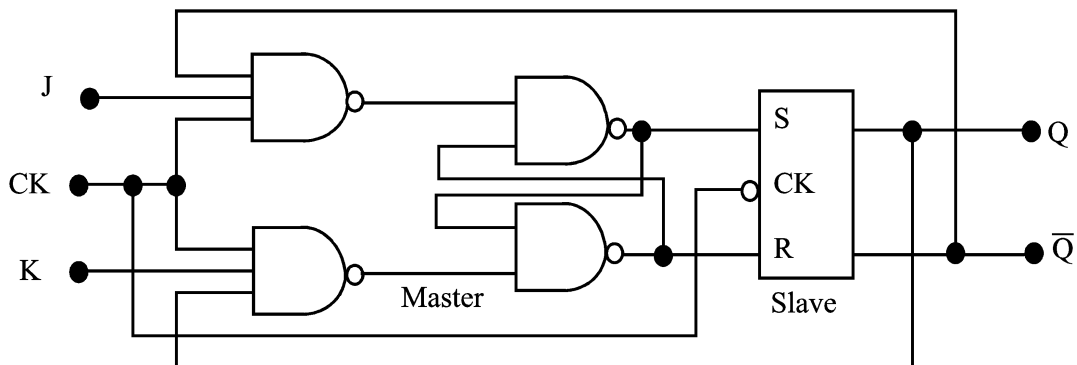


The system is manufactured commercially and has a symbol:

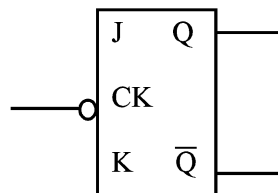


5.4.4 The J-K flip flop

If we now use a clocked SR master slave flip flop with 3-input NAND gates for the master, we create the J-K flip flop.



Its commercial symbol is:



In operation it behaves exactly like an SR FF where J equals the S input and K equals the R input. However, you will remember that in an SR FF we could never put $S = R = 1$. In this flip flop we can put $J = K = 1$ and when we do the flip flop toggles on each falling clock edge and hence it becomes a 'T' flip flop.

| J | K | Q_{n+1} |
|---|---|-------------------------|
| 0 | 0 | Q_n (Q no change) |
| 1 | 0 | 1 (Q set) |
| 0 | 1 | 0 (Q reset) |
| 1 | 1 | \bar{Q}_n (Q toggles) |

Q_n = current state; Q_{n+1} = state following falling edge of input clock pulse



Self assessment 5.1

1. Sketch the diagram of a Master-Slave flip-flop using commercial SR flip-flop symbols and with the aid of a timing diagram explain the meaning of input/output isolation.
2. Sketch a logic diagram for a clocked 'D' flip flop utilising 5 NAND gates. Explain its operation by drawing its input, clock and output waveforms on a timing diagram.
3. Sketch the logic diagram of a clocked SR flip-flop constructed of NOR gates and with the aid of a timing diagram, explain its operation.

5.5 Basic counters

Digital logic counters are an integral part of most digital systems. Counters are used in generating timing sequences to control precise operations such as clock pulses. They may be used in counting the number of times a particular event occurs in a system.

A counter is a logic device which counts through a prescribed sequence of states upon the application of some input pulses. The input pulses may arrive from some other external device or they may be clock pulses arriving regularly or at random.

Counters may be asynchronous, simply counting whenever input pulses are received or they may be synchronous meaning their entire counting operation can only occur when a clock pulse is received as well as the input pulses to be counted.

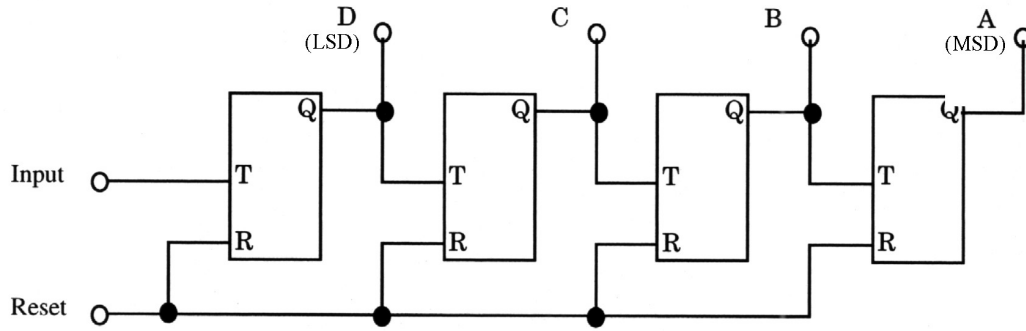
The flip flop is usually used as the basic building block for counter construction. The flip flops may be cascaded serially in which case the operation of any one flip flop depends upon the operation of its predecessor, or they may be connected in parallel in which case all flip flops change state simultaneously usually on receipt of a clock pulse.

One of the simplest forms of counting is the binary sequence counter.

An N-bit binary counter consists of N flip flops and can count from zero to $2^N - 1$.

5.5.1 Serial counting

Let us examine a logic system which is known as a serial counter or ‘ripple through’ counter. In this, the input to be counted is applied to the first flip flop and its output is connected to the next flip flop etc. The diagram shows a basic 4 bit counter constructed by T-flip flops.



4 Bit Ripple Through Binary Counter

We know from section 5.4 that a T flip flop changes state on one-to-zero transitions applied to its T input.

Let us now consider the code table for a binary count from zero to fifteen.

| A | B | C | D |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 |

Notice in column D, that whenever it changes from a one to a zero the next column (C) will also change. Zero-to-one transitions have no effect. Notice also this applies to all columns, A B C and D.

