SNA: A Siamese Network Accelerator to Exploit the Model-Level Parallelism of Hybrid Network Structure

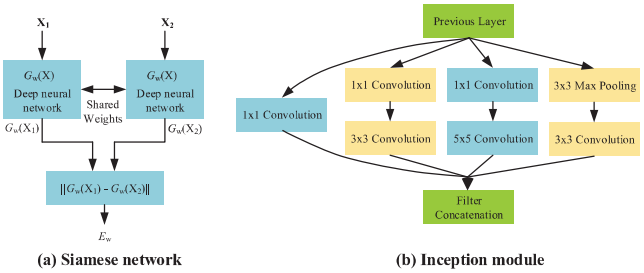
Xingbin Wang*†, Boyan Zhao*, Rui Hou*‡, Dan Meng*
*State Key Laboratory of Information Security, Institute of Information Engineering, CAS, Beijing, China
†School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

***Abstract*—Siamese network is compute-intensive learning model with growing applicability in a wide range of domains. However, state-of-art deep neural network (DNN) accelerators would not work efficiently for siamese network, as their designs do not account for the algorithm properties of siamese network. In this paper, we propose a siamese network accelerator called SNA, the first Simultaneous Multi-Threading (SMT) hardware architecture to perform siamese network inference with high performance and energy efficiency. We devise an adaptive inter-model computing resource partition and flexible on-chip buffer management mechanism based on the model parallelism and SMT design philosophy. Our architecture is implemented in Verilog and synthesized in a 65nm technology using Synopsys design tools. We also evaluate it with several typical siamese networks. Compared to the state-of-art accelerator, on average, the SNA architecture offers 2.1x speedup and 1.48x energy reduction.**

I. Introduction

Siamese network is an important development direction of deep neural networks (DNNs), which has been widely used in many fields such as image matching and retrieval, face verification, object/visual tracking [1]–[5]. The similarity metric is applied to compare or match new samples from previously-unseen categories (e.g. faces from people not seen during training), which is realized by training siamese network [1]. A typical siamese network architecture is composed of two identical DNN sub-networks sharing the same weights as shown in Fig. 1(a). And hybrid network structure is an important characteristic of siamese network. As shown in Fig. 1(b), the inception module [6] is a typical hybrid network structure that can reduce the parameters and computation of siamese network model, and improve its feature extraction ability.

Existing DNNs have rich data-level parallelism, and DNN accelerators focus mainly on exploiting data-level parallelism within Processing Element (PE) array [7]. Accordingly, siamese networks also contain rich model-level parallelism. Siamese network is usually composed of two identical DNN models that process input data by sharing parameters. This model-level parallelism cannot be fully exploited by typical DNN accelerator architecture. Moreover, the typical DNN accelerator is a one-way computation channel and unidirectional data read/write channel, so it can only execute a DNN model in serial. Therefore, siamese network requires the design of new architecture to exploit model-level parallelism.



(a) Siamese network **(b) Inception module**

Fig. 1. Siamese network and the typical hybrid network structure.

To fully utilize the model parallelism in siamese network, one natural idea is using multi-core or Simultaneous Multi-Threading (SMT) technology to accelerate it. However, we observe that using multi-core computing architecture to perform siamese network inference calculations will cause some problems.

- Processing cores designed with fixed computing resources and storage resources have low PE utilization when performing single-model inference. For example, the PE utilization of TPU for AlphaGo and LSTM is 46.2% and 8.2% [8]. The waste of resources will be more prominent in the multi-core structure.
- Static resource (computing resource and on-chip buffer) allocation on typical multi-core architecture does not adapt well to the computational characteristics of siamese network.

Based on the consideration of resource sharing, dynamic resource allocation and architectural support for chip security [17], we take the idea of SMT architecture to design siamese network accelerator.

According to our best knowledge, this paper is the first to propose a siamese network accelerator architecture called SNA, which can effectively exploit model-level parallelism of the hybrid network structures. Specific contributions are as follows:

- SNA architecture provides adaptive inter-model computing resource partition for siamese network acceleration.

