Energy Performance of FPGAs on PERFECT Suite Kernels

Sanmukh R. Kuppannagari, Ren Chen, Andrea Sanny, Shreyas G. Singapura, Geoffrey Phi C. Tran, Shijie Zhou, Yusong Hu,
†Stephen P. Crago, Viktor K. Prasanna
Ming Hsieh Department of Electrical Engineering, †Information Sciences Institute
University of Southern California
Los Angeles, USA 90007
Email: {kuppanna, renchen, sanny, singapur, geoffret, shijiezh, ysh_055, prasanna}@usc.edu, †crago@isi.edu

***Abstract*—Energy efficiency as a metric has gained significant importance in the recent years. The goal of the TAPAS project is to discover and develop revolutionary approaches that will generate energy-efficient designs on embedded computing systems. One suitable target platform for implementing energy-efficient algorithms is the FPGA, due to its low power consumption, high performance and efficiency, and high degree of programmability. We develop optimizations targeted at reducing memory and communication energy dissipation on FPGAs. We design parameterized architectures for six kernels selected from the PERFECT benchmark suite, applying our memory and communication optimizations. Our optimizations demonstrate 3x to 110x energy efficiency improvement compared to baseline implementations.**

I. Introduction

The Tunable Algorithms for PERFECT ArchitectureS (TAPAS) project is part of the DARPA PERFECT program [1] and is dedicated towards developing algorithmic innovations and optimizations for power efficiency in the emerging landscape of embedded computing platforms. We follow a design methodology which uses algorithm-architecture pairs to enable the efficient exploration of design spaces at both design time (static) and runtime (dynamic). We have developed several high-level optimizations targeted at reducing the memory and communication energy dissipation. We also created a high-level abstraction of an embedded computing system and defined our optimizations with this abstraction. FPGAs have garnered interest as an embedded computing platform [2] and [3]. As a result, we use the FPGA to implement six kernels from the DARPA PERFECT benchmark suite [4], applying our optimizations to achieve an improvement of 3x to 110x in energy efficiency when compared against the unoptimized designs on the same platform.

Section II details related work for each of the selected kernels. In Section III, we define the various memory and communication energy optimizations that we have developed. Section IV details the improvements in energy efficiency obtained for each of the kernels and we conclude our findings in Section V.

II. Related Works

We selected 2-D convolution, FFT, Histogram Equalization, Sorting, Discrete Wavelet Transform and Debayer from the DARPA PERFECT benchmark suite for this work. Extensive research work has been done to optimize each of these kernels, with a focus on various metrics such as performance, resource utilizations, throughput etc.

There are some existing works which discuss the energy efficiency of 2-D convolution on an FPGA. In [5], the spatial redundancy of the image is exploited to avoid the computation of higher bits of neighboring pixels. However, the optimization is only a point solution and the design space is not fully explored. The authors in [6] compare the energy efficiency of a GPU and FPGA, while their research focus is the productivity of an FPGA programming model. In [7], energy efficiency is improved by mapping the application on embedded memory blocks on a FPGA, though the architecture design is not discussed.

A low-power FFT was explored in [8] and [9] using a cache-based FFT algorithm and a delay-balanced pipeline architecture based on a split-radix algorithm. In [10], the authors use techniques such as clock gating and memory binding to achieve high energy efficiency. However, none of the above mentioned works focus on improving the energy efficiency of the FFT architecture.

The histogram equalization implementation on an FPGA, such as [11], achieves a real-time implementation with high processing speed. In [12], histogram statistics and equalization are computed in parallel to create a fast, simple and flexible hardware implementation. Both of these works, however, look only at small image sizes and do not focus on improving the energy efficiency.

Though sorting algorithm implementations on an FPGA have been studied thoroughly, most of the research has been directed towards improving throughput and latency. [13] uses bitonic sort to achieve low latency and high throughput. [14] focuses on a parallel recursive sorting algorithm on hardware. [15] focuses on a scalable sorting implementation to handle large data. [16] analyzes the power consumption of sorting networks on FPGAs; however, the work does not provide the individual power dissipation of memory, logic control unit, etc.

