Digital Electronics

Diego Trapero

Table of contents

1 Digital Electronics

2 Binary

The binary numeral system, or base-2 numeral system, represents numeric values using two symbols: typically 0 and 1. More specifically, the usual base-2 system is a positional notation with a radix of 2. #

$[x]_{10}$	$[x]_2$	$[x]_{16}$
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Powers of 2

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	264
9	512
10	1024

Numbers you can represent with n bits

System	Range

Natural binary	0 to $2^n - 1$
One's complement	$-2^{n-1} - 1$ to $2^{n-1} - 1$
Two's complement	-2^{n-1} to $2^{n-1} - 1$

2.1 Binary - Decimal Conversion

2.1.1 Binary to Decimal Conversion

Digit weighting

2.1.2 Decimal to Binary Conversion

Sum of weights

Repeated division by 2

2.2 Binary Arithmetic

2.2.1 Binary Addition

Addition rules

- $0 + 0 = 0$

- $0 + 1 = 1$
- $1 + 0 = 1$
- $1 + 1 = 10$ (1 and carry 1)

```
  1 1 1 1 1    (carried digits)
    0 1 1 0 1
+   1 0 1 1 1
-------------
= 1 0 0 1 0 0 = 36
```

2.2.2 Binary Substraction

Substraction rules

- $0 - 0 = 0$
- $0 - 1 = 0$
- $1 - 0 = 1$
- $10 - 1 = 1$

```
    *   * * *   (starred columns are borrowed from)
  1 1 0 1 1 1 0
-     1 0 1 1 1
----------------
= 1 0 1 0 1 1 1
```

2.2.3 Binary Multiplication

Multiplication rules

- $0 \cdot 0 = 0$
- $0 \cdot 1 = 0$
- $1 \cdot 0 = 0$
- $1 \cdot 1 = 1$

```
        1 0 1 1   (A)
      × 1 0 1 0   (B)
      ---------
        0 0 0 0   ← Corresponds to the rightmost 'zero' in B
+     1 0 1 1     ← Corresponds to the next 'one' in B
+   0 0 0 0
+ 1 0 1 1
---------------
= 1 1 0 1 1 1 0
```

```
          1 0 1 . 1 0 1     A (5.625 in decimal)
        × 1 1 0 . 0 1       B (6.25  in decimal)
        ------------------
              1 . 0 1 1 0 1   ← Corresponds to a 'one' in B
  +           0 0 . 0 0 0 0   ← Corresponds to a 'zero' in B
  +         0 0 0 . 0 0 0
  +       1 0 1 1 . 0 1
  +     1 0 1 1 0 . 1
  --------------------------
  =   1 0 0 0 1 1 . 0 0 1 0 1  (35.15625 in decimal)
```

2.2.4 Binary Division

2.2.5 Complements

In mathematics and computing, the method of complements is a technique used to subtract one number from another using only addition of positive numbers. This method was commonly used in mechanical calculators and is still used in modern computers. To subtract a number y (the subtrahend) from another number x (the minuend), the radix complement of y is added to x and the initial '1' of the result is discarded. Discarding the initial '1' is especially convenient on calculators or computers that use a fixed number of digits: there is nowhere for it to go so it is simply lost during the calculation. #

1's Complement

The One's complement of a binary number is defined as the value obtained by inverting all the bits in the binary representation of the number (swapping 0s for 1s and vice-versa). #

The ones' complement of the number then behaves like the negative of the original number in some arithmetic operations, although it presents some problems that are solved by the two's complement, like the offset by -1.

Examples

$$1C[11010] = 00101$$

2's Complement

Obtain the 2's complement

- **From the 1's complement**

$$2C[x] = 1c[x] + 1$$

- **Working from LSB to MSB**

2.3 Signed numbers

Sign-magnitude form

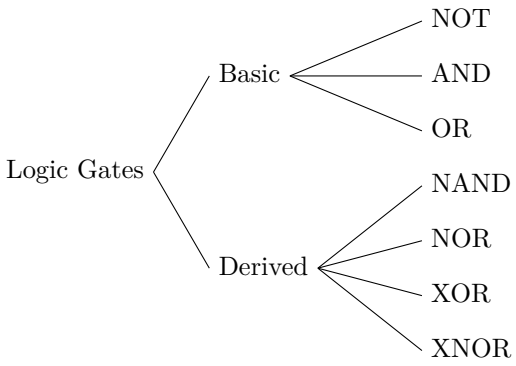
1's complement form

2's complement form

2.4 Boolean Algebra

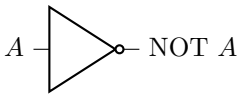
3 Digital components

3.1 Logic Gates



- **Basic gates** perform the three basic boolean operations: conjunction (AND), disjunction (OR) and negation (NOT). Only one the two other gates is neccesary in conjuntion with the NOT gate to implement everty boolean function. The other operation can be expressed in therms of the other two.
- **Derived gates** perform derived boolean operations, operations that can be composed from basic operations. The most common are NAND, NOR, XOR, and XNOR gates.

