A Comparison of Accelerator Architectures for Radio-Astronomical Signal-Processing Algorithms

John W. Romein
ASTRON (Netherlands Institute for Radio Astronomy)
Postbus 2, 7990 AA Dwingeloo, The Netherlands
Email: romein@astron.nl

Abstract—**In this paper, we compare a wide range of accelerator architectures (GPUs from AMD and NVIDIA, the Xeon Phi, and a DSP), by means of a signal-processing pipeline that processes radio-telescope data. We discuss the mapping of the algorithms from this pipeline to the accelerators, and analyze performance. We also analyze energy efficiency, using custom-built, microcontroller-based power sensors that measure the instantaneous power consumption of the accelerators, at millisecond time scale. We show that the GPUs are the fastest and most energy efficient accelerators, and that the differences in performance and energy efficiency are large.**

I. Introduction

This paper compares a wide range of accelerators, by means of a signal-processing pipeline that processes data from a radio telescope. We implemented and optimized the algorithms from this pipeline on GPUs from AMD and NVIDIA, the Intel Xeon Phi, a Digital Signal Processor (DSP) from Texas Instruments, and a dual Intel Xeon CPU as reference platform. The algorithms have quite different compute and memory access characteristics, hence we assess various subsystems of the accelerators. The pipeline performs some domain-specific operations like tracking a sky source, but also contains standard signal-processing algorithms found in many more application domains (e.g., radar, WiFi equipment, audio processing, imaging).

The main contribution of this paper is an analysis of the performance and energy efficiency of these accelerators. We describe the optimizations needed to obtain high performance. Additionally, we describe how we use custom-built, microcontroller-based devices to measure the energy drawn by the (PCIe) boards, at high time resolution. The accelerated applications are instrumented to gather power sensor data during their execution, and determine their own energy efficiency. We show that GPUs yield the best performance and energy efficiency; even better than the DSP, that has optimized hardware support for signal processing. We also show that the differences between the (energy) efficiencies of the accelerators are large.

This paper is structured as follows. The algorithms and architectures used in this study are described in II and Sections III, respectively. Section IV describes the implementations and optimizations of the algorithms on the different architectures. Sections V and VI analyze performance and energy efficiency. Section VII discusses related work, and Section VIII concludes.

II. Signal-processing algorithms

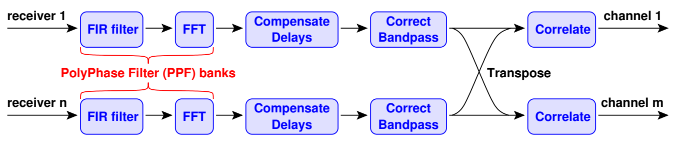


Fig. 1: Correlator pipeline.

Many radio telescopes use multiple receivers, to increase sensitivity and angular resolution. Together, they form a distributed sensor network. In the field, receiver data is digitized, preprocessed, and sent via dedicated WAN links to a central location. There, the data is received, aligned, filtered, corrected, and combined (correlated), by the *correlator pipeline* (Fig. 1). Finally, data that is affected by interference is detected and discarded, and the remaining data is calibrated and imaged. This paper only considers the correlator pipeline. This pipeline runs in real time and is offloaded to accelerators, because of the high computational demands and data rates.

The first step is a *Poly-Phase Filter (PPF) bank* that splits a frequency band into narrower frequency channels. The PPF bank consists of a series of Finite Impulse Response (FIR) filters and an FFT block (see Fig. 2). Input samples are round-robin distributed over the FIR filters, and the FIR filter outputs are subsequently Fourier transformed (FFT).

The *FIR filters* are band pass filters that attenuate high and low frequencies. The complex-valued samples are convolved with real filter weights (Fig. 3). For this paper, we use a PPF bank with 64 FIR filters of 16 weights and history entries each.

The next step is *delay compensation*, to align the signals from the observed sky source. Its wave front hits the receivers at different times (see Fig. 4). The delay is implemented as a phase rotation of the signal (a complex multiplication per sample), and is interpolated in time and frequency (requiring another complex multiplication).

Bandpass correction flattens a ripple (Fig. 5) that is caused by another PPF bank near the receivers, in the field. The amplitude of the signal is multiplied by a constant, real, channel-dependent weight which is computed in advance.

