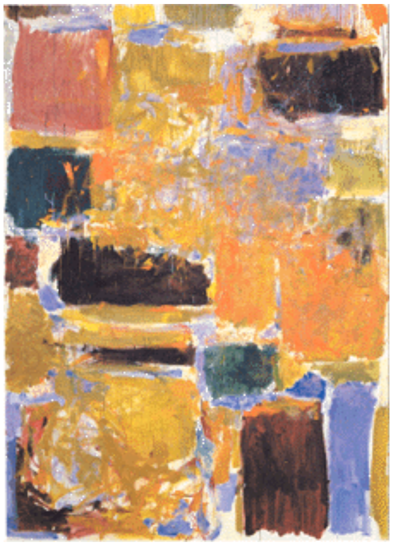


EE141-Spring 2010 Digital Integrated Circuits

Lecture 13
Complex CMOS - Revisited

Administrativia

- ❑ No more re-grading of Midterm after today
- ❑ Hw 5 due on Friday.
- ❑ Hw 6 posted early next week. You get TWO weeks for this one.
- ❑ Project phase 1 introduced today
  - ▪ Actual launch is on Friday
- ❑ Out of town next week
  - ▪ We lecture offered by Stanley
  - ▪ Fr lecture cancelled – Make-up on Tu March 16 at 3:30pm

Class Material

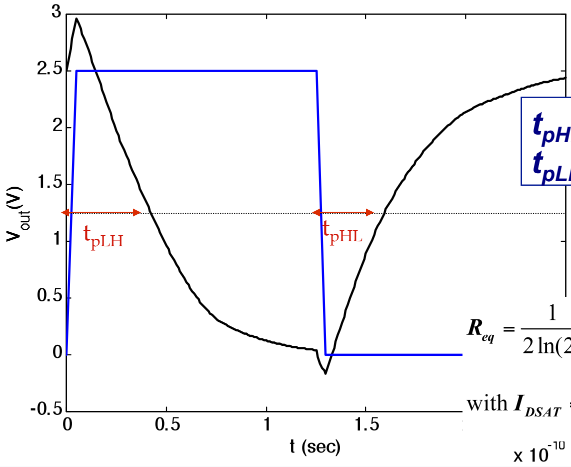
- Last lecture
  - Inverter delay
- Today's lecture
  - Inverter energy
  - Project launch
  - Optimizing complex CMOS
- Reading (Ch 5, 6)

EECS141 Lecture #13 3

Propagation Delay

EECS141 Lecture #13 4

Transient Response



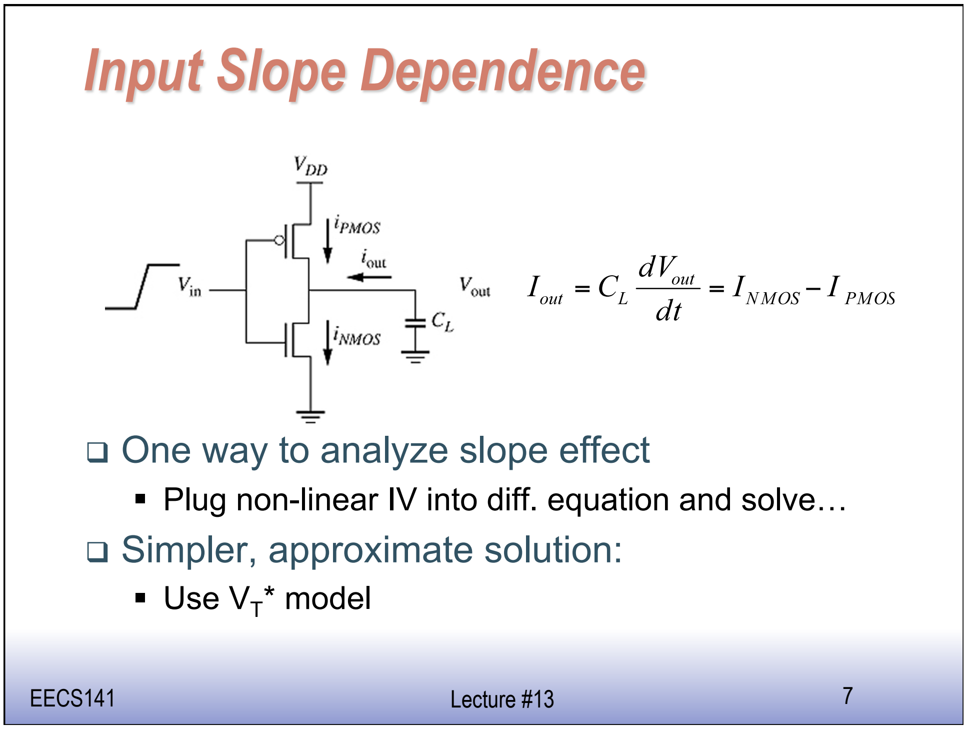
t_{pHL} = 0.69 C_L R_{eqn}
t_{pLH} = 0.69 C_L R_{eqp}

$$R_{eq} = \frac{1}{2\ln(2)} \frac{1}{(1+\lambda V_{DD})} \frac{V_{DD}}{I_{DSAT}}$$

$$\text{with } I_{DSAT} = k'\frac{W}{L}\left(V_{DD} - V_T - \frac{V_{DS,eff}}{2}\right)V_{DS,eff}$$

Step Inputs?

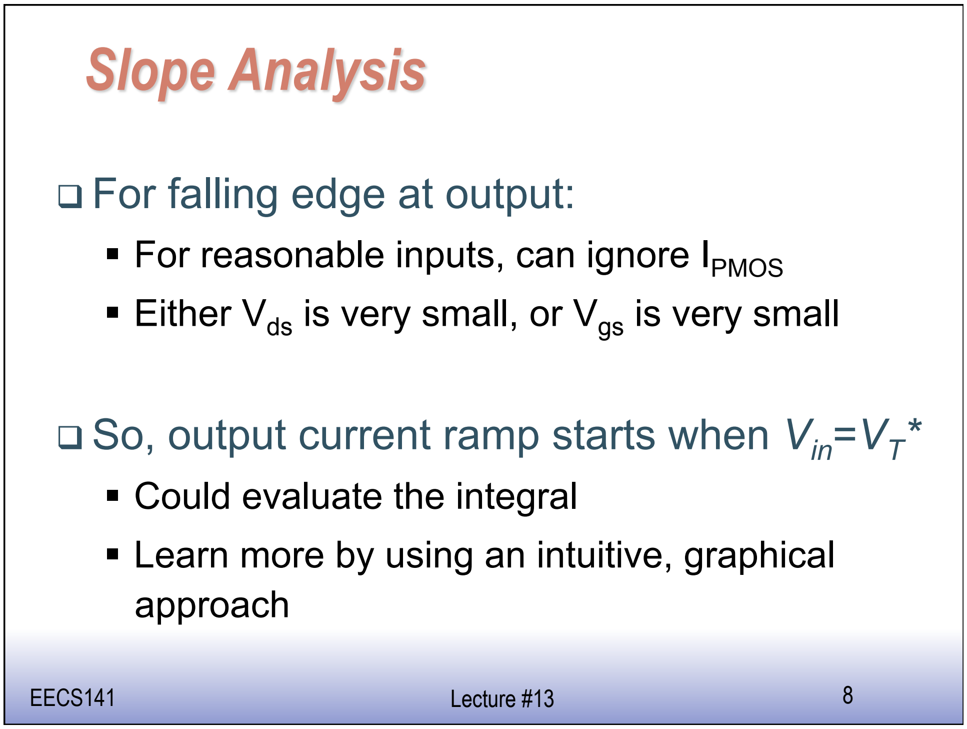
- ❑ Derived RC model assuming input was a step
  - But input is not a step
  - Transistor turns on gradually
- ❑ Let's look at gate switching more carefully
  - Use our models to understand the effect of input slope



Input Slope Depedence

$$I_{out} = C_L \frac{dV_{out}}{dt} = I_{NMOS} - I_{PMOS}$$

- ❑ One way to analyze slope effect
  - ▪ Plug non-linear IV into diff. equation and solve…
- ❑ Simpler, approximate solution:
  - ▪ Use V_T* model



Slope Analysis

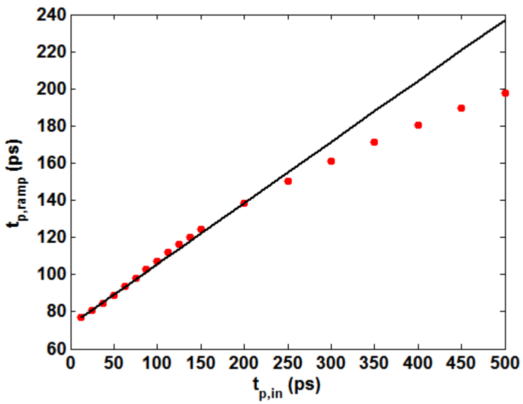
- ❑ For falling edge at output:
  - ▪ For reasonable inputs, can ignore I_{PMOS}
  - ▪ Either V_{ds} is very small, or V_{gs} is very small
- ❑ So, output current ramp starts when V_{in}=V_T*
  - ▪ Could evaluate the integral
  - ▪ Learn more by using an intuitive, graphical approach

Result Summary

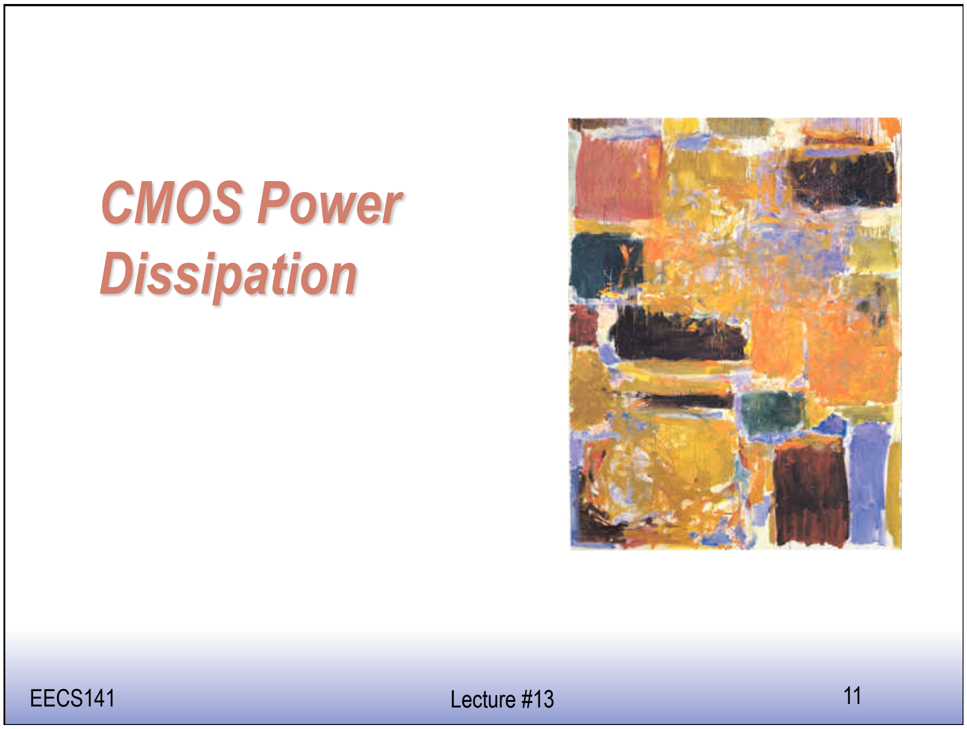
- ❑ For reasonable input slopes:

$$\boxed{t_{p,ramp} = t_{p,step} + \frac{V_T{}^*}{V_{DD}} \cdot t_{p,in}}$$

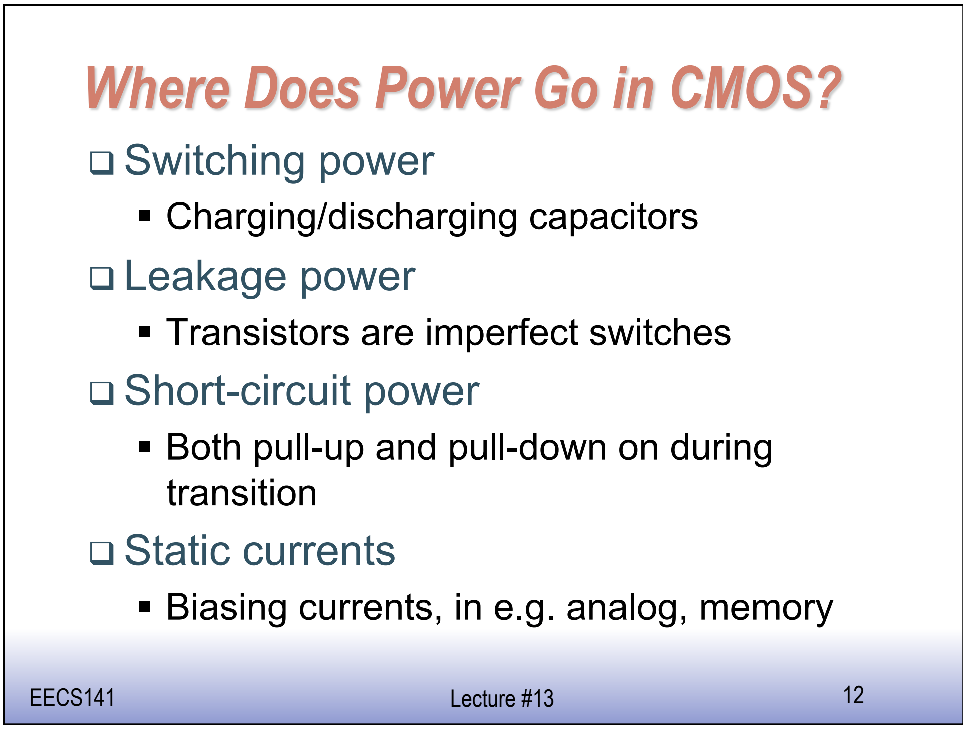
Model vs. Spice Data



- ❑ For reasonable input slope
  - Model matches Spice very well
- ❑ Model breaks with very large t_r
  - Input looks “DC” – traces out VTC
  - Have other problems here anyways
    - – Short-circuit current

