# Report 15.10.20

15/10/2020

**Matteo Perotti**
**Luca Bertaccini**
**Pasquale Davide Schiavone**
**Stefan Mach**

**Professor Luca Benini**
**Integrated Systems Laboratory**
**ETH Zürich**

# Summary

- Completed **testing** environment for **NoDenormal** functions
- Further **optimized code** after performance evaluation
- **Completing** the **library** (division, cast functions)
- Environment to **benchmark Tiny FPU** and evaluate its **energy efficiency**

# Testing environment for NoDenormal

- Completed testing environment:
  - RV32**I**M **+ Denormals** → **Verified with TestFloat (level 2, max number of inputs)**
  - RV32**I**M No Denormals
  - RV32**E**M No Denormals

  } → **Verified with TestFloat (level 2, max number of inputs)**

# Updated Code size

| Code Size (B) | Mine | libgcc | Mine ND (rv32em) | SEGGER ND * |
|---|---|---|---|---|
| **f32_add** | 414 | 786 | 334 | 410 |
| **f32_sub** | 8 | 816 | 8 | 10 |
| **f32_mul** | 374 | 538 | 172 | 178 |
| **f32_eq** | 38 | 86 | 38 | 44 |
| **f32_le** | 60 | 120 | 60 | 54 |
| **f32_lt** | 52 | 120 | 52 | 58 |
| **f64_add** | 750 | 1520 | 572 | 724 |
| **f64_sub** | 10 | 1538 | 10 | 10 |
| **f64_mul** | 560 | 1096 | 288 | 286 |
| **f64_eq** | 64 | 106 | 46 | 52 |
| **f64_le** | 104 | 166 | 84 | 70 |
| **f64_lt** | 92 | 166 | 76 | 70 |
| **Total** | 2526 (-64%) | 7058 | 1740 (-11.5%) | 1966 |

* Data from: https://www.segger.com/products/development-tools/runtime-library/technology/floating-point-library/

# Updated Code size

| Code Size (B) | Mine | libgcc | Mine ND (rv32em) | SEGGER ND * |
|---|---|---|---|---|
| **f32_add** | 414 | 786 | 334 | 410 |
| **f32_sub** | 8 | 816 | 8 | 10 |
| **f32_mul** | 374 | 538 | 172 | 178 |
| **f32_eq** | 38 | 86 | 38 | 44 |
| **f32_le** | 60 | 120 | 60 | 54 |
| **f32_lt** | 52 | 120 | 52 | 58 |
| **f64_add** | 750 | 1520 | 572 | 724 |
| **f64_sub** | 10 | 1538 | 10 | 10 |
| **f64_mul** | 560 | 1096 | 288 | 286 |
| **f64_eq** | 64 | 106 | 46 | 52 |
| **f64_le** | 104 | 166 | 84 | 70 |
| **f64_lt** | 92 | 166 | 76 | 70 |
| **Total** | 2526 (-64%) | 7058 | 1740 (-11.5%) | 1966 |

* Data from: https://www.segger.com/products/development-tools/runtime-library/technology/floating-point-library/

# Updated performance

| | **Mine** (cycles) | **libgcc** (cycles) | **Improvement** |
|---|---|---|---|
| **f32_add** | 75.20 | 95.50 | 26.99% |
| **f32_sub** | 79.00 | 102.80 | 30.13% |
| **f32_mul** | 51.20 | 117.20 | 128.91% |
| **f32_eq** | 15.00 | 27.30 | 82.00% |
| **f32_le** | 17.70 | 36.80 | 107.91% |
| **f32_lt** | 16.70 | 36.80 | 120.36% |
| **f64_add** | 102.40 | 115.00 | 12.30% |
| **f64_sub** | 107.10 | 123.50 | 15.31% |
| **f64_mul** | 91.40 | 219.50 | 140.15% |
| **f64_eq** | 24.00 | 31.30 | 30.42% |
| **f64_le** | 25.00 | 40.10 | 60.40% |
| **f64_lt** | 25.00 | 40.10 | 60.40% |

# Notes

- Mine vs. SEGGER:
  - Code size improvement (-11.5%)
  - Lower code size for the addition
  - Lower performance for the addition

- Mine vs. libgcc:
  - Code size improvement (-64%)
  - The real improvement is higher, as libgcc depends on count leading zero function and table!
  - Multiplication/comparison performance boost (also +140% speed)

# Completing the library

## Functions to add:

- Division (32b-64b)
- Conversion (32b-64b)
  - Float <--> int
  - Float <--> unsigned integer
  - Float <--> signed long
  - Float <--> unsigned long

# Open-Source Release

**Future** plans:

- **Release** the **library** under **GPL** license
- **Complete** and **test** the missing functions
- **Add** them to the library

