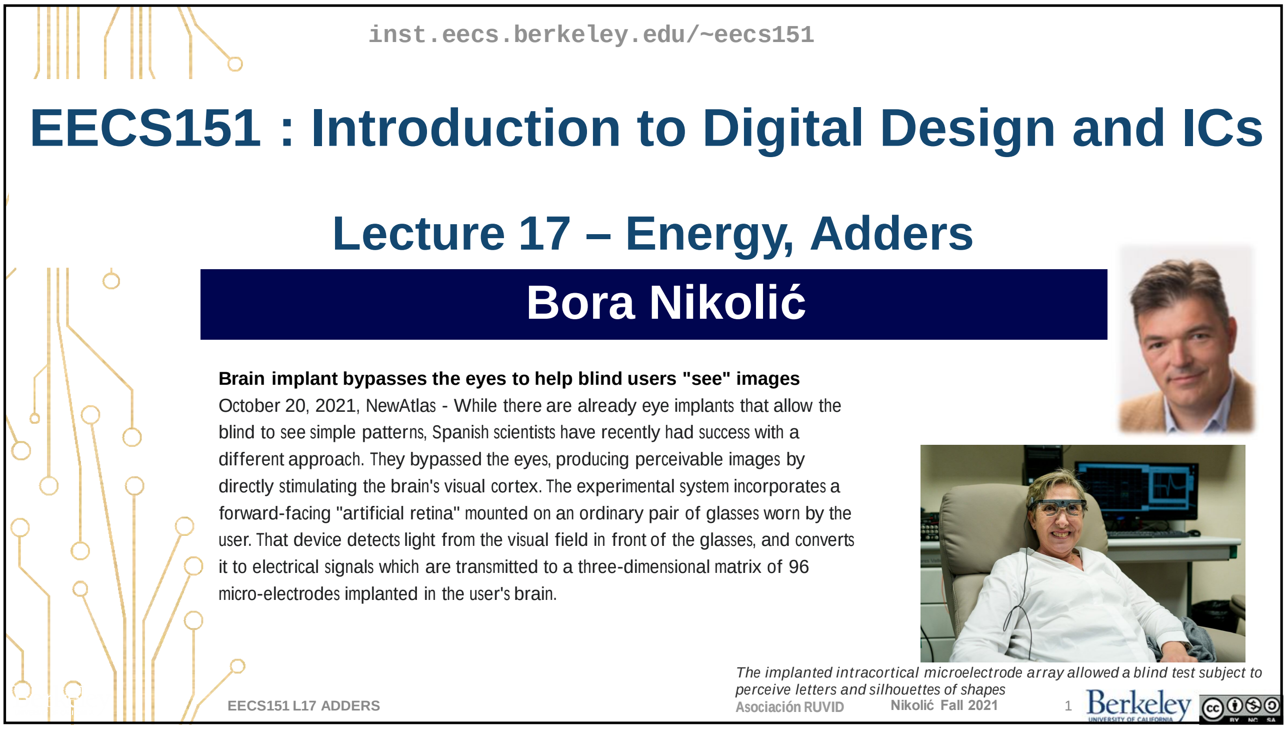


inst.eecs.berkeley.edu/~eecs151

EECS151 : Introduction to Digital Design and ICs

Lecture 17 – Energy, Adders

Bora Nikolić

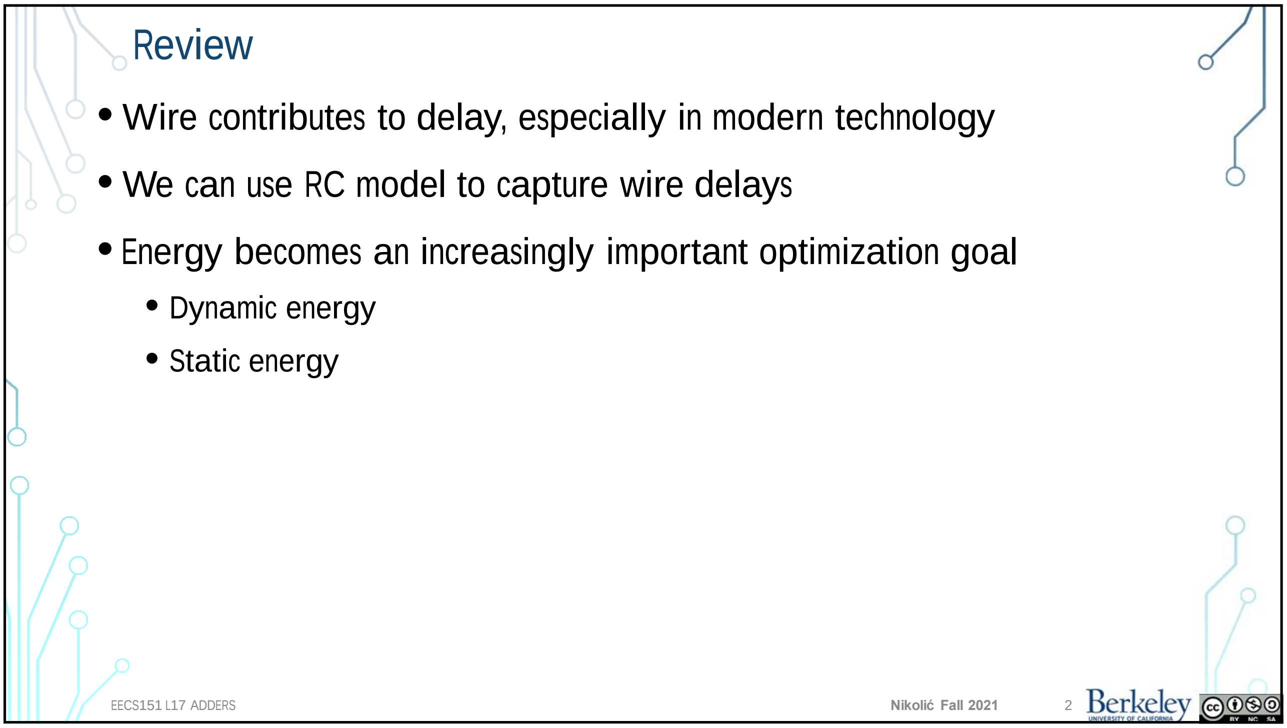
Brain implant bypasses the eyes to help blind users "see" images

October 20, 2021, NewAtlas - While there are already eye implants that allow the blind to see simple patterns, Spanish scientists have recently had success with a different approach. They bypassed the eyes, producing perceivable images by directly stimulating the brain's visual cortex. The experimental system incorporates a forward-facing "artificial retina" mounted on an ordinary pair of glasses worn by the user. That device detects light from the visual field in front of the glasses, and converts it to electrical signals which are transmitted to a three-dimensional matrix of 96 micro-electrodes implanted in the user's brain.

The implanted intracortical microelectrode array allowed a blind test subject to perceive letters and silhouettes of shapes

Asociación RUVID

1



Review

- Wire contributes to delay, especially in modern technology
- We can use RC model to capture wire delays
- Energy becomes an increasingly important optimization goal
  - Dynamic energy
  - Static energy

