Digital Electronics

Contents

1 Analog vs Digital

Electronic circuits can be divided in two big categories: analog and digital. Analog signals have a continous range of values. Up to now we have been dealing with analog circuits (RC filters, rectifiers, op-amps...) Digital signals, have a discrete number of values (states). There are some situations in which the input signals is discrete by nature: pass of a particle, press a keyboard, etc... In these cases the use of digital circuits is desirable. Although the number of discrete states of a signal can be big, in general when we are talking about digital signals we assume only two discrete values or states:

- HIGH or '1' or TRUE
- LOW or '0' or FALSE

1.1 Digital signal representation

The states '0' and '1' are represented as voltage or current pulses that propagate down in a cable or free space. The voltage or current levels should follow some rules, defining some minimal and maximal values for the representation of '0' and '1'. In figure 1 are shown the voltage range definitions:

- $V_L^{min} < LOW < V_L^{max}$
- $V_H^{min} < HIGH < V_H^{max}$
- The values in the range $[V_L^{max}, V_H^{min}]$ are undefined.
- V_{th} is the typical logic threshold between '0' and '1'.

In the table 1 are the values of different logic levels. (Beware, the reference is a bit old, update when possible).

		V_{cc}	V_L^{min}	V_L^{typ}	V_L^{max}	V_{th}	V_H^{min}	V_H^{typ}	V_H^{max}	NI
TTL	inp	+5V	-0.4V		+0.8V	+1.2 V	+2.0V		+5.4V	0.4V
			+0.0V	+0.2V	+0.4V		+2.4V	+3.3V	+5.0V	
	out									
CMOS	inp	+5V	-0.2V		+1.5V	+2.5 V	+3.3V		+5.4V	1.1V
			+0.0V	+0.2V	+0.4V		+4.6V	+4.9V	+5.0V	
	out									
ECL	inp	-5.2V	-5.0V		-1.5V	-1.3V	-1.1V		-0.0V	0.1V
			-1.9V	-1.8V	-1.6V		-1.0V	-0.9V	-0.8V	
	out									
NIM	inp	$\pm 12V$	-2.0V		+1.5V		+3.0V		+12.0V	0.5V
(slow)	out	$\pm 24V$	-2.0V	+0.0V	+1.0V		+4.0V	+6.0V	+12.0V	
NIM	inp	$\pm 12V$	-4.0mA		+20mA		-12mA		-36mA	2.0mAV
(fast)	out	$\pm 24V$	-1.0mA	+0.0mA	+1mA		-14mA		-18mA	

Table 1: Digital standards

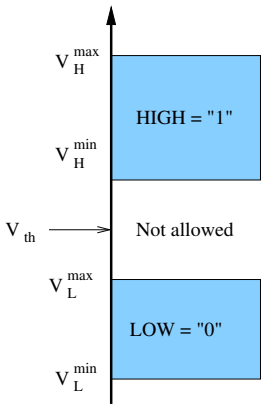


Figure 1: Voltage ranges

1.2 Advantages of digital signals

Basically we can think about two situations in which we should use digital electronics: 1) when the problem to solve leads to a discrete number of states, we have already seen some examples as the pass of a particle, and 2) when there is any signal transmission. Transmission of information in analog form degrades the signal irreversibly because of noise pickup, some of which is not reducible (ex. radio station) while as we have seen in previous section digital levels are defined in a wide range of voltage, in such a way that any noise that disturb the digital level can be recovered as far as the final signal still stays in the level definition range. We can define noise inmunity as the miinal voltage needed to mistake the logical level.

1.3 Combinatorial and sequential logic

In digital electronics digital outputs are generated from digital inputs. If the output o the logic circuit depends only on the present input values, we refer to these circuits as combinatorial logic circuits. We can said that combinatorial circuits have no memory and they are based in the so called logical gates.

On the other hand, if the output of the circuit depends on present as well as past input values, we will say that these circuits have memory and we refer to them as sequential logic circuits. Such circuits are more complicated and require the presence of a clock signal to regulate the response of the circuit to new circuits.

2 Base Logical Operations

In digital electronics we are treating basically with logics. Even in the definition of the states we have already pinted out defining the two states as TRUE and FALSE. In logics we can define three basic operations:

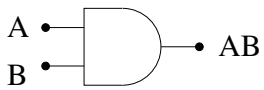
- NOT ($\overline{x}$). This operation changes a logic level by its complemetary
- AND ($\times$). This operation produces a HIGH logic level only if all inputs are HIGH

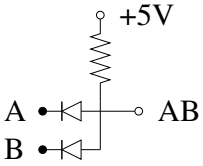
- OR ($+$). This operations produces a HIGH logic level if any of the inputs is HIGH.

2.1 Logical gates

Basic to digital electronics are switching gates. They control the flow of information which is in the form of pulses. If we list all possible inputs to a circuit and the corresponding outputs we obtain what is called a truth table. In general if in any input or output there is a small circle means that the input or output is inverted.