The VLSI implementations of Discrete Wavelet Transform (DWT) can be classified into two broad categories: lifting-based and convolution-based [17]. The lifting-based architecture is computationally less intensive and requires a smaller amount of memory [18]. Efforts have been made to reduce

This work has been funded by DARPA under the cooperative agreement number HR0011-12-2-0023.

978-1-4799-6233-4/14/$31.00

the critical path delay in lifting-based architectures in [19], [20]. Several works [21], [22] have also focused on reducing memory size. Parallelism in scanning has been explored in works such as [20], [23]. To the best of our knowledge, none of the works have focused on energy efficiency.

Debayer is the process by which a full-resolution image is built from a "Bayer" image. Research done in debayer implementation has primarily focused on maintaining the quality of image and improving the Signal to Noise Ratio (SNR) during operation. [24] focuses on improving the SNR. [25], [26] aim to reduce the cost in terms of computations and although [27] has addressed energy efficiency, the platform used for implementation was not FPGAs.

III. Memory and Communication Energy Optimizations

Memory and communication operations are the major sources of energy dissipation when a design is implemented on an FPGA. We identify various high level optimizations targeted at reducing memory and communication energy.

We only focus on the dynamic energy consumption of FPGA. We do not consider the static energy which is dissipated irrespective of whether a component is active or not. Our optimizations can be applied to FPGAs as well as to SoCs which consist of processor cores and accelerators along with the reconfigurable logic. These architectures can be interfaced with external memory such as a DDR3 for storing large amount of data.

We implement a *Memory Activation Scheduler* to switch on or switch off the memory blocks and *Address Generation Unit* to generate the memory addresses used to access memory in a given clock cycle.

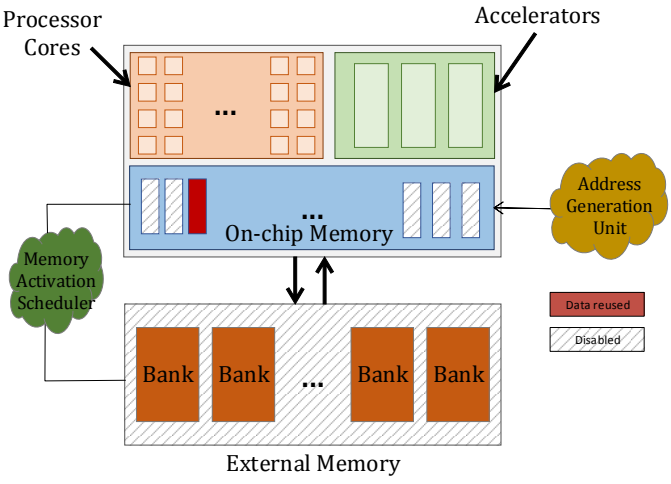

Fig. 1: Spatial Locality and Data Reuse Optimization: reusing local data and disabling unused memory

A. Spatial Locality and Data Reuse Optimization

In naive implementations, data is stored in memory without considering locality of accesses. Many algorithms exhibit some form of spatial locality in accessing the data which can be exploited to reduce the number of individual accesses to the memory. Buffers can be used to store the data that is fetched in bulk from the external memory and computation operations can be performed on the data in the subsequent clock cycles without a need to access the external memory. Figure 1 shows an architecture that implements these optimizations.

Consider an algorithm in which the computation operations require B bits in a single clock. If the available buffer size is $k \times B$, the algorithm will be able to consume the data stored in the buffer for k cycles. So, the data can be fetched from the external memory and stored in the buffer in one clock cycle. The algorithm can then access the data in k cycles without accessing the external memory. This allows us to turn off the external memory for $100(k-1)/k\%$ clock cycles. The energy efficiency improvement is determined by the following observation.

Observation 1: Energy consumed by memory can be reduced by a factor of $c \times k$, where k is the number of words stored in the buffers and c is an implementation constant.

B. Memory Activation Scheduling

On an FPGA, memory is composed of several blocks which can be turned on or off individually in any clock cycle. Assuming that the computation operations need to access input from only k blocks out of n available blocks in each clock cycle, we have the following observation:

Observation 2: Using a suitable memory activation schedule that controls the number of active memory blocks, the energy performance can be improved by $n/(n-k)$.

