Journal of Advanced Zoology

ISSN: 0253-7214
Volume 44 Special Issue-2 Year 2023 Page 4603:4612

Implementation of An Extremely Effective Modified Reconfigurable Constant Coefficient Multiplier for Neural Network Architecture using FPGA

Manibharathi.K1, C. Saranya2, P.Kumar3

[1]PG scholar., [2]AssistantProfessor, [3]Professor&Head,
Department of Electronics and Communication engineering,
K. S. Rangasamy College of Technology, Thiruchengode, Chennai, Tamilnadu, India.

Article History

Received: 06 Aug 2023
Revised: 05 September 2023
Accepted:11 November 2023

Abstract

Due to their potential to reduce silicon area or boost throughput, low-precision computations were widely studied to speed up deep learning applications on field-programmable gate arrays (FPGAs). However, the precision suffers as a result of these advantages. proving the superiority of modified reconfigurable constant coefficient multipliers (MRCCMs) over low-precision math in terms of silicon area savings. MRCCMs can be highly optimized for FPGAs because they only use subtractors, adders, multiplexers, and bit shifts (MUXs) to multiply input values by a constrained set of coefficients. suggested a family of MRCCMs designed specifically for FPGA logic components to guarantee their effective use. Create innovative training methods that convert potential MRCCM coefficient models to the weight value ranges of neural networks to reduce information loss due to quantization. As a result, hardware can still use MRCCMs while keeping high accuracy. Utilizing the ResNet-18, ResNet-50, and AlexNet networks, illustrates the advantages of these methods. The resulting implementations reduce resource consumption by up to 50% compared to conventional 8-bit quantized networks, which results in substantial speedups and power savings. All other methods with MRCCMs accomplish accuracy that is at least comparable to an 8-bit uniformly quantized system while significantly reducing resource usage, while our MRCCM has the lowest consumption of resources and surpasses 6-bit fixed point accuracy. Similar to that, this study compared the MRCCM approach using Xilinx FPGA on various sizes of MRCCM like ADD-3, ADD-4, and ADD-2.

Keywords: *neural networks, reconfigurable, multiplier, gate arrays, training technique.*

1. Introduction

One of the most widely applied methods in digital signal processing is multiplication with constants. (DSP). In DSP applications, which were formerly managed by Application Specific Integrated Circuit (ASIC) versions, FPGAs are growing in popularity. The rising cost of ASIC manufacturing and the

adaptability offered by FPGAs' reprogrammability are the causes of this tendency. FPGA designs are generally larger, slower, and more power-hungry than comparable ASIC realisations as a trade-off for the reprogrammability of these devices. Therefore, optimised DSP code designs for FPGAs are becoming increasingly crucial. One of the reasons embedded multipliers are found in FPGAs is for this purpose. They are undesirable, though, due to their stringent word limits and availability limitations for those pre-existing coarse-grained blocks. Limited supply is especially important in industrial applications where it is critical to choose affordable FPGAs with few embedded multipliers and other design components are vying for DSP resources. Alternative logic-based constant multiplication techniques that are not reliant on embedded special purpose hardware are needed to close the development gap to the realisation of an ASIC. As a consequence, the shift-add circuit, which employs constant multiplication, is the subject of extensive research on how to use it most effectively. However, it is equally essential to use smaller, more focused constant factors during run-time as opposed to larger, more comprehensive ratios. Reconfigurable constant multipliers are used to create hardware-efficient run-time adaptable filters, such as for adaptive control and video coding apps. It is shown how an application with stringent resource and reconfiguration time constraints requires highly optimised RCMs on FPGAs. The control system of the particle accelerator uses an FPGA as a co-processor there.

Because of their superior prediction abilities, convolution neural networks (CNNs) have been extensively used in contemporary computer vision applications. Researchers frequently favour larger networks with more complex processing and memory requirements. It has been shown that field-programmable gate array (FPGA) implementations outperform central processing unit (CPU) and graphics processing unit (GPU) implementations in terms of latency and resource efficiency. They allow customised data paths that boost parallelism and require less data movement than CPU/GPU technologies. Through the use of custom hardware tailored to specific applications, this design flexibility provides the chance to maximise system efficiency.

