International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 04 | Apr 2021 www.irjet.net p-ISSN: 2395-0072

Automatic Generation of Hardware Architecture of 2D Convolvers.

A. Al-Marakeby

Systems and Computers Dept. Faculty of Engineering, Al-Azhar Univ. Cairo, Egypt

---

Abstract - *Convolution is a very important operation in signal processing, image processing, and computer vision. The convolution operation is implemented easily on software, but designing a custom hardware architecture for 2D convolution helps in obtaining fast, and power efficient systems. Many hardware architectures are proposed, however there is no a suitable architecture for all cases. The most efficient architecture depends on many factors, which add a large effort on the designer to analyze and design an efficient architecture. IP cores and HLS give a general solution which needs optimizations and customization to work efficiently for a specific problem. In this research, an automatic generation system is introduced which helps the designer to easily analyses the performance and accuracy of the architecture. Also, this system automatically generates an architecture to realize the 2D convolution based on some parameters and customizations given by the designer. The automatic generation system starts by receiving a kernel coefficients or an equation as an input, then generates a Verilog code as an output which can be realized on FPGA or ASIC. The experimental results show that, not only the automatic generation tool reduces the designer efforts, but also in many cases it generates more efficient systems than the human-based designs.*

Key Words: 2D convolution, image processing, automatic generation, FPGA.

1. INTRODUCTION

2D Convolution are commonly used in very large applications ranging from image processing and computer vision to CNN (Convolutional Neural Networks) and deep learning. When working with low power and small size systems the hardware implementation of 2D convolution is more efficient than software implementation. However, designing an efficient hardware architecture of 2D convolvers is not an easy task. The architecture depends on many factors, parameters, optimizations, and customizations. Usually, the designer will spend a large time to take all these variations into consideration. Another solution is to use IP cores and HLS which facilitates the design and reduces the time and efforts. However, these ready systems are not efficient, and they require many customizations and optimizations to match the given problem requirements. For example, a 5×5 Gaussian kernel can be implemented using the architecture shown in fig.1.

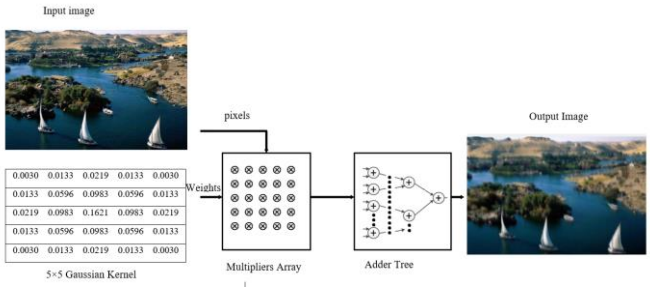


Fig -1: 2D Convolution using parallel architecture.

This architecture consists of a multiplier array contains 25 floating point multipliers and an adder tree contains 24 floating point adders. This architecture can be considered as a general 2D convolution architecture for any 5×5 kernel, but it can be improved dramatically depending on many factors. One main factor is the decision by the designer to use fixed point instead of floating point with a reduced accuracy (sometimes without accuracy losses depending on the kernel coefficients). Floating point arithmetic consumes a lot of hardware circuitry, and fixed-point approximations are more suitable and efficient for hardware implementations [1][10]. It is required to select the suitable bit width for fixed point approximation which affects both of the accuracy and hardware complexity.

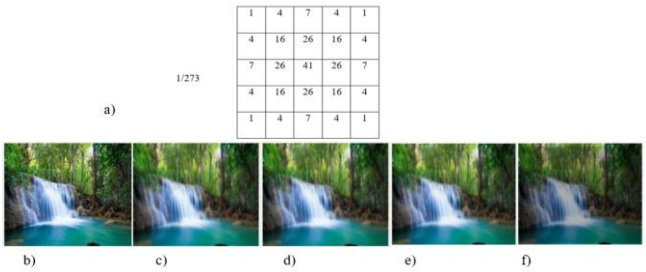


Fig -2: a) Fixed point approximation for 5×5 Gaussian kernel b)input image c) output image using floating point d) Fixed point 16 bit Error=0.003 e) Fixed point 8 bit Error=0.547 f) Fixed point 4 bit Error=10.14

