Digital electronics

•Explosive growth over 10-20 years - now dominates applications, still growing...

computing

communications: mobile/fixed phones, radio, data links,...

other: digital audio, consumer goods,...

•Why?

binary logic

almost complete noise immunity

high speed, still increasing

Ethernet: ~Gb/s, phones, links: >Gb/s, computing ~GHz

ease of use

many analogue functions now easier to implement by digital summation, etc

availability

basic logic to complete IC assemblies of big range of complex functions

•Bits, Bytes & Words

byte = 8 bits   word: usually multiple of bytes 8, 16, 32, 64 bits

Basic logic

•bits can be represented in several ways, almost invariably voltage

0/1: Low/High (voltage level) ... or High/Low

values and range depend on families, most common are...

•TTL (bipolar) Transistor-Transistor Logic

usually V_S = 0 to +5V

V_T ~ 1.5V ΔV ~ 1V

outputs & inputs sink/source currents

not identical levels

•CMOS - now most common

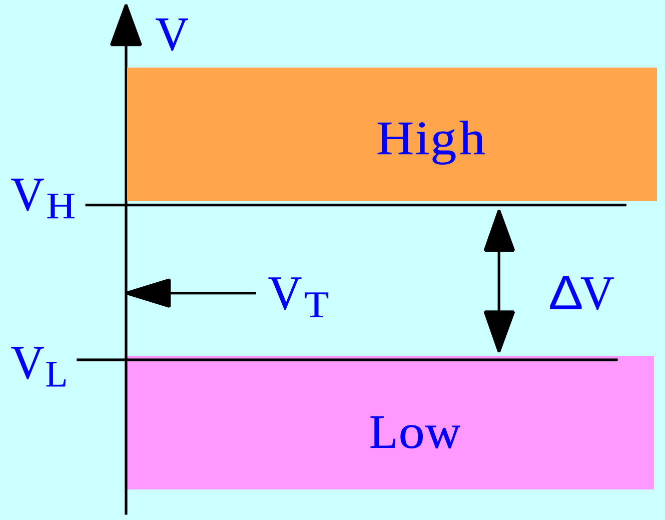
V_S = 0 to +5V but +12V, +3.5V and lower

$V_T \sim V_S/2$ $\Delta V \sim 0.4\ V_S$

outputs swing between supplies

•ECL Emitter Coupled Logic

high speed, but power hungry



designs must tolerate variations

component manufacture

operating temperature

supply voltage

loading

noise

Logic gates

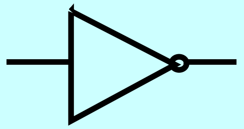
•Logical functions, described by truth table

define output for inputs A & B use High = 1

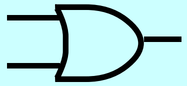
•NOT (Inverter)

$\overline{A}$ or A'

A	out
1	0
0	1



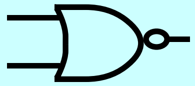
•OR



A+B

A	B	out
1	1	1
1	0	1
0	1	1
0	0	0

NOR

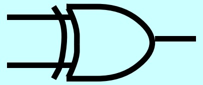


(A+B)'

A	B	out
1	1	0
1	0	0
0	1	0
0	0	1

XOR

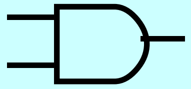
exclusive OR



$A \oplus B$

A	B	out
1	1	0
1	0	1
0	1	1
0	0	0

•AND

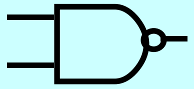


A.B

or AB

A	B	out
1	1	1
1	0	0
0	1	0
0	0	0

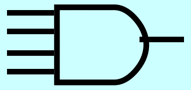
NAND



(AB)'

