PROYECTO 1

Presentado por:

Isabella Ocampo Sánchez

Curso:

Organización de computadores

Profesor:

Jose Luis Montoya Pareja

Universidad EAFIT

Ingeniería de Sistemas

Medellín, Colombia

2026

UNIVERSIDAD EAFIT

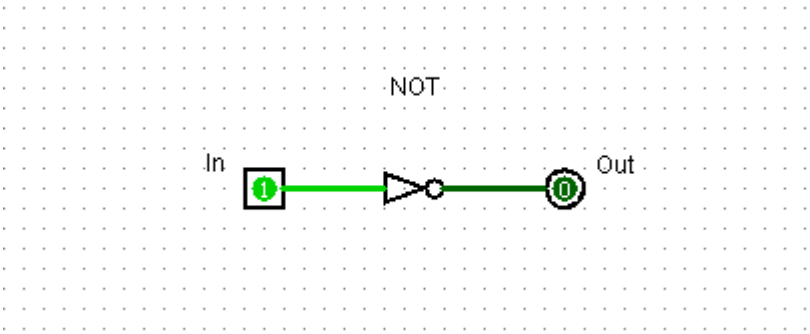
Tabla de Contenido

CHIP NOT

•Not

in	out
0	1
1	0

```
CHIP Not {

   IN in;

   OUT out;


   PARTS:

   Nand(a=in, b=in , out= out);

}
```



CHIP AND

• AND

a	b	out
0	0	0
0	1	0
1	0	0
1	1	1

CHIP And {

IN a, b;

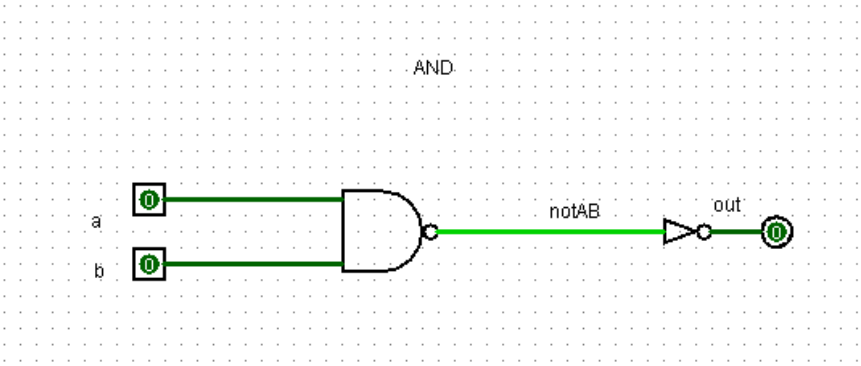
OUT out;

PARTS:

Nand(a=a , b=b , out=nand1);

Not(in=nand1 , out= out);

}

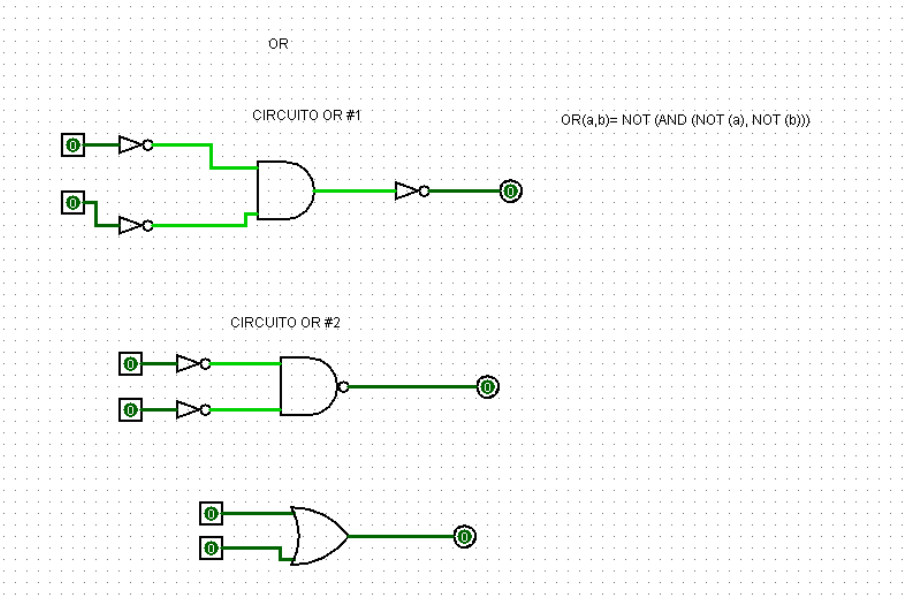


CHIP OR

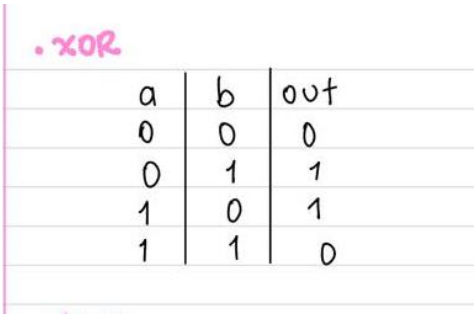
•OR

a	b	out
0	0	0
0	1	1
1	0	1
1	1	1

```
CHIP Or {

   IN a, b;

   OUT out;


   PARTS:

   Not(in=a , out=notA );

   Not(in=b , out=notB );


   Nand(a=notA, b=notB , out=out );

}
```



