Precision Modeling and Bitwidth Optimization of Floating-Point Applications

Zhihong Zhao
Alternative System Concepts, Inc.
Windham, NH, USA

Miriam Leeser
Northeastern University
Boston, MA, USA
Outline

- Variable Bitwidth Computing
- Precision Modeling Methodology
  - Behavioral Profiling
  - Error Modeling
  - Verification
- Bitwidth Optimization
  - Problem Formulation
  - Optimization Algorithms
  - Optimization Results
- Future Work
- Conclusion
Variable Bitwidth Computing

- Why Variable Bitwidth Computing?

- Obtaining Optimized Bitwidths
  - Other’s approach: simulation-based bitwidth searching
  - Our approach: model-based bitwidth optimization
  - System flowchart of the model-based bitwidth optimization
Why Variable Bitwidth Computing?

- Standard FP representation (Single Precision): 32 bits
  - sign: 1 bit
  - exponent: 8 bits
  - mantissa: 23 bits
- Implementation of standard FP operations in custom circuits is expensive
- Mantissa bitwidth can be reduced without compromising precision requirements
Relative Error VS Bitwidth

Experiment results of three examples

- DCT-1
- DCT-2
- ANN
Simulation-based Searching

- Initial B
- Simulation
- \( B \leq B-1 \)
  - good
    - \( Y \)
  - \( N \)
    - \( B \leq B+1 \)
- Optimal B

> Iterative simulation
> No explicit arithmetic precision model generated or needed
> Time consuming
Model-based Optimization

- No iterative simulation
- Application-specific arithmetic precision model generated
- One bitwidth value for each operation node
**System Flowchart**

- **Test Data**
- **Data Path**
- **Behavioral Profiling**
- **Profile Data**
- **Precision Modeling**
- **Error Range**
- **Bitwidth Optimization**
- **Optimized Bitwidths**
Precision Modeling Methodology

- Behavioral Profiling
- Error Models of FP Operations
  - Rounding Error
  - Propagation Error
- Constructing Precision Model
Profile-Driven Modeling

- Behavioral profiling gathers profile data through one-time simulation
  - bit probability
  - statistical values of variables in data-path
- Profiling is performed on a graphical representation (usually CDFG) of the application
- Profile data is used in precision modeling
- Selecting stimuli is important
Behavioral Profiling

- Watcher insertion & simulation

Diagram:
- Watcher
- CDFG node
- Data
- Stimuli
The overall error of a FP operation:

\[(x \cdot y) \cdot \text{fp}(\hat{x} \cdot \hat{y}) \cdot (x \cdot y) \cdot \text{fp}(\hat{x} \cdot \hat{y}) \cdot (x \cdot y) \cdot \text{fp}(\hat{x} \cdot \hat{y})\]

Overall error at the result of an operation:
Propagation Error(PE) + Rounding Error(RE)
Propagation Error (1)

Calculation of PE

\[ f(x, y) \approx f(\hat{x}, \hat{y}) \approx \frac{f(\hat{x}, \hat{y})}{x} (x \approx \hat{x}) \approx \frac{f(\hat{x}, \hat{y})}{y} (y \approx \hat{y}) \]

\[ \frac{f(x, y)}{f(\hat{x}, \hat{y})} \approx \frac{f'(\hat{x}, \hat{y})\hat{x}}{\hat{f'(\hat{x}, \hat{y})}} \approx \frac{f'(\hat{x}, \hat{y})\hat{y}}{\hat{f'(\hat{x}, \hat{y})}} \approx k_x \approx k_y \]

- \( K \) is the amplification factor, determined based on the operation’s type and data.
Propagation Error (2)

Equation:

\[
\frac{\partial z}{\partial p} \frac{f(x, y) - f(\hat{x}, \hat{y})}{f(x, y)} \quad \frac{\partial f(\hat{x}, \hat{y})}{\partial \hat{x}} \quad \frac{\partial f(\hat{x}, \hat{y})}{\partial \hat{y}} \quad k_x \quad k_y
\]

MULT:

\[
k_x \quad \frac{\partial f(\hat{x}, \hat{y})}{\partial \hat{x}} \quad \frac{\hat{y}}{\hat{y}} \quad 1.0
\]

ADD:

\[
k_y \quad \frac{\partial f(\hat{x}, \hat{y})}{\partial \hat{y}} \quad \frac{\hat{x}}{\hat{y}} \quad 1.0
\]

SQRT:

\[
k_x \quad \frac{\partial f(\hat{x})}{\partial \hat{x}} \quad 0.5
\]
Rounding Error

Mantissa:

\[
\begin{array}{cccccccc}
|a_1|a_2| \cdots |a_b|a_{b+1}| \cdots |a_{22}|a_{23}
\end{array}
\]

Precise value:

\[ z \approx (1.0 ? a_1 2^{?1} ? a_2 2^{?2} ? \quad ? a_b 2^{?b} ? a_{b+1} 2^{?b+1} ? \quad ? a_{22} 2^{?22} ? a_{23} 2^{?23}) ? 2^e \]

FP value:

\[ \hat{z} \approx (1.0 ? a_1 2^{?1} ? a_2 2^{?2} ? \quad ? a_b 2^{?b}) ? 2^e \]