According to[1] the hardware platform uses four Altera Stratix III field-programmable gate arrays and considers both the cellular and network layers. This makes it possible to execute a large-scale neural network with realistic biophysical dynamics.

said that a number of uses for picture processing heavily rely on two-dimensional convolution[2]. Using different kernel sizes for image convolving improves the total performance of image processing applications. As a result, it is imperative to construct a reconfigurable convolver that takes the list of recommended kernel sizes into consideration. The proposed model performs considerably better than previous works in terms of resource utilization, accuracy, and maximum clock frequency, according to simulations and hardware implementations on FPGA[3]

In reaction to this discovery, we propose LUTNet, a complete hardware-software framework for the design of space-efficient FPGA-based neural network accelerators using native LUTs as inference operators[4].

In the area of computer vision, convolutional neural networks (CNNs) have recently achieved great success. The weights in a CNN layer are represented by a limited number of quantized segments in this approach [5]. Hardware acceleration of CNNs using these devices has become a viable strategy. The most effective method for computing convolution for smaller filter sizes is Winograd filtering based convolution [6].

Computer, biological, and electronic disciplines are all combined in the comparatively new interdisciplinary field of neuromorphic research. Neuromorphic systems, made up of software and hardware systems, are used to create neural networks that are built on the capabilities of the human brain [7]. 3-D convolutional neural networks are extensively used in computer recognition software. (3-D CNNs). The bulk of earlier work in this area only concentrated on the design and optimisation of accelerators for 2-D CNNs, and there have been few attempts to accelerate 3-D CNNs on FPGA [8].

Notable features of FPGAs include variant peripheral support and flexibility to partial or major design modifications. However, due to their high power consumption, large die area, and poor performance, FPGA-based designs are not commonly used in low-power embedded devices with soft-core processors [9].

According to [10] a modified leaky integrate-and-fire (LIF) model is used to realise the hierarchical SNN, which achieves both high efficiency and minimal hardware usage. CNNs result in improved results but require more memory and processing power. Therefore, convolutional neural network reasoning is frequently carried out in centralised high-performance systems[11].

Convolutional neural networks (CNNs) have lately been accelerated using a variety of accelerators on FPGAs in many domain-specific application areas. Additionally, the theoretical computational burden of CNN inference has been considerably reduced by some optimisation methods, such as network sparsity and quick algorithms[12]. Multipliers are crucial circuit elements for increasing the energy economy of CNN and image processing hardware solutions[13]. Quantization is a crucial form of optimisation for floating-point deep neural network (DNN) accelerators. Digital signal processing (DSP) blocks on field-programmable gate arrays are ineffectively used when the accelerator precision is considerably lower than the DSP precision [14].

To enable a range of DNN models, efficient DNN inference implementations on edge-computing platforms, such as ASICs, FPGAs, and embedded systems, are carefully investigated. Because DNN models are so large and demand so much computation, model compression is an essential step in the deployment of DNN models on edge devices [15].

2. Methodology

Proposed system

In a field programmable gate array, a deep learning app will involve very low-accuracy arithmetic operations, which will require more silicon area. To improve performance in the silicon area and use little accuracy in arithmetic applications, an MRCCM was introduced in the suggested work. With only subtractors, adders, multipliers, and bit shifts as their only coefficients, MRCCMs multiply the incoming value by a small number of coefficients. Utilizing a low power and space-efficient carry select adder, this way to lower the logic area in MRCCM-based computations will result in a greater reduction in silicon area for deep learning apps. (figure 1) In a similar vein, this study compared the currently used ripple carry adder-oriented MRCCM approach via Xilinx FPGA, on various sizes of MRCCM including ADD-3, ADD-4, and ADD-2.