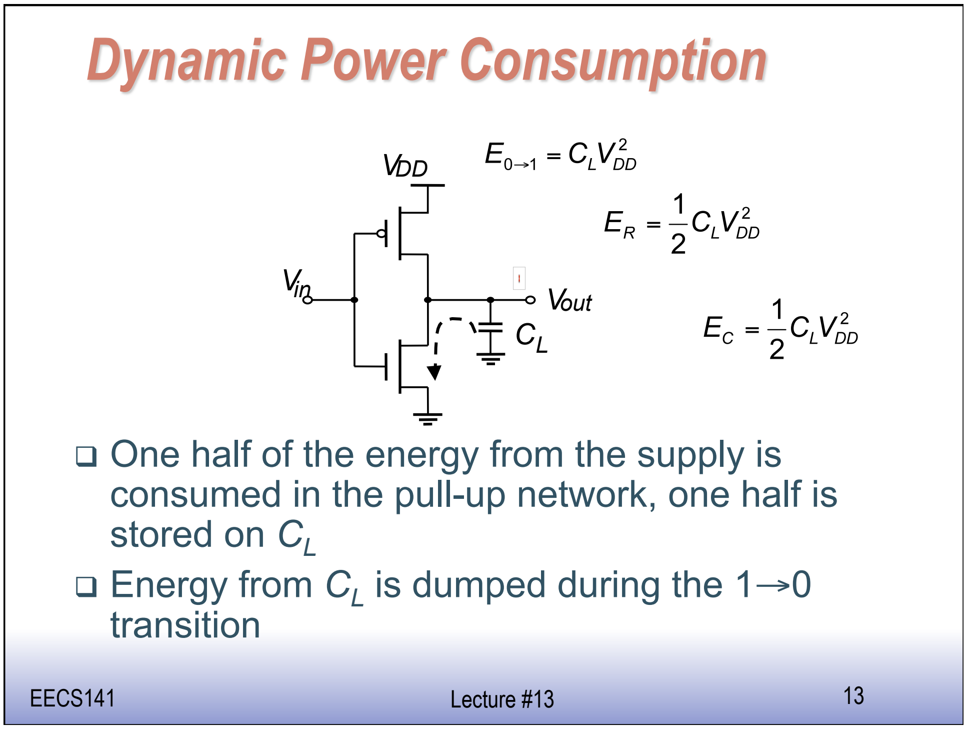


CMOS Power Dissipation



Where Does Power Go in CMOS?

- ❑ Switching power
  - Charging/discharging capacitors
- ❑ Leakage power
  - Transistors are imperfect switches
- ❑ Short-circuit power
  - Both pull-up and pull-down on during transition
- ❑ Static currents
  - Biasing currents, in e.g. analog, memory



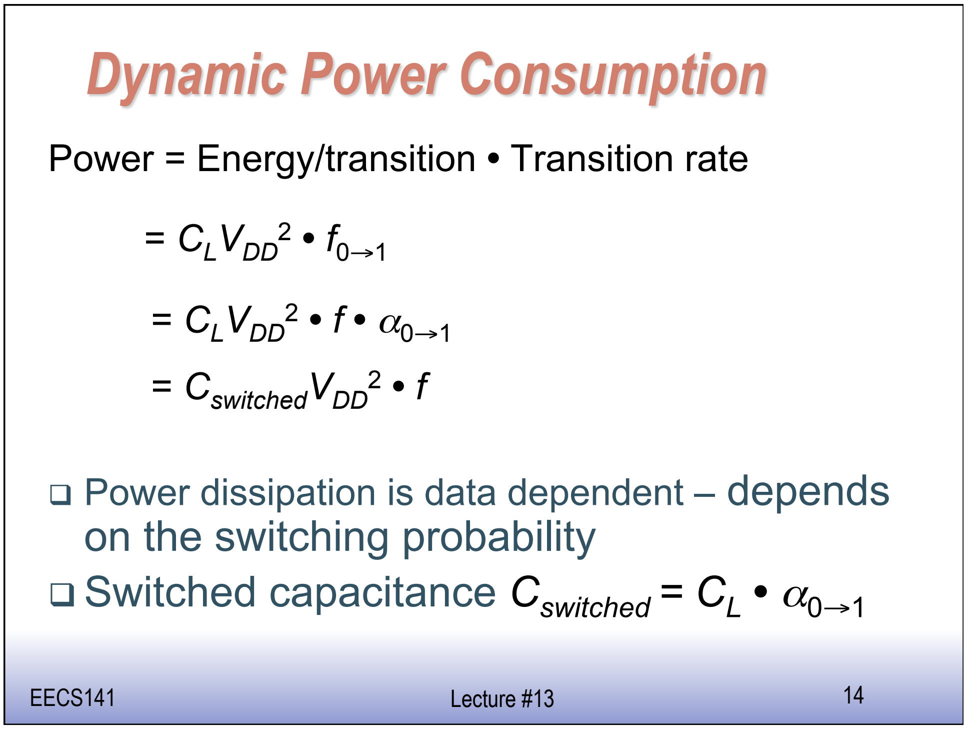
Dynamic Power Consumption

$E_{0\rightarrow1} = C_L V_{DD}^2$

$E_R = \frac{1}{2} C_L V_{DD}^2$

$E_C = \frac{1}{2} C_L V_{DD}^2$

- ❑ One half of the energy from the supply is consumed in the pull-up network, one half is stored on C_L
- ❑ Energy from C_L is dumped during the 1→0 transition



Dynamic Power Consumption

Power = Energy/transition • Transition rate

$$= C_L V_{DD}^2 \bullet f_{0\rightarrow1}$$

$$= C_L V_{DD}^2 \bullet f \bullet \alpha_{0\rightarrow1}$$

$$= C_{switched} V_{DD}^2 \bullet f$$

- ❑ Power dissipation is data dependent – depends on the switching probability
- ❑ Switched capacitance $C_{switched} = C_L \bullet \alpha_{0\rightarrow1}$

Transition Activity and Power

- ❑ Energy consumed in N cycles, E_N:

$$E_N = C_L \bullet V_{DD}^2 \bullet n_{0\rightarrow 1}$$

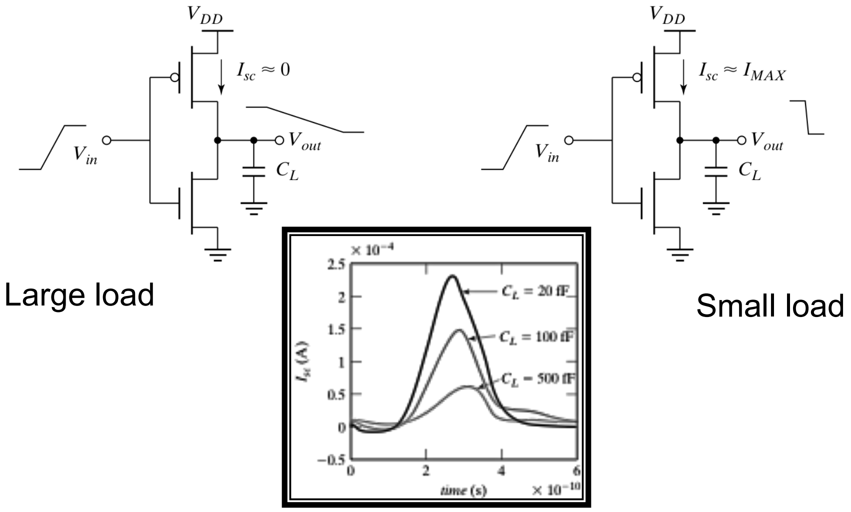
$n_{0\rightarrow 1}$ – number of 0→1 transitions in N cycles

$$P_{avg} = \lim_{N\rightarrow\infty} \frac{E_N}{N} \cdot f = \left(\lim_{N\rightarrow\infty} \frac{n_{0\rightarrow 1}}{N} \right) \cdot C_L \cdot V_{DD}{}^2 \cdot f$$

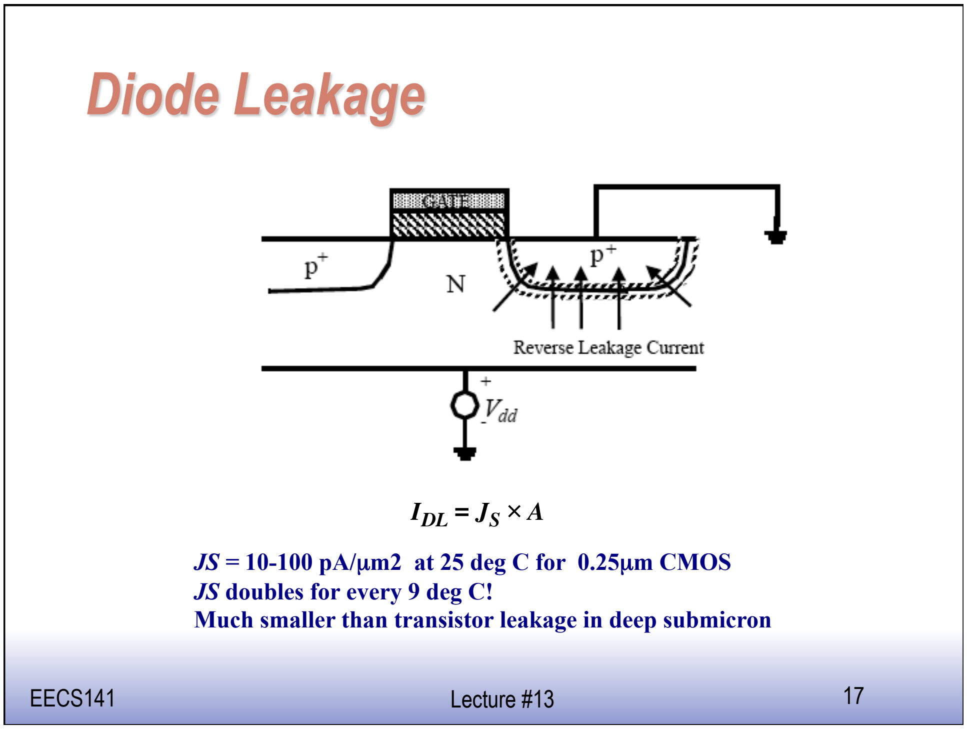
$$\alpha_{0\rightarrow 1} = \lim_{N\rightarrow\infty} \frac{n_{0\rightarrow 1}}{N}$$

$$\boxed{P_{avg} = \alpha_{0\rightarrow 1} \cdot C_L \cdot V_{DD}{}^2 \cdot f}$$

Short Circuit Current



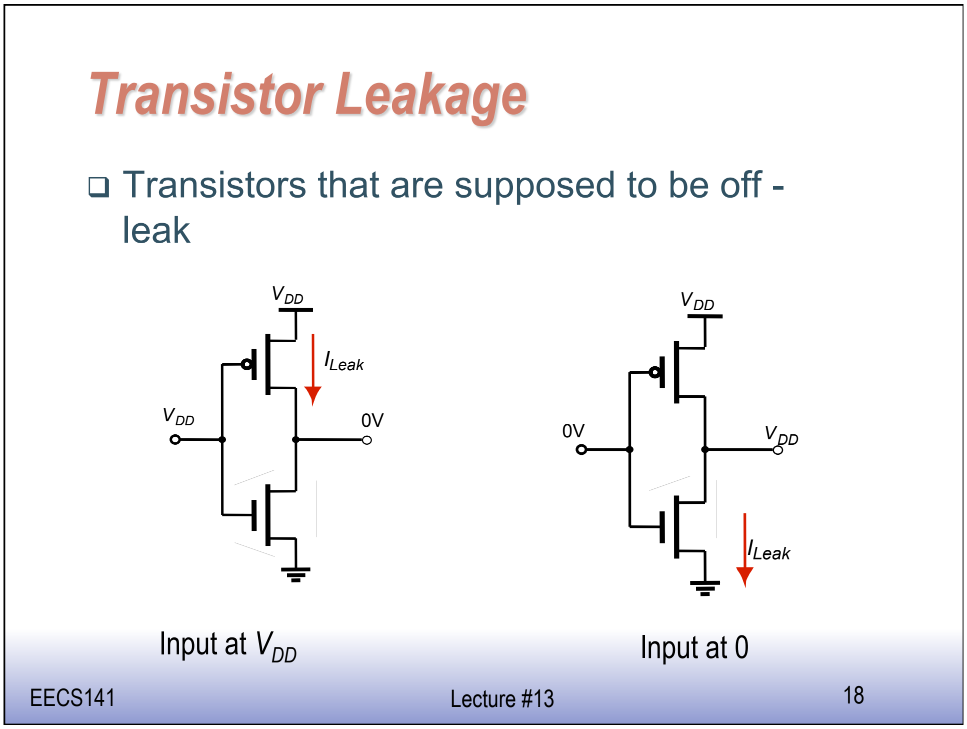
- ❑ Short circuit current usually well controlled



