Report 02.09.20

02/09/2020

Matteo Perotti
Luca Bertaccini
Pasquale Davide Schiavone
Stefan Mach

Professor Luca Benini
Integrated Systems Laboratory
ETH Zürich

Roadmap

- **RISC-V FP library** (32-/64- bit)
  - Add/Sub
  - Mul
  - FMA
  - Comparison functions

- FP library **comparison** for code **size**/**performance**
  - Translated
  - libgcc
  - SoftFloat
  - SEGGER
  - ARM libgcc (size)

- **FP library optimization**

Where we are

- **RISC-V FP translated library**:
  - f32_add
  - f32_sub
  - f32_mul
  - f32_fma
  - f64_add
  - f64_sub
  - f64_mul ← Debugging now
  - f64_fma
  - comparison functions

Code size

Code size (B)	libgcc	Translated	SEGGER
f32_add	786*	414*	410
f32_sub	816	8*	10
f32_mul	538	388	178
f64_add	1516*	722*	724
f64_sub	1534	10*	10
clz support	316	316	-
Total	5506	1858 (-66%)	1332 (-76%)

*depend on **clz support** from libgcc. We are optimizing it for code size.

All the information relative to SEGGER are taken from their official website
https://www.segger.com/products/development-tools/runtime-library/technology/floating-point-library/

Code size

- The FMA issue:
  - libgcc does not provide low level support for FMA
  - There are high-level FMA functions provided by libm
  - f32_fma provided calling f64_mul and f64_add

Problem if the program needs only single-precision FP!

Code size (B)	libgcc	Translated
f32_fma	> 3000	~780

Performance

- Ideally, two measurements
  - Function itself, avg. cycle count (SEGGER settings)
    - Random inputs in (0, 1)
    - Zero-delay memory
  - Benchmark program score

- Preliminary measurements on CV32E40P for f32_mul
  - libgcc: > 100 cycles
  - Translated: 51 cycles
  - SEGGER (on another processor, so not fair comparison): 39.3 cycles

Further improvements and notes

1. **SEGGER flushes denormals** to signed zero. Try one version with denormals to zero.
2. **SEGGER** measures **performance** and **size** using **two different versions** of its **library**
3. **Remove special cases** (e.g. power of 2 multiplications)
4. **Play** with the **clz implementation** (inline, don’t use the table)
5. **Optimize** r**egister allocation**, condition checking and general **coding style**