Lecture 11

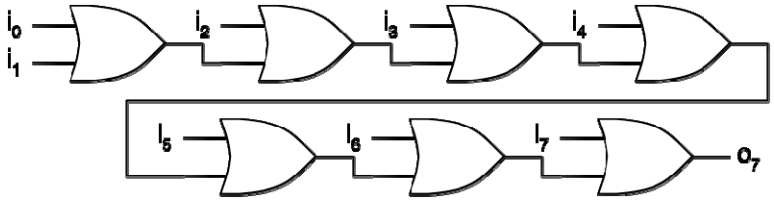
- Logistics
  - Midterm 1: ave 88, median 90, std 9. good job!
  - HW4 due on Wednesday
- Last lecture
  - Multi-level logic
  - Timing diagrams
- Today
  - Time/space trade offs: Parallel prefix trees
  - Adders
    - Ripple-carry
    - Carry-lookahead
    - Carry-select
  - The conclusion of combinational logic!!!

Arithmetic circuits

- General-purpose building blocks
  - Critical components in processor data paths
    - Adders
    - Multipliers
    - ALUs
  - Perform most computer instructions
  - Time ↔ space tradeoff
    - Fast circuits usually require more logic

What do we mean by time/speed tradeoff?: Linear chains vs. trees

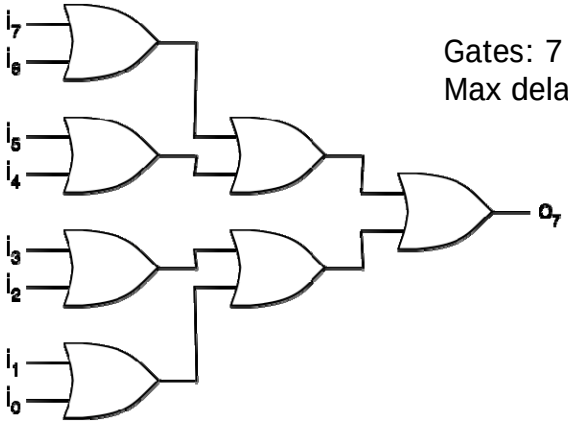
- Lets say we want to implement an 8-input OR function with only 2-input gates



Gates: 7
Max delay: 7

Linear chains vs. trees

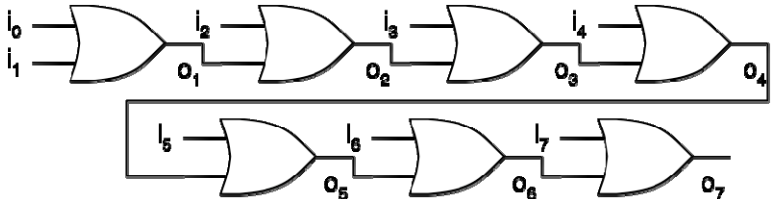
- Now consider the tree version



Gates: 7
Max delay: 3

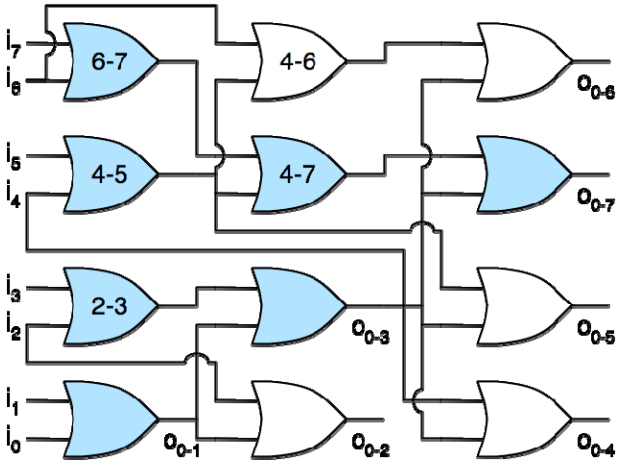
And now we change the problem slightly

- Build a circuit that takes 8 single bit inputs and calculates the OR of the first 2, the OR of the first 3, the OR of the first 4, and so on, up to the OR of all 8



