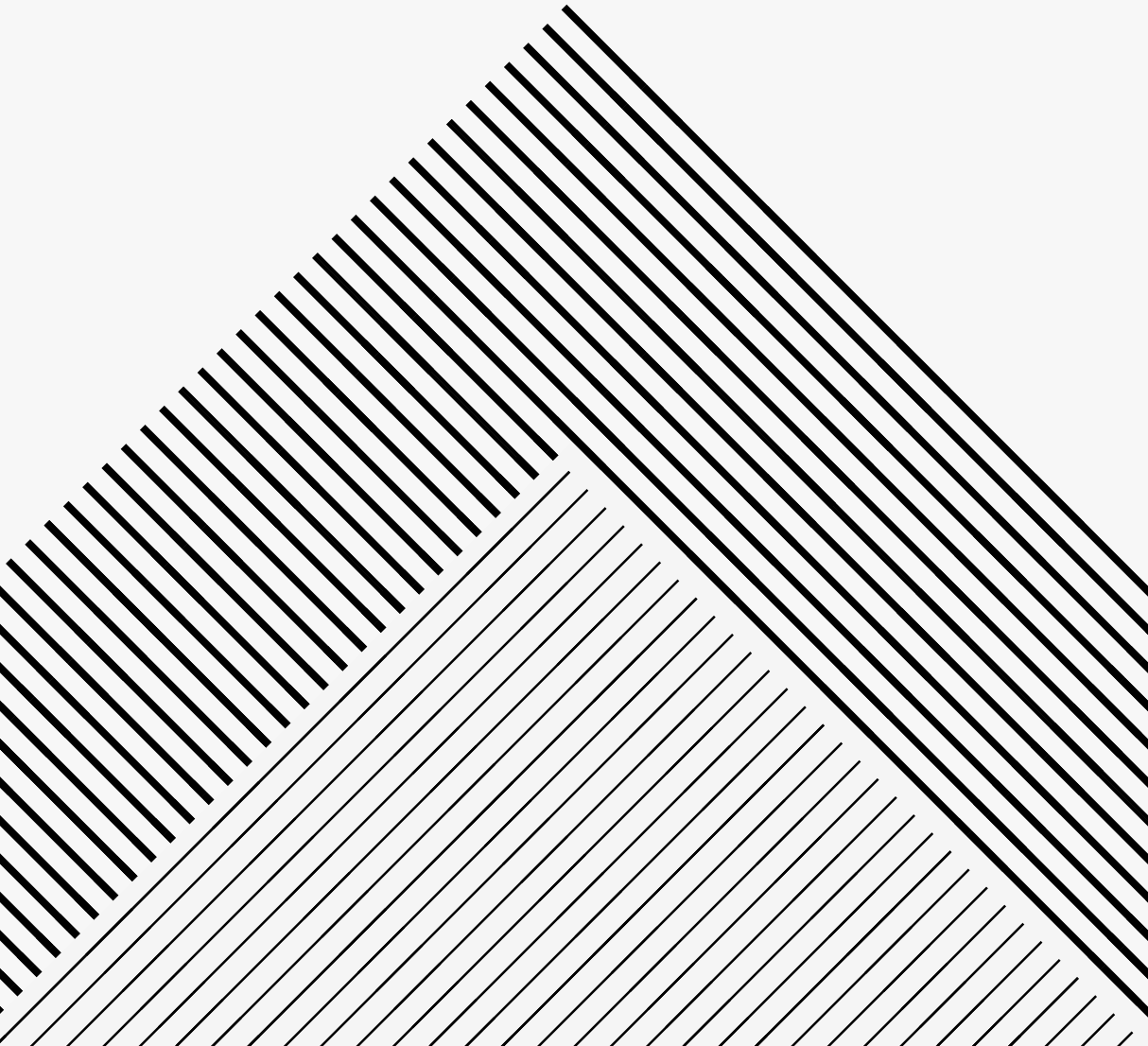
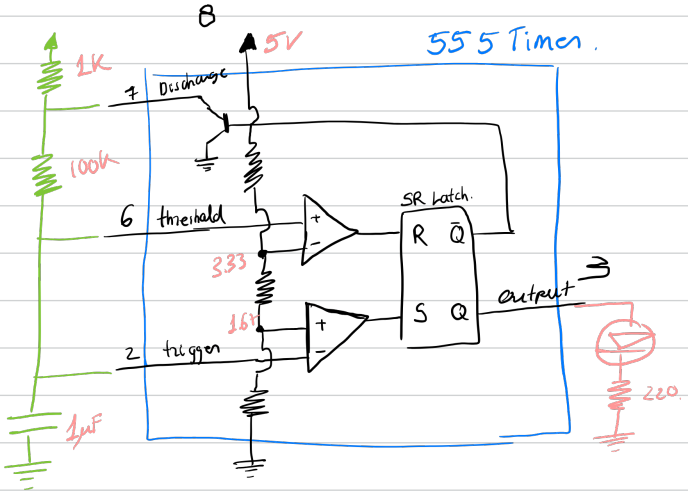


Astable 555 timer.



