EECS 151/251A Discussion 8

04/13/2018

Announcements

- That extra discussion with Taehwan will be in two weeks
  - Location/time TBA, slides will be available if you can’t make it.
- Homework 10 out by Sunday

Agenda

- Memories: Adders
- Your questions

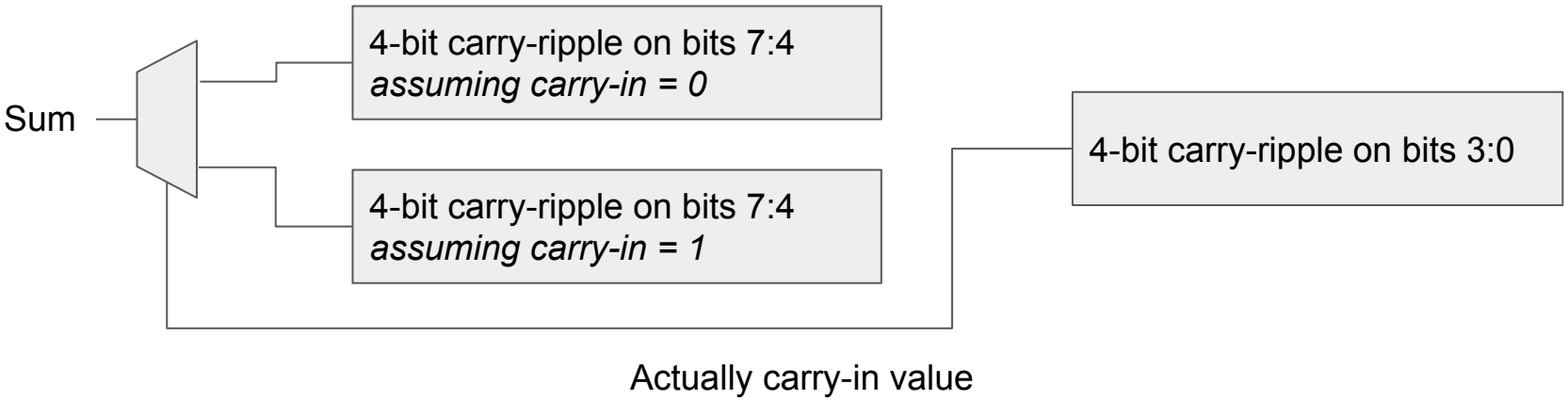
Carry-ripple adder

Problem?

Slow!

Carry-select adder

e.g. adding two 8-bit numbers



What do you need to consider when picking: size of group, number of groups?

What's best? sqrt(N) stages of sqrt(N) bits

Better (from your notes)

- Compare to ripple adder delay:

  $T_{total} = 2\ sqrt(N)\ T_{FA} - T_{FA}$, assuming $T_{FA} = T_{MUX}$

  For ripple adder $T_{total} = N\ T_{FA}$

  "cross-over" at N=3, Carry select faster for any value of N>3.

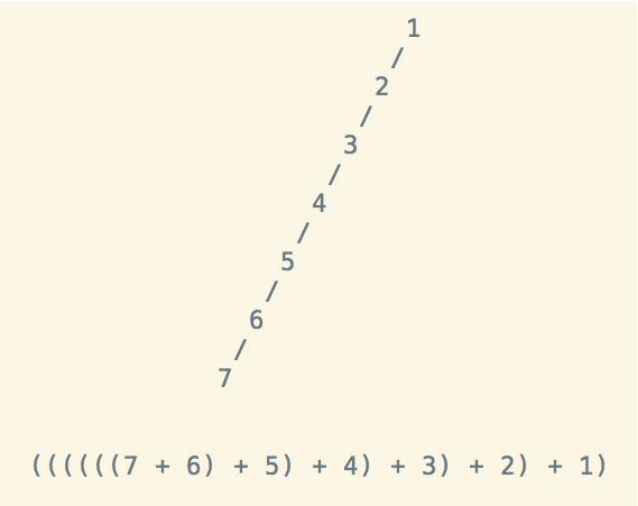
- Is sqrt(N) really the optimum?
  - From right to left increase size of each block to better match delays
  - Ex: 64-bit adder, use block sizes [12 11 10 9 8 7 7], the exact answer depends on the relative delay of mux and FA

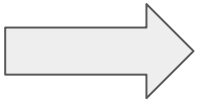
Carry-select adder

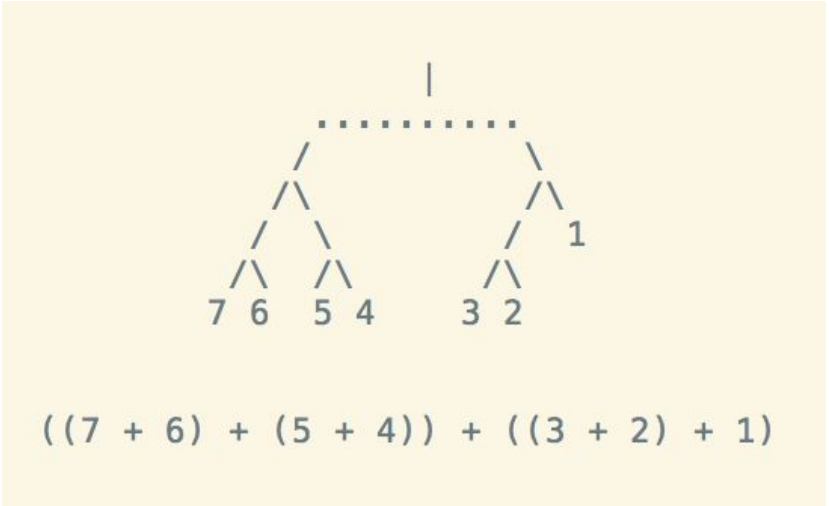
Problem?

Groups still need to propagate addition in series.

Reduction trees







6 time steps to compute

ceil(log2(6)) = 3 steps to compute

Because? Associativity of addition.

Carry-lookahead adders

- Try to turn addition into reduction operation
- Need associative computation for carry values

$a\ b\ c_i$	c_{i+1}	s	
0 0 0	0	0	*carry "kill"*
0 0 1	0	1	$k_i = a_i' b_i'$
0 1 0	0	1	
0 1 1	1	0	*carry "propagate"*
1 0 0	0	1	$p_i = a_i \oplus b_i$
1 0 1	1	0	
1 1 0	1	0	*carry "generate"*
1 1 1	1	1	$g_i = a_i b_i$

$$c_{i+1} = g_i + p_i c_i$$
$$s_i = p_i \oplus c_i$$

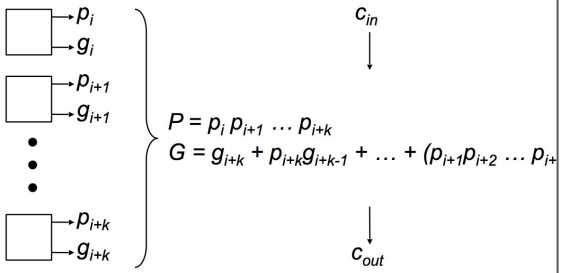
(From your notes)

But no advantage if we do this for every bit as before...

Carry-lookahead adders

Carry Look-ahead Adders

- "Group" propagate and generate signals:

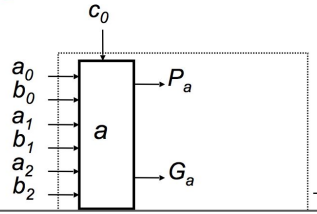


- P true if the group as a whole propagates a
- G true if the group as a whole generates a

$$c_{out} = G + Pc_{in}$$

- Group P and G can be generated hierarchi

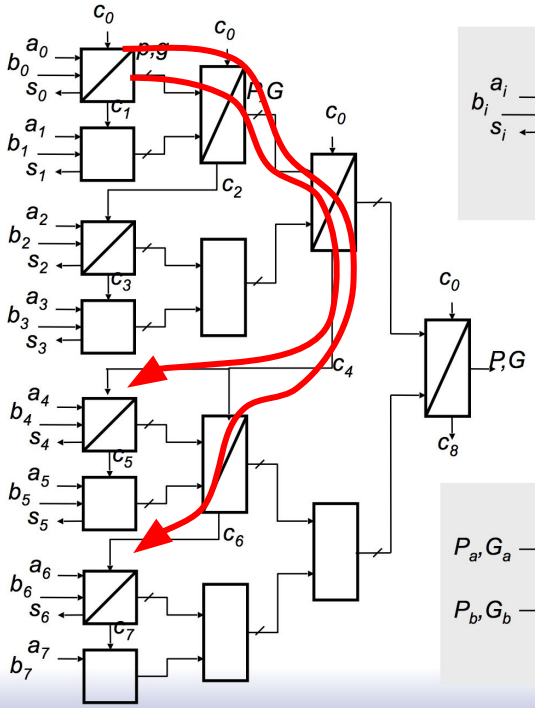
Carry Look-ahead Adders

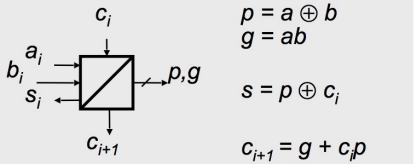


9-bit Example of hierarchically generated P and G signals:

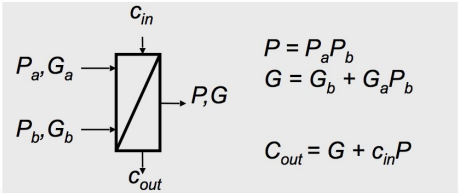
$$P = P_aP_bP_c$$

$$+ P_bP_cG_a$$





8-bit Carry Look-ahead Adder

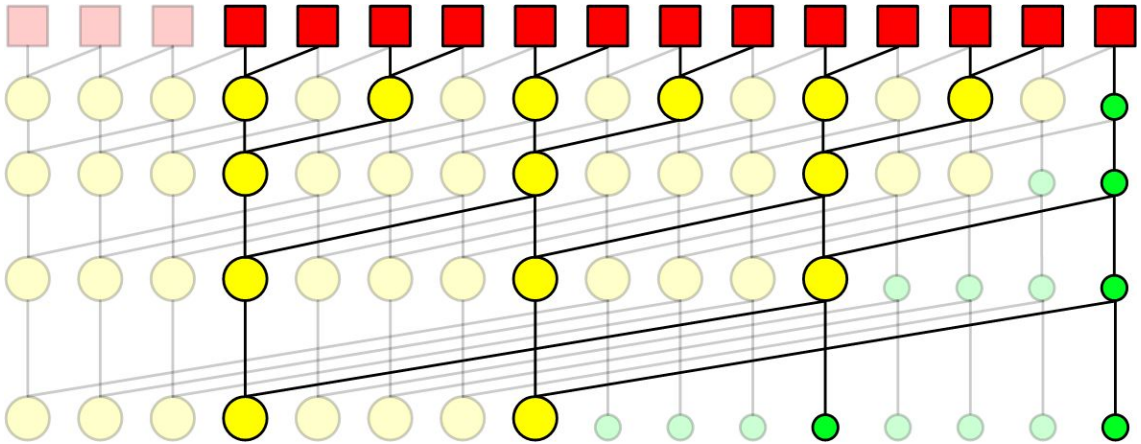


Carry-lookahead

- Use combinational logic to calculate propagate (P) and generate (G) values for groups of bits
- Provide generated values to other groups well before their sum would ripple through
- Can combine with other techniques depending on specific design

Parallel-prefix adders

- Generate carries for each bit directly; don’t do any grouping there
- E.g. Kogge-Stone adder (radix 2, sparsity 4)



- (each line is a bit, each level is a successive computation)
- Each of these carries can feed a carry-select or carry-ripple (or other) adder (shown by transparency)

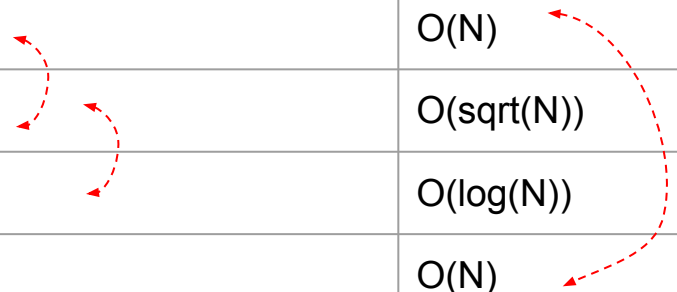
Bit serial adders

- Draw one
- Pros?
  - Simple, require very few components, can feed other serial computations
- Cons?
  - Take ~N cycles

Adder summary

Where N is the operand width in bits:

Adder	Cost	Delay
Ripple	O(N)	O(N)
Carry-select	O(N)	O(sqrt(N))
Carry-lookahead	O(N)	O(log(N))
Bit-serial	O(1)	O(N)



Usual O-notation caveat: constants are hidden, but here may be important. **See notes!** *Lecture 20, slide 33.*

References

- Lecture notes
- Wikipedia