Algorithm-Architecture Co-Exploration of Systolic Arrays Using High-Level Synthesis

Chu-Chun Yang*†, Gwo Giun (Chris) Lee*††, Tsung-Ying Tsai*, Jie-Ren Zheng*, Yue-Cong Kuo*,
Wei-Chieh Lee*, Ryan Karthik Pary**
* National Cheng Kung University, Taiwan
E-mail: †n26124905@gs.ncku.edu.tw, ††clee@ncku.edu.tw
** Nanyang Technological University, Singapore

***Abstract*—This paper introduces an Architecture Compiler technique which jointly explores the algorithm and architecture co-design. This methodology facilitates a systematic mechanism by which design spaces are crossed or traversed from algorithmic functionality to potentially synthesizable microarchitecture designs having optimized placement and routing with low power, starting from high level synthesis. This will be demonstrated for parallel architecture designs for AI accelerators of general matrix multiplication (GEMM) with high demands in Generative AI on EDGE devices. In this paper, we explored the systolic array dataflow which includes weight, input, and output stationarity for parallel computing in CNN convolution. Different data granularities based on kernel sizes resulting in potentially different architectures will be explored at the high level. This algorithm/architecture co-design methodology will be exploring the potential design space solutions for all LeNet5, AlexNet, VGG16, and VGG19 CNN most suitable for the pre-defined specification quickly.**

I. Introduction

Artificial intelligence (AI) such as Deep learning (DL) [1] and Large Language Models (LLM) [2] has seen rapid growth in recent years, driving advancements across various fields such as visual recognition, natural language processing (NLP), and autonomous systems. Many of these applications require real- time processing to enhance user experience and operational efficiency. Convolutional neural networks (CNNs) are widely used for feature extraction and image classification, with dedicated hardware accelerators playing a crucial role in improving performance.

State-of-the-art System-on-Chip (SoC) designs for deploying complex AI algorithms on edge devices demand vertical integration, which in addition to software/hardware co-design, requires joint algorithm/architecture co-design (AAC). That is, starting from the algorithm, architecture including physical design aspects such as automatic place and route (APR) and power constraints are required to be considered early in the design process. This is particularly crucial to assure that anticipated predefined specification could be designed into the SoC, where balancing power, performance, and area (PPA) requires a holistic, cross-level co-design approach that jointly optimizes both algorithm and architecture [3]. AAC provides information on both the software and hardware requirements of SoC platforms during the early design phase, beginning from algorithms at the top level, which mitigates the potential pitfalls of system integration failure.

This paper explores the AAC based top-down design methodology using, as an example, the systolic array for parallel computing in CNNs. Due to its high parallelism, the systolic array is a prominent technique for computational efficiency enhancement [4] and hence used to showcase the joint exploration of algorithm and architecture design spaces in both general matrix multiplication (GEMM) and direct convolution modes. We model four dataflows: output stationary (OS), weight stationary (WS), input stationary (IS) in GEMM mode [5], and a reconfigurable weight stationary approach with weight decomposition (WS2) for direct convolution [6]. To support GEMM-based modeling, we analyze CNN convolutions and develop analytical models capturing key complexity metrics to evaluate trade-offs. Our methodology enables efficient design space exploration (DSE) for CNNs such as LeNet-5, AlexNet, VGG16, and VGG19, optimizing configurations based on predefined specifications.

Leveraging Cadence Stratus High Level Synthesis (HLS) tool, we model these dataflows to balance performance, area, and power trade-offs, providing insights into memory access patterns, computational efficiency, and hardware constraints for informed architectural decisions before RTL implementation. AAC provides information on both the software and hardware requirements of architectural platforms during the early design phase, beginning from algorithms at the top level, which mitigates the potential pitfalls of system integration failure in SoC design.

II. Methodology

A. AAC Based Architectural Exploration via Dataflow Modeling

As depicted in Fig. 1, the key essence of vertical integration, a cross-level, top-down methodology is to obtain lower-level information, especially software and hardware design

979-8-3315-7206-8/25/$31.00

parameters for the SoC, to make appropriate design choices during early design stage, starting from top level. According to specification pre-defined for targeted application, a proper design solution must be chosen at each abstraction level, starting from algorithm, system architecture, microarchitecture to physical circuit levels. Making proper decisions as early as the algorithmic level is pertinent in preventing design changes, which are more difficult at later design stages at lower levels.