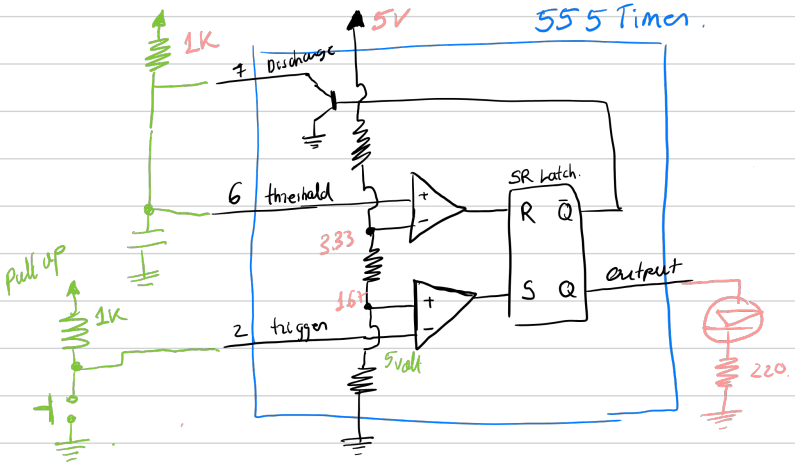
$$T = t_1 + t_2 = t_{1\,\text{charge time}} + t_{2\,\text{Discharge}} = 0.693 \cdot (R_A + R_B) \cdot C + 0.693 \cdot R_B \cdot C = 0.693 (R_A + 2R_B) C$$

$C = 1\mu C$
$R_A = 1K$
$R_B = 10K$

$t = 1.386\,s$ $\quad f = \frac{1}{T} = 0.7215$

Monostable 555 timer.

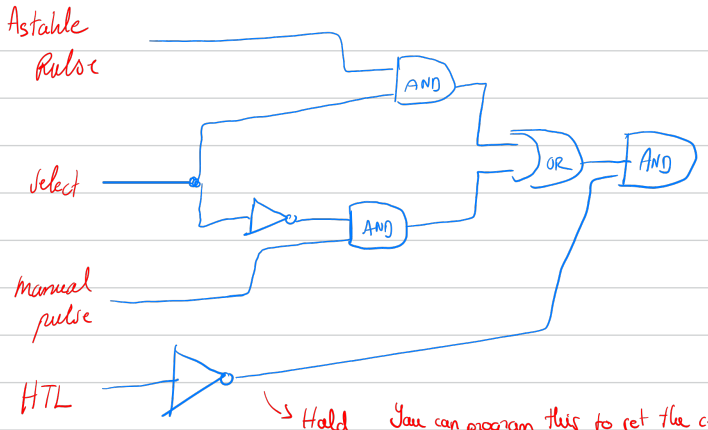
Problem Bouncing pulses.



So when I press the button down the comparator set the SR latch to L so the output goes high, If don't press the latch inverse the output ant let the current flows out through the transistor and discharge

Monostable only when the button is press.

Pin 4 goes to 5V just to prevent that I from accidentally tripping

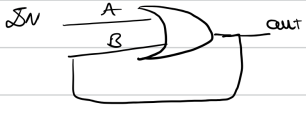


↳ Hold. You can program this to set the clock to 0

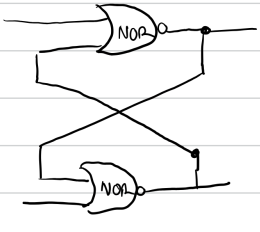
AND	
0 0	0
0 1	0
1 0	0
1 1	1

OR	
0 0	0
0 1	1
1 0	1
1 1	1

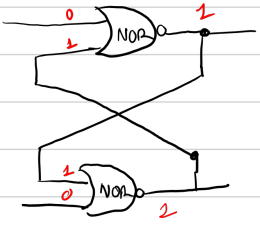
SR LATCH



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

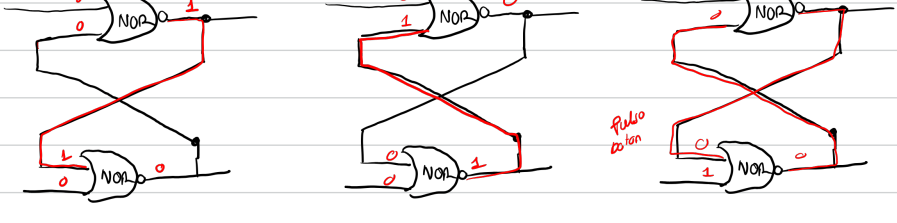


Confundí el chip SN74LS32N ⇒ OR
×
SN74LS02N ⇒ NOR

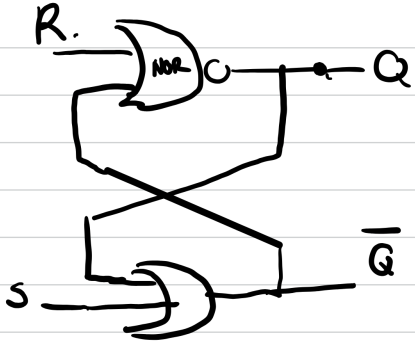


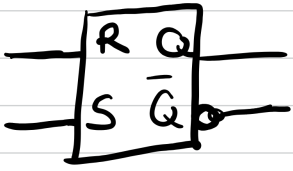
NOR	
0 0	1
0 1	0
1 0	0
1 1	0

T=0s

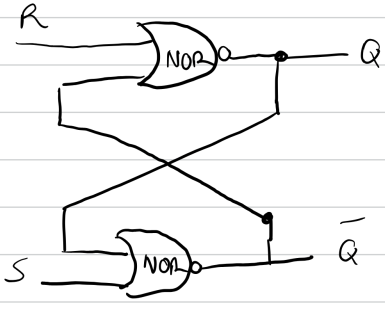


SR Latch

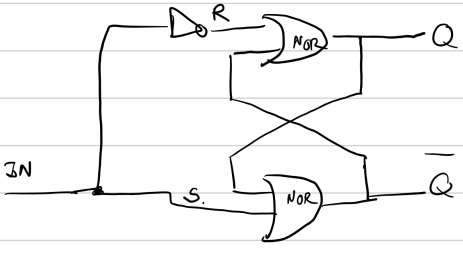




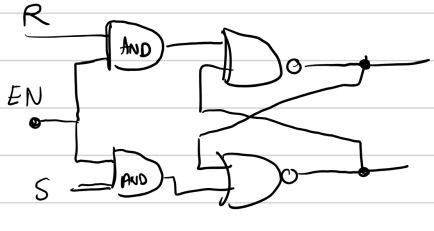
D LATCH



Queremos con un solo botón poder cambiar de estado. Si colocamos una inversión en una de las entradas y la conectamos a el botón tenemos