This work was supported by the Strategic Priority Research Program of Chinese Academy of Sciences under grant No. XDC02010000. We thank the anonymous reviewers for their valuable comments and suggestions.
‡Corresponding author: Rui Hou (E-mail: hourui@iie.ac.cn)

It can make flexible adjustments according to the requirements of siamese network and configure the optimal computing resources for each branch of the hybrid network structure.

- We propose a flexible on-chip buffer management mechanism that can meet the different buffer requirements of siamese network. The siamese network requires two sets of input/output buffer and a shared weight buffer which can also be split into multiple sub-weight buffers based on the branch number of the hybrid network structure.
- SNA provides a hardware-software solution for better resource utilization and computing performance. We develop a compiler to generate the optimized data flow and hardware configurations for different siamese networks.

II. Background and Motivation

A. The algorithm properties of Siamese network

Siamese network is an important branch of DNN and has been successfully used in image local matching and face verification [1], [5]. Compared to the classical DNN structure, siamese network contains two levels of model parallelism: ❶ Siamese network usually consists of two identical DNN sub-networks with the same weights. Two sub-network models need to perform inference operations simultaneously, which is called *coarse-grained model parallelism*. ❷ DNN sub-network usually adopts a hybrid network structure that contains multiple branches. These branches can be executed simultaneously to improve system performance, which is called *fine-grained model-level parallelism*.

B. Challenges of siamese network

The existing single-chip ASIC architecture cannot effectively exploit the model-level parallelism of siamese network. From the perspective of computing architecture, most of the chips adopt the computing mode of the systolic array that cannot effectively support multi-path computing. For example, both TPU [9] and Eyeriss [7] take systolic array as the computing core, and such one-way and single computing path cannot effectively exploit the parallelism of siamese network. From the perspective of on-chip buffer, their input/output buffer design is centered around the systolic array and cannot effectively support multi-channel data paths. For example, in order to enable parallel processing of the two models of siamese network, the on-chip buffer needs to be divided into two parts, which are corresponding to the input and output buffer of each model. From the perspective of data communication, current architectures cannot effectively support data communication between models. Siamese network requires efficient communication mechanism in a single chip to meet its performance requirement, rather than moving data to off-chip DRAM for data communication. Accordingly, our new DNN accelerator architecture should satisfy the following properties:

- **Adaptive inter-model computing resource partition.** In order to satisfy the multi-branch parallel computation of the hybrid network structure and the simultaneous inference of double models, we need to design the computing resources in an adaptive way. It can be flexibly divided into multiple computing groups according to different computing requirements.
- **Flexible inter-model buffer management.** The granularity of traditional buffer allocation is at the function level. It cannot split the weight buffer according to the requirement of the hybrid network structure.
- **Efficient Inter-model communication.** Data communication exists between the two sub-DNN models of siamese network [2]. Using the existing DNN accelerator architecture, data interaction between two sub-DNN models can only be realized through accessing the off-chip DRAM which results in low communication efficiency.

III. SNA hardware architecture

A. Overview

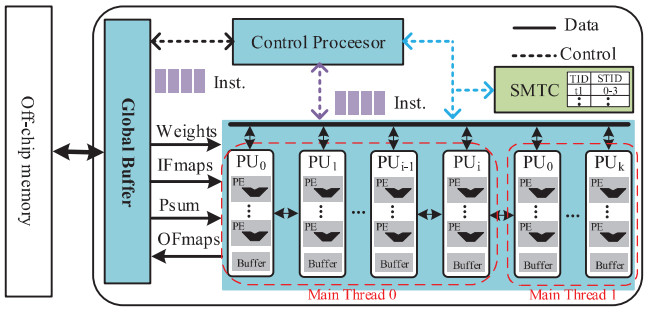


Fig. 2. The SNA architecture and its multi-level memory hierarchy.

As mentioned, SMT architecture is more suitable for exploiting the model-level parallelism than multi-core architecture. Specifically, we design an adaptive inter-model computing resource partition and a flexible on-chip buffer management unit to share computing and storage resources. The SNA architecture can provide double main threads for siamese network to fully exploit the coarse-grained model parallelism. Meanwhile, it also offers a sub-thread for each branch of the hybrid network structure to fully utilize the fine-grained model parallelism.

Typical DNN accelerators usually employ the systolic array as computing unit. The management circuit around the computing resources (internal connection bus and scheduling logic) has considerable area and power consumption cost. For example, researches [10], [11] show that the connection bus and scheduling logic occupy 8.9% of the chip area and 15.3% of the power consumption. To support multi-thread computation, we design a flexible Processing Units (PUs) array and PU controller to avoid the complex bus design and scheduling logic.

The siamese network accelerator architecture is shown in Fig. 2. The accelerator consists of a global buffer, a Control Processor (CP), SMT Controller (SMTC) and multiple identical PUs. Each PU which contains multiple separate compute lanes is equipped with an input buffer and an output buffer that communicate with the global buffer. The architecture also consists of dedicated units to support the computations of ReLU activation and pooling layer.

B. Processing unit

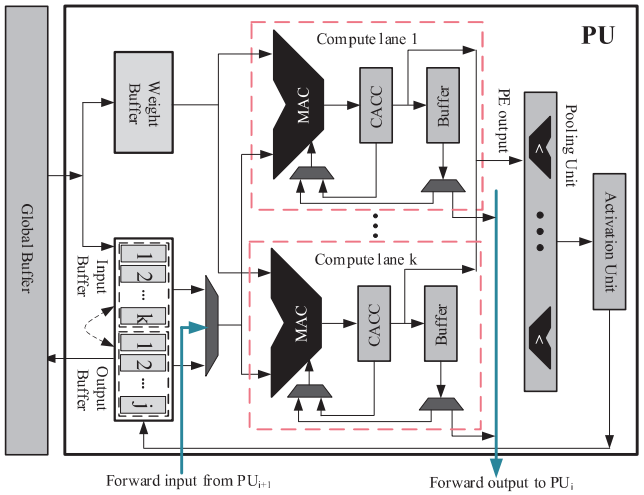


Fig. 3. The microarchitecture of PU.

Fig. 3 shows the microarchitecture of PU. Unlike classical PE array, each PU comprises multiple compute lanes, a weight buffer, an input/output buffer. The weight and input/output buffers are shared across all the compute lanes. Each compute lane consists of one dedicated Multiply-and-Accumulate (MAC) unit, Convolution Accumulator (CACC) and Forward Output Buffer (FOB).

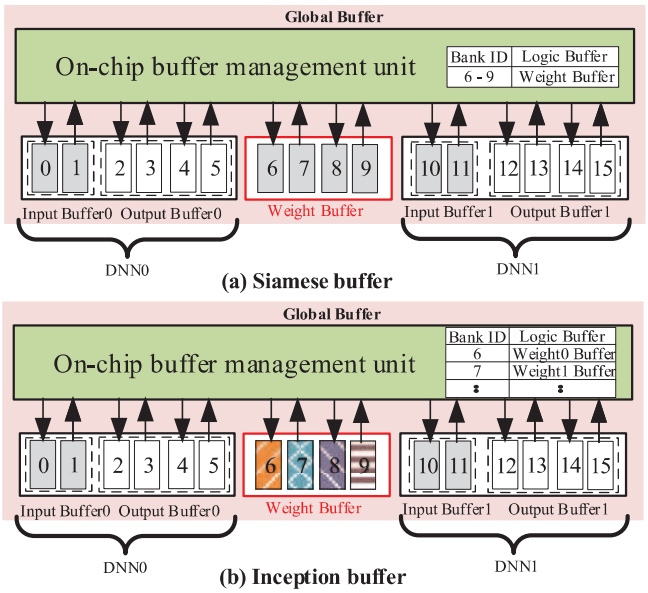


Fig. 4. The global on-chip buffer sharing and splitting.

