

コンピュータシステムの 理論と実装

2025/2/23

1章 ハードウェア実装

最初 NAND ゲートのみ しか与えられる。

NAND ゲートから各ゲートを実装していく。

NAND (No + And)

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

NOT

a	out
0	1
1	0
0	1
1	0

OR

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

AND

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

✓ Xor (And, Or, Not)

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

a, b が異なる時にビットが立つ
a, b が同じ時にビットは立たない。

Xor.hdl ファイルの if 文を見て解いてしまったので、
表から if 文作って hdl 文作成しようか
勉強にしよう

Mux (マルチプレクサ)

sel	out
0	a
1	b

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

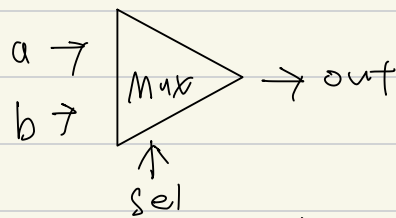
a Or sel	= out	(a Or sel)	or	(b And sel)
0	0	0	0	0
0	0	0	1	0
1	0	0	0	0
1	0	1	0	0

Max

$(a \text{ Or } sel) \text{ Or } (\text{Not } b \text{ And } sel)$

0	0	1	1	0	0	1
0	0	1	0	0	0	1
1	0	1	1	0	0	1
1	0	1	0	0	0	1
			Or			
	0	1			0	1
	0	1			0	0
	1	1			0	1
	1	1			0	0

	Or
0	1
0	1
1	1
1	1

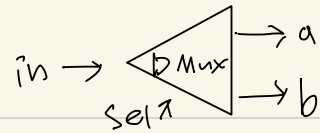


sel	out
0	a
1	b

	Xor
0	0
0	1
1	0
1	1

$(a \text{ Or } sel) \text{ Xor } (\text{Not } b \text{ And } sel)$

D Mux (71 2114 70677)



sel	a	b
0	in	0
1	0	in

if (sel == 0)
out = a, b

{a, b} = {in, 0}

else

{a, b} = {0, in}

in	sel	a	Xor	And	or	NAnd
0	0	0	0	0	0	1
1	0	1	1	0	1	1
0	1	0	1	0	1	1
1	1	0	0	1	1	0

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

outb = And(a = in, b = sel, out = b)

outa = Xor(a = in, b = outb, out = a)

ifDL:

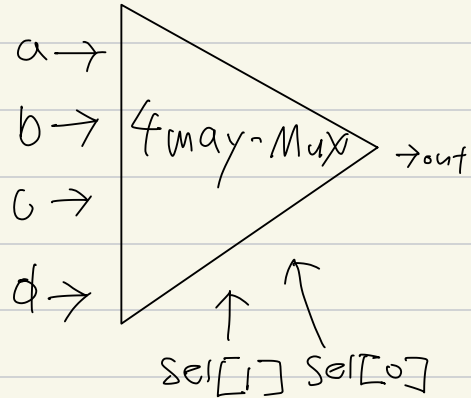
And(a = in, b = sel, out = outb, out = b);

Xor(a = in, b = outb, out = a);

