# Report 12.08.20

12/08/2020

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

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

# Tiny Floating-Point Unit

- Architecture: SNITCH
- Tiny FPU: Top-level
- Microcode
- Target Conference: DATE21

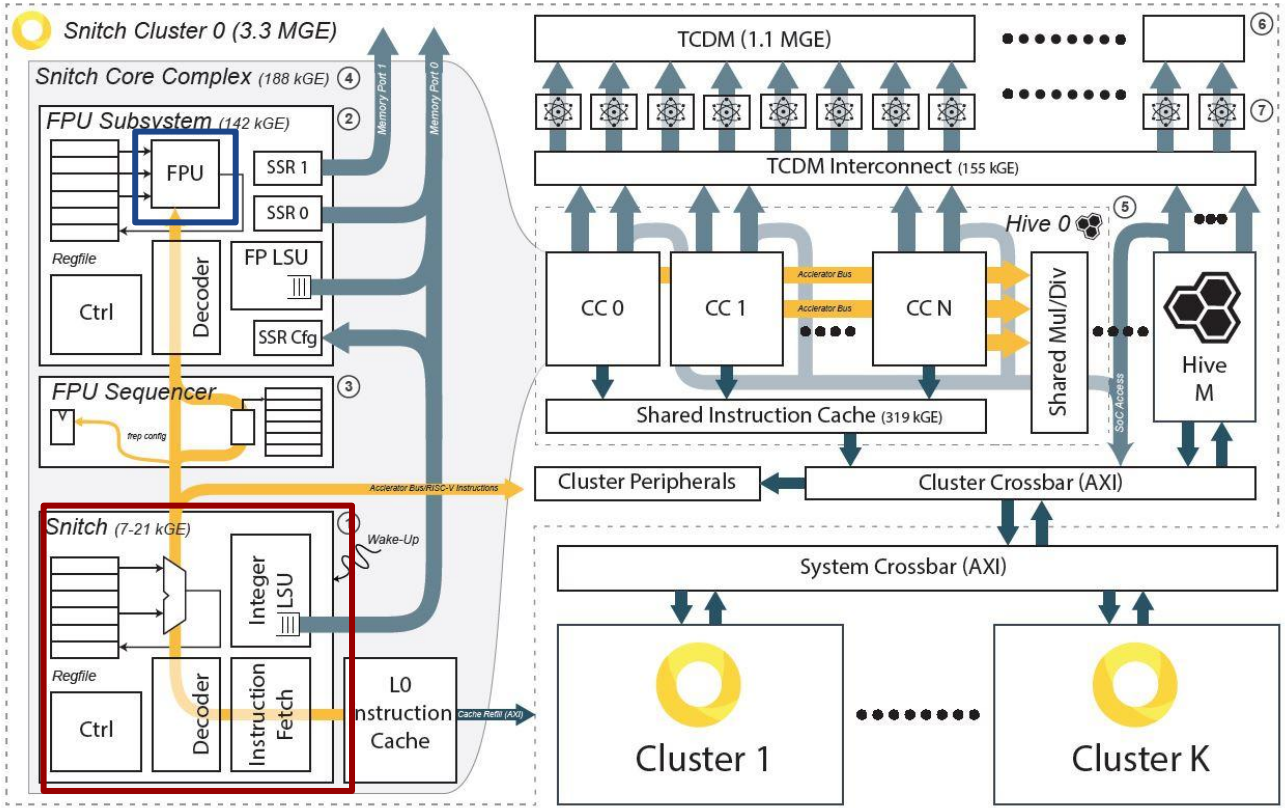
# SNITCH

- **Snitch** RISC-V Processing Element with Streaming Registers (**RV32[I|E]MAFD**ZifenceiZicsrXssrXfrep)
  - Support for FP32 and FP64
  - **32-bit INT register file**
  - **64-bit FP register file**
- Snitch paper: https://arxiv.org/abs/2002.10143
- *"A general-purpose, **single-stage**, **single-issue core** tuned for utmost **energy efficiency**. Aiming to maximize the compute/control ratio (making the **FPU the dominant part of the design**) mitigating the effects of deep pipelines and dynamic scheduling"*