Private input/output Buffers. A whole physical buffer can be logically divided into two sub-buffers for holding the input and output data of each layer. The logical partitioning supports the use of arbitrary sub-buffer as an input or an output buffer, thereby reusing the Output Feature maps (OFmaps) of layer i-1 as the Input Feature maps (IFmaps) of layer i. In this case, the data of each sub-buffer are logically swapped rather than copying data across banks physically. The weight buffer is also shared across all the compute lanes within each PU.

Data forwarding. To make the communication between the two sub-models of siamese network, we add a dedicated unidirectional link between adjacent PUs to forward their outputs. As shown in Fig. 3, these data are forwarded to be reused for next PU. This step not only reduces the bus traffic, but also improves the data communication capability between PUs. It is important to note that FOB can not only cache the output data of PUs, but also store the partial sum.

C. Flexible on-chip buffer management unit

Different network structures have different functional requirements for on-chip buffer. The traditional double buffer allocation for AlexNet and VGG is effective, but it is not efficient enough for DNNs with complex graph topology. For siamese network, we need to assign two private input/output buffers and shared weight buffer for the two sub-models. For inception module, the weight buffer needs to be split into sub-buffers to store the weights of different branches. Therefore, we propose a flexible on-chip buffer management unit (FBMU) to meet the various requirements of DNNs for the on-chip buffer.

Weight buffer sharing. As shown in Fig. 4(a), the compiler divides the global on-chip buffer into three parts according to the requirements of siamese network. The controller fetches one weight from the shared weight buffer and broadcasts it to all compute lanes. Simultaneously, the controller reads their IFmaps and delivers them to the corresponding compute lane to perform multiplication.

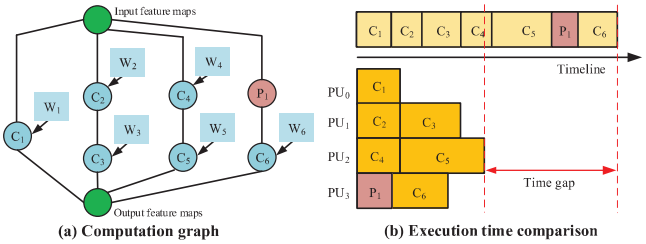


Fig. 5. The parallel compute flow for inception module.

Weight buffer splitting. As shown in Fig. 4(b), the weight buffer is divided into several sub-buffers according to the different branches of the hybrid network structure. Fig. 5(a) shows the computation graph of inception module, we observe that the lifecycles of dierent branches could be different, which provides the opportunity to economize buffer resource by sharing. Fig. 5(b) shows the execution time comparison between SMT architecture and typical DNN accelerator architecture. The SMT architecture significantly shortens the execution cycle of the inception module. We apply accurate liveness analysis for the whole computation graph to determine sub-weight buffer size for each branch, and use the buffer allocation algorithm [12] to allocate total weight buffer. The power consumption introduced by online sub-weight buffer calculation is negligible because the operation is accomplished by the configuration of the instruction registers.

D. Data communication within SNA architecture

In the design of SNA architecture, we employ two communication methods to meet the needs for data communication within siamese network.

Data movement between PUs. For the need of a high communication rate, data flow between PUs is the first choice. The unidirectional transmission paths between PUs can provide higher bandwidth. The data forwarding unit forwards the data to the adjacent PU in single layer, especially suitable for tiling techniques.

Data communication within FBMU. Owing to the deterministic execution sequence of DNN model, we design an on-chip buffer management unit to access data by instructions directly. Furthermore, a sophisticated bus arbiter is provided to reuse on-chip feature maps and reduces off-chip feature maps traffic. We also adopt tilling technology to split feature maps.