Error:

\[ \frac{z - \hat{z}}{z} \approx \frac{(a_{b+1} 2^{?b+1} ? \quad ? a_{22} 2^{?22} ? a_{23} 2^{?23})}{(1.0 ? a_1 2^{?1} ? a_2 2^{?2} ? \quad ? a_{23} 2^{?23})^2} \approx \frac{p_{b+1} 2^{?b+1} ? \quad ? p_{22} 2^{?22} ? p_{23} 2^{?23}}{1.0 ? p_1 2^{?1} ? p_2 2^{?2} ? \quad ? p_{23} 2^{?23}} \]
Constructing Precision Model

- Establish error model for each node in data-path using the RE+PE model
- Construct precision model for the application based-on data-path structure
- The precision model is a function of output error of the application in terms of bit-widths in data-path and input error of the application
- The precision model can be used to predict output error and optimize bitwidths
Experimental Results

- Experimental Procedure
- Example Data-paths (CDFGs)
- Comparison of Predicted Error Range and Actual Errors
Experimental Procedure
CDFG: PID Controller

\[ I_{refk} \times (K_p \times E_k) \times K_i \times (E_k \times \frac{1}{T}) \times K_d \times \frac{(E_k)}{T} \]
Result: PID Controller
Bitwidth Optimization

- Problem Formulation
- Optimization Methods
  - Grid Steepest Descent (GDS)
  - Accelerated Grid Steepest Descent (GSD-A)
- Optimization Results
Problem Formulation

- Precision model based on CDFG and profile data:
  \[ F \approx f(b_1, b_2, \ldots, b_N) \]

- Objective function:
  \[ \min_{i=1}^{N} b_i \]

- Constraints:
  - \( N \): number of nodes in CDFG
  - \( b_i \): bit-width of node \( i \)
  - \( P \): accuracy requirements
  - \( Z \): range for bit-width selection
  \[ Z \approx [1, 2, \ldots, 23] \]
Optimization Algorithms (1)

- **Regular Steepest Decent**
  - In each step, direction is calculated, step length is determined by searching in the direction

- **Grid Steepest Decent (GSD)**
  - In each step, step length is fixed (\(\alpha = 1\)), direction is determined by searching

\[
\min f(x) \\
\n\begin{align*}
\alpha & x_k \leftarrow x_k + \alpha d_k \\
\alpha & f(x_k) \\
\end{align*}
\]

x: vector of bitwidths
k: search step
**Grid Steepest Decent (GSD)**

1. Initialize bitwidths
2. Search neighbors
3. Adopt best direction
4. $F < P$?
5. $y$
6. Lower bound
7. Only one of the bitwidths increases by 1
8. $? = 1$
9. $d = [0 \ 0 \ldots \ 0 \ 1 \ 0 \ldots \ 0]$
Optimization Algorithms (3)

- Accelerated GSD (GSD-A)
  - “Smart” Initial Point
  - Binary search to locate initial point
  - Total search time is reduced to a fraction
  - Initial Point: All bitwidths have the same initial value
  - GSD: Each bitwidth is calculated individually

\[
\begin{align*}
L = 1, & \quad H = 23 \\
B = \frac{(L + H)}{2} \\
\text{if } F(B) < P & \\
H = B, & \quad L = B \\
\text{if } H - L > 1 & \\
\text{Initial} = L \\
& \quad \text{GDS}
\end{align*}
\]
## Optimization Results (1)

### Result Comparison (P = 5%)

<table>
<thead>
<tr>
<th>EXAMPLE</th>
<th>IEEE BITWIDTH</th>
<th>OPTIMIZED BITWIDTH</th>
<th>RGSD STEPS</th>
<th>RGSD-A STEPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIFFEQ</td>
<td>230</td>
<td>132</td>
<td>98</td>
<td>9</td>
</tr>
<tr>
<td>PID</td>
<td>253</td>
<td>136</td>
<td>87</td>
<td>8</td>
</tr>
<tr>
<td>Three MULT</td>
<td>96</td>
<td>37</td>
<td>59</td>
<td>2</td>
</tr>
</tbody>
</table>
Optimization Result (2)

- IEEE Format
- Total Bitwidth: 253
- Output Error: 0

Variable Bitwidth
- Total Bitwidth: 136
- Output Error: 5%

Optimize Bitwidths

Error=0  ➔  Error=5%
Future Work

- Validate the optimized bitwidths
  - *C++ floating-point library* supporting variable bitwidth
  - *VHDL Variable precision floating-point component library*, developed by Rapid Prototyping Lab at Northeastern University, available under GPL at [http://www.ece.neu.edu/groups/rpl/projects/floatingpoint/](http://www.ece.neu.edu/groups/rpl/projects/floatingpoint/)

- Improve error models
  - Propagation error models and rounding error models
  - Singularity issues

- Integrate in high level synthesis flow
  - IEEE 1076.3 working group: variable bitwidth floating-point for synthesis
Variable bitwidth FP computing is viable

Model-based bitwidth optimization has advantages over simulation-based searching

A methodology of FP precision modeling has been developed

The precision model predicts output error and can be used for bit-width optimization
A customized optimization algorithm, Grid Steepest Descent (GSD), has been developed.

Search acceleration techniques have been applied to GSD.

Optimized bitwidths for a given precision target can be found quickly.

Sum of the optimized bitwidths is significantly smaller than that of standard IEEE format.