A Parameterizable Floating-Point Logarithm Operator for FPGAs

Jérémie Detrey    Florent de Dinechin

Projet Arénaire – LIP
UMR CNRS – ENS Lyon – UCB Lyon – INRIA 5668
http://www.ens-lyon.fr/LIP/Arenaire/
Outline

▶ Context

▶ Evaluation algorithm

▶ Architecture

▶ Error analysis

▶ Results

▶ Conclusion
Context

- Context
- Evaluation algorithm
- Architecture
- Error analysis
- Results
- Conclusion
Context: Floating-point on FPGAs

- floating-point reaches FPGAs

- basic operators libraries (+, −, ×, ÷, √):
  - Belanović and Leeser, 2002
  - Lee and Burgess, 2002
  - Detrey and de Dinechin, 2003 (FPLibrary)
  - deLorimier and DeHon, 2005
  - Dou et al., 2005

- more and more applications: matrix operations, convolutions, filters

- but slow operators: 10 times slower than typical processor operators
Context: Floating-point elementary functions?

- in current processors, **elementary functions** are **micro-coded** or implemented in **software**

- porting those algorithms to FPGAs would result in very slow operators

- need for **ad-hoc hardware algorithms**
Context: Previous works

- only two references:
  - sine (Ortiz et al., 2003)
  - exponential (Doss and Riley, 2004)

- but...
  - not really targeted to FPGAs
  - too close to software implementation
Context: Motivations

- logarithm operator: first attempt at a library of floating-point elementary functions
- parameterized operator
- compatible with FPLibrary
- use of recent fixed-point evaluation methods
Evaluation algorithm
Evaluation algorithm: Number format

- 2 parameters: $w_E$ (range) and $w_F$ (precision)

- inspired from the IEEE-754 standard:

$$X = (-1)^{S_X} \cdot 1.F_X \cdot 2^{E_X-E_0}$$
Evaluation algorithm: Number format

- 2 parameters: $w_E$ (range) and $w_F$ (precision)

- inspired from the IEEE-754 standard:

$$X = (-1)^{S_X} \cdot 1.F_X \cdot 2^{E_X - E_0}$$

- 2 extra bits for exceptional cases:
  - $\text{exn}_X = 00$: zero, $X = \pm 0$
  - $\text{exn}_X = 01$: regular case
  - $\text{exn}_X = 10$: infinity, $X = \pm \infty$
  - $\text{exn}_X = 11$: Not-a-Number (NaN)
we need to compute $R = \log X$, with $X > 0$
we need to compute $R = \log X$, with $X > 0$

with $X = M \cdot 2^E$, we have:

$$R = \log M + E \cdot \log 2$$
Evaluation algorithm: Fixed-point logarithm
Evaluation algorithm: Fixed-point logarithm

- when $M \approx 1$ and $E = 0$, $R \approx 0$
- possibly large renormalization for $R$
Evaluation algorithm: Fixed-point logarithm

- when $M \approx 1$ and $E = 0$, $R \approx 0$
- possibly large renormalization for $R$
- very large precision ($2w_F$ bits) to evaluate $\log M$
Evaluation algorithm: Fixed-point logarithm

- Taylor formula: $\log M = (M - 1) - \frac{1}{2}(M - 1)^2 + \ldots$
Evaluation algorithm: Fixed-point logarithm

- Taylor formula: \[ \log M = (M - 1) - \frac{1}{2}(M - 1)^2 + \ldots \]

- with \( f(M) = \frac{\log M}{M - 1} \), we have:
  - reconstruction: \( \log M = f(M) \cdot (M - 1) \) where \( M - 1 \) is exact
  - only \( w_F + g_0 \) bits to evaluate \( f(M) \)
Evaluation algorithm: Fixed-point logarithm

- evaluation of $f(M)$ by the Higher-Order Table-Based Method (HOTBM):
  - piecewise polynomial approximation
  - terms computed in parallel
  - ad-hoc powering units
  - optimized look-up tables
  - small multipliers
  - guaranteed faithful rounding
Architecture

► Context

► Evaluation algorithm

► Architecture

► Error analysis

► Results

► Conclusion
Architecture

\[ f(x) = \frac{\log x}{x - 1} \]

\[ \tilde{R} \approx \log X \]
Architecture

- range reduction:
  - comparison with $\sqrt{2}$ on a few bits only
  - right shift of at most one position
Architecture

- range reduction:
  - comparison with $\sqrt{2}$ on a few bits only
  - right shift of at most one position

- sign detection:
  - sign of $E$ when $E \neq 0$
  - sign of $M - 1$ otherwise

$\tilde{R} \approx \log X$
Architecture

- range reduction:
  - comparison with $\sqrt{2}$ on a few bits only
  - right shift of at most one position
- sign detection:
  - sign of $E$ when $E \neq 0$
  - sign of $M - 1$ otherwise
- fixed-point logarithm:
  - HOTBM operator to compute $f(M)$
  - in parallel, compute $M - 1$ and change sign if needed
  - large mult ($\sim w_F \times w_F + g_0$) for reconstruction
**Architecture**

- **range reduction:**
  - comparison with $\sqrt{2}$ on a few bits only
  - right shift of at most one position

- **sign detection:**
  - sign of $E$ when $E \neq 0$
  - sign of $M - 1$ otherwise

