ECE4740: Digital VLSI Design

Lecture 20: Pipelining and adders

727

Semiconductor revenue grew by 22.2% in 2017

2017 Rank	2016 Rank	Vendor	2017 Revenue	2017 Market Share (%)	2016 Revenue	2016-2017 Growth (%)
1	2	Samsung Electronics	61,215	14.6	40,104	52.6
2	1	Intel	57,712	13.8	54,091	6.7
3	4	SK Hynix	26,309	6.3	14,700	79.0
4	6	Micron Technology	23,062	5.5	12,950	78.1
5	3	Qualcomm	17,063	4.1	15,415	10.7
6	5	Broadcom	15,490	3.7	13,223	17.1
7	7	Texas Instruments	13,806	3.3	11,901	16.0
8	8	Toshiba	12,813	3.1	9,918	29.2
9	17	Western Digital	9,181	2.2	4,170	120.2
10	9	NXP	8,651	2.1	9,306	-7.0
		Others	**174,418**	**41.6**	**157,736**	**10.6**
		Total Market	**419,720**	**100.0**	**343,514**	**22.2**

Source: Gartner (January 2018)

https://www.gartner.com/newsroom/id/3842666

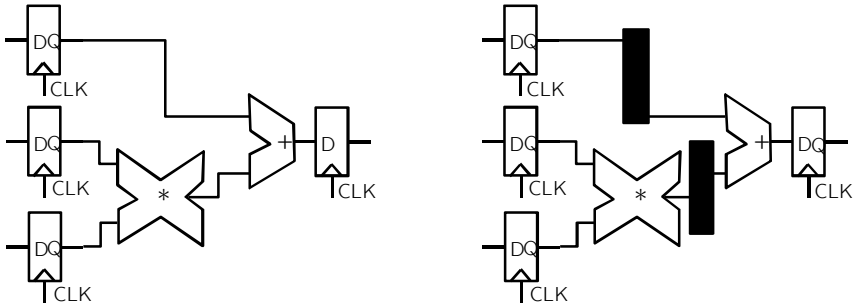
728

Recap

Pipelining and more

729

Pipelining increases throughput

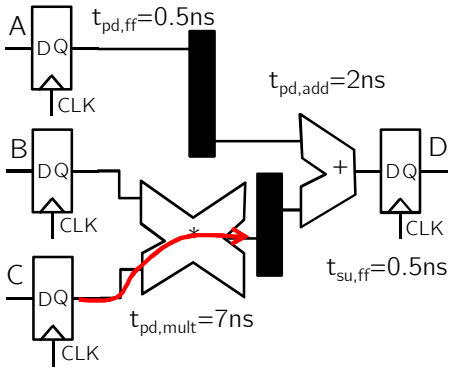


- Process same amount of data per clock cycle
- But clock frequency is higher because critical path has been reduced

730

Can we do better?

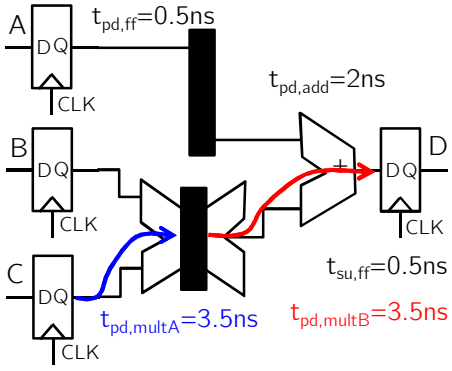
- Assume you want only 1 pipeline stage but faster clock frequency



