FloPoCo
an arithmetic code generator for FPGAs
http://www.ens-lyon.fr/LTP/Arenaire/Ware/FloPoCo/
Florent de Dinechin, Bogdan Pasca, projet Arénaire

IEEE-754-compatible
•, . , +, −, √
○ Massive parallelism
○ Each operator 10X slower than a processor’s
   no match to GPPU, ClearSpeed, ...
Use FPGA flexibility!

FPGAs as floating-point accelerators?

Flexibility in precision
• There is life between single and double precision
• Never compute more accurately than needed
  mix-and-match fixed-point and floating-point [2]
  • FP interfaces, internal fixed-point computations [4]
  • Automated error analysis [6, 7, 8, 4]

Flexibility in algorithms
• Operators may be specialized to a context
  • Multipliers by a constant [1]
  • Squares [3]
• Specific hardware may be designed for coarser operators
  • elementary functions (10- the throughout of a Pentium) [6, 7, 8]
  • application-specific accelerator [4] – see below

Not a library, a generator
• An infinite virtual library
• Greater parameterization and flexibility
  • Optimizes for different hardware targets (timings, LUT and DSP)
• Better design-space exploration
  • Example: evaluation of arbitrary functions using HOTBM [5]
  • Generate simple and tidy VHDL code
• Written in C++ using GMP, MPFR and Sollya
  • Freely available under the LGPL

Naive approach
• Each operator 10X slower than a processor’s
• Designed as accurate as the application requires
• Designed to never overflow
• Never compute more accurately than needed
• Designed to never overflow
• Never compute more accurately than needed

FioPoCo - the framework

Class hierachy overview

Automatic pipeline management
• Designers think in terms of pipeline levels
• Attributes for: cycle of a signal, currentCycle, lifeSpan
• Automatic register insertions – Ask for a demo!

Goal: Not your neighbour’s FPU
• Basic operations, with a bit of pepper
• Elementary functions (sine, exponential, logarithm…)
• Algebraic functions (1/x, √x, polynomials, …)
• Compound functions (log(x+1)) , e−x, …)
• Sums, dot products, sums of squares, norms…
• Interval arithmetic

Example 1 - Accumulation of FP numbers

<table>
<thead>
<tr>
<th>Accumulator significand never needs to be shifted</th>
</tr>
</thead>
<tbody>
<tr>
<td>128 single, 6 cycles @ 242 MHz</td>
</tr>
<tr>
<td>128 single, 8 cycles @ 434 MHz</td>
</tr>
<tr>
<td>295 single, 10 cycles @ 428 MHz</td>
</tr>
<tr>
<td>375 single, 11 cycles @ 414 MHz</td>
</tr>
</tbody>
</table>

Example 2 - Collision detection predicate

<table>
<thead>
<tr>
<th>Need to compute: x² + y² + z² &lt; r²</th>
</tr>
</thead>
</table>
| Naive approach
  • 3 FP multipliers + 2 FP adders + comparison |
| Specific operator
  • Squares instead of multipliers
  • Addition of positive numbers is simpler
  • Mantissa alignment in parallel
  • Less rounding logic |

References

This work was partly supported by the Egide Brancusi programme of the French government, and the XitremData university programme.