Diode Leakage

$$I_{DL} = J_S \times A$$

JS = 10-100 pA/µm2 at 25 deg C for 0.25µm CMOS
JS doubles for every 9 deg C!
Much smaller than transistor leakage in deep submicron

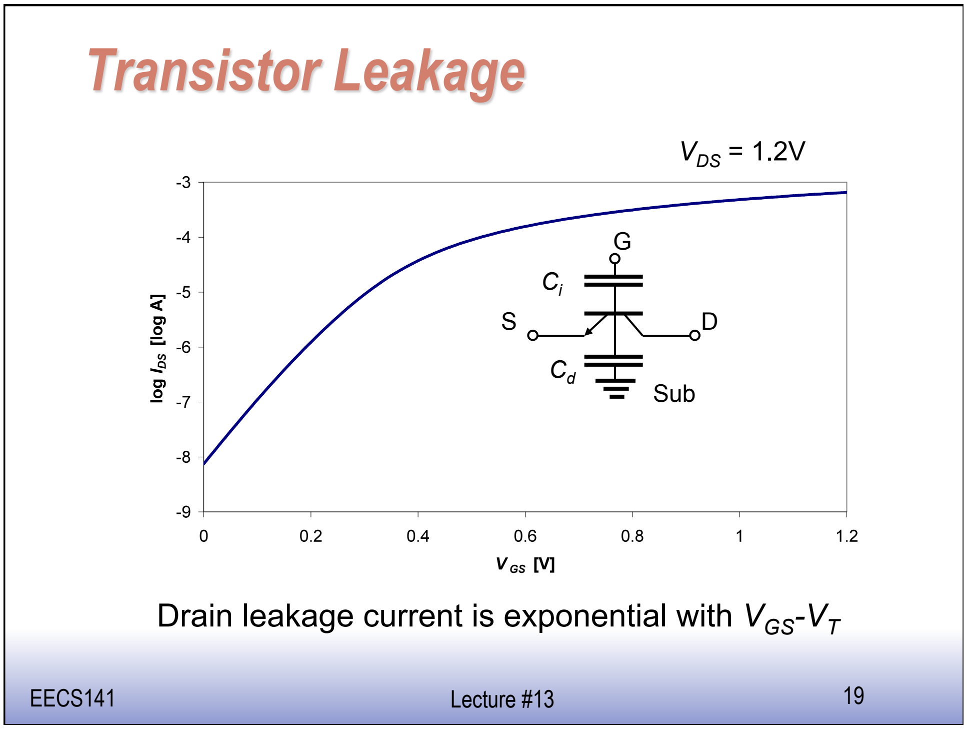


Transistor Leakage

- ❑ Transistors that are supposed to be off - leak

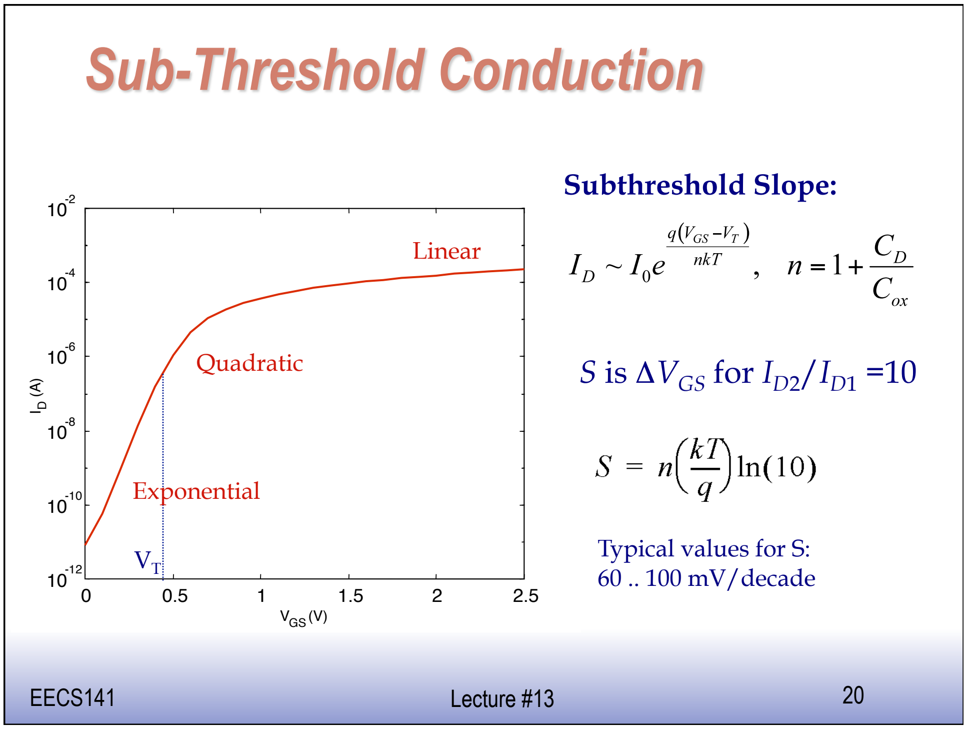
Input at V_{DD}

Input at 0



Transistor Leakage

Drain leakage current is exponential with V_{GS}-V_T



Sub-Threshold Conduction

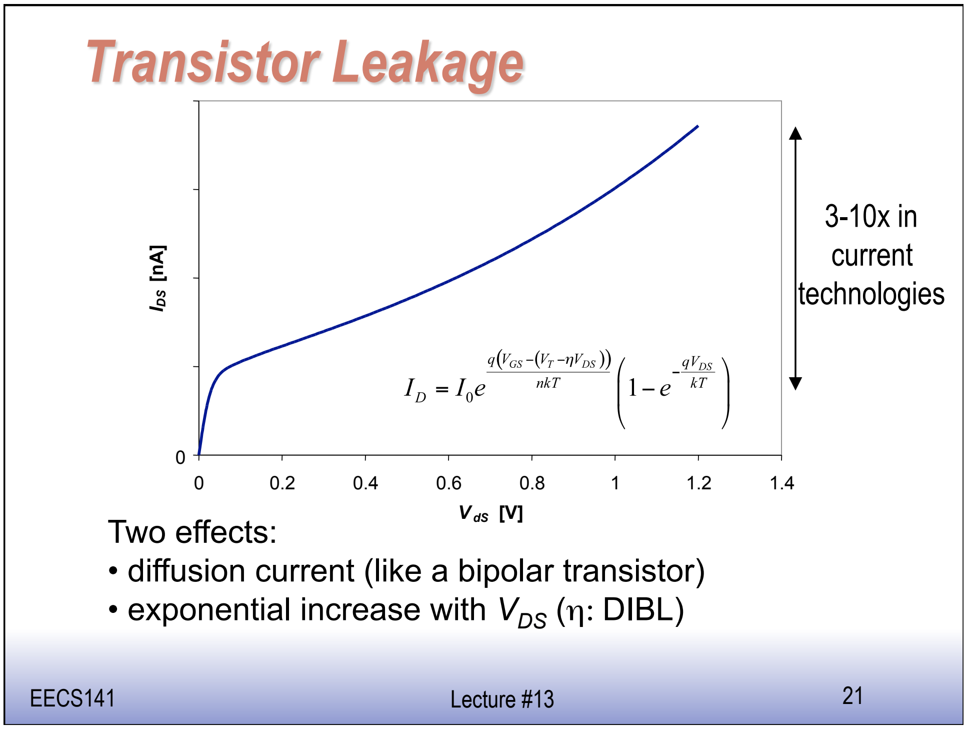
Subthreshold Slope:

$$I_D \sim I_0 e^{\frac{q(V_{GS}-V_T)}{nkT}}, \quad n = 1 + \frac{C_D}{C_{ox}}$$

S is ΔV_{GS} for I_{D2}/I_{D1} =10

$$S = n\left(\frac{kT}{q}\right)\ln(10)$$

Typical values for S:
60 .. 100 mV/decade



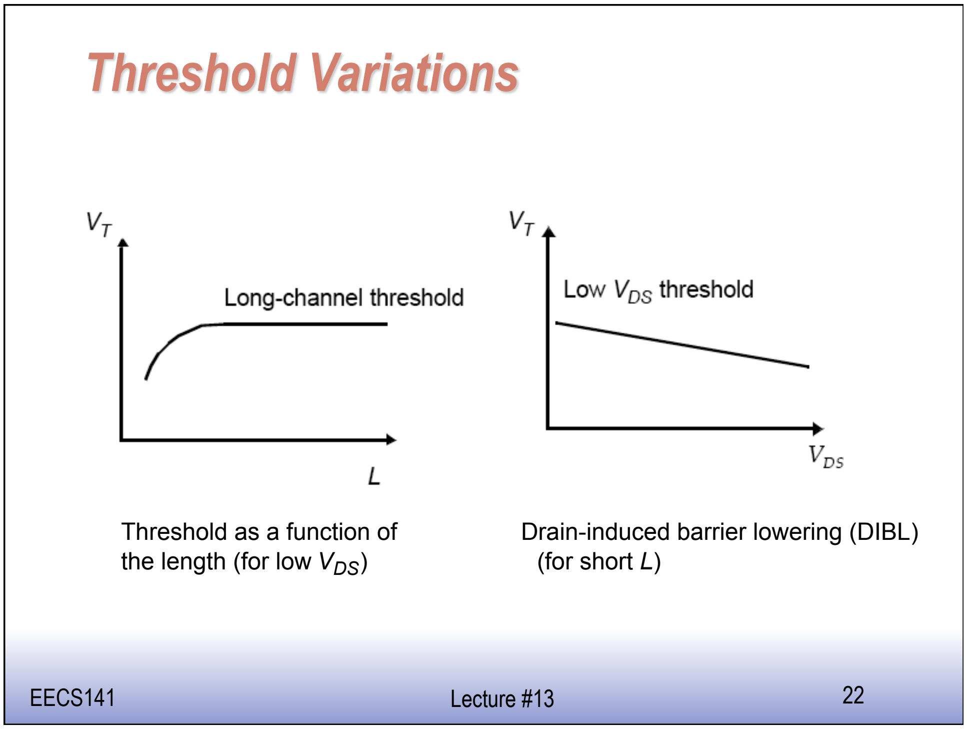
Transistor Leakage

$$I_D = I_0 e^{\frac{q(V_{GS}-(V_T-\eta V_{DS}))}{nkT}}\left(1-e^{-\frac{qV_{DS}}{kT}}\right)$$

3-10x in current technologies

Two effects:

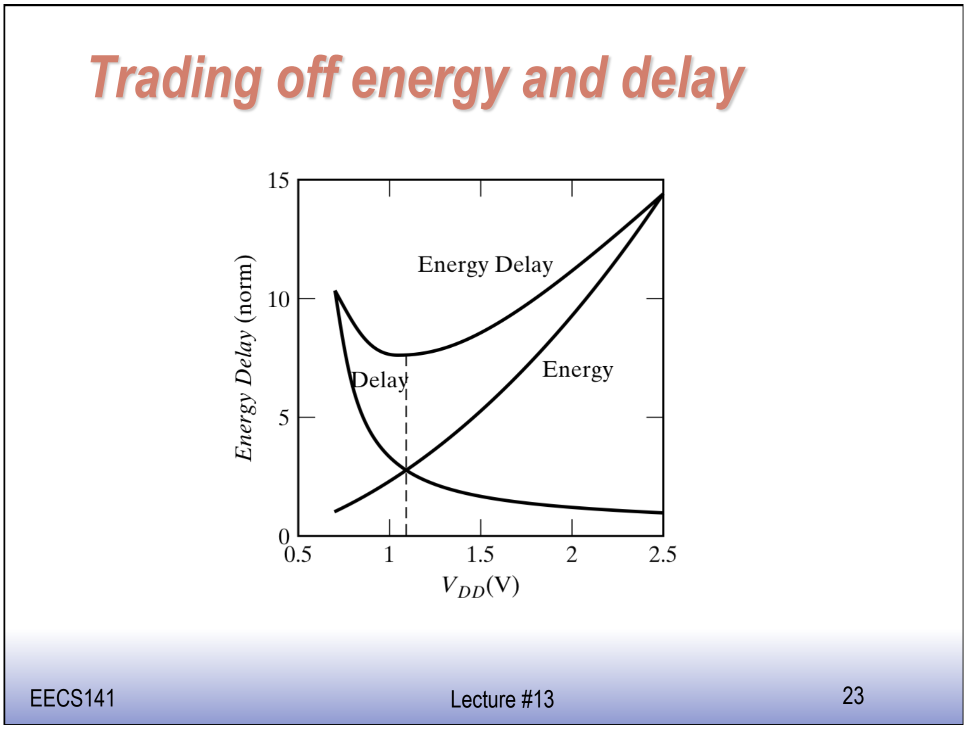
- diffusion current (like a bipolar transistor)
- exponential increase with V_{DS} (η: DIBL)



Threshold Variations

Threshold as a function of the length (for low V_{DS})

Drain-induced barrier lowering (DIBL) (for short L)



Trading off energy and delay

EECS141 Lecture #13 23

Intermezzo

EECS141 Lecture #13 24

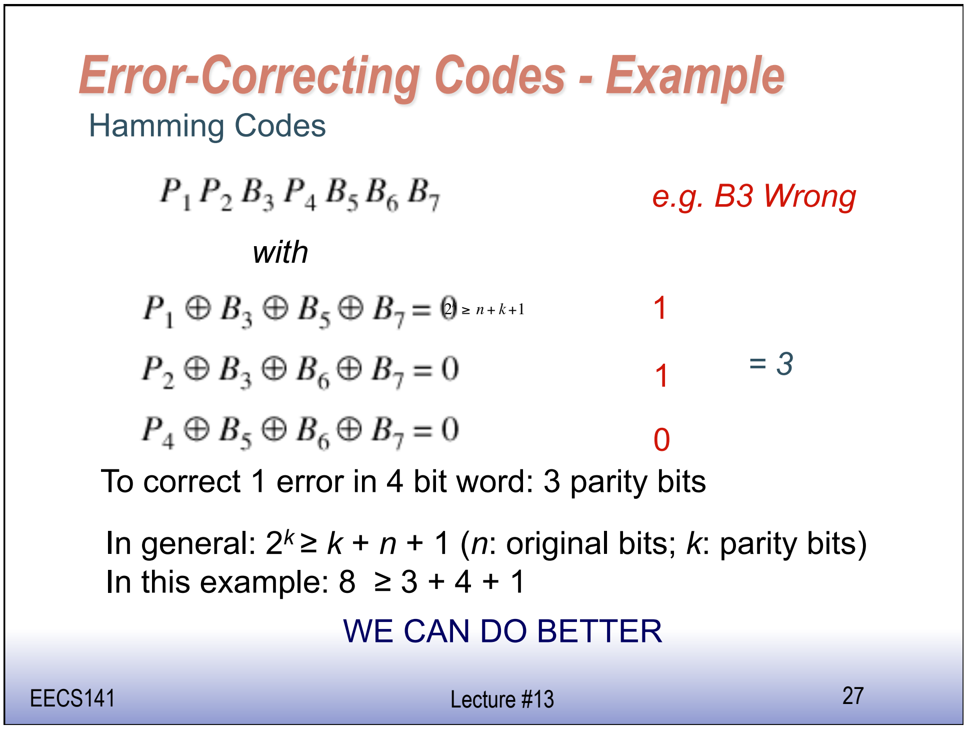
Error-Correcting Codes

- ❑ Data transmissions, computations or storage may often fail
  - Because of yield issues
  - Noise
  - Interference
- ❑ Often used in wireless transmissions, hard disks, memory
- ❑ Redundancy can help to eliminate the impact of these errors
  - Triple-modular redundancy
  - Coding – add extra bits so that errors can be detected and corrected

Richard Wesley Hamming

- ❑ In telecommunication, a Hamming code is a linear error-correcting code named after its inventor, Richard Hamming. Hamming codes can detect up to two simultaneous bit errors, and correct single-bit errors; thus, reliable communication is possible when the Hamming distance between the transmitted and received bit patterns is less than or equal to one. By contrast, the simple parity code cannot correct errors, and can only detect an odd number of errors.
- ❑ Because of the simplicity of Hamming codes, they are widely used in computer memory (RAM). In particular, a single-error-correcting and double-error-detecting variant commonly referred to as SECDED.

R. Hamming, 1915-1998



Error-Correcting Codes - Example

Hamming Codes

$P_1 P_2 B_3 P_4 B_5 B_6 B_7$

e.g. B3 Wrong

with

$P_1 \oplus B_3 \oplus B_5 \oplus B_7 = 0$ $2^k \geq n+k+1$ 1

$P_2 \oplus B_3 \oplus B_6 \oplus B_7 = 0$ 1 *= 3*

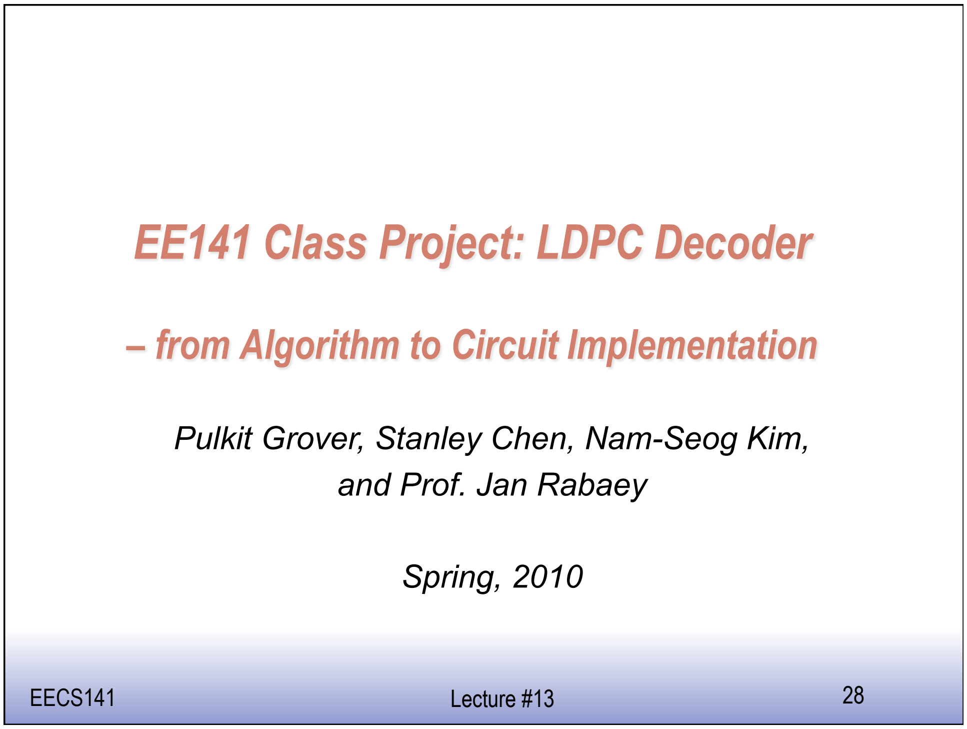
$P_4 \oplus B_5 \oplus B_6 \oplus B_7 = 0$ 0

To correct 1 error in 4 bit word: 3 parity bits

In general: $2^k \geq k + n + 1$ (n: original bits; k: parity bits)
In this example: 8 ≥ 3 + 4 + 1

WE CAN DO BETTER

EECS141 Lecture #13 27



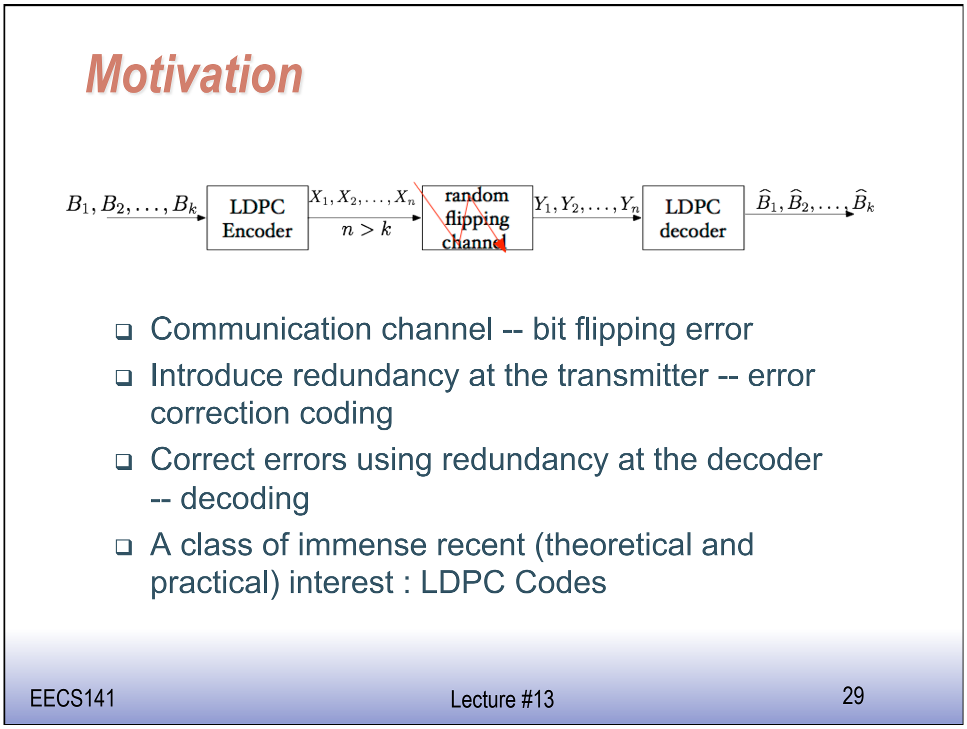
EE141 Class Project: LDPC Decoder

– from Algorithm to Circuit Implementation

Pulkit Grover, Stanley Chen, Nam-Seog Kim,
and Prof. Jan Rabaey

Spring, 2010

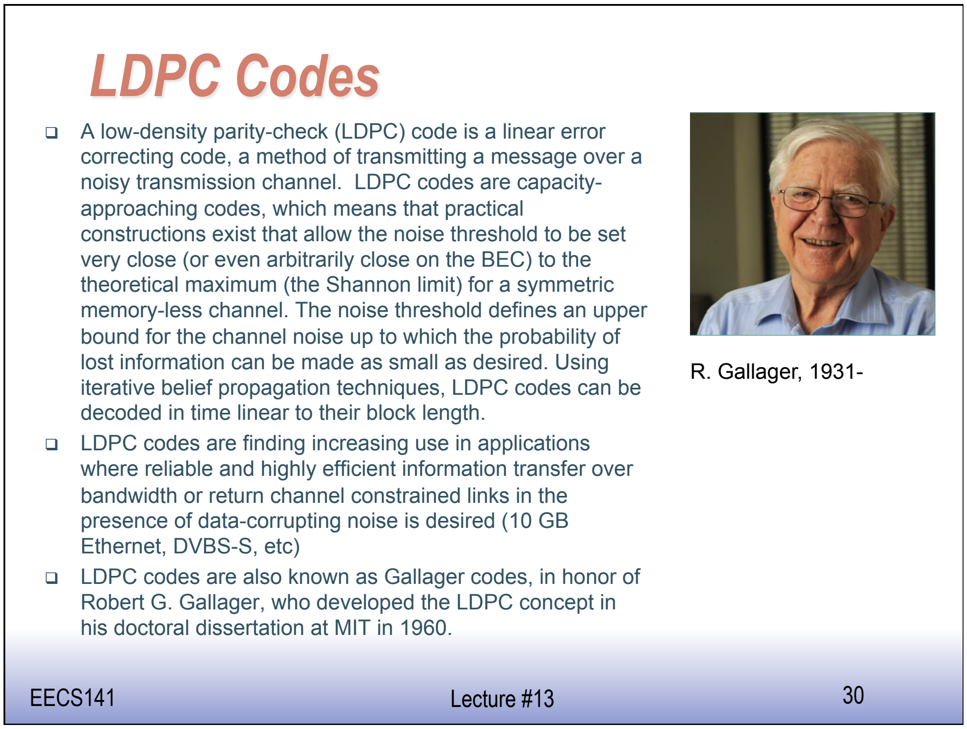
EECS141 Lecture #13 28



Motivation

$B_1, B_2, \ldots, B_k$ → LDPC Encoder → $X_1, X_2, \ldots, X_n$, $n > k$ → random flipping channel → $Y_1, Y_2, \ldots, Y_n$ → LDPC decoder → $\widehat{B}_1, \widehat{B}_2, \ldots, \widehat{B}_k$