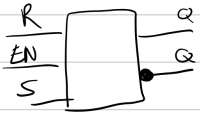


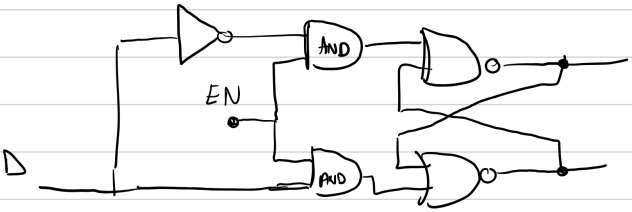
Solución: Meter un enable



¿Qué pasa? Que no podemos memorizar ni 0 ni 1 ni otra

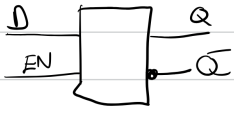
Si $Q=1$ $\bar{Q}=0$





Si ya de ultimas le metemos un inversor al reset nos ahorramos el tener que setear y resetear de manera dependiente

Solo cambia siempre y cuando el enable esté activo



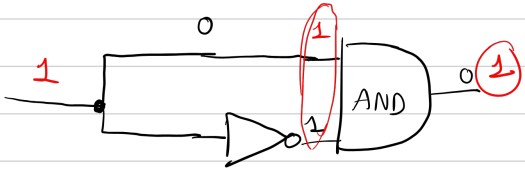
D stands for data

Lo que pasa es que necesitamos que todo esté sincronizado. para eso usamos el clk necesitas un clk en todos los circuitos

La cosa es que ahora el reloj tendrá prioridad y da igual si el D está cambiando que solo nos preocupamos por cuando empieza el reloj en donde estabamos.

Ejemplo

Ahora queremos conseguir un detector de pulsos para el clk ya que ralo nos interesa ese pulso que determina cuando cambia todo



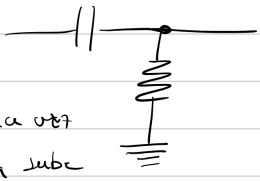
Solo cuando las dos inputs sean 1 saldrá un 1

La cosa es que el inversor toma tiempo para invertir lo que hace que dispare la salida a 1 durante ese tiempo perfecto para lo que queremos

$0,1\mu F = 0.1 \times 10^{-6}$

$1k = 10^3$

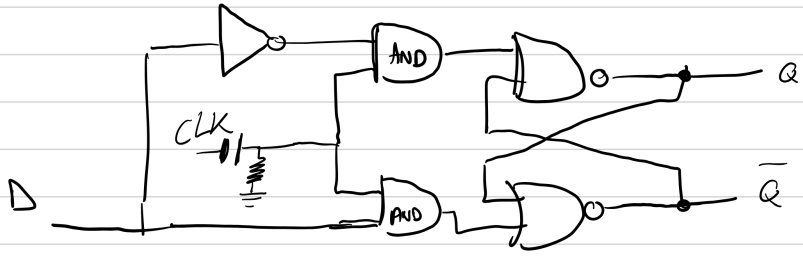
$0.1 \cdot 10^{-6} \cdot 10^3 = 0'1\ ms$

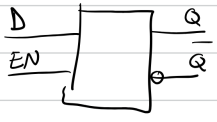


La corriente fluye, una vez el condensador se carga sube el flanco y se descarga

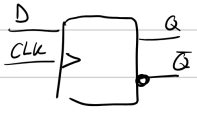
Finally we got the D flip flop.

D flip flop





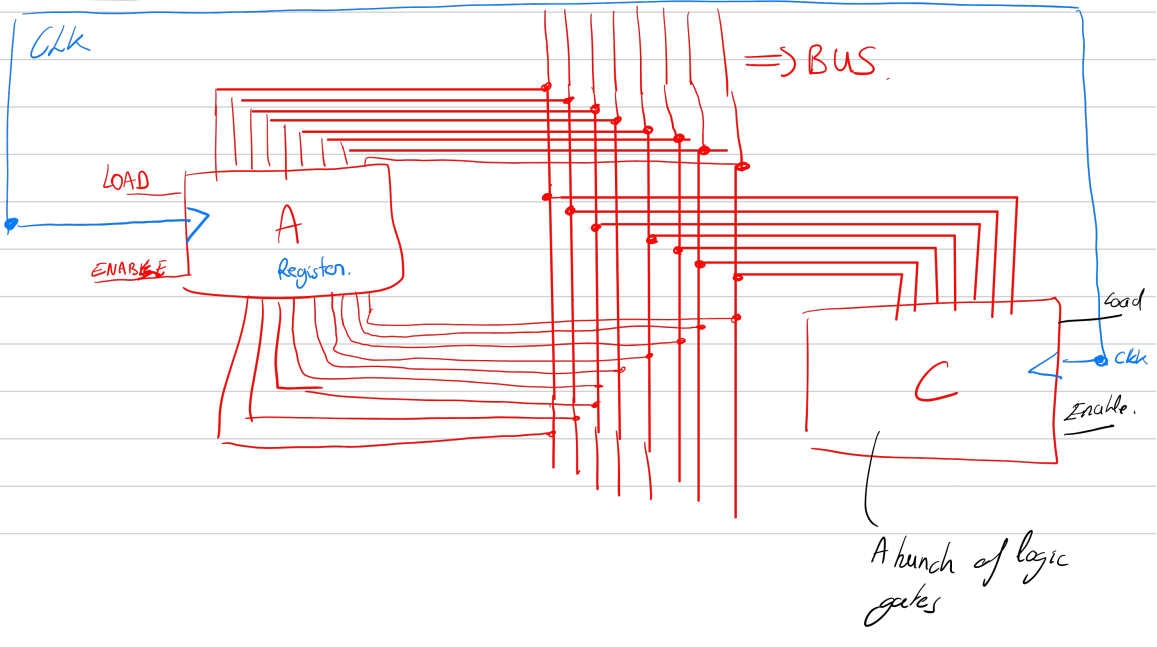
D Latch.



