Digital Design and Computer Organization

Carry-lookahead and Prefix adders - 1

Reetinder Sidhu

Department of Computer Science and Engineering

Digital Design and Computer Organization

Carry-lookahead and Prefix adders - 1

Reetinder Sidhu

Department of Computer Science and Engineering

Course Outline

- Digital Design
  - Combinational logic design
  - Sequential logic design
    - **Carry-lookahead and Prefix adders - 1**
- Computer Organization
  - Architecture (microprocessor instruction set)
  - Microarchitecure (microprocessor operation)

Concepts covered

- Adder Performance Evaluation
- Carry-Lookahead Adder

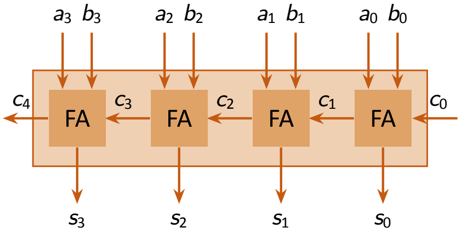
Carry-lookahead and Prefix adders - 1

Preliminaries

- We evaluate the preformance of various adder types by estimating their area and time requirements
  - Area is estimated by number of logic gates required
  - Time is estimated based on critical path delay
- We assume that all two input AND, OR and XOR gates:
  - Occupy area a_g
  - Have a propagation delay t_g
  - Somewhat simplifying, but realistic assumption
- How about gates with more than two inputs? For a k-input AND/OR/XOR gate:
  - Its area is estimated to be $(k-1)a_g$
    - ★ Since a k input AND/OR/XOR can be constructed using $k-1$ two input gates of the type
  - Its propagation delay is estimated to be $\lceil(\log_2 k)\rceil t_d$
    - ★ Shortest delay above obtained when the $k-1$ gates are arranged in a “tree-like” fashion
- No inverters in adder designs considered
  - If present, can be ignored in initial analysis

Carry-lookahead and Prefix adders - 1

Ripple Carry Adder Area and Time



- Area requirements:
  - Each full adder contains five two input gates (for carry) and one three input gate (for sum)
  - So each full adder occupies $7a_g$ area
- An n-bit ripple carry adder thus occupies $7na_g$ area

- Time requirements: For an n-bit ripple carry adder, critical path delay is composed of:
  - Propagation delay from c_0 to c_{n-1}
    - Signal passes through two gates in each of the $n - 1$ stages
    - $2(n - 1)t_g$ time required
  - Sum computation
    - $2t_g$ time required for three input XOR gate
- An n-bit ripple carry adder thus occupies $2nt_g$ time

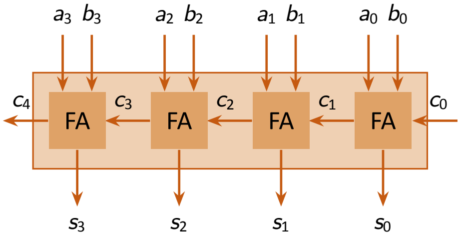
The Carry Chain Problem

- Time requirement for addition increases linearly with number of bits to be added
  - If 16-bit addition requires x time, 64-bit addition will require $4x$ time
- Same for subtraction, increment and decrement
- For such a fundamental operation faster solutions are required
- Speed can be improved by eliminating the carry chain

Carry-lookahead and Prefix adders - 1

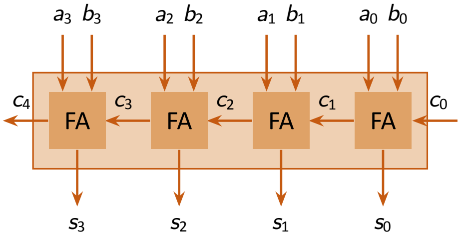
Direct generation of carry?

- Compute carry (c_i) directly from inputs (a_i and b_i):



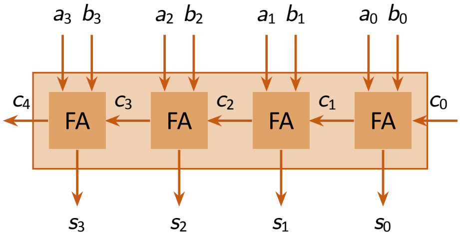
Carry-lookahead and Prefix adders - 1
Direct generation of carry?

