Energy Efficient Image Convolution on FPGA*

Agrim Gupta**
BITS-Pilani
Rajasthan, India
Email:agrimgupta92@gmail.com

Viktor K. Prasanna
Ming Hsieh Department of Electrical Engineering
University of Southern California
Los Angeles, USA 90089
Email:prasanna@usc.edu

***Abstract*—2-D Convolution is widely used in image and video processing. Despite its computational simplicity it is memory and energy intensive. In this paper, a parameterized Image Convolution architecture in terms of the convolution window size and coefficients, the input pixel resolution, the image size and the type of memory used is proposed to identify design trade-offs in achieving energy efficiency. The proposed architecture uses memory scheduling and parallelism to achieve energy efficiency. To understand the efficiency of our implementations, we implement a baseline architecture and evaluate the energy efficiency (defined as the number of operations per Joule). From the experimental results, a design space is generated to demonstrate the effect of these parameters on the energy efficiency of the designs. A performance model is constructed to estimate the energy consumption for varying parameters. On a state of the art FPGA, for $N \times N$-image ($64 \leq N \leq 512$), our designs achieve energy efficiency up to 32.98 Gops/Joule. Our implementations sustain up to 34.38% of the peak energy efficiency.**

***Index Terms* — Image Convolution, FPGA computing, Energy efficient design**

I. Introduction

FPGAs have become an attractive option for implementing signal processing applications because of their high processing power and customizability. State-of-the-art FPGAs offer high operating frequency, unprecedented logic density and a host of other features. As FPGAs are programmed specifically for the problem to be solved, they can achieve higher performance with lower power consumption than general purpose processors.

Most image processing algorithms are local and two dimensional (2-D) by nature. The most popular of which is Image Convolution. Conceptually 2-D convolution is the process of performing sum of the products of kernel matrix with corresponding image pixels. However, its software implementation requires huge amount of resources in terms of computational power, energy, latency and memory. Several algorithms have been proposed for area efficient implementation on FPGA. They focus mainly on different buffering models for performance optimization in terms of bandwidth and data transfer to/from the memory [1]. Algorithms have been proposed [2], [3] and developed for high performance, operating frequency and reduced resources. Although there are many example in literature of 2D convolution implementations, to the best of our knowledge they have not explored the design space for energy efficiency.

Energy is a key metric in computing today. To obtain an energy efficient design, it is necessary to analyze the trade-offs between energy and latency on a parameterized architecture. We implement a baseline architecture parameterized in terms of convolution window size and coefficients, the input pixel resolution and the image size to identify energy hot-spots. For on-chip storage of image a significant amount of energy is consumed in memory. Inherent parallelism of the image convolution algorithm is exploited to increase the performance. More than one pixel can be simultaneously convoluted using multiple Multiplier Accumulators (MAC). However, the number of MACs that can be used is limited by the bandwidth of the memory access ports and the number of bits per pixel (data width). Optimized architecture is parameterized in terms of number of MACs to identify their optimum number for a particular image size and data width. The energy consumption can be further reduced by memory scheduling. In memory scheduling only the memory being accessed is kept active, the rest is switched off. In this paper, we make the following contributions:

1) A parameterized architecture of 2-D Image Convolution. Parameters are image size, kernel size and number of Multiplier Accumulators.
2) A novel energy efficient architecture based on memory scheduling and simultaneous convolution of multiple pixels.
3) An upper bound on the energy efficiency of any image convolution implementation on a given target device.
4) A performance model to estimate the energy efficiency of the implementation for varying parameters.
5) Implementations that can sustain upto 34.38% peak energy efficiency for image size 128×128 (16 bits per pixel).

The rest of the paper is organized as follows. Section II covers the background and related work. Section III introduces the proposed architecture and its implementation on FPGA. Section IV presents experimental results and analysis. Section V concludes the paper.

II. Background and related work

A. Background

Spatial convolution in two dimension is a neighborhood operation and is defined by:

$$f[x,y] * g[x,y] = \Sigma_{i,j} f[j,k] * g[x-j, y-k] \tag{1}$$

where $f[x,y]$ represents the input image and $g[x,y]$ the convolution kernel.

*This work has been funded by DARPA under grant number HR0011-12-2-0023.
**Participation sponsered by the Viterbi India 2013 Program, University of Southern California.

