PERFECT Case Studies Demonstrating Order of Magnitude Reduction in Power Consumption

David K. Wittenberg*, Edin Kadric‡, André DeHon‡, Jonathan Edwards*, Jeffrey Smith*, and Silviu Chiricescu*
*BAE Systems
‡University of Pennsylvania

***Abstract*—We propose three methods for reducing power consumption in high-performance FPGAs (field programmable gate arrays). We show that by using continuous hierarchy memory, lightweight checks, and lower chip voltage for near-threshold voltage computation, we can both reduce power consumption and increase reliability without a decrease in throughput.**

We have implemented these techniques in two different, realistic wide-area motion imagery algorithms on FPGAs. We demonstrated greatly improved performance/efficiency compared to two flight-tested platforms, getting up to a 250X reduction in power use (measured in giga operations per second per watt). This paper summarizes these two case studies.

I. INTRODUCTION

As part of DARPA's (Defense Advanced Research Projects Agency) PERFECT (Power Efficiency Revolution For Embedded Computing Technology) Program, we developed the LEARN (Low-Energy Architecture of Reconfigurable Nodes) architecture, which represents a reconfigurable hardware approach. To highlight its capabilities, we have used a wide-area motion imagery (WAMI) application as a test. WAMI systems have to fly on power-constrained aircraft, with limited bandwidth to the ground, requiring most of the computation to take place on the aircraft. This implies a requirement to reduce power consumption as much as possible.

WAMI algorithms require a representative mixture of processing techniques and kernels: it includes linear algebra and image arithmetic operations. This makes it a good test case for systems with high vector/matrix computational requirements.

A. Advances

We introduce three innovations which together result in lower power use without a loss of throughput or reliability. They are introduced here, and more fully described in Section III.

1) Continuous Hierarchy Memory (CHM): By splitting the memory blocks into multiple banks, it is possible to reduce energy consumption by choosing the optimal memory size for an application. Memory block size is chosen at fabrication time. Further optimization is possible at fabrication time by choosing the size of the memory blocks. The optimal size of the memory blocks is application dependent.

2) Lightweight Checks (LWC): For many problems, it is much easier to check an answer than to find it in the first place. For example sorting takes time $n \log n$, but can be checked in linear time. We use LWC to verify the computations and repeat them if a failure is detected. As a result, this process makes up for reliability lost when we calculate with low votage (V_{dd}).

3) Lower Voltage: By running the chip at a near-threshold voltage, we reduce the power consumption of the chip while paying a reliability and throughput penalty. However, we recover the reliability loss (by employing LWC) and restore the original throughput (by increasing the parallelism of the design) while maintaining most of the (substantial) power savings advantage.

If the platform has additional resources, the power savings could be used to move more logic onto the platform while maintaining a fixed power envelope. For example, on one of the case studies, the WAMI processing was only a small part of a larger computation mapped across multiple computing nodes. By lowering the power required by the WAMI computation we could do additional computations on one FPGA, and in the process remove the need for some interfaces and auxiliary circuitry, resulting in additional power savings.

B. Two Case Studies

We demonstrate these advances on two related WAMI systems: ARGUS-IS [1] (Autonomous Realtime Ground-Ubiquitous Surveillance-Imaging System) and TAILWIND (Tactical Aircraft to Increase Long-Wave Infrared Nighttime Detection) [2]. ARGUS is an advanced camera system that uses hundreds of cellphone cameras to record video and autotrack moving objects. ARGUS was chosen as a common WAMI system, across all DARPA PERFECT performers, where power reduction was mission critical. These technologies were also applied to TAILWIND, as a second case study, as it represented the next-generation WAMI processing pipeline that goes beyond simple detection and tracking of vehicles and pedestrians, and is capable of performing 3-D scene reconstruction, working in the Infrared (IR) spectrum. The 3-D scene structure was postulated to improve detection of small targets (e.g. pedestrians) occupying very few pixels. By understanding 3-D structure, nuisance false-alarm sources (e.g. those caused by parallax) could be mitigated in ways not easily accessible in traditional

WAMI processing pipelines lacking this information. TAILWIND was designed to fly at lower altitude than ARGUS to concentrate more on tracking pedestrians.

