DEVELOPMENT OF FLOATING-POINT MAC ENGINE FOR 2-D CONVOLUTION OF IMAGE

BommadeniCharan Raj[1],Dr. M.Pushpalatha[2], Dr.T. Krishnarjuna Rao[3]
[1]PG Scholar, Dept. of VLSI & Embedded System, SIET, Hyderabad.
charanraj0811@gmail.com
[2]Associate Professor, Dept. of ECE, SIET, Hyderabad
[3]Associate Professor, Dept. of ECE, SIET, Hyderabad

ABSTRACT
IEEE 754-2008 Floating point numbers are frequently employed in the new trend of Graphics Processing Architecture. One of the common procedures used in image processing applications is convolution, which is computationally demanding and necessitates the deployment of an effective image processing architecture. The sliding window approach for 2-D image convolution is proposed as an acceleration method in this study using a single-precision floating point MAC engine. The engine offers programmable options to define filter and picture size and employs a modified approach for virtual zero-padding to conserve memory.The MAC design has been suggested to be enhanced by a low power multiplier with lower dynamic power, particularly when acting on pixels, and a quicker increment by one circuit based on AND-EXOR gate topologies. The research concludes by comparing the picture quality of the image acquired from the RTL Simulation of the suggested design and displaying the post-synthesis power dissipation, area estimation, and results.

1.INTRODUCTION

Various imaging applications such as object detection, classification, and visual search requires image filtering. Among these applications, visual search involves a massive number of calculations for feature detection in the input image. In such scenarios, rather than using software (SW) implementations that are not capable of providing real-time performance, hardware (HW) implementations of image filtering can be more performance optimized for real-time processing of higher resolution images.

Over the years, several designs have been proposed but are tailored for Gaussian kernels only. These methods and designs cannot be used for applications that work on kernels other than Gaussian kernels. The design presented in [1] is compatible with such scenarios as it does not recur to the separability property of the Gaussian kernels but requires a complex arrangement of SRAM modules. This complexity grows as the dimensions of the tile to be processed increase. The HW convolution module in [2] is confined to only 3x3 kernels. Its convolution operation relies on a tree-based structure. However, it is not flexible to incorporate convolution of any other size; neither such structure is suitable for parallelism in the convolution of an image. The architecture proposed in [3] is designed for kernels of fixed coefficients and fixed no. of input sets; it is well suited when processing continuous data streams. Moreover, the implementation is in FPGA, which does not reveal the design's actual throughput and power efficiency.

To address the challenges discussed above, it is crucial to design an architecture that can support a parallel computation of convolution operation while reducing the data movement from off-chip and should support the different sizes of the kernel. This paper contributes to the HW design of the single-precision floating-point based MAC engine, a hardware accelerator for image convolution. It is a standalone module that can be integrated into a microcontroller or SOC system for image processing applications. The aim is to develop an architecture that can independently perform 2-D convolution without relying on the host system and operate with kernels of any size rather than having a fixed-size kernel buffer. The parallel arrangement of functional units and linear indexing of the pixel data & kernel weights using a small micro-code processor (MCP) has achieved this.

While being a separate unit, it allows overcoming the limitations of SW-based image convolution. The 2-D convolution is computed in a loop-wise manner to save power. We show that the proposed sliding window algorithm can retain the boundary pixels without the need for padding. The proposed algorithm also addresses the additional local memory requirement for HW implementation of convolution, specifically when trying to exploit parallelism in the filtering process. The data movement is reduced by mapping the entire image into the internal register files (RF) of the functional units (FU). The hardware has configurable parameters that support different shapes in convolution. Meanwhile, an application specific multiplier with reduced dynamic power and a faster increment by one circuit has also been presented to optimize
the FU.

1.2 EXISTING SYSTEM

A. CHALLENGES IN IMAGE PROCESSING HW DESIGN

Several workloads, such as Image Processing, require specialized hardware to meet expected performance. However, such responsive hardware comes at the cost of flexibility, and sometimes power is sacrificed. There is a need to create a specialized HW that offers reasonable performance, power efficiency, and flexibility. Image processing calls for such specialized HW since CPUs are not optimized for such a high level of data parallelism. The traditional solution followed to exploit this data parallelism is an implementation using SIMD Units which is highly flexible but way slower. The alternative solution to this is using GPUs which provide great flexibility and are more performant than SIMD units; however, they consume too much power. Another alternative solution is deploying ASIC accelerators which will be very performant and power-efficient at the cost of flexibility since ASIC can implement only one algorithm. An Engine is an architecture with reasonable performance, flexibility, and power for a specific algorithm.

B. FLOATING POINT FORMAT

Floating-point data is commonly used in scientific images, representing measurements using 32 or 64 bits per pixel. In the rest of literature 32-bit Floating Point (FP32) is used which is a IEEE-754 standard format [4] comprised of a sign bit (S), 8 exponent bits (E) and 23 mantissa bits (M). Figure 1 shows the internals of FP32 and its equivalent value is calculated as shown in equation (1)

$$\text{Value} = (-1)^S \times 2^E \times (1.M)$$

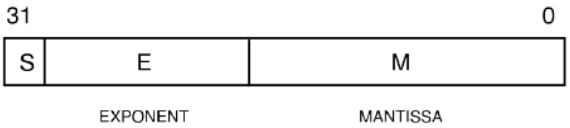


FIGURE 1. IEEE-754 single-precision floating-point format.

C. INTEGER VS FLOATING POINT FUNCTIONAL UNIT FOR IMAGE PROCESSING

Several works have been dedicated to integer functional units for image processing that processes 8-bit grayscale channel, 24b-bit RGB channels, and 32-bit RGBA channels. However, very few pieces of literature are available related to floating point image processing hardware. Floating-point pixels have application in HDR Imaging in mobile cameras, satellite imaging as elevation of the land surface has different dynamic ranges. The filter being used can also be categorized into integer-order and fractional-order filters. Fractional order filters collect more image details in their high-frequency region than integer ones and provide better noise sensitivity. Due to this very advantage, floating-point image processing is also considered a suitable option for biomedical applications. Floating-point images can use 32-bit floats or 16-bit half precision format, but half-precision does not have as much dynamic range as required in practical applications. Floating point pixels have some advantages over Integer pixels but have extra memory overhead.

