Digital Design and Computer Organization

Carry-lookahead and Prefix adders - 2

Reetinder Sidhu

Department of Computer Science and Engineering

Digital Design and Computer Organization

Carry-lookahead and Prefix adders - 2

Reetinder Sidhu

Department of Computer Science and Engineering

Carry-lookahead and Prefix adders - 2

Course Outline

- Digital Design
  - Combinational logic design
  - Sequential logic design
    - **Carry-lookahead and Prefix adders - 2**
- Computer Organization
  - Architecture (microprocessor instruction set)
  - Microarchitecure (microprocessor operation)

Concepts covered

- Carry-Lookahead Adder

Scaling Carry-Lookahead Adders

- Ripple carry adders are compact but slow
- Carry-lookahead adders are fast enough but difficult to scale to large sizes

Scaling Carry-Lookahead Adders

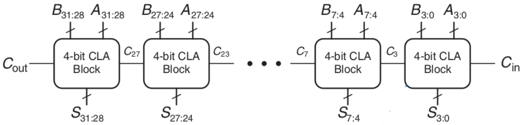
PES UNIVERSITY ONLINE

- Ripple carry adders are compact but slow
- Carry-lookahead adders are fast enough but difficult to scale to large sizes
- One solution is a hybrid approach:
  - Split the adder into a number of blocks
  - Use carry-lookahead technique to add bits in each block
  - Combine the blocks together using ripple carry technique

CARRY-LOOKAHEAD AND PREFIX ADDERS - 2

Approach

- Carry-lookahead for blocks which are combined using ripple carry technique:

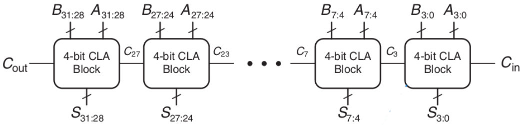


Source: *Elsevier*

Carry-lookahead and Prefix adders - 2

Approach

- Carry-lookahead for blocks which are combined using ripple carry technique:



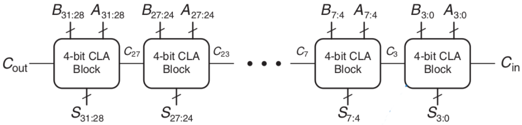
Source: *Elsevier*

- Each 4-bit block can be a carry-lookahead adder

Carry-lookahead and Prefix adders - 2

Approach

- Carry-lookahead for blocks which are combined using ripple carry technique:



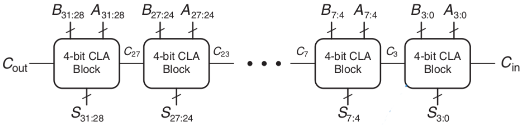
Source: *Elsevier*

- Each 4-bit block can be a carry-lookahead adder
- Critical path is from a_0, b_0 and c_{in} to s_{31}

CARRY-LOOKAHEAD AND PREFIX ADDERS - 2

Approach

- Carry-lookahead for blocks which are combined using ripple carry technique:



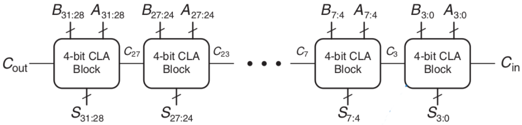
Source: *Elsevier*

- Each 4-bit block can be a carry-lookahead adder
- Critical path is from a_0, b_0 and c_{in} to s_{31}
- So it is important to compute $c_3, c_7, \ldots, c_{27}$ quickly

Carry-lookahead and Prefix adders - 2

Approach

- Carry-lookahead for blocks which are combined using ripple carry technique:



Source: *Elsevier*

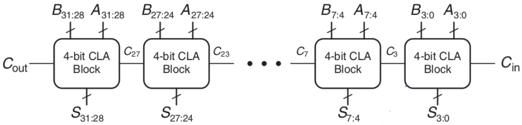
- Each 4-bit block can be a carry-lookahead adder
- Critical path is from a_0, b_0 and c_{in} to s_{31}
- So it is important to compute $c_3, c_7, \ldots, c_{27}$ quickly

