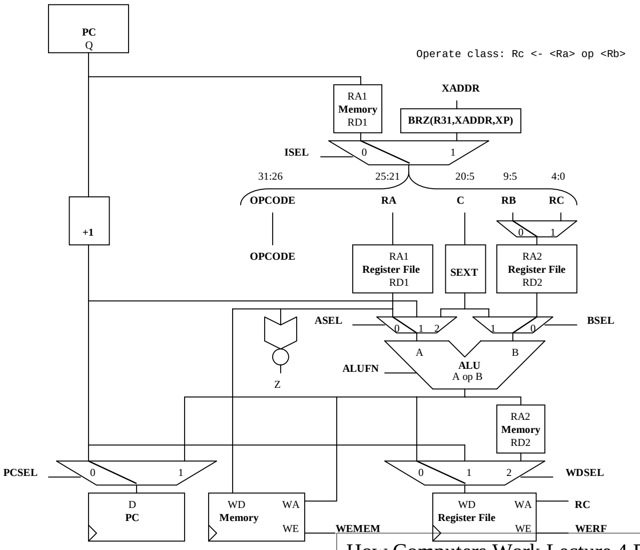
How Computers Work

Lecture 4

Computer Arithmetic

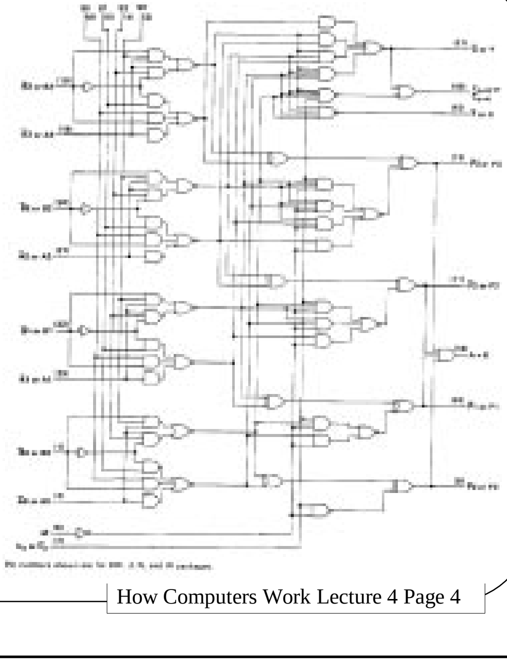
A Descending Data Flow View of the Beta



What are we going to learn today?

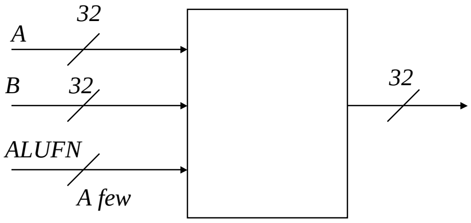
- How to build the ***A***rithmetic/***L***ogical ***U***nit
  - – Integer adder and multiplier architectures
  - – Time/Space/Cost Trade-offs

The 74181 ALU



A Wild and Crazy Idea:

- Arithmetic / Logic Unit is describable by a table:
  - ergo, we can implement it with a memory:

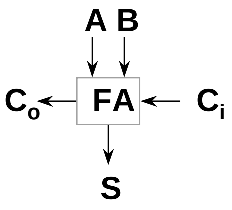


Bad Idea, because:
$2^{\sim 70}$ is a large number of rows!

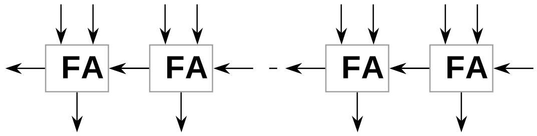
A 1 Bit Full Adder

C_i	A	B	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- Generates 1 sum bit, carry
- Can be cascaded to N bits



Ripple-carry N-bit adder:

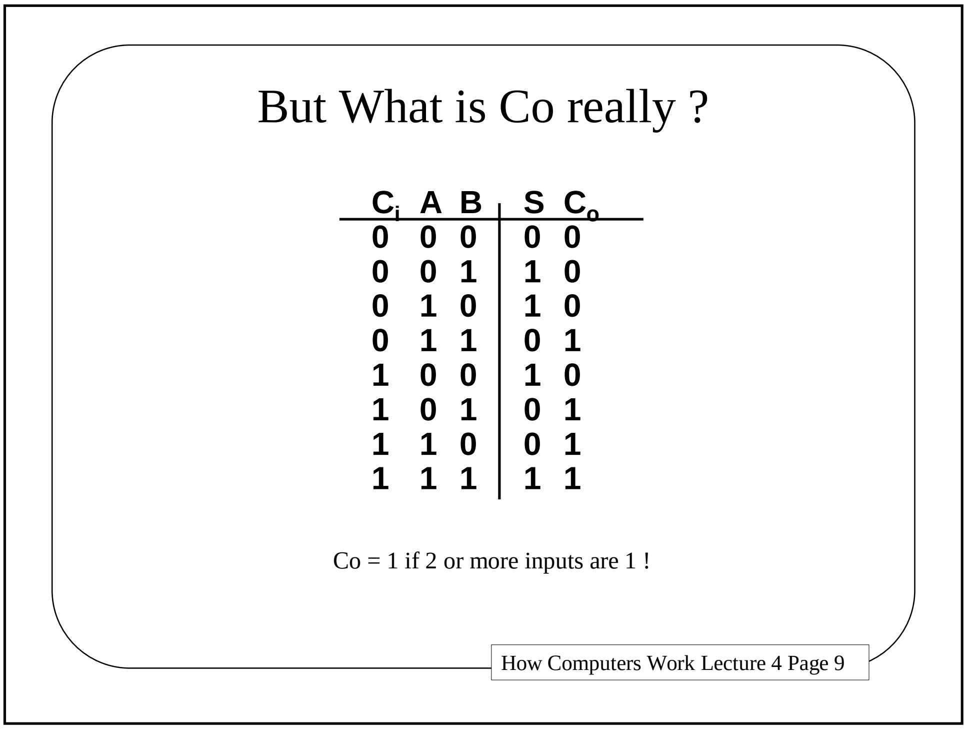


Problem: It's *Slow!*

What is Co as a function of Ci, A, B ?

C_i	A	B	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

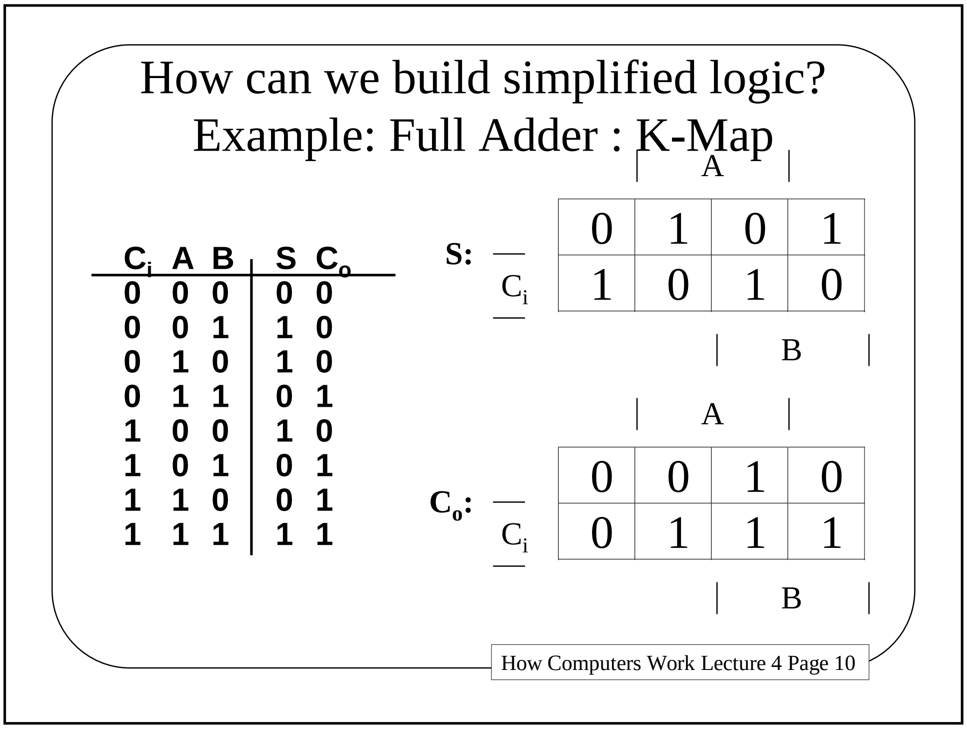
$$Co = \overline{Ci}AB + Ci\overline{A}B + CiA\overline{B} + CiAB$$



But What is Co really ?