Consider now the operation of the 4 bit binary counter as shown in the diagram. The counter can be reset to all zeros when a pulse is applied to the reset line.

Input pulses may arrive in random sequence and their pulse width may vary (as long as they are greater than some minimum value) without affecting the process of counting. FFD will change state on the trailing edge of the input signal (i.e. $1 \rightarrow 0$) FFC changes state every time FFD goes $1 \rightarrow 0$ etc.

As the count proceeds, the outputs of the flip flops change as per the code table until all are set to a 1 state (i.e. $1111_2 = 15_{10}$). The next input pulse resets FFD to 0 and this $1 \rightarrow 0$ change ‘ripples through’ the counter making all the outputs zero.

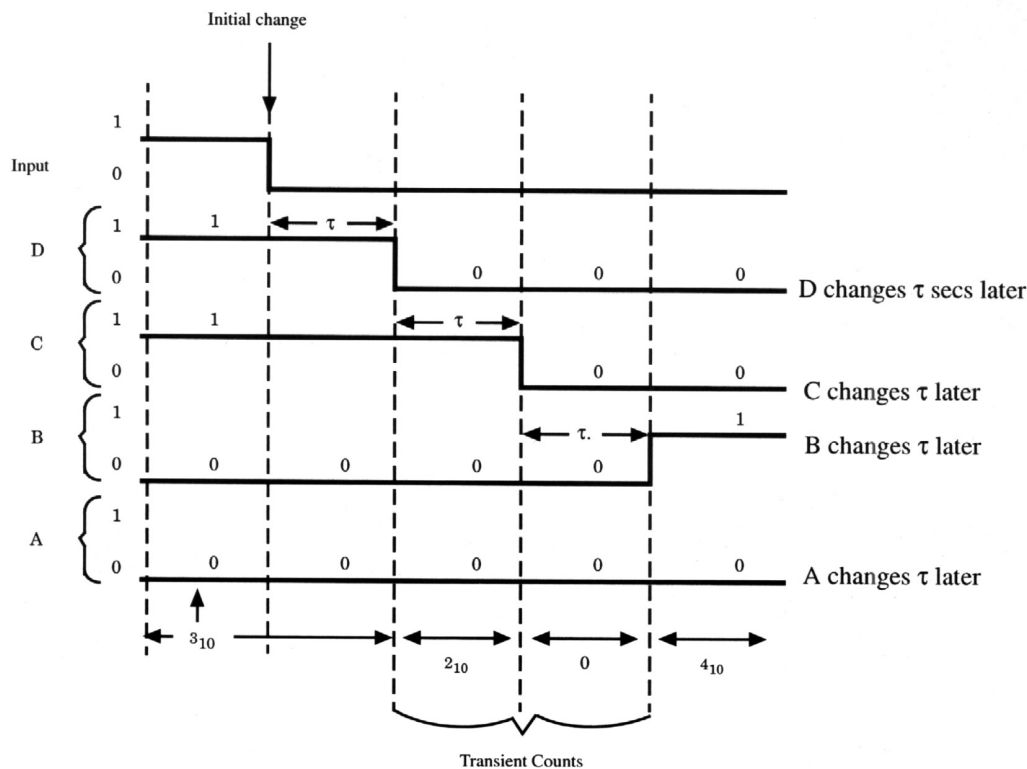
5.5.2 Propagation delay

In practice, due to the time taken for electronic signals to travel through the circuitry, each flip flop has a time delay between the input signal and its output change. This is called the

propagation delay of the device and is usually quite small in individual electronic devices ($< 1 \mu\text{S}$) but can be of the order of milliseconds for whole systems. Problems could occur because of this delay.

Consider a binary counter as it counts from decimal 3 (0011) to decimal 4 (0100).

Assume that each flip flop has a propagation delay of τ seconds. If we observe a timing diagram as our counter goes from 3_{10} to 4_{10} we have:



The system finally gets from 3 to 4 after taking 3τ seconds (3 transient counts) to do so **but** notice that it also had 2 transient counts of 2_{10} and 0 on the way! Sometimes these errors or transient states don't matter, sometimes they do. A detailed consideration of any proposed system is required to evaluate if this could be a problem. Note, the time to count from 111_2 to 0000 will take 4τ seconds and transient states of 14, 12 and 8 will occur on the way.

The counter can be pre-set to some initial value by generating a pre-set signal at the final stage of the count and by connecting this signal to the S or R inputs of each flip flop as appropriate.

5.5.3 Reversible binary counting

Reverse binary counting can be achieved from the code table for a binary counter by reading it in reverse order. We note that each stage, except the least significant stage, changes state when the previous stage changes from 0 to 1. The least significant stage changes state for each input pulse.

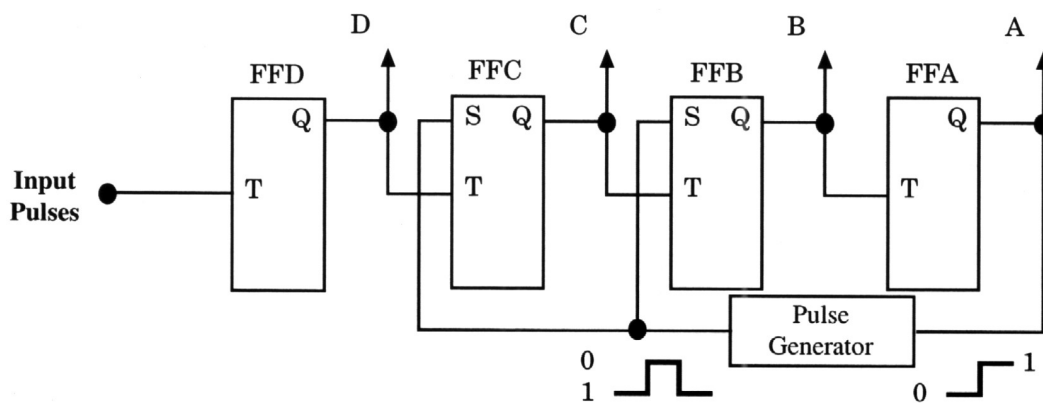
For example, if the state of the counter is 0000 at some time, then the next input pulse must set it to 1111 and the next to 1110, to 1101 etc.