In this paper, based on AAC, we analyzed different algorithmic realizations in the form of dataflow model (DFM), which consist of both algorithmic behavior and architectural implementation information. Each dataflow contains a different processing scheduling or order at a specific data granularity and can be considered a different architecture instantiation or realization for the same algorithm. As depicted in Fig. 1, AAC explores the joint algorithm and architecture co-design space during architectural exploration. In essence, it provides guidance in exploring various DFMs of an algorithm. Subsequently, we performed quantitative analysis based on algorithmic intrinsic complexity metrics and chose the most suitable DFM to be mapped onto an optimal architectural design, in satisfying the pre-defined specification. As such, the dataflow modelled using SystemC in this research serves a bridge in mapping algorithms onto architectural platforms.

B. Algorithmic Intrinsic Complexity Analysis for HLS

Complexity analysis of AI algorithms, in accommodating versatile scenarios and platforms, poses a major challenge in system design. Conventional assessment of computer algorithm's complexity was performed via software execution times based on an ideal single-thread machine with considerations of infinite memory and bandwidth. However, advanced SoC designs require highly precise complexity measures, that are platform independent, which could be used for both software and/or hardware. As such, we have used the four metrics to measures the algorithmic intrinsic complexity of different algorithmic realizations or DFGs: (I) number of operations, (II) data storage (III) data transfer rate, and (IV) degree of parallelism.

The *number of operations* characterizes the number of arithmetic operations in a DFM. Based on the specification required, the datapath hardware and initial clock cycles could be estimated using this parameter.

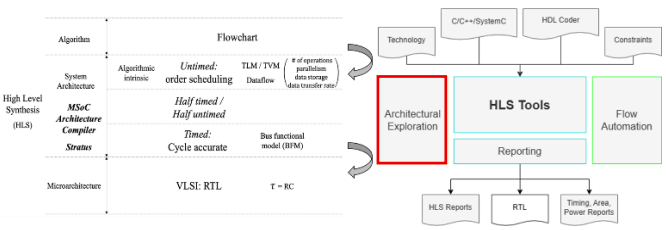

Fig 1. High Level Synthesis: Joint Algorithm/Architecture Co-Design Space Exploration

Data storage, as estimated from the DFM, is the buffer size required owing to data lifetime. This is affected by data causality, whereby some necessary data must be maintained for a certain period for subsequent processing. By exploring different data granularities and processing orders of DFGs, the system might achieve lower requirements for data storage. This data storage metric constitutes the second highest power consumption factor among these complexity metrics.

The *data transfer rate* is the amount of data communicated between modules and when estimated from DFG, characterizes average bandwidth which is algorithmic-intrinsic. The peak bandwidth is measured when the targeted platform is specified. The data transfer metric constitutes the highest power consumption factor in complexity analysis.

The *degree of parallelism*, at the data level, is the potential of a DFG to be parallelized to increase the throughput. This feature enables a reduction in power consumption by lowering the working frequency or shortening the working speed while providing the same throughput.

Transparent to either software or hardware, these four metrics quantitatively measure the complexity which is intrinsic to the algorithm and are therefore platform independent. As such, the DSE facilitates complexity assessment of algorithms which could be mapped onto all computational platforms, including embedded systems and in this case SoC. The purpose is primarily to first gain insight into the hardware and software ingredient required to map the unrolled datapath hardware pipelining, memory requirement, and average bandwidth with corresponding clock frequency in timing.

Upon progression to later system level design stages, parameters corresponding to selected platform for targeted applications, would be incorporated. As such, the system is further optimized via software/hardware co-design towards embedded software and/or VLSI hardware design at the microarchitecture level. In our current study for SoC design, 45nm process technology was synthesized using Cadence's Stratus HLS tool. Re-timing in this step facilitates refinement of the goodness-of-fit of the unrolled datapath pipeline estimated from algorithmic intrinsic complexity to optimally fit the architectural platform, under the constraint of physical parameters, which is the 45nm process for this study.

It is noteworthy that these timing and retiming iterations in the previous two corresponding steps are required to assure joint optimization of the algorithm and architecture to assure the design meets the pre-defined specification! Subsequently HLS was used to automatically convert high-level algorithmic descriptions (typically written in C, C++, or SystemC) into RTL implementations.