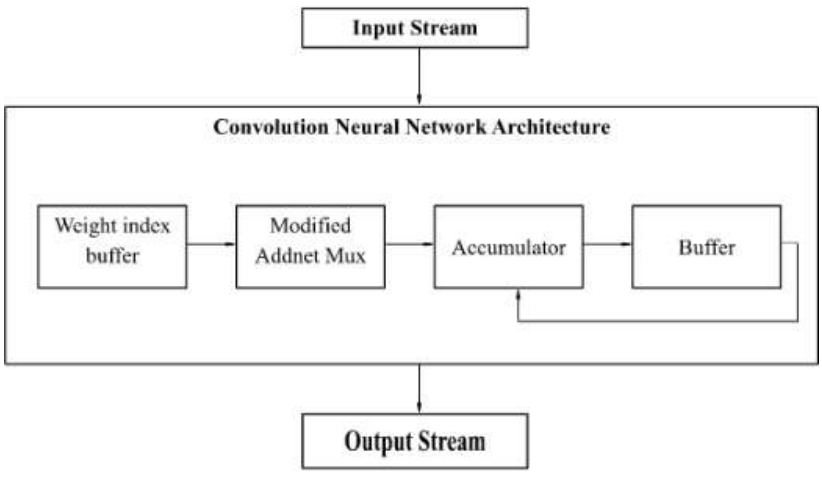


Figure 1 Architecture of VLSI

Reconfigurable Multipliers

A device known as a constant coefficient multiplier calculates y = cx using only additions, bit shifts, and subtractions, where c is a fixed value. To calculate y = 6x, for instance, using adds and shifts. A circuit known as an MRCCM computes y = cs xs, where cs is a discontinuous coefficient set C = c0, c1,..., cN1 selected from a [log2(N)] bit choose signal. The most common methods for realizing MRCCMs include subtractions, additions, MUXes, and bit shifts. A sample of an MRCCM with the coefficient set C = 12305, 20746 is shown in Fig. 2.

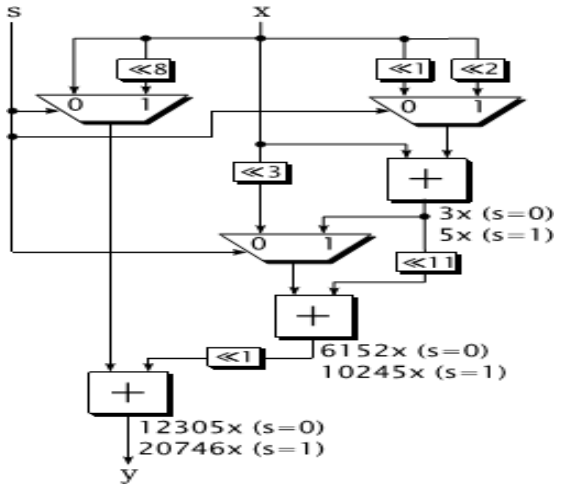


Figure 2 Programmable multiplier with coefficients fixed to 12305 and 20746

FPGA Multipliers Mapping

It looked for building elements that effectively map to an FPGA's logic fabric. Although comparable circuits can be obtained for other FPGAs, our concepts are optimized for the most recent Xilinx UltraScale/UltraScale+ FPGAs). As a result, we made sure the MUXes fell into the same LUTs that were necessary for the adders when designing our base structures. The two base configurations used to construct the MRCCM units in this paper are depicted in Fig. 3.

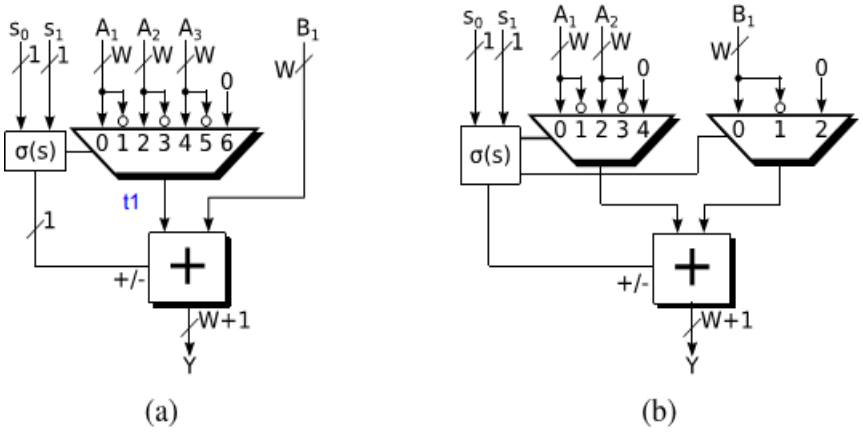


Figure 3 Reconfigurable multiplier base configurations (a) Topology A and (b) Topology B.