C_i	A	B	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Co = 1 if 2 or more inputs are 1 !



How can we build simplified logic?
Example: Full Adder : K-Map

C_i	A	B	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

S:

A

	0	1	0	1
C_i	1	0	1	0

B

C_o:

A

	0	0	1	0
C_i	0	1	1	1

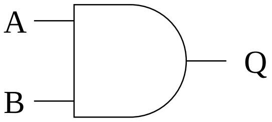
B

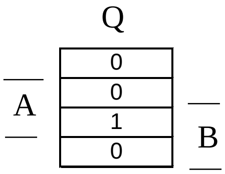
The Karnaugh Map

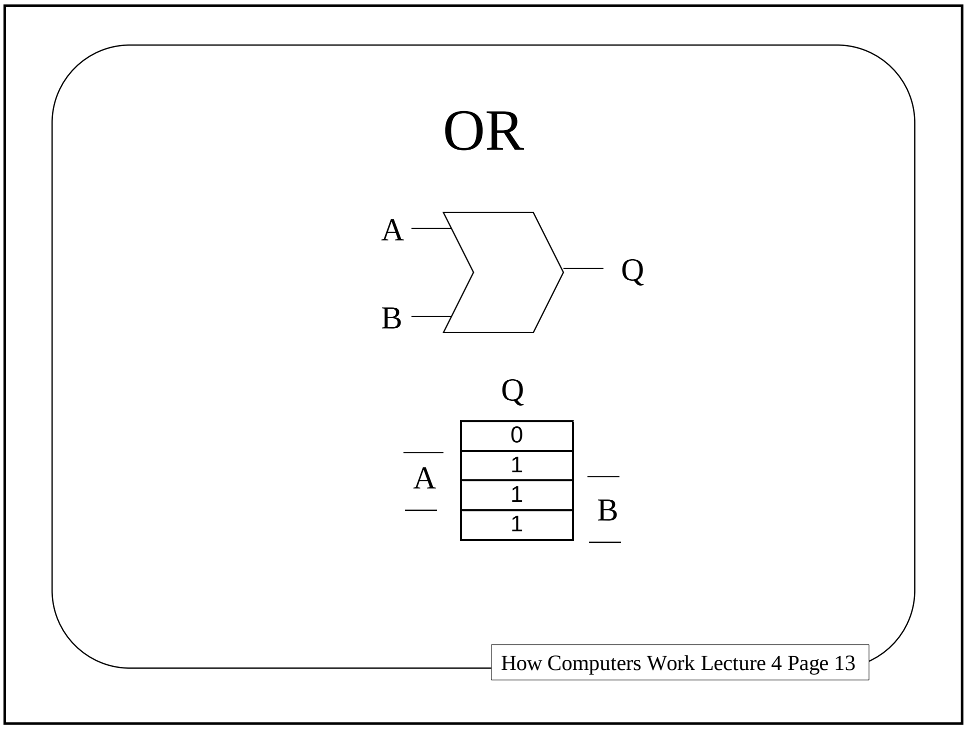
Characteristics:

1: Unit-Distance Input Labels
2: Wrap-Around

AND

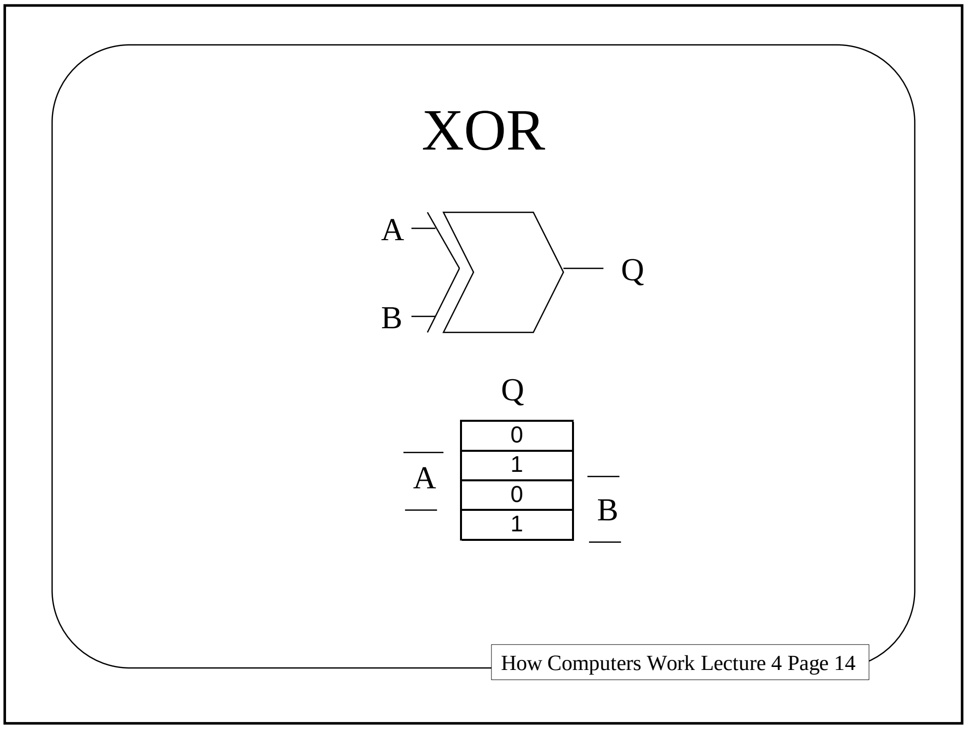






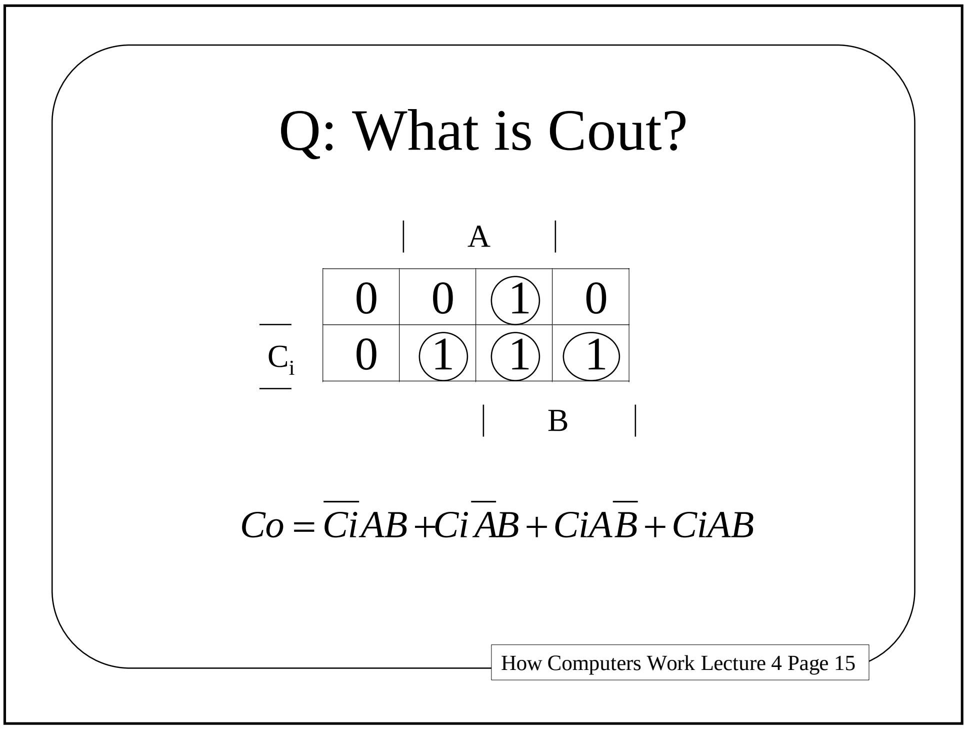
OR

	Q	
	0	
A	1	
	1	B
	1	



XOR

	Q	
	0	
A	1	
	0	B
	1	

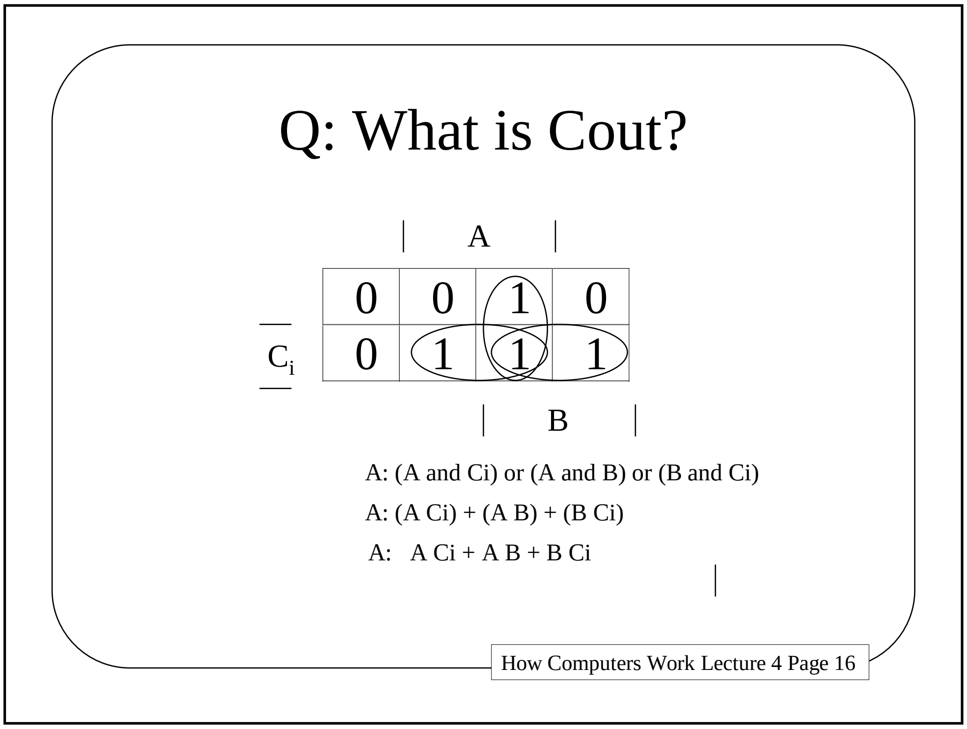


Q: What is Cout?

$$\begin{array}{c|cccc} & & A & A & \\ \hline & 0 & 0 & 1 & 0 \\ C_i & 0 & 1 & 1 & 1 \\ \hline & & & B & B \end{array}$$

$$Co = \overline{Ci}AB + Ci\overline{A}B + CiA\overline{B} + CiAB$$

How Computers Work Lecture 4 Page 15



Q: What is Cout?

$$\begin{array}{c|cccc} & & A & A & \\ \hline & 0 & 0 & 1 & 0 \\ C_i & 0 & 1 & 1 & 1 \\ \hline & & & B & B \end{array}$$

