# The Efficiency of Convolution on Gemmini Deep Learning Hardware Accelerator

Dennis Agyemanh Nana Gookyi, Michael Wilson, Roger Kwao Ahiadormey, Derek Kwaku Pobi Asiedu, Paul Danquah, Raymond Gyaang

▶ **To cite this version:**

Dennis Agyemanh Nana Gookyi, Michael Wilson, Roger Kwao Ahiadormey, Derek Kwaku Pobi Asiedu, Paul Danquah, et al.. The Efficiency of Convolution on Gemmini Deep Learning Hardware Accelerator. AFRICON 2023: IEEE AFRICON Conference, Sep 2023, Nairobi, Kenya. 10.1109/africon55910.2023.10293709 . hal-04599896

**HAL Id: hal-04599896**

**https://imt-atlantique.hal.science/hal-04599896v1**

Submitted on 4 Jun 2024

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

2023 IEEE AFRICON

# The Efficiency of Convolution on Gemmini Deep Learning Hardware Accelerator

Dennis Agyemanh Nana Gookyi
*Institute for Scientific and Technological Information*
*Council for Scientific and Industrial Research*
Accra, Ghana
dennisgookyi@gmail.com

Michael Wilson
*Institute for Scientific and Technological Information*
*Council for Scientific and Industrial Research*
Accra, Ghana
yboagengwilson@gmail.com

Roger Kwao Ahiadormey
*Institute for Scientific and Technological Information*
*Council for Scientific and Industrial Research*
Accra, Ghana
rogerkwao@ieee.org

Derek Kwaku Pobi Asiedu
*Department of Mathematical and Electrical Engineering*
*IMT-Atlantique*
Brest, France
kwakupobi@gmail.com

Paul Danquah
*Institute for Scientific and Technological Information*
*Council for Scientific and Industrial Research*
Accra, Ghana
pauldanquah@yahoo.com

Raymond Gyaang
*Department of Electrical and Electronic Engineering*
*Bolgatanga Technical University*
Bolgatanga, Ghana
gyaangraymond@bolgatu.edu.gh

***Abstract*—The successful use of deep learning (DL) algorithms in a variety of applications is conceptually based on convolutions. Though convolution is a simple operation, it suffers from severe performance degradation when implemented in software. Recently, with the advancement of CMOS technology, the convolution operation in DL algorithms has been accelerated by being delegated to specialized hardware platforms such as Field Programmable Gate Array (FPGA) devices. On hardware platforms, the convolution operation can be implemented on a synthesizable processor core or custom hardware accelerators based on systolic arrays (SA). Choosing an optimal hardware implementation should not be done analytically but instead employ the use of tools for fast and accurate estimation of metrics such as execution cycles, hardware resource utilization, and power consumption. This work evaluates the efficiency of implementing the convolution operation on various SA dimensions (8 × 8, 16 × 16, and 32 × 32) on the open-source Gemmini DL hardware accelerator with a comparison to the synthesizable RISC-V Rocket processor core. In terms of execution cycles the 8 × 8, 16 × 16, and 32 × 32 Gemmini configurations offer speedups of 323×, 249×, and 204× relative to the Rocket core. This work shows that, unlike the General Matrix to Matrix Multiplication (GEMM), the performance of the convolution operation degrades by an average factor of 2 when the Gemmini SA is doubled. In terms of hardware resource utilization on the Zynq Ultrascale+ ZCU104 FPGA evaluation board, the area and power consumption increased by 3.1× and 2.7× when the Gemmini SA dimension is doubled. Overall, the 8 × 8 Gemmini SA dimension recorded the highest performance-per-area metric making it the most efficient for a popular convolution configuration.**

***Keywords—deep learning, convolution, FPGA, systolic array, hardware accelerator, Gemmini, Rocket core***

## I. INTRODUCTION

Convolutional Neural Networks (CNNs) have exhibited superior accuracy than humans when used in applications such as translation [1], video/image classification [2], and object detection [3]. CNNs are greatly successful because of the use of convolution operations in layers for feature extraction. Though convolution is a simple operation, it is computationally intensive and hence a challenge for real-time and high-performance processing [4]. Convolution can be implemented on platforms such as conventional Central Processing Units (CPUs), Graphic Processing Units (GPUs), or Domain-Specific Architectures (DSAs). Implementation in conventional CPUs is flexible but suffers from a lack of parallel processing and limitation in both power and frequency [5]. The parallel processing issue of the conventional CPU implementation can be solved using GPU which also suffers from low latency and enormous power consumption [6]. The convolution implementation issues in CPU and GPU have led to the design of DSAs (hardware accelerators) on FPGAs to improve energy efficiency and latency.