A	B	out
1	1	0
1	0	1
0	1	1
0	0	1

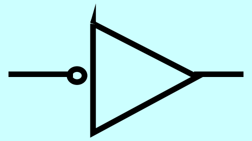
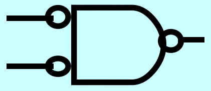
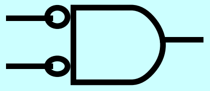
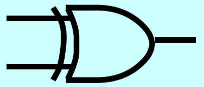
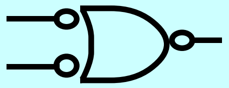
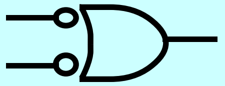
multiple inputs are often allowed, eg



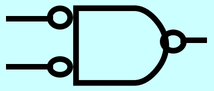
Negative-true logic

- What if - instead of High = 1 - we choose Low = 1?

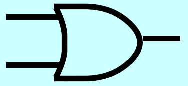
  equivalent functions but different symbols

- NOT 
- OR  NOR  XOR 
- AND  NAND 

it may be easier to think in terms of High & Low, eg...



Negative-true							OR		
A	B	A'	B'	A'.B'	Q		A	B	Q
L	L	H	H	H	L		L	L	L
L	H	H	L	L	H		L	H	H
H	L	L	H	L	H		H	L	H
H	H	L	L	L	H		H	H	H

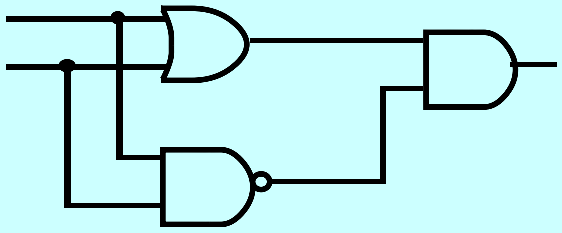


Which gates are needed?

- Logic functions can be constructed from other combinations

eg XOR

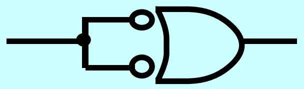
start from OR, change...

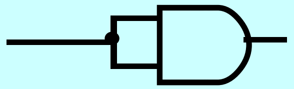


A	B	OR
1	1	1
1	0	1
0	1	1
0	0	0

A	B	XOR
1	1	0
1	0	1
0	1	1
0	0	0

- A couple of examples...





what purpose could they serve?

- Several simple theorems of logic help to do translations if needed (see H&H)

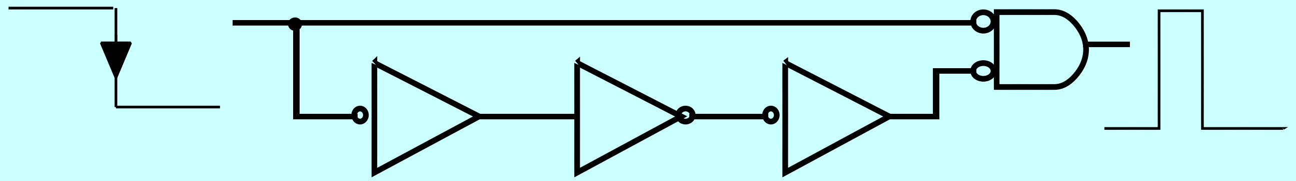
DeMorgan's: Not(A+B) = NotA.NotB

Not(A.B) = NotA + NotB

NotA = A' = $\overline{A}$

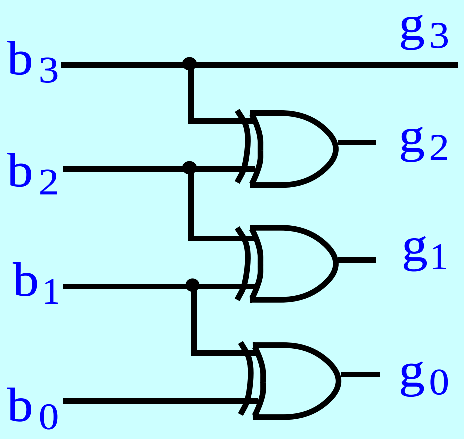
A few applications

•Pulse on falling edge



•Gray coding

binary	Gray
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
0110	0101
0111	0100
...	