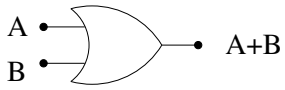
2.1.1 AND Gate

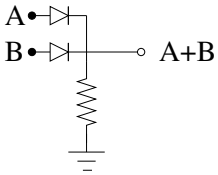




A	B	AB
0	0	0
1	0	0
0	1	0
1	1	1

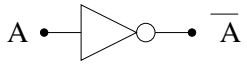
2.1.2 OR Gate

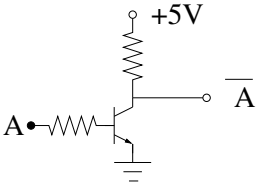




A	B	A+B
0	0	0
1	0	1
0	1	1
1	1	1

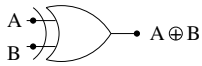
2.1.3 NOT Gate

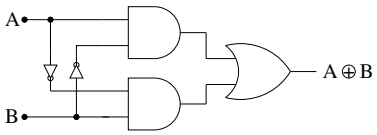




A	$\overline{A}$
0	1
1	0

2.1.4 XOR Gate





A	B	A+B
0	0	0
1	0	1
0	1	1
1	1	0

2.2 Boolean Algebra

The algebra that can be done with binary states and the above operations is known as boolean algebra. We are going to present this algebra as a series of axioms:

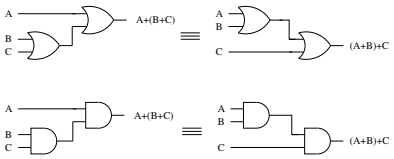
- Commutative composition laws:

$$A + B = B + A \qquad AB = BA$$

- Associative laws

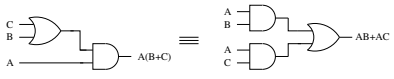
$$A + (B + C) = (A + B) + C$$

$$A(BC) = (AB)C$$



- Distributive law

$$A(B + C) = AB + AC$$



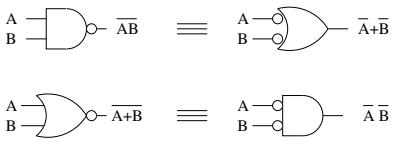
- Boolean Algebra rules

1. $A + 0 = A$	7. $A \cdot A = A$
2. $A + 1 = 1$	8. $A \cdot \overline{A} = 0$
3. $A \cdot 0 = 0$	9. $\overline{\overline{A}} = A$
4. $A \cdot 1 = A$	10. $A + AB = A$
5. $A + A = A$	11. $A + \overline{A}B = A + B$
6. $A + \overline{A} = 1$	12. $(A + B)(A + C) = A + BC$

- DeMorgan laws.

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

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



3 Numerical Systems

3.1 Binary numbers

Digital electronics leads directly to the use of binary numbers. Then quantities are going to be represented as binary numbers:

$$abc_2 = a \times 2^2 + b \times 2^1 + c \times 2^0$$

Each digit is known as a bit and can take only two values 0 and 1. The left most bit is the highest-order bit and represent the most significant bit (MSB) while the lowest-order bit is the least significant bit (LSB). Some useful definitions are:

word a binary number consisting of an arbitrary number of bits

nibble 4-bit word (one hexadecimal digit) 16 values.

byte an 8-bit word 256 values.

3.1.1 1-complement

The 1-complement of a binary number is obtained just changing each 0 to 1 and each 1 to 0:

Binary number	1	0	1	1	1	0	1	0
	↓	↓	↓	↓	↓	↓	↓	↓
1-complement	0	1	0	0	0	1	0	1

3.1.2 2-complement

The 2-complement of a binary number is obtained adding 1 to the 1-complement of this number:

2-complement = 1-complement+1

Binary number		1	0	1	1	1	0	1	0
1-complement		0	1	0	0	0	1	0	1
	+								1
2-complement		0	1	0	0	0	1	1	0

There is a simple recipe to obtain the 2-complement:

1. Begining with the LSB, just write down bits as they are moving to left till the first 1, including it.
2. Substitute the rest of bits by their 1-complement.

3.1.3 Signed numbers

In a signed number, the left most bit is the so called sign bit: 0=positive number 1=negative number.

Sign-value notation In this notation the left-most bit is the sign bit and the others are used to represent the absolut value notation.

1-complement In this notation the positive numbers have the same representation as the sign-value notation, and the negative numbers are obtained taking the 1-complement of the positive correspondants.

2-complement The positive numbers have the same representation as the sign-value notation, and the negative numbers are obtained taking the 1-complement of the positive correspondants.