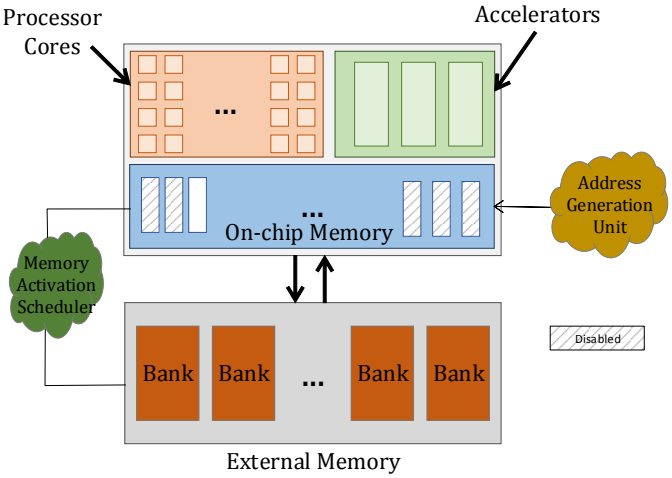

Fig. 2: Memory Activation Scheduling: switch off unused memory

Figure 2 shows an architecture that implements memory activation scheduling.

C. Inter-stage Communication

Authors in [28] develop a three-stage permutation block which is a mapping of a Clos network. A Clos network is a

multistage circuit-switching network, and a three-stage Clos network can realize all $2^n!$ permutations where 2^n is the size of input/output vector. The three-stage permutation block of [28] consists of two stages (Stage 0 and Stage 2) performing permutation in space and one stage (Stage 1) performing permutation in time on streaming data. Since it is a mapping from the three-stage Clos network, it can also realize all $2^n!$ permutations.

Several applications, such as FFT, are executed in stages and the data generated in intermediate stages can be stored into buffers. Taking into account the access pattern of the data buffer, an efficient data remapping technique can be developed. Consider the memory access pattern in which the memory is written in-order sequentially and read out of order. A dual-ported memory is required to ensure reads and writes occur simultaneously at different addresses in the same cycle. To reduce energy consumption, we can remap the data when writing the data buffer so that each memory location in the data buffer is written with a new value once the old value is read out. This allows us to have a single-ported memory. As a dual-ported memory is typically implemented using two single port memories the reduction in memory energy is 50%.

D. External Memory Row Activation Schedule

For external memories, such as DRAM, the memory is separated into banks, with each bank separated further into rows [29]. To read from a row, the row must first be opened. However there is a cost associated with a row activation. Without a suitable layout, the kernel will have to activate different rows in consecutive clock cycles. This causes significant energy consumption due to the cost associated with row activation as well as the cost associated with an active row.

By using a suitable data layout, the number of row activations can be minimized and the page hit rate improved. A burst of data is defined as the data accessed in consecutive clock cycles. By storing a burst of data in contiguous locations, the number of row activations can be reduced.

Given a row with c columns and a burst length of b, we can have a maximum of $c/b - 1$ consecutive read commands to an open row in order to read every data element stored in that row, with the first read command opening the row.

Observation 3: By maximizing the number of consecutive read commands to an open row to $c/b - 1$, a page hit rate of $(c/b - 1)/(c/b)$ can be achieved, with only the first read requiring row activation for each c/b reads to a row.

IV. Kernel Results

A. Experimental Setup

The experiments were performed using the Xilinx Vivado 2013.4 development tools [30], and power dissipation was determined using the Vivado Power Analysis tool. The experiments for convolution, histogram equalization and discrete wavelet transform were simulated targeting the state-of-the-art Xilinx Virtex 7 XC7VX980 with -2L speed grade [31]. The simulation platform for 1-D FFT, 2-D FFT, sorting and debayer was the state-of-the-art Xilinx Artix 7 XC7A200TL with -2L speed grade. We define *energy efficiency* as the ratio of total number of operations performed to the total energy consumed and evaluate the kernels' implementations using this metric. The designs were verified by post place-and-route simulation, using randomly generated inputs and an average switching activity of 50%. We either used the VCD (value change dump) file or the SAIF (switching activity interchange format) as input to the Xilinx Vivado Power Analyzer to produce accurate power dissipation estimations. Entire DRAM energy consumption is considered in our energy efficiency calculations. The baseline architecture is the same architecture as the optimized one but without applying any of the optimizations mentioned in Section III.