in Gray code, only 1 bit changes between states

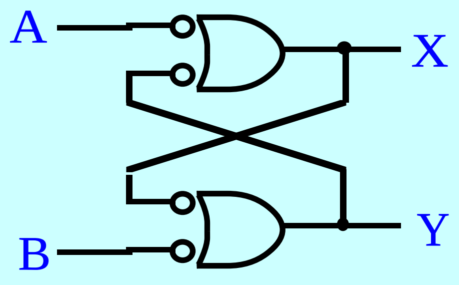
eg, valuable for controlling stepping motors

simplifying logic sequences

Memory - the Flip-flop

- Work out the truth table

A	B	X	Y
L	L	H	H
L	H	H	L
H	L	L	H
H	H	L	H
H	H	H	L



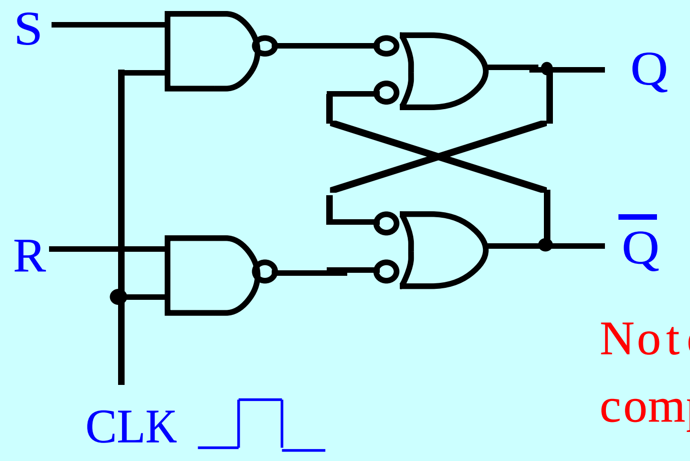
Note the symmetry

so the state depends on the previous history of the flip-flop

- Clocked flip-flop

if CLK = L, state is maintained

S	R	Q_{n+1}
L	L	Q_n
L	H	L
H	L	H
H	H	-



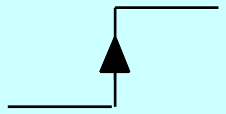
Note the complementary outputs

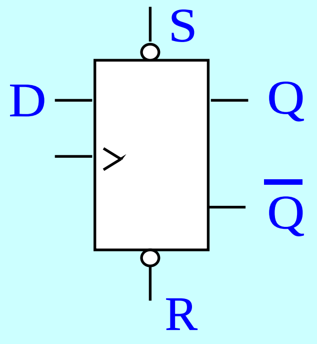
why is HH output undefined?

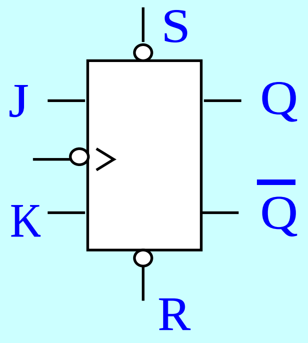
- We now have a memory device with Set & Reset, which runs sequentially

D and JK flip flops

- even the clocked flip-flop is not quite sufficient
  - can't guarantee inputs will not change during clock high
  - the solutions are Master-Slave and edge triggered D flip-flops
- D-type flip-flop
  - transmits value at Data input to Q, on clock edge
    - *both positive and negative edge types available*







J	K	Q_{n+1}
L	L	Q_n
L	H	L
H	L	H
H	H	Q_n'