P. 388

Mux 4Way lb

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



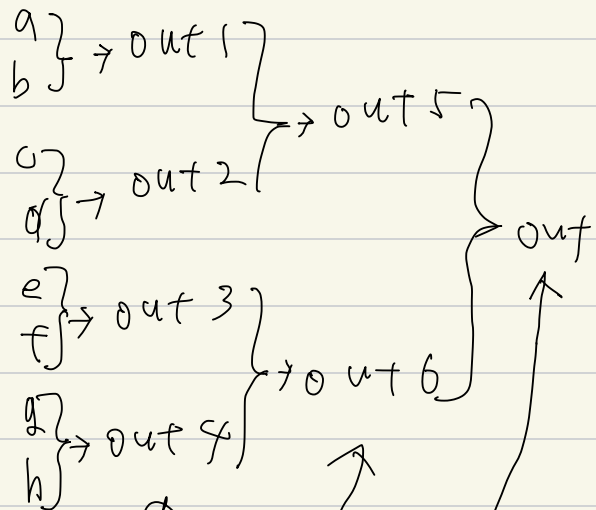
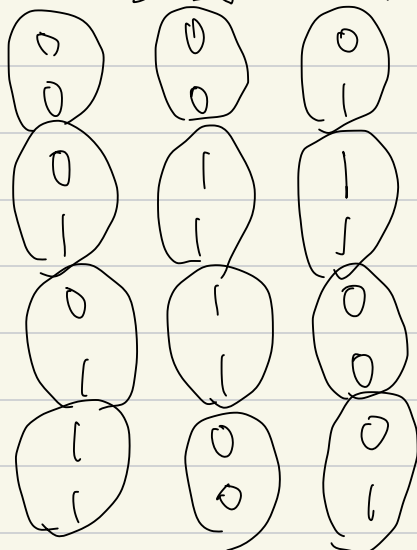
a	b	c	d	sel[1]	sel[0]	out
1	0	0	0	0	0	1
1	1	0	0	0	1	1
1	1	1	0	1	0	1
1	1	1	1	1	1	1

sel[1]が0のは a, b を出力, sel[1]が1のは c, d を出力

$\text{Mux lb}(a=a, b=b, \text{sel}=\text{sel}[0], \text{out}=\text{out}_0); \leftarrow \text{sel}[1]=0$
 $\text{Mux lb}(a=c, b=d, \text{sel}=\text{sel}[0], \text{out}=\text{out}_1); \leftarrow \text{sel}[1]=1$
 $\text{Mux lb}(a=\text{out}_0, b=\text{out}_1, \text{sel}=\text{sel}[1], \text{out}=\text{out});$

Mux 8 Way 1b

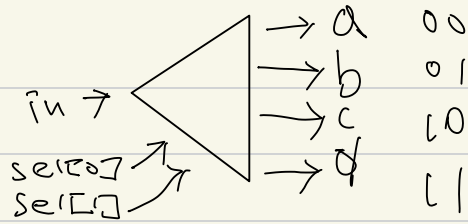
sel0 sel1 sel2



Mux 4 Way 1b

Mux 1b

DMax 4Way



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

sel[0] が 0 のときは c, d を 0 に置き、1 のときは a-b を出す。

DMax (in=in, sel=sel[0], a=out a, b=out b);

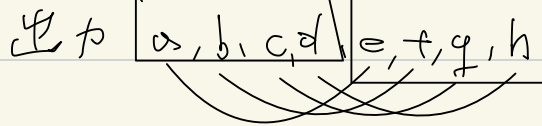
~~DMax (in=in, sel=sel[0], a=out c, b=out d);~~

DMax (in=out a, sel=sel[1], a=a, b=c);

DMax (in=out b, sel=sel[1], a=b, b=d);

DMux 8Way

入力 in, sel[3]



- sel[2] が 0 のときは e, f, g, h が出力される
- sel[2] が 1 のときは a, b, c, d が出力される。

DMux 4Way (in=in, sel=sel[0..1], a=a, b=b, c=c, d=d);

DMux 4Way (in=in, sel=sel[0..1], a=e, b=f, c=g, d=h);

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

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

DMux (in=c, sel=sel[2], a=c, b=g);

DMux (in=d, sel=sel[2], a=d, b=h);

2025/2/27 2章 7-10 算術

半加算器 (Half Adder)

入力 a, b

a	b	carry	sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

出力 Sum, Carry
 $a+b$ の
最下位ビット
最上位ビット

Xor ($a=a, b=b, \text{out}=\text{sum}$);

AND ($a=a, b=b, \text{out}=\text{carry}$);

全加器 (Full Adder)

输入 a, b, c

输出 $carry, sum$

a	b	$carry$	sum	$c + sum$	$c\ carry$	$csum$
0	0	0	0	0	0	0
0	1	0	1	0	1	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0
0	0	0	0	1	0	1
0	1	0	1	1	1	0
1	0	0	1	1	1	0
1	1	1	0	1	0	1