- Compute carry (c_i) directly from inputs (a_i and b_i):
  - $c_1 = a_0 b_0 + b_0 c_0 + c_0 a_0$



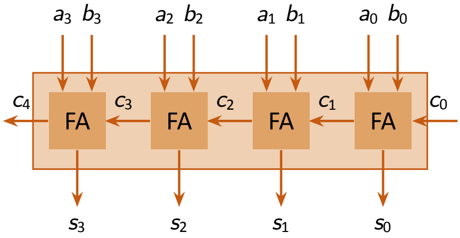
Direct generation of carry?

- Compute carry (c_i) directly from inputs (a_i and b_i):
  - $c_1 = a_0 b_0 + b_0 c_0 + c_0 a_0 = a_0 b_0 + (a_0 + b_0) c_0$



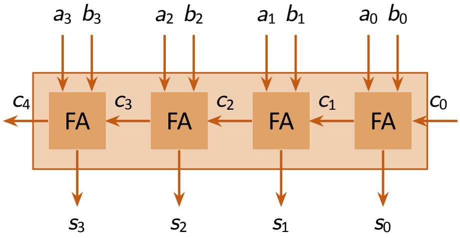
Direct generation of carry?

- Compute carry (c_i) directly from inputs (a_i and b_i):
  - $c_1 = a_0 b_0 + b_0 c_0 + c_0 a_0 = a_0 b_0 + (a_0 + b_0) c_0$
  - $c_2 = a_1 b_1 + (a_1 + b_1) c_1$



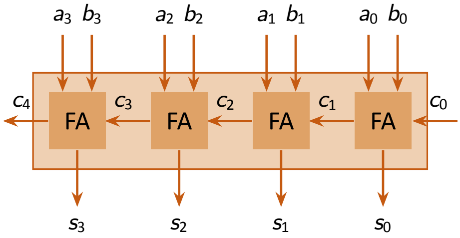
Direct generation of carry?

- Compute carry (c_i) directly from inputs (a_i and b_i):
  - $c_1 = a_0 b_0 + b_0 c_0 + c_0 a_0 = a_0 b_0 + (a_0 + b_0) c_0$
  - $c_2 = a_1 b_1 + (a_1 + b_1) c_1$
    $= a_1 b_1 + (a_1 + b_1)(a_0 b_0 + (a_0 + b_0) c_0)$



Direct generation of carry?

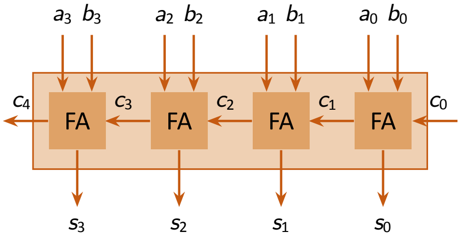
- Compute carry (c_i) directly from inputs (a_i and b_i):
  - $c_1 = a_0 b_0 + b_0 c_0 + c_0 a_0 = a_0 b_0 + (a_0 + b_0) c_0$
  - $c_2 = a_1 b_1 + (a_1 + b_1) c_1$
    $= a_1 b_1 + (a_1 + b_1)(a_0 b_0 + (a_0 + b_0) c_0)$
    $= a_1 b_1 + (a_1 + b_1) a_0 b_0 + (a_1 + b_1)(a_0 + b_0) c_0$



Carry-lookahead and Prefix adders - 1

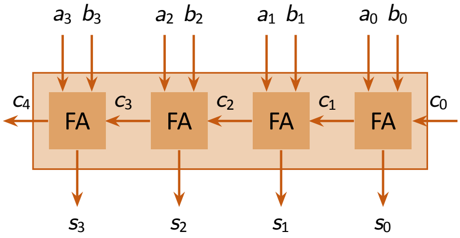
Direct generation of carry?

