Energy-Performance Design Exploration of a Low-Power Microprogrammed Deep-Learning Accelerator

Giulia Santoro, Mario R. Casu, Valentino Peluso, Andrea Calimera, Massimo Alioto[†]
Politecnico di Torino, 10129 Turin, IT [†]National University of Singapore, 119077 Singapore, SG

***Abstract*—This paper presents the design space exploration of a novel microprogrammable accelerator in which PEs are connected with a Network-on-Chip and benefit from low-power features enabled through a practical implementation of a Dual-V_{dd} assignment scheme. An analytical model, fitted with post-layout data obtained with a 28nm FDSOI design kit, returns implementations with optimal energy-performance tradeoff by taking into consideration all the key design-space variables. The obtained Pareto analysis helps us infer optimization rules aimed at improving quality of design.**

I. Introduction

Deep Learning (DL) is a brain-inspired computational model that implements data reasoning through multiple layers of abstraction [1]. The DL paradigm, *Deep Convolutional Networks* in particular, evolved from a theoretical model to a practical solution with the introduction of GPUs. However, dedicated HW accelerators are more suited for green-computing and embedded applications. Prior works explored implementations on different technologies. Those based on FPGAs achieve high efficiency while guaranteeing reconfigurability and fast prototyping; results collected in [2] show a performance upper bound of 50 GOPS/W, 100x more than GPUs. Those based on custom ASICs further improve energy efficiency at the cost of flexibility and design complexity; the custom architecture proposed in [3] shows an energy efficiency of 800 GOPS/W. The latest in state-of-art consists of Application-Specific Processors (ASIPs) which combine the energy efficiency of ASICs with the flexibility of programmable arrays; the architectural solution proposed in [4] reaches performance levels up to 200 GOPS/W. The right choice depends on the field of application and the resources available, but even when restricting the range to a single technology domain, the huge number of design variables make it difficult for designers to identify the set of optimal solutions. The same deep-learning kernels can become computation- or memory-bounded by simply changing some of the architecture variables or just tuning the operating conditions, like the supply voltage and the associated clock frequency. Here is why standard architectural-level synthesis flows, built to accommodate the specifications of common data-path architectures, should be integrated with proper models that take into account the complexity of DL accelerators while exploring the interdependence among the many design variables.

Within this context, the main contributions of this work are: (i) introduce the design of a novel DL accelerator, called *LoC-1*, consisting of a microprogrammable architecture with all the essential ingredients to meet performance and energy efficiency; (ii) introduce an analytical model that brings under the same roof all the key architectural parameters; (iii) provide a Pareto analysis of the *Energy-Performance* space.

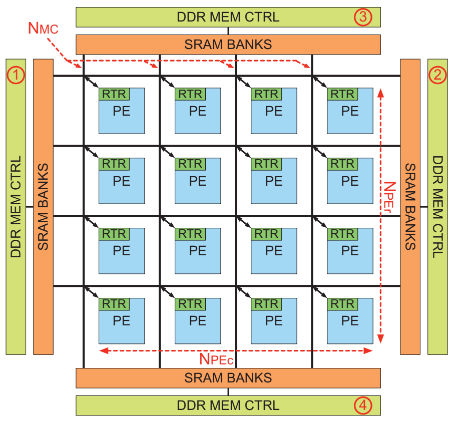

Figure 1. *Loc-1* Architecture with a 4×4 Processing Array.

II. Previous Works

Prior works addressed design space exploration for deep-learning accelerators. Some focused on the computational side of the problem overlooking the limits imposed by memory bandwidth [5], [6] Some others addressed this issue but only for a specific case of convolutional networks with ad-hoc optimizations [7], [8]. All targeted FPGA implementations. The proposed study covers a larger spectrum of implementations as we consider many degrees of flexibility. A different class of works, e.g. [9], do focus on techniques that accelerate the design space exploration (which is not the scope of this work).

III. The *LoC-1* Accelerator

A. Processing Array

The *LoC-1* is a $N_{PEr} \times N_{PEc}$ array of PEs connected through a standard wormhole-switching Network-on-Chip (NoC) [10] and interfaced with the external DRAM through N_{MC} DDR channels; N_{PEr}, N_{PEc}, N_{MC} and the NoC parallelism are design variables. An additional level of memory hierarchy is represented by SRAM banks bridging the NoC with the off-chip DRAM. The presence of these SRAM banks and their size is another design variable. Fig. 1 shows an example of a 4×4 architecture with 4 memory channels and SRAM banks.

