

Símbolos de lenguaje Ladder

—|— Normalmente abierto (NA)

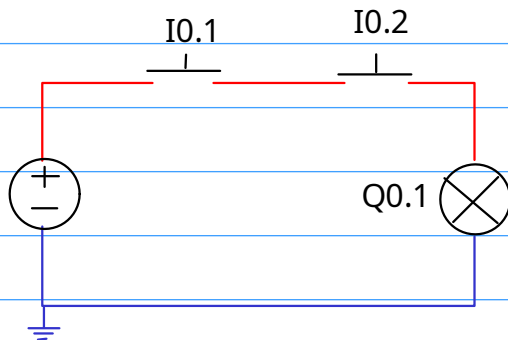
—|/— Normalmente cerrado (NC)

—()— Coil, representación de una salida

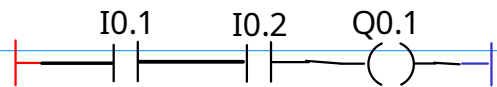
—| —| Rail, rieles de referencia

Ejemplo de un circuito eléctrico en lógica cableada y lenguaje ladder

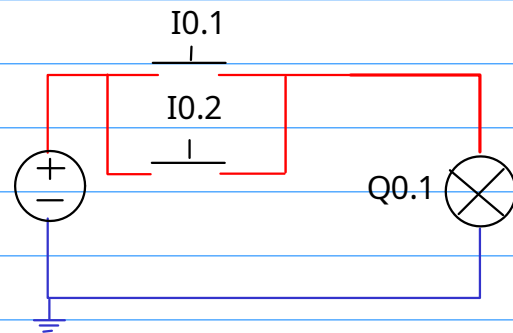
Eje1:



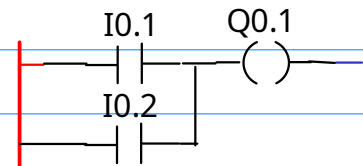
Lógica cableada



Eje2:

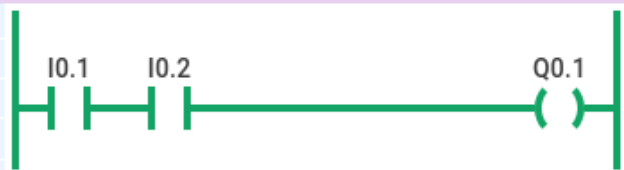


Lógica cableada



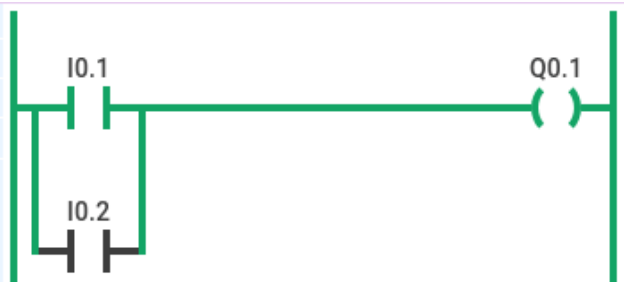
Del Eje1: Del Ej1, se deben pulsar (true) ambos contactos (I0.1 e I0.2) para que se active Q0.1

Name	Type	Value
I0.1	Bool	True
I0.2	Bool	True
Q0.1	Bool	True



Del Eje2: Del Ej2, con que al menos un contacto (I0.1 o I0.2) esté pulsado (True) se activa active Q0.1

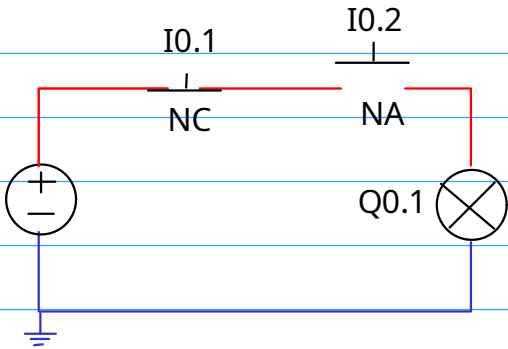
Name	Type	Value
I0.1	Bool	True
I0.2	Bool	False
Q0.1	Bool	True



La simulación en lenguaje ladder la puede realizar en:
<https://app.plcsimulator.online/>

