Storage Elements

Basic Computational Model:

• So far we have discussed ***combinational components***, i.e. the output is a function only of the current inputs

• How do we "store" the values of variables for later use?

Sequential Components:

• In ***sequential components***, the output is a function of the current inputs AND the ***current state*** of the circuit

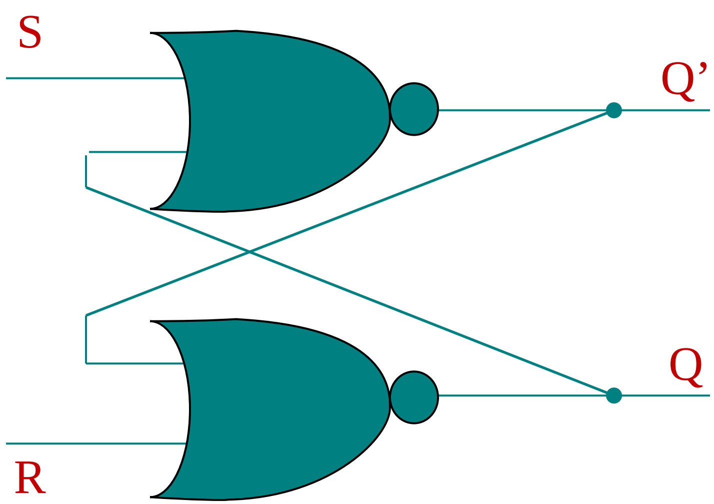
• The ***current state*** is a function of a sequence of inputs over multiple time steps, starting from a known state.

• The state is stored in storage elements:
(e.g. Latches, Flip-Flops, Registers, Memory)

Set/Reset Latch (S/R)

Symbol:

Gate-Level Implementation:



Truth Table:

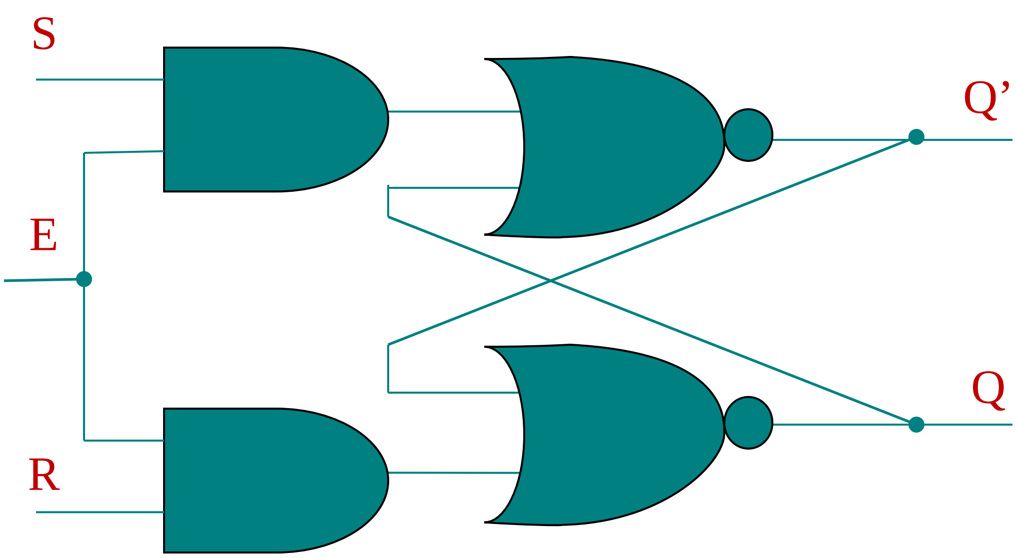
	S	R	Q_t	Q_{t+1}	Q'_{t+1}
Keep{	0	0	0	0	1
	0	0	1	1	0
Reset{	0	1	0	0	1
	0	1	1	0	1
Set{	1	0	0	1	0
	1	0	1	1	0
Illegal{	1	1	0	?	?
	1	1	1	?	?

Notice that for the first time we allow “feedback” between the combinational gates

Gated S/R Latch

Symbol:

Gate-Level Implementation:



Truth Table if E=1:

	S	R	Q_t	Q_{t+1}	Q'_{t+1}
Keep{	0	0	0	0	1
	0	0	1	1	0
Reset{	0	1	0	0	1
	0	1	1	0	1
Set{	1	0	0	1	0
	1	0	1	1	0
Illegal{	1	1	0	?	?
	1	1	1	?	?

Truth Table if E=0:

IF Q_t=1,
Q_{t+1} =1 and Q'_{t+1} =0
IF Q_t=0,
Q_{t+1} =0 and Q'_{t+1} =1

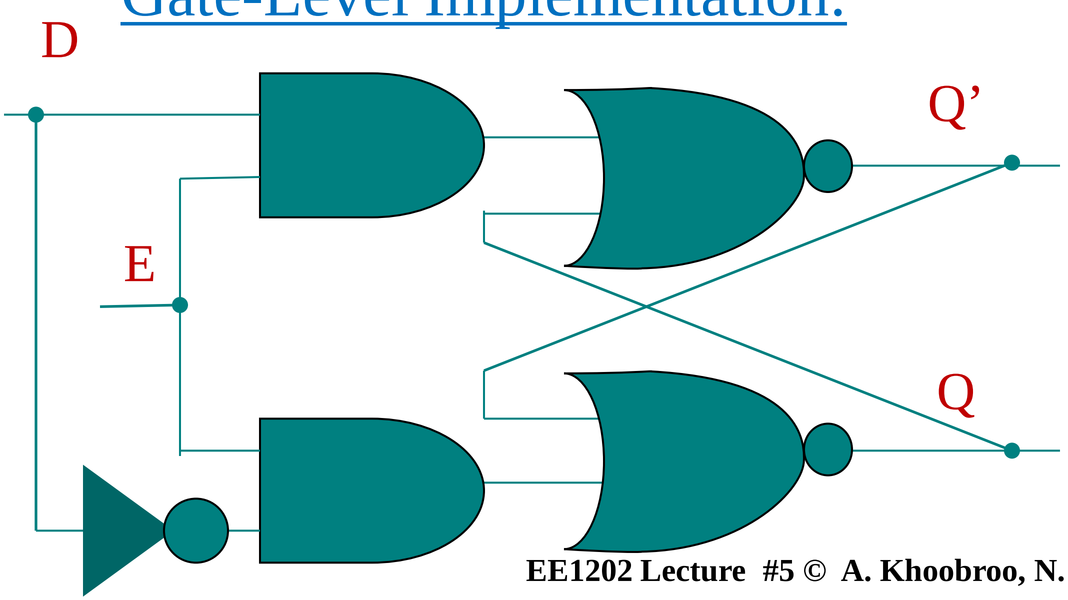
D-Latch

Avoiding the "illegal combinations":

Force S and R to be always different

Symbol:

Gate-Level Implementation:



Truth Table:

	E	D	Q_t	Q_{t+1}	Q'_{t+1}
Keep	0	0	0	0	1
	0	0	1	1	0
	0	1	0	0	1
	0	1	1	1	0
Read	1	0	0	0	1
	1	0	1	0	1
	1	1	0	1	0
	1	1	1	1	0

D stands for "Data"!!!