# Benchmark environment

**Comparison** between FastFPU, TinyFPU, uCode, SW-libgcc:

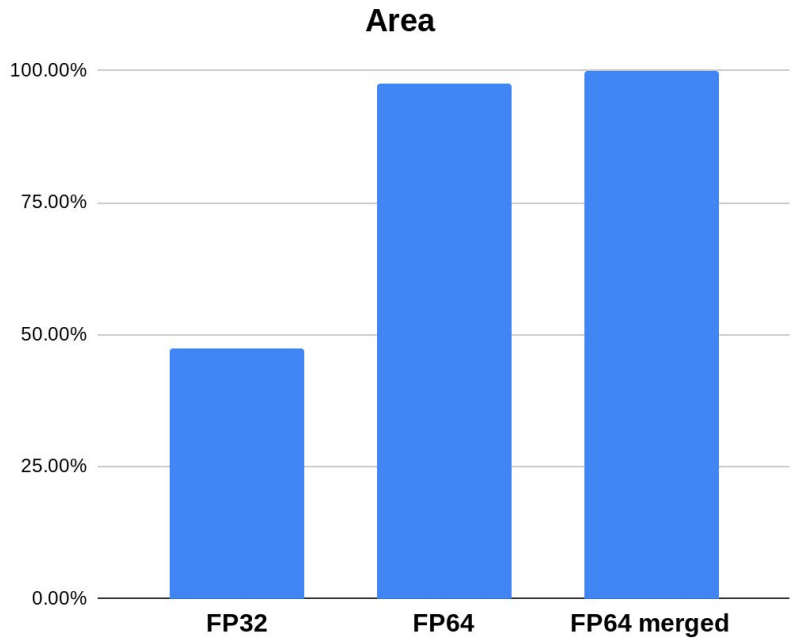
- Estimate **Power**, **FP-Throughput**, **Energy Efficiency** (y-axis)
- Vary **% FP operations** in the program (x-axis)
- 1 **synthetic** and some **real** programs from CoreMarkPRO (https://github.com/eembc/coremark-pro/tree/master/benchmarks/fp)
- Find **intervals** of **% FP** operations in which **one implementation** is **more energy efficient** than the others

# Tiny Floating-Point Unit

- Merged implementation
- FPU power analysis
- Architectures

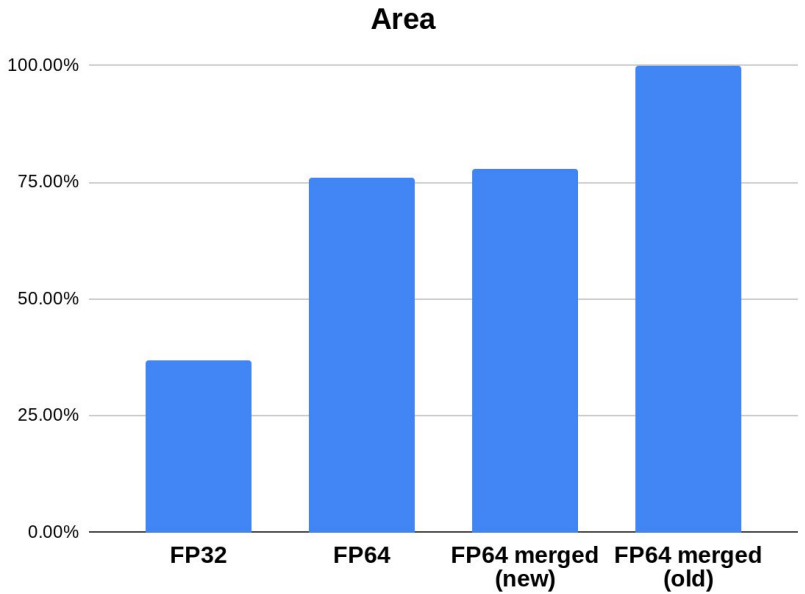
# TinyFPU - Merged Implementation (Area Overhead)

- Target: **RVFD**
- Merged implementation instead of FP32 TinyFPU and FP64 TinyFPU