Positive	All	00011001	(+25)
Negative	Sign-value	10011001	
	1-complement	11100110	
	2-complement	11100111	

3.2 Hexadecimal numbers

The hexadecimal system has a base 16, that is, it is composed by 16 numbers. It can be represented by 4 bits (a nibble). Usually the hexadecimal numbers are represented with a 0x before the number:

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

3.3 Binary Decimal Code (BDC)

It is a way of represent the decimal digits in a binary code. It is as simple as:

Decimal digit	0	1	2	3	4	5	6	7	8	9
BCD code	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

$$135 \rightarrow 0001\ 0011\ 0101$$

3.4 Code Gray

Not valid for arithmetic calcualtions. It allows to pass from a number to the following just changing 1 bit. The Gray code can be defined for all possible numbers of bits.

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

3.4.1 Binary to Gray

$$\begin{array}{ccccccccc} 1 & -+\rightarrow & 0 & -+\rightarrow & 1 & -+\rightarrow & 1 & -+\rightarrow & 0 \quad \text{Binary} \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow \\ 1 & & 1 & & 1 & & 0 & & 1 \quad \text{Gray} \end{array}$$

3.4.2 Gray to Binary

$$\begin{array}{ccccccccc} 1 & & 1 & & 1 & & 0 & & 1 \quad \text{Gray} \\ \downarrow & + \nearrow \downarrow & & + \nearrow \downarrow & & + \nearrow \downarrow & & + \nearrow \downarrow & \\ 1 & & 1 & & 1 & & 0 & & 1 \quad \text{Binary} \end{array}$$

3.5 ASCII code

3.6 Transmission integrity

3.6.1 Parity

3.6.2 Cycle Redundancy Check (CRC)

4 Combinatorial Circuits

Combinatorial circuits are those circuits which output depends only on the input values at that instant. Another way to say that is that combinatorial circuits have no memory.

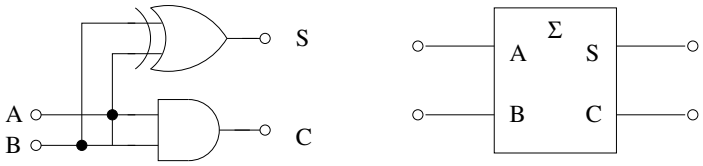
4.1 Adder circuits

Addition of binary numbers is the basic of binary arithmetics.

4.1.1 Half Adder

A half adder is a dispositive with two inputs and two outputs: A sum digit and a carry bit. Truth table and a possible implementation is shown down. These circuits are useful as far as we need only to add 1 bit values.

A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

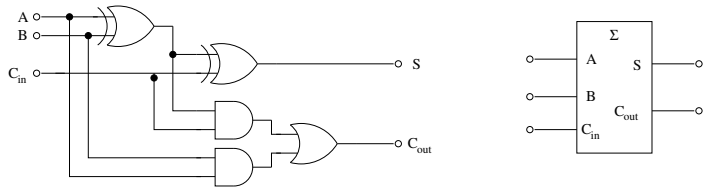


4.1.2 Full Adder

A full adder takes two bits and a carry bit as input and provides two output bits. Truth table and a possible implementation are shown down. If you look carefully, the full adder can be

built with two half adders, what gives sense to the name choice.

A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



4.1.3 Parallel Adder

The conexion of full adders in cascade forms a parallel adder. With parallel adders we can add multibit words. MSI (Medium Scale Integration) chips with 4-bit parallel adders are 74xx83 and 74xx283. In figure 2, is sketched how is built a 4-bit adder with full adders and how to connect two 4-bit adders to create a 8-bit adder.

