Hybrid Floating-Point Modules with Low Area Overhead on a Fine-Grained Processing Core

UC DAVIS COLLEGE OF ENGINEERING

Jon Pimentel and Bevan Baas

University of California, Davis: VLSI Computation Laboratory

VCL

1. Research Motivation

- **Floating-Point (FP) is the most commonly used method for real number representation** [1]
- **Certain architectures are limited to fixed-point arithmetic due to the large area and power requirements for floating-point hardware** [2,3]

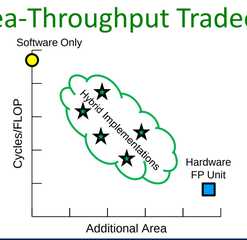
- **The goal**:
  - Increase throughput of floating-point arithmetic without the area overhead of full hardware floating-point units (FPUs)

Full Software
Low Area
Low Throughput

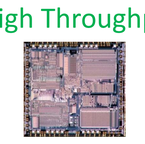
MOVE R2 R4
ADD R1 R2 R3
SHR R1 R1 R6
SUB R7 R7 R6

Fixed-point Software Instructions

Hybrid
Area-Throughput Tradeoff



Full Hardware
High Area
High Throughput



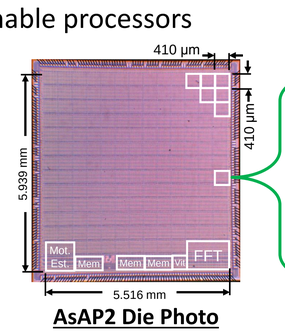
Floating-Point Processor [4]

[1] J.-M. Muller et al., Handbook of Floating-Point Arithmetic, 2009.
[3] D. Truong, et al., JSSC, 2009.
[2] S. Gilani et al., ASAP, 2011.
[4] http://www.cpu-world.com/CPUs/80387/

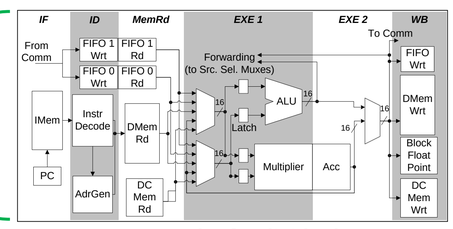
2. Targeted Many-Core Architecture

- AsAP2 [1]: **Fine-grained many-core system**
- **Example of a platform whose datapath is limited to fixed-point arithmetic**
- General purpose and capable of computing complex DSP workloads: e.g 802.11a, SAR, H.264
- **No specialized instructions**
- 164 programmable processors

AsAP 2 Single Processor	
Area	0.17 mm^2
Technology	65 nm low-leakage CMOS
Max Freq.	1.2 GHz @ 1.3 V
Instruction Memory	128 x 35-bit
Data Memory	128 x 16-bit



AsAP2 Die Photo



Datapath of Individual Processing Core

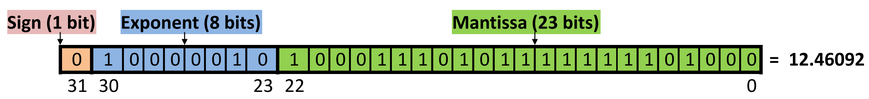
[1] D. Truong, et al., JSSC, 2009.

3. Floating-Point Format Modifications

- IEEE-754 is the technical standard for floating-point [1]
  - Defines data format, rounding modes, operations, exception handling

- One way to decrease the overhead of floating-point hardware is to **modify the floating-point format**

- **Subset of the IEEE-754 standard's requirements are implemented**
  - Addition/subtraction, and multiplication are implemented
  - Exception handling, NaNs, ±Inf, and denormal values are not supported
  - Only round to nearest even supported
- Division and square root can be performed using addition/subtraction and multiplication
- Many multimedia applications do not rely on extra modes/special number support [2,3,4]



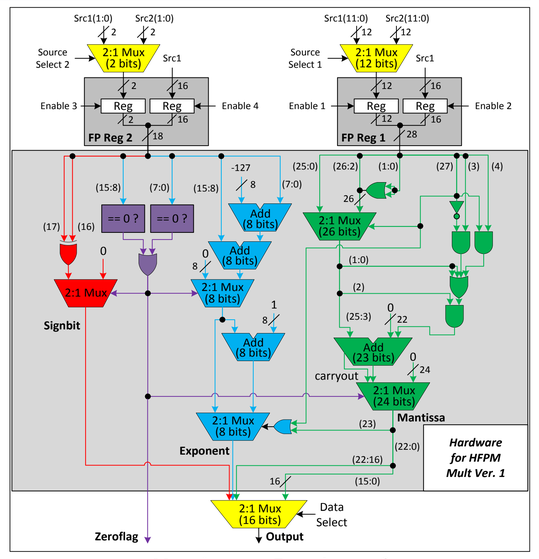
IEEE-754 32-bit Floating-Point Bit Layout

[1] IEEE Standard for Floating-Point Arithmetic, 2008, IEEE Std 754-2008.
[3] F. Fang et al., EURASIP JASP, 2002.
[2] H.-J. Oh et al., JSSC, 2006.
[4] S.-W. Lee et al., ISCAS, 2002.