A: (A and Ci) or (A and B) or (B and Ci)

A: (A Ci) + (A B) + (B Ci)

A: A Ci + A B + B Ci

How Computers Work Lecture 4 Page 16

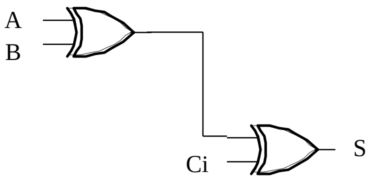
How about S ?

C_i	A	B	S
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

		A		
	0	1	0	1
C_i	1	0	1	0
			B	

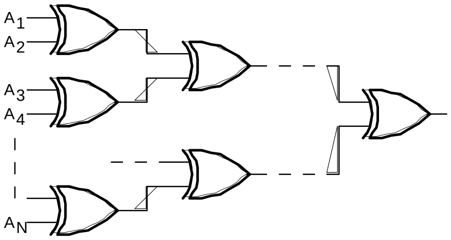
How Computers Work Lecture 4 Page 17

Parity



How Computers Work Lecture 4 Page 18

Tree Structure



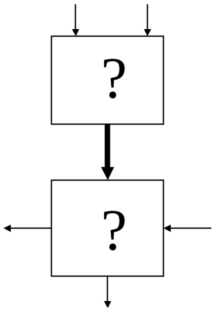
N-input TREE has O*(log(n))* levels...

Signal propagation takes O*(log(n))* gate delays.

O*(n)* gates.

An Idea !

