Lecture 12: Adders
Outline

- Single-bit Addition
- Carry-Ripple Adder
- Carry-Skip Adder
- Carry-Lookahead Adder
- Carry-Select Adder
- Carry-Increment Adder
- Tree Adder
Single-Bit Addition

Half Adder

\[ S = A \oplus B \]
\[ C_{out} = A \cdot B \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>(C_{out})</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

Full Adder

\[ S = A \oplus B \oplus C \]
\[ C_{out} = MAJ(A, B, C) \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>(C_{out})</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
For a full adder, define what happens to carries (in terms of A and B)

- Generate: $C_{out} = 1$ independent of C
  - $G = A \cdot B$
- Propagate: $C_{out} = C$
  - $P = .$
- Kill: $C_{out} = 0$ independent of C
  - $K = \sim A \cdot \sim B$
Full Adder Design I

- Brute force implementation from eqns

\[ S = A \oplus B \oplus C \]

\[ C_{\text{out}} = \text{MAJ}(A, B, C) \]
Full Adder Design II

- Factor $S$ in terms of $C_{out}$
  
  $$S = ABC + (A + B + C)(\sim C_{out})$$

- Critical path is usually $C$ to $C_{out}$ in ripple adder
Clever layout circumvents usual line of diffusion
  – Use wide transistors on critical path
  – Eliminate output inverters
Full Adder Design III

- Complementary Pass Transistor Logic (CPL)
  - Slightly faster, but more area
Dual-rail domino

- Very fast, but large and power hungry
- Used in very fast multipliers
N-bit adder called CPA

- Each sum bit depends on all previous carries
- How do we compute all these carries quickly?
Carry-Ripple Adder

- Simplest design: cascade full adders
  - Critical path goes from $C_{in}$ to $C_{out}$
  - Design full adder to have fast carry delay

```
A_4 B_4
A_3 B_3
A_2 B_2
A_1 B_1

C_{out} + --------- + --------- + --------- + --------- + C_{in}
S_4 C_3 S_3 C_2 S_2 C_1
```
Inversions

- Critical path passes through majority gate
  - Built from minority + inverter
  - Eliminate inverter and use inverting full adder
Generate / Propagate

1. Equations often factored into $G$ and $P$
2. Generate and propagate for groups spanning $i:j$

$$G_{i:j} =$$
$$P_{i:j} =$$

3. Base case

$$G_{i:i} =$$
$$G_{0:0} =$$
$$P_{i:i} =$$
$$P_{0:0} =$$

4. Sum:

$$S_i =$$
PG Logic

1: Bitwise PG logic

2: Group PG logic

3: Sum logic

17: Adders
$G_{i:0} = G_i + P_i \cdot G_{i-1:0}$
Carry-Ripple PG Diagram

\[ t_{\text{ripple}} = \]
PG Diagram Notation

Black cell

\[
i:k \quad k-1:j
\]

\[
i:j
\]

Gray cell

\[
i:k \quad k-1:j
\]

\[
i:j
\]

Buffer

\[
i:j
\]
Carry-Skip Adder

- Carry-ripple is slow through all N stages
- Carry-skip allows carry to skip over groups of n bits
  - Decision based on n-bit propagate signal

\[
\begin{array}{c}
\text{P}_{16:13} \quad \text{A}_{16:13} \quad \text{B}_{16:13} \\
\text{S}_{16:13}
\end{array}
\quad \begin{array}{c}
\text{P}_{12:9} \quad \text{A}_{12:9} \quad \text{B}_{12:9} \\
\text{S}_{12:9}
\end{array}
\quad \begin{array}{c}
\text{P} \quad \text{A}_{8:5} \quad \text{B}_{8:5} \\
\text{S}_{8:5}
\end{array}
\quad \begin{array}{c}
\text{P} \quad \text{A}_{4:1} \quad \text{B}_{4:1} \\
\text{S}_{4:1}
\end{array}
\]

\[
\text{C}_{12} \quad \text{C}_8 \quad \text{C}_4 \quad \text{C}_{\text{in}}
\]

\[
\begin{array}{c}
1 \\
0 \\
1 \\
0 \\
1 \\
0 \\
1 \\
0
\end{array}
\]