1.3 PROPOSED SYSTEM

The proposed sliding window algorithm can retain the boundary pixels without the need for padding. The proposed algorithm also addresses the additional local memory requirement for HW implementation of convolution, specifically when trying to exploit parallelism in the filtering process. The data movement is reduced by mapping the entire image into the internal register files (RF) of the functional units (FU). The hardware has configurable parameters that support different shapes in convolution. Meanwhile, an application specific multiplier with reduced dynamic power and a faster increment by one circuit has also been presented to optimize the FU.

2.LITERATURE SURVEY

"High-performance SIFT hardware accelerator for real-time image feature extraction", F.-C. Huang, S.-Y. Huang, J.-W. Ker, and Y.-C. Chen,

Feature extraction is an essential part in applications that require computer vision to recognize objects in an image processed. To extract the features robustly, feature extraction algorithms are often very demanding in computation so that the performance achieved by pure software is far from real-time. Among those feature extraction algorithms, scale-invariant feature transform (SIFT) has gained a lot of popularity recently. In this paper, we propose an all-hardware SIFT accelerator-the fastest of its kind to our knowledge. It consists of two interactive hardware components, one for key point identification, and the other for feature descriptor generation. We successfully developed a segment buffer scheme that could not only feed data to the computing modules in a data-streaming manner, but also reduce about 50% memory requirement than a previous work. With a parallel architecture incorporating a three-stage pipeline, the processing time of the key point identification is only 3.4 ms for one video graphics array (VGA) image. Taking also into account the feature descriptor generation part, the overall SIFT processing time for a VGA image can be kept within 33 ms (to support real-time operation) when the number of feature points to be extracted is fewer than 890.

"Convolution accelerator designs using fast algorithms", Y. Zhao, D. Wang, and L. Wang,

Convolutional neural networks (CNNs) have achieved great success in image processing. However, the heavy

computational burden it imposes makes it difficult for use in embedded applications that have limited power consumption and performance. Although there are many fast convolution algorithms that can reduce the computational complexity, they increase the difficulty of practical implementation. To overcome these difficulties, this paper proposes several convolution accelerator designs using fast algorithms. The designs are based on the field programmable gate array (FPGA) and display a better balance between the digital signal processor (DSP) and the logic resource, while also requiring lower power consumption. The implementation results show that the power consumption of the accelerator design based on the Strassen–Winograd algorithm is 21.3% less than that of conventional accelerators.

"Design of a convolutional two-dimensional filter in FPGA for image processing applications", G. Licciardo, C. Cappetta, and L. Di Benedetto,
Exploiting the Bachet weight decomposition theorem, a new two-dimensional filter is designed. The filter can be adapted to different multimedia applications, but in this work it is specifically targeted to image processing applications. The method allows emulating standard 32 bit floating point multipliers using a chain of fixed point adders and a logic unit to manage the exponent, in order to obtain IEEE-754 compliant results. The proposed design allows more compact implementation of a floating-point filtering architecture when a fixed set of coefficients and a fixed range of input values are used. The elaboration of the data proceeds in raster-scan order and is capable of directly processing the data coming from the acquisition source thanks to a careful organization of the memories, avoiding the implementation of frame buffers or any aligning circuitry. The proposed architecture shows state-of-the-art performances in terms of critical path delay, obtaining a critical path delay of 4.7 ns when implemented on a Xilinx Virtex 7 FPGA board.

"A compression method for arbitrary precision floating-point images",C. Manders, F. Farbiz, and S. Mann,
The paper proposes a method of compressing floating-point images of arbitrary precision. The concept of floating-point images is used frequently in such areas as high dynamic range imaging, where pixel data stored as 8 or 12-bit integers are insufficient. The compression scheme presented in the paper organizes the floating-point data in a manner such that already existing compression algorithms such as JPEG or Zlib compression may be used once the data re-organization has taken place. The paper compares the result to a popular (but restrictive) form of image compression, openEXR, and shows significant gains over this format. Furthermore, the compression scheme presented is scalable to deal with floating point images of arbitrary precision.

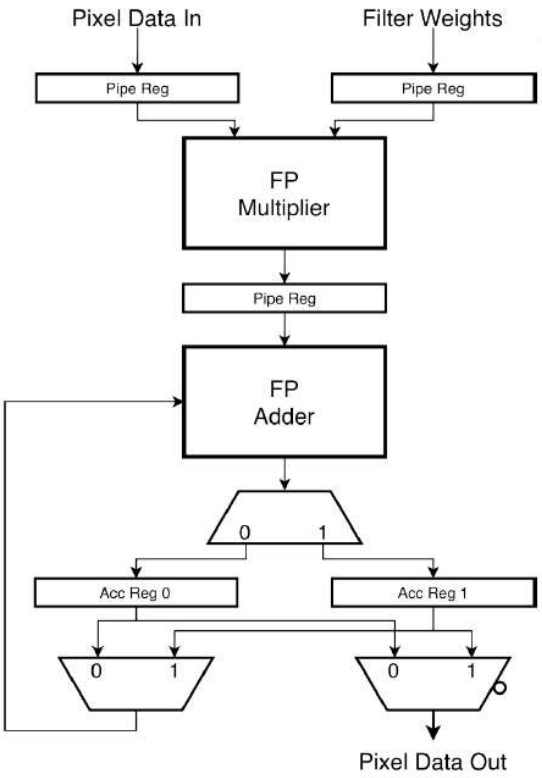
"A study on convolution using half-precision floating-point numbers on GPU for radio astronomy deconvolution", M. Seznec, N. Gac, A. Ferrari, and F. Orieux
The use of IEEE 754-2008 half-precision floating-point numbers is an emerging trend in Graphical Processing Units' architecture. Being such a compact way of representing data, its use may speed up programs by reducing the memory bandwidth usage and allowing hardware designers to fit more computing units within the same die space. In this paper, we highlight the acceleration offered by the use of half floating-point numbers over different implementations of the same operation, a 2D convolution. We show that even though it may lead up to a significant speed-up, the degradation brought by this new format is not always negligible. Then, we choose a deconvolution problem inspired by the SKA radio-telescope processing pipeline to show how half floats behave in a more complex application.