4. Hybrid Floating-Point Modules (HFPMs)

- Another method to increase throughput without the overhead of floating-point hardware is to **utilize hybrid floating-point modules**

- HFPMs are one of the methods used for performing floating-point arithmetic
- HFPMs are an **alternative to Full Hardware or Full Software Floating-Point Modules**
- **Hybrid of Software/Hardware**
  - Fixed-point software
  - Custom FP instructions
- ***Fixed-point Software***
  - Keeps area low
  - Existing ALU is used to perform simple steps
- ***Custom FP Hardware***
  - Increases throughput
  - Added to perform part of a floating-point operation



Hardware Portion of Hybrid Multiplication Module

5. Full Software FP Modules

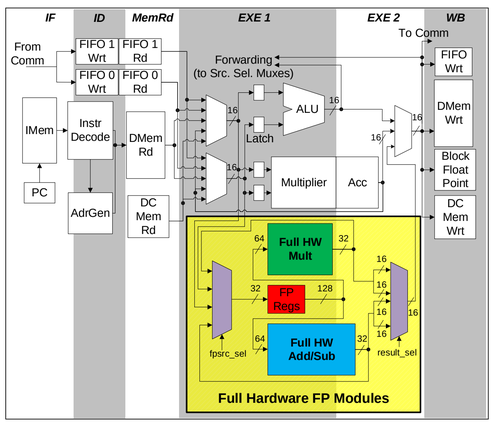
- Full software modules are one of the methods used for performing floating-point arithmetic
- **FP hardware emulated using 16-bit fixed-point ALU operations**
- No custom FP instructions
- **Software library created in assembly**
  - Addition/Subtraction Modules in Software
    - IMEM usage: 2 cores
  - Multiplication Module in Software
    - IMEM usage: 1 core
- **Large program sizes**
- **No additional area to implement**
- **Low throughput**

```
// Compute new sign bit
XOR DMEM[0] DMEM [18] DMEM [14]
// Compute the new exponent
SUB NULL DMEM [16] DMEM [5]
ADD DMEM [1] DMEM [1] regbp1
// Perform multiplication
MACC NULL DMEM [3] DMEM [18] nop1
// Grab bits that we shift off
AND NULL ACC DMEM [6]
```

Section of Full SW Mult Program

6. Full Hardware FP Modules

- Full hardware modules are one of the methods used for performing floating-point arithmetic
- **Arithmetic performed using floating-point hardware only**
- 16-bit operands loaded into 32-bit FP registers
- **Separate addition/subtraction and multiplication modules created in hardware (allows modularity)**
  - Fused and cascade multiply-add FPUs have large overhead for this platform
- **High area to implement**
- **High throughput**



Full Hardware FP Modules Integrated into Datapath of a Single Processor

7. Comparison of Floating-Point Modules

- Throughput and area for each type of floating-point module are plotted on the right
- The figure compares the **additional area** required for each floating-point module **versus** the **cycles per floating-point operation (FLOP)**

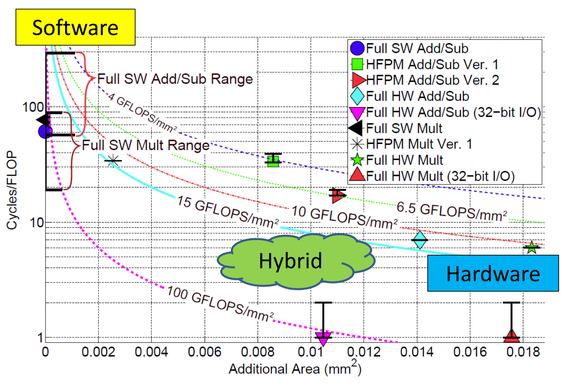
Hybrid Modules Compared Against Full Software:

✳ **HFPM Mult Ver. 1**
- Area increase: **1.5%**, Multiplication speedup: **2.3x**