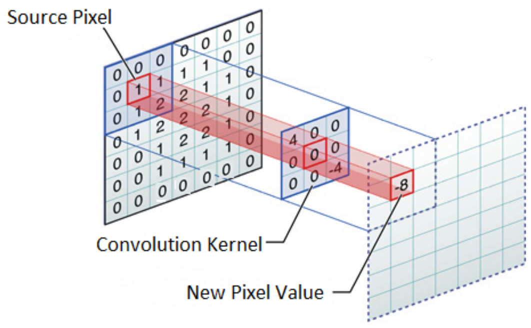

Fig. 1: Convolution using 3×3 kernel

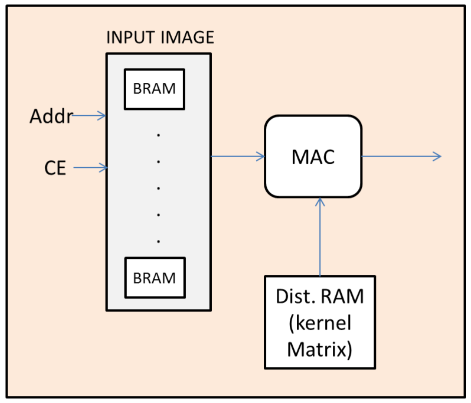

Fig. 2: Baseline Architecture

Convolution is a basic operation in many image and video processing tasks. Fig. 1 illustrates the convolution process for a $N \times N$ image. Each pixel of the input image $f[x, y]$ is convoluted with the $k \times k$ window centered on $f[x, y]$. The window acts like a sliding template, called convolution kernel. Each window element is multiplied with the corresponding image element. The $k \times k$ products hence obtained are added to produce the convoluted pixel. The image is zero padded to compute the convolution of boundary pixels.

B. *Related Work*

There are many examples in literature of 2D convolution implementations, but to the best of our knowledge none of them take into account energy efficiency constraint as the main requirement. Most of the prior work has been focused on high performance, reduction of FPGA resources and area.

In [3], a high performance fully re-configurable FPGA based 2D convolution processor was developed. The proposed architecture operates on image pixels coded with various bit resolutions and varying kernel weights avoiding power and time-consuming reconfiguration. In [1], an area efficient implementation of 2D convolution for space applications is presented.

In [4], kernel specific alternative architectures for image convolution were proposed which focused on reduced power consumption, FPGA resources and throughput. In [5], [6], a FPGA based configurable systolic architecture specially tailored for real-time window-based image operations is presented.

In [2], the focus was on reduction of resource use and high operating frequency by developing a new architecture for image convolution. Optimization techniques like increased parallelism by replication and pipelining, search for high regularity and re-utilization of common resources (adders), optimization of multipliers by means of adder trees were used in the proposed architecture. Though these architectures can achieve high operating frequency and reduction of resources, energy efficiency has not been explored and evaluated in these works.

In this work, we explore the design space for energy efficiency. The design space exploration is performed on the current state-of-the-art FPGAs. Like [2], we replicate MACs, the functional units of the architecture to apply convolution operation simultaneously on multiple image pixels. Further optimization like re-utilization of common resources and memory scheduling are used to reduce the energy consumption of the proposed architecture.

III. Architecture and Optimization

The most common FPGA architecture consists of an array of logic blocks called Configurable Logic Block, I/O pads, and routing channels. The Configuration Logic Block in most of the Xilinx FPGAs contain small single port or double port RAM. This RAM is normally distributed throughout the FPGA over many LUTs and so it is called distributed RAM. Other type of memory available is block RAM or BRAM. A block RAM is a dedicated two port memory containing several kilo-bits of RAM and cannot be used to implement other functions like digital logic. Additionally, the BRAM architecture supports power saving by allowing only a portion of the memory to be active on each memory access.

A. *Baseline Architecture*

We implement the basic image convolution algorithm as explained in Section II-A. The image to be convoluted is assumed to be stored on-chip. Memory used to store the input image and the convolution kernel can be either Distributed RAM or Block RAM. According to [7], when used for large size memories, BRAM consumes less power than dist. RAM. Image convolution being a memory intensive operation, the input image is stored in BRAM. The image is zero padded to compute the convolution of the boundary elements. The kernel matrix is stored in Distributed RAM.