"Design of efficient floating point convolution module for embedded system", J. Li, X. Zhou, B. Wang, H. Shen, and F. Ran
The convolutional neural network (CNN) has made great success in many fields, and is gradually being applied in edge-computing systems. Taking the limited budget of the resources in the systems into consideration, the implementation of CNNs on embedded devices is preferred. However, accompanying the increasingly complex CNNs is the huge cost of memory, which constrains its implementation on embedded devices. In this paper, we propose an efficient, pipelined convolution module based on a Brain Floating-Point (BF16) to solve this problem, which is composed of a quantization unit, a serial-to-matrix conversion unit, and a convolution operation unit. The mean error of the convolution module based on BF16 is only 0.1538%, which hardly affects the CNN inference. Additionally, when synthesized at 400 MHz, the area of the BF16 convolution module is 21.23% and 18.54% smaller than that of the INT16 and FP16 convolution modules, respectively. Furthermore, our module using the TSMC 90 nm library can run at 1 GHz by optimizing the critical path. Finally, our module was implemented on the Xilinx PYNQ-Z2 board to evaluate the performance. The experimental results show that at the frequency of 100 MHz, our module is, separately, 783.94 times and 579.35 times faster than the Cortex-M4 with FPU and Hummingbird E203, while maintaining an extremely low error rate.

"MacSim: A MAC-enabled high-performance low-power SIMD architecture", T. Geng, L. Waeijen, M. Peemen, H. Corporaal, and Y. He
Single-Instruction-Multiple-Data (SIMD) architectures, which exploit data-level parallelism (DLP), are widely used to achieve high-performance and low-power computing. In most of streaming applications, such as CNN-based detection and recognition, color space conversion and various kinds of filters, multiply-accumulate is one of the most important and expensive operations to be executed. In this paper, we propose a high-performance low-power SIMD architecture with advanced multiply accumulator (MAC) support (MacSim) to improve the computational efficiency. In addition, a smart loop tiling scheme is proposed. To support this tiling even further, the MAC unit is equipped with multiple accumulator registers. According to the Design Space Exploration (DSE) of the proposed MAC unit, a MAC instance with four accumulator registers (MAC4reg) is selected as a good choice for target kernels. In this paper, a 64-PE 16-bit (processing element) SIMD instance without MAC support is taken as the baseline. For a head-to-head comparison, a 64-PE 16-bit SIMD with MAC4reg (MacSim4) and the baseline SIMD are all implemented in HDL and synthesized with a TSMC 40nm low-power library. Five streaming application kernels are mapped to both architectures. Our experimental results show with MAC4reg the runtime and energy consumption are reduced up to 38% and 42% respectively. Besides, a 4-layer CNN-based detection application is also fully mapped onto the proposed MacSim4. Working at 950MHz, MacSim4 reaches a throughput of 62.4 GOPS, which meets the requirement of real-time (720P HD, 30fps) detection. The energy consumption per PE per operation is very low, 4.7pJ/Op excluding SRAM (Static Random Access Memory) and 4.8pJ/Op including a 2k-entry SRAM bank. As a prototype, the proposed SIMD is mapped into an FPGA and can run all the kernels.

"Ultra-low-power adder stage design for exascale floating point units", A. A. Del Barrio, N. Bagherzadeh, and R. Hermida
Currently, the most powerful supercomputers can provide tens of petaflops. Future many-core systems are estimated to provide an exaflop. However, the power budget limitation makes these machines still unfeasible and unaffordable. Floating Point Units (FPUs) are critical from both the power consumption and performance points of view of today's microprocessors and supercomputers. Literature offers very different designs. Some of them are focused on increasing performance no matter the penalty, and others on decreasing power at the expense of lower performance. In this article, we propose a novel approach for reducing the power of the FPU without degrading the rest of parameters. Concretely, this power reduction is also accompanied by an area reduction and a performance improvement. Hence, an overall energy gain will be produced. According to our experiments, our proposed unit consumes 17.5%, 23% and 16.5% less energy for single, double and quadruple precision, with an additional 15%, 21.5% and 14.5% delay reduction, respectively. Furthermore, area is also diminished by 4%, 4.5 and 5%.

3. PROPOSED METHODOLOGY
BLOCK DIAGRAM



Architecture of floating-point MAC unit

3.2 MODULE EXPLANATION:
A. CONVOLUTION OPERATION
A 2-D convolution collects necessary data from image pixels; mathematically, it is an element-wise multiplication of image kernel (filter weights) and input feature map (pixel data). This process of extracting essential features in image pixels is called Convolution, a fundamental building block of any Convolutional neural network. Convolution operator is denoted by `_', equation (2) & (3) expresses the entire convolution operation.

$$y[m,n] = x[m,n] * h[m,n]$$
$$y[m,n] = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x[i,j] \times h[m-i, n-j]$$

where x is function from image location to pixel value, h is filter applied to the image, and y is filtered image. There are multiple ways to compute image convolution, such as sliding window transform, shift-multiply add Fourier transform. A pictorial representation of the Sliding window algorithm is shown in Figure 2. The stencil of kernel matrix slides over the image matrix so that the target pixel appears precisely in the center and then performs multiply accumulate for that particular region; this goes on till the whole image area is covered.
The convolution is indeterminate at the boundaries of the image, an i x i image, and k x k kernel produces an (i - k + 1) x (i - k + 1) image. These can be prevented by padding the image with a layer of zeros or sometimes any arbitrary value. To prevent the loss of pixels at the boundary of the image, the Algorithm in Algorithm 1 is modified to incorporate virtual zero padding. It does not actually use zeros for padding; instead arranges the remaining pixels ofthe zero-padded [I]kxk matrix with its corresponding kernel coefficients for the current window.