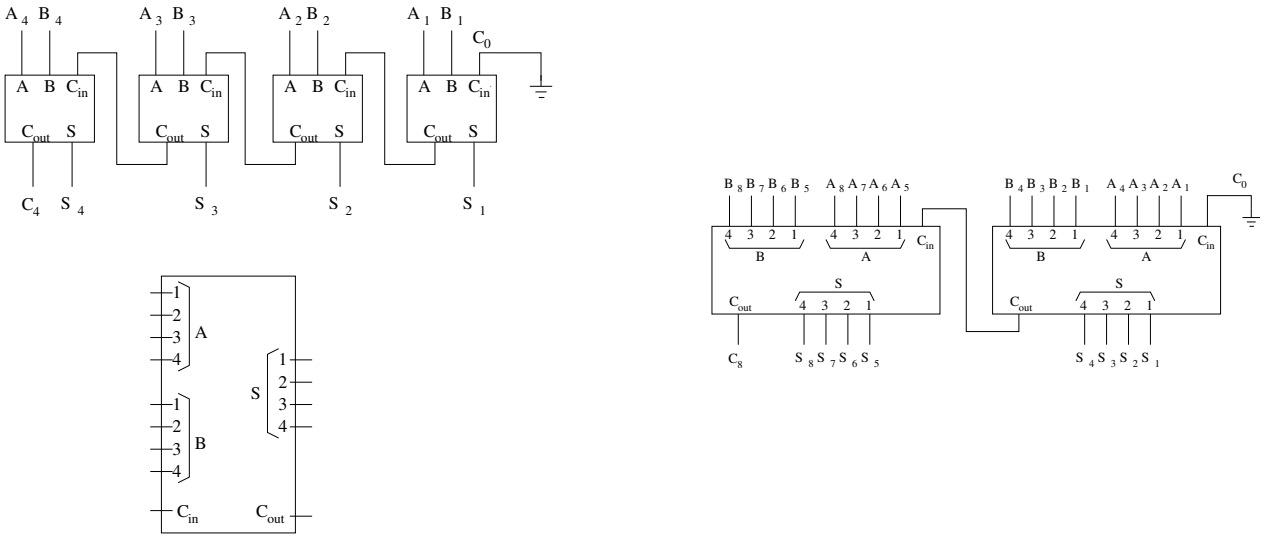


Figure 2: 4-bit and 8-bit parallel adders

4.2 Decoders

Decoders are circuits that detect an specific combination of bits in the input generating an output acording with the input. So if there are n inputs, the maximal number of outputs can be up to 2^n. An example of the usefulness of such devices is shown in figure **??**, where a decoder is used to chose the IO ports of a processor. Please, note that this is a simplified sketch on how is actually the real IO addres implementation inside your computer!!!!

74xx154 is a MSI chip with 4 input bins, and 16 output bins, like the o ne showed in the example.

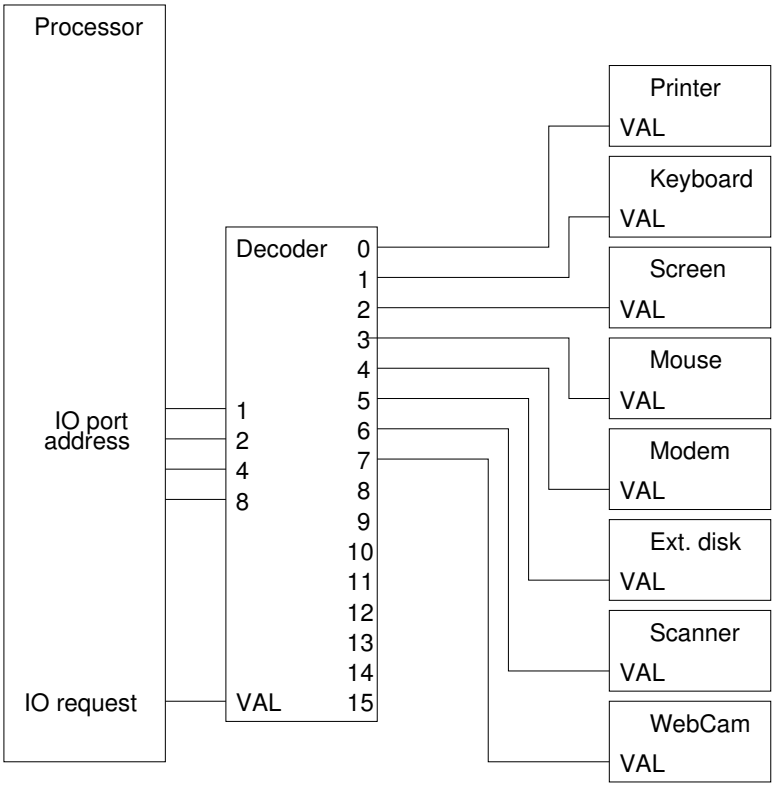


Figure 3: 4-bit and 8-bit parallel adders

Another example is the DCB-7 segments decoder (74xx47) that interfaces DCB code with a 7 segments display in order to visualize the code. This decoder is extensively use in quite a lot of human interfaces.

4.3 Coders

A coder performs the oposite action of a decoder, given up to 2^n input signals, they can be coded in n bits. A typical example of an application of a coder is for instance a keyboard. Once we touch a key in the keyboard, we expect that the signal is coded to some code that can be used by the computer (DCB,ASCII, etc..).

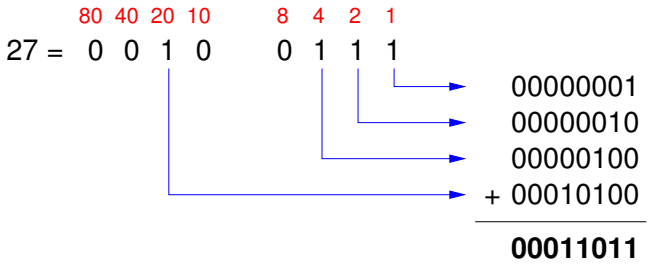
4.4 Code converters

These are circuits that translate a code into another. Usually this converters are programmed in logic arrays.

4.4.1 DCB-binary conversion

A simple method to convert DCB code into binary code is using adder circuits.

1. The value of each bit in a BCD number is reoresented by a binary number
2. All binary representation with weight equal to 1 are added
3. The result of the addition is the binary equivalent of the BCD number



The inverse operation can also easily figured out.

