IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 1, June 2023

Low Power Implementation of Mitchells Approximate Logarithimic Multiplication for Convolutional Neural Networks

Kanuparthi Venkata Siva Prasad Reddy, Pamuluru Ganesh, Pala Mohan Sai, Gorla Prateesh
Department of Electronics and Communication Engineering
Prathyusha Engineering College, Thiruvallur, Tamil Nadu, India

Abstract: *Approximate computing (AC) is an emerging paradigm that leverages the inherent error tolerance of many applications—such as image recognition, multi- media processing, and machine learning (ML)—to allow some accuracy to be traded off to save energy consumption. AC techniques can be applied at both the circuit and/or architecture levels, possibly in coordination with software- level techniques. Multiplication is one of the most resource- and power-hungry operations in many error-tolerant computing applications, such as image processing, neural networks (NN), and digital signal processing (DSP). In this research project, we focus on the design and implementation of hardware-efficient approximate computing circuits, aiming to simplify the multiplication operation and/or to reduce the number of required multiplications. Two 4×4 approximate multiplier designs are proposed in which approximation is employed in the partial product reduction tree, the most expensive part of the design of a multiplier. The two proposed designs are then used to construct larger approximate multipliers. Multiplication is the computational bottleneck in NNs. For the first time, we attempt to find the critical features in an approximate multiplier that make it superior to others for use in a NN. Inspired by the insight that adding small amounts of noise can improve the performance of NNs, we replaced the ex- act multipliers in two representative NNs with 600 approximate multipliers and then experimentally measured the effect on classification accuracy. Interestingly, some approximate multipliers improved the performance of NNs. Insight into which features of an approximate multiplier make it superior to others in the NN applications was gained by training a statistical predictor that anticipates how well a given approximate multiplier is likely to work in a NN application. In the logarithmic number system (LNS) the multiplication operation is converted into simple shift and addition operations. We have proposed a novel exact leading-one detector (LOD) to speed up the calculation of the base-2 logarithm of the input operands to a logarithmic multiplier. In addition, since the logarithmic multipliers that use LODs always underestimate the actual multiplication product, a nearest-one detector (NOD) is proposed for a logarithmic multiplier that has a double-sided error distribution. Finally, we investigate the design of multiply-accumulate (MAC) units. An approximate logarithmic MAC (LMAC) unit is proposed for the first time. Furthermore, a soft-dropping low-power (SDLP) architecture is specifically designed for convolutional neural networks (CNNs) that, unlike the existing accelerators that simplify the multiplication/addition operations, reduces the number of required multiplications. The SDLP takes advantage of the spatial dependence between the input image pixels and skips some of the multiplications during the convolution operation and, thereby, reduces the energy consumption of the CNN inference calculation*

Keywords: Approximate computing

I. INTRODUCTION

The discontinuation of Dennard scaling and the fading of Moore's law have motivated the coming forth of new devices, architectures, and design techniques for computing. Nowadays, power and energy efficiency have become significant design concerns for modern computing systems. On the other hand, current applications and workloads, such as image

IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.301

Volume 3, Issue 1, June 2023

processing, computer vision, graphics, machine learning, data mining, and financial and physical simulations, are part of a set of applications classified as Recognition, Mining, and Synthesis, which have been reported as error-tolerant. This means that even in the presence of deliberately introduced errors, these applications produce acceptable results as a golden result does not exists, the application deals with noisy input data, or even it presents iterative refinement. approximate computing techniques have been reported in the literature. Still, these show the need to exploit this design paradigm to improve computation efficiency at all layers where computation occurs. Recent work has proposed to exploit inherent resilience to errors in applications by using approximate accelerators. In general, hardware accelerators have reported significant benefits for reducing energy consumption, and they have been used to overcome the utilization wall challenge. In a nutshell, an accelerator is used to offload a highly-frequent and compute-intensive section of an application to dedicated hardware, while a host processor executes the rest of the application. Accelerators can be in the form of a GPU, a DSP, or a specialized FPGA design. From the approximate computing perspective, approximate accelerators exploit error resilience as frequently-executed, but error-tolerant sections of an application are performed by dedicated approximate hardware designs. One approach proposed is to implement these accelerators as neural networks and take advantage of the approximate nature of the results produced by this computational model. Another approach proposes the usage of approximate arithmetic circuits to replace exact calculations in hardware accelerator designs. Nevertheless, many approximate adders and multipliers have been reported in the literature. For an ongoing number of such approximate arithmetic circuits, and considering their usage in building approximate designs, such as approximate accelerators, a question arises: given a design for an error-tolerant application and a set of approximate components, which approximate arithmetic circuits
should be used to minimize the computational effort, for instance, the required area, delay, power, or energy, while satisfying a defined accuracy? Traditional approaches required exhaustive synthesis and simulation of by-hand designed approximate accelerators, which might be infeasible due to the large design space even considering a reduced set of approximate arithmetic circuits. For instance, to satisfy accuracy constraints in these accelerators, required to guarantee good enough results despite the on-purpose errors, it is imperative to assess how the errors introduced by approximate circuits propagate through other exact and approximate computations, and finally accumulate at the output. This is, in particular, crucial to enable the high-level synthesis of approximate accelerators. Bridging the gap between many approximate arithmetic circuits and the automated design and implementation of approximate accelerators is crucial to further enable cross-layer approximate computing.

1.1 Literature Survey

Approximate computing enables improvement in speed, reductions in area and power, and savings in energy compared to accurate computing at the expense of an acceptable loss in the accuracy of results [1]. There are many practical applications that are inherently error-resilient, and they have been considered as suitable candidates to evaluate the efficacy and practicality of approximate computing. Examples of such practical applications include multimedia [2], low-power graphics processing [3], memory for multi-core processors [4], the hardware implementation of neural networks for machine learning and artificial intelligence [5], software engineering [6], memory storage [7], big data mining and analytics [8], and neuromorphic computing [9]. Approximate computing broadly covers hardware, software, and memory storage, and approximate hardware includes approximate arithmetic circuits [10] and approximate logic circuits [11]. Within the domain of approximate arithmetic circuits, the design of approximate adders and multipliers has attracted significant attention [12], which is understandable given that addition and multiplication are frequently performed in microprocessors and digital signal processors. For example, in [13], it was found that additions constituted nearly 80% of the operations in an ARM processor's arithmetic and logic unit, and it was noted in [14] that adders and multipliers contributed to about 80% of the total power consumption of a fast Fourier transform processor.
This paper discusses approximate adders and multipliers. Approximate adders are categorized into two types: Static Approximate Adders (SAAs) and Dynamic Approximate Adders (DAAs). SAAs have a fixed approximation, and they could enable significant reductions in delay, area, and power compared to accurate adders for an increase in the approximation. However, prior knowledge of the target application could be useful to determine an optimal approximation for an SAA. On the other hand, DAAs have a flexible approximation and could be configured to produce

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 1, June 2023

an approximate or accurate results on demand, i.e., the accuracy of results could be adjusted as per need and prior knowledge about a target application may not be necessary. However, to achieve this, DAAs incorporate additional error detection and correction logic to facilitate a variable approximation, which forms a design overhead. Furthermore, multiple computational cycles might be required to achieve a result that corresponds to a desired accuracy in a DAA. These two tend to negatively impact the design metrics of DAAs in general. In [15], for a digital video-encoding application, it was observed that the savings in power achieved by an SAA over an accurate adder is comparable with a DAA.

Many multiplier architectures such as Braun array, Booth algorithm, Wallace tree, Baugh Wooley algorithm, and the Dadda tree are available for unsigned and signed multiplication [16]. These accurate multiplier architectures have been modified to obtain approximate multiplier architectures in the literature [17]. With respect to unsigned multiplication, and especially for small multiplications that are typically encountered in digital image processing, the Braun array multiplier (BAM) [18] is preferable because it has a simple and regular structure. Moreover, BAM allows for easy pipelining to increase the throughput as required. Approximate (array) multiplier architectures can be derived by making vertical and/or horizontal cuts in an accurate BAM [19] and assigning different combinations of binary constants to the dangling internal inputs and dangling product bits.

In this article, we describe Approximator, which is a software tool developed to automatically generate Verilog HDL codes of approximate adders and multipliers of any size, corresponding to the following approximate arithmetic circuit architectures proposed by us: approximate adders (HEAA [20], HOERAA [21], HOAANED [22], and M-HERLOA [23]) and approximate (array) multipliers (AAM01 [24,25]). Though we proposed three approximate array multiplier architectures in [24], among them AAM01 [25] (also called PAAM01 [24]) was found to have better optimized error characteristics, and its superior performance was confirmed for a couple of digital image processing applications, namely digital image denoising and digital image blending. Hence, we decided to only incorporate AAM01 into Approximator. The approximate adder and multiplier architectures comprising Approximator correspond to static approximation.

Approximator has been made open for access on GitHub for the benefit of the research community and a beta version of the tool is available for free download [26]. Documentation about the tool is also provided for a user's reference [27]. Approximator has been made available in a convenient graphical user interface (GUI) format for ease of use by an end-user. Approximator asks for input specifications from a user to: (i) generate Verilog HDL codes of approximate adders, (ii) generate Verilog HDL codes of approximate multiplier, (iii) perform error analysis of approximate arithmetic circuits, and (iv) perform accuracy analysis of approximate arithmetic circuits.