B. 2-D Convolution

The input image is $M \times N$ pixels, which is convolved with a 9×9 Gaussian filter to produce an output image with the same number of rows and columns. Images of small (640×480), medium (1920×1080) and large (3840×2160) sizes are considered for implementation. The proposed architecture (Fig. 3) has the following properties:

- Spatial locality and data reuse: the partial results generated by the Row-Processing Unit (PU) are stored and transposed in the internal buffer for the column-wise convolution
- Activation scheduling of image memory blocks: Block RAMs (BRAM)
- Parallelism exploration: varying the number of Multipliers and Accumulators in Row-PU and Col-PU

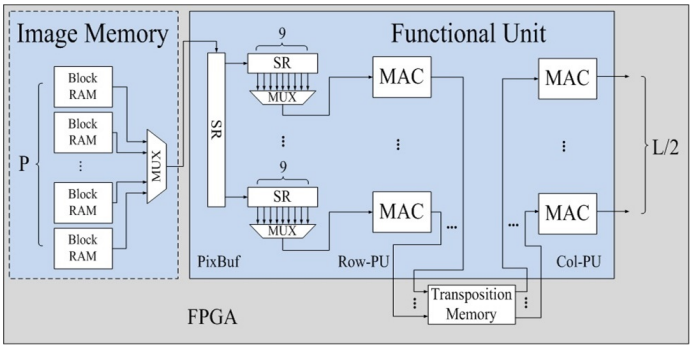

Fig. 3: Optimized Convolution Architecture

Experimental results for various input sizes are shown in Table I.

TABLE I: Energy Efficiency Comparison: Convolution

	Baseline	Optimized Architecture	
Input	GFLOPS/W	GFLOPS/W	Improvement
Small	0.28	6.67	23.11x
Medium	0.47	9.66	19.63x
Large	0.46	8.82	18.05x

C. Fast Fourier Transform (FFT)

For 1-D FFT, there are N input and output points. The sizes of N we used are 256, 4096 and 32768 (small, medium,

large). The 2-D FFT is performed by using a row-wise 1-D FFT kernel followed by a column-wise 1-D FFT kernel. For a $N \times N$-point 2-D FFT, the sizes of N are 1024, 4096 and 8192. The N input/output points are represented in single-precision floating point numbers. The architecture of the 1-D FFT shown in Fig. 4 is composed of components such as Radix Block, Data Path Permutation Unit (DAP), Twiddle Factor Computation Unit (TWC), and Parallel-to-Serial/Serial-to-Parallel MUX. The optimized 2-D FFT architecture is shown in Fig. 5. The various optimizations in the proposed architecture are:

- Memory activation scheduling: BRAMs (TWC)
- Energy efficient data remapping: BRAMs, DRAM
- Spatial locality and data reuse: BRAMs

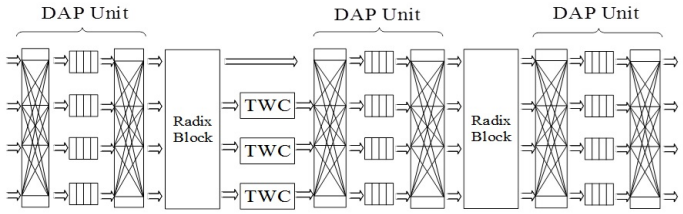

Fig. 4: Optimized 1-D FFT architecture

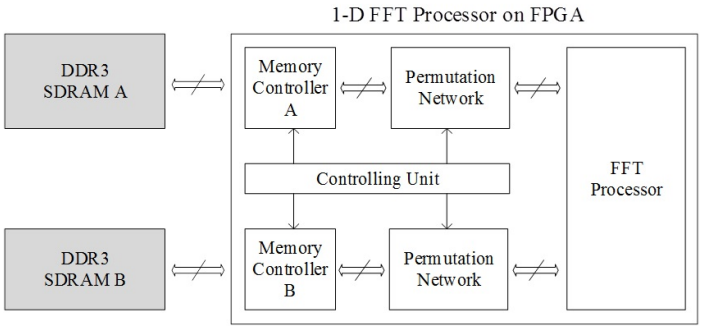

Fig. 5: Optimized 2-D FFT architecture

The performance comparison between the baseline and optimized architectures is tabulated in Table II and Table III.

