Table of Contents for this manuscript

Does Arithmetic Logic Dominate Data Movement? A Systematic Comparison of Energy-Efficiency for FFT Accelerators

Tung Thanh-Hoang[1], Amirali Shambayati[1], Henry Hoffmann[1], and Andrew A. Chien[1, 2]

[1]Department of Computer Science, University of Chicago, Illinois, USA
[2]Argonne National Laboratory, Chicago, Illinois, USA
{hoangt, fywkevin, achien}@cs.uchicago.edu

***Abstract*—In this paper, we perform a systematic comparison to study the energy cost of varying data formats and data types w.r.t. arithmetic logic and data movement for accelerator-based heterogeneous systems in which both compute-intensive (FFT accelerator) and data-intensive accelerators (DLT accelerator) are added. We explore evaluation for a wide range of design processes (e.g. 32nm bulk-CMOS and projected 7nm FinFET) and memory systems (e.g. DDR3 and HMC).**

First, our result shows that when varying data formats, the energy costs of using floating point over fixed point are 5.3% (DDR3), 6.2% (HMC) for core and 0.8% (DDR3), 1.5% (HMC) for system in 32nm process. These energy costs are negligible as 0.2% and 0.01% for core and system in 7nm FinFET process in DDR3 memory and slightly increasing in HMC. Second, we identify that the core and system energy of systems using fixed point, 16-bit, FFT accelerator is nearly half of using 32-bit if data movement is also accelerated. This evidence implies that system energy is highly proportional to the amount of moving data when varying data types.

I. Introduction

Fast Fourier Transforms (FFT) has been widely used in many numeric computations ranging from scientific computing to embedded signal processing. To achieve high performance and energy efficiency, FFT compute is often performed by hardware accelerator [1]. In a natural manner, numerical algorithms are usually developed with floating point data format in order to achieve sufficient accuracy and error detection mechanisms. However, the implementation of floating point arithmetics suffers from performance, power and area overhead comparing to fixed-point format. Therefore, there is a need to develop efficient floating-to-fixed point conversion algorithms to minimize quantization error that requires high design time (30-50% of total development cycle of software) [2]. While the technology node is being shrunk down that reduces the relative energy cost of arithmetic logic for varying data formats. Meanwhile, the improvement of memory technology (e.g. off-chip memory) is far behind logic. This causes the impact of data movement, as a significant fraction of system energy, becomes an important concern for not only traditional architectures but also future accelerator-based heterogeneous systems in that application executions is often memory-limited [3].

Motivation: The impacts of data format and data type selection on the energy of accelerator-integrated systems are raising two interesting questions: *1) Does the energy cost of arithmetic logic dominate data movement when varying data formats? 2) Is the system energy proportional to the amount of moving data when varying data types?*

Contributions: Our result suggests that -*with the system perspective*- the relative energy cost using floating point over fixed point is insignificant (at most 6.2%)in accelerated-based heterogeneous system for a range of process and memory. Therefore, we can make use of floating point arithmetics and its advantages. This estimation also shows that both core and system energy are strongly dependent on the amount of moving data due to varying data types which envisions the design of energy-proportional accelerator-based systems.

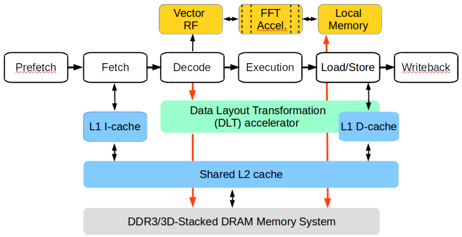


Fig. 1: Architecture of the Accelerated-integrated Systems.

II. FFT Accelerator Design and Integration

We consider the combination of one FFT accelerator and a Data Layout Transformation (DLT) accelerator which is used to accelerator data movement and transposition for the FFT compute. The high-level architecture is illustrated in Figure 1.

FFT accelerator: We consider three FFT accelerators: *1)* Fft64Fl32 is an optimized hardware accelerator generated by using Spiral tool [4]. This accelerator can compute 64 complex value of single-precision, floating point. In addition, the output of FFT accelerator is extended to perform parallel multiplication with twiddle factors stored in vector register which is a necessary step to compute large FFTs. Synthesizing in 32nm process, the Fft64Fl32 accelerator runs at 1Ghz and contain 3.8M gates which is extremely larger than the RISC core. *2)* Fft64Fx32 is designed in the same manner with the Fft64Fl32 accelerator to run at 1Ghz with same latency (2.3M gates in 32nm process). *3)* Fft64Fx16 is designed in the same manner with the Fft64Fx32 accelerator to run at 1Ghz with same latency (1.1M gates in 32nm process).

