

ECE 274 Digital Logic

Optimization and Tradeoffs

Carry-Lookahead Adders

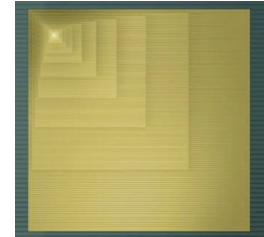
Digital Design 6.4



Digital Design

Chapter 6: Optimization and Tradeoffs

Slides to accompany the textbook *Digital Design*, First Edition,
by Frank Vahid, John Wiley and Sons Publishers, 2007.
<http://www.dvvhid.com>

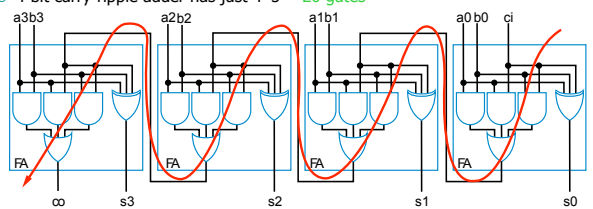
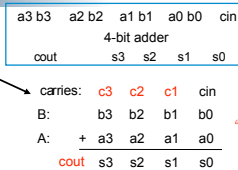


Copyright © 2007 Frank Vahid

Instructors of courses requiring Vahid's *Digital Design* textbook (published by John Wiley and Sons) have permission to modify and use these slides for customary course-related activities, subject to keeping this copyright notice in place and unmodified. These slides may be posted as ~~unmodified~~ pdf versions on publicly-accessible course websites. PowerPoint source (or pdf with animations) may ~~not~~ be posted to publicly-accessible websites, but may be posted for students on internal protected sites or distributed directly to students by other electronic means. Instructors may make printouts of the slides available to students for a reasonable photocopying charge, without incurring royalties. Any other use requires explicit permission. Instructors may obtain PowerPoint source or obtain special use permissions from Wiley – see <http://www.dvvhid.com> for information.

Carry-Lookahead Adder Faster Adder

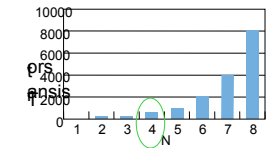
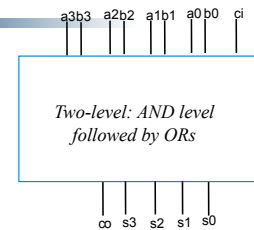
- Built carry-ripple adder in Ch 4
 - Similar to adding by hand, column by column
 - Con: Slow
 - Output is not correct until the carries have rippled to the left
 - 4-bit carry-ripple adder has $4 \times 2 = 8$ gate delays
 - Pro: Small
 - 4-bit carry-ripple adder has just $4 \times 5 = 20$ gates



3

Carry-Lookahead Adder Faster Adder

- Faster adder – Use two-level combinational logic design process
 - Recall that 4-bit two-level adder was big
 - Pro: Fast
 - 2 gate delays
 - Con: Large
 - Truth table would have $2^{(4+4)} = 256$ rows
 - Plot shows 4-bit adder would use about 500 gates
- Is there a compromise design?
 - Between 2 and 8 gate delays
 - Between 20 and 500 gates



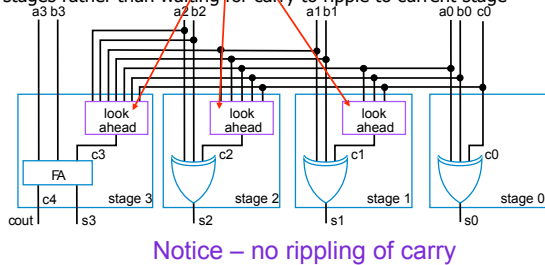
4

Carry-Lookahead Adder

Faster Adder – (Bad) Attempt at "Lookahead"

Idea

- Modify carry-ripple adder – For a stage's carry-in, don't wait for carry to ripple, but rather *directly compute* from inputs of earlier stages
- Called "lookahead" because current stage "looks ahead" at previous stages rather than waiting for carry to ripple to current stage

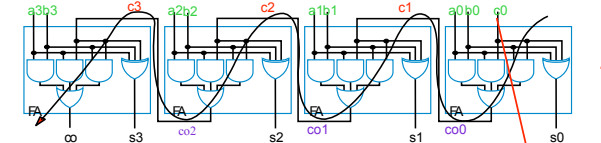


5

Carry-Lookahead Adder

Faster Adder – (Bad) Attempt at "Lookahead"

- Want each stage's carry-in bit to be function of external inputs only (a's, b's, or c0)



