Nand to Tetris

2025年1月28日(火)

Project One

~~Part~~
~~Logic Gate 1 = Nand~~
~~The start~~

Logic Gate 1 — Nand

- This gate shall be treated as a primitive gate, and so its implementation shall be assumed.

Logic Gate 2 — Not

- $Nand(x,y) = \overline{x \cdot y}$
- Looking at the Nand(x,y) truth table:

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

Another solution is $Not(x) = Nand(x,x)$

By setting the first input to "1", and only using the second input, the output becomes the Not of the second output.

- Therefore, $\boxed{Not(x) = Nand(1,x)}$ or $\boxed{Not(x) = Nand(x,x)}$

Logic Gate 3 - And

- $And(x,y) = x \cdot y$
- $Nand(x,y) = \overline{x \cdot y}$, and $Not(x) = \overline{x}$
- Therefore, $\boxed{And(x,y) = Not(Nand(x,y)) = \overline{(\overline{x \cdot y})} = x \cdot y}$

Logic Gate 4 - Or

- Truth table for And:

x	y	And(x,y)
0	0	0
0	1	0
1	0	0
1	1	1

- Try $Not(And(Not(x), y))$:

x	y	$\overline{(\overline{x} \cdot y)}$
0	0	0
0	1	0
1	0	1
1	1	1

Doesn't work

- Try $Not(And(Not(x), Not(y)))$:

x	y	$\overline{(\overline{x} \cdot \overline{y})}$
0	0	0
0	1	1
1	0	1
1	1	1

Same as $Or(x,y)$

(2)

- Therefore, $Or(x,y) = Not(And(Not(x), Not(y)))$

Logic Gate 5 – Xor

- Try truth table of $Xor(x,y)$:

x	y	Xor(x,y)
0	0	0
0	1	1
1	0	1
1	1	0

Truth table is the same as Or(x,y) besides this row

- Assuming same function of both inputs:
  - $Xor(x,y) = Or(f(x), f(y))$ ✗
  - This cannot be since $f(x) = Not(x)$ would only inverse the truth table
  - $Xor(x,y) = Or(f(x,y), g(x,y))$
  - f and g must be different because the same output is yielded with both asymmetrical inputs.
  - If $x=1$ when $y=0$, out = 1, if $y=1$ when $x=0$ then out = 1
  - Try Truth table of $And(x, Not(y))$ and $And(Not(x), y)$:

x	y	$x \cdot \bar{y}$
0	0	0
0	1	0
1	0	1
1	1	0

x	y	$\bar{x} \cdot y$
0	0	0
0	1	1
1	0	0
1	1	0

} Superimpose outputs with Or to preserve ones

- Therefore, $Xor(x,y) = Or(And(x, Not(y)), And(Not(x), y))$

Logic Gate 6 – Multiplexer (Mux)

- 3 inputs – a, b, sel
- Truth table:

a	b	sel	out(Mux(a,b,sel))	And(sel, And(b, sel))	And(Not(sel), Or(a, sel))
0	0	0	0	0	0
0	1	0	0	0	0
1	0	0	1	0	1
1	1	0	0	0	1
0	0	1	0	0	0
0	1	1	1	1	0
1	0	1	0	0	0
1	1	1	1	1	0

✓ ✓

(3)

- And(sel, And(b,sel)) and And(Nob(sel), Or(a,sel)) can be superimposed with Or(x,y) to yield Mux(a,b,sel)
- Therefore, **Mux(a,b,sel) = Or(And(Nob(sel), Or(a,sel)), And(sel, And(b,sel)))**

Logic Gate 7 – Demultiplexer (DMux)

- Truth table:

DMux(in, sel) → a, b

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

- Seperate sets of gates for each output a and b:
  - a:
    - **And(Nob(sel), or(in,sel))**
  - b:
    - **And(sel, And(In, sel))**

Logic Gate 8 – Multi-bit Not (16-bit)

- Using the same boolean logic for the single-bit gate, the multi-bit gate only needs to be implemented on arrays of bits.

Logic Gate 9 – Multi-bit And (16-bit)

- Same ~~as~~ Method as Logic Gate 8.

Logic Gate 10 – Multi-bit Or (16-bit)

- Same method as Logic Gate 8.