Gates: 7
Max delay: 7

The tree version of the prefix circuit

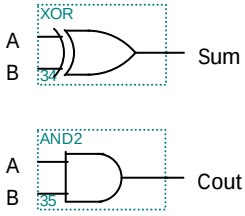


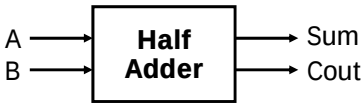
Gates: 12
Max delay: 3

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

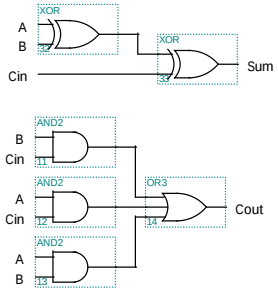


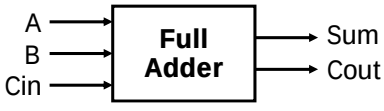


Binary full adder

- 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





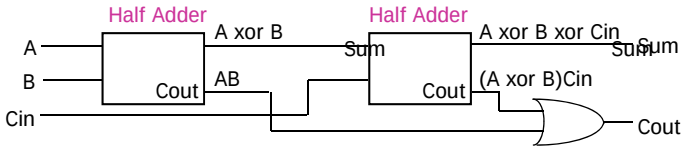
Full adder: Alternative implementation

- Multilevel logic
  - Slower
  - Fewer gates
    - 2 XORs, 2 ANDs, 1 OR

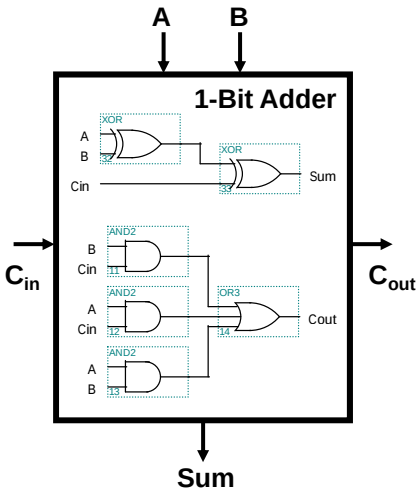
Sum = (A ⊕ B) ⊕ Cin

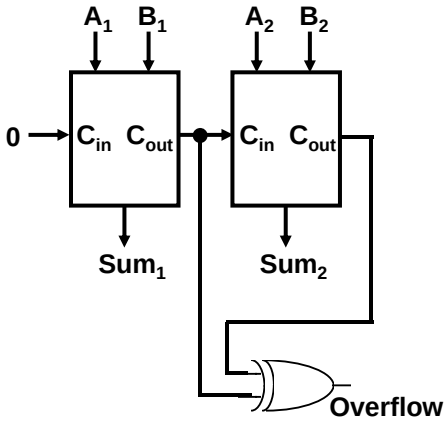
Cout = ACin + BCin + AB
= (A ⊕ B)Cin + AB

