

EE141- Spring 2003
Lecture 25

Interconnect Effects
Input-Output



EE141

Schedule for the rest of the
semester

	Tu	Th
Week 13	Interconnect Launch Project 2	Interconnect (cntd) hw 9 due hw 10
Week 14	NO LECTURE (Faculty retreat)	Memory 1 midterm 2 results hw 10 due hw 11 (not graded)
Week 15	Memory 2	Future perspectives Project posters (1:30-5pm)

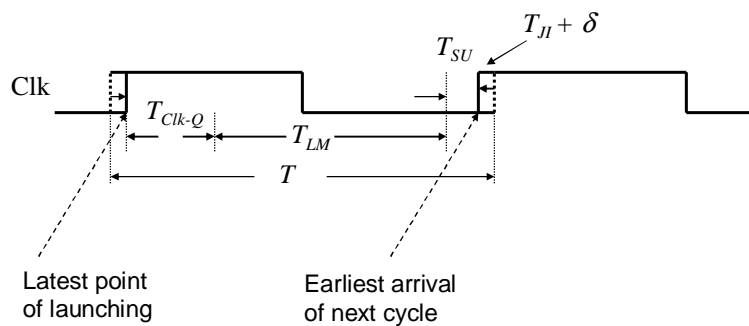
EE141

Today

- Short overview of last lecture
- Launch project 2 – discussion of dividers
- Start discussion of “Coping with interconnect – Chapter 9)

EE141

Longest Logic Path in Edge-Triggered Systems



EE141

Clock Constraints in Edge-Triggered Systems

If launching edge is late and receiving edge is early, the data will not be too late if:

$$T_{c-q} + T_{LM} + T_{SU} < T - T_{JL,1} - T_{JL,2} + \delta$$

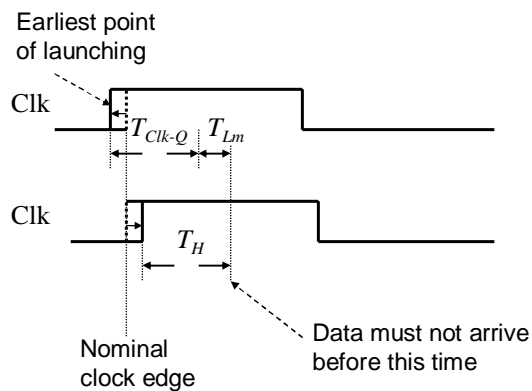
Minimum cycle time is determined by the maximum delays through the logic

$$T_{c-q} + T_{LM} + T_{SU} - \delta + 2 T_{JL} < T$$

Skew can be either positive or negative

EE141

Shortest Path



EE141

Clock Constraints in Edge-Triggered Systems

If launching edge is early and receiving edge is late:

$$T_{c-q} + T_{LM} < T_H + \delta$$

Minimum logic delay

$$T_{c-q} + T_{LM} < T_H + \delta$$

EE141

Summary

- Jitter always works against you. Should minimize it.
- Clock skew can work for or against you.
- Overall strategy: deliver clock to the different nodes in the network with minimum skew!

EE141

EE141- Spring 2003 Project 2

Divider

With contributions of J. Kubiawicz (CS152)



EE141

Divide: Paper & Pencil

	1001	Quotient
Divisor 1000	1001010	Dividend
	-1000	
	10	
	101	
	1010	
	-1000	
	10	Remainder (or Modulo result)

See how big a number can be subtracted, creating quotient bit on each step

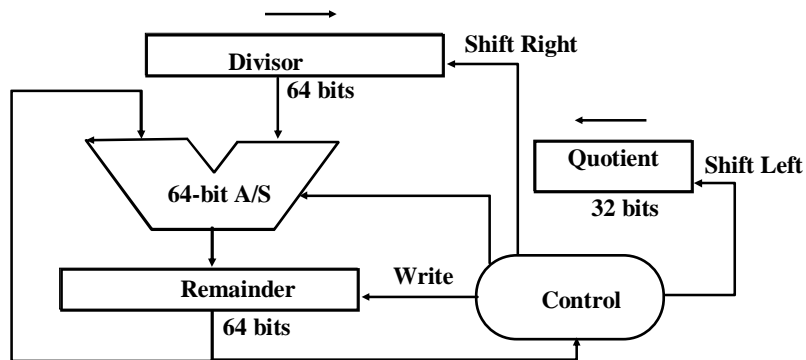
Binary => 1 * divisor or 0 * divisor

Dividend = Quotient x Divisor + Remainder
=> | Dividend | = | Quotient | + | Divisor |

