Truncated Logarithmic Approximation

Michael Sullivan and Earl E. Swartzlander, Jr.
University of Texas

April 10, 2013
Introduction

Approximate arithmetic units have the potential to save power, area, and latency over conventional circuits.

Approximate logarithmic conversion is attractive because it can estimate multiplication, division, rooting, and raising to a power with bounded relative error.

Known application areas include:

- Graphics
- Neural networks
- DSP applications
- Specialized circuitry
  - period meters for nuclear power plants
Example: Approximate division

For example, division of logs can be performed using subtraction.

\[
\log_2(a/b) = \log_2(a) - \log_2(b)
\]

\[
a/b = 2^{(\log_2(a) - \log_2(b))}
\]

Approximate logarithms can be used to cheaply approximate fixed-point division.

\[
a/b \approx 2^{(\tilde{\log}_2(a) - \tilde{\log}_2(b))}
\]
This Work

Many approximate conversion schemes exist (differ in precision, latency, cost, and flexibility).

The **least costly** (and least precise) schemes use piece-wise linear interpolation. All such schemes refine a simple linear interpolation scheme (Mitchell, 1962).

**Intuition:**
The precision of interpolation should be proportional to the precision of the final result.

**The Idea:**
Rounding off the log and anti-log approximation reduces their costs, and can actually improve the average precision of the result.

**Truncated logarithmic approximation** can be used as a drop-in replacement for Mitchell’s scheme, improving its cost and precision.
A fixed-point input, $N$ can be written as $N = 2^k \times (1 + f)$.

- $k$ is the characteristic ($0 \leq k < \log_2(N)$)
- $f$ is the fractional component ($0 \leq f < 1$)

The binary logarithm of $N$ is $\log_2(N) = k + \log_2(1 + f)$. Approximate logarithmic computations approximate $\log_2(1 + f)$ with enough fidelity to achieve a set precision.
Mitchell’s Approximation (cont.)

Mitchell estimates $\log_2(1+f)$ using a single straight-line approximation to the logarithm curve, $f \approx \log_2(1+f)$

![Graph showing Mitchell's log approximation.]

**Figure:** Mitchell’s log approximation.
Mitchell’s Logarithm Generation

(a) Logarithm Generation

Figure: Approximate log and anti-log conversion.
Mitchell Hardware Cost

Hardware costs are dominated by two shifters:

(a) Logarithm Shifter

(b) Anti-Log Shifter (after multiplication)

Figure: Shifters for approximate (anti-)logarithmic conversion.
Mitchell’s Approximation

Figure: The error in Mitchell’s log approximation.
Truncated logarithmic approximation replaces Mitchell’s algorithm, retaining only the $t$ most-significant bits of the fractional component.

\[
\log_{2^{-T}}(X, t) = k + (f \mod 2^{-t}) + 2^{-t} \approx \log_2(X) \tag{1}
\]
Hardware Savings - Log Generation

(a) Logarithm Shifter

(b) Trunc. Log Shift ($t = 4$)

Figure: The impact of truncation on the log generation shifter.
Hardware Savings - Anti-Log Generation

(a) Full Anti-Logarithm Shifter

(b) Truncated Anti-Log Shift ($t = 4$)
Truncated Log Error

(c) Up-Rounded Values

(d) Up-Rounded Error

Figure: The error of upward-rounded truncation.
Cost Savings

(a) Cost Across $t$ ($n=32$)

(b) Cost Across $n$ ($t=4$)

Figure: The relative costs of truncated log gen/anti-gen.
Figure: Exploring the $n/t$ landscape.
Piece-wise Linear Logarithmic Approximation


Using as a Drop-In Replacement for Combet et. al.

(a) Truncated Combet et. al.  (b) Truncated Combet $t$ Exploration

**Figure:** Truncation applied to Combet et. al.’s correction scheme.
A Novel Error Correction Scheme with Truncation

(a) $\log_2 T \uparrow + EC_4 (t_{ec} = 4)$

(b) $\log_2 T \uparrow + EC_4 (t_{ec} = 4)$ cost

Figure: Novel error correction for truncated logarithms.
To Recap (Numerical Results)

Table: Precision and Cost of Truncated Mitchell Analogues ($n = 32$).

<table>
<thead>
<tr>
<th>Technique</th>
<th>Min. Error</th>
<th>Max. Error</th>
<th>Max. Abs. Error</th>
<th>Average Abs. Error</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mitchell</td>
<td>-0.086</td>
<td>0</td>
<td><strong>0.086</strong></td>
<td><strong>0.057</strong></td>
<td>1.0</td>
</tr>
<tr>
<td>$\log_2 T_{\uparrow}(t = 4)$</td>
<td>-0.086</td>
<td>0.062</td>
<td><strong>0.086</strong></td>
<td><strong>0.035</strong></td>
<td>0.57</td>
</tr>
<tr>
<td>$\log_2 T_{\uparrow}EC4(t_{ec} = 2)$</td>
<td>-0.081</td>
<td>0.078</td>
<td>0.081</td>
<td>0.028</td>
<td>0.60</td>
</tr>
<tr>
<td>$\log_2 T_{\uparrow}EC4(t_{ec} = 3)$</td>
<td>-0.066</td>
<td>0.070</td>
<td>0.070</td>
<td>0.024</td>
<td>0.61</td>
</tr>
<tr>
<td>$\log_2 T_{\uparrow}EC4(t_{ec} = 4)$</td>
<td>-0.061</td>
<td>0.066</td>
<td><strong>0.061</strong></td>
<td><strong>0.023</strong></td>
<td>0.63</td>
</tr>
</tbody>
</table>
Truncated approximate logarithms improve piecewise-linear approximate logarithm computations. They are based off of the intuition that the internal precision of a conversion scheme should be proportional to the precision of the approximation.

Benefits:

- Decrease cost (up to \( \sim 50\% \))
- May improve the precision of results
- Amenable to existing error reduction techniques
- May allow unique truncation-specific error reduction
Methodology - Unit Gate Model

Delay and cost (energy) estimated using a unit-gate model:

- Simple 2-input gates (AND, OR) \([C = 1, T = 1]\)
- 2-input XOR gates and MUXes \([C = 2, T = 2]\)
- \(m\)-input gates composed of a tree of 2-input gates

Advantages of the unit gate model:

- Offers a rough technology-agnostic model for circuit efficiency
- Betters understanding of scaling properties, bottlenecks
- Can be used for rapid design-space exploration

Inverters, buffering, and wiring concerns are ignored, but:

- Limited fan-out components are used
- Wiring stays roughly equivalent after truncation