IV. SNA SOFTWARE ARCHITECTURE

The compiler plays a key role in software architecture. It not only maps the high-level network definitions to customized instructions for SNA architecture, but also schedules and allocates hardware resources required by the main thread and sub-threads of siamese network, such that maximize data reuse and best utilize accelerator's on-chip buffer. And it also creates a static schedule for the global buffer to fetch IFmaps or weights for off-chip memory and feed the PUs through the bus fabric. Static scheduling avoids contention on the bus and alleviates the need for PUs to perform complex handshaking. This approach improves the scalability and efficiency of SNA.

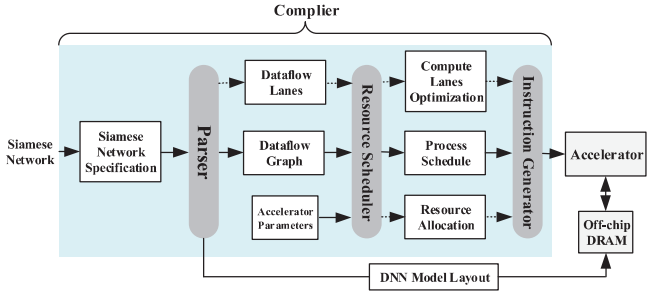


Fig. 6. Software workflow for SNA and complier generates execution schedule instructions.

A. SNA Software Workflow

Fig. 6 shows the software workflow of SNA which takes siamese network specification (e.g. Caffe format) as input. The first step is to elicit the structure and weights of siamese network and make a directed graph with nodes representing the network operations through *Parser*. Then the *Parser* generates dataflow lanes or dataflow graphs depending on the parallel mode and the model layout in off-chip memory.

Resource scheduler (include hardware resource search algorithm) allocates resource based on the dataflow graphs to optimize the computing resource and on-chip buffer. Compute lanes optimization and resource allocation aim to strike a balance between model parallelism and maximum performance by splitting buffer and slicing computation. Experimental results show that the inference performance of hybrid network structures or sub-models is positively correlated with the allocated computing resource and has low sensitivity to the allocated on-chip buffer resource. Therefore, we first allocate sufficient computing resource to them, and then allocate corresponding on-chip buffer resource according to their requirements.

Instruction generator produces a static execution schedule and the configuration parameters of SNA in accordance to resource allocation results from hardware resource search algorithm. Although increasing the amount of PU on a branch of the network structure potentially provides higher parallelism, higher computing resource allocation does not mean better performance. The computation asynchronization leads to computation redundancy, which seriously affect execution performance. Therefore, it is imperative to properly allocate hardware resources.

B. SNA Software Optimization

Hardware Resource Search algorithm. The algorithm takes siamese network specification and hardware parameters, and returns a list named ParamSNA that stores the value of the resource allocation parameters. The algorithm first divides the accelerator into two main threads which contain multiple sub-threads according to the characteristic of siamese network. Then, it adjusts the hardware resource for the two models of siamese network through a greedy search such that they cooperatively achieve maximum performance while maintaining high resource utilization. Accordingly, we break the algorithm into two main stages as follows:

Profiling stage. We propose a function called *ThreadResourceProfiling* which profiles the number of sub-threads for two main threads and corresponding computation resource and on-chip buffer size. The proper profiling results in terms of the SNA are accumulated in a list.

Optimization stage. The optimization stage evaluates the combined effects of two main threads and their sub-threads and determines the proper resource allocation for them. To avoid the complexity of evaluating the combined effects, the optimization stage consists of two functions: *ComputeLanesOptimization* and *GlobalOptimazition*.

ComputeLanesOptimization. To utilize the algorithm properties of siamese network effectively, the function needs to determine: ❶ its associated number of PU for each branch while the compute unit utilization is maximized; ❷ maximum overall performance of parallel hybrid network structure (e.g. Inception module). To determine a proper set of parameters, we formulate the problem as a multi-objective optimization problem, and provide a greedy algorithm to solve it.

The second function, *GlobalOptimazition*, evaluates the effect of multi-threads performed in siamese network simultaneously and adjusts their resource allocation to achieve maximum performance. The output of the function is the

hardware resource allocation parameters in siamese network which are stored in the list ParamSNA.