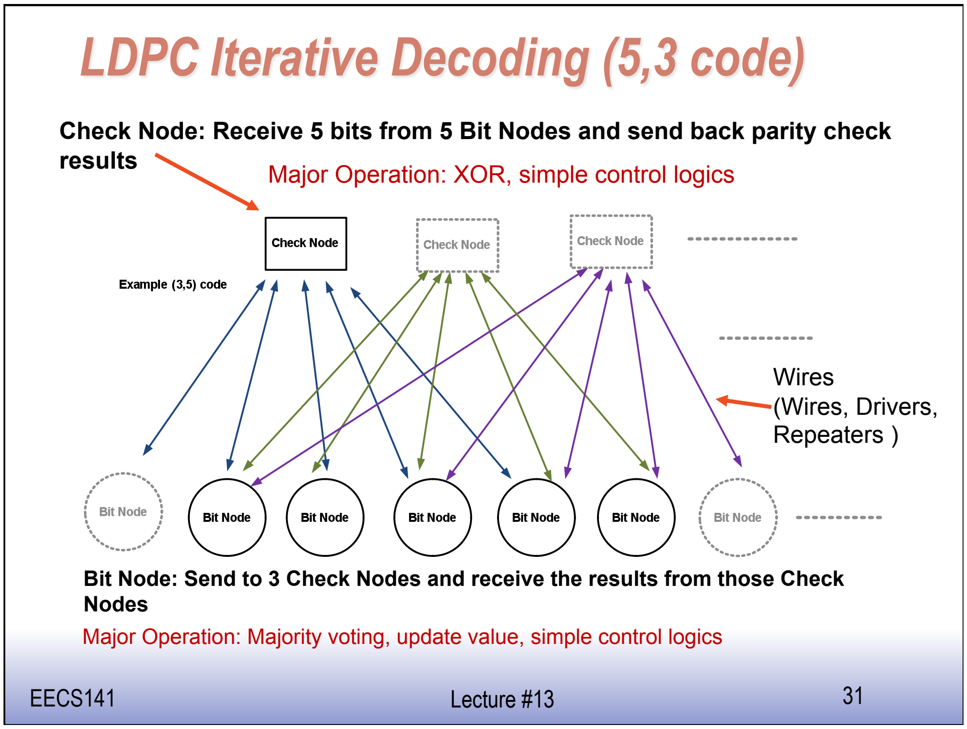
- Communication channel -- bit flipping error
- Introduce redundancy at the transmitter -- error correction coding
- Correct errors using redundancy at the decoder -- decoding
- A class of immense recent (theoretical and practical) interest : LDPC Codes



LDPC Codes

- A low-density parity-check (LDPC) code is a linear error correcting code, a method of transmitting a message over a noisy transmission channel. LDPC codes are capacity-approaching codes, which means that practical constructions exist that allow the noise threshold to be set very close (or even arbitrarily close on the BEC) to the theoretical maximum (the Shannon limit) for a symmetric memory-less channel. The noise threshold defines an upper bound for the channel noise up to which the probability of lost information can be made as small as desired. Using iterative belief propagation techniques, LDPC codes can be decoded in time linear to their block length.
- LDPC codes are finding increasing use in applications where reliable and highly efficient information transfer over bandwidth or return channel constrained links in the presence of data-corrupting noise is desired (10 GB Ethernet, DVBS-S, etc)
- LDPC codes are also known as Gallager codes, in honor of Robert G. Gallager, who developed the LDPC concept in his doctoral dissertation at MIT in 1960.

R. Gallager, 1931-



LDPC Iterative Decoding (5,3 code)

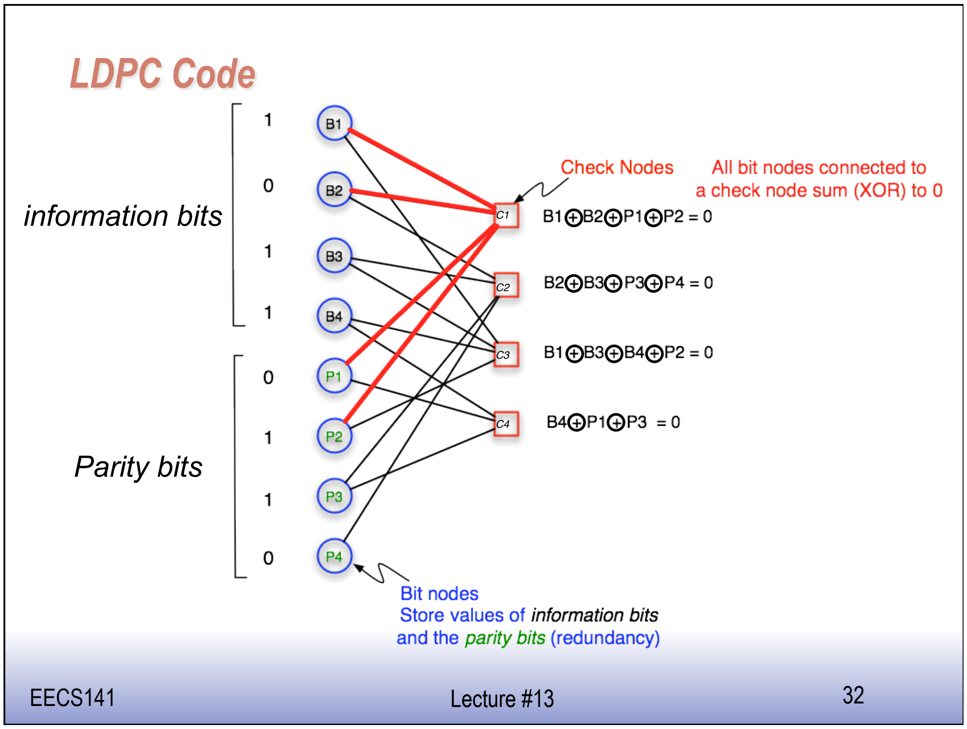
Check Node: Receive 5 bits from 5 Bit Nodes and send back parity check results

Major Operation: XOR, simple control logics

Wires (Wires, Drivers, Repeaters)

Bit Node: Send to 3 Check Nodes and receive the results from those Check Nodes

Major Operation: Majority voting, update value, simple control logics



LDPC Code

information bits

Parity bits

Check Nodes

All bit nodes connected to a check node sum (XOR) to 0

$B1 \oplus B2 \oplus P1 \oplus P2 = 0$

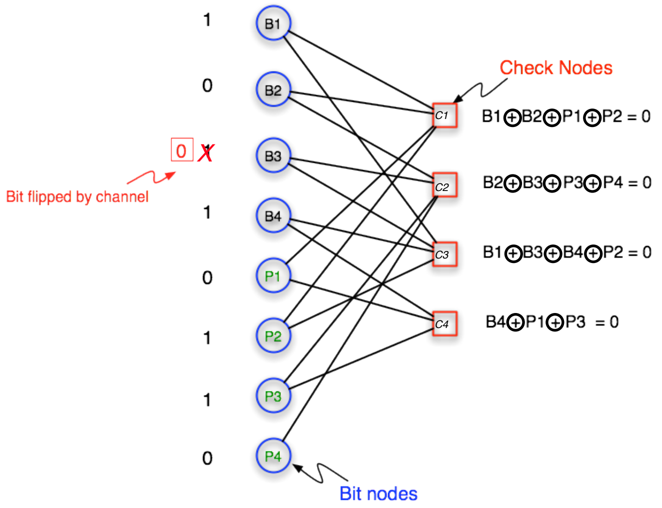
$B2 \oplus B3 \oplus P3 \oplus P4 = 0$

$B1 \oplus B3 \oplus B4 \oplus P2 = 0$

$B4 \oplus P1 \oplus P3 = 0$

Bit nodes
Store values of *information bits* and the *parity bits* (redundancy)

LDPC Decoding



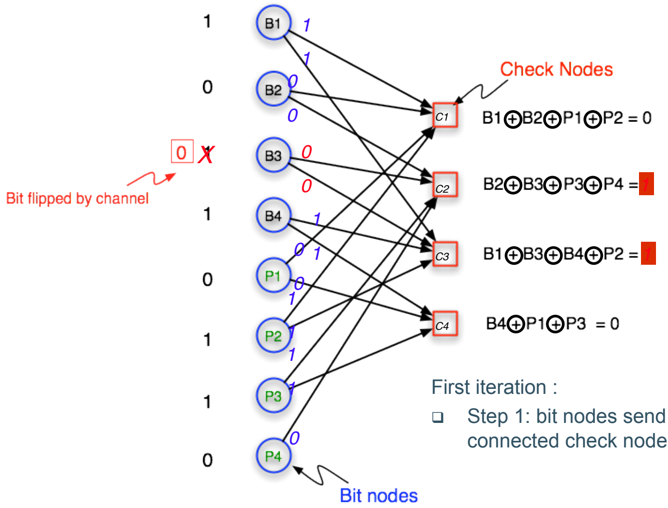
B1⊕B2⊕P1⊕P2 = 0

B2⊕B3⊕P3⊕P4 = 0

B1⊕B3⊕B4⊕P2 = 0

B4⊕P1⊕P3 = 0

LDPC Decoding : First iteration



B1⊕B2⊕P1⊕P2 = 0

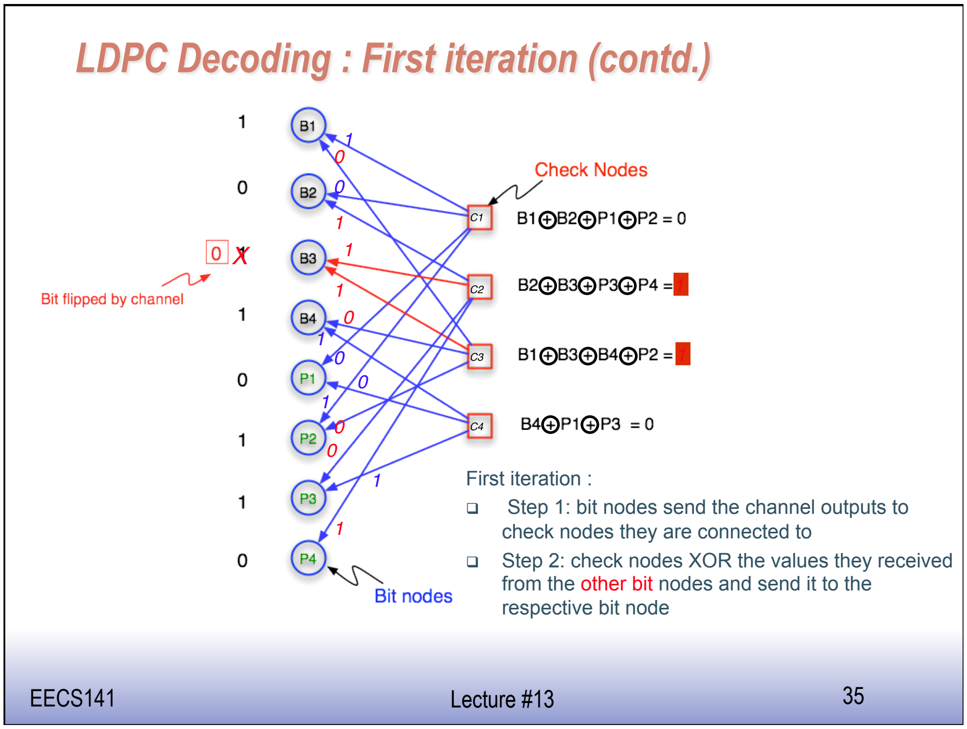
B2⊕B3⊕P3⊕P4 =🟥

B1⊕B3⊕B4⊕P2 =🟥

B4⊕P1⊕P3 = 0

First iteration :

- ❑ Step 1: bit nodes send the channel outputs to the connected check nodes



LDPC Decoding : First iteration (contd.)

