Overview

- Last lecture
  - PLDs
  - ROMs
  - Tristates
  - Design examples
- Today
  - Adders
    - Ripple-carry
    - Carry-lookahead
    - Carry-select
  - The conclusion of combinational logic!!!

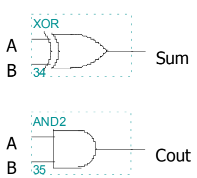
Arithmetic circuits

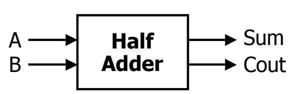
- General-purpose building blocks
  - Critical components in processor datapaths
    - Adders
    - Multipliers (integer, floating-point)
    - ALUs
  - Perform most computer instructions
  - Time ↔ space tradeoff
    - Fast circuits usually require more logic

Binary half adder

- 1-bit half adder
  - Computes sum, carry-out
    - No carry-in
  - Sum = A'B + AB' = A xor B
  - Cout = AB

A	B	S	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

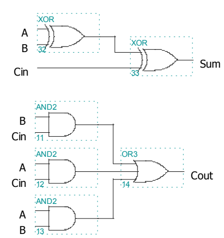


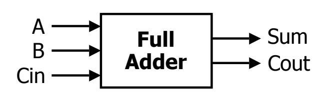


Binary full adder

- 1-bit full adder
  - Computes sum, carry-out
    - Carry-in allows cascaded adders
  - Sum = Cin xor A xor B
  - Cout = ACin + BCin + AB

A	B	C_{in}	S	C_{out}
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





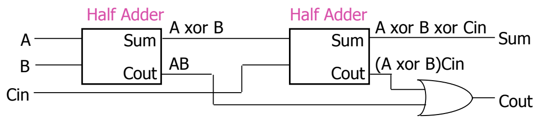
Full adder: Alternative implementation

- Multilevel logic
  - Slower
  - Less gates
    - ➪ 2 XORs, 2 ANDs, 1 OR

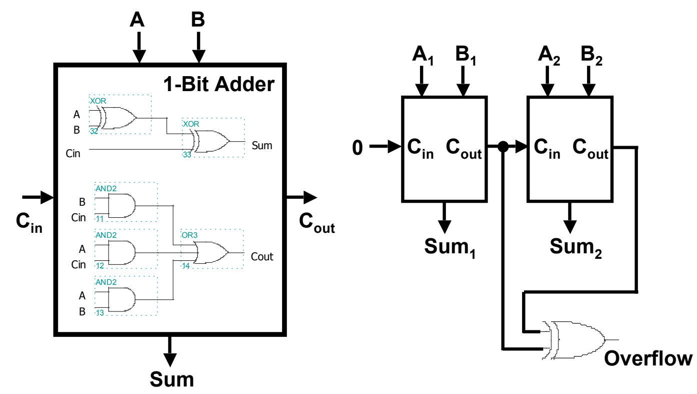
A	B	C_{in}	S	C_{out}	C_{out}
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	0	1	1
1	1	0	0	1	1
1	1	1	1	1	1

Sum = (A ⊕ B) ⊕ Cin

Cout = ACin + BCin + AB
= (A ⊕ B)Cin + AB

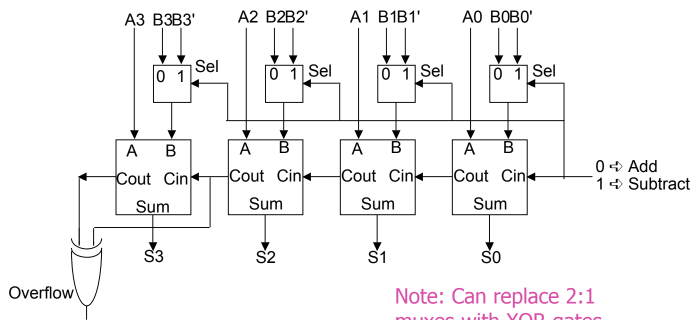


2-bit ripple-carry adder



4-bit ripple-carry adder/subtractor

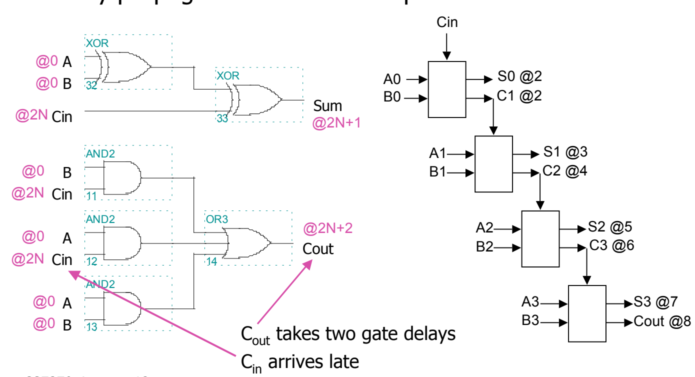
- Circuit adds or subtracts
  - 2s complement: A – B = A + (–B) = A + B' + 1