The kernel is moved in a sliding manner over the entire image starting from the beginning, keeping in mind that each localization in the input image will generate one output pixel. In every clock cycle the MAC accepts two operands, a multiplier and a multiplicand, and produces a product ($A \times B = Prod$) that is added to the previous result ($S = S + / - Prod$). For the case of image convolution the two inputs are the kernel matrix element and the corresponding image pixel. The entire memory is kept active till the convolution of all pixels is computed. To perform convolution of one pixel $2 \times k^2$ read

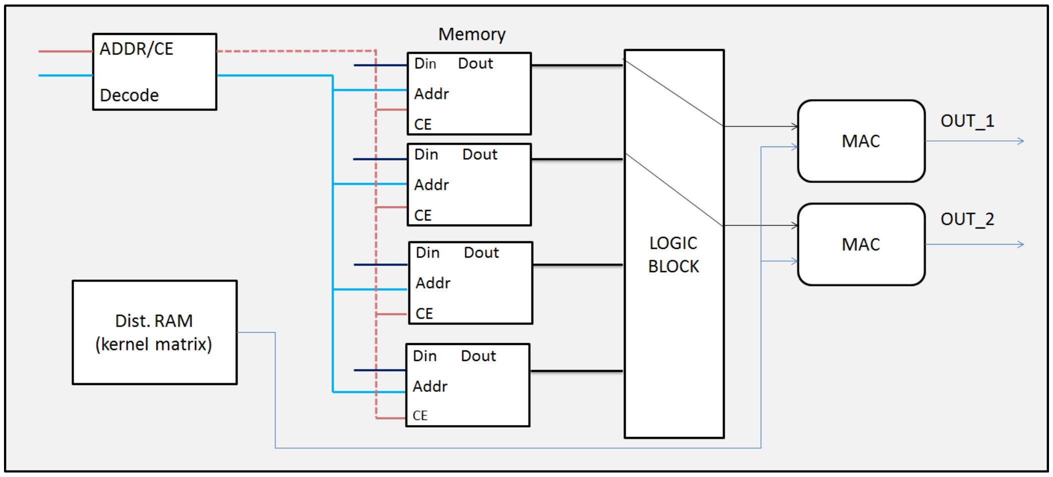

Fig. 3: Overall Architecture

operations have to be performed to obtain input image pixels and kernel matrix elements. It takes k^2 clock cycles to produce one output pixel for kernel size $k \times k$. The baseline architecture is parameterized in terms of user defined parameters like image size, kernel size and elements and data width. Convolution is a memory and computation intensive operation. The number of additions and multiplications per image pixel are $k^2 - 1$ and k^2 respectively. The algorithm performs $2N^2k^2$ computations in N^2k^2 clock cycles using one MAC (multiplier/accumulator). It performs one multiplication and addition operation per clock cycle. Let the power consumed by one MAC be C_1 and C_2 be power consumed to store one pixel in BRAM. The total energy consumed would be:

$$E = C_1N^2k^2 + C_2(N^2k^2)N^2 \tag{2}$$

We see from Equation 2, majority of the energy is consumed in the memory. The next sections proposes optimizations to reduce the energy consumption.

B. Energy Efficient Architecture

Energy efficient image convolution can be performed by reducing the energy consumption in the memory and the latency of the system by processing multiple pixels simultaneously. We propose the following optimizations for energy efficient image convolution:

- *Memory Scheduling*: Major source of energy consumption in on-chip designs is memory energy. An effective way to reduce the energy consumption is to switch off the BRAM blocks which are not in use at a particular instant of time. The overhead in logic power is compensated by significant reduction in the memory power. Memory scheduling can be done using a decoder which takes input as the address of the memory location to be accessed. It generates corresponding enable signal such that only the BRAM block containing the pixel to be accessed is enabled. The output of all the BRAM blocks is then multiplexed to give the final output. As compared to the trivial implementation in which all the BRAM blocks are active this implementation requires more area and logic but results in significant reduction in energy consumption especially for large image sizes.
- *Multiple number of MACs (L)*: The energy consumption can also be reduced by decreasing the latency. Multiple image pixels can be convoluted simultaneously to reduce the latency of the system by exploiting the inherent parallelism of the image convolution algorithm. This also reduces the number of memory read operations for the kernel matrix elements.

Detailed procedure for the optimized architecture is shown in Algorithm 1. In our design, we initialize the BRAM with the input image. Each BRAM can be characterized as $b \times w$ where b is the size of each memory location and w is the number of such memory locations. The kernel matrix is stored in dist. RAM. The number of image pixels to be simultaneously convoluted is a user defined parameter specified by number of MACs. The overall architecture is shown in Fig. 2.

