1.6 RESISTOR TRANSISTOR LOGIC (RTL)

The resistor-transistor Logic(RTL) circuit is one of the basic logic circuits in digital logic families. It is a bipolar saturated device. The RTL logic is popular because of its simplicity.

The RTL circuit consists of resistors at inputs and transistors at the output side. Transistors are used as the switching device. The emitter of the transistor is connected to the ground. The collector terminals are tied together and given to the supply through the resistor R_C. The collector resistor is known as a passive pull-up resistor.

2-input RTL NOR gate

The following figure shows the circuit diagram of 2-input RTL NOR gate. Q_1 and Q_2 are the two transistors. A and B are the two inputs, given to the base of two transistors and Y is the output.

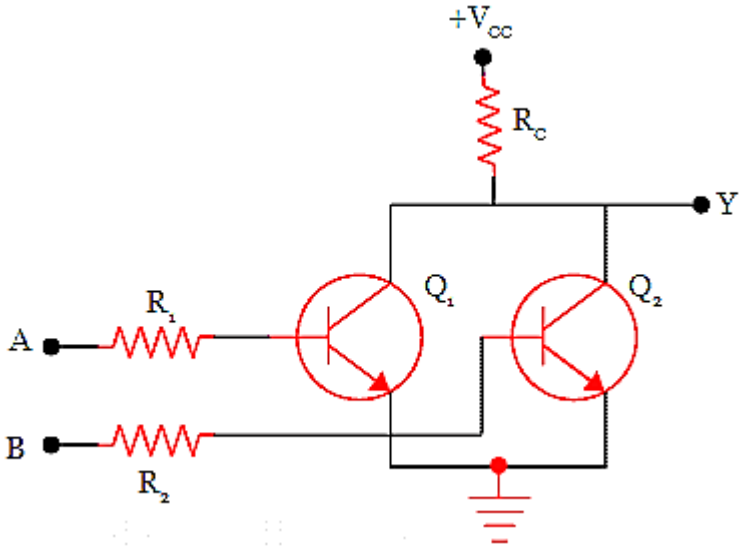


Figure 1.6.1 2-input RTL NOR gate

[Source: https://www.electrically4u.com/resistor-transistor-logic-rtl/]

When both the inputs A and B are at 0V or logic 0, it is not enough to turn on the gates of both the transistor. So the transistors will not conduct. Due to this, the voltage +VCC will appear at the output Y. Hence the output is logic 1 or logic HIGH at terminal Y. When any one of the inputs, either A or B is given HIGH voltage or logic 1, then the transistor with HIGH gate input will be turned on. This will make a path for the supply voltage to go to the ground through the resistor R_C and transistor. Thus there will be 0 v

at the output terminal Y. When both the inputs are HIGH, it will drive both the transistor to turn on. It will make a path for the supply voltage to flow to the ground through resistor R_C and transistor. Therefore, there will be 0 v at the output terminal Y.

The below table shows the truth table for NOR gate.

Inputs		Output
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Table 1.6.1 Truth Table – Nor Gate

[Source: https://www.electrically4u.com/resistor-transistor-logic-rtl/]

3-input RTL NOR gate

The above discussed 2-input RTL NOR gate is the basis for all the logic circuits built with resistors and transistors. The 3-input Resistor-Transistor Logic NOR gate can also be constructed as shown below. The operation is similar to the 2-input RTL NOR gate.

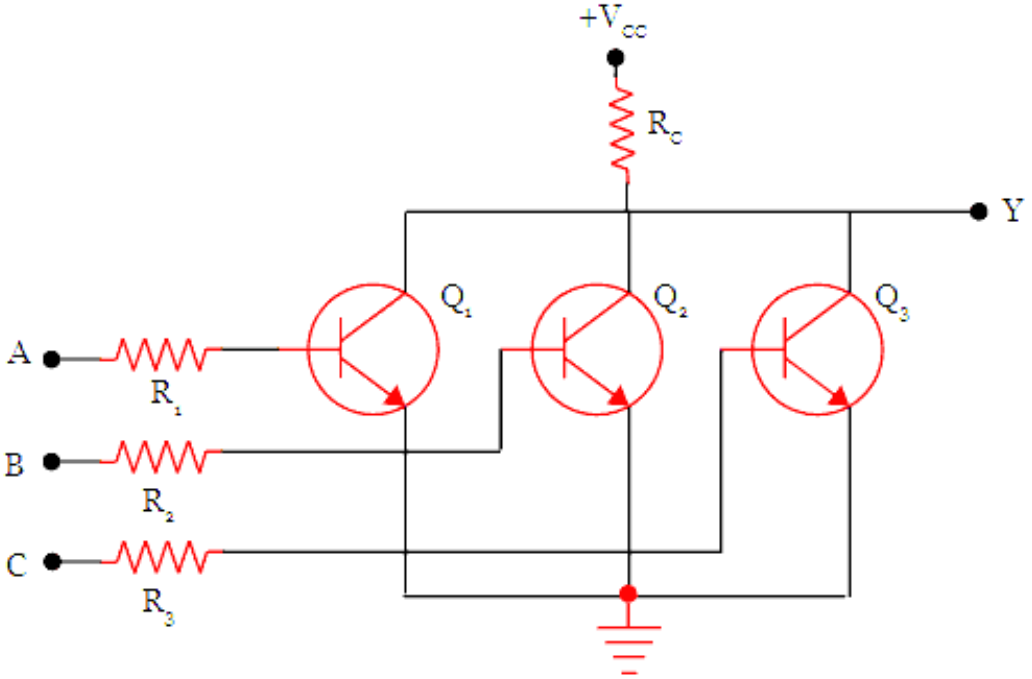


Figure 1.6.2 3-input RTL NOR gate

[Source: https://www.electrically4u.com/resistor-transistor-logic-rtl/]

Limitations

When the transistor is switched on, the power dissipation increases as the current flows through base and collector. Also, the RTL gate has poor noise margin, poor fan-out and the propagation delay is more.

1.3 BINARY CODES

Binary codes are codes which are represented in binary system with modification from the original ones. There are two types of binary codes: Weighted codes and Non-Weighted codes. BCD and the 2421 code are examples of weighted codes. In a weighted code, each bit position is assigned a weighting factor in such a way that each digit ca n be evaluated by adding the weight of all the 1's the in coded combination.

8421 code/BCD code

The BCD (Binary Coded Decimal) is a straight assignment of the binary equivalent. It is possible to assign weights to the binary bits according to their positions. The weights in the BCD code are 8,4,2,1.

Example: The bit assignment 1001, can be seen by its weights to represent the decimal 9 because 1x8+0x4+0x2+1x1 = 9

Weighted Code

– 8421 code

- Most common
- Default
- The corresponding decimal digit is determined by adding the weights associated with t code group.

– 62310 = 0110 0010 0011

– 2421, 5421,7536, etc… codes

• The weights associated with the bits in each code group are given by the name of the code

Nonweighted Codes

Non Weighted codes are codes that are not positionally weighted. That is, each position within the binary number is not assigned a fixed value.

- Actually weighted 74210 except for the digit 0
- Used by the post office for scanning bar codes for zip codes
- Has error detection properties

2421 code

This is a weighted code; its weights are 2, 4, 2 and 1. A decimal number is represented in 4-bit form and the total four bits weight is 2 + 4 + 2 + 1 = 9. Hence the 2421 code represents the decimal numbers from 0 to 9.

5211 code

This is a weighted code; its weights are 5, 2, 1 and 1. A decimal number is represented in 4-bit form and the total four bits weight is 5 + 2 + 1 + 1 = 9. Hence the 5211 code represents the decimal numbers from 0 to 9.

Reflective code

A code is said to be reflective when code for 9 is complement for the code for 0, and so is for 8 and 1 codes, 7 and 2, 6 and 3, 5 and 4. Codes 2421, 5211, and excess-3 are reflective, whereas the 8421 code is not.

Sequential code

The BCD (Binary Coded Decimal) is a straight assignment of the binary equivalent. It is possible to assign weights to the binary bits according to their positions. The weights in the BCD code are 8,4,2,1.