The two common open-source hardware accelerators for DL algorithms are the Gemmini framework [7] and the Nvidia Deep Learning Accelerator (NVDLA) [8]. Both accelerators are based on the SA [9] architecture which is designed by interconnecting processing units (PEs) to form two-dimensional arrays. Though the NVDLA accelerator is estimated to be 3.77× faster in running ResNet-50 [10] neural network, this work focused on the Gemmini framework because it offers a complete System-on-Chip (SoC) synthesizable hardware architecture for DL exploration. This work, therefore, studies the efficiency of the convolution on various configurations of the Gemmini accelerator compared to the synthesizable RISC-V Rocket processor core. The contributions of this paper include:

- Extracting cycle-accurate performance metrics (execution cycles) from convolution operation on three SA dimensions of the Gemmini accelerator with comparison to implementations on Rocket processor core.
- Synthesizing the hardware modules of the various SA dimensions of the Gemmini accelerator to extract resource utilization metrics such as hardware area, frequency, and power consumption.
- Identifying the most efficient Gemmini configuration using the performance and resource utilization metrics.

This work was carried out with the aid of a grant in the UNESCO-TWAS programme, "Seed Grant for African Principal Investigators" financed by the German Federal Ministry of Education and Research (BMBF) (TWAS-SG-NAPI-4500474961).

2023 IEEE AFRICON | 979-8-3503-3621-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/AFRICON55910.2023.10293709

979-8-3503-3621-4/23/$31.00 ©2023 IEEE

Authorized licensed use limited to: IMT ATLANTIQUE. Downloaded on January 22,2024 at 08:12:29 UTC from IEEE Xplore. Restrictions apply.

The rest of this paper is organized as follows: Section II presents some background and related works, Section III discusses the Gemmini accelerator hardware template, Section IV outlines the methodology used in this research, Section V documents the experimental results of the research, and Section VI concludes the paper.

## II. Background

### A. Convolution on Systolic Arrays

The convolution operation in CNNs is a mathematical operation that serves as the building block for feature extraction involving an input image and kernels [11]. Fig. 1 illustrates a simple 2D direct convolution of input $A$ of $i\_dim_h$ height and $i\_dim_w$ width and kernel $B$ of $k\_dim_h$ height and $k\_dim_w$ width. The output $C$ of $o\_dim_h$ height and $o\_dim_w$ width is computed as a dot-product by sliding the kernel $B$ over submatrices of input $A$ with dimensions of $k\_dim_h \times k\_dim_w$. The sliding is both vertical and horizontal with the sliding parameter of $s_h$ and $s_w$ respectively.

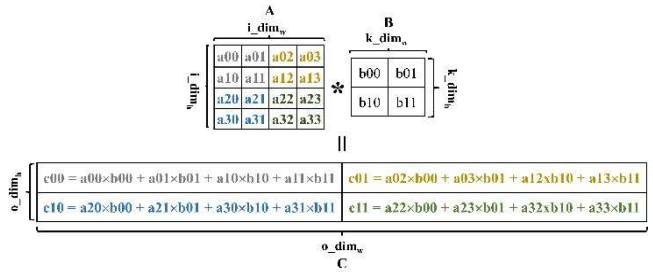


Fig. 1. 4 × 4 by 2 × 2 direct convolution operation.