Next, the data is transposed in memory; the need for this is explained in Section IV. Finally, the filtered and corrected data from the different receivers are *correlated*. For each *pair* of stations, we multiply a filtered and corrected sample from one

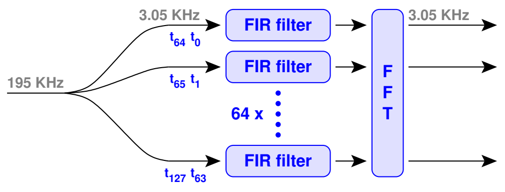


Fig. 2: PPF bank.

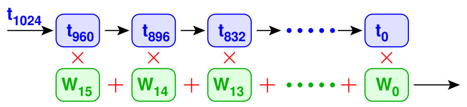


Fig. 3: FIR filter.

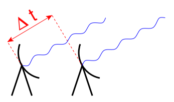


Fig. 4: Receive time difference.

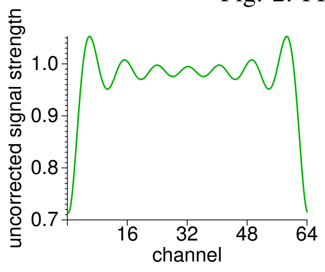


Fig. 5: Uncorrected signal powers.

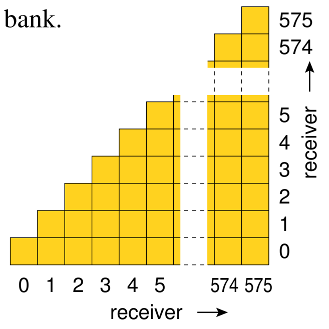


Fig. 6: Correlation triangle.

station with the complex conjugate of a sample from the other station. The products are accumulated over time, and the result is stored in a triangular data structure (see Fig. 6). There is no need to compute the upper triangle, as the correlation matrix is Hermitian. The correlator kernel resembles the BLAS routine CHERK, but our triangular output data structure is twice as compact. Computing the correlations is computationally the most expensive operation of the pipeline.

For simplicity, we ignore polarizations; we assume dual-polarized receivers throughout this paper.

III. Architectures

Table I lists the characteristics of the used accelerators. We include server-grade and consumer-grade hardware, as the consumer-grade hardware is typically one generation ahead of the server-grade hardware. Most accelerators are powerful devices that consume hundreds of Watts, but the DSP is a low-power System-on-Chip (SoC) that needs about 20W.

All architectures are highly parallel, and use three levels of parallelism. First, loosely-coupled cores provide MIMD[1] concurrency. As these cores operate independently, they are supposed to exploit coarse-grain parallelism. Second, each of these cores uses SIMD (vector-like) instructions, so that only one instruction decode and issue is necessary to perform many identical operations. SIMD instructions are suitable to exploit fine-grain parallelism. Third, all architectures also support instruction-level parallelism, by dispatching multiple instructions to different execution pipelines (ALUs, load-store units, branch units, etc.) in a single cycle.

All architectures have multiple layers of memory: device memory, transparent caches, and a register file. In addition, the GPUs and DSP have fast, local memories that can be used explicitly by the programmer to share and reuse data.

The AMD and NVIDIA GPUs are similar. Each processing unit runs groups of light-weight threads that cooperate on a single task, usually follow the same instruction path, share a piece of fast local memory, and synchronize through barrier instructions. Memory and instruction latencies are hidden by having tens of instructions per processing unit in flight, so that most cycles the instruction scheduler finds enough instructions ready for dispatch to keep all execution units busy. Hence, the programmer should provide more parallelism than on a regular CPU. Computations and PCIe I/O overlap by submitting commands to different command queues.

The Xeon Phi (Knights Corner) has up to 61 regular CPU cores. Each core is equipped with a vector unit that operates on 16 single-precision floating-point values simultaneously. Prefetching data into the L2 and L1 caches (either compiler generated or manually inserted), alignment to 64 bytes, and the use of non-sequentially-consistent streaming writes are essential to obtain good performance.

The EVMK2H is a SoC with 4 ARM cores and 8 DSP cores. The DSP has a VLIW architecture with eight specialized execution units. It has hardware features that support signal processing, such as circular addressing, complex vector-matrix instructions, and optimized support for 16-bit complex numbers. Loops can be pipelined, i.e., multiple loop iterations are processed simultaneously and out of phase, subject to resource constraints, timing constraints, and (loop-carried) dependencies. DMA controllers can be used to move data between the different types of memories. Altogether, the DSP is a complex processor with many specialized hardware features exposed to the programmer.