EE141 - Project 2

DIVIDE HARDWARE Version 1

- 64-bit Divisor reg, 64-bit ALU, 64-bit Remainder reg, 32-bit Quotient reg



EE141 - Project 2

Divide Algorithm Version 1

Start: Place Dividend in Remainder

- Takes $n+1$ steps for n -bit Quotient & Rem.

Remainder	Quotient	Divisor
0000	0111	0000 0010 0000

1. Subtract the Divisor register from the Remainder register, and place the result in the Remainder register.

Test
Remainder ≥ 0 Remainder < 0

2a. Shift the Quotient register to the left setting the new rightmost bit to 1.

2b. Restore the original value by adding the Divisor register to the Remainder register, & place the sum in the Remainder register. Also shift the Quotient register to the left, setting the new least significant bit to 0.

3. Shift the Divisor register right 1 bit.

$n+1$ repetition? No: $< n+1$ repetitions

Yes: $n+1$ repetitions ($n = 4$ here)

Done

EE141 - Project 2

Divide Algorithm I example (7 / 2)

	Remainder	Quotient	Divisor
	0000 0111	00000	0010 0000
1:	1110 0111	00000	0010 0000
2:	0000 0111	00000	0010 0000
3:	0000 0111	00000	0001 0000
1:	1111 0111	00000	0001 0000
2:	0000 0111	00000	0001 0000
3:	0000 0111	00000	0000 1000
1:	1111 1111	00000	0000 1000
2:	0000 0111	00000	0000 1000
3:	0000 0111	00000	0000 0100
1:	0000 0011	00000	0000 0100
2:	0000 0011	00001	0000 0100
3:	0000 0011	00001	0000 0010
1:	0000 0001	00001	0000 0010
2:	0000 0001	00011	0000 0010
3:	0000 0001	00011	0000 0001

Answer:
Quotient = 3
Remainder = 1

EE141 - Project 2

Observations on Divide Version 1

- 1/2 bits in divisor always 0
=> 1/2 of 64-bit adder is wasted
=> 1/2 of divisor is wasted
- Instead of shifting divisor to right,
shift remainder to left?

EE141 - Project 2

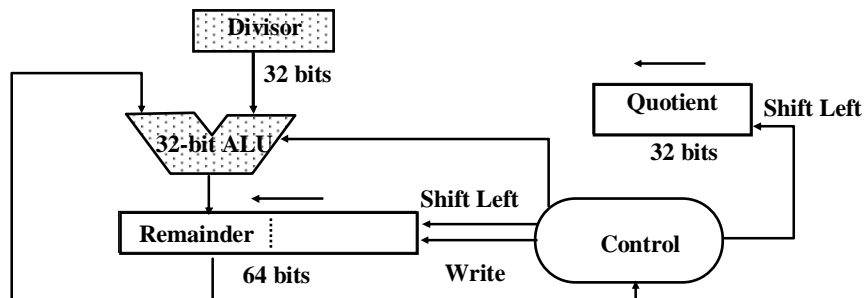
Divide Algorithm I example: wasted space

	Remainder	Quotient	Divisor
	0000 0111	00000	0010 0000
1:	1110 0111	00000	0010 0000
2:	0000 0111	00000	0010 0000
3:	0000 0111	00000	0001 0000
1:	1111 0111	00000	0001 0000
2:	0000 0111	00000	0001 0000
3:	0000 0111	00000	0000 1000
1:	1111 1111	00000	0000 1000
2:	0000 0111	00000	0000 1000
3:	0000 0111	00000	0000 0100
1:	0000 0011	00000	0000 0100
2:	0000 0011	00001	0000 0100
3:	0000 0011	00001	0000 0010
1:	0000 0001	00001	0000 0010
2:	0000 0001	00011	0000 0010
3:	0000 0001	00011	0000 0010