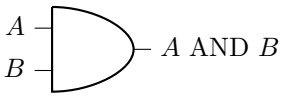
3.1.1 NOT Gate



Rule: The NOT gate output is the complementary of the input

A	NOT A
0	1
1	0

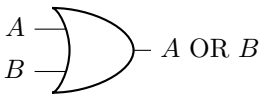
3.1.2 AND Gate



Rule: The AND gate output is 1 only when all the inputs are 1.

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

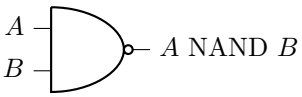
3.1.3 OR Gate



Rule: The OR gate output is 1 when any of the inputs is 1.

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

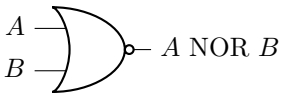
3.1.4 NAND Gate



Rule: The NAND gate output is 0 when all the inputs are 1

- The NAND gate is a NOT-AND gate
- The NAND gate can be viewed as a negative-OR whose output is 1 when any inputs is 0

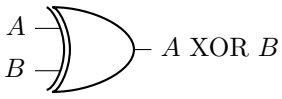
3.1.5 NOR Gate



Rule: The NOR gate output is 0 when any of the inputs is 1.

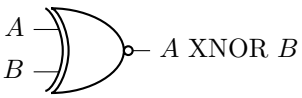
- The NOR gate is a NOT-OR gate
- The NOR gate can be viewd as a negative-AND whose output is 1 only when all the inputs are 0.

3.1.6 XOR Gate



Rule: The XOR gate output is 1 when all the inputs are not the same

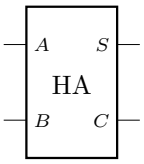
3.1.7 XNOR Gate



Rule: The XNOR gate output when the inputs are not the same

3.2 Adders

3.2.1 Half adder



A half adder is a two input, two output circuit, that takes two input bits, A and B, and ouputs the two-digit sum, $A + B$. The MSB of the sum is referred as C (carry) and the LSB digit is S (sum). If performing addition digit by digit, the S ouput it's the result digit and the C output is the carry resulting of the operation. This adder does't take into account the carry of the previous digit addition, this is why it is called half adder.

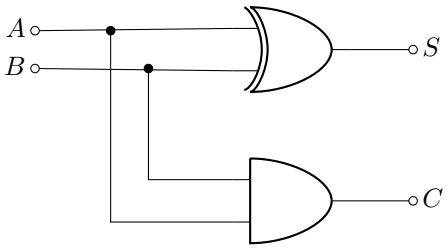
Truth table

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

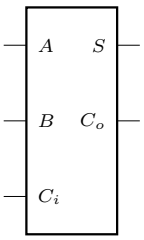
Logic operations

- $S = A \oplus B$
- $C = AB$

Gate implementation



3.2.2 Full adder



A full adder is a three input, two output circuit, that takes three input bits, A, B and C_i, and ouputs the two-digit sum, $A + B + C_i$. If performing addition digit by digit, A and B are the operands, C_i the previous carry, the S ouput it's the result digit and the C output is the carry resulting of the operation.

Truth table

A	B	C_i	C_o	S
0	0	0	0	0
1	0	0	0	1
0	1	0	0	1
1	1	0	1	0
0	0	1	0	1
1	0	1	1	0
0	1	1	1	0
1	1	1	1	1