Algorithm 1 Sliding Window for 2-D Convolution
With an Image matrix $[I]_{ixi}$ and Kernel Matrix $[K]_{k\,x\,k}$
for cr**in range** (0, orows-1)
    for cc **in range** (0, ocols-1)
        for i **in range** (cr, krows + cr-1)
            for j **in range** (cc, kcols + cc-1)
                op[cr][cc] + = ip[i][j] x kernel[i-cr][j-cc];
where, orows = irows - krows + 1;
ocols = icols - kcols + 1,
irows, icols, krows, kcols are the configurable parameters

Now the convolution of i x i images and k x k kernel with p level of padding produces an (i - k + 2p + 1) x (i - k + 2p + 1) image. A typical HW implementation of convolution with padding would require storing this padding value as we use image partitioning; the no. of padding values that need to be stored increases. The proposed modified algorithm shown in Algorithm 2 brings power saving as well since there is no need for storing zeros for padding and wasting the operation cycle in the calculation of redundant element whose value is going to be zero. Table 1 shows the memory depth required in different scenarios; from Table 1, it is evident that Algorithm 2 can save significant local memory space for HW based convolution.

Algorithm 2 Modified Sliding Window With Virtual Zero
Padding
With an Image matrix $[I]_{ixi}$ and Kernel Matrix $[K]_{k\,x\,k}$
for cr**in range** (0, irows-1)
    for cc **in range** (0, icols-1)
        for i **in range** (cr-p_b, krows + cr-p-a-1)
            for j **in range** (cc -p_d, kcols + cc-p-c-1)
                op[cr][cc] + = ip[i][j]_kernel[i-cr+p][j-cc+p];
Where p is number of padding level

$$a = \begin{Bmatrix} 1, cr == irows - 1 \\ 0, otherwise \end{Bmatrix}, \quad c = \begin{Bmatrix} 1, cc == icols - 1 \\ 0, otherwise \end{Bmatrix}$$
$$b = \begin{Bmatrix} 0, cr == 0 \\ 1, otherwise \end{Bmatrix}, \quad d = \begin{Bmatrix} 0, cc == 0 \\ 1, otherwise \end{Bmatrix}$$

B. FLOATING MAC UNIT (FPMAC)
FPMAC composes the data path of the engine. It is responsible for performing a two-stage pipelined multiplication & accumulation and producing an output pixel. A signal from controller controls the magnitude of this accumulation, and it has two accumulator registers to support continuous multiplication & accumulation also shown in Figure 3. The use of a second accumulator allows the accumulation of the new incoming window data without breaking the flow of MAC operation. The MAC is fed with the kernel weights and the input pixel data coming from the register file. After each k x k cycle, the target pixel surrounded by k^2-1 pixels is produced into a filtered pixel. FU performs convolution in a multiplication and accumulation manner till it covers every element of the current window, the result is saved in one of the accumulation registers, and for the next target pixel, the other accumulator register becomes active.

C. FLOATING POINT ADDER
The following are the steps to be pursued to add two single precision floating point numbers, block diagramof FP Adder is show in Figure 4. Let X and Y be the two numbers to be added.
i. First, the exponents e_x and e_y are compared and the tentative exponent is the larger of these two.
ii. The mantissa with smaller exponent is shifted right by 'e_x - e_y' position and set as 2nd operand to addition/subtraction while the other mantissa is set as 1st operand.
iii. The residual mantissa bits left out due to shifting are used for guard bit, rounding bit and sticky bit (GRS) computation.

iv. The aligned mantissas are added or subtracted depending upon the sign of both operands and sign of result is set to the sign of largest exponent operand.
v. The sum is normalized in two cases:
- When there is carry out of significand addition, then the mr is shifted right by one position through the GRS bits and the tentative exponent is incremented by one.
- When there is cancellation in significand, in this case the number leading zeros `z' in the tentative significand is counted or anticipated and the tentative significand is shifted left by `z' position where as tentative exponent is set to e_r- z. Accordingly G, R, S bits also gets modified.
vi. The guard bit, rounding bit and sticky bit changes during normalization based on carry out bit of addition. And at last rounding logic decides the rounding status of mantissa.

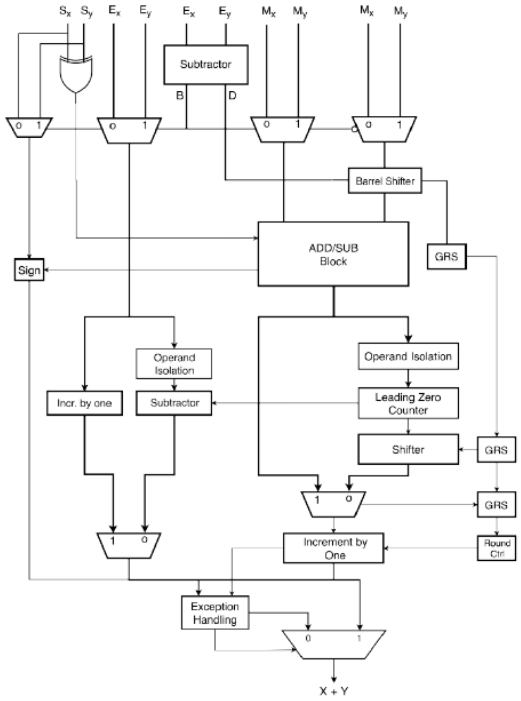


FIGURE 4. Single precision floating point adder.