Algorithm 1 Optimized Architecture

```
 1: {$N \times N$ Input image and $k \times k$ convolution kernel is read into
    on chip memory}
 2: for all $j$, $1 \le j \le N^2$ do
 3:     for all $l$, $1 \le l \le L$, do in parallel do
 4:         for all $i$, $1 \le i \le k^2$ do
 5:             Decode the pixel address
 6:             Enable BRAM block corresponding to input address
 7:             Input to MAC image pixel and kernel element
 8:         end for
 9:     end for
10: end for
```

Number of pixels that can be accessed in one clock cycle depends on the bandwidth of the memory access port specified by b. The number of BRAMs that need to be active for L MACs and d data width is given by $L \times d/b$. L pixels are fetched from $L \times d/b$ BRAMs in each clock cycle and multiplied with the corresponding kernel element. One added advantage of this implementation is that for all L pixels the kernel element is fetched only once. In contrast, in baseline implementation for every pixel the kernel element had to be fetched. As shown in Fig. 3 the decoder block takes the

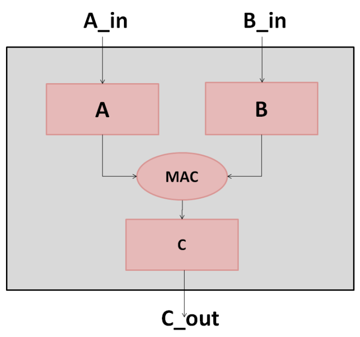

Fig. 4: Minimal Architecture

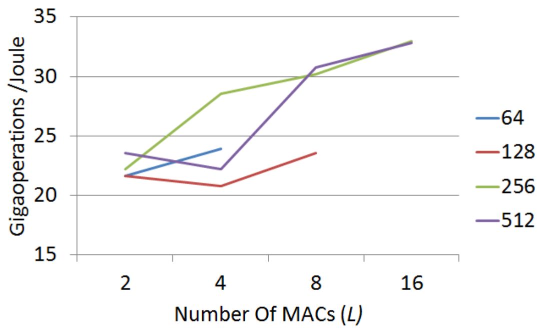

Fig. 5: Energy Efficiency for various problem sizes with varying L

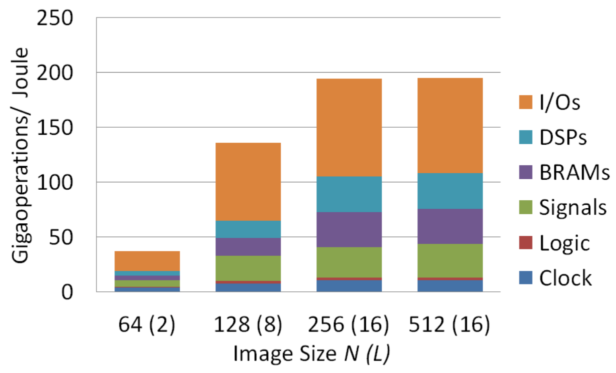

Fig. 6: Power consumed by components for optimized implementation for varying problem sizes

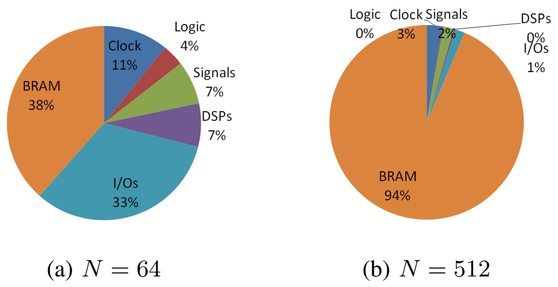

Fig. 7: Power Consumption Profile for Baseline Implementation

addresses of the pixels as input and activates the BRAM in which the pixels are stored. Higher the number of simultaneous access, more BRAMs are required to be activated. There is commensurate increase in logic, DSP and I/O power. Fig. 3 shows an architecture with four BRAMs and two MACs for $L = 2$, $d = 16$, $b = 18$. Therefore, at any instant two BRAMs are active. Initially, pixels are fetched from the first two BRAMs. When all the pixels in the two BRAMs are convoluted, then for the subsequent clock cycles, the logic block switches the connections to the last two BRAMs. This process is repeated for larger designs depending upon number of MACs and data width.