There are two chips 74184 and 74185 that converts DCB to binary and viceversa. In figure 4 are shown how to convert 2 digits DCB numbers in binary and 8 bit binary numbers in 3 digit DCB numbers.

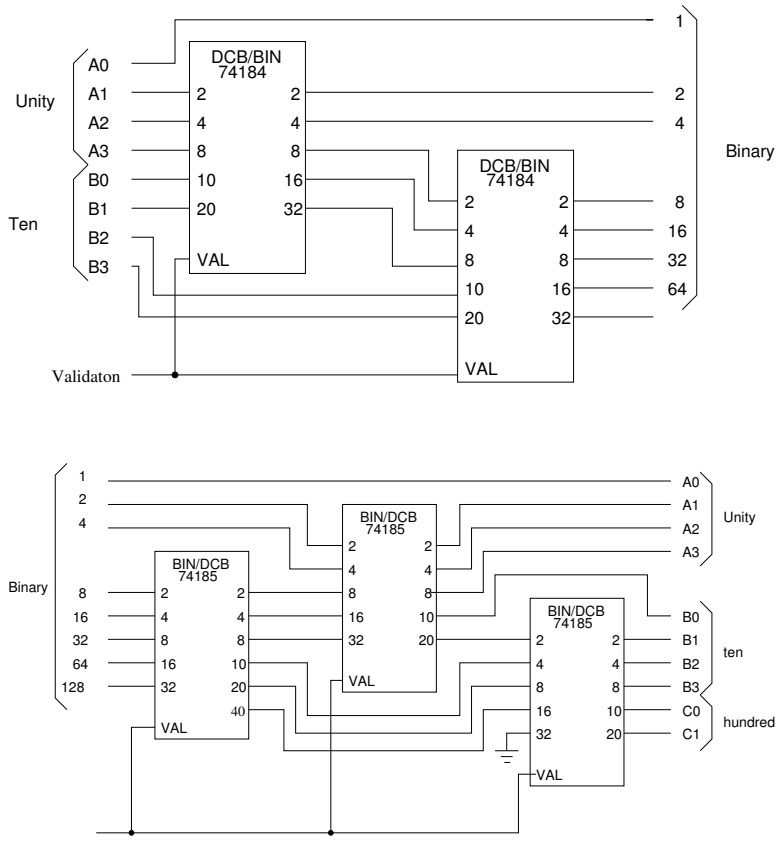
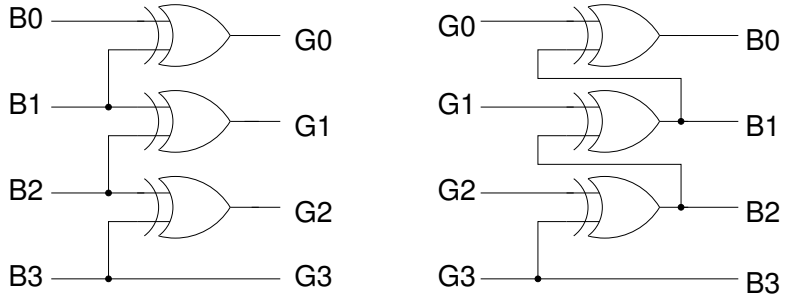


Figure 4: 2 digit BCD to binary converter (74184 circuit) and 8 bit binary to 3 digit BCD converter (74185 circuit)

4.4.2 Binary-Grey conversion

Simple circuits used to convert binary to Grey and viceversa are. Compare with the rules given in the Binary given in the section describing Gray code.



4.5 Multiplexers

A multiplexer is a device that is able to select 1 out of N input data sources and to transmit the selected data to a single information channel. We can see it also like a serializer in the sense that serializes the parallel information that cames in N parallel lines. Besides the N input lines multiplexers have also n inputs ($N = 2^n$) that provides the information of the selected line. An enable line can also be present. In figure 5 is shown a possible and simple implementation of 4-to-1 multiplexer. MSI chip that act as multiplexers are 74xx250 (16-to-1), 74xx151 (8-to-1), 74xx253 (4-to-1), 74xx157 (2-to-1).