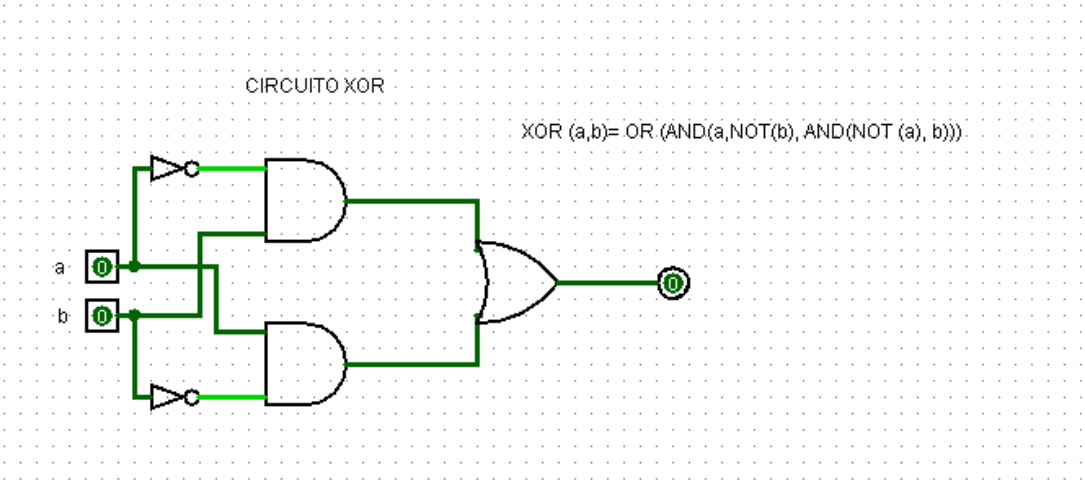
CHIP XOR



• XOR

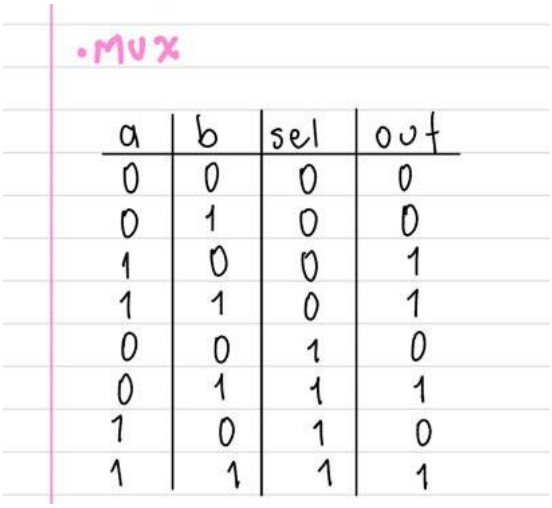
a	b	out
0	0	0
0	1	1
1	0	1
1	1	0

```
CHIP Xor {

   IN a, b;

   OUT out;

   PARTS:

   Not(in=a , out=notA );

   Not(in=b , out=notB );


   And(a=notA , b=b, out=outand1 );

   And(a=a , b=notB, out=outand2 );

   Or(a=outand1 , b=outand2 , out=out );

}
```



UNIVERSIDAD EAFIT

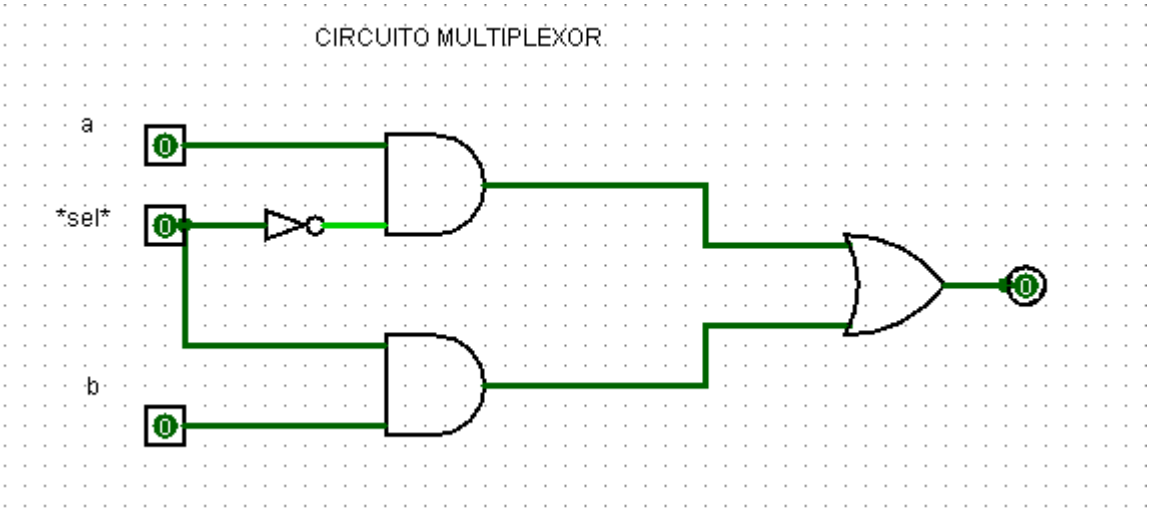
CHIP MUX



•MUX

a	b	sel	out
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	0	1	0
0	1	1	1
1	0	1	0
1	1	1	1

```
CHIP Mux {

    IN a, b, sel;

    OUT out;


    PARTS:

    Not(in=sel, out=notSel);


    And(a=a, b=notSel, out=andA);

    And(a=b, b=sel, out=andB);


    Or(a=andA, b=andB, out=out);

}
```



