Enhancing Computation-Efficiency of Deep Neural Network Processing on Edge Devices through Serial/Parallel Systolic Computing

Iraj Moghaddasi and Byeong-Gyu Nam *

Department of Computer Science and Engineering, Chungnam National University, Daejeon 305-764, Republic of Korea; iraj.moghaddas@gmail.com
* Correspondence: bgnam@cnu.ac.kr

Abstract: In recent years, deep neural networks (DNNs) have addressed new applications with intelligent autonomy, often achieving higher accuracy than human experts. This capability comes at the expense of the ever-increasing complexity of emerging DNNs, causing enormous challenges while deploying on resource-limited edge devices. Improving the efficiency of DNN hardware accelerators by compression has been explored previously. Existing state-of-the-art studies applied approximate computing to enhance energy efficiency even at the expense of a little accuracy loss. In contrast, bit-serial processing has been used for improving the computational efficiency of neural processing without accuracy loss, exploiting a simple design, dynamic precision adjustment, and computation pruning. This research presents Serial/Parallel Systolic Array (SPSA) and Octet Serial/Parallel Systolic Array (OSPSA) processing elements for edge DNN acceleration, which exploit bit-serial processing on systolic array architecture for improving computational efficiency. For evaluation, all designs were described at the RTL level and synthesized in 28 nm technology. Post-synthesis cycle-accurate simulations of image classification over DNNs illustrated that, on average, a sample 16 × 16 systolic array indicated remarkable improvements of 17.6% and 50.6% in energy efficiency compared to the baseline, with no loss of accuracy.

Keywords: systolic array; DNN accelerator; serial inference engine; TPU; energy efficiency

1. Introduction

The growing importance of deep learning (DL) lies in solving problems that may be difficult or even impossible for human experts. In this context, deep neural networks (DNNs) have demonstrated excellent accuracy in emerging DL applications [1,2]. To enhance their accuracy further, modern DNNs are becoming increasingly more complicated. For example, some DNNs need more than ten million parameters to perform billions of multiply-accumulate (MAC) operations in the inference phase, requiring significant data movement to support the process [3]. A direct parallel processing approach requires extensive data movement, resulting in numerous problems such as increased energy consumption or “power wall” [4,5]. In this regard, edge computing can improve energy efficiency by enabling processing near the data sources and mitigating data transmission, as it consumes more energy than data processing [6]. However, edge devices require special-purpose accelerators to provide better computation efficiency, even though these resource-limited devices struggle to execute complex DNNs.

DNN accelerators are often constructed from highly parallel processing elements (PEs), arranged in a two-dimensional systolic array (SA) architecture, to address intensive processing demands. In summary, SAs have two key features for efficiently accelerating DNNs, including straight support for matrix/vector multiplication as the main DNN computational block and a simple and efficient control mechanism. The SAs operate in different dataflows, e.g., weight stationary (WS) and output stationary (OS) [7], capable
of executing an enormous number of operations concurrently, reaching tera-operations per second (TOPS). For instance, the Google Tensor Processing Unit (TPU) presents an SA with WS dataflow and a raw throughput of 92 TOPS [8]. Similarly, Samsung incorporates a Neural Processing Unit (NPU) for edge devices, featuring the 6 K MAC configuration, capable of delivering up to 14.7 TOPS [9]. This enormous performance of TOPS is achieved at the cost of higher power density, thermal effects, and reliability challenges [10,11], which can restrict them from being used in critical applications, e.g., healthcare. Therefore, there is an urgent need to improve the computational efficiency of SA-based DNN accelerators.