TABLE I. Notations of Convolution for Algorithmic-Intrinsic Parameters Analysis

Convolution	
Ifmap size	$W \times H$
Ofmap size	$W_o \times H_o$
Filter size	$K \times K$
Input channel	C
Filter numbers/ Output channel	N
PE array size	$P \times P$

TABLE II. Notations and Matrix Size of GEMM for Algorithmic-Intrinsic Parameters Analysis

GEMM		
Matrix	Rows	Columns
Input matrix X	CK^2	W_oH_o
Weight matrix H	N	CK^2
Output matrix Y	N	W_oH_o

TABLE III. Algorithmic-Intrinsic Parameters Analysis of OS, WS and IS Dataflow Models

Parameter	**Unit**	**OS**	**WS**	**IS**
Number of Operations	MACs	$W_oH_oCNK^2$	$W_oH_oCNK^2$	$W_oH_oCNK^2$
Degree of Parallelism	MACs	P^2	P^2	P^2
Unit Time	Cycles	$CK^2 + 3P$	$H_oW_o + 3P$	$N + 3P$
Total Time	Cycles	$W_oH_oN(CK^2 + 3P) \,/\, P^2$	$CNK^2 \times (H_oW_o + 3P) \,/\, P^2$	$W_oH_oCK^2 \times (N + 3P) \,/\, P^2$
Data Storage	Bytes	$2CPK^2$	$2PW_oH_o + CPK^2$	$CPK^2 + 2NP$
Data Transfer Rate	Bytes/sec	$\frac{2CP^3K^2 \times clock\_rate}{W_oH_oN \times (CK^2 + 3P)}$	$\frac{(2P^2 + 2PW_oH_o) \times clock\_rate}{W_oH_o + 3P}$	$\frac{(2NP + 2P^2) \times clock\_rate}{N + 3P}$

C. *Architectural Exploration Example: Dataflow Modeling of Systolic Array*

To evaluate the effectiveness of these models in achieving a balance between speed, power, and scalability, the CNN algorithm is formatted in General Matrix Multiplication (GEMM) mode. In this mode, the three primary dataflow models—OS , WS, and IS—are based on data stationarity and on distinct aspects of data reuse. These dataflow models focus visually represented in Fig. 2. Fig. 2 (a) illustrates how convolution operation maps to GEMM. Fig. 2 (b), (c), (d) illustrate how the input data, filter data and output data are arranged and move through the systolic array. In OS, the input data and filter data stream in the PE array from the buffer, while partial sums accumulate in each PE. After all accumulation is over, the output data flow out the PE array. In WS, filter data are preloaded in the PE array. Input data stream in the PE array from the buffer, while partial sums accumulate downward and flow out of the PE array to the buffer. In IS, input data are preloaded in the PE array. Filter data stream in the PE array from the buffer, while partial sums accumulate downward and flow out of the PE array to the buffer.

In this work, we explore three systolic array dataflow models—OS, WS, IS. Each model optimizes distinct aspects of computation for improved memory reuse and performance: OS reduces output memory bandwidth by optimizing output reuse, WS minimizes weight memory access to enhance weight reuse and IS focuses on input reuse, particularly when input activations dominate. Each dataflow model offers an optimal trade-off depending on the specific computational scenario.

We estimated the algorithmic-intrinsic parameters of OS, WS and IS with four parameters: number of operations, degree of parallelism, data transfer rate and data storage. The estimation result is shown in Table III. The notations are presented in Table I. W and H are the width and height of the ifmap. W_o and H_o are the width and height of the ofmap. C is the number of ifmap channels and filter channels. N is the number of filters and ofmap channels. P is the width and height of the PE array. In GEMM, the size of the input matrix, weight matrix and output matrix are shown in Table II.

Number of operations estimates the number of MACs for one convolution layer. Degree of parallelism estimates the number of MACs can be processed at the same time. Data transfer rate estimates the average number of data transferred per second for one convolution layer. Data storage estimates the buffer size for ifmap, filter and ofmap. Unit time estimates the cycles to process the input data and filter data from the buffer. Total time estimates the cycles to complete a convolution layer. Unit time and total time are used to estimate the data transfer rate and throughput of the dataflow. All three dataflow models have the same number of operations and degree of parallelism due to the same convolution layer and PE array size. OS can process more data at a time which requires larger data storage and data transfer rate, it can have higher throughput with the tradeoff of larger chip area. IS and WS have similar dataflow, resulting in similar parameters, with IS having slightly higher data storage and data transfer rate.