Using L MACs reduces the total latency of the design to N^2k^2/L. Each BRAM block can store R rows depending on the size of the image. Total number of pixels active at any point during the design would be $R \times N$ per BRAM. Number of active BRAMs is given by $L \times d/b$. Hence we can write the energy equation as:

$$E = C_1 N^2 k^2 + C_2 (N^2 k^2) RNd/b \qquad (3)$$

Neglecting the energy spent in I/O, access to memory, cache and other buffers the energy consumed by MAC remains the same. This can be verified by Equation 2 and Equation 3. However, the total energy consumption reduces. Even though the logic energy remains the same for the case of multiple MACs, the energy efficiency increases because of reduced latency. Intuitively, an optimal implementation would have memory and logic power of the same order of N and k. Theoretically, its possible when $b = RNb$. However it's implementation is limited by the maximum number of independent data access that can be done from BRAM per clock cycle as the above results gives a very large bandwidth requirement. In the next Section we vary the number of MACs to obtain an optimum number for a given data width and problem size.

IV. Experiments And Evaluation

All the designs were implemented in Verilog on Virtex-7 FPGA (XC7VX690T, speed grade -3) using Xilinx ISE 14.5. The input image size lies in the range $64 \leq N \leq 512$ with bits per pixel varying in the range $8 \leq d \leq 32$. The convolution kernel size lies in the range $3 \leq N \leq 7$ with bits per element varying in the range $8 \leq d \leq 32$. Xilinx Multiply Accumulator core [8] was used in our implementation. We used manual latency of one as configuration option when generating the core. The designs were verified by post place-and-route simulation. The reported results are post place-and-route results. As we are interested in the power consumed by the architecture, we considered only the dynamic power in our experiments. After post place and route we measured the power using Xpower estimator [9]. All our results are reported at operating frequencies of 200 MHz.

A. *Performance Metric*

We consider *Energy Efficiency* as the metric for performance evaluation. Energy Efficiency is defined as the number of operations per unit energy consumed. For image convolution, with $N \times N$ image and kernel size $k \times k$ energy efficiency