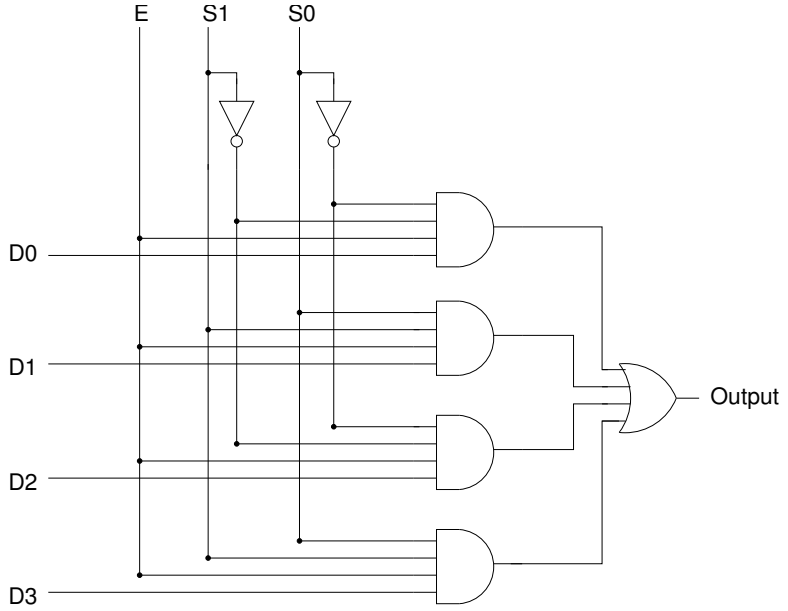


Figure 5: 4-to-1 multiplexer

4.6 Demultiplexers

A demultiplexer is a system for transmitting a binary signal (serial data) on one of N lines, the particular line being selected by means of an address. The only difference between

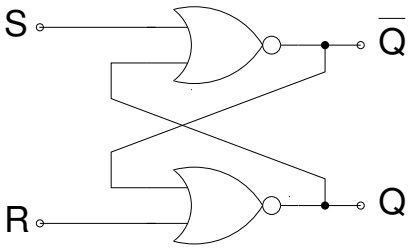
a decoder and a multiplexer is the presence of an enable signal. MSI demultiplexers are 74xx154 (4-to-16, 4 addres lines and 16 outputs), 74xx139 (2-to-4), 74xx138 (3-to-8).

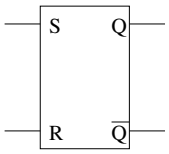
5 Sequential Circuits

The potential of digital circuits increases when we add *memory* to logic gates. Sequential circuits are those which output values depends not only on the input values at that instant, but also on the input values in the past. The sequential environment is then more complicated as present and past inputs have to be made available in an ordered way. The main way to obtain this is to regulate the circuits with a clock signal, to synchronize the different elements of the circuit.

5.1 Bistables

The basic brick in sequential logic circuits are the so called bistables. The name of this circuits cames because the existance of two stable states. A possible implementation of a bistable is shown in the figure 6





S	R	Q_n	
0	0	Q_{n-1}	store
1	0	1	set
0	1	0	reset
1	1		not valid

Figure 6: Bistable implementation, symbol and truth table of a SR bistable

The cross-coupling of the outputs assure that the outputs Q and $\bar{Q}$ have opposite states. The equations of the outputs can be written as:

$$Q = \overline{R + \overline{Q}}$$
$$\overline{Q} = \overline{S + Q}$$

Let's analize now all possible states.

	$(Q\bar{Q})_{n-1} = 10$	$(Q\bar{Q})_{n-1} = 01$
$SR = 00$	$Q_n = \overline{0+0} = 1$ $\bar{Q}_n = \overline{0+1} = 1$	$Q_n = \overline{0+1} = 0$ $\bar{Q}_n = \overline{0+0} = 1$
$SR = 10$	$Q_n = \overline{0+0} = 1$ $\bar{Q}_n = \overline{1+1} = 0$	$Q_n = \overline{0+1} = 0 \rightarrow 1$ $\bar{Q}_n = \overline{1+0} = 0 \rightarrow 0$
$SR = 01$	$Q_n = \overline{1+0} = 0 \rightarrow 0$ $\bar{Q}_n = \overline{0+1} = 0 \rightarrow 1$	$Q_n = \overline{1+1} = 0$ $\bar{Q}_n = \overline{0+0} = 1$
$SR = 11$	$Q_n = \overline{1+0} = 1$ $\bar{Q}_n = \overline{1+1} = 1$	$Q_n = \overline{1+1} = 1$ $\bar{Q}_n = \overline{1+0} = 1$

Few comments on the above table, there are two cases, $SR = 10, Q\overline{Q} = 01$ and $SR = 01, Q\overline{Q} = 10$ in which the direct application of the above equations leads to an inestable output configuration ($Q = \overline{Q} = 0$), hence the system becames unstable and one of the outputs flip into stable configuration that fulfills the Q and $\overline{Q}$. On the other hand, in the case of $SR = 11$, in all cases we obtain that $Q\overline{Q} = 11$, in this case, the output becames unstable, and the output oscillates, but it doesn't matter to which configuration goes $Q\overline{Q} = 10$ or 01, the output doesn't fulfill the equation, so the system becomes again unstable and oscillates. This is why this state is not valid. A MSI chip containing SR bistables is the 74xx279

The bistable described above is called SR (Set Reset). An extra validation input can be added to this configuration, see figure 7.