2

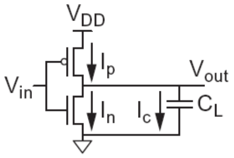
Dynamic Power

Charging and Discharging a Capacitor

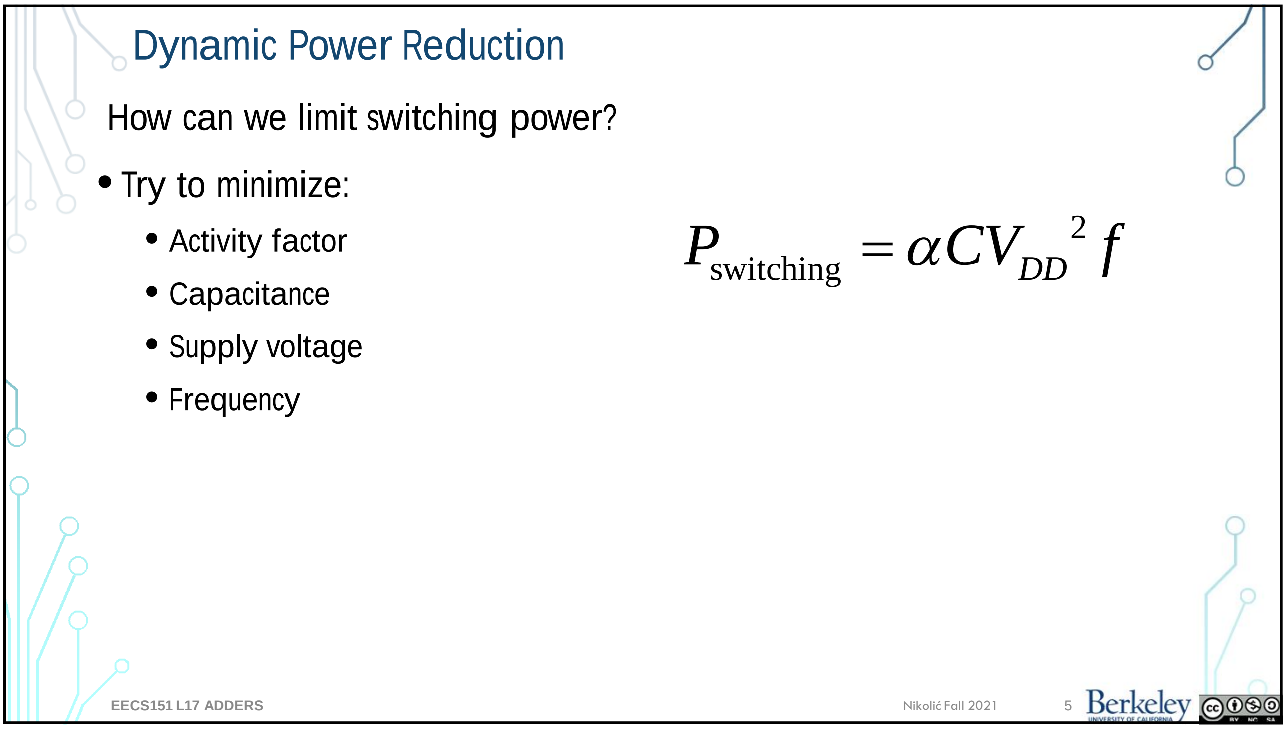
- When the gate output rises
  - Energy stored in capacitor is $E_C = \frac{1}{2} C_L V_{DD}^2$
  - But energy drawn from the supply is

$$E_{VDD} = \int_0^\infty I(t) V_{DD}\, dt = \int_0^\infty C_L \frac{dV}{dt} V_{DD}\, dt$$

$$= C_L V_{DD} \int_0^{V_{DD}} dV = C_L V_{DD}^2$$



  - Half the energy from V_{DD} is dissipated in the pMOS transistor as heat, other half stored in capacitor
- When the gate output transitions HL
  - Energy in capacitor is dumped to GND
  - Dissipated as heat in the NMOS transistor

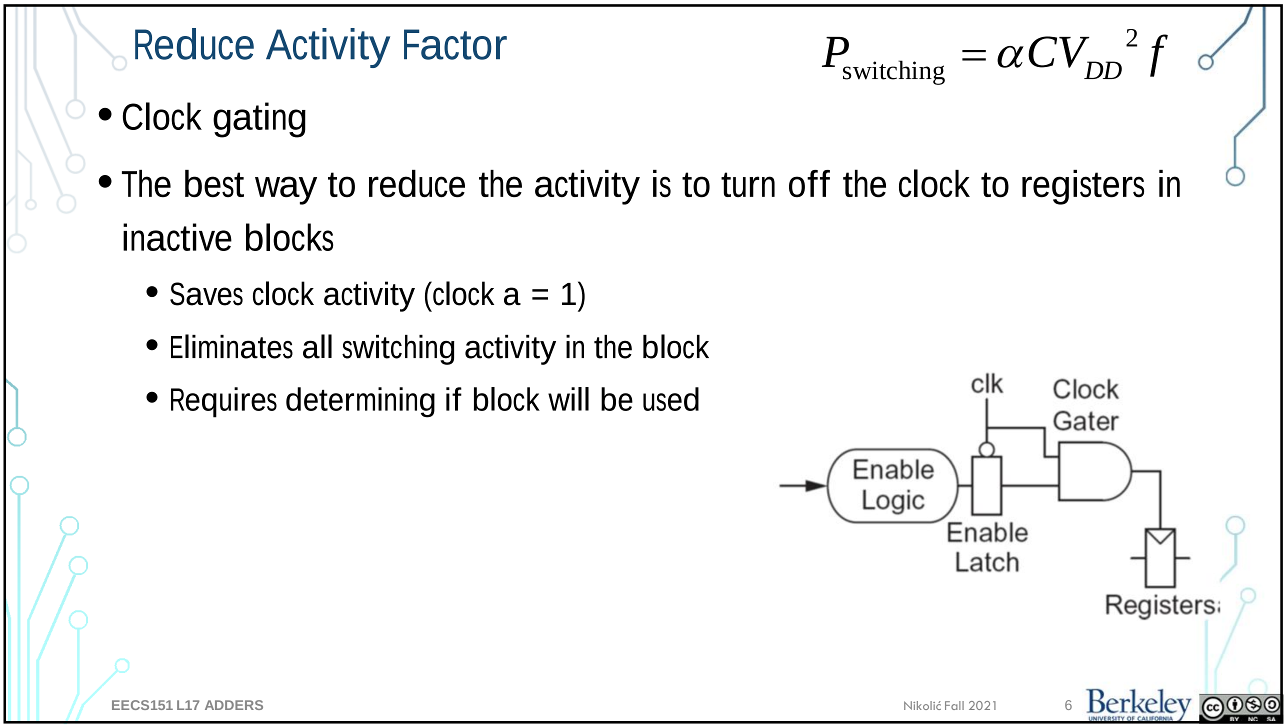


Dynamic Power Reduction

How can we limit switching power?

- Try to minimize:
  - Activity factor
  - Capacitance
  - Supply voltage
  - Frequency

$$P_{\text{switching}} = \alpha C V_{DD}{}^{2} f$$



Reduce Activity Factor

$$P_{\text{switching}} = \alpha C V_{DD}{}^{2} f$$

- Clock gating
- The best way to reduce the activity is to turn off the clock to registers in inactive blocks
  - Saves clock activity (clock a = 1)
  - Eliminates all switching activity in the block
  - Requires determining if block will be used

Reduce Capacitance

$$P_{\text{switching}} = \alpha C V_{DD}{}^{2} f$$

- Gate capacitance
  - Fewer stages of logic
  - Smaller gate sizes
- Wire capacitance
  - Good floorplanning to keep communicating blocks close to each other

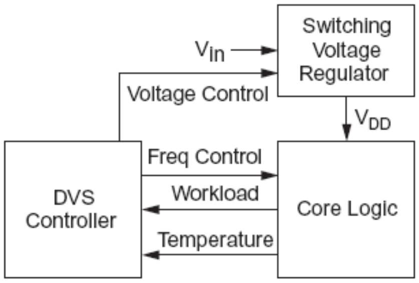
Reduce Voltage / Frequency

$$P_{\text{switching}} = \alpha C V_{DD}{}^{2} f$$

- Run each block at the lowest possible voltage and frequency that meets performance requirements

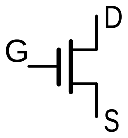
- Voltage domains
  - Provide separate supplies to different blocks

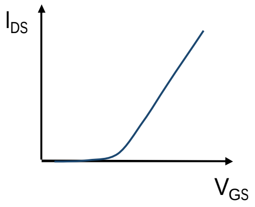
- Dynamic voltage / frequency scaling
  - Adjust V_{DD} and f according to workload