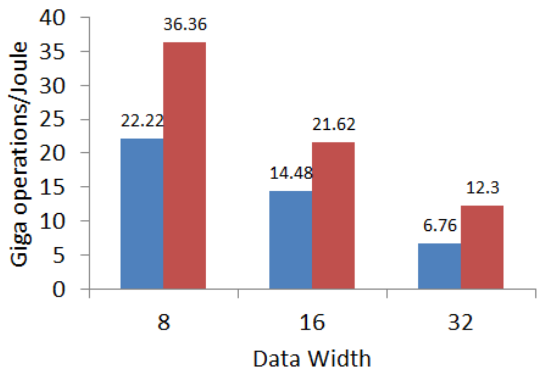

Fig. 8: Energy Efficiency comparison for $N = 64$

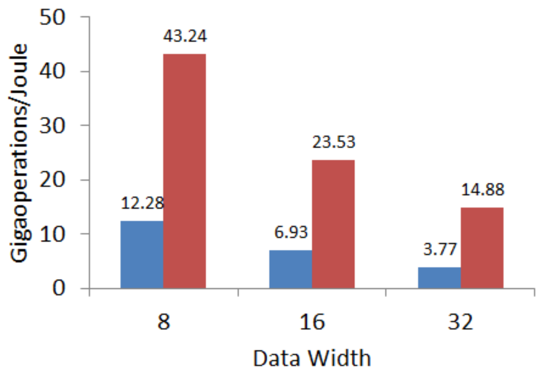

Fig. 9: Energy Efficiency comparison for $N = 128$

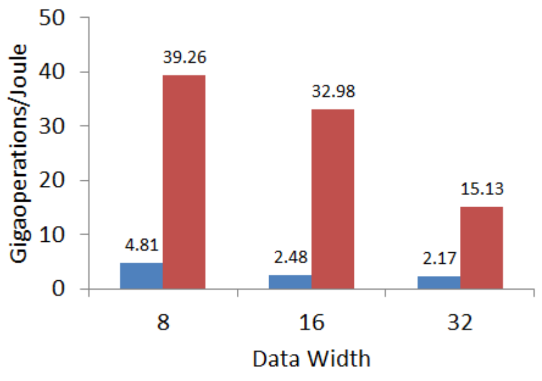

Fig. 10: Energy Efficiency comparison for $N = 256$

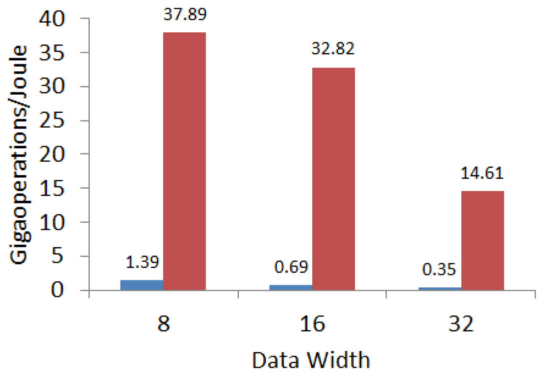

Fig. 11: Energy Efficiency comparison for $N = 512$

is given by N^2k^2/energy consumed by the design. Energy of the design = time taken by the design/ average power dissipation of the design. Alternatively Energy efficiency of the design is Power efficiency (Power efficiency = number of operations per second/Watt).

B. *Peak Energy Efficiency*

The energy efficiency of any image convolution design is upper bounded by the inherent peak performance of the platform. This depends on the target device and the IP cores used to perform the arithmetic operations. We measure this peak energy efficiency by using a minimal architecture for the processing element under ideal conditions. Thus, we ignore all the overheads such as memory energy, I/O, access to memory, cache and other buffers that may be employed by an implementation. This minimal architecture consists of only multiplication and addition as these are the basic operations in any image convolution design. The design consists of two inputs A and B and one MAC as shown in Fig. 4. The other components such as routing, memory will only increase the energy consumption. Based on the experiments, this architecture can operate at a maximum frequency of 345 MHz, consumes 154.1 mW when operating at maximum frequency, and occupies 44 slices.

Energy efficiency for this design is given by $2 \times$ *frequency / power consumed by the design*. For identifying the frequency at which this design achieves peak energy efficiency, we measured the power consumed by the design at intervals of 50 MHz in the range of 0 to 350 MHz frequencies. Peak energy efficiency is observed at 200 MHz. The peak energy efficiency at 50% toggle rate is 95.92 Gops/Joule (16 bits per pixel).

We use this peak energy efficiency as an upper bound on the performance of any algorithm and architecture for image convolution and compare the sustained performance of an implementation against this bound. Note that as the IP cores improve, we can expect a corresponding increase in the sustained performance of our algorithm and the architecture for Image convolution.

C. *Energy hot spots*

In this experiment, we identify the energy hot spots in the baseline implementation. The baseline implementation has one MAC and no memory scheduling. Based on the experimental results, we observe that for large problem size, significant amount of energy is consumed in BRAM. Since a small amount of memory is required to be active at any instant, memory scheduling helps in reducing energy consumption. Fig. 7 also suggests that for small image size I/Os consume a significant amount of energy.