731

Can we do better? → retiming

- Retiming helps to balance critical paths
- Not always possible ☹

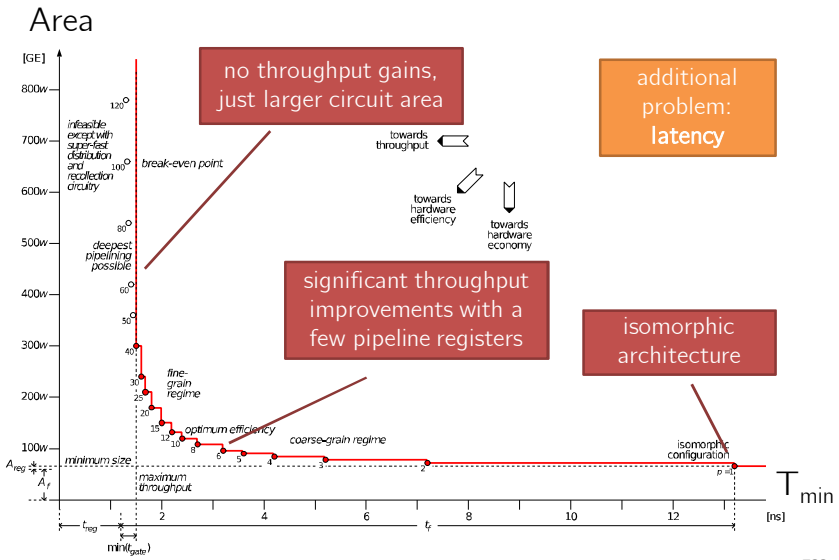


$T_{min}=t_{pd,ff}+t_{pd,multB}+t_{pd,add}+t_{su,ff}=6.5ns$

$f_{max}=153MHz$

732

Coarse grain vs. fine grain pipelining



H. Kaeslin, "Digital Integrated Circuit Design", Cambridge Univ. Press

733

Pipelining/retiming summary

- Pipelining and retiming can significantly improve throughput of a VLSI design
- For coarse-grained pipelining, hardware overhead (area) is rather small
- Pipelining suppresses glitches!
- Problems:
  - Feedback causes data-dependency problems
  - Too many registers do, in general, not help much but increase area (fine-grained pipelining)

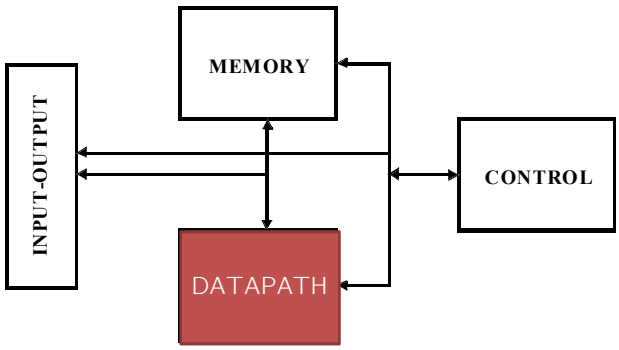
734

Very important basic building blocks

Adder circuits

735

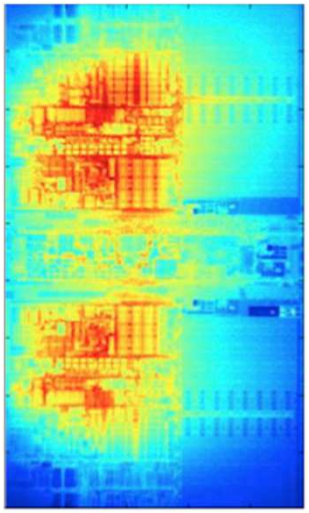
Architecture of generic processor



- Datapath contains main arithmetic units: adder, multiplier, divider, shifter, etc.

736

Processor thermal map

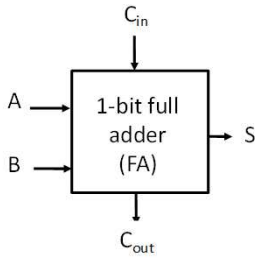


- AMD Athlon II dual core (45nm, 2.1GHz, SPEC CPU 2006)
- Highest power consumption in integer and floating-point arithmetic units

http://scale.engin.brown.edu/tools/

737

The full adder (FA)



A	B	C_{in}	C_{out}	S	carry status
0	0	0	0	0	kill (K)
0	0	1	0	1	kill (K)
0	1	0	0	1	propagate (P)
0	1	1	1	0	propagate (P)
1	0	0	0	1	propagate (P)
1	0	1	1	0	propagate (P)
1	1	0	1	0	generate (G)
1	1	1	1	1	generate (G)

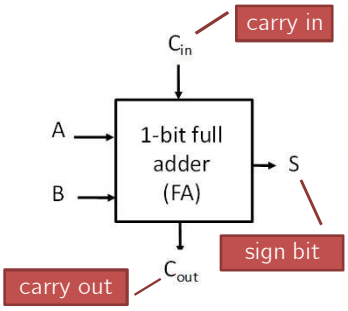
- G = A*B
- P = A⊕B
- K = !A*!B

- S = $A \oplus B \oplus C_{in} = P \oplus C_{in}$
- C_{out} = $A*B + A*C_{in} + B*C_{in}$ = $G + P*C_{in}$

738

The full adder (cont

carry signals are critical!!!!



