Digital Logic (256 Review)

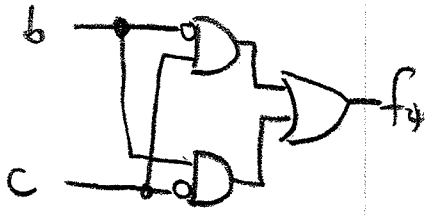
Basic logic gates

a	f
0	1
1	0

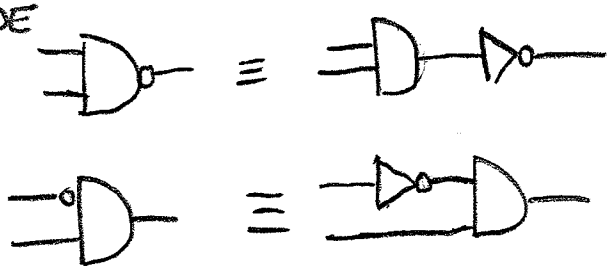
a	b	c	f_1	f_2	f_3
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	0	1	1
1	0	1	0	1	0
1	1	0	0	1	0
1	1	1	1	1	1

f_4

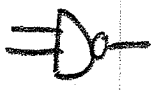
$f_4 = \bar{b}c + b\bar{c}$

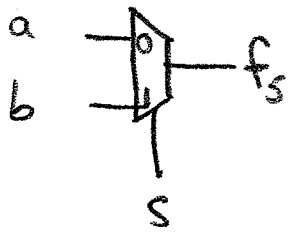


ASIDE



- can also build using just 4x

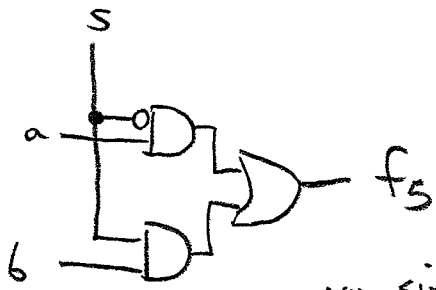
In fact, is a universal logic gate.
You can build <u>any</u> digital system using just 



s	a	b	f_s
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

if $s=1$
$f_s = b$
else
$f_s = a$

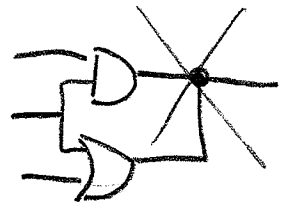
$f_s = \bar{s}a + sb$



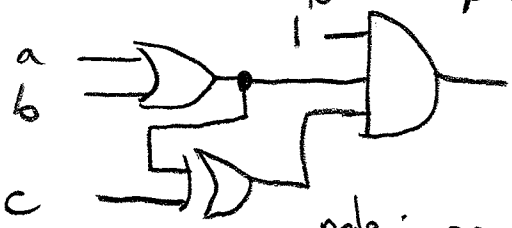
very similar to XOR!

<u>Some Logic Rules</u>: inputs + outputs are <u>directional</u> + have rules

① never connect 2 outputs together



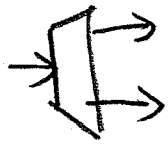
② make sure every input to a logic gate is connected
- to a 0 or 1 (constant)
- to the output of just 1 other gate
- to a "primary input" from outside world



note: an output can connect to multiple inputs

③ never form a logic loop

④ never use a mux backwards



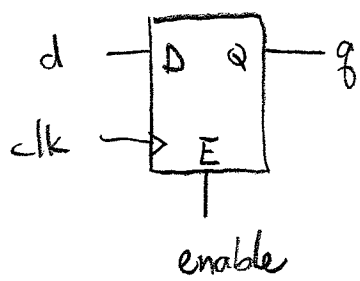
Digital Logic Review (cont'd)

D flip flop - positive edge triggered

clk	d	q_{n+1}
0	x	q_n
1	x	q_n
↑	0	0 } captures d
↑	1	1 } captures d
↓	x	q_n

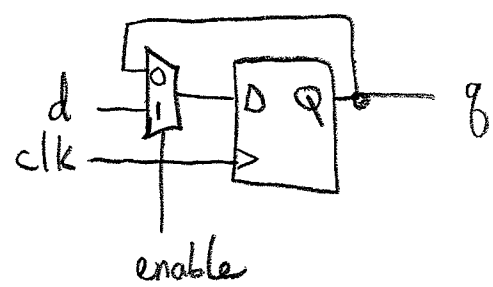
$q_{n+1} = d$ on ↑ clk

DFF + enable input

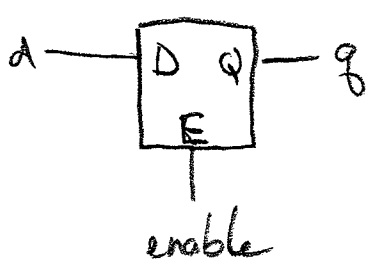


$$q_{n+1} = \begin{cases} d & \text{on } \uparrow \text{clk } \underline{\text{only if}} \ E=1 \\ q_n & \text{otherwise} \end{cases}$$

≡



Level-Sensitive Latch aka Flow-through Latch




```
if enable = 1
    q_{n+1} = d
else
    q_{n+1} = q_n
end if
```

continuously

captures last d while enable = 1

Binary Numbers

humans – powers of 10 : 1, 10, 100, 1000

$2113 = 2113_{10}$

why? 10 fingers! (digits)

computers – powers of 2 : 1, 2, 4, 8

$2113 = 2048 + 64 + 1 = 1000\ 0100\ 0001_2$

$= \%1000\ 0100\ 0001$

why? 2 states: on, off

binary digits ≡ bits

base 2 –	binary	%	0, 1
base 10 –	decimal	(no prefix)$_{10}$	0-9
base 8 –	octal	—	0-7
base 16 –	hexadecimal "hex"	$, 0x	0-9, A-F

computers use binary exclusively

- everything stored as sequence 0s, 1s.
- usually grouped:

nibble	4 bits	(1 hex digit)
byte	8 bits	(2 hex digits)
word/half	16 bits	(4 " ")
longword/word	32 bits	(8 " ")
long longword/long word	64 bits	(16 " ")

Examples

2113_{10}

$= 2 \times 10^3 + 1 \times 10^2 + 1 \times 10^1 + 3 \times 10^0$

note increasing exponent.

Converting Binary to Decimal

nibble 1001_2 = %1001 to decimal:

$$= 1\times 2^3 + 0\times 2^2 + 0\times 2^1 + 1\times 2^0$$
$$= 1\times 8 + 0\times 4 + 0\times 2 + 1\times 1$$
$$= 8 + 0 + 0 + 1$$
$$= 9_{10}$$

convert to decimal

byte %10010111 to decimal:

$$= 1\times 2^7 + 0\times 2^6 + 0\times 2^5 + 1\times 2^4 + 0\times 2^3 + 1\times 2^2 + 1\times 2^1 + 1\times 2^0$$
$$= 1\times 128 + 0\times 64 + 0\times 32 + 1\times 16 + 0\times 8 + 1\times 4 + 1\times 2 + 1\times 1$$
$$= 128 + 16 + 4 + 2 + 1$$
$$= 144 + 7$$
$$= 151$$

Notice, like decimal we can add leading 0s.

2113 = 02113 = 002113

%1001 = %0101 = %001001

In this course, we will often be working with bytes

- although we often drop leading 0s in written form, the computer cannot! it must store all 8 bits.

Other definitions

bit 7 bit 1 bit Ø

%10010111

most significant bit — MSB — bit 7 of a byte, bit 15 of a word

least significant bit — LSB — always bit Ø

The powers of 2

You should memorize them:

0	2^0	=	1	*
1	2^1		2	*
2	2^2		4	*
3	2^3		8	*
4			16	*
5			32	*
6			64	*
7			128	*
8			256	*
9			512	
10			1024	*
11			2048	
12			4096	*
13	⋮		8192	
14			16384	*
15			32768	
16	2^{16}	=	65536	*

* memorize these

also handy:

2^{20}	1 Mega	=	1,048,576
2^{24}	16 Mega	=	16,771,216
2^{30}	1 Giga	=	1,073,741,824

Converting decimal to binary

53_{10} = %110101 how?

first verify: = $1 \times 32 + 1 \times 16 + 1 \times 4 + 1$

= $48 + 5$

Try: 2113_{10} to binary!

the other way:

		odd?		
	53	Y	— 1	LSB
÷2 =	26	N	— 0	
÷2 =	13	Y	— 1	
÷2 =	6	N	— 0	
÷2 =	3	Y	— 1	
÷2 =	1	Y	— 1	MSB
÷2 =	0	N	— 0	

⋮ etc

= %110101 ← LSB