Using T flip flops, reverse counting may be obtained by connecting an inverter in the input of each next stage.

5.5.4 Non-pure binary counting

In the counters we have considered so far the output of all the flip flops is zero simultaneously on a count of 0000 and again after the 16th count, 32nd count and so on.

Suppose the counter needs to be reset to zero after some value e.g. 5 or 10 etc., extra logic elements can be used to effect this. Many circuits are possible and we will look at one here.



The pulse generator gives a short duration pulse **output** when its input sees a $0 \rightarrow 1$ transition. In this case that pulse is fed back to the **set** inputs of FFB and FFC. Consider the situation if the initial state is all zeros and we allow input pulses to enter the first flip flop (FFD).

| Input Pulse | Count (BIN) | | | | Count (DEC) |
|-------------|-------------|---|---|---|-------------|
| | A | B | C | D | |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | 2 |
| 3 | 0 | 0 | 1 | 1 | 3 |
| 4 | 0 | 1 | 0 | 0 | 4 |
| 5 | 0 | 1 | 0 | 1 | 5 |
| 6 | 0 | 1 | 1 | 0 | 6 |
| 7 | 0 | 1 | 1 | 1 | 7 |
| 8 | 1 | 0 | 0 | 0 | 8 |
| 8 | 1 | 1 | 1 | 0 | 14 |
| 9 | 1 | 1 | 1 | 1 | 15 |

This count only appears momentarily until FF's B and C are set by the feedback.

On the 8th pulse FFA changes to a 1 and this produces a pulse from the pulse generator which in turn sets FFB and FFC back to a 1 giving a count of 1110 or 14 decimal. The next input pulse causes the count to go to 15₁₀ and the next pulse after that resets the counter to 0000.

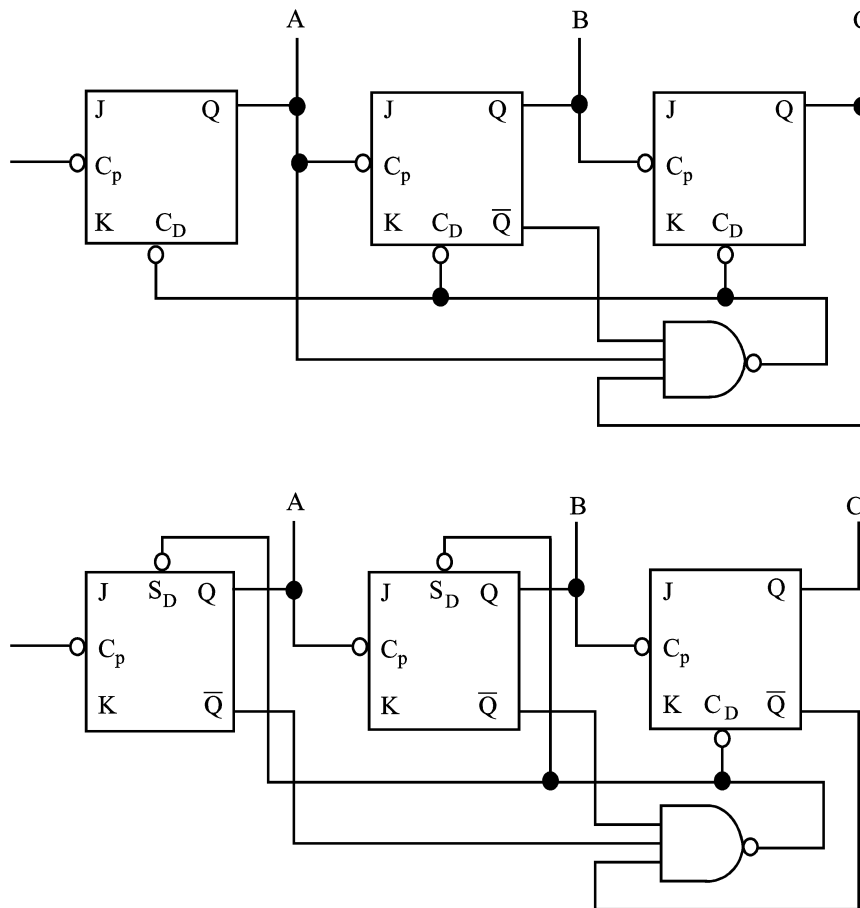
We have returned to our original state after 10 pulses. This is a divide by 10 counter. The result is a $(\div 2^N - X)$ counter where,

N = number of stages in counter

X = **Extra** count set in by the pulse generator

In this case, $\div 2^4 - 6 = 16 - 6 = 10 \therefore$ a \div by 10 counter.

5.5.5 Feedback using logic gates



In some situations standard logic gates such as a NAND gate or a NOR gate are used to provide the feedback detector.

Two diagrams are shown indicating how this may be achieved.