In fig. 2 the Gaussian kernel is approximated using different bit widths and the accuracy is given for each approximation. The losses should be analysed by the designer to carefully

select the suitable approximation. In addition to that, the efficient design of 2D convolution circuits should consider the separable kernels, which can improve the convolution operation both in terms of speed, and resource utilization [7][11]. Using separable kernels reduces the number of multipliers for 5×5 kernels from 25 to 10 , and reduces the number of adders from 24 to 8. For large kernel sizes , very large reduction in circuit complexity can be achieved using separable kernels. Also the designer should take into consideration the special coefficients of the kernel such as zeros and power of two coefficients. The multiplication by zero not only save one multiplier but also affects the addition and change the adder tress architecture. The multiplication by power of two factors can be implemented easily using simple shift operation instead of using complex multiplication circuits. The Soble kernel shown in fig.3 can be implemented without using any multiplication circuit, because the kernel coefficients are limited to zeros and special case numbers. All these design efforts and optimization required for efficient design of one kernel are repeated if the kernel size or kernel coefficients are changed. If only, the sigma value of Gaussian kernel is changed the complete design is repeated. For large kernel these optimizations are vey tedious for humans. In this work, an automatic generation system for 2D convolvers is presented. The input of this automatic generation system is the kernel coefficients or the equation. The output is Verilog code which can be realized on FPGA or ASIC. This automatic generation system analyse the conversion of floating point values into fixed point with a specific bit widths and estimate the quantization effects to allow the designer to choose between accuracy, power , and resource utilization. Also, the system is equipped with different architectures to utilize special cases optimizations such as separable convolution. The system can detect zeros values in the kernel and automatically modify the architecture to remove unneeded units.

2	2	4	2	2
1	1	2	1	1
0	0	0	0	0
-1	-1	-2	-1	-1
-2	-2	-4	-2	-2

Fig -3: Sobel Kernel Sb_y (5×5)

Also, the system can deal with special cases multiplication such as power of two to replace complex multiplier by more simple units. Many optimizations such as pre-computed values and pipelining are programmed in this system to automatically deal with them. The paper is organized as follows. Section 2 presents related work. Section 3 discusses the different optimizations and customization used in designing efficient 2D convolovers. Section 4 introduces the automatic generation system. Experiments and results are given in section 5, and finally a conclusion is given in section 6.

2. RELATED WORK

A lot of techniques are proposed to efficiently design hardware architectures of 2D convolovers. Some architectures are designed to calculate the convolution for fixed and specific kernels only[7][10][11], while other architectures are designed to work with different kernels by loading the kernel values in registers[3][8][12]. Dynamic kernel architectures are very important in hardware implementation of CNN. Debasish and Susanta presented a hardware architecture for 16-bit, 5×5 fixed-point 2D Gaussian kernel[10]. They proposed two filters, one using separable kernel and the other using generalized kernel. They have achieved an improvement in terms of frames per second and max speed using the proposed architectures[10]. In [11] a one-dimensional (1D) convolution processor with reconfigurable architecture is Implemented. The proposed architecture, can simultaneously execute convolution operation with different kernels, with a maximum kernel size of up to 24 × 24 with 194 frames per second [11]. Kaiyuan Guo .et. al. proposed Angel-Eye, a CNN accelerator architecture, with a compilation tool and data quantization strategy [3]. The compilation tool is used to map a certain CNN model onto a hardware. Optimization is done on compilation to utilize the on-chip cache and the parallelism between calculation and data I/O[3]. Automatic generation systems are proposed which facilitate the design of architectures for a specific problem and generates the Verilog or VHDL codes[2][5][9]. In [9], a Spiral hardware generation framework for linear transforms is introduced. The system takes a problem specification as input and also some directives that define characteristics of the required datapath. The system automatically generates an algorithm, maps it to a datapath, and outputs the Verilog description[9]. In [5] , an automatic generation tool is proposed which allows developers, through a configurable user-interface, to automatically generate VHDL code for their desired CNN model. The generated code or architecture is reconfigurable ,modular, massively parallel, fully pipelined, and adaptive to different CNN models. They tested the system using “LeNet” and “AlexNet”[5]. The previous systems introduce techniques

of manual optimizations of convolovers or automatic generation systems but for other problems. In our system, the different optimizations, customizations, and analysis are executed automatically using the proposed automatic generation systems. The system works with fixed kernels only which generates a circuit for implementing specific kernel coefficients.

3. OPTIMIZATIONS AND CUSTOMIZATIONS

In this section a set of optimizations is introduced which enhance the performance and resource utilization of a 2D convolver circuit. The automatic generation system discussed in section 4 is prepared with these optimizations to generate fast and efficient systems. Also, the designer can choose from many customizations and options according to the given problem. These customizations and their effects on accuracy, power, speed , and resource utilizations are discussed here.

3.1 Fixed Point Approximation

Many kernel coefficients are represented as floating point numbers which requires floating-point arithmetic units (multiplications and addition). The fixed-point arithmetic units are faster and simpler than floating point units, but with a lower accuracy. In fig.2 an image is convolved using floating point and fixed point with different quantization levels. The output image generated using floating point representation is considered as a reference image and the difference between this image and other images represents the error. The error is given by equation 1.

$$Error = \frac{1}{w.h}\sum_{x=1}^{w}\sum_{y=1}^{h}(g(x,y) - \hat{g}(x,y))^2 \qquad (1)$$

Where g is the image obtained by applying floating point kernels , $\hat{g}$ is the image obtained by applying fixed point kernels, *w* the image width , and h is the image height.

3.2 Separable Architectures

The resources required for 2D convolution can be reduced by using separable kernels [7]. A separable kernel of size m×m can be decomposed into a vertical m ×1 kernel and a horizontal 1×m kernel, thus, only 2m multiplications and (2m −2) additions are required per pixel. The 3×3 kernel given in (2) can be divided into two separable kernels of size 3×1 and 1×3 which performs the same operations of the original kernel but with simpler and faster circuitry. For large kernels, the number of units required for 2D convolution can be 10 times the units for separable kernels as shown in fig.4.

$$\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} = \begin{bmatrix}1\\ 2\\ 1\end{bmatrix} * [1 \quad 2 \quad 1] \qquad (2)$$

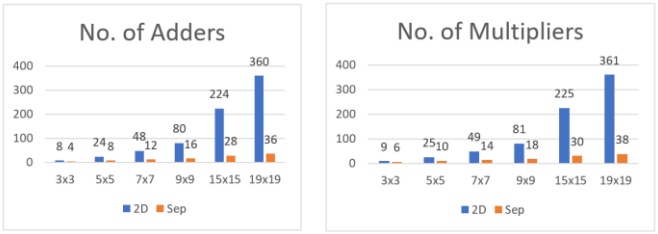


Fig -4: Number of multipliers and adder required for convolution using 2D kernels and Separable kernels.

3.3 Special Cases Coefficients

Multipliers units are designed to multiply any two variable numbers, but when one operand is constant, a great reduction in circuit complexity can be obtained. For constant multiplication the multiplier complexity can be reduced, or it can be completely discarded for some special cases [4][6]. For example, there is no meaning to use a multiplier unit for zero coefficients multiplication. These multipliers should be discarded, and the adder tree should be modified to match the number of inputs after discarding these zero operands.

3.4 Pre-Computed Values

It is common in convolution kernels to find similar coefficients. The multiplication can be discarded if a pre-computed value is used instead of repeating the same operation with the same operands. The number of registers required to store the pre-computed values depends on the location of repeated coefficients.

3.5 Pipelining

It is required to increase the max frequency clock of the system so; pipelining is very important in designing the hardware architecture of the convolvers. The multiplication stage is in parallel while the adder tree works in different levels where each level waits the results of the previous levels.