[1]Flynn's taxonomy: SIMD = Single Instruction, Multiple Data; MIMD = Multiple Instruction, Multiple Data.

	model	type	intro date	architecture	tech. (nm)	clock (GHz)	core config[a] = #FPUs	peak TFLOPS	mem sz (GB)	mem bw (GB/s)	TDP[d] (W)
server grade	AMD ***FirePro S10000***	GPU	Q4'12	Tahiti	28	0.825	2×28×1×64 = 3584	5.91	2×6	480[c]	375
	Intel ***Xeon Phi 7120X***	CPU	Q2'13	Knights Corner	22	1.33[b]	1×61×1×16 = 976	2.60[b]	16	352[c]	300
	NVIDIA ***Tesla K40***	GPU	Q4'13	Kepler	28	0.875[b]	1×15×6×32 = 2880	5.04[b]	12[c]	<288[c]	235
	TI ***EVMK2H***	DSP	Q4'13	KeyStone 2	28	1.167	1× 8×2× 8 = 128	0.15	2+2	10.7	>19
	Intel ***Xeon E5-2697v3***	CPU	Q3'14	Haswell EP	22	2.60[b]	2×14×2× 8 = 448	2.78[b]	≤1536	136[c]	290
consumer	NVIDIA ***Titan X***	GPU	Q1'15	Maxwell	28	1.113[b]	1×24×4×32 = 3072	6.84[b]	12	336	250
	AMD ***R9 Fury X***	GPU	Q2'15	Fiji	28	1.050	1×64×1×64 = 4096	8.60	4	512	275
	NVIDIA ***GTX 1080***	GPU	Q2'16	Pascal	16	1.80[b]	1×40×2×32 = 2560	9.22[b]	8	320	180

[a] #ICs × #compute units × FPU instructions/cycle × vector size
[b] turbo mode enabled
[c] ECC (error correction) enabled; this reduces memory size (on K40) and bandwidth
[d] Thermal Design Power

TABLE I: Processor characteristics.

IV. Implementations and optimizations

We implemented the processing pipeline as described in Section II on all platforms. We programmed all accelerators in their native languages: CUDA for NVIDIA GPUs, OpenCL for AMD GPUs, C++/OpenMP with auto-vectorization, pragmas, and intrinsics on the Xeon and Xeon Phi, and a mix of assembly, C++, and OpenCL on the DSP. For the FFT, we use vendor-supplied libraries (NVIDIA cuFFT, Apple/AMD clFFT, Intel MKL, TI DSPlib).

The pipeline consists of two parts: all kernels except the correlator, and the correlator kernel itself. The first part is coarse-grain parallel along the number of receivers, as these are processed independently. The frequency channels provide fine-grain parallelism; they are created simultaneously by the PPF bank. Conversely, in the second part, the correlator processes the frequency channels independently, but combines all receivers in a fine-grain parallel way. This has two major implications: we need to change parallelization strategies between the first and the second part of the pipeline, and we need to transpose data in memory.

The implementations for the different GPUs are similar. Delay compensation, bandpass correction, and transpose are combined into one kernel, to make fewer passes over the data. Some optimizations are architecture specific, e.g., texture memory reads are only beneficial on the Kepler-based K40.

On all architectures, the correlator kernels minimize memory bandwidth usage by accumulating correlations in registers, and by caching samples in all levels of the memory hierarchy, including registers [1]. The correlation triangle (Fig. 7) is decomposed hierarchically; e.g., on GPUs, in blocks of 32x32 receivers. One kernel computes triangular blocks near the diagonal (green), another kernel computes square blocks (yellow), and, if the number of receivers is not a multiple of 32, a third kernel processes rectangular blocks on the right (gray). Each block is processed by a (blue) CUDA thread block or OpenCL work group, and is decomposed further into (red) 2x2 subblocks that are processed by individual threads. The threads within a thread block collectively transfer samples from 32 receivers from device memory or L2 cache to local memory, and each thread transfers samples from local memory to registers. Different channels are correlated independently by different thread blocks. Other architectures decompose the work and minimize memory bandwidth usage similarly.

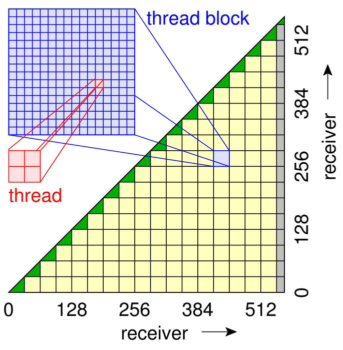


Fig. 7: Decomposition of work in the correlate kernel.

The Xeon and Xeon Phi implementations are vectorized by using a mix of intrinsics and OpenMP SIMD directives. As the instruction set does not support complex-complex vector multiplications without explicit shuffling, complex data is stored in separate arrays of real and imaginary numbers (where the other implementations store arrays of interleaved real and imaginary numbers). The Xeon Phi implementation uses OpenMP 4.0 offloading. The Xeon and Xeon Phi implementations of the correlator kernel are similar (except for vector length), but the other kernels are implemented quite differently. The Xeon Phi FIR filter profits from the four times larger vector register file by keeping weights and samples in registers; the register file of the regular Xeon is too small for this. However, this optimization disallows another optimization, namely, that the Fourier-transformed data is kept in L1 cache, as the Xeon implementation does. Additionally, the transpose implementations are different, as the penalty for scattering data to memory is acceptable on the Xeon and prohibitively expensive on the Xeon Phi. Therefore, the Xeon Phi transposes data hierarchically, transposing blocks of 16x16 values within vector registers using a long sequence of permute, swizzle, and shuffle operations. The Xeon Phi makes one more pass over the data than the Xeon, but we found that this was still the most efficient way.

We developed two implementations for the DSP: one that operates on single-precision floating-point values (like the other accelerators), and one that operates on mixed-precision integers that fits the DSP architecture better. The accuracy loss of limited-precision integers is tolerable. All kernels are written in assembly, as the C compiler does not generate efficient code, not even when using intrinsics. We only use the OpenCL runtime to dispatch work to the DSP cores. Also, all DSP-specific features like software-managed caches, DMA controllers, some complex instructions, and circular addressing are extensively used. We cannot overemphasize that some of these features, especially the DMA controllers, severely complicate programming.

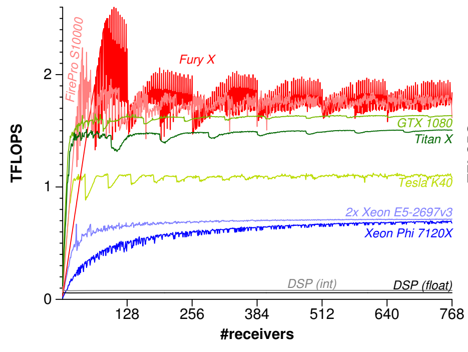
V. Performance results

In this section, we analyze the performance that we obtain on the different architectures. Figures 8a–8d show the performance of the four kernels for up to 768 receivers. Figure 8e shows the performance of the whole processing pipeline. Generally, the GPUs perform best, followed by the Xeon, Xeon Phi, and the low-power DSP.

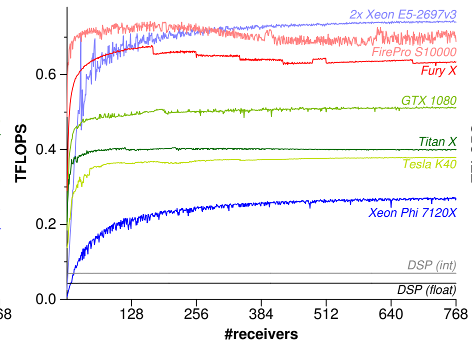
The fluctuations in the Fury X curve of Fig. 8a and all GPU curves in Fig. 8c are caused by the sensitivity of the memory subsystems to particular access strides. Normally, padding helps to avoid inefficient access strides, but we do not do so here as it would affect FFT performance negatively.

The GPU curves of Fig. 8d (and 8e) show that multiples of 32 receivers perform better than non-multiples of 32 receivers. The latter require a third kernel invocation that processes the rectangular leftovers at the right of the correlation triangle (the gray rectangles in Fig. 7). This kernel may leave some cores idle, provides less parallelism to hide all latencies, and needs to execute additional instructions to check boundaries.

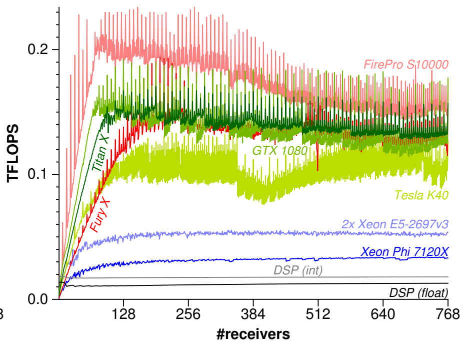
Below, we analyze the performance results in more detail.