Problems may occur using this system if the output signal from the gate is not maintained long enough for all transient count states to settle.

5.5.6 Synchronous counting

In synchronous systems **all** stages change state simultaneously because a clock pulse is applied to **all** stages at once.

Thus the total time taken for the new state to occur in **one** propagation delay time (τ). Compare this with a ripple through counter propagation delay time of 4τ .

Synchronous systems are usually more complex than asynchronous systems since it is necessary to prepare the input logic gates to count the pulse, in advance of the pulse being received. However the increase in complexity is offset by the increase in operating speed.

Synchronous systems are however often prone to instability problems which do not exist in asynchronous systems.

5.5.7 Design of parallel counters

Several methods have been evolved, some analytical, some using map techniques. The main advantage of a map technique is its simplicity and ease of understanding. An example is the simplest way to illustrate this technique.



Example

Map method for 2421 BCD counter design using J-K flip flops

A code table for this counter is:

Table 5.1

| Weighting | 2 | 4 | 2 | 1 |
|-----------|---|---|---|---|
| DEC count | A | B | C | D |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 |
| 7 | 0 | 1 | 1 | 1 |
| 8 | 1 | 1 | 1 | 0 |
| 9 | 1 | 1 | 1 | 1 |
| 10–15 | X | X | X | X |

A=MSB
2421 code
X=Don't care

The equivalent decimal values are marked on a Karnaugh map:

Map 1: Karnaugh map

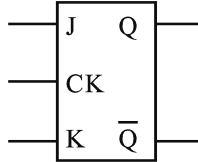
| CD \ AB | AB | | | |
|---------|----|----|----|----|
| | 00 | 01 | 11 | 10 |
| 00 | 0 | 4 | X | X |
| 01 | 1 | 5 | X | X |
| 11 | 3 | 7 | 9 | X |
| 10 | 2 | 6 | 8 | X |

As previously stated, in parallel counters all flip flops change state simultaneously, so the input to each flip flop must be prepared in advance for the clock.

Consider the Q output change of a clocked JK flip flop under various input conditions.

Table 5.2

| | Input | | Output Change Q |
|-------------|-------|---|--------------------|
| | J | K | |
| Set by CK | 1 | X | $0 \rightarrow 1$ |
| Reset by CK | X | 1 | $1 \rightarrow 0$ |
| No Change | 0 | X | $0 \rightarrow 0$ |
| No Change | X | 0 | $1 \rightarrow 1$ |



X = Don't Care

A 2421 counter would require 4 flip flops (one for each digit) to construct. In this case the four are marked FFA, FFB, FFC and FFD. If we first consider the operation of FFA we see from the code table, that its output is zero for all counts up to decimal 6, after 7 it changes to a 1, maintains during 8 and after 9 changes to a zero again. These observations can now be marked on another map called a code change map for flip flop A corresponding to the counts shown on the Karnaugh Map 5.1 above.

Map 5.1: Karnaugh Map for 2421 BCD Code

| CD \ AB | 00 | 01 | 11 | 10 |
|---------|----|----|----|----|
| 00 | 0 | 4 | X | X |
| 01 | 1 | 5 | X | X |
| 11 | 3 | 7 | 9 | X |
| 10 | 2 | 6 | 8 | X |

Map 5.2: Code Change Map for FFA

| | | | |
|-------------------|-------------------|-------------------|--|
| $0 \rightarrow 0$ | $0 \rightarrow 0$ | | |
| $0 \rightarrow 0$ | $0 \rightarrow 0$ | | |
| $0 \rightarrow 0$ | $0 \rightarrow 1$ | $1 \rightarrow 0$ | |
| $0 \rightarrow 0$ | $0 \rightarrow 0$ | $1 \rightarrow 1$ | |

Counts 0 to 6, FFA = 0, show as $0 \rightarrow 0$ on code change map.
 After count 7, FFA $\rightarrow 1$, show as $0 \rightarrow 1$ on code change map.
 Maintain FFA = 1 for count 8, show as $1 \rightarrow 1$ on code change map.
 After count 9, FFA $\rightarrow 0$, show as $1 \rightarrow 0$ on code change map.

The code change Map 5.2 is now used to determine what the J and K signals will be for FFA. This is normally done by marking the characteristics of a JK flip flop (table 5.2) onto two maps, one for the J input and one for the K input. These are called input logic maps.

| AB \ CD | 00 | 01 | 11 | 10 |
|---------|----|----|----|----|
| 00 | X | X | X | X |
| 01 | X | X | X | X |
| 11 | X | X | 1 | X |
| 10 | X | X | 0 | X |

K-Map

$$\mathbf{K}_A = \mathbf{D}$$

Input Logic Maps

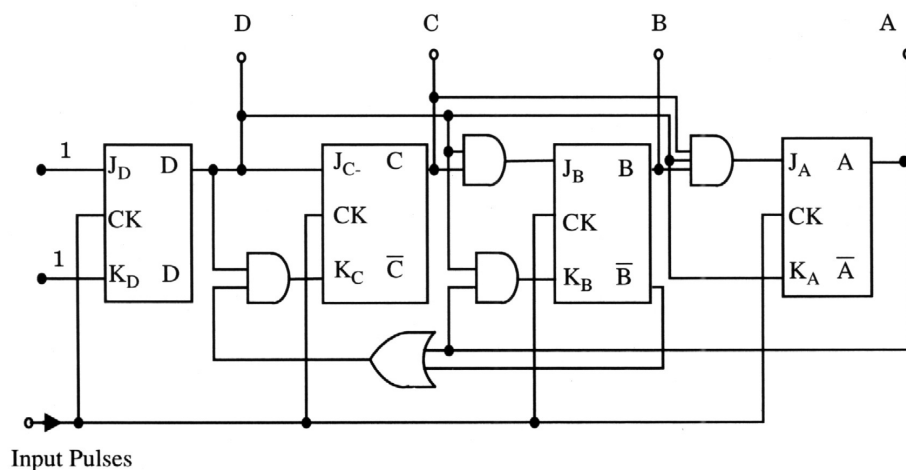
The input signals for FFA have now been determined. A similar procedure is now followed to obtain the input signals for FFB, FFC and FFD.