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



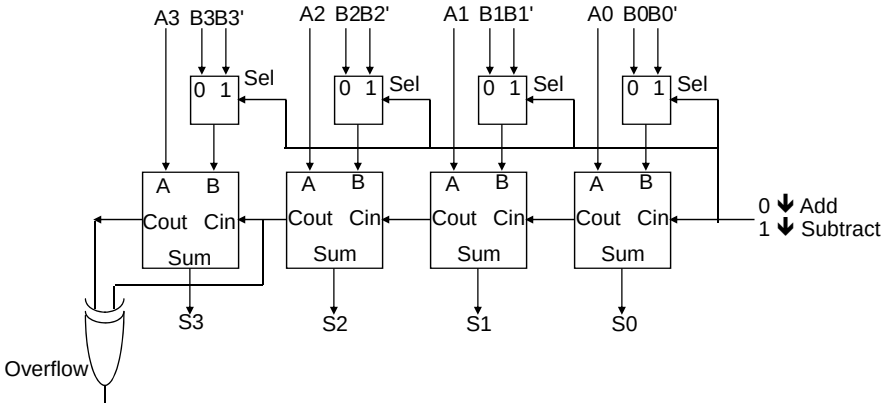
2-bit ripple-carry adder





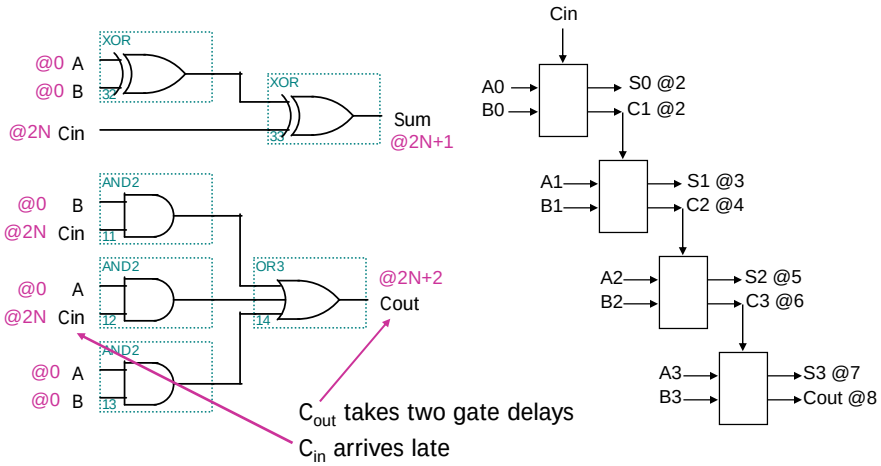
4-bit ripple-carry adder/subtractor

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



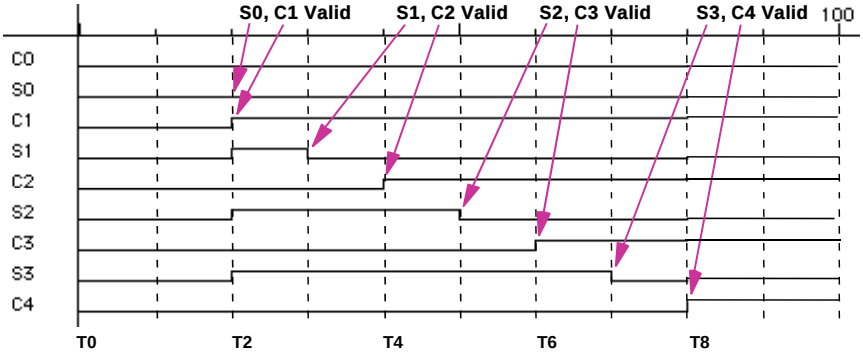
Problem: Ripple-carry delay

- Carry propagation limits adder speed



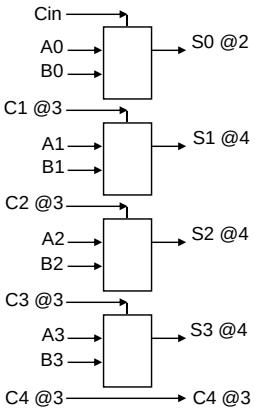
Ripple-carry adder timing diagram

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

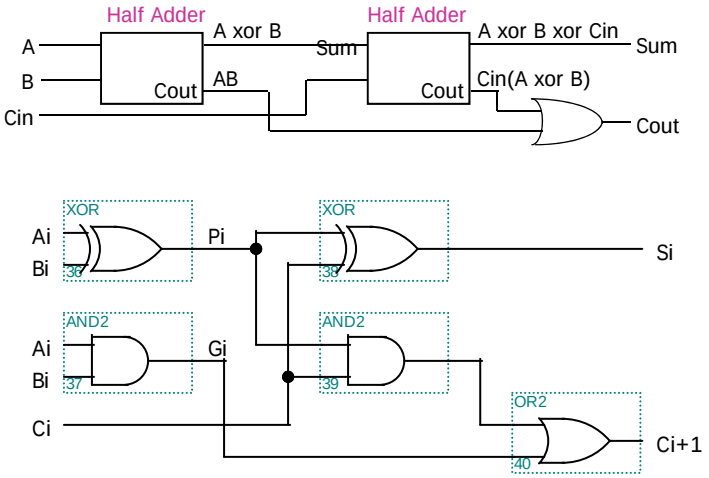


One solution: Carry lookahead logic

- 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



Full adder: getting Pi an Gi



Carry-lookahead logic

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

Carry-lookahead logic (cont'd)

- Re-express the carry logic in terms of G and P

  $C_1 = G_0 + P_0C_0$

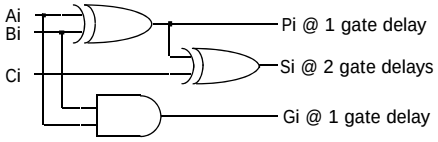
  $C_2 = G_1 + P_1C_1 = G_1 + P_1G_0 + P_1P_0C_0$