The rest of the article is structured as follows. Section 2 and Section 3 describes the approximate adders and the approximate (array) multiplier, respectively, which form a part of Approximator. Next, the development and working principle of the GUI version of Approximator are described in Section 4 through some example screenshots. Finally, Section 5 concludes the article

In inexact computing, approximate adders are the essential building block for the arithmetic circuits. Approximate adders have inaccurate outputs for carry and sum with some combinations of inputs, which have incorrect outputs for sum and carry. So, the hardware requirement of the system gets reduced for inexact computing. As a result, approximate computing yields high speed and low power consumption for the design. However, approximate computing is a suitable choice for DSP applications like video, image, and audio processing, where accurate results are not essential. Adders are implemented by using various digital CMOS technologies (Ashim et al., 2016; Chip-Hong et al., 2005) such as Transmission Gate Adder (TGA), Complementary Pass Transistor Logic (CPL), and (Uming et al., 1995) Double Pass Transistor Logic (DPL) in order to reduce the power consumption of the design. High Performance Error Tolerant Adders and Multiplexer based arithmetic full adders (MBAFA) are proposed (R. Jothin et al., 2018) to reduce the design parameters such as area and power consumption and also to improve the accuracy. Various types of approximate adders are proposed (Fazel Sharifi et al., 2017; Jeevan Jot Singh et al., 2018; Zhixi Yang et al., 2013; Vaibhav Gupta et al., 2013), and their performance is analyzed based on the area, power, speed and accuracy of the results. In Honglan Jiang et al. (2015), various types of approximate adder techniques are analyzed. Error and circuit characteristics are compared. Equal segmentation Adder (ESA) is proposed, which results in better hardware efficiency, but with the lowest accuracy in terms of error. The approximate full adders (AFA) are compared (Sunil Dutt et al.,

IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

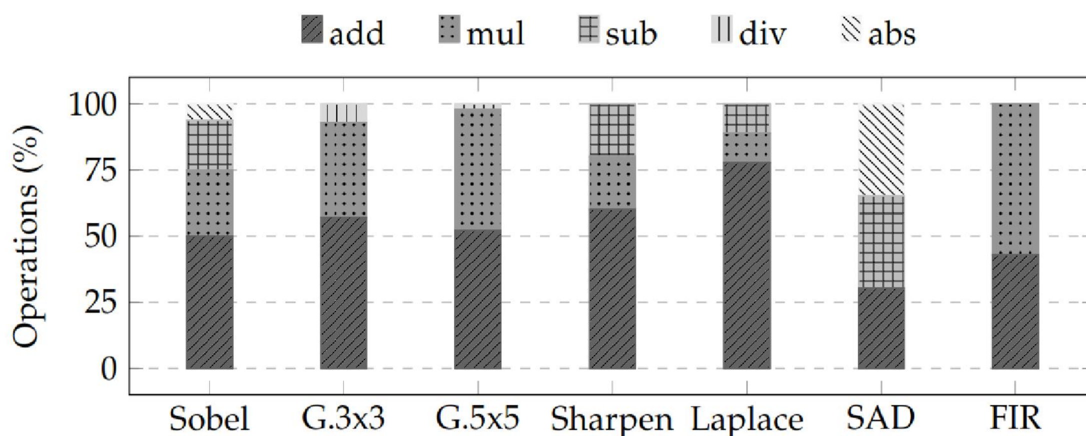
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.301

Volume 3, Issue 1, June 2023

2017) with the existing ripple carry adder (RCA) in terms of area and power. In Tongxin Yang et al. (2018), the approximate adder is based on the carry look ahead adder, and accuracy is realized by masking the carry propagation at run time. It results in reduced power and delay and is used for error tolerant applications. Approximate adders are used in data mining and multimedia signal processing, which can tolerate error, and the exact computing is not necessary. A 4-2 compressor tree was proposed (J.Anjana et al., 2018), and one of the Xor gate is replaced by OR gate to reduce the hardware requirement. Approximate multipliers are also designed (Kalvala et al., 2017; Suganthi et al., 2017) by using approximate adders, which are used in image processing applications. Algorithmic noise Tolerant (ANT) schemes (Rajamohana Hegde et al., 2001) are used to compensate the degradation of the algorithmic performance due to input dependent errors, and this error control scheme is used in soft DSP. Prediction based error control scheme was proposed to improve the performance of the filtering algorithm even when the errors occurred due to the approximate computation. Approximate computing was used in DCT image compression (Haider A.F.Almurib et al., 2018; S Geetha and P Amritvalli. 2017). A set of images are compressed to analyze the different parameters such as delay, energy consumption, and PSNR. Accurate adders (G. Narmadha et al., 2015 and 2016; Manickam Ramasamy et al., 2019) are also designed, implemented, and analyzed for achieving the power efficiency and reducing the hardware requirement. Section 2 depicts the existing approximate adder structures. Section 3 presents the proposed approximate adder structure. Section 4 provides the performance analysis of existing and proposed approximate adders. 1 Approximate Arithmetic Circuits

As discussed in Chapter 1, with the coming forth of Approximate Computing (AxC)as an energy- and accuracy-aware design paradigm, many approximate arithmeticcircuits have been proposed, mainly approximate adders and multipliers. A quick search in the Scopus database1shows more than 1400 publications related to AxC in the last decade, and about 37%of those works correspond to approximate adders or multipliers. The main ideabehind this approximate circuits is to perform the mathematical operations fasteror with less area, power, or energy than the accurate circuits while introducingerrors in the results.In the literature, several methods have been proposed to generate approximatecircuits from accurate descriptions, for instance, by transforming gate-level representationsof the circuit or exploitingdelay in non-critical circuit paths. However, the dominant trend has beento propose approximate designs directly from accurate counterparts.



Other approximate arithmetic circuits have been proposed, for example, approximatedividers. However, this dissertation focuses on approximate addersand m5-tap FIR filter. Figure 1 presents a profile of the requiredmathematical operations for these applications. As it can be noticed, in most ofthese applications, excluding SAD, 75% or more of the operations correspond toadditions and multiplications, and, for instance, none have division as a significantoperation.As many approximate arithmetic circuits have been proposed, and are still beingproposed in the community, this chapter does not cover them extensively. Althoughthe main idea is to reduce the complexity of the arithmetic circuits, some of the keyconcepts behind these approximate units are here mentioned. Extensiv approximate adders. The first considers the substitution of 1-bit fulladder (FA) for simplified versions,aiming to reduce the power consumption of the addition. This type is knownas low-power (LP) approximate adder. For instance, Figure 2 depicts

DOI: 10.48175/IJARSCT-11268

ISSN 2581-9429 IJARSCT

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 1, June 2023

an 8-bitapproximate adder built from a Ripple-Carry Adder (RCA). This adder, namedLower-part-OR adder (LOA), replaces the 1-bit additions of least significant bits.

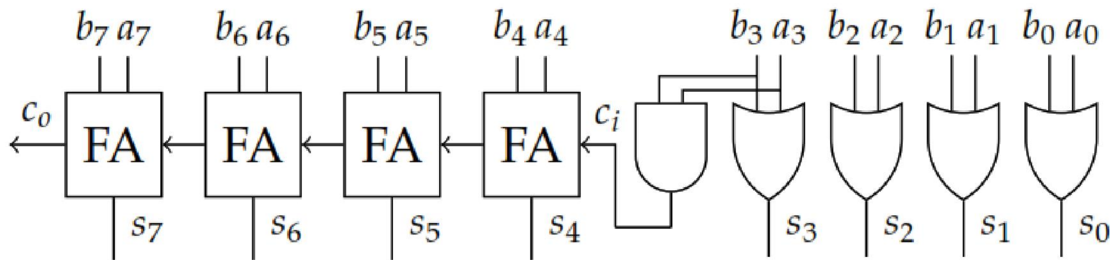


Figure 1

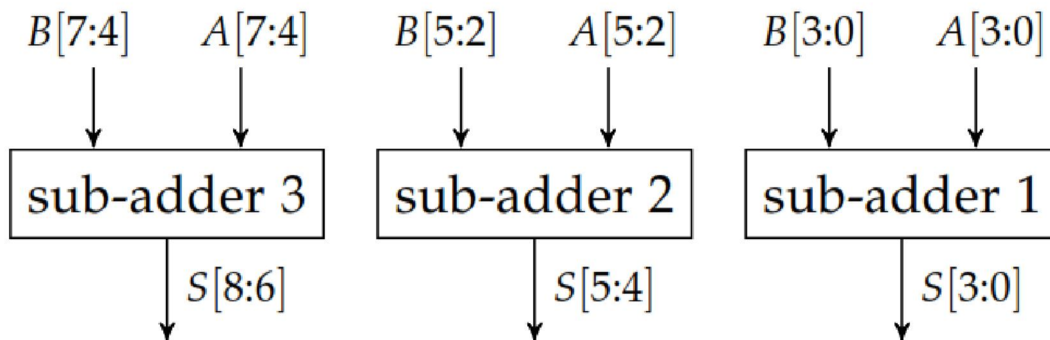


Figure 2

(LSB) performed by a FA with a single OR gate. By doing so, for instance, the required area is reduced as fewer logic gates are needed. Also, the circuit delay is reduced, as the carry propagation chain is cut. For this LOA example, area and power delay are reduced by about 35%, respectively, compared to an 8-bit RCA. The second type of approximate adder is known as high-performance (HP) approximate adder. For this approximate adder, the addition is computed by breaking the carry propagation chain of the exact addition and using multiple (sometimes overlapping) sub-adders to generate the addition result, aiming to reduce the latency of the computation. Figure 3 depicts an example of an HP approximate adder. This adder, called Generic Accuracy Configurable adder, performs an N-bit addition using multiple sub-adders of smaller size. In this example, an 8-bit addition is done by three 4-bit adders. The most significant R-bits of the sub-adders are considered as resultant bits, and they are used in the actual result. The remaining P-bits, known as previous bits, are used to estimate the carry propagation to the upper bits. As shown in Figure 2.3, only the sub-adder 1 contributes with all its partial result to the final result, while sub-adder 2 and 3 provide 2 and 3 bits to the result. This 8-bit example reduces the delay in 40% with respect to an 8-bit RCA.

1.2 Existing Systems

In R. Jothin et al. (2018), MBAFA1 and MBAFA2 have been designed to reduce the number of gates required and delay with minimum errors. MBAFA1 and MBAFA2 are shown in Figures 1 and 2, respectively, and are assumed as Approximate 1 and 2. Adder has three inputs A, B, and C and two outputs sum and carry.

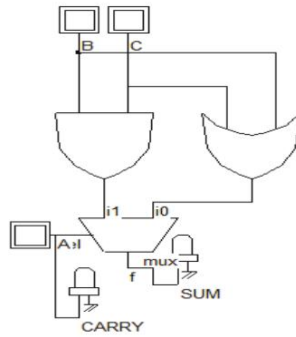


Figure 3

(CNFET) in Fazel Sharifi et al. (2017) to reduce the power consumption, and its design in terms of gates was shown in Figure 3. This design was assumed as Approximate 3.

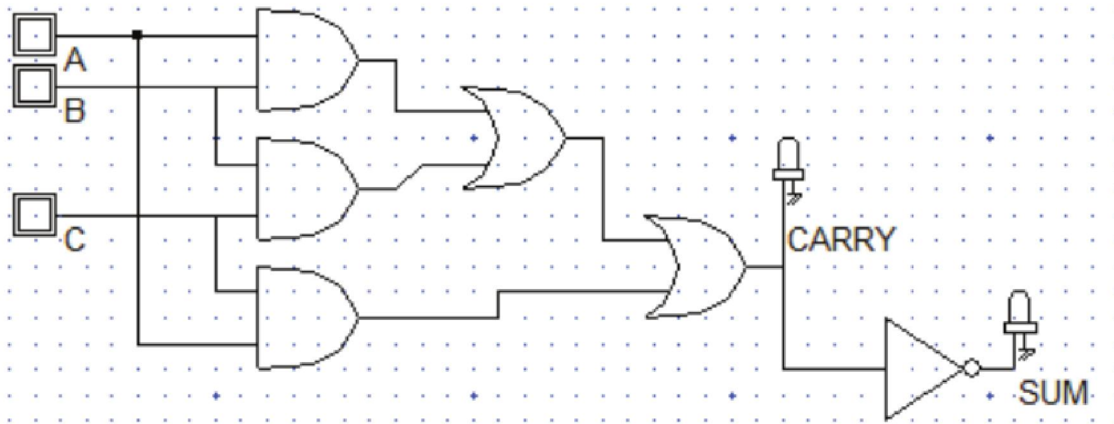


Figure 4

In Jeevan Jot Singh et al. (2018), different types of inexact adders have been designed for image compression techniques. These designs are taken as Approximate 4, Approximate 5, and Approximate 6 and are given in Figure 4, Figure 5, and Figure 6 respectively. Approximate adder 4 incurs a smaller number of gates and reduces the chip size.

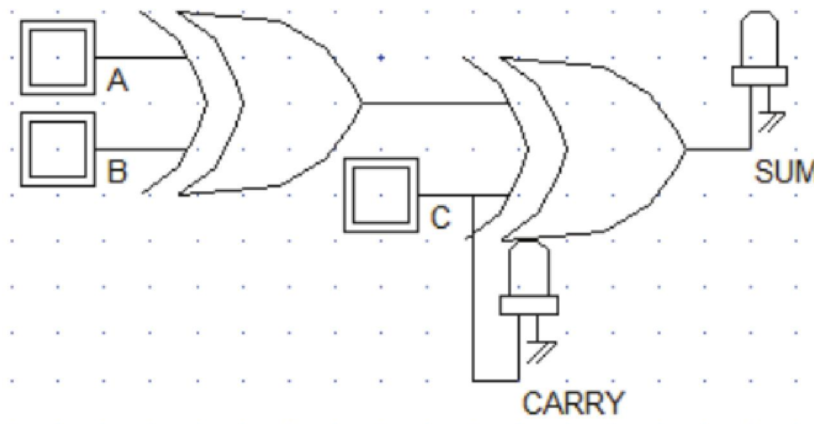


Figure 5

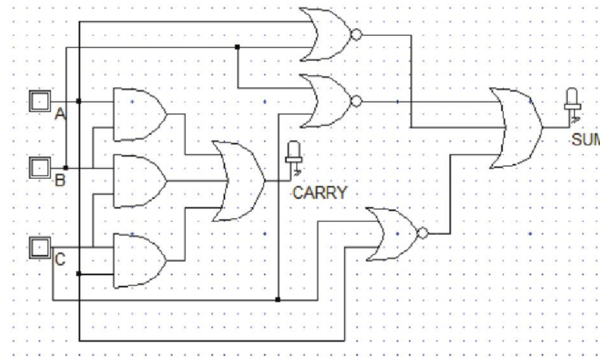


figure 6

XNOR and XOR based adders have been designed in Zhixi Yang et al. (2013) to reduce the count of transistors and power. The three types of adders are assumed here as Approximate.

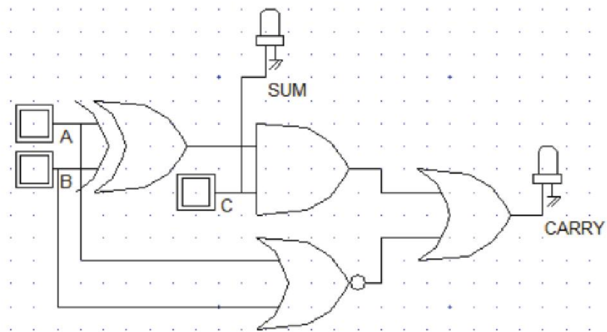


Figure 6

Area efficient and low power approximate multipliers have also been designed in Suganthi et al. (2017) by using approximate adder shown in Figure 10, which is assumed as Approximate 10. In accurate adder, 2 XOR gates are required for sum generation. To reduce the transistor count, one of the XOR gate is replaced by OR gate in sum generation and carry was generated by using only one AND gate. High speed error tolerant adder has also been designed for image processing and multimedia applications (S Geetha et al., 2017) and is shown in Figure 11. This type of adder is assumed as Approximate 11. These types of adders are used in Discrete Cosine Transform (DCT) and Inverse Discrete Cosine Transform (IDCT) for image compression techniques.

INPUTS			OUTPUTS		R. Jothin et al., 2018				Fazel Sharifi et al., 2017		Jeevan Jot Singh et al., 2018					
			Accurate		Approximate 1		Approximate 2		Approximate 3		Approximate 4		Approximate 5		Approximate 6	
A	B	C	Sum	Carry	Sum	Carry	Sum	Carry	Sum	Carry	Sum	Carry	Sum	Carry	Sum	Carry
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0
0	0	1	1	0	1	0	1	0	1	0	1	1	1	0	1	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	1	1	0	1	1	0	1	0	0	1	0	1	1	1	0	1
1	0	0	1	0	0	1	1	0	1	0	1	0	1	0	1	0
1	0	1	0	1	0	1	1	0	0	1	0	1	1	1	0	1
1	1	0	0	1	0	1	0	1	0	1	0	0	0	1	0	1
1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1

Copyright to IJARSCT
www.ijarsct.co.in
DOI: 10.48175/IJARSCT-11268

IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.301

Volume 3, Issue 1, June 2023

II. NEURAL NETWORKS

NNs process information in an entirely different way than a conventional (von Neumann) computer [19]. Weights are adjusted in the neurons of a NN to allow the NN to perform certain computations (e.g., pattern recognition and classification on vectors or arrays of input values) [20]. Note that the neurons in a NN are arranged in several layers including an input layer, a variable number of hidden layer(s) (of the same or different types) followed by an output layer. The neurons within the same layer process inputs from the earlier layer in parallel. The outputs from the output layer are often used to signal the likelihood of membership in two or more disjoint classes. As experience is being gained in machine learning tasks, diverse types of hidden NN layers have been proposed. The authors in [21] employed convolutional layers that function as local filters to data from the previous layers. Other common types of hidden layers are the average and max pooling layer that are used for weighted sub-sampling [22]. More recently, several application-specific layers have been proposed for image classification [23], seg mentation [24] and speech processing [25].

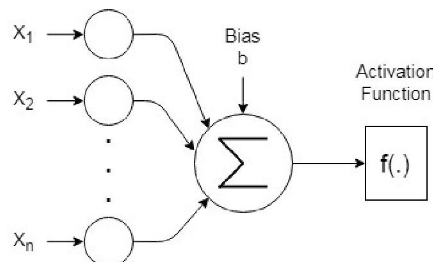


Figure 2.2: Model of an artificial neuron

Artificial neuron

Neurons are the main processing units of NNs that compute a weighted sum of their inputs and then send the result through an activation function (AF). The AF introduces non-linearity into a NN's behaviour and maps the resulting output values either into either the interval (-1, 1) or (0, 1) [8]. The AF can be either a hard-limiting function (e.g., a step function) or a soft-limiting function (e.g., a sigmoid function) [26].

Fig. 2.2 shows the structure of an artificial neuron. A neuron has $n \geqslant 2$ inputs (depending on the network structure) and one output. Each input xi is multiplied by its corresponding synaptic weight wi, i = 0, 1, ..., n. An adder tree is then used to sum up the products. The resulting sum is then input to the AF. An external bias b is often included to increase or lower the sum that is the input to the AF [20].

Feed-forward neural networks

The two major operating modes for NNs are training and inference. The training process is usually performed infrequently and off-line and, therefore, its energy consumption is less of a concern [26]. The inference process, on the other hand, is done frequently. Although it is less computation-intensive than the training process, inference still requires significant computation for large networks. Note that a trained network can be retrained and used to perform a different tack on a different dataset. Usually only a few steps of retraining are required to fine-tune the pre-trained network for another problem. Fig. 2.3 shows a feed-forward NN with n, k, and m neurons in the input, hidden, and output layers, respectively.

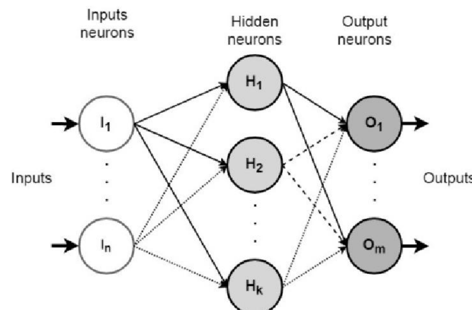


Figure 2.3: Structure of a feed-forward NN.

Convolutional neural networks

CNNs are a class of deep neural networks, which are mainly used to analyze visual imagery [6], [27], [28]. CNNs are feed-forward neural networks consisting of a pipeline of layers. Each layer inputs a set of data, known as a feature map (FM), and produces a new set of FMs with higher-level semantics [6]. The four main computations involved in the major types of a typical CNN layers are:

1. Convolutional layer: A convolutional layer applies a set of trained con- volution filters Θ to a set of input volumes X^{conv} (i.e., a color image in the case of the first convolutional layer or an output generated by previous layers in the network) and outputs a set of FMs, Y^{conv}. The computations involved in a convolutional layer are thus:

$$Y^{conv} = conv(X^{conv}, \Theta) + \beta[n] \quad (2.7)$$

where β denotes the trained bias term. Note that during convolution, the kernel Θ slides across the whole range of X^{conv}.

2. Activation layer: A convolutional layer is usually followed by an activa- tion layer that applies a non-linear function to all of the FM's values. The most common activation function, which is also used in this work, is the rectified linear unit (ReLU) that implements $Y^{act} = \max(0, X^{act})$ [23], where X^{act} denotes the input to this layer.

3. Pooling layer: A pooling layer sub-samples the output of the convolution layer and reduces the spatial dimension by discarding irrelevant detail [29]. The intuitive reasoning behind this layer is that the exact location of a specific feature (which is extracted in the convolution layer) is not as important as its location relative to the other features [30]. The typical pooling layers are the maximum and average pooling layers, which produce almost identical results [29]. The average pooling layer, which slides over the input to this layer and outputs the average of every sub- region that the filter convolves around, is used in this research study.

4. Fully-connected layer: The fully-connected layers are usually form the last few layers of a CNN. A fully-connected layer takes the output of the previous layer (i.e., the activation maps of high-level features) and determines which features most strongly correlate to a particular class

Fig. 2.4 shows an illustrative example of the feed-forward propagation in the convolution and activation layers. The bias is omitted in Fig. 2.4 for sim- plicity. Parameter N in this figure indicates the number of filters.