TABLE II: Energy Efficiency Comparison: 1-D FFT

	Baseline	Optimized Architecture	
Input	GFLOPS/W	GFLOPS/W	Improvement
Small	5.862	20.657	3.52x
Medium	5.432	17.905	3.29x
Large	4.641	15.099	3.25x

D. Histogram Equalization

There are three steps involved in performing histogram equalization. First, a histogram is computed, consisting of the probability distribution function of the pixel values. The next step is to calculate the cumulative probability distribution function (CDF) on the histogram. Finally, the CDF is used to scale each input image pixel to produce the output image. To maintain a realistic throughput, our architecture has a minimum frame rate of 30 frames per second. There are three input image sizes: small (640×480), medium (1920×1080) and large (3840×2160). The pixels are represented by unsigned 32 bit integers. The optimized architecture shown in Fig. 6 has the following features:

TABLE III: Energy Efficiency Comparison: 2-D FFT

	Baseline	Optimized Architecture	
Input	GFLOPS/W	GFLOPS/W	Improvement
Small	2.11	8.71	4.13x
Medium	1.91	8.31	4.35x
Large	1.80	8.34	4.63x

- Memory activation scheduling: BRAMs and DRAM
- Data layout to minimize DRAM row activation energy

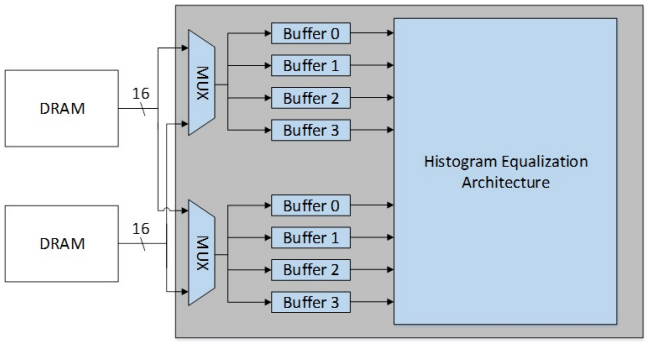

Fig. 6: Optimized Histogram Equalization Architecture

Experimental results for various input sizes are shown in Table IV.

TABLE IV: Energy Efficiency Comparison: Histogram Equalization

	Baseline	Optimized Architecture	
Input	GOPS/W	GOPS/W	Improvement
Small	0.72	8.97	12.42x
Medium	1.35	4.33	3.21x
Large	1.34	4.22	3.15x

E. Sorting

The input data is made up of small windows. Each element in the window consists of keys to be sorted and auxiliary data to be carried along. A set of 1024 windows is to be sorted. There are three window sizes: small (9), medium (64) and large (1024). The elements in each set are single-precision floating point numbers. The baseline architecture consists of on-chip BRAM, control logic, and a comparison unit. The optimized architecture depicted in Fig. 7 is composed of parameterized Distributed RAMs (Dist. RAM), local registers, a comparison unit and control logic. Optimizations employed are:

- Memory activation scheduling: BRAMs and DRAM
- Spatial locality and data reuse optimization: Distributed RAM is used for local data