- Compute carry (c_i) directly from inputs (a_i and b_i):
  - $c_1 = a_0 b_0 + b_0 c_0 + c_0 a_0 = a_0 b_0 + (a_0 + b_0) c_0$
  - $c_2 = a_1 b_1 + (a_1 + b_1) c_1$
    $= a_1 b_1 + (a_1 + b_1)(a_0 b_0 + (a_0 + b_0) c_0)$
    $= a_1 b_1 + (a_1 + b_1) a_0 b_0 + (a_1 + b_1)(a_0 + b_0) c_0$
- Lots of terms of type $a_i b_i$ and $(a_i + b_i)$. So to make the formulas simpler, we substitute:



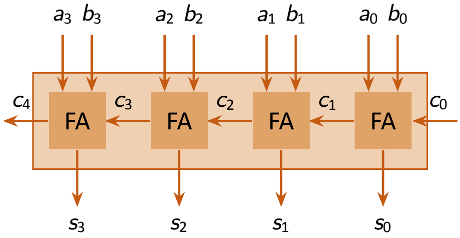
Direct generation of carry?

- Compute carry (c_i) directly from inputs (a_i and b_i):
  - $c_1 = a_0 b_0 + b_0 c_0 + c_0 a_0 = a_0 b_0 + (a_0 + b_0) c_0$
  - $c_2 = a_1 b_1 + (a_1 + b_1) c_1$
    $= a_1 b_1 + (a_1 + b_1)(a_0 b_0 + (a_0 + b_0) c_0)$
    $= a_1 b_1 + (a_1 + b_1) a_0 b_0 + (a_1 + b_1)(a_0 + b_0) c_0$
- Lots of terms of type $a_i b_i$ and $(a_i + b_i)$. So to make the formulas simpler, we substitute:
  - $g_i = a_i b_i$
  - $p_i = a_i + b_i$



Direct generation of carry?

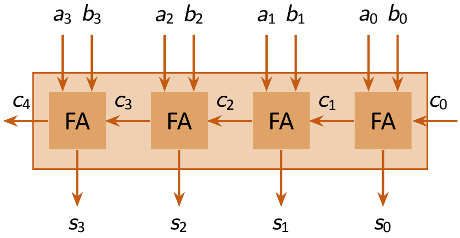
- Compute carry (c_i) directly from inputs (a_i and b_i):
  - $c_1 = a_0 b_0 + b_0 c_0 + c_0 a_0 = a_0 b_0 + (a_0 + b_0) c_0$
  - $c_2 = a_1 b_1 + (a_1 + b_1) c_1$
    $= a_1 b_1 + (a_1 + b_1)(a_0 b_0 + (a_0 + b_0) c_0)$
    $= a_1 b_1 + (a_1 + b_1) a_0 b_0 + (a_1 + b_1)(a_0 + b_0) c_0$
- Lots of terms of type $a_i b_i$ and $(a_i + b_i)$. So to make the formulas simpler, we substitute:
  - $g_i = a_i b_i$
  - $p_i = a_i + b_i$
- Above substitution yields:
  - $c_1 = g_0 + p_0 c_0$



Carry-lookahead and Prefix adders - 1

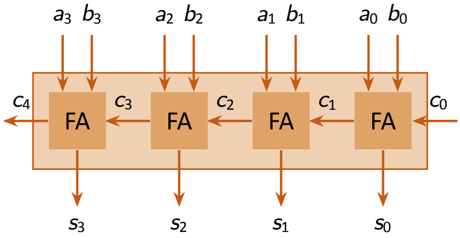
Direct generation of carry?

- Compute carry (c_i) directly from inputs (a_i and b_i):
  - $c_1 = a_0 b_0 + b_0 c_0 + c_0 a_0 = a_0 b_0 + (a_0 + b_0) c_0$
  - $c_2 = a_1 b_1 + (a_1 + b_1) c_1$
    $= a_1 b_1 + (a_1 + b_1)(a_0 b_0 + (a_0 + b_0) c_0)$
    $= a_1 b_1 + (a_1 + b_1) a_0 b_0 + (a_1 + b_1)(a_0 + b_0) c_0$
- Lots of terms of type $a_i b_i$ and $(a_i + b_i)$. So to make the formulas simpler, we substitute:
  - $g_i = a_i b_i$
  - $p_i = a_i + b_i$
- Above substitution yields:
  - $c_1 = g_0 + p_0 c_0$
  - $c_2 = g_1 + p_1 g_0 + p_1 p_0 c_0$



Direct generation of carry?