The computational workload of a CNN inference is the result of an intensive use of the multiply-accumulate (MAC) operations. Most of these MACs occur in the convolutional layers and, therefore, convolutional layers are responsible more than 90% of execution time during the inference [31]

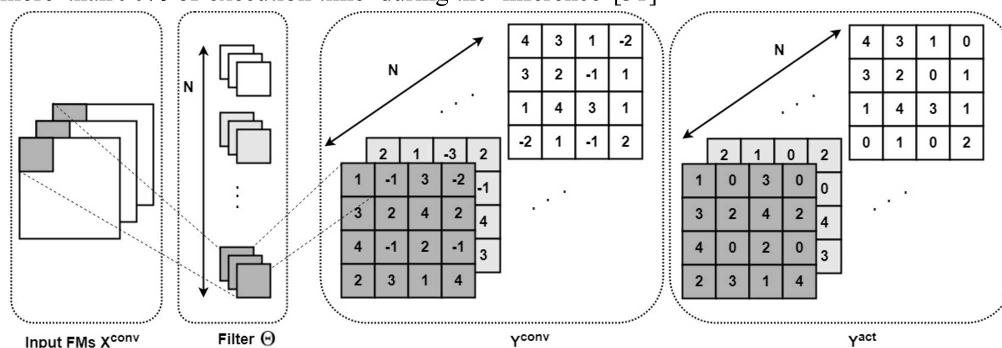


Figure 2.4: Feed-forward propagation in convolutional and activation layers.

2.2. Image Processing Application

The performance of approximate adders was evaluated based on a digital image processing (DIP) application. Digital images of 512 × 512 pixels, with a grayscale resolution of 8 bits were used for experimentation. Fast Fourier transform (FFT) was performed on the images and then inverse FFT (IFFT) was performed to reconstruct the images following the procedure in [28]. Integer FFT and IFFT operations were performed. In the FFT and IFFT computations,

IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.301

Volume 3, Issue 1, June 2023

multiplications were accurately performed, while additions were accurately and approximately performed, separately, to compare the performance of accurate and approximate adders.

In general, the savings in design metrics achieved by an SAA compared to the accurate adder are proportionate to the degree of incorporated approximation [21,22]. Therefore, an optimum approximation has to be determined to strike an acceptable compromise between maximizing the savings in design metrics and ensuring the good quality of results (here, the quality of DIP results). Based on an extensive trial-and-error, a 32-bit approximate adder comprising a 10-bit inexact part was found to be acceptable for the DIP application [22], and this was adopted for this work. The images reconstructed using different approximate adders such as LOA [19], LOAWA [29], APPROX5 [30], HEAA, OLOCA [31], HOERAA, HOAANED, HERLOA [32], and M-HERLOA were compared with the original image on the basis of two well-known figures of merit, namely the peak signal to noise ratio (PSNR) [33] and the structural similarity index metric (SSIM) [34]. PSNR (in dB) varies from zero to infinity, and SSIM varies from 0 to 1 decimal. A higher PSNR indicates low noise or distortion, and a higher SSIM indicates greater structural similarity between the reference (original) image and the target image. The image reconstructed using the accurate adder had a PSNR of infinity and SSIM of 1 since no noise was introduced in the accurate computation of FFT and IFFT, and the original image was faithfully reconstructed. On the contrary, the images reconstructed using approximate adders did not have ideal PSNR and SSIM values since noise was introduced during the approximate computation of FFT and IFFT. Figure 2, Figure 3 and Figure 4 show example DIP results obtained for 'lena', 'cameraman', and 'woman with dark hair' images.

Figure 2. Image processing result for 'lena' image obtained using accurate and approximate adders:

2.3 Proposed Multiplier

Several 4:2 compressors are required to implement one 4× 4 multiplier. How- ever, the function of an exact 4:2 compressor can be approximated to reduce the hardware cost. Ignoring C_{out} (due to its small impact on the compressor's accuracy [54]) as well as our goal to use as few gates as possible led to the approximate compressor truth table given in Table 3.2.

As shown in Table 3.2, there are five/seven incorrect values for the approximate Carry/Sum outputs which correspond to an output error. To reduce this source of inaccuracy, we encode the inputs to the compressor using conventional propagate and generate signals given by:

$$P(i,j) = pp(i,j) + pp(j,i), G(i,j) = pp(i,j) \cdot pp(j,i$$

X1	X2	X3	X4	Carry	Sum
x1	x2	x3	x4	Exact / Approximate	Exact / Approximate

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 1, June 2023

0	0	0	0	0/0 ✓	0/0 ✓
0	0	0	1	0/0 ✓	1/1 ✓
0	0	1	0	0/0 ✓	1/1 ✓
0	0	1	1	1/1 ✓	0/1 ✗
0	1	0	0	0/0 ✓	1/1 ✓
0	1	0	1	1/0 ✗	0/1 ✗
0	1	1	0	1/0 ✗	0/1 ✗
0	1	1	1	1/1 ✓	1/1 ✓
1	0	0	0	0/0 ✓	1/1 ✓
1	0	0	1	1/0 ✗	0/1 ✗
1	0	1	0	1/0 ✗	0/1 ✗
1	0	1	1	1/1 ✓	1/1 ✓
1	1	0	0	1/1 ✓	0/1 ✗
1	1	0	1	1/1 ✓	1/1 ✓
1	1	1	0	1/1 ✓	1/1 ✓
1	1	1	1	0/1 ✗	0/1 ✗

encoding ensures that, although the approximate circuit may have a fairly large number of faulty output entries in the truth table, it in fact rarely produces those outputs. To see how this approach affects the compressor's accuracy, consider Stage 2 in which the following terms are added: $pp_{2,0}$, $pp_{1,1}$Table 3.2: Truth table of the proposed approximate compressor.

$$\text{Approximate Sum} = (x_1 + x_2) + (x_3 + x_4)$$

$$\text{Approximate Carry} = (x_1.x_2) + (x_3.x_4)$$

$pp_{0,2}$, and c_1. Table 3.3, where NA stands for Not Applicable, shows how encoding the PPs using (3.2) helps to improve the design accuracy compared to the situation in Table 3.2. Note that all possible input combinations for the 4×4 multiplier were considered ($2^4 \times 2^4 = 256$) to obtain the probability of each input combination shown in Table 3.3.

Using the proposed technique, the number of faulty Carry/Sum values is reduced from 5/7 to 2/4. Note that the two approximated cases for the Carry signal occur only with a small probability of 0.078 (0.0624+0.0156), see Table 3.3. It is also worth mentioning that the following combinations in Table 3.3 cannot occur, so they do not contribute to the output errors for the approximate compressor:

• (0,1) for ($pp_{(1,1)}$, c_1): since $c_1 = pp_{(0,1)}.pp_{(1,0)} = (\alpha_0.\beta_1).(\alpha_1.\beta_0)$, c_1 = '1' means that α_0, β_1, α_1, and β_0 are '1'. Consequently, $pp_{(1,1)} = \alpha_1.\beta_1 = 1$. Hence, it is impossible to have the (0,1) combination for ($pp_{(1,1)}$, c_1).

III. HARDWARE IMPLEMENTATION TECHNOLOGIES

The construction of a typical real-time imaging or video embedded system is usually an integration of a range of electronic devices, e.g. image acquisi- tion device, signal processing units, memories, and a display. Driven by the market demand to have faster, smarter, smaller and more interconnected prod- ucts, designers are under greater pressure to make decisions on selecting the appropriate technologies in each one of the devices among many of the alter- natives. Trade-offs are constantly made concerning e.g. cost, speed, power, and configurability. In this chapter, a brief overview of the varied alternative technologies is given along with elaborations on the plus and minus sides of each of the technologies, which motivates the decisions made on the selection of the right architecture for each of the devices used in the projects.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 1, June 2023

3.1 ASIC vs. FPGA

The core devices of an real-time embedded system are composed of one or several signal processing units implemented with different technologies such as Micro-controller units (MCUs), Application Specific Signal Processors(ASSPs), General Purpose Processors (GPPs/RISCs), Field Programmable Gate Arrays (FPGAs) and Application Specific Integrated Circuits (ASICs). A comparison is made for the areas where each of these technologies prevails [2], which is a bit biased to DSPs. This is shown in Table 2.1. No perfect technology exists that is competent in all areas. For a balanced embedded system design, a combination of some of the alternative technologies is a necessity. In general, an embedded system design is initiated with Hardware/Software partitioning, once the original specifications are settled under various system requirements.

Speed In terms of maximum achievable clock frequency, ASICs are typically much faster than an FPGA given the same manufacture process technol- ogy. This is mainly due to the interconnect architecture within FPGAs.

Speed In terms of maximum achievable clock frequency, ASICs are typically much faster than an FPGA given the same manufacture process technol- ogy.

Table 2.1: Comparisons of different types of signal processing units. Sources are from [2].

	Performance	Price	Power	Flexibility	Time to market
ASIC	Excellent	Excellent [1]	Good	Poor	Fair
FPGA	Excellent	Poor	Fair	Excellent	Good
DSP	Excellent	Excellent	Excellent	Excellent	Good
RISC	Good	Fair	Fair	Excellent	Excellent
MCU	Fair	Excellent	Fair	Excellent	Excellent

The partitioning is carried out by either a heuristic approach or by a certain kind of optimization algorithm, e.g. simulated annealing [3] or tabu search [4]. Software is executed in processors (DSPs, MCUs, ASSPs, GPPs/RISCs) for features and flexibility, while dedicated hardware are used for parts of the algorithm which are critical regarding timing constraints. With the main focus of the thesis being on the blocks that need to be accelerated and optimized by custom hardware for better performance and power, only ASIC and FPGA implementation technologies are discussed in the following sections.

With the full freedom to customize the hardware to the very last single bit of logic, both ASICs and FPGAs can achieve much better system performance compared to other technologies. However, as they differ in the inner structure of logic blocks building, they posses quite different metrics in areas such as speed, power, unit cost, logic integration, etc. In general, designs implemented with ASIC technology is optimized by utilizing a rich spectrum of logic cells with varied sizes and strengths, along with dedicated interconnection. In contrast, FPGAs with the aim of full flexibility are composed of programmable logic components and programmable interconnects. A typical structure of an FPGA is illustrated in figure 2.1. Figure 2.2 and 2.3 show the details of programable logic components and interconnects. Logic blocks can be formed on site through programming look up tables and the configuration SRAMs which control the routing resources. The programmability of FPGAs comes at the cost of speed, power, size, and cost, which is discussed in details in the following.