HFPM Add/Sub Ver. 1
- Area increase: **5.1%**, Add/Sub speedup: **1.8x**

HFPM Add/Sub Ver. 2
- Area increase: **6.5%**, Add/Sub speedup: **3.6x**

- **Full hardware modules:** have the **highest throughput**, but require the **most area**
- **Full software modules:** **don't require any additional area**, but have the **lowest throughput**
- **Hybrid modules:** **offer midpoints** between full software and full hardware

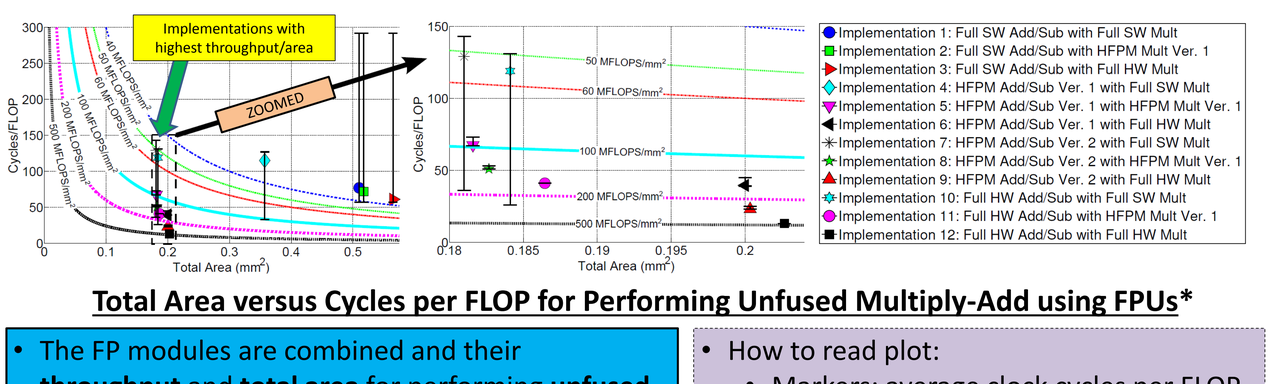


Additional Area versus Cycles per FLOP for all FP Modules*

- How to read plot:
  - Markers: average clock cycles per FLOP
  - Endpoints of interval bars: min/max cycle counts
  - Contour lines: throughput per additional area tradeoffs

*Results based on synthesis in 65 CMOS with a supply voltage of 1.3 V at 1.2 GHz.

8. Comparison of Floating-Point Modules Combined into FPUs



Total Area versus Cycles per FLOP for Performing Unfused Multiply-Add using FPUs*

- The FP modules are combined and their **throughput** and **total area** for performing **unfused multiply-add** is plotted above
- Each floating-point unit consists of:
  - 1 addition/subtraction module
  - 1 multiplication module

- How to read plot:
  - Markers: average clock cycles per FLOP
  - Endpoints of interval bars: min/max cycle counts
  - Contour lines: throughput per additional area tradeoffs

- **Implementations with a HFPM or Full HW Add/Sub module have the largest throughput per area.**
- ● **Imp. 1, Full Software FPU:** No additional area, **15.6 MFLOPS** throughput
- ■ **Imp. 12, Full Hardware FPU:** ~20% additional area, **92.3 MFLOPS** throughput
- **9 implementations provide higher throughput/area than Full Software FPU**
- ✳ **Imp. 7:** lowest throughput (due to IMEM usage), 1.7x increase in throughput per area
- ★ **Imp. 8:** only hybrid modules, 4.1x increase in throughput, 9.95% smaller than Full Hardware FPU

*Results based on synthesis in 65 CMOS with a supply voltage of 1.3 V at 1.2 GHz.

9. Summary

- 3 hybrid floating-point modules were presented for a fine-grained processor
- 12 FPU implementations were synthesized in 65 nm CMOS
- Throughput was increased over a software implementation by utilizing custom FP instructions
- Area overhead was kept low by reusing the processor's fixed-point ALU

- Nine increase throughput/area by 1.05-8.5x versus a Full Software FPU
- Nine use 1.08-12.5x less area than a Full Hardware FPU
- **The throughput of floating-point arithmetic was increased without incurring the area overhead of full hardware floating-point units**

10. Acknowledgments: The authors gratefully acknowledge support from C2S2 Grant 2047.002.014, NSF Grant 1018972 and 0903549 and CAREER Award 0546907, SRC GRC Grant 1598, 1971, and 2321 and CSR Grant 1659, Intel, UC Micro, SEM, Fudan University, Frances Muenzer, Stanley Hsu, Nima Mostafavi, Aaron Stillmaker, Bin Liu, and Brent Bohnenstiehl. The authors would like to thank STMicroelectronics for donating the chip fabrication.