Excess- 3 code

Excess-3 is a non weighted code used to express decimal numbers. The code derives its name from the fact that each binary code is the corresponding 8421 code plus 0011(3). Example: 1000 of 8421 = 1011 in Excess-3

Gray code

The gray code belongs to a class of codes called minimum change codes, in which only one bit in the code changes when moving from one code to the next. The Gray code is non-weighted code, as the position of bit does not contain any weight. In digital Gray code has got a special place.

Decimal Number	Binary Code	Gray Code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

The gray code is a reflective digital code which has the special property that any two

subsequent numbers codes differ by only one bit. This is also called a unit-distance code. Important when an analog quantity must be converted to a digital representation. Only one bit changes between two successive integers which are being coded.

Error Detecting and Correction Codes

Error detecting codes

When data is transmitted from one point to another, like in wireless transmission, or it is just stored, like in hard disks and memories, there are chances that data may

get corrupted. To detect these data errors, we use special codes, which are error detection codes.

Error correcting code

Error-correcting codes not only detect errors, but also correct them. This is used normally in Satellite communication, where turn-around delay is very high as is the probability of data getting corrupt.

Hamming codes

Hamming code adds a minimum number of bits to the data transmitted in a noisy channel, to be able to correct every possible one-bit error. It can detect (not correct) two-bit errors and cannot distinguish between 1-bit and 2-bits inconsistencies. It can't - in general - detect 3(or more)-bits errors.

Parity codes

A parity bit is an extra bit included with a message to make the total number of 1's either parity codes, every data byte, or nibble (according to how user wants to use it) is checked if they have even number of ones or even number of zeros. Based on this information an additional bit is appended to the original data. Thus if we consider 8-bit data, adding the parity bit will make it 9 bit long.

At the receiver side, once again parity is calculated and matched with the received parity (bit 9), and if they match, data is ok, otherwise data is corrupt.

Two types of parity

-Even parity: Checks if there is an even number of ones; if so, parity bit is zero. When the number of one'sis odd then parity bit is set to 1.

-Odd Parity: Checks if there is an odd number of ones; if so, parity bit is zero. When the number of one'sis even then parity bit is set to 1.

Alphanumeric codes

The binary codes that can be used to represent all the letters of the alphabet, numbers and mathematical symbols, punctuation marks, are known as alphanumeric codes or character codes. These codes enable us to interface the input-output devices like the keyboard, printers, video displays with the computer.

ASCII codes

Codes to handle alphabetic and numeric information, special symbols, punctuation marks, and control characters.

• ASCII (American Standard Code Information Interchange) is the best known.

• Unicode –a 16-bit coding system provides for foreign languages, mathematical symbols, geometrical shapes, dingbats, etc. It has become a world standard alphanumeric code for microcomputers and computers. It is a 7-bit code representing $2^7 = 128$ different characters. These characters represent 26 upper case letters (A to Z), 26 lowercase letters (a to z), 10 numbers (0 to 9), 33 special characters and symbols and 33 control characters.

EBCDIC codes

EBCDIC stands for Extended Binary Coded Decimal Interchange. It is mainly used with large computer systems like mainframes. EBCDIC is an 8-bit code and thus accommodates up to 256 characters. An EBCDIC code is divided into two portions: 4 zone bits (on the left) and 4 numeric bits (on the right).

Example 1: Give the binary, BCD, Excess-3, gray code representations of numbers: 5,8,14.

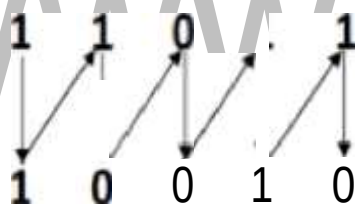
Derimal Number	Billil code	BCD code	Excess-3 code	Cray code
5	0l 81	8101	IONS	01 I I
8	18	I ti0	I 0 11	11 OF
14	i iie	0001 0100	0100 0111	iooi

Example 2: Binary To Gray Code Conversion

1 + 0 + 0 + 1 + 0 (BINARY)

1 1 0 1 1 (CONVERTSO 6RAY CODE)

Example 3: Gray Code To Binary Code Conversion



1.2 BOOLEAN ARITHMETIC

Binary Addition

Rules of Binary Addition

✓ **Binary Addition**

✓ **Rules of Binary Addition**

- 0 + 0 = 0
- 0 + 1 = 1
- 1 + 0 = 1
- 1 + 1 = 0, and carry 1 to the next more significant bit

✓ Example

00011010 + 00001100 = 00100110

```
      1  1
    0  0  0  1  1  0  1  0
+   0  0  0  0  1  1  0  0
-------------------------
    0  0  1  0  0  1  1  0
```


Note: The rules of binary addition (without carries) are the same as the truths of the XOR gate.

Binary Subtraction

Rules of Binary Subtraction

0 - 0 = 0

0 - 1 = 1, and borrow 1 from the next more significant bit

1 - 0 = 1

1 - 1 = 0

Example

00100101 - 00010001= 00010100

```
  0 0 1 0 0 1 0 1
+ 0 0 0 1 0 0 0 1
-----------------
  0 0 0 1 0 1 0 0
```

Binary Multiplication

Rules of Binary Multiplication

0 x 0 = 0

0 x 1 = 0

1 x 0 = 0

1 x 1 = 1, and no carry or borrow bits

Example

00101001 × 00000110 = 11110110

```
    0 0 1 0 1 0 0 1
  × 0 0 0 0 0 1 1 0
-----------------------
    0 0 0 0 0 0 0 0
  0 0 1 0 1 0 0 1
  0 1 0 1 0 0 1
-----------------------
  0 1 1 1 1 0 1 1 0
```

Note: The rules of binary multiplication are the same as the truths of the AND gate.

Binary Division

Binary division is the repeated process of subtraction, just as in decimal division.

Example 1: 00101010 ÷ 00000110 = 00000111

00101010÷00000110 1 1 1 = 7(base 10)

=00000111

1 1 0 0 0 1 1 0 1 0 = 42(base 10)
- 1 1 0 = 6(base 10)

1 borrows

1 0 1 0 1
- 1 1 0

1 1 0
- 1 1 0

0

Example 2: 10000111 ÷ 00000101 = 00011011 1.3 1.4 1.5

10000111÷00000101 1 1 0 1 1 = 27(base 10)
=00011011

1 0 1 0 0 0 1 0 0 1 1 1 = 135(base 10)
- 1 0 1 = 5(base 10)

1 1 1 0
- 1 0 1

1 1
- 0

1 1 1
- 1 0 1

1 0 1
- 1 0 1

0

1.7 DIODE TRANSISTOR LOGIC (DTL)

Diode transistor logic(DTL) belongs to the digital logic family. This logic circuit has diodes at the input side and transistor at the output side and so the name diode transistor logic. It has more advantages than resistor transistor logic(RTL).

Logic circuit of 2-input DTL NAND gate

The following figure shows the circuit for the 2-input DTL NAND gate. It consists of two diodes and a transistor. The two diodes D_A, D_B and the resistor R_1 form the input side of the logic circuit. The common emitter configuration of transistor Q_1 and resistor R_2 forms the output side.

