Announcements

- Today: wrap up dynamic circuits, start discussing arithmetic circuits
- Homework 6 – due next Monday
- Lab 5 (simulation) – starts this week
- Midterms handed back on Wednesday
A Generic Digital Processor
Building Blocks for Digital Architectures

Datapath (Arithmetic Unit)
- Bit-sliced datapath (adder, multiplier, shifter, comparator, etc.)

Memory
- RAM, ROM, Buffers, Shift registers

Control
- Finite state machine (PLA, random logic.)
- Counters

Interconnect
- Switches
- Arbiters
- Bus
Bit-Sliced Design

Tile identical processing elements
Full Adder

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$C_i$</th>
<th>$S$</th>
<th>$C_o$</th>
<th>Carry status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>delete</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>delete</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>propagate</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>propagate</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>propagate</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>propagate</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>generate</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>generate</td>
</tr>
</tbody>
</table>
The Binary Adder

\[ S = A \oplus B \oplus C_i \]
\[ = A\overline{B}\overline{C}_i + \overline{A}B\overline{C}_i + \overline{A}\overline{B}C_i + ABC_i \]
\[ C_0 = AB + BC_i + AC_i \]
The Ripple-Carry Adder

Worst case delay linear with the number of bits

\[ t_d = O(N) \]

\[ t_{adder} \approx (N - 1)t_{carry} + t_{sum} \]

Goal: Make the fastest possible carry path circuit
Complimentary Static CMOS Full Adder

28 Transistors
A Closer Look

- **Drawbacks**
  - Tall PMOS Stack
    - Slows down circuit
  - $C_o$ load is 2 diffusion and 6 gate capacitances
  - $C_i$ goes through the extra output inverter to $C_o$
    - Could optimize with next stage
  - Sum generation has extra inverter on output
    - Not the critical path

- **Positive**
  - $C_i$ closest to output node
Inversion Property

\[ S(A, B, C_i) = S(\overline{A}, \overline{B}, \overline{C_i}) \]

\[ \overline{C_o}(A, B, C_i) = C_o(\overline{A}, \overline{B}, \overline{C_i}) \]
Minimize Critical Path by Reducing Inverting Stages

Exploit Inversion Property

Note: need 2 different types of cells
Applying Inversion Property

With the next stage, invert A and B. You will get as outputs S and C... so take away inverters on these outputs.

Invert A and B inputs

To Ci

Co
Express Sum and Carry as Function of P, G, D

Define 3 new variable which ONLY depend on A, B

**Generate (G) = AB**  \[ C_0 = 1 \text{ if } G = 1 \]

**Propagate (P) = A \oplus B**  \[ C_0 = C_i \text{ if } P = 1 \]

**Delete = A \bar{B}**  \[ C_0 = 0 \text{ if } D = 1 \]

\[ C_o(G, P) = G + PC_i \]

\[ S(G, P) = P \oplus C_i \]

Can also derive expressions for S and \( C_o \) based on \( D \) and \( P \)
A Better Structure: the Mirror Adder

24 transistors
The Mirror Adder I

• The NMOS and PMOS chains are completely symmetrical. This guarantees identical rising and falling transitions if the NMOS and PMOS devices are properly sized. A maximum of two series transistors can be observed in the carry-generation circuitry.

• When laying out the cell, the most critical issue is the minimization of the capacitance at node $C_o$. The reduction of the diffusion capacitances is particularly important.

• The capacitance at node $C_o$ is composed of four diffusion capacitances, two internal gate capacitances, and six gate capacitances in the connecting adder cell.
The Mirror Adder II

- The transistors connected to $C_i$ are placed closest to the output.
  - Fastest for late arriving inputs, $C_i$ tends to arrive late
- Only the transistors in the carry stage have to be optimized for optimal speed. All transistors in the sum stage can be minimal size.
Adder Architectures

- In addition to optimizing each full adder cell and exploiting inversion property, we can also reorganize the add computation to speed things up.
- Basic idea is to overlap propagating the carry with computing the Propagate and Generate functions.
- Discuss three basic architectures:
  - Carry-Bypass
  - Carry-Select
  - Carry-Lookahead
 Carry-Bypass Adder

Idea: If \((P0 \text{ and } P1 \text{ and } P2 \text{ and } P3 = 1)\) then \(C_{o3} = C_0\), else “kill” or “generate”. 
Carry-Bypass Adder (cont.)

Note that this is done at the expense of a MUX in the carry delay path !!
Carry Ripple vs. Carry Bypass

Essentially greater than 4 bits is needed to overcome the overhead of the MUX
Carry-Select Adder

Evaluate possibilities for both $C_i = 1$ and $C_i = 0$ and then select when $C_i$ comes in.