B. Processing Element (PE)

The PE is the basic block of the accelerator. Its programmable architecture supports all the main operations found in DL algorithms and can be dynamically resized to match different input feature-map and kernel size. A PE consists of a micro-programmed SIMD-like (Single Instruction Multiple Data) unit. It has a control unit that orchestrates the flow of instructions executed by four parallel lanes. The instruction set includes: convolution, maxpooling, ReLU. Each lane has a local memory

Table I
VOLTAGE-FREQUENCY (VF) POINTS

low-V_{dd} high-V_{dd}	0.6-0.7				0.7-0.8				0.8-0.9				0.9-1.0			
Norm. F.	1.00	1.12	1.22	1.33	1.43	1.53	1.63	1.74	1.84	1.95	2.05	2.15	2.24	2.33	2.41	2.49
VF-index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

and the execution unit. The execution unit (EU) contains a Multiply-and-Accumulate (MAC) unit whose adder can also be used independently. The the local memory is organized as follows:

• **Input Buffer (IB):** made up of two banks, a private (PVT) and a shared (SHD) bank. PVT is accessible only locally; SHD can be accessed by other lanes too.
• **Scratchpad Memory (SM):** used for intermediate results.
• **Output Buffer (OB):** it has a small capacity and it is used to buffer data bound to other PEs and/or memory.

The lanes can work independently or in groups. The latter mode enables operations on large data that cannot be handled by a single lane. For instance, convolutions with high dimension kernels are split in smaller chunks assigned to different lanes; each lane computes a partial sum that is then summed with the results coming from other lanes. This mechanism saves data movement from/to memory.

In order to guarantee a high throughput, the IB of each lane should be ceaseless fed with fresh data. However, during data refill, the EU may stall. To reduce idleness, the IB can be programmed to work as a double buffer.

The PE hosts the NoC router that dispatches data/instructions received from the network to the IBs/control unit; output data are sent to the NoC through the same router. The PE and the NoC are developed at register-transfer level and then mapped onto a CMOS 28-nm FDSOI technology; performance and energy characterization is done at a post-routing stage.

C. Low-Power Design

Previous works, e.g., [4], already demonstrated voltage-frequency scaling is a desirable option for DL accelerators. This work improves over prior art through a fine-grain dual-V_{dd} strategy [11]. The latter leverages a finer spatial V_{dd} assignment; within each PE, different layout regions (*tiles*) are supplied with two voltages, low-V_{dd} and high-V_{dd} corresponding to the two boundaries of a supply voltage interval. We considered the V_{dd} range [0.6V-1.0V] with a 100-mV step, i.e., four V_{dd} intervals. Each tile is then assigned to a V_{dd} s.t. the f_{clk} constraint is met and the number of tiles at low-V_{dd} is maximized. For each of the four 100-mV voltage intervals, four different f_{clk} are available – sixteen voltage-frequency (VF) points in total (Table I).

IV. DEFINING THE DESIGN SPACE

The *Loc-1* architecture works as a template to generate instances with different throughput and energy consumption. The design space is obtained by the cartesian product of the sets of all the design variables involved. Feasible solutions in the design space must respect area and memory constraints. The Pareto analysis returns solutions which are not dominated in terms of *energy efficiency* [GOPS/W] or *throughput* [GOPS].

A. Design Variables and Constraints

• $\mathbf{N_{PE} = N_{PEr} \times N_{PEc}}$: number of PEs organized in an array to match the layout of an NoC mesh. The number of rows and columns ($\mathbf{N_{PEr}}$ and $\mathbf{N_{PEr}}$) are independent. The maximum $\mathbf{N_{PE}}$ depends on the size of the target die. It ranges from 20 for a 2-mm^2 die, up to 100 for a 10-mm^2 die.
• $\mathbf{N_{MC}}$: number of DRAM memory controllers and channels, from 1 to 4. More channels imply more memory bandwidth but also more power spent in the external DRAM.
• $\mathbf{S_{\%}}$: the size of the SRAM buffers as a percentage of the maximum die area, from 0 to 80% in 10% steps. This SRAM lies between the NoC periphery and the memory controllers, hence it is partitioned in $\mathbf{N_{MC}}$ blocks. Each block is banked to guarantee the same access bandwidth of the NoC. Each bank size is a function of $\mathbf{S_{\%}}$, $\mathbf{N_{MC}}$, and the die area.
• $\mathbf{B_{NoC}}$: the NoC channels bit-width, from 32 to 256 bits.
• $\mathbf{M_{IB}}$: private part of the input buffer in each lane, from 32 to 256 words (the shared part is 1/4 of the private one).
• $\mathbf{IF_{IB}}$: the portion of the input buffer that stores the Input Features working set: 8, 16, or 32 words. This portion and the weights portion (possibly doubled for double-buffer use) sum up to M_{IB} words (up to 256 in total).
• $\mathbf{DB_{IF}}$: a Boolean variable used to enable the partitioning of the input-buffer memory dedicated to Input Features in two buffers for double-buffering.
• $\mathbf{DB_{W}}$: a Boolean variable used to enable double buffering on the part of input-buffer memory dedicated to Weights.
• $\mathbf{VF}$: a number from 1 to 16 indicating the voltage/frequency pair to be chosen as the preferred operating pair for the design instance (currently 16 pairs as reported in Table I).

