

ECE4740: Digital VLSI Design

Lecture 23: Arithmetic & logic circuits

829

Important basic functions

Comparators

830

Different comparator/detector types

- Comparators are used in virtually all digital VLSI designs, processors, GPUs, etc.
- Different types:
 - 0's detector: $A=[0000\ 0000]$
 - 1's detector: $A=[1111\ 1111]$
 - Equality comparator: $A=B$
 - Magnitude comparator: $A<B$ or $A\leq B$

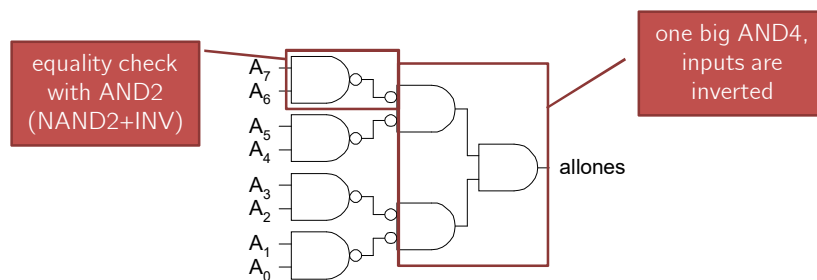


Image taken from: <http://littleandersenglish.blogspot.com/2018/01/unit-4-comparative-and-superlative.html>

831

1's and 0's detectors

- All 1's detector:



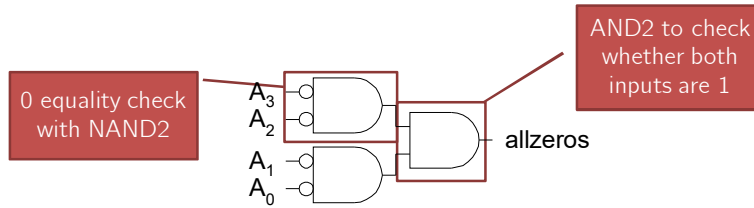
- Tree structure: $T=O(\log N)$, $A=O(N*\log N)$

Image adapted from: CMOS VLSI Design: A Circuits and Systems Perspective by Weste, Harris

832

1's and 0's detectors (cont'd)

- All 0's detector:



- Tree structure: $T=O(\log N)$, $A=O(N*\log N)$

Image adapted from: CMOS VLSI Design: A Circuits and Systems Perspective by Weste, Harris

833

Equality comparator

- Check if **individual bits are equal**
 - XNOR = equality gate
 - 1's detector on bitwise checks

A	B	XNOR(A,B)
0	0	1
0	1	0
1	0	0
1	1	1

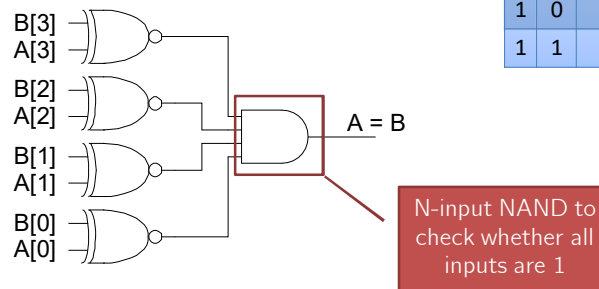


Image adapted from: CMOS VLSI Design: A Circuits and Systems Perspective by Weste, Harris

834

Magnitude comparator

- More complicated than the detectors before
- Idea: compute $B-A$ and look at sign
 - If $B-A \geq 0$ then $B \geq A$ and $B < A$ otherwise
- Two's complement identity: $B-A = B + !A + 1$
- Requires a carry propagate adder!

