Ripple Adder

- e.g., 4-bit adder (MSB is right)
Signal Delay

- Ripple Adder slow – carry needs to propagate
- linear in number of bits

- speed up: Carry Lookahead Adder
  - extra (more complex) circuits to determine carry
  - gates can switch in parallel
  - hierarchical application
Carry Lookahead

- determine carry bits in parallel to main addition
- unroll sequential computation
- AND and OR can have more than 2 inputs

- basic observation about bit pair A,B:
  - Carry Generate: $G(A,B) = A \land B$
  - Carry Propagate: $P(A,B) = A \lor B$
**4-Bit Carry Computation**

- **input:** $A_0...A_3, B_0...B_3, C_0$
- **output:** $C_1...C_4$
- **intermediate:** $G_0...G_3, P_0...P_3$

\[
\begin{align*}
C_1 &= G_0 \lor P_0 \land C_0 \\
C_2 &= G_1 \lor P_1 \land C_1 = G_1 \lor P_1 \land G_0 \lor P_1 \land P_0 \land C_0 \\
C_3 &= G_2 \lor P_2 \land C_2 = \ldots \\
C_4 &= G_3 \lor P_3 \land C_3 = \ldots
\end{align*}
\]
4-Bit Carry Computation

- \( C_4 = G_3 \lor \)
  \[ P_3 \land G_2 \lor \]
  \[ P_3 \land P_2 \land G_1 \lor \]
  \[ P_3 \land P_2 \land P_1 \land G_0 \lor \]
  \[ P_3 \land P_2 \land P_1 \land P_1 \land C_0 \]

  - with

- \( G_x = A_x \land B_x \)

- \( P_x = A_x \lor B_x \)
Computing $C_4$
Partial Full Adder

- Note: \((A \land B) \lor (A \lor B) = (A \land B) \lor (A \oplus B)\)
  - \(\implies\) Can use \(P = A \oplus B\)
Carry Lookahead Adder

- Step 1: Compute all $P_i, G_i$
- Step 2: Compute all $C_i$
- Step 3: Compute all $S_i$

- in practice: limited to 4 bits
  - scheme can be used recursively/hierarchical
Principles

- sequential execution
  - turned into parallel execution
  - trade-off: number of gates vs. speed
Basic Memory Circuit – Flip-flop gate

- stored bit is at Q;
- Output depends on the inputs \textit{and} the previous output (stored bit)
- S – set, R – reset
Revisit Binary Addition

- If you are building circuits to handle the data, then you have a limit to the number of bits available to represent values
- fixed width n-bit representation: *overflow*
  - modular arithmetic
  - 4 bits: $14 + 4 = 2$
Sign Representation

- fixed width n-bit representation
  - most significant bit: left-most (highest value)
  - least significant bit: right-most (lowest value)
- sign extension: treat MSB as sign
  - 0 means positive, 1 means negative
- two zeros: 0000 and 1000
- cannot use basic addition
  - e.g. 3-1 = -4 ??
Ones' Complement

- negative number: invert bits
- still two zeros: 0000 and 1111
- addition possible
  - add carry-over to sum
Arithmetic

00001101    13
+ 11111011    - 4
= 100001000    8 ?
+ 1
= 00001001    9
Two's Complement

- negative number: invert bits and add 1
- single zero: 0000
- range: $-2^{n-1} \ldots -2^{n-1}$
- straightforward addition
Arithmetic

- 00001101        13
- + 11111100       - 4
- =100001001       9

• ignore carry over
Overflow

- assume 8-bit integers in two's complement
  - $100 + 50 - 25 = ?$
  - $100 + (50 - 25) = 100 + 25 = 125$
  - $(100 + 50) - 25 = -106 - 25 = -131 = 125$
More Arithmetic

- addition, subtraction – done
- multiplication, division? – *Not In Course*
- integer vs. fraction?
Shift Operations

- shift bitstring to left or right
  - with or without carry-over
  - simplest: no carry-over
- equivalent to multiplication/division by 2
- very fast machine instructions
- programming languages, operators $\ll$ and $\gg$
  - $a \ll b \quad a \times 2^b$
  - $a \gg b \quad a \div 2^b$