Results in about 30% extra transistors
Carry Select Adder: Critical Path

Bit 0-3

Setup

"0" Carry

"1" Carry

Multiplexer

Sum Generation

S₀-₃

Bit 4-7

Setup

"0" Carry

"1" Carry

Multiplexer

Sum Generation

S₄-₇

Bit 8-11

Setup

"0" Carry

"1" Carry

Multiplexer

Sum Generation

S₈-₁₁

Bit 12-15

Setup

"0" Carry

"1" Carry

Multiplexer

Sum Generation

S₁₂-₁₅
Linear Carry Select

\[ t_{add} = t_{setup} + \left( \frac{N}{M} \right) t_{carry} + Mt_{mux} + t_{sum} \]
Carry-Select Adder Observations

- The inputs to the final multiplexer are steady long before the Mux select (Ci) arrives
  - Path is the same as is the number of bits
- Would be helpful to try and even out the delays so that the critical path is balanced between inputs and Mux select.
  - Make logic simpler with the least significant bits by reducing the number of bits handled in the FA or half adder (HA). HA is FA without Ci (2 ins, 2 outs)
  - Add bits progressively as you move to the MSB
Square Root Carry Select

\[ t_{\text{add}} = t_{\text{setup}} + P \cdot t_{\text{carry}} + (\sqrt{2N}) t_{\text{mux}} + t_{\text{sum}} \]
Adder Delays: Comparison

- Ripple adder
- Linear select
- Square root select

Graph showing the comparison of adder delays with respect to the number of inputs (N) and delay time (tp).
Carry Look Ahead: Basic Idea

\[ A_0, B_0 \quad A_1, B_1 \quad \ldots \quad A_{N-1}, B_{N-1} \]

\[ C_{i,0} \quad P_0 \quad C_{i,1} \quad P_1 \quad \ldots \quad C_{i,N-1} \quad P_{N-1} \]
Look-Ahead: Topology

- No more than $N = 4$ bits
- Delay still increases linearly with number of bits
- Capacitance, resistance too high for $N > 4$
Binary Multiplication

\[
Z = \overline{X} \times Y = \sum_{k=0}^{M \cdot N - 1} Z_k 2^k \\
= \left( \sum_{i=0}^{M-1} X_i 2^i \right) \left( \sum_{j=0}^{N-1} Y_j 2^j \right) \\
= \sum_{i=0}^{M-1} \left( \sum_{j=0}^{N-1} X_i Y_j 2^{i+j} \right)
\]

with

\[
X = \sum_{i=0}^{M-1} X_i 2^i \\
Y = \sum_{j=0}^{N-1} Y_j 2^j
\]
Binary Multiplication

\[ \begin{array}{cccccc}
0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 \\
\times & 1 & 0 & 1 & 1 \\
\hline
1 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
+ & 1 & 0 & 1 & 0 & 1 & 0 \\
\hline
1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\
\end{array} \]

AND operation

Partial Products
The Array Multiplier
The MxN Array Multiplier: Critical Path

\[ t_{mult} \approx [(M - 1) + (N - 2)] t_{carry} + (N - 1) t_{sum} + t_{and} \]
Adder Cells in Array Multiplier

Identical Delays for Carry and Sum
Multiplier Floorplan

X and Y signals are broadcasted through the complete array.

(         )
Array Multiplier Reflections

- Many equal critical paths
  - Very hard to optimize by transistor sizing
- We could pass the carry bits diagonally down instead of across
  - Output does not change
  - Need to add an extra stage to accommodate this
Carry Save Multiplier

Vector Merging Adder

\[ t_{mult} = (N-1)t_{carry}^+ + t_{and}^+ + t_{merge} \]
The Tree Multiplier

- Note that the partial products layout looks as follows:

  ![Diagram of partial products layout]

- Note that we can rearrange and add the partial products differently
- Reduce number of adder circuits and logic depth
- FA compresses 3b to 2b, HA has 2b in and 2b out
Tree Multiplier

- Re arranging

1st Stage
Half Adders

6 5 4 3 2 1 0
Tree Multiplier

- Re arranging

<table>
<thead>
<tr>
<th>1st Stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2nd Stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

- Ellipses represent the processing elements in each stage.
Tree Multiplier

* Re arranging

1st Stage

2nd Stage Full Adders
Tree Multiplier

- Re arranging

1st Stage

2nd Stage Full Adders

3rd Stage

6 5 4 3 2 1 0

6 5 4 3 2 1 0

6 5 4 3 2 1 0
Tree Multiplier

- Re arranging

1st Stage

2nd Stage Full Adders

3rd Stage Half Adders

6 5 4 3 2 1 0

6 5 4 3 2 1 0
Wallace-Tree Multiplier

Partial Products:
- $X_3Y_3$
- $X_2Y_2$
- $X_1Y_2$
- $X_3Y_0$
- $X_1Y_1$
- $X_2Y_0$
- $X_0Y_1$
- $X_0Y_0$

First Stage:
- HA
- HA

2nd Stage:
- FA
- FA
- FA
- HA

Final Adder:
- $Z_7$
- $Z_6$
- $Z_5$
- $Z_4$
- $Z_3$
- $Z_2$
- $Z_1$
- $Z_0$

Digital Integrated Circuits
Arithmetic
© Prentice Hall 1995
Multipliers: Summary

- Optimization goals different than Adder
  - Identify critical path
  - More system level optimization then individual cell optimization