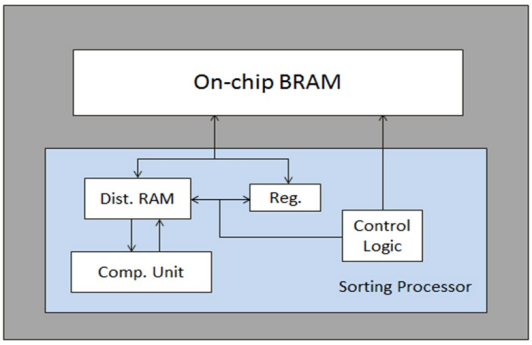

Fig. 7: Optimized Sorting Architecture

The experimental results and a comparison between the baseline and optimized architecture are tabulated in Table V. The improvement in the medium size input is much higher compared to the small and large size inputs. The reason is, in the case of the small and medium size inputs, $\frac{N-1}{N+3}$ comparison per cycle are performed and this ratio is higher for the medium size input. In the case of the large size input, concurrent read and write cannot be performed and hence the improvement is overshadowed by the energy consumed due to memory accesses.

TABLE V: Energy Efficiency Comparison: Sorting

	Baseline	Optimized Architecture	
Input	GOPS/W	GOPS/W	Improvement
Small	1.28	23.71	18.52x
Medium	0.23	25.34	110.17x
Large	0.37	5.2	14.06x

F. Discrete Wavelet Transform (DWT)

For DWT, the input image is $M \times N$ pixels. Each pixel is a unsigned 32-bit floating point number. There are three sizes for input image: small (640×480), medium (1920×1080) and large (3840×2160). The 2-D discrete wavelet transform is calculated by applying a 1-D DWT to the rows and then again to the columns. The process is repeated for 30 input frames. In the optimized architecture, the size of the vertical window scanned is R. Hence, the size of the temporal buffer used to store results between stages is reduced from N in the baseline architecture to R in the optimized architecture. This results in less memory usage and hence lower power dissipation. The optimized architecture is shown in Fig. 8. The results comparing the baseline architecture and the optimized architecture are shown in Table VI.

G. Debayer

An input image in a Bayer pattern of size $Y \times X$ is to be converted to a 3-D image of size $3 \times (Y-2) \times (X-2)$ using interpolation. In the output image, each pixel is defined by a triple of red, green and blue values. The input and

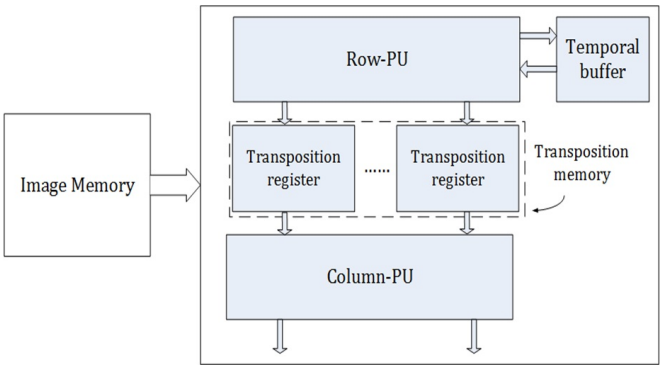

Fig. 8: Optimized DWT Architecture

TABLE VI: Energy Efficiency Comparison: DWT

	Baseline	Optimized Architecture	
Input	GFLOPS/W	GFLOPS/W	Improvement
Small	0.184	1.18	5.46x
Medium	0.176	2.64	14.04x
Large	0.176	2.54	13.46x

output images will consist of unsigned 16-bit integer pixel samples. The interpolation footprint is 5×5 centered on the pixel. There are three input image sizes: small (512×512), medium (1024×1024) and large (2048×2048). The baseline architecture consists of external DRAM, on-chip BRAMs and debayer processors. The optimized architecture shown in Fig. 9 contains a memory scheduler in addition to the aforementioned components. A minimum frame rate of 30 frames per second is maintained in both architectures. Optimizations used are:

- Memory activation scheduling: BRAMs and DRAM
- Spatial locality and data reuse: registers used for storing pixels in the current footprint and re-use these registers for the next footprint

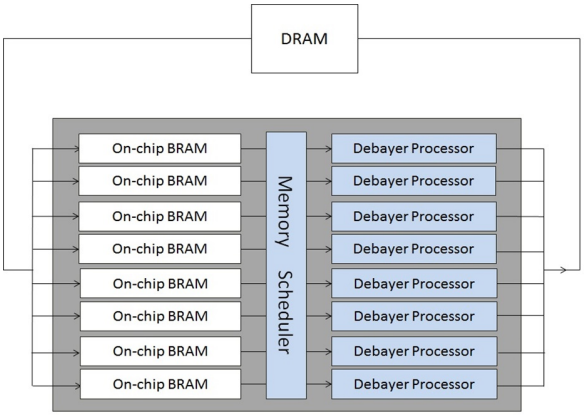

Fig. 9: Optimized Debayer Architecture

A performance comparison between the baseline and optimized architecture is shown in Table VII.