Experimental Design

XILINX

A provider of customizable logic devices was Xilinx. It was the first semiconductor firm with a flexible manufacturing approach, and it is credited with creating the FPGA. The business was established in Silicon Valley in 1984 and has offices in California, San Jose, Dublin, Tokyo, the United States, Ireland, and Singapore. Corporate headquarters for the business are located in North America, Asia, and Europe. For the generation and evaluation of HDL designs, Xilinx ISE is a discontinued piece of software from the company. It was mainly used to create embedded code for the FPGA and CPLD IC (integrated circuit) product families from Xilinx. Xilinx Vivado replaced it in the market.

ISE DESIGN SUITE: SYSTEM EDITION

The top high-level tool for constructing high-performance digital signal processing (DSP) systems using Xilinx programmable devices is called System Generator for DSP. It offers system modeling and automatic generation of code from MATLAB and Simulink.

VERILOG

One of the two primary Hardware Description Languages (HDL) utilized by academic and industrial hardware programmers is Verilog. Since most computer and electrical engineers study C in college, Verilog is very similar to C and is well-liked by them. Cadence had its private language called Verilog HDL. With the hope that the language would gain more acceptance, the market for software products linked to Verilog HDL would expand more quickly, Cadence was driven to make the language available to the public domain. Cadence acknowledged the demand from Verilog HDL customers for other service and software providers to adopt the language and create Verilog-compatible design tools.

Register-Transfer Level

Designs that use the Register-transmission Level define the properties of a circuit using processes and data transmission between the registers. Any code that can be synthesized is considered an RTL code, according to modern meaning.

Behavioral level

An algorithmic system is described at this level. (Behavioural). Each algorithm is sequential, which implies that each of its commands is carried out one at a time. The primary components are blocks, functions, and tasks. The design's structural implementation is not taken into consideration.

Multiple Library Flow

ModelSim makes two applications of libraries: two ways: 1) as a resource library; 2) as a local working library containing the compiled form of your design. As you alter your design and recompile, the items in your working collection will also change. A resource library usually contains static components that are used in your design. It can build its resource libraries or use ones that are already available from other design teams or outside sources. When the design is built, it defines which data libraries are to be employed, and there are guidelines to determine the order in which they are examined. When your gate-level development and testing bench is built into the working repository and the design refers to gate-level models in a different resource library, you are using both a resource library and a working library. (figure 4) The fundamental stages for modeling with numerous libraries are depicted in the diagram below.

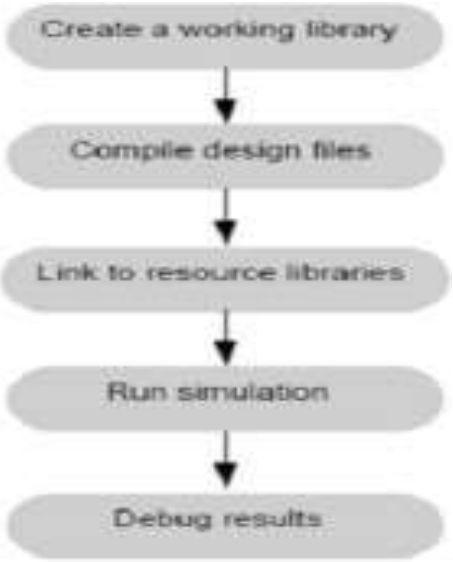


Figure. 4 Flow chart of Multiple Library

Gate Level

Logical links and their timing features inside the logical level define the features of a system. Every signal is distinct. They can have the discrete logical numbers "0," "1," "X," and "Z." The predefined logic primitives are the usable actions. (basic gates). His netlist is utilized for gate-level backend and modeling, and tools like synthesis tools are used to create gate-level code.