V. Evaluation

A. Methodology

Architecture implementation. To evaluate the performance and efficiency of SNA, we implement the SNA architecture in Verilog. We use Synopsys Design Compiler and a SMIC 65-nm standard-cell library to synthesize it to evaluate the performance and the cost of chip area, timing, and energy.

TABLE I
SUMMARY OF SEVERAL POPULAR SIAMESE NETWORKS.

Siamese Network	Layers	Max Weights	Total MACs	Total Weights
Siamese [1]	6/6	400KB	8.95M	862K
MSP-CNN [3]	7/7	276.5KB	1.98G	12.3M
SiamRPN++ [2]	16/16	0.98MB	2.72G	19.2M

Benchmarks. Three popular siamese networks are selected as targeted workloads. Table I summarizes the details of these siamese networks.

Baselines. We compare SNA with state-of-the-art DNN accelerator Eyeriss [7]. The SNA architecture contains 64 PUs, each PU with 8 compute lanes, with a total of 512 MAC units. Table 2 lists the major architectural components of the SNA and Eyeriss. For a fair comparison, we re-implement Eyeriss with the same number of PEs as ours. Besides, we also allocate the same on-chip buffer size (1.5MB) to Eyeriss.

TABLE II
SNA AND EYERISS DESIGN PARAMETERS AND AREA BREAKDOWN.

	Architectural components	SNA		Eyeriss	
		Size	*Area (mm^2)*	*Size*	*Area (mm^2)*
PU	Compute Lane/PE	8	0.048	4	0.006
	Partial Sum Register	N/A	0	48B	0.004
	Input Register	N/A	0	24B	0.002
	Weight Buffer	5KB	0.291	0.5KB	0.028
	Input/Output Buffer	10KB	0.26	N/A	0
Accl*	Compute Unit	64 PUs	38.3	512 PEs	20.48
	Control Unit	–	2.64	–	1.61
	Global Buffer	800KB	12.26	1.5MB	32.19
Total area		57.89 mm^2		54.28 mm^2	

*represents accelerator.

Microarchitecture simulator. We develop a cycle-level microarchitecture simulator that takes the configuration of the SNA architecture as input and simulates the execution to calculate the cycle counts as well as the number of accesses to on-chip global buffer and off-chip memory. The cycle counts of the simulator are verified by the Verilog implementation of the SNA architecture. To evaluate the energy efficiency, we use CACTI-P [13] obtain the area and the power of the register files and on-chip buffers.

B. Experimental Results

1) Comparison to Eyeriss

Performance. Fig. 7 shows the performance and energy reduction of SNA in comparison with Eyeriss. On average, SNA delivers 2x speedup since the two-level model parallelism of siamese network is fully utilized. In addition, FBMU offers an opportunity to reuse feature maps and reduce off-chip memory traffic. Depending on the types of siamese network computation, the benchmarks have different performance gains. SNA achieves the highest speedup of 2.31x and lowest speedup of 1.78x over Eyeriss for MSP-CNN [3] and siamese network [1], respectively. MSP-CNN [3] with inception module and SiamRPN [2] with shortcut connection achieve higher performance gains because their convolution operations have rich parallelism and are more amenable for data reuse.

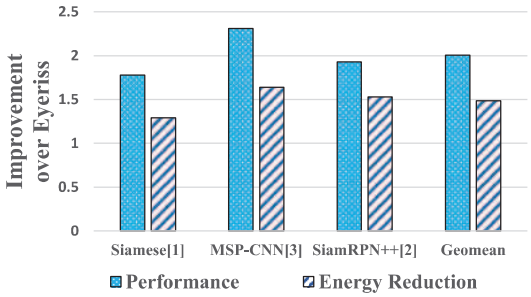


Fig. 7. SNA performance and energy reduction over Eyeriss.

