Wednesday EXAM-2, Nov 18
Magnetoelectronics
Optoelectronics
Digital Electronics
Pulsed ICs

Due Wed, Nov 18
Homework Ch. 21
Lab-8a and 8b in **one report**

Review Digital Circuits

- ***Inside Computers***
  Very large-scale integration (VLSI)
  CPU, central processing unit
  > 10^9 transistors (Minecraft computer)
- ***Truth Table***
  Output for each miniterm (=1)
- ***Karnaugh Map*** (*simplification →→→*)
  graphical matrix solution
  combine groups of miniterms (yellow)
  wrap around sides to combine (blue)
  use miniterm more than once (orange)

Truth Table

Row	A	B	C	D	Out
1	1	1	1	1	0
2	1	1	1	0	0
3	1	1	0	1	1
4	1	1	0	0	0
5	1	0	1	1	1
6	1	0	1	0	0
7	1	0	0	1	1
8	1	0	0	0	1
9	0	1	1	1	0
10	0	1	1	0	0
11	0	1	0	1	0
12	0	1	0	0	0
13	0	0	1	1	0
14	0	0	1	0	0
15	0	0	0	1	0
16	0	0	0	0	1

Karnaugh Map Rules

RULE-1: Order top/side table axes,
vary only one bit
when moving to next cell

RULE-2: group **even numbers** of
"1"s that are **adjacent**
(You can wrap around the cylinder,
as in AB=10 → CD=00)

RULE-3: Each group is one miniterm

RULE-4: If input is both "0" and "1"
you don't need that input

K-map Simplification →

Truth Table

AB / CD	00	01	11	10
00	1	0	0	1
01	0	0	1	1
11	0	0	0	1
10	0	0	0	0

5 & 6-variable K-map

Calendar of Topics Covered

Physics PHYS 2371/2372, Electronics for Scientists

Don Heiman and Hari Kumarakuru
Northeastern University, Fall 2020

Also see **Course Description** and **Syllabus**

This is a schedule of the topics covered, but it may be modified occasionally (11/07/2020).

Week #	Lectures	Weekly Topics (Chs.)	Homework (Ch-Problem)	Lab Experiments (always look for latest version)
VII Oct 19, 21-23 **MON/WED**	**MONDAY** **EXAM-I**	**Wed Lecture** Magnetoelectronics Magnetoelectronics Lecture Magnetic induction/flux Transformers (Ch-11)	11-all	**Lab-6, *Build a Magnetometer*** Lab-6 video, Lab-6 data
VIII Oct 28-30	**Wed Lecture** Optoelectronics Optoele Lecture	*Photodiode, LED, laser*	none	**Lab-7, *Optoelectronics*** (coupled LED-photodiode) Lab-7 Optoele video
IX Nov 2, 4-6 **MON/WED**	MON Digital-1 Digital-1 video WED Digital-2 Digital-2 Lecture	**Mon/Wed Lectures** Digital Logic (Ch-19,22), Binary Numbers (Ch-54) Logical Networks (Ch-20)	19-all, 20-all	**Lab-8a, *Digital Circuits*** (truth table, 4-bit decoder) Lab-8a Digital video Lab-8a video
X Nov 11-13	**Monday Lecture** Pulsed ICs Pulsed Lecture	Lecture: Pulsed ICs Digital Summary	21-1/2	**Lab-8b, *Pulsed Digital*** (Flip-flops, counter, displays) Lab-8b Pulsed video
XI Nov 18-20 **WED EXAM**	**EXAM-II - Wed** Final Project	**EXAM-II: Magnetoelectronics, Optoelectronics, Digital/Pulsed**		***Final Project***
XII Nov 25-27	No Lecture	**Thanksgiving**		No Lab
XIII Dec 2	**Wed Lecture**	Future Electronics		***Project PowerPoint*** **due Monday Dec 2** (EG361 or email file)
XIV Dec 7-9	**No Classes**			

Pulsed Digital Circuits

- **Moore's Law**
  - growth of technology
- **Clock Speed**
- **Flip-flops**
  - RS flip-flop
  - clocked FF
  - JK flip-flop
- **Lab-8b**
  - Digital Counter

Keyboard → Computer → Monitor

What happens when you press a key on the keyboard?

Key press sends an ASCII code to the computer.
ASCII Code is a number 0-255

Keyboard effectively sends:

- **numbers for math**
- **characters for word processing**
- **special characters for functions**

The computer central processing unit (CPU)
→ converts the ASCII code to **binary** numbers (e.g. 10110...)
→ uses machine code and assembly language to process the information (**add/multiply/spell check**...)

The memory (ROM/RAM/NonVolatile) stores the information

The graphics card (GPU) converts the information for the monitor

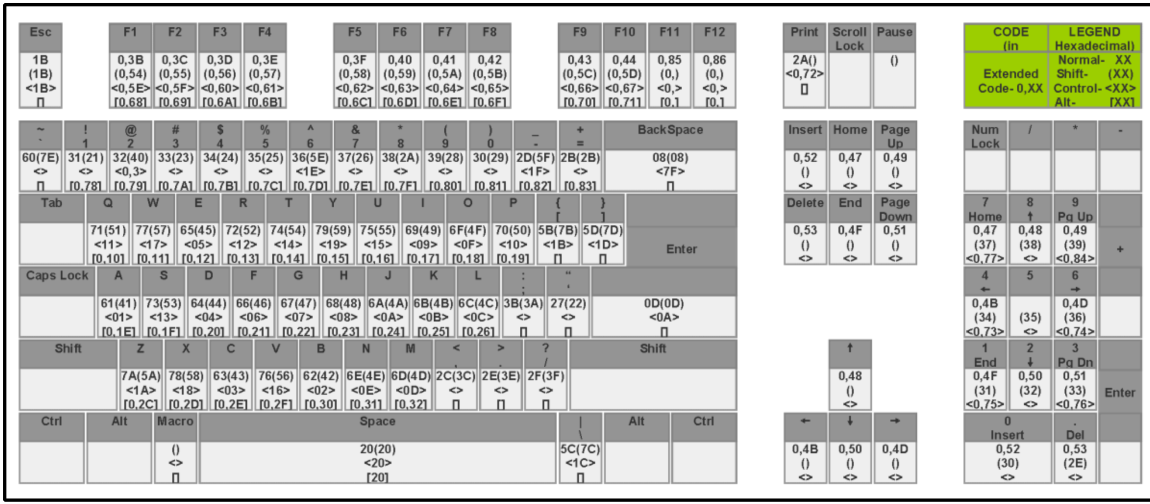
The information from the graphics card (GPU)
→ converts the information to **pixels** to display on the **LCD/LEDs** in the monitor.

