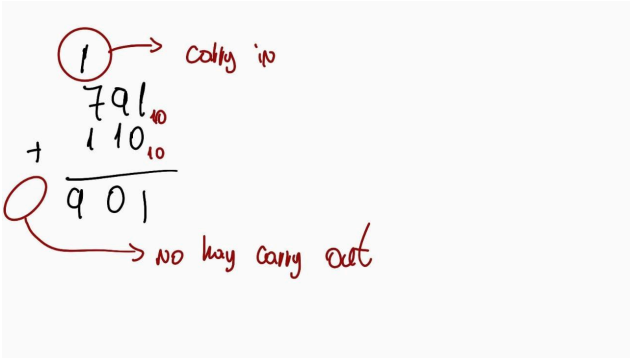
Informe de laboratorio
Guía 0
ELO-212

Grupo 9 (1001)
Integrantes:
 * Joaquin Aguilera
 * Marcelo Fernandez
 * Felipe Vega
Paralelo: 2

1.1 Diseño de un circuito digital para un full-adder

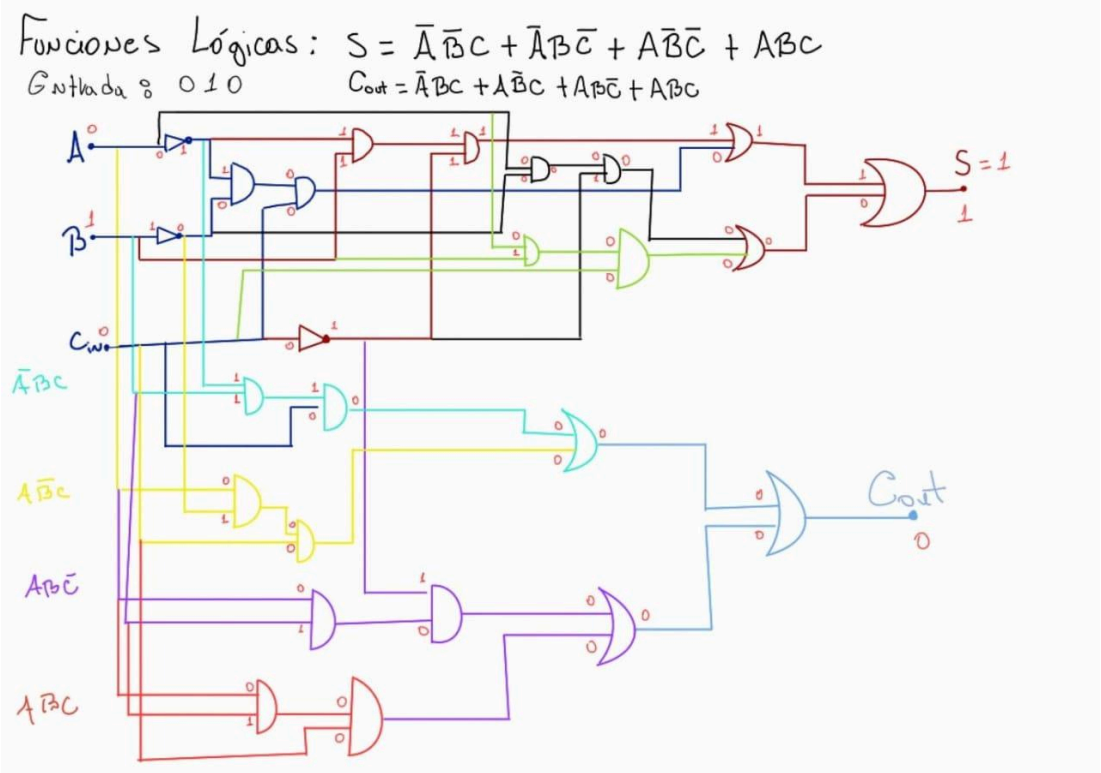
En el caso de la suma de 791+110, podemos observar como el carry out sería 0, debido a que la suma no sobrepasa la cantidad máxima representable con la respectiva base numérica.



La tabla muestra muestra las entradas y salidas lógicas de un circuito full-Adder, Para diseñar el circuito que represente dicha tabla escribiremos dicha tabla en función de sus entradas como una función lógica, primero para las entradas en donde la salida corresponde a "1" y luego para las salidas en donde Cout resulta "1".

Obtenida la función lógica, se decide no simplificar mediante ningún método, expresando el circuito solo con elementos de 2 entradas en su forma extendida.