The rest of the paper is organized as follows: Section II discusses ARGUS and TAILWIND in more detail, as well as describing the kernels which make up both systems. Section III describes the novel techniques we developed for lowering FPGA power consumption. Section IV shows the results of our improvements and Section V concludes.

II. Case Studies

In this section we describe three implementations of a WAMI pipeline. Two of the implementations have been flight tested and the third one is based on our LEARN architecture and compared against the other two. WAMI systems are designed for use on unmanned aerial vehicles (UAVs) and long-flight duration airships, so the need for power efficiency cannot be overstated. These reconnaissance platforms are designed to loiter, and contain other payloads, resulting in extremely limited available power. Ground users are generally able to communicate with such platforms through a constrained wireless downlink, but there is also a premium on this bandwidth. Ultimately, this limits the ability to process large amounts of data on the ground in real time. For that reason, significant processing must occur on the platform and the system is highly power constrained.

The first implementation targets a combination of CPU/GPU platform derived from the original ARGUS architecture. Specifically for testing purposes, we have used a Tesla K40 (with 6 GHz memory clock, 384 b memory width, 235 W power consumption and 4.29 TFLOPS single precision). The second implementation maps the advanced wide-area image processing algorithms in TAILWIND onto a typical low-power x86 CPU (Intel Core i5-4250U processor) platform fabricated in 22 nm. The third implementation (PERFECT WAMI) was developed during the DARPA PERFECT Program [3] and can be targeted to a custom-defined differentially reliable LEARN architecture.

A. Kernels

The PERFECT WAMI application software was adapted from the original reference implementations of three algorithm kernels created by the BAE Systems PERFECT team, namely:

- Debayer (for image decolorization) – Kernel for converting a 1-channel Bayer color filter pattern image into a 3-channel RGB image by interpolating pixels in a 5 x 5 pixel window.
- Lucas-Kanade (LK for image registration) – Kernel for image alignment between successive frames of a moving image. The LK algorithm is used as an iterative solver for the affine warp parameters that relate successive images.
- Gaussian Mixture Models (GMM for image change detection) – Kernel for separating background objects from foreground or moving objects in a video stream. GMM models the probability that a given pixel is in the foreground using multiple Gaussian probability models for each pixel.

The 3 kernels were designed using Bluespec SystemVerilog[1] and mapped onto the LEARN architecture using a customized set of tools based on VTR [4].

Table I shows key differences between ARGUS and TAILWIND.

ARGUS	**TAILWIND**
Visual spectrum	Infrared spectrum
Debayer	No Debayer, as IR doesn't need it
Registration - LK	Registration SLAM (FAST 9)
H.264 Frames	PNGs
Multiple GMMs	Multiple GMMs, RFD

TABLE I
ARGUS vs. TAILWIND differences

For image registration, ARGUS utilizes the Lucas-Kanade (LK) algorithm [5], while TAILWIND utilizes FastSLAM [6], a modern simultaneous localization and mapping (SLAM) algorithm [7].

The SLAM algorithm consists of three steps. In the first step, a set of feature points is extracted from the image using the Rosten and Drummond [8], [9] algorithm. The second step determines the correspondence of the feature points across images using a RANdom SAmple Consensus (RANSAC) algorithm [10], which has been proven to be robust to outlier corresponding points. The last step estimates the camera rigid body motion between two consecutive frames as an 8 parameter homographic transform, using the Levenberg-Marquardt algorithm [11].

The SLAM approach is particularly relevant in an unknown 3-D environment such as that addressed on TAILWIND. It is a natural framework within which: 1) perspective homographies can be recovered, or 2) 3-D scene structure can be incorporated.

For detection, ARGUS uses a traditional Gaussian mixture model (GMM) algorithm (e.g. Stauffer-Grimson GMM) [12]. The observed intensity of each pixel is modeled by K mixtures of Gaussian distributions (each one with its own mean, standard deviation, and weight). The algorithm has three basic steps: (i) the intensity of each pixel is matched (based on a threshold) against all the Gaussian distributions; (ii) based on the results from the previous step, the parameters of the matched Gaussian distribution (if there is one) or that of the least likely distribution (if none matched) are updated; and (iii) the number of background distributions is updated and the pixel is classified as background or foreground.

The GMM algorithm works well for images produced by stationary cameras when the lighting changes slowly. However,

[1] Bluespec SystemVerilog is a high-level, Haskell-extended, functional hardware description programming language used to accelerate reliable chip design and electronic design automation.

if used on images produced by moving platforms (such as UAVs), the algorithm produces high rates of false alarms, especially in regions with high contrast (e.g. a crosswalk) due to image alignment errors, parallax handling, and post-transform image interpolation. To deal with these challenges, TAILWIND provides three improved versions of GMM: GMM + edge suppression, interval-based GMM, and robust frame differences (RFD) [13]. The edge suppression lowers the false-alarm rate, but can introduce fragments in detection. The interval-based GMM extends the search of matching GMM to surrounding warped pixel location to deal with registration drift. The RFD applies the main idea in interval-based GMM to the traditional frame differences algorithm.

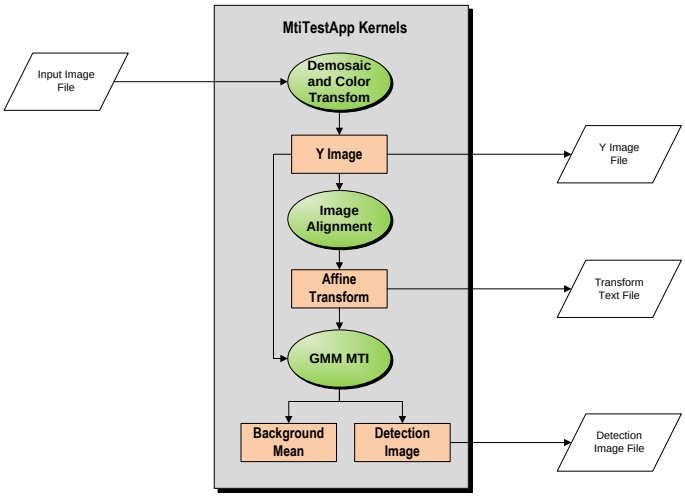

Fig. 1. The ARGUS wide-area imaging processing pipeline test application

B. ARGUS

BAE Systems and DARPA developed the ARGUS-IS sensor (Fig. 1) and the first- and second-generation Airborne Processing System (APS) [1]. The ARGUS-IS sensor is a 1.85 GPixel, Bayer pattern, motion imaging system consisting of 368 individual, 5 MPixel CMOS image sensors, with a maximum frame rate of 10 Hz. The systems were developed to support several military objectives for wide-area persistent overwatch, including tracking of pedestrians and vehicles with an operational concept emphasizing wide-area persistent surveillance of city-sized ground areas. A typical use case would involve simultaneously: 1) creating and downlinking many simultaneous geostationary area or object (vehicle and pedestrian) tracking video feeds, 2) lossy compression and storage of the full-scene image stream, and 3) full-scene vision processing for motion detection or tracking of vehicles.

C. TAILWIND

TAILWIND is the next-generation WAMI processing pipeline that goes beyond simple tracking of vehicles and pedestrians.

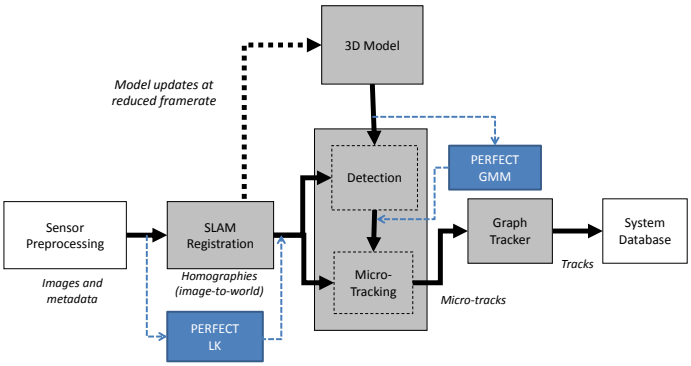

Fig. 2. The TAILWIND image processing subsystem instrumented with PERFECT kernels (in blue).