One method for accelerating convolution is by using SA which was proposed in 1979 [12] but has recently become popular due to the advancement of parallel processing hardware architectures. The SA is made up of an interlocking grid of PEs with each internal architecture made up of a multiply-accumulate (MAC) unit as shown in Fig. 2. The operation of SAs starts by arranging input $A$, kernel $B$, and the results $C$ from Fig. 1 into memories $A$, $B$, and $C$ (also known as the accumulator) respectively. The total memory locations needed for $A$, $B$, and $C$ are computed using Equations (1), (2), and (3) respectively. The $B$ values are preloaded from the top into the SA PEs from cycle $t_1$ to $t_4$. From cycle $t_5$ to $t_8$, the A values are pushed from the left into the PEs for partial multiplication while the results are accumulated in memory C.

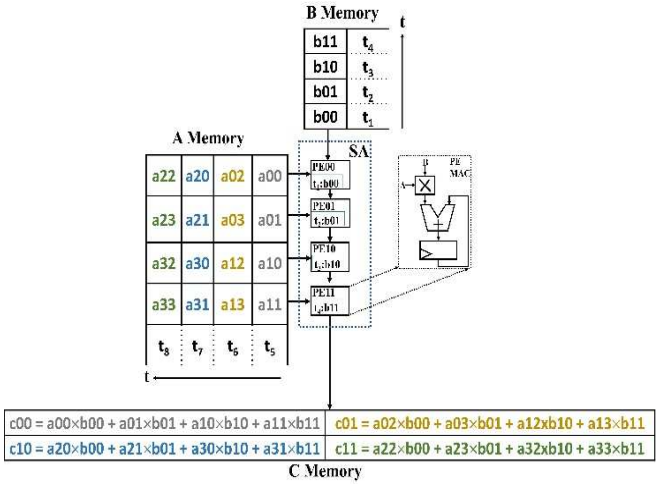


Fig. 2. 4 × 4 by 2 × 2 convolution using systolic arrays.

$$k\_dim_h \times k\_dim_w \tag{1}$$

$$(k\_dim_h \times k\_dim_w) \times [(i\_dim_h - s_h) \times (i\_dim_w - s_w)] \tag{2}$$

$$[(i\_dim_w - k\_dim_w) / s_w] \times [(i\_dim_h - k\_dim_h) / s_h] \tag{3}$$

### B. Related Work

Gemmini is a component of the open-source RISC-V ecosystem [13] developed for DL accelerator exploration by researchers at the University of California, Berkeley. Since its inception in 2019, a number of works [14], [15] focused on improving some individual components of the framework. Few researchers explored the performance of DL building blocks including convolutions and GEMM on the Gemmini framework. The authors of [16] measured the performance of various convolutions on a single SA configuration (16 × 16) of the accelerator. They concluded that convolutions on the Gemmini framework are faster than implementation on a traditional CPU. The authors failed to explore convolution performance on other SA dimensions. They also failed to measure the effects of the generated Gemmini architecture on hardware resource consumption.

This work investigates the efficiency of the convolution operation on different configurations of the Gemmini framework. The performance in terms of cycles of a popular convolution parameter on the various SA dimensions is measured using an open-source cycle-accurate simulator. The hardware resource utilization of the various SA dimensions of the framework is extracted using a commercial FPGA board. The efficiency of the various SA dimensions is measured using the performance and hardware resource utilization results.

## III. The Gemmini Opens-source Template

The Gemmini configurable hardware template for generating DL accelerators is shown in Fig. 3. The accelerator comprises a configurable SA arranged as tiles. The SA is loaded with data from the Scratchpad which is made up of configurable SRAMs for storing computational data. The results of the computations are stored in the Accumulator also made up of SRAMs. The Controller is responsible for scheduling data to and from the SA and also sends/receives data from the DRAM. The accelerator can be configured to use components such as Transposer for transposing the kernel matrix, Mat-Scaler Mult for matrix scaler multiplication, ReLU for rectified linear unit operation as well Pool for the pooling operation in DL algorithms. The framework can be configured to use one or more RISC-V synthesizable CPUs including Berkeley Out-of-Order Machine (BOOM) and Rocket core. The CPUs communicate with the Gemmini accelerator by sending commands through an interface known as the Rocket Custom Coprocessor (RoCC). A number of configurable hierarchical caches including the Level 1 (L1) and Level 2 (L2) can be added.

Authorized licensed use limited to: IMT ATLANTIQUE. Downloaded on January 22,2024 at 08:12:29 UTC from IEEE Xplore. Restrictions apply.