Leakage Power

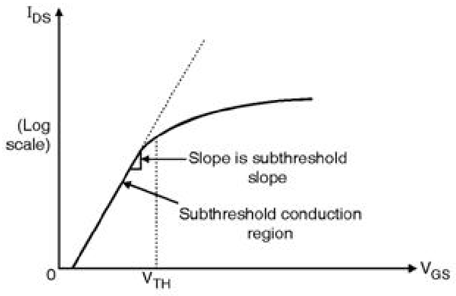
Subthreshold Leakage



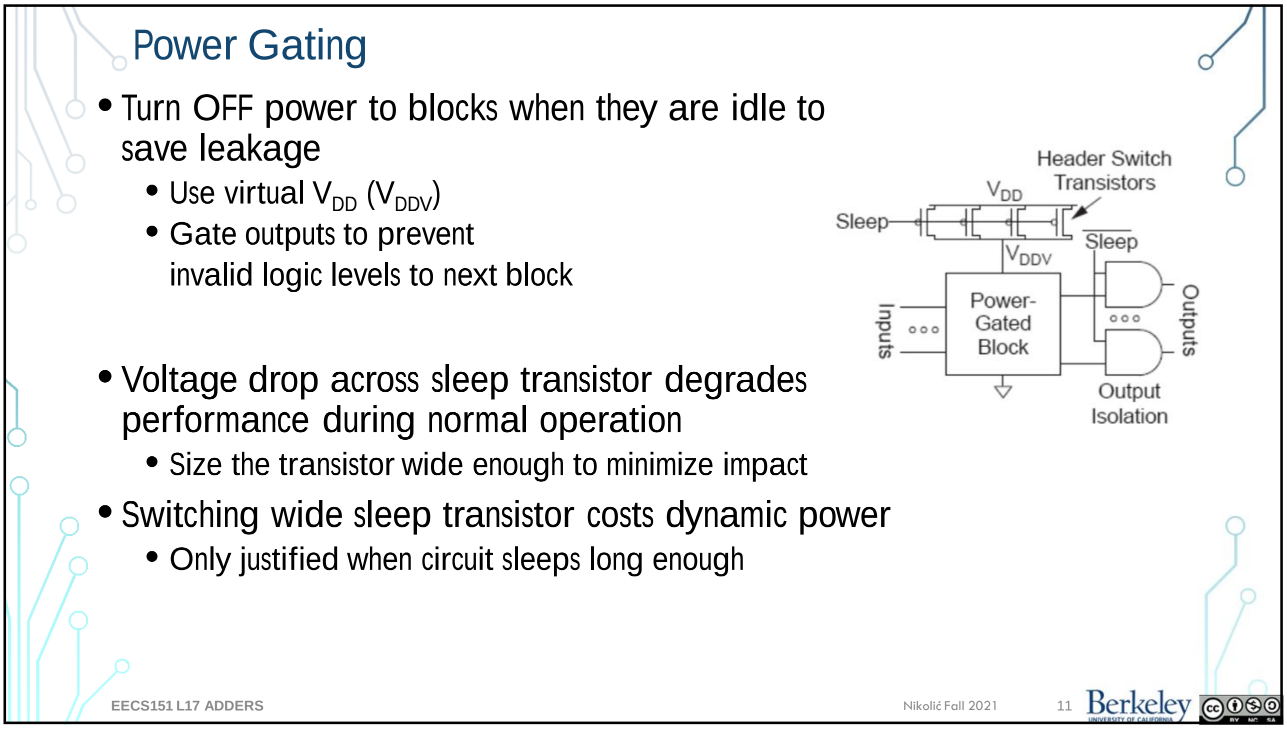


Nearly linear

$I_{DS} \sim K(V_{GS}\text{-}V_{Th})$



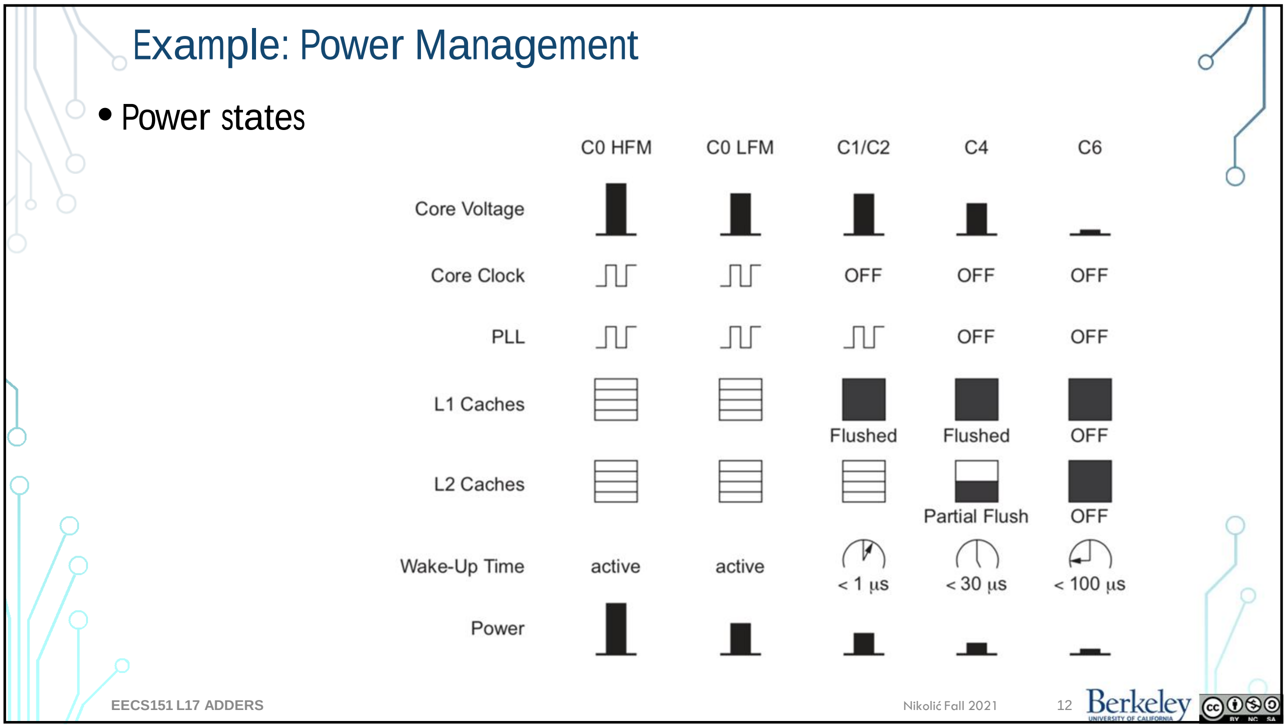
I_{DS} Vs V_{GS} characteristics in log scale



Power Gating

- Turn OFF power to blocks when they are idle to save leakage
  - Use virtual V_{DD} (V_{DDV})
  - Gate outputs to prevent invalid logic levels to next block
- Voltage drop across sleep transistor degrades performance during normal operation
  - Size the transistor wide enough to minimize impact
- Switching wide sleep transistor costs dynamic power
  - Only justified when circuit sleeps long enough

11

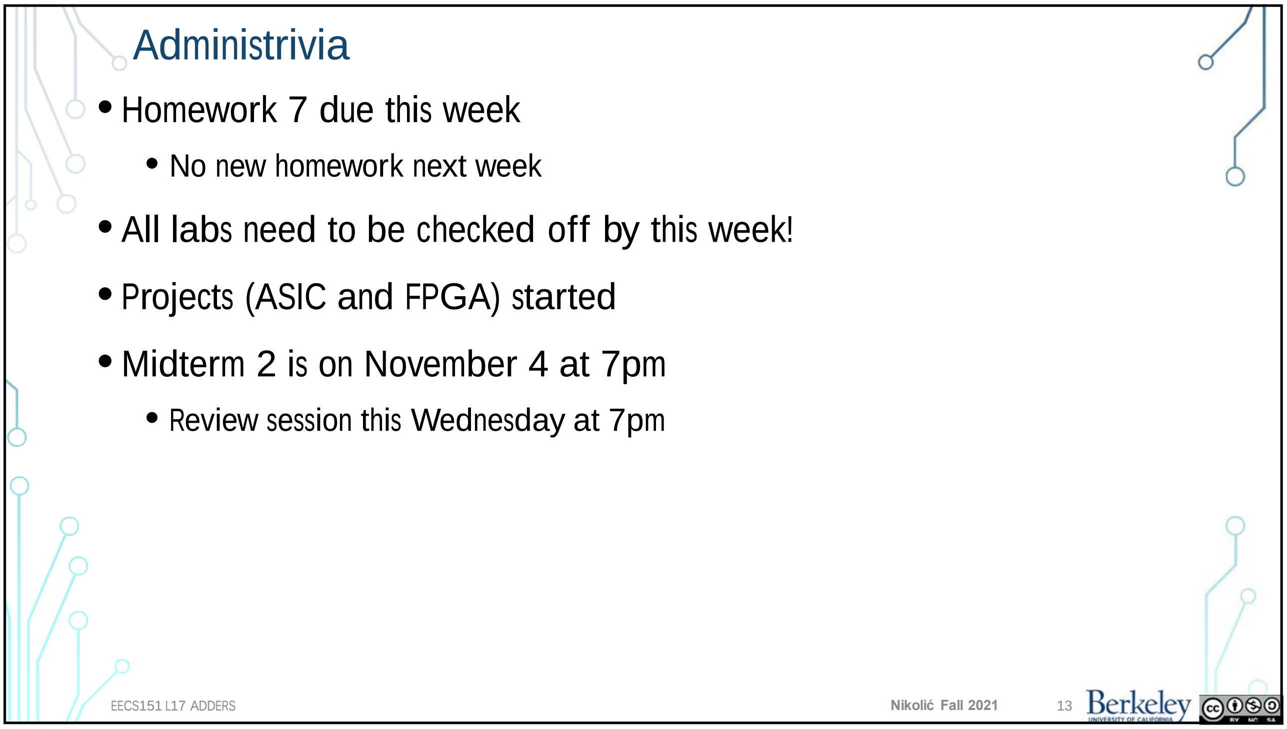


Example: Power Management

- Power states

	C0 HFM	C0 LFM	C1/C2	C4	C6
Core Voltage	▮	▮	▮	▮	▁
Core Clock	⎍⎍	⎍⎍	OFF	OFF	OFF
PLL	⎍⎍	⎍⎍	⎍⎍	OFF	OFF
L1 Caches	▤	▤	■ Flushed	■ Flushed	■ OFF
L2 Caches	▤	▤	▤	⬓ Partial Flush	■ OFF
Wake-Up Time	active	active	< 1 µs	< 30 µs	< 100 µs
Power	▮	▮	▮	▬	▁

