Project 1

NAND

x	y	f
0	0	1
0	1	1
1	0	1
1	1	0

→ Not(x) & Not(y)
→ Not(x) & y
→ x & Not(y)

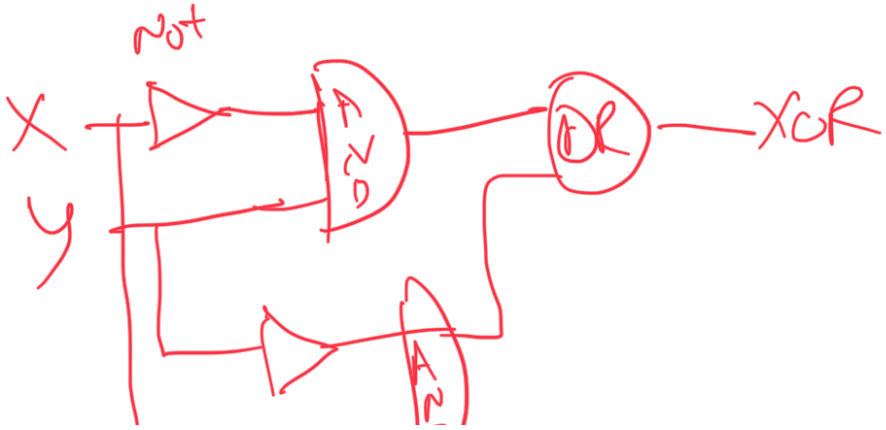
(Not(x) & Not(y)) (Not(x) & y) Or (x & Not(y))

Not(x & y)

XOR

x	y	f
0	0	0
0	1	1
1	0	1
1	1	0

Not(x) & y
Not(y) & x



AND

x	y	AND
0	0	0
0	1	0
1	0	0
1	1	1

NAND

x	y	NAND(x, y)
0	0	1
0	1	1
1	0	1
1	1	0

NAND (NAND(x, y), x)

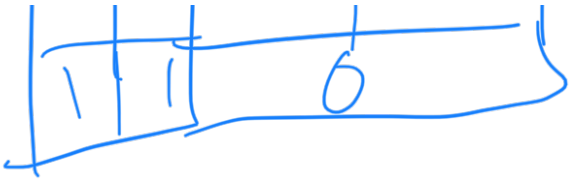
y	x	NAND(x, y)	f	f(Nand(x, y), y)
0	0	1	0	0
0	1	1	1	1
1	0	1	0	0
0	0	1	0	0
0	1	0		

Not Gate

x	Not(x)
0	1

NAND

x	x	Nand(x, x)
0	0	1



XOR GATE

X	y	f
0	0	0
0	1	1
1	0	1
1	1	0

And(Not(x), y) OR Not (y) And x

OR

X	Y	OR
0	0	0
0	1	1
1	0	1
1	1	1

And(Not (x), y)
OR
And(Not(y), x)
OR
(x)&(y)

NAND

X	Y	NAND
0	0	1
0	1	1
1	0	1
1	1	0

And = Nand (Nand (x,y) , Nand (x,y))

X	y	!X	!Y	Nand
0	0	1	1	0
0	1	1	0	1
1	0	0	1	1
1	1	0	0	1

MUX (Multiplexor)

3 Inputs → a, b, Sel

Sel	out
0	a
1	b

a	b	Sel	Mux
0	0	0	0
1	0	0	1
0	1	0	0
1	1	0	1
0	0	1	0
1	0	1	0
0	1	1	1
1	1	1	1

a & Not(b)

a & B

Not(a) & B

a & B

a & Not(b) & Not(c)

+

a & B & Not C

+

Not A & D & C

A & D & C

a	Sel	Not Sel
0	0	1
0	1	0
1	0	1
1	1	0

And (a, NotSel)

0 ✓
0 ✓
1 ✓
0 ✓

We want 0's index's ↑

2

& Find pair (b, Not Sel)

+

b	Sel	!Sel
0	0	1
0	1	0
1	0	1
1	1	0

a	b	Sel	A & Not Sel	B & Not Sel	Desired
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	1	0
0	1	1	0	0	1
1	0	0	1	0	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	0	1

x	y	D
0	0	0
0	0	0
0	1	0
0	0	1
1	0	1
0	0	0
1	1	1
0	0	1

Arranged

0 0 -> 1
0 1 -> 0
1 0 -> 1
1 1 -> 1

DMUX

Input - Sel, in

Output - a, b

if sel = 0
output = in, 0
else
out = 0, in

In Mux
if sel = 0
  out = a
else
  out = b

XGate

1) Mux(a= in, b=0, Sel, out=a)

MUX					Desired	
in	Sel	a	b	MO	a	b
0	0	0	1	0	0	0
0	1	0	0	0	0	0
1	0	1	1	1	1	0
1	1	1	0	0	0	1

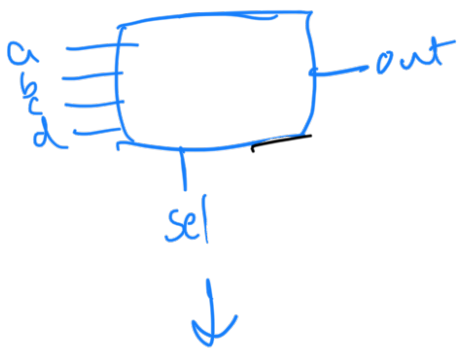
Or16, And16, Mux16 are Easy
Same thing again & again

MUX4WAY16

Implement Mux 4 Way with just 1 bit

lets first ...