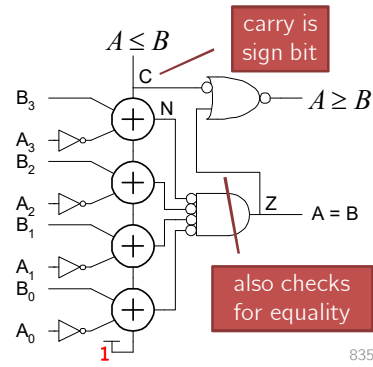


Image adapted from: CMOS VLSI Design: A Circuits and Systems Perspective by Weste, Harris

835

Recap: signed numbers

- Most common: 2's complement number format
- Use B bit to represent every integer in the range:

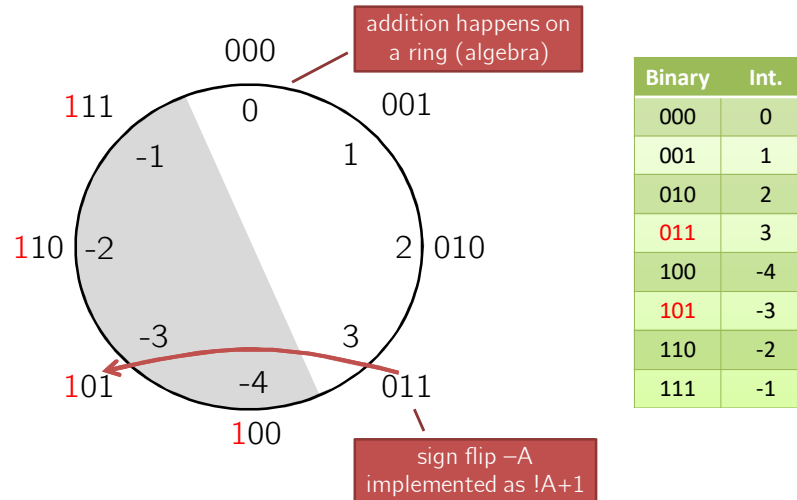
$$-2^{(B-1)} \text{ to } 2^{(B-1)} - 1$$
- Addition, subtraction, and multiplication are very simple!

Binary	Int.
000	0
001	1
010	2
011	3
100	-4
101	-3
110	-2
111	-1

MSB indicates sign of number

836

Two's complement numbers



837

Advantages of 2's complement

- Addition can be carried out with standard adder circuits (ripple, Kogge-Stone, etc.)
- Multiplication can be carried out with standard multiplier circuits
- Sign can easily be extracted (=MSB)
- Negating requires only INV+increment*
- Fixed-point numbers!

*increment can often be done with carry in at LSB

838

Other number formats

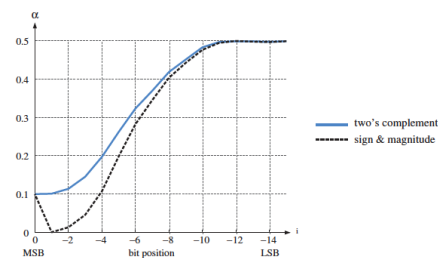
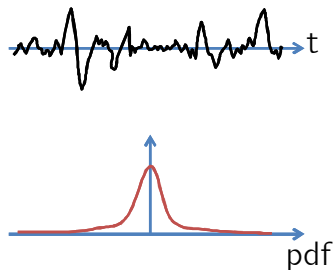
- Two's complement has an **asymmetric range** but addition/subtraction is very easy
- Alternative: **sign magnitude**
 - Stores sign and magnitude separately
 - Symmetric range
 - Two zeros... 😊
 - Addition, subtraction, and multiplication requires more complex VLSI circuits

Binary	Int.
000	0
001	1
010	2
011	3
100	-0
101	-1
110	-2
111	-3

839

Sign-magnitude can reduce power

- Example: Audio/speech signals
- **Lower switching activity!**



taken from Kaeslin, 2008

840

Signed vs. unsigned comparison

- Magnitude comparison harder for signed numbers:
 - C = carry out
 - Z = zero (all bits of B-A are 0)
 - N = negative (MSB of result)
 - V = overflow (input had different signs)
 - S = sign of result
XOR(N,V)

Condition	Unsigned	Signed
A=B	Z	Z
A!=B	!Z	!Z
A<B	C*!Z	!S*!Z
A>B	!C	S
A<=B	C	!S
A>=B	!C+Z	S+Z

841

Useful arithmetic and logic circuits

Shifters and rotators



842

Logical and arithmetic shifters

- Shifters shift bits to right or left
 - Left shift (can be multiplication by 2)
 - Right shift (can be division by 2)
 - Used in floating-point units or CORDICs (coordinate rotation digital computers)
- Inserts/extends sign bit
- | | |
|-----------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> • Logical shift right:
1011 LSR 1 = 0101 • Logical shift left:
1011 LSL 1 = 0110 | <ul style="list-style-type: none"> • Arithmetic shift/right:
1011 ASR 1 = 1101 • Arithmetic shift left:
1011 ASL 1 = 0110 |
|-----------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|

843

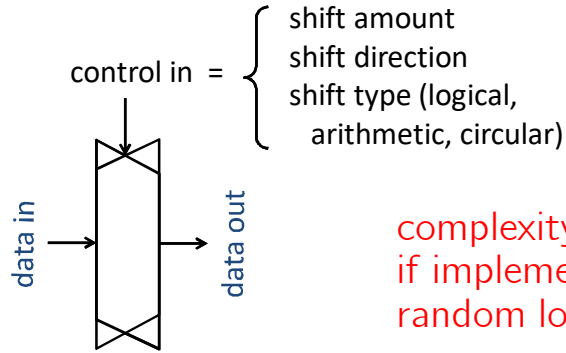
Rotators

- Shifts number to left or right and fills with lost bits on other side
 - Used for cryptography, encoding and decoding circuits, number conversion, etc.
- Rotate right: 1011 ROR 1 = 1101 
 - Rotate left: 1001 ROL 1 = 0011 

844

Programmable shifters/rotators

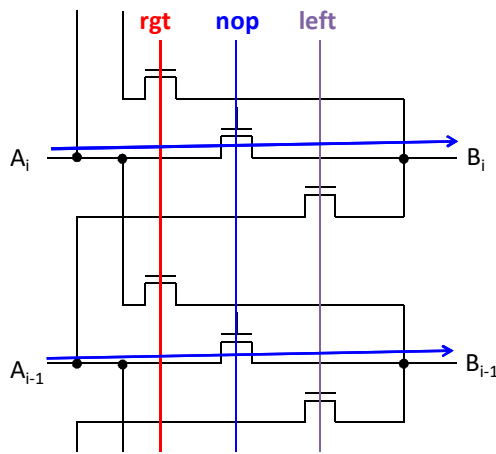
- Fixed shifters/rotators are **just wires**
- Programmable shifters have multiple modes



