OPTIMO: A 65nm 270MHz 143.2mW Programmable Spatial-Array-Processor with a Hierarchical Multi-cast On-Chip Network for Solving Distributed Optimizations

Muya Chang, Li-Hsiang Lin, Justin Romberg, Arijit Raychowdhury
School of ECE, Georgia Institute of Technology, GA, USA

***Abstract*—This paper presents OPTIMO, a 65nm, 16-b, fully-programmable, spatial-array processor with 49-cores and a hierarchical multi-cast network for solving distributed optimizations. We demonstrate six template algorithms and their applications and measure a peak energy-efficiency of 0.279 TOPS/W.**

***Keywords*—optimizations; array-processing; multi-cast network**

I. Introduction

The explosion of big-data problems arising in statistics, machine learning (ML), image processing, 5G systems and other related areas [1] have accelerated the development of hardware prototypes that rely on data-flow architectures and near-memory processing to address the memory-bottleneck. Looking beyond the success of neural network (NN) accelerators for classification [2-5], we recognize a growing need for solving complex optimization problems, which arise in all areas of signal processing such as ML model-training, computational imaging (medical, optical and hyper-spectral), resource-allocation in 5G massive MIMO networks and solving inverse problems such as LDPC decoding. In spite of the diversity of applications, a common mathematical framework, namely solving constrained-optimizations (i.e., *minimize l(x)* under a constraint *r(x)=0* for a vector *x* and functions *l* and *r*) has emerged; and recent advances in the alternating direction method of multipliers (ADMM) has been particularly successful for large-scale problems [1]. A review of the ADMM algorithm is beyond the scope of this article, but it suffices to say that ADMM is one of the most common iterative algorithms for solving a large class of optimization problems and interested readers are pointed to [1] for a detailed survey. From a hardware perspective, ADMM in a distributed form is very powerful as it relies on local, iterative computing on a subset of the data (local memory) along with periodic exchange of information with near/distant neighbors (reconfigurable data-flow) to converge to a global solution (called consensus), as shown in Fig. 1. In this paper, we present OPTIMO, a spatial-array-processor with near-memory processing, a hierarchical and multi-cast on-chip network, and full-programming support for solving distributed optimizations via ADMM. We demonstrate six template algorithms: (1) least-squares optimizations, (2) least absolute shrinkage and selection operator (LASSO), (3) elastic-net, (4) linear support-vector machine (SVM), (5) group-LASSO and (6) distributed averaging. The algorithms use different objectives and constraints and represent a vast majority of statistical algorithms that are used on big-data sets. We note a 4.77x (4.18x) improvement in energy (performance) compared to a GPU-style SIMT machine with a shared memory (Fig. 1). To the best of our knowledge, this is the first *at-scale* demonstration of a programmable array processor for solving a set of optimization problems that are used for various emerging applications, such as training of NN, computational imaging etc.

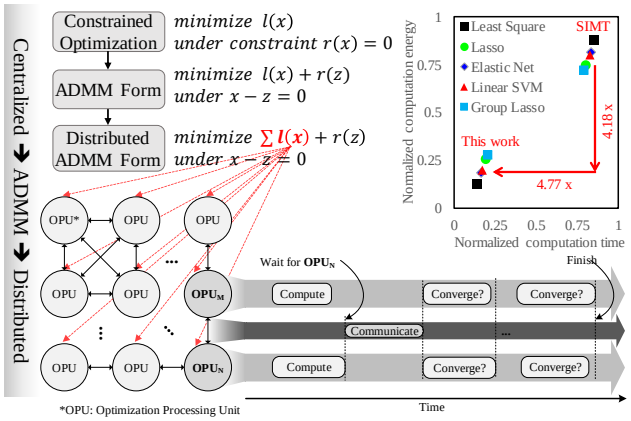


Fig 1: OPTIMO: A patial array processor for solving distributed optmizations via distributed ADMM. This work (measured) shows 4.77x (4.18x) improvement in energy (performance) compared to a GPU-style SIMT machine (simulated).