- Recall full-adder equations:

$$s = a \text{ xor } b$$

$$c = bc + ac + ab$$

Stage 0: Carry-in is already an external input: c_0

$$\text{Stage 1: } c_1 = c_0$$

$$c_0 = b_0c_0 + a_0c_0 + a_0b_0$$

$$c_1 = b_0c_0 + a_0c_0 + a_0b_0$$

$$\text{Stage 2: } c_2 = c_1$$

$$c_1 = b_1c_1 + a_1c_1 + a_1b_1$$

$$c_2 = b_1c_1 + a_1c_1 + a_1b_1$$

$$c_2 = b_1(b_0c_0 + a_0c_0 + a_0b_0) + a_1(b_0c_0 + a_0c_0 + a_0b_0) + a_1b_1$$

$$c_2 = b_1b_0c_0 + b_1a_0c_0 + b_1a_0b_0 + a_1b_0c_0 + a_1a_0c_0 + a_1a_0b_0 + a_1b_1$$

Continue for c_3

6

Carry-Lookahead Adder

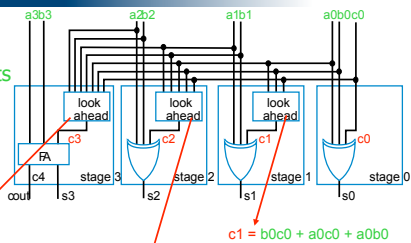
Faster Adder – (Bad) Attempt at "Lookahead"

- Carry lookahead logic function of external inputs

- No waiting for ripple

Problem

- Equations get too big
- Not efficient
- Need a better form of lookahead



$$c_1 = b_0c_0 + a_0c_0 + a_0b_0$$

$$c_2 = b_1b_0c_0 + b_1a_0c_0 + b_1a_0b_0 + a_1b_0c_0 + a_1a_0c_0 + a_1a_0b_0 + a_1b_1$$

$$c_3 = b_2b_1b_0c_0 + b_2b_1a_0c_0 + b_2b_1a_0b_0 + b_2a_1b_0c_0 + b_2a_1a_0c_0 + b_2a_1a_0b_0 + b_2a_1b_1 + a_2b_1b_0c_0 + a_2b_1a_0c_0 + a_2b_1a_0b_0 + a_2a_1b_0c_0 + a_2a_1a_0c_0 + a_2a_1a_0b_0 + a_2a_1b_1 + a_2b_2$$

7

Carry-Lookahead Adder

Better Form of Lookahead

- Have each stage compute two terms

- Propagate:** $P = a \text{ xor } b$

- Generate:** $G = ab$

- Compute lookahead from P and G terms, not from external inputs

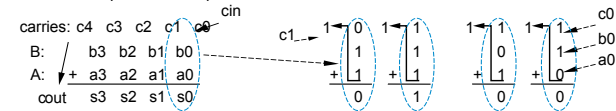
- Why P & G ? Because the logic comes out much simpler

- Very clever finding; not particularly obvious though

- Why those names?

- G : If a and b are 1, carry-out will be 1 – "generate" a carry-out of 1 in this case

- P : If only one of a or b is 1, then carry-in will equal the carry-in – propagate the carry-in to the carry-out in this case

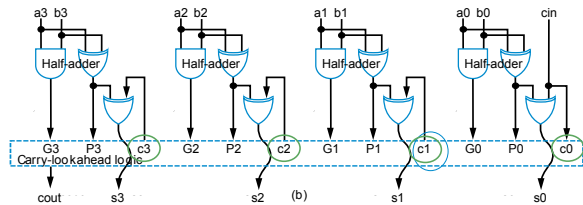


(a) if $a_0b_0 = 1$ then $c_1 = 1$
if $a_0 \text{ xor } b_0 = 1$ then $c_1 = 1$ if $c_0 = 1$
(call this G:Generate) (call this P:Propagate)