B1⊕B2⊕P1⊕P2 = 0

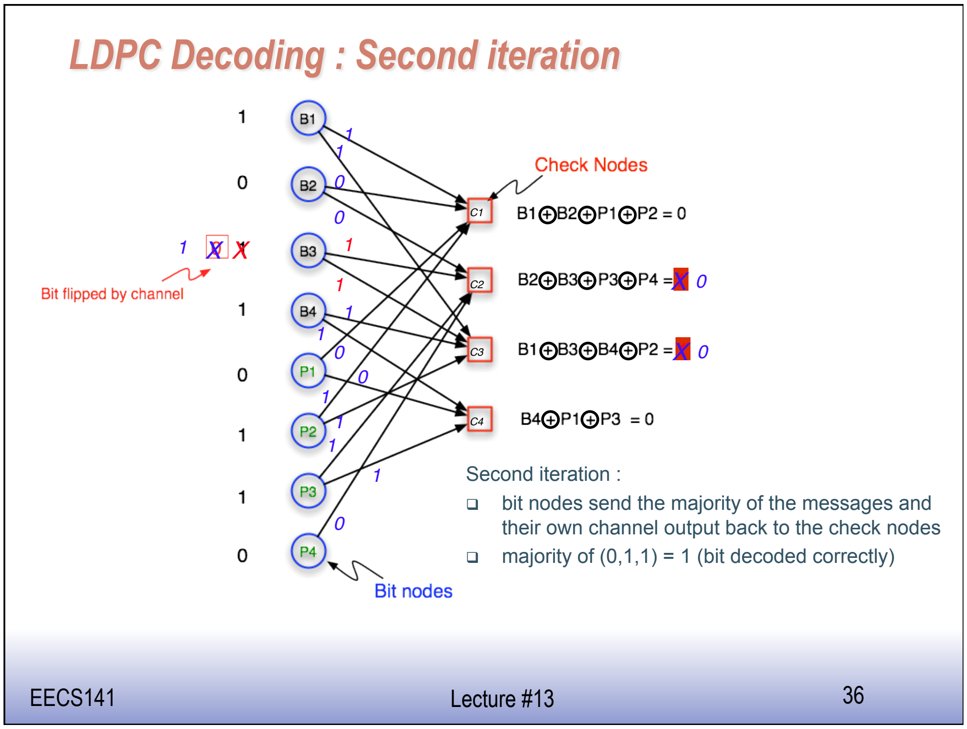
B2⊕B3⊕P3⊕P4 = 🟥

B1⊕B3⊕B4⊕P2 = 🟥

B4⊕P1⊕P3 = 0

First iteration :

- Step 1: bit nodes send the channel outputs to check nodes they are connected to
- Step 2: check nodes XOR the values they received from the other bit nodes and send it to the respective bit node



LDPC Decoding : Second iteration

B1⊕B2⊕P1⊕P2 = 0

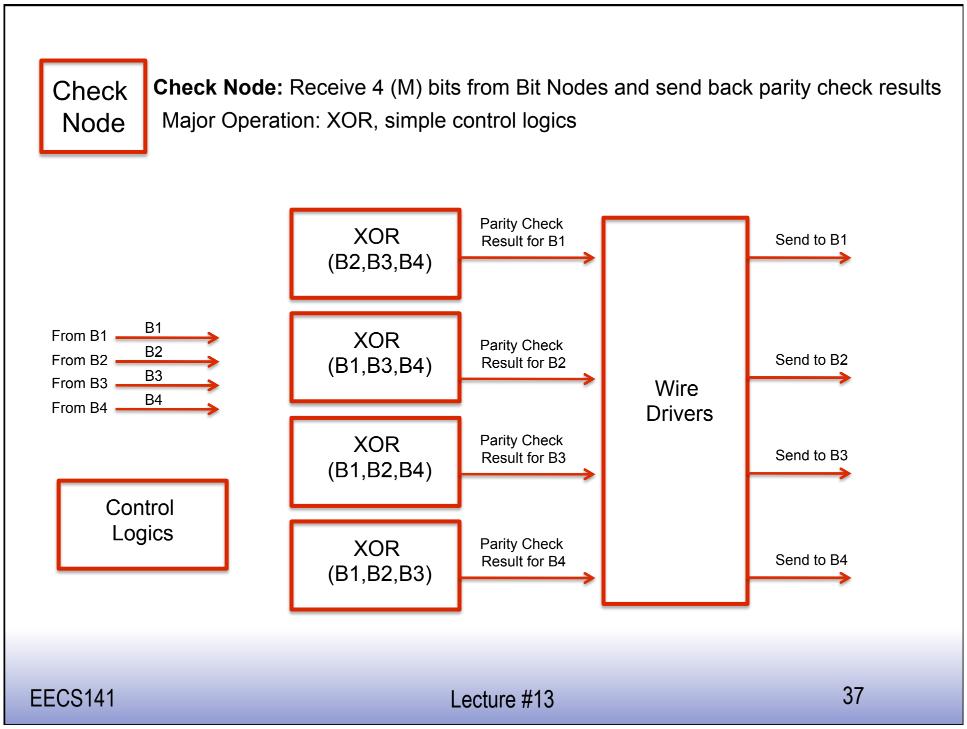
B2⊕B3⊕P3⊕P4 = 0

B1⊕B3⊕B4⊕P2 = 0

B4⊕P1⊕P3 = 0

Second iteration :

- bit nodes send the majority of the messages and their own channel output back to the check nodes
- majority of (0,1,1) = 1 (bit decoded correctly)



Check Node

Check Node: Receive 4 (M) bits from Bit Nodes and send back parity check results
Major Operation: XOR, simple control logics

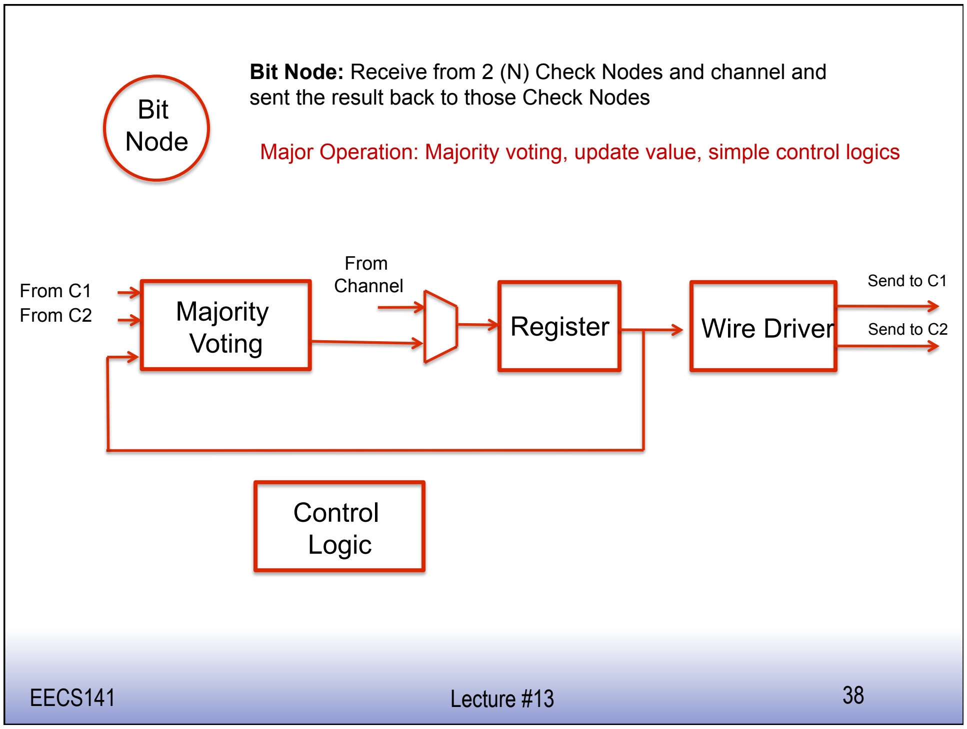
From B1 B1
From B2 B2
From B3 B3
From B4 B4

Control Logics

XOR (B2,B3,B4) Parity Check Result for B1
XOR (B1,B3,B4) Parity Check Result for B2
XOR (B1,B2,B4) Parity Check Result for B3
XOR (B1,B2,B3) Parity Check Result for B4

Wire Drivers

Send to B1
Send to B2
Send to B3
Send to B4



Bit Node

Bit Node: Receive from 2 (N) Check Nodes and channel and sent the result back to those Check Nodes