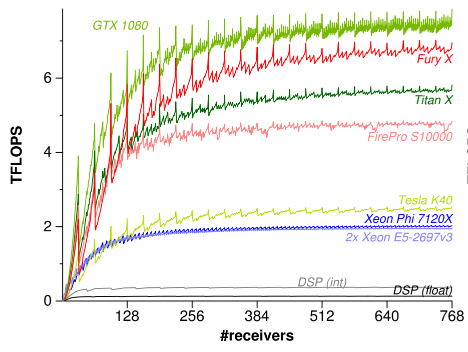
(a) FIR filter



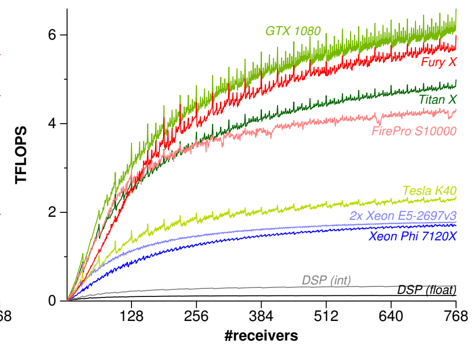
(b) FFT



(c) delay/bandpass/transpose



(d) correlate



(e) full pipeline

Fig. 8: Performance of the individual kernels, as well as the full pipeline.

A. *Compute and memory bounds*

To determine if a kernel is compute bound or memory I/O bound, we start with a roofline analysis [2], as shown in Fig. 9. The Y-axis represents performance, i.e., the number of operations per second. The X-axis represents the *operational intensity* (OI), i.e., performance divided by memory bandwidth usage, in FLOP/byte. A compute-bound kernel has a high OI; a memory-I/O bound kernel has a low OI. The roofline shows the architecture's performance limit, as function of the OI.

Each dot in a graph represents a kernel on an architecture. A dot closely below the horizontal part of the roofline indicates that the kernel is compute bound; a dot closely below the diagonal part indicates that the kernel is memory-I/O bound. A dot far below the roofline indicates that it is bound by something else. We only count FLOPs that contribute to the result, not overhead instructions like address calculations and branch instructions. To measure the bandwidth, we use the device's hardware performance counters to measure DRAM bandwidth usage, so cache hits do not count, but, for example, mis-speculated prefetches do.

For this analysis, we keep the problem size fixed: 576 receivers, the largest number correlated by any radio telescope (AARTFAAC).

Theoretically, the FIR filter, FFT, and delay/bandpass/transpose kernels (Fig. 9a–9c) have expected FLOP/byte ratios of 6.4, 1.87, and 0.75, respectively. In most cases, the measured FLOP/byte ratios match the theoretical values. The ratio is lower than expected on the Tesla K40, due to ECC overhead. It is much lower on the Xeon Phi 7120X in Fig. 9c, as this kernel makes a second pass over the data. It is much higher on the DSP and the dual E5-2697v3 CPU, because the intermediate data remains in SRAM/cache. The three kernels are I/O bound, which explains why the AMD GPUs outperform the other accelerators: these GPUs have the highest memory bandwidth.

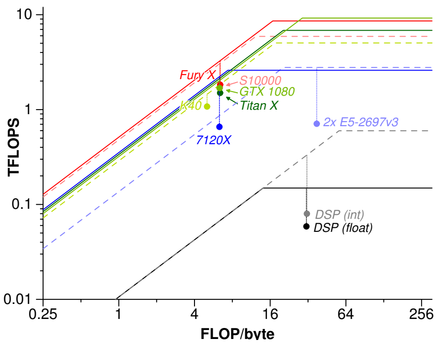
For the correlator kernel (Fig. 9d) the FLOP/byte ratio varies, but the high data reuse through caches and the register files makes this kernel compute bound on all architectures. On most architectures, the correlator reaches more than 80% of the FPU peak performance.