D flip flop

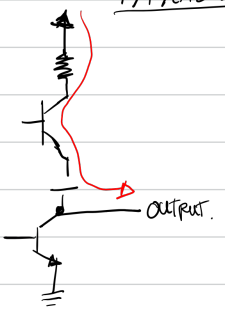
BUS ARCHITECTURE and how register transfers work.

8bit Register.

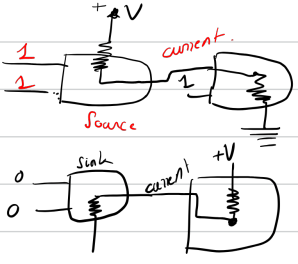


Podemos llegar a tener un problema si las outputs de diferentes dispositivos como output A y output B salen al mismo cable al mismo tiempo

TYPYCAL LOGIC GATE



output is able to sync signal
Also you can take that to the ground

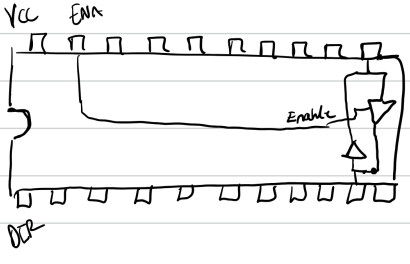


Sinking
Sending 0 ⇒ Pulling current
Sending 1 ⇒ Pulling current also

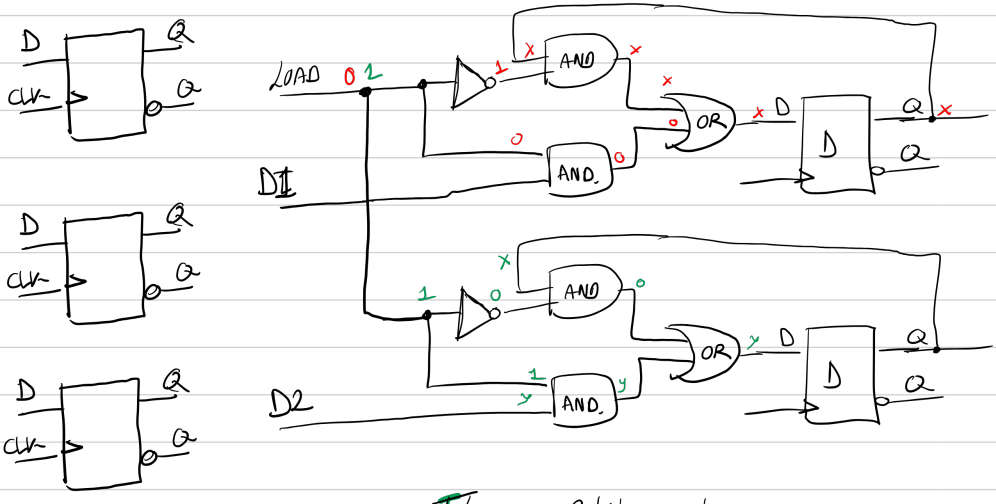
↳ We need other state
Z do nothing leave the wire alone

For this we got

SN54/74LS245

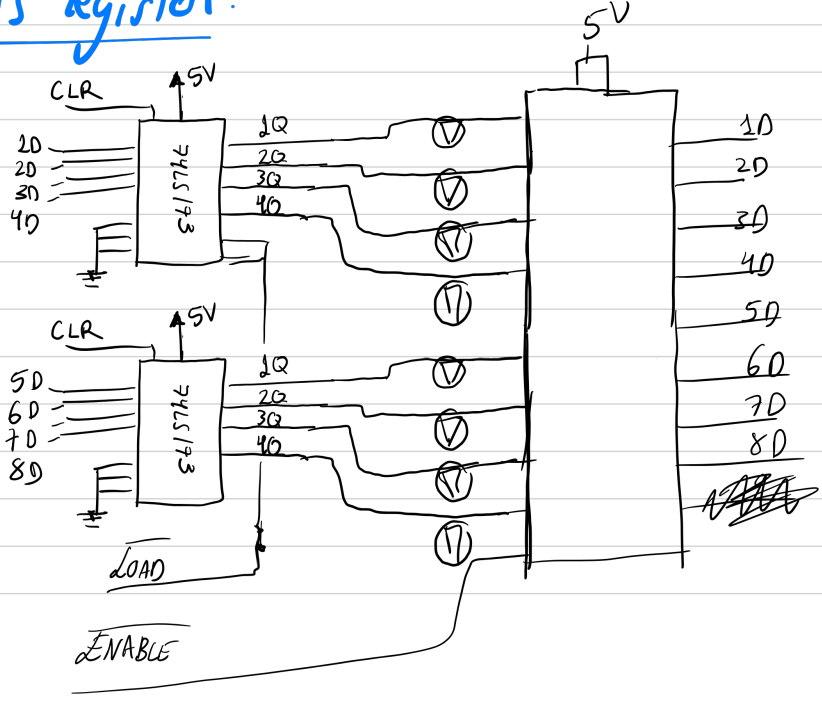


Designing and building 1bit - 8bits registers.



This is a 2 bits registers.

8bits registers.



Negative numbers in binary.