Tools of Debugging

Numerous tools are available in Model Sim to analyze and fix your design. The following lessons will address several of these instruments, including:

1. Using tasks

2. utilizing several frameworks

3. placing breakpoints and navigating the source code

4. Observing patterns and timing

5. Examining and starting recollections

6. Using the Waveform Editor, make stimuli

7. Simulator automation

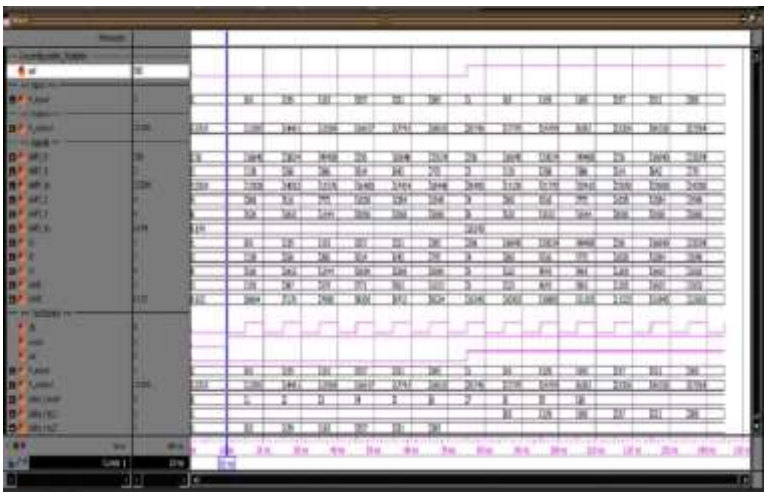
3. Results and Discussion

XILINX Simulation of Reconfigurable Multipliers

When the I/Q block was built on a XILINX VirtexE2000-6 FPGA for the 16-bit computation, the implementation expenses of the alternatives were 33% and 50%, respectively, of the device utilization. Power usage data is comparable. Power usage drops by 33% when switching from 28-bandwidth to 19-bit-width. A study against a specialized hardware multiplier was made using a configuration on a XILINX XC4000XV FPGA.(figure 6) Researcher observed the maximum bandwidth for a typical

1616 multiplier of 51 MHz for a single 32-bit product in a single clock cycle and a maximum bandwidth for the proposed multiplier of 42.8 MHz for 16 32-bit products in three pipeline cycles.

Figure 5 The output of the suggested 16-bit reconfigurable multiplier's test with the coefficients set to



12305 and 20746

Modeling Base Topology A Used To Construct Variable Multipliers

The reduction in silicon area might be utilized as well to increase parallelism, which would increase throughput, decrease latency, or allow the design to work on a smaller FPGA. For instance, although the Conv Layer and AlexNet methods already calculate one PE for each output feature map, the researcher can raise pl and calculate more output map feature pixels concurrently to cut down on the quantity of PE iterations needed to compute a layer. Most big neural networks might be accelerated using this simple optimization. Such networks typically have a lot of inherent redundancy and high processing demands, and because of resource limitations, FPGA accelerators employ some kind of layer folding. This is particularly true for implementations with higher accuracy when precision maintenance is crucial. As a result, the AddNet multiplier was a tool that can be used very broadly to increase the parallelism of FPGA NN designs that are already in use.