The component honors overflow and underflow, and accordingly settles the output value. One of the common operations in floating-point arithmetic is increment by one used for normalization or rounding purposes. Increment by One is part of the critical path and hence should be fast. A typical way to increment a binary number is to use a carry look-ahead adder but it offers a great amount of delay. Nevertheless, there is a more efficient way to do this. One way to accomplish increment by one is to use binary to the excess-1 converter, but it only works for fewer bits and will face higher propagation delay for many bits. The design in [4] suggests using a carry-propagate adder for this. Another way is trailing one detection of the input word, then special encoding followed by partial complementing. Special encoding encodes the results of trailing one detector to generate a string `S'. All the bit positions in `S' which follows trailing zero along with trailing zero position itself are set to one. This work proposes a new method that works by producing a mask string `S' at first, without the need of trailing one detector and then passing it through ex-or gates along with input bits for partial complementing. This complete method of incrementing a 24-bit binary number by one is shown in Figure 5.

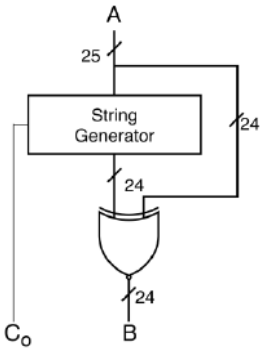


FIGURE 5. Proposed increment by one circuit.

A string expression to determine the one-hot representation is mentioned in [15], where only those bits in string expression that follow the leading one are set to `1'. To derive the string S required for proposed circuit, the logical expression illustrated in [15] is modified, the ith bit of S denoted as Si, 0 < i < n-1 is defined as follows:

$$S_i = A_{n-1} \bullet A_{n-2} \bullet \ldots\ldots \bullet A_{i+2} \bullet A_{i+1} \bullet C_0$$

where . denotes the logical AND operation. Ai is the input to increment by one circuit. The string S thus can be defined as:

$$S_0 = C_0$$
$$S_i = 1, \quad \text{if } i \leq R$$
$$0, \quad \text{otherwise}$$

where R-1 is the position of trailing one in binary word A and C0 is carry-in (control bit). Translating the equation (4) & (5) into the gate-level circuit will result in a long chain of AND gates. It can be avoided when the AND gate chain is broken into identical blocks, each generating 4-bits of string S as shown in Figure 6, and the output of each block is isolated using the MSB from the previous block.

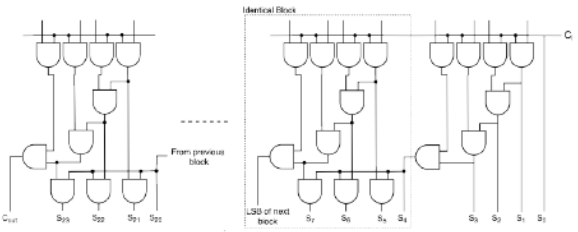


FIGURE 6. Proposed 24-bit string generator for increment by one circuit.

The advantage of the latter method is that it buffers the input word when the incremented value is not desired and eliminates the need for a multiplexer in the normalization or rounding section.

D. FLOATING POINT MULTIPLIER

Below are the steps to be followed to multiply two single precision floating point numbers, block diagram of FP multiplier is shown in Figure 7. Let X and Y be the two numbers to be multiplied.

i. The exponents of both operands are added and then subtracted with bias `127'.

ii. The significand $1.m_x$ and $1.m_y$ are multiplied and the product of two significands will be less than `4'. The lower 23-bits are forwarded as pre-normalized mantissa and rest of the bits (residual products) are used for sticky bit computation. Few bits are used for guard and round bits.

iii. As the step of normalization the Least Significant Bit (LSB) of product is checked for bit `1'and if so pre-normalized mantissa is shifted right by one position.

In parallel of Normalization, rounding logic decides the rounding status of mantissa.

iv. The carry out bit from rounding of normalized mantissa, along with LSB of product decides whether to increment the tentative exponent or not.

v. The value is also checked for overflow before normalization and rounding while it is checked for underflow after rounding.

The significand multiplication itself is fixed-point multiplication and has a significant share in the power consumption of MAC. The relatively close elements in the image matrix are almost equal in value and do not require to be fully multiplied again and again. Based on this observation, this work proposes a low power multiplier (LPM) specific to the image processing application.

From the population graph of the 7x7 pixel matrix shown in Figure 8(c), it can be observed that the consecutive pixels have a small dynamic range, and the upper bits of these pixels toggle infrequently; the flat line curve in the graph depicts this exactly.

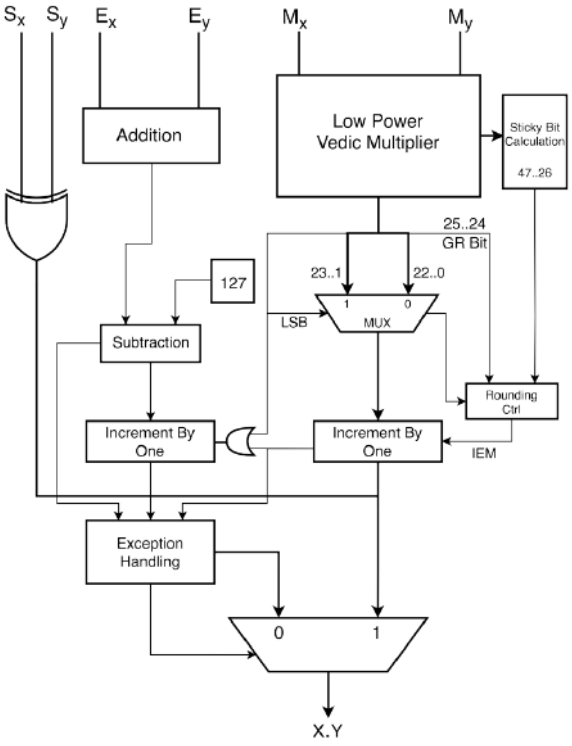
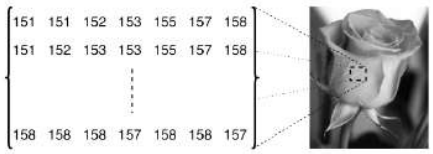


FIGURE 7. Single precision floating point multiplier.



(a) (b)

(c)

FIGURE 8. (a) A 7 x 7 pixel matrix, (b) A grayscale Image, (c) Graph of consecutive image pixels of a small region.