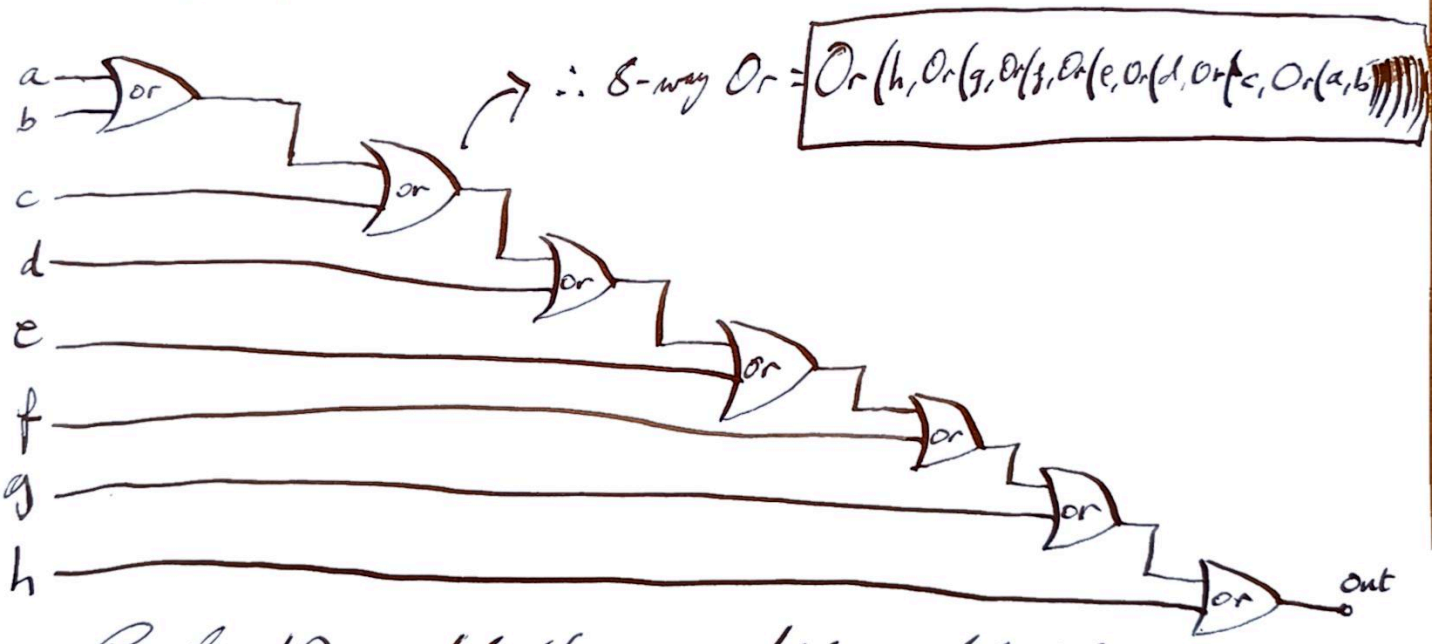
(4)

Logic Gate 11 – Multi-bit Multiplexer

- Same method as Logic Gate 8, using the single-bit multiplexer implemented on arrays of bits for input.

Logic Gate 12 – ~~Multi~~ Multi-way Or

- Specifically, only 8-way Or gates are needed.
- Looks would look like this:



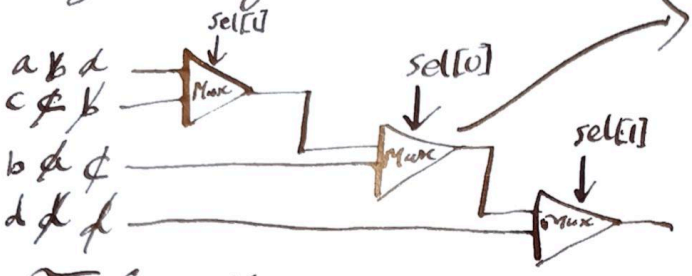
Logic Gate 13 – Multi-way/Multi-bit Multiplexer (4-way)

- Only 4-way and 8-way 16-bit multiplexers are needed
- Truth table for 4-way 16-bit multiplexer:

sel[1]	sel[0]	out
0	0	a
0	1	b
1	0	c
1	1	d

5

- Try these forks!

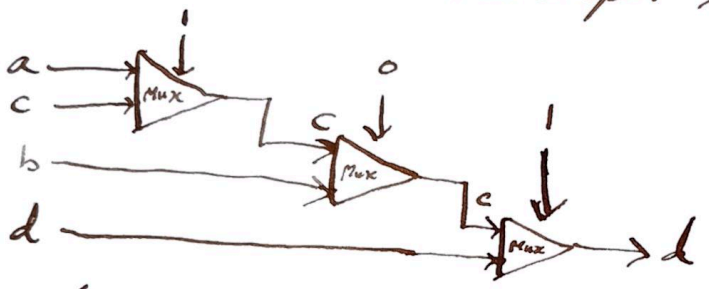


→ "Mux" gates are 16-bit multi-bit gates.