III. Experimental Result & Discussion

The HLS evaluation of systolic array configurations was conducted using the Cadence Genus 45nm technology library. We compared frequency, area, power consumption, and accumulator bit-width requirements across different configurations to assess their efficiency. The OS, WS, and IS

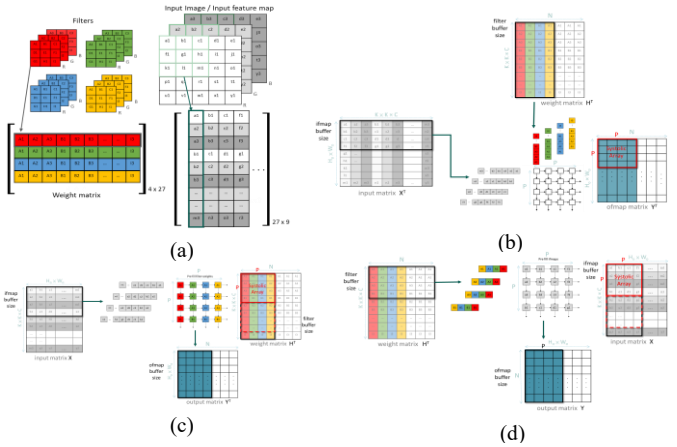

Fig. 2. Dataflow Model in GEMM Mode, where (a) GEMM, (b) OS Dataflow Model, (c) WS Dataflow Model and (d) IS Dataflow Model

systolic arrays, each consisting of a 4×4 PE structure, are depicted in Fig. 3, where computed values accumulate downward.

For the Output Stationary (OS) configuration, the design utilized a 4×4 systolic array with 16 processing elements (PEs), where each PE was responsible for 8-bit input and weight, and 16-bit multipliers. Each PE performed 27 multiplications and 26 additions, requiring a accumulator of 21 bits to store the cumulative results. The OS configuration achieved the best performance in terms of output reuse, minimizing output memory bandwidth.

In the Weight Stationary (WS) configuration, the design also used a 4×4 systolic array with 16 PEs, where input and weight were 8 bits, and 16-bit multipliers were used. The accumulator width increased from 16 bits in the top row to 18 bits in the bottom rows of the systolic array. Each PE preloaded filter weights, while the input feature map streamed through, optimizing weight reuse. This configuration effectively minimizes weight memory access and maximizes weight reuse during computation.

For the Input Stationary (IS) configuration, a similar 4×4 systolic array with 16 PEs was used, where each PE handled 8-bit input and weight, with 16-bit multipliers. The accumulator width increased from 16 bits in the top row to 18 bits in the bottom rows, just like the WS configuration. This setup optimized input reuse by preloading the input data into the PEs, while filter weights streamed through and partial sums accumulated, providing efficient input reuse.

To compare the logic synthesis results and the impact of different SystemC dataflow models and directives on area and power consumption, we present the synthesis result in Table IV. OS has higher area, higher power, and lower clock frequency because of the larger bitwidth of PE and larger buffer size, increasing the delay and power consumption of the circuit. But OS can provide better throughput than WS and IS. WS and IS have similar area and power. From the synthesis result, we can see that algorithmic-intrinsic parameters analysis helps us

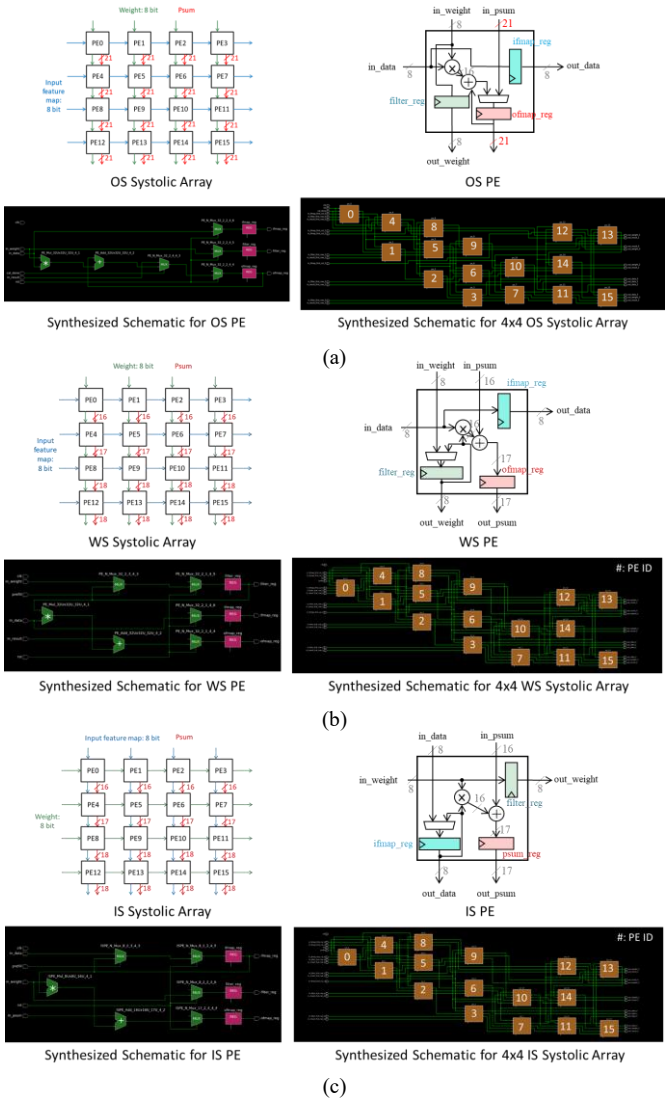

Fig. 3. Block Diagrams and Synthesized Schematics of Systolic Array and Processing Element (PE) for (a) OS, (b) WS, and (c) IS Configuration

estimate different dataflow models and lets us discover the optimal design for the specification.