A	B	C_{in}	C_{out}	S	carry status
0	0	0	0	0	kill (K)
0	0	1	0	1	kill (K)
0	1	0	0	1	propagate (P)
0	1	1	1	0	propagate (P)
1	0	0	0	1	propagate (P)
1	0	1	1	0	propagate (P)
1	1	0	1	0	generate (G)
1	1	1	1	1	generate (G)

- $G = A*B$
- $P = A \oplus B$
- $K = !A*!B$

independent of C_{in}

- $S = A \oplus B \oplus C_{in} = P \oplus C_{in}$
- $C_{out} = A*B + A*C_{in} + B*C_{in}$
  $= G + P*C_{in}$

majority function

739

GPK: generate, propagate, kill

A	B	C_{in}	C_{out}	S	carry status
0	0	0	0	0	kill (K)
0	0	1	0	1	kill (K)
0	1	0	0	1	propagate (P)
0	1	1	1	0	propagate (P)
1	0	0	0	1	propagate (P)
1	0	1	1	0	propagate (P)
1	1	0	1	0	generate (G)
1	1	1	1	1	generate (G)

- $G = A*B$
- $P = A \oplus B$
- $K = !A*!B$

- Generate C_{out}=1 independent of C_{in}
- $C_{out} = G + P*C_{in}$

740

GPK: generate, propagate, kill

A	B	C_{in}	C_{out}	S	carry status
0	0	0	0	0	kill (K)
0	0	1	0	1	kill (K)
0	1	0	0	1	propagate (P)
0	1	1	1	0	propagate (P)
1	0	0	0	1	propagate (P)
1	0	1	1	0	propagate (P)
1	1	0	1	0	generate (G)
1	1	1	1	1	generate (G)

- G = A*B
- P = A⊕B
- K = !A*!B

- Propagate $C_{out} = C_{in}$
- $C_{out} = G+P*C_{in}$

741

GPK: generate, propagate, kill

A	B	C_{in}	C_{out}	S	carry status
0	0	0	0	0	kill (K)
0	0	1	0	1	kill (K)
0	1	0	0	1	propagate (P)
0	1	1	1	0	propagate (P)
1	0	0	0	1	propagate (P)
1	0	1	1	0	propagate (P)
1	1	0	1	0	generate (G)
1	1	1	1	1	generate (G)

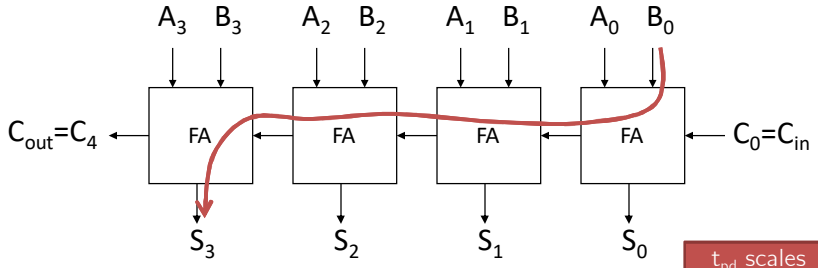
- G = A*B
- P = A⊕B
- K = !A*!B

- Kill C_{out}, i.e., $C_{out} = 0$ independent of C_{in}
- $C_{out} = G+P*C_{in} = 0 + 0*C_{in}$

742

How to build an N-bit adder?

- Simplest solution: ripple carry adder (RCA)



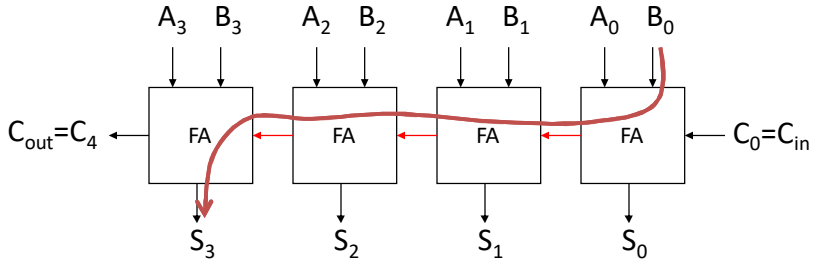
- $t_{pd,RCA} \approx (N-1)t_{pd,carry} + t_{pd,sum}$
- Too slow in most applications (e.g., in a processor where N=64 or N=128)

t_{pd} scales with O(N)

743

Ripple carry adder (RCA)

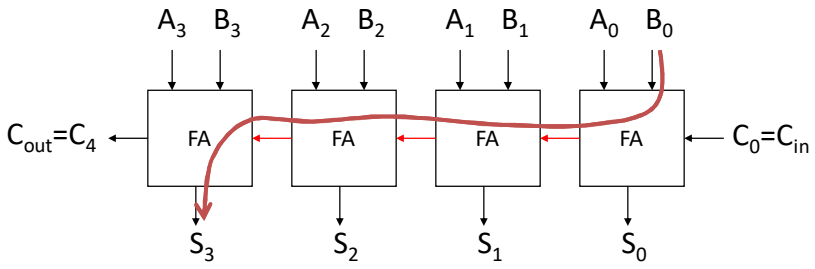
- Carry signal is critical!



- $t_{pd,RCA} \approx (N-1)t_{pd,carry} + t_{pd,sum}$
- Goal: make fastest possible carry path!

744

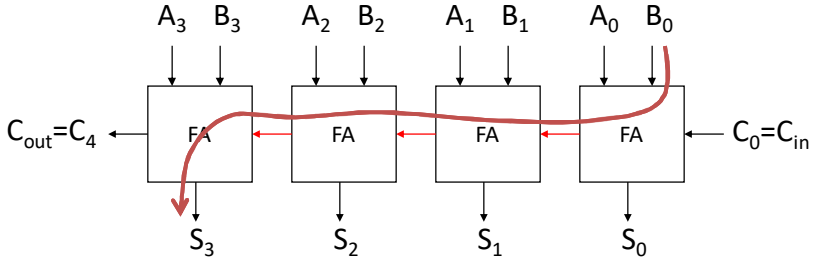
Worst/best-case input patterns?



- What is a worst-case input pattern?
- What is a best-case input pattern?

745

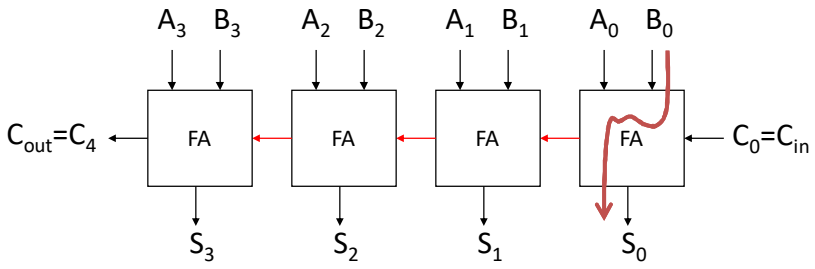
Worst-case input patterns



- Not that easy… one needs to know FA design
- One "solution": A=[0111], B=[0000], C=0
  A=[0000], B=[0001], C=0

746

Best-case input patterns



- Not that easy… one needs to know FA design
- One "solution": A=[0000], B=[0000], C=0
  A=[0000], B=[0000], C=1

