Implementation-Specific Verification of Divide and Square Root Instructions

Ariel J. Birnbaum, Elena Guralnik,
Avi Kaplan, Anatoly Koyfman
Verification of the Floating Point Unit

Formal verification

Restricted by the unit’s complexity

FPU

Simulation tests

Practically unbounded number of different cases to test

Certain implementation areas are not sufficiently verified with the existing methods.
The Usual Approach in Simulation Based Verification

- generic test
- directed tests

FPU

inputs

outputs

IEEE spec
Test Generation Using FPgen

FPgen – a test generator, creating tests by solving constraints on inputs, outputs and intermediate result.

Intermediate result is the result before rounding. Defining coverage cases in terms of constraints on the intermediate result is our first step toward “white box” verification approach.
Motivation for Our Work

- Some bug prone cases could not be characterized in terms of the inputs, outputs and intermediate result.

- These bugs occurred in operations not implemented directly as a logic circuit. They are executed via “microcode”.

- The trigger for the bug occurrence can be described only in terms of specific implementation algorithm.

- The typical scenario was found in execution of instructions implemented via iterative numerical methods.
Typical Iterative Algorithm in Hardware

1. Initial (linear) approximation of the result.
2. A relative error is calculated.
3. A correction term (rational function of a relative error and a previous approximation) is added.

Input: another iteration
Output:
Real Life Bugs:

- **Square root:**
  - Rounding algorithm in the implementation depends on the sign of the final result.
  - The sign of the result was erroneously determined by the sign of the approximation after the first iteration.
  - In very rare cases this value was negative. It can occur only when the relative error is close to its maximal positive value.

- **Divide:**
  - A different problem of a similar nature was found.
Operations Implemented via Iterative Numerical Methods:

Reciprocal: $1/b$.
Iterative refinement of an initial approximation is used to achieve a final result.

Divide: $a/b$.
The result is calculated using the approximation of $1/b$, since $a/b = a \cdot 1/b$.

Reciprocal Square Root: $1/\sqrt{b}$.
The algorithm used is similar to that of Reciprocal.

Square Root: $\sqrt{b}$.
The result is calculated using the approximation of $1/\sqrt{b}$, since $\sqrt{b} = b \cdot 1/\sqrt{b}$. 
Formulating the Problem

- Corner cases can be described as intervals around extreme values of the relative error. We thus proceeded to study the following problem:

Given a binary floating point operation from the previous slide, the implementation algorithm, the iteration number, and an interval, find random inputs for the operation that, after the requested iteration, yield a relative error within the specified interval.
Relative Error

\( \tilde{q} \) is an approximation of a quantity \( q \).

Its *signed relative error* is defined by:

\[
e_q = \frac{q - \tilde{q}}{q}
\]

\( \tilde{q}^{(i)} \) is an approximation of a quantity \( q \) after the \( i \)-th iteration.

\( e_q^{(i)} \) is its signed relative error.

If we approximate the value \( q = r \cdot s \) using the approximation of \( s \), i.e. \( \tilde{q} = r \cdot \tilde{s} \). Then:

\[
e_q = \frac{q - \tilde{q}}{q} = \frac{r \cdot s - r \cdot \tilde{s}}{r \cdot s} = \frac{s - \tilde{s}}{s} = e_s
\]
Reduction of the problem to simple operations

As we mentioned before, given an approximation \( \tilde{q} \) of \( q \), we can approximate \( a \cdot q \) by \( a \cdot \tilde{q} \) with the same relative error. Therefore, \( \frac{e_a}{b} = e_1 \frac{1}{b} \) and \( e_{\sqrt{b}} = e_1 \frac{1}{\sqrt{b}} \), hence we can limit our discussion to the reciprocal and reciprocal square root operations.
Reduction to inputs in the interval \([1,2)\)

For \(b = (-1)^s \cdot 2^e \cdot (1 + f)\) \((0 \leq f < 1)\),

its reciprocal is \(\frac{1}{b} = (-1)^s \cdot 2^{-e} \cdot \frac{1}{1 + f}\).

Since \(\frac{e}{b} = e \frac{1}{1+f}\), we can assume that \(b \in [1,2)\).

For the same \(b\) its reciprocal square root is

\[
\frac{1}{\sqrt{b}} = 2^{\frac{e}{2}} \cdot \frac{1}{\sqrt{1 + f}} \quad \text{if } b \text{ has an even exponent, and}
\]

\[
\frac{1}{\sqrt{b}} = 2^{\frac{e+1}{2}} \cdot \frac{\sqrt{2}}{\sqrt{1 + f}} \quad \text{if } b \text{ has an odd exponent.}
\]

Using the same argument, we can assume that \(b \in [1,2)\),

if we treat both \(1/\sqrt{b}\) and \(\sqrt{2} / \sqrt{b}\).
Simplified Problem Definition

Given:

- an iterative method to compute $q(b) \in \{\frac{1}{b}, \frac{1}{\sqrt{b}}, \sqrt{2}\}$;
- an iteration number $i$;
- an interval $[L, H]$;

find:

random $b \in [1, 2)$ such that $e^{(i)}_{q(b)} \in [L, H]$. 
Solution outline

We present here a solution for a simplified problem: we assume that the domain is continuous and the initial approximation is piecewise linear.

The full solution can be found in the article.