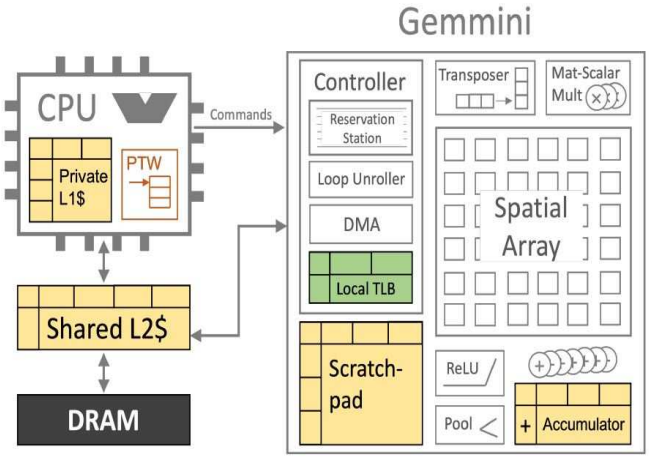


Fig. 3. Gemmini accelerator open-source hardware template [7].

## IV. Methodology

This work employed both commercial and open-source hardware development tools to examine the efficiency of the convolution operation on different SA dimensions of the Gemmini hardware accelerator and the RISC-V Rocket processor core. The workflow is shown in Fig. 3.

The flow starts by customizing and generating the Gemmini SoC Register Transfer Level (RTL) codes by editing Scala configuration files in the Gemmini repository [17]. Some of the customizable components in the SoC include the RISC-V CPU, the Gemmini accelerator, caches, and some peripherals. Fig. 5 illustrates the parameters used for configuring and generating the modules in the Gemmini SoC. The Gemmini accelerator is composed of one tile with three SA dimensions including 8 × 8, 16 × 16, and 32 × 32. The processor configured is the 5-stage in-order RISC-V Rocket core. The Direct Memory Access (DMA) controller has a data width of 128-bit with a total of 16 transfers per burst. The Accumulator has two banks with a capacity of 64 KB, while the Scratchpad has four banks with a capacity of 256 KB.

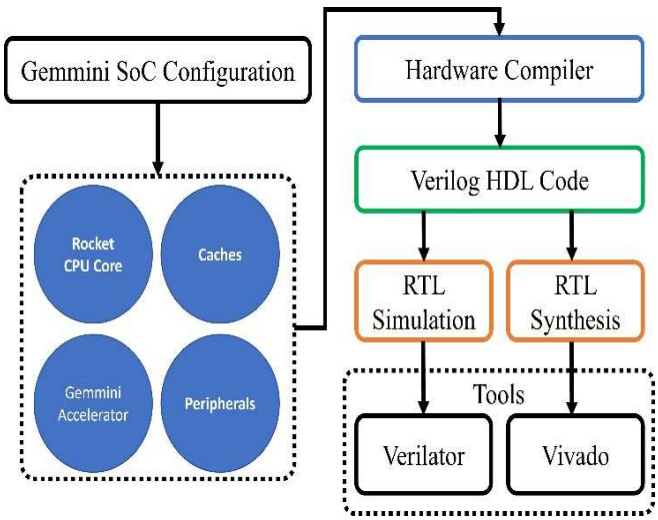


Fig. 4. Gemmini accelerator exploration workflow.

| | | | | |
|---|---|---|---|---|
| Datatype | inputType | Int 8-bit | | |
| | accType | Int 32-bit | | |
| Scratchpad (KB) | sp_capacity | 256 | | |
| | sp_banks | 4 | | |
| Accumulator (KB) | acc_capacity | 64 | | |
| | acc_banks | 2 | | |
| DMA | max_in_flight_req | 16 | | |
| | dma_bus_width | 128 | | |
| Processor | Rocket Core | BigCores | | |
| Spatial Array | tileRows | 1 | | |
| | tileColumns | 1 | | |
| | meshRows | 8 | 16 | 32 |
| | meshColumns | 8 | 16 | 32 |

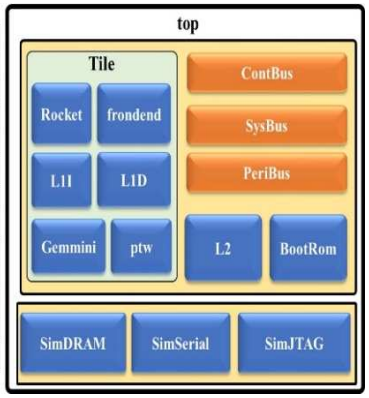


Fig. 5. Gemmini accelerator SoC generation parameters and modules.