Meanwhile, bit-serial processing has been crucial for the efficient deployment of DNNs on resource-limited edge devices [12–15]. Briefly, there are three factors with serial PEs that help to improve performance and energy efficiency: (1) Dynamic precision adjustment, which could be excellent for heterogeneous precision across DNN layers, or dynamic adaptation according to the requested accuracy for the running workload. For DNN inferences without losing the accuracy, layer-wise precision adjustment can achieve a $2.29 \times$ speedup or energy reduction, on average, in 16-bit precision implementation (or equivalent $1.145 \times$ speedup in 8-bit) [13]. (2) Maximizing the operating frequency due to a simpler design over complex bit-parallel processing units. (3) Increasing throughput via concurrent input bit-group processing [16,17]. The key idea is to process a bit column of concurrent data items in each cycle, which can be useful when applying the same calculation to all data bits.

This research introduces Serial/Parallel SA (SPSA) as a novel approach for SA-based DNN processing. Following this, Octet Serial/Parallel SA (OSPSA) is proposed to improve SPSA energy efficiency even more. Both architectures exploit the benefits of improving efficiency through activation serial processing. Instead of design-time fixed precision assignment, they allow runtime and layer-wise precision adjustment according to the DNN model, requested accuracy, and operating conditions, e.g., performance, energy, temperature, and even reliability. The major contributions of this study are as follows:

1. We propose a serial/parallel systolic array architecture and data flow;
2. We introduce a bit-serial processing of activation with zero skipping capability;
3. Our design exploits activation precision adjustment in a systolic array accelerator;
4. We improve energy efficiency by replacing complicated multipliers with simpler and low-cost serial circuits.

Overall, the focus of this work is to combine the concepts of (a) serial processing, (b) systolic computing, and (c) heterogeneous bit-precision accelerator at the same time and to evaluate their collaborative impact on improving computational efficiency. The rest of this paper is organized as follows. Section 2 presents the baseline design and limitations. Section 3 provides a brief survey of related works. The proposed serial/parallel processing approach is clarified in Section 4. Section 5 overviews the proposed architecture and processing elements. Section 6 presents the experimental results and comparison over the baseline. Finally, Section 7 concludes the paper.

2. Baseline System

The proposed architectures extend TPU as the conventional SA-based DNN accelerator, replacing bit-parallel processing units with serial/parallel elements. The TPU targets the inference task, boasting 64 K (TPU1) and 4 K (TPU3) systolic MAC arrays for cloud and edge, respectively [8]. The TPU can be regarded as a hardware accelerator with spatial architecture and WS dataflow, which proposes a raw throughput of 92 TOPS. The TPU leverages the significant reduction in energy and area by 8-bit integer SA multipliers over the 32-bit floating-point GPU data path. It compresses 25 times MACs and 3.5 times on-chip memory while using less than half the power of GPU in a relatively small die. Meanwhile, the bit-parallel designs seem to have limitations:

- Specifying fixed precision at design-time;
- Static trade-offs among operational factors like accuracy;
- High latency (cycle-time) due to complicated operations.
In this study, the proposed architectures improve the computational efficiency of the TPU-like bit-parallel design by serial/parallel processing, but the throughput is reduced due to performing operations sequentially. The OSPSA processes multiple bit-instances of concurrent activations simultaneously to compensate for the throughput reduction, exploiting the inherited parallelism in the DNNs. Instead of design-time fixed precision assignment for all layers, the proposed designs allow runtime layer-wise precision adjustment according to the DNN model, required accuracy, and operational conditions. In other words, they enable runtime trade-offs among different parameters, e.g., accuracy, energy, performance, temperature, and lifetime [11].

3. Related Work

In recent years, numerous research works have been presented toward efficient deployment of DNNs on edge devices. This section concisely overviews previous works on developing systolic array-based and bit-serial DNN accelerators. In 1978, the systolic array concept was formally introduced [18] as a highly parallel and pipelined array of simple PEs that operate in a lockstep, wave-like fashion. Thanks to spatial architecture and simple data flow, systolic arrays were adapted in parallel computing, finding successful applications, e.g., signal processing and image processing. Despite the initial excitement, the trend in computer architecture has shifted toward more adaptable temporal processing architectures, e.g., CPUs and GPUs [19]. However, with the rise of DNNs, there has been a renewed interest in systolic arrays due to three primary advantages [8,20]: (1) extensive throughput and parallel processing, (2) straightforward structure and data flow that improve energy efficiency by eliminating complex control mechanisms [20], and (3) data caching and reuse within the array for substantial energy consumption reduction by minimizing memory accesses. These attributes have thrust systolic arrays into the forefront of research on DNN accelerator architectures [21]. Regarding this, Google released a systolic array-based TPU based on constant precision bit-parallel PEs [8]. TPU is a programmable and reconfigurable hardware platform to compute neural networks faster and more energy efficiently over conventional CPUs and GPUs [19,20].