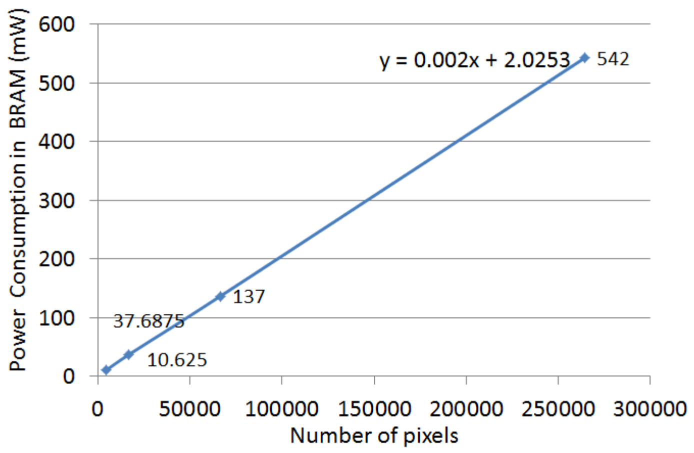

Fig. 12: Power consumed in BRAM for varying Problem Sizes

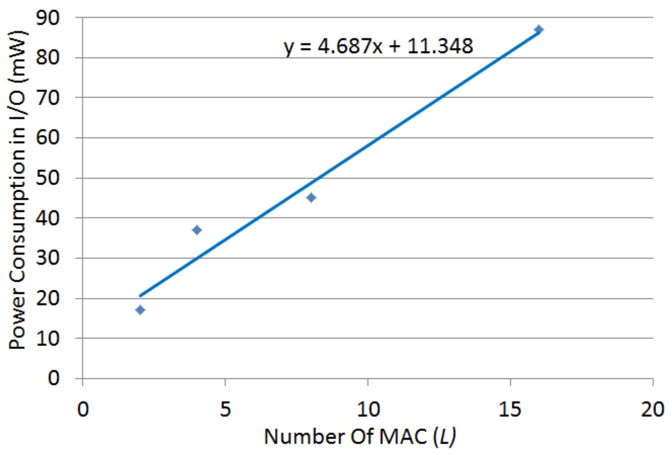

Fig. 13: Power consumed in I/O for varying L

D. Design Space Exploration

We explore the effect our proposed optimizations have on the energy efficiency performance metric. Design space exploration is performed by altering the number of MACs for various problem sizes with 16 bits per pixel (bpp). Optimal value of MAC for each problem size is identified and experiments for data widths 8 and 32 are performed using them. The maximum number of MACs that can be used is limited by the bandwidth capacity of the BRAM and total number of BRAMs used. For example, if we consider $N = 64$ and 8 bpp we require 3 single port RAMs (port width = 18 bits). Maximum number of MACs is 6 in this case. From Fig. 6 we notice as the number of MACs increases the power consumption in DSP, Signal and Logic increases commensurately. But there is significant increase in the power consumption in I/Os. Taking into account these factors the experiments were limited to the number of MACs as shown in Fig. 5.

We finally compare our optimized implementation with the baseline architecture. The image size is varied over the range $64 \leq N \leq 512$ for data sizes 8, 16 and 32. The comparison for optimized architecture with optimum number of MACs is depicted in Fig. 6. Based on the experimental results we have the following observations:

- Energy efficiency of the baseline architecture reduces for all data sizes as the problem size increases. This is due to significant amount of energy being consumed in BRAMs that are idle. The proportion of which is higher for large image sizes.
- Energy efficiency of the optimized architecture reduces for all data sizes as the problem size increases. This is due to significant amount of energy begin consumed in I/Os. As the number of MACs is increased I/Os become a major contributor of energy consumption which is similar to the case of small image size.

E. Performance Model

In this section we revisit Equation 2 and Equation 3 to determine the constants C_1 and C_2. With the help of these equations we can determine the energy efficiency accurately to a great extent. This can help speed up the design space exploration. The power consumed in DSP is primarily due to MAC. Experimental results show that power consumed by one MAC at 200 MHz frequency is 2 mW for 16 bits per pixel. To determine the value of C_2, we plot total BRAM power consumption vs. Number of pixels. Slope of the line in Fig. 12 gives the value of the constant C_2. The power required to store one pixel in BRAM is .002 mW. Fig. 14 shows a comparison between the values of energy efficiency found out experimentally with the values that were obtained using Equation 2. For smaller image sizes there is more error because we have ignored energy consumption in I/Os, Signal and Clock. As the image size increases their proportion of energy consumption decreases. For large designs our model accurately predicts the value of the energy efficiency metric.

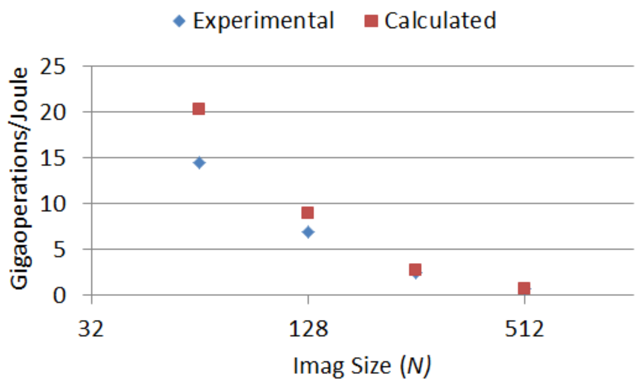

Fig. 14: Comparison of Energy Efficiency for baseline architecture

Fig. 6 shows that to accurately model the optimized design it is necessary to incorporate the energy consumption in Signal and I/O. Let the power consumed per MAC in Signal be C_3 and C_4 be the power consumed per MAC in I/Os for 16 bits