Luego de realizarse el circuito, se analiza la entrada A=0, B=1, Cout=0. De lo cual se espera una salida S=1 y un Cout resultante igual a 0.Siendo correcto el resultado, se verifica la correcta estructura del circuito.



1.2 Diagrama temporal para un full-adder

Realizado el diagrama temporal, se observa la directa representación de una tabla de verdad pero en sentido del tiempo, siendo representada en dicho tiempo la salida S y Cout ante las diferentes entradas.
Sí se puede “reemplazar” una tabla de verdad con un diagrama temporal, y para pasar un diagrama temporal a una tabla de verdad es posible SOLO en el caso de estar seguros que dicho diagrama muestra todas las posibles combinaciones de entradas. Si ese no es el caso, la tabla quedará incompleta provocando falta de información y provocando que no se pueda analizar y/o modelar correctamente un circuito.

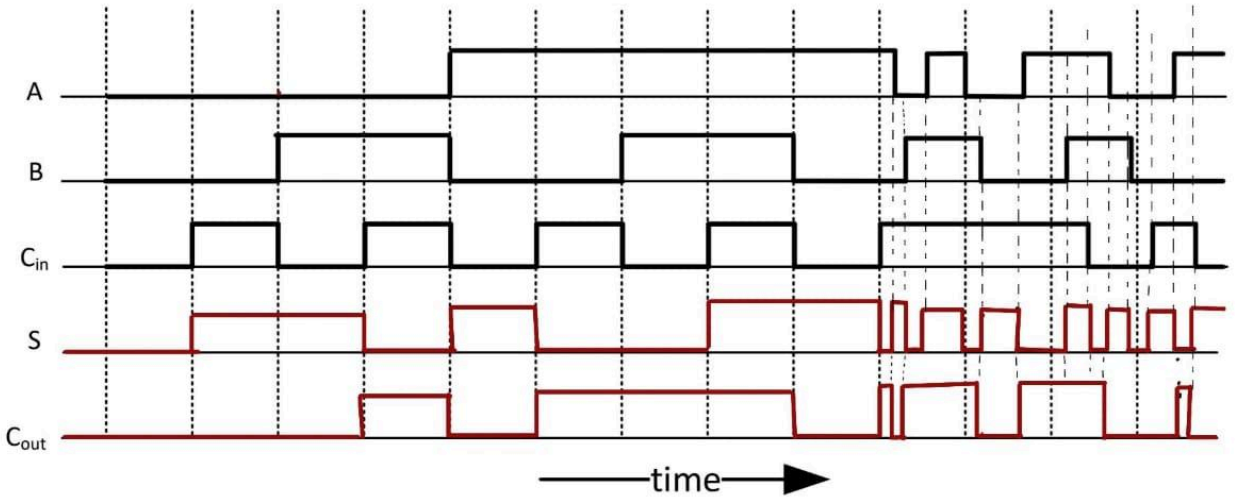
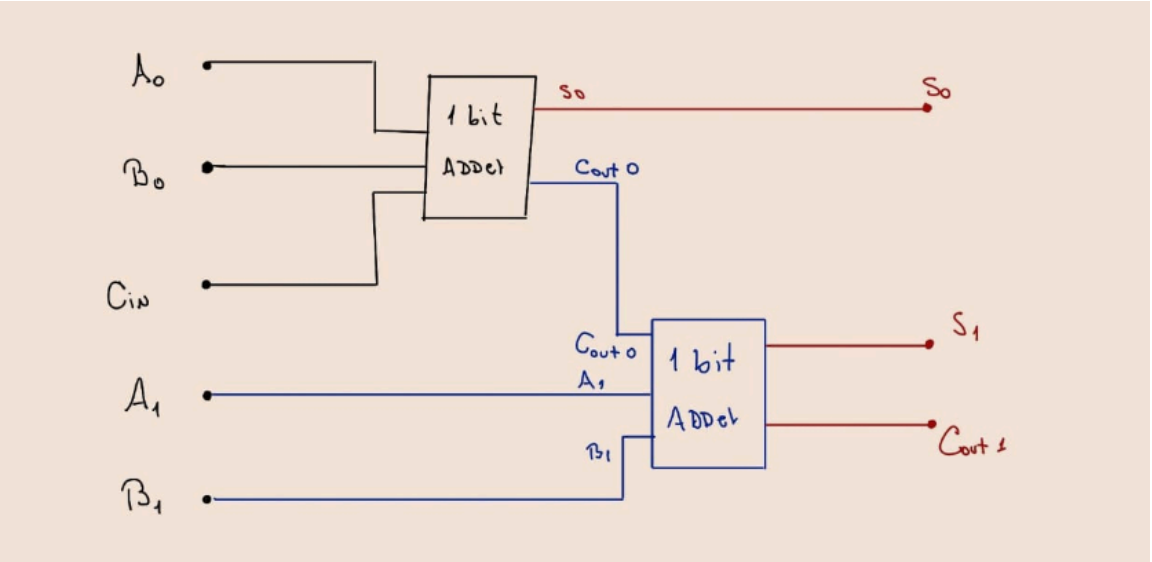


Figura 1: Diagrama temporal de un full-adder.