Logic operations

- $S = A \oplus B \oplus C_i$
- $C_o = (A \cdot B) + (C_i \cdot (A \oplus B))$

Gate implementation

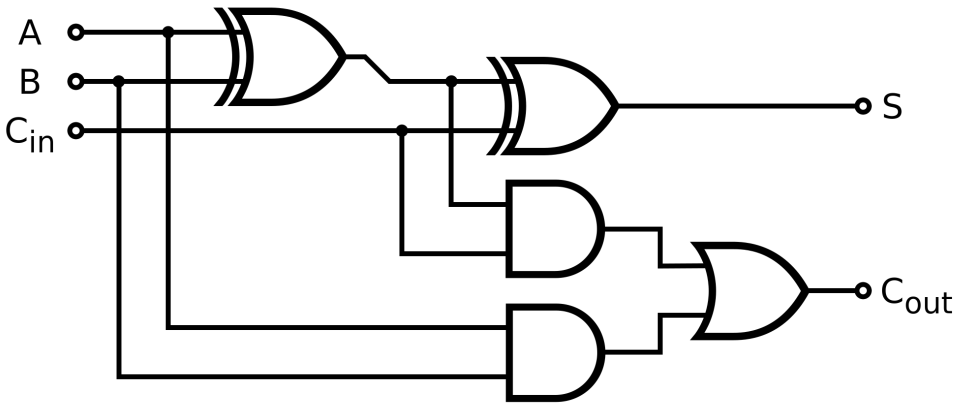
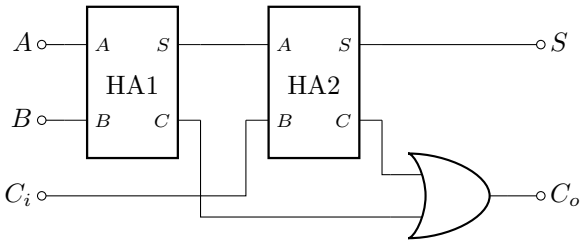


Figure 1: Source: http://commons.wikimedia.org/wiki/File:Full_Adder.svg

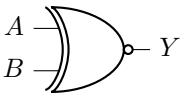
Full adder from two half adders A full adder can be implemented from two half adders and an OR gate:



3.3 Comparators

3.3.1 Equality Comparators

1-bit equality comparator A 1 bit equality comparator can be implemented with a XNOR gate:



A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

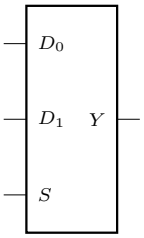
n-bit equality comparator A n-bit comparator can be implemented by comparing two numbers digit by digit with n XNOR gates and then processing the outputs through an AND gate to check if every bit is the same in the two numbers

3.4 Multiplexers, Demultiplexers, Encoders, Decoders

3.4.1 Multiplexer

A Multiplexer is...

2 to 1 Multiplexer



$$Y = (D_1 \cdot \overline{S}) + (D_0 \cdot S)$$

S	D_0	D_1	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

S	Y
0	D_0
1	D_1

A straightforward realization of this 2-to-1 multiplexer would need 2 AND gates, an OR gate, and a NOT gate.

MUX implementation of truth table A mux based circuit can be used to imple-

ment a boolean function.

A logic function with n input bits and 1 ouput bit can be implemented in a MUX with n selection bits and 2^n data inputs, i.e. a 2^n-to-1 MUX.

[***** poner ejemplo]

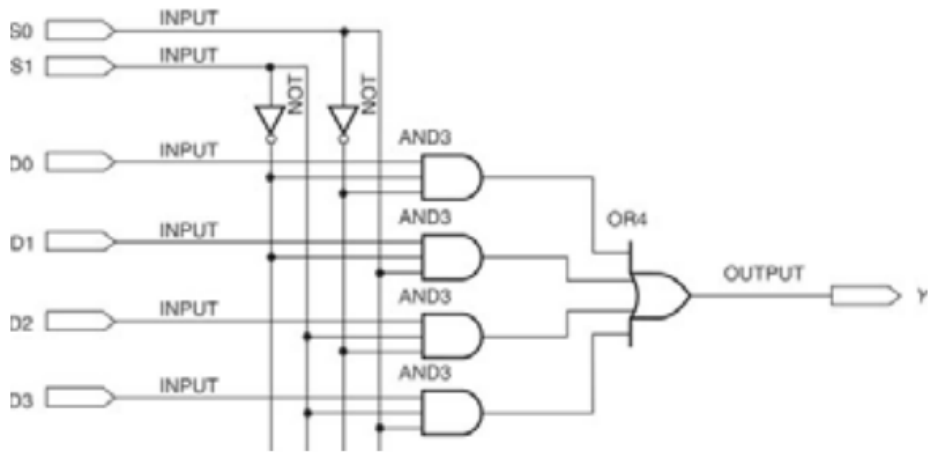


Figure 2: 4 to 1 multiplexer. Source: http://www.ecgf.uakron.edu/grover/web/ee263/slides/Chapter%2006B.pdf