D-Latch provides basic memory capability

Introducing the “Clock”

Asynchronous vs. Synchronous Design:

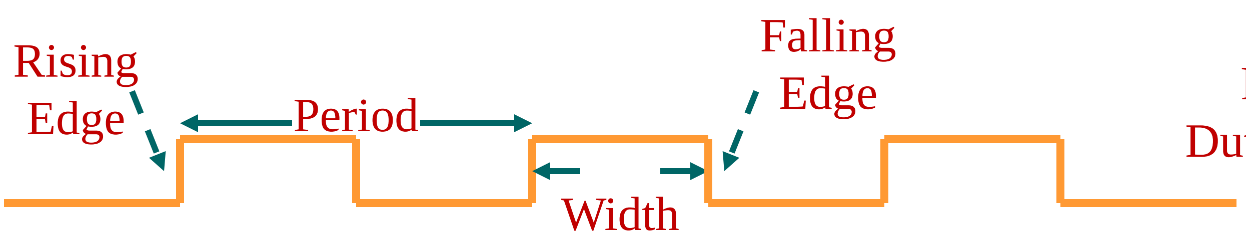
• Signals may take different paths in the design and arrive at any arbitrary time and order

• The Latches that we examined will respond immediately to any signal change at their inputs, so the result may depend on the time and order of signal arrival

• This makes our circuit unpredictable
How do we synchronize the computation ?

Clock:

A periodic signal that imposes a synchronization point



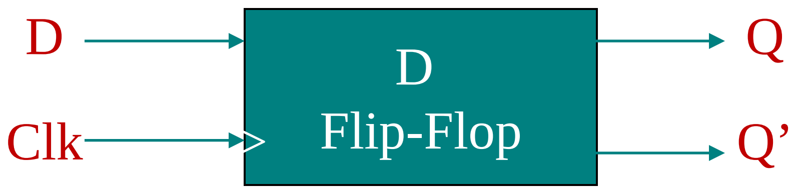
Frequency = 1 / Period
Duty Cycle = Width / Period

D Flip-Flop Master/Slave

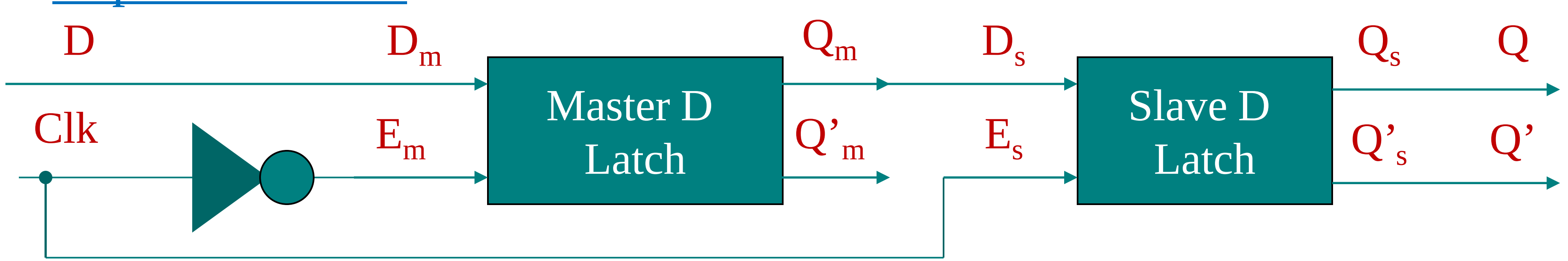
Basic Idea:

Connect two latches in series, the "Master" and the "Slave". Use a clock signal to enable the Slave and the inverse of the clock signal to enable the Master. The Master stores the result while the clock is 0 and then passes it to the Slave when the clock is 1.

Symbol:



Implementation:



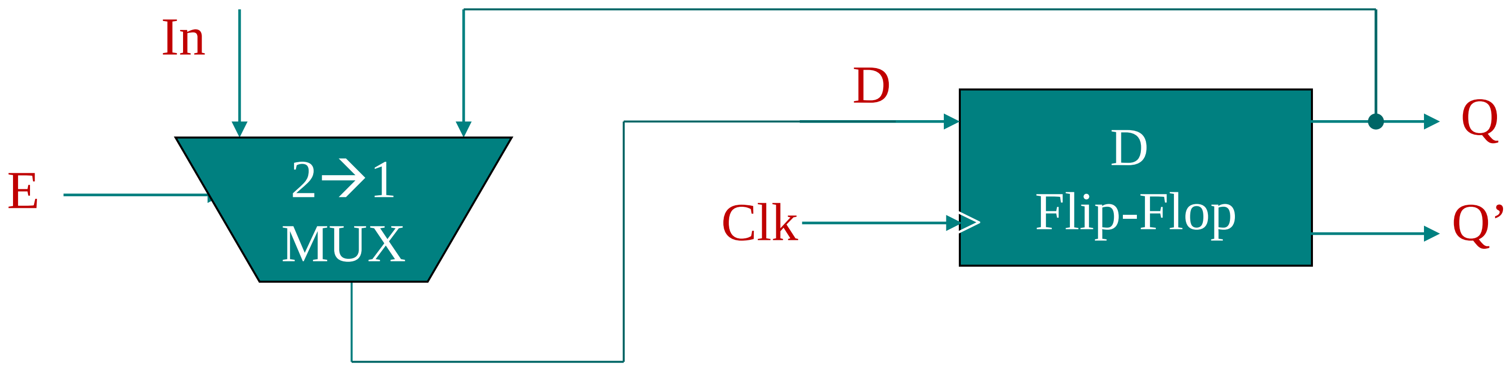
D Flip- Flop with Enable

Operation:

E	Function	Comment
0	$Q_{t+1}=Q_t$	Keep Previous Value
1	$Q_{t+1}=In_t$	Load Input Data

Symbol:

Implementation:



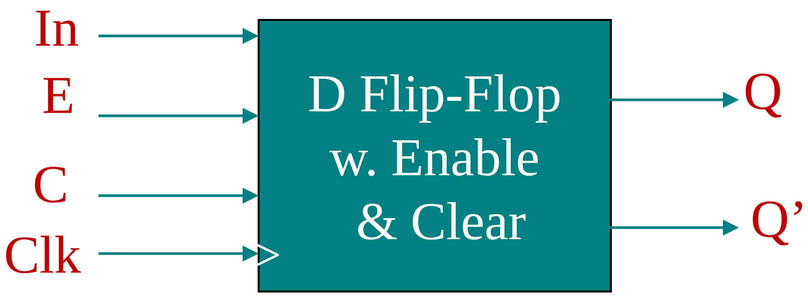
Note: The MUX Passes the In to the D when E=1 and the Q to the D when E=0

D Flip- Flop with Enable and Clear

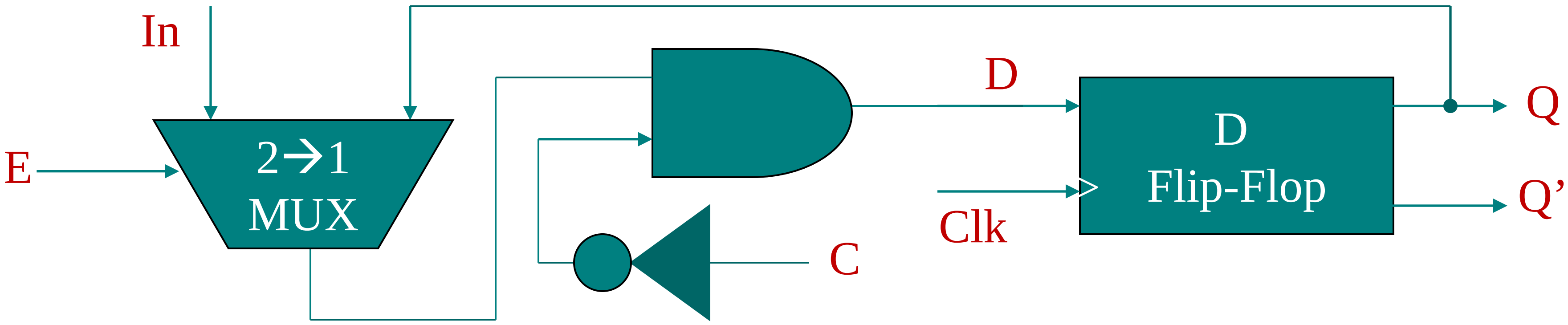
Operation:

C E	Function	Comment
0 0	$Q_{t+1}=Q_t$	Keep Previous Value
0 1	$Q_{t+1}=In_t$	Load Input Data
1 0	$Q_{t+1}=0$	Clear to 0
1 1	$Q_{t+1}=0$	Clear to 0

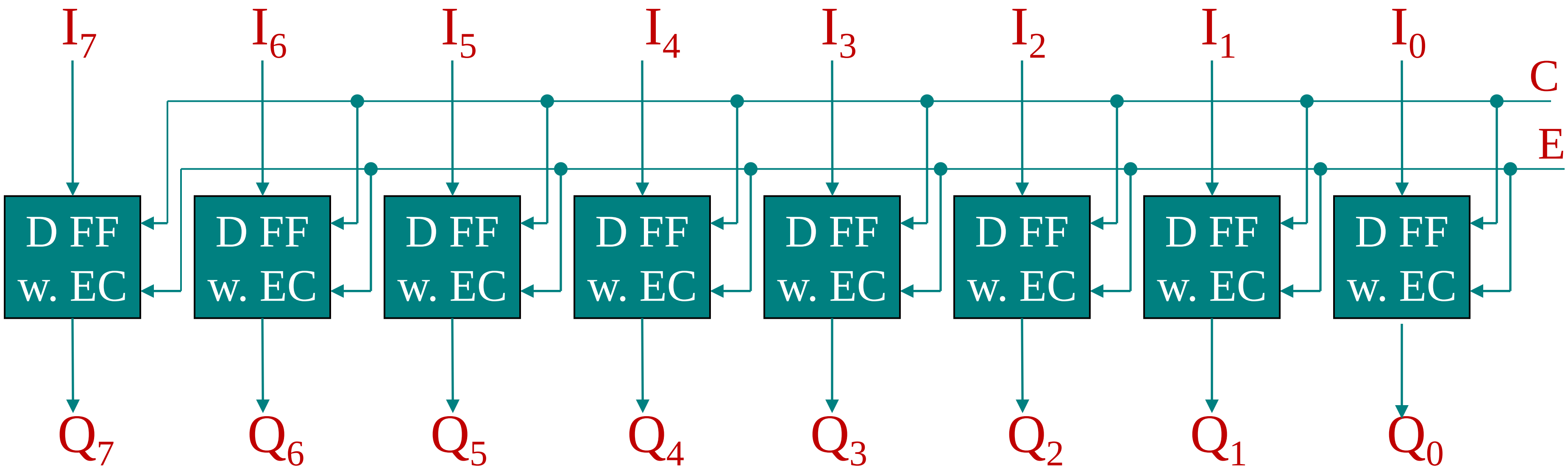
Symbol:



Implementation:



8-Bit Register with Enable and Clear

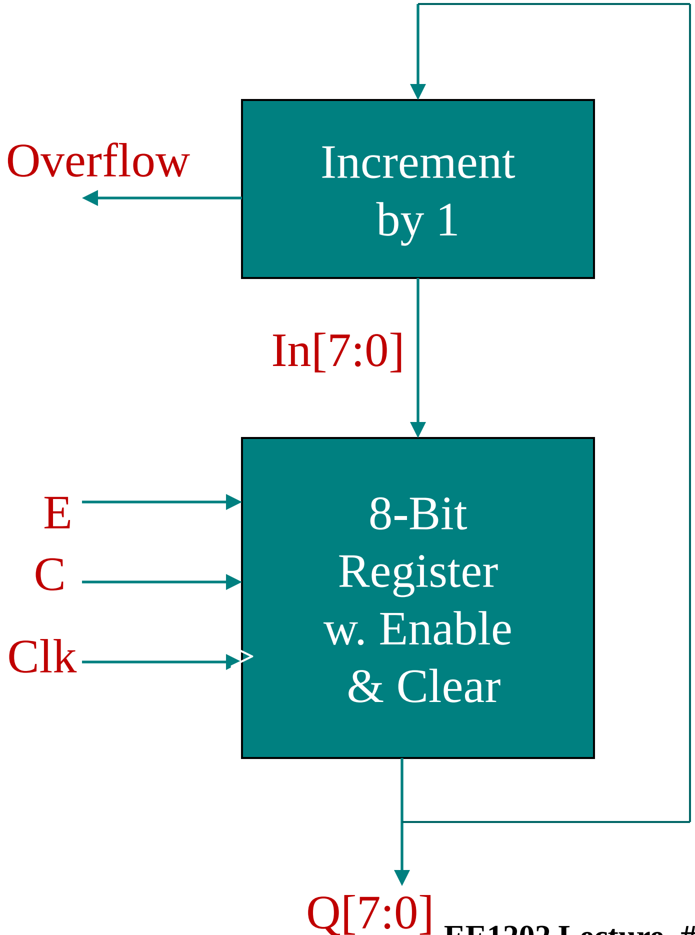


Note: The same Enable E and Clear C signals are used to drive all D Flip-Flops so that they all Clear, Load from the inputs I[7:0], or Keep the previous Values Q[7:0]

(The clock signals are not shown for clarity of the figure)

Register-Based Circuit

Block Diagram:



Operation:

- Initialize the Register to "00000000" by C=1 and sending a clock pulse

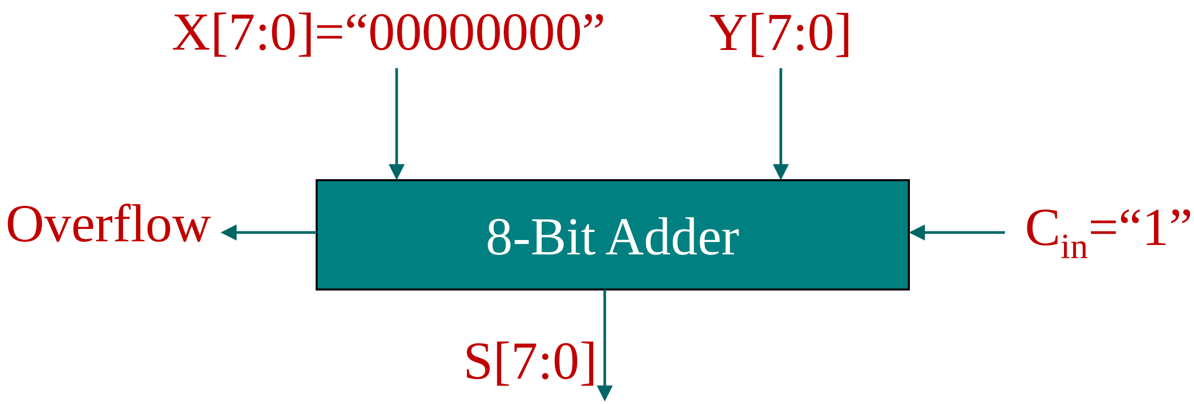
- Take the current value of the Register, Q[7:0],
increment it by "1" and store it back into the Register by C=0, E =1, and sending a clock pulse

It's a counter!!!

- Overflow asserted when counter "wraps around"

Building an increment by “1” Circuit

Simple Idea – Use an Adder:



Hardware Reduction:

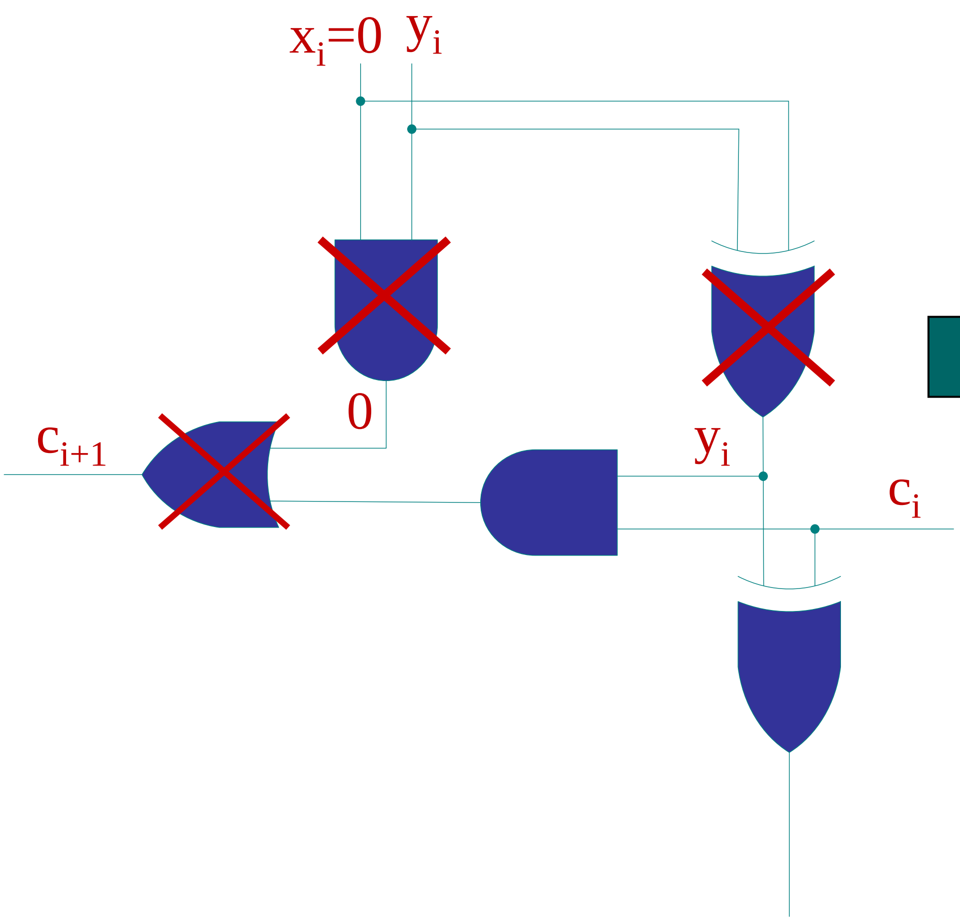
$s_i = x_i \oplus y_i \oplus c_i$ but $x_i=0$, so $s_i = 0 \oplus y_i \oplus c_i$ => $\mathbf{s_i = y_i \oplus c_i}$