1.3 Extensión a sumador de múltiples bits

Para ampliar el esquema obtenido (basado en los sumadores ya planteados anteriormente para 1 bit) se escala la estructura utilizando dichos sumadores como si fueran “white box”. Si se quiere hacer un sumador de 32 bits habría que crear un nuevo esquema en el que aumenta la probabilidad de fallar al crearlo o al calcular sus resultados. Para disminuir su complejidad se puede modularizar el circuito según las operaciones ya calculadas y sus representaciones en circuitos que ya son conocidos, disminuyendo así su tamaño y priorizando la simplicidad (abstracción) de la solución. Sin embargo, a pesar de simplificar el circuito mediante la abstracción del mismo, su complejidad aumenta a medida que se aumenta el número de bits, aumentando en este casó el numero de “adders” que se deberán usar e interconectar en cascada.



Sumador de múltiples bits utilizando sumados de 1 bit

A-0	A-1	B-0	B-1	Cin		S0	s1	cout
0	0	0	0	0		0	0	0
0	0	0	0	1		1	1	0
0	0	0	1	0		1	1	0
0	0	0	1	1		1	0	0
0	0	1	0	0		1	0	0
0	0	1	0	1		0	1	1
0	0	1	1	0		1	1	0
0	0	1	1	1		0	0	1
0	1	0	0	0		0	1	0
0	1	0	0	1		1	0	0
0	1	0	1	0		0	0	0
0	1	0	1	1		1	1	0
0	1	1	0	0		1	1	0
0	1	1	0	1		0	0	1
0	1	1	1	0		1	0	0
0	1	1	1	1		0	1	1
1	0	0	0	0		1	0	0
1	0	0	0	1		0	1	1
1	0	0	1	0		1	1	0
1	0	0	1	1		0	0	1
1	0	1	0	0		0	0	1
1	0	1	0	1		0	1	1
1	0	1	1	0		0	1	1
1	0	1	1	1		0	0	1
1	1	0	0	0		1	1	0
1	1	0	0	1		0	0	1
1	1	0	1	0		1	0	0
1	1	0	1	1		0	1	1
1	1	1	0	0		0	1	1
1	1	1	0	1		0	0	1
1	1	1	1	0		0	0	1
1	1	1	1	1		1	1	1

Tabla de verdad sumador de 2 bit

1.4

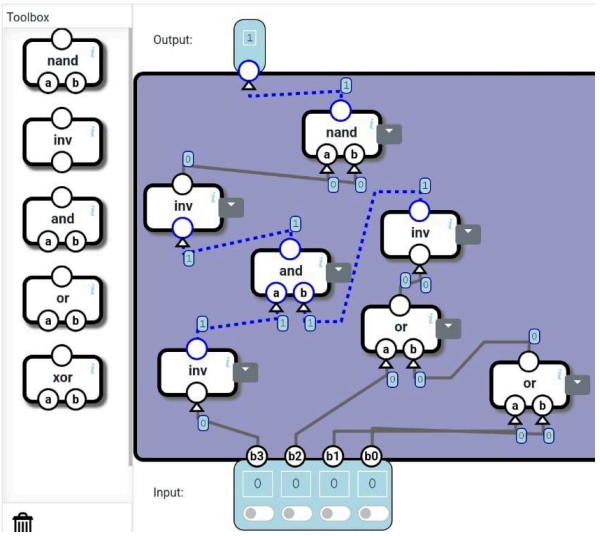
The NAND game

Logic Gates	Arithmetics
✓ Nand	✓ Half Adder
✓ Invert	✓ Full Adder
✓ And	✓ Multi-bit Adder
✓ Or	✓ Increment
✓ Xor	✓ Subtraction
	✓ Equal to Zero

Podemos observar que la complejidad de las tablas de verdades resultantes va aumentando, a medida que se van sumando compuertas más complejas, estas mismas compuertas (por ej ADDER) están hechas en base a las compuertas lógicas más básicas que son AND, NOT y OR, lo cual presenta una ventaja a la hora de hacer niveles más altos, ya que el diseño del circuito no resulta tan engorroso.
La siguiente imagen corresponde al nivel 11 de NAND game (incluida como evidencia del desarrollo):

