Energy-Efficient Histogram Equalization on FPGA

Andrea Sanny
Ming Hsieh Dept. of Electrical Engineering
University of Southern California
Email: sanny@usc.edu

Yi-Hua E. Yang
Xilinx Inc.
Santa Clara, CA
Email: edward.yang@xilinx.com

Viktor K. Prasanna
Ming Hsieh Dept. of Electrical Engineering
University of Southern California
Email: prasanna@usc.edu

***Abstract*—Histogram equalization is a common kernel used for image processing, a widely-used procedure for many present-day applications. Much of the work done emphasizes throughput and area-efficient designs, yet energy efficiency is a relatively untapped field. In this work, we develop an energy-efficient histogram equalization architecture and propose a memory activation schedule to minimize energy consumption. For larger image sizes, we design an efficient buffering and power-down scheme to reduce external DRAM power computation. Pipelining and data hazard prevention are employed to achieve a realistic frame rate of 30+ frames per second. The image sizes range from 240×128 to 3840×2160, with a width of 16 bits per pixel. We compare our results against the theoretical peak performance of histogram equalization on the target device, maintaining up to 77% of the peak performance. Post place-and-route results show that our optimized architecture achieves up to $12.8\times$ higher energy efficiency than the baseline architecture.**

***Index Terms*—histogram equalization, FPGA, energy efficiency, memory activation scheduling, DRAM**

I. Introduction

Image enhancement is one of the major focuses of image processing and is often used for backlit images. It is an important technique for a plethora of diverse applications including medical applications, satellite imaging and thermal imaging [6], [2]. One of the most common algorithms for contrast enhancement is histogram equalization, used for its simplistic nature and effectiveness as an algorithm. Often implemented in image processing pipelines, histogram equalization has been well-researched, particularly in the fields of improving frame rate and low-cost designs.

In this paper, we propose an energy-efficient implementation of histogram equalization. We maintain a realistic frame rate for real-world applications, while attempting to reduce the power consumption of the architecture. Many applications now have low-power modes and there is a strong desire in current technology for high energy efficiency to reduce cost, avoid of overheating, etc. To improve efficiency, we create a power profile for the architecture and determine the bottleneck on power consumption, which is inferred to be memory power. To lessen the power impact of memory, we propose the separation of memory into individual blocks and the use of a memory activation schedule to activate and deactivate these blocks for optimal power reduction. Additionally, by estimating the peak energy efficiency of the target device, we create an upper bound for any histogram equalization algorithm on that device and can compare against this bound to evaluate the sustained energy efficiency of our implementation.

Depending on the application, the images processed can vary significantly in size. Smaller-sized images can be stored directly into on-chip memory, allowing the full structure of the histogram equalization architecture to be completely situated on-chip. However, on-chip resources are limited, and for image sizes which cannot be fully stored using these resources, external memory becomes a requirement. In our case, we use 1 Gb DDR3 memory to temporarily store the incoming input image while maintaining the histogram and cumulative distribution function in on-chip memory. Off-chip memory comes with new factors that must be considered such as the higher frequency of DRAM in comparison with the on-chip structure as well as understanding the opportunities available for lower power consumption when using off-chip memory.

We use post place-and-route results on a state-of-the-art Virtex 7 XC7VX960T FPGA [8] to compare our baseline and optimized designs. We also use the Micron power estimator tool [5] to approximate the amount of power consumed by DRAM for the larger-sized images. The image size varies from 240×128 to 3840×2160 with a width of 16 bits per pixel, which implies a histogram with 65536 in order to fully capture the potential pixel values. The contributions of this work are summarized below:

- A detailed power profile of the histogram equalization components (Section III-C)
- A memory activation schedule for an energy-efficient implementation of histogram equalization (Section III-C1)
- Performance comparison of the baseline and optimized architecture with respect to throughput and energy efficiency (Section IV)
- An evaluation of the effects of using on-chip versus off-chip memory for image storage (Section IV-B2)

The paper is organized into the following sections, starting with Section II which describes the algorithm and related work. Section III gives an overview of our architecture and the optimizations employed to achieve high throughput and low power consumption. Section IV details the experiments and performance comparisons. Finally, we conclude with Section V, summarizing our research and future direction.