Energy reduction. Compared to the Eyeriss, the average energy reduction of the SNA architecture is about 48%. The energy reduction mainly attributes to off-chip memory access reduction and communication between two sub-DNN models. MSP-CNN [3] has the maximum benefit of 1.64× energy reduction. SNA can effectively support the simultaneous execution of two DNN models and is equipped with two communication modes to improve the communication efficiency for them. SNA architecture offers a fundamentally different approach from Eyeriss and explores the model-level parallelism for siamese network, which significantly improves performance and energy.

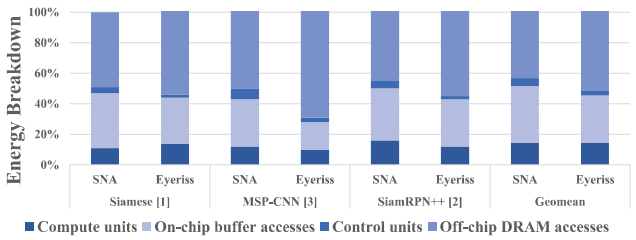


Fig. 8. Energy breakdown of SNA and Eyeriss.

Energy breakdown. To see the sources of the energy reduction, we break down the energy consumption for each hardware component (Compute units, On-chip buffer accesses, Off-chip DRAM accesses, Control units). Fig. 8 illustrates the per-component energy dissipation for SMT and Eyeriss. The results show that both accelerators consume more than 80% of energy for on-chip and off-chip memory access. The FBMU and SMT process mechanism increase energy consumption for on-chip buffers, but they lead to fewer off-chip memory access. Eyeriss uses the local register file within PE, which constitutes a significant portion of energy consumption. The SNA architecture employs explicit data sharing for input and

partial sum to avoid the need for register files. In turn, this incurs more on-chip buffer accesses.

Area breakdown. Table II summarizes the area of the major micro-architectural components in SNA and Eyeriss. Overall, SNA needs about 6.67% more area compared to Eyeriss. The increase in the area is mainly attributed to the control logic of FBMU and the SMT controller.

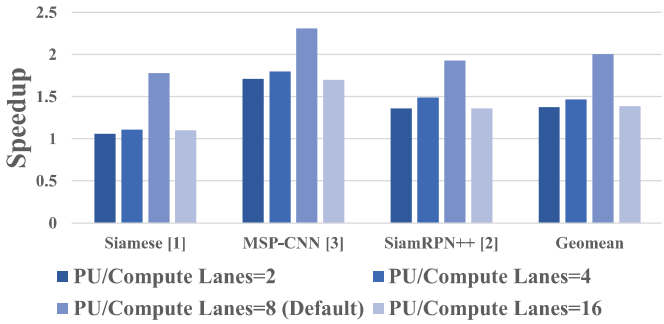


Fig. 9. Sensitivity of speedup with DNN-SMT over Eyeriss to the number of compute lanes per each PU.

2) Sensitivity study

Sensitivity to the number of compute lanes. Fig. 9 shows the impact of varying the number of compute lanes within each PU. We select SNA configuration of eight compute lanes for each PU as baseline. And the Eyeriss is equipped with the same number of compute elements. Experimental results show that increasing the number of compute lanes to 16 incurs a 29% performance decrease. Increasing the number of compute lanes potentially improves the parallelization level. However, the performance improvement diminishes due to the underutilization of computing resource and the limitation on private buffer size in each PU. Also, if we reduce the number of compute lanes to 2 and 4, the performance decreases are 27.7% and 22%, respectively. The reason for this behavior is mostly because of the insufficient computing power for PU. The diversity of applications requires different hardware resource allocation strategies, resulting in different performance. Therefore, these parameters need to be determined based on hardware architecture and computation requirement.

VI. Related work

The new and popular siamese networks are highly parallel hybrid network structure. We exploit the inherent algorithmic structure of siamese network and runtime information to judiciously execute SMT and improve the performance of SNA. We also study the rich and unexplored area of model parallelism in the domain of siamese network. Below, we discuss the most related works.