Our motivation for adding the TAILWIND case is that TAILWIND: 1) is capable of performing 3-D (actually multiple 2-D planes) reconstruction of a scene from IR data for improved clutter removal, 2) is a locally modifiable, closest matching, WAMI program that actually flew on real avionic platforms, 3) has a framework that incorporates elements from tactical, small-SWaP and exploitation systems, 4) has a variety of WAMI datasets with associated ground truth data for electro-optic and infrared sensors needed for measurement, 5) has algorithm metrics for functional verification (e.g. probability of detection and false alarm rate) and 6) has performance metrics for comparison to conventional architectures (CPU usage, memory, run time) - See Table II.

TAILWIND code runs on an low-power x86 platform and is optimized for performance with minimal consideration given to energy consumption. To test the applicability of the PERFECT technologies to the TAILWIND system, we have substituted TAILWIND's image registration and change detection kernels with functionally equivalent kernels from the ARGUS/TAV suite (see Fig. 2) and evaluated the system-level performance.

III. Advances

We introduced the LEARN architecture, an FPGA architecture that supports operation at near threshold voltages with customized banked internal memories [14]. When error checking is performed with LWC [15], a net energy savings can be realized with no loss of reliability. The LEARN architecture is characterized by:

- Very low energy consumption
- Low system clock rates
- Increased silicon area to accommodate application parallelism to compensate for slow clocks with no decrease in throughput.

A. Continuous Hierarchy Memory

CHM was introduced in [16], and better described in [14]. Because the energy cost of addressing memory grows as the

Data Set	System Level Metric	SLAM + RFD	LK + RFD	SLAM + GMM
Quantico Scene 1	P(vehicle detection)	0.9	0.53	0.5
	P(dismout detection)	0.64	0.38	0.46
	False Alarm/minute	1.35	103	163
	P(tracking vehicle)	0.3	0.1	0
	P(tracking dismount)	0.33	0.14	0.14
Quantico Scene 2	P(vehicle detection)	0.82	0.89	0.3
	P(dismout detection)	0.52	0.71	0.25
	False Alarm/minute	8.9	149	124
	P(tracking—vehicle)	0.42	0.25	0
	P(tracking—dismount)	0.44	0.28	0.1
Quantico Scene 3	P(vehicle detection)	0.85	0.54	0.24
	P(dismout detection)	0.6	0.42	0.19
	False Alarm/minute	4.1	117	126
	P(tracking—vehicle)	0.15	0.13	0.03
	P(tracking—dismount)	0.21	0.09	0.27

TABLE II
TAILWIND SYSTEM-LEVEL EVALUATION METRICS FOR DIFFERENT KERNELS

square root of the size of the memory block, excessively large memories increase the energy cost. Excessively small memories have increased energy costs because of the longer wires necessary to reach the extra memory blocks. CHM minimizes the energy cost by dividing memory blocks into several smaller blocks, allowing the designer to use an optimal number of memory blocks. One can further optimize power consumption by choosing the appropriate-size memory blocks when the FPGA is made.

B. Lightweight Checks

In many applications, particularly in signal processing, we can tolerate increased error rate while still maintaining proper operation. These are applications that already tolerate a level of signal-to-noise ratio (SNR), such as compression, object matching, and feature detection. Therefore, we often have a margin to decrease voltage and SNR, yet maintain an acceptable signal, image or sound quality, even with more errors at the output.

Some kernels also have inherent fault mitigation mechanisms. For example, consider using GMM [17] to separate foreground objects from the background. The algorithm itself is inherently noisy. As a result, the downstream processing typically performs morphological operations to tolerate isolated pixels that are misidentified [12]. As long as the errors introduced due to transient circuit failures are small compared to the noise inherent in the algorithm, the downstream operations can easily tolerate them along with the algorithm noise. We tolerate an isolated foreground or background pixel that would be removed by the morphological operations.

Other kernels require correct results and do not tolerate errors introduced by circuit failures. For those kernels, we use LWC to detect failures in the computation and re-execute the computations. A detailed classification of LWC classes is presented in [18]. Here, we present only two classes:

1) Operations with Checksums: Many signal, image processing, and scientific computing tasks are based on linear weighted sums. It is often possible to identify sums that remain invariant between the input data and the output data, or at least change in easily predictable ways. These sums serve as an LWC on the operation.