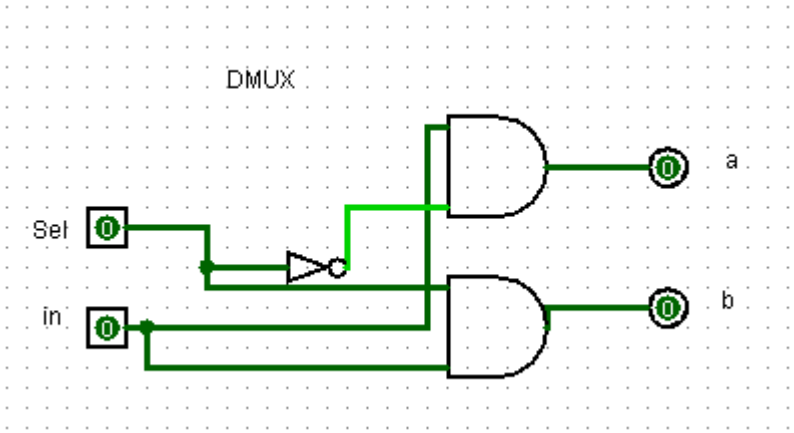
CHIP DMUX

• DMUX

in	sel	a	b
0	0	0	0
1	0	1	0
0	1	0	0
1	1	0	1

```
CHIP DMux {

    IN in, sel;

    OUT a, b;


    PARTS:

    Not(in=sel , out=Notsel );

    And(a=in , b=Notsel, out=a );

    And(a=in , b=sel, out=b );
```

}



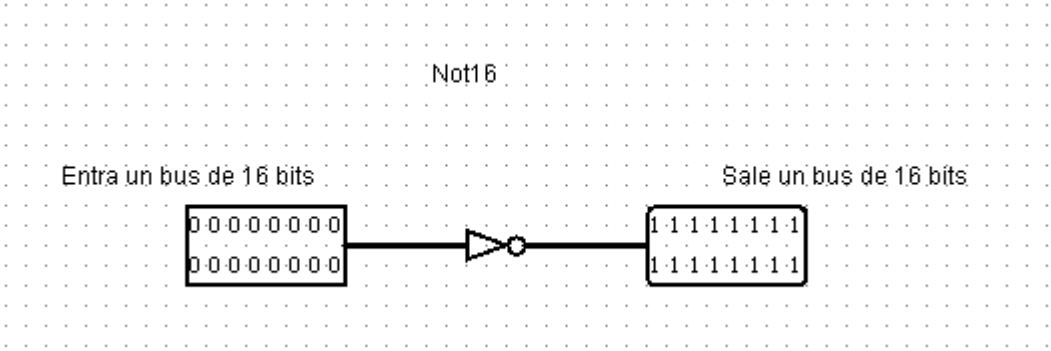
CHIP NOT 16

- NOT 16

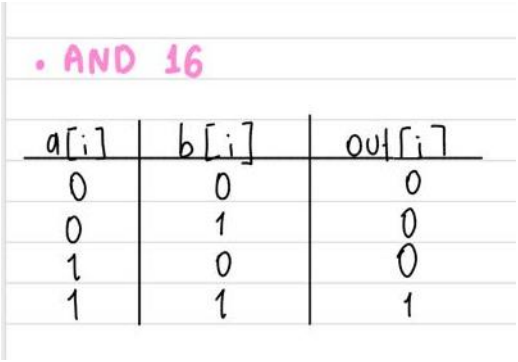
Entrada (in[i])	Salida (out[i])
0	1
1	0

```
CHIP Not16 {

    IN in[16];

    OUT out[16];


    PARTS:

    Not(in=in[0] , out=out[0]);

    Not(in=in[1] , out=out[1]);

    Not(in=in[2] , out=out[2]);

    Not(in=in[3] , out=out[3]);

    Not(in=in[4] , out=out[4]);

    Not(in=in[5] , out=out[5]);
```

```
    Not(in=in[6] , out=out[6]);
    Not(in=in[7] , out=out[7]);
    Not(in=in[8] , out=out[8]);
    Not(in=in[9] , out=out[9]);
    Not(in=in[10] , out=out[10]);
    Not(in=in[11] , out=out[11]);
    Not(in=in[12] , out=out[12]);
    Not(in=in[13] , out=out[13]);
    Not(in=in[14] , out=out[14]);
    Not(in=in[15] , out=out[15]);
}
```

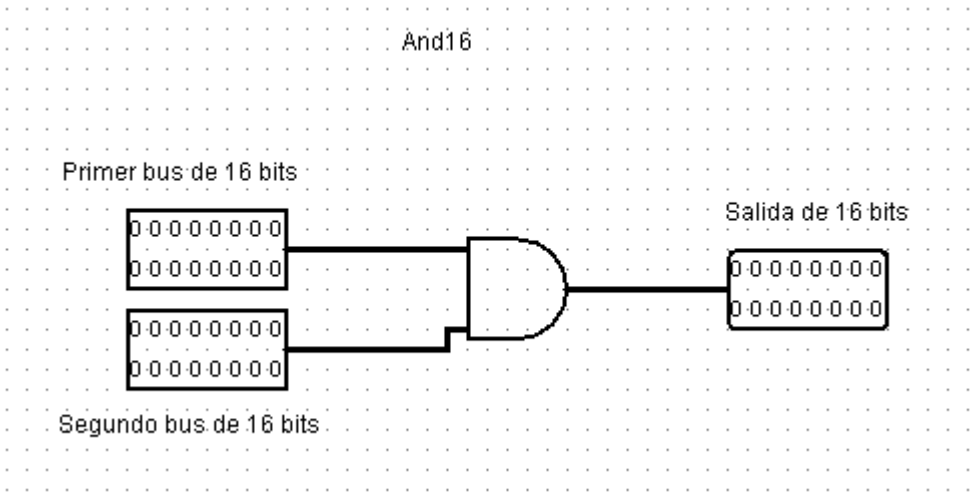


CHIP AND 16




```
CHIP And16 {
    IN a[16], b[16];
```

```
    OUT out[16];

    PARTS:
    And(a=a[0] , b= b[0], out= out[0]);
    And(a=a[1] , b= b[1], out= out[1]);
    And(a=a[2] , b= b[2], out= out[2]);
    And(a=a[3] , b= b[3], out= out[3]);
    And(a=a[4] , b= b[4], out= out[4]);
    And(a=a[5] , b= b[5], out= out[5]);
    And(a=a[6] , b= b[6], out= out[6]);
    And(a=a[7] , b= b[7], out= out[7]);
    And(a=a[8] , b= b[8], out= out[8]);
    And(a=a[9] , b= b[9], out= out[9]);
    And(a=a[10] , b= b[10], out= out[10]);
    And(a=a[11] , b= b[11], out= out[11]);
    And(a=a[12] , b= b[12], out= out[12]);
    And(a=a[13] , b= b[13], out= out[13]);
    And(a=a[14] , b= b[14], out= out[14]);
    And(a=a[15] , b= b[15], out= out[15]);

}
```