- Speed things up by doing as much work as possible on A & B Inputs **<u>before</u>** the carry arrives:



*G*enerate and P*ropagate*

(35)

S:

	G	G		
	0	0	X	1
C_i	1	1	X	0
		P	P	

G:

	A	A	
0	0	1	0
		B	B

P:

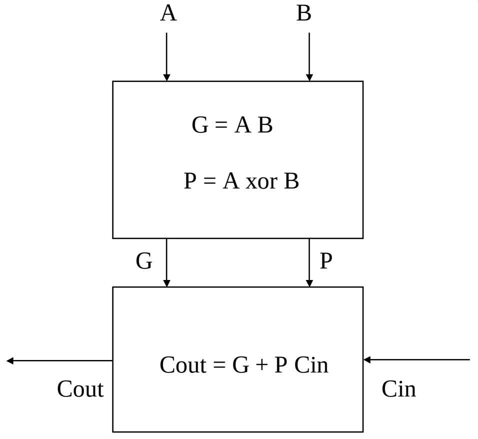
	A	A	
0	1	0	1
		B	B

(36)

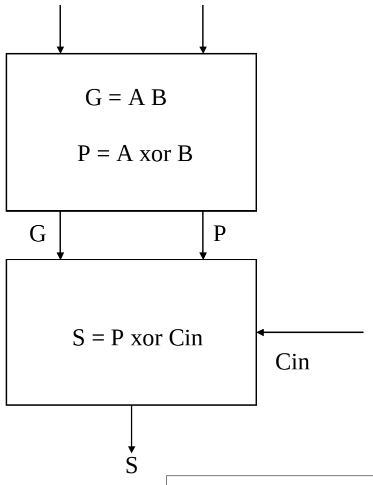
C_o:

	G	G		
	0	1	X	0
C_i	0	1	X	1
		P	P	

Implementation of C_o

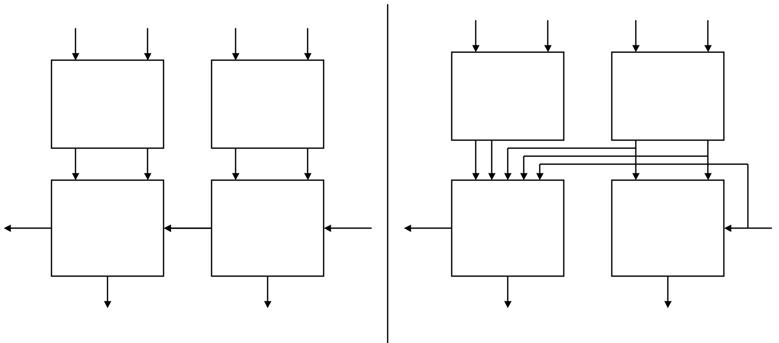


Implementation of S

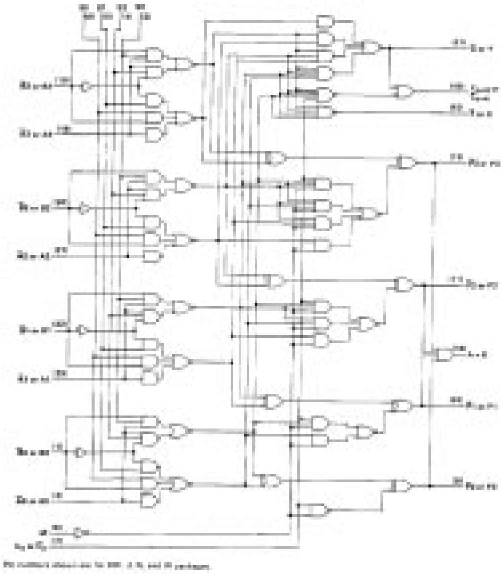


Yet Another Idea !

- Carry Look-Ahead



The 74181 ALU



How Fast Can an Adder Get ?

- Input Sensitivity Analysis: Ultimately, some bits of the answer are dependent on all bits of the inputs.
- Given an infinite number of **bounded fan-in** gates, what is the minimum growth of t_{pd} vs. the number of inputs (n)?
  - Answer: O(log(n))

Any more tricks to go faster?

- What about changing the Encoding of the inputs (i.e. base 4 !!!!!!!)
  - O(log(n)) limitation still there, but converting to a higher radix, doing the computation, then going back to binary CAN be faster than doing it naively in binary.
- How about analog computing?
  - Works, but watch out for noise.
- How about parallel computing?
  - Works, but watch out for cost.
- How about pipelined computing?
  - Q: What's a pipelined computer?
    - A: You're going to find out real soon.

Summary

- Today's Lecture:
  - How to build the ***A***rithmetic/***L***ogical ***U***nit
  - Time/Space/Cost Trade-offs
- Recitation
  - K-maps and sum-of-products form
  - Multipliers