2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

128 64 32 16 8 4 2 1

0 0 0 0 0 1 0 1 = 5

⊖ 1 0 0 0 0 1 0 1 = -5

1 1 1 1	-7
1 1 1 0	-6
1 1 0 1	-5
1 1 0 0	-4
1 0 1 1	-3
1 0 1 0	-2
1 0 0 1	-1
1 0 0 0	-0
0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7

+5 + (-5)

1 0 ②
0 1 0 1
1 1 0 1
(1) 0 0 1 0

Con el complemento a 1 se obtiene invirtiendo todos los bits del numero positivo

5 = 0 1 0 1

1 0 1 0

↘ sigue el problema del -0.

Solución ⇒ Complemento a 2.

Invertes y sumas para sacar la equivalencia y poder operar todos numeros positivos y negativos.

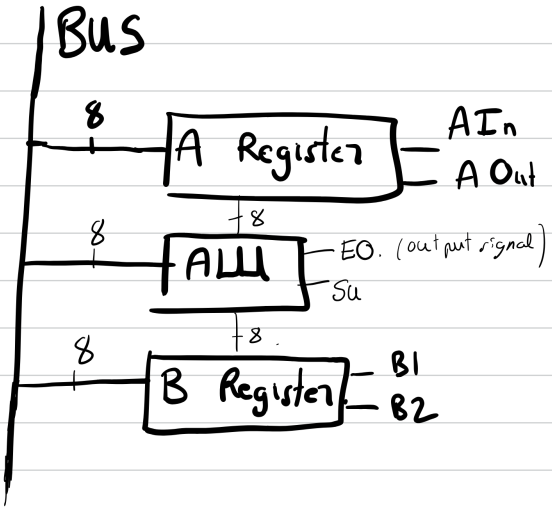
En complemento a 2: Todo hay un 0

Invertimos

Sumamos.

ALU DESIGN

Arimethic logic Unit



Dentro de la ALU vamos a necesitar 1 sumador. Lógica de los sumadores:

Tabla verdad

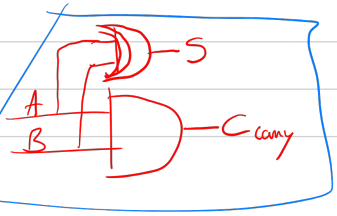
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

haciendo las combinaciones

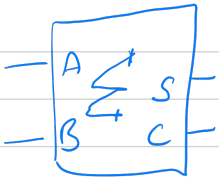
+	0	1	1
0	1	0	1
0	1	1	10
			C S

Esto es $A\bar{B} + \bar{A}B = A \oplus B$ XOR

El carry es una AND $C = A \cdot B$

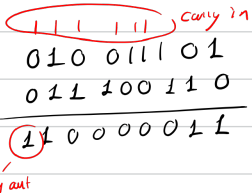


Sumador de 1 bit



Este sumador no tiene en cuenta el carry de entrada solo el de salida.

En operaciones como estas:



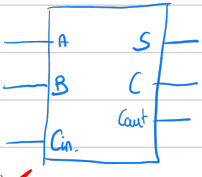


Tabla de verdad

A	B	Cin	Cout	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

Mapa de carnot

Cout

A \ BCin	00	01	11	10
0	0	0	1	0
1	0	1	1	1

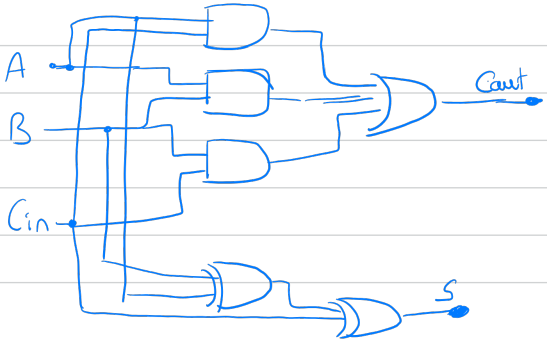
BCin

ACin AB

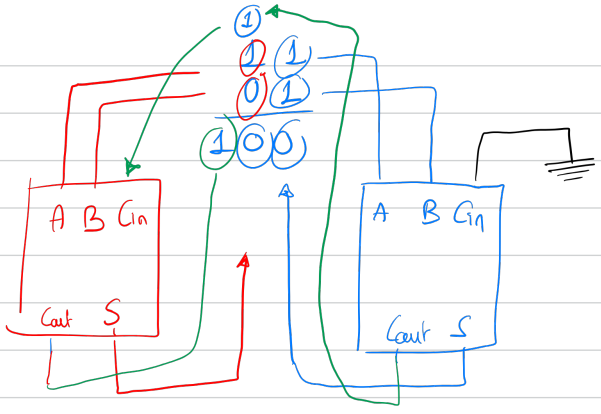
$Cout = A\bar{C}in + BCin + AB$

$S = A\bar{B}\bar{C} + A\bar{B}C + ABC + AB\bar{C} + \bar{A}B\bar{C}$

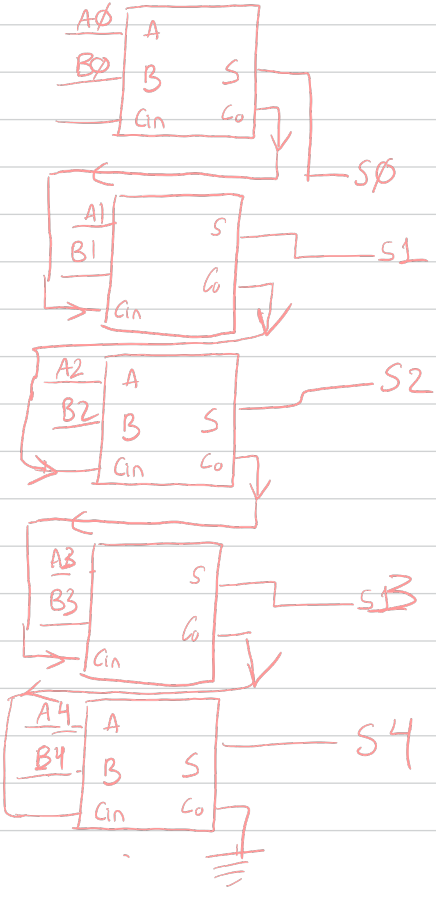
$S = (A \oplus B) \oplus C$

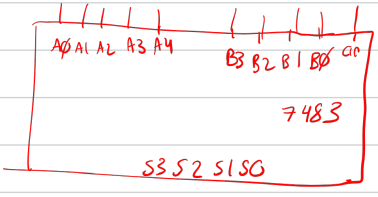


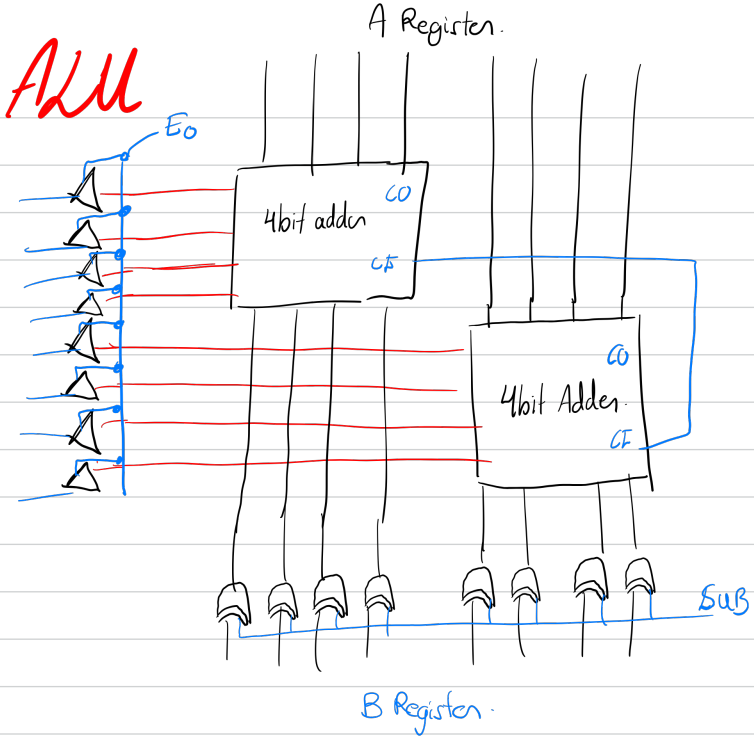
Sumadan 2 bits



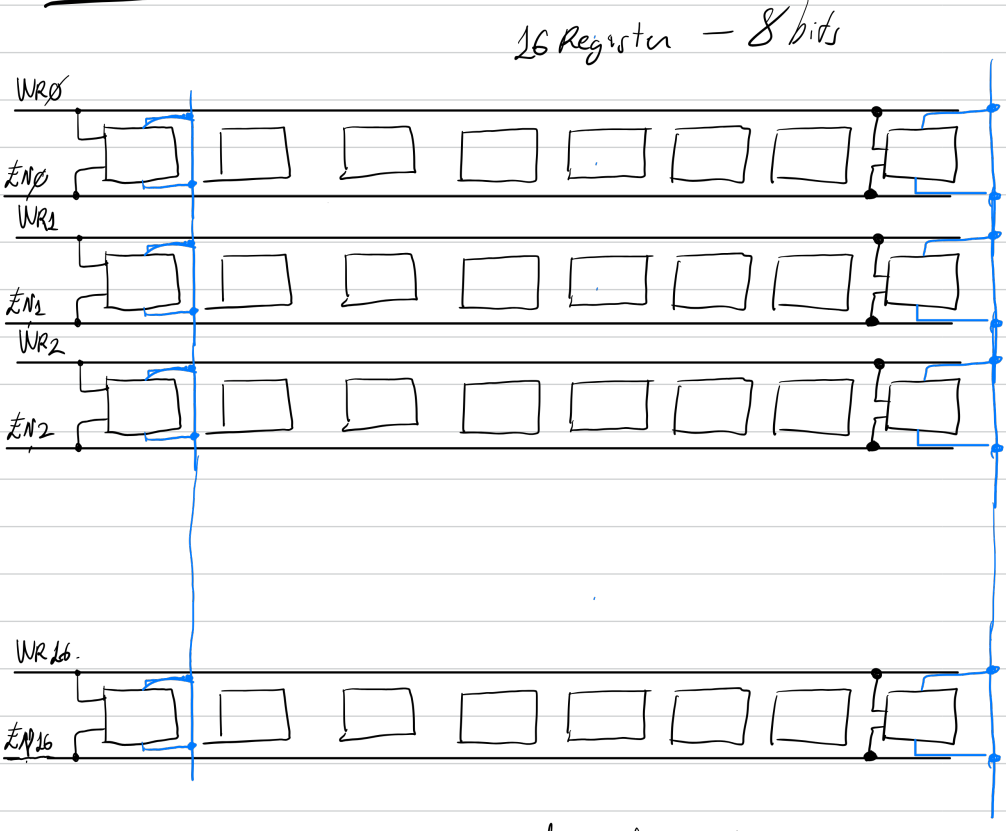
CHIP 74LS283. 4Bit adder.