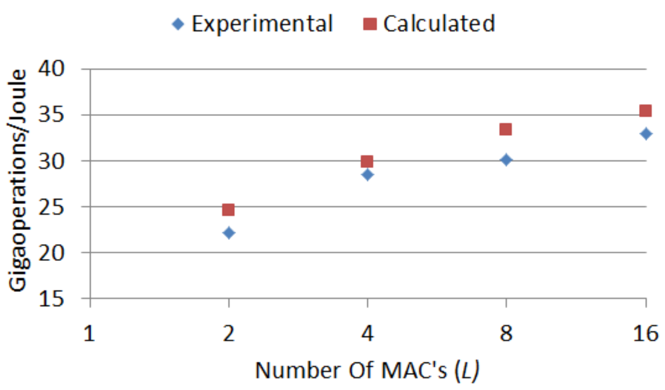

Fig. 15: Comparison of Energy Efficiency for optimized architecture ($N = 256$)

per pixel. We can rewrite Equation 3 as:

$$E = C_1 N^2 k^2 + C_2 (N^2 k^2) R N d / b + C_3 N^2 k^2 + C_4 N^2 k^2 \tag{4}$$

The value of C_3 can be determined by plotting power consumption in Signal vs. Number of MACs. From the slope of the graph the value of C_3 was determined to be 1.89 mW/MAC. Similarly, the constant C_4 can be determined from Fig. 13. The value of constant in the equation of the line being large, we incorporate it separately in the equation below. Equation 4 can be modified by grouping common terms to obtain:

$$E = 8.577 \times N^2 k^2 + C_2 (N^2 k^2) R N d / b + 11.35 \ (mJ) \tag{5}$$

We use the above expression to calculate the energy efficiency for $N = 256$ as shown in Fig. 15. Unlike the baseline architecture as the problem size increase, in the optimized architecture the energy consumption due to Clock and Logic is not small in comparison to total energy consumption. However, to great extent the value of energy efficiency metric obtained from the performance model is in close approximation to the actual experimental values.

V. Conclusion

In this work, we presented a parameterized architecture for energy efficient implementation of image convolution on FPGA. A baseline architecture was implemented and studied for various problem and data sizes to identify the energy hot spots. A novel energy efficient architecture based on memory scheduling and simultaneous convolution of multiple pixels was developed. Design space was explored in terms of number of MAC to determine the optimum number of MACs for specific problem size. A performance model was developed to estimate the energy consumption of the design. In the future we plan to work on an accurate high-level performance model for energy-efficiency estimation, which can be used to accelerate design space exploration to obtain an energy efficient design.

References

[1] S. Di Carlo, G. Gambardella, M. Indaco, D. Rolfo, G. Tiotto, and P. Prinetto, “An area-efficient 2-d convolution implementation on fpga for space applications,” in *Design and Test Workshop (IDT), 2011 IEEE 6th International*. IEEE, 2011, pp. 88–92.

[2] M. A. Vega-Rodriguez, J. M. Sanchez-Perez, and J. A. Gomez-Pulido, “An optimized architecture for implementing image convolution with reconfigurable hardware,” in *Proc. Sixth Biannual World Automation Congress*, vol. 16, 2004, pp. 131–136.

[3] S. Perri, M. Lanuzza, P. Corsonello, and G. Cocorullo, “Simd 2-d convolver for fast fpga-based image and video processors,” in *Conference on Military and Aero-space Prog. Logic Devices*, 2003.

[4] J. Y. Mori, C. H. Llanos, and P. A. Berger, “Kernel analysis for architecture design trade off in convolution-based image filtering,” in *Integrated Circuits and Systems Design (SBCCI), 2012 25th Symposium on*. IEEE, 2012, pp. 1–6.

[5] C. Torres-Huitzil and M. Arias-Estrada, “Fpga-based configurable systolic architecture for window-based image processing,” *EURASIP Journal on Applied Signal Processing*, vol. 2005, pp. 1024–1034, 2005.

[6] V. Hecht and K. Ronner, “An advanced programmable 2d-convolution chip for, real time image processing,” in *Circuits and Systems, 1991., IEEE International Sympoisum on*. IEEE, 1991, pp. 1897–1900.

[7] “XST User Guide for Virtex-6, Spartan-6, and 7 Series Devices,” http://www.xilinx.com/support/documentation.

[8] “Xilinx LogiCORE IP Multiply Accumulator v2.0,” http://www.xilinx.com/support/documentation/ip_documentation/xbip_multaccum_ds716.pdf.

[9] “Xilinx Power Tools Tutorial,” http://www.xilinx.com/support/documentation/user guides/ug440.pdf.