After configuring the Gemmini SoC, the configuration commands can be executed using the RISC-V open-source toolchain [18] which includes the Flexible Intermediate Representation for RTL (FIRRTL) compiler to generate the RTL codes in the form of Verilog Hardware Description Language (HDL). The Verilog files are used for both simulation and synthesis to extract performance and hardware resource utilization results respectively.

For simulation, this work compiled bare-metal convolution programs that are executed in the various SA dimension of the Gemmini accelerator as well as the Rocket core. The convolution parameters include batch size (*batch_size*) of 1, input dimension (*in_dim*) of 224, input channel (*in_channel*) of 3, output channel (*out_channel*) of 7, kernel dimension (*kernel_dim*) of 3, *padding* of 1, *stride* of 2, and output dimension (*out_dim*) of 112. These are common convolution parameters for image classification in DL. The RISC-V compiler version riscv64-unknown-elf-gcc is used to compile the convolution programs to generate executable files for simulation. The executable files together with the generated Verilog files are fed into the open-source Verilator tool [18] for simulation. The Verilator tool generates performance results of executing the convolution operation in the Gemmini accelerators as well as the Rocket core.

For hardware synthesis, Vivado version v2020.2 [19] is used to extract resource utilization reports on the Zynq Ultrascale+ ZCU104 FPGA device by feeding the tool with the generated Verilog files. The Vivado tool converts the Verilog codes into hardware gate-level netlist files for analysis.

## V. Experimental Results

This section elaborates on the results obtained by following the methodology discussed in Section IV to measure the performance and resource utilization of the convolution operation on various Gemmini SA dimensions.

### A. Performance

The convolution operation with an input image dimension of 224 × 224 is run on 8 × 8, 16 × 16, and 32 × 32 Gemmini SA dimensions and the Rocket CPU core. The Verilator simulator with an operating frequency of 100 MHz is used to extract the performance results in the form of execution cycles. Fig. 6 illustrates the execution cycles of convolution on the Gemmini configured accelerators and the Rocket core.

Authorized licensed use limited to: IMT ATLANTIQUE. Downloaded on January 22,2024 at 08:12:29 UTC from IEEE Xplore. Restrictions apply.

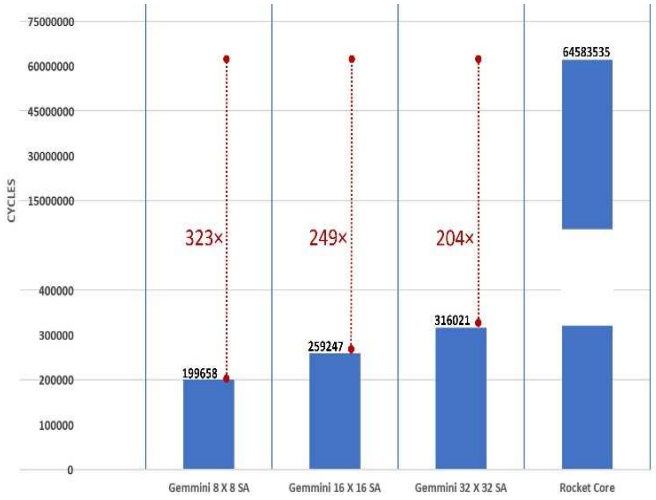


Fig. 6. Convolution performance of Gemmini and Rocket core.

The Gemmini SA of 8 × 8, 16 × 16, and 32 × 32 reported cycles of 199658, 259247, and 316021 respectively while the Rocket core reported cycles of 64583535. Overall, all the Gemmini SA configurations performed better than the Rocket core. The Gemmini SA of 8 × 8, 16 × 16, and 32 × 32 reported speedups of 323×, 249×, and 204× relative to the Rocket core.