CHIP OR 16

• OR 16

a[i]	b[i]	Out[i]
0	0	0
0	1	1
1	0	1
1	1	1

CHIP Or16 {

IN a[16], b[16];

OUT out[16];

PARTS:

Or(a=a[0] , b= b[0], out= out[0]);

Or(a=a[1] , b= b[1], out= out[1]);

Or(a=a[2] , b= b[2], out= out[2]);

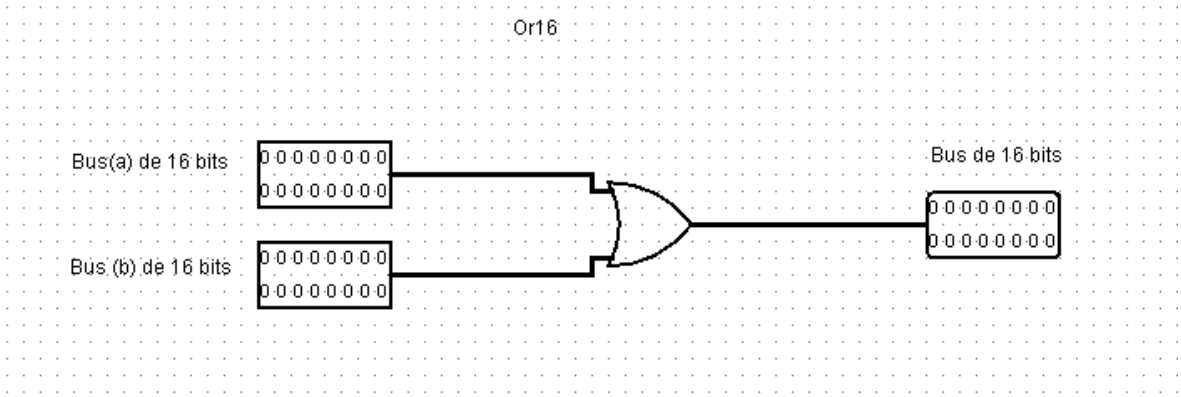
Or(a=a[3] , b= b[3], out= out[3]);

Or(a=a[4] , b= b[4], out= out[4]);

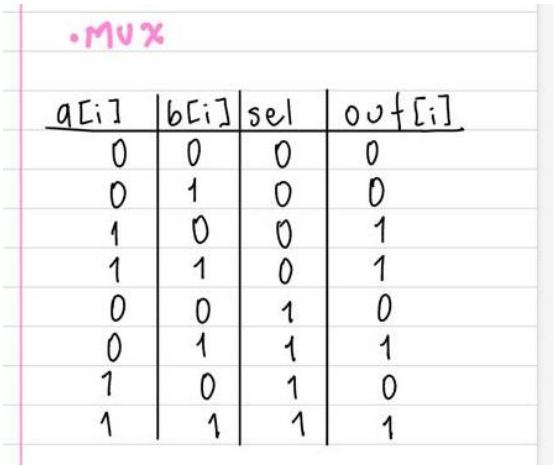
Or(a=a[5] , b= b[5], out= out[5]);

Or(a=a[6] , b= b[6], out= out[6]);

```
    Or(a=a[7] , b= b[7], out= out[7]);
    Or(a=a[8] , b= b[8], out= out[8]);
    Or(a=a[9] , b= b[9], out= out[9]);
    Or(a=a[10] , b= b[10], out= out[10]);
    Or(a=a[11] , b= b[11], out= out[11]);
    Or(a=a[12] , b= b[12], out= out[12]);
    Or(a=a[13] , b= b[13], out= out[13]);
    Or(a=a[14] , b= b[14], out= out[14]);
    Or(a=a[15] , b= b[15], out= out[15]);
}
```



CHIP MUX 16



•MUX

a[i]	b[i]	sel	out[i]
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	0	1	0
0	1	1	1
1	0	1	0
1	1	1	1

UNIVERSIDAD EAFIT

```
CHIP Mux16 {
    IN a[16], b[16], sel;
    OUT out[16];

    PARTS:
    Not(in=sel , out= Notsel);
    And(a=a[0] , b=Notsel , out=andA1 );
    And(a=a[1] , b=Notsel , out=andA2 );
    And(a=a[2] , b=Notsel , out=andA3 );
    And(a=a[3] , b=Notsel , out=andA4 );
    And(a=a[4] , b=Notsel , out=andA5 );
    And(a=a[5] , b=Notsel , out=andA6 );
    And(a=a[6] , b=Notsel , out=andA7 );
    And(a=a[7] , b=Notsel , out=andA8 );
    And(a=a[8] , b=Notsel , out=andA9 );
    And(a=a[9] , b=Notsel , out=andA10 );
    And(a=a[10] , b=Notsel , out=andA11 );
    And(a=a[11] , b=Notsel , out=andA12 );
    And(a=a[12] , b=Notsel , out=andA13);
    And(a=a[13] , b=Notsel , out=andA14);
    And(a=a[14] , b=Notsel , out=andA15 );
    And(a=a[15] , b=Notsel , out=andA16);

    And(a=b[0] , b=sel , out=andB1 );
    And(a=b[1] , b=sel , out=andB2 );
```

```
And(a=b[2] , b=sel , out=andB3 );

And(a=b[3] , b=sel , out=andB4 );

And(a=b[4] , b=sel , out=andB5 );

And(a=b[5] , b=sel , out=andB6 );

And(a=b[6] , b=sel , out=andB7 );

And(a=b[7] , b=sel , out=andB8 );

And(a=b[8] , b=sel , out=andB9 );

And(a=b[9] , b=sel , out=andB10 );

And(a=b[10] , b=sel , out=andB11 );

And(a=b[11] , b=sel , out=andB12 );

And(a=b[12] , b=sel , out=andB13);

And(a=b[13] , b=sel , out=andB14);

And(a=b[14] , b=sel , out=andB15 );

And(a=b[15] , b=sel , out=andB16);


Or(a=andA1 , b= andB1, out= out[0]);

Or(a=andA2 , b= andB2, out= out[1]);

Or(a=andA3 , b= andB3, out= out[2]);

Or(a=andA4 , b= andB4, out= out[3]);

Or(a=andA5 , b= andB5, out= out[4]);

Or(a=andA6 , b= andB6, out= out[5]);

Or(a=andA7 , b= andB7, out= out[6]);

Or(a=andA8 , b= andB8, out= out[7]);

Or(a=andA9 , b= andB9, out= out[8]);

Or(a=andA10 , b= andB10, out= out[9]);

Or(a=andA11 , b= andB11, out= out[10]);
```

Or(a=andA12 , b= andB12, out= out[11]);

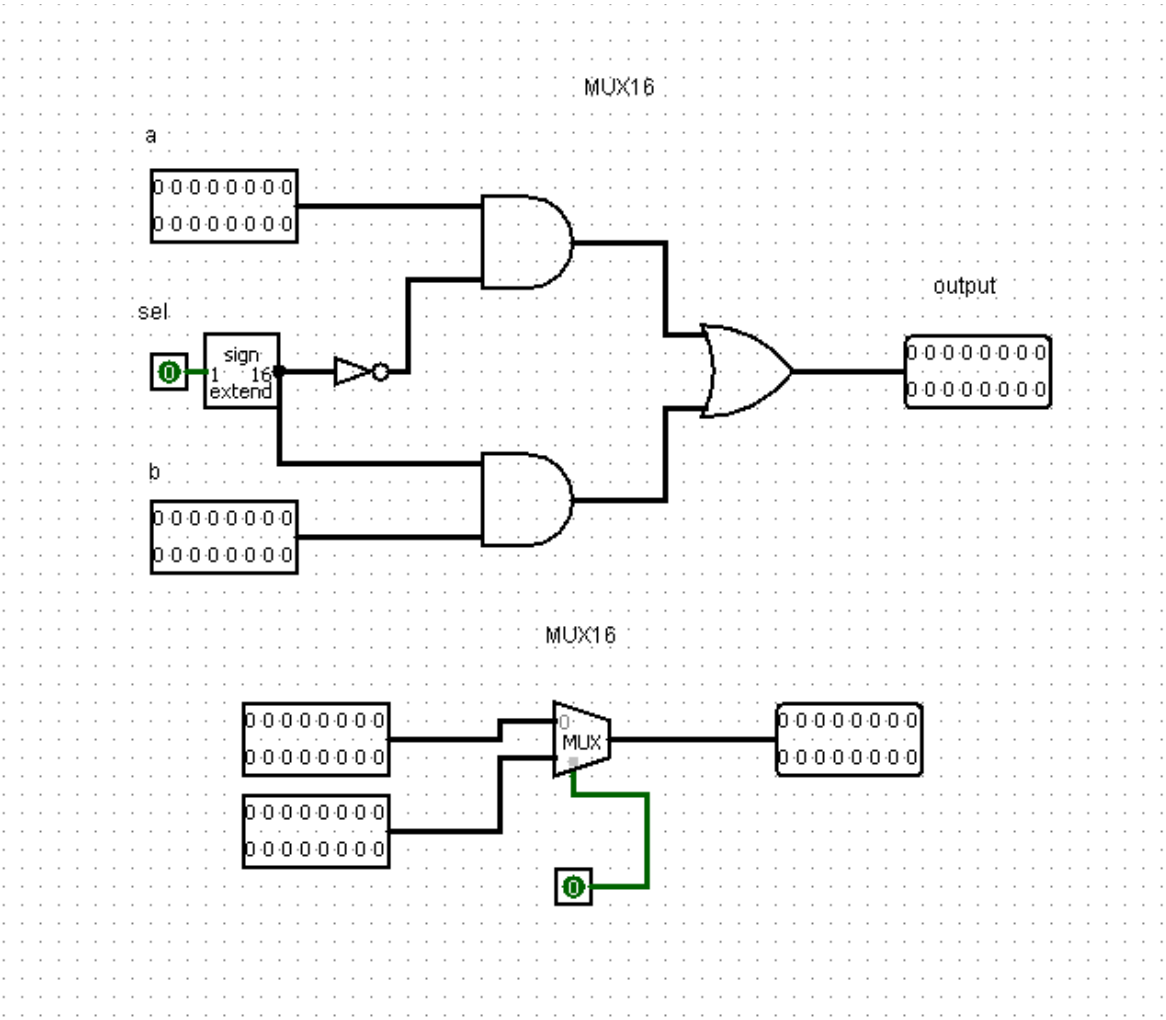
Or(a=andA13 , b= andB13, out= out[12]);

Or(a=andA14 , b= andB14, out= out[13]);