  $C_3 = G_2 + P_2C_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$

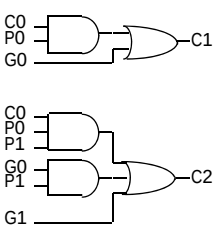
  $C_4 = G_3 + P_3C_3 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$

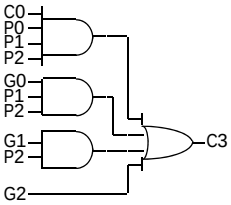
- Implement each carry equation with two-level logic
  - Derive intermediate results directly from inputs
    - Rather than from carries
  - Allows "sum" computations to proceed in parallel

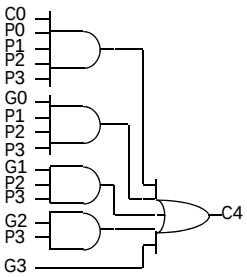
Implementing the carry-lookahead logic



Logic complexity increases with adder size

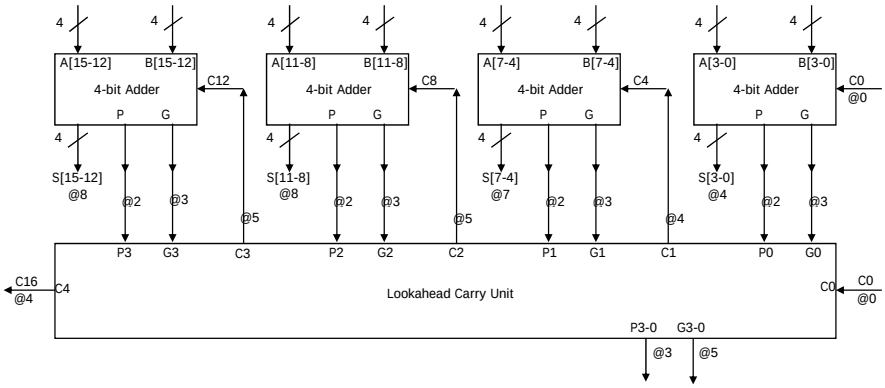






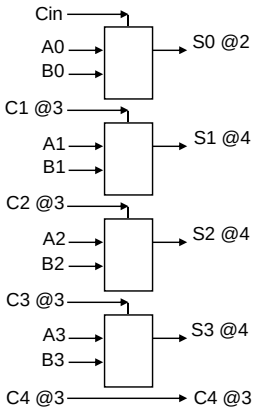
Cascaded carry-lookahead adder

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



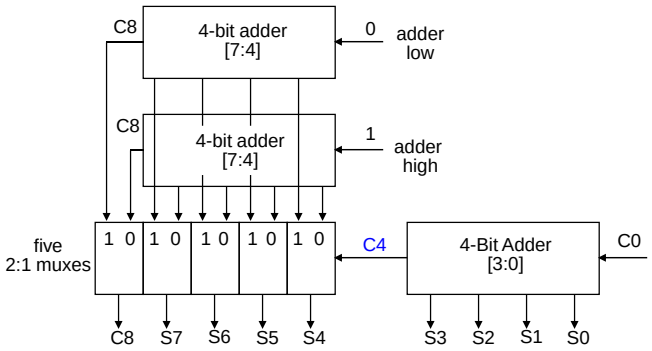
Carry lookahead logic summary

- 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



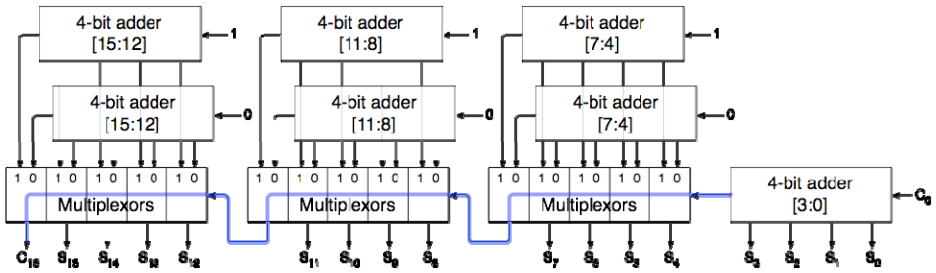
Another solution: Carry-select adder

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



Scaling of carry-select adders

- Size: roughly twice the size of a ripple-carry
  Delay: delay through a 4-bit ripple-carry plus the multiplexor path highlighted in blue (3 2-1 multiplexors, in this example)



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