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

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

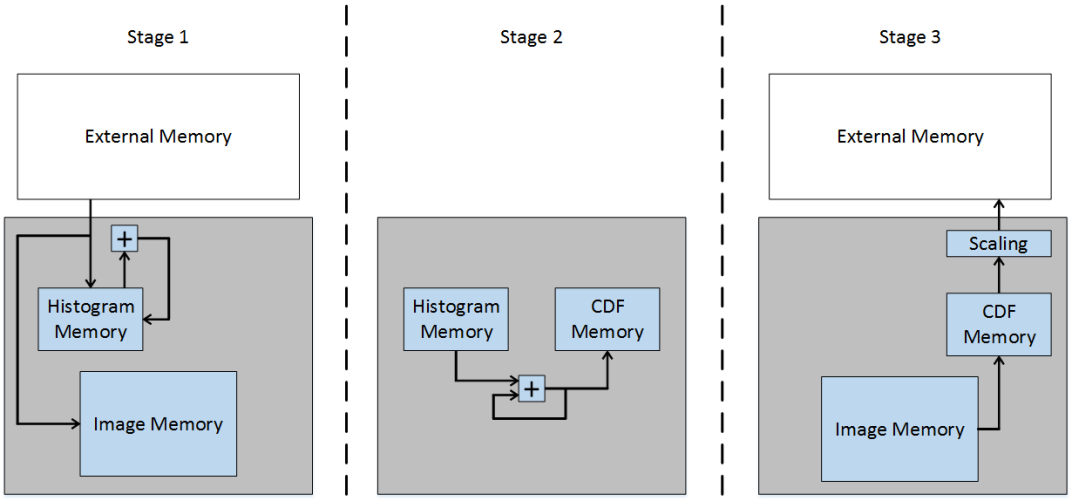

Fig. 1: Histogram equalization architecture stages

II. Background and Related Work

A. Background

Histogram equalization is frequently used in image processing and enhancement, often in conjunction with other kernels to create an image processing pipeline. The basic approach for histogram equalization is as follows:

1) Create a histogram of the image pixels
2) Create the cumulative distribution function (CDF) from the histogram
3) Scale the image pixels using the CDF

There are two standard histogram equalization methods available in literature: global histogram equalization (GHE) and local/adaptive histogram equalization (AHE). Global histogram equalization is considered a simple and fast method, requiring full image knowledge when creating the histogram while adaptive histogram equalization localizes the operation by only considering a window of pixels for local histograms and scaling.

B. Related Work

As a commonly-applicable kernel for image processing and enhancement, the histogram equalization algorithm is a well-studied problem with a conventional focus on hardware implementations, real-time applications and fast performance. Of the two main methods for histogram equalization, AHE is more often used in literature. An example is [7], which uses small windows (or sub-images) of image pixels to create sub-image histograms and scale each pixel locally according to the window around it. The benefit of AHE is the ability to parallelize the process, processing several windows at the same time independently. In comparison with global histogram equalization though, the computational complexity is very high. In [9], three accelerative techniques are combined in order to form a fast AHE method in order to bypass the computational complexity. In [3], the authors attempt to combine the benefits of both methods with a low-pass filter-type mask in order to actualize a nonoverlapped sub-block histogram equalization function. However, in both cases, these papers do not delve into the problem of energy efficiency. The focal point remains on improving real-time processing speed to prevent high computation time when applying the histogram equalization kernel.

There have also been histogram equalization architectures developed on FPGA, such as [4], which achieves a real-time histogram equalization implementation with high processing speed for calculation completion and generation of a look-up table result. In [1], a non-conventional scheme is used to compute the histogram statistics and equalization in parallel, with the intent to create a fast, simple and flexible hardware. Both of these works only study small image sizes and do not consider potential modifications to the architecture, required for large images which may necessitate long processing times and cannot be placed on-chip for convenience.

Although the design space of the histogram equalization architecture is extensively explored, none of the abovementioned works detail methods for improvement of the energy efficiency of the architecture. Energy efficiency, though, has become one of the most important metrics for computing today. We propose a full exploration of the algorithm-architecture mapping space of GHE in order to analyze different image accessing sequences, computation restructuring and memory scheduling with various memory access schemes. GHE is selected due to its fast and simple nature. By exploring the energy-performance-area trade-offs at multiple levels, we obtain an energy-efficient design for histogram equalization.