II. System Architecture and Design

A. System Design

Fig. 2 illustrates the chip-architecture where 49 programmable 16b OPUs (optimization processing units) indexed as (row, column) (1) compute locally and iteratively and (2) transmit/receive data from the neighbors. The OPUs interact via a 2-layered multi-cast network with (1) layer-0 establishing near-neighbor (neighborhood of 8) bi-directional connections

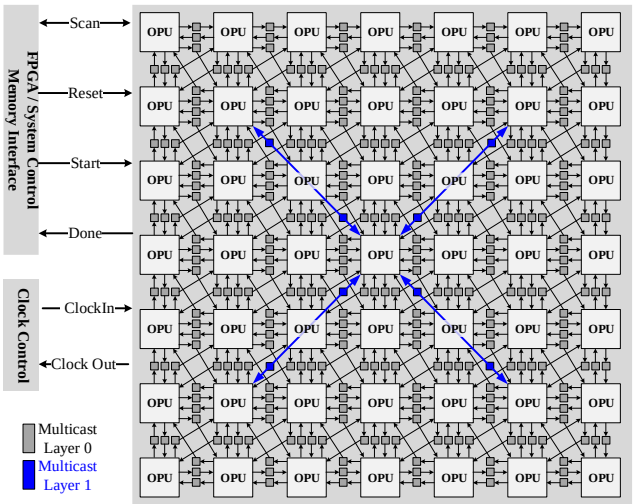


Fig 2 System Architecture showing the 49 OPUs and a 2-layer multi-cast on-chip network.

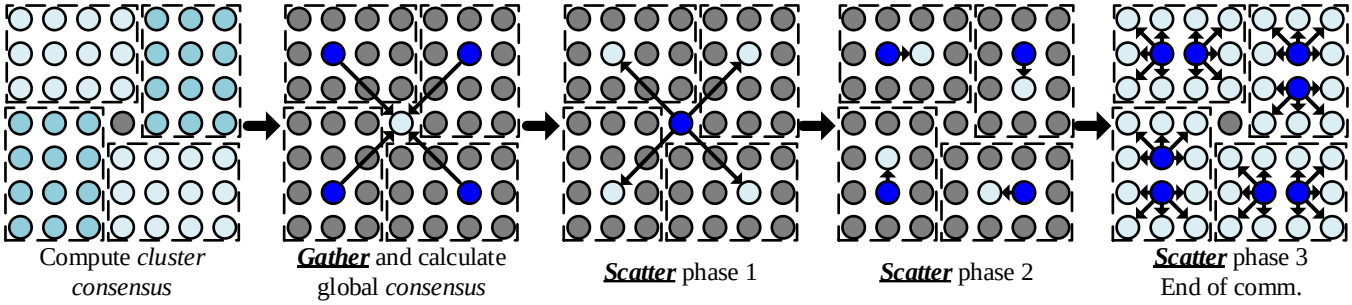


Fig. 3: Gather and scatter processes of communication enabled by a hierarchical on-chip network result in fast convergence to a global consensus.

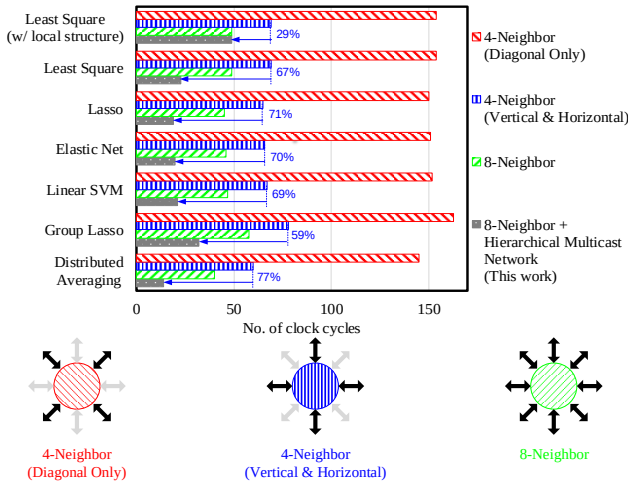


Fig 4: Time for convergence for template algorithms as a function of their connectivity with their neighbors.

and (2) layer-1 connecting 4 cluster center OPUs i.e, (2,2), (6,2), (2,6) and (6,6) with the chip-center OPU i.e, (4,4). Depending on the algorithm and structure of the data, optimization algorithms require complex data-flow patterns where both near-neighbor (layer-0 connections) as well as global information (layer-0 and layer-1 connections) are used. The 48-OPUs are divided into four clusters as shown with one in the center as shown in Fig. 3. Global consensus is reached in each iteration via: (1) the four clusters reach cluster-level consensus (layer-0), (2) gather process where the chip-center obtains cluster-level consensus information from cluster centers (layer-1) and calculates the global data, (3) scatter (step-1) process where the chip-center scatters the global data back to the cluster centers, and (4) a final scatter (step 2 and 3) process where the cluster-centers spreads the data across all the OPUs. This readies all the OPUs for the next iteration. The scatter and gather processes are intrinsic to distributed optmizations as the system computes locally, distributes information gloabally and iterates to reach concensus. We compare the proposed hierarchical multi-cast network with networks that allow 4 or 6 connections to the neighbors – as is common in convolutonal and deep neural networks [2]. Architectural and network simulations of various optmization algorithms on >10000 random data-sets reveal a 29%-77% reduction in convergence time compared to a fixed, 4-neighbor TPU-style connection (Fig. 4).

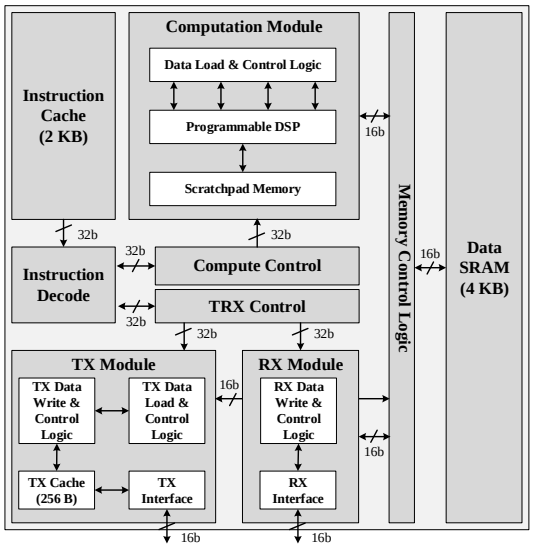
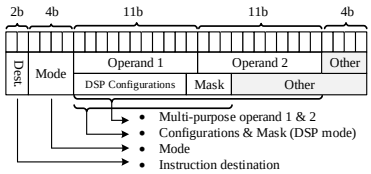


Fig 5: Architecture of the Optimization Processing Unit (OPU) showing the principal modules

B. *Optimization Processing Unit*

Each OPU features (Fig. 5): (1) one computation module consisting of a programmable digital signal processor (P-DSP), a scratchpad memory and control logic, (2) 2KB of instruction cache, (4) 4KB of data memory (for local data R/W), and (5) a transceiver module for the gather and scatter processes. Programming is supported via 32b instructions (Fig. 6) and each inter-OPU data-movement is supported on dedicated links. Before data-transmission, a transmit buffer temporarily stores the data and it is flushed out at the end of the transmission. Received data is not buffered; instead the control logic directly writes the incoming traffic to the data cache thereby reducing both latency and energy. The design supports synchronous, mesochronous as well as asynchronous communication among OPUs with bidirectional FIFOs enabling fast and parallel data exchange across clock-crossing boundaries. Fig. 6 further



Functional Blocks	No. of Instr.
Computation Controller	11
TRX Controller	8
Programmable DSP	768

Macro Functions	No. of instr.	Algorithm Type					
		1*	2*	3*	4*	5*	6*
L_1 Norm	3		✓	✓			
$(L_2 \text{ Norm})^2$	3	✓	✓	✓	✓	✓	
L_2 Norm	6		✓				
L_{Inf} Norm	3		✓				
Shrinkage	3		✓	✓			
MAC	4	✓	✓	✓	✓	✓	✓
Hinge Loss	3				✓		
Distributed Averaging	10	✓	✓	✓	✓	✓	

1*. Least Square
2*. Lasso
3*. Group Lasso
4*. Elastic Net
5*. Linear SVM
6*. Distributed Averaging

Fig 6: Programming support is enabled via a custom instruction set architecture with a 32-b Instruction format and macro functions