Otros ejemplos de lógica cableada y de lenguaje Ladder

Eje3:

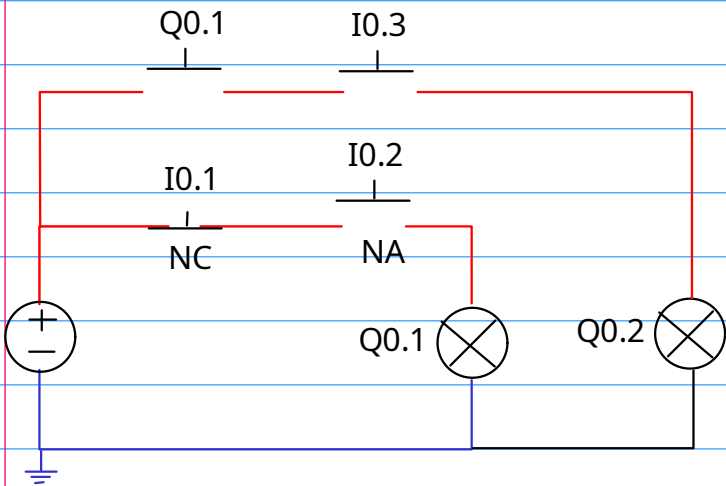


Del Eje3, para que Q0.1 se active (True)
Se debe pulsar I0.2 y dejar a I0.2 sin pulsar.
Observe que I0.1 es un contacto normalmente cerrado (NC) y que I0.2 es un contacto normalmente abierto (NA)

Name	Type	Value
I0.1	Bool	False
I0.2	Bool	True
Q0.1	Bool	True

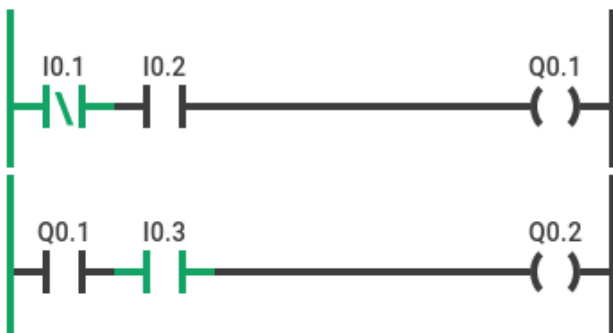


Eje4:

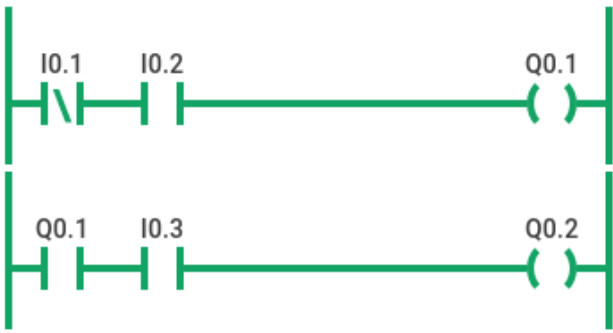


Teniendo presente el funcionamiento del Eje3, observe que en el Ej4, para activar Q0.2, se requiere que Q0.1 esté activo junto con I0.3. Observe los valores lógicos tomados por las variables en función de la activaciones en los ejemplos de abajo.

Name	Type	Value
I0.1	Bool	False
I0.2	Bool	False
Q0.1	Bool	False
I0.3	Bool	True
Q0.2	Bool	False



Name	Type	Value
I0.1	Bool	False
I0.2	Bool	True
Q0.1	Bool	True
I0.3	Bool	True
Q0.2	Bool	True



Compuertas lógicas, símbolos, lógica de contacto, ecuación y tabla de verdad

AND

Símbolo:

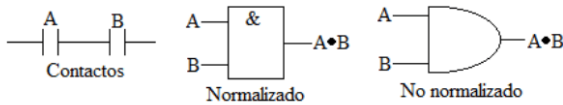


Figura 7.2.: Compuerta lógica AND

Ecuación:

$$S = A \bullet B$$

Tabla de verdad:

Tabla 7.2.: Compuerta lógica AND

Entrada A	Entrada B	Salida S
0	0	0
0	1	0
1	0	0
1	1	1

Lenguaje verilog:

```
assign s = a & b; // Operador AND lógico
```

Ejemplo de uso:

```
module and_gate (
    input a,
    input b,
    output s
);
    assign s = a & b; // Operador AND lógico
endmodule
```

OR

Símbolo:

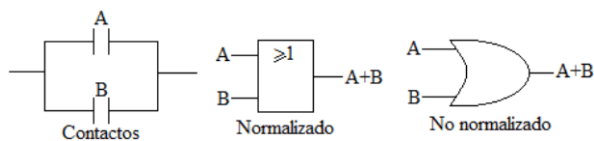


Figura 7.1.: Compuerta lógica OR

Ecuación:

$$S = A + B$$

Tabla de verdad:

Tabla 7.1.: Compuerta lógica OR

Entrada A	Entrada B	Salida S
0	0	0
0	1	1
1	0	1
1	1	1

Lenguaje verilog:

```
assign s = a | b; // Operador OR lógico
```

Ejemplo de uso:

```
module or_gate (
    input a,
    input b,
    output s
);
    assign s = a | b; // Operador OR lógico
endmodule
```

NOT

Símbolo:

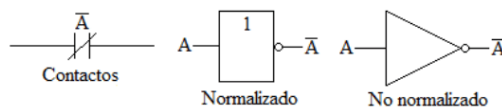


Figura 7.3.: Compuerta lógica NOT

Ecuación:

$$S = \bar{A}$$

Tabla de verdad:

Tabla 7.3.: Compuerta lógica NOT

Entrada A	Salida S
0	1
1	0

Lenguaje verilog:

```
assign s = ~a; // Operador NOT lógico
```

Ejemplo de uso:

```
module not_gate (
    input a,
    output s
);
    assign s = ~a; // Operador NOT lógico
endmodule
```

Compuertas lógicas, símbolos, lógica de contacto, ecuación y tabla de verdad

NAND

Símbolo:

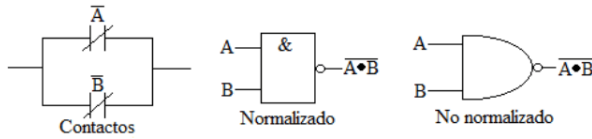


Figura 7.4.: Compuerta lógica NAND

Ecuación:

$$S = \overline{A \cdot B} = \overline{A} + \overline{B}$$

Tabla de verdad:

Tabla 7.4.: Compuerta lógica NAND

Entrada A	Entrada B	Salida S
0	0	1
0	1	1
1	0	1
1	1	0

Lenguaje verilog:

```
assign s = ~(a & b); // Operador NAND lógico
```

Ejemplo de uso:

```
module nand_gate (  
    input a,  
    input b,  
    output s  
);  
    assign s = ~(a & b); // Operador NAND lógico  
endmodule
```

NOR

Símbolo:

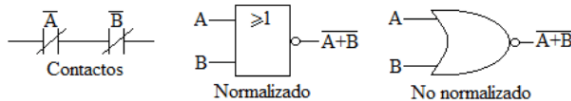


Figura 7.5.: Compuerta lógica NOR

Ecuación:

$$S = \overline{A + B} = \overline{A} \cdot \overline{B}$$

Tabla de verdad:

Tabla 7.5.: Compuerta lógica NOR

Entrada A	Entrada B	Salida S
0	0	1
0	1	0
1	0	0
1	1	0

Lenguaje verilog:

```
assign s = ~(a | b); // Operador NOR lógico
```

Ejemplo de uso:

```
module nor_gate (  
    input a,  
    input b,  
    output s  
);  
    assign s = ~(a | b); // Operador NOR lógico  
endmodule
```

XOR

Símbolo:

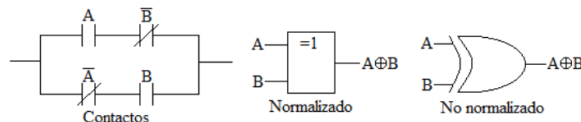


Figura 7.6.: Compuerta lógica XOR

Ecuación:

$$S = A \oplus B$$

$$S = \overline{A} \cdot B + A \cdot \overline{B}$$

Tabla de verdad:

Tabla 7.6.: Compuerta lógica XOR

Entrada A	Entrada B	Salida S
0	0	0
0	1	1
1	0	1
1	1	0

Lenguaje verilog:

```
assign s = a ^ b; // Operador XOR lógico
```

Ejemplo de uso:

```
module xor_gate (  
    input a,  
    input b,  
    output s  
);  
    assign s = a ^ b; // Operador XOR lógico  
endmodule
```

Compuertas lógicas, símbolos, lógica de contacto, ecuación y tabla de verdad

XNOR

Símbolo:

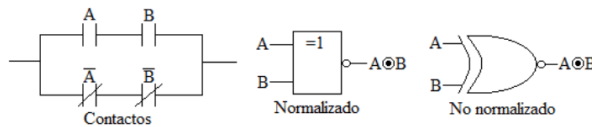


Figura 7.7.: Compuerta lógica XNOR

Ecuación:

$$S = \overline{A \oplus B}$$

Tabla de verdad:

Tabla 7.7.: Compuerta lógica XNOR

Entrada A	Entrada B	Salida S
0	0	1
0	1	0
1	0	0
1	1	1

Lenguaje verilog:

```
assign s = ~(a ^ b); // Operador XNOR lógico
```

Ejemplo de uso:

```
module xnor_gate (  
    input a,  
    input b,  
    output s  
);  
    assign s = ~(a ^ b); // Operador XNOR lógico  
endmodule
```