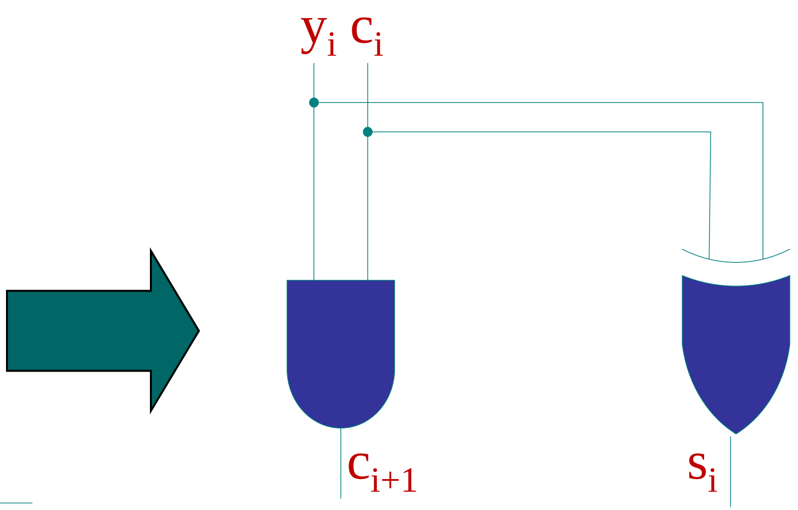
$c_{i+1} = x_i \cdot y_i + c_i \cdot (x_i \oplus y_i)$ but $x_i=0$, so $c_{i+1} = 0 \cdot y_i + c_i \cdot (0 \oplus y_i)$ => $\mathbf{c_{i+1} = c_i \cdot y_i}$

Special Case: $s_0 = c_{in} \oplus y_0$ => $s_0 = 1 \oplus y_0$ => $\mathbf{s_0 = y_0'}$
and $c_1 = c_{in} \cdot y_0$ => $c_1 = 1 \cdot y_0$ => $\mathbf{c_1 = y_0}$

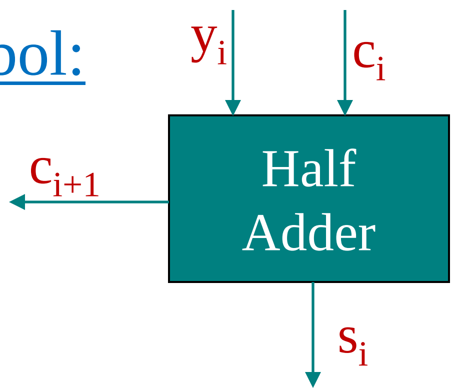
Reduced Hardware Implementation

Full Adder Reduces to Half Adder:



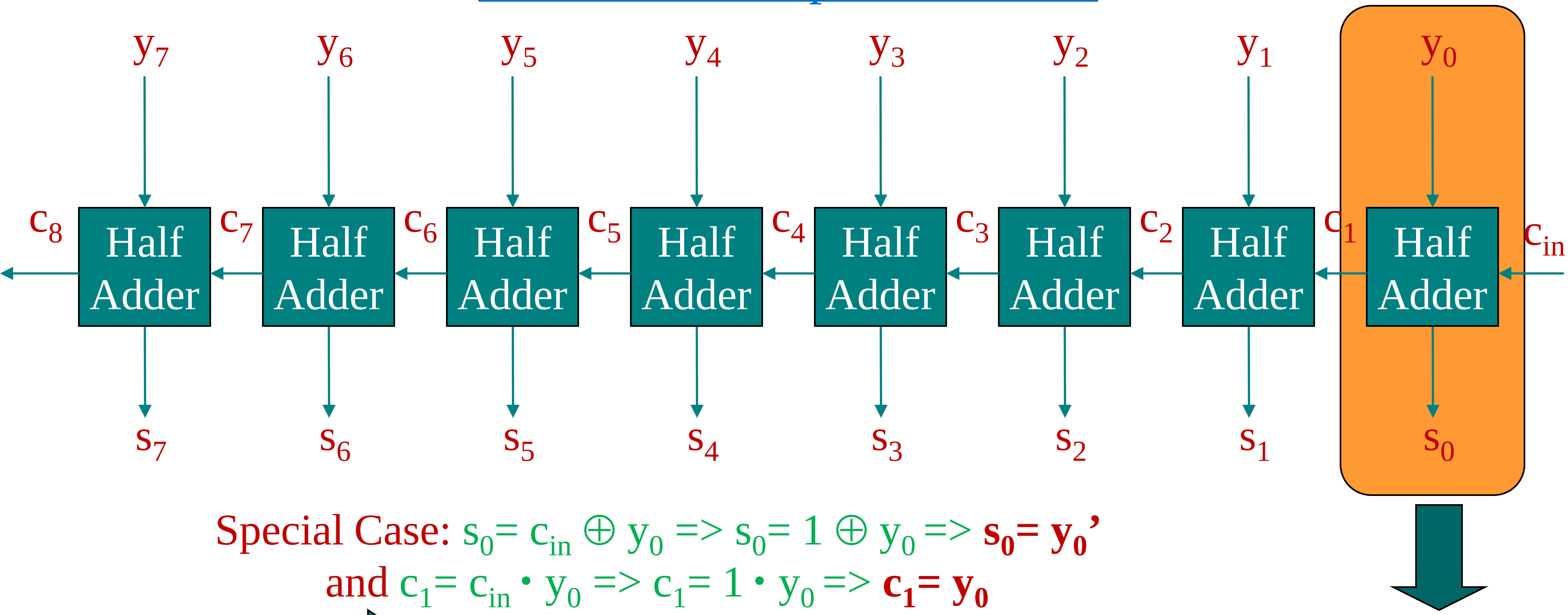


Symbol:



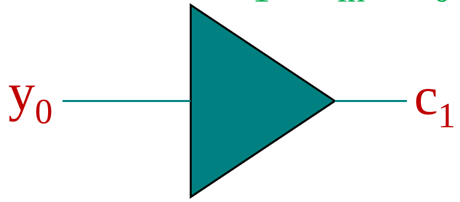
8-Bit increment Circuit

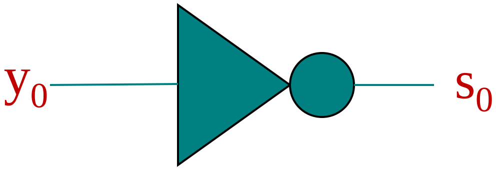
Hierarchical Implementation:



Special Case: $s_0 = c_{in} \oplus y_0 \Rightarrow s_0 = 1 \oplus y_0 \Rightarrow \mathbf{s_0 = y_0'}$

and $c_1 = c_{in} \cdot y_0 \Rightarrow c_1 = 1 \cdot y_0 \Rightarrow \mathbf{c_1 = y_0}$



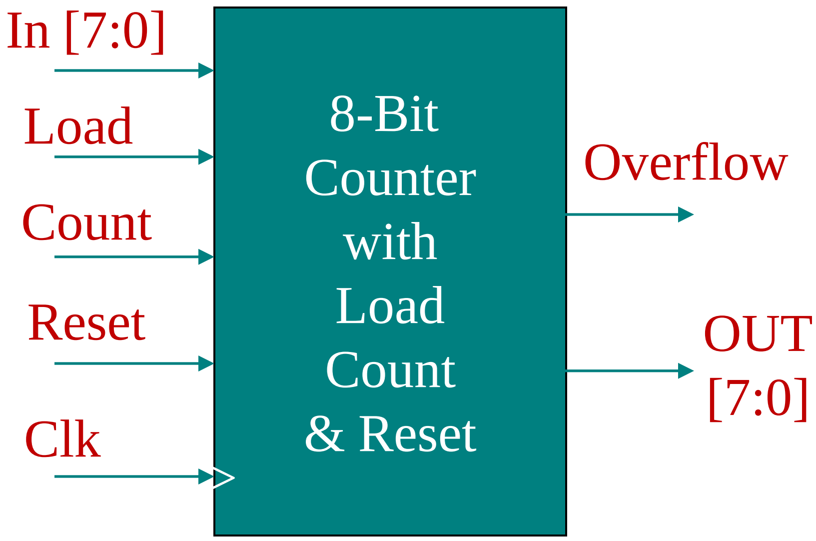


8-Bit Counter with Load, Count and Reset

Operation:

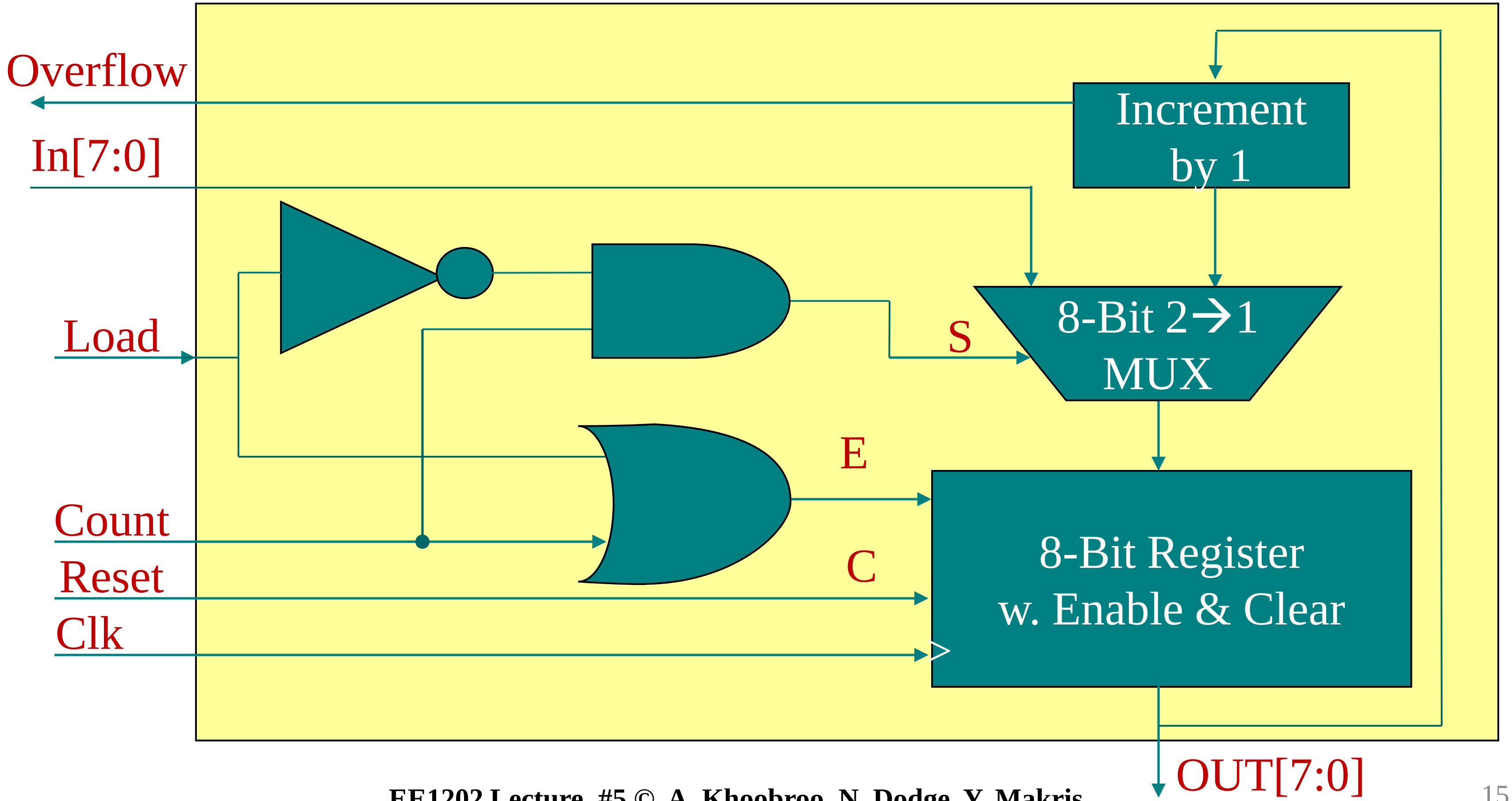
Reset	Count	Load	Operation
0	0	0	Keep Previous
0	0	1	Load from IN
0	1	0	Count Up by 1
0	1	1	Load from IN
1	0	0	Reset to 0
1	0	1	Reset to 0
1	1	0	Reset to 0
1	1	1	Reset to 0