12

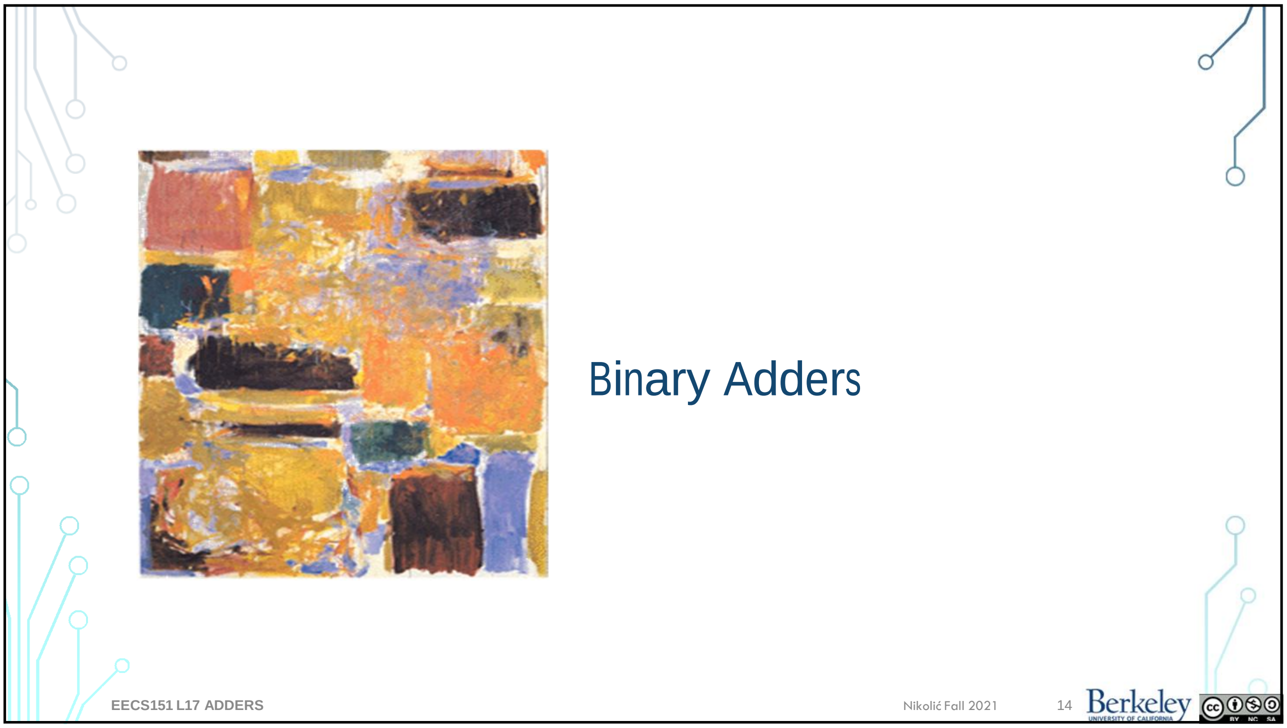


Administrivia

- Homework 7 due this week
  - No new homework next week
- All labs need to be checked off by this week!
- Projects (ASIC and FPGA) started
- Midterm 2 is on November 4 at 7pm
  - Review session this Wednesday at 7pm

EECS151 L17 ADDERS Nikolić Fall 2021 13

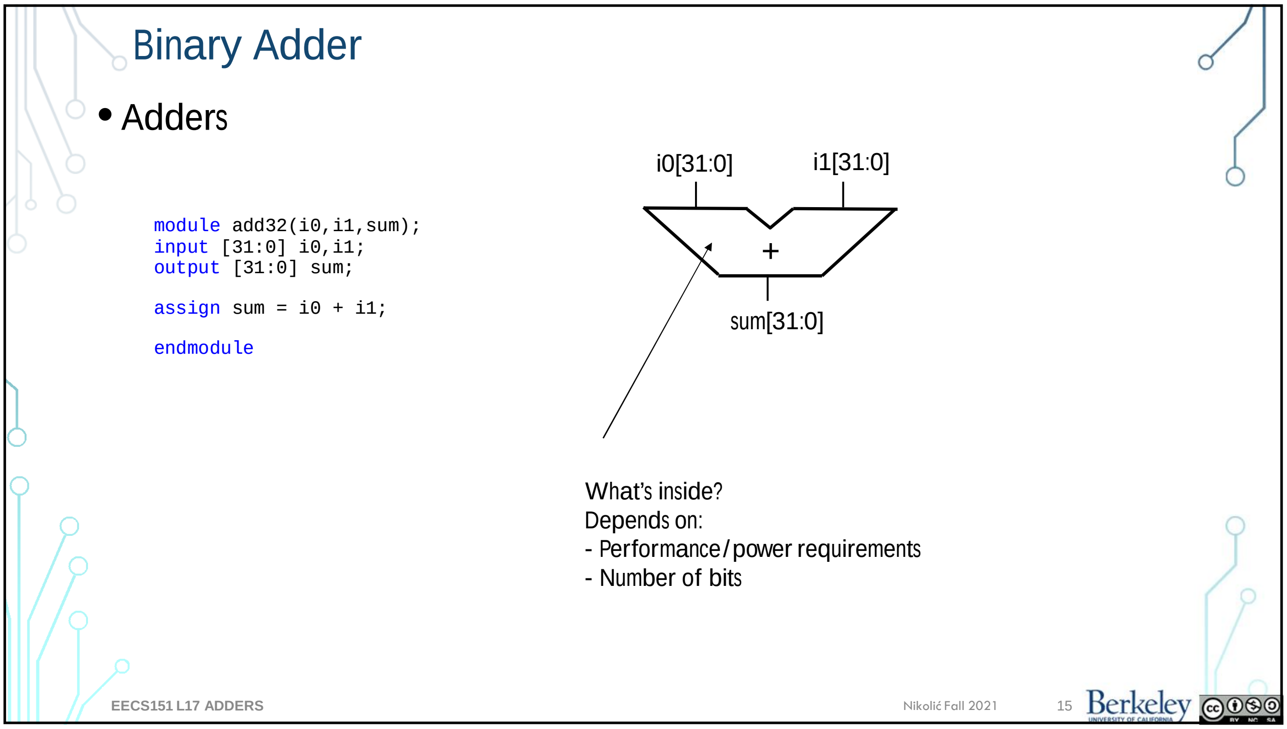
13



Binary Adders

EECS151 L17 ADDERS Nikolić Fall 2021 14

14



Binary Adder

- Adders

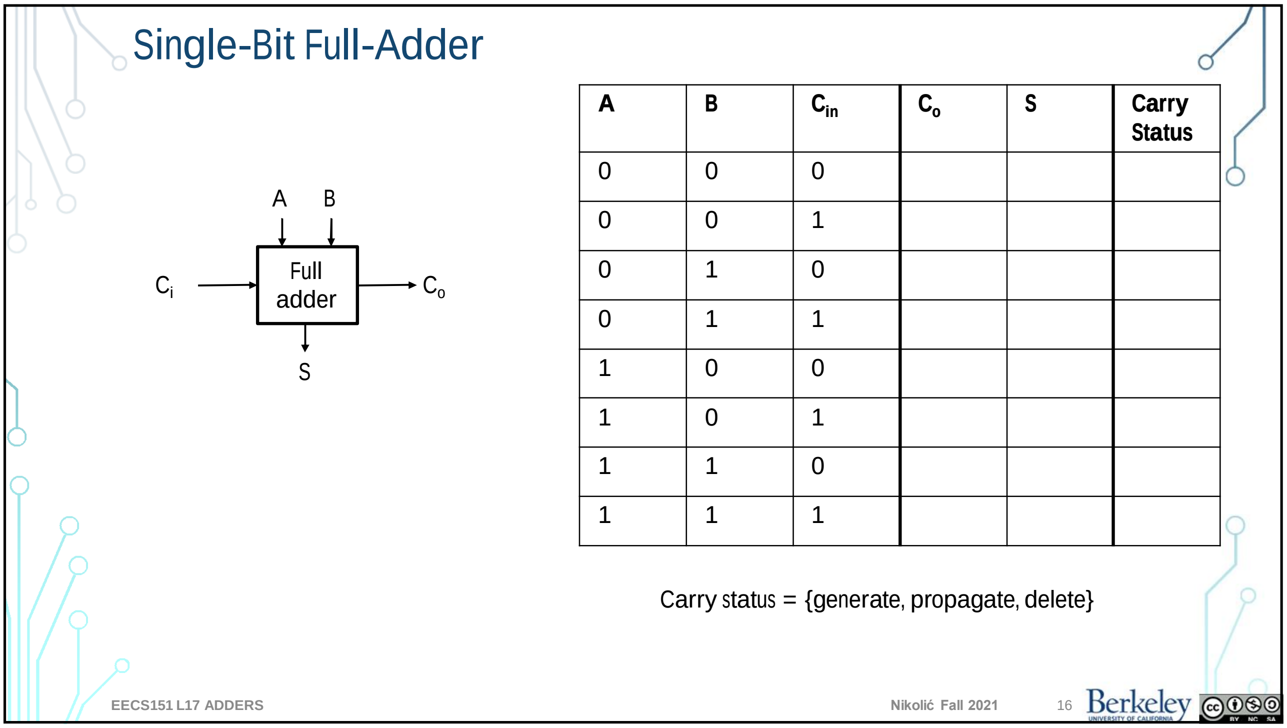
```
module add32(i0,i1,sum);
input [31:0] i0,i1;
output [31:0] sum;

assign sum = i0 + i1;

endmodule
```