This infrequent change gives an opportunity where the ith pixel is checked for its dynamic range with respect to the iC1th pixel. If the upper bits of these pixels are equal, the multiplier disables switching in the higher section of the multiplier using operand isolation. The block diagram of this method is shown in Figure 9. The structure of the Vedic Multiplier based on

UrdhvaTriyakbhyam consists of several sections, each working independently on the multiplication of different bit combinations of two operands; this allows disabling the unused section of multiplier dynamically, thereby saving significant power without affecting the working of the rest of the multiplier.

The block diagram in Figure 9 depicts the implementation of LPM using the Vedic Multiplication method, where a 2-stage line buffer for both the operands compares the two consecutive operands and saves significant power without affecting the delay of the data path.

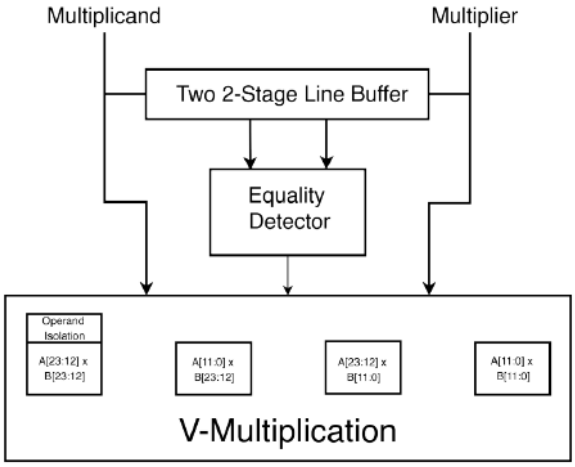


FIGURE 9. Proposed low power vedic multiplier.

E. ENGINE ARCITECTURE

The engine uses FPMAC as the basis of the functional Unit whose operation is controlled by the controller. The MAC Engine architecture is shown in Figure 10. To accelerate the sliding window algorithm, the architecture employs an array of 2n functional units, each working in parallel, where `n' is the scalable factor for the array. The parallelism of the 2-D convolution on an image is based on a basic approach of partitioning the image [17] into blocks of rows and columns, each of these blocks forming a square matrix. The architecture shown in Figure 10 partitions the image into 4 blocks. For a given image, each FPMAC out of 2n functional units performs the 2-D convolution only on a single block, thereby exploiting parallelism and acceleration of image convolution.

F. MEMORY SYSTEM

The primary concern of Algorithms 1 & 2 is that the degree of data replication of input pixels is very high, which leads to complex and costly memory access patterns when the image pixels are stored off-chip. Inside the functional unit, a deep internal memory (register file) is merged directly with FPMAC to avoid the frequent data travel from the host memory; this will reduce the interconnect power related to multiple memory access of the overlapped data when pixels are stored off-chip. This tight coupling of register file with FU will come alive at the physical design of the architecture.

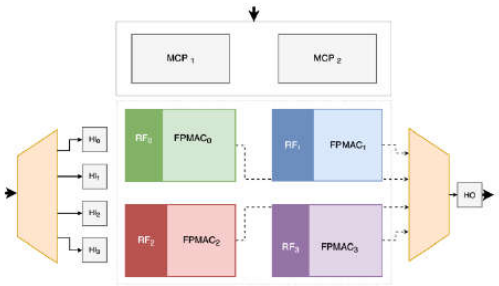


FIGURE 10. Architecture of FP MAC engine.

And the power reduction in data access will be achieved without the need for data/pixel reuse but at the cost of a larger internal storage unit. However, various existing designs also have large on-chip/internal memory to minimize the energy. To support the image partitioning within the engine, pixel transfer from external memory to the engine and within the engine plays a crucial role. The data written into the external DRAM by the host CPU is collected one by one at the engine via a buffer. The pixel data distribution among the functional units analogous to time division de-multiplexing (TDDM). Using this method helps eliminate the data hindrance to the other functional units, which otherwise would occur if other functional units are written into only after the writing of image block into the first functional unit has finished. Respective data to each functional unit are transferred one by one in consecutive cycles via data handlers that monitor each register file's write addresses. To implement the TDDM like data distribution, a four-level nested for-loop as mentioned in proposed Algorithm 3 is used which can also be viewed as image partitioning algorithm. Figure 11 shows how the pixel data for a 4 x 4 image is partitioned and distributed among the functional units using the mentioned algorithm.

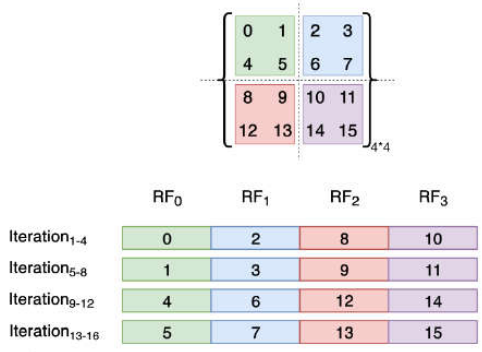


FIGURE 11. Pixel data distribution of 4 x 4 matrix using Algorithm 3.

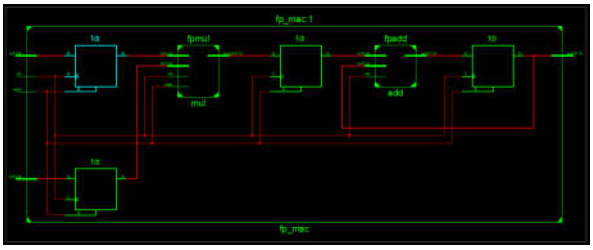
Each element of the incoming image block is flattened into a single row vector and stored sequentially into vertical register file, also shown in Figure 12; this makes indexing the desired pixel easier. The other advantage of this method is that the convolution is not fixed to particular kernel size but instead it can support convolution with any size of the kernel. The maximum size of the kernel will be decided at the design time by describing the depth of the register file for storing the kernel. The data access pattern of content stored in this RF follows the modified sliding window algorithm mentioned in Algorithm 2. Each processed pixel value from the 2n units is picked up and multiplexed out to the external memory via a data handler.