ANY COMPUTER
All information is coded in NUMBERS.
These numbers can be converted to text, symbols or images.

ASCII Keyboard, Hexadecimal

Keyboard uses 8-bits bytes (two 4-bit numbers).
This requires **Hexadecimal** numbers.

Deci	Hexa
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F



Number "6_{10}" → "36_{hex}"
"36"=0011 0110

Letter "k" → "$6B_{hex}$"
"6B"=0110 1011

Character "+" → "$2B_{hex}$"
"2B"=0010 1011

Mistake: "=" equals "$3D_{hex}$" not "$2B_{hex}$"

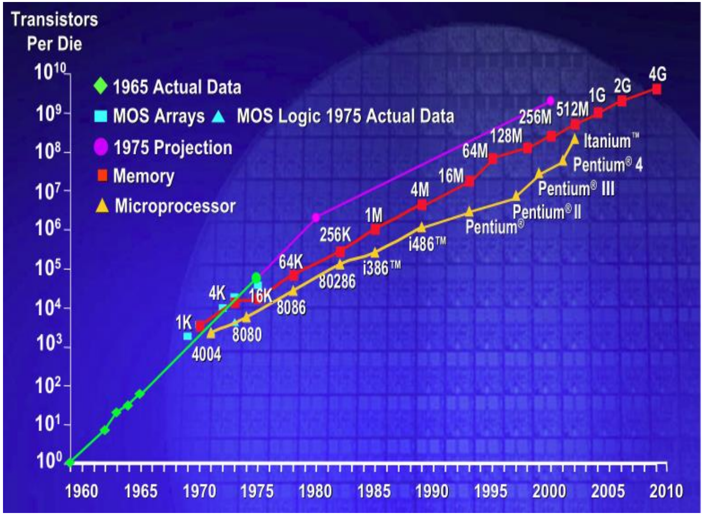
Moore's Law

The law is named after Intel co-founder Gordon E. Moore, who described the trend in his 1965 paper.

Moore's Law is the observation that, over the history of computing hardware, the number of transistors on integrated circuits doubles about every two years.

Exponential Increase with Time

- memory capacity (Moore to 1965)
- number of transistors, processing speed
- number and size of pixels in cameras



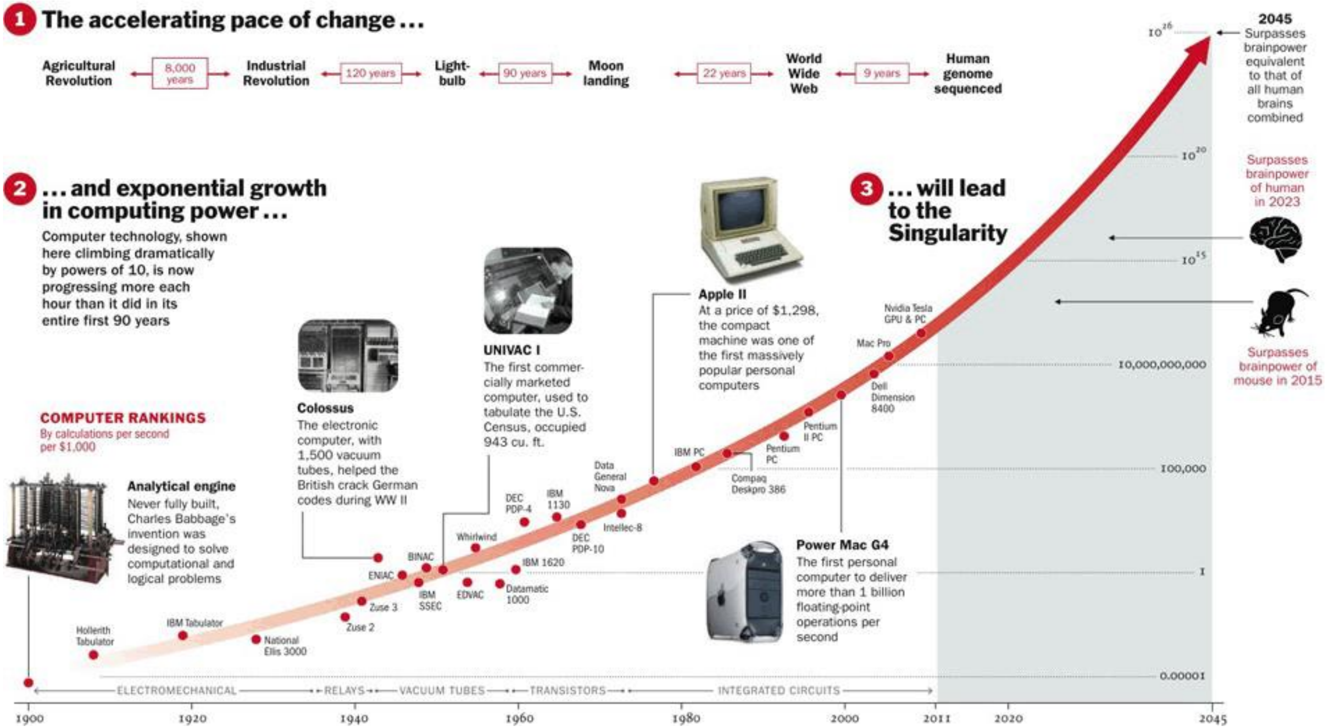
Moore's Law Got Me! **

(1:42, Mythbusters *)

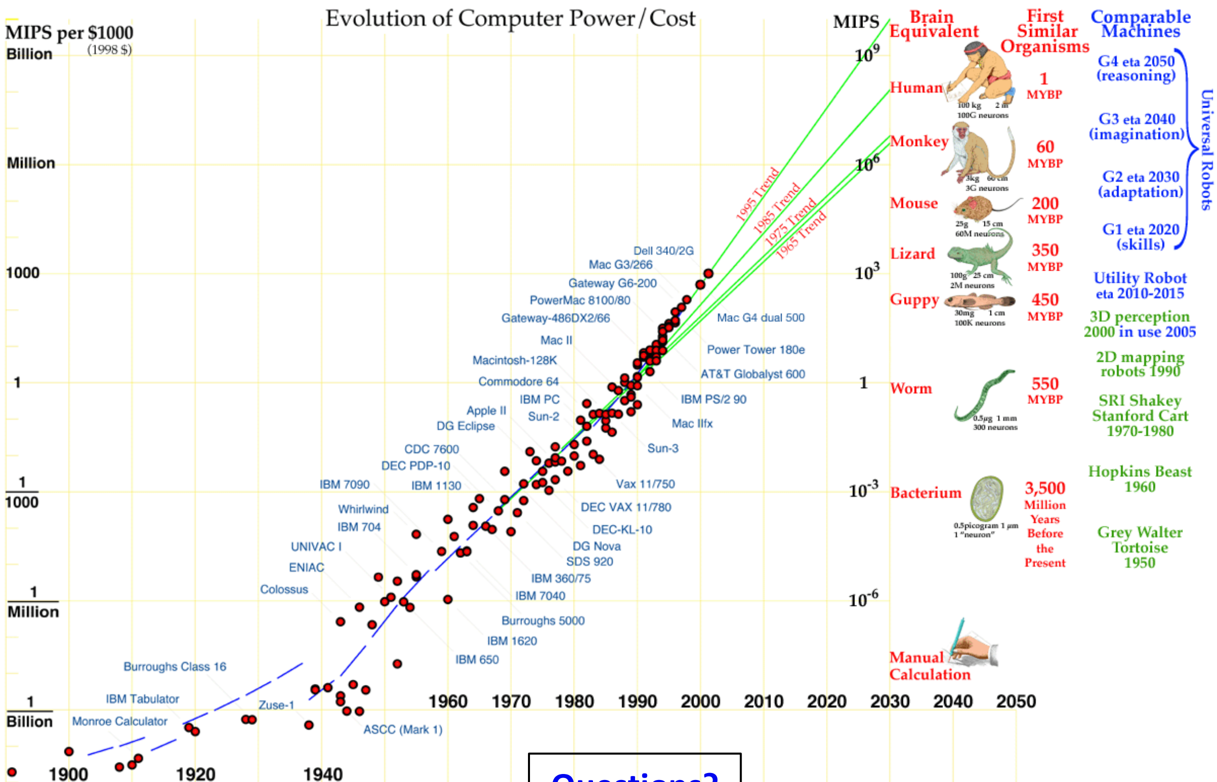
What is Moore's Law ** (2:25, 2007)

Moore's Law (11:51)

Moore's Law – Computing Power (speed/$1,000)



Moore's Law – Computing Power/Cost



Questions?

Today – Pulsed Digital ICs

Electronic Clocks

require pulsed ICs

- Digital Watches -
1 MHz crystal
divided down to 1 Hz

- **All Computers –**
need to **synchronize**
gate operations
(GHz)

Digital Pulsed ICs

- **One-Shot, Oscillator, Ch-23**
  - 74121 one-shot
  - 555 timer/oscillator
- **Flip-Flops, Ch-21**
  - RS flip-flops
  - D flip-flop (D latch)
  - T flip-flop
  - JK flip-flop (toggle)
- **Registers and Counters, Ch-24**
  - binary counter

Computer Speed – Clock Rate, Architecture, Cores

1980
Intel 8088
4.77 MHz
↓
2000
1 GHz
↓
Now
3+ GHz

Clock tic
0.3 nsec
300 psec

Clock Speed

Also called ***clock rate***, the speed at which a microprocessor executes instructions.

Every computer contains an internal clock that regulates the rate at which instructions are executed and synchronizes all the various computer components.

The CPU (central processing unit) requires a fixed number of clock ticks (or *clock cycles*) to execute each instruction.
The faster the clock, the more instructions the CPU can execute.

The internal architecture of a CPU also effects a CPU's performance, so two different CPUs with the same clock speed will not necessarily perform equally.

Whereas an Intel 80286 (16-bit) microprocessor requires 20 cycles to multiply two numbers, an Intel 80486 (32-bit) performs the same calculation in a single clock tick. 20 times faster!

Also, increasing the number of "**cores**" operating in parallel increases computation speed. (dual core, quad core, etc.)

Measuring Computer Performance – IPS and FLOPS

Instructions per Second (IPS)

Computer performance can be measured in IPS or MIPS **(million IPS).** Examples of **integer** operation include data movement (A to B) or value testing (If A = B, then C). MIPS as a performance benchmark is adequate when a computer is used in database queries, **word processing**, and **spreadsheets** (wiki).