Equal to Zero

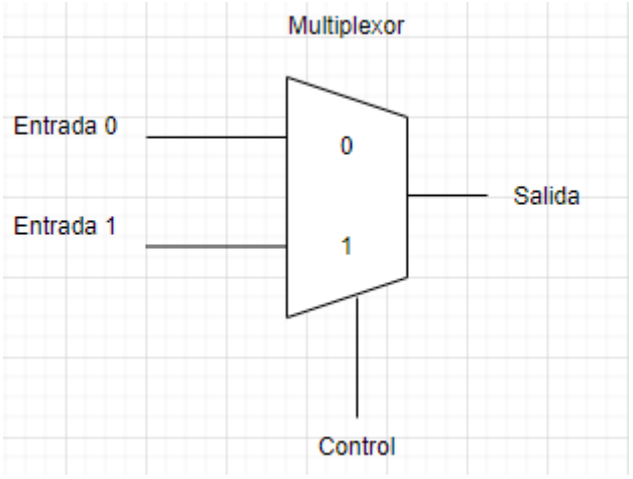
Should output 1 if and only if all bits in the input are 0.



1.5

- Multiplexores (MUX):

Tiene múltiples entradas y solo una salida, además una señal de control que dependiendo de su valor, selecciona una entrada diferente para enviarlo a la salida.Prácticamente funciona como las luces de una casa y su interruptor siendo la señal de control.



● Look-up Table:

Una tabla de verdad, Es un registro que tiene todas las entradas posibles (Todos los casos existentes) y además te entrega la salida asociada a esta entrada.

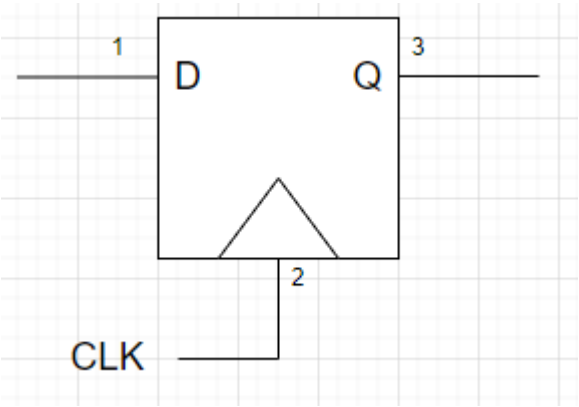
Ejemplo Look-up table AND:

Input A	Input B	Output C
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

● Flip Flop Tipo D

Un Flip flop se encarga de recibir un bit de información , almacenarlo según indica una entrada CLK (Clock), y finalmente manda esa información a la salida.

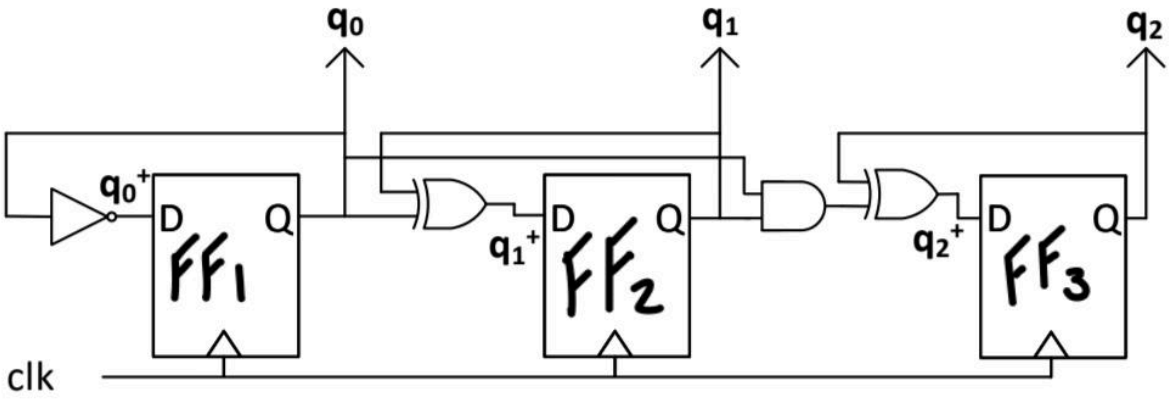
Ejemplo Flip Flop



1 Llega el bit de información (D)
2 Luego de obtener la señal de CLK, el bit de información es almacenado
3 Finalmente el bit es enviado hacia la salida (Q)