Focusing on the Gemmini configurations it is observed that doubling the SA dimension increases the cycles by a factor of 2. This observation is contrary to the operation of GEMM on the Gemmini SA in which the cycles are reduced by a factor of 2 when the SA dimension is doubled [20]. The reason for the increase in the cycles when the SA area is doubled is that given a convolution operation with a kernel size of 3 × 3 limits the number of PEs used for the MAC operation to 3. The partial results have to be pushed down and accumulated in the accumulator as shown in Fig. 2. When the SA is increased while the kernel size is fixed, the partial results must pass through unused PEs before reaching the accumulator and for that matter, the latency of cycles increase.

### *B. Resource Utilization*

The hardware resource utilization of the three Gemmini SA dimensions including 8 × 8, 16 × 16, and 32 × 32 was obtained using Vivado software with Zynq Ultrascale+ ZCU104 FPGA device. The ZCU104 FPGA device is a high-end hardware development board for running machine learning applications. It is equipped with 230400 Configurable Logic Block Look-up Tables (CLB LUTs) with other modules including 312 Block Random Access Memory (BRAM) tiles and 1728 Digital Signal Processing (DSP) blocks. Table I shows the results from synthesizing the three Gemmini SA configurations with the selected FPGA device. The results include metrics such as CLB LUTs, maximum frequency, and power consumption.

The hardware gates of FPGAs are grouped into CLB LUTs which gives them flexibility in terms of programmability. A CLB LUT is made up of a small RAM together with adders, flip-flops, and registers for storing information. From Table I, the 8 × 8, 16 × 16, and 32 × 32 Gemmini SA dimensions used 90371 with a utilization of 39.2%, 245928 with a utilization of 106.7%, and 860052 with a utilization of 373.3% respectively. This shows that only the 8 × 8 SA dimension can fit into the FPGA device with the 16 × 16 and 32 × 32 needing over 6% and 273% extra CLB LUTs respectively. It can also be observed that doubling the SA dimension increase the CLB LUTs by a factor of 3.1.

The maximum frequency metric of a digital system determines the critical path of the circuit which is the longest/slowest path a signal travels from the input to the output. The critical path is important because it restricts the speed at which the circuit operates. The longer the critical path, the lower the frequency. From Table I, the 8 × 8 SA dimension recorded the highest maximum frequency of 58.8 MHz with a critical path delay of 17 ns while the 16 × 16 and 32 × 32 SA both recorded a maximum frequency of 52.6 MHz with a critical path delay of 19 ns.

The Vivado software consists of an in-built tool for analyzing the power consumption of a design. The tool reports both dynamic and static power consumption. The static power is recorded when the design is in a sturdy state. The static power does not change if the parameters are kept constant. The dynamic power on other hand is recorded when the design is operational. This work, therefore, reported the dynamic power of the three Gemmini SA dimensions as shown in Table I. From the table, the 8 × 8, 16 × 16, and 32 × 32 recorded power consumption of 0.960 W, 2.211 W, and 7.387 W respectively. It can be observed that when the Gemmini SA dimension is doubled, the dynamic power consumed is increased by an average factor of 2.7.

TABLE I. GEMMINI HARDWARE RESOURCE UTILIZATION

| Gemmini SA Config | CLB LUTs (% Utilization) | Frequency (MHz) | Power (W) |
|---|---|---|---|
| 8 × 8 | 90371 (39.2%) | 58.8 | 0.960 |
| 16 × 16 | 245928 (106.7%) | 52.6 | 2.211 |
| 32 × 32 | 860052 (373.3%) | 52.6 | 7.387 |

### *C. Efficiency*

The experimental results of this work in Section V produced the performance in terms of execution cycles as well as the hardware gate count in terms of CLB LUTs of three Gemmini SA configuration dimensions which include 8 × 8, 16 × 16, and 32 × 32. This section, therefore, employs the performance and hardware resources utilized by the various Gemmini SA dimension to estimate a figure of merit referred to as the Performance-per-Area (PPA). The PPA metric is a simplified method to measure the efficiency of a digital system. The PPA estimates how the convolution operation makes use of the available hardware resources (CLB LUTs) of the three Gemmini SA configurations. The PPA metric in this work is computed as the ratio of the convolution operation performance (execution cycles) and the resource consumption (CLB LUTs) of the Gemmini SA configuration.