complexity prohibitive if implemented with random logic gates

845

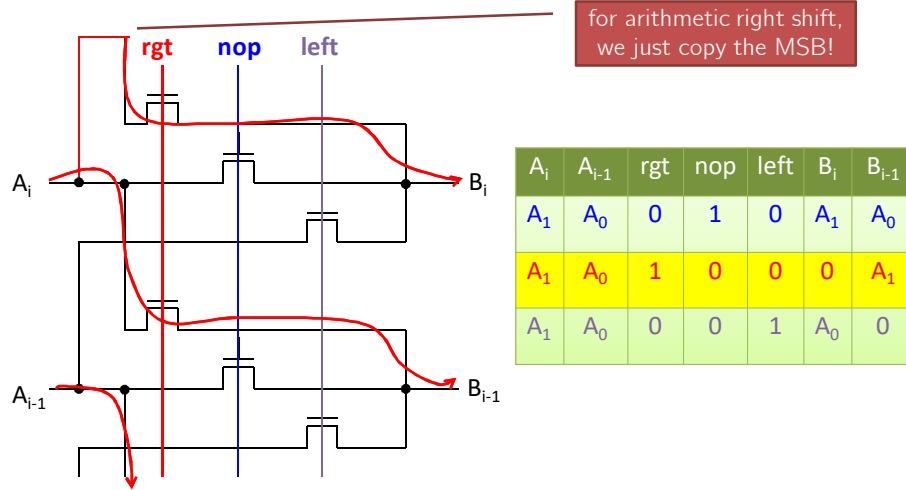
Programmable binary shifter: nop



A_i	A_{i-1}	rgt	nop	left	B_i	B_{i-1}
A_1	A_0	0	1	0	A_1	A_0
A_1	A_0	1	0	0	0	A_1
A_1	A_0	0	0	1	A_0	0

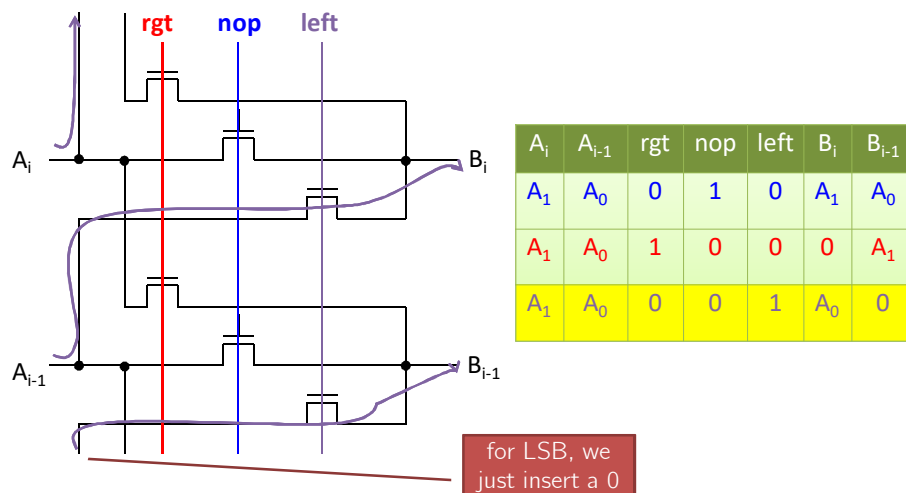
846

Programmable binary shifter: right



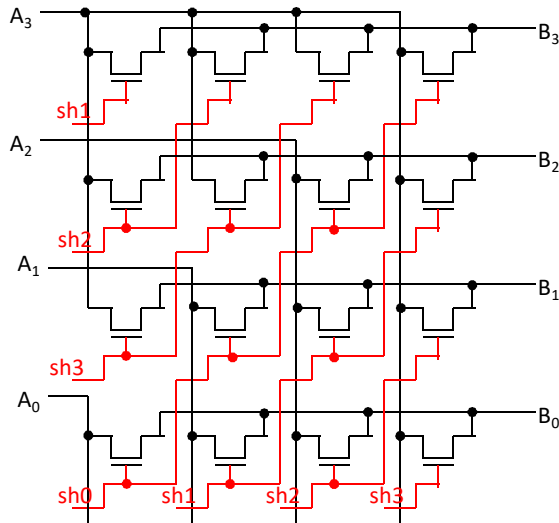
847

Programmable binary shifter: left



848

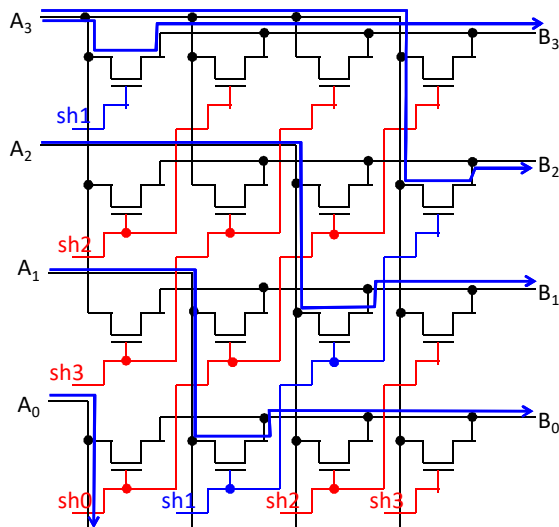
4-bit arithmetic barrel shifter



- Does sign-bit extension
- Very regular structure
- Area heavily affected by wiring