Feasible solutions must respect the three following constraints:
• $\mathbf{A_{MAX}}$: maximum die area; the sum of the logic and the SRAM area cannot exceed this bound.
• $\mathbf{BW_{DRAM}}$: maximum bandwidth of the DRAM; the accesses per unit time to the DDR channels multiplied by the DDR channel bit-width cannot exceed this bound.
• $\mathbf{BW_{SRAM}}$: maximum bandwidth of the SRAM; similar limit as for the DRAM.

B. Analytical Model

1) Workload Model: The computation of CONV and FC layers is modeled following the approach of standard DL frameworks like Caffe [12], which consists in transforming the related operations in matrix multiplications. Standard methods for matrix block partitioning for parallel processing are used. These methods allow us to model computation in each PE as a sequence of block data fetching from memory (first weights, then input features that reuse those weights several times) followed by multiplications and additions (row-column dot products). We do not model POOL layers because they do not represent a computation bottleneck.

2) Throughput Model: The throughput can be either limited by the computation capacity, the memory bandwidth (DRAM or

SRAM), or the NoC bandwidth. The proposed model identifies the bottleneck and computes the throughput accordingly.

The system throughput Th_S is the product of the number of active PEs and the throughput of the PE, $Th_S = N_{PE}Th_{PE}$. The Th_{PE} is the product of the number of SIMD lanes N_L (four in the *Loc-1* architecture) and the throughput $Th_L$of a single lane, $Th_{PE} = N_L Th_L$. Th_L is the number of basic operations performed by the lane during an *Initiation Interval*. In the following we measure the latter in number of clock cycles, therefore the actual Th_L [GOPS] is obtained by multiplying by the f_{clk} value.

The computation consists of two repeated nested cycles: in the outer cycle weights W are loaded from the DRAM and stored in the lane input buffer IB; in the inner cycle, input features IF are streamed N_W times (weights reuse factor) from either the DRAM (first access) or the SRAM (subsequent accesses, for the fraction that fits in the SRAM banks) to the input buffer. The (W, IF) pair are read from the IB and sent to the lane execution unit EU. The outer cycle has initiation interval II_W, whereas the inner cycle has initiation interval II_{IF}. If there are no stalls due to either the NoC bandwidth or the unavailability of a double buffer (variable $\mathbf{DB_W} = 0$), the outer cycle initiation interval corresponds to an integer number of times (N_W) the initiation interval of the inner cycle. The three conditions are expressed as follows:

$$II_W = \max\left(L_W, N_W II_{IF}, \frac{L_W + N_W II_{IF}}{1 + DB_W}\right) \quad (1)$$

where L_W is the communication latency of weights, which depends on the bandwidth of either the PE input system, the NoC, or the DRAM as follows:

$$L_W = \frac{W D_W N_L}{\min(BW_{PE}, BW_{NoC}, BW_{DRAM})} \quad (2)$$

with D_W the bit-width parallelism of the W weights loaded in the input buffer. Like II_W, the expression of II_{IF} takes into account possible stalls due to communication latency and the possibility that data are immediately available in the local memory, which is simply the time needed to stream the (W,IF) pair to the EU:

$$II_{IF} = \max(L_{IF}, L_C, \frac{L_{IF} + L_C}{1 + DB_{IF}}) \quad (3)$$

In (3) L_{IF} is the communication latency of weights, which is similar to (2) but takes into account the possibility of streaming IFs from the SRAM other than the DRAM, and L_C is the computation latency, which is basically $L_C = IF + L_{ovh}$, where L_{ovh} is a clock-cycles overhead due to control logic (one cycle in our implementation). The throughput of the lane is finally given by:

$$Th_L = (N_W N_{OPS} F_{CK})/II_W, \quