Automated Systolic Array Architecture Synthesis for High Throughput CNN Inference on AWS F1 FPGA

Xuechao Wei1,3, Peng Zhang3, Cody Hao Yu2,3, and Jim Wu3

1Center for Energy-efficient Computing and Applications, School of EECS, Peking University, China
2Computer Science Department, University of California, Los Angeles, CA, USA
3Falcon Computing Solutions, Inc, Los Angeles, CA, USA

ABSTRACT
FPGAs are more and more deployed in datacenters as reconfigurable hardware accelerators for applications leveraging deep neural networks (DNNs). Meanwhile, in-datacenter FPGAs are prone to adopt the Stacked Silicon Interconnect (SSI) technology to integrate multiple dies together in order to incorporate more resources. However, previous designs are not scalable to efficiently utilize resources on these large FPGAs to further improve performance due to timing violation caused by crossing-die critical paths. In this work, we propose a Stacked Systolic Array (SSA) architecture by placing multiple systolic arrays on different dies and chaining them by non-critical paths. And we further improve the DSP utilization of SSA by individual designs for different layers of DNN models. We also present an automation flow to do design space exploration and accelerator generation. Experiment results show that our accelerator achieves up to 85% performance improvement than previous designs.

1. INTRODUCTION
DNNs are compute-intensive learning models with growing applicability in a wide range of domains. FPGA is a compelling computing resource in datacenters as accelerator architecture. However, enabling efficient FPGA-based acceleration of DNNs in datacenters is not straightforward for the following reasons:
• Developing high performance DNN architecture is challenging.
• Crossing SLRs on SSI devices may result in frequency degradation.
• Irregularity in DNN models may cause dynamic underutilization of FPGA resources.

In this paper we propose an DNN accelerator architecture for large FPGAs to solve these design challenges.
• A low latency DNN accelerator design based on stacked systolic arrays achieving 2 Tops for AWS F1 FPGA
• A resource partitioning algorithm between systolic arrays and FPGA dies achieving more than 90% resource utilization and 240MHz clock frequency.
• An end-to-end automation flow from high-level C code to FPGA-acceleration of DNNs in datacenters

2. ARCHITECTURE DESIGN FOR DATA-CENTERS

2.1 DNN Accelerator Designs
CNN models, such as AlexNet, VGG, and ResNet [1], consist of several to hundreds of cascaded layers. The basic computations in each layer are common—such as convolutional, fully connected, pooling, sigmoid and ReLU. Among these computation blocks, convolutional layers contribute over 90% of the computational complexity.

The state-of-the-art FPGAs in datacenters such as AWS EC2 F1 FPGA utilize Stacked Silicon Interconnect (SSI) [2] technology that packages thousands DSP blocks on different dies on the same FPGA. Traditional designs are able to achieve the massive parallelization with full pipelining in the architecture design. But the implementation of the design may have difficulty in meeting the timing for a high clock frequency due to SLR crossing, high fan-out, long wires, and large multiplexes issues [3].

2.2 Stacked Systolic Array Architecture

We present a novel stacked systolic array architecture for CNN on large FPGAs in Figure 1. As shown in this figure, each systolic array is designed to process one or a group of convolutional layers, and constrained within a single die. Systolic arrays are connected by stream buffers that are used to store intermediate input features. We can see that only interconnection between dies is the FIFO used to transmit input features from last layer’s output. The single systolic array architecture is shown in Figure 2. The PE array part is used to compute convolution operations. The other parts include ReLU, normalization, pooling and zero padding. This 2-D topology of systolic array matches the 2-D structure in the FPGA layout, and crossing-die paths are FIFOs that are not timing critical. Hence SSA can achieve timing constraints easily because of low routing complexity.

![Figure 1: Stacked systolic array architecture](image)

Although the SSA architecture is able to significantly benefit designs on large FPGAs in datacenters, mapping a CNN model with multiple layers onto a SSA architecture can be challenging in the following areas:
• Determine the single systolic array structure
• Determine the number of systolic arrays
• Stream buffer management
3. ANALYTICAL MODELS AND DESIGN SPACE EXPLORATION

3.1 Analytical Models
The design challenges of CNN on FPGA need to be considered in a unified way with high-level modeling. In this section, we formulate the overall optimization problem as maximizing system throughput under the single die and total resource constraints. The model is constructed in the following steps:

- **Architecture Abstraction**
- **Die Assignment**
- **Resource Utilization Modeling**
- **Performance Modeling**

3.2 Design Space Exploration
Under the performance and resource modeling, our design space exploration identifies a valid design option with the highest throughput. We develop a two-phase process in Figure 2 which first filters the design space into a small set of candidates using the proposed analytical model with a given clock frequency, and then goes through the hardware generation flow for the selected designs to obtain the one that has the best on-board performance.

Figure 2: Two-Phase Design Space Exploration

4. IMPLEMENTATION
We implement a push-button design flow framework to generate an executable system on AWS EC2 F1 FPGAs from a userwritten intuitive CNN program in Figure 3. A user only needs to specify the CNN model configuration using Caffe prototxt, as shown in the left side of Figure 4. Our automation flow shown in the right side of Figure 3 first analyzes the user program using Merlin compiler infrastructure to obtain necessary information such as iteration domains and data access patterns. Subsequently, we perform design space exploration to identify multiple valid design options with the highest estimated throughput. The design options are parameterized to instantiate systolic array implementation (kernel), as well as the C/C++ software program (host).

5. EXPERIMENT
We first compare the performance results of the generated SSA architecture with baseline implementations, as shown in Figure 5. We use two kinds of baseline manifestations, whose results are known as "Baseline" and "SSA-Oracle" in Figure 5. And the results of SSA architecture is "SSA". "Baseline" is the original implementation in [3]. And "SSA-Oracle" is SSA result using the same frequency as "Baseline". We can see that from resource utilization improvement, SSA has 30% performance improvement. With die assignment, SSA could obtain 20% performance improvement.

We also compare our results with state-of-the-art designs. The comparison results are shown in Table 1. We could see that our design has better throughput and performance density results than previous designs.

6. CONCLUSIONS
Large FPGAs manufactured by multiple dies are recently deployed in datacenters as reconfigurable hardware accelerators for applications leveraging deep neural networks (DNNs). But previous designs are not scalable to efficiently utilize resources on these large FPGAs to further improve performance due to timing violation caused by crossing-die critical paths. In this work, we propose
Table 1: Comparison to State-of-the-art Implementations

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Frequency (MHz)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>150</td>
<td>150</td>
<td>303</td>
<td>370</td>
<td>385</td>
</tr>
<tr>
<td></td>
<td>150</td>
<td>150</td>
<td>150</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td><strong>Precision</strong></td>
<td>fixed</td>
<td>fixed</td>
<td>float</td>
<td>float</td>
<td>fixed</td>
</tr>
<tr>
<td></td>
<td>16 bit</td>
<td>8-16 bit</td>
<td>16 bit</td>
<td>32 bit</td>
<td>16 bit</td>
</tr>
<tr>
<td></td>
<td>N/A</td>
<td>N/A</td>
<td>42K (25%)</td>
<td>42K (25%)</td>
<td>8-16 bit</td>
</tr>
<tr>
<td><strong>Logic Utilization</strong></td>
<td>300K (81%)</td>
<td>161K (58%)</td>
<td>246K (58%)</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>DSP Utilization</strong></td>
<td>2833 (78%)</td>
<td>1518 (100%)</td>
<td>1476 (97%)</td>
<td>1320 (87%)</td>
<td>2756 (91%)</td>
</tr>
<tr>
<td><strong>BRAM Utilization</strong></td>
<td>1248 (42%)</td>
<td>1900 (70%)</td>
<td>2487 (92%)</td>
<td>1250 (46%)</td>
<td>1450 (54%)</td>
</tr>
<tr>
<td><strong>Throughput (Gops)</strong></td>
<td>354</td>
<td>645.25</td>
<td>2487</td>
<td>1450</td>
<td>2260.9</td>
</tr>
<tr>
<td><strong>Performance Density (ops/DSPslice/cycle)</strong></td>
<td>0.83</td>
<td>1.42</td>
<td>3.09</td>
<td>1.77</td>
<td>1.84</td>
</tr>
</tbody>
</table>

a Stacked Systolic Array (SSA) architecture by placing multiple systolic arrays on different dies and chaining them by non-critical paths. And we further improve the DSP utilization of SSA by individual designs for different layers of DNN models. We also present an automation flow to do design space exploration and accelerator generation. Experiment results show that our accelerator achieves up to 9X performance improvement than previous designs.

7. REFERENCES