$abcarry \text{ or } ccarry = carry$

$csum = sum$

Half Adder ($a = a, b = b, carry = abcarry, sum = absum$);
 Half Adder ($a = c, b = absum, carry = ccarry, sum = sum$);
 Or ($a = abcarry, b = ccarry, out = carry$);

16ビット加算器 (Add16)

2ビットの場合

10 a

01 + b

11 out

11

11 a

11 + b

110 out

入力 a[16], b[16]

出力 out[16]

Overflow (入力ビット幅を超え) は無視

Half Adder (a = a[0], b = b[0], carry = 0, sum = out[0])

Full Adder (a = a[i], b = b[i], c = 0, carry = 0, sum = out[i])

i: ビット繰り返し

a[0] = 1, b[0] = 1, c = 0, out[0] = 0 i = 0

a[1] = 1, b[1] = 1, c = 1, out[1] = 1 i = 1

16ビットインクリメント (Inc16)

入力 In[6]

出力 Out[6]

関数 out = in + 1

・インクリメントする子のための1が必要 → +true p374

・最下位ビット+1はXorで求めらる → Sum = out[0]

・繰り上げが1発生するかはAndで求めらる → Carry

Xor(a = true, b = in[0], out = out[0]);
And(a = inc, b = in[0], out = carry0);
Xor(a = carry0, b = in[1], out = out[1]);
And(a = carry0, b = in[1], out = carry1);

この方法で実装すると

$(Xor \times 16) + (And \times 15)$

の構成となる

Add16.hdlを利用

するよりも効率が良い

carry1 = carry0 And in[1]

carry0 = inc And in[0]

out[0] = inc Xor in[0]

+1 or inc

out[1] = carry0 Xor in[1]

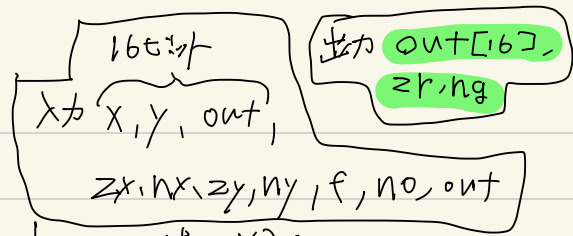
加算器チップ (Add16.hdl) は

(HalfAdder x1) + (FullAdder x15) で構成される。

$(Xor \times 1) + (And \times 1) + (HalfAdder \times 2) + (Or \times 1)$ $(Xor \times 3) + (And \times 3) + Or$
 $(Xor + And) \times 2$

Add16チップを利用すると $(Xor \times 3) + (And \times 3) + Or$ を利用する。

算術論理演算器 (ALU)



$zx=1$ の時の $x=0$ $And16(a=false, b=x, out=x);$

$nx=1$ の時の $x=!x$ $Not16(in=x, out=x);$

$f=1$ の時の $out=x+y$ $Add16(a=x, b=y, out=out);$

$f=0$ の時の $out=x \& y$ $And16(a=x, b=y, out=out);$

$no=1$ の時の $out=!out$ $Not16(in=out, out=out);$

分岐出力を Mux の a, b に x と y , sel に 条件 (zx, nx, f, no)
 を x と y に 出力すれば 1 条件分岐により 値の出力が できる

$Not16(in=0zx, out=not x);$

$Add16(a=x, b=y, out=x+add y);$

$And16(a=x, b=y, out=x \text{ and } y);$

$Mux16(a=x, b=false, sel=zx, out=0zx);$
 $Mux16(a=0zx, b=not x, sel=nx, out=0nx);$ } y も同じ

$Mux16(a=x \text{ and } y, b=x+add y, sel=f, out=of);$

$Not16(in=of, out=not of);$

$Mux16(a=of, b=not of, sel=no, out=out, out[0..7]=0low8, out[8..15]=0nohigh8, out[16]=ng);$