\[
\begin{array}{c}
1 \\
0 \\
1 \\
0 \\
1 \\
0 \\
1 \\
0
\end{array}
\]
For k n-bit groups (N = nk)

\[ t_{\text{skip}} = \]
Variable Group Size

Delay grows as $O(\sqrt{N})$
Carry-Lookahead Adder

- Carry-lookahead adder computes $G_{i:0}$ for many bits in parallel.
- Uses higher-valency cells with more than two inputs.
Higher-Valency Cells

\[ i:k \quad k-1:l \quad l-1:m \quad m-1:j \]

\[ G_{i:k} \quad P_{i:k} \quad G_{k-1:l} \quad P_{k-1:l} \quad G_{l-1:m} \quad P_{l-1:m} \quad G_{m-1:j} \quad P_{m-1:j} \]

\[ G_{i:j} \quad P_{i:j} \]
Carry-Select Adder

- Trick for critical paths dependent on late input $X$
  - Precompute two possible outputs for $X = 0, 1$
  - Select proper output when $X$ arrives
- Carry-select adder precomputes n-bit sums
  - For both possible carries into n-bit group
Factor initial PG and final XOR out of carry-select

\[ t_{\text{increment}} = \]
Also buffer noncritical signals
Tree Adder

- If lookahead is good, lookahead across lookahead!
  - Recursive lookahead gives $O(\log N)$ delay
- Many variations on tree adders
Kogge-Stone

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0


15:0 14:0 13:0 12:0 11:0 10:0 9:0 8:0 7:0 6:0 5:0 4:0 3:0 2:0 1:0 0:0
Tree Adder Taxonomy

- Ideal N-bit tree adder would have
  - $L = \log N$ logic levels
  - Fanout never exceeding 2
  - No more than one wiring track between levels

- Describe adder with 3-D taxonomy ($l, f, t$)
  - Logic levels: $L + l$
  - Fanout: $2^f + 1$
  - Wiring tracks: $2^t$

- Known tree adders sit on plane defined by
  $$l + f + t = L - 1$$
Tree Adder Taxonomy

- Adder Types:
  - Kogge-Stone
  - Brent-Kung
  - Sklansky

- Taxonomy Structure:
  - Fanout (f)
  - Wire Tracks (t)
  - Logic Levels (l)

- Key Levels:
  - 0 (1)
  - 1 (2)
  - 2 (4)
  - 3 (8)
  - 4 (16)
  - 5 (32)
  - 6 (64)
  - 7 (128)
  - 8 (256)
  - 9 (512)

- Example:
  - Kogge-Stone adder typically uses fewer logic levels than Brent-Kung or Sklansky.
Han-Carlson
17: Adders

CMOS VLSI Design 4th Ed.
Adder architectures offer area / power / delay tradeoffs. Choose the best one for your application.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Classification</th>
<th>Logic Levels</th>
<th>Max Fanout</th>
<th>Tracks</th>
<th>Cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carry-Ripple</td>
<td></td>
<td>N-1</td>
<td>1</td>
<td>1</td>
<td>N</td>
</tr>
<tr>
<td>Carry-Skip n=4</td>
<td></td>
<td>N/4 + 5</td>
<td>2</td>
<td>1</td>
<td>1.25N</td>
</tr>
<tr>
<td>Carry-Inc. n=4</td>
<td></td>
<td>N/4 + 2</td>
<td>4</td>
<td>1</td>
<td>2N</td>
</tr>
<tr>
<td>Brent-Kung</td>
<td>(L-1, 0, 0)</td>
<td>2log₂N – 1</td>
<td>2</td>
<td>1</td>
<td>2N</td>
</tr>
<tr>
<td>Sklansky</td>
<td>(0, L-1, 0)</td>
<td>log₂N</td>
<td>N/2 + 1</td>
<td>1</td>
<td>0.5 Nlog₂N</td>
</tr>
<tr>
<td>Kogge-Stone</td>
<td>(0, 0, L-1)</td>
<td>log₂N</td>
<td>2</td>
<td>N/2</td>
<td>Nlog₂N</td>
</tr>
</tbody>
</table>