2016 – Intel i7 CPU, 238,000 MIPS at 3.0 GHz
> 200 billion instructions/second

FLOPS – better measure of performance

In computing, **FLOPS** (for **FL**oating-point **O**perations **P**er **S**econd) is a measure of computer performance, useful in fields of **scientific calculations** that make heavy use of floating-point calculations. For such mathematical cases it is a more accurate measure than the generic IPS.

2017 – using 3 AMD commercial graphic cards ($2,500) achieved 75 TFLOPS (~ 10^{14} operations/sec)

Floating point number
e.g. **1.528535×10^{15}**

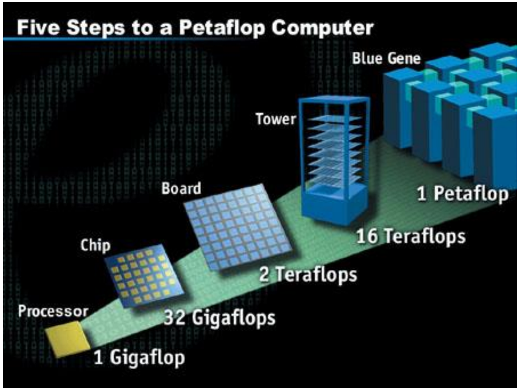
20-40 IPS ~ 1 FLOPS

Computer performance

Name	FLOPS
yottaFLOPS	10^{24}
zettaFLOPS	10^{21}
exaFLOPS	10^{18}
petaFLOPS	10^{15}
teraFLOPS	10^{12}
gigaFLOPS	10^{9}
megaFLOPS	10^{6}
kiloFLOPS	10^{3}

Supercomputer Architecture

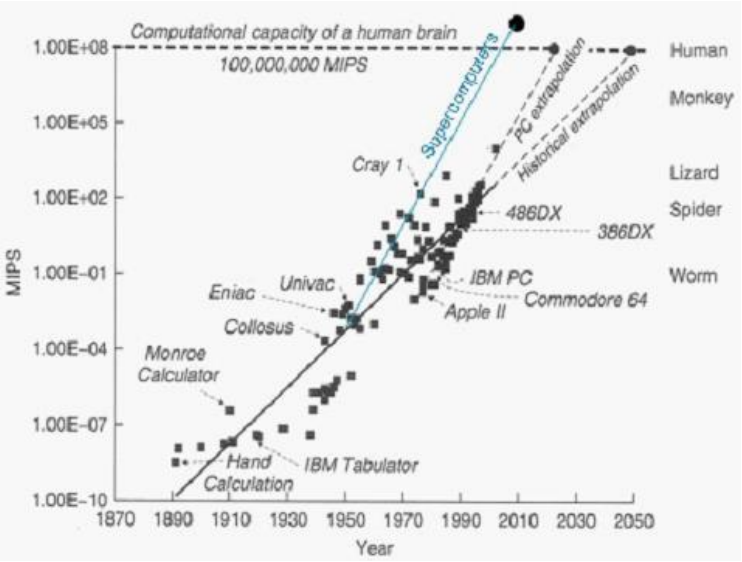
Supercomputers are "scalable"



2008 IBM Roadrunner
> 1 PFLOPS

19,000 processors, 296 computer racks
2.4 MW power, $100M

2020 Fujitsu Fugaku, Japan
415 petaFLOPS (1E13 MIPS) →



Questions?

Pulsed ICs

We will now briefly cover the following pulsed ICs:

Multivibrators

- **74121 One-Shot**
- **555 Timer/Oscillator**

Flip-Flops

- **RS Flip-Flop**
- **D Flip-Flop**
- **T Flip-Flop** (toggle)
- **JK Flip-Flop** (universal)

74121 One-Shot

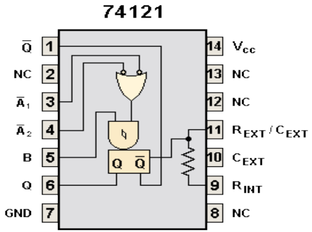
skip

Suppose you want to lengthen a pulse to a preset time.

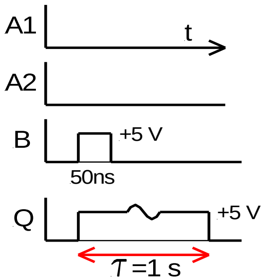
Example: Lengthen a 50 ns TTL pulse to light an LED. Turning on an LED for only 50 ns would be nearly invisible.

Stretch 50 ns → 1 s

74121 One-Shot
Monostable Multivibrator



Monostable Multivibrator
(One-Shot)

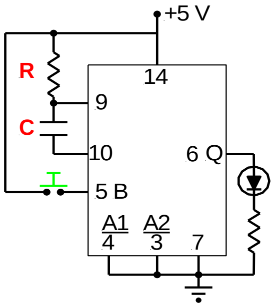


Input triggers:
A1, A2 – edge triggers
B – level trigger
Q – output

$\tau(s) = \ell n(2)\, R(\Omega) C(F)$

R=140 kΩ, C=10 μF

τ = 1 s



555 as Monostable Multivibrator (one-shot)

skip

555 as a One-Shot

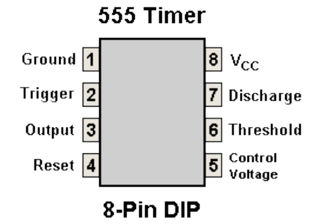
Use: lengthen a pulse to a preset time.
Stretch 50 ns → 1 s