C. DRAM Access

The state-of-the-art FPGA utilized in our experiments only provides up to 66 Mb on-chip memory (or 188036 Kb BRAMs). For larger images sizes, the available on-chip memory is inadequate for temporary image storage. Larger image sizes will require off-chip memory for storage, such as DDR3 memory, which is useful due to its high frequency and low

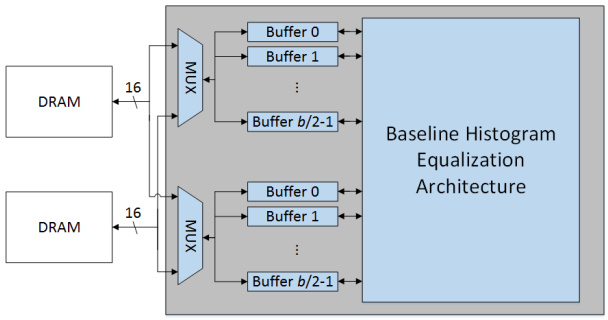

Fig. 2: Baseline architecture with external memory

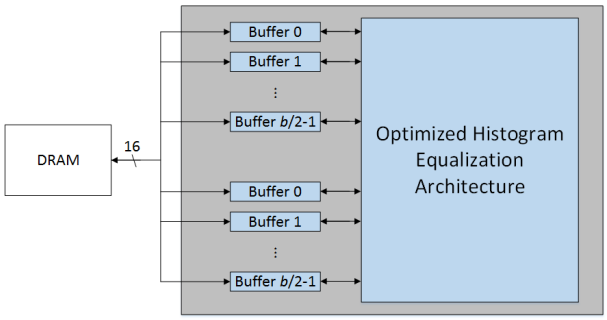

Fig. 3: Optimized architecture with external memory

cost.

We define DRAM power in three categories: active power, read/write/termination power and background power. Active power is the power dissipated when activating or opening a row for future reads or writes, also including the power consumed when precharge the array bitlines. Read/write/term power is the power consumed when data moves in or our of a row. Background power is independent of the DRAM access activity and consists of transistor leakage, peripheral circuitry and data refresh operations. The DRAM memory cells store data using capacitors and, as a result, have cell leakage. Therefore, periodic refreshing is used to maintain the data's validity.

III. Histogram Equalization

A. Architecture

We split our histogram equalization method into three stages of interest which are shown in Figure 1. Stage 1 stores the incoming image into temporary on-chip memory for future accesses, while the incoming pixels are also used to address the histogram memory and create the final histogram. Stage 2, which transpires after the completion of Stage 1, creates the cumulative distribution function based on the created histogram. Finally, Stage 3 begins once the CDF has been fully formed, accessing the previously-stored image data to scale each pixel, using the CDF memory to determine the final result which can be used by other kernels if the histogram equalization architecture is implemented within an image processing pipeline. We assume that the results are read immediately after completion, since our architecture was developed for applications that require streaming.

The baseline architecture is designed as a generic architecture which can maintain a reasonable throughput without the use of any of the power optimizations developed. The baseline optimizes for throughput performance while assuming all memory is active at all time. The optimized architecture uses throughput optimizations together with memory power optimizations such as memory activation scheduling to improve energy efficiency. For small image sizes, only on-chip memory is required.

The baseline architecture is shown in Figure 2 and our optimized external-memory-based architecture is shown in Figure 3. The frequency of the processing on-chip and the memory off-chip may be significantly disparate; for our experiments, 200 MHz and 800 MHz, respectively. Depending on the pixel width, on-chip and off-chip frequencies and the configuration of the DRAM, a number of buffers, b, are required between external memory and on-chip processing to compensate for the difference in frequency.

The baseline architecture requires the image to be fully stored into the DRAM before being accessed by the histogram equalization circuit; it then requires the equalized image to be written back to the DRAM before being sent to the output destination. Therefore, two DRAMs are needed for the baseline architecture in order to satisfy the bandwidth requirement.

The optimized architecture, on the other hand, recognizes that the algorithm requires only one pass of access to the entire image, bypassing the need for storing both input and output images in the DRAM. Instead, the DRAM is only used as temporary storage while sending data to the histogram and CDF generation circuit in a streaming fashion. Therefore the optimized version requires only one DRAM to satisfy its bandwidth requirements.