849

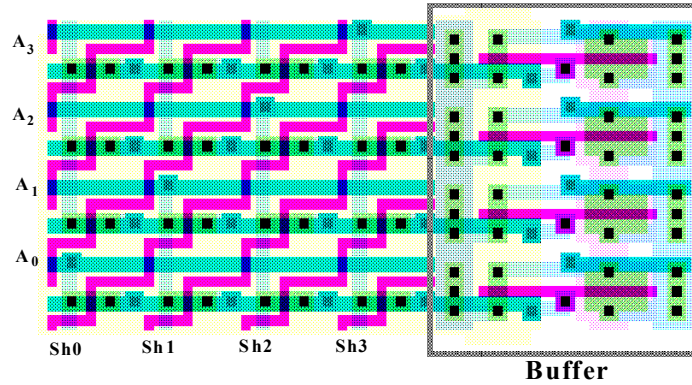
4-bit arithmetic barrel shifter (cont'd)



- How much is the output voltage drop?
- Needs buffers at each outputs

850

Barrel shifter layout

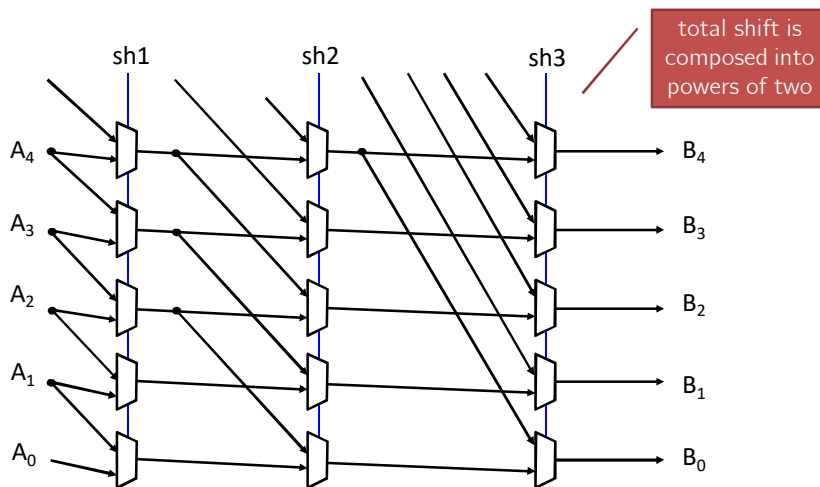


- Width $\approx 2 * p_m * N$, $N = \text{max. shift amount}$, $p_m = \text{metal pitch}$
- Delay $\approx 1\text{FET} + N \text{ diffusion capacitances} + 1\text{INV}$