4. AUTOMATIC GEENRATION SYSTEM

The optimizations introduced in section 3 are difficult to be analyzed and designed humanly specially for large kernels. Also, the manual design is not adaptable where the simple modification requires repeating the analysis and design process. For example, the convolver for 3×3 Gaussian kernel with σ=1 is not efficient when σ =2. In this section, the automatic generator system is introduced which helps the designer to easily analyze and generate the hardware architecture of convolovers for arbitrary kernel sizes with different coefficients. The input of the system is the kernel either by values or by formula as shown in fig.5. The system stores some common kernels equations such as Gaussian, , Laplacian , and Sobel. . Instead of feeding the system with values, the designer can choose one of the available kernel formulas and choose the size and some other parameters to automatically generate the kernel coefficients as shown in fig.6. Either the kernel coefficients are generated automatically or entered manually, the analyzer will start analyzing these coefficients. The analysis process contains many steps which extracts useful information to optimize the architecture. These information include zero coefficients, power of 2 coefficients, repeated values ,...etc. Also The analysis process helps the designer to select design points through drawing quantization error curves.

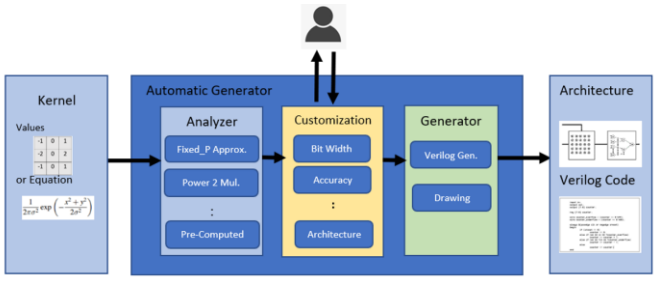


Fig -5: Automatic generation system

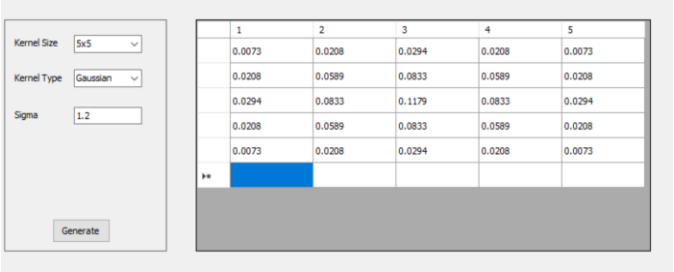


Fig -6: Kernel Coefficients generation

The customization subsystem allows the designer to select some parameters such as accuracy, bit widths, and architecture. Based on the designer choices, the generator generates the Verilog code for the customized architecture. In fig.7 the accuracy curve indicates the effects of bit widths on the accuracy. This curve helps the designer to choose the appropriate accuracy and automatically generates fixed point values for his choice.

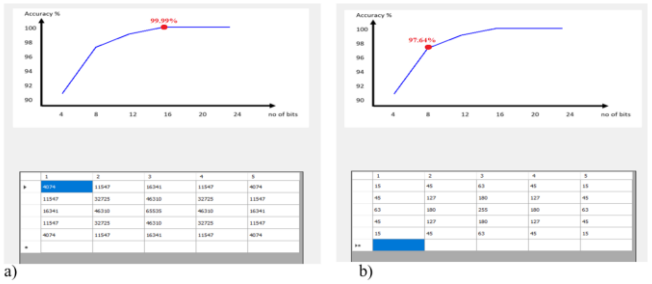


Fig -7: Quantization Error Analysis and automatic fixed-point coefficients generation a) 16 bit b) 8 bit

The Verilog code generator consists of different modules for generating port definition and body of the different modules. In fig.8 (a) a 7x7 Gaussian kernel architecture is automatically generated using the generation system. The multiplier array is sketched and the adder tree is partially sketched for its large size. The Verilog code for this system is also generated automatically.

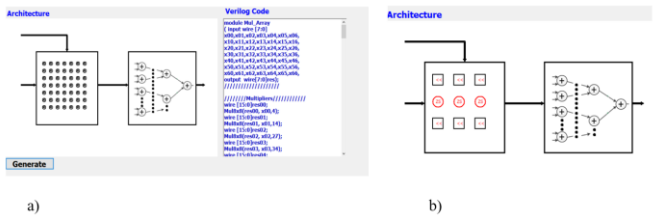


Fig -8: automatic generation of a) 7x7 Gaussian kernel sigma=1.5 b) 3x3 Sobel Kernel