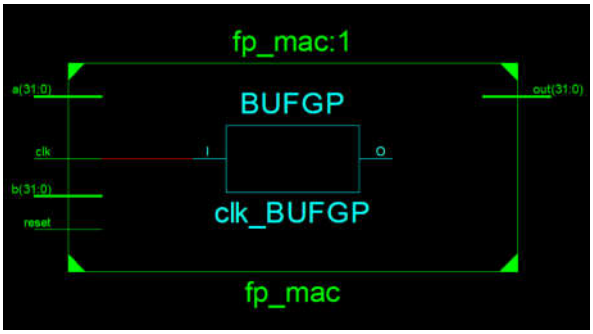
Algorithm 3 Image Partitioning Algorithm
Ina Image matrix $[I]_{ixi}$
for p **in range** (0, t-1)
for q **in range** (0, t-1)
for fu_r**in range** (0, log2(fu)-1)
for fu_c**in range** (0, log2(fu)-1)
read = (rows x (t x fu_r + p)) + (t x fu_c + q);
Where t = *i*/log2 (*fu*),
fu is number of functional units/ no. of blocks of partitioned image. `fu' should be 4 or 16 or 64.

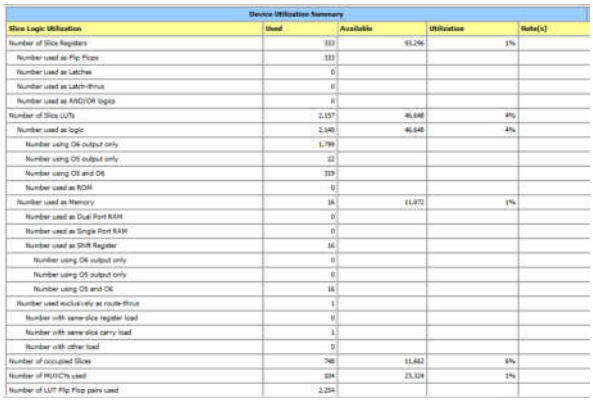
G. MICROCODE PROCESSOR

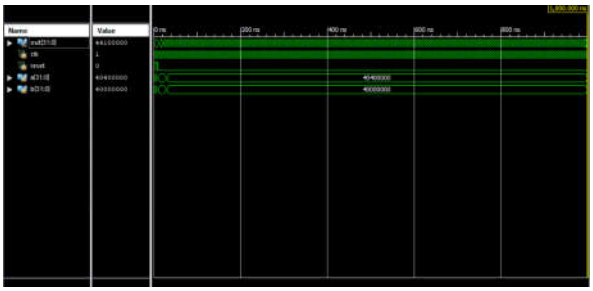
The host processor can implement the modified convolutionalgorithm presented in Section III.A. Rather than relying onthe host processor to compute memory addressing offsetsfor register files, the MAC Engine exploits a microcode processor similar to one presented in [18]. The microcode processor (MCP) is built as a part of this image convolution system. Information related to image size & kernel size is received from the host system, and its sets various flag to indicate completion of the image read operation and convolution. The microcode processor architecture is based on a small set of Instructions that implements the nested loops of the algorithm.
The implementation of this ISA is a single cycle execution architecture that has 8-bit instructions such as MOV, ADD, INCREMENT, JUMP, and COMPARE. The corresponding opcodes related to Algorithms 2 & 3 are stored in the two separate ROMs. Two different processors fetch these opcodes each cycle and perform desirable operations to obtain the address values for the external and internal memory. The whole operation of the MAC engine is orchestrated by these two MCPs. MCP 1 controls the data flow from external memory to internal memory and maps the partitioned image block to its respective functional unit, whereas MCP 2 controls data flow in the internal engine, from RF to FPMAC.

5. SIMULATION RESULTS:









6.CONCLUSION

In order to enhance the HW-based convolution, a convolution module improved at the algorithmic and architectural levels is given in this research. The picture results for a floating-point pixel data are not displayed in the earlier research connected to HW implementation of convolution. Additionally, the padding in image convolution is not covered in such literatures. When the host system grants the MAC engine's asynchronous overall architecture Direct Memory Access to the main memory, the MAC engine may operate entirely autonomously. A lot of transaction power may be saved, and internal memory is used more effectively, if the picture is stored off-chip or directly mapped onto it (register files).

The benefit of this method in this study is that leveraging parallelism becomes simple, and utilizing the low power multiplier, substantial dynamic power may be saved. Using floating-point picture data, this study demonstrates how the 2-D image convolution may realistically be padded with zeros, partitioned, and accelerated at the hardware level. The resultant image will have higher noise performance.

7. FUTURE SCOPE

Future work willconsider more complexsystems and theimplementation of double precision floating-point calculationengines.

REFERENCES:

[1] F.-C. Huang, S.-Y. Huang, J.-W. Ker, and Y.-C. Chen, ``High-performance SIFT hardware accelerator for real-time image feature extraction," IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 3, pp. 340351, Mar. 2012.

[2] Y. Zhao, D. Wang, and L. Wang, ``Convolution accelerator designs using fast algorithms," Algorithms, vol. 12, no. 5, p. 112, May 2019.

[3] G. Licciardo, C. Cappetta, and L. Di Benedetto, ``Design of a convolutional two-dimensional filter in FPGA for image processing applications," Computers, vol. 6, no. 2, p. 19, May 2017.

[4] J.-M. Müller, N. Brunie, F. de Dinechin, C.-P. Jeannerod, M. Joldes, V. Lefévre, G. Melquiond, N. Revol, and S. Torres, *Handbook of Floating- Point Arithmetic*. Basel, Switzerland: Birkhäuser Science, 2018.

[5] C. Manders, F. Farbiz, and S. Mann, ``A compression method for arbitrary precision _oating-point images," in *Proc. IEEE Int. Conf. Image Process.*, Oct. 2007, pp. IV-165_IV-168.