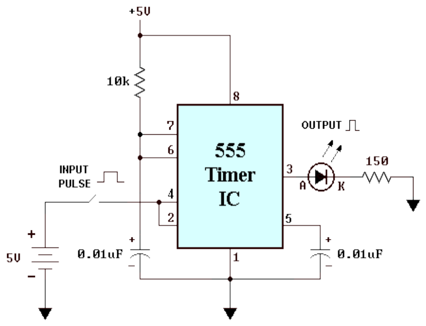
Trigger

Trigger (pin 2) is pulled HIGH when the switch is closed

$\tau(s) = \ell n(3)\, R(\Omega) C(F)$
$= 1.1RC$

$R = 100\ k\Omega$, $C = 10\ \mu F$
$\tau = 1.1\ s$





555 Timer as a Monostable Multivibrator

555 as Astable Multivibrator (oscillator)

skip

555 as an Oscillator

Triggers itself

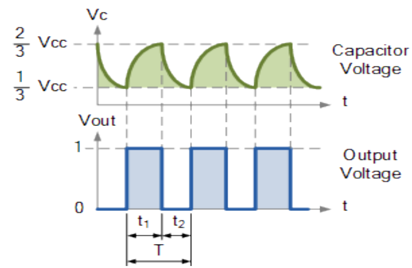
C_1 charges up from
1/3 V_{cc} to 2/3 V_{cc}

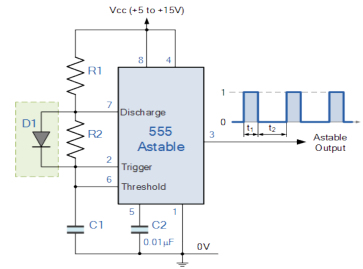
Charges to $t_1 = \ell n(2)\,(R_1+R_2)C_1$

At 2/3 V_{cc} discharge $t_2 = \ell n(2)\,R_2C_1$

On time: $t_1=0.693\,(R_1+R_2)C_1$
Off time: $t_2=0.693\,R_2C_1$

Period: $T=t_1+t_2=0.693(R_1+2R_2)C_1$
Frequency: $f=1.44/[(R_1+2R_2)C_1]$
Duty Cycle: on-time/period
$D = R_1/(R_1+2R_2)$





Example
Flash an LED on for 0.1 s every second
For $t_{off} > t_{on}$, add diode across R_2
Then on time = t_1 = 0.693 R_1C
Off time = t_2 = 0.693 R_2C
For C=10 µF
R_1=14.4 kΩ, R_2=130 k Ω

See 555 Tutorial

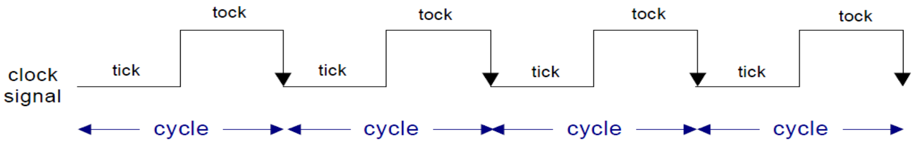
* Vary components in a running oscillator

Digital ICs that use Pulses

These include **Flip-Flops**, **Counters**, and **Displays**

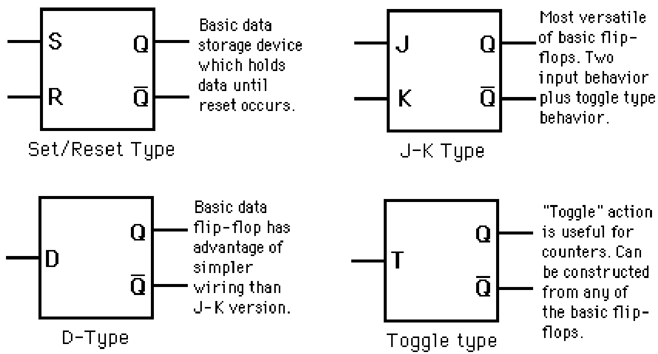
Before, we had digital voltages that were more or less constant in time. When sending digital information, or performing computations, you need a train of digital pulses.
→ Sequential Logic

Computers operate by performing logic operations (AND/OR/NOT) sequentially in time. There is a **clock-rate** that runs at GHz pulse rates. Thus, logic gate operations change several times every nanosecond. And 64 operations (*64-bit*) can be performed at one time (Parallel Logic).



Flip-Flop Types

Flip-flops are heavily used for digital data **transfer** and **storage** and are commonly used in banks called "registers" for the storage of binary numerical data.



1 or 2 inputs

2 outputs
Q and Q

See: [Flip-flop (electronics)] (wiki)

Flip-Flops

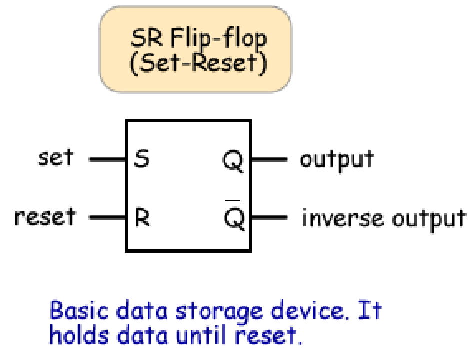
Why Flip-Flips?

Basic building block of all **memory**, counters, binary math

What is a Flip-Flop?

"Flip-flop" is the common name given to two-state devices which offer basic memory for sequential logic operations.

- **Two** outputs, two stable states
  Outputs (Q, $\underline{Q}$), (Q=1 or 0)
- **Bistable** Multivibrator
  Also called a **Latch**, pulse sets Q and it remains there



RS Flip-Flop

S=Set → **Q=1, $\underline{Q}$=0**
R=Reset → **Q=0, $\underline{Q}$=1**
Set or reset the output Q

The first electronic flip-flop was invented in 1918 by William Eccles and F. W. Jordan. It consisted of two active elements (vacuum tubes).