DLT accelerator: this is a specialized hardware to accelerate *i)* data movement (i.e., gather/scatter stride data) between the local memory and off-chip memory *ii)* transposition of data layout inside the local memory. Data access pattern is compacted into a descriptor which is composed of the number of moving elements, stride of element, and size of element. The DLT accelerator supports sufficient data movement types and incurs low performance/implementation cost compared to advanced DMA [5]. The DLT accelerator consists of pipeline stages (instruction decoder, fast parallel address generator and read/write memory access units) which can be tightly integrated into RISC pipeline without performance overhead.

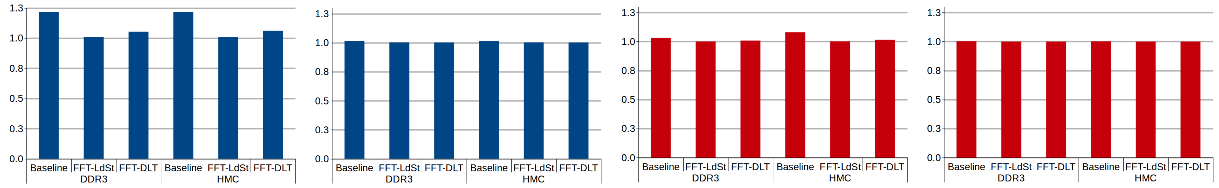
Programmability: Table I shows C-intrinsic functions which are directly mapped to ISA and implemented in microarchitecture for the FFT and DLT accelerators.

978-1-4799-1925-3/15/$31.00 © 2015 IEEE

Table of Contents for this manuscript

TABLE I: The Instruction Set Architecture (ISA) of FFT and DLT accelerators.

Instruction	Intrinsic	Functional description
Fft64Fl32	Fft64Fl32(Opt, *srcA0, *srcA1, *tdmA0, *tdmA1)	Compute 64 complex samples, 32-bit floating FFT. The FFT output can be multiplied with 64 complex floating point twiddle factors via *Opt* argument.
Fft64Fx32	Fft64Fx32(Opt, *srcA0, *srcA1, *tdmA0, *tdmA1)	Provide the same function as *Fft64Fx32* except data type is fixed point 32-bit.
Fft64Fx16	Fft64Fx16(Opt, *srcA, *tdmA)	Provide the same function as *Fft64Fx32* except data type is fixed point 16-bit.
LoadLm2Vr	LoadLm2Vr(vrDst, *srcA)	Load 256 bytes from local memory memory to vector register.
StoreVr2Lm	StoreVr2Lm(vrSrc, *dstA)	Store 256-byte vector register to local memory memory.
DltFormDesc	int32 DltFormDesc(num_elems, stride, elem_size)	Form 32-bit descriptor of DLT for data movement.
DltGather	DltGather(*mmA, *lmA, desc)	Gather data from main memory (mmA) to local memory (lmA).
DltScatter	DltScatter(*lmA, *mmA, desc)	Scatter data from local memory (lmA) memory to main memory (mmA).
DltGatherFence	DltGatherFence()	Memory synchronization used to check the completion of all issued gather inst.
DltScatterFence	DltScatterFence()	Provide the same function as *DltGatherFence* but for scattering.



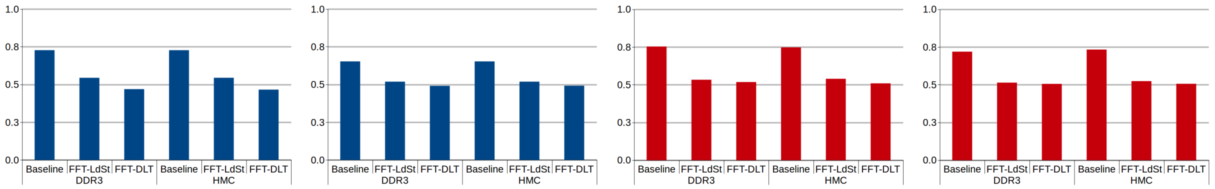
(a) Core/32nm.