B. *Other bounds*

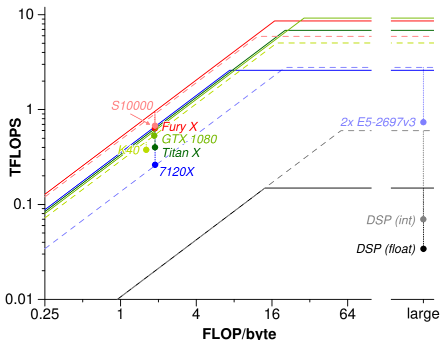
Figure 9 shows that in several cases, the performance is quite distant from its roofline, usually indicating that the kernel is neither compute bound, nor memory-I/O bound. The large distances for the Xeon Phi 7120X (Fig. 9(a–c)) are still due to memory performance; the roofline is based on an advertised bandwidth of 352 GB/s, but the practical upper bound is around 160 GB/s, so the diagonal part of the roofline is unrealistically high. Fang et al. [3] attribute this to ring saturation, contention on the distributed tag directories, and ECC overhead.

On the DSP, the peak integer performance can only be achieved by using powerful complex vector-matrix multiplication instructions, but a FIR filter with real filter weights and the FFT do not map well to these instructions. This is surprising; one would expect excellent hardware support on a DSP for such basic signal-processing algorithms. The correlator algorithm also does not map well to these instructions, as the input data must be reordered to use them.

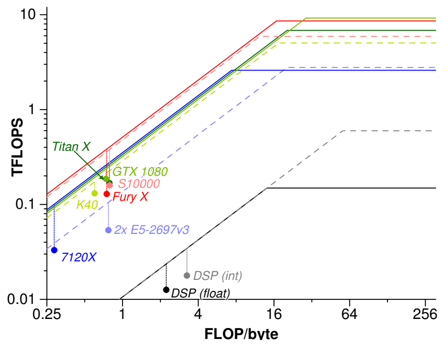
The Kepler-based Tesla K40 achieves only 50% of the FPU peak performance on the correlator kernel; it cannot fully hide the FPU and memory latencies. Its successor architecture,



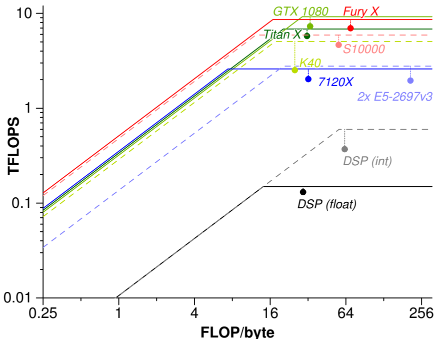
(a) FIR filter



(b) FFT



(c) delay/bandpass/transpose



(d) correlate

Fig. 9: Roofline plots for all kernels and architectures.

Maxwell, has many architectural improvements (shorter instruction latencies, improved instruction scheduling, etc.), and indeed: the Titan X achieves 85%.

The Xeon CPU loses performance in the FIR filter (Fig. 9a) due to vector register spilling and the inability to hide the latencies of the AVX2 fused multiply-add instructions.

VI. Energy efficiency

We study energy efficiency not only because this is important from environmental and cost perspectives, but also because this is a fair way to compare accelerators with different power demands. The devices cannot measure their own energy consumption at the high time resolutions necessary to measure individual kernels. Hence, we built custom power meters out of microcontrollers, current sensors, and PCIe riser cards, similar to PowerInsight [4]. Unlike PowerInsight, we only measure currents and not voltages; we verified that the voltages are sufficiently stable to compute the power consumption. The measurements are reported back to the host via USB, and a small library allows the accelerated applications to determine their own power consumption and energy efficiency. For PCIe cards, we measure the 3.3V and 12V lines of the PCIe slot, and the external power cables. We measure the power of the entire DSP board, thus the power consumption of the ARM cores is included, unlike the PCIe boards. We do not measure the Xeon reference platform with the power meters, but use LIKWID [5] to measure the CPU package and DRAM power.

Most devices can trade performance for better energy efficiency (see Price et al. [6] for an in-depth study), but for these energy measurements, we kept the same, aggressive clock settings as used in Section V.

As the power measurements are near-millisecond accurate, we see how the energy consumption varies when an accelerator runs the consecutive kernels of the pipeline. Figure 10 shows the instantaneous power consumption of the Titan X. The figure shows that some kernels draw much more energy than others.

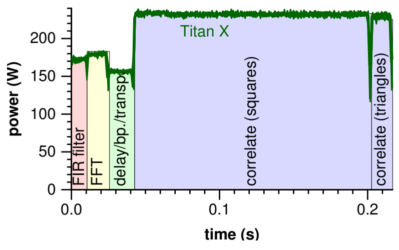


Fig. 10: Energy consumption traces.