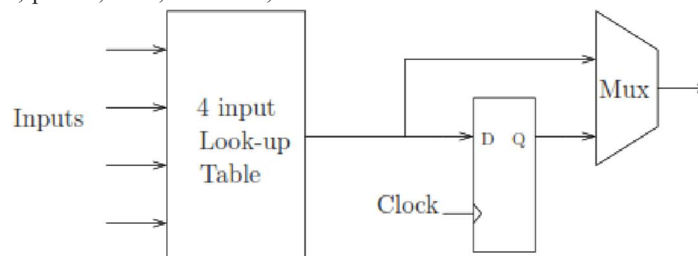


Figure 7 Simplified programmable logic elements in an typical FPGA architecture.

IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.301

Volume 3, Issue 1, June 2023

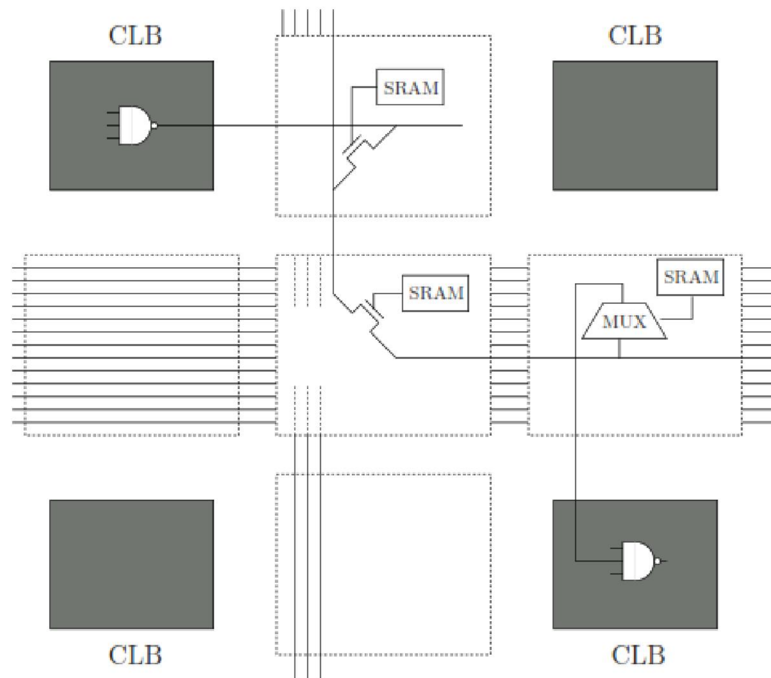


Figure 8 Configurable routing resources controlled by SRAMs.

Clock speed	ASICs	FPGAs
Power	Low	High
Unit cost with volume production	Low	High
Logic Integration	High	Low
Flexibility	Low	High
Back-end Design Effort	High	Low
Integrated Features	Low	High

Table 2: Comparisons between ASICs and FPGAs.

To ensure programmability, many FPGA devices utilize pass transistors to connect different logic cells dynamically, see figure 2.3. These active routing resources add significant delays to signal paths. Furthermore, the length of each wire is fixed to either short, medium, and long types. No further optimization can be exploited on the wire length even when two logic elements are very close to each other. The situation could get even worse if high logic utilization is encountered, in which case it is difficult to find a appropriate route within certain regions. As a result, physically adjacent logic elements do not necessarily get a short signal path. In contrast, ASICs has the facility to utilize optimally buffered wires imple- mented with metal in many layers, which can even route over logic cells. Another contributor to FPGAs speed degradation lies in its logic granu- larity. In order to achieve programmability, look-up tables are used which usually have a fixed number of inputs. Any logic function with slightly more input variables will take up additional look-up tables, which will again introduce additional routing and delay. On the contrary, ASICs, usually with a rich spectrum types of logic gates of varying functionality and drive strength (e.g. over 500 types for UMC 0.13 µm technology used at the department), logic functions can be very fine tuned during synthesis process to meet a better timing constraint.

Power The active routing in FPGA devices does not only increase signal path delays, it also introduce extra capacitance. Combined with large capaci- tances caused by the fixed interconnection wire length, the

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 1, June 2023

capacitance in FPGA signal path is in general several times larger than that of an ASIC. Substantial power consumption is dissipated during signal switching that drives such signal paths. In addition, FPGAs have pre-made dedicated clock routing resources, which are connected to all the flip flops on an FPGA in the same clock domain. The capacitance of the flip flop will contribute to the total switching power even when it is not used. Fur- thermore, the extra SRAMs used to program look-up tables and wires also consume static power.

Logic density The logic density on an FPGA is usually much lower compared to ASICs. Active routing device takes up substantial chip area. Look-up tables waste logic resource when they are not fully used, which is also true for flip-flops following each look-up table. Due to relatively low logic density, around 1/3 of large ASIC designs in the market usually could not fit into one single FPGA [5]. Low logic density increase the cost per unit chip area, which makes ASIC design more preferable for industry designs in mass production.

Despite of all the above drawbacks, FPGA implementation also comes with quite a few advantages, which is served as the motivation in the thesis work.

Verification Ease Due to its flexibility, an FPGA can be re-programmed as requested when a design flaw is spotted. This is extremely useful for video projects, since algorithms for video applications usually need to be verified over a long time period to observe long term effects. Computer simulations are inherently slow. It could take a computer weeks of time to simulate a video sequences lasting for only several minutes. Besides, an FPGA platform is also highly portable compared to a computer, which makes it more feasible to use in heterogeneous environments for system robustness verification.

Design Facility Modern FPGAs comes with integrated IP blocks for design ease. Most importantly, microprocessors are shipped with certain FP- GAs, e.g. (hard Power PC and soft Microblaze processor cores on Virtex II pro and later version of Xilinx FPGAs). This gives great benefit to hardware/software co-design, which is essential in the presented video surveillance project. Algorithm such as feature extraction and tracking is more suitable for software implementation. With the facilitation of various FPGA tools, interaction between software and hardware can be verified easily in an FPGA platform. Minor changes in hardware/software partitioning are easier and more viable compared to ASICs.

Minimum Effort Back-end Design The FPGA design flow eliminates the complex and time-consuming floor planning, place and route, timing anal- ysis, and mask/re-spin stages of the project, since the design logic is al- ready synthesized to be placed onto an already verified, characterized FPGA device. This will facilitate hardware designers more time to con- centrate mainly on architecture and logic design task.

From the discussions above, FPGAs are selected as our implementation technology due to its fair performance and all the flexibilities and facilities.

Dynamic Range	CCD High	CMOS Moderate
Speed	Moderate	High
Windowing	Limited	Extensive
Cost	High	Low
Uniformity	High	Low to moderate
System Noise	Low	High

Table 2.3: Image sensor technology comparisons: CCD vs. CMOS.

An image sensor is a device that converts light intensity to an electronic signal. They are widely used among digital cameras and other imaging devices. The two most commonly used sensor technologies are based on Charge Coupled De- vices (CCD) or Complementary Metal Oxide Semiconductor(CMOS) sensors. Descriptions and comparisons of the two technologies are briefly discussed in the following which are based on [6–8]. A summary of the two sensor types is given in Table 2.3. Both devices are composed of a array of fundamental light sensitive elements called photodiodes, which excite electrons (charges)

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 1, June 2023

When there is light with enough photons striking on it. In theory, the trans- formation from photon to electron is linear so that one photon would release one electron. In general, this is not the case in the real world. Typical image sensors intended for digital cameras will release less than one electron. The photodiode measures the light intensity by accumulating light incident for a short period of time (integration time), until enough charges are gathered and ready to be read out. While CCD and CMOS sensors are quite similar in these basic photodiode structure, they mainly differs in the way how these charges are processed, e.g. readout procedure, signal amplification, and AD conver- sion. The inner structures of the two devices are illustrated in figure 2.4 and CCD sensors read out charges in a row-wise manner: The charges on each
row are coupled to the row above, so when the charges are moved down to the row below, new charges from the row above will fill the current position, thus the name Coupled Charged Device. The CCD shifts one row at a time to the readout registers, where the charges are shifted out serially through a charge- to-voltage converter. The signal coming out of the chip is a weak analog signal, therefore an extra off-chip amplifier

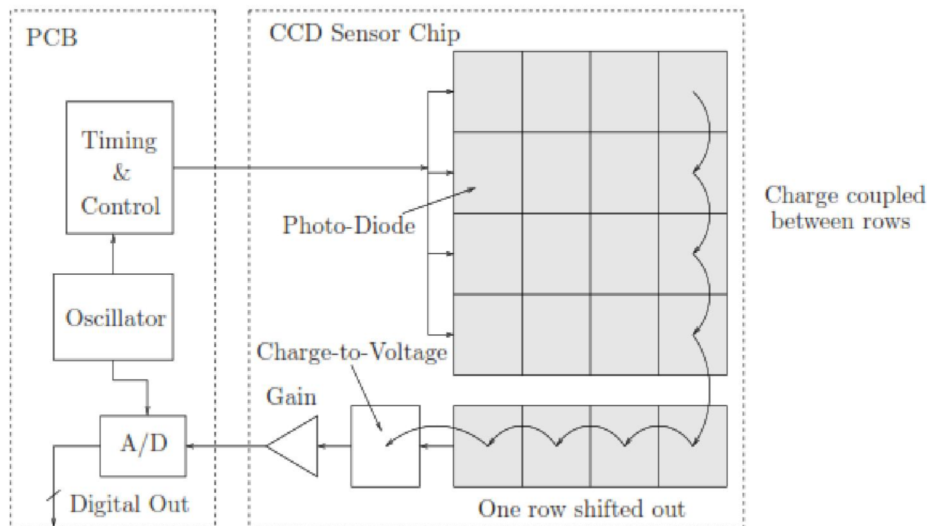


Figure 9 A typical CCD image sensor architecture.