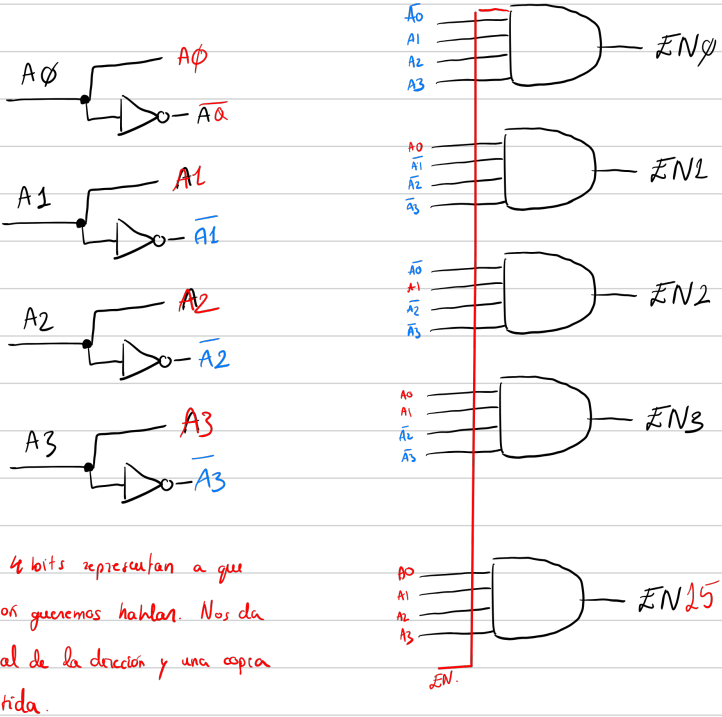

ALU
A Register
E0
4bit adder
CO
CI
4bit Adder
CO
CI
Sub
B Register

RAM MEMORY



Static Ram. more expensive and complex, advantages you don't have to the refresh, and it's faster.

ADDRESS DECODE LOGIC.



Estos 4 bits representan a que dirección queremos hablar. Nos da la señal de la dirección y una copia invertida.

Si todo son 0 en los 4 bits de dirección habilitaríamos la primera dirección. Entonces tendríamos las combinaciones 2^{16}

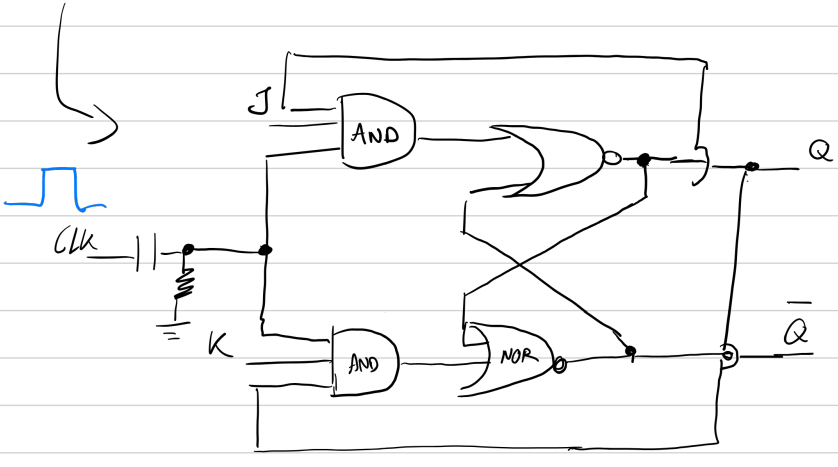
0000
0001
0010
...
1111

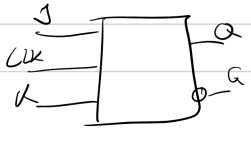
This is the logic for 74LS189 although this chip is a 64 bit RAM instead of 128 bits so we have to add a second chip.

EVOLUCION FLIP FLOPS

SR latch => Enable + Block => SR Flip flop → + Retroalimentación = JK Flip Flop.

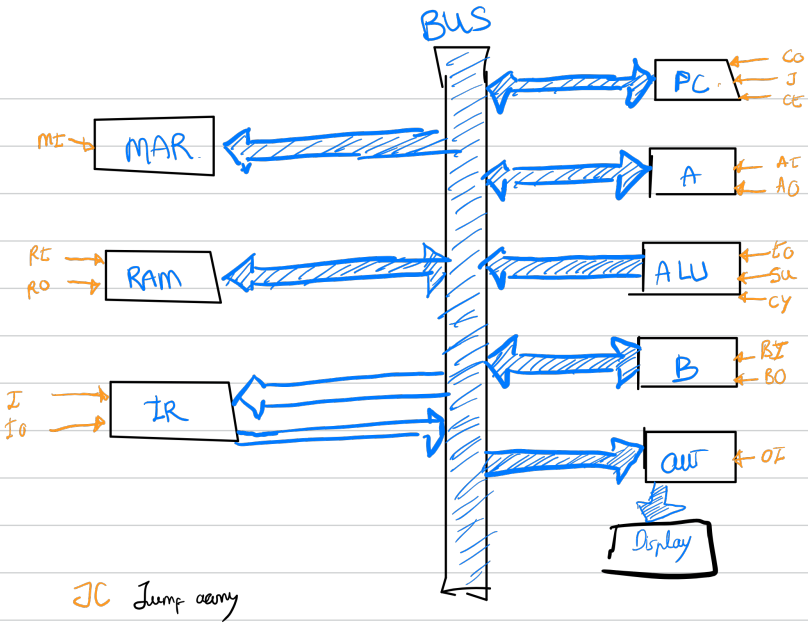
Tipo	Ventaja	Problema
SR latch	Almacena 1 bit con lógica simple.	Estado invalido si S = R = 1
SR + Enable	se controla cuando cambia con el enable	sigue teniendo estado invalido
SR flip flop	Solo cambia en flanco del reloj	Aun tiene
JK flip flop	Elimina invalido	Más complejo pero robusto.





Inversión

J	K	CLK	Q	$\bar{Q}$
0	0	↓	Last state	
0	1	↓	0	1
1	1	↓	1	0
X	X	0	Toggle	
			Last state	



JC Jump carry

HLT Halt system.