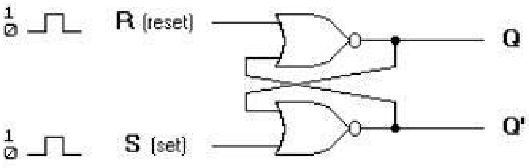
RS Flip-Flop

Example: RS Flip-Flop

S=Set,   R=Reset

Uses positive-going pulses

Contains 2 NORs



NOR - any 1 gives a 0

0 and 0 gives a 1

A	B	OR	NOR
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

RS Flip-Flop with pulse input

R	S	new Q	new Q	
0	0	nc	-	
0	1	1	0	set
1	0	0	1	reset
1	1	?	?	

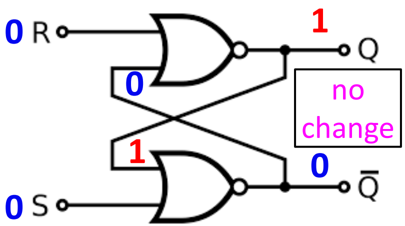
nc=no change (last value)

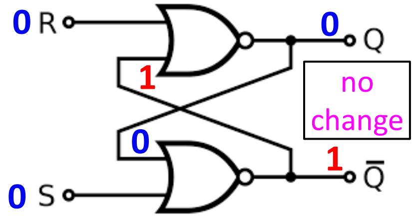
Feedback wires maintain constant output values

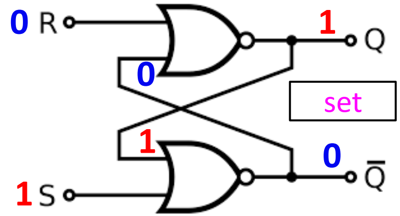
See how an RS FF switches (wiki)

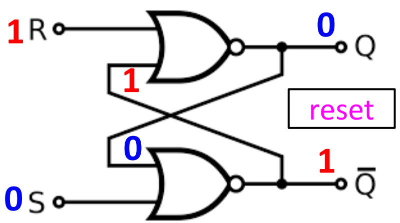
RS Flip-Flop

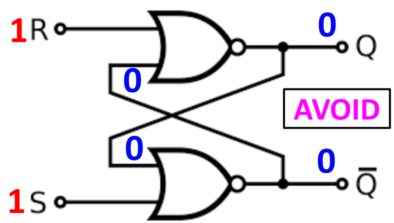
NOR – (any 1 gives a 0) – (0 and 0 gives a 1)











RS Flip-Flop with pulse input

R	S	new Q	new $\bar{Q}$	
0	0	nc	-	
0	1	1	0	set
1	0	0	1	reset
1	1	?	?	

nc=no change (last value)

Questions?

RS Flip-Flop

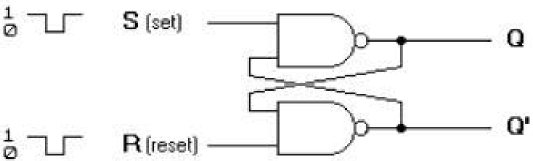
skip

Example: RS Flip-Flop

S=Set, R=Reset

Uses negative-going pulses

Contains 2 NANDs



NAND - any 0 gives a 1

A	B	AND	NAND
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

RS Flip-Flop with pulse input

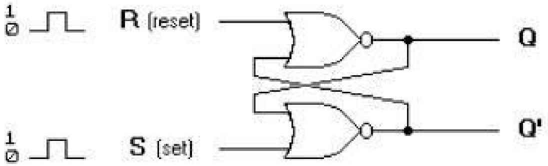
S	R	new Q	new Q	
1	1	nc	-	
0	1	1	0	(set)
1	0	0	1	(reset)
0	0	?	?	

nc=no change (last value)

Summary: RS and RS Flip-Flops

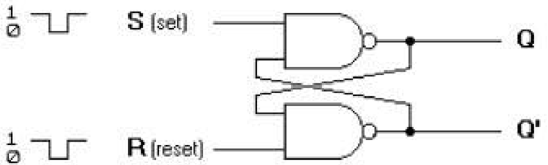
RS Flip-Flop

Uses positive-going pulses
Contains 2 NORs
Any 1 gives a 0



RS Flip-Flop

Uses negative-going pulses
Contains 2 NANDs
Any 0 gives a 1

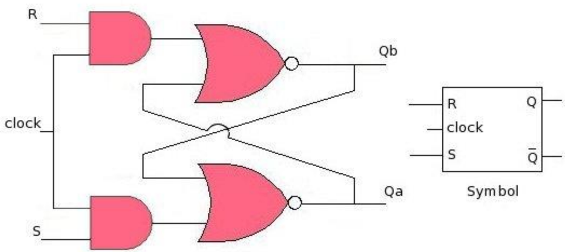


Don't get confused about the RS or RS notation.
RS refers to a positive-going pulse.
RS refers to a negative-going pulse.
Most people simple drop the (NOT) bars on R and S and assume either positive-going or negative-going pulses.

Questions?

Clocked Flip-Flop

Clock Pulse Notation
Clock = CK = CLK = CP = Enable
CLK, also called **Enable**

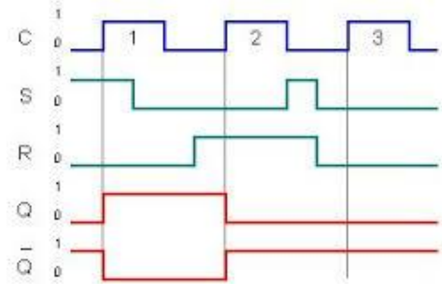


Clocked SR Flip-Flop

Clock pulse enables inputs

Nothing changes unless there is a clock pulse

CLK = 0 → no change in outputs
CLK = 1 → new RS changes output

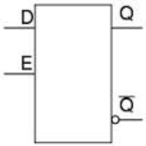


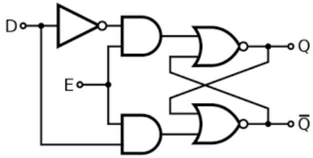
Edge (rising) triggering

D Latch – Value Enabled

skip

D latch
Only one input = **D**
plus enable = **E**

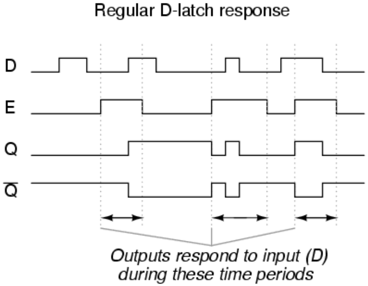




The D Latch captures the value of the D-input whenever enable is on, E=1.