# SNITCH

**Tiny FPU**

**Snitch**



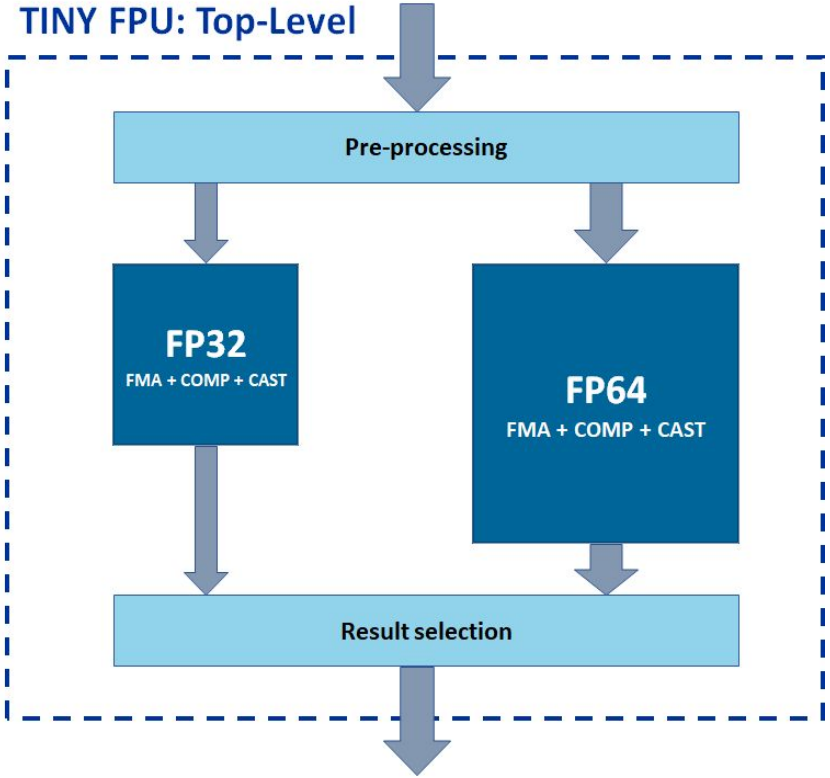

*Zaruba, F., Schuiki, F., Hoefler, T., & Benini, L. (2020). Snitch: A 10 kGE Pseudo Dual-Issue Processor for Area and Energy Efficient Execution of Floating-Point Intensive Workloads. arXiv preprint cs.AR/2002.10143.*

# Tiny FPU: Top-Level

- Same **interface** as **FPNEW**
- The **top-level** contains:
  - 1 instantiation of **FP32** FMA+COMP+CAST
  - 1 instantiation of **FP64** FMA+COMP+CAST
  - **pre/post-processing** depending on the instruction and on the precision
- Tiny FPU **integrated into Snitch**

# Tiny FPU: Top-Level

- **FP64 module** instantiated just if **double-precision** is **supported**
- **Pre-processing**: depending on the instruction some operands may be set to zero
- **Result selected** depending on input and output **format**



# Microcode

- Compile with **F/D extensions** enabled (e.g. fadd.s f1,f2,f3)
- Employ a memory-mapped **ROM** to store the SW emulation instructions, which uses RV32 instructions
- FP instruction will be interpreted by the decoder as a ***jal*** to the ROM address

# Microcode

**Issue**:

- RISC-V ISA does not contain MV from FP64 register file to INT32 register file

**Proposed solution**:

- Add two custom instructions to perform the move from FP64 to INT32 and vice versa

- Push & Pop of the integer registers used by the microcode

or

- Add some 64-bit **shadow registers** to save function call overhead

# Target Conference: DATE21

**DATE21**:

- **Abstract** submission: **September, 14**
- **Paper** submission: **September, 21**

# DATE21: Paper content

- Comparison among:
  - **FPU**
  - **Tiny FPU**
  - **SW emulation**
  - **Microcode**

- Performance:
  - Latency
  - Area
  - Code size
  - Power consumption

# Next steps

- Microcode implementation
- Optimized SW libraries
- Benchmarks (fft, matmul, knn, ...)

# Appendix: FMA + COMP + CAST HW - Single Precision

| FP32 | fpnew_fma + fpnew_noncomp + fpnew_multi_cast (NON-REDUCED) | reduced_fma_comp_cast (last presentation) | reduced_fma_comp_cast tested |
|---|---|---|---|
| **Overall Area** | 100% | ~57.1% | ~58.3% |
| **Latency** | 1 cycle (ADD/MUL/FMADD/COMP) | 2 cycles COMP<br>9 cycles CAST<br>10-12 cycles (ADD)<br>22-24 cycles (FMADD/MUL) | 2 cycles COMP<br>9 cycles CAST<br>10-12 cycles (ADD)<br>22-24 cycles (FMADD/MUL) |
| **Optimization** | - | **~42.9%** | **~41.7%** |