In Fig.8 (b) a 3x3 Sobel kernel architecture is automatically generated. The Sobel kernel consists of 3 zeros which allow the removal of the multiplier units in corresponds locations. The red circle contains "ZS", refers to a zero skipped multiplier in this location. The remaining coefficients of the Sobel kernel are 1's or -1's , which can discard the multiplier units also. The ones or multiplier of 2 coefficients are represented here using a rectangle contain "<<" shift symbol. Both ones and zeros coefficients are discarding multiplier units, but they are different in the adder tree modules. Zero coefficients reduce the number of inputs to the adder tree

and they don't contribute to the addition process, while one coefficients contribute to the addition process. Fig. 9 illustrates code generated for 3x3 Sobel kernel, where the module "adder tree " port definition contains only 6 inputs because 3 inputs are discarded which correspond to the 3 zeros in the Sobel kernel.

```
////////Adder Tree//////////////////
module Adder_Tree
( input wire [16:0]A0,A1,A2,A3,A4,A5,
 output reg[7:0] res);
  \\ Level0 \\
wire [15:0] A0_1;
Add8x8(A0_1,A0,A1)
wire [15:0] A2_3;
```

Fig -9: Adder tree automatic code generation for Sobel Kernel

5. RESULTS AND DISCUSSION

The automatic generation system is tested using different kernels with different sizes. Two references are used to measure the efficiency and performance of the automatically generated architecture. The first system called REF1 represents the standard architecture shown in fig.2 using floating point multiplication units and addition units. The second system called REF2 represents the same standard architecture but using 16 bit fixed point. The automatic generated system is compared with these two reference systems.

TABLE.1

Accuracy and Resources utilization for Gaussian Kernel 3x3

Resources	REF1	REF2	Aut_Gen
Logic Elements	3,629	782	650
Registers	3,262	450	320
Embedded Multipliers	63	18	18
Accuracy	100%	99.6%	99.6%

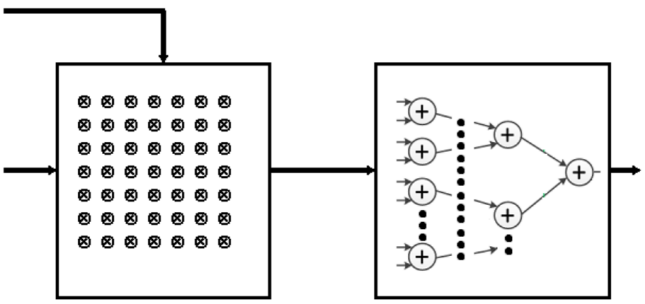
In Table.1 the resource utilization (number of logic elements, embedded multipliers , and registers) is illustrated for REF1, REF2 and the automatic generation system. It is clear that the floating point systems are very complex and consume a large number of multipliers and other resources. Using fixed point reduces the required resources dramatically with a small accuracy losses. For large kernel number of multipliers and adders grow exponentially. The automatic generation system not only helps the designer to analyze the error, but also generate optimized architectures by analyzing the coefficients and generates the Verilog code. For example in table.2 it is clear that that automatic generation systems has reduced the required resources dramatically without any losses in accuracy. The reason is that: the coefficients are integers and they also contain zeros and ones only which can be implemented without multipliers.

TABLE.2

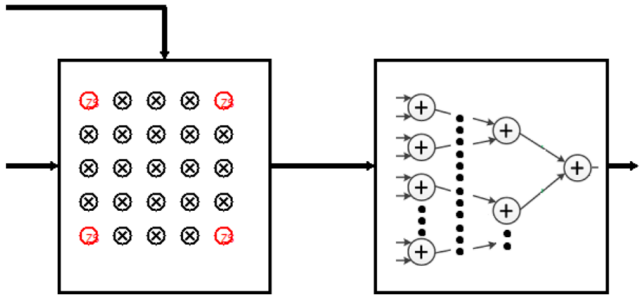
Accuracy and Resources utilization for Sobel Kernel 3x3

Resources	REF1	REF2	Aut_Gen
Logic Elements	3,629	782	235
Registers	3,262	450	25
Embedded Multipliers	63	18	0
Accuracy	100%	100%	100%