Major Operation: Majority voting, update value, simple control logics

From C1
From C2

Majority Voting

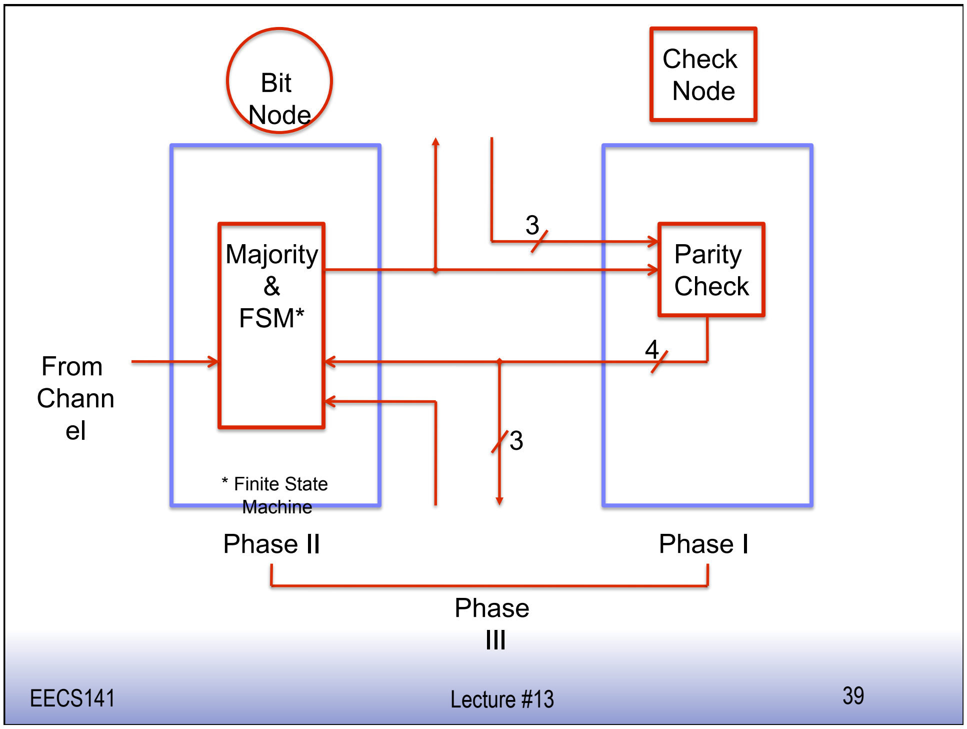
From Channel

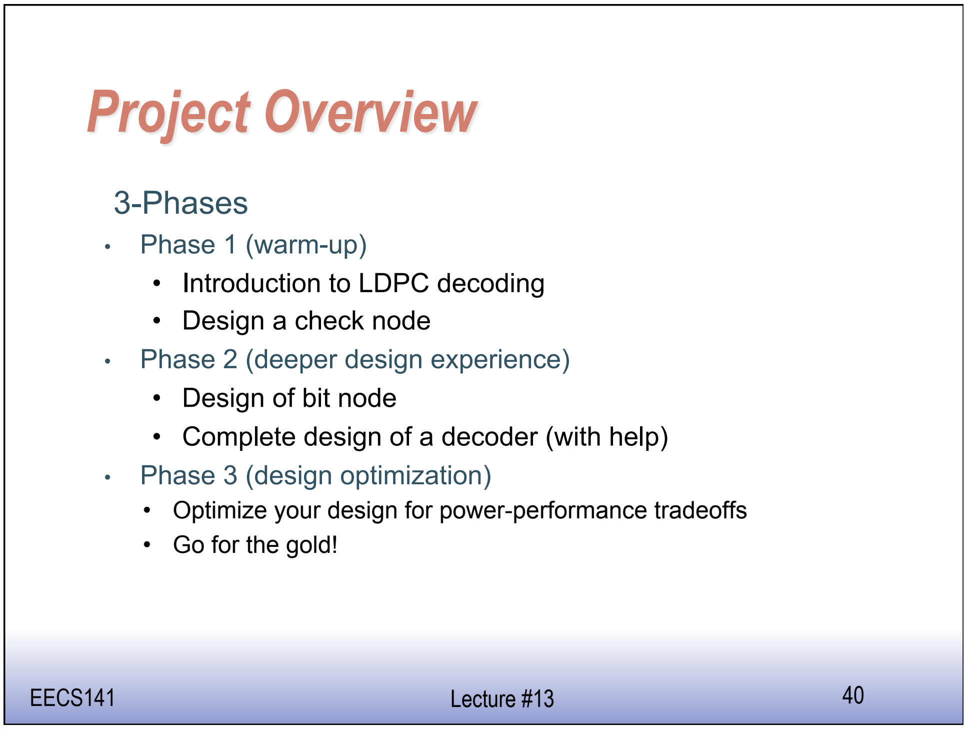
Register

Wire Driver

Send to C1
Send to C2

Control Logic





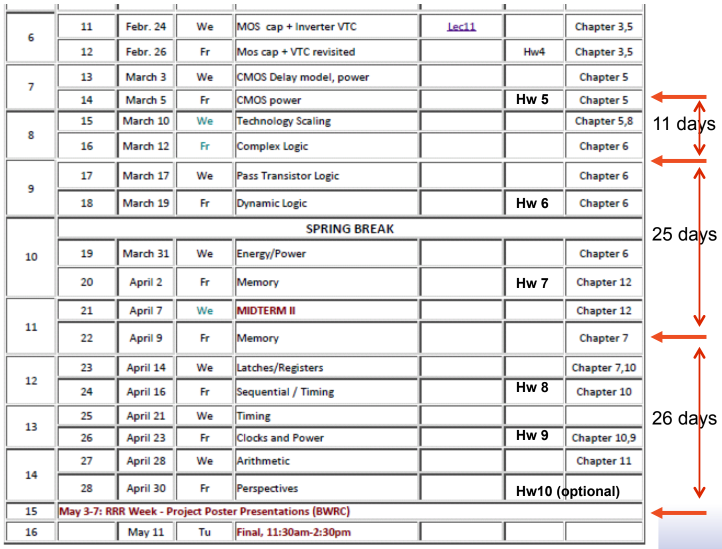
Project Overview

3-Phases

- Phase 1 (warm-up)
  - Introduction to LDPC decoding
  - Design a check node
- Phase 2 (deeper design experience)
  - Design of bit node
  - Complete design of a decoder (with help)
- Phase 3 (design optimization)
  - Optimize your design for power-performance tradeoffs
  - Go for the gold!

Time Line

Prj-ph1: 03/05~03/15, 11 days
Prj-ph2: 03/15~04/09, 25 days
Prj-ph3: 04/09~05/05, 26 days



6	11	Febr. 24	We	MOS cap + Inverter VTC	Lec11		Chapter 3,5
	12	Febr. 26	Fr	Mos cap + VTC revisited		Hw4	Chapter 3,5
7	13	March 3	We	CMOS Delay model, power			Chapter 5
	14	March 5	Fr	CMOS power		Hw 5	Chapter 5
8	15	March 10	We	Technology Scaling			Chapter 5,8
	16	March 12	Fr	Complex Logic			Chapter 6
9	17	March 17	We	Pass Transistor Logic			Chapter 6
	18	March 19	Fr	Dynamic Logic		Hw 6	Chapter 6
10	SPRING BREAK						
	19	March 31	We	Energy/Power			Chapter 6
	20	April 2	Fr	Memory		Hw 7	Chapter 12
11	21	April 7	We	MIDTERM II			Chapter 12
	22	April 9	Fr	Memory			Chapter 7
12	23	April 14	We	Latches/Registers			Chapter 7,10
	24	April 16	Fr	Sequential / Timing		Hw 8	Chapter 10
13	25	April 21	We	Timing			
	26	April 23	Fr	Clocks and Power		Hw 9	Chapter 10,9
14	27	April 28	We	Arithmetic			Chapter 11
	28	April 30	Fr	Perspectives		Hw10 (optional)	
15	May 3-7: RRR Week - Project Poster Presentations (BWRC)						
16		May 11	Tu	Final, 11:30am-2:30pm			

11 days

25 days

26 days

Prj-ph1 launched
(Friday Mar. 5)

Prj-ph1 due
(Monday Mar. 15)
& Prj-ph2 launched

Prj-ph2 due
&Prj-ph3 launched
(Friday Apr. 9)

Poster Session
(Wednesday May 5)

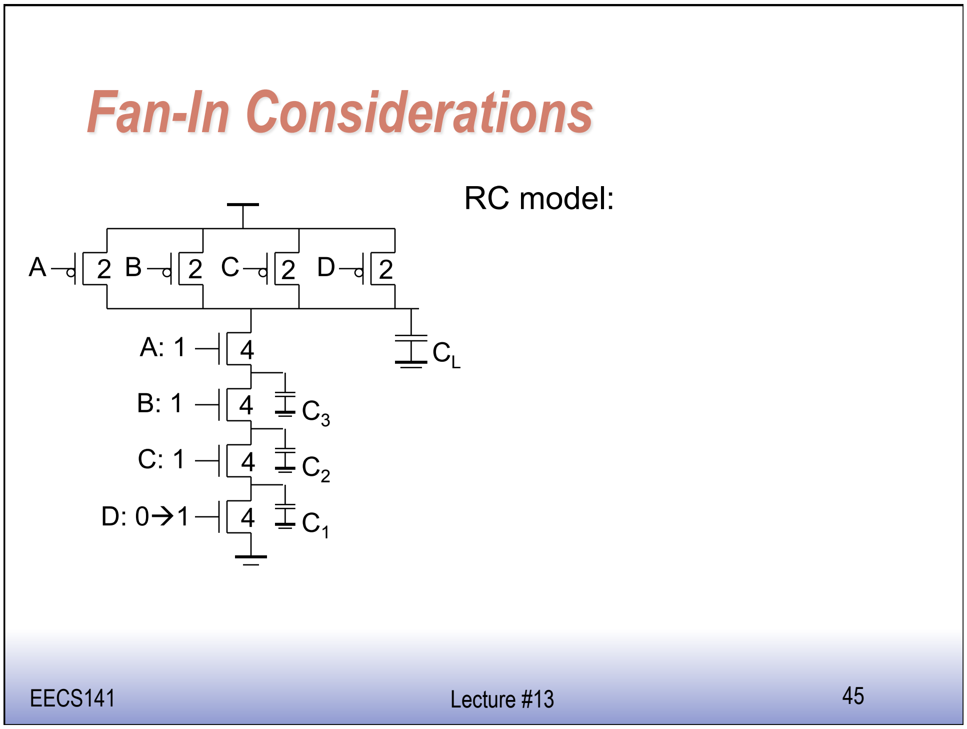
Project Phase 1 - Tasks

- ❑ Do an exercise on LDPC decoding
  - Given an input vector + decoding procedures, derive the final results step by step
- ❑ Derive the schematic for Check Node
- ❑ Design and size logic gates (w/ preset wire loading) for delay
- ❑ Simulate the design in Cadence SPECTRE

CMOS Logic Revisited

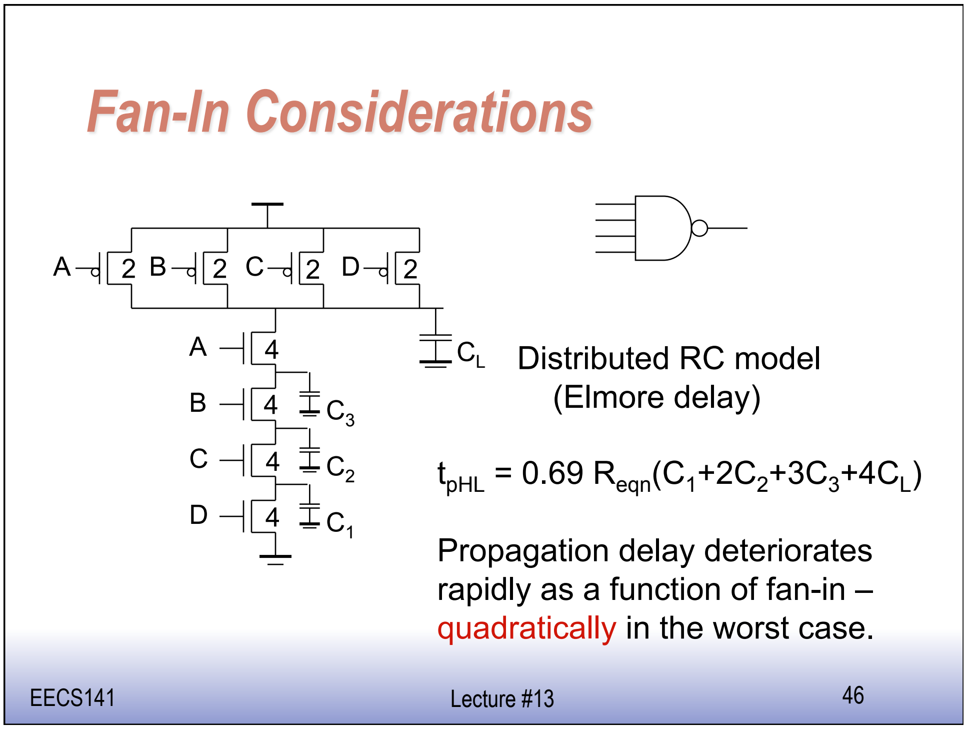
Analyzing and Optimizing Complex CMOS Gates

- ❑ Techniques very similar to the inverter case
- ❑ Logical Effort technique as the means for gate sizing and topology optimization
- ❑ However … some other things to be aware of



Fan-In Considerations

RC model:

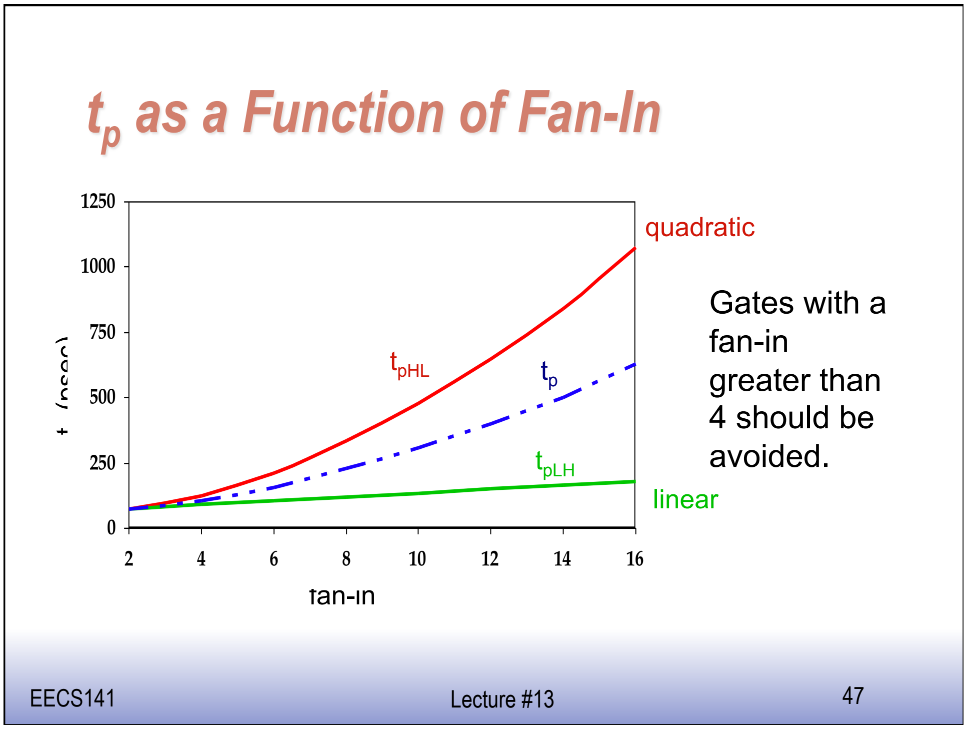


Fan-In Considerations

Distributed RC model
(Elmore delay)

$$t_{pHL} = 0.69\ R_{eqn}(C_1+2C_2+3C_3+4C_L)$$

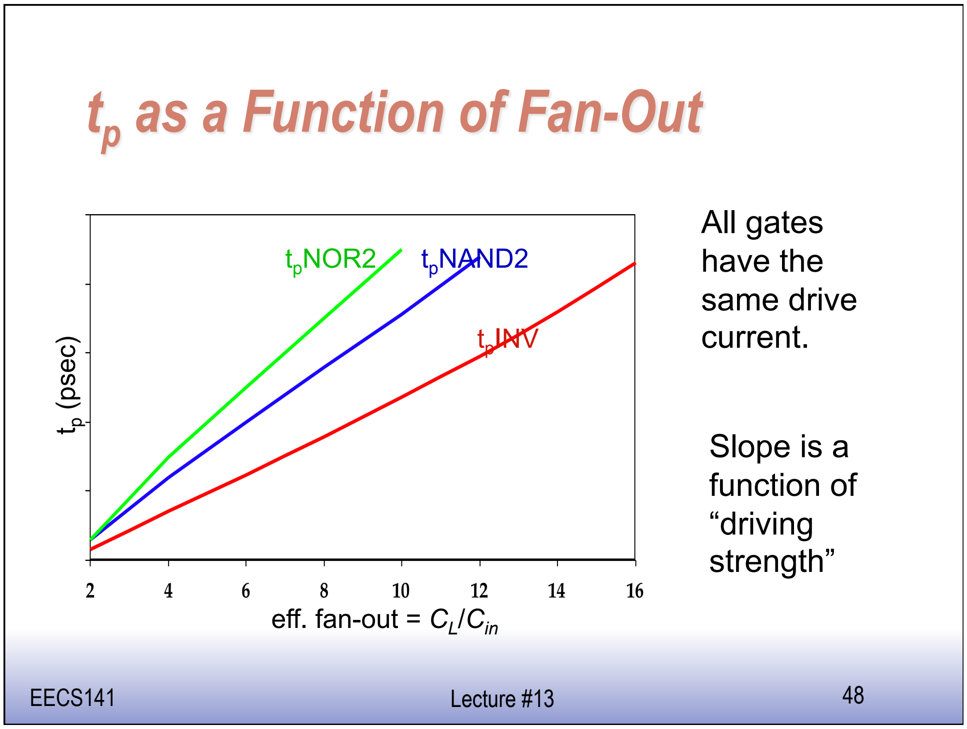
Propagation delay deteriorates rapidly as a function of fan-in – quadratically in the worst case.



t_p as a Function of Fan-In

Gates with a fan-in greater than 4 should be avoided.