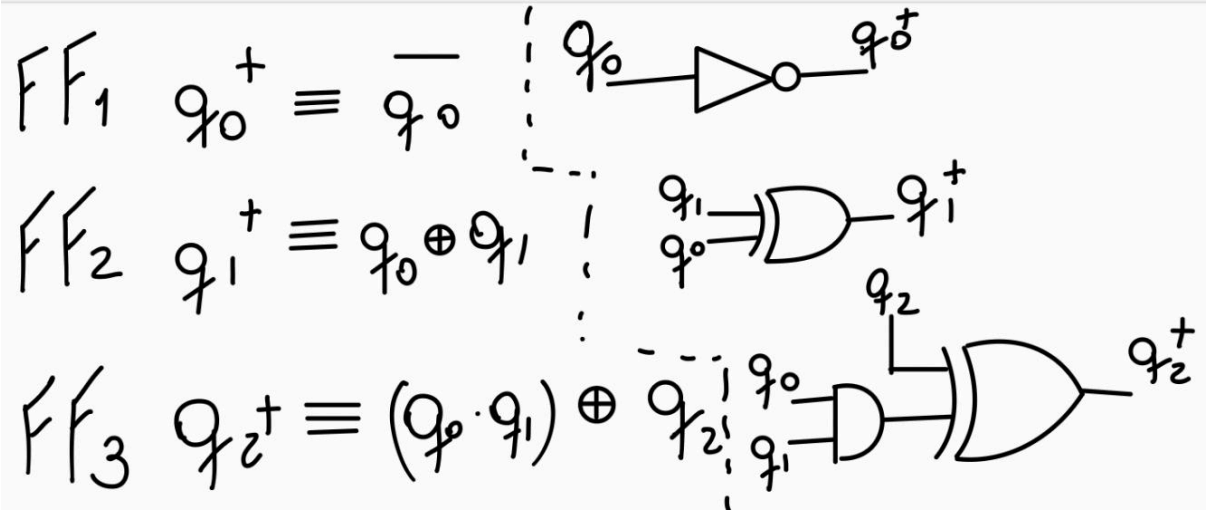
Diagrama temporal:

Luego de analizar la figura, a cada Flip FLop le asignaremos los siguientes nombres:



Después obtenemos las siguientes operaciones y compuertas logicas con el siguiente formato:

FF#/Operación Lógica/Compuerta



Para finalmente, Obtener el siguiente diagrama temporal, considerando que antes del primer clk q0 y q1 están en alto (Valor Lógico 1) y q2 está en bajo (Valor lógico 0):

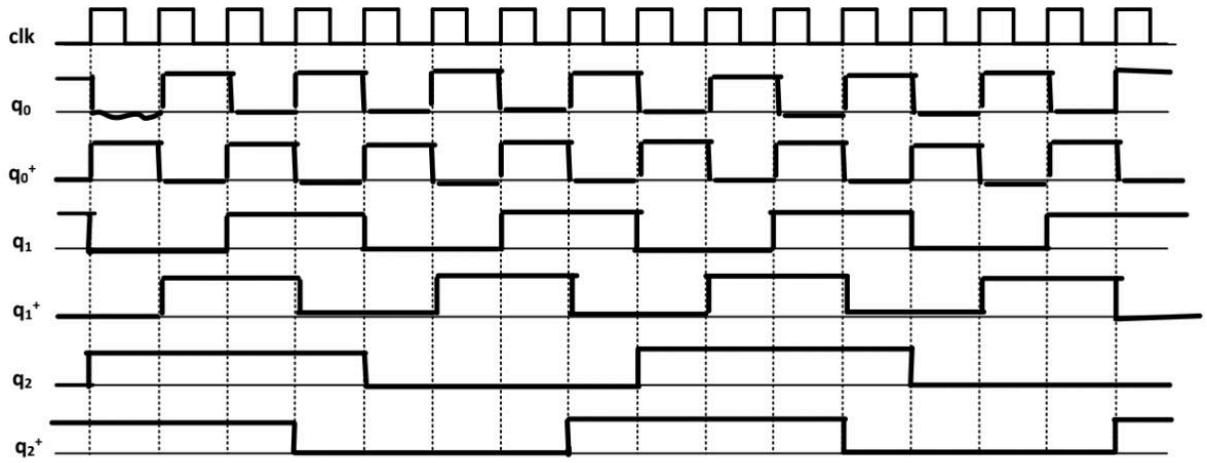


Figura 3: Diagrama temporal del esquemático con FF-D.