8

Carry-Lookahead Adder

Better Form of Lookahead



- With P & G , the carry lookahead equations are much simpler

- Equations before plugging in

- $c1 = G0 + P0c0$
- $c2 = G1 + P1c1$
- $c3 = G2 + P2c2$
- $cout = G3 + P3c3$

After plugging in:

$$c1 = G0 + P0c0$$

$$c2 = G1 + P1c1 = G1 + P1(G0 + P0c0)$$

$$c2 = G1 + P1G0 + P1P0c0$$

$$c3 = G2 + P2c2 = G2 + P2(G1 + P1G0 + P1P0c0)$$

$$c3 = G2 + P2G1 + P2P1G0 + P2P1P0c0$$

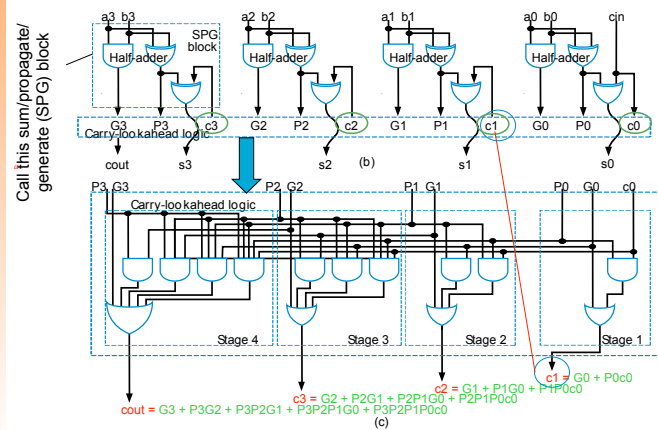
$$cout = G3 + P3G2 + P3P2G1 + P3P2P1G0 + P3P2P1P0c0$$

Much simpler than the "bad" lookahead

9

Carry-Lookahead Adder

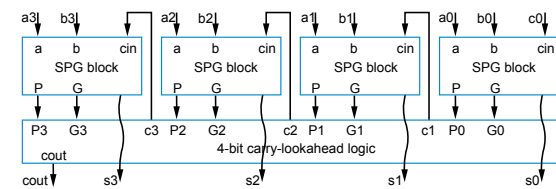
Better Form of Lookahead



10

Carry-Lookahead Adder

Carry-Lookahead Adder -- High-Level View



- Fast -- only 4 gate delays

- Each stage has SPG block with 2 gate levels
- Carry-lookahead logic quickly computes the carry from the propagate and generate bits using 2 gate levels inside

- Reasonable number of gates -- 4-bit adder has only 26 gates

- 4-bit adder comparison (gate delays, gates)
 - Carry-ripple: (8, 20)
 - Two-level: (2, 500)
 - CLA: (4, 26)
 - Nice compromise

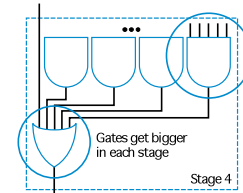
11

Carry-Lookahead Adder

Carry-Lookahead Adder -- 32-bit?

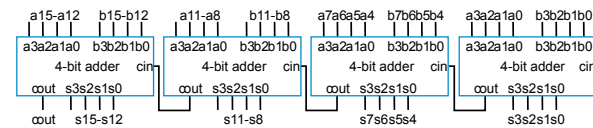
- Problem: Gates get bigger in each stage

- 4th stage has 5-input gates
- 32nd stage would have 33-input gates
 - Too many inputs for one gate
 - Would require building from smaller gates, meaning more levels (slower), more gates (bigger)



- One solution: Connect 4-bit CLA adders in ripple manner

- But slow (4 + 4 + 4 + 4 gate delays)

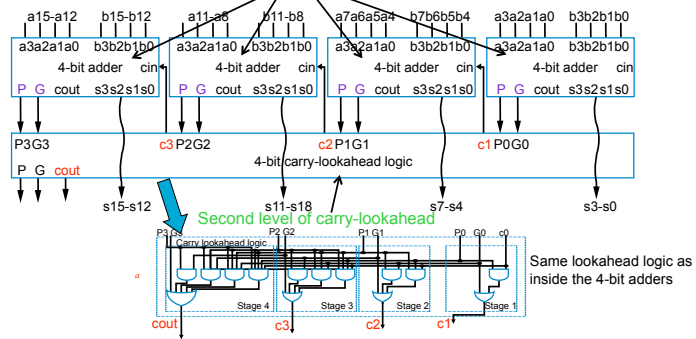


12

Carry-Lookahead Adder

Hierarchical Carry-Lookahead Adders

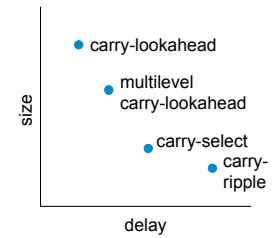
- Better solution -- Rather than rippling the carries, just *repeat* the carry-lookahead concept
 - Requires minor modification of 4-bit CLA adder to output P and G



13

Optimizations and Tradeoffs

Adder Tradeoffs



- Designer picks the adder that satisfies particular delay and size requirements
 - May use different adder types in different parts of same design
 - Faster adders on critical path, smaller adders on non-critical path

14