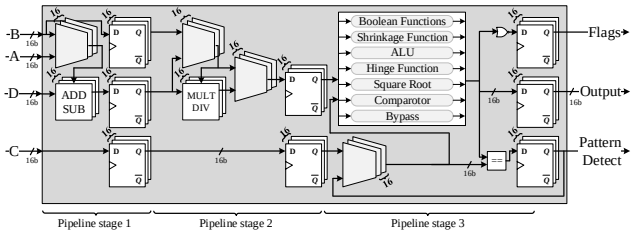


Fig 7: Programmable DSP Architecture showing a 3-stage pipeline

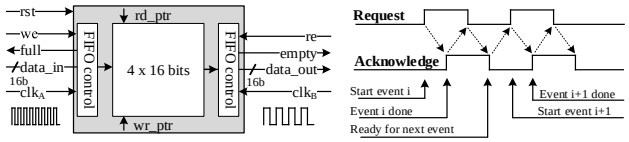


Fig 8: FIFO Architecture and the corresponding timing diagram

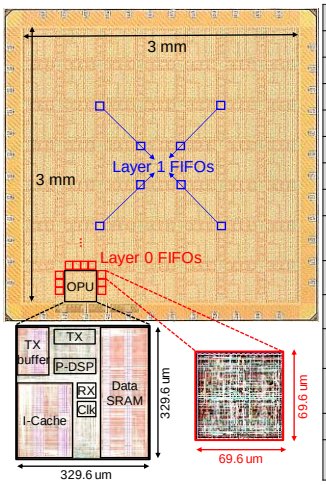
summarizes: (1) the number of instructions supported by each module, (2) the commonly-used macro functions and the number of instructions per function, and (3) their usage in the six template algorithms.

C. *Programmable DSP Architecture*

Fig 7. illustrates the principal components of the P-DSP, which consists of three pipeline stages in an architecture designed to maximize energy-efficiency. The first two stages support add/subtract and multiply/divide and the final stage supports a class of fixed-function blocks as shown in Fig. 7. Instruction level control of the pipeline and variable latency through the P-DSP is maintained via a program counter. High level program-code and data-sets are translated to instruction and data, scanned in to the chip and then executed. Convergence is declared either (1) after a fixed number of iterations, or (2) when the maximum cycle-to-cycle change of data in an OPU falls below a threshold. The clock-crossing FIFO features a 64b buffer and operates on a 4-phase handshaking scheme (Fig. 8). The timing diagram for executing ADMM is shown in Fig. 9. The system is clocked either by (1) a single global clock (synchronous) or (2) DCO based clock per-OPU enabling asynchronous/mesochorous links.

D. *Die micrograph and chip characteristics*

The test-chip is fabricated in TSMC 65nm GP CMOS process and occupies a total area of 12mm^2. It features 306.25KB of on-die memory distributed across 49-cores. The die micrograph and chip characteristics are shown in Fig. 10.



Technology	TSMC 65nm GP 1P9M
Chip Size	3.41 mm x 3.41 mm
Core Area	3 mm x 3 mm
Package	QFN6x6-48
Pin Count	48
Gate Count (logic only)	2725 kGates (NAND2)
On-Chip SRAM	306.25 KB
Number of OPUs	49
No. of pipeline stages in programmable DSP	3
Core Supply Voltage	0.5-1.2 V
IO Supply Voltage	2.5 V
Clock Rate	10-270 MHz
Network	Asynchronous & Mesochronous
Peak Energy Efficiency	279 GOPS/W
Arithmetic Precision	16-bit fixed-point

Fig 10: Die shot and chip characteristics

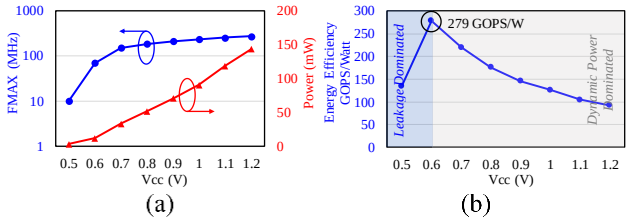


Fig 11: Measured (a) Power-performance trade-off (b) Energy-efficiency as a function of the supply voltage