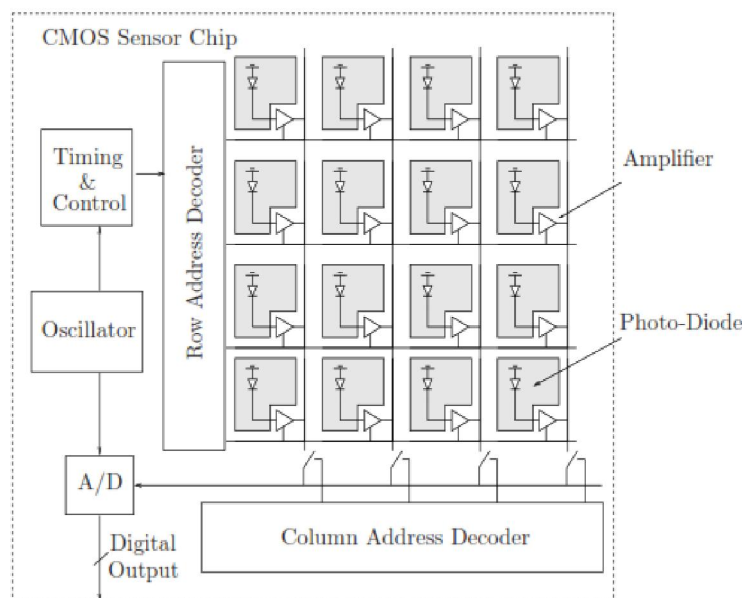


Figure 10 A typical CMOS image sensor architecture

DOI: 10.48175/IJARSCT-11268

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 1, June 2023

Board (PCB) which results in a higher cost. On the chip level, although CMOS sensor can be manufactured using a foundry process technology that is also capable of producing other circuits in volume, the cost of the chip is not considerable lower than a CCD. This is due to the fact that special, lower volume, optically adapted mixed-signal process has to be used by the requirement of good electro-optical performance [6].

Image Quality The image quality can be measured in many ways:

Noise level CMOS sensors in general have a higher level of noises due to the extra circuits introduced. This can be compensated to some extent by extra noise correction circuits. However this could also increase the processing time between frames.

Uniformity CMOS sensors use separate amplifier for each pixel, the offset and gain of which can vary due to wafer process variations. As a result, the same light intensity will be interpreted as different value. CCD sensor with an off-chip amplifier for every pixel, excel in uniformity.

Light Sensitivity CMOS sensors are less sensitive to light due to the fact that part of each pixel site are not used for sensing light but for processing. The percentage of a pixel used for light sensing is called fill factor, which is shown in figure 2.2. In general, CCD sensors have a fill factor of 100% while CMOS sensor has much less, e.g. 30%－60% [9]. Possibly, such a drawback can be partially solved by adjusting integration time of each pixel. Speed and Power In general, a CMOS sensor is faster and consumes lower power compared to a CCD. Moving auxiliary circuits on chip, parasitic capacitance is reduced, which increase the speed at the same time con- sumes less power.

Windowing The extra row and column decoders in CMOS sensors enable data reading out from arbitrary positions. This could be useful if only portion of the pixel array is of interest. Reading out data with using different resolution is made easy on CMOS sensor without having to discard pixels outside the active window as compared to a CCD sensor.

IV. HARDWARES AND SOFTWARE

4.1 Introduction

The electronics industry has achieved a phenomenal growth over the last two decades mainly due to the rapid advances in integration technologies, large-scale systems design – in short, due to the advent of VLSI. Typically,the required computational power of these applications is the driving force for the fast development for this field. One of the most important characteristics of information service is their increasing need for very high processing power and bandwidth. The other important characteristics is that the information services tend to become more and more personalized,which means that the devices must be more intelligent to answer individual demands,and at the same time they must be portable to allow more flexibility and mobility. More complex function are required in various data processing and telecommunications devices,the need to integrate these functions in a small system,packages is also increasing.

The level of integration as measured by the number of logic gates in a monolithic chip has been steadily rising for almost three decades,mainlydue to the rapid progress in processing technology and interconnect technology.

The monolithic integration of large number of functions on a single chip usually provides

1.Less area/volume and therefore compactness.
2. Less power consumption.
3. Less testing requirements at system level.
4. High reliability,mainly due to improve on-chip interconnects.
5. High speed,due to significantly reduced interconnection length.
6. Significant cost saving.

Therefore,the current trend of integration will also continue in the future. Advances in devices manufacturing technology and especially the steady reduction of minimum feature size support the trend. A minimum size of 0.25 microns was readily achieved. Logic chip such as microprocessor chips and digital signal processing chips contain not only large array of memory(SRAM) Cells, but also many different functional units. As a result, their design complexity is considered much higher than that of memory chips. Sophisticated computer-aided design tools and methodologies are developed and applied in order to manage the rapidly increasing design complexity.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 1, June 2023

4.2 VLSI DESIGN FLOW

Fig 2.1 provides the most simplified view of the VLSI design flow,taking into account the various representation or abstraction of design-behavioral,logic circuit and mask layout. Note that the verification of design plays a very important role in every step during this process.

Although top-down design flow provides an excellent process control,in reality,there is no truly unidirectional top-down design. Both top-down and bottom-up approaches have to be combined. For instance, is a chip designer defined architecture without close estimation of the corresponding chip area then it is very likely that the resulting chip layout exceeds the area limit of the available technology. In such a case,in order to fit the architecture into the allowable chip area,some functions have to be removed and the design process must be repeated.

Such changes may require significant modification of the original requirement. Thus it is very important to feed forward low-level information to higher level as early as possible.

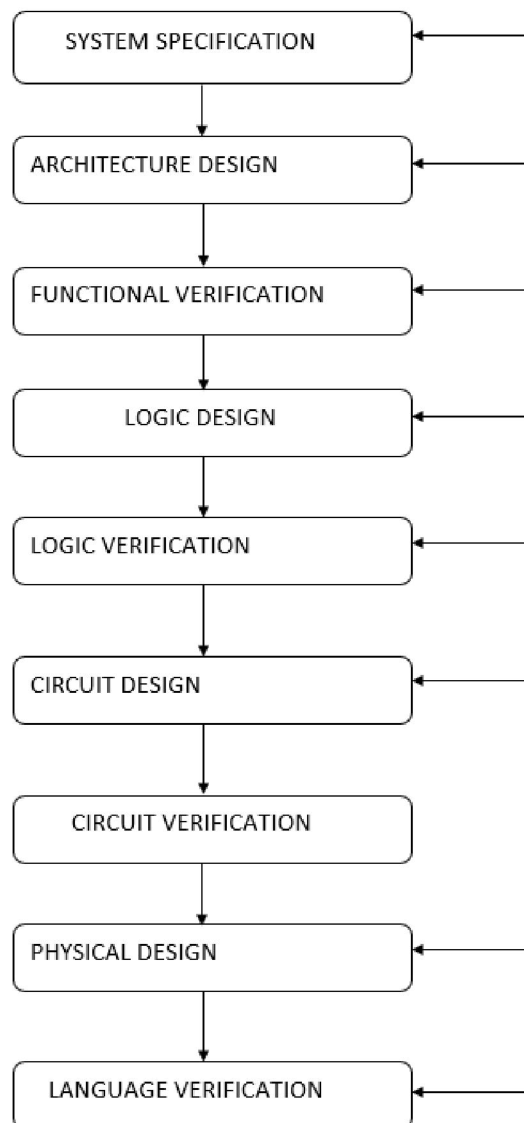


Figure 11 VLSI Design Flow

Copyright to IJARSCT
www.ijarsct.co.in
DOI: 10.48175/IJARSCT-11268

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 1, June 2023

4.3 FPGA INTRODUCTION

A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by the designer after manufacturing-hence "field-programmable".The FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC). FPGAs can be used to implement any logical function that an ASIC could perform. The ability to update the functionality after shipping partial re-configuration of the portion of the design and the low non-recurring, engineering costs relative to an ASIC design(notwithstanding the generally higher unit cost), offer advantages for many applications.The gate array where the logic network can be programmed into the device after its manufacture. An FPGA consists of an array of logic elements, either gates or lookup table RAMs, flip-flops and programmableinterconnectwiring.

Most FPGAs are reprogrammable, since their logic functions and interconnect are defined by RAM cells. The Xilinx LCA, Altera FLEX and AT&TORCA devices are examples. The Actel FPGAs are the leading example of such devices. Atmel FPGAs are currently (July 1997) the only ones in which part of the array can be reprogrammed while other parts are active. As of 1994, FPGAs have logic capacity up to 10K to 20K 2-input-NAND-equivalent gates, up to about 200 I/O pins and can run at clock rates of 50 MHz or more. FPGA designs must be prepared using CAD software tools, usually provided by the chip vendor, to do technology mapping, partitioning and placement, routing, and binary output.

The resulting binary can be programmed into a ROM connected to the FPGA or downloaded to the FPGA from a connected computer. In addition to ordinary logic applications, FPGAs have enabled the development of logic emulators There is also research on using FPGAs as computing devices, taking direct advantage of their re-configurability into problem-specific hardware processors.

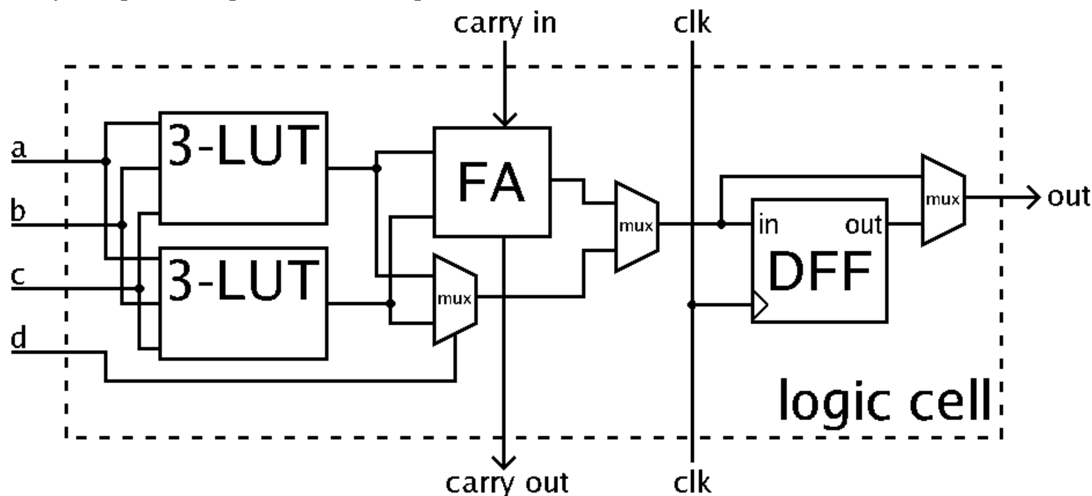


Figure 12 FPGA Logic block

FPGAs contain programmable logic components called "logic blocks" and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together" (changeable) logic gates that can be inter-wired in different configurations. Logic blocks can be configured to perform complex combinational functions or merely simple logic gates like AND and XOR. In most FPGAs logic blocks also include memory elements which may be simple flip-flops (FF) or more complex blocks of memory.In addition to digital functions, some FPGAs have analog features.

The most common analog feature is programmable slew rate and drive strength on each output pin, allowing the engineer to set slow rates on lightly loaded pins that would otherwise ring unexpectedly, and to set stronger, faster rates on heavily loaded pins on high speed channels that would otherwise run too slow. Another relatively common analog feature is differential comparators on input pins designed to be connected to differential signaling channels.

