Announcements

- Midterm 2: Thurs. Nov. 1st, 6:30-8:00pm, 60 Evans
  - Exam starts at 6:30pm sharp
  - Review session: Tues., Oct. 30th, 6pm, 550 Cory
- Project phase 2 out this Thurs., due next Fri.

Class Material

- Last lecture
  - Dynamic logic
- Today’s lecture
  - Adders
- Reading
  - Chapter 11

An Intel Microprocessor

Itanium has 6 64-bit integer execution units like this

Bit-Sliced Design

Tile identical processing elements
Itanium Integer Datapath

Data Paths Are Thermal Hogs

Full-Adder

The Binary Adder

Express Sum and Carry as a function of P, G, K

Simplest Adder: Ripple-Carry
**Complementary Static CMOS Full Adder: “Direct” Implementation**

![Diagram of Complementary Static CMOS Full Adder]

**Minimize Critical Path by Reducing Inverting Stages**

![Diagram of Minimize Critical Path]

**Inversion Property**

\[ \tilde{S}(A, B, C_i) = S(A, \bar{B}, \bar{C_i}) \]
\[ \tilde{C}_o(A, B, C_i) = C_o(A, B, \bar{C_i}) \]

**A Better Structure: The Mirror Adder**

![Diagram of A Better Structure]

**Sizing the Mirror Adder: Fanout**

- Since LE of carry gate is 2, want f of 2 to get EF of 4
- Use min. size sum gates to reduce load on carry.
- Total load on carry gate is:
  \[ C_{\text{total}} = C_{\text{C0}} + (6+6+9) \]
  \[ C_{\text{total}} = 2C_{\text{C0}} \]
Sizing the Mirror Adder

\[ \text{C}_{\text{load}} = \text{C}_{\text{Ci}} + (6+6+9) = 2\text{C}_{\text{Ci}} \]
\[ \Rightarrow \text{C}_{\text{Ci}} = 21 \]

Minimum size G and K stacks to reduce diffusion loading

Mirror Adder Summary

- The NMOS and PMOS chains are completely symmetrical. Maximum of two series transistors in the carry-generation gate.
- When laying out the cell, the most critical issue is the minimization of the capacitance at node \( C_0 \). Reduction of the diffusion capacitances is particularly important.
- Carry signals are critical - transistors connected to \( C_i \) are placed closest to the output.
- Only the transistors in the (propagate) carry chain have to be optimized for speed. All transistors in the sum stage can be minimal size.

Manchester Carry Chain

Carry-Bypass Adder

Idea: If \( (P0 \text{ and } P1 \text{ and } P2 \text{ and } P3 = 1) \) then \( C_{out} = C_0 \), else “kill” or “generate.”

Dynamic Manchester Carry Chain
Carry-Bypass Adder (cont.)

\[ t_{\text{adder}} = t_{\text{setup}} + (M-1)t_{\text{carry}} + \left(\frac{N}{M}\right)t_{\text{bypass}} + (M-1)t_{\text{carry}} + t_{\text{sum}} \]

Carry Ripple versus Carry Bypass

Linear Carry Select

Square Root Carry Select
Adder Delays - Comparison

Many Kinds of Tree Adders
- Many ways to construct these tree (or "carry lookahead") adders
  - Many of these variations named after the people who first came up with them
- Most of these vary three basic parameters:
  - Radix: how many bits are combined in each PG gate
    - Previous example was radix 2; often go up to radix 4
  - Tree Depth: how many stages of logic you go through to get the final carry. Must be at least \( \log_2(N) \)
  - Fanout: Maximum logical branching in the tree

Logarithmic (Tree) Adders – Basic
- "Look ahead" across groups of multiple bits to figure out the carry
  - Example with two bit groups:
    \[ P_{1:0} = P_1 \cdot P_0, G_{1:0} = G_1 + P_1 \cdot G_0 \rightarrow C_{out1} = G_{1:0} + P_{1:0} \cdot C_{in} \]
- Combine these groups in a tree structure:
  - Delay is now \( ~\log_2(N) \)
  - Instead of \( ~N \)

Rest of the Tree
- Previous picture shows only half of the algorithm
  - Need to generate carries at individual bit positions too
- Brent-Kung Tree

Tree Adders
- 16-bit radix-2 Kogge-Stone tree
Tree Adders

16-bit radix-2 sparse tree with sparseness of 2

Tree Adders

16-bit radix-4 Kogge-Stone Tree

Next Lecture

- Domino Logic