What's inside?
Depends on:
- Performance/power requirements
- Number of bits

15



Single-Bit Full-Adder

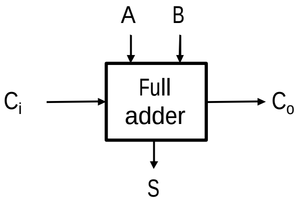
A	B	C_{in}	C_o	S	Carry Status
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Carry status = {generate, propagate, delete}

16

Single-Bit Full Adder

- Logic equations



$$S = A \oplus B \oplus Ci$$

$$S = A\,\overline{B}\,\overline{C_i} + \overline{A}\,B\,\overline{C_i} + \overline{A}\,\overline{B}\,C_i + A\,B\,Ci$$

$$C_o = A\,B + B\,Ci + A\,Ci$$

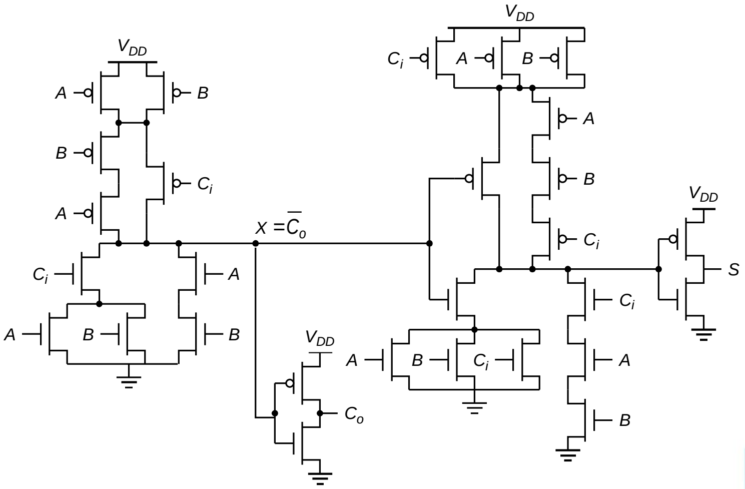
Static CMOS Full Adder

- Direct mapping of logic equations

$$C_o = A\,B + B\,Ci + A\,Ci$$

$$\overline{C_o} = \overline{AB + C_i(A + B)}$$

$$S = ABC_i + (A + B + C_i)\,\overline{C_o}$$



28 Transistors

Express Sum and Carry as a function of P, G, D

- Define generate, propagate and delete as functions of A, B
  - Will use two at a time

Generate (G) = AB

Propagate (P) = A + B (or A ⊕ B)

Delete = $\overline{A}\,\overline{B}$

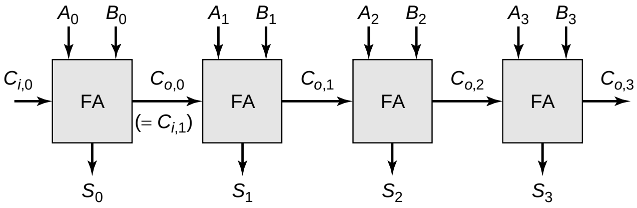
C_o = A B + B Ci + A Ci = G + P C_i

A	B	C_{in}	G	P	K	C_o	S
0	0	0	0	0	1	0	0
		1				0	1
0	1	0	0	1	0	0	1
		1				1	0
1	0	0	0	1	0	0	1
		1				1	0
1	1	0	1	X	0	1	0
		1				1	1

Can also derive expressions for C_o based on D and P

The Ripple-Carry Adder

- 4-bit adder



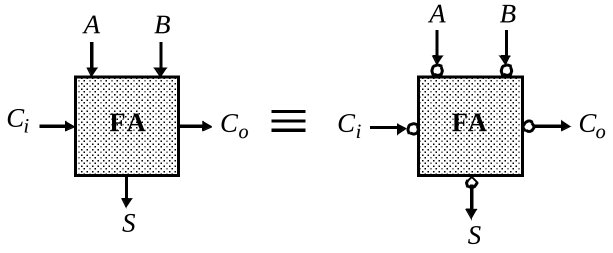
Worst case delay linear with the number of bits

t_d = O(N)

t_{adder} = (N-1)t_{carry} + t_{sum}

Goal: Make the fastest possible carry path circuit

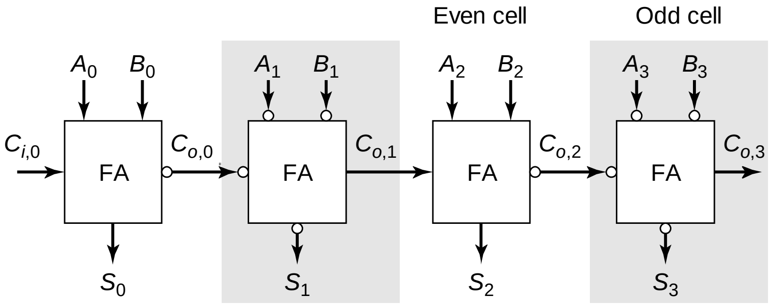
Inversion Property



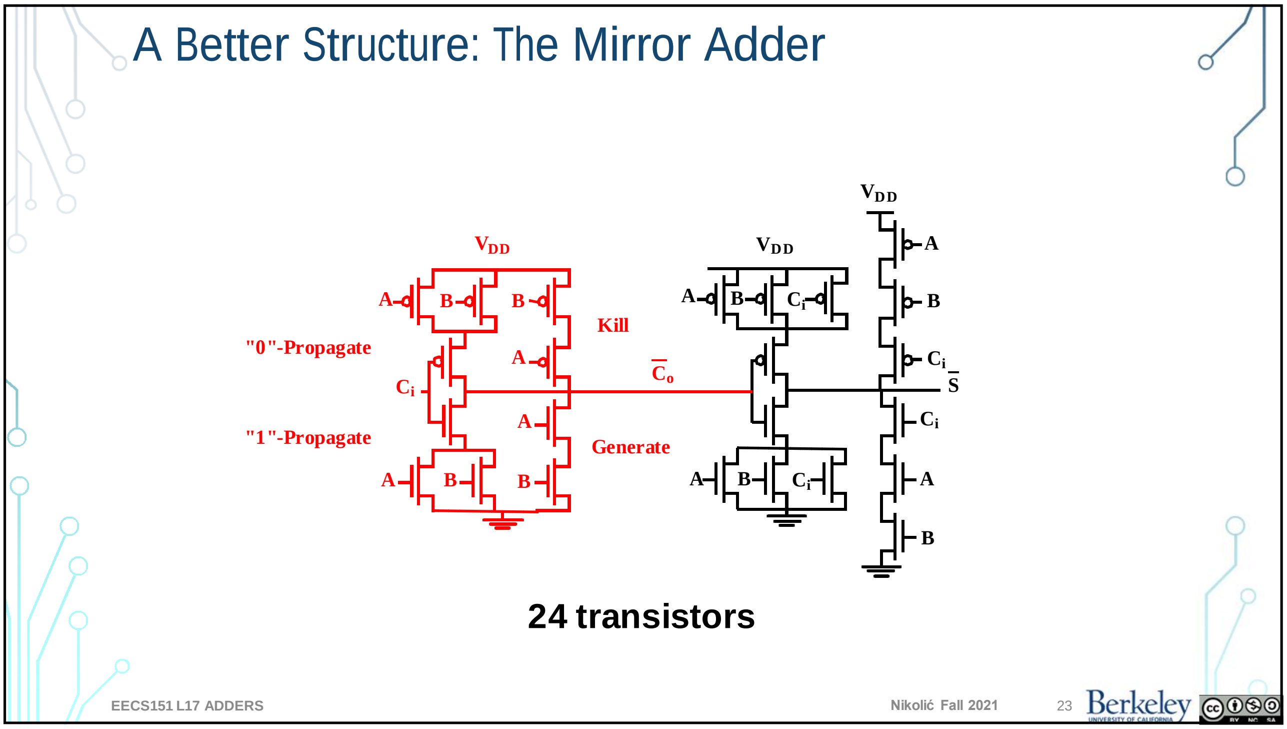
$$\bar{S}(A,B,C_i) = S(\bar{A},\bar{B},\bar{C}_i)$$