III. Measurement Results

Fig. 11 illustrates the measured power-performance trade-off showing a peak F_{MAX} (in a synchronous setting) of 270 MHz (at 0.5V) and operation down in 0.5V (with FMAX= 10 MHz). Energy-efficiency (considering both dynamic and leakage power) shows a peak of 0.279TOPS/W at 0.6V, below which the design is leakage-dominated owing to the large (306.25 KB) on-die SRAM. It should also be noted that an operation here represents execution of a single pipeline-stage of the P-DSP and is computationally more demanding than MAC operations native to NN accelerators. Per-OPU DCO-based clocking reduces the overhead of routing a global clock and enables 2.7%-7.75% power savings compared to global clocking at *iso-performance*. The power-breakdown among computation, communication and storage at 0.6 V and 1.0 V is shown in Fig. 12. We use the hardware prototype to execute template algorithms across multiple data-sets and plot the time-to-

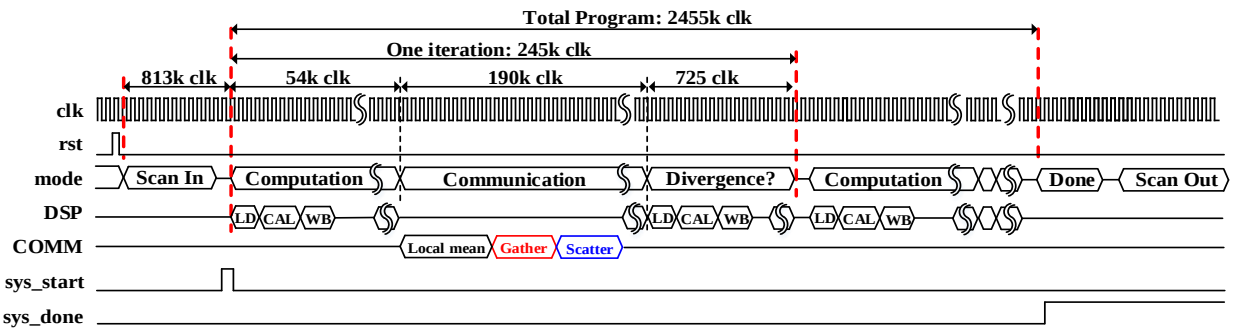


Fig 9: Timing diagram of showing the various steps of a template optimization problem

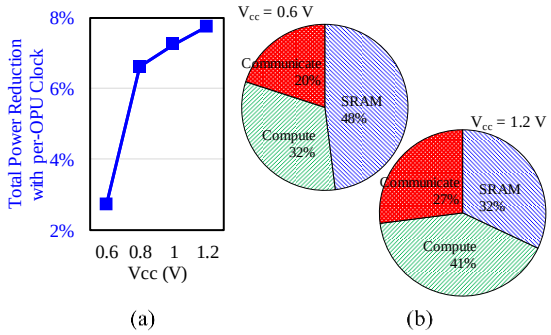


Fig 12: Measured (a) Power reduction via per-OPU clocking (b) Break-down of power consumption

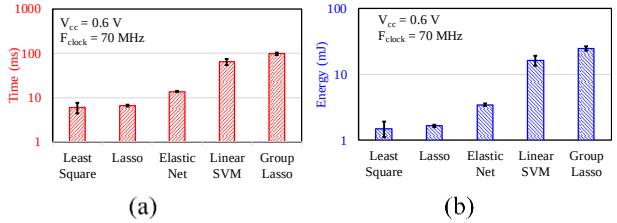


Fig 13: Measured algorithm-level benchmarking showing the time to compute and energy for six template algorithms. The errors bars show different problem instances that were characterized.

compute and energy at 0.6 V (Fig. 13). Although we demonstrate the capability of this near-memory spatial-architecture in solving distributed optimizations, the proposed hardware and programming model can also support a variety of other array processing tasks including inference in deep and convolutional neural networks.

IV. Applications

The programmable and iterative optimization solver is capable of addressing multiple applications. MRI image reconstruction from non-uniformly sampled data-points is computationally challenging and requires patients to lie in the machine for a long time. Our solution uses iterative least-squares optimization (Fig. 14 (a)) to reconstruct MRI images with high PSNR in less than 8ms. Similarly binary SVMs (Fig. 14 (b)), a popular choice in ML classification problems shows convergence with increasing number of iterations (multi-class SVM records 91% accuracy on the MNIST data-set, which is the state-of-the-art). Further, feature extraction with LASSO (L_1 regularization) used in ML, is shown in Fig. 14 (c). In Table I, a comparison with the state-of-the-art shows a (1) a highly-programmable, iterative optimization solver with peak efficiency of 0.279TOPS/W (2) a hierarchical multi-cast