Or(a=andA15 , b= andB15, out= out[14]);

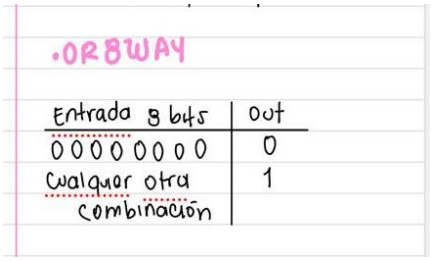
Or(a=andA16 , b= andB16, out= out[15]);

}



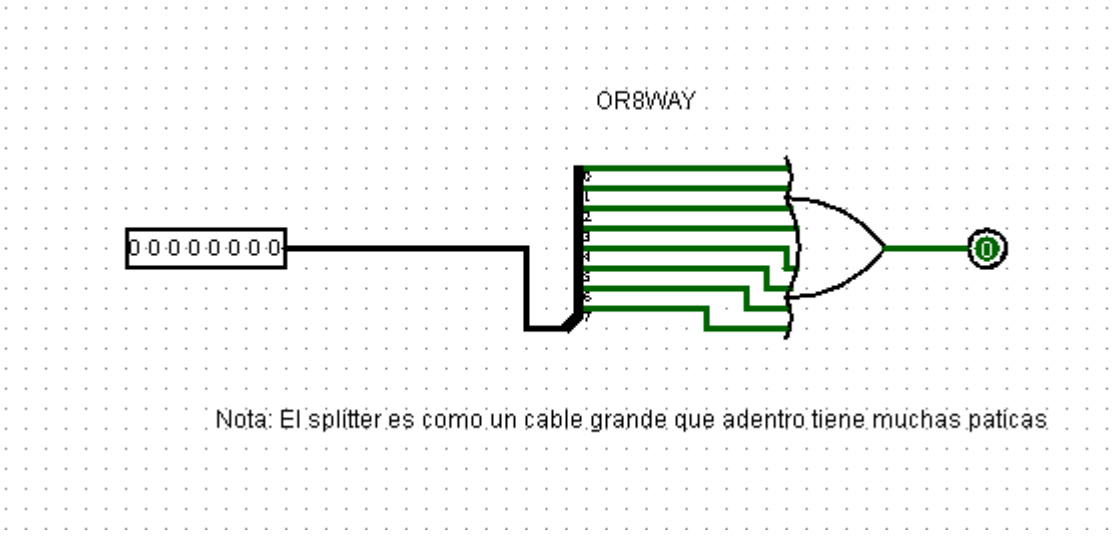
UNIVERSIDAD EAFIT

CHIP OR8WAY




```
CHIP Or8Way {

    IN in[8];

    OUT out;


    PARTS:

      /* FORMA CON NAND

       Not(in=in[0], out=not0 );

       Not(in=in[1], out=not1 );

       Not(in=in[2], out=not2 );

       Not(in=in[3], out=not3 );

       Not(in=in[4], out=not4 );

       Not(in=in[5], out=not5 );

       Not(in=in[6], out=not6 );

       Not(in=in[7], out=not7 );


       And(a=not0 , b= not1, out=not01 );

       And(a=not2 , b= not3, out=not23 );

       And(a=not4 , b= not5, out=not45 );

       And(a=not6 , b= not7, out=not67 );


       And(a=not01 , b= not23, out=nota1 );
```

```
And(a=not45 , b= not67, out=notb1 );


Nand(a=nota1 , b=notb1 , out=out );*/


   // FORMA SOLO CON OR
Or(a=in[0] , b=in[1] , out= or01);
Or(a=in[2] , b=in[3] , out= or23);
Or(a=in[4] , b=in[5] , out= or45);
Or(a=in[6] , b=in[7] , out= or67);


Or(a=or01 , b=or23 , out= ora1);
Or(a=or45 , b=or67 , out= ora2);


Or(a=ora1 , b=ora2 , out= out);


}
```

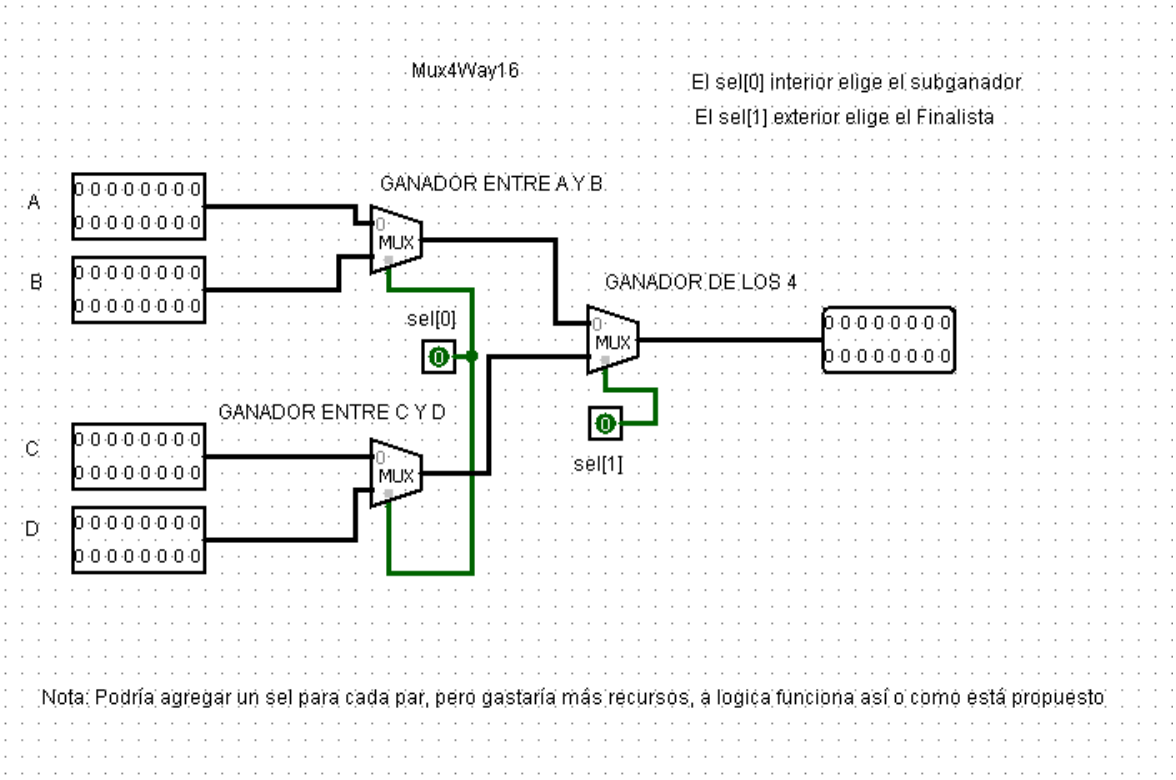


Nota: El splitter es como un cable grande que adentro tiene muchas patitas

UNIVERSIDAD EAFIT

CHIP MUX4WAY16


```
CHIP Mux4Way16 {

    IN a[16], b[16], c[16], d[16], sel[2];

    OUT out[16];


    PARTS:

    Mux16(a=a , b=b, sel=sel[0] , out=Ganadorm1 );

    Mux16(a=c , b=d, sel=sel[0] , out=Ganadorm2 );


    Mux16(a=Ganadorm1 , b=Ganadorm2, sel=sel[1] , out=out );

}
```



CHIP MUX8WAY16


```
CHIP Mux8Way16 {
   IN a[16], b[16], c[16], d[16],
      e[16], f[16], g[16], h[16],
      sel[3];
   OUT out[16];
```

PARTS:

Mux16(a=a , b=b , sel=sel[0] , out=ganador1);

Mux16(a=c , b=d , sel=sel[0] , out=ganador2);

Mux16(a=e , b=f , sel=sel[0] , out=ganador3);

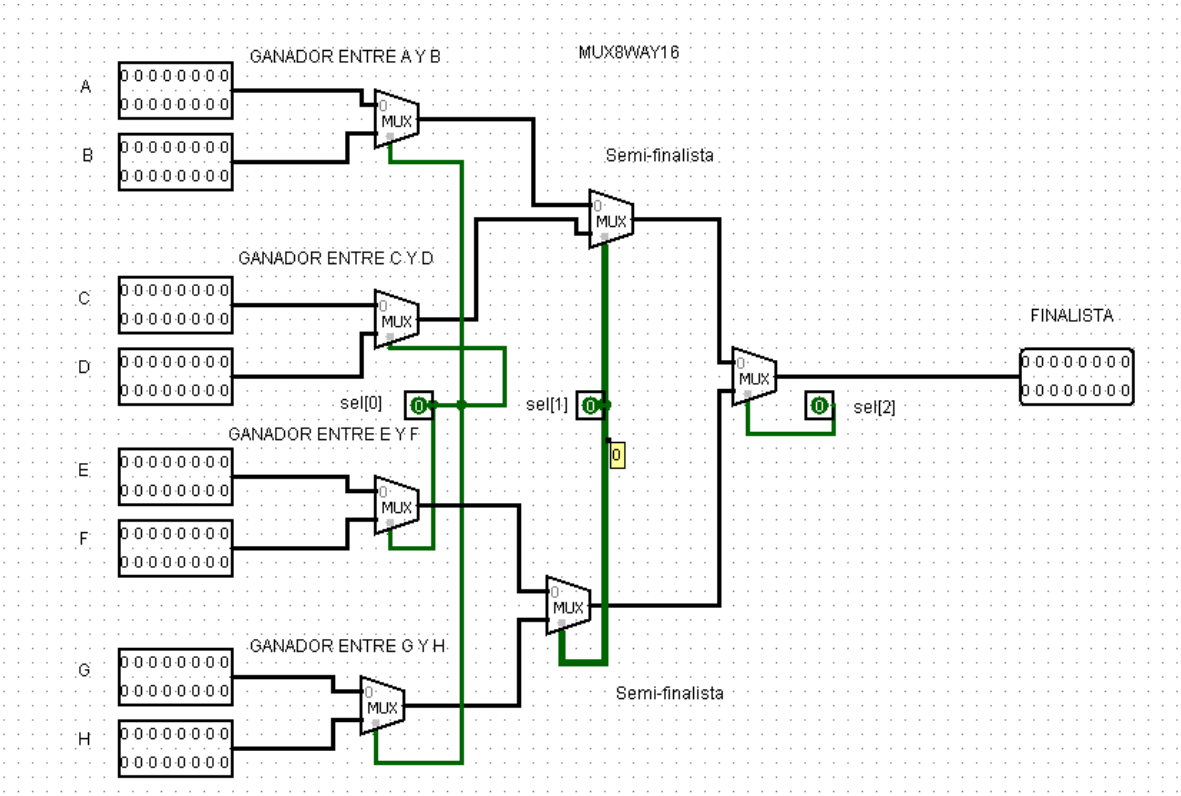
Mux16(a=g , b=h , sel=sel[0] , out=ganador4);

Mux16(a= ganador1, b= ganador2, sel= sel[1], out= semifinalista1);

Mux16(a= ganador3, b= ganador4, sel= sel[1], out=semifinalista2);

Mux16(a= semifinalista1, b= semifinalista2, sel= sel[2], out=out);