A few "mixed signal FPGAs" have integrated peripheral Analog-Digital converters (ADCs) and Digital-to-Analog Converters (DACs) with analog signal conditioning blocks allowing them to operate as a system-on-a-chip. Such devices blur the line between an FPGA which carries digital ones and zeros on its internal programmable interconnect fabric, and field-programmable analog array (FPAA), which carries analog values on its internal programmable interconnect fabric.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 1, June 2023

4.3.1 APPLICATIONS

Applications of FPGAs include digital signal processing, software-defined radio, aerospace and defense systems, ASIC prototyping, medical imaging, computer vision, speech recognition, cryptography, bioinformatics, computer hardware emulation, radio astronomy, metal detection and a growing range of other areas. With the introduction of dedicated multipliers into FPGA architectures in late 1990s, applications which had traditionally been the sole reserve of DSPs began to incorporate FPGAs instead.

Traditionally, FPGAs have been reserved for specific vertical applications where the volume of production is small. For these low volume applications, the premium that company pay in hardware costs per unit for a programmable chip is more affordable than the development resources spent on creating an ASIC for a low-volume application. Today new cost and performance dynamics have broadened the range of viable applications.

4..3.2 ARCHITECTURE

The most common FPGA architecture consists of an array of logic blocks (called Configurable Logic Block, CLB, or Logic Array Block, LAB, depending on vendor), I/O pads, and routing channels. Generally, all the routing channels have the same width (number of wires). Multiple I/O pads may fit into the height of one row or the width of one column in the array.An application circuit must be mapped into an FPGA with adequate resources. While the number of CLBs/LABs and I/Os required is easily determined from the design, the number of routing tracks needed may vary considerably even among designs with the same amount of logic. For example, a crossbar switch requires much more routing than asystolicarray with the same gate count. Since unused routing tracks increase the cost (and decrease the performance) of the part without providing any benefit, FPGA manufacturers try to provide just enough tracks so that most designs that will fit in terms of Lookup tables (LUTs) and I/Os can be routed. Together, they control over 80 percent of the market.

Other competitors include Lattice Semiconductor (SRAM based with integrated configuration flash, instant-on, low power, live reconfiguration), Actel (now Microsemi, antifuse, flash-based, mixed-signal), Silicon Blue Technologies (extremely low power SRAM-based FPGAs with optional integrated nonvolatile configuration memory; acquired by Lattice in 2011), Achronix (SRAM based, 1.5 GHz fabric speed), and QuickLogic (handheld focused CSSP, no general purpose FPGAs).In general, a logic block (CLB or LAB) consists of a few logical cells (called ALM, LE, Slice etc.). A typical cell consists of a 4-input LUT, a Full adder (FA) and a D-type flip-flop, as shown below. The LUTs are in this figure split into two 3-input LUTs. In normal mode those are combined into a 4-input LUT through the left MUX. In arithmetic mode, their outputs are fed to the FA. The selection of mode is programmed into the middle multiplexer.

The output can be either synchronous or asynchronous, depending on the programming of the MUX to the right, in the figure example. In practice, entire or parts of the FA are put as functions into the LUTs in order to save space.

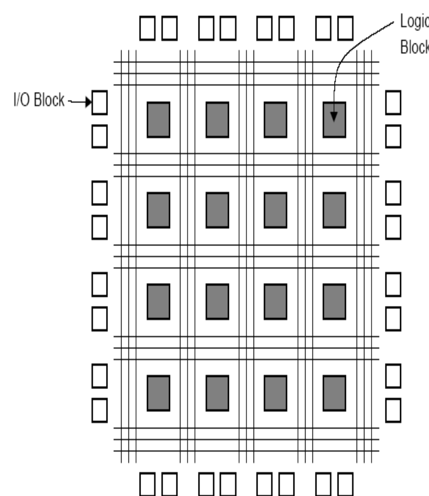


Figure 13: FPGA Architecture

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 1, June 2023

4.4 XILINX

Xilinx designs, develops and markets programmable logic products including integrated circuits (ICs), software design tools, predefined system functions delivered as intellectual property (IP) cores, design services, customer training field engineering and technical support. Xilinx sells both FPGAs and CPLDs programmable logic devices for electronic equipment manufacturers in end markets such as communications, industrial, consumer, automotive, and data processing.

Xilinx's FPGAs have been used for the ALICE (A Large Ion Collider Experiment) at the CERN European laboratory on the French-Swiss border to map and disentangle the trajectories of thousands of subatomic particles. Xilinx also engaged in a partnership with the United States Air Force Research Laboratory's space vehicles Directorate to develop FPGAs to withstand the damaging effects of radiation in space for deployment in new satellites, which are 1,000 times less sensitive to space radiation than the commercial equivalent.

Xilinx FPGAs can run a regular embedded OS (such as Linux or works) and can implement processor peripherals in programmable logic. Xilinx's IP cores include IP for simple functions (BCD encoders, counters, etc.,) for domain specific cores (digital signal processing, FFT and FIR cores) to complex systems (multi-gigabit networking cores, Micro Blaze soft microprocessor, and the compact Pico Blaze microcontroller). Xilinx also creates custom cores for a fee.The ISE Design Suite is the central electronic design automation (EDA) product family sold by Xilinx. The ISE Design Suite features include design entry and synthesis supporting Verilog or VHDL, place and route (PAR), completed verification and debug using Chip Scope Pro tools, and creation of the bit files that are used to configure the chip.

Xilinx announced the architecture for an Extensible Processing Platform, which licenses the ARM Cortex-A9 MPCore processor for embedded systems designers familiar with the ARM platform. The Extensible Processing Platform architecture abstracts much of the hardware burden away from the embedded software developers point of view, giving them an unprecedented level of control in the developmentprocess. With this platform, software developers can leverage their existing system code based on ARM technology and utilize vast off-the-shelf open-source and commercially available software component libraries. Because the system boots an OS at reset, software development can get under way quickly within familiar development and debug environments using tools such as ARM's Real View development suite and related third-party tools, Eclipse-based IDEs, GNU, the Xilinx Software Development Kit and others.

The platform targets embedded designers working on market applications that require multi functionality and real-time responsiveness, such as automotive driver assistance, intelligent video surveillance, industrial automation, aerospace and defense, and next-generation wireless. Xilinx announced, in early 2011, a new sync product family specifically based on its extensible processing platform.

Following the introduction of its 28nm 7-series FPGAs, Xilinx revealed that several of the highest-density parts in those FPGA product lines will be constructed using multiple dice in one package, employing technology developed for 3D construction and stacked-die assemblies. The technology stacks several (three or four) active FPGA dice side-by-side on a silicon interposer – a single piece of silicon that carries passive interconnect. The individual FPGA dice are conventional, and are flip-chip mounted by micro bumps on to the interposer.

The interposer provides direct interconnect between the FPGA dice, with no need for transceiver technologies such as high-speed Serdes.

4.5 MODEL SIM

ModelSim is a multi-language HDL simulation environment by Mentor Graphics, for simulation of hardware description language such as VHDL, Verilog and systemC and includes a built-in C debugger.

4.5.1 Creating Project

A project is a collection of entity for a Verilog module under specification or test. Projects ease interaction with the tool and are useful for organizing files and simulation settings. At a minimum, projects have a work library and a session state that is stored in a .mpf file. A project may also consist of:

HDL source files or references to source files.

Other files such as READMEs or other project documentations.

Local libraries.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 1, June 2023

References to global libraries.
Start ModelSim with one of the following:
1. Create and change to a new directory to make it the current directory. You can make the directory current by invoking ModelSim from the new directory or by using the File > Change Directory command from the ModelSim Main window.
2. Copy the Verilog files (files with ".v" extension) from the \\modeltech\examples directory into the current directory. Before you can compile a Verilog design, you need to create a design library in the new directory. Since Modelsim is a compiled Verilog simulator, it requires a target design library for the compilation. Modelsim can compile both VHDL and Verilog code into the same library if desired.
3. Invoke Modelsim: from a Windows shortcut icon, from the Start menu.
4. Create library before you compile any HDL code, you'll need a design library to hold the compilation results. To create a new design library, make this menu selection in the Main window: File > New > Library. Make sure Create: a new library and a logical mapping to it is selected. Type "work" in the Library Name field and then select OK. This creates a subdirectory named work - your design library - within the current directory. Modelsim saves a special file named _info in the subdirectory.
5. Compile the counter.v, and tcounter.v files into the work library by selecting Compile > Compile from the menu.
6. Load the design by selecting Simulate > Simulate from the menu.The Simulate dialog appears. Click the "+" sign next to 'work' to see the counter and test_counter design units. (You won't see this dialog box if you invoke vsim with test counter from the command line.)
7. Bring up the Signals, Source, and Wave windows by entering the following command at the VSIM prompt within the main windows, view signals source wave (Main MENU: View >)
8. Add signals now let's add signals to the Wave window with ModelSim's drag and drop feature. In the Signals window, select Edit > Select All to select the three signals. Drag the signals to either the pathname or the values pane of the Wave window.

4.5.2 Basic Verilog simulation

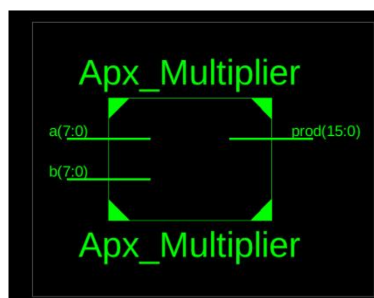
The goals for this lesson are:
1. Compile a Verilog design
2.List signals in the design
3. Examine the hierarchy of the design
4. Simulate the design
5. Change the default run length
6. Set a breakpoint

The project feature covered in A executes several actions automatically such as creating and mapping work libraries. In this part we will go through the entire process so you get a feel for how ModelSim really works

V. SIMULATION RESULTS

Schematic View

Simulation Output



Copyright to IJARSCT
www.ijarsct.co.in
DOI: 10.48175/IJARSCT-11268

IJARSCT

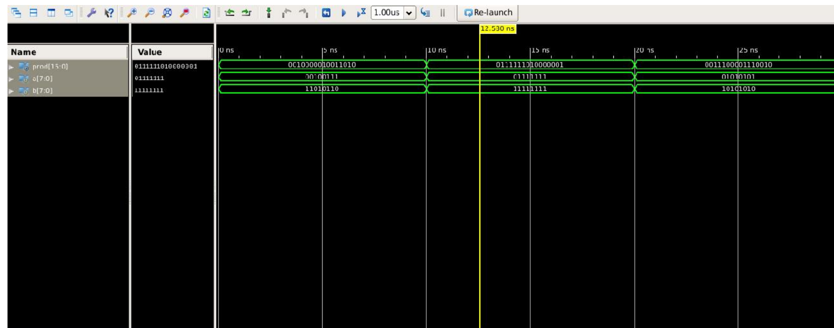
ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.301

Volume 3, Issue 1, June 2023



VI. CONCLUSION

logarithmic multiplication for convolutional neural networks (CNNs), with a The project investigates whether or not it is possible to perform approximate particular emphasis on reducing the amount of power required to do so. The research presents a design for a low power Mitchell's approximate logarithmic multiplier that may be utilized in CNNs to execute multiplication in a manner that is both efficient and accurate.