That captured value becomes the Q output. At other times, the output Q does not change.

Value Enabled
Depends on the value of E, not edge triggered.



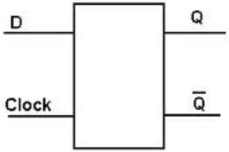
Value Enabled

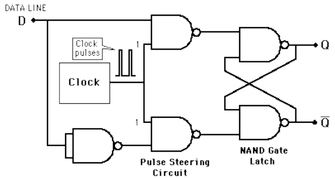
D-latch and Flip-Flop (0-4:53, shows timing), 15:42

Clocked D Flip-Flop – Edge Triggered

By far the most important FF
Stores one input (MEMORY)

Clocked D Flip-Flop
Only one input = **D**
plus clock pulse = **CLK**

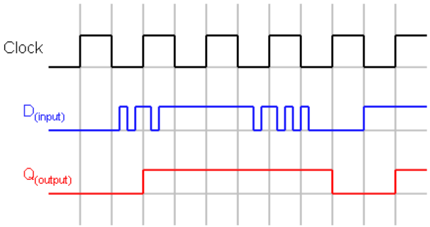




The D flip-flop captures the value of the D-input at a definite portion of the clock cycle (such as the **rising edge** of the clock).
Edge Triggering

That captured value becomes the Q output.
At other times, the output Q does not change.

The D flip-flop can be viewed as a **memory cell**, a zero-order hold, or a delay line.



Edge triggering (rising)

* Watch: **D-latch and Flip-Flop** (11:20-12:10), 15:42

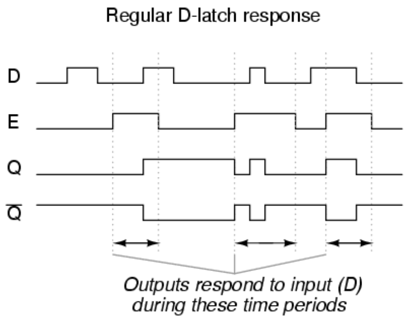
Enabling/Triggering Flip-Flops

skip

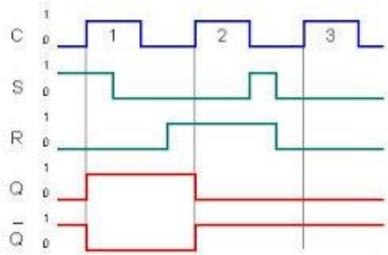
D Latch – value enabled

D Flip-Flop – edge triggered

Value Enabling



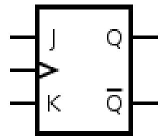
Edge triggering

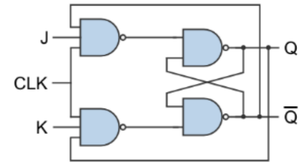


JK Flip-Flop

JK Flip-Flip

Most powerful
Can be configured as
RS-FF, D-FF, T-FF





The JK flip-flop augments the behavior of the SR flip-flop by interpreting **J=Set** and **K=Reset**.

Q_0 goes into Q_{next}

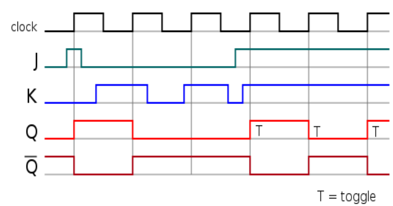
- The combination J=1, K=0 is a command to **set** the flip-flop ($Q_{next}=1$)
- The combination J=0, K=1 is a command to **reset** the flip-flop ($Q_{next}=0$)
- The combination J=K=1 is a command to **toggle** the flip-flop ($Q_0 \rightarrow \overline{Q_0}$)
  On the CLK edge, Q_0 is set to $Q_{next}=\overline{Q_0}$.

$$Q_{next} = J\,\overline{Q_0} + \overline{K}\,Q_0$$

JK Flip-Flop

J	K	CLK	Qnext	
0	0	↑	Q_0	hold
1	0	↑	1	set
0	1	↑	0	reset
1	1	↑	$\overline{Q_0}$	toggle

OK



Watch: JK and T Flip-Flop, (0-8:15) 13:09 *

Questions?

T Flip-Flop: TOGGLE

skip

Clocked T Flip-Flop: TOGGLE

Only one input = **T**
plus clock pulse = **CP**

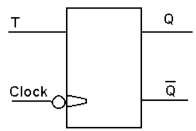
If the T input is high, the T flip-flop changes state ("toggles") whenever the clock input is strobed.

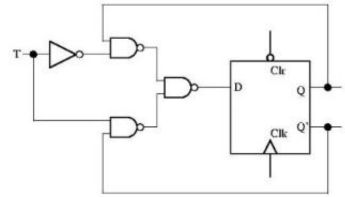
Simply $Q \rightarrow \overline{Q}$

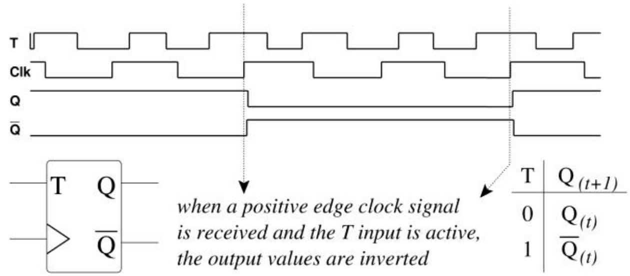
If the T input is low, the flip-flop holds the previous value.