I will leave their determination as an exercise for you.

The results should be:

$$\begin{array}{llll} J_A & = & BCD & K_A = D \\ J_B & = & CD & K_B = AD \\ J_C & = & D & K_C = D(A + \overline{B}) \\ J_D & = & 1 & K_D = 1 \end{array}$$

Implementation:



Complete 2421 BCD Counter



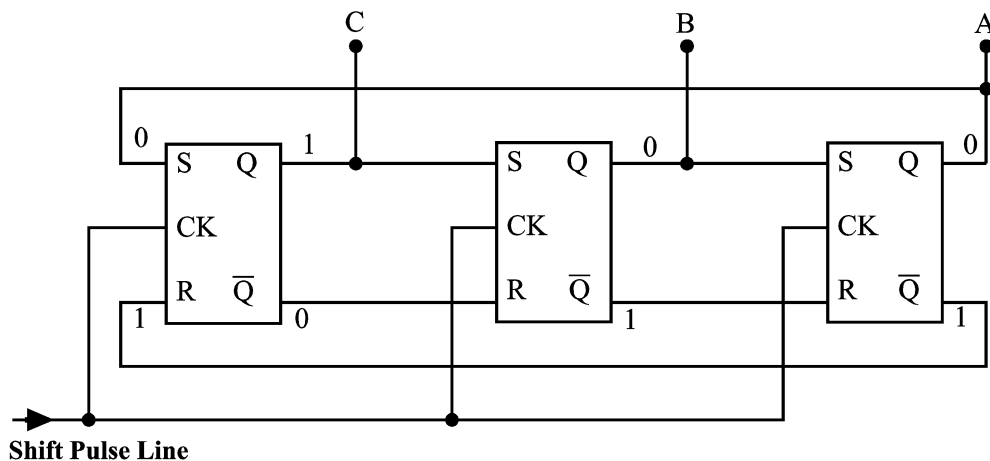
Self assessment 5.2

1. Sketch the diagram of a 3-bit ripple through binary counter using commercial SR flip-flop symbols and with the aid of a truth table explain its operation.
2. Devise a logic system using JK flip-flops connected as T flip-flops to divide a continuous 1 Hz pulse train by 10. Draw the complete logic system and explain its operation in detail.

5.6 Shift registers

A shift register comprises a number of cascaded flip flops which contain a preset pattern of binary digits. The application of a 'shift pulse' moves the whole pattern one step along the register.

Consider the flip flops connected as shown:



Assume all stages except FFC are initially set to zero. These logic states are marked on the diagram. When the first shift pulse arrives, FFC is reset, FFB is set, FFA is unchanged. The next shift pulse sets FFA, resets FFB and leaves FFC unchanged. You can see that the logic 1 originally in FFC has progressively moved through the register. Continual shift pulses move the pattern around and around as though it were a circular device or ring.

For this reason these devices are sometimes called 'ring counters'. The number of times a given pattern is repeated is determined by the initial code set into the register.

Let us consider the pattern generated by this shift register.

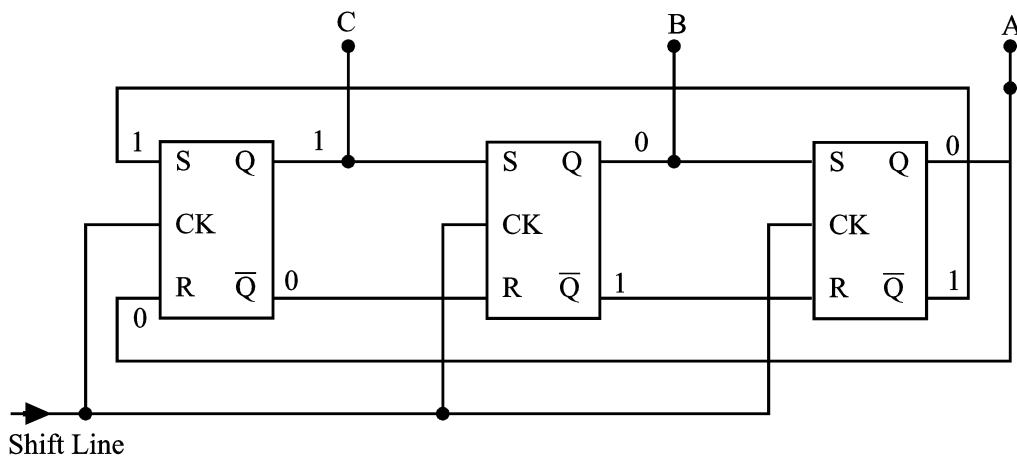
| | A | B | C |
|-------------------|---|---|---|
| Initial Condition | 0 | 0 | 1 |
| First Shift Pulse | 0 | 1 | 0 |
| Second Shift | 1 | 0 | 0 |
| Third Shift | 0 | 0 | 1 |

← (Back to initial conditions again)

5.6.1 Feedback variations

By feeding back various functions of the state of the register to its input, shift registers can be made to generate a sequence of binary combinations which need not necessarily follow conventional sequences. For instance by feeding back the **'complement'** of the last stage we can generate a pattern of length $2N$, where N is the number of stages.