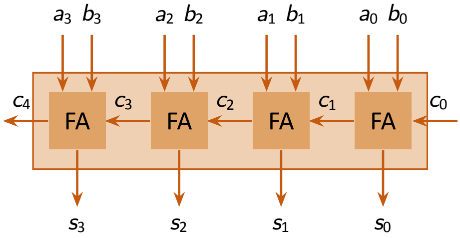
- Compute carry (c_i) directly from inputs (a_i and b_i):
  - $c_1 = a_0 b_0 + b_0 c_0 + c_0 a_0 = a_0 b_0 + (a_0 + b_0) c_0$
  - $c_2 = a_1 b_1 + (a_1 + b_1) c_1$
    $= a_1 b_1 + (a_1 + b_1)(a_0 b_0 + (a_0 + b_0) c_0)$
    $= a_1 b_1 + (a_1 + b_1) a_0 b_0 + (a_1 + b_1)(a_0 + b_0) c_0$
- Lots of terms of type $a_i b_i$ and $(a_i + b_i)$. So to make the formulas simpler, we substitute:
  - $g_i = a_i b_i$
  - $p_i = a_i + b_i$
- Above substitution yields:
  - $c_1 = g_0 + p_0 c_0$
  - $c_2 = g_1 + p_1 g_0 + p_1 p_0 c_0$
  - $c_3 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0$



Carry-lookahead and Prefix adders - 1

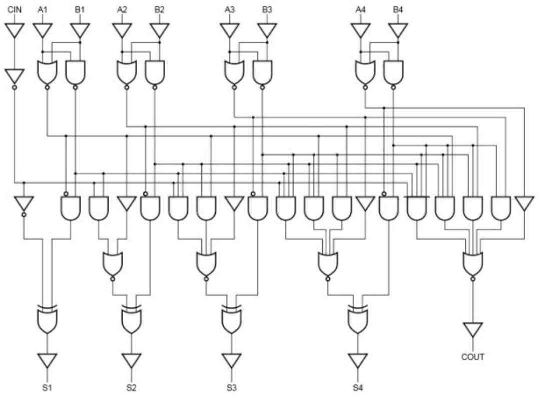
Direct generation of carry?

- Compute carry (c_i) directly from inputs (a_i and b_i):
  - $c_1 = a_0 b_0 + b_0 c_0 + c_0 a_0 = a_0 b_0 + (a_0 + b_0) c_0$
  - $c_2 = a_1 b_1 + (a_1 + b_1) c_1$
    $= a_1 b_1 + (a_1 + b_1)(a_0 b_0 + (a_0 + b_0) c_0)$
    $= a_1 b_1 + (a_1 + b_1) a_0 b_0 + (a_1 + b_1)(a_0 + b_0) c_0$
- Lots of terms of type $a_i b_i$ and $(a_i + b_i)$. So to make the formulas simpler, we substitute:
  - $g_i = a_i b_i$
  - $p_i = a_i + b_i$
- Above substitution yields:
  - $c_1 = g_0 + p_0 c_0$
  - $c_2 = g_1 + p_1 g_0 + p_1 p_0 c_0$
  - $c_3 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0$
  - $c_4 = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_0$



4-bit Carry-Lookahead Adder

4-bit Carry-Lookahead Adder



Source: *electronicshub*

Carry-Lookahead Adder Area Estimate

- All gates mentioned below are two input gates
- From the carry formulas, we see that for a carry-lookahead adder:

Carry-Lookahead Adder Area Estimate

- All gates mentioned below are two input gates
- From the carry formulas, we see that for a carry-lookahead adder:
  - For c_1 4 gates are required