B. Throughput Optimization

We use two forms of pipelining to achieve reasonable throughput: circuit-level and block-level pipelining. Circuit-level pipelining is defined as the pipelining between computational units and memory within each stage. Through the use of this pipelining, we can improve our operation rate to one pixel per cycle, resulting in an overall reduction in the number of required cycles. We can complete the histogram equalization on an $M \times N$ image and a histogram of L bins in $2MN + L$ cycles with the knowledge that Stage 1 requires at least MN cycles to operate on each pixel, Stage 2 requires at least L cycles and Stage 3 requires MN. For various image sizes, this level of pipelining will ensure a realistic frame rate of greater than 30 fps. However, the use of circuit-level pipelining introduces potential data hazards during operation. In Stage 1, the value of the incoming pixel is unknown, resulting in irregular access patterns to the histogram memory during this stage. There is a strong possibility, since many images have a high likelihood of similar values in small sections, that consecutive image pixels will, at some point, have the same values and access the same histogram location. When accessing consecutively, it is imperative to ensure that the

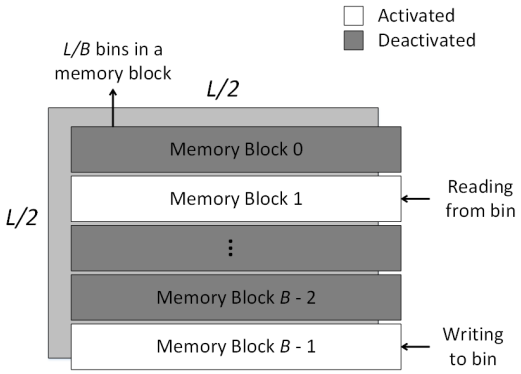

Fig. 4: Memory activation scheduling

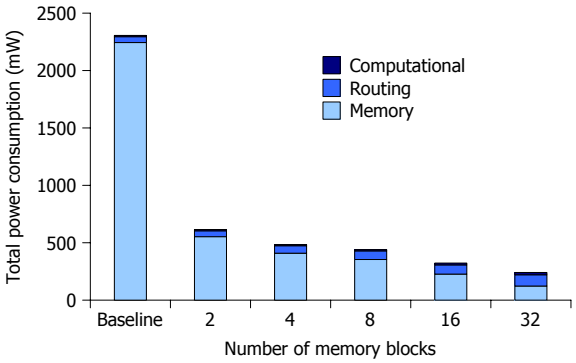

Fig. 5: Power profile at 200 MHz

latest histogram bin value is available during each read and write back to the histogram memory. However, the data is read out and written back in a pipeline-manner, with several bin values in transit between the initial read to a bin, the processing and the final write-back to the bin. Therefore, there is a chance for read-after-write (RAW) hazards during Stage 1. One possible resolution is to stall the pipeline in order to achieve valid results, however this solution impairs throughput. Instead, we include additional circuitry to implement data forwarding during Stage 1 to maintain the actual value written back to the histogram bin during operation.

Though circuit-level pipelining can improve throughput, the number of cycles increases linearly with the number of pixels. Therefore, for larger image sizes, the throughput can be heavily impacted by the size of the image, reducing the improvements possible. To further lower the number of required cycles, we propose the use of block-level pipelining as well. There are natural dependencies between Stages 2 and 3 and Stages 1 and 2. Since the creation of the CDF in Stage 2 is dependent on the completion of the full histogram in Stage 1, Stages 1 and 2 cannot be parallelized. For similar reasons, Stages 2 and 3 cannot be parallelized as well, since the scaling of the pixels is dependent on the completed CDF. However, Stage 1 is not dependent on the previous Stage 3, which enables the incorporation of block-level pipelining at this cross-section. We parallelize Stage 1 and Stage 3 to achieve $MN + L$ cycles. For large MN much greater than L, this is almost a 50% reduction in the total cycle time.

C. Power Optimization

Before selecting optimizations for energy efficiency improvement, we first determine the areas that limit our performance by developing a power profile of the histogram equalization architecture and separating the power of its components. By lowering our power consumption without affecting the number of operations per second, we can improve energy efficiency through power reduction. We assume for this analysis that the histogram is entirely stored into on-chip memory using block RAMs. We separate power into three categories: computational power, interconnection power and memory power. As shown in Figure 5 for a 640×480 image, memory power is a significant factor in power consumption, notably larger than either computational or routing power. To offset the high memory costs associated with active block RAM, we propose the use of memory activation scheduling, separating the memory into blocks and deactivating blocks that are not required at the current time for reading or writing. The baseline use a single large memory block as shown in the figure and has a much higher power consumption in comparison to the optimized architecture as the number of memory blocks increases. The cost of the memory activation schedule lies in the addition of memory scheduler logic and wiring required to control each block, however the increase in routing power is insignificant when compared with the overall decrease in the total power consumption.