With the increasing complexity of modern DNNs, the conventional systolic arrays may not efficiently cover new features and requirements due to exceeding the energy budgets of running embedded edge devices [22]. Meanwhile, designers must consider numerous factors in the systolic array design, including data flow, array size, and bandwidth. These factors collectively influence the final accelerator’s performance, computation efficiency, and memory footprint [23,24]. Based on this consideration, two common solutions have been adopted for speeding up and improving the efficiency of inference engines, including compression and pruning [25–27]. Compression reduces processing and memory complexity by cutting down the bit-precision, including quantization and approximation. The state-of-the-art research [10] applied approximate multipliers to lower power density and temperature at the expense of nearly 1% accuracy loss. Although some applications can tolerate the loss of accuracy, it is not acceptable for critical applications such as healthcare and automotive. Conversely, pruning decreases ineffectual computation amounts such as zero skipping [28,29] or early computation termination [12,25]. Overall, compression can improve efficiency at the price of potential output accuracy loss, while pruning relies on detecting and pruning ineffectual computation, incurring area and power overheads. In this context, multi-precision computing is a major approach for conserving accuracy at the expense of the area overhead of including processing elements with different precisions at the same time. Accordingly, designers tried to increase the efficiency via heterogeneous DNN accelerators (HDAs) [30,31]. Heterogeneity refers to utilizing multiple NPUs with different precision. HDA balances the latency/power trade-off, providing a restricted scaling and imposing area overhead. The core idea is integrating multiple NPUs with different precisions and arranging them in various topologies. This better fits the specific requirements of various computational blocks intra and inter-DNNs.
On the other hand, numerous efforts have focused on enhancing computation efficiency through serial processing, leveraging circuit simplification and dynamic precision adjustment. Generally, serial designs encompass bit-level and bit-serial architectures. In bit-level designs, processing elements (PEs) are constructed from lower-precision components and feature spatiotemporal processing [32,33]. In contrast, bit-serial PEs handle input data bits sequentially and are categorized into serial and serial/parallel types. Serial engines [34,35] use bit-serial approaches for both weights and activations. Despite their low energy consumption, which suits sparsely active systems with low data rates, serial engines suffer from degraded performance and increased response times. Serial/parallel designs [13,27] employ PEs with combined serial and parallel inputs. Regarding this, Ref. [14] reduces latency and energy consumption by nearly 50% on average via dynamic precision adjustment and storing reusable partial products in LUTs. However, this approach incurs memory and computation overheads of refreshing LUT contents when fetching new data. It also relies on tightly coupled data memory (TCDM), significantly reducing data movement by close coupling to the PEs and avoiding deep memory hierarchies. Although dynamic precision adjustment significantly accelerates processing, these designs still face high latency and response times, requiring more optimizations, e.g., exploiting available sparsity. For example, Ref. [27] skips ineffectual bits of inputs, providing an accelerator for CNNs that processes only the essential activation bits, thus excluding all zero bits.

Overall, our work differentiates from the existing state-of-the-art as it combines the concepts of (a) systolic computing, (b) multi-precision architecture, and (c) bit-serial processing. Distinguishing from the forefront studies, we are the first to design a serial and variable bit-precision systolic MAC array for DNN inference in which we reduce resource usage while conserving accuracy. Hence, we satisfy the tight accuracy, area, and latency constraints, while delivering better computation efficiency.