851

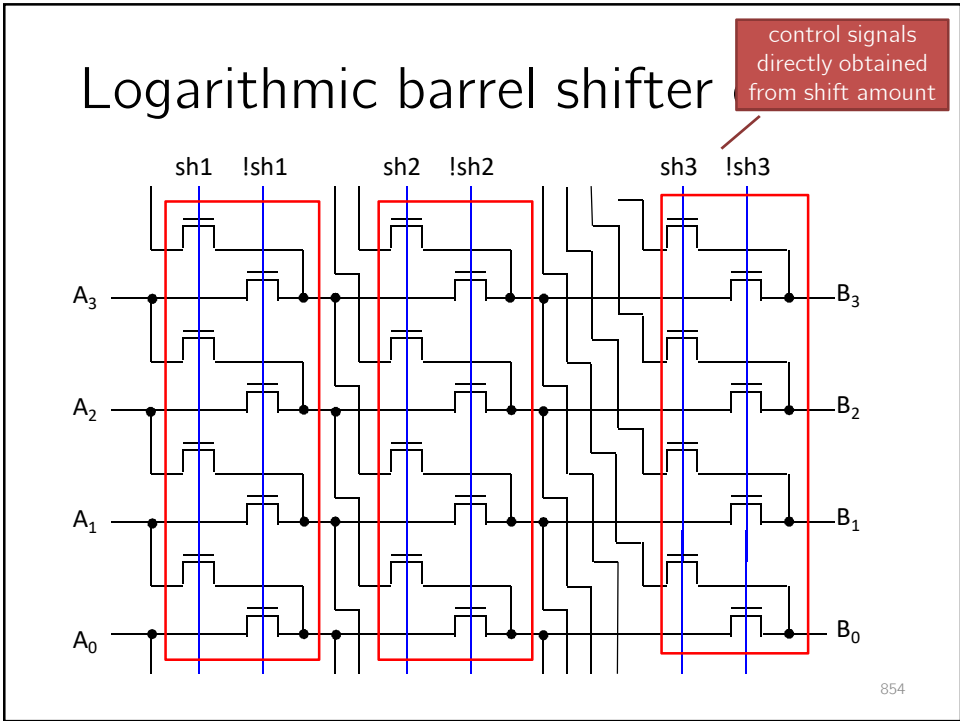
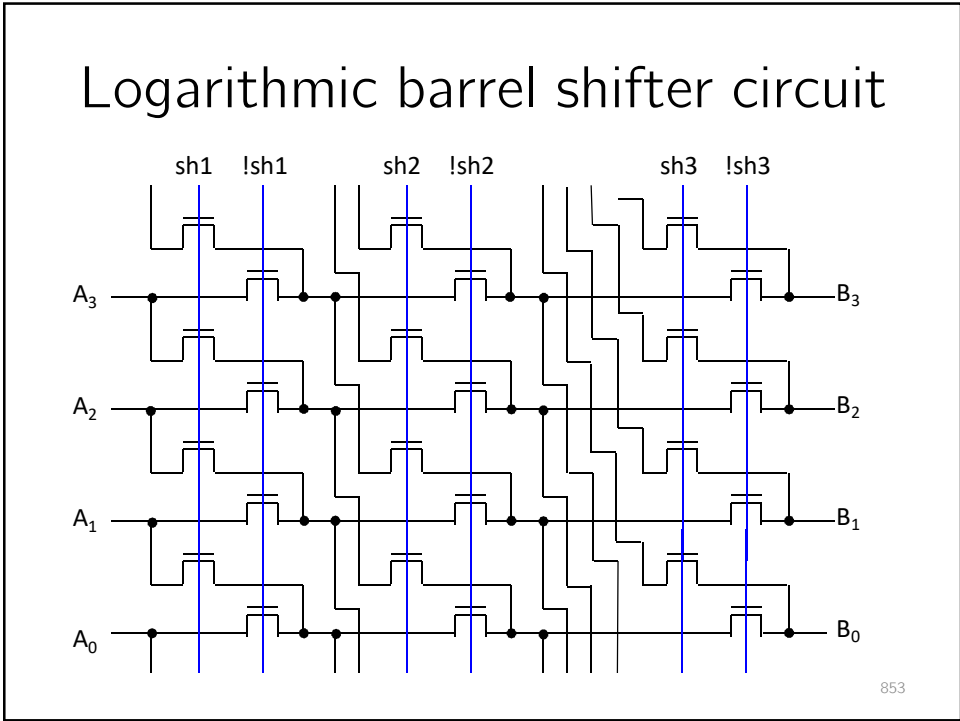
Image taken from: *An Interconnect-Centric Approach to Cyclic Shifter Design* David M. Harris Harvey Mudd College.