1. First, we solve the following problem:
   a. iteration number is 0;
   b. the initial approximation is linear.

2. Then we adopt our algorithm to a piecewise linear initial approximation.

3. Afterwards, we extend the solution to all iterations.
Step 1: Linear Initial Approximation.

All the operations discussed \( \frac{1}{b}, \frac{1}{\sqrt{b}} \) and \( \sqrt{2} \) are of the form \( \beta \cdot b^{-\alpha} \).

The initial approximation is \( \tilde{q}(b) = \beta(A \cdot b + C) \).

\[
e_{q(b)}(b) = \frac{q(b) - \tilde{q}(b)}{q(b)} = \frac{\beta \cdot b^{-\alpha} - \beta(A \cdot b + C)}{\beta \cdot b^{-\alpha}} = 1 - b^\alpha \cdot (A \cdot b + C).
\]

It suffices to solve the following problem: given a smooth function \( e(b) \) over an interval \([X, Y]\), find \( b \in [1, 2] \), s.t. \( e(b) \in [L, H] \).

This can be solved in a finite number of iterations by the Interval Halving Method.
Step 1: Linear Initial Approximation (cont)

Interval Halving Method (aka Binary Search):

**inputs:** \( x, y \in [X, Y] \) s.t. \( e(x) \leq L \) and \( e(y) \geq H \).

**output:** \( r \in [X, Y] \) s.t. \( e(r) \in [L, H] \)

\[ r \leftarrow \frac{x + y}{2} \]

loop

if \( e(r) < L \) then \( x \leftarrow r \)

else if \( e(r) > H \) then \( y \leftarrow r \)

else return \( r \) (Solution Found)

\[ r \leftarrow \frac{x + y}{2} \]
Step 1: Linear Initial Approximation (cont)

- The algorithm’s correctness is ensured by the invariant that a solution exists between both endpoints of the search.

- The continuity of the relative error function and the Mean Value Theorem promises that invariant is kept.

- The solution so far doesn’t supply a random solution!

- To obtain the randomization of the solution, the initial guess can be chosen randomly within the interval \([x,y]\) and the invariant is still kept. We can also randomize the choice of \(r\) from \([x,y]\) at the end of every iteration.
Step 2: a Piecewise Linear Initial Approximation

- When the initial approximation is piecewise linear, we can extend our solution by dividing the input domain into the sectors, on which it is linear.

- In this case we precalculate the bounds and extrema of the relative error function for each sector.

- Actually, we created a table, containing the following information about each sector:
  - \( \min(e(r)) \) and \( x \) such that \( e(r) = \min(e(r)) \),
  - \( \max(e(r)) \) and \( x \) such that \( e(r) = \max(e(r)) \).
Step 3: Following Iterations

In most iterative methods used in practice the approximation is refined at each step by adding a correction term:
\[
\tilde{q}^{(i+1)} = \tilde{q}^{(i)} + C, \quad \text{where } C = \tilde{q}^{(i)} \cdot F(e_q^{(i)}).
\]

F is a rational function of the relative error at that step.

Then we have that the relative error after the iteration \((i+1)\) is:
\[
e_q^{(i+1)} = \frac{q - \tilde{q}^{(i+1)}}{q} = \frac{q - (\tilde{q}^{(i)} + \tilde{q}^{(i)} \cdot F(e_q^{(i)}))}{q} = 1 - \frac{\tilde{q}^{(i)}}{q} \cdot (1 + F(e_q^{(i)})) = 1 - (1 - e_q^{(i)}) \cdot (1 + F(e_q^{(i)}))
\]
Step 3: Following Iterations (example)

For example, for Double Precision Divide \( q = a/b \) :

\[
C = \left( a - b \cdot \widetilde{q}^{(0)} \right) \cdot \frac{\widetilde{q}^{(0)}}{a} \cdot P(e^{(i)}_q), \text{ where } P(x) = \sum_{i=1}^{5} x^i.
\]

Thus:

\[
C = \frac{a - b \cdot \widetilde{q}^{(0)}}{a} \cdot \widetilde{q}^{(0)} \cdot P(e^{(i)}_q) = \widetilde{q}^{(0)} \cdot \frac{a/b - \widetilde{q}^{(0)}}{a/b} \cdot P(e^{(i)}_q),
\]

then by defining \( F(x) \equiv x \cdot P(x), \)

we have \( C = \widetilde{q}^{(0)} \cdot F(e^{(0)}_q). \)
Step 3: Following Iterations (cont)

- The relative error after the iteration \((i+1)\) is a rational function of the error in the \(i\)-th iteration.

- Hence, the relative error after the iteration \((i+1)\) is also a smooth function of the input \(b\).

- The extremal values of it can be precomputed using numerical techniques.

- Knowing the extreme values in every sector, we can apply the same Halving Method to obtain a solution.
New Solving Abilities => New Coverage Tasks

A new kind of constraint can be solved now.

It gives us the ability to provide a new set of coverage tasks.

Each task defines an interval for the relative error in a given iteration, for a given operation, on a certain precision.

- **operation:** $1/b, a/b, \sqrt{b}$ or $1/\sqrt{b}$
- **relative error:** sign & range $[L,H]$
- **iteration number**
Summary and Future Work

- For complex operations such as divide and square root it’s necessary to “peek inside the black box”.
- There is a need for implementation oriented testing of instructions implemented using approximation algorithms.
- A new kind of coverage models treat the problems that motivated our research.
- Currently we can’t solve constraints placed on both inputs and relative error – there is a need to extend the solvers ability.