4. Octet Serial Processing Approach

This study proposes two serial/parallel SA (SPSA and OSPSA) designs for DNN accelerators with higher capabilities than conventional bit-parallel SA (PSA), previously used in famous accelerators, e.g., TPU. They mainly improve efficiency by reducing resource consumption through spatiotemporal serial processing, which makes them suitable for DL applications on edge devices. Overall, SPSA and OSPSA (1) reduce energy consumption via the simpler architecture of serial/parallel PEs, (2) enable ineffectual computation pruning in two ways: (a) layer-wise precision adjustment and (b) bit-column-wise zero skipping, and (3) allow higher throughput without trading accuracy through latency reduction and bit-column-wise computing. Although bit-parallel multiplication is useful for high-performance DNN inference, it seems less appropriate for resource-constrained edge devices. Figure 1a shows the computing model of PSA. Herein, activations and weights are fetched in bit-parallel and are multiplied to produce the partial product. Then, the partial product will be added to the input partial sum to produce output for the consequent layer. PEs are in an SA architecture with a target operation of matrix multiplication $O = S \times N$, where $S$ and $N$ represent weights and activations matrices, respectively.

![Figure 1. Processing models: (a) bit-parallel, (b) serial/parallel, and (c) octet serial/parallel.](image-url)
Figure 1b,c illustrate the proposed computing model of PEs with serial/parallel and octet serial/parallel architectures. In this method, weights are in bit-parallel, and activations arrive in bit-serial. Herein, serial activation bits are ANDed by all corresponding weight bits to produce partial products. To boost the throughput to the level of a bit-parallel design, octet serial/parallel PE processes a column of eight concurrent bits of separate activations in the same bit-position. In OSPSA, eight partial products feed to the adder tree (compressor) to generate a partial sum for the current bit-position. Both PEs are working in an 8-cycle loop, starting from MSB and processing one bit of activations per cycle. Figure 2 illustrates the processing approach of the OSPSA for \(2 \times 2\) SA and 2-bit inputs, for simplicity. As shown, four concurrent activations per cycle are fed to the OSPSA array, which iterates in a 2-cycle loop. Herein, the square activations matrices have been shifted into something more akin to a parallelogram so that each input activation reaches the right PE at the right cycle.

![Octet serial/parallel systolic dataflow](image)

**Figure 2.** Octet serial/parallel systolic dataflow.

5. SPSA Accelerator Architecture

5.1. Overall Accelerator Architecture

The accelerator architecture is introduced by borrowing the general structure from TPU with a WS dataflow. As shown in Figure 3, the PEs are replaced by components with serial/parallel design. Like TPU, the Matrix Multiply Unit (MMU) is the heart of the accelerator including \(16 \times 16\) serial/parallel MAC elements in this sample design. This design can be extended for bigger array sizes, like \(32 \times 32\) or \(64 \times 64\) like TPU, only by increasing the bit lines of partial sums. However, the number of rows is preferred to be low to increase the PE utilization percentage, considering the limited filter size of the target DNNs. In the case of bigger networks, like Transformers, the row count can be increased even more. In this design, PEs perform signed operations over separate activations bits and 8-bit parallel weights. The partial products are collected by the Shifter and Accumulator unit below the MMU. The MMU computes at half-speed in case of 8-bit activations and 16-bit parallel weights. The partial products are collected by the Shifter and Accumulator unit below the MMU. The MMU computes at half-speed in case of 8-bit activations and 16-bit parallel weights. The partial products are collected by the Shifter and Accumulator unit below the MMU. The MMU computes at half-speed in case of 8-bit activations and 16-bit parallel weights. The partial products are collected by the Shifter and Accumulator unit below the MMU. The MMU computes at half-speed in case of 8-bit activations and 16-bit parallel weights. The partial products are collected by the Shifter and Accumulator unit below the MMU.
a quarter-speed. In this design, a Transposer element is added to convert the bit-parallel activations and read from memory to serial bit stream for processing by PEs.