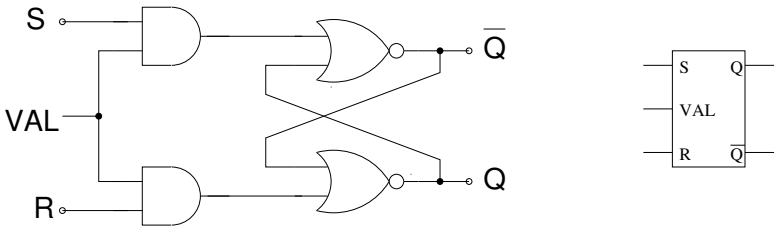
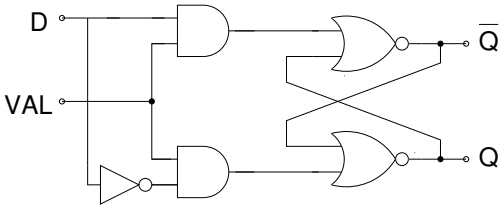


Figure 7: Bistable implementation, symbol and truth table of a SR bistable

Another type of bistable is the so called D bistable. In figure 8 there is shown a possible implementation that is based in the validated bistable described before and has the advantage that the forbidden state $SR = 11$ will never occur. The truth table is shown also. A MSI chip with D bistables is 74xx75.



D	VAL	Q_n
1	1	1
0	1	0
X	0	Q_{n-1}

Figure 8: Bistable implementation, symbol and truth table of a D bistable

5.2 Monostables

A monostable is a circuit that will have only one stable state. Initially a monostable, also called one-shot, is in its stable state and only pass to a quasistable state when is triggered. Then monostable stays in this quasi stable state a certain period of time, returning later to its stable state. This behaviour describe the usefulness of monostable, generate an impulsion once is triggered. The widht of the impulsion is regulated with an RC network. In figure **??** is shown an implementation of a monostable:

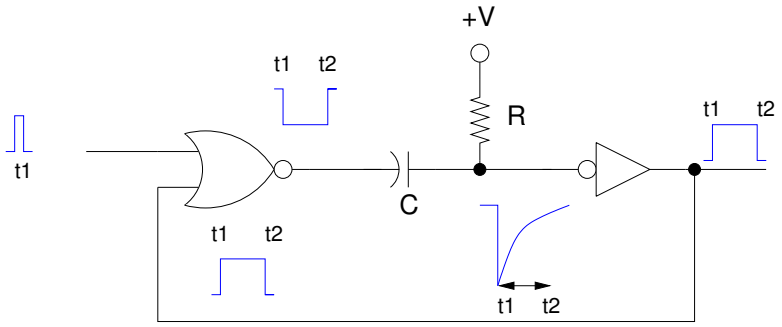


Figure 9: Simplified implementation of a monostable

The RC acts as a delay line, when the trigger signal arrives, the capacitor start to be charged throug the resistance, with a time constant determined by the product RC. During the charging process the output is high, and only when the cacapacitor is charged, the output becames again to one.

Practical monostables based in MSI are 74121 or 74221 (non-retriggerable) and 74122 (retriggerable). The length of the impulsion can be tuned thanks to the internal and external R and C. A rule of thumb is that $\tau = 0.7RC$. Minimal values of the impulsions are in the range of 30 to 100 ns. For the exact expression consult the datasheet of each dispositive.

An important characteristic of monostables is the retriggerability. A monostable is retriggerable if it can starts a new cycle during the duration of the output pulse. In that case the output pulse is will be longer. In the case of non retriggerable monostables input transitions are ignored during the duration of the pulse.

5.3 Flip-Flops

Flip-Flops or bistables multivibrators are synchronous bistable cirucits, where by synchronous means that the output can change only in presence of a falling or raising edge of a clock signal. In figure 10 are represented the a SR and a D flip-flop. Besides these two flip-flops, based directly on SR and D bistables, there is a third type of flip-flip called JK.

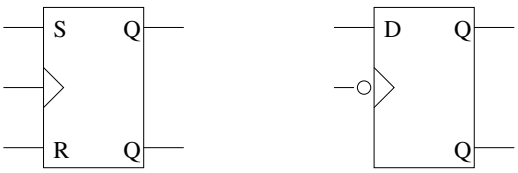


Figure 10: Logical symbols of SR and D flip-flop triggered by rising edge and falling edge respectively

Transition Detector .

Flip-flops are triggered by falling or rising edge of a clock. In figure 11 is shown an implementation of a basic transition detector.