[6] M. Seznec, N. Gac, A. Ferrari, and F. Orieux, ``A study on convolution using half-precision _oating-point numbers on GPU for radio astronomy deconvolution," in *Proc. IEEE Int. Workshop Signal Process. Syst. (SiPS)*, Oct. 2018, pp. 170_175.

[7] J. Li, X. Zhou, B.Wang, H. Shen, and F. Ran, ``Design of efficient floating point convolution module for embedded system," *Electronics*, vol. 10, no. 4, p. 467, Feb. 2021.

[8] T. Geng, L. Waeijen, M. Peemen, H. Corporaal, and Y. He, ``MacSim: A MAC-enabled high-performance low-power SIMD architecture," in *Proc. Euromicro Conf. Digit. Syst. Design (DSD)*, Aug. 2016, pp. 160_167.

[9] O. H. Moustafa and S. M. Ismail, ``FPGA-based floating point fractional order image edge detection," in *Proc. 15th Int. Comput. Eng. Conf. (ICENCO)*, Dec. 2019, pp. 91_94.

[10] A. A. Del Barrio, N. Bagherzadeh, and R. Hermida, ``Ultra-low-power adder stage design for exascale _oating point units," *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 3s, pp. 1_24, Mar. 2014.

[11] W. Qadeer, R. Hameed, O. Shacham, P. Venkatesan, C. Kozyrakis, and M. A. Horowitz, ``Convolution engine: Balancing efficiency & flexibility in specialized computing," in *Proc. 40th Annu. Int. Symp. Comput. Archit.*, Jun. 2013, pp. 24_35.

[12] R. Mudassir, M. Anis, and J. Jaffari, ``Switching activity reduction in low power booth multiplier," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2008, pp. 3306_3309.

[13] T. Ahn and K. Choi, ``Dynamic operand interchange for low power," *Electron. Lett.*, vol. 33, no. 25, pp. 2118_2120, Dec. 1997.

[14] J.-F. Lin, C.-Y. Chan, and S.-W. Yu, ``Novel low voltage and low power array multiplier design for IoT applications," *Electronics*, vol. 8, no. 12, p. 1429, Nov. 2019.

[15] G. Dimitrakopoulos, K. Galanopoulos, C. Mavrokefalidis, and D. Nikolos, “Low-power leading-zero counting and anticipation logic for high-speed floating-point units," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 7, pp. 837_850, Jul. 2008.

[16] A. Deepa and C. N. Marimuthu, ``Design of a high speed Vedic multiplier and square architecture based on Yavadunam Sutra," *S adhan a*, vol. 44, no. 9, pp. 1_10, Aug. 2019.

[17] D. G. Bailey, ``Image processing," in *Design for Embedded Image Processing on FPGAs*, vol. 8. Hoboken, NJ, USA: Wiley, 2011, ch. 1, sec. 1, pp. 14_17.

[18] F. Conti, P. D. Schiavone, and L. Benini, ``XNOR neural engine: A hardware accelerator IP for 21.6-fJ/op binary neural network inference," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2940_2951, Nov. 2018.

AUTHORS PROFILES

BOMMADENI CHARAN RAJ is a proficient PG Scholar, Master of Technology, Dept of ECE,VLSI & ES, Siddhartha Institute of Engineering & Technology (SIET), JNTUH, Ibrahimpatnam,T.S. Along with initial degrees of Bachelor of Technology in Electronics and Communication Engineering (ECE) from Mahatma Gandhi Institute of Technology(MGIT), JNTUH,Gandipet, Hyderabad, T.S. He is currently bounded and working as a Hardware Engineer in Sigma Advanced Systems Private Limited, Hardware Park, Shamshabad, Hyderabad, T.S. He is having 2+ years of experience in the Design of schematics for complicated military application PCB's and systems. He could able to write RTL and VHDL coding procedures and perform simulations for the FPGA devices like, Artix, Virtex, Zynq, Cyclone, Arria, etc. He had expertise in the typical FPGA coding platforms such as Xilinx Vivado, Intel Quartus.

Dr. M. PUSHPALATHA working as an Associate professor for Siddhartha Institute of Engineering and Technology Hyderabad. She Completed Ph.D. in the area of wireless communication, from Sri Satya Sai University of technology and medical Science, Bhopal, Madhya Pradesh. She received his Doctoral Degree in the year 2021 from SSSUTMS Sehore, Bhopal, Madhya Pradesh from school of Engineering. She has completed herMaster's degree with specialization in Digital Electronics and communication systems from Mahaveer Institute of Science & Technology affiliated to JNTU Hyderabad in the year 2008. Prior to this has completed bachelor's degree in Electronics and communication engineering from G. Narayanamma Institute of Technology and Science affiliated to JNTU Hyderabad.

Her carrier started as a lecturer and has total **15 years** of experience in teaching field. Out of which 3 years worked as Assistant Professor and 4 years as Senior Assistant Professor in a single organization named Abhinav Hi-Tech College of Engineering. She is a permanent member of ISTE and IETE.She published **30** papers in National and International journals and published papers in international conferences, **2 SCI Journals**. She has **3 patents**. She attended and also conducted many workshops and conferences. She is very much interest to do research on DECS, wireless technology and signals.

Dr.T.KRISHNARJUNA RAO working as an Associate Professor for Siddhartha Institute Of Engineering and Technology Hyderabad in the department of ECE. He completed PhD in Low power VLSI Technology Domain. He received his Doctoral Degree in the year 2021 from SSSUTMS Sehore,Bhopal, Madhya Pradesh from school of Engineering. He got B.Tech degree from ADAM's Engineering college in ECE branch from Khammam.. He got M.Tech degree from ANURAG Engineering College with VLSI System Design from Kodad. He is the permanent member of ISTE and IETE. He has **12 years** of teaching experience in engineering field. He attended many workshops related to VLSI and Low power VLSI. He **published 24** papers in various National and International journals, attended so many conferences. He has **2** patent publications. He has very much interest to do research on VLSI Technology and Design, communication systems and Digital Electronics and Wireless Communication.