EE241 - Spring 2003
Advanced Digital Integrated Circuits

Lecture 20
Multipliers

Binary Multiplication

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

with

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

<p>| 1 0 1 0 1 0 | N bits |
| × | 1 0 1 1 | M bits |
|-----------------|--------|
| 1 0 1 0 1 0 | AND operation |
| 0 0 0 0 0 0 | Partial Products |</p>
<table>
<thead>
<tr>
<th>+</th>
<th>1 0 1 0 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 0 0 1 1 0</td>
<td>N+ M bits in the final sum</td>
</tr>
</tbody>
</table>

### Shift-and-Add Multiplier

- Standard adder and shift-in the multiplicand
- Shift the result as well and add
- N cycles
- Parallel adders add more hardware (adders) instead.
Array Multiplier

MxN Array Multiplier — Critical Path

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

Identical Delays for Carry and Sum

Carry-Save Multiplier

Vector Merging Adder

\[ t_{\text{mult}} = (N-1) t_{\text{carry}} + (N-1) t_{\text{and}} + t_{\text{merge}} \]
X and Y signals are broadcasted through the complete array.

Multipliers

- Partial product generation
- Partial product accumulation
- Final summation
Generating Partial Products

- All partial products: AND

- Booth’s recoding – reduction of partial product count

---

Booth Recoding

- Instead of generating all the partial products
  \[ 0 \times x = 0, \quad 1 \times x = x, \quad x = \{0, 1\} \]

- Reduce the number of partial products by grouping

\[
\begin{array}{cccccc}
0 & 0 & 0 \\
0 & 1 & 1^* \\
1 & 0 & 2^* \quad \text{(shift)} \\
1 & 1 & 3^* \quad \text{(or 4\(^*\) -1)}
\end{array}
\]

Booth’51
Booth Recoding

- Instead of using set \{0, 1*Y, 2*Y, 3*Y\}
- Use \{0, 1*Y, 2*Y, 4*Y, -Y\}
- Shifting and complementing
  - \(3*Y = 4*Y - Y\)
- Can be simplified by looking into three bits – modified Booth recoding

Modified Booth Recoding

Two bits and the MSB of previous two

<table>
<thead>
<tr>
<th>(x_i + 2x_{i+1}x_{i-1})</th>
<th>Add to partial product</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>+0Y</td>
</tr>
<tr>
<td>001</td>
<td>+1Y</td>
</tr>
<tr>
<td>010</td>
<td>+1Y</td>
</tr>
<tr>
<td>011</td>
<td>+2Y</td>
</tr>
<tr>
<td>100</td>
<td>-2Y</td>
</tr>
<tr>
<td>101</td>
<td>-1Y</td>
</tr>
<tr>
<td>110</td>
<td>-1Y</td>
</tr>
<tr>
<td>111</td>
<td>-0Y</td>
</tr>
</tbody>
</table>
Accumulating Partial Products

Partial product matrix

Reorganized matrix

Wallace-Tree Multiplier
Wallace-Tree Multiplier

Wallace, Trans on Comp. 2/64
Tree Multipliers

- Time is proportional to log N
- Wiring is complicated
- Different wire lengths
- Optional pipelining
- Wallace tree: reduce the number of operands at earliest opportunity
- Dadda tree: reduce the number of operands with fewest adders

Minimum Number of Stages

<table>
<thead>
<tr>
<th>Number of bits in the multiplier</th>
<th>Number of stages</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>6 ≤ n ≤ 9</td>
<td>3</td>
</tr>
<tr>
<td>9 ≤ n ≤ 13</td>
<td>4</td>
</tr>
<tr>
<td>13 ≤ n ≤ 16</td>
<td>5</td>
</tr>
<tr>
<td>16 ≤ n ≤ 19</td>
<td>6</td>
</tr>
<tr>
<td>20 ≤ n ≤ 28</td>
<td>7</td>
</tr>
<tr>
<td>28 ≤ n ≤ 42</td>
<td>8</td>
</tr>
<tr>
<td>42 ≤ n ≤ 63</td>
<td>9</td>
</tr>
</tbody>
</table>

Dadda, '65
Generalized Counters

Stenzel, Trans on Comp 10/77

UEC Berkeley EE241
B. Nikolic, J. Rabaey
Generalized Counters

32x32b using (5,5,4) with (3,2) in the last stage

4:2 Counters (Compressors)

4-2 carry-save module

Truth table for the 4:2 adder: 

<table>
<thead>
<tr>
<th>n</th>
<th>Cin</th>
<th>Cout</th>
<th>Carry</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

*Either Cout or Carry may be one, for two or three inputs equal to 1 but NOT both.

Cout may NOT be a function of the Cout from the adjacent block or a ripple carry may occur.
4:2 Compressors

Built of CSAs

Pipelined version compresses 8 partial products per cycle

Interconnect can be more regular than in Wallace tree
Three Dimensional Optimization

Oklobdzija, Vileger, Liu, Trans on Comp 3/96

Vertical Slices in TDM

UC Berkeley EE241  B. Nikolic, J. Rabaey
Final Addition

Latest-Fewest Output Profile For TDM FFRT

Optimal Uniform Input Adder
Hybrid Ripple-Carry/1-level Carry-Skip/Carry Select
Final Addition

Example: CPL Multiplier

Block Diagram

Critical Path

Yano, JSSC 4/90
Example: DPL Multiplier

Booth encoder

Partial product generator
Example: DPL Multiplier

FA-based 4:2

Modified 4:2

Example: DPL Multiplier

Tree construction

P: 4x4 Partial product generators
C: 4:2 Compressor
M: Half adder
F: Full adder
Example: DPL Multiplier

Regularly Structured Tree

4W: 4-2 compressor
3W: 1-b full adder
P: Partial product-bit generator

Goto, JSSC 9/92
Regularly Structured Tree
Regularly Structured Tree

Itoh, JSSC 2/01
Regularly Structured Tree