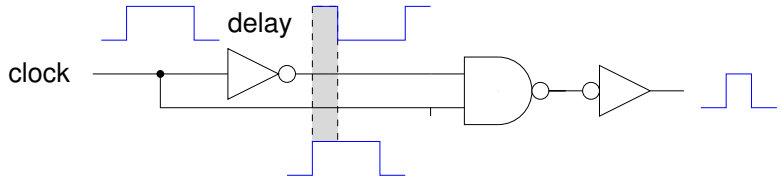
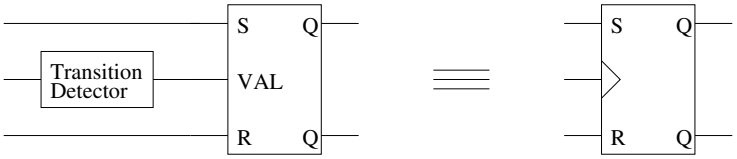


Figure 11: Rising edge transition detector



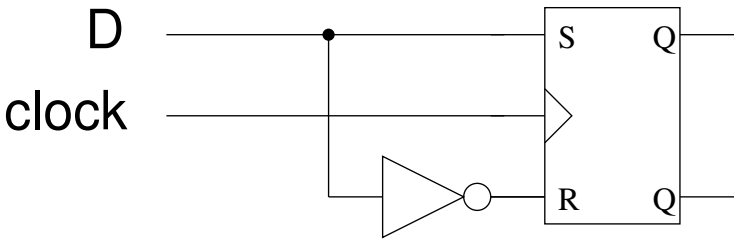
S	R	clock	Q_n
0	0	X	Q_{n-1}
1	0	↑	1
0	1	↑	0
1	1	X	not valid

Figure 12: SR Flip-Flop implementation

SR flip-flops An SR flip-flop is based on a SR bistable. Indeed the only difference between a SR flip-flop and a validated SR bistable will be the nature of the validation signal, in the case of o flip-flop the validation will came from a transition detector. In figure 12 is shown how to build one of such devices.

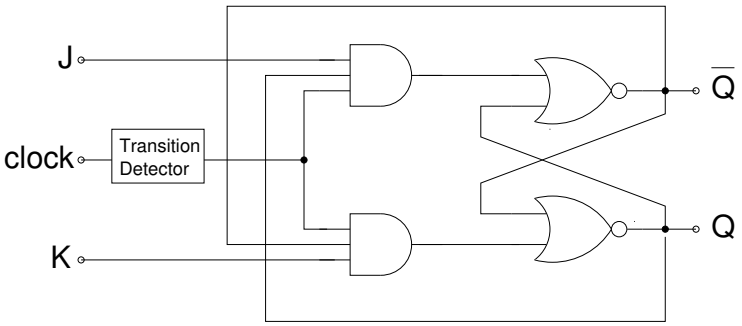
D flip-flops

JK flip-flops . Logical behavior of JK flip-flops is exactly equal to that of the SR flip-flop for the *set* and *reset* states and *no change*. The difference cames from the fact that in this flip-flop there is not a *no valid* state. In figure **??** is shown the implementation of this flip-flop and its truth table.



D	clock	Q_n
1	↑	1
0	↑	0

Figure 13: D Flip-Flop implementation



S	R	clock	Q_n
0	0	$\uparrow$	Q_{n-1}
1	0	$\uparrow$	1
0	1	$\uparrow$	0
1	1	$\uparrow$	$\overline{Q}_{n-1}$

Figure 14: JK Flip-Flop implementation

The equations the outputs is then:

$$Q = \overline{K \cdot Q + \overline{Q}}$$

$$\overline{Q} = \overline{J \cdot \overline{Q} + Q}$$

We have In this case the analysis of all cases is:

	$(Q\bar{Q})_{n-1} = 10$	$(Q\bar{Q})_{n-1} = 01$
$JK = 00$	$Q_n = \overline{01 + 0} = 1$ $\bar{Q}_n = \overline{00 + 1} = 1$	$Q_n = \overline{00 + 1} = 0$ $\bar{Q}_n = \overline{01 + 0} = 1$
$JK = 10$	$Q_n = \overline{01 + 0} = 1$ $\bar{Q}_n = \overline{10 + 1} = 0$	$Q_n = \overline{00 + 1} = 0 \rightarrow 1$ $\bar{Q}_n = \overline{11 + 0} = 0 \rightarrow 0$
$JK = 01$	$Q_n = \overline{11 + 0} = 0 \rightarrow 0$ $\bar{Q}_n = \overline{00 + 1} = 0 \rightarrow 1$	$Q_n = \overline{10 + 1} = 0$ $\bar{Q}_n = \overline{01 + 0} = 1$
$JK = 11$	$Q_n = \overline{11 + 0} = 0 \rightarrow 0$ $\bar{Q}_n = \overline{10 + 1} = 0 \rightarrow 1$	$Q_n = \overline{10 + 1} = 0 \rightarrow 1$ $\bar{Q}_n = \overline{11 + 0} = 0 \rightarrow 0$

Asynchronous inputs Besides the synchronous inputs, flip-flops have also two asynchronous inputs, in order to allow the user to set a known state. This inputs are called SET or PRESET to choose state $Q = 1$ and RESET or CLEAR to select $Q = 0$. It is important to repeat that these inputs are asynchronous so it is not needed the presence of any clock to make them work.

5.4 Counters

Synchornous counters

5.5 Shift Registers

5.6 555 Timer

6 Memories and data storage

7 ROM

8 RAM

9 Bibliography

- Millman
- Floyd
- Horowtiz
- Plonus