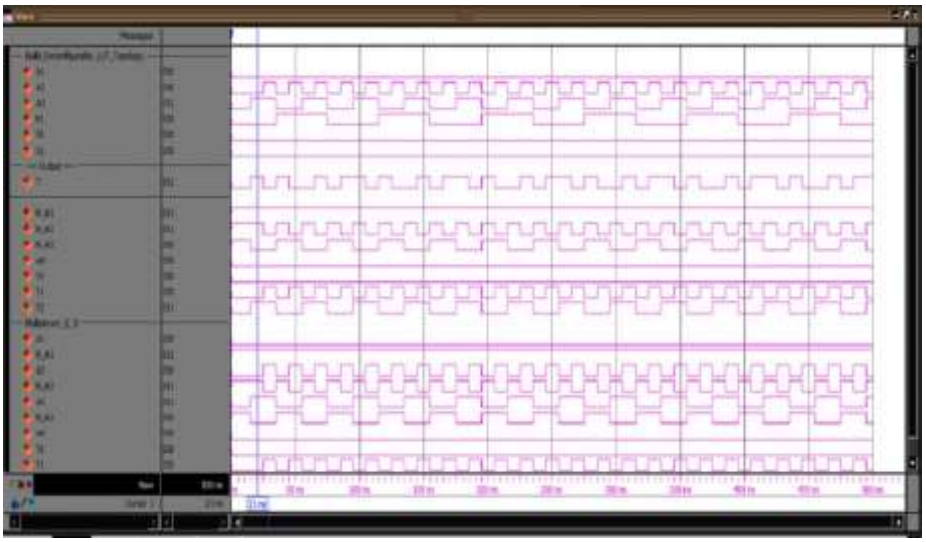


Figure 6 Base Topology Output from Simulation used to create a variable factor

Base Topology B Simulation Used To Create Reconfigurable Multiplier

Figure 7 shows the maximum multiplier units versus frequency values for various parameter configurations. The only additional processing time is needed to reconfigure the LUTs when the constant needs to be changed. The setup time is displayed for two programming schemes, of various constant sizes, and the setup time. About 20% more effort is needed to reconfigure the direct scheme than the reverse one. The restructuring procedure moves very quickly—on the order of 100 ns. The initial 2 units also contain two logic modules that control both the output stream, or the outcome, and the inbound data stream, which includes the multiplicand and multiplier operands. (figure 7) The IU specifically handles the data transmission from and to the bus. Typically, the output and input bus bandwidth and data word length are distinct, so an internal control logic circuit is created to address this problem.

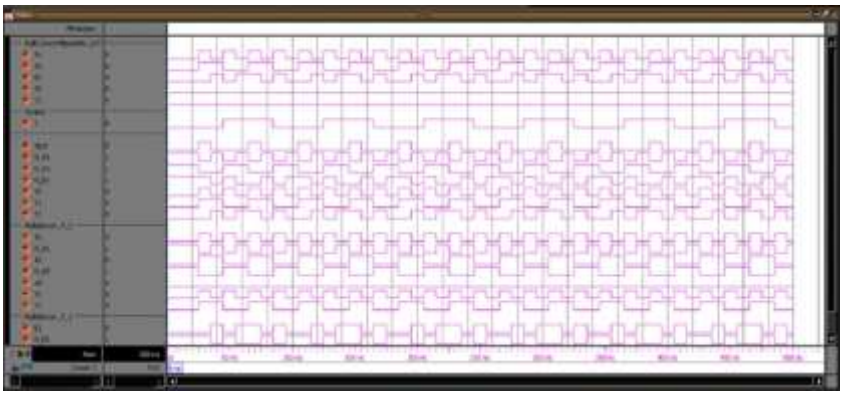


Figure 7 Base topology B simulation data for the reconfigurable multiplier

A plan for a reconfigurable grid of multipliers was offered. The processor may be changed or at runtime modified to exchange matrix size for bandwidth. For example, it takes a total of (s/m)2 =4k mm multipliers to build an array of multipliers with the size s=maximum input unit bitwidth=2k+2 (k=0,1,2,...), and each pipeline phase can handle an input array combined with the size h=(s/b)2 b-bit components in parallel. (table 1) By reusing components, the design achieves superiority. This fact makes it possible to use a portion of the structure, for instance, when electricity reduction is required.

Table 1 Number of cases where the proposed optimal shift reassignment allows for the preservation of a certain number of 2:1 multiplexers compared to the original DAG fusion solution.

saved multiplexers	number of configurations															
	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	sum
0	**64**	26	20	13	20	7	5	6	5	6	2	2	1	1	-	178
1	27	**35**	26	31	12	12	15	11	8	9	10	6	8	2	7	219
2	8	22	**29**	**29**	**28**	**26**	23	20	17	20	17	18	13	13	7	290
3	1	11	16	10	23	23	**25**	16	**24**	**22**	**25**	**20**	17	14	15	**262**
4	-	4	8	15	8	19	18	**24**	19	17	16	16	**23**	22	18	227
5	-	2	0	1	5	12	6	11	12	10	16	17	13	**23**	**24**	152
6	-	-	1	1	3	1	6	6	6	14	10	15	18	10	12	103
7	-	-	-	-	1	-	2	6	7	0	1	2	4	8	10	41
8	-	-	-	-	-	-	-	-	1	1	3	2	3	3	3	16
9	-	-	-	-	-	-	-	-	1	1	-	2	-	4	2	10
10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1
11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1

4. Conclusion

Reconfiguration techniques used in RCM creation are LUT-based and multiplexer-based. However, because Reconfigurable Constant coefficient multipliers are limited to a specific number of target values, their use is currently only possible in certain DSP application areas, such as linear changes and digital filtering. Reconfigurable Constant coefficient multipliers with coefficient sets that closely resemble the desired range of neural network weights are designed using the suggested technique, Add Net. Additionally, we create a technique for preparing neural networks to utilize RCCMs. According to this research, Add Net exceeds low accuracy computation in terms of precision for a particular silicon area budget when optimizing neural networks.