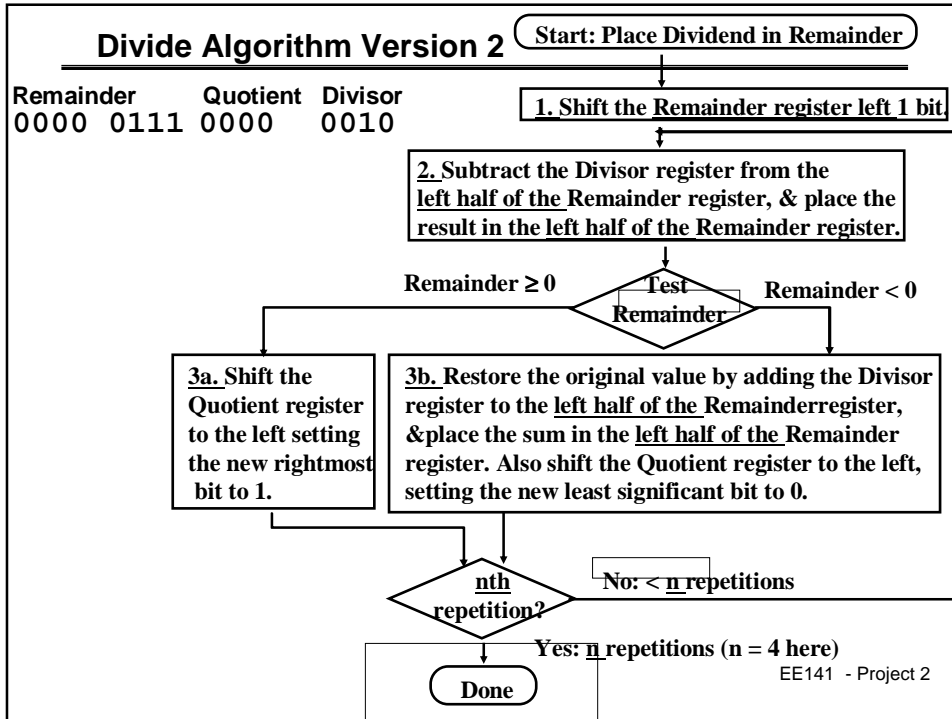
EE141 - Project 2

DIVIDE HARDWARE Version 2

- **32-bit Divisor reg, 32-bit ALU, 64-bit Remainder reg, 32-bit Quotient reg**



EE141 - Project 2



Divide Algorithm I version 2 (shift remainder)

	Remainder	Quotient	Divisor
	0000	0111	00000 0010
1:	1110	0111	00000 0010
2:	0000	0111	00000 0010
3:	0000	1110	00000 0010
1:	1110	1110	00000 0010
2:	0000	1110	00000 0010
3:	0001	1100	00000 0010
1:	1111	1100	00000 0010
2:	0001	1100	00000 0010
3:	0011	1000	00000 0010
1:	0001	1000	00001 0010
2:	0001	1000	00001 0010
3:	0011	0000	00001 0010
1:	0001	0000	00011 0010
2:	0001	0000	00011 0010

EE141 - Project 2

Divide: Revisited

Non-restoring divider

	1001	Quotient
Divisor	1000	Dividend
	1001010	
	-1000	
	00010	
	-1000	
	110101	
	+1000	
	111010	
	+1000	
	00010	Remainder (or Modulo result)

Avoids extra step of “restoration” when partial result is negative.
 Instead of subtract, adds divisor on next iteration

EE141 - Project 2

Divide Algorithm I example: non-restoring

	Remainder	Quotient	Divisor
	0000	0111	00000
1:	1110	0111	00000
2:	1100	1110	00000
1:	1110	1110	00000
2:	1101	1100	00000
1:	1111	1100	00000
2:	1111	1000	00000
1:	0001	1000	00001
2:	0011	0000	00001
1:	0001	0000	00011
			0010