Logarithmic barrel shifter

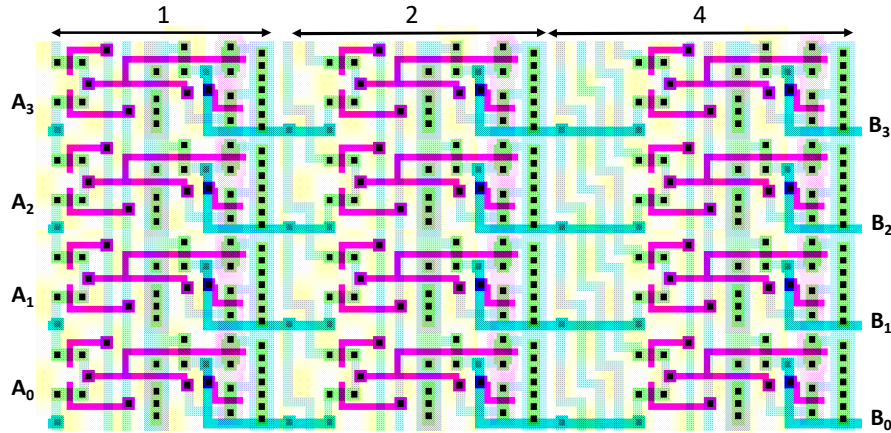


852

Image taken from: *An Interconnect-Centric Approach to Cyclic Shifter Design* David M. Harris Harvey Mudd College.



Logarithmic barrel shifter layout



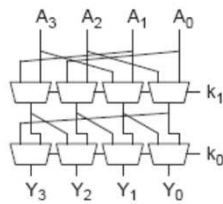
- Width $\approx p_m(2^K + 2K - 1)$, $K = \log_2(N)$
- Delay $\approx K - \text{FETs} + 2 \text{ diffusion capacitances (+1 INV)}$

855

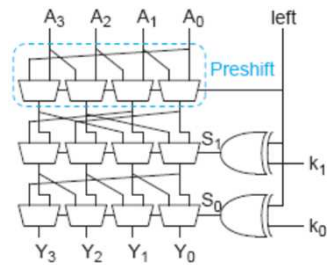
Image taken from: *An Interconnect-Centric Approach to Cyclic Shifter Design* David M. Harris Harvey Mudd College.

Logarithmic barrel rotator

- Very similar to shifter



right shift only



right and left shift

- Left rotations are right rotations by $N - k$ bit

856

Image taken from: *CMOS VLSI Design: A Circuits and Systems Perspective* by Weste, Harris

(Shifter/rotator comparison)

N	K	Barrel		Logarithmic	
		Width	Speed	Width	Speed
		$2 N p_m$	$1 + N$ diffs	$p_m(2^K+2K-1)$	$K + 2$ diffs
8	3	$16 p_m$	$1 + 8$	$13 p_m$	$3 + 2$
16	4	$32 p_m$	$1 + 16$	$23 p_m$	$4 + 2$
32	5	$64 p_m$	$1 + 32$	$41 p_m$	$5 + 2$
64	6	$128 p_m$	$1 + 64$	$75 p_m$	$6 + 2$

- Barrel better for small (faster, not much bigger)
- Logarithmic shifters always smaller and better for large shifters, **but be careful with PTs in series!**

857

Build trees!