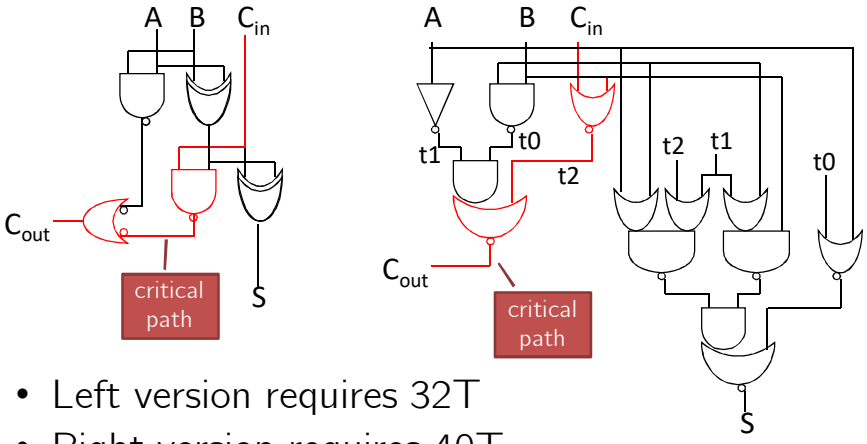
747

Some basic tricks

How to speed up adders

748

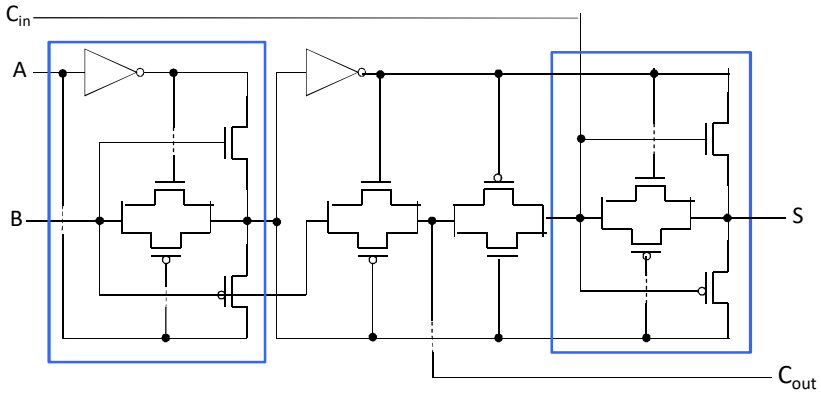
(bad) FA gate-level implementations



- Left version requires 32T
- Right version requires 40T

749

Recall the TG XOR FA

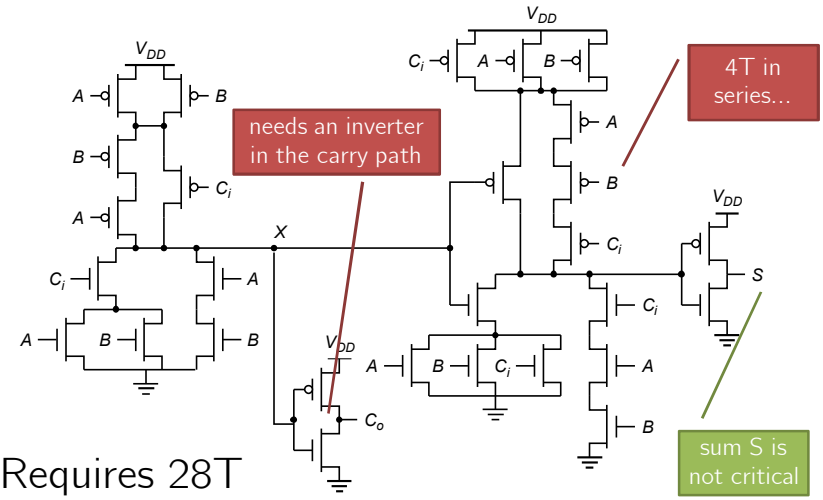


- Requires only 16T
- Circuit is not very fast (TGs in series)

Vesterbacka, Mark. "A 14-transistor CMOS full adder with full voltage-swing nodes." *Signal Processing Systems, 1999. SiPS 99. 1999 IEEE Workshop on*. IEEE, 1999.

750

Static CMOS FA



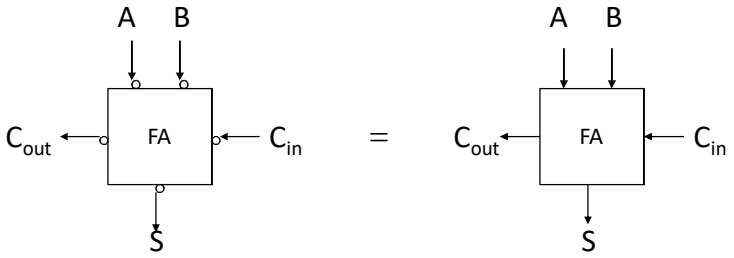
- Requires 28T

751

Image taken from: Digital Integrated Circuits (2nd Edition) by Rabaey, Chandrakasan, Nikolic

FA inversion property

- Inverting all inputs of FA results in inverted values at all outputs



752

FA inversion property (cont'd)

- Inverting all inputs of FA results in inverted values at all outputs

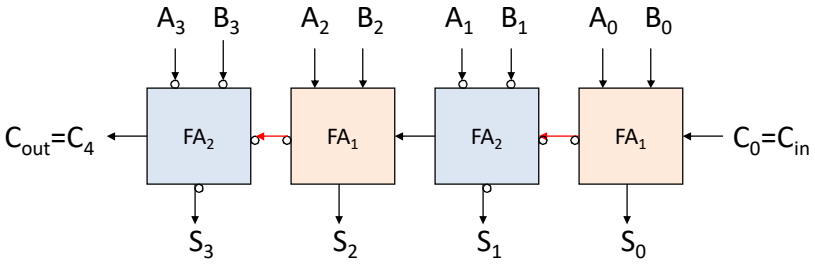
A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