EE141 - Project 2

Project 2

- Goal: Design Divider with Minimum Clock Frequency
 - » Supply voltage fixed at 2 V, 0.25 μm CMOS
 - » 4 bit dividend, divisor, quotient, remainder
 - » Two's complement, all words positive
 - » Choice of static and/or pass-transistor logic
 - » Given register schematics
 - » Given output loads, input waveforms, clock waveforms

EE141

Design Phases

- Determine block diagram of divider that will lead to minimum clock-cycle (be inspired!)
- Design schematics of basic cells
- Demonstrate functionality of divider
- Determine worst-case critical path
- Size transistors, and simulate critical path using SPICE
(make sure you include all loading factors needed)

EE141

Reporting

- Poster session on Th May 8; 1:30-5pm
- Prepare 9 slides poster (powerpoint template will be provided)
 - » Choice of schematics
 - » Show functionality
 - » Transistor sizing and cell design
 - » Critical path analysis
- 10' per group oral presentation (2 parallel sessions)
- End of the semester celebration (cookies and soda)

EE141

Interconnect Issues



EE141

Impact of Interconnect Parasitics

- Reduce Robustness
- Affect Performance

Classes of Parasitics



- Capacitive
- Resistive
- Inductive

EE141

INTERCONNECT

Dealing with Capacitance

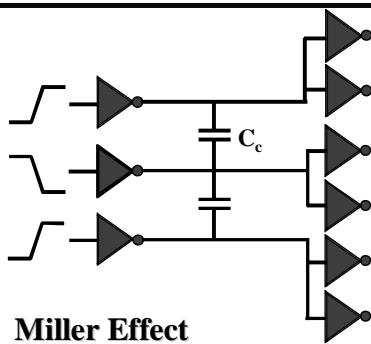
EE141

Dealing with Capacitive Cross Talk

- Avoid floating nodes
- Protect sensitive nodes
- Make rise and fall times as large as possible
- Differential signaling
- Do not run wires together for a long distance
- Use shielding wires
- Use shielding layers

EE141

Delay Degradation



- Impact of neighboring signal activity on switching delay
- When neighboring lines switch in opposite direction of victim line, delay increases

Miller Effect

- Both terminals of capacitor are switched in opposite directions ($0 \rightarrow V_{dd}$, $V_{dd} \rightarrow 0$)
- Effective voltage is doubled and additional charge is needed (from $Q=CV$)

EE141

Impact of Cross Talk on Delay

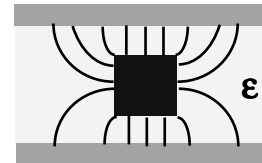
bit $k-1$	bit k	bit $k+1$	Delay factor g
↑	↑	↑	1
↑	↑	—	$1+r$
↑	↑	↓	$1+2r$
—	↑	—	$1+2r$
—	↑	↓	$1+3r$
↓	↑	↓	$1+4r$

r is ratio between capacitance to GND and to neighbor

EE141

Interconnect Projections Low-k dielectrics

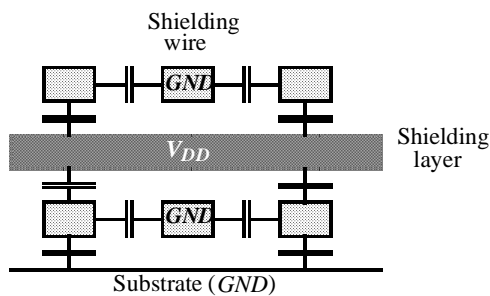
- Both *delay and power are reduced* by dropping interconnect capacitance
- Types of low-k materials include: inorganic (SiO_2), organic (Polyimides) and aerogels (ultra low-k)
- The numbers below are on the conservative side of the NRTS roadmap



Generation	0.25 μm	0.18 μm	0.13 μm	0.1 μm	0.07 μm	0.05 μm
Dielectric Constant	3.3	2.7	2.3	2.0	1.8	1.5

EE141

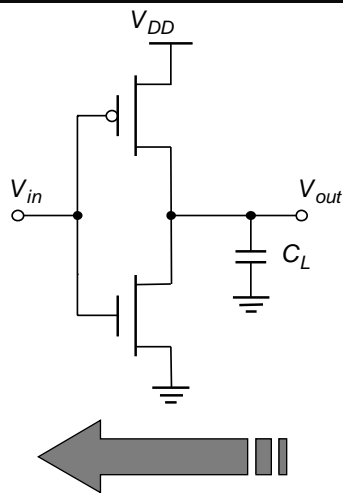
How to Battle Capacitive Crosstalk



- Avoid large crosstalk cap's
- Avoid floating nodes
- Isolate sensitive nodes
- Control rise/fall times
- Shield!
- Differential signaling