DNN accelerators. Several DNN accelerators have been proposed [7], [9], [11], [12]. Eyeriss [7] presents an optimized row-stationary dataflow for DNNs to improve efficiency. DaDianNao [14] uses eDRAM to eliminate off-chip accesses and provides high performance and efficiency for DNNs. HyPar [15] uses multiple accelerators to explore the coarse-grain parallelism. Learn-to-Scale [16] proposes the communication-aware sparsified parallelization on the embedded on-chip multi-core accelerators. None of these accelerators evaluates the benefits of model parallelism in the convolution operations and also accelerates the siamese network.

VII. Conclusion

Motivated by the observation that siamese networks employ highly parallel hybrid network structures which contain rich model parallelism, we propose a siamese network accelerator architecture to perform inference with high parallelism. The SNA architecture employs the SMT technology with the adaptive inter-model computing resource partition and flexible on-chip buffer management mechanism that enable it to fully exploit the model parallelism of siamese network. It also provides a unified solution that works for the existing DNNs. Specifically, DNNs with a hybrid structure can be accelerated with a higher performance boost.The experimental results show that SNA achieves significant speedup and energy reduction compared to state-of-the-art accelerators. Our SNA architecture provides the first proof-of-concept for realizing siamese network hardware, and open a new avenue for its high-performance inference deployment.

References

[1] Sumit Chopra et al. Learning a similarity metric discriminatively, with application to face verification. CVPR'05, pages 539–546, 2005

[2] Bo Li et al. Siamrpn++: Evolution of siamese visual tracking with very deep networks. CVPR'19, pages 4282–4291, 2019

[3] Chen Shen et al. Deep siamese network with multi-level similaritytyperception for person re-identification. In 25th ACM MM'17, pages 1942–1950. ACM, 2017.

[4] Qiang Wang et al. Learning attentions: residual attentional siamese network for high performance online visual tracking. CVPR'18, pages4854–4863, 2018

[5] Zagoruyko et al. Learning to compare image patches via convolutional neural networks. CVPR'15, pages 4353–4361, 2015.

[6] Christian Szegedy et al. Rethinking the inception architecture for computer vision. CVPR'16, pages 2818–2826, 2016.

[7] Yu-Hsin Chen et al. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. ISCA'16, pages 367–379, 2016.

[8] Liu Bosheng et al. Addressing the issue of processing element underutilization in general-purpose systolic deep learning accelerators. ASP-DAC'19, pages 733–738. ACM, 2019.

[9] Jouppi et al. In-datacenter performance analysis of a tensor processing unit. ISCA'17, pages 1–12. IEEE, 2017.

[10] Nowatzki et al. Hybrid optimization/heuristic instruction scheduling for programmable accelerator codesign. InPACT, pages 36–1, 2018.

[11] Hyoukjun Kwon et al. Maeri: Enabling flexible dataflow mapping over dnn accelerators via reconfigurable interconnects. SIGPLANNotices, volume 53, pages 461–475. ACM, 2018.

[12] Yijin Guan et al. Fp-dnn: An automated framework for mapping deep neural networks onto fpgas with rtl-hls hybrid templates. FCCM'17, pages 152–159. IEEE, 2017.

[13] Sheng Li et al. Cacti-p: Architecture-level modeling for sram-based structures with advanced leakage reduction techniques. ICCAD'11, pages 694–701. IEEE Press, 2011.

[14] Yunji Chen et al. Dadiannao: A machine-learning supercomputer. MICRO'14, pages 609–622. IEEE Computer Society, 2014.

[15] Linghao Song et al. Hypar: Towards hybrid parallelism for deep learning accelerator array. HPCA'19, pages 56–68. IEEE, 2019.

[16] Kaiwei Zou et al. Learn-to-scale: Parallelizing deep learning inference on chip multiprocessor architecture. DATE'19, pages 1172–1177. IEEE,2019.

[17] Dan Meng et al. Security-first architecture: deploying physically isolated active security processors for safeguarding the future of computing, Cybersecurity, 2018.1.5, 1(2): 1 12.