Fig. 7 illustrates the PPA of the three Gemmini SA configurations together with the execution cycles and total CLB LUTs utilized. From the figure, the cycles and the CLB LUTs increase with an increase in the Gemmini SA while the PPA reduces linearly. It must be noted that a high PPA value indicates that the convolution operation made good use of the available hardware resources. The Gemmini 8 × 8 SA configuration recorded the highest PPA of 2.2 while the 16 × 16 and 32 × 32 SA configurations recorded PPAs of 1.05 and 0.37 respectively. This indicates that the Gemmini 8 × 8 SA configuration is the most efficient for convolutions among the three configurations.

Authorized licensed use limited to: IMT ATLANTIQUE. Downloaded on January 22,2024 at 08:12:29 UTC from IEEE Xplore. Restrictions apply.

The authors of [20] mapped GEMM operations to four Gemmini SA configurations including 8 × 8, 16 × 16, 32 × 32, and 64 × 64. They observed that, unlike this work, the cycles are reduced when the Gemmini SA dimension is doubled while the CLB LUTs are doubled as confirmed in this work. They concluded that the Gemmini 16 × 16 SA was the most efficient in running GEMM operations. It is therefore recommended that to get the best performance out of the Gemmini accelerators, the convolution operation should be transformed into a GEMM operation before being mapped to the SA of the Gemmini accelerator.

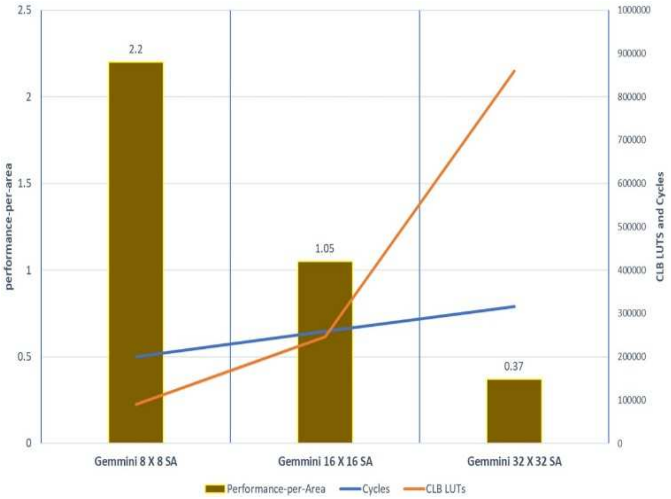


Fig. 7. Efficiency of Gemmini SA configurations.

## VI. Conclusion

This work studied the efficiency of convolution operation on the open-source Gemmini DL hardware accelerator template with three SA dimensions including 8 × 8, 16 × 16, and 32 × 32. For measuring the performance of convolution, the open-source Verilator simulator was used to obtain the execution cycles on the three Gemmini SA configurations as well as the RISC-V synthesizable Rocket processor core. It was observed that the 8 × 8, 16 × 16, and 32 × 32 Gemmini SA configurations offer speedups of 323×, 249×, and 204× relative to the Rocket core. Also, the performance of the convolution operation degrades by an average factor of 2 when the Gemmini SA is doubled. For measuring the hardware resource utilization in terms of CLB LUTs, frequency, and power, the three Gemmini SA configurations were mapped on the Zynq Ultrascale+ ZCU104 FPGA evaluation board. It was observed that area and power consumption increased by 3.1× and 2.7× when the Gemmini SA dimension is doubled. The 16 × 16 and 32 × 32 SA configurations used over 100% of the available CLB LUTs on the FPGA and therefore not recommended for low-cost devices. The performance and resource utilization results were used to estimate the efficiency of the Gemmini SA configurations in the form of PPA. Overall, the 8 × 8 Gemmini SA dimension recorded the highest PPA metric making it the most efficient for a popular convolution configuration.

Future work will deal with efficient methods of transforming convolutions to GEMM operation before being mapped to the Gemmini accelerator to improve performance.