in: a, b, c, d, sel, sel2

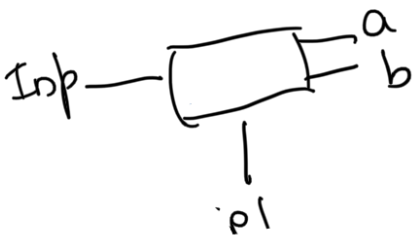


sel	
a	0 0
b	0 1
c	1 0
d	1 1

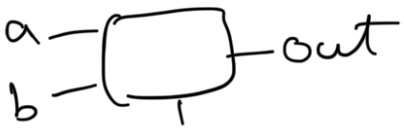
x	y
1	0
1	0
0	1
0	1

let a, b be x
let c, d be y

DMUX



MUX



sel

(1) first Complete for a, b which be done using
Mux16(a=a, b=b, sel=sel[0])
Store it in temp 1

b[0]
a, b are only chosen when sel is 01 or 00
So we only care abt sel[0]

(2) Now find for c, d.
this can be done by
Mux16(a=c, b=d, sel=sel[0])
Store it in temp (2)

b[0], c & d are only called when sel is 10 or 11

Remember that in reality there are just 2 bits for sel

1	0
x	x

and exists 4 possible states

0 0
0 1
1 0
1 1

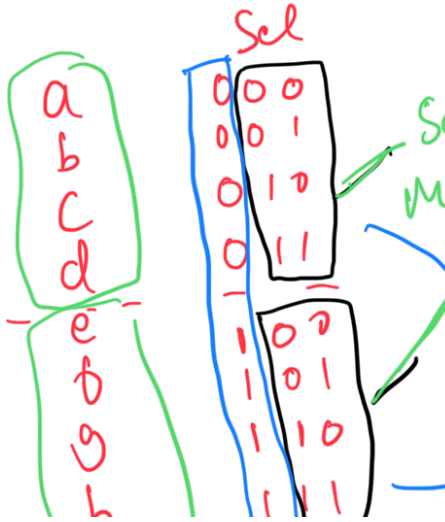
So, lets say we get sel = 10,

over temp 1 will choose [a]
but our temp 2 should choose [c]
On the other hand if sel = 11, temp 1 = b
& temp 2 should be d. Did You Get it!!!
we just need to use the Mux for c & d with sel[0]!

Now we need to choose b/w temp 1 & temp 2. if you look closely temp 1 is always chosen when Sel = 00 or 01. Basically when Sel[1] = 0 temp 1 is chosen & when Sel 1 is 1 we need temp 2!
So just use Mux16(a=temp1, b=temp2, sel=sel[1], out=out

Voila!

MUX 8 Way 16!!!



This is actually much easier if u understand Mux 4 Way 16

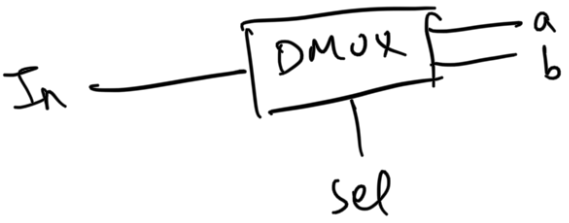
LOOK AT MUX4WAY16 for better

Understanding

Now to choose b/w temp1 & temp2

Use this

DMux 4 Way



if sel = 0 -> in, 0
else -> 0, in

2 Inputs -> in, sel (2)
4 outputs -> a, b, c, d

in	0	0	0	sel = 00
0	in	0	0	01
0	0	in	0	10
0	0	0	in	11

Similar to the way we solved Mux 4 Way (16), Here also we will first solve for a & b

So -> DMux (in = in, sel = sel[0], a = tempa, b = tempb)

this will give us in, 0 or 0, in

tempb [0] we are not sure with the vals yet, as they can just be 0s

Now, as a & b can just be 0s So to check that we need to look at sel[1]. We find that whenever sel[1] is 0, a & b can have "in" val. at sel = 00, 01. But whenever sel[1] is 1, a & b are guranteed to be 0. at sel 10, 11

we need to see for sel

So Now we need to see for [illegible]

DMux(in=tempa, sel=sel[1], a=a, b=c)

this bcoz if sel[1] is 0, a will be tempa and that would mean it will have in value so naturally c will be 0!

Same this is true for b & D

DMux(in=tempb, sel=sel[1], a=b, b=d)

DMux 8 Way

a
b
c
d
e
f
g
h

in 0 0 0 0 0 0 0 0
0 in 0 0 0 0 0 0 0
0 0 in 0 0 0 0 0 0
.
.
.
0 0 0 0 0 0 in 0 0
0 0 0 0 0 0 0 in 0
0 0 0 0 0 0 0 0 in

sel = 0 0 0
0 0 1
0 1 0
0 1 1
1 0 0
1 0 1
1 1 0
1 1 1

So I tried doing the same thing as the DMux 4 Way but that did not work, so I did this =>

just like we divided DMux4Way into 2 parts A&C here I divided it into A&E. group A i.e a,b,c,d only have a chance of having "in" if Sel[2] = 0 .else E group i.e. efgh will have "in"!

So -

DMux(in=in, sel=sel[2], a=tempa, b=tempe)

Bcoz we are not sure if exactly "a" or "e" one is or their group member

if Leb say tempa is "in" that means s[2]=0 & one out of a, b, c, d has e. We don't know which one.

So for that.

DMux4Way(in=tempa, sel=sel[0..1], a=a, b=b, c=c, d=d);

if tempa is 0, which means tempe is 1, then all of abcd will be 0, which is what we want

Similarly -

DMux4Way(in=tempe, sel=sel(0..1), a=e, b=f, c=g, d=h)

Voila!