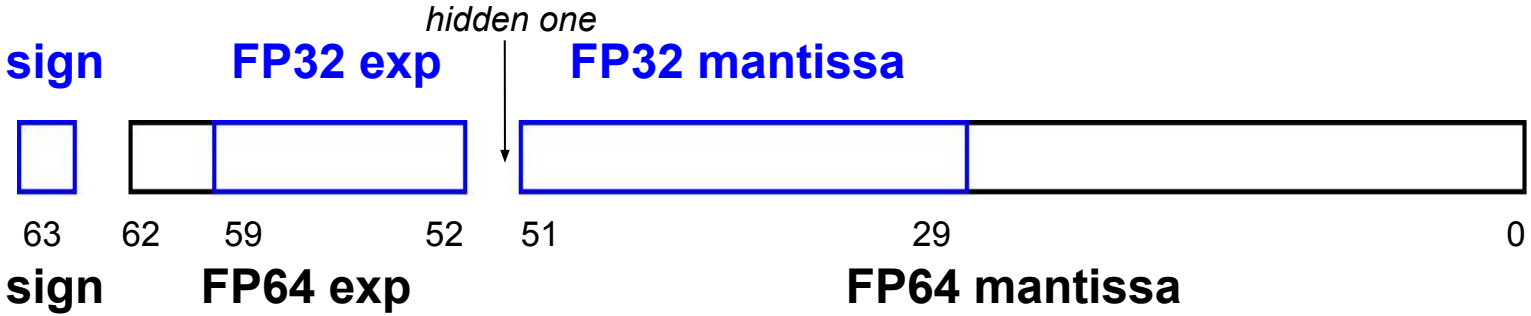


# TinyFPU - Merged Implementation (Area Overhead)

- Target: **RVFD**
- Merged implementation instead of FP32 TinyFPU and FP64 TinyFPU
- **3% area overhead**
- **+1 cycle** (FMADD, FMUL)



# TinyFPU - Merged Implementation




- **FP32 exp** mapped into **FP64 exp LSBs**
- **FP32 mantissa** mapped into **FP64 mantissa MSBs**
- FP32 mantissa + hidden one = 24 bits
- FP64 mantissa + hidden one = 53 bits

# TinyFPU vs FastFPU - Power analysis

- Topographical synthesis
- Stand-alone FPUs: power analysis (relaxed timing - **Tclk=15000ps**):
  - **LKG** power **scales** similarly to **area**
  - **Overall power scales** similarly to **area** (FP32 vs FP64 version of the **same FPU**)
  - FP64 TinyFPU is **2.2x** smaller than FP64 FastFPU but consumes **5.5x** less power
  - Lower **switching activity** for TinyFPU

# Snitch

- **IPU, DMA, SSRs** removed
- **Memories reduced**:
  - FP64 comparison: 4-bank **TCDM** (64x256), 2-bank **CACHE** (data: 64x128, tag: 41x128)
  - FP32 comparison: 4-bank **TCDM** (32x512), 2-bank **CACHE** (data: 64x128, tag: 27x128)

# Comparison

**Single-precision** support (*Tclk=2000ps*):

- 32-bit Snitch + FP32 Fast-FPU*
- 32-bit Snitch + FP32 Tiny-FPU
- 32-bit Snitch

**Double-precision** support (*Tclk=2400ps*):

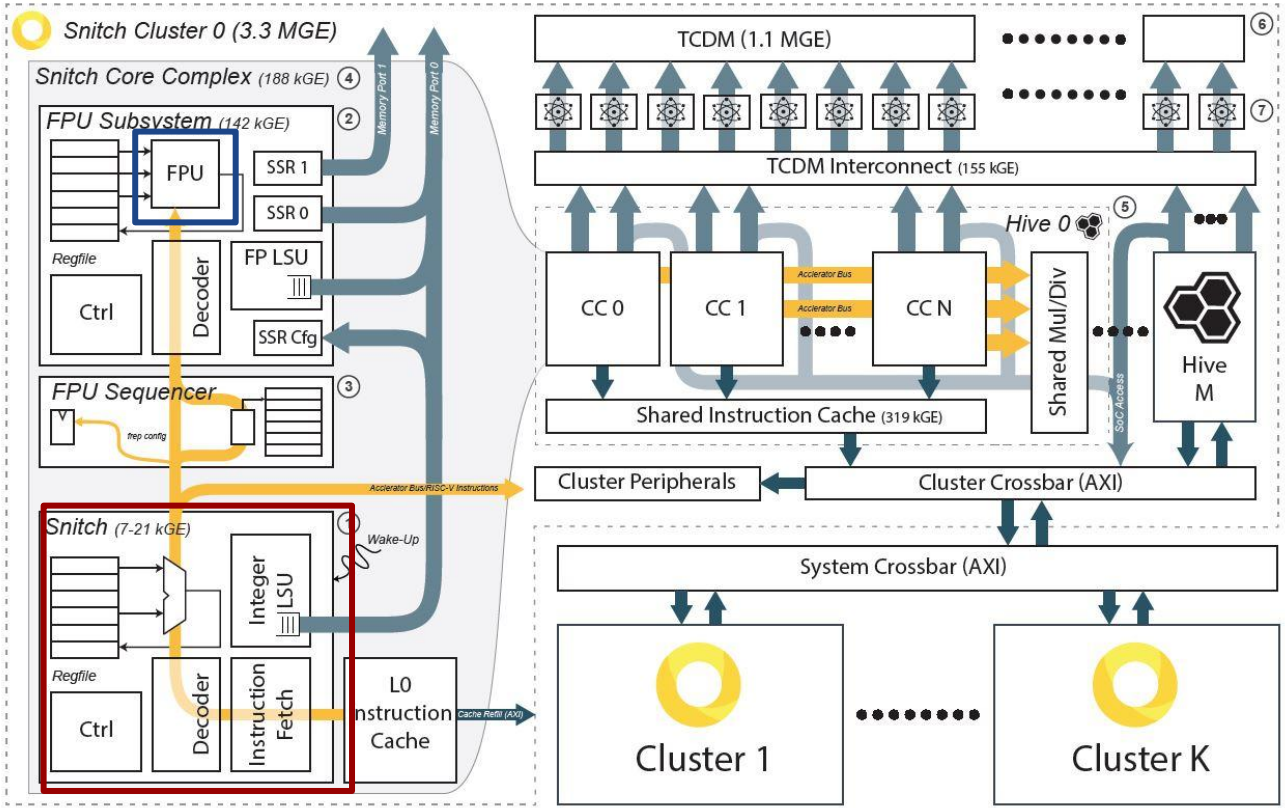
- 32-bit Snitch + FP64 Fast-FPU* (merged)
- 32-bit Snitch + FP64 Tiny-FPU (merged)
- 32-bit Snitch