## References

[1] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformer for language understanding,” in Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2019, pp. 4171–4186.

[2] Y. Jeong, S. Son, B. Lee, and S. Lee, “The braking-pressure and driving direction determination system (BDDS) using road roughness and passenger conditions of surrounding vehicles,” Sensors, vol. 22, pp. 1–19, June 2022.

[3] Q. Zhao, T. Sheng, Y. Wang, Z. Tang, Y. Chen, L. Cai, and H. Ling, “M2Det: A single-shot object detector based on multi-level feature pyramid network,” in 33rd AAAI Conference on Artificial Intelligence, 2019, pp. 9259–9266.

[4] S. S. Park and K. S. Chung, “CONNA: Configurable matrix multiplication engine for neural network acceleration,” Electronics, vol. 11, pp. 1–19, July 2022.

[5] H. Esmaeilzadeh, E. Blem, R. Amant, K. Sankaralingam, and D. Burger, “Dark silicon and the end of multicore scaling,” in IEEE Annual International Symposium on Computer Architecture, 2011, pp. 365–376.

[6] S. D. Manasi, S. Banerjee, A. Davare, A. A. Sorokin, S. M. Burns, D. A. Kirkpatrick, and S. S. Sapatnekar, “Reusing GEMM hardware for efficient execution of depthwise separable convolution on ASIC-based DNN accelerators,” in 28th Asia and South Pacific Design Automation Conference, 2023, pp. 475–482.

[7] H. Genc, S. Kim, A. Amid, A. A. Haj, V. Iyer, P. Prakash, J. Zhao, D. Grubb, H. Liew, and H. Mao, “Gemmini: Enabling systematic deep learning architecture evaluation via full-stack integration,” in ACM/IEEE Automation Conference, 2021, pp. 1–6.

[8] NVDLA, “NVDLA open source project,” Available: http://nvdla.org/hw/contents, Accessed on 14 March 2023.

[9] H. T. Kung, “Why systolic architectures,” IEEE Computer, vol. 15, pp. 37–46, June 1982.

[10] A. Gonzalez and C. Hong, “A chipyard comparison of NVDLA and Gemmini,” Available: https://charleshong3.github.io/projects/nvdla_v_gemmini.pdf, Accessed on 14 March 2023.

[11] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, pp. 2278–2324, June 1998.

[12] V. K. P. Kumar and Y. C. Tsai, “Designing linear systolic arrays,” Journal of Parallel and Distributed Computing, vol. 7, pp. 441–463, June 1998.

[13] Chipyard, “Chipyard version 1.8.1,” Available: https://chipyard.readthedocs.io/en/stable/, Accessed on 14 March 2023.

[14] K. Inayat and J. Chung, “Hybrid accumulator factored systolic array for machine learning,” IEEE Transaction on Very Large Scale Integration, vol. 30, pp. 881–892, June 2022.

[15] F. N. Peccia and O. Bringmann, “Integration of a systolic array based hardware accelerator into a DNN operator auto-tuning framework,” arXiv:2212.03034v1, pp. 1–6, December 2022.

[16] C. Viera, A. Lorenzon, L. Schnorr, P. Navaux, and A. C. Beck, “Exploring direct convolution performance on the Gemmini accelerator,” in 21st Brazilian Symposium on High-Performance Computing Systems, 2020, pp. 1–12.

[17] GEMMINI, “Gemmini,” Available: https://github.com/ucb-bar/gemmini, Accessed on 14 March 2023.

[18] VERIPOL, “Verilator,” Available: https://veripool.org/verilator/, Accessed on 14 March 2023.

[19] AMD Xilinx, “Vivado,” Available: https://www.xilinx.com/support/download.html, Accessed on 14 March 2023.

[20] D. A. N. Gookyi, E. Lee, K. Kim, S. J. Jang, and S. S. Lee, “Deep learning accelerators’ configuration space exploration effect on performance and resource utilization: A Gemmini case study,” Sensors, vol. 23, pp. 1–26, February 2023.

Authorized licensed use limited to: IMT ATLANTIQUE. Downloaded on January 22,2024 at 08:12:29 UTC from IEEE Xplore. Restrictions apply.