Feedback shift registers generate one 'progressive' or 'creeping' code of length $2N$ in addition to other codes. Consider a three stage system ABC with complementary feedback initially set to 001.



In this case the pattern generated is a creeping code of $2N = 6$.

| A | B | C | Count Sequence |
|---|---|---|----------------|
| 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 3 |
| 1 | 1 | 1 | 7 |
| 1 | 1 | 0 | 6 |
| 1 | 0 | 0 | 4 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |

Code Length = 6

Count Repeats

We have $2^3 = 8$ possible combinations for a three flip flop system, so another code equal to

$2^N - 2N$ is $8 - 6 = 2$ exists. Examination of the count sequence above reveals that count 2 is missing. If we therefore started our system at 2 we have a sequence:

| A | B | C | Count Sequence |
|---|---|---|----------------|
| 0 | 1 | 0 | 2 |
| 1 | 0 | 1 | 5 |
| 0 | 1 | 0 | 2 |

} Code Length = 2
→ Count Repeats

The code of length $2N$ is called the **main code**.

The code of length $2^N - 2N$ is called the **auxiliary code**.

In general for N stages,

3. Length of each main code = $2N$
4. Number of main codes = integer part of $\frac{2^N}{2N}$
5. Length of auxiliary code = $2^N - 2N$

The main disadvantages with this counter are 1. the short length of the main codes and 2. the counter may commence operating in any code. The latter can be overcome by starting the system with a preset code set in.

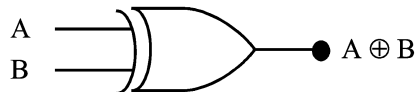
5.6.2 Linear feedback shift registers

The length of the main code can be increased by feeding back a **not equivalent** function of the states of the register.

The **not equivalent** of two signals A and B is $A \not\sim B$.

Logically this is $\overline{A}B + A\overline{B}$ which you will recognise as the **exclusive-or** function, $A \oplus B$.

The symbol is:



Where $A \oplus B$ equals $\overline{A}B + A\overline{B}$

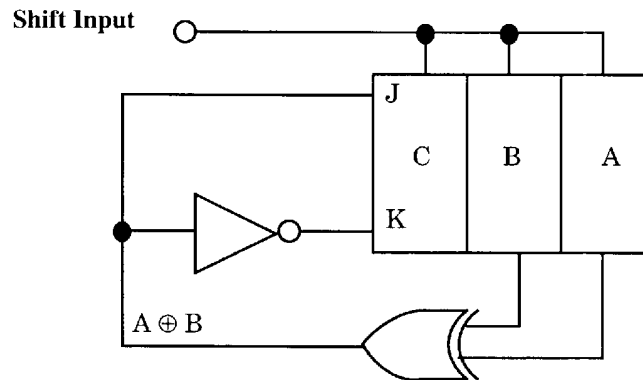
The term **modulo-2** is used for this function also. It derives from the **modulus** of the function.

The modulus of a counter is the number of input pulses for a complete count cycle. For one standard 4 flip flop binary counter without feedback, the modulus is 16, so its called **modulo-16**.



Example 1

Let us consider a 3 stage linear feedback shift register with **not-equivalent** feedback. These shift registers are usually drawn as shown.



If the initial conditions are $ABC = 001$, the count sequence is shown. In each case the previous state of A and B will determine C's state after the shift pulse.

| Count Sequence | A | B | C | |
|----------------|---|---|---|---|
| 1 | 0 | 0 | 1 | Initial conditions $A = B$ |
| 2 | 0 | 1 | 0 | First shift pulse sets B, resets C |
| 5 | 1 | 0 | 1 | $A \neq B$ so C is set on next shift pulse |
| 3 | 0 | 1 | 1 | $A \neq B$ so C is maintained |
| 7 | 1 | 1 | 1 | same again |
| 6 | 1 | 1 | 0 | $A = B$ so reset C |
| 4 | 1 | 0 | 0 | same again |
| 1 | 0 | 0 | 1 | $A \neq B$ so C and cycle repeats |

Code Length = 7

This system with 'not equivalent' feedback results in a code length of 7 or $2^N - 1$. The missing state is 000 which if allowed to occur would stop the counter.

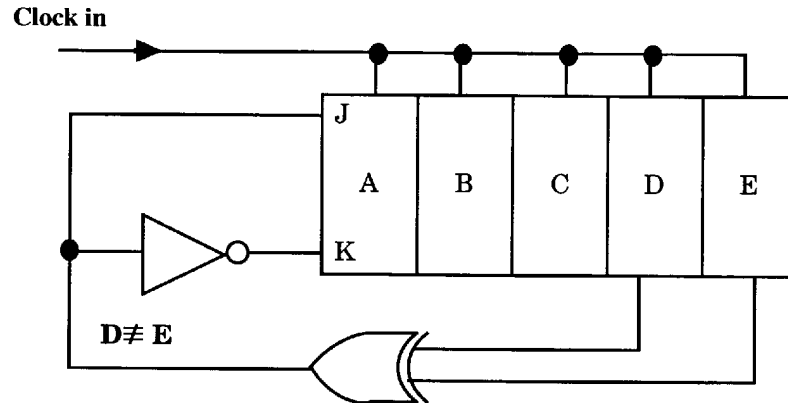
The length and number of cycles is dependant on the points from which feedback is taken.