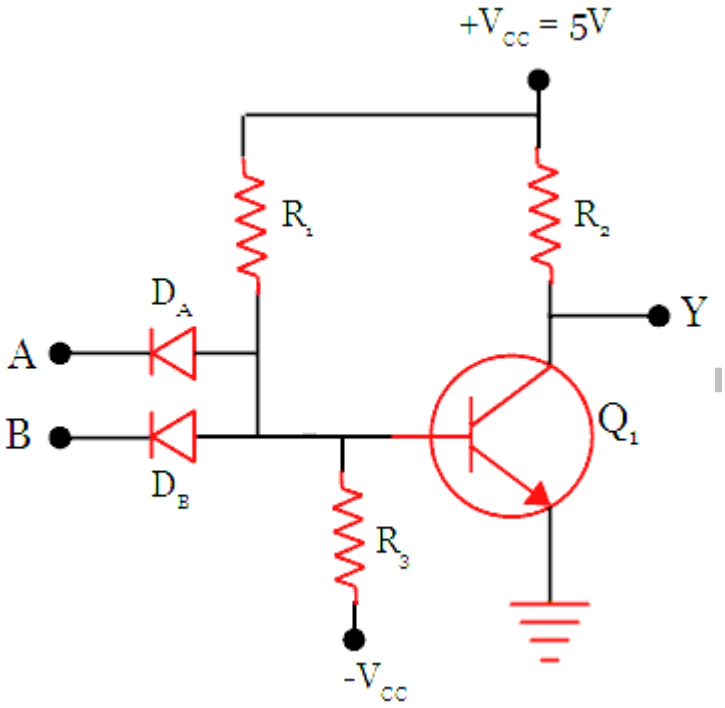


Figure 1.7.1 Circuit of 2-input DTL NAND gate

[Source: https://www.electrically4u.com/diode-transistor-logic]

When both the inputs A and B are LOW, the diodes D_A and D_B become forward biased and so both diodes will conduct in the forward direction. So the current due to the supply voltage $+V_{CC}$ = 5 V will go to the ground through R_1 and the two diodes D_A and D_B. The supply voltage gets dropped in the resistor R_1 and it will not be sufficient to turn ON the transistor. So the transistor will be in cut off mode.

Therefore, the output at the terminal Y will have HIGH value, that is Logic 1. The operation of the gate with the current flow path is shown in the below figure.

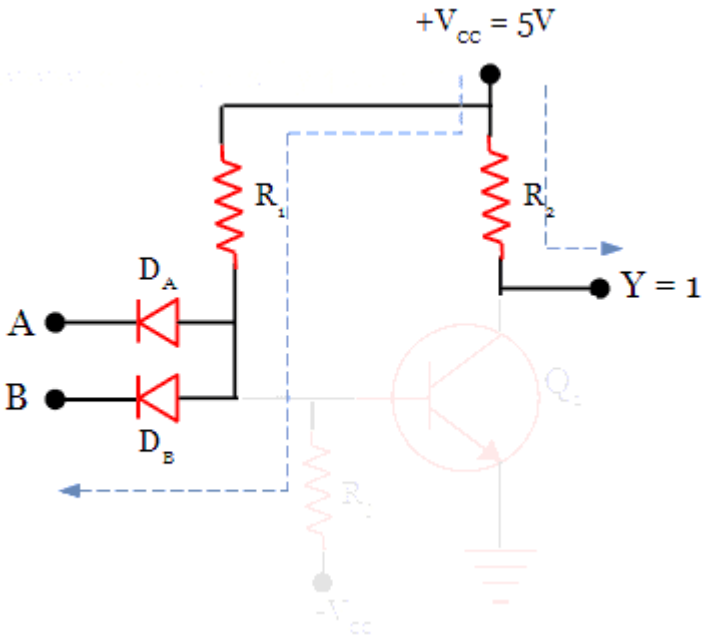


Figure 1.7.2 Case:1 When both inputs are low, the output is high

[Source: https://www.electrically4u.com/diode-transistor-logic]

Now, if anyone of the input, either A or B is given LOW, which makes the corresponding diode to be forward biased. In this case, the same operation will take place.

Since any one of the diodes is forward biased, the current will go the ground through the forward-biased diode and so the transistor will be in cut off mode. The output at the terminal Y will also be at logic 1.

When both the inputs A and B are HIGH, which will reverse bias both the diode. So both diodes will not conduct. In this case, the voltage from the supply $+V_{CC}$, will be enough to drive the transistor into conduction mode.

Thus the transistor will conduct through collector and emitter. The entire voltage gets dropped in the resistor R_2 and the output at the terminal Y will have LOW output, which is considered as logic 0. This operation is shown in the below figure.

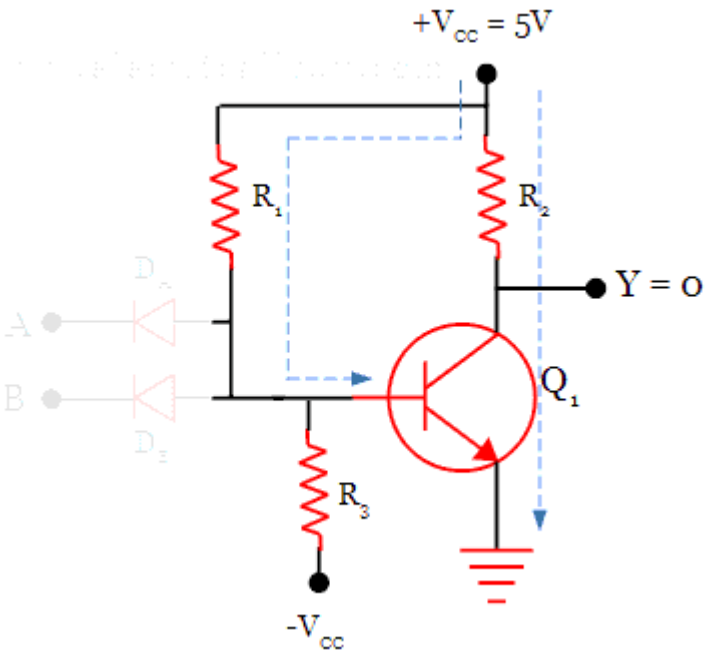


Figure 1.7.3 Case:2 When both inputs are high, the output is low

[Source: https://www.electrically4u.com/diode-transistor-logic]

Advantages

It has better advantages than RTL Logic. The Diode Transistor Logic has improved noise margin, greater fan-out. However, the propagation delay is more for this device, when compared to Transistor-transistor logic(TTL). But the speed is better than RTL.

1.8 EMITTER COUPLED LOGIC (ECL)

Emitter-coupled logic is the fastest of all digital logic families. It was invented by Hannon S. Yourke in the year 1956 at IBM. It is also called as current mode logic. The design of ECL circuit consists of transistors and resistors.

By preventing the transistor from entering into saturation, the high-speed operation is achieved in ECL logic family. Very small voltage swing is necessary to switch between the two different voltage levels. This cannot be achieved in transistor-transistor logic, as the transistors enter into saturation mode, while in operation.

Emitter-coupled logic family offers an incredible propagation delay of 1ns. The delay is more reduced in the latest ECL families. In this section, you will learn about the operation of basic emitter coupled logic implemented for inverter circuit and OR/NOR gate.

Inverter circuit of emitter-coupled logic

The circuit shown below represents the emitter-coupled logic circuit of an inverter. It has two NPN transistors connected in differential single-ended input mode.

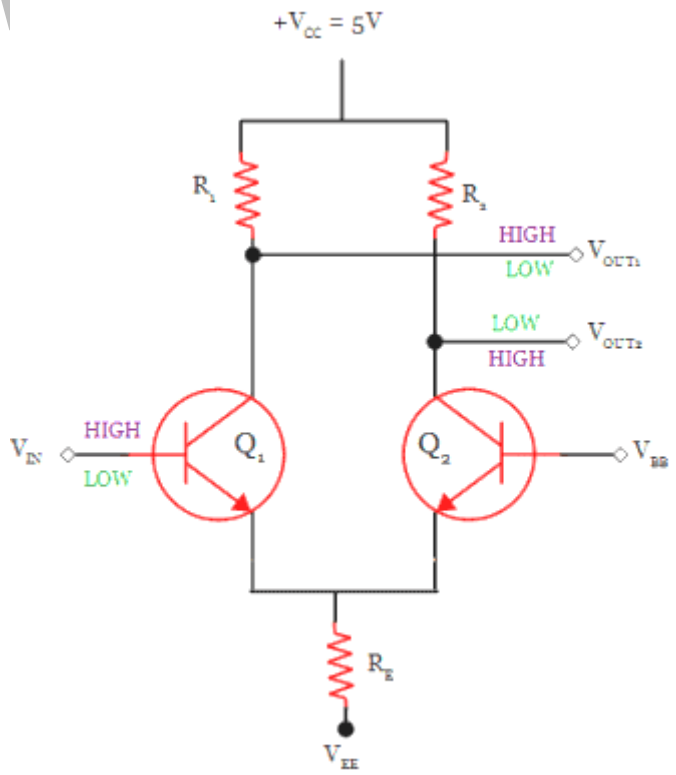


Figure 1.8.1 Emitter-Coupled Logic

[Source: https://www.electrically4u.com/emitter-coupled-logic]

Both the emitters are connected together with common resistance R_E. It is a current limiting resistance, used to prevent the transistor from entering into saturation.

It has two outputs: inverting output(V_{OUT1}) and non-inverting output(V_{OUT12}). V_{IN} is the input terminal, where LOW or HIGH input is given.

When the input is HIGH, it will turn ON the transistor Q_1 but not saturated and the transistor Q_2 is turned OFF. This will pull the output V_{OUT2} to HIGH but due to the drop in resistant R_1, the output at terminal V_{OUT1} will be at LOW value.

On the other side, when the input V_{IN} is given LOW value, it will turn OFF the transistor Q_1 and the transistor is turned ON. The transistor Q_2 will not enter into saturation.

It will make the output at terminal V_{OUT1} to be pulled HIGH value. Due to the drop in resistance R_2, the output at terminal V_{OUT2} will have LOW value.

Two input ECL OR/NOR gate

The following circuit is the Emitter-coupled logic circuit of the 2-input OR/NOR gate. It is the slight modification of the inverter circuit given above. In this, an additional transistor is used at the input side.

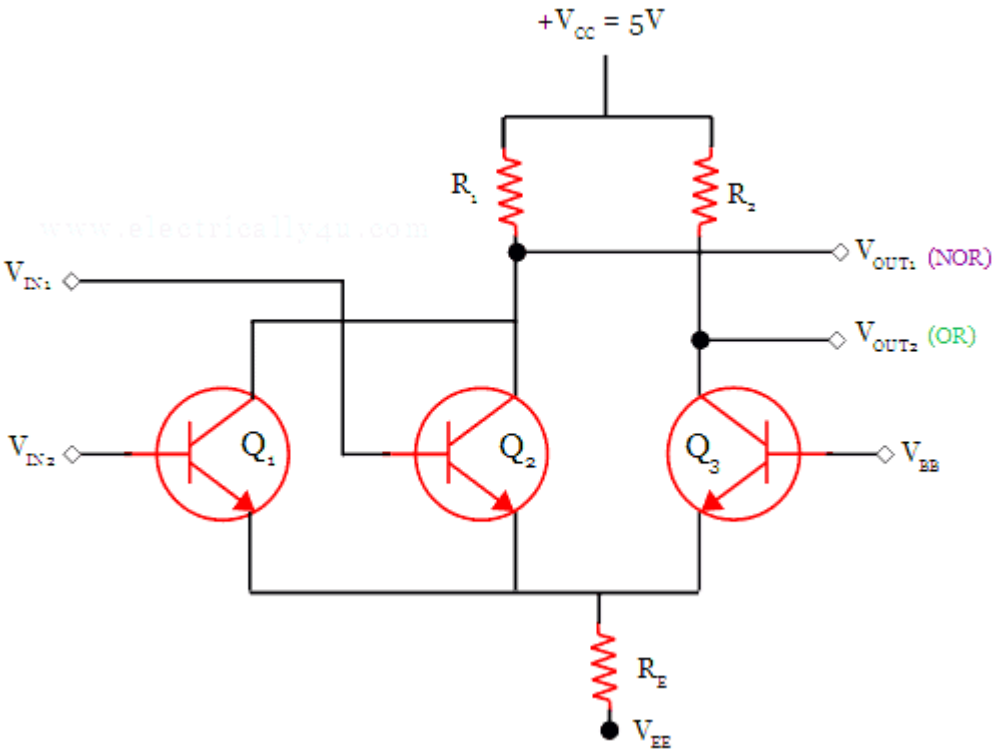


Figure 1.8.2 Two input ECL OR/NOR gate

[Source: https://www.electrically4u.com/emitter-coupled-logic]

The operation is simple as explained above. If the input at both the transistors Q_1 and Q_2 are LOW, it will make V_{OUT1} to HIGH value. It corresponds to the NOR gate output. At the same time, transistor Q_3 is turned ON, which will make the V_{OUT2} to be HIGH. It corresponds to the OR gate output.

Similarly, if both the input of transistors Q_1 and Q_2 are HIGH, it will turn on both the transistors. It will drive the output at terminal V_{OUT1} to be LOW. The transistor Q_3 is

turned OFF during this operation. It will drive the output at terminal V_{OUT2} to be HIGH.

The truth table for OR/NOR gate is shown below.

Inputs		OR	NOR
A	B	Y	Y
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Advantages

- High-speed operation is possible and so the fastest logic family.
- Since transistors are not allowed to enter into saturation, which reduces the storage delay.
- Fan-out capability is high.

Apart from the advantages, it also has its own disadvantage. For the fast switching of transistors, the low and high logic levels are kept close. It reduces the noise margin. Since transistors are not allowed to enter into saturation, the power consumption is more.

1.5 ERROR DETECTING AND ERROR CORRECTING CODES

When bits are transmitted over the computer network, they are subject to get corrupted due to interference and network problems. The corrupted bits leads to spurious data being received by the receiver and are called errors.

Error-correcting codes (ECC) are a sequence of numbers generated by specific algorithms for detecting and removing errors in data that has been transmitted over noisy channels. Error correcting codes ascertain the exact number of bits that has been corrupted and the location of the corrupted bits, within the limitations in algorithm.

ECCs can be broadly categorized into two types −

- Block codes − The message is divided into fixed-sized blocks of bits, to which redundant bits are added for error detection or correction.
- Convolutional codes − The message comprises of data streams of arbitrary length and parity symbols are generated by the sliding application of a Boolean function to the data stream.

Hamming Code

Hamming code is a block code that is capable of detecting up to two simultaneous bit errors and correcting single-bit errors. It was developed by R.W. Hamming for error correction.

In this coding method, the source encodes the message by inserting redundant bits within the message. These redundant bits are extra bits that are generated and inserted at specific positions in the message itself to enable error detection and correction. When the destination receives this message, it performs recalculations to detect errors and find the bit position that has error.

Encoding a message by Hamming Code

The procedure used by the sender to encode the message encompasses the following steps –

- Step 1 − Calculation of the number of redundant bits.

- Step 2 − Positioning the redundant bits.
- Step 3 − Calculating the values of each redundant bit.

Once the redundant bits are embedded within the message, this is sent to the user.

Step 1 − Calculation of the number of redundant bits.

If the message contains *mm*number of data bits, *rr*number of redundant bits are added to it so that *mr* is able to indicate at least (*m* + *r*+ 1) different states. Here, (*m* + *r*) indicates location of an error in each of (*m* + *r*) bit positions and one additional state indicates no error. Since, *rr* bits can indicate $2^r r$ states, $2^r r$ must be at least equal to (*m* + *r* + 1). Thus the following equation should hold $2^r \geq m+r+1$

Step 2 − Positioning the redundant bits.

The *r* redundant bits placed at bit positions of powers of 2, i.e. 1, 2, 4, 8, 16 etc. They are referred in the rest of this text as r_1 (at position 1), r_2 (at position 2), r_3 (at position 4), r_4 (at position 8) and so on.

Step 3 − Calculating the values of each redundant bit.

The redundant bits are parity bits. A parity bit is an extra bit that makes the number of 1s either even or odd. The two types of parity are −

- Even Parity − Here the total number of bits in the message is made even.
- Odd Parity − Here the total number of bits in the message is made odd.

Each redundant bit, r_i, is calculated as the parity, generally even parity, based upon its bit position. It covers all bit positions whose binary representation includes a 1 in the i^{th} position except the position of r_i. Thus −

- r_1 is the parity bit for all data bits in positions whose binary representation includes a 1 in the least significant position excluding 1 (3, 5, 7, 9, 11 and so on)
- r_2 is the parity bit for all data bits in positions whose binary representation includes a 1 in the position 2 from right except 2 (3, 6, 7, 10, 11 and so on)

- r_3 is the parity bit for all data bits in positions whose binary representation includes a 1 in the position 3 from right except 4 (5-7, 12-15, 20-23 and so on)

Decoding a message in Hamming Code

Once the receiver gets an incoming message, it performs recalculations to detect errors and correct them. The steps for recalculation are −

- Step 1 − Calculation of the number of redundant bits.
- Step 2 − Positioning the redundant bits.
- Step 3 − Parity checking.
- Step 4 − Error detection and correction

Step 1 − Calculation of the number of redundant bits

Using the same formula as in encoding, the number of redundant bits are ascertained.