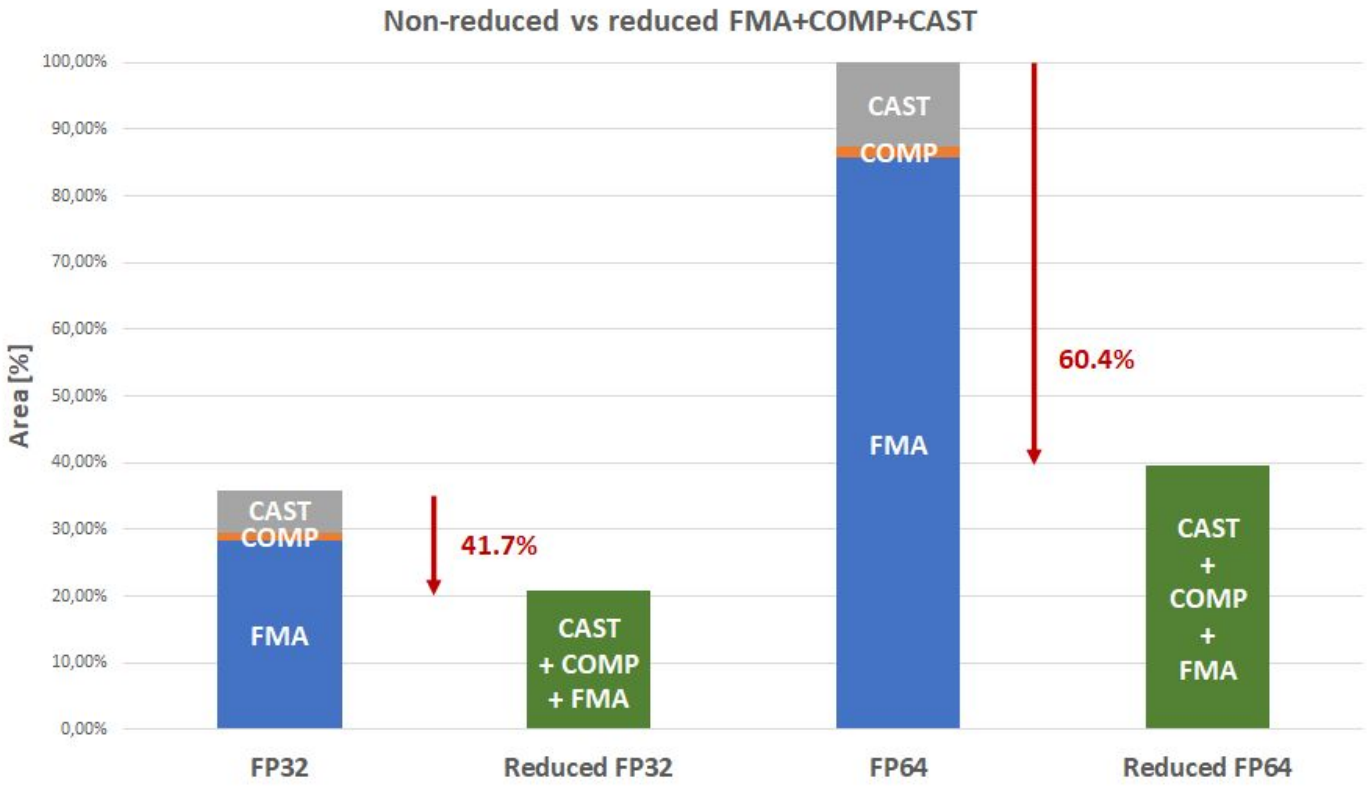
- No changes in the latencies after testing

# Appendix: FMA + COMP + CAST HW - Double Precision

| FP64 | fpnew_fma + fpnew_noncomp + fpnew_multi_cast (NON-REDUCED) | reduced_fma_comp_cast (last presentation) | reduced_fma_comp_cast tested |
|---|---|---|---|
| **Overall Area** | 100% | ~39.3% | ~39.6% |
| **Latency** | 1 cycle (ADD/MUL/FMADD/COMP) | 2 cycles COMP<br>9 cycles CAST<br>10-12 cycles (ADD)<br>36-38 cycles (FMADD/MUL) | 2 cycles COMP<br>9 cycles CAST<br>10-12 cycles (ADD)<br>36-38 cycles (FMADD/MUL) |
| **Optimization** | - | **~60.7%** | **~60.4%** |