EE141

Driving Large Capacitances

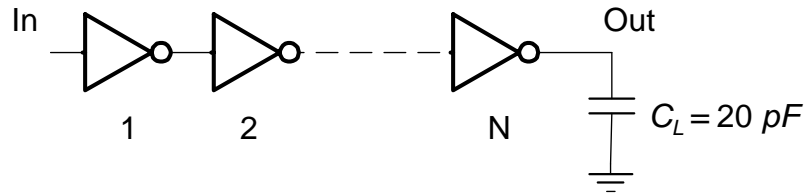


$$t_p = \frac{C_L V_{swing}}{I_{av}}$$

- Transistor Sizing
- Cascaded Buffers

EE141

Using Cascaded Buffers



0.25 μm process
 $C_{in} = 2.5 \text{ fF}$
 $t_{p0} = 30 \text{ ps}$

$F = C_L/C_{in} = 8000$
 $f_{opt} = 3.6 \quad N = 7$
 $t_p = 0.76 \text{ ns}$

(See Chapter 5)

EE141

Output Driver Design

Trade off Performance for Area and Energy

Given t_{pmax} find N and f

- Area

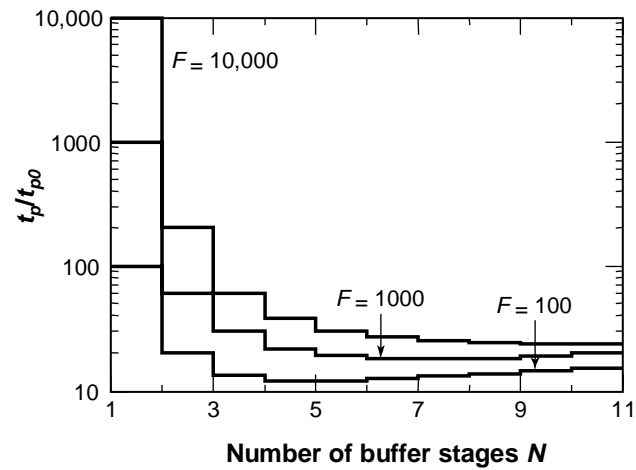
$$A_{driver} = (1 + f + f^2 + \dots + f^{N-1}) A_{min} = \frac{f^N - 1}{f - 1} A_{min} = \frac{F - 1}{f - 1} A_{min}$$

- Energy

$$E_{driver} = (1 + f + f^2 + \dots + f^{N-1}) C_i V_{DD}^2 = \frac{F - 1}{f - 1} C_i V_{DD}^2 \approx \frac{C_L}{f - 1} V_{DD}^2$$

EE141

Delay as a Function of F and N



EE141

Output Driver Design

0.25 μm process, $C_L = 20 \text{ pF}$

Transistor Sizes for optimally-sized cascaded buffer $t_p = 0.76 \text{ ns}$

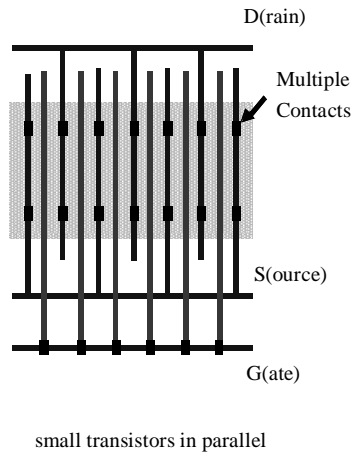
Stage	1	2	3	4	5	6	7
W_n (μm)	0.375	1.35	4.86	17.3	63	229.8	816.3
W_p (μm)	0.71	2.56	9.2	33.1	119.2	429.3	1545.5