2) Convergent Algorithms: Iterative convergent algorithms are an important class of kernels in both image processing [5] and scientific computing [19]. They often have the useful property that an LWC is built into the algorithm—the convergence acceptance test. Assuming the the acceptance check is reliable, it guarantees that no result is produced until it is correct. In many cases, the algorithm will self-correct if errors occur in an iterative improvement computation. This means, for correctness, we only need to focus on the convergence test regardless of the iterative improvement computation. If neither fault mitigation or built-in LWCs apply, we have to engineer custom LWCs to add to the algorithm.

IV. RESULTS

For the WAMI pipeline of ARGUS and TAILWIND, we have employed the profiling and power-estimation tools provided by the manufacturers of plaforms onto which the application was mapped. For the WAMI pipeline mapped onto the LEARN architecture, we have provided detailed simulations and energy estimations using a set of tools developed (and validated) during the PERFECT program.

A. ARGUS

1) Measurement Methodology: Power and calculation operations were profiled on a 3000-image, 5 MPixel (2592 x 1944 pixels with 8-bit/color RGB per pixel) focal plane array (FPA) representative test data set consisting of 3.3 Hz ARGUS-IS imagery representing an observation duration of 16.7 seconds (four FPAs were repeated fifteen times to approximately match the 60-FPA, node-level load of the fielded system).

NVML API [20] and nvprof [21] sampling profilers from NVIDIA were used to obtain power and timing estimates,

from which energy estimates were derived. Pure calculation algorithm timing estimates were obtained by subtracting PCIe device-transfer and CUDA overhead time. The profiler also specified calculation-operation and memory-transaction counts.

2) Results: The ARGUS WAMI application shown in Fig. 1 mapped onto a Tesla K40 platform has achieved a total of $940 * 10^9$ operations and 17 giga-transfers (i.e. 384-bit global memory read/write interactions) while maintaining a constant power draw (across all kernels) of 77.4 W for 3000 FPAs in 6.5 seconds of wall-clock processing time. This suggested per-pixel values of: 33.5 nJ, 62 calculation operations, and 18 B of memory access. ARGUS achieves 1.86 GOPS/W (giga operations per second per watt).

B. TAILWIND

1) Measurement Methodology: The TAILWIND WAMI pipeline (shown in Fig. 2) was compiled and run on a fourth-generation ultra-low power Intel Core i5-4250U processor fabricated on a 22 nm node and running at 1.3 GHz. The Intel performance analyzer, Intel®$VTune^{TM}$ Amplifier 2015 [22], was used to implement extensive energy *measurements* of the code. Since the performance analyzer samples the performance counters present inside the Haswell microarchitecture, for each input data set, we have conducted 11 runs and averaged the results (e.g. energy, time) across all the runs. Each input data set consists of 500 frames at 2048 x 1536 pixels.

2) Results: Here we present the energy and latency of only the image registration and change-detection blocks from Fig. 2. Although we have the overall performance numbers for the entire pipeline, we present in Figure 3 only the energy per frame and latency per frame for the image registration and change detection only (see the columns with green background). This is done because: (i) we want to assess the applicability of the techniques discussed in Section I-A to the above-mentioned kernels; and (ii) together, these kernels account for a significant percentage (i.e. 21+6.6=27.6%) of the overall energy of the TAILWIND system.

	TAILWIND SYSTEM		*PERFECT SYSTEM (1PE/kernel)*		
	Image Registration (SLAM)	Motion Detection (RFD)	Image Registration (LK)	Motion Detection (GMM)	
Energy per Frame(mJ)/ % of total Energy	752/21.00%	237/6.61%	175/129	19/14	Energy per Frame(mJ) STD/CHM
Std. Dev. Energy	0.015	0.026			
Time per Frame (ms)	253.5	82.5	444.04	37.3	Time per Frame (ms)
Std. Dev. Time	2.962	8.292			

Fig. 3. TAILWIND vs PERFECT kernels

C. LEARN

1) Measurement Methodology: A WAMI application similar to the one presented in Fig. 1 has been mapped onto the LEARN architecture. Since we are able to vary any architectural parameter, we have conducted a sweep of the most important architectural parameters (i.e. memory size and organization, location of the memory within the FPGA fabric, type of hard resources available, operating voltage, algorithm parallelism) to arrive at an optimal LEARN architecture that meets the throughput requirement and minimizes the overall energy.