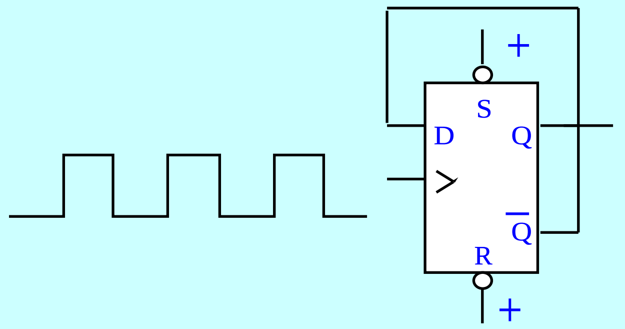
- JK flip-flop - two data inputs
  - both low: unchanged
  - both high: complement of last state
  - complementary data: output follows J input

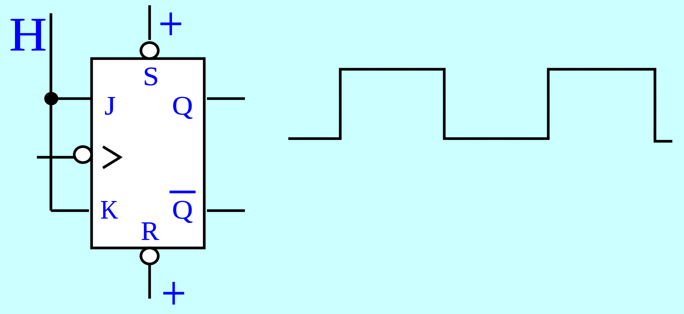
> symbol = edge

o> = negative going edge

Examples

- Divide clock frequency by 2



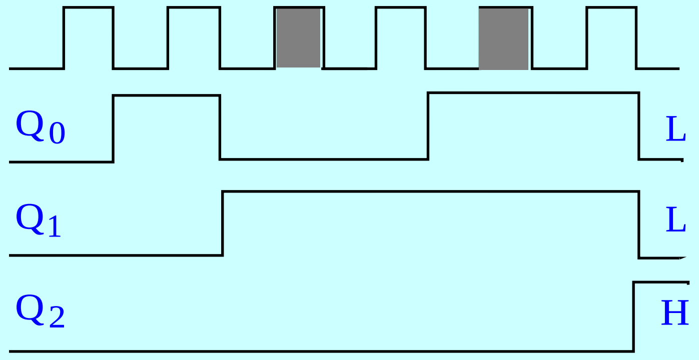


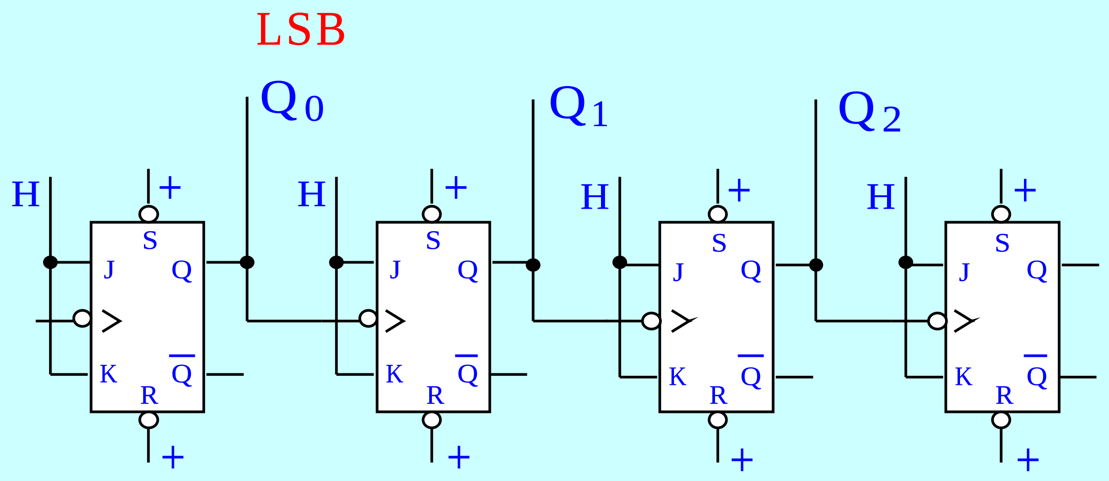
- n-bit Counter

set all Q_i = L

output changes on negative edge

follow outputs





output = HLL = 100_2 = 4

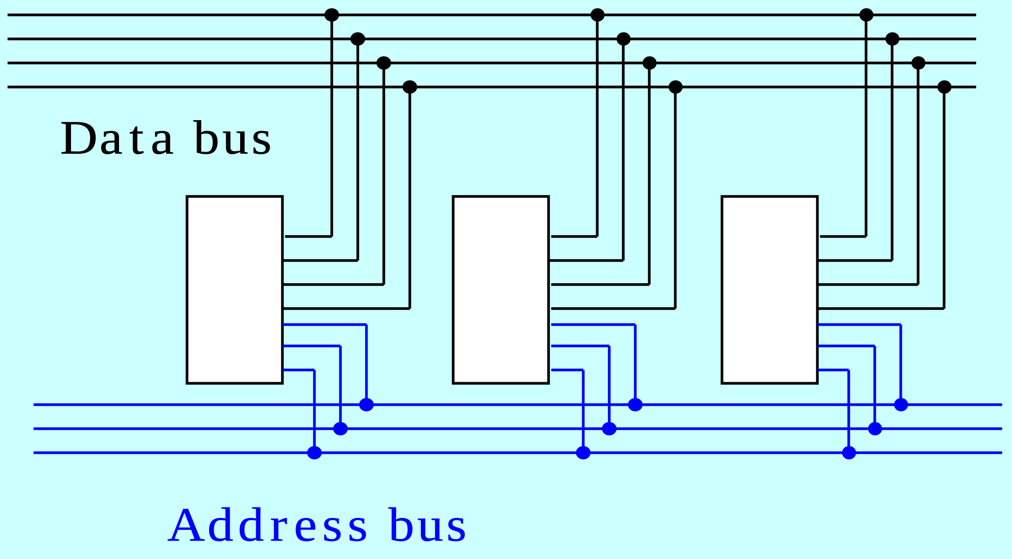
Tri-state logic

- In some applications there is a need to connect to a "bus"
- Bus

    series of parallel lines shared

    between multiple devices

    typically 8-bit, 16-bit, 32-bit,..



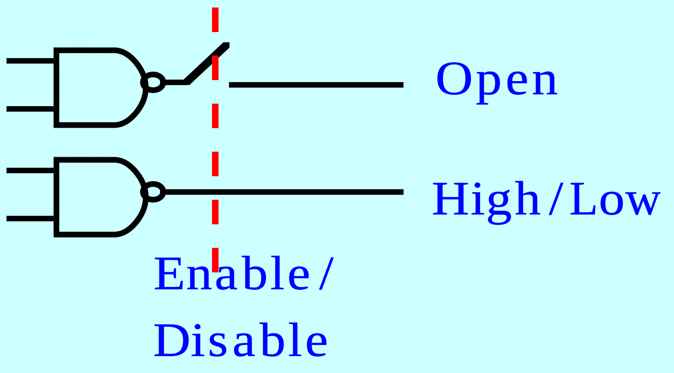
- but if all devices are connected to the bus... who has priority?

    potential recipe for confusion

- Solution, third logic state: "open circuit"

    ie HIGH, LOW and "OPEN"

    controlled by additional input: Enable



Programmable logic

- For very complex logic, it's time consuming and risky to develop circuits
  - programmable logic solves both problems
- PLA programmable logic array
  - transistor array with connections set by fuses to burn
- FPGA field programmable gate array
  - MOS array of uncommitted gates - few k to several M
  - connections made by downloading code which sets biasing of circuits
  - fully re-programmable
- DSP digital signal processor
  - cut-down microprocessor with limited instruction set
- Various levels of complexity and skills to learn
  - eg 2M gate FPGA needs sophisticated design and simulation software