- Not so important to compute sum outputs s_0 to s_{30} quickly

Carry-lookahead and Prefix adders - 2

Approach

- Carry-lookahead for blocks which are combined using ripple carry technique:



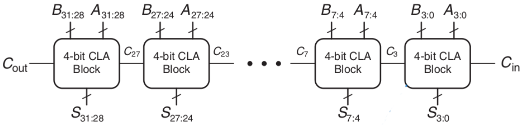
Source: *Elsevier*

- Each 4-bit block can be a carry-lookahead adder
- Critical path is from a_0, b_0 and c_{in} to s_{31}
- So it is important to compute $c_3, c_7, \ldots, c_{27}$ quickly
  - So use carry lookahead approach to compute above carry values
- Not so important to compute sum outputs s_0 to s_{30} quickly

Carry-lookahead and Prefix adders - 2

Approach

- Carry-lookahead for blocks which are combined using ripple carry technique:



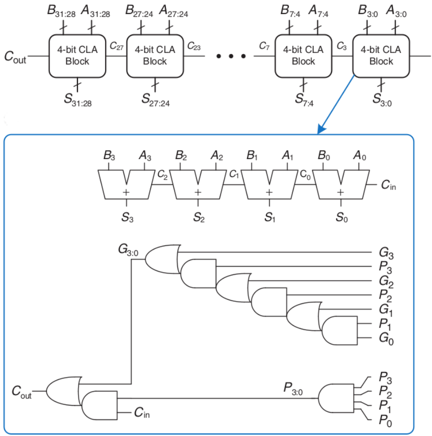
Source: *Elsevier*

- Each 4-bit block can be a carry-lookahead adder
- Critical path is from a_0, b_0 and c_{in} to s_{31}
- So it is important to compute $c_3, c_7, \ldots, c_{27}$ quickly
  - So use carry lookahead approach to compute above carry values
- Not so important to compute sum outputs s_0 to s_{30} quickly
  - So use ripple carry technique inside each block as well to compute the sum outputs

Carry-lookahead and Prefix adders - 2

Block structure

- $$C_{out} = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_{in}$$
  $$= g_3 + p_3(g_2 + p_2 g_1 + p_2 p_1 g_0) + p_3 p_2 p_1 p_0 c_{in}$$
  $$= g_3 + p_3(g_2 + p_2(g_1 + p_1 g_0)) + p_3 p_2 p_1 p_0 c_{in}$$
- Logic circuit for above Boolean formula shown in figure

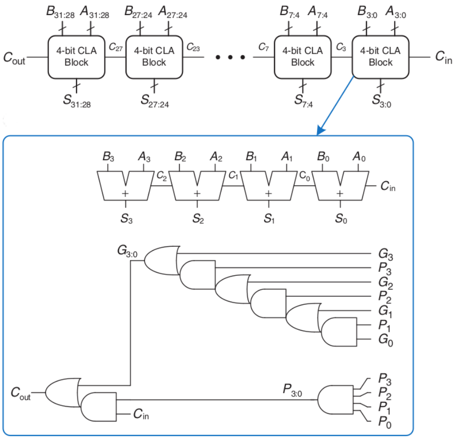


Source: *Elsevier*

CARRY-LOOKAHEAD AND PREFIX ADDERS - 2

Critical Path Delay

- Critical path is from a_0, b_0 and c_{in} to s_{31}
- Three parts of the critical path delay are the time required to:
  - compute various p and g
  - for carry to propagate from c_0 to c_{27}
  - compute sum s_{31}