The entire WAMI application is represented as a parameterized Bluespec SystemVerilog design, which is mapped onto our architecture using an internally developed customized flow based in part on VTR [23]. To get accurate energy estimation, we use activity factors obtained by *simulating* the gate-level design processing the same inputs as the ones mentioned in Section IV-A1. This way, we know precisely how often any architectural element (i.e. look-up table, routing segment, memory access, etc.) is toggled. Since we share inputs with the ARGUS implementation of the WAMI pipeline, we can compare the energy required by the two implementations.

Our energy estimation also considers the operating voltage of every architectural element and the characteristics of the process technology. We operate the memory elements at nominal voltage and use CACTI [24] to model their energy consumption. We operate the remaining architectural elements at close to threshold voltage to save power.

2) Results: This section presents the energy consumption of both the full WAMI application and its constituent kernels. We have mentioned above that we have conducted a sweep of the architectural parameters to determine the architecture that minimizes the energy. It comes as no surprise that the architectures that minimize the energy consumption of each WAMI kernel (while maintaining throughput requirements) have different parameters. Specifically, they have different memory block sizes. For example, the optimal energy for GMM and DeBayer kernels is achieved for an architecture with a memory block size of 128Kb x 64, and 32Kb x 64, respectively.

To assess the improvements afforded by the techniques described in Section I-A on a flight-tested system, we replace the image-registration and change-detection kernels in the TAILWIND system with "functionally" equivalent kernels used with the LEARN architecture (see Fig. 2) and evaluate the resulting system-level performance. The blue background columns in Figure 3 show the energy consumption of the above-mentioned kernels. There are two energy numbers for each of the kernels – one for an architecture with "regular" memory blocks (STD), and one for an architecture employing the CHM memory block. In both cases, the size of the memory block is kept constant at 64 Kb x 64.

Figure 3 shows that the "PERFECT kernels" (i.e. LK and GMM) consume only a small fraction of the energy consumed by the functionally equivalent "TAILWIND kernels" (i.e. SLAM and RFD): 129/752≈17% and 14/237≈6% for

the image registration and change detection respectively. This is due to multiple factors such as: running frequency of the kernels, spatial distribution of the computations in the "PERFECT kernels" and memory access optimization. In addition, the "PERFECT kernels" could further benefit from having the computations run at near-threshold voltage (e.g. 0.6 V) resulting in additional energy savings.