$Or8Way(in=0nohigh8, out=0 \text{ or } 8ways);$

$Or(a=0 \text{ or } 8ways1, b=0 \text{ or } 8ways2, out=0 \text{ or });$

$Not(in=0 \text{ or } , out=zr);$

2025/3/6 ~ Xメモリ

1ビットレジスタ

ビット(Bit)の実装

フリップフロップ (DFF) は自身で
(実装せず) にビットインを利用する。
入力 in
出力 out

入力 in, load

出力 out

if load == 1 then out(t+1) = in(t)

else out(t+1) = out(t)

Mux (a=odff, b=in, sel=load, out=outmux);

DFF (in=outmux, out=out, out=odff);

16ビットレジスタ (Register) 入力 in[16], load

出力 out[16]

1ビットレジスタを16個内包している。

Bit (in=in[0], load=load, out=out[0]);

⋮

Bit (in=in[15], load=load, out=out[15]);

$x[k]$ in $[16]$, load, address $[k]$ ($k = \log_2 n$)

RAM 8

出力 out $[16]$

$\begin{matrix} 6 \\ 3 \end{matrix}$

RAMに内蔵された16ビットレジスタの数

addressを指定したレジスタにloadを行います。Dmux 8Way

Dmux 8Way (in = load, sel = address, a = 01a, b = 01b, c = 01c, d = 01d,
e = 01e, f = 01f, g = 01g, h = 01h);

register (in = in, load = 01a, out = org0);

∴ ∴ , load = 01b, out = org1);

⋮

load = 01h, out = org7);

Mux 8Way 16 (a = org0, b = org1, c = org2, d = org3, e = org4,
f = org5, g = org6, h = org7, sel = address, out = out);

λ $\text{in}[16]$, load , $\text{address}[6]$ ($k = \log_2 n$) 64

RAM64 λ $\text{out}[16]$

RAM8 ≤ 8 肉包する

$k[0..2]$ での RAM8 に λ が入るのを 選択する。

$k[3..5]$ での RAM8 内の λ の Register に λ が入るのを 選択する。

$\text{DMux8Way}(\text{in} = \text{load}, \text{sel} = \text{address}[0..2], a = 01a, b = 01b, c = 01c, \\ d = 01d, e = 01e, f = 01f, g = 01g, h = 01h);$

$\text{RAM8}(\text{in} = \text{in}, \text{load} = 01a, \text{address} = \text{address}[3..5], \text{out} = \text{oram0});$

$\text{RAM8}(\quad, \text{load} = 01b, \quad = \quad, \text{out} = \text{oram1});$
 \vdots

$\text{RAM8}(\quad, \text{load} = 01h, \quad \text{out} = \text{oram7});$

$\text{Mux8Wayb}(a = \text{oram0}, b = \text{oram1}, c = \text{oram2}, d = \text{oram3}, e = \text{oram4}, \\ f = \text{oram5}, g = \text{oram6}, h = \text{oram7}, \text{sel} = \text{address}[0..2], \\ \text{out} = \text{out});$

λ 入力 $in[1b]$, $load$, inc , $reset$

カウンタ (PC)

出力 $out[1b]$

if $reset(t)$ then $out(t+1) = 0$

else if $load(t)$ then $out(t+1) = in(t)$

else if $inc(t)$ then $out(t+1) = out(t) + 1$

else

$out(t+1) = out(t)$

$out(t+1) \leftarrow$ 次のように出力させるために使われる

Register($in = 0selreset$, $load = true$, $out = out$);

Inclb($in = incout$, $out = 0inc$);

Register($in = 0selload$, $load = inc$, $out = incout$);

カウンタ

Mux1b($a = 0selload$, $b = false$, $sel = reset$, $out = 0selreset$);

Mux1b($a = 0selinc$, $b = iselload$, $sel = load$, $out = 0selload$);

Mux1b($a = incout$, $b = 0inc$, $sel = inc$, $out = 0selinc$);

Mux1b($a = in$, $b = false$, $sel = reset$, $out = iselload$);