1) Memory Activation Scheduling: We develop a memory activation schedule to select the memory blocks for activation and deactivation as shown in Figure 4. By ensuring that only the minimum number of BRAM are active at a time, we reduce the total power consumed by memory. For our implementation, this selection requires a minimization on the number of blocks active for temporary image memory, histogram memory and CDF memory. Depending on the currently operating stage of histogram equalization, certain memories may be completely inactive during the stage, while memories which still are accessed have a significant reduction in power by deactivating most of the memory.

2) DRAM Power-down Mode: When modeling the power of a DRAM, there are two main states: standby mode and power-down mode. In standby mode, the DRAM is available for all possible tasks such as activating a row or reading/writing from a bank. Active standby mode consumes the highest amount of power per cycle. Power-down mode can be categorized as either active or precharged, without the ability to read or write. A row can remain active during power-down mode, however no row can be written to or read from. Power-down mode can be used to operate at a lower current and will consume the lowest amount of power in the precharge power-down mode.

When the external DRAM is used for temporary memory storage for large images, the optimized algorithm ensures that the external memory is always accessed sequentially. In a first order estimation, the total DRAM bandwidth requirement (both read and write) of the optimized circuit at 200 MHz

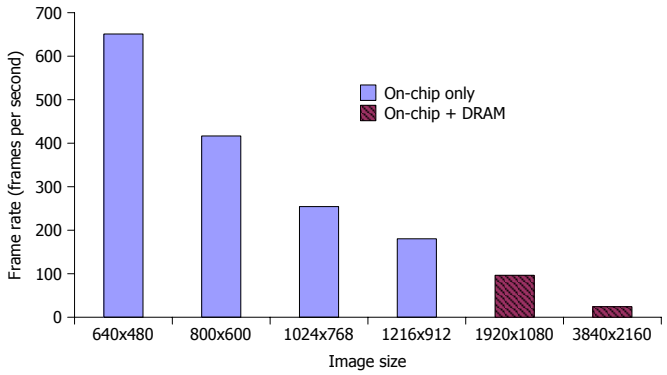

Fig. 6: Frame rate at 200 MHz

is about one-quarter of the available bandwidth of the double data rate (DDR) DRAM at running 800 MHz. Therefore, the DRAM can be put into the power-down mode for up to 75% of total run time.

In power-down mode, no row can be read from or written to though any bank of the DRAM. The DRAM consumes significantly less power in power-down mode than in active mode. By accessing the DRAM in bursts and coordinating the data transfer with that in the on-chip buffer, we could put the DRAM into power-down mode at almost 75% of total run time, resulting in up to $2\times$ reduction of average DRAM power consumption.

IV. Performance Evaluation

Our experiments were conducted on a state-of-the-art Xilinx Virtex 7 XC7VX980T FPGA [8] with a -2L speed grade. The experiments were implemented using the Vivado 2013.4 development tools and the Vivado Power Analysis tool to determine the power dissipation of the designs. All the designs were verified by post place-and-route simulation, using the VCD (value change dump) file as input to the Power Analysis tool for accurate power dissipation elements. The Micron DDR3 SDRAM System-Power Calculator [5] was utilized to determine an accurate estimation of the off-chip power. When analyzing power consumption, we only consider dynamic power in our experiments. We used a wide variety of common image sizes from 240×128 to 3840×2160 with a pixel depth of 16 bits per pixel. The Virtex-7 device has 153K logic slices, a total of 54 Mb block RAM available and 880 Input/Output (I/O) pins.

A. Throughput

Throughput is defined in this work as the frame rate or number of frames per second. We define our minimum frame rate as 30 frames per second. Simple dual-port RAMs are utilized for histogram memory to allow data streaming with one bin being read each cycle while one bin is written to, resulting in a higher performance than if single-port RAM was selected. Higher throughput could also be achieved by employing multiple pipelines, however additional logic would also be required to avoid further data hazards between pipelines. We focus on a serial implementation though, since the focus of our work is not on throughput but on energy efficiency.