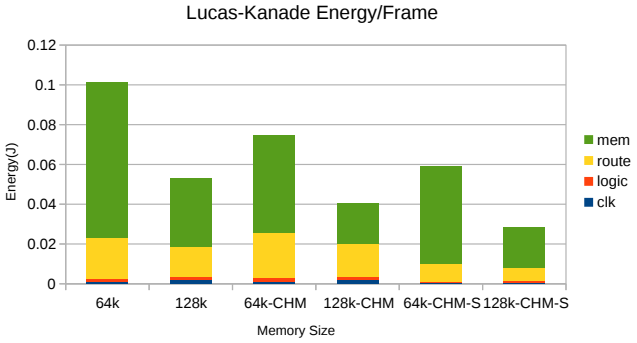

Fig. 4. LK Energy/frame using one PE

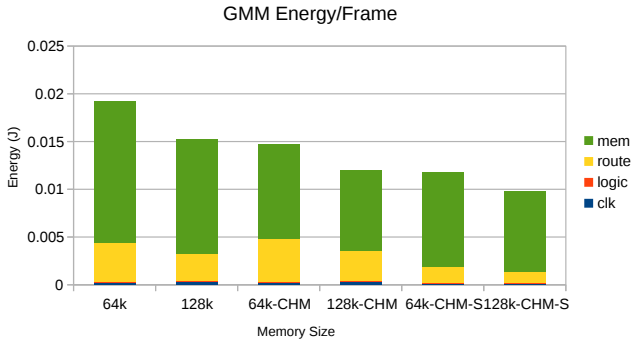

Fig. 5. GMM Energy/frame using one PE

Figures 4 and 5 show the energy per frame for the LK and GMM kernels for two different architectures (with memory blocks of size 64Kb x 64 and 128 Kb x 64) along with the energy contribution breakdown – using a frame size of 1024 x 1024. As expected, the memory and route energy dominate the energy expenditure. Taking as basis an architecture with a single processing element and standard (non-banked) memory blocks of size 128 Kb x 64, the GMM and LK kernels achieve an energy savings of 21.5% and 24% respectively, for simply employing CHM (e.g. banking existing memory - see the "128k-CHM" bars); an additional 14.1% and 23% savings is achieved by scaling the voltage down from the nominal 0.95 V to the near-threshold value (see the "128k-CHM-S" bars). Scaling the voltage from the nominal value to the near-threshold value comes at the expense of increasing the delay by a factor of 29. To make up for the significant decrease in throughput, one can increase the parallelism by adding more processing elements, but that in turn increases the area required by the circuit. A more appropriate choice is to scale the voltage down to 0.75 V which in turn would increase the delay by a factor of $\approx$ 3.5; this throughput loss can be easily made up by quadrupling the number of processing elements for a kernel like GMM. Increasing the number of PEs also reduces the energy consumption (due to needing less memory per PE) up to a certain point after which the energy increases due to interconnect energy [14].

To determine the parameters of the LEARN architecture that minimize the energy of the entire WAMI application, we have conducted preliminary studies with small input image sets (i.e. input frames of size 256x256, 512x512, 1024x1024). These experiments allowed us to determine that the architecture that minimizes the overall energy consumption of the WAMI application has a block memory size of 64Kb x 64, with memory block columns placed every 20 logic block columns. We then scaled up the simulation to the 5 Mpixel FPA size (same input size as the experiments in Section IV-A) and 3000 frame data set. Multiple V_{dd} values and technology nodes were simulated. In the optimal configuration of 64 Kb x 64/memory block, using a 7 nm technology node and a V_{DD}= 0.4, the simulation was capable of executing $4 * 10^{12}$ operations in 17.1 s using 0.5 W, thus providing 467 GOPS/W. This energy efficiency result compares favorably with the similar result reported in Section IV-A. We used the Intel VTune Amplifier profiler. We didn't include the GOPS/W measurement metric, in this case, since VTune did not provide it. The energy efficiency improvement of 467/1.86$\approx$250x is partially due to the difference in the technology nodes: 28 nm for the ARGUS vs 7 nm for the LEARN; even when accounting for additional, non-trivial parallelism performance gains in the ARGUS WAMI implementation, our approach suggest an order-of-magnitude gain in energy efficiency over what is likely achievable in the future 7 nm technology.

V. Conclusion

Successful application of the LEARN architecture to WAMI permits the employment of larger sensors or increases in capability on the same platform, enabling exploitation techniques that have previously been inconceivable with conventional technology or longer missions. We have shown that optimizing FPGAs can produce great savings in power consumption, apart from the power reductions that decreasing feature size will provide. For the cost of extra silicon area, we show that reliability can be increased and power consumption decreased with no decrease in throughput.

VI. Acknowledgments

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

REFERENCES

[1] B. Leininger, J. Edwards, J. Antoniades, D. Chester, D. Haas, E. Liu, M. Stevens, C. Gershfield, M. Braun, J. D. Targove, S. Wein, P. Brewer, D. G. Madden, and K. H. Shafique, "Autonomous real-time ground ubiquitous surveillance-imaging system (ARGUS-IS)," vol. 6981, 2008, pp. 69 810H–69 810H–11. [Online]. Available: http://dx.doi.org/10.1117/12.784724

[2] J. Keller, "BAE Systems to develop wide-area motion-analysis sensors to detect and track large numbers of ground forces," June 2011. [Online]. Available: http://www.militaryaerospace.com/articles/2011/06/bae-systems-to-develop.html

[3] BAE Systems, University of Pennsylvania, Princeton University, and Brown University, "PRACTICE: power-reducing adaptive computing technologies: Intelligent, cross-layer, and efficient," in *Response to DARPA-BAA-12-24 (PERFECT)*, April 2012.