TABLE VII: Energy Efficiency Comparison: Debayer

	Baseline	Optimized Architecture	
Input	GOPS/W	GOPS/W	Improvement
Small	3.62	13.7	3.78x
Medium	3.6	13.8	3.83x
Large	3.6	13.5	3.75x

V. Conclusion

In this paper, we detailed the various optimizations that were developed focusing on reducing memory and communication energy consumption on FPGAs. The benefits obtained by the optimizations were empirically analyzed by implementing them on kernels from the PERFECT benchmark suite resulting in an improvement of 3x to 110x in energy efficiency. The energy efficiency achieved by the kernel implementations was as high as 25 GFLOPS/W as in the case of the sorting kernel, which shows that FPGAs are a suitable candidate for low power embedded computing.

References

[1] Power efficiency revolution for embedded computing technologies(perfect). DARPA. [Online]. Available: http://www.darpa.mil/Our_Work/MTO/Programs/

[2] W. J. MacLean, "An evaluation of the suitability of fpgas for embedded vision systems," in *Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*. IEEE, 2005, pp. 131–131.

[3] Y. Qu, Y. Yang, and V. K. Prasanna, "Large-scale multi-flow regular expression matching on fpga," in *High Performance Switching and Routing (HPSR), 2012 IEEE 13th International Conference on*. IEEE, 2012, pp. 70–75.

[4] DARPA, "The perfect suite manual," Tech. Rep.

[5] H. Ngo and V. Asari, "Neighborhood dependent approach for low power 2d convolution in video processing applications," in *Industrial Electronics and Applications, 2009. ICIEA 2009. 4th IEEE Conference on*. IEEE, 2009, pp. 656–661.

[6] Betkaoui, B. and Thomas, D.B. and Luk, W., "Comparing performance and energy efficiency of FPGAs and GPUs for high productivity computing," *Field-Programmable Technology (FPT), 2010 International Conference on*, pp. 94–101, 2010.

[7] Ghosh, A.; Paul, S.; Bhunia, S., "Energy-Efficient Application Mapping in FPGA through Computation in Embedded Memory Blocks," *VLSI Design (VLSID), 2012 25th International Conference on*, pp. 424–429, 2012.

[8] B. M. Baas, "A low-power, high-performance, 1024-point fft processor," *Solid-State Circuits, IEEE Journal of*, vol. 34, no. 3, pp. 380–387, 1999.

[9] W.-C. Yeh and C.-W. Jen, "High-speed and low-power split-radix fft," *Signal Processing, IEEE Transactions on*, vol. 51, no. 3, pp. 864–874, 2003.

[10] S. Choi, R. Scrofano, V. K. Prasanna, and J.-W. Jang, "Energy-efficient signal processing using fpgas," in *Proceedings of the 2003 ACM/SIGDA eleventh international symposium on Field programmable gate arrays*. ACM, 2003, pp. 225–234.

[11] J.-Y. Kim, L.-S. Kim, and S.-H. Hwang, "An advanced contrast enhancement using partially overlapped sub-block histogram equalization," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 11, no. 4, pp. 475–484, 2001.

[12] Li, Xiying and Ni, GuoQiang and Cui, Yanmei and Pu, Tian and Zhong, Yanli, "Real-time image histogram equalization using FPGA," *Proc. SPIE*, vol. 3561., 1998, pp. 293–299.

[13] A. Farmahini-Farahani, A. Gregerson, M. Schulte, and K. Compton, "Modular high-throughput and low-latency sorting units for fpgas in the large hadron collider," in *Application Specific Processors (SASP), 2011 IEEE 9th Symposium on*. IEEE, 2011, pp. 38–45.