$2^r \geq m + r + 1$ where *m* is the number of data bits and *r* is the number of redundant bits.

Step 2 − Positioning the redundant bits

The *r* redundant bits placed at bit positions of powers of 2, i.e. 1, 2, 4, 8, 16 etc.

Step 3 − Parity checking

Parity bits are calculated based upon the data bits and the redundant bits using the same rule as during generation of c_1,c_2 ,c_3 ,c_4 etc. Thus

c_1 = parity(1, 3, 5, 7, 9, 11 and so on)

c_2 = parity(2, 3, 6, 7, 10, 11 and so on)

c_3 = parity(4-7, 12-15, 20-23 and so on)

Step 4 − Error detection and correction

The decimal equivalent of the parity bits binary values is calculated. If it is 0, there is no error. Otherwise, the decimal value gives the bit position which has error. For example, if $c_1c_2c_3c_4$ = 1001, it implies that the data bit at position 9, decimal equivalent of 1001, has error. The bit is flipped to get the correct message.

1.5.1 Parity code

The parity code is used for the purpose of detecting errors during the transmission of binary information. The parity code is a bit that is included with the binary data to be transmitted.

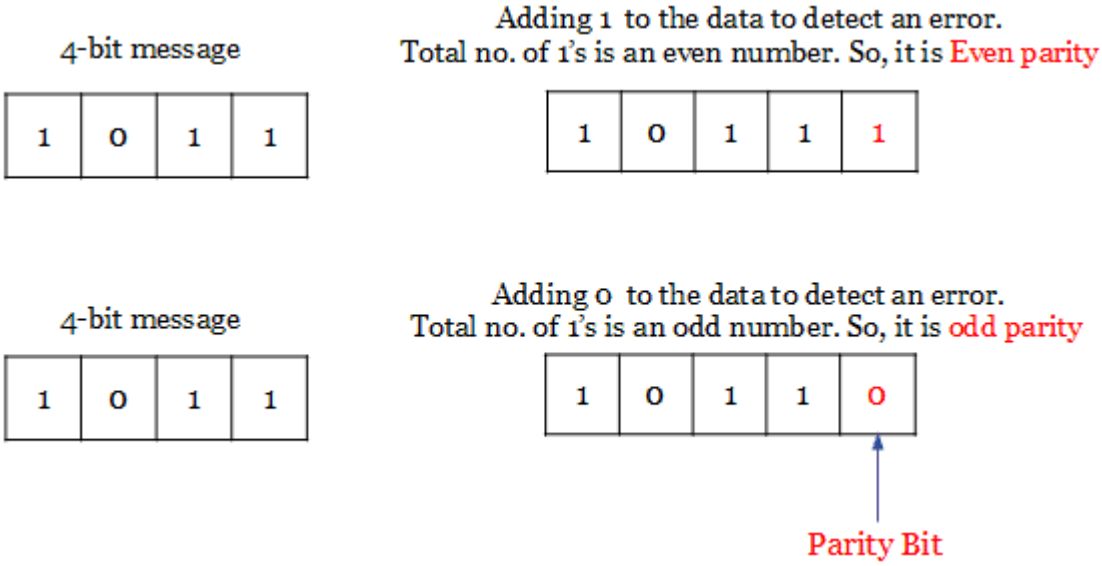
The inclusion of a parity bit will make the number of 1's either odd or even. Based on the number of 1's in the transmitted data, the parity code is of two types.

- Even parity code
- Odd parity code

In even parity, the added parity bit will make the total number of 1's an even number. If the added parity bit make the total number of 1's as odd number, such parity code is said to be odd parity code.

Let us consider the 4-bit message(1011) to be transmitted. Adding 1 to the message will make the total number of 1's in the message to be an even number. Hence it is called as even parity.

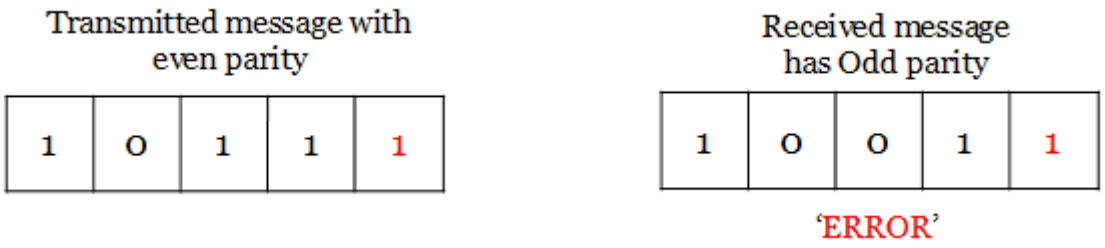
For the same message, adding 0 with the transmitted message will make the total number of 1's to be an odd number. Hence it is called as odd parity. It is shown in the example below.



The following table shows the some of the 4-bit messages to be transmitted along with the parity bits. The bits in red color are the parity bits.

4-bit message	Message with Odd parity	Message with Even Parity
0000	0000**1**	0000**0**
0001	0001**0**	0001**1**
0010	0010**0**	0010**1**
0011	0011**1**	0011**0**
0100	0100**0**	0100**1**
0101	0101**1**	0101**0**
0110	0110**1**	0110**0**
0111	0111**0**	0111**1**

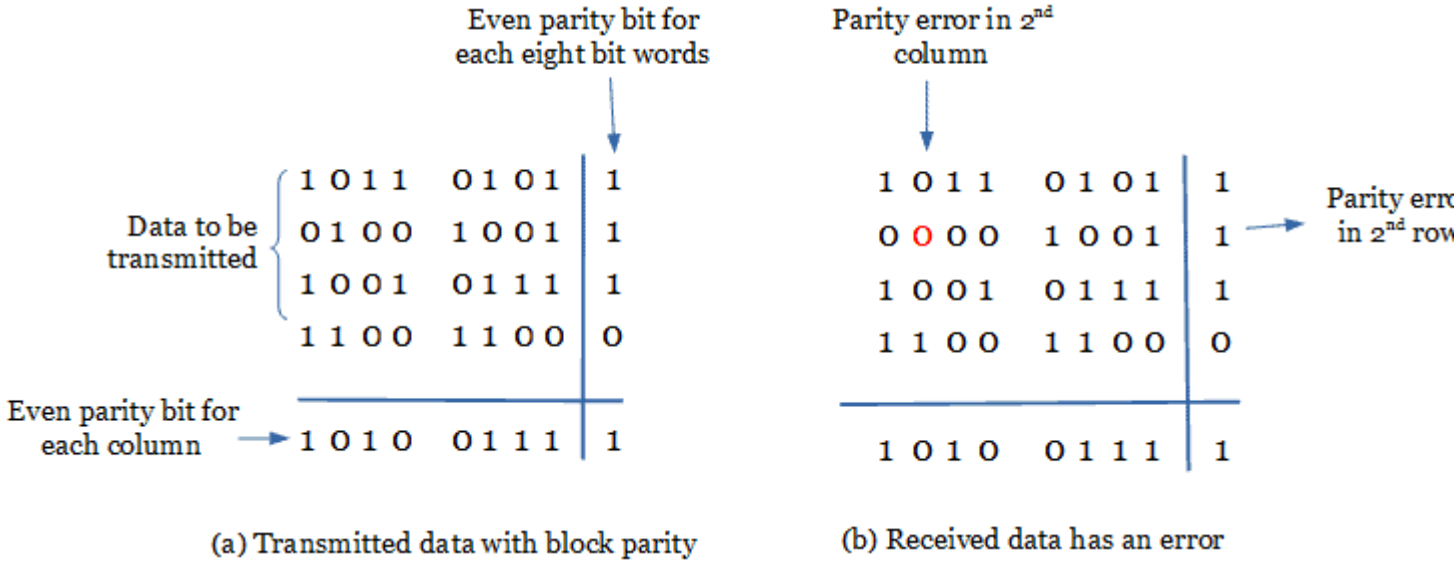
On the receiver side, if the received data is other than the sent data, then it is an error. If the sent date is even parity code and the received data is odd parity, then there is an error.