(b) Core/7nm.

(c) System/32nm.

(d) System/7nm.

Fig. 2: Energy impact of varying data formats (32-bit, floating/fixed point energy ratio).



(a) Core/32nm.

(b) Core/7nm.

(c) System/32nm.

(d) System/7nm.

Fig. 3: Energy impact of varying data types (fixed point, 16-bit/32-bit energy ratio).

Execution phases: We separate FFT compute in separate execution phases such as *i) Marshal*, *ii) Compute*, *iii) Transpose*, *iv) Vector LD/ST*, *v) De-marshal* (see [6] for the detail definition of phases).

TABLE II: Simulation Platform Configuration.

Parameter	Value
Core type	In-order, 5-stage pipeline, MIPS-like 16/32-bit ISA
Vector RF	2048b x 18 registers
Wide IO local mem.	64 banks (32bx16k), total 2048b wide
Cache hierarchy	L1-I: 32KB, 2-cyc. latency, L1-D: 24KB, 2-cyc. latency Shared L2: 512KB, 10-cycle latency
Main memory	2GB/4-rank/16-dev. DDR3 or 4GB/4-rank/8-dev. HMC

III. Methodology and Experiment

Table II show the system configuration which is used to evaluate following systems: **Baseline** (stand-alone RISC32 processor), **FFT-LdSt** (using FFT accelerator and general Load/Store instructions for data movement) and **FFT-DLT** (federation of FFT and DLT accelerators).

IV. Experimental Results

We define evaluation metrics as: a) **CORE logic energy** is sum of energy of Compute, Vector Ld/St and Transpose phases (excluding DRAM energy) b) **SYSTEM energy** is sum of energy of all phases including DRAM energy[1].

A. Energy Impact of Varying Data Formats

At the core logic level when both FFT and DLT accelerators are used to accelerate compute and data movement (FFT-DLT designs) the core energy costs of same 32-bit floating over fixed point with same 32-bit length are *relatively small* such as: 32nm (Fig. 2a), 5.3% (DDR3) and 6.2% (HMC), 7nm (Fig. 2b), 0.3% (DDR3) and slightly higher (HMC). At the system level, the system energy costs due to varying data formats of FFT-DLT designs are *decreasing (and eventually negligible)*: 32nm (Fig. 2c), 0.8% (DDR3) and 1.5% (HMC), 7nm (Fig. 2d), $<$ 0.01% (DDR3 and HMC).

B. Energy Impact of Varying Data Types

Comparing the core energy of fixed point, 16-bit and 32-bit FFT compute, our evaluation shows that using 16-bit fixed point results in 47% (DDR3), 48% (HMC) energy reduction compared to using 32-bit fixed point in 32nm bulk-CMOS (Fig. 3a) and 7nm FinFET (Fig. 3b) process respectively. At the system level, the relative energy of varying data types for fixed point, 32-bit over 16-bit data format is negligible across memory systems. We observe the energy cost due to using 16-bit fixed point is almost 50% in 32nm CMOS process (Fig. 3c) and even getting closer in 7nm FinFET (Fig. 3d).

References

[1] A. Pedram et al. “Transforming a linear algebra core to an FFT accelerator”. In: *ASAP*. 2013.

[2] M. Clark et al. *Accelerating Fixed-Point Design for MB-OFDM UWB Systems*. Tech. rep. CommsDesign, 2005.

[3] H. T. Tung et al. “Performance and Energy Limits of a Processor-integrated FFT Accelerator”. In: *HPEC*. 2014.

[4] P. Milder et al. “Computer Generation of Hardware for Linear Digital Signal Processing Transforms”. In: *ACM Trans. Des. Autom. Electron. Syst.* 17.2 (Apr. 2012).

[5] *TMS320C6748/46/42 and OMAP-L138 Processor Enhanced Direct Memory Access Controller*. Tech. rep. Texas Instruments, 2010.

[6] H. T. Tung et al. *A Systematic Comparison of Energy-Efficiency: Float and Fixed Point Processor-integrated FFT Accelerators*. Tech. rep. 2015.

[1] The detail evaluation for phase energy can be found in [6].