[4] J. Rose, J. Luu, C. W. Yu, O. Densmore, J. Goeders, A. Somerville, K. B. Kent, P. Jamieson, and J. Anderson, "The VTR project: architecture and CAD for FPGAs from verilog to routing," in *Proceedings of the International Symposium on Field-Programmable Gate Arrays*. New York, NY, USA: ACM, 2012, pp. 77–86. [Online]. Available: http://doi.acm.org/10.1145/2145694.2145708

[5] S. Baker and I. Matthews, "Lucas-Kanade 20 years on: A unifying framework," *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221–255, Febuary 2004.

[6] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *AAAI-02*, July 2002, pp. 593–598.

[7] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *International Journal of Robotics Research*, vol. 5, no. 4, 1986. [Online]. Available: http://www.frc.ri.cmu.edu/~hpm/project.archive/reference.file/Smith&Cheeseman.pdf

[8] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," in *IEEE International Conference on Computer Vision*, vol. 2, October 2005, pp. 1508–1511. [Online]. Available: http://www.coxphysics.com/work/rosten_2005_tracking.pdf

[9] ——, "Machine learning for high-speed corner detection," in *European Conference on Computer Vision*, vol. 1, May 2006, pp. 430–443. [Online]. Available: http://www.coxphysics.com/work/rosten_2006_machine.pdf

[10] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *CACM*, vol. 24, no. 6, pp. 381–395, June 1981. [Online]. Available: http://www.dtic.mil/dtic/tr/fulltext/u2/a460585.pdf

[11] D. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *SIAM Journal on Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[12] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 2. Los Alamitos, CA, USA: IEEE, Aug. 1999, pp. 246–252 Vol. 2. [Online]. Available: http://dx.doi.org/10.1109/cvpr.1999.784637

[13] T. Pollard and M. Antone, "Detecting and tracking all moving objects in wide-area aerial video," in *Computer Vision and Pattern Recognition Workshops*, 2012, pp. 15–22.

[14] E. Kadric, D. Lakata, and A. DeHon, "Impact of Memory Architecture on FPGA Energy Consumption," in *Proceedings of the International Symposium on Field-Programmable Gate Arrays*, 2015, pp. 146–155.

[15] E. Kadric, K. Mahajan, and A. DeHon, "Energy reduction through differential reliability and lightweight checking," in *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines*, 2014.

[16] ——, "Kung Fu data energy—minimizing communication energy in FPGA computations," in *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines*, 2014.

[17] M. Genovese and E. Napoli, "ASIC and FPGA implementation of the gaussian mixture model algorithm for real-time segmentation of high definition video," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 3, pp. 537–547, March 2014.

[18] E. Kadric, "Energy reduction through voltage scaling and lightweight checking," Ph.D. dissertation, University of Pennyslvania, 2016.

[19] Y. Saad, *Iterative Methods for Sparse, Linear Systems*, 2nd ed. SIAM, 2003.

[20] NVIDIA, "NVIDIA management library (NVML)." [Online]. Available: https://developer.nvidia.com/nvidia-management-library-nvml

[21] ——, "CUDA toolkit profiler user's guide," 2015. [Online]. Available: http://docs.nvidia.com/cuda/profiler-users-guide/#axzz44PXT7f9M

[22] "Intel VTune amplifier 2015," 2015. [Online]. Available: https://software.intel.com/en-us/intel-vtune-amplifier-xe

[23] J. Luun, V. Betz, T. Campbell, W. M. Fang, P. Jamieson, I. Kuon, A. Marquardt, A. Ye, and J. S. Rose, "VPR user's manual," 2012. [Online]. Available: http://vtr-verilog-to-routing.googlecode.com/files/VPR_User_Manual_6.0.pdf

[24] P. Shivakumar and N. P. Jouppi, "Cacti 3.0: An integrated cache timing, power, and area model," Technical Report 2001/2, Compaq Computer Corporation, Tech. Rep., 2001. [Online]. Available: http://www.cs.utexas.edu/~cart/publications/cacti3.pdf