So, both even and odd parity codes are used only for the detection of error and not for the correction in the transmitted data. Even parity is commonly used and it has almost become a convention.

Block Parity

When several binary words are transmitted and received at a time, then such information is regarded as a block of data, having rows and columns. For example, let us consider four eight-bit words, which are to be transmitted, forms a 4 x 8 block. Parity bits are assigned to both rows and columns.



(a) Transmitted data with block parity

(b) Received data has an error

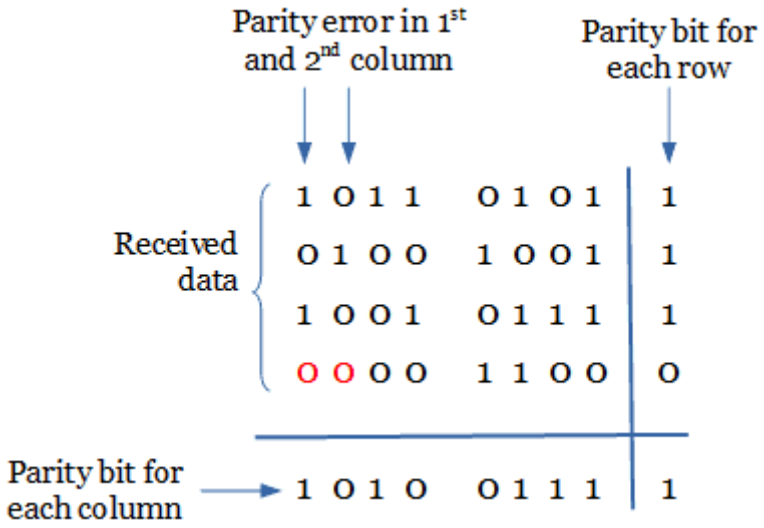
As shown above, even parity is given to each eight-bit word and for each column. In this case, the parity bit is called block parity. The block of data is transmitted from the transmitter side.

Upon analyzing the received block of data(b), the first row has no error as the even parity is maintained. In the second row, the even parity is changed to odd parity. So there is an error in the 2nd row. Even parity is maintained in the third and fourth row and thus, there is no error.

To find out the error bit in the second row, let's check the column-wise parity. If you do so, in the 2nd column, you can notice the even parity is changed to odd parity. So there is an error in the second column. In all other columns, there is no change in the even parity and hence no error in other columns.

The bit, which is at the intersection of the 2nd row and 2nd column is an error bit. In the above illustration, the error bit is marked in red color. Since the detected error is a single bit, we can change the bit 0 to 1. Thus in block parity, detection and correction of an error are possible with the parity codes.

Now, take a look at the following received message.



In the above figure, you can observe that there is no parity bit error for each row. But for the 1st and 2nd column the even parity is changed to odd parity, which is an error.

Thus the error is seen in the 1st and 2nd column. In this case, it is possible only to detect the error. It is not possible to correct the error as there is no information revealing the row where the errors occurred.

1.1 REVIEW OF NUMBER SYSTEMS

Many number systems are in use in digital technology. The most common are the decimal, binary, octal, and hexadecimal systems. The decimal system is clearly the most familiar to us because it is tools that we use every day.

Types of Number Systems are

1Decimal Number system

2 Binary Number system

3 Octal Number system

4 Hexadecimal Number system

Decimal	Binary	Octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Numbering Systems		
System	Base	Digits
Binary	2	0 1
Octal	8	0 1 2 3 4 5 6 7
Decimal	10	0 1 2 3 4 5 6 7 8 9
Hexadecimal	16	0 1 2 3 4 5 6 7 8 9 A B C D E F

Figure 1.1 Number system and their Base value

[Source: https://www.brainkart.com/article/Review-of-Number-Systems_6744/]

Decimal system:

Decimal system is composed of 10 numerals or symbols. These 10 symbols are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Using these symbols as digits of a number, we can express any quantity. The decimal system is also called the base-10 system because it has 10 digits. Even though the decimal system has only 10 symbols, any number of any magnitude can be expressed by using our system of positional weighting.

Example: 3.14_{10} , 52_{10} ,1024_{10}

Binary System:

In the binary system, there are only two symbols or possible digit values, 0 and 1. This base-2 system can be used to represent any quantity that can be represented in decimal or other base system.

In digital systems the information that is being processed is usually presented in binary form. Binary quantities can be represented by any device that has only two operating states or possible conditions. E.g.. A switch is only open or closed. We arbitrarily (as we define them) let an open switch represent binary 0 and a closed switch represent binary 1. Thus we can represent any binary number by using series of switches.

Binary 1: Any voltage between 2V to 5V Binary 0: Any voltage between 0V to 0.8V

Octal System: The octal number system has a base of eight, meaning that it has eight possible digits: 0,1,2,3,4,5,6,7.

Hexadecimal System: The hexadecimal system uses base 16. Thus, it has 16 possible digit symbols. It uses the digits 0 through 9 plus the letters A, B, C, D, E, and F as the 16 digit symbols.

1.4 THEOREMS OF BOOLEAN ALGEBRA

The theorems of Boolean algebra can be used to simplify many a complex Boolean expression and also to transform the given expression into a more useful and meaningful equivalent expression. The theorems are presented as pairs, with the two theorems in a given pair being the dual of each other. These theorems can be very easily verified by the method of 'perfect induction'. According to this method, the validity of the expression is tested for all possible combinations of values of the variables involved. Also, since the validity of the theorem is based on its being true for all possible combinations of values of variables, there is no reason why a variable cannot be replaced with its complement, or vice versa, without disturbing the validity. Another important point is that, if a given expression is valid, its dual will also be valid

Theorem 1 (Operations with '0' and '1')

(a) 0.X = 0 and (b) 1+X= 1

Where X is not necessarily a single variable – it could be a term or even a large expression.

Theorem 1(a) can be proved by substituting all possible values of X, that is, 0 and 1, into the given expression and checking whether the LHS equals the RHS:

• For X = 0, LHS = 0.X = 0.0 = 0 = RHS.

• For X= 1, LHS = 0.1 = 0 = RHS.

Thus, 0.X =0 irrespective of the value of X, and hence the proof.

Theorem 1(b) can be proved in a similar manner. In general, according to theorem 1,

0. (Boolean expression) = 0 and 1+ (Boolean expression) =1.

For example: 0. (A.B+B.C +C.D) = 0 and 1+ (A.B+B.C +C.D) = 1, where A, B and C are Boolean variables.

Theorem 2 (Operations with '0' and '1')

(a) 1.X = X and (b) 0+X = X

where X could be a variable, a term or even a large expression. According to this theorem, ANDing a Boolean expression to '1' or ORing '0' to it makes no difference to the expression:

• For X = 0, LHS = 1.0 = 0 = RHS.

• For X = 1, LHS = 1.1 = 1 = RHS. Also,

1. (Boolean expression) = Boolean expression and 0 + (Boolean expression) = Boolean expression.

For example,

1.(A+B.C +C.D) = 0+(A+B.C +C.D) = A+B.C +C.D

Theorem 3 (Idempotent or Identity Laws)

(a) X.X.X……X = X and (b) X+X+X +⋯+X = X

Theorems 3(a) and (b) are known by the name of idempotent laws, also known as identity laws.

Theorem 3(a) is a direct outcome of an AND gate operation, whereas theorem 3(b) represents an OR gate operation when all the inputs of the gate have been tied together. The scope of idempotent laws can be expanded further by considering X to be a term or

an expression. For example, let us apply idempotent laws to simplify the following Boolean expression:

$$(A.\overline{B}.\overline{B}+C.C).(A.\overline{B}.\overline{B}+A.\overline{B}+C.C)=(A.\overline{B}+C).(A.\overline{B}+A.\overline{B}+C)$$
$$=(A.\overline{B}+C).(A.\overline{B}+C)=A.\overline{B}+C$$

Theorem 4 (Complementation Law)

(a) X_X = 0 and (b) X+X = 1

According to this theorem, in general, any Boolean expression when ANDed to its complement yields a '0' and when ORed to its complement yields a'1', irrespective of the complexity of the expression:

- For $X=0, \overline{X}=1$. Therefore, $X.\overline{X}=0.1=0$.
- For $X=1, \overline{X}=0$. Therefore, $X.\overline{X}=1.0=0$.