When compared to traditional methods of multiplication, the experimental findings demonstrate that the design that was proposed offers a reduction in the amount of power consumed as well as the area overhead. Because the proposed design has an accuracy that is on par with that of traditional methods of multiplication, it is an option worth considering for convolutional neural networks (CNNs) that must have both low power consumption and high performance.

Overall, the study highlights the potential of employing approximate logarithmic multiplication in CNNs to produce low power consumption without compromising accuracy or performance. This is accomplished through the use of approximate logarithmic multiplication. The concept that was presented can serve as a foundation for further work in the field of CNNs that are both power-efficient and effective

REFERENCES

[1]. Venkataramani, S.; Chakradhar, S.T.; Roy, K.; Raghunathan, A. Approximate computing and the quest for computing efficiency. In Proceedings of the 52nd Design Automation Conference, San Francisco, CA, USA, 8–12 June 2015. [Google Scholar]

[2]. Breuer, M.A. Multi-media applications and imprecise computation. In Proceedings of the 8th Euromicro Conference on Digital System Design, Porto, Portugal, 30 August–3 September 2005. [Google Scholar]

[3]. Zhang, H.; Putic, M.; Lach, J. Low power GPGPU computation with imprecise hardware. In Proceedings of the 51st Design Automation Conference, San Francisco, CA, USA, 1–5 June 2014. [Google Scholar]

[4]. Shoushtari, M.; Rahmani, A.M.; Dutt, N. Quality-configurable memory hierarchy through approximation. In Proceedings of the 14th International Conference on Compilers, Architecture, and Synthesis for Embedded Systems, Taipei, Taiwan, 9–14 October 2011. [Google Scholar]

[5]. Sarwar, S.S.; Srinivasan, G.; Han, B.; Wijesinghe, P.; Jaiswal, A.; Panda, P.; Raghunathan, A.; Roy, K. Energy efficient neural computing: A study of cross-layer approximations. IEEE J. Emerg. Sel. Top. Circuits Syst. 2018, 8, 796–809. [Google Scholar] [CrossRef]

[6]. Sampson, A.; Deitl, W.; Fortuna, E.; Gnanapragasam, D.; Ceze, L.; Grossman, D. EnerJ: Approximate data types for safe and general low-power computation. In Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation, San Jose, CA, USA, 4–8 June 2011. [Google Scholar]

[7]. Sampson, A.; Nelson, J.; Strauss, K.; Ceze, L. Approximate storage in solid-state memories. In Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture, Davis, CA, USA, 7–11 December 2013. [Google Scholar]

[8]. Nair, R. Big data needs approximate computing: Technical Perspective. Commun. ACM 2015, 58, 104. [Google Scholar] [CrossRef]

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 1, June 2023

[9]. Panda, P.; Sengupta, A.; Sarwar, S.S.; Srinivasan, G.; Venkataramani, S.; Raghunathan, A.; Roy, K. Cross-layer approximations for neuromorphic computing: From devices to circuits and systems. In Proceedings of the 53rd Annual Design Automation Conference, Austin, TX, USA, 5–9 June 2016. [Google Scholar]

[10]. Jiang, H.; Santiago, F.J.H.; Mo, H.; Liu, L.; Han, J. Approximate arithmetic circuits: A survey, characterization, and recent applications. Proc. IEEE 2020, 108, 2108–2135. [Google Scholar] [CrossRef]

[11]. Scarabottolo, I.; Ansaloni, G.; Constantinides, G.A.; Pozzi, L.; Reda, S. Approximate logic synthesis: A survey. Proc. IEEE 2020, 108, 2195–2213. [Google Scholar] [CrossRef]

[12]. Jiang, H.; Liu, C.; Liu, L.; Lombardi, F.; Han, J. A review, classification, and comparative evaluation of approximate arithmetic circuits. ACM J. Emerg. Technol. Comput. Syst. 2017, 13, 1–37. [Google Scholar] [CrossRef][Green Version]

[13]. Garside, J.D. A CMOS VLSI implementation of an asynchronous ALU. In Proceedings of the IFIP Working Conference on Asynchronous Design Methodologies, Manchester, UK, 31 March–2 April 1993. [Google Scholar]

[14]. Wanhammar, L. DSP Integrated Circuits, 1st ed.; Academic Press: Cambridge, MA, USA, 1999; ISBN 9780127345307. [Google Scholar]

[15]. Raha, A.; Jayakumar, H.; Raghunathan, V. Input-based dynamic reconfiguration of approximate arithmetic circuits for video encoding. IEEE Trans. VLSI Syst. 2016, 24, 846–857. [Google Scholar] [CrossRef]

[16]. Ercegovac, M.D.; Lang, T. Digital Arithmetic; Morgan Kaufmann: Burlington, MA, USA, 2003; ISBN 978-1558607989. [Google Scholar]

[17]. Jiang, H.; Liu, C.; Maheshwari, N.; Lombardi, F.; Han, J. A comparative evaluation of approximate multipliers. In Proceedings of the IEEE/ACM International Symposium on Nanoscale Architectures, Beijing, China, 18–20 July 2016. [Google Scholar]

[18]. Vai, M.M. VLSI Design; CRC Press: Boca Raton, FL, USA, 2000; ISBN 978-0849318764. [Google Scholar]

[19]. Mahdiani, H.R.; Ahmadi, A.; Fakhraie, S.M.; Lucas, C. Bio-inspired computational blocks for efficient VLSI implementation of soft-computing applications. IEEE Trans. Circuits Syst. I Regul. Pap. 2010, 57, 850–862. [Google Scholar] [CrossRef]

[20]. Balasubramanian, P.; Maskell, D.L. Hardware efficient approximate adder design. In Proceedings of the IEEE Region 10 Conference, Jeju, Korea, 28–31 October 2018. [Google Scholar]

[21]. Balasubramanian, P.; Maskell, D.L. Hardware optimized and error reduced approximate adder. Electronics 2019, 8, 1212. [Google Scholar] [CrossRef][Green Version]

[22]. Balasubramanian, P.; Nayar, R.; Maskell, D.L.; Mastorakis, N.E. An approximate adder with a near-normal error distribution: Design, error analysis and practical application. IEEE Access 2021, 9, 4518–4530. [Google Scholar] [CrossRef]

[23]. Balasubramanian, P.; Nayar, R.; Maskell, D.L. An approximate adder with reduced error and optimized design metrics. Accepted for publication. In Proceedings of the 17th IEEE Asia Pacific Conference on Circuits and Systems, Penang, Malaysia, 22–26 November 2021. [Google Scholar]

[24]. Balasubramanian, P.; Nayar, R.; Maskell, D.L. Approximate array multipliers. Electronics 2021, 10, 630. [Google Scholar] [CrossRef]

[25]. Balasubramanian, P.; Nayar, R.; Min, O.; Maskell, D.L. Image blending using approximate multiplication. In Proceedings of the IEEE 32nd International Conference on Microelectronics, Nis, Serbia, 12–14 September 2021. [Google Scholar]

[26]. Approximator. Available online: https://github.com/OkkarMin/approximator-tool (accessed on 7 November 2021).

[27]. Approximator Tool Documentation. Available online: https://tool-documentation.vercel.app (accessed on 7 November 2021).

[28]. Zhu, N.; Goh, W.L.; Zhang, W.; Yeo, K.S.; Kong, Z.H. Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing. IEEE Trans. VLSI Syst. 2010, 18, 1225–1229. [Google Scholar]

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 1, June 2023

[29]. Albicocco, P.; Cardarilli, G.C.; Nannarelli, A.; Petricca, M.; Re, M. Imprecise arithmetic for low power image processing. In Proceedings of the 46th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, 4–7 November 2012. [Google Scholar]

[30]. Gupta, V.; Mohapatra, D.; Raghunathan, A.; Roy, K. Low-power digital signal processing using approximate adders. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 2013, 32, 124–137. [Google Scholar] [CrossRef]

[31]. Dalloo, A.; Najafi, A.; Garcia-Ortiz, A. Systematic design of an approximate adder: The optimized lower part constant-OR adder. IEEE Trans. VLSI Syst. 2018, 26, 1595–1599. [Google Scholar] [CrossRef]

[32]. Seo, H.; Yang, Y.S.; Kim, Y. Design and analysis of an approximate adder with hybrid error reduction. Electronics 2020, 9, 471. [Google Scholar] [CrossRef][Green Version]

[33]. Bovik, A. Handbook of Image and Video Processing, 2nd ed.; Academic Press: Orlando, FL, USA, 2005; ISBN 978-0080533612. [Google Scholar]

[34]. Wang, Z.; Bovik, A.C.; Sheikh, H.R.; Simoncelli, E.P. Image quality assessment: From error visibility to structural similarity. IEEE Trans. Image Processing 2004, 13, 600–612. [Google Scholar] [CrossRef] [PubMed][Green Version]

[35]. Chan, W.-T.J.; Kahng, A.B.; Kang, S.; Kumar, R.; Sartori, J. Statistical analysis and modeling for error composition in approximate computation circuits. In Proceedings of the 31st IEEE International Conference on Computer Design, Asheville, NC, USA, 6–9 October 2013. [Google Scholar]

[36]. Balasubramanian, P.; Maskell, D.L. Factorized carry lookahead adders. In Proceedings of the IEEE 14th International Symposium on Signals, Circuits and Systems, Iasi, Romania, 11–12 July 2019. [Google Scholar]

[37]. Synopsys SAED_EDK32/28_CORE Databook. Revision 1.0.0, January 2012. Available online: https://www.synopsys.com/community/university-program/teaching-resources.html (accessed on 27 September 2021).

[38]. Yamamoto, T.; Taniguchi, I.; Tomiyama, H.; Yamashita, S.; Hara-Azumi, Y. A systematic methodology for design and analysis of approximate array multipliers.

Copyright to IJARSCT
www.ijarsct.co.in
DOI: 10.48175/IJARSCT-11268
ISSN 2581-9429 IJARSCT