Carry-Lookahead Adder Area Estimate

- All gates mentioned below are two input gates
- From the carry formulas, we see that for a carry-lookahead adder:
  - For c_1 4 gates are required
  - For c_i number of gates required is $2i + 2$ greater than gates required for c_{i-1}

Carry-Lookahead Adder Area Estimate

- All gates mentioned below are two input gates
- From the carry formulas, we see that for a carry-lookahead adder:
  - For c_1 4 gates are required
  - For c_i number of gates required is $2i + 2$ greater than gates required for c_{i-1}
  - So $i(i+3)$ gates are required for c_1 to c_i

Carry-Lookahead Adder Area Estimate

- All gates mentioned below are two input gates
- From the carry formulas, we see that for a carry-lookahead adder:
  - For c_1 4 gates are required
  - For c_i number of gates required is $2i + 2$ greater than gates required for c_{i-1}
  - So $i(i+3)$ gates are required for c_1 to c_i
  - Each three input XOR gate (for sum) would count as two gates

Carry-Lookahead Adder Area Estimate

- All gates mentioned below are two input gates
- From the carry formulas, we see that for a carry-lookahead adder:
  - For c_1 4 gates are required
  - For c_i number of gates required is $2i + 2$ greater than gates required for c_{i-1}
  - So $i(i+3)$ gates are required for c_1 to c_i
  - Each three input XOR gate (for sum) would count as two gates
- So an n-bit carry-lookahead adder would require $n^2 + 5n$ gates

Carry-Lookahead Adder Time Estimate

- Carry formulas for 4-bit carry-lookahead adder:
  - $c_1 = g_0 + p_0 c_0$
  - $c_2 = g_1 + p_1 g_0 + p_1 p_0 c_0$
  - $c_3 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0$
  - $c_4 = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_0$

CARRY-LOOKAHEAD AND PREFIX ADDERS - 1
Carry-Lookahead Adder Time Estimate

- Carry formulas for 4-bit carry-lookahead adder:
  - $c_1 = g_0 + p_0 c_0$
  - $c_2 = g_1 + p_1 g_0 + p_1 p_0 c_0$
  - $c_3 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0$
  - $c_4 = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_0$
- Critical path delay for a carry-lookahead adder is composed of:

CARRY-LOOKAHEAD AND PREFIX ADDERS - 1

Carry-Lookahead Adder Time Estimate

- Carry formulas for 4-bit carry-lookahead adder:
  - $c_1 = g_0 + p_0 c_0$
  - $c_2 = g_1 + p_1 g_0 + p_1 p_0 c_0$
  - $c_3 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0$
  - $c_4 = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_0$
- Critical path delay for a carry-lookahead adder is composed of:
  - p and g computation
    - t_g time required required for two input AND/OR gate

Carry-Lookahead Adder Time Estimate

- Carry formulas for 4-bit carry-lookahead adder:
  - $c_1 = g_0 + p_0 c_0$
  - $c_2 = g_1 + p_1 g_0 + p_1 p_0 c_0$
  - $c_3 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0$
  - $c_4 = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_0$
- Critical path delay for a carry-lookahead adder is composed of:
  - p and g computation
    - t_g time required required for two input AND/OR gate
  - carry computation delay
    - time required for c_i depends on i

Carry-Lookahead Adder Time Estimate

- Carry formulas for 4-bit carry-lookahead adder:
  - $c_1 = g_0 + p_0 c_0$
  - $c_2 = g_1 + p_1 g_0 + p_1 p_0 c_0$
  - $c_3 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0$
  - $c_4 = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_0$
- Critical path delay for a carry-lookahead adder is composed of:
  - p and g computation
    - t_g time required required for two input AND/OR gate
  - carry computation delay
    - time required for c_i depends on i
  - sum computation
    - $2t_g$ time required for three input XOR