A	B	C_{in}	C_{out}	S
1	1	1	1	1
1	1	0	1	0
1	0	1	1	0
1	0	0	0	1
0	1	1	1	0
0	1	0	0	1
0	0	1	0	1
0	0	0	0	0

- $S=!A \oplus !B \oplus !C_{in}$ $C_{out}=!A*!B+!A*!C_{in}+!B*!C_{in}$

753

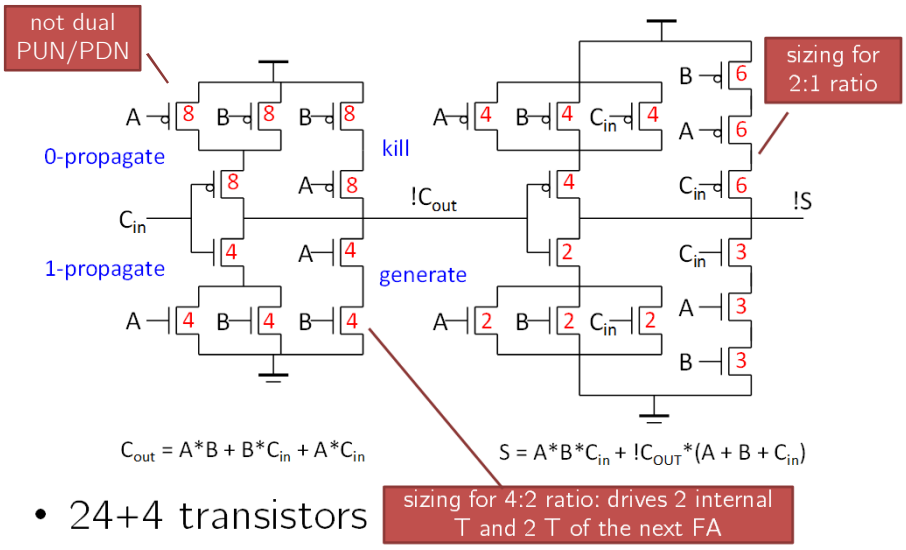
Exploiting the inversion property



- Reduces critical path in carry chain by eliminating inverters between FAs
- Requires two different FA designs

754

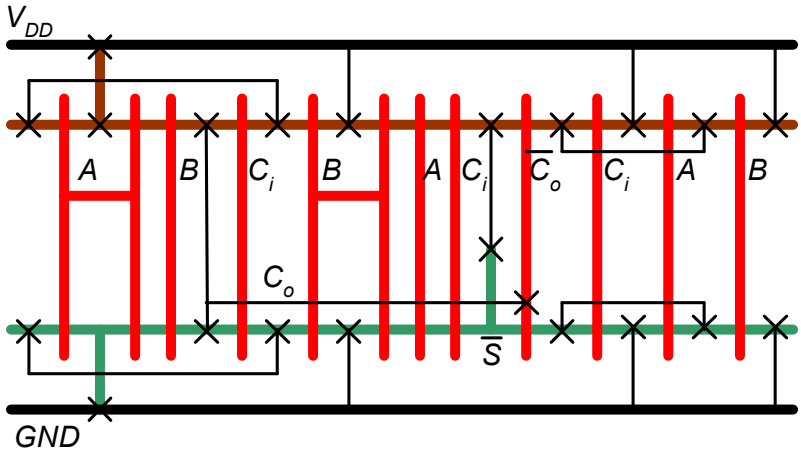
A better FA structure: mirror adder



- 24+4 transistors

755

Stick diagram of mirror adder



756

Image taken from: Digital Integrated Circuits (2nd Edition) by Rabaey, Chandrakasan, Nikolic

Features of the mirror adder

- PUN and PDN are completely symmetrical
- Proper sizing → symmetric rise & fall times
- Transistors connected to C_{in} are placed closest to output → faster
- For layout, critical to minimize capacitance at node $!C_{out}$ → use shared diffusions
- In carry circuit, maximum of 2T in series
- Only transistors in carry stage need to be optimized for speed, others → minimum size!

757