}

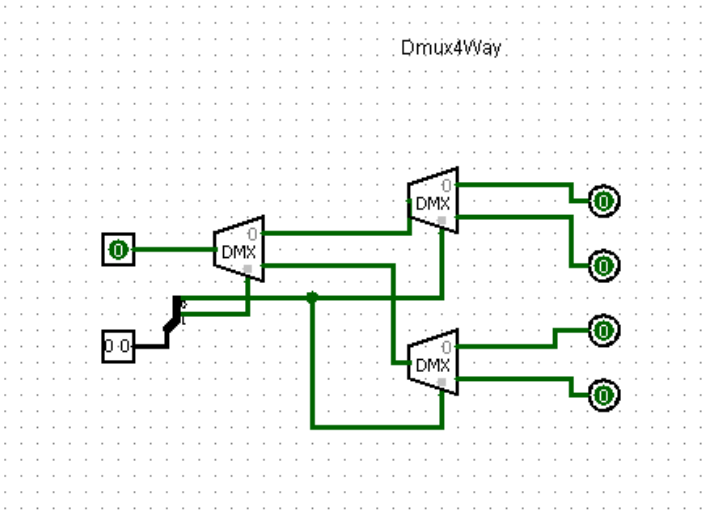


CHIP DMUX4WAY

· DMUX 4WAY

in	sel[1]	sel[0]	a	b	c	d
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	x	x	0	0	0	0

```
CHIP DMux4Way {

   IN in, sel[2];

   OUT a, b, c, d;


   PARTS:


   DMux(in=in , sel=sel[1] , a=grupoAB , b=grupoCD );

   DMux(in=grupoAB , sel=sel[0] , a= a, b=b );

   DMux(in=grupoCD , sel=sel[0] , a= c, b=d );


}
```



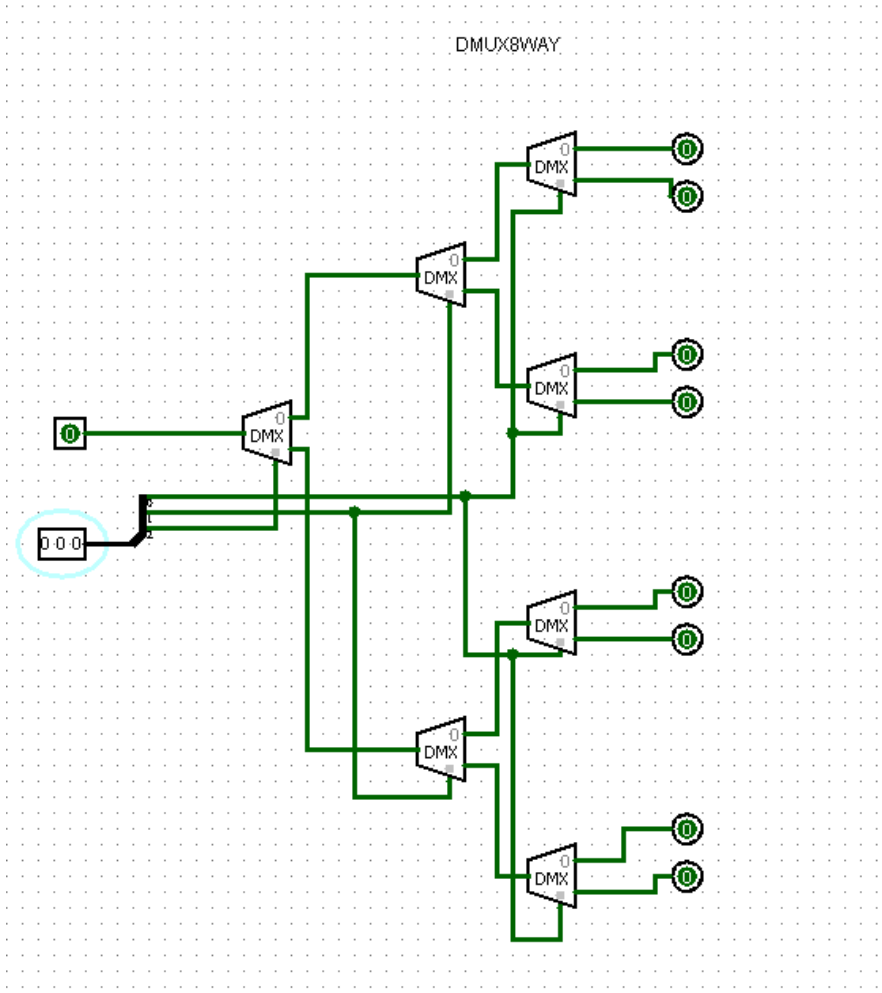
UNIVERSIDAD EAFIT

CHIP DMUX8WAY

• DMUX8WAY

in	sel[2]	sel[1]	sel[0]	salida → =1 (resultado 0)
1	0	0	0	a
1	0	0	1	b
1	0	1	0	c
1	0	1	1	d
1	1	0	0	e
1	1	0	1	f
1	1	1	0	g
1	1	1	1	h

```
CHIP DMux8Way {

    IN in, sel[3];

    OUT a, b, c, d, e, f, g, h;


    PARTS:

    DMux(in=in , sel=sel[2] , a=grupoABCD , b=grupoEFGH );

    DMux(in=grupoABCD , sel=sel[1] , a= grupoAB, b=grupoCD );

    DMux(in=grupoEFGH , sel=sel[1] , a= grupoEF, b=grupoGH );

    DMux(in=grupoAB , sel=sel[0] , a= a, b=b );

    DMux(in=grupoCD , sel=sel[0] , a= c, b=d );

    DMux(in=grupoEF , sel=sel[0] , a= e, b=f );

    DMux(in=grupoGH , sel=sel[0] , a= g, b=h );
```


DMUX8WAY
DMX
DMX
DMX
DMX
DMX
DMX
DMX
0 0 0