3.4.2 Demultiplexer

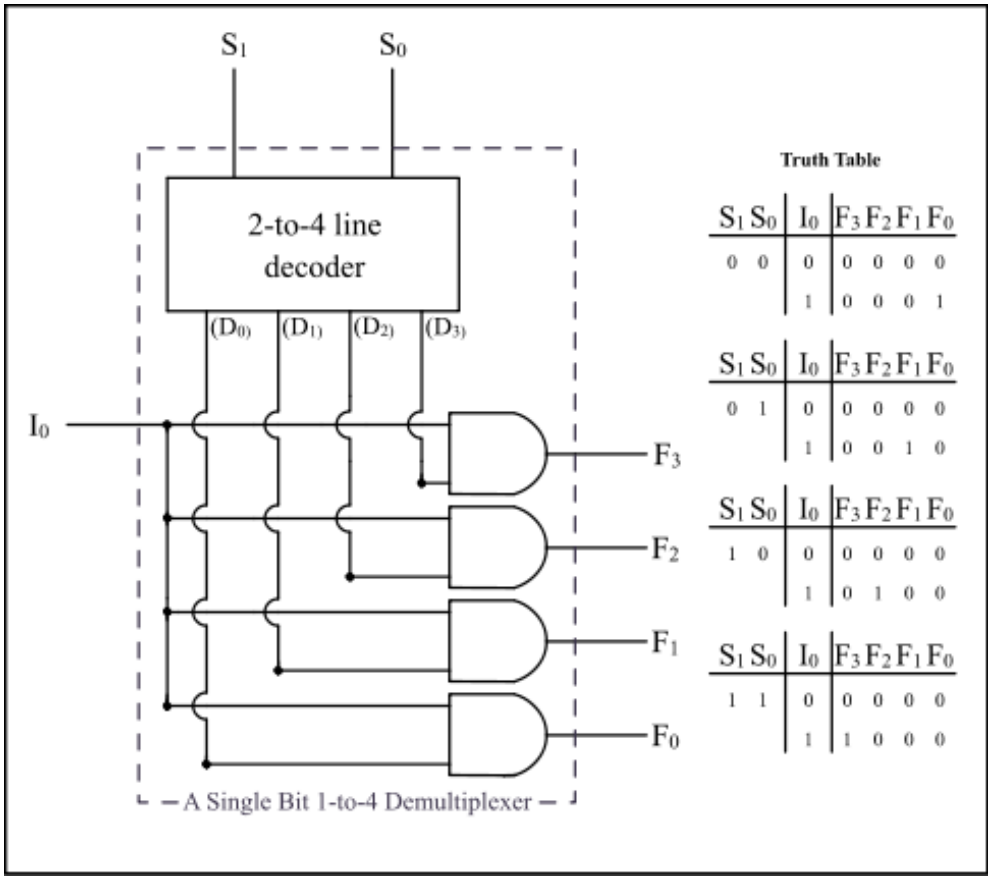


Figure 3: 4 to 1 demultiplexer. Source: http://en.wikipedia.org/wiki/File:Demultiplexer_Example01.svg

3.4.3 Encoder

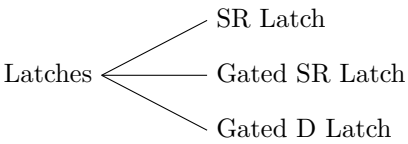
3.4.4 Decoder

3.5 Biestables

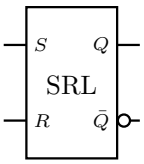
Biestables are devices that have two stable states (SET and RESET); they can retain either of these states indefininitly, making them useful as storage devices.

3.5.1 Latches

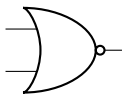
Latches are...

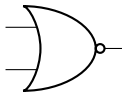


SR (SET-RESET) Latch

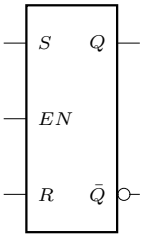


Gate implementations

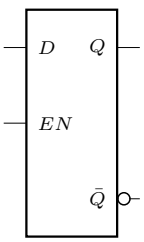




Gated SR Latch

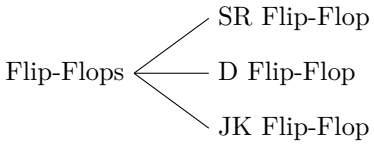


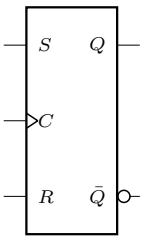
Gated D Latch



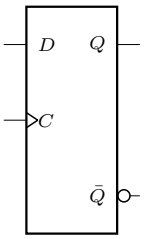
3.5.2 Flip-Flops

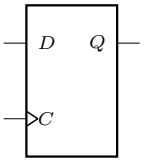
Flip-flops are...