- Test with sel = 10 :

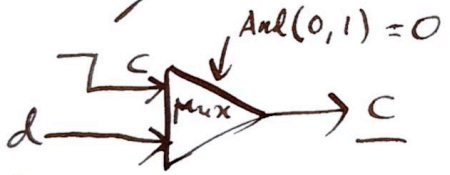
sel[1] sel[0]

- Sel = 10 should mean the output is c



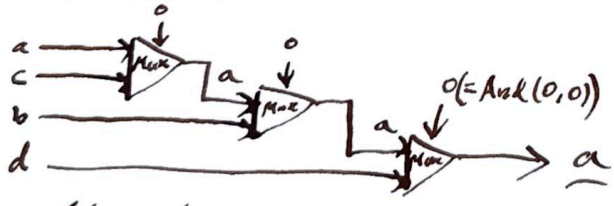
∴ Doesn't work!

- Try with sel = 10, and last Mux gate selection being And(sel[0], sel[1])
  - Last gate:



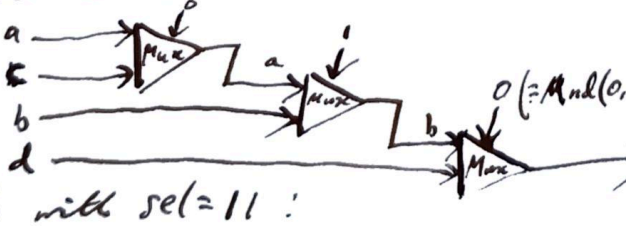
∴ Works

- Try with sel = 00 :



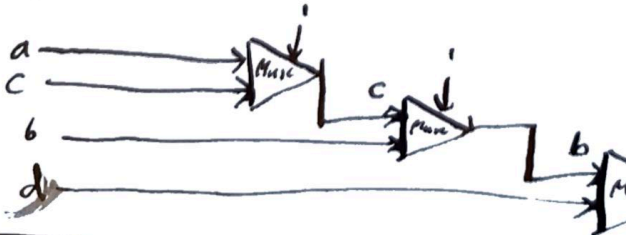
∴ Works

- Try with sel = 01 :



∴ Works

- Try with sel = 11 :



∴ Works

∴ Mux(____ Mux(Mux(a, c, sel[1]), b, sel[0]), d, And(sel[0], sel[1]))

16-bit

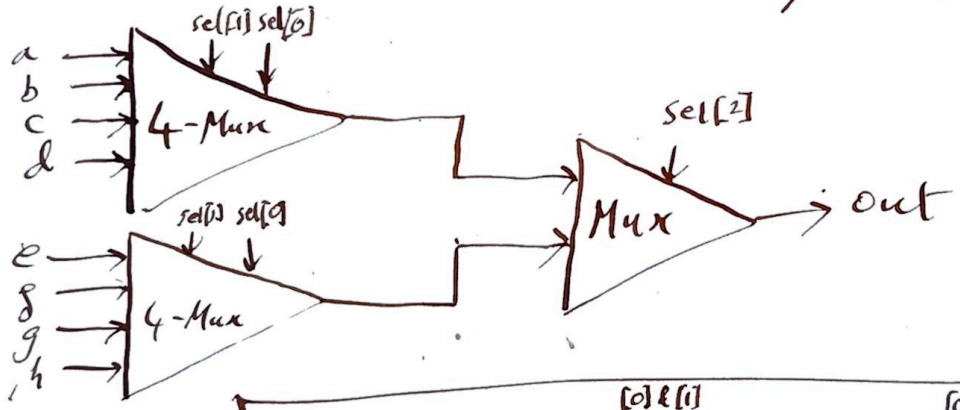
6

Logic Gate 14 - Multi-way/Multi-bit Multiplexer (8-way)

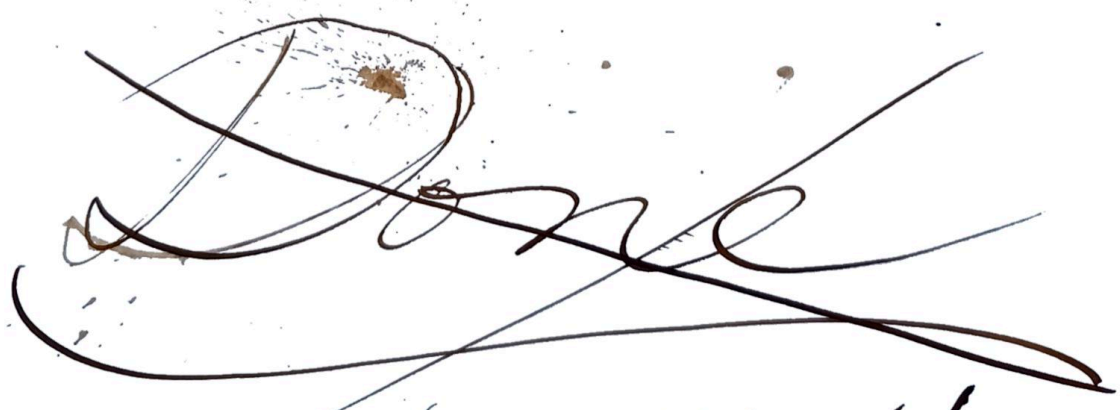
- Truth table for 8-way 16-bit multiplexer:

sel[2]	sel[1]	sel[0]	out	
0	0	0	a	if sel[2]=0, 4-way Mux on a,b,c,d
0	0	1	b	
0	1	0	c	
0	1	1	d	
1	0	0	e	if sel[2]=1, 4-way Mux on e,f,g,h
1	0	1	f	
1	1	0	g	
1	1	1	h	

- Can a 4-way 16-bit multiplexer be used? Yes
- Must be passed through 16-bit Multiplexer.



- Therefore, Mux(Mux4(a,b,c,d, sel[0]&[1]), Mux4(e,f,g,h, sel[0]&[1]), sel[2]) — 16-bit



↳ Not yet!

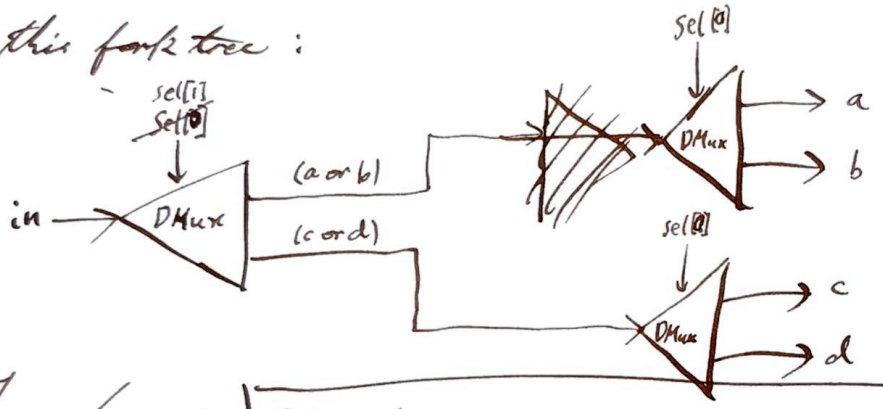
(7)

Logic Gate 15 - 4-way 1-bit Demultiplexer

- Truth table for 4-way, 1-bit demultiplexer:

sel[1]	sel[0]	a	b	c	d	(for any 1-bit input "in")
0	0	in	0	0	0	} if sel[1]=0 and sel[0]=0, or 1 input is routed to a or b
0	1	0	in	0	0	
1	0	0	0	in	0	} if sel[1]=1 and sel[0]=0 or 1, input is routed to c or d
1	1	0	0	0	in	

- Try this fork tree:



- Therefore: DMux(in, sel[1]) → DMux(in, sel[0]), DMux(in, sel[0])

Logic Gate 16 - 8-way 1-bit Demultiplexer

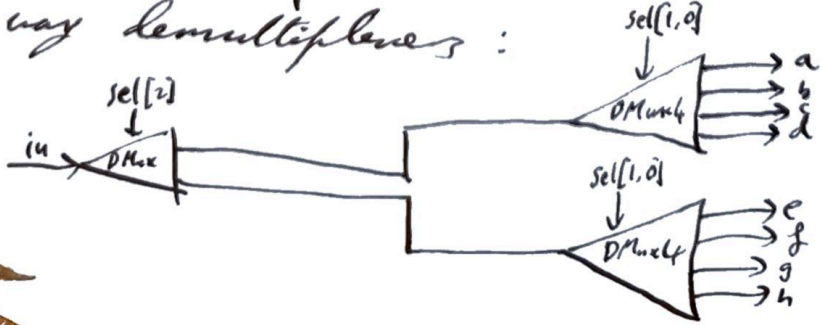
- Truth table:

sel[2]	sel[1]	sel[0]	a	b	c	d	e	f	g	h
0	0	0	in	0	0	0	0	0	0	0
0	0	1	0	in	0	0	0	0	0	0
0	1	0	0	0	in	0	0	0	0	0
0	1	1	0	0	0	in	0	0	0	0
1	0	0	0	0	0	0	in	0	0	0
1	0	1	0	0	0	0	0	in	0	0
1	1	0	0	0	0	0	0	0	in	0
1	1	1	0	0	0	0	0	0	0	in

- Therefore:

DMux(in, sel[2]) → DMux4(in, sel[1,0]), DMux4(in, sel[1,0])

- Using 4-way demultiplexers:



Done