Carry-Lookahead Adder Time Estimate

- Carry formulas for 4-bit carry-lookahead adder:
  - $c_1 = g_0 + p_0 c_0$
  - $c_2 = g_1 + p_1 g_0 + p_1 p_0 c_0$
  - $c_3 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0$
  - $c_4 = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_0$
- Critical path delay for a carry-lookahead adder is composed of:
  - p and g computation
    - t_g time required required for two input AND/OR gate
  - carry computation delay
    - time required for c_i depends on i
  - sum computation
    - $2t_g$ time required for three input XOR
- Carry computation delay: for an n-bit carry-lookahead adder, longest delay is for c_{n-1}, which is composed of:

Carry-Lookahead Adder Time Estimate

- Carry formulas for 4-bit carry-lookahead adder:
  - $c_1 = g_0 + p_0 c_0$
  - $c_2 = g_1 + p_1 g_0 + p_1 p_0 c_0$
  - $c_3 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0$
  - $c_4 = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_0$
- Critical path delay for a carry-lookahead adder is composed of:
  - p and g computation
    - t_g time required required for two input AND/OR gate
  - carry computation delay
    - time required for c_i depends on i
  - sum computation
    - $2t_g$ time required for three input XOR
- Carry computation delay: for an n-bit carry-lookahead adder, longest delay is for c_{n-1}, which is composed of:
  - Delay for the minterm $p_{n-2} p_{n-3} \cdots p_0 c_0$
    - n input AND gate requires $\lceil \log_2(n-1) \rceil t_g$ time

Carry-lookahead and Prefix adders - 1
Carry-Lookahead Adder Time Estimate

- Carry formulas for 4-bit carry-lookahead adder:
  - $c_1 = g_0 + p_0 c_0$
  - $c_2 = g_1 + p_1 g_0 + p_1 p_0 c_0$
  - $c_3 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0$
  - $c_4 = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_0$
- Critical path delay for a carry-lookahead adder is composed of:
  - p and g computation
    - t_g time required required for two input AND/OR gate
  - carry computation delay
    - time required for c_i depends on i
  - sum computation
    - $2t_g$ time required for three input XOR
- Carry computation delay: for an n-bit carry-lookahead adder, longest delay is for c_{n-1}, which is composed of:
  - Delay for the minterm $p_{n-2} p_{n-3} \cdots p_0 c_0$
    - n input AND gate requires $\lceil \log_2(n-1) \rceil t_g$ time
  - Delay for the OR of all minterms
    - OR of n inputs requires $\lceil \log_2(n-1) \rceil t_g$ time

Carry-lookahead and Prefix adders - 1

Carry-Lookahead Adder Time Estimate

- Carry formulas for 4-bit carry-lookahead adder:
  - $c_1 = g_0 + p_0 c_0$
  - $c_2 = g_1 + p_1 g_0 + p_1 p_0 c_0$
  - $c_3 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0$
  - $c_4 = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_0$
- Critical path delay for a carry-lookahead adder is composed of:
  - p and g computation
    - t_g time required required for two input AND/OR gate
  - carry computation delay
    - time required for c_i depends on i
  - sum computation
    - $2t_g$ time required for three input XOR
- Carry computation delay: for an n-bit carry-lookahead adder, longest delay is for c_{n-1}, which is composed of:
  - Delay for the minterm $p_{n-2} p_{n-3} \cdots p_0 c_0$
    - n input AND gate requires $\lceil \log_2(n-1) \rceil t_g$ time
  - Delay for the OR of all minterms
    - OR of n inputs requires $\lceil \log_2(n-1) \rceil t_g$ time
- So total time required (critical path delay) for an n-bit carry-lookahead adder: $2\lceil \log_2(n-1) \rceil t_g + 3t_g$

Carry-lookahead and Prefix adders - 1
Performance Comparison

- Area and time estimates for n-bit adders:

	Area	Time
Ripple carry	$7na_g$	$2nt_g$
Carry-lookahead	$(n^2+5n)a_g$	$2\lceil \log_2(n-1) \rceil t_g + 3t_g$

- Compared to the ripple carry adder's linear delay increase with size, the carry-lookahead adder delay increase only logarithmically, resulting in dramatically faster adders
- However, the area of the carry-lookahead adder increase qudratically with size
- Is there an adder design that retains the carry-lookahead adder's speed but has significantly lesser area?