$$\overline{C_o}(A,B,C_i) = C_o(\bar{A},\bar{B},\bar{C}_i)$$

Minimize Critical Path by Reducing Inverting Stages



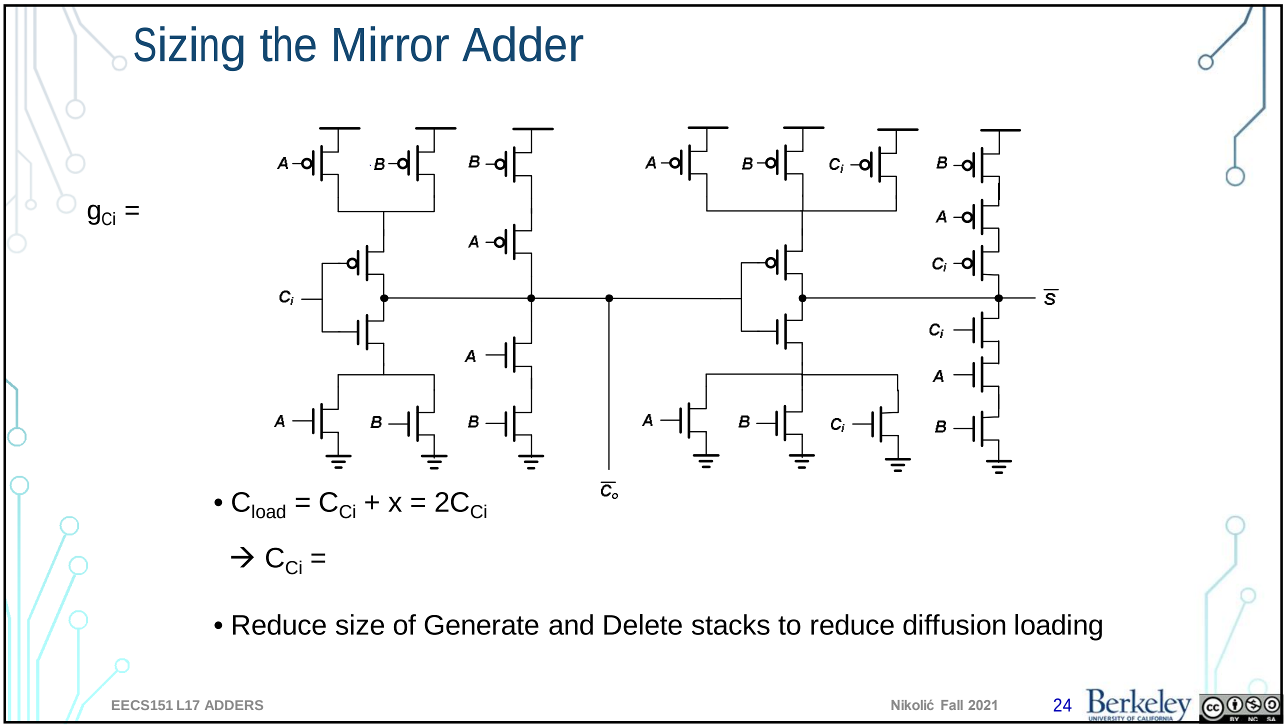
Exploit Inversion Property



A Better Structure: The Mirror Adder

24 transistors

23



Sizing the Mirror Adder

g_{Ci} =

- $C_{load} = C_{Ci} + x = 2C_{Ci}$

  → C_{Ci} =

- Reduce size of Generate and Delete stacks to reduce diffusion loading

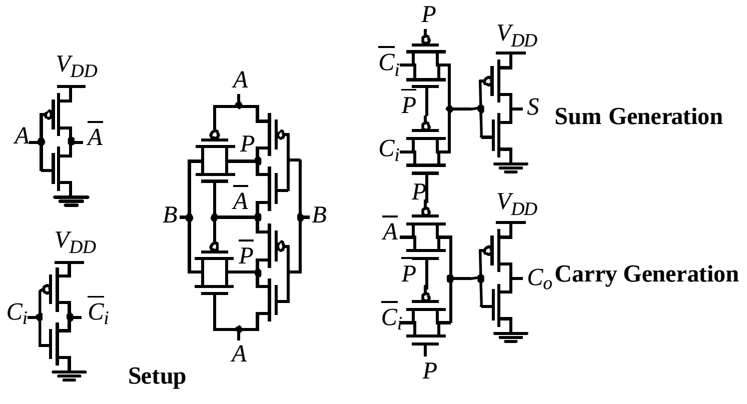
24

The Mirror Adder

- The NMOS and PMOS chains are completely symmetrical.
  A maximum of two series transistors in the carry-generation stack.
- Only the transistors in the carry stage have to be optimized for optimal speed. All transistors in the sum stage can be smaller.
- The transistors connected to C_i are placed closest to the output.
- Minimize the capacitance at node C_o.

25

Transmission Gate Full Adder

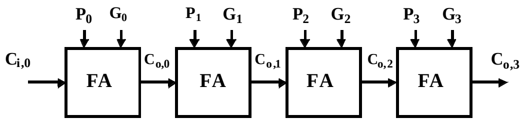


26

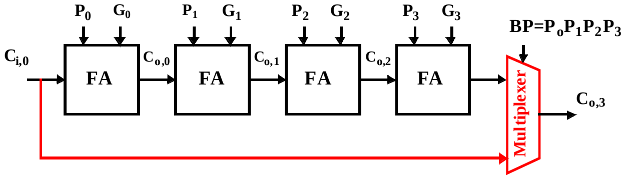
Carry Bypass Adders

Carry-Bypass Adder

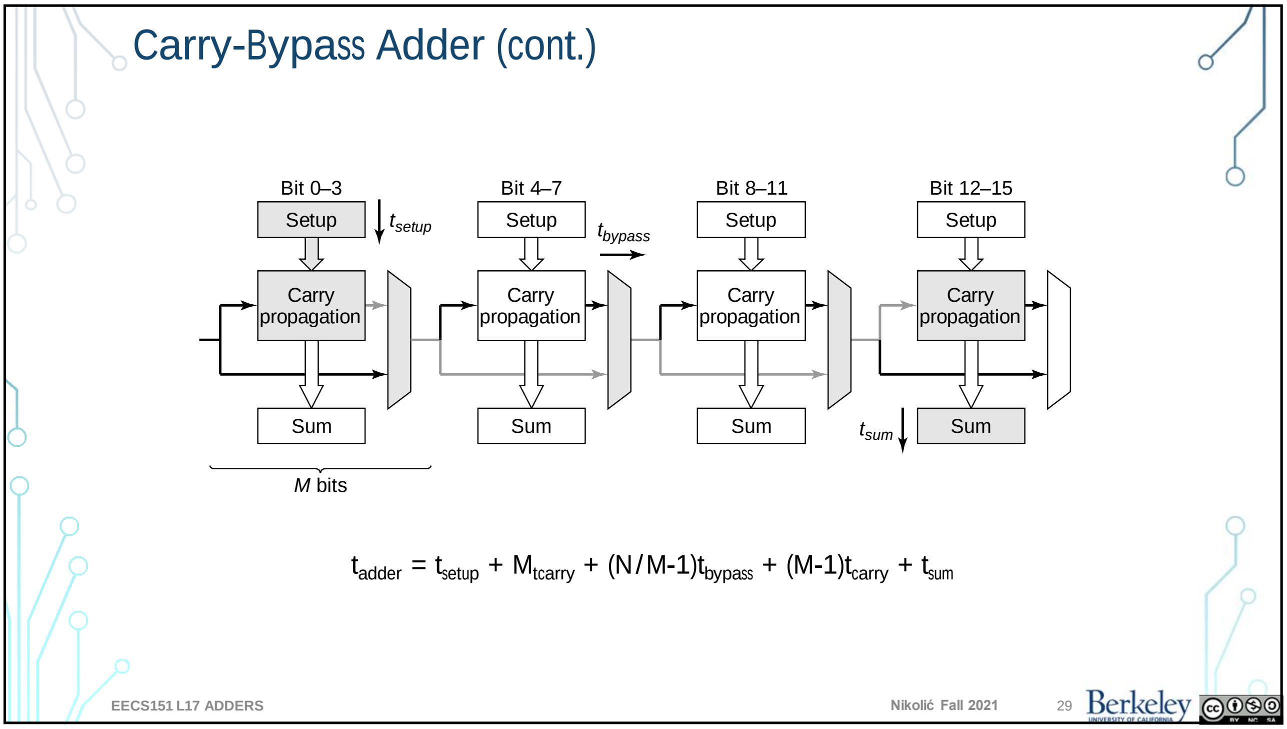
- Also called ‘carry skip’