DISPLAY 7 SEGMENTI

Vamos a resolver el problema de como mostramos el número que queremos, para ello solo cogemos 4 bits para representar los numeros.

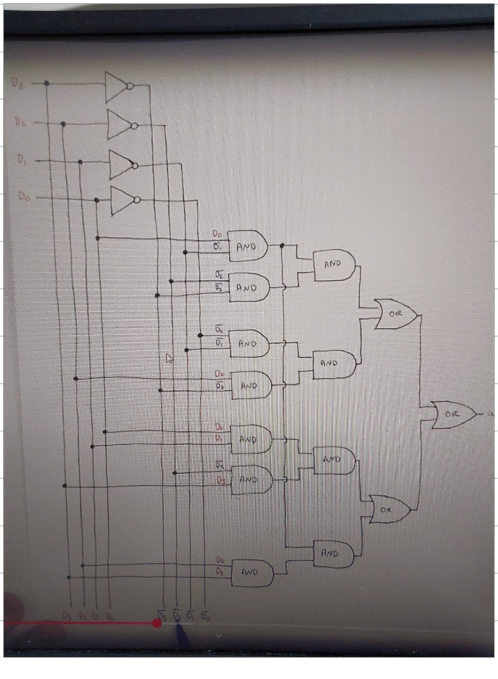
CATODO comun

0 1 2 3 4 5 6 7 8 9 A B C D E F

0000 0001 - - - - - - - 1111

7 Segmentos

	d3	d2	d1	d0	A	B
0	0	0	0	0	1	1
1	0	0	0	1	0	1
2	0	0	1	0	1	1
3	0	0	1	1	1	1
4	0	1	0	0	0	1
5	0	1	0	1	1	0
6	0	1	1	0	1	0
7	0	1	1	1	1	1
8	1	0	0	0	1	1
9	1	0	0	1	1	1
A	1	0	1	0	1	1
b	1	0	1	1	0	0
C	1	1	0	0	1	0
d	1	1	0	1	0	1
E	1	1	1	0	1	0
F	1	1	1	1	1	0



Lógica

En verdad para nuestro caso es diferente está invertido al ser cátodo común.

EPROMs.

Erasable programmable read only memory: It means that retains data even when power its turn off. If you expose the die to ultraviolet light you will erase that memory

We got another option EEPROMs you can erase without ultraviolet

En el chip AT26C16 se puede programar la lógica del 7 segmentos asignando las direcciones de memoria en la tabla de verdad del 7 segmentos.

CPU Control Logic.

La RAM en la unidad 0 tiene eso

	ADDRESS	WHAT DOES IT MEAN	EQUIVALENCE
LDA 14.	0000	0001	1110 [14]
ADD 15	0001	0010	1111 [15]
OUT	0010	1110	0000

En el registro de instrucción (IR) hay 8bits binarios

Bits MSB [7:4] Representan el OPCODE osea que hacer. LDA, ADD, OUT

LSB [3:0] La dirección de memoria.

¿Porqué esta dividido así? Porque en una sola instrucción puedo codificar una instrucción entera en un solo byte

Que operación ejecutar MSB

Sobre que dirección actuar. LSB

MODELO MENTAL

1. FETCH Buscar la instrucción en memoria
2. EXECUTE Ejecutar lo que dice

1. FETCH Cargan la instrucción desde la RAM al registro de instrucciones.

From RAM ⟹ Instruction register.

1 Contador como dirección
2 Lees la dirección con el registro de instrucciones
3 Aumento del contador para que apunte a la siguiente

2. EXECUTE

4 Paso el registro de instrucciones al registro de direcciones (RAM)

5 Saco la palabra de la RAM y lo meto al registro A.

OPP CODE ADD
LDA
SUB
MOV.

CPU PART 3

	INSTRUCCION	STEP	HLT	MI	RI	RO	IO	II	AI	AO	EO	SU	BI	OI	CE	CO	J
Fetch.	X X X X	0 0 0		1												1	
	X X X X	0 0 1				1		1							1		
LDA	0 0 0 1	0 1 0		1			1										
	0 0 0 1	0 1 1				1			1								
	0 0 0 1	1 0 0															
ADD	0 0 1 0	0 1 0		1			1										
	0 0 1 0	0 1 1				1							1				
	0 0 1 0	1 0 0							1		1						
OUT	1 1 1 0	0 1 0								1				1			
	1 1 1 0	0 1 1															
	1 1 1 0	1 0 0															

EPPROM

- Instrucciones ⇒ van los pines A3 A4 A5 A6
- Steps ⇒ van los pines A2 A1 A0.

A7 A8 A9 A10
como las tortugas en
Verano

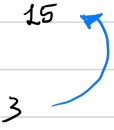
Señales primera EEPROM

HLT MI RI RO IO II AI AO

S0 S1 I2 I3 I4 I5 I6 I7

PROGRAM

0:	LDI 3	0000	0101	0011
1:	STA 15	0001	0100	1111
2:	LDI 0	0010	0101	0000
3:	ADD 15	0011	0011	1111
4:	OUT	0100	1110	0000
5:	JMP 3	0101	0110	0011



CONDITIONAL JUMP TURING COMPLETE

0 OUT

1 ADD 15

2 JUMP CARRY 4

3 Jump 0

4

5

6

CONDITIONAL JUMP TURING COMPLETE

		OPPCODE	OPERANDO
0	OUT	1110	0000
1	ADD 15	0010	1111
2	JUMP CARRY 4	0111	0100
3	JUMP 0	0110	0000
4	SUB 15	0011	1111
5	OUT	1110	0000
6	JUMP ZERO	1000	0000
7	JUMP 4	0110	0100
15	15	0000	1111