![Overall SPSA-OSPSA architecture](image-url)

**Figure 3.** Overall SPSA-OSPSA architecture.

### 5.2. SPSA-MAC Processing Elements

SPSA and OSPSA replace power-hungry bit-parallel multipliers with simple serial circuits and reduce the adder size and output p-sum bit-lines. In the WS dataflow, 8-bit weights are pre-loaded before processing starts. Then, activations arrive bit-group-wise sequentially to exploit the benefits of temporal serial processing and spatial parallel processing. To consider signs, all weights activations, outputs, and intermediate data are in two’s complement. The MSB input indicates the arrival of the most significant sign bit in PEs. Figure 4a,b demonstrate the detailed designs of SP-MAC and OSP-MAC. The number inside the brackets indicates the bit position is processed in each cycle. In SP-MAC, serial activations bits are ANDed with preloaded 8-bit weights per cycle, which is extended to eight concurrent serial activations in OSP-MAC. In OSP-MAC, an adder tree compresses $8 \times 8$-bit partial products to produce an 11-bit partial sum. The partial sum is summed with the input partial sum to generate the output partial sum. In this sample design with a $16 \times 16$ MAC array, the partial sum bit-width is set to 11 and 15 for SPSA and OSPSA, respectively, to accumulate generated partial sums in one column of SA.

![PEs detailed design: (a) SP-MAC and (b) OSP-MAC](image-url)

**Figure 4.** PEs detailed design: (a) SP-MAC and (b) OSP-MAC.
6. Evaluation and Comparison

In this study, a cross-layer framework is deployed to evaluate the computational efficiency of the proposed designs. Initially, all designs, including SPSA, OSPSA, and PSA, are described in RTL (Verilog) and synthesized using a 28 nm cell library to produce the netlist and standard delay format (SDF) files. In parallel, several DNN models on different datasets are deployed in Python (3.12.0) and Matlab (2022) to provide evaluation benchmarks of activation and weight values. Here, the VGG16 on the MNIST dataset and the AlexNet and ReNet18 on the ImageNet dataset are profiled in 8-bit and 16-bit two’s complement format. Next, cycle-accurate simulations are conducted on generated netlists and benchmarks to produce the output data, timing reports, and value change dump (VCD) profiles. After that, the VCD files are used to generate power reports using a power analysis tool. Finally, energy efficiency factors are reported. A summary of the reported primary features by the synthesis tool is shown in Table 1. Herein, the array sizes for PSA and SPSA are assumed to be 16 × 16. However, the equivalent dimension of OSPSA with the same functionality is 2 × 16. Demonstrably, all design factors are improved through serial/parallel SA processing, including area, power efficiency, latency, frequency, performance/area, and performance/watt.

<table>
<thead>
<tr>
<th>Feature</th>
<th>PSA (16 × 16)</th>
<th>SPSA (16 × 16)</th>
<th>Improvement</th>
<th>OSPSA (2 × 16)</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area</td>
<td>243,624</td>
<td>63,475</td>
<td>+74%</td>
<td>37,401</td>
<td>+84%</td>
</tr>
<tr>
<td>Leakage Power</td>
<td>1.456 mW</td>
<td>0.504 mW</td>
<td>+65%</td>
<td>0.299 mW</td>
<td>+79%</td>
</tr>
<tr>
<td>Dynamic Power</td>
<td>17.88 mW</td>
<td>4.42 mW</td>
<td>+72%</td>
<td>2.44 mW</td>
<td>+88%</td>
</tr>
<tr>
<td>Latency (Cycle Time)</td>
<td>2.30 ns</td>
<td>1.00 ns</td>
<td>+56%</td>
<td>1.38 ns</td>
<td>+40%</td>
</tr>
<tr>
<td>Max Frequency</td>
<td>434 MHz</td>
<td>1000 MHz</td>
<td>+130%</td>
<td>724 MHz</td>
<td>+67%</td>
</tr>
<tr>
<td>Max Performance</td>
<td>111 GMac/s</td>
<td>32.0 GMac/s</td>
<td>−71%</td>
<td>23.2 GMac/s</td>
<td>−0.79%</td>
</tr>
<tr>
<td>Max Perf./Area (PPA)</td>
<td>456 × 10^3</td>
<td>504 × 10^3</td>
<td>+10%</td>
<td>619 × 10^3</td>
<td>+35%</td>
</tr>
<tr>
<td>Max Perf./Watt (PPW)</td>
<td>5.74 × 10^12</td>
<td>6.50 × 10^12</td>
<td>+13%</td>
<td>8.47 × 10^12</td>
<td>+47%</td>
</tr>
</tbody>
</table>

