Pushing the Limits of CMOS Technology: A Wave-Pipelined Multiplier

Fabian Klass, Michael Flynn, and Ad J. van de Goor
Stanford University
Delft University

Conventional Pipelining
New data are applied after previous data are available:

Pipeline rate is limited by the maximum delay of one logic section + the clocking overhead
Wave Pipelining

New data are applied before previous data are available:

Pipeline rate is limited by the difference between maximum and minimum delay across the logic section + the clocking overhead

Advantages of Wave Pipelining

- Achieve very high pipeline rates, approaching the physical speed limit of the technology
- Increase pipeline rate without significant latency increase
- Minimize clocking overhead
- Use fewer registers thus saving area
- Reduce clock distribution problems
Disadvantages of Wave Pipelining

- Require path delay equalization to maximize pipeline rate
- Clock signals must be synchronized with the waves
- Process and temperature variations are more critical than in conventional systems
- Difficult to handle at system level

Delay Variations

Sources of delay variation in digital circuits:

- Difference in path lengths (gate level)
- Different input patterns (transistor level)
- Coupling capacitance effects
- Power supply changes (noise)
- Fabrication process variations
- Temperature changes
Path Equalization Techniques

Delay elements are inserted to increase short path delays:

![Diagram of unbalanced and balanced delay elements with a rough tuning adjustment](image1)

Gate delays are tuned to equalize path delays:

![Diagram of unbalanced and balanced delay elements with a fine tuning adjustment](image2)

Delay in CMOS

The delay of a CMOS gate varies substantially with different input patterns:

![Diagram of a 2-NAND gate with superimposed waveforms](image3)
CMOS Wave Pipelining

By using proper design constraints, e.g., 2-NAND and inverters, and path equalization techniques, the overall delay variation of a CMOS circuit is relatively small, between 10-20%, despite substantial delay variations of individual gates of about 60%.

Example

Delay variations cancel out, as if all gates have average delay.

16x16-bit Multiplier

Algorithm

[Diagram showing the 16x16-bit multiplier algorithm and architecture]
(4,2) Counter

It is the building block of the partial-product tree:

![Block Diagram](image)

<table>
<thead>
<tr>
<th>Logic Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cin</td>
</tr>
<tr>
<td>+</td>
</tr>
<tr>
<td>C</td>
</tr>
<tr>
<td>C</td>
</tr>
</tbody>
</table>

Row of (4,2) counters

(4,2) Counter Implementation

<table>
<thead>
<tr>
<th>Padding Elements</th>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
</tr>
</thead>
<tbody>
<tr>
<td>C out</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Minimum Delay Test Vectors</th>
<th>Maximum Delay Test Vectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>0t 0t 0s 1t</td>
<td>0t 1s 1s 0t</td>
</tr>
<tr>
<td>0s 1t 0t 0t</td>
<td>1s 0t 0t 1s</td>
</tr>
<tr>
<td>0t 0t 1t 1s</td>
<td>1t 1t 0t 0s</td>
</tr>
<tr>
<td>1t 0s 0t 0t</td>
<td>1t 1t 0s 0t</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Delay</td>
<td>1.46ns</td>
<td>1.46ns</td>
</tr>
<tr>
<td>Minimum Delay</td>
<td>1.19ns</td>
<td>1.28ns</td>
</tr>
<tr>
<td>Average Delay</td>
<td>1.33ns</td>
<td>1.38ns</td>
</tr>
<tr>
<td>Max-Min Delay</td>
<td>0.28ns</td>
<td>0.18ns</td>
</tr>
</tbody>
</table>

0s (1s) : steady 0 (1)
0t (1t) : transition 1-0 (0-1)
(4,2) Counter Layout

Partial-Product Reduction Tree
Wiring Delays

The output capacitance of a cell in the tree is dominated by the capacitance of the interconnection wires.

\[ C_{\text{Load}} = C_{\text{wire}} + C_{\text{logic}} \]

Delay at stage \( i \):

\[ D_i = \sum d_i + \sum R_i \left( C_{\text{wire}}^{(i+1)} + C_{\text{logic}}^{(i+1)} \right) \]

The overall delay of the tree is balanced since all stages are equal and the sum of the lengths of the interconnection wires is the same along any path.

16-bit Carry Propagate Adder

Block Diagram:

- Input A
- Input B
- g,p Generator
- Carry Generator
- Sum
- Output

Parallel Carry Generator:

- P,G block
- Padding block
Test Chip

Input A  Input B

FIFO       FIFO

WAVE-PIPELINED MULTIPLIER

FIFO

Output

External Clock

Clock Generator & Steering Logic

Control

Simulation Results

Pattern #1

Pattern #2
Wave Pipeline versus Regular Pipeline

<table>
<thead>
<tr>
<th>Pipeline</th>
<th>Latency</th>
<th>Min Clock</th>
<th>Speed-up</th>
<th>Area</th>
<th>Latency x Clock</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>10.1ns</td>
<td>10.1ns</td>
<td>1.0x</td>
<td>A</td>
<td>102ns²</td>
</tr>
<tr>
<td>Wave</td>
<td>10.2ns</td>
<td>1.45ns</td>
<td>7.0x</td>
<td>1.21A</td>
<td>14.8ns²</td>
</tr>
<tr>
<td>Regular</td>
<td>16.2ns</td>
<td>2.7ns</td>
<td>3.7x</td>
<td>1.33A</td>
<td>43.7ns²</td>
</tr>
</tbody>
</table>
Conclusions

Significant performance increase can be achieved in CMOS by using the technique of wave pipelining.

Wave pipelining can achieve a product Latency x Cycle-Time about 2x better than Regular Pipelining.