Hence, theorem 4(a) is proved. Since theorem 4(b) is the dual of theorem 4(a), its proof is implied.

The example below further illustrates the application of complementation laws:

$$(A+B.C)(\overline{A+B.C})=0 \quad \text{and} \quad (A+B.C)+(\overline{A+B.C})=1$$

Theorem 5 (Commutative property)

Mathematical identity, called a ‖property‖ or a ‖law,‖ describes how differing variables relate to each other in a system of numbers. One of these properties is known as the commutative property, and it applies equally to addition and multiplication. In essence, the commutative property tells us we can reverse the order of variables that are either added together or multiplied together without changing the truth of the expression:

Commutative property of addition

A + B = B + A

Commutative property of multiplication

AB = BA

Theorem 6 (Associative Property)

The Associative Property, again applying equally well to addition and multiplication. This property tells us we can associate groups of added or multiplied variables together with parentheses without altering the truth of the equations.

Associative property of addition

A + (B + C) = (A + B) + C

Associative property of multiplication

A (BC) = (AB) C

Theorem 7 (Distributive Property)

The Distributive Property, illustrating how to expand a Boolean expression formed by the product of a sum, and in reverse shows us how terms may be factored out of Boolean sums-of-products:

Distributive property

A (B + C) = AB + AC

Theorem 8 (Absorption Law or Redundancy Law)

(a) X+X.Y = X and (b) X.(X+Y) = X

The proof of absorption law is straightforward:

X+X.Y = X. (1+Y) = X.1 = X

Theorem 8(b) is the dual of theorem 8(a) and hence stands proved.

The crux of this simplification theorem is that, if a smaller term appears in a larger term, then the larger term is redundant. The following examples further illustrate the underlying concept:

$$A + A.\overline{B} + A.\overline{B}.\overline{C} + A.\overline{B}.C + \overline{C}.B.A = A$$

and

$$(\overline{A} + B + \overline{C}).(\overline{A} + B).(C + B + \overline{A}) = \overline{A} + B$$

Demorgan's Theorem

De-Morgan was a great logician and mathematician. He had contributed much to logic. Among his contribution the following two theorems are important

De-Morgan‘s First Theorem

It States that ―The complement of the sum of the variables is equal to the product of the complement of each variable‖. This theorem may be expressed by the following Boolean expression.

$$\overline{A + B} = \overline{A} . \overline{B}$$

De-Morgan‘s Second Theorem

It states that the ―Complement of the product of variables is equal to the sum of complements of each individual variables‖. Boolean expression for this theorem is

$$\overline{A.B} \quad = \quad \overline{A}+\overline{B}$$

A	B	A . B	$\overline{A.B}$ L.H.S	$\overline{A}$	$\overline{B}$	$\overline{A}+\overline{B}$ R.H.S
0	0	0	1	1	1	1
1	0	0	1	0	1	1
0	1	0	1	1	0	1
1	1	1	0	0	0	0

Table 1.4 De-Morgan's Second Theorem

[Source https://www.brainkart.com/article//]

1.9 TRANSISTOR-TRANSISTOR LOGIC (TTL)

Transistor-Transistor Logic belongs to the digital logic family. It consists of transistors at both input and output side, diodes and few resistors. Unlike Resistor-transistor logic and Diode transistor logic, both the logic function and amplifying function are performed by the transistors.

The TTL integrated circuits are very popular in different applications including computer controls, consumer electronics, industrial control systems, etc.

There are various subfamilies in transistor-transistor logic, which includes standard TTL, Low power TTL, Schottky TTL, Advanced Schottky TTL, High power TTL, fast TTL, etc. In this section, you will learn about the circuit and operation of standard TTL and different output configurations of TTL.

The following figure shows the circuit diagram of the 2-input TTL NAND gate. It has four transistors Q_1, Q_2, Q_3 and Q_4. Transistor Q_1 has 2-inputs on the emitter side. Transistor Q_3 and Q_4 form the output side, called Totem pole output.

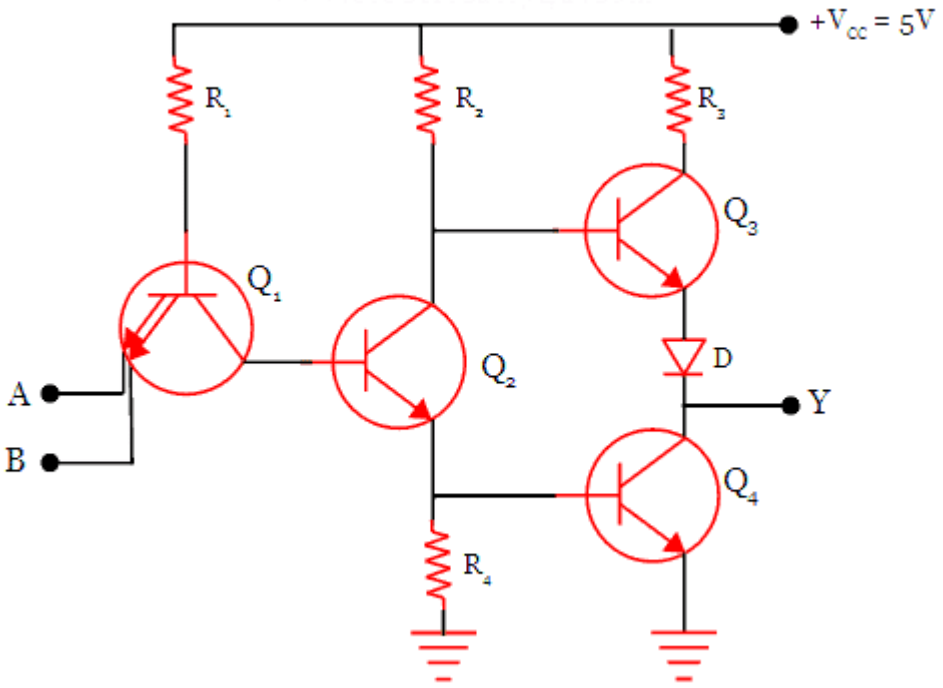


Figure 1.9.1 2-input TTL NAND gate

[Source: https://www.electrically4u.com/transistor-transistor-logic/]

When both inputs A and B are low, both the diodes are forward biased. So the current due to the supply voltage $+V_{CC} = 5$ V will go to the ground through R_1 and the two diodes D_A and D_B.

The supply voltage gets dropped in the resistor R_1 and it will not be sufficient to turn ON the transistor Q_2. With Q_2 open, the transistor Q_4 will also cut off. But the transistor Q_3 is pulled high. Since Q_3 is an emitter follower, the output at the terminal will also be HIGH, which is at logic 1.

When any one input, either A or B is low, the diode with low input will be forward biased. The same operation will take place as explained above. In this case, the output will be HIGH.

When both the inputs A and B are high, both the diodes at the emitter-base junction will be reverse biased. The diode D_C at the collector-base junction is forward biased. It will turn on the transistor Q_2. With Q_2 turned ON, transistor Q_4 will also be turned ON.

Both the transistors at the output side will conduct and so the output at terminal will have LOW value, which is considered as logic 0.

1.9.1 Output configuration of TTL

There are three different output configurations in transistor-transistor logic

1. Totem-pole output
2. open collector output
3. Tri-state gate output

Totem-pole output

In the circuit shown below, the shaded portion shows the totem-pole output. Transistor Q3, Q4, diode D and current limiting resistor R3 form the totem-pole output configuration of TTL.

There are a few advantages of using this configuration. When the output switches from LOW to a HIGH state, the output transistor Q4 goes from saturation to cut off. During this transition, the load capacitance across Q3 charges exponentially from low to a high voltage level.