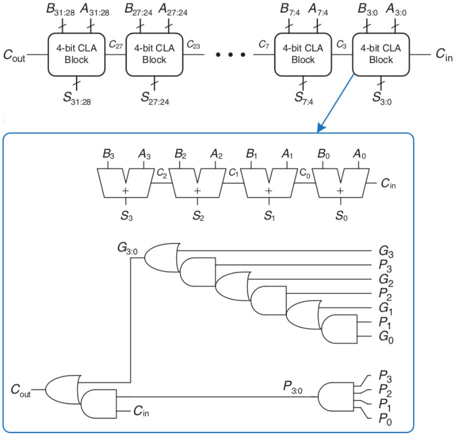


Source: *Elsevier*

Carry-lookahead and Prefix adders - 2

Critical Path Delay

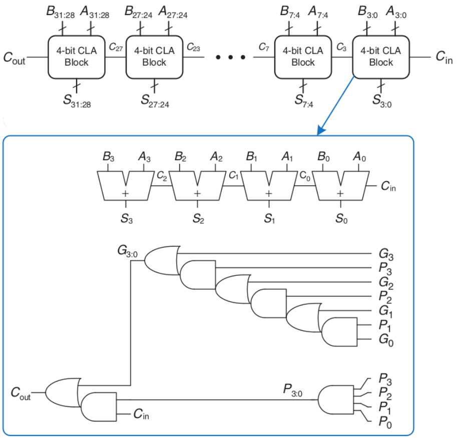
- Time required to compute various p and g:
  - Compute p_i and g_i $(0 \leq i < 4)$ in each block in time t_{pg}
  - Compute $g_{3:0}$ in each block in time t_{pg_block}



Source: *Elsevier*

Carry-lookahead and Prefix adders - 2
Critical Path Delay

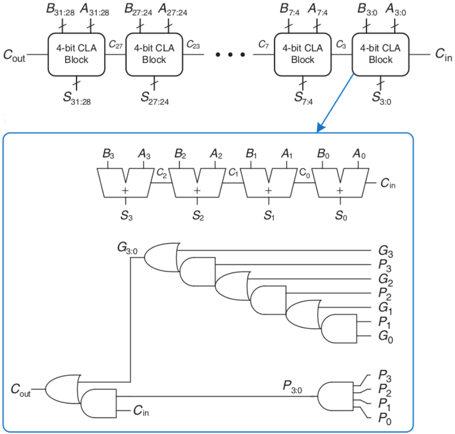
- Time required for carry to propagate from c_0 to c_{27}:
  - In each block, c_{in} propagates through an AND gate and an OR gate to emerge as c_{out} in time t_{AND_OR}
  - Since carry propagates in above manner through first seven blocks, time required is $7t_{AND\_OR}$



Source: *Elsevier*

CARRY-LOOKAHEAD AND PREFIX ADDERS - 2
Critical Path Delay

- Time required to compute sum s_{31}:
  - Once c_{27} is available, it needs to propagate through the four full adders, each of which takes time t_{FA}
  - So total time required is $4t_{FA}$
- So critical path delay is:
  - $t_{CLA} = t_{pg} + t_{pg\_block} + 7t_{AND\_OR} + 4t_{FA}$

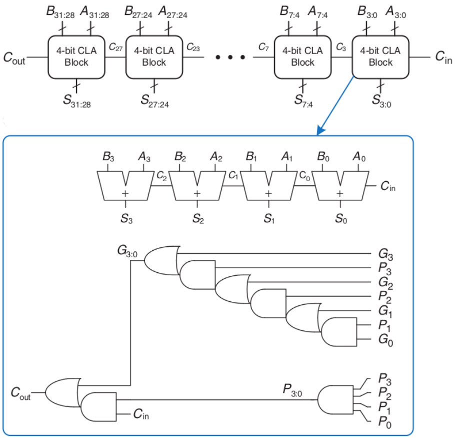


Source: *Elsevier*

CARRY-LOOKAHEAD AND PREFIX ADDERS - 2
Critical Path Delay

- Generalizing, if we assume an N-bit adder is constructed using k-bit blocks:
  - $\frac{N}{k}$ blocks each if size k will be used
  - Hence critical path delay would be:

    $t_{CLA} =$
    $t_{pg} + t_{pg\_block} + (\frac{N}{k} - 1)t_{AND\_OR} + kt_{FA}$



Source: *Elsevier*