- No changes in the latencies after testing

# Appendix: FMA + COMP + CAST: HW optimization (tested)



# Appendix: FMA - Single Precision

| FP32 | Latency reduced FPU | Average Latency SW emulation (SEGGER) | Code size (SEGGER) | Speed-up |
|---|---|---|---|---|
| **ADD** | 10-12 cycles | 49.5 cycles | 410 B | ~4.5x |
| **SUB** | 10-12 cycles | 62.2 cycles | 10 B | ~5.5x |
| **MUL** | 22-24 cycles | 39.3 cycles | 178 B | ~1.7x |
| **FMADD** | 22-24 cycles | 49.5 + 39.3 = 88.8 cycles | | **~3.9x** |

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

# Appendix: COMP - Single Precision

| FP32 | Latency reduced FPU | Average Latency SW emulation (SEGGER) | Code size (SEGGER) | Speed-up |
|---|---|---|---|---|
| < | 2 cycles | 11 cycles | 58 B | 5.5x |
| <= | 2 cycles | 10 cycles | 54 B | 5x |
| > | 2 cycles | 10 cycles | 50 B | 5x |
| >= | 2 cycles | 11 cycles | 62 B | 5.5x |
| == | 2 cycles | 10 cycles | 44 B | 5x |
| != | 2 cycles | 10 cycles | -- | 5x |

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

# Appendix: CAST - Single Precision

| FP32 | Latency reduced FPU | Average Latency SW emulation (SEGGER) | Code size (SEGGER) | Speed-up |
|---|---|---|---|---|
| **INT32 -> FP32** | 9 cycles | 32.6 cycles | 66 B | 3.6x |
| **FP32 -> INT32** | 9 cycles | 14 cycles | 74 B | 1.5x |

Overall code size - SW emulation (FMA+COMP+CAST) = **1006 B**

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

# Appendix: FMA - Double Precision

| FP64 | Latency reduced FPU | Average Latency SW emulation (SEGGER) | Code size (SEGGER) | Speed-up |
|---|---|---|---|---|
| **ADD** | 10-12 cycles | 62.8 cycles | 724 B | ~5.7x |
| **SUB** | 10-12 cycles | 82.8 cycles | 10 B | ~7.5x |
| **MUL** | 36-38 cycles | 75.0 cycles | 286 B | ~2x |
| **FMADD** | 36-38 cycles | 62.8 +75 = 137.8 cycles | - | **~3.7x** |

Overall code size - SW emulation (FMA+COMP+CAST) = **2332 B**

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

# Appendix: COMP - Double Precision

| FP64 | Latency reduced FPU | Average Latency SW emulation (SEGGER) | Code size (SEGGER) | Speed-up |
|---|---|---|---|---|
| < | 2 cycles | 16 cycles | 70 B | ~8x |
| <= | 2 cycles | 16 cycles | 70 B | ~8x |
| > | 2 cycles | 16.1 cycles | 70 B | ~8x |
| >= | 2 cycles | 16.1 cycles | 70 B | ~8x |
| == | 2 cycles | 14 cycles | 52 B | ~7x |
| != | 2 cycles | 14 cycles | -- | ~7x |

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

# Appendix: CAST - Double Precision

| FP64 | Latency reduced FPU | Average Latency SW emulation (SEGGER) | Code size (SEGGER) | Speed-up |
|---|---|---|---|---|
| **INT32 -> FP32** | 9 cycles | 32.6 cycles | 66 B | ~3.6x |
| **FP32 -> INT32** | 9 cycles | 14 cycles | 74 B | **~1.5x** |
| **INT64 -> FP32** | 9 cycles | 49.1 cycles | 96 B | **~5.5x** |
| **FP32 -> INT64** | 9 cycles | 23.2 cycles | 146 B | ~2.6x |
| **FP32 -> FP64** | 9 cycles | 14.1 cycles | 64 B | ~1.5x |
| **FP64 -> INT64** | 9 cycles | 26.9 cycles | 146 B | ~3.0x |
| **FP64 -> INT32** | 9 cycles | 16.8 cycles | 84 B | ~3.6x |
| **INT32 -> FP64** | 9 cycles | 31.6 cycles | 46 B | ~1.9x |
| **INT64 -> FP64** | 9 cycles | 45.1 cycles | 128 B | ~5x |
| **FP64 -> FP32** | 9 cycles | 25.1 cycles | 130 B | ~2.8x |

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