Transistor Sizes of redesigned cascaded buffer $t_p = 1.8 \text{ ns}$

Stage	1	2	3
W_n (μm)	0.375	7.5	150
W_p (μm)	0.71	14.4	284

EE141

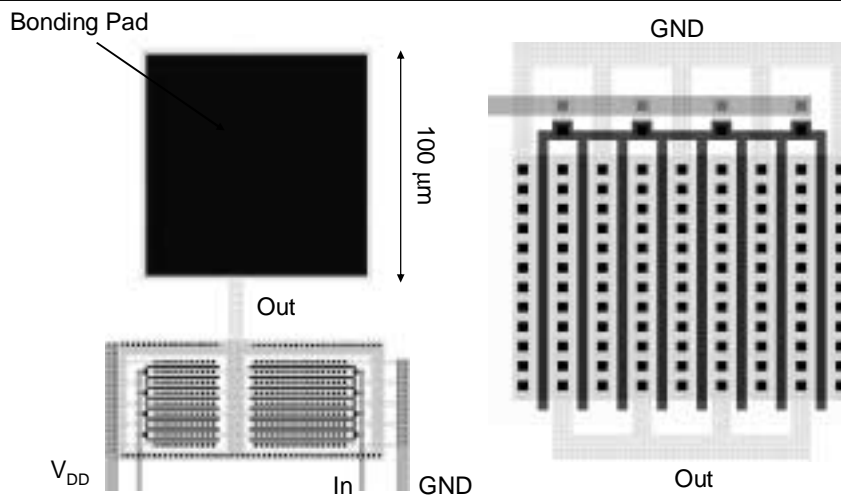
How to Design Large Transistors



Reduces diffusion capacitance

EE141

Bonding Pad Design



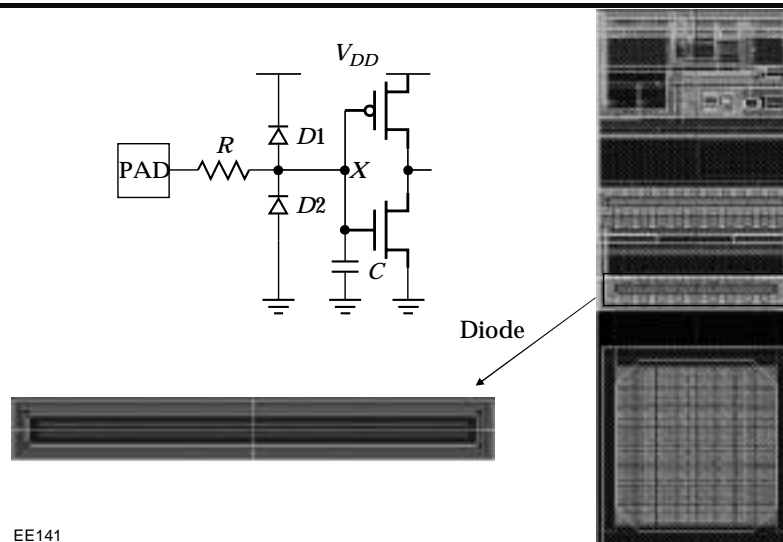
EE141

ESD Protection

- When a chip is connected to a board, there is unknown (potentially large) static voltage difference
- Equalizing potentials requires (large) charge flow through the pads
- Diodes sink this charge into the substrate – need guard rings to pick it up.

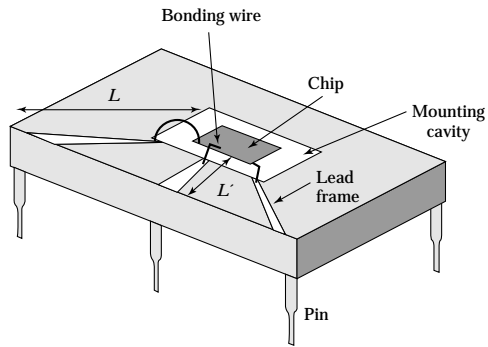
EE141

ESD Protection



EE141

Chip Packaging



- Bond wires ($\sim 25\mu\text{m}$) are used to connect the package to the chip

- Pads are arranged in a frame around the chip

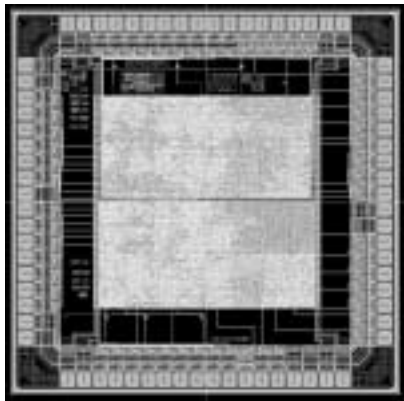
- Pads are relatively large ($\sim 100\mu\text{m}$ in $0.25\mu\text{m}$ technology), with large pitch ($100\mu\text{m}$)