EECS141 Lecture #13 47



t_p as a Function of Fan-Out

All gates have the same drive current.

Slope is a function of "driving strength"

EECS141 Lecture #13 48

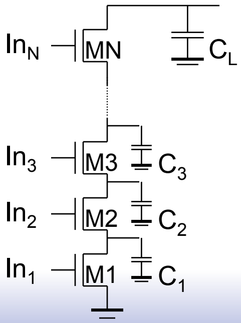
t_p as a Function of Fan-In and Fan-Out

- ❑ Fan-in: quadratic due to increasing resistance and capacitance
- ❑ Fan-out: each additional fan-out gate adds two gate capacitances to C_L

$$t_p = a_1FI + a_2FI^2 + a_3FO$$

Fast Complex Gates: Design Technique 1

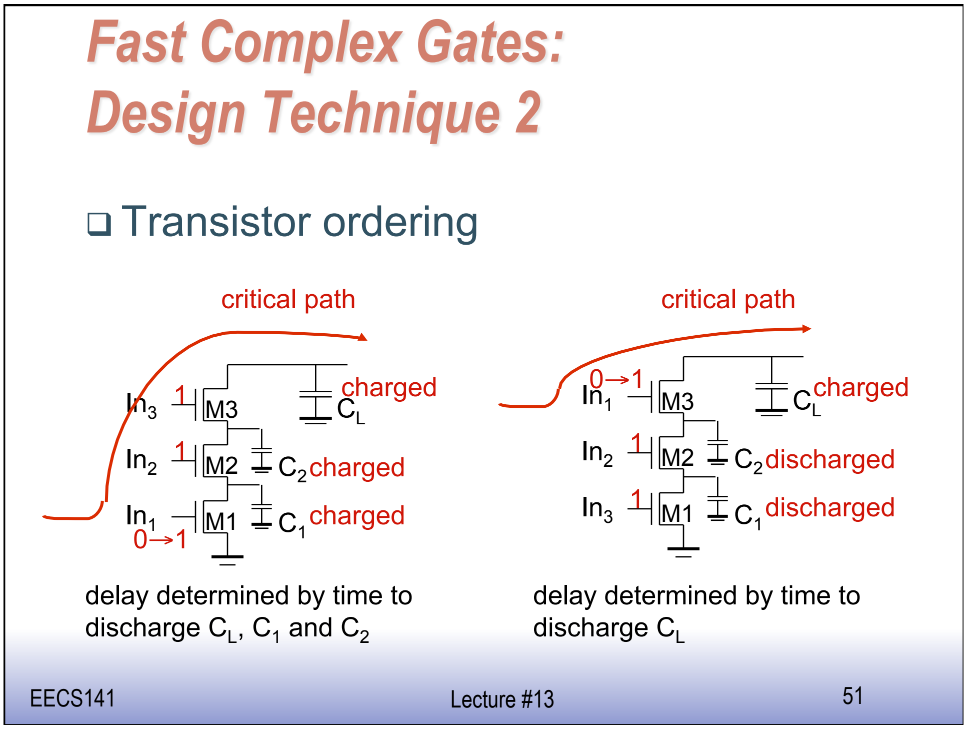
- ❑ Transistor sizing
  - ▪ as long as fan-out capacitance dominates
- ❑ Progressive sizing



Distributed RC line

M1 > M2 > M3 > … > MN
(the FET closest to the
output is the smallest)

Can reduce delay by more than 20%;
Be careful: input loading, junction caps,
decreasing gains as technology shrinks

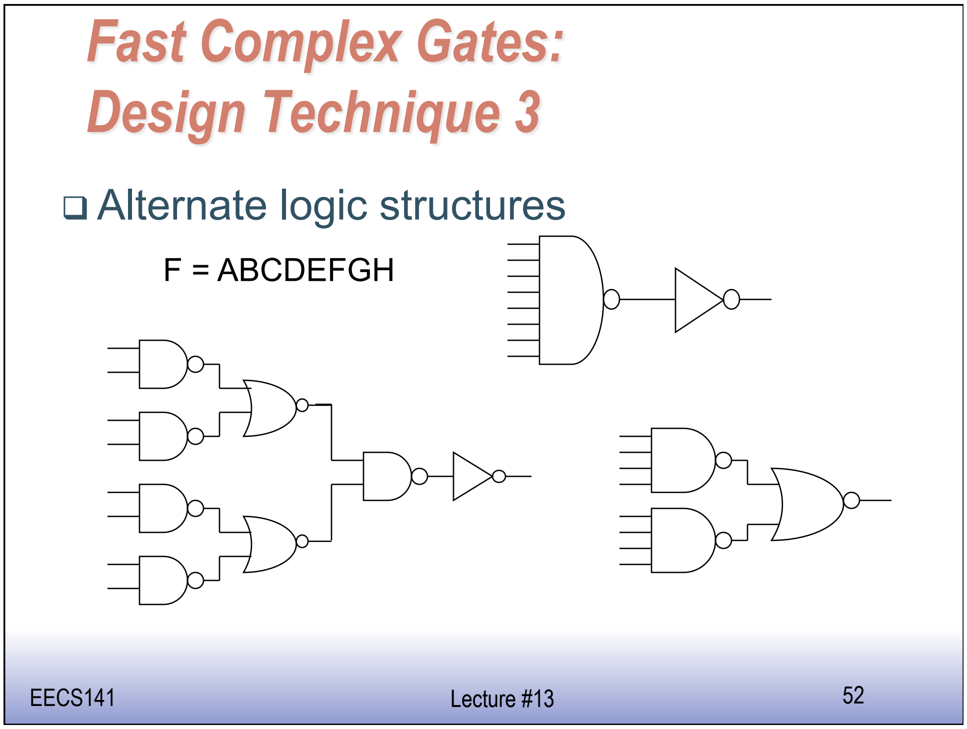


Fast Complex Gates: Design Technique 2

- ❑ Transistor ordering

delay determined by time to discharge C_L, C_1 and C_2

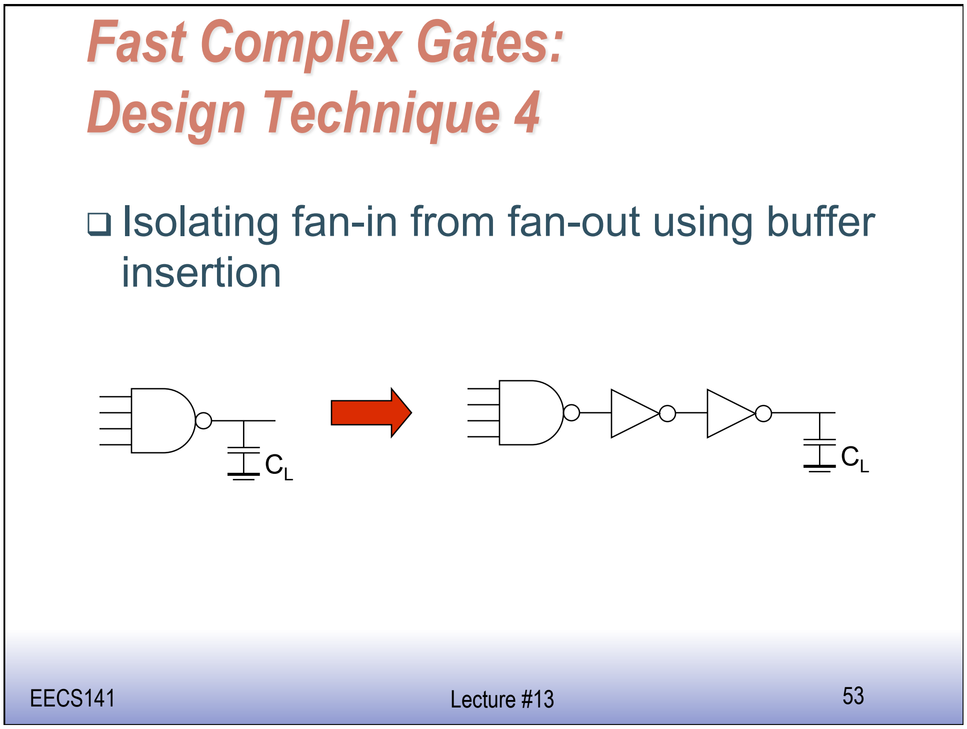
delay determined by time to discharge C_L



Fast Complex Gates: Design Technique 3

- ❑ Alternate logic structures

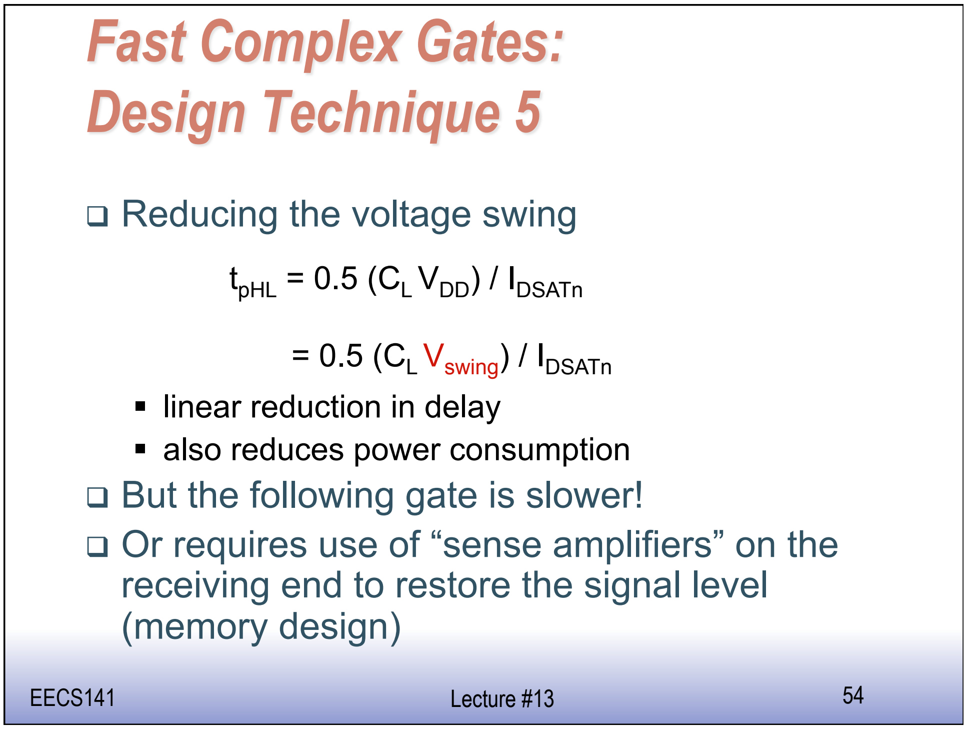
F = ABCDEFGH



Fast Complex Gates: Design Technique 4

- ❑ Isolating fan-in from fan-out using buffer insertion

EECS141 Lecture #13 53



Fast Complex Gates: Design Technique 5

- ❑ Reducing the voltage swing

$$t_{pHL} = 0.5\ (C_L\ V_{DD})\ /\ I_{DSATn}$$

$$= 0.5\ (C_L\ V_{swing})\ /\ I_{DSATn}$$

  - linear reduction in delay
  - also reduces power consumption
- ❑ But the following gate is slower!
- ❑ Or requires use of "sense amplifiers" on the receiving end to restore the signal level (memory design)

EECS141 Lecture #13 54