Also called Carry-Skip

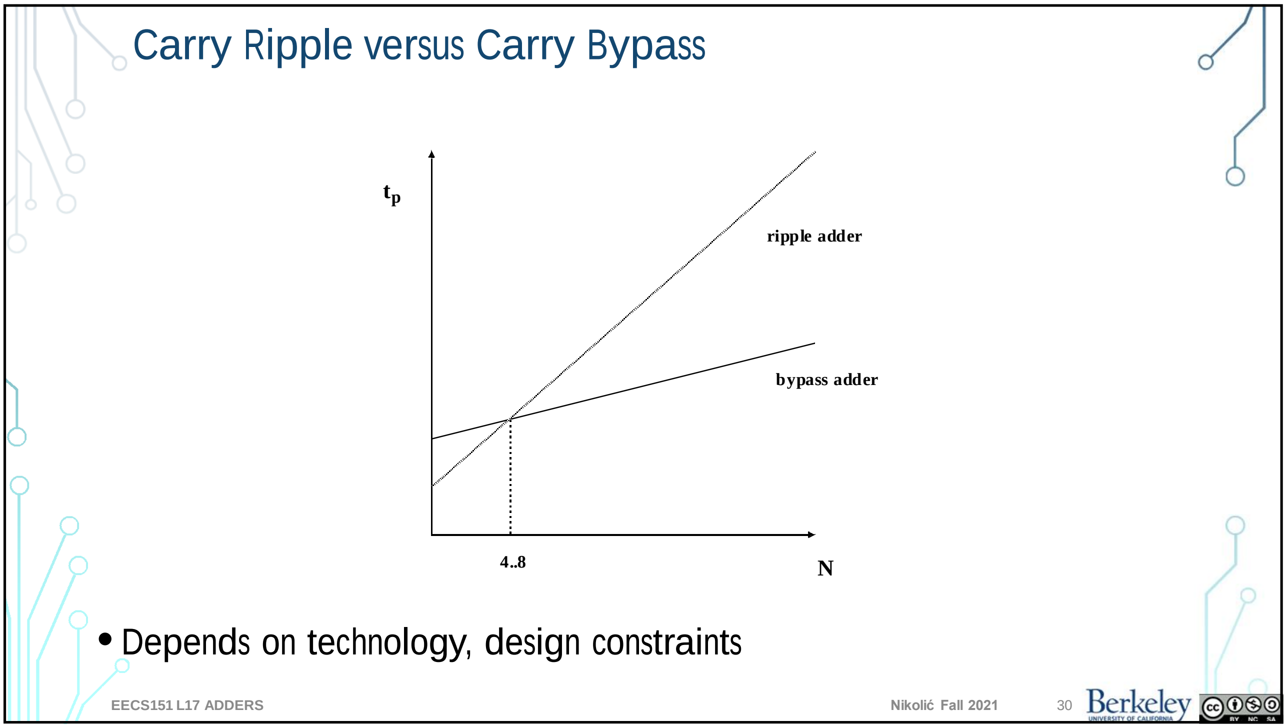


Idea: If (P0 and P1 and P2 and P3 = 1) then $C_{o3} = C_0$, else “kill” or “generate”.



Carry-Bypass Adder (cont.)

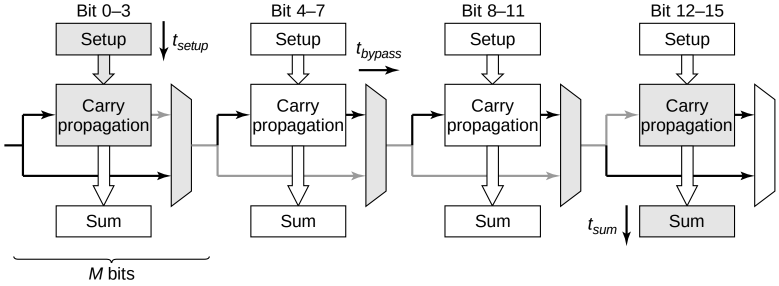
$$t_{adder} = t_{setup} + M_{tcarry} + (N/M-1)t_{bypass} + (M-1)t_{carry} + t_{sum}$$