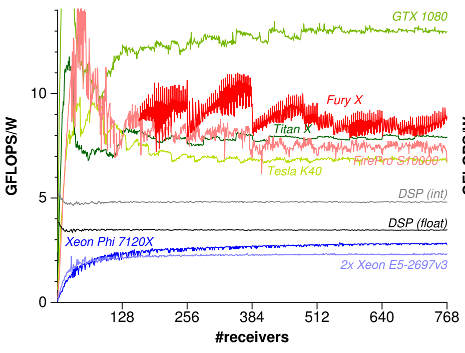
Figure 11 shows energy efficiency, as function of number of stations. The recently introduced, 16 nm technology-based GTX 1080 is by far the most energy-efficient platform, nearly seven times more efficient than the reference CPU platform. The high energy-efficiency of the FFT kernel on the DSP is due to the data being kept in L1 SRAM, not due to special signal-processing features.

VII. Related work

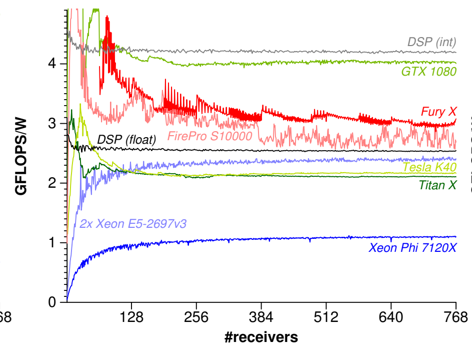
Five years ago, we studied the (compute-bound) correlator kernel [7, 8] on contemporary accelerators. This paper extends that work by not only considering the correlator kernel, but a whole pipeline of signal-processing algorithms with a mixture of compute-bound and memory-I/O-bound kernels. In addition, we now assess the energy efficiency of the accelerators.

Many radio telescopes worldwide use GPUs to correlate data: LOFAR, the LWA, MWA, and GMRT use NVIDIA GPUs, while AARTFAAC and CHIME use AMD GPUs. The LOFAR and AARTFAAC correlators are based on the program code used for this paper, while the LWA and MWA are based on the xGPU [9] library. Fiorin et al. [10] propose a custom architecture for the future SKA telescope.

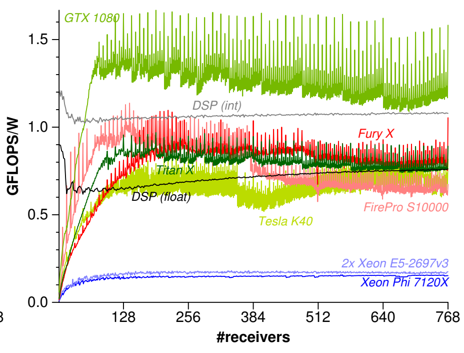
There are more papers that compare multiple accelerators (typically GPUs and the Xeon Phi), for example in the domain of pulsar searching [11], image processing [12], and pattern



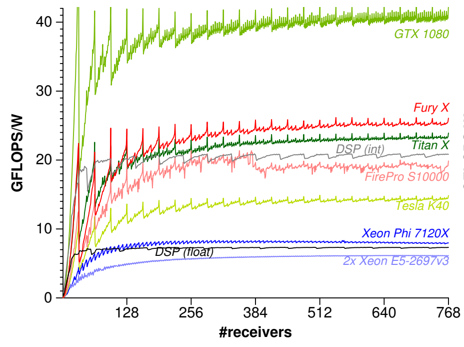
(a) FIR filter



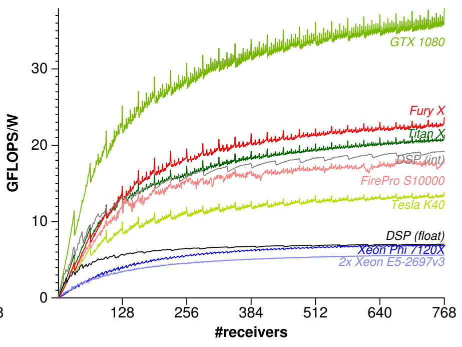
(b) FFT



(c) delay/bandpass/transpose



(d) correlate



(e) full pipeline

Fig. 11: Energy efficiency of the individual kernels, as well as the full pipeline.

matching [13]. These papers compare performance, but none of them assesses energy efficiency.