Symbol:



8-Bit Counter with Load, Count and Reset

Block Diagram:

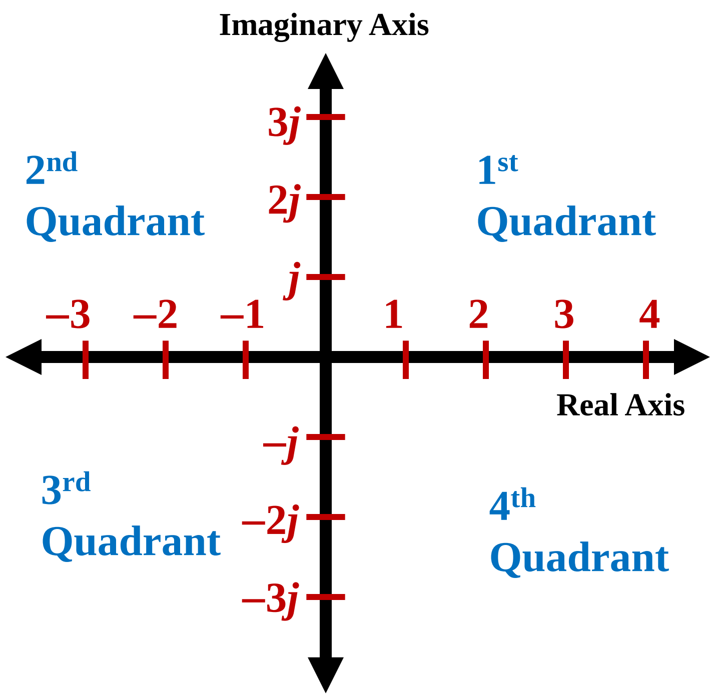


AC RL and RC Circuits

- When a sinusoidal AC voltage is applied to an RL or RC circuit, the relationship between voltage and current is altered.
- The voltage and current still have the same frequency and cosine-wave shape, but voltage and current no longer rise and fall together.
- To solve for currents in AC RL/RC circuits, we need some additional mathematical tools:
  – Using the complex plane in problem solutions.
  – Using transforms to solve for AC sinusoidal currents.

Imaginary Numbers

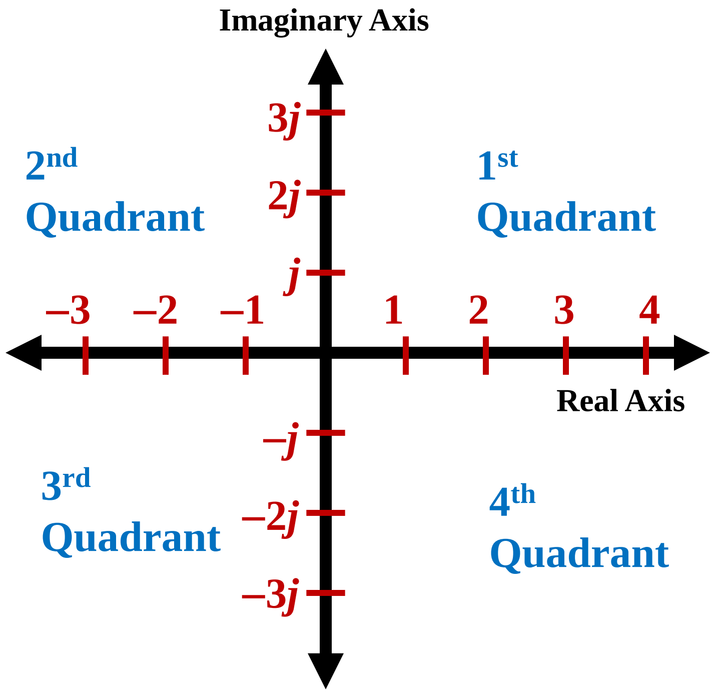
- Solutions to science and engineering problems often involve $\sqrt{-1}$.
- Scientists define $\boldsymbol{i = -\sqrt{-1}}$.
- As we EE's use *i* for AC current, we define $\boldsymbol{j = +\sqrt{-1}}$.
- Thus technically, $j = -i$, but that does not affect the math.
- Solutions that involve *j* are said to use "imaginary numbers."
- Imaginary numbers can be envisioned as existing with real numbers in a two-dimensional plane called the "Complex Plane."