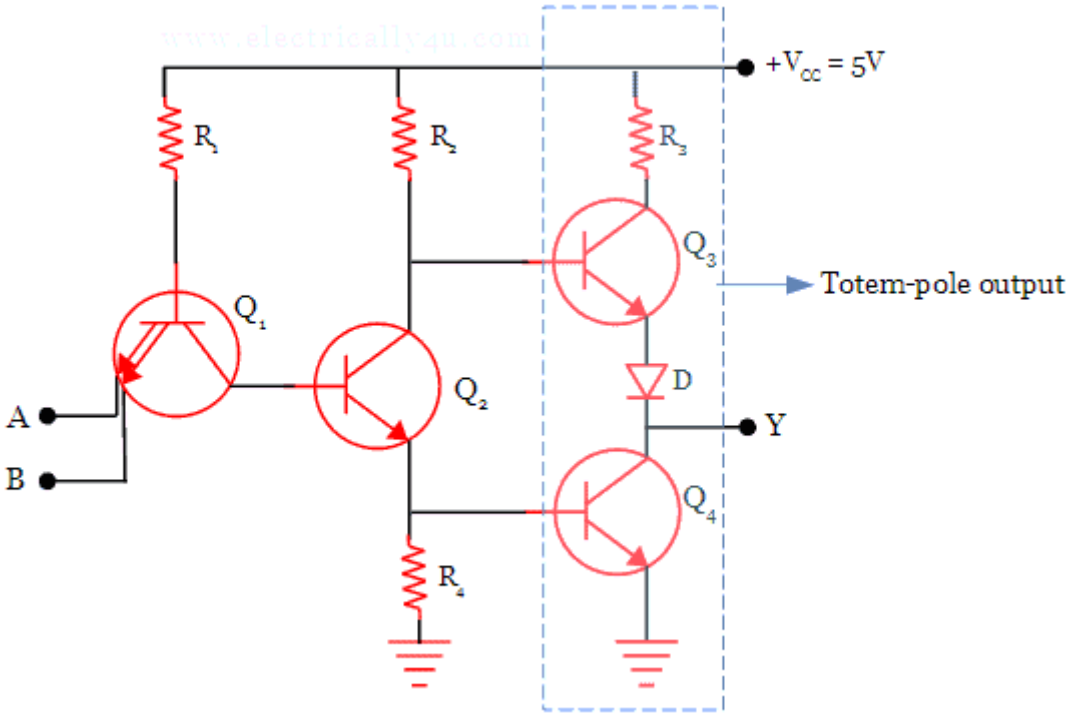


Figure 1.9.2 Totem-pole output configuration in Transistor-Transistor Logic

[Source: https://www.electrically4u.com/transistor-transistor-logic/]

Due to the low output impedance of both transistors Q3 and Q4, the output voltage can change quickly from LOW to HIGH value as the capacitance charge and discharge quickly.

Open collector output

The open-collector output configuration of Transistor-Transistor Logic is shown in the below figure. In this configuration, the transistor Q3 and pull-up resistor is eliminated. Instead, external pull up resistor for proper operation as shown in the figure.

The output is taken from the open collector terminal of Q4. When transistor Q4 is OFF, the output Y will be HIGH and when Q4 is ON, the output will be LOW.

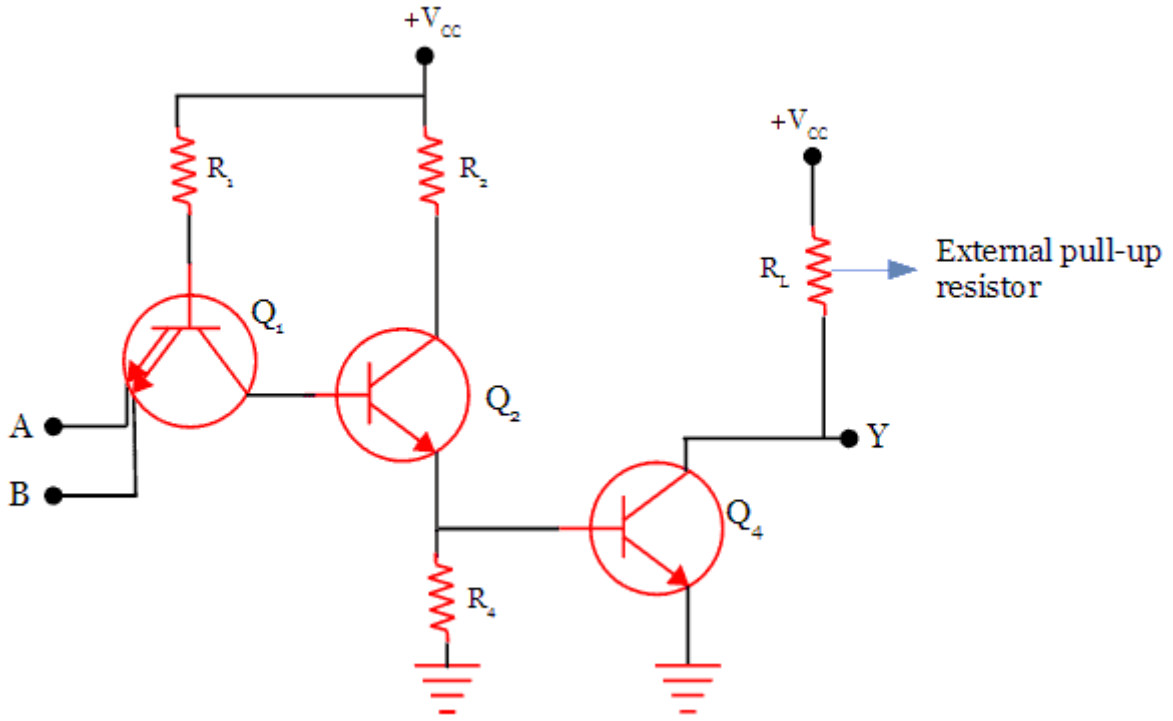


Figure 1.9.3 Open-Collector Output Configuration Of Transistor-Transistor Logic

[Source: https://www.electrically4u.com/transistor-transistor-logic/]

Tri-state gate output

While operating the transistor in this output configuration, it is possible to attain high speed. Three output state is possible: HIGH, LOW and high impedance.

When the transistor Q3 is ON, the output at terminal Y is HIGH. The output is LOW when the transistor Q4 is turned ON. The first and second states are the normal operation of TTL. In the third state, both the transistors Q3 and Q4 are turned OFF, which results in neither LOW nor HIGH output.

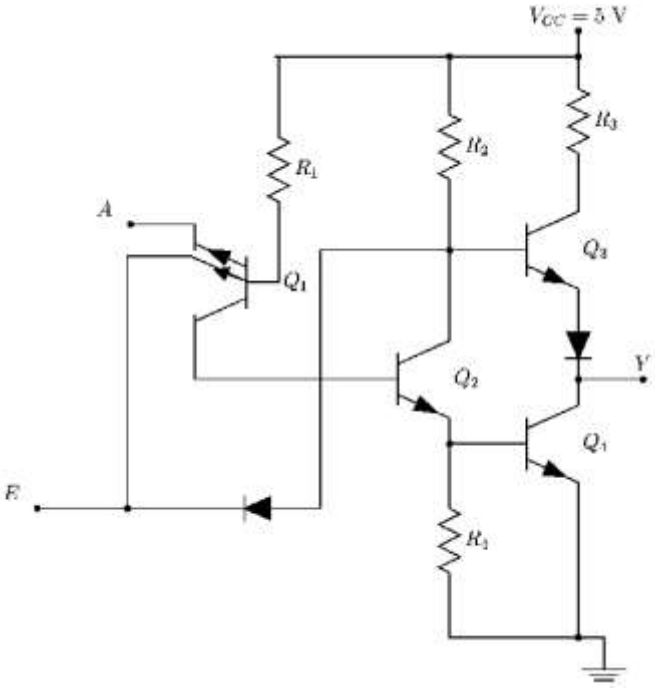


Figure 1.9.4 Tri-state gate output

[Source: http://users.cecs.anu.edu.au/]

Advantages

High-speed operation. The propagation delay is around 10 ms, which is fast compared to DTL and RTL logic devices.

Less power dissipation compared to DTL and RTL.

Low cost

Better fan-out

Reliable operation for noise.