For example if Modulo-2 feedback is taken from stages 4 and 5 of a 5 stage linear feedback shift register, it generates a main code length of 21 and two auxiliary codes of length 7 and 3 respectively.



Example 2

Consider 5 stages with Modulo-2 feedback as shown:



| | E | D | C | B | A | DEC | |
|-------------------|---|---|---|---|---|-----|--------------------------------------|
| Initial Condition | 0 | 0 | 0 | 0 | 0 | 0 | stays in this state so it is avoided |
| | 0 | 0 | 0 | 0 | 1 | 1 | ← Initial Condition |
| | 0 | 0 | 0 | 1 | 0 | 2 | ← First Shift Pulse etc. |
| | 0 | 0 | 1 | 0 | 0 | 4 | |
| | 0 | 1 | 0 | 0 | 0 | 8 | |
| | 1 | 0 | 0 | 0 | 1 | 17 | |
| | 0 | 0 | 0 | 1 | 1 | 3 | |
| | 0 | 0 | 1 | 1 | 0 | 6 | |
| | 0 | 1 | 1 | 0 | 0 | 12 | |
| | 1 | 1 | 0 | 0 | 1 | 25 | |
| | 1 | 0 | 0 | 1 | 0 | 18 | |
| | 0 | 0 | 1 | 0 | 1 | 5 | |
| | 0 | 1 | 0 | 1 | 0 | 10 | |
| | 1 | 0 | 1 | 0 | 1 | 21 | |
| | 0 | 1 | 0 | 1 | 1 | 11 | |
| | 1 | 0 | 1 | 1 | 1 | 23 | |
| | 0 | 1 | 1 | 1 | 1 | 15 | |
| | 1 | 1 | 1 | 1 | 1 | 31 | |
| | 1 | 1 | 1 | 1 | 0 | 30 | |
| | 1 | 1 | 1 | 0 | 0 | 28 | |
| | 1 | 1 | 0 | 0 | 0 | 24 | |
| | 1 | 0 | 0 | 0 | 0 | 16 | |
| | 0 | 0 | 0 | 0 | 1 | 1 | ← Cycle Repeats |

Code Length = 21

The first decimal number missing in the main code is 7. So start at count 7, for the first auxiliary code.

| E | D | C | B | A | DEC |
|---|---|---|---|---|-----|
| 0 | 0 | 1 | 1 | 1 | 7 |
| 0 | 1 | 1 | 1 | 0 | 14 |
| 1 | 1 | 1 | 0 | 1 | 29 |
| 1 | 1 | 0 | 1 | 0 | 26 |
| 1 | 0 | 1 | 0 | 0 | 20 |
| 0 | 1 | 0 | 0 | 1 | 9 |
| 1 | 0 | 0 | 1 | 1 | 19 |
| 0 | 0 | 1 | 1 | 1 | 7 |

Code Length = 7

Cycle Repeats

The second auxiliary code starts at 13.

| | | | | | |
|---|---|---|---|---|----|
| 0 | 1 | 1 | 0 | 1 | 13 |
| 1 | 1 | 0 | 1 | 1 | 27 |
| 1 | 0 | 1 | 1 | 0 | 22 |
| 0 | 1 | 1 | 0 | 1 | 13 |

Code Length = 3

Cycle Repeats

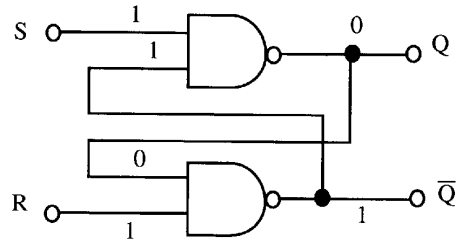
If we were to consider the sequence of binary digits from any of the 5 flip flops outputs it appears to have no pattern and is random. Strictly speaking is not truly random as it repeats itself after 21 digits. Hence it is known as a pseudo-random binary sequence, (PRBS). These PRBS signals are often used to test a data transmission system for errors in the transmitted signal.

If a feedback system other than a modulo-2 is used, then a non-linear feedback shift register is created.



Solution to activity 5.1

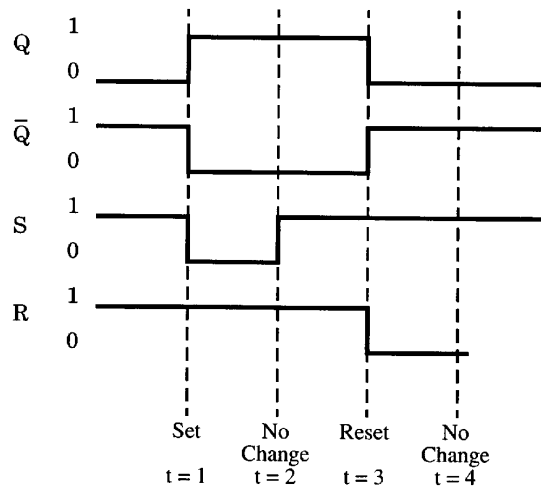
The same SR FF can be made from NAND gates.



| A | B | \overline{AB} |
|---|---|-----------------|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Suppose we have initially:

$S = 1, R = 1, Q = 0$ (we have a stable condition)



We avoid the simultaneous condition change
 $S = R = 0$.

Note this FF changes state whenever S or R changes from a one to a zero.

This FF is sometimes called a **latch**.



Activity 5.2

1. A 10-stage ripple-through counter is constructed using SN7472 integrated circuits. These flip-flops are capable of toggling at a 15 MHz rate and have delay times for 1 to 0 transitions of 25 nS and 0 to 1 transitions of 16 nS.