Carry Ripple versus Carry Bypass

- Depends on technology, design constraints

To Design a Faster Carry-Bypass Adder



a) Uniform groups of 4 are optimal

b) Uniform groups >4 are optimal

c) Uniform groups <4 are optimal

d) Increasing group size with higher bit position

e) Wider groups around mid bit positions are optimal

www.yellkey.com / cover

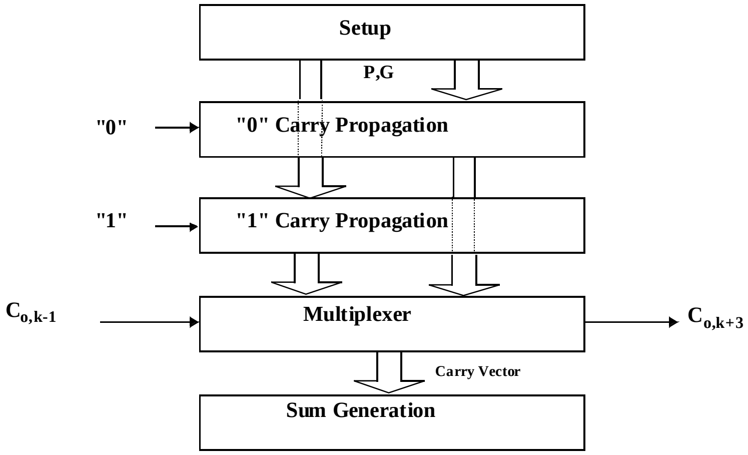
31

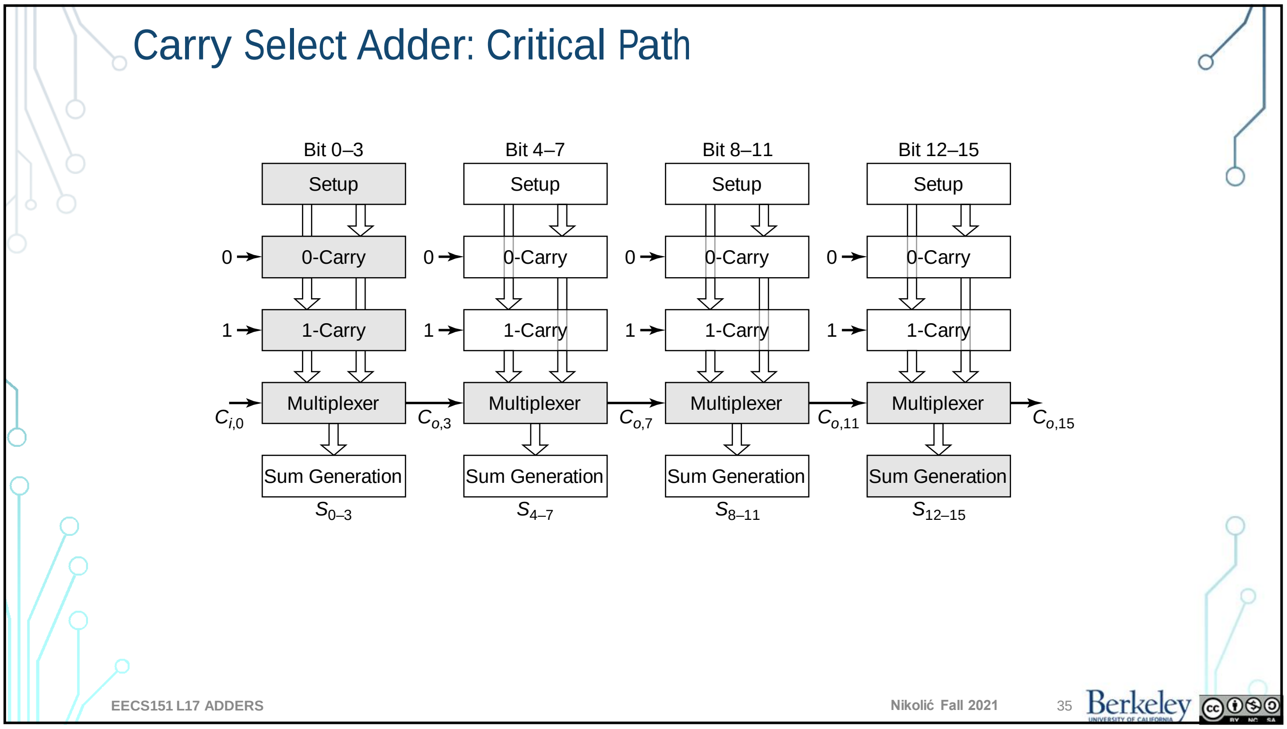
Faster Carry-Bypass

32

Carry-Select Adders

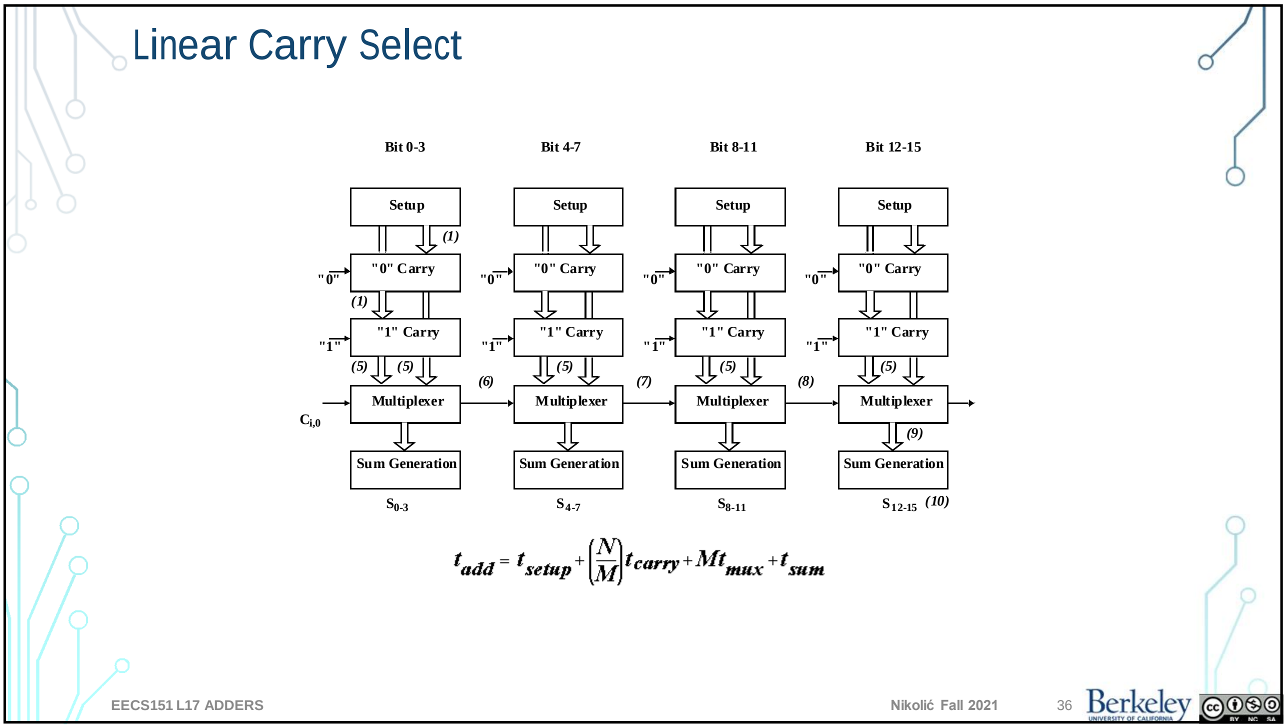
Carry-Select Adder





Carry Select Adder: Critical Path

35

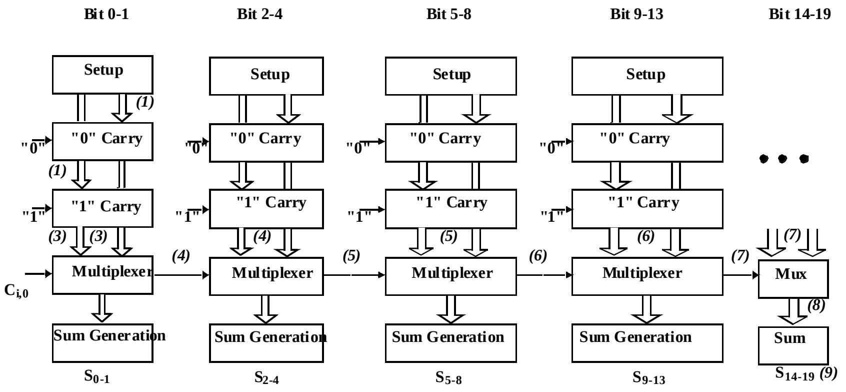


Linear Carry Select

$$t_{add} = t_{setup} + \left(\frac{N}{M}\right) t_{carry} + Mt_{mux} + t_{sum}$$

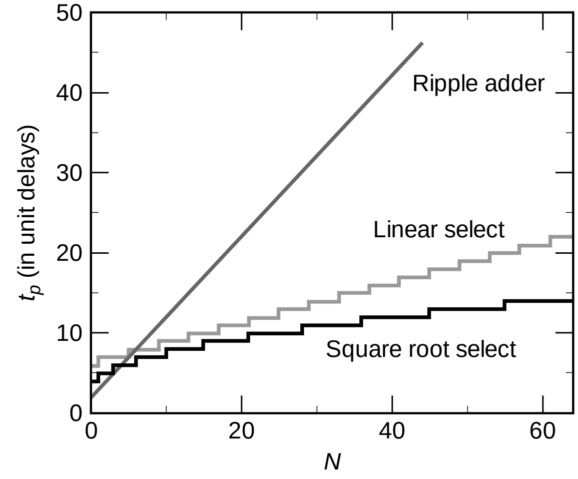
36

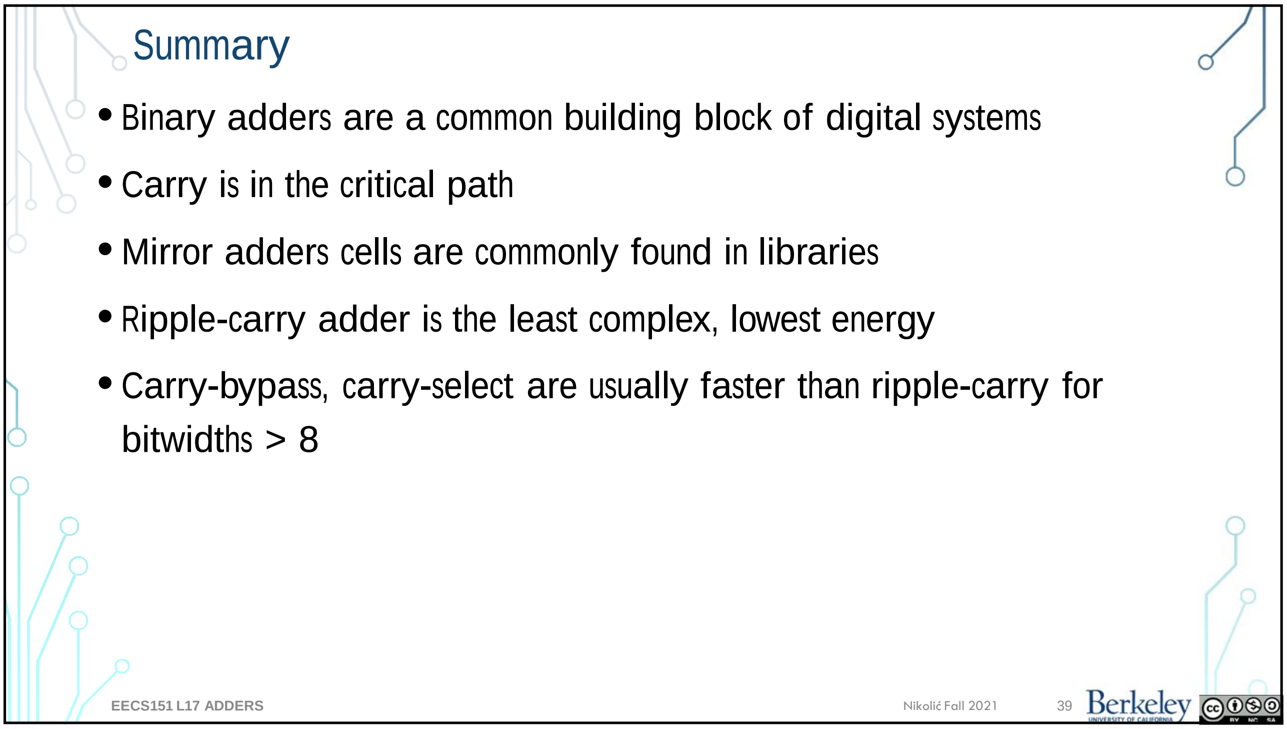
Square Root Carry Select



$$t_{add} = t_{setup} + P \cdot t_{carry} + (\sqrt{2N}) t_{mux} + t_{sum}$$

Adder Delays - Comparison





Summary

- Binary adders are a common building block of digital systems
- Carry is in the critical path
- Mirror adders cells are commonly found in libraries
- Ripple-carry adder is the least complex, lowest energy
- Carry-bypass, carry-select are usually faster than ripple-carry for bitwidths > 8