**Fast-FPU: combinational*

# SNITCH

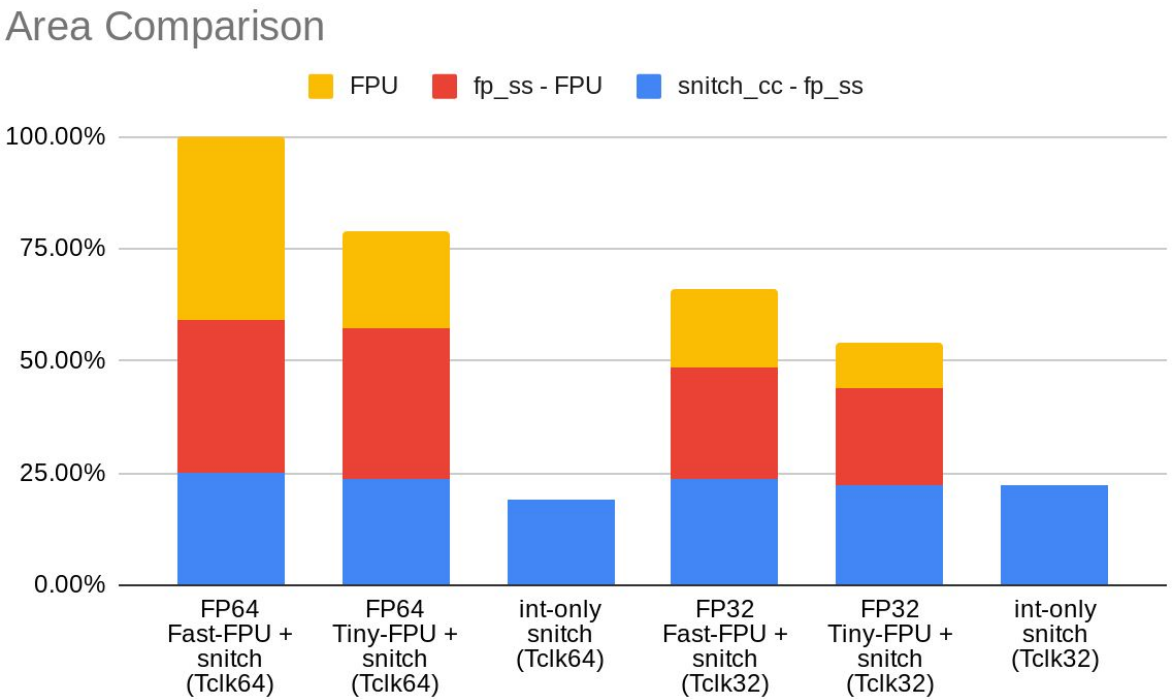
**FPU**

**Snitch**




*Zaruba, F., Schuiki, F., Hoefler, T., & Benini, L. (2020). Snitch: A 10 kGE Pseudo Dual-Issue Processor for Area and Energy Efficient Execution of Floating-Point Intensive Workloads. arXiv preprint cs.AR/2002.10143.*

# Comparison (Area)



# Next Steps

- Power analysis
- Complete the paper
- Micro-ops