The Complex Plane

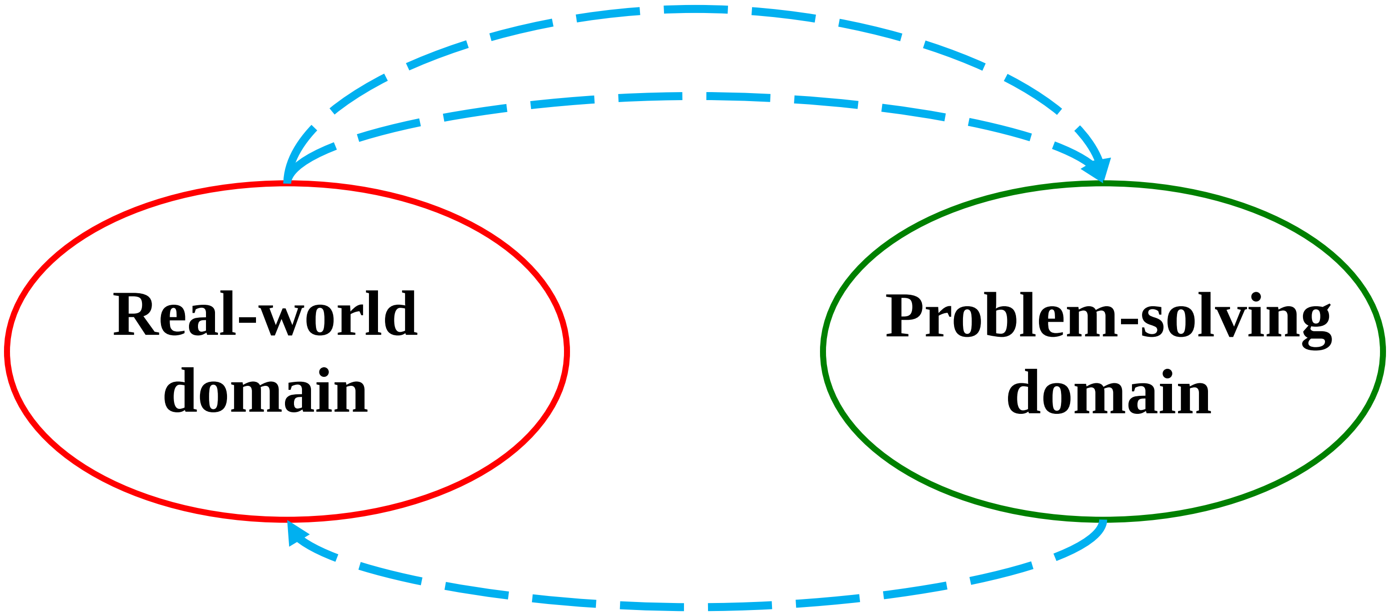
Complex Plane

- In the complex plane, imaginary numbers lie on the y-axis, real numbers on the x-axis, and complex numbers (mixed real and imaginary) lie off-axis.
- For example, 4 is on the +x axis, –8 is on the –x axis, *j*6 is on the + y axis, and –*j*14 is on the –y axis.
- Complex numbers like 6+*j*4, or –12 –*j*3 lie off-axis, the first in the first quadrant, and the second in the third quadrant.



The Complex Plane

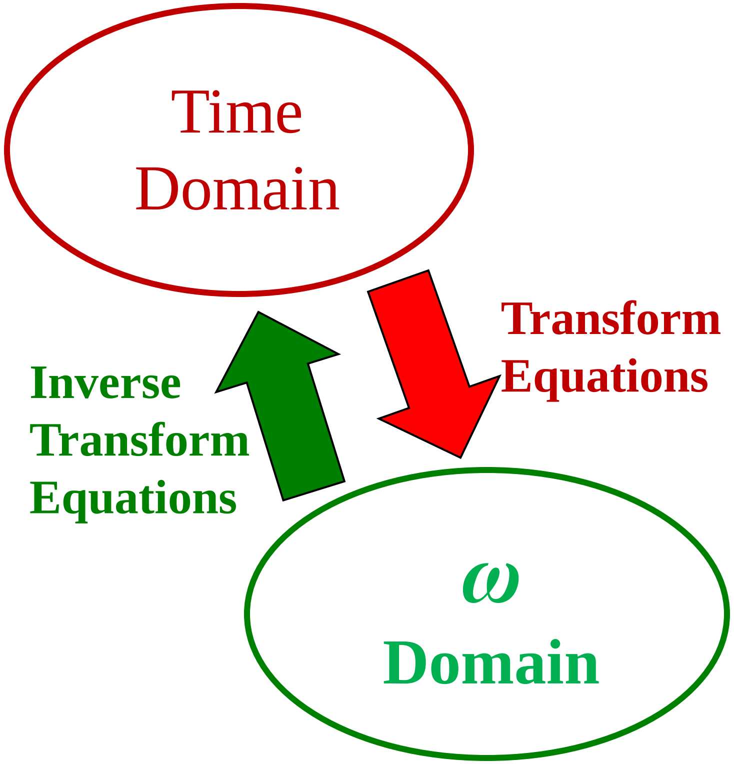
Transformation



- Transforms move a problem from the real-world domain, where it is hard to solve, to an alternate domain where the solution is easier.
- Sinusoidal AC problems involving *R*-*L*-*C* circuits are hard to solve in the "real" time domain but easier to solve in the ω-domain.

The ω Domain

- In the time domain, *RLC* circuit problems must be solved using calculus.
- However, by transforming them to the ω domain (a radian frequency domain, $\omega = 2\pi f$), the problems become algebra problems.
- A catch: We need transforms to get the problem to the ω domain, and inverse transforms to get the solutions back to the time domain!



Euler's formula

- You should remember Euler's formula from trigonometry (if not, get out your old trig textbook and review): .
- The alternate expression for $e^{\pm jx}$ is a complex number. The real part is $\cos x$ and the imaginary part is $\pm j\sin x$.
- We can say that $\cos x = \text{Re}\{e^{\pm jx}\}$ and $\pm j\sin x = \text{Im}\{e^{\pm jx}\}$, where Re = "real part" and Im = "imaginary part."
- We usually express AC voltage as a cosine function. That is, an AC voltage $v(t)$ and be expressed as $v(t) = V_p \cos \omega t$, where V_p is the peak AC voltage.
- Therefore we can say that $v(t) = V_p \cos \omega t = V_p \text{Re}\{e^{\pm j\omega t}\}$. This relation is important in developing inverse transforms.

Frequency Domain Transformation

Element	Time Domain	ω Domain Transform
AC Voltage	$V_p \cos \omega t$	V_p
Resistance	R	R
Inductance	L	$j\omega L$
Capacitance	C	$1/j\omega C$

- The time-domain, sinusoidal AC voltage is normally represented as a cosine function, as shown above.
- R, L and C are in Ohms, Henrys and Farads.
- Skipping some long derivations (which you will get in EE 3301), transforms for the ω domain are shown above.
- Notice that the AC voltage ω-transform has no frequency information. However, frequency information is carried in the L and C transforms.

Frequency Domain Transformation

Element	Time Domain	ω Domain Transform
AC Voltage	$V_p \cos \omega t$	V_p
Resistance	R	R
Inductance	L	$j\omega L$
Capacitance	C	$1/j\omega C$

- Because we are studying constant-frequency sinusoidal AC circuits, the ω-domain transforms are constants.
- This is a considerable advantage over the time-domain situation, where t varies constantly (which is why solving for sinusoidal currents in the time domain is a calculus problem).
- Two other items:
  - In the ω-domain, the units of R, $j\omega L$, and $1/j\omega C$ are Ohms.
  - In the ω-domain, Ohm's Law and Kirchoff's voltage and current laws still hold.