VIII. Conclusions and Future Work

The take-away message of this study is that GPUs are the most (energy) efficient accelerators — at least for a radio-astronomical correlator pipeline — and that the differences in performance and energy efficiency between the accelerators are large. It is difficult to obtain good memory performance on the Xeon Phi. The DSP has architectural features that support signal processing (like circular addressing, loop pipelining, some complex instructions, and multi-level DMA controllers), but these features make it difficult to program and do not make it more energy efficient than GPUs. The custom-built power sensors that we introduced in Sec. VI are highly useful to analyze energy efficiency, at millisecond timescales.

Future work includes a comparison with FPGAs. In a separate work, we study how to efficiently create sky images from correlated data on these accelerators.

Acknowledgments

This work is supported by the NWO through Open Competition (Triple-A) and NWO-M grants (DAS-4, DAS-5 [14]), and by the Dutch Ministry of EZ and the province of Drenthe through the Dome grant. We thank AMD, Intel, and NVIDIA for their support and for providing us with hardware.

References

[1] J. Romein, P. Broekema, J. Mol, and R. van Nieuwpoort, "The LOFAR Correlator: Implementation and Performance Analysis," in *PPoPP'10*, Bangalore, India, January 2010, pp. 169–178.

[2] S. Williams, A. Waterman, and D. Patterson, "Roofline: An Insightful Visual Performance Model for Multicore Architectures," vol. 52, no. 4, pp. 65–76, April 2009.

[3] J. Fang, H. Sips, L. Zhang, C. Xu, Y. Che, and A. Varbanescu, "Test-Driving Intel Xeon Phi," in *ICPE'14*, Dublin, Ireland, March 2014, pp. 137–148.

[4] J. Laros III, P. Pokorny, and D. DeBonis, "PowerInsight – A Commodity Power Measurement Capability," in *Int. Workshop on Power Measurement and Profiling*, Arlington Va, June 2013.

[5] J. Treibig, G. Hager, and G. Wellein, "LIKWID: A Lightweight Performance-oriented Tool Suite for x86 Multicore Environments." in *ICPPW'10*, San Diego, CA, September 2010, pp. 207–216.

[6] D. Price, M. Clark, B. Barsdell, R. Babich, and L. Greenhill, "Optimizing performance-per-watt on GPUs in high performance computing," *Computer Science – Research and Development*, pp. 1–9, Sept. 2015.

[7] R. van Nieuwpoort and J. Romein, "Correlating Radio Astronomy Signals with Many-Core Hardware," *Int. Journal of Parallel Processing*, vol. 39, no. 1, pp. 88–114, February 2011.

[8] ——, "Building Correlators with Many-Core Hardware," *IEEE Signal Processing Magazine*, vol. 27, no. 2, pp. 108–117, March 2010.

[9] M. Clark, P. Plante, and L. Greenhill, "Accelerating Radio Astronomy Cross-Correlation with Graphics Processing Units," *Int. Jour. of High Performance Computing Applications*, vol. 27, no. 2, pp. 178–192, 2013.

[10] L. Fiorin, E. Vermij, J. Lunteren, R. Jongerius, and C. Hagleitner, "An Energy-Efficient Custom Architecture for the SKA1-low Central Signal Processor," in *CF'15*, Ischia, Italy, May 2015, pp. 5:1–5:8.

[11] A. Sclocco, H. Bal, J. Hessels, J. Leeuwen, and R. van Nieuwpoort, "Auto-Tuning Dedispersion for Many-Core Accelerators," in *IPDPS'14*, Phoenix, AZ, May 2014, pp. 952–961.

[12] G. Teodoro, T. Kurc, J. Kong, L. Cooper, and J. Saltz, "Comparative Performance Analysis of Intel (R) Xeon Phi (TM), GPU, and CPU: A Case Study from Microscopy Image Analysis," in *IPDPS'14*, Phoenix, AZ, May 2014, pp. 1063–1072.

[13] T. Tran, Y. Liu, and B. Schmidt, "Bit-parallel Approximate Pattern Matching: Kepler GPU versus Xeon Phi," *Parallel Computing*, vol. 54, pp. 128–138, November 2015.

[14] H. Bal et al., "A Medium-Scale Distributed System for Computer Science Research: Infrastructure for the Long Term," *IEEE Computer*, vol. 49, no. 5, pp. 54–63, May 2016.