6.1. Computation Pruning by Zero Skipping

Compared with bit-parallel processing, there is a higher probability of zero observation and skipping in sequential bit-column-wise computing of activations. This differs from zero skipping in the bit-parallel processing of sparse matrixes and is even applicable to non-sparse matrixes. This is because of the separate processing of bit groups with different positions. For example, if we consider an input image file, there is a higher probability of a zero-bit observation in the most significant bits of neighbor pixels. Figure 5 shows the average potential of zero skipping by SPSA/OSPSA (8.74%) for 8-bit precision compared to PSA in non-sparse input activations.

![Figure 5. Zero skipping in PSA and SPSA-OSPSA.](image)

6.2. Energy Efficiency Improvement

The power-delay product (PDP) is a measure of merit correlated with the energy efficiency of a circuit. The PDP is the product of average power consumption and the input-
output delay, or the duration of a workload run. It has an energy dimension and measures the energy consumption per workload execution. Here, simulations are performed for VGG16, AlexNet, and ResNet18 benchmarks. Figure 6 demonstrates an average of 17.6% and 50.6% reduction in PDP for SPSA and OSPSA compared to the conventional bit-parallel baseline. This is achieved for 8-bit activations precision, which could be increased for higher activations bit-precisions, e.g., 16-bit due to dynamic precision adjustment.

![Figure 6. Improving energy efficiency by serial processing.](image)

7. Conclusions and Future Works

DNNs have achieved amazing accuracy with increasing complexity, making it difficult for them to be deployed on edge devices. So, there is an urgent need for accelerators with higher computation efficiency on edge devices. In this study, new serial/parallel architectures, namely SPSA and OSPSA, have been introduced based on the conventional bit-parallel systolic accelerators to increase the computation efficiency of DNN execution on edge devices. The proposed design exploited serial processing to significantly improve computational efficiency. As far as we know, this is the first DNN inference engine designed with a serial systolic array architecture. The functionality and efficacy of the SPSA were evaluated based on different DNN models and datasets. The experimental results proved that the proposed designs significantly improve energy efficiency without trading accuracy. Furthermore, the proposed architectures demonstrate a higher probability of zero activation skipping by utilizing bit-serial processing.

For future work, we aim to improve the SPSA and OSPSA designs to support a fully serial systolic array with adjustable bit-precision of the activations and weights. Additionally, we will explore other methods to exploit zero skipping more in the weights and activations, leveraging the available bit-sparsity to enhance the energy efficiency even further. Ultimately, we intend to expand the SPSA and OSPSA capabilities to support the training and inference phases.

Author Contributions: Conceptualization, I.M. and B.-G.N.; methodology, I.M.; software, I.M.; validation, I.M. and B.-G.N.; formal analysis, I.M.; investigation, I.M.; resources, I.M.; data curation, I.M.; writing—original draft preparation, I.M.; writing—review and editing, B.-G.N.; visualization, I.M.; supervision, B.-G.N.; project administration, B.-G.N.; funding acquisition, B.-G.N. All authors have read and agreed to the published version of the manuscript.

Funding: The research fund of Chungnam National University supported this work.

Data Availability Statement: The dataset is available from the corresponding author and can be provided on reasonable request.

Acknowledgments: The authors would like to thank IDEC for CAD support.

Conflicts of Interest: The authors declare no conflicts of interest.
References


Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.