[14] D. Mihhailov, V. Sklyarov, I. Skliarova, and A. Sudnitson, "Parallel fpga-based implementation of recursive sorting algorithms," in *Reconfigurable Computing and FPGAs (ReConFig), 2010 International Conference on*. IEEE, 2010, pp. 121–126.

[15] D. Koch and J. Torresen, "Fpgasort: a high performance sorting architecture exploiting run-time reconfiguration on fpgas for large problem sorting," in *Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays*. ACM, 2011, pp. 45–54.

[16] R. Mueller, J. Teubner, and G. Alonso, "Sorting networks on fpgas," *The VLDB JournalThe International Journal on Very Large Data Bases*, vol. 21, no. 1, pp. 1–23, 2012.

[17] C. Cheng and K. K. Parhi, "High-speed vlsi implementation of 2-d discrete wavelet transform," *Signal Processing, IEEE Transactions on*, vol. 56, no. 1, pp. 393–403, 2008.

[18] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *Journal of Fourier analysis and Applications*, vol. 4, no. 3, pp. 247–269, 1998.

[19] C. Xiong, J. Tian, and J. Liu, "Efficient architectures for two-dimensional discrete wavelet transform using lifting scheme," *Image Processing, IEEE Transactions on*, vol. 16, no. 3, pp. 607–614, 2007.

[20] B. K. Mohanty and P. K. Meher, "Memory efficient modular vlsi architecture for highthroughput and low-latency implementation of multilevel lifting 2-d dwt," *Signal Processing, IEEE Transactions on*, vol. 59, no. 5, pp. 2072–2084, 2011.

[21] C.-C. Cheng, C.-T. Huang, C.-Y. Chen, C.-J. Lian, and L.-G. Chen, "On-chip memory optimization scheme for vlsi implementation of line-based two-dimentional discrete wavelet transform," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 17, no. 7, pp. 814–822, 2007.

[22] C.-Y. Xiong, J.-W. Tian, and J. Liu, "Efficient high-speed/low-power line-based architecture for two-dimensional discrete wavelet transform using lifting scheme," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 16, no. 2, pp. 309–316, 2006.

[23] X. Tian, L. Wu, Y.-H. Tan, and J.-W. Tian, "Efficient multi-input/multi-output vlsi architecture for two-dimensional lifting-based discrete wavelet transform," *IEEE transactions on computers*, vol. 60, no. 8, pp. 1207–1211, 2011.

[24] Y.-K. Cho, J.-S. Lee, H.-M. Yang, and H.-S. Kim, "An efficient color demosaicing using approximated directional line averages," in *SoC Design Conference, 2008. ISOCC'08. International*, vol. 2. IEEE, 2008, pp. II–125.

[25] N.-X. Lian and Y.-P. Tan, "An efficient and effective color filter array demosaicking method," in *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, vol. 4. IEEE, 2007, pp. IV–225.

[26] J. Aelterman, B. Goossens, J. De Vylder, A. Pižurica, and W. Philips, "Computationally efficient locally adaptive demosaicing of color filter array images using the dual-tree complex wavelet packet transform," *PloS one*, vol. 8, no. 5, p. e61846, 2013.

[27] W. Qadeer, R. Hameed, O. Shacham, P. Venkatesan, C. Kozyrakis, and M. A. Horowitz, "Convolution engine: balancing efficiency & flexibility in specialized computing," in *Proceedings of the 40th Annual International Symposium on Computer Architecture*. ACM, 2013, pp. 24–35.

[28] R. Chen and V. K. Prasanna, "Energy-efficient architecture for stride permutation on streaming data."

[29] K. K. Matam and V. K. Prasanna, "Energy-efficient large-scale matrix multiplication on fpgas."

[30] Vivado design suite user guide: Design flows overview. Xilinx. [Online]. Available: http://www.xilinx.com/support/documentation/sw_manuals/xilinx2013_4/ug892-vivado-design-flows-overview.pdf

[31] Xilinx. virtex-7 fpga family. Xilinx. [Online]. Available: http://www.xilinx.com /products/silicon-devices/FPGA/virtex-7/index.htm