Figure 6 shows the frame rate of different image sizes at different frequencies. At 200 MHz, which was used for our experiments on power consumption, we ensure the minimum frame rate 30 frames per second. For our serial implementation, the frame rate is directly related to image size and as the size increases, the throughput will decrease. To improve throughput, several pixels can be processed simultaneously in Stage 1 or Stage 3, requiring some additional hardware to avoid data hazards during the memory accesses to the histogram or CDF memory blocks. The image sizes smaller than 640×480 were not shown due to the very high frame rate which would distort the results of the other larger images in the graph.

B. Energy Efficiency

1) Peak Performance: Peak performance or peak energy efficiency is the upper bound to possible energy efficiency on a chosen target device. This upper bound is defined by the inherent peak performance of the platform, dependent on the target device and IP cores used. We interpret the realistic minimal architecture for histogram equalization to be the processing unit for processing and a memory block for read or write accesses under ideal conditions. Ideal conditions consist of ignoring overheads such as I/O, buffering, and additional logic that may be used for different implementations of histogram equalization. The common elements in every histogram equalization implementation is an adder unit, multiplier unit or divider core which are used during specific stages of the algorithm. This minimal architecture makes the assumption that histogram and CDF memory can be stored on-chip and, for most real-time applications, the histogram memory will be too large to consider distributed RAM. Even in the case of image memory being stored off-chip, there will always be an access to histogram or CDF memory on-chip at every stage. The minimal memory element for the histogram or CDF is an 18 Kb block RAM, which is the smallest possible memory unit available for BRAM implementations. Based on experimental results, the resulting peak performance is 17.8 GOPS/W. This performance is the maximum energy efficiency that can be achieve on our selected FPGA device for a histogram equalization technique, and we use this result to compare against our implementation to determine the efficiency of sustained performance in relation with this bound.

2) Energy Efficiency Comparison: We compare the energy efficiency of our defined baseline with our proposed architecture for a variety of image sizes in Figure 7. Two memory types were utilized: on-chip block RAM or off-chip DRAM, dependent on the image size requirement. The same overall histogram equalization architecture is used for all image sizes; the main difference between the designs is the additional buffering required by external memory or the additional memory control used for the scheduling. The 240×128 result is not shown for $B = 32$ blocks, due to the fact that the block RAM size is limited to a minimum of 18

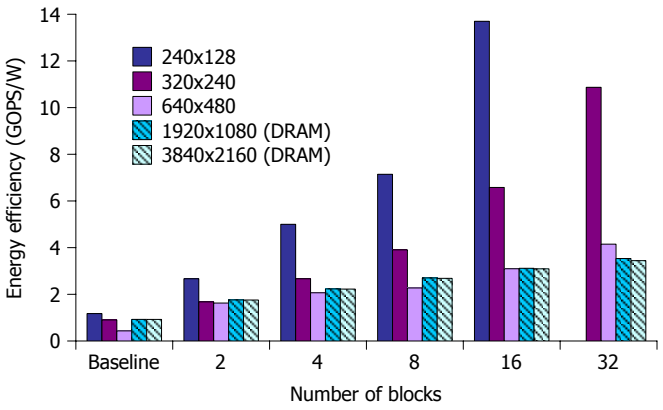

Fig. 7: Energy efficiency comparison

Kb. The maximum number of blocks which can be completely filled is $B = 30$ blocks for this case.

We present two of the larger image sizes in Figure 7 which utilize off-chip memory for image storage: 1920×1080 and 3840×2160. The resulting improvement of the optimized version over the baseline is from $12.8\times$ to $3.72\times$, with the minimum improvement caused by the limitations of control over the DRAM power states. Though a 1920×1080 image can fit on-chip, there is a crosspoint at which the use of DRAM improves energy efficiency further than using on-chip memory. The two images have similar performance to the 640×480 on-chip image, and if on-chip memory was used for image sizes larger than 640×480, the area and communication costs due to block RAM's inflexible placement on the board outweigh the power costs of DRAM. If we compare our final energy efficiency results of the optimized architecture against the upper bound of peak performance, the proposed architecture achieves up to 77% of the peak energy efficiency.