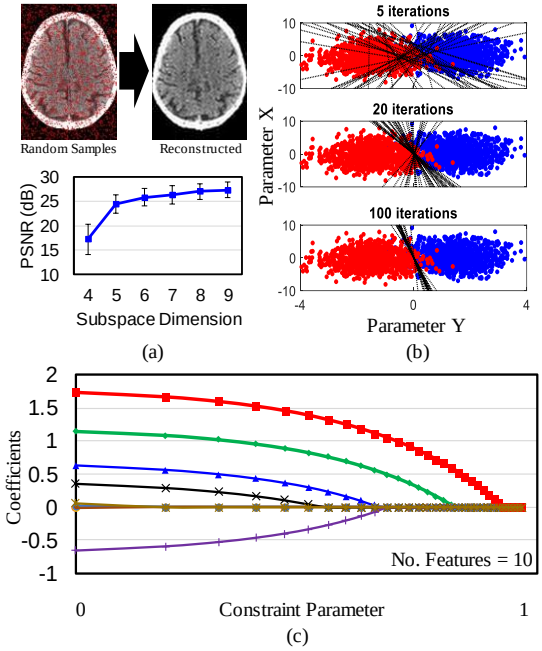


Fig 14: Application of OPTIMO in (a) MRI image reconstruction (b) Binary SVM (c) Lasso feature extraction for sample problems.

network for program-specific data-movement and (3) competitive energy-efficiency and voltage-scalability.

V. Conclusions

We present a 49-core fully-programmable spatial array processor for solving distributed optimizations with support for a large class of algorithms and applications. We note a peak performance of 270MHz and peak energy-efficiency of 0.279 TOPS/W.

Acknowledgements

This work was supported in part by C-BRIC, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, and in part by the NSF under grant 1640081, and the Nanoelectronics Research Corporation (NERC), a wholly-owned subsidiary of the SRC, through Extremely Energy Efficient Collective Electronics (EXCEL), an SRC-NRI Nanoelectronics Research Initiative under Research Task ID 2698.002.

References

[1] S. Boyd, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers Foundations and Trends® in Machine Learning", 3(1), 1–122. https://doi.org/10.1561/2200000016

[2] Y. Chen, et. al,, JSSC, vol. 52, no. 1, pp. 127-138, Jan. 2017.

[3] B. Moons et. al., ISSCC, 2017, pp. 246-247.

[4] J. Sim et. al., ISSCC, 2016, pp. 264-265.

[5] S. Choi et. al., ISSCC, 2018, pp. 220-222.

[6] T. Chen et. al., ISSCC, 2018, pp. 226-228

	This work	[6]	[3]	[4]	[5]	[2]
Application	Distributed Optimization	ECG Signal Reconstruction	CNN Inference	DNN Inference	CNN Inference	CNN Inference
Optimization algorithm	ADMM implementation	subspace pursuit	none	none	none	none
Technology	65nm	40nm	180nm	65nm	65nm	65nm
Area	12mm^2	3.06mm^2	3.3mm^2	16mm^2	16mm^2	16mm^2
On-die SRAM	306.25 KB	192KB	144 KB	36 KB	490.5 KB	181.5 KB
Programming support	yes	fixed function	fixed function	fixed function	fixed function	fixed function
On chip network	8 neighbors with hierarchical	not reported	systolic (4 neighbor)	not reported	systolic (4 neighbor)	systolic (6 neighbor)
Resolution	16b	32b	4b-16b	16b	16b	16b
Power	3.63 - 143.2 mW	21.8 - 93 mW	7.5-300 mW	45 mW	6.57 mW	278 mW
Frequency	10 - 270 MHz	67.5 MHz	200 MHz	125 MHz	10 - 100 MHz	200 MHz
Supply voltage	0.5-1V	0.9V	1V	1.2V	0.7-1.2V	0.82-1.17V
Performance/Watt	0.279 TOPS/W	21.5 MOPS/W	0.26-10TOPS/W	1.42TOPS/W	11.8 - 19.7 GOPS	0.21TOPS/W

Table I: Comparison of the proposed array-processor with competitive spatial-array processors. The proposed design addresses distributed optimization which presents a more complex data-flow and compute than traditional CNN and DNN inference architectures.