- Many chips areas are 'pad limited'

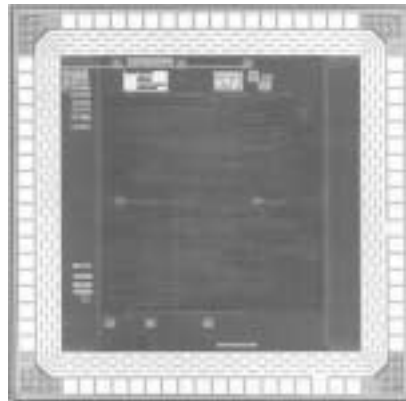
EE141

Pad Frame

Layout



Die Photo



EE141

Chip Packaging

- An alternative is 'flip-chip':
 - » Pads are distributed around the chip
 - » The soldering balls are placed on pads
 - » The chip is 'flipped' onto the package
 - » Can have many more pads

EE141

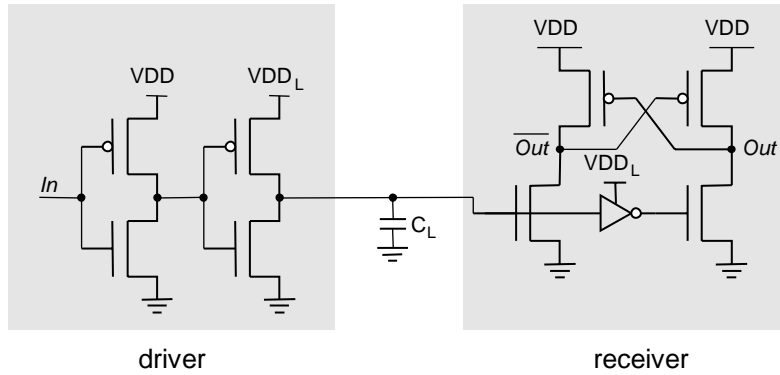
Reducing the swing

$$t_{pHL} = \frac{C_L V_{swing}/2}{I_{av}}$$

- Reducing the swing potentially yields linear reduction in delay
- Also results in reduction in power dissipation
- Delay penalty is paid by the receiver
- Requires use of "sense amplifier" to restore signal level
- Frequently designed differentially (e.g. LVDS)

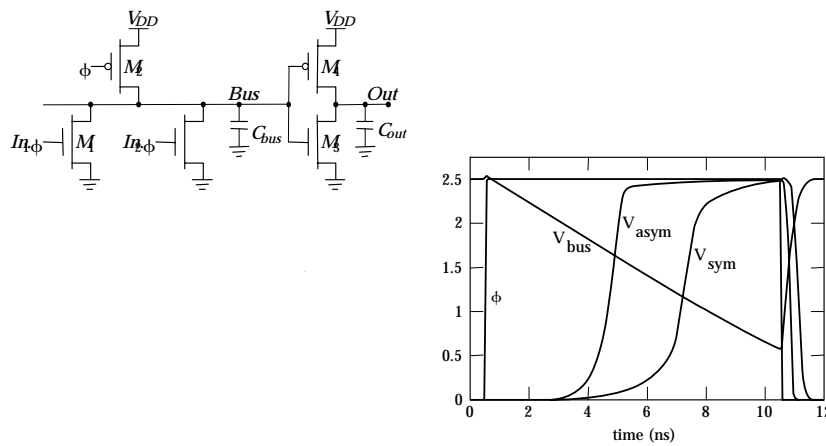
EE141

Single-Ended Static Driver and Receiver



EE141

Dynamic Reduced Swing Network



EE141