- **fixed-point logarithm:**
  - HOTBM operator to compute $f(M)$
  - in parallel, compute $M - 1$ and change sign if needed
  - large mult ($\sim w_F \times w_F + g_0$) for reconstruction

- $E \cdot \log 2$:
  - change sign if needed
  - rectangular mult ($\sim w_E \times w_F + g_1$)
range reduction:
- comparison with $\sqrt{2}$ on a few bits only
- right shift of at most one position

sign detection:
- sign of $E$ when $E \neq 0$
- sign of $M - 1$ otherwise

fixed-point logarithm:
- HOTBM operator to compute $f(M)$
- in parallel, compute $M - 1$ and change sign if needed
- large mult ($\sim w_E \times w_F + g_0$) for reconstruction

$E \cdot \log 2$:
- change sign if needed
- rectangular mult ($\sim w_E \times w_F + g_1$)

final step:
- reconstruction with a single large addition
- normalization using a leading-zero counter
- rounding
Error analysis

- Context
- Evaluation algorithm
- Architecture
- Error analysis
- Results
- Conclusion
Error analysis: Error propagation

\[ f(x) = \frac{\log x}{x - 1} \]

\( M \approx \log X \)

\( \tilde{R} \approx \log X \)
Error analysis: Error propagation

$\epsilon_f$:
- evaluation error for $f(M)$
- multiplied by $M - 1$
Error analysis: Error propagation

► $\epsilon_f$:
  - evaluation error for $f(M)$
  - multiplied by $M - 1$

► $\epsilon_{\log 2}$:
  - discretization error for the constant $\log 2$
  - $g_1$ guard bits: $|\epsilon_{\log 2}| < 2^{-w_F-g_1-1}$
  - multiplied by $E$
Error analysis: Error propagation

\( \epsilon_f \):
- evaluation error for \( f(M) \)
- multiplied by \( M - 1 \)

\( \epsilon_{\log 2} \):
- discretization error for the constant \( \log 2 \)
- \( g_1 \) guard bits: \( |\epsilon_{\log 2}| < 2^{-w_F-g_1-1} \)
- multiplied by \( E \)

final addition
Error analysis: Faithful rounding

▶ table maker’s dilemma: we cannot guarantee correct rounding
Error analysis: Faithful rounding

- table maker’s dilemma: we cannot guarantee correct rounding
Error analysis: Faithful rounding

- table maker’s dilemma: we cannot guarantee correct rounding
Error analysis: Faithful rounding

- **table maker’s dilemma:** we cannot guarantee correct rounding

- **ensure faithful rounding:** one *unit in the last place* error

- from the **constraints** we compute the **number of guard bits**
Error analysis: Faithful rounding

▶ more than 96% of all cases are correctly rounded

▶ single precision \((X > 0\) only):
Results

- Context
- Evaluation algorithm
- Architecture
- Error analysis
- Results
- Conclusion
Results: Operator area

single precision \((w_E, w_F) = (8, 23)\):
1399 slices (27% of a Virtex-II 1000 FPGA)
Results: Operator latency

- single precision \((w_E, w_F) = (8, 23)\): 64 ns
Results: Using embedded multipliers

- **Single precision** \((w_E, w_F) = (8, 23)\):

<table>
<thead>
<tr>
<th></th>
<th>LUT-based mults</th>
<th>1399 slices (27%)</th>
<th>64 ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>18 \times 18 mults</td>
<td>830 slices (16%) 9 mults</td>
<td>61 ns</td>
<td></td>
</tr>
</tbody>
</table>
Results: Pipelined version

- all operators are pipelined at 100 MHz on a Virtex-II 1000 FPGA
- single precision \((w_E, w_F) = (8, 23)\):

<table>
<thead>
<tr>
<th></th>
<th>Combinatorial</th>
<th>1399 slices (27%)</th>
<th>64 ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipelined</td>
<td>1618 slices (31%)</td>
<td>11 cycles</td>
<td></td>
</tr>
</tbody>
</table>
Results: Comparison

single precision comparison with a 2.4 GHz Intel Xeon processor:

<table>
<thead>
<tr>
<th></th>
<th>cycles</th>
<th>latency</th>
<th>throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4 GHz Intel Xeon</td>
<td>196</td>
<td>82 ns</td>
<td>12 Mop/s</td>
</tr>
<tr>
<td>100 MHz Virtex-II FPGA</td>
<td>11</td>
<td>64 ns</td>
<td>100 Mop/s</td>
</tr>
</tbody>
</table>
Conclusion

➢ Context

➢ Evaluation algorithm

➢ Architecture

➢ Error analysis

➢ Results

➢ Conclusion
Conclusion

- floating-point logarithm operator
  - fully parameterized (range and precision)
  - up to single precision
  - part of the FPLibrary operator suite

- careful error analysis
  - faithful rounding
  - optimized data width

- fast operators
  - same latency as in current processors
  - 10 times increase in throughput
Future work

- explore other algorithms to target \textit{double precision}

- implement other \textit{elementary functions}:
  - exponential: already developed (FPT'05)
  - sine
  - cosine
  - ...

Thank you for your attention

more information & download page:
http://www.ens-lyon.fr/LIP/Arenaire/
Thank you for your attention

▶ more information & download page:
  http://www.ens-lyon.fr/LIP/Arenaire/

Questions?