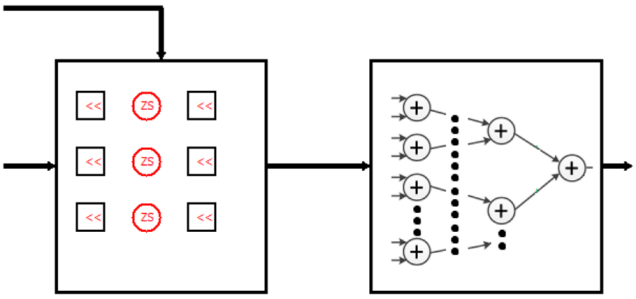
In fig.10 different kernels are generated using the automatic generation system which analyze the coefficients and generates the optimized architecture and Verilog code for these kernels. In fig.10 (b) the analyzer has detected some zeros and hence, discarded the multiplier units in these locations. Also, the adder tree has been automatically modified to reflect the new change. All these analysis and generation are very difficult for manual processing specially for large kernels , and this tool has verified its efficiency , ease of use and fast analysis and architecture generation.



a)



b)



c)

Fig -10: Different Architectures generated using the automatic generation System. a) Gaussian 7x7 b) gaussian 5X5 c) Sobel 3x3

6. CONCLUSIONS

Automatic generation of hardware architectures of convolvers is very useful in designing fast and efficient systems which are difficult to be designed manually. The importance of this automatic generation system increases for large kernel sizes where it is difficult to trace all coefficients and optimize the architecture for these values. The combination of many optimizations techniques in one system achieved high performance. Usually, the researcher focuses on one optimization technique and ignore some important techniques which can enhance the performance. The automatic generation system reduces the designing efforts and time and generates a Verilog code which is suitable for direct realization on FPGA or ASIC.

REFERENCES

[1] Elboher, Elhanan, and Michael Werman. "Efficient and accurate Gaussian image filtering using running sums." 2012 12th International Conference on Intelligent Systems Design and Applications (ISDA). IEEE, 2012.

[2] Froehlich, Saman, Daniel Große, and Rolf Drechsler. "Approximate hardware generation using symbolic computer algebra employing grobner basis." 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2018.

[3] Guo, Kaiyuan, et al. "Angel-eye: A complete design flow for mapping cnn onto embedded fpga." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 37.1 (2017): 35-47

[4] Gwee, Bah-Hwee, et al. "A low-voltage micropower asynchronous multiplier with shift–add multiplication approach." IEEE Transactions on Circuits and Systems I: Regular Papers 56.7 (2008): 1349-1359.

[5] Hamdan, Muhammad K., and Diane T. Rover. "VHDL generator for a high performance convolutional neural network FPGA-based accelerator." 2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig). IEEE, 2017.

[6] Hsiao, Shen-Fu, Jun-Hong Zhang Jian, and Ming-Chih Chen. "Low-cost FIR filter designs based on faithfully rounded truncated multiple constant multiplication/accumulation." IEEE Transactions on Circuits and Systems II: Express Briefs 60.5 (2013): 287-291.

[7] Joginipelly, Arjun Kumar, and Dimitrios Charalampidis. "Efficient separable convolution using field programmable gate arrays." Microprocessors and Microsystems 71 (2019): 102852

[8] Kala, S., et al. "High-performance CNN accelerator on FPGA using unified Winograd-GEMM architecture." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 27.12 (2019): 2816-2828.

[9] Milder, Peter, et al. "Computer generation of hardware for linear digital signal processing transforms." ACM Transactions on Design Automation of Electronic Systems (TODAES) 17.2 (2012): 1-33.

[10] Mukherjee, Debasish, and Susanta Mukhopadhyay. "Fast hardware architecture for fixed-point 2D Gaussian filter." AEU-International Journal of Electronics and Communications 105 (2019): 98-105.

[11] Rao, Lei, Bin Zhang, and Jizhong Zhao. "Hardware implementation of reconfigurable 1D convolution." Journal of Signal Processing Systems 82.1 (2016): 1-16.

[12] Zhang, Hui, Mingxin Xia, and Guangshu Hu. "A multiwindow partial buffering scheme for FPGA-based 2-D convolvers." IEEE Transactions on Circuits and Systems II: Express Briefs 54.2 (2007): 200-204.