If the input frequency to this counter is 10MHz determine if it is possible for this counter to register the state 1000000000.

Hint: Consider the worst case for propagation delays for this counter.

2. Devise a logic system, using J.K. Flip Flops connected as T Flip Flops and other standard logic units, to divide a continuous 1 Hz pulse train by the following numbers:
 - a. 9 if the control signals are $XY = 00$
 - b. 10 if the control signals are $XY = 01$
 - c. 11 if the control signals are $XY = 10$
 - d. 12 if the control signals are $XY = 11$

When the division ratio is an even number the output signal of your system should be a square wave.

Note: The control signals may change independently of any other signal. Spurious outputs after a change in control signal are allowed before the system settles into its new division ratio.

Hint: Consider section 5.5.5.

3. Refer to the ELE1301 course page on 'Study Desk' and complete the following experiments:
 - a. Home experiment 5-1 – Clocked S-R flip-flop
 - b. Home experiment 5-2 – Master-slave flip-flop.
 - c. Home experiment 5-3 – Serial counter.



Solutions to activity 5.2

1. This example is designed to illustrate the propagation delay in a serial counter.

The worst case (i.e. the largest number of propagations) must be the count proceeding the desired state of 1000000000 which must be 0111111111.

So to go from 0111111111 to 1000000000, nine FFs must change state from a 1 to a 0 and one FF must change 0 to 1.

Times are: $9 \times \downarrow = 9 \times 25\text{ns} = 225\text{ns}$

$1 \times \uparrow = 1 \times 16\text{ns} = 16\text{ns}$

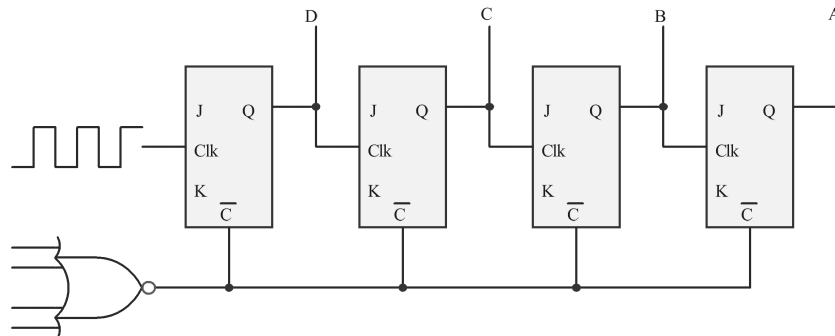
total time = 241ns

Compare this time with the period of a 10MHz clock signal

$$\text{i.e. } \frac{1}{10 \times 10^6} = 10^{-7} = 100 \times 10^{-9} = 100\text{ns}$$

Therefore the input (10MHz (100ns signal)) will change faster than the signals can propagate (241ns) through the counter, so the count of 1000000000 cannot be set up before the first FF changes again.

2. Logic circuit:



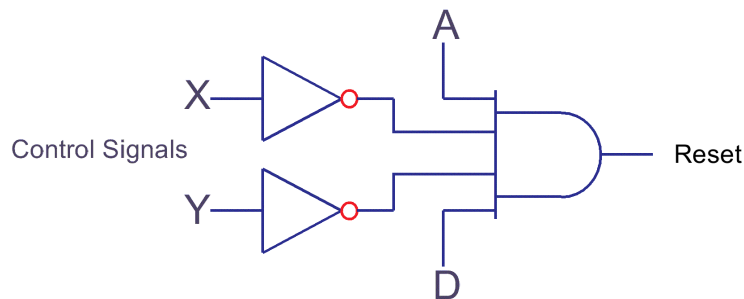
Reset Conditions

Control signals:

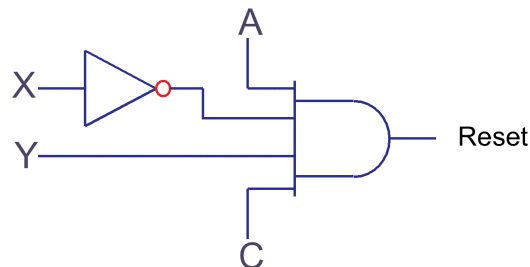
| | | X | Y | | | A | B | C | D | |
|-----|---|----------|----------|---|--------|----------|----------|----------|----------|---------|
| ÷9 | → | 0 | 0 | → | detect | 1 | 0 | 0 | 1 | & reset |
| ÷10 | → | 0 | 1 | → | detect | 1 | 0 | 1 | 0 | & reset |
| ÷11 | → | 1 | 0 | → | detect | 1 | 0 | 1 | 1 | & reset |
| ÷12 | → | 1 | 1 | → | detect | 1 | 1 | 0 | 0 | & reset |

Detectors (2 only shown):

÷9 - detect A & D then and reset if X = Y = 0



÷10 - detect A & C then and reset if X = 0, Y = 1



Output of each circuit is then fed to a 4 input NOR gate to reset the FFs.



Self assessment 5.3

1. Sketch the logic diagram of a three stage feedback shift register using clocked SR flip-flops. The feedback is effected by connecting the complement output of the last stage (A) to the input of the first stage (C). If the initial conditions ABC = 001, derive truth tables for the register operation following each clock pulse.
2. Explain the meaning and use of not equivalent feedback as applied to a linear feedback shift register.
3. Sketch the schematic diagram for a three stage, linear feedback shift register using JK flip-flops, with not equivalent feedback connected from the last two stages (A & B) to the first stage (C) and show by means of a truth table its counting sequence, commencing at the state 001 = ABC.