This behavior is described by:

$$Q_{next} = T\,\overline{Q} + \overline{T}\,Q$$







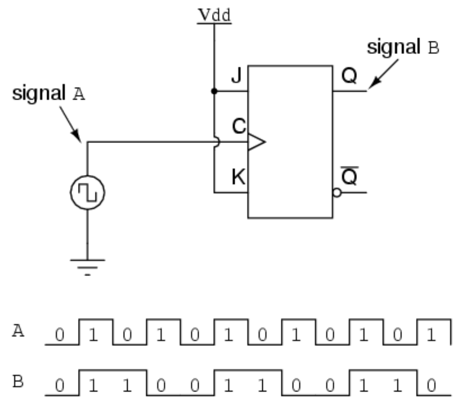
* Watch: [JK and T Flip-Flop](), (8:15+) 13:09

Toggle from JK Flip-Flop

Make JK-FF into
simple **Toggle**

Set J=K=1
Each time a clock pulse edge comes,
the output is toggled.

Q → $\underline{Q}$ → Q → $\underline{Q}$ → Q →



Use toggle from JK flip-flop
as a **binary counter**

Frequency is ½
f → *f*/2

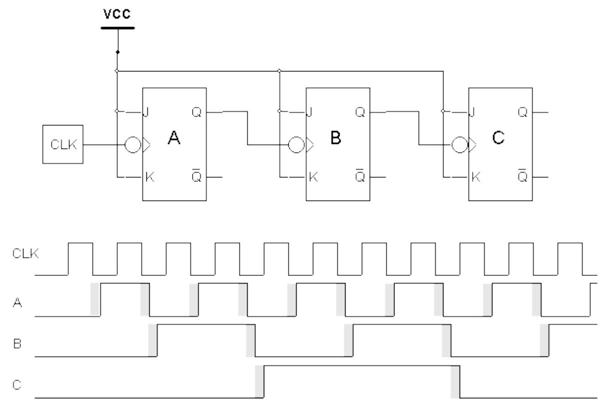
Questions?

Binary Counter, Ch. 24

Binary Counter

Counts how many clock pulses come by

Use the JK-FF in **TOGGLE** mode



Binary Number DCBA

D	C	B	A
2^3	2^2	2^1	2^0
8	4	2	1
Q_2	Q_1	Q_0	CLK
FF2	FF1	FF0	CLK

Note:
"A" is the least significant bit in DCBA (e.g. 0101)

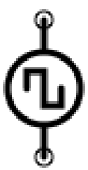
* Watch: Binary Counter, 20 sec

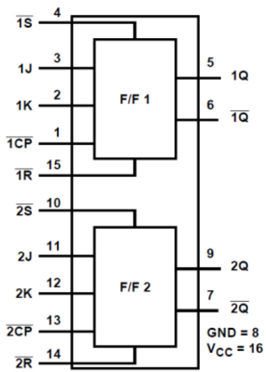
See Binary Counter details, 10 min

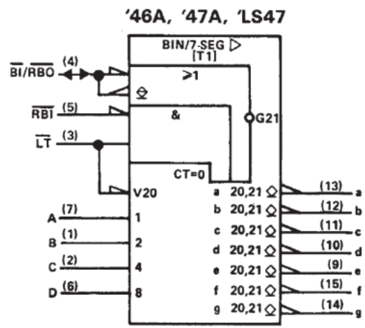
Lab-8b
Flip-Flop, Counters, Displays

- Design and construct a binary counter circuit using JK flip-flops
- The circuit cycles through the binary numbers 000-111
- Convert binary numbers to BCD
- Light LED digital number display

Square wave generator	+	Combine 3 74112 flip flops	+	SN74LS47N BCD-to-seven-segment decoder/driver	+	LTS-4801B 7-segment LED for digits 0-9

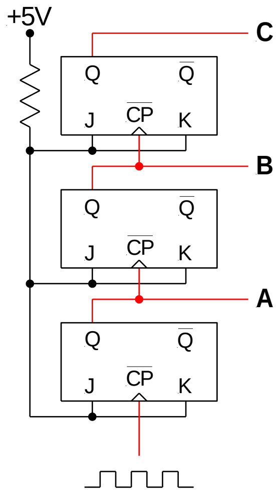




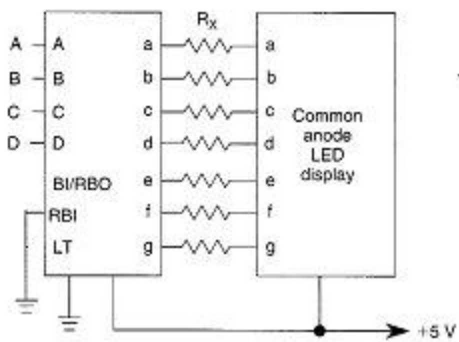


Lab-8b, Roll the Dice

Make 4-bit Counter
Three 74112 JK-FF



SN74LS47N
BCD-to-seven-segment
common anode
decoder/driver



R_x=200-500 Ω

LTS-4801B
7-segment LED
for digits 0-9
common anode

See Decoder/Display

Note: "A" is the least significant bit (CBA)

Wednesday EXAM-2, Nov 18
Magnetoelectronics
Optoelectronics
Digital Electronics
Pulsed ICs

Due Wed, Nov 18
Homework Ch. 21
Lab-8a and 8b in **one** report

mwisho