Large multiplexers

858

Remember the TG 2-in MUX?

note that this one is inverting

$F = !((in_1 \& S) | (in_2 \& !S))$

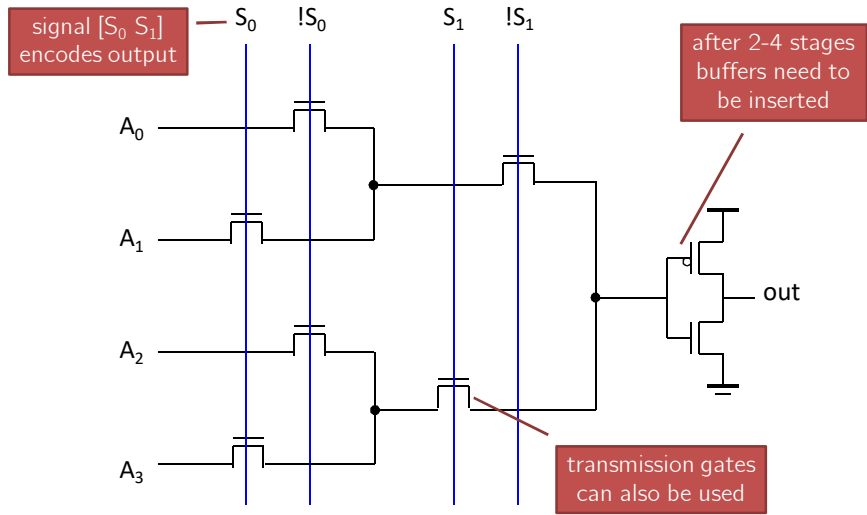
859

Building large MUXs

- Signal $[S_0 S_1]$ automatically encodes input to pass to output
- Delay grows logarithmically: $T=O(\log N)$
- Area: $A=N\log_2(N)$

860

Simpler circuit: MUX4 example



Not only used in multipliers

Multi-operand addition

862

Multi-operand addition

- Add four N-bit numbers: $\text{Sum} = A+B+C+D$
- Straightforward solution: Use 3 N-bit carry propagate adders \rightarrow large and slow

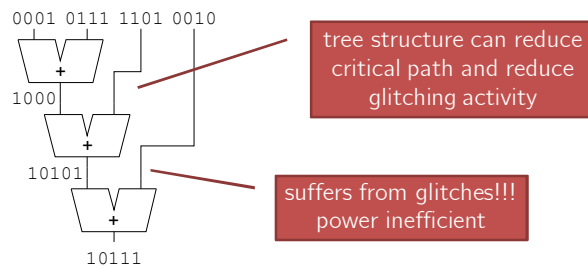


Image taken from: CMOS VLSI Design: A Circuits and Systems Perspective by Weste, Harris

863

Better: carry save adder (CSA)

- Remember: Full adder sums 3 inputs and produces 2 outputs (3:2 compressor)
 - Essentially adding three 1-bit numbers
- N full adders in parallel \rightarrow carry save adder

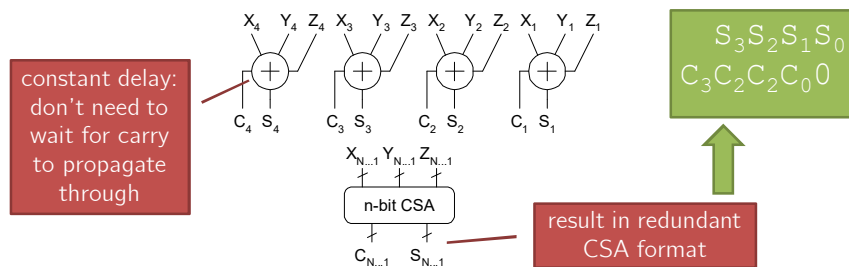


Image taken from: CMOS VLSI Design: A Circuits and Systems Perspective by Weste, Harris

864

Redundant CSA format

- Assume we compute $0110 + 0011$
- Carry propagate adders compute:
- Carry save adders compute:

$$\begin{array}{r} 0110 \\ +0011 \\ \hline =1001 \end{array}$$

carry propagated

$$\begin{array}{r} 0110 \\ +0011 \\ \hline =0121 \end{array}$$

think non-binary, redundant number format

$$\begin{array}{r} S_3 S_2 S_1 S_0 = 0101 \\ C_3 C_2 C_1 C_0 = 00100 \end{array}$$

adding these numbers → final result

865