Note: Can replace 2:1 muxes with XOR gates

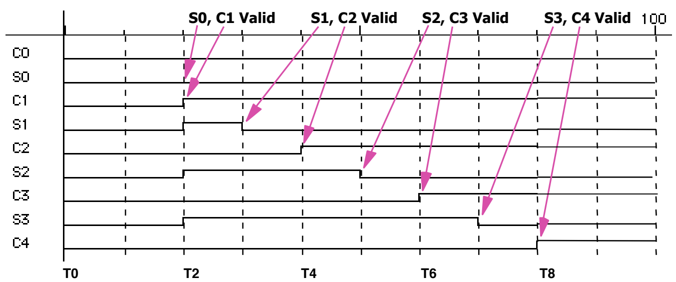
Problem: Ripple-carry delay

- Carry propagation limits adder speed



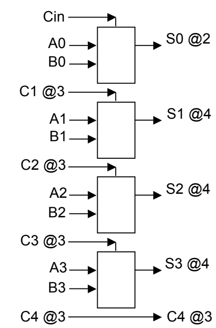
Ripple-carry adder timing diagram

- Critical delay
  - Carry propagation
  - 1111 + 0001 = 10000 is worst case

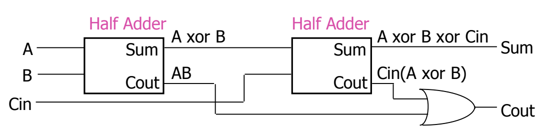


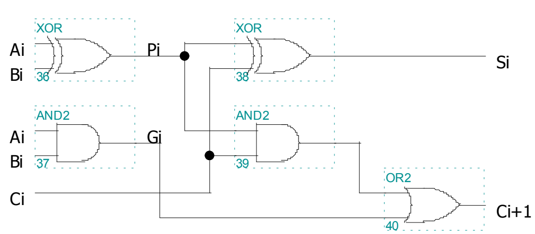
One solution: Carry lookahead logic

- Compute all the carries in parallel
  - Derive carries from the data inputs
    - Not from intermediate carries
    - Use two-level logic
  - Compute all sums in parallel
- Cascade simple adders to make large adders
- Speed improvement
  - 16-bit ripple-carry: ~32 gate delays
  - 16-bit carry-lookahead: ~8 gate delays
- Issues
  - Complex combinational logic



Full adder again





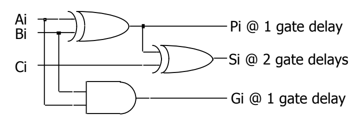
Carry-lookahead logic

- Carry generate: $G_i = A_iB_i$
  - Generate carry when A = B = 1
- Carry propagate: $P_i = A_i \text{ xor } B_i$
  - Propagate carry-in to carry-out when (A xor B) = 1
- Sum and Cout in terms of generate/propagate:
  - $S_i = A_i \text{ xor } B_i \text{ xor } C_i$
    $= P_i \text{ xor } C_i$
  - $C_{i+1} = A_iB_i + C_i(A_i \text{ xor } B_i)$
    $= G_i + C_iP_i$

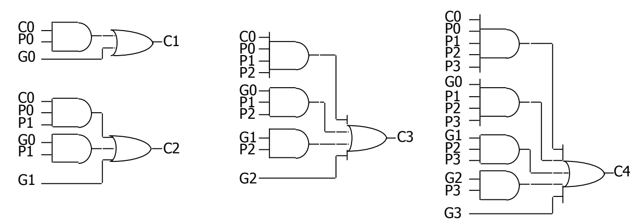
Carry-lookahead logic (cont'd)

- Re-express the carry logic in terms of G and P
  - $C_1 = G_0 + P_0C_0$
  - $C_2 = G_1 + P_1C_1 = G_1 + P_1G_0 + P_1P_0C_0$
  - $C_3 = G_2 + P_2C_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$
  - $C_4 = G_3 + P_3C_3 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$
- Implement each carry equation with two-level logic
  - Derive intermediate results directly from inputs
    - Rather than from carries
  - Allows "sum" computations to proceed in parallel

Implementing the carry-lookahead logic

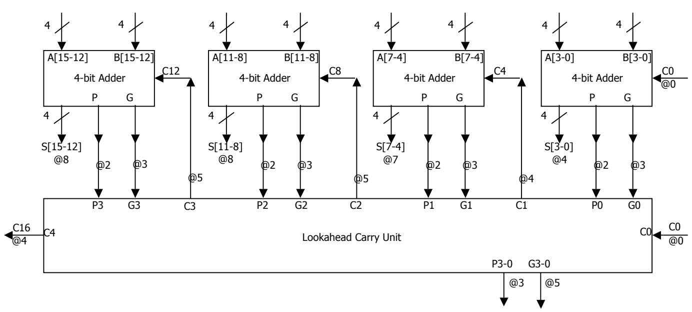


Logic complexity increases with adder size



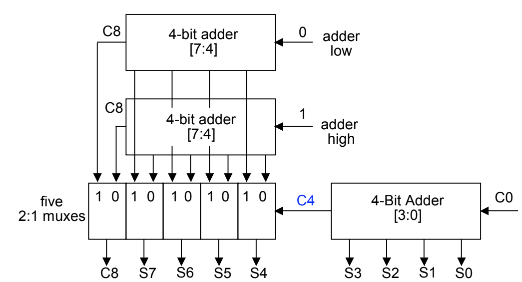
Cascaded carry-lookahead adder

- 4 four-bit adders with internal carry lookahead
  - Second level lookahead extends adder to 16 bits



Another solution: Carry-select adder

- Redundant hardware speeds carry calculation
  - Compute two high-order sums while waiting for carry-in (C4)
  - Select correct high-order sum after receiving C4



We've finished combinational logic...

- What you should know
  - Twos complement arithmetic
  - Truth tables
  - Basic logic gates
  - Schematic diagrams
  - Timing diagrams
  - Minterm and maxterm expansions (canonical, minimized)
  - de Morgan's theorem
  - AND/OR to NAND/NOR logic conversion
  - K-maps, logic minimization, don't cares
  - Multiplexers/demultiplexers
  - PLAs/PALs
  - ROMs
  - Adders