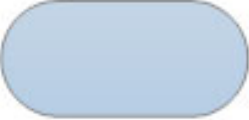

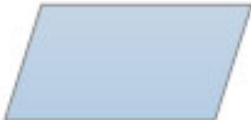


Crear un archivo con el nombre gates_test.v con el siguiente contenido

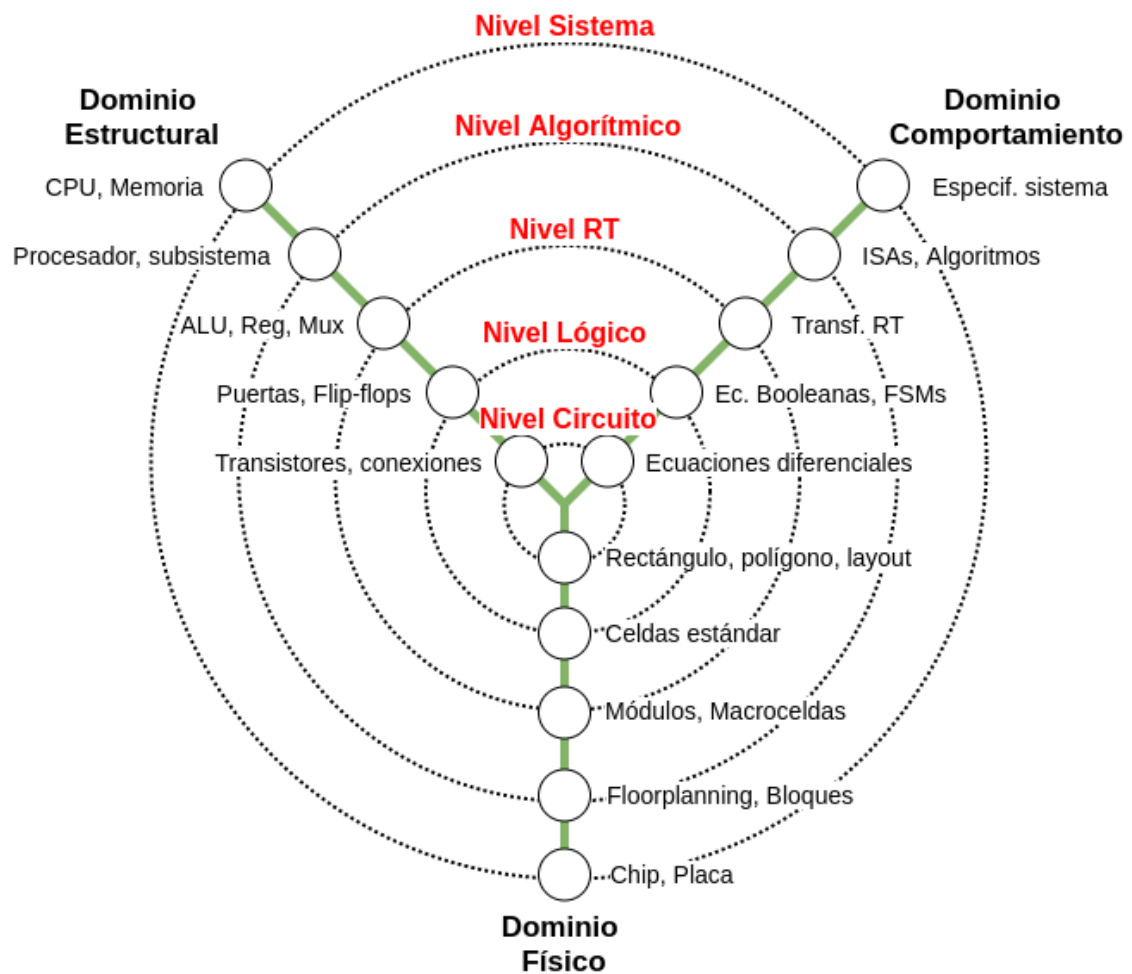
```
// filename: gates_test.v  
// Compuerta AND  
module and_gate (  
    input a,  
    input b,  
    output s  
);  
    assign s = a & b; // Operador AND lógico  
endmodule  
  
// Compuerta OR  
module or_gate (  
    input a,  
    input b,  
    output s  
);  
    assign s = a | b; // Operador OR lógico  
endmodule  
  
// Compuerta XOR  
module xor_gate (  
    input a,  
    input b,  
    output s  
);  
    assign s = a ^ b; // Operador XOR lógico  
endmodule  
  
// .. continúa aquí el archivo  
// Módulo de prueba  
module test_gates;  
    reg a, b; // Entradas  
    wire and_s, or_s, xor_s; // Salidas  
  
    // Instanciación de las compuertas  
    and_gate u1 (.a(a), .b(b), .s(and_s)); // Compuerta AND  
    or_gate u2 (.a(a), .b(b), .s(or_s)); // Compuerta OR  
    xor_gate u3 (.a(a), .b(b), .s(xor_s)); // Compuerta XOR  
  
    // Bloque inicial para probar diferentes combinaciones  
    initial begin  
        a = 0; b = 0; #10;  
        a = 0; b = 1; #10;  
        a = 1; b = 0; #10;  
        a = 1; b = 1; #10;  
        $finish; // Terminar la simulación  
    end  
  
    // Monitor para mostrar los resultados  
    initial begin  
        $monitor("Time=%0d, a=%b, b=%b, AND=%b, OR=%b, XOR=%b",  
            $time, a, b, and_s, or_s, xor_s);  
    end  
end
```

Comando iverilog en terminal para ejecutar el ejemplo:

```
iverilog -o gates_test_tb gates_test.v  
vvp gates_test_tb
```

Símbolos para diagramas de flujo

Símbolo	Nombre	Función
	Inicio / Final	Representa el inicio y el final de un proceso
	Línea de Flujo	Indica el orden de la ejecución de las operaciones. La flecha indica la siguiente instrucción.
	Entrada / Salida	Representa la lectura de datos en la entrada y la impresión de datos en la salida
	Proceso	Representa cualquier tipo de operación
	Decisión	Nos permite analizar una situación, con base en los valores verdadero y falso



Nivel de Abstracción	Valores	Medidas
Sistema	Relaciones entre subsistemas, sincronización y protocolos.	Ancho de banda, MIPS.
Algorítmico	Estructuras abstractas. Se usan las dependencias en lugar del tiempo.	Latencia, cadencia de datos, número de módulos.
RT (Register Transfer)	Palabras con valores discretos. Control y procesamiento en tiempo discreto.	Tiempos de ciclo, márgenes y puertas equivalentes.
Lógico	Valores lógicos. Computación en tiempo continuo.	Tiempos de conmutación, Skew y áreas equivalentes.
Circuito	Valores continuos. Todo es electrónica en tiempo continuo.	Tiempos de subida, bajada y consumos de área.