REFERENCE

Cheng, Wei, Ing-Chao Lin, and Yun-Yang Shih. 2022. "An Efficient Implementation of Convolutional Neural Network With CLIP-Q Quantization on FPGA." *IEEE Transactions on Circuits and Systems. I, Regular Papers: A Publication of the IEEE Circuits and Systems Society* 69 (10): 4093–4102.

Dehghani, Abbas, Ali Kavari, Mahdi Kalbasi, and Keyvan RahimiZadeh. 2022. "A New Approach for Design of an Efficient FPGA-Based Reconfigurable Convolver for Image Processing." *The Journal of Supercomputing* 78 (2): 2597–2615.

Deng, Bin, Yanrong Fan, Jiang Wang, and Shuangming Yang. 2021. "Reconstruction of a Fully Paralleled Auditory Spiking Neural Network and FPGA Implementation." *IEEE Transactions on Biomedical Circuits and Systems* 15 (6): 1320–31.

Farsa, Edris Zaman, Arash Ahmadi, Mohammad Ali Maleki, Morteza Gholami, and Hima Nikafshan Rad. 2019. "A Low-Cost High-Speed Neuromorphic Hardware Based on Spiking Neural Network." *IEEE Transactions on Circuits and Systems II: Express Briefs* 66 (9): 1582–86.

Jokar, Ehsan, Hadis Abolfathi, Arash Ahmadi, and Majid Ahmadi. 2019. "An Efficient Uniform-Segmented Neuron Model for Large-Scale Neuromorphic Circuit Design: Simulation and FPGA Synthesis Results." *IEEE Transactions on Circuits and Systems. I, Regular Papers: A Publication of the IEEE Circuits and Systems Society* 66 (6): 2336–49.

Kala, S., Jimson Mathew, Babita R. Jose, and S. Nalesh. 2019. "UniWiG: Unified Winograd-GEMM Architecture for Accelerating CNN on FPGAs." In *2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID)*, 209–14. IEEE.

Rasoulinezhad, Seyedramin, Hao Zhou, Lingli Wang, and Philip H. W. Leong. 2019. "PIR-DSP: An FPGA DSP Block Architecture for Multi-Precision Deep Neural Networks." In *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 35–44. IEEE.

Shen, Junzhong, You Huang, Mei Wen, and Chunyuan Zhang. 2020. "Toward an Efficient Deep Pipelined Template-Based Architecture for Accelerating the Entire 2-D and 3-D CNNs on FPGA." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39 (7): 1442–55.

Véstias, Mário P. 2019. "A Survey of Convolutional Neural Networks on Edge with Reconfigurable Computing." *Algorithms* 12 (8): 154.

Wang, Erwei, James J. Davis, Peter Y. K. Cheung, and George A. Constantinides. 2020. "LUTNet: Learning FPGA Configurations for Highly Efficient Neural Network Inference." *IEEE Transactions on Computers. Institute of Electrical and Electronics Engineers* 69 (12): 1795–1808.

Wang, Xuan, Chao Wang, Jing Cao, Lei Gong, and Xuehai Zhou. 2020. "WinoNN: Optimizing FPGA-Based Convolutional Neural Network Accelerators Using Sparse Winograd Algorithm." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39 (11): 4290–4302.

Yang, Shuangming, Bin Deng, Jiang Wang, Huiyan Li, Meili Lu, Yanqiu Che, Xile Wei, and Kenneth A. Loparo. 2020. "Scalable Digital Neuromorphic Architecture for Large-Scale Biophysically Meaningful Neural Network With Multi-Compartment Neurons." *IEEE Transactions on Neural Networks and Learning Systems* 31 (1): 148–62.