In Figure 8 and Figure 9, we display the power consumption breakdown of the various components with and without external DRAM for the baseline and the optimized version, respectively. We compare the two large images against one of the smaller image sizes (640×480). Without optimizations, BRAM dissipates a significant amount of power, however with the scheduling, the power is considerably reduced. DRAM power, due to its power dissipation during any state, does not have the same capabilities for power reduction. The advantage of off-chip memory is the lower routing power since the scheduler does not need to control the on-chip image memory, which results in markedly less wiring. The power breakdown infers that we can maintain comparable energy efficiency to the on-chip version using off-chip memory.

V. Conclusion and Future Work

We propose an optimized histogram equalization architecture which utilizes a memory activation schedule for lower memory power consumption as well as two levels of pipelining along with data hazard prevention circuitry to ensure a reasonable throughput of at least 30 frames per second. In the future, we plan to analyze and optimize additional kernels

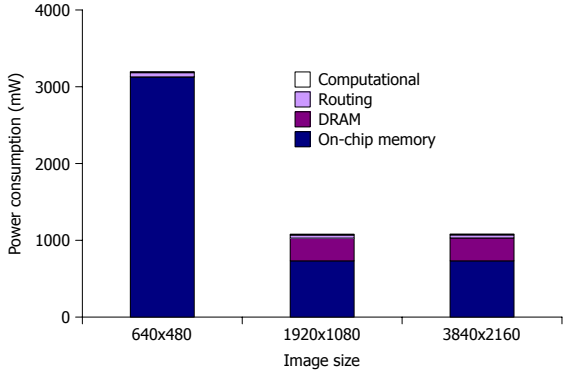

Fig. 8: High-level power analysis (baseline)

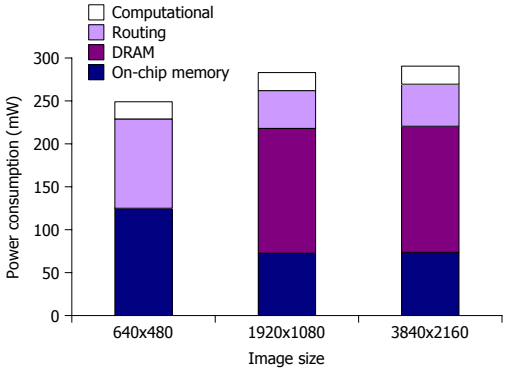

Fig. 9: High-level power analysis (optimized, B = 32)

within a common image processing pipeline, as well as look further into external memories and their controls which can result in the optimal energy efficiency for image storage and processing.

References

[1] Abduallah M Alsuwailem and Saleh A Alshebeili. A new approach for real-time histogram equalization using fpga. In *Intelligent Signal Processing and Communication Systems, 2005. ISPACS 2005. Proceedings of 2005 International Symposium on*, pages 397–400. IEEE, 2005.

[2] R Kathiravan, R Shanmugasundaram, and N Santhiyakumari. Satellite image resolution enhancement using contrast limited adaptive histogram equalization. *Digital Image Processing*, 6(3):161–165, 2014.

[3] Joung-Youn Kim, Lee-Sup Kim, and Seung-Ho Hwang. An advanced contrast enhancement using partially overlapped sub-block histogram equalization. *Circuits and Systems for Video Technology, IEEE Transactions on*, 11(4):475–484, 2001.

[4] Xiying Li, GuoQiang Ni, Yanmei Cui, Tian Pu, and Yanli Zhong. Real-time image histogram equalization using fpga. In *Photonics China'98*, pages 293–299. International Society for Optics and Photonics, 1998.

[5] Micron. Micron ddr3 sdram system-power calculator. http://www.micron.com/products/support/power-calc.

[6] K Raj Mohan and G Thirugnanam. A dualistic sub-image histogram equalization based enhancement and segmentation techniques for medical images. In *Image Information Processing (ICIIP), 2013 IEEE Second International Conference on*, pages 566–569. IEEE, 2013.

[7] Stephen M Pizer, E Philip Amburn, John D Austin, Robert Cromartie, Ari Geselowitz, Trey Greer, Bart ter Haar Romeny, John B Zimmerman, and Karel Zuiderveld. Adaptive histogram equalization and its variations. *Computer vision, graphics, and image processing*, 39(3):355–368, 1987.

[8] Xilinx. Virtex-7 fpga family. http://www.xilinx.com/products/silicon-devices/fpga/virtex-7/index.htm.

[9] Wang Zhiming and Tao Jianhua. A fast implementation of adaptive histogram equalization. In *Signal Processing, 2006 8th International Conference on*, volume 2. IEEE, 2006.