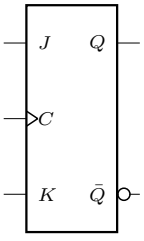
D FlipFlop





CLK	D	Q_{next}
^	0	0
^	1	1
0	X	Q
1	X	Q

JK FlipFlop



3.6 Finite State Machines

Finite State Machines (FSM) are...

- Moore machine: The FSM uses only entry actions, i.e., output depends only on the state.
- Mealy machine: The FSM uses only input actions, i.e., output depends on input and state.

3.7 Lookup tables

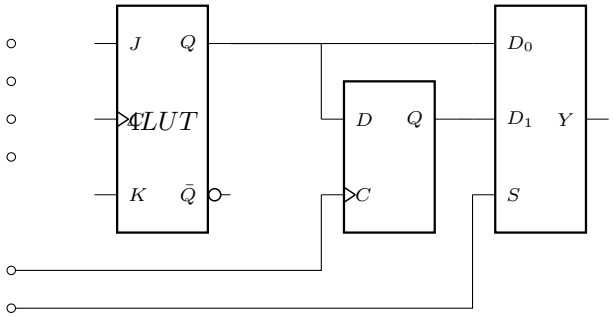
A Lookup Table (LUT) can enconde a n-bit boolean function, with 1 bit output.

A n-bit LUT can be implemented with a mux whose select lines are the n inputs of the LUT and whose 2^n inputs are constants.

2-bit LUT

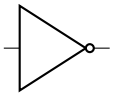
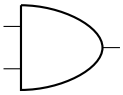
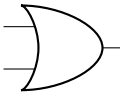
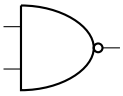
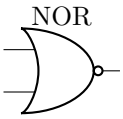
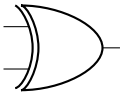
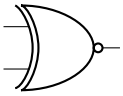
X_0	X_1	Y
0	0	y_0
0	1	y_1
1	0	y_2
1	1	y_3

FPGA logic cell FPGAs are built of many interconnected logic cells. A basic logic cell could be composed of a 4 LUT, a flipflop and a 2to1 mux to bypass the register.

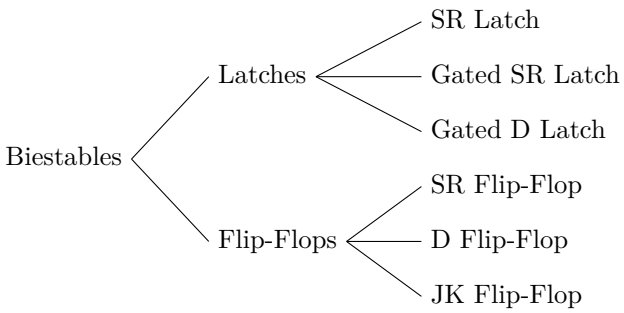


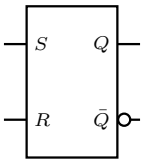
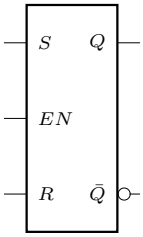
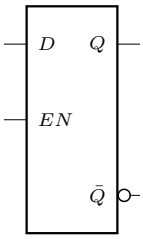
4 Reference tables

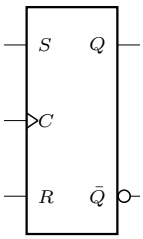
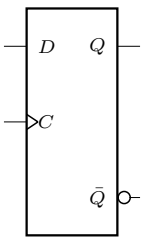
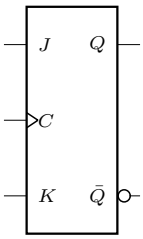
4.1 Logic Gates

Gate	Symbol	Rule	Truth table
NOT		The NOT gate output is the complementary of the input	not
AND		The AND gate output is 1 only when all the inputs are 1	and
OR		The OR gate output is 1 when any of the inputs is 1	or
NAND		The NAND gate output is 0 when all the inputs are 1	or
NOR		The NOR gate output is 0 when any of the inputs is 1	or
XOR		The XOR gate output is 1 when all the inputs are not the same	or
XNOR		The XNOR gate output when the inputs are not the same	or

4.2 Biestables



Latch	asd
	q
	q
	q

FlipFlop	asd
	q
	q
	q

5 Bibliography

- Digital Fundamentals, Thomas L. Floyd, 10th ed. Pearson
- https://en.wikipedia.org/wiki/Binary_number
- https://en.wikipedia.org/wiki/Ones'_complement

6 More