We also synthesize and compare GEMM algorithmic code which computes GEMM directly without the systolic array dataflow models. Although the GEMM algorithmic code presents tradeoffs between throughput and area across different architectures by adding directives of retiming, unurolling and pipelining, the result is worse than systolic arrays

In summary, the experimental results highlighted the tradeoffs across the different systolic array configurations and the importance of architecture exploration. The OS configuration excelled in output reuse with minimal memory bandwidth, the WS configuration optimized weight memory access, and the IS configuration focused on input reuse. With architecture exploration, we explore various dataflow models and extract their algorithmic-intrinsic parameters to evaluate the tradeoffs of different dataflows and find the feasible design at the early design stage.

IV. CONCLUSION

This paper integrates system-level architecture exploration for early-stage design evaluation with high-level synthesis for micro-architectural optimization. It employs parallel computing using a systolic array for CNN matrix multiplication, offering a novel perspective on architecture evaluation and PPA optimization in the early design phase. The results highlight the effectiveness of co-design in enhancing both algorithmic efficiency and architectural implementation for deep learning models. This approach paves the way for further hardware design innovations, advancing AI technology through comprehensive design space exploration.

ACKNOWLEDGMENT

We sincerely appreciate the Cadence Group for their valuable support and funding for this project with Stratus HLS tools for dataflow modeling.

REFERENCES

[1] LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. Nature 521, 436–444 (2015). https://doi.org/10.1038/nature14539

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser and Illia Polosukhin, *Attention Is All You Need*, arXiv:1706.03762,2023

[3] K.-C. Chen, W.-H. Peng, and C. G. G. Lee, "Overview of intelligent signal processing systems," *APSIPA Transactions on Signal and Information Processing*, vol. 12, no. 1, e36, 2023, [Online]. Available: http://dx.doi.org/10.1561/116.00000053.

[4] H. T. Kung, Charles Eric Leiserson "Systolic Arrays for (VLSI)," *Carnegie-Mellon University. Department of Computer Science*; 79/103

[5] A. Samajdar, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, *Scale-sim: Systolic cnn accelerator simulator*, arXiv preprint arXiv:1811.02883, 2018.

[6] L. Du *et al.*, "A reconfigurable streaming deep convolutional neural network accelerator for internet of things," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 1, pp. 198–208, Jan. 2018.

[7] S.-Y. Chen, G. G. C. Lee, T.-P. Wang, C.-W. Huang, J.-H. Chen, and C.-L. Tsai, "Reconfigurable edge via analytics architecture," in *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, Hsinchu, Taiwan, 2019, pp. 117–121.

[8] T.-Y. Tsai, "System architecture exploration and dataflow model design for convolutional neural network accelerator based on systolic array," M.S. thesis, Department of Electrical Engineering, National Cheng Kung University, 2023.

TABLE IV. Synthesis Results of OS, WS and IS Systolic Array and GEMM Algorithmic Code

Systolic Array Dataflow Models			
Process	**SC_METHOD**		
Architecture	4×4 OS Systolic Array	4×4 WS Systolic Array	4×4 IS Systolic Array
Frequency (MHz)	448.11	488.04	488.04
Area (μm^2)	17119.49	13762.08	13762.08
Power (W)	1.28E-03	1.01E-03	1.01E-03
Process	**SC_CTHREAD with retiming directive**		
Architecture	4×4 OS Systolic Array	4×4 WS Systolic Array	4×4 IS Systolic Array
Frequency (MHz)	447.35	473.6	493.32
Area (μm^2)	11487.44	11115.34	10593.11
Power (W)	4.98E-04	4.77E-04	3.78E-04

GeMM Algorithmic Code				
Process	**SC_METHOD**	**SC_CTHREAD**		
Architecture	No	Fully Pipelined	No Loop Unrolling	Inner Loop Unrolling*
Frequency (MHz)	320.57	327.63	425.13	242.32
Area (μm^2)	407441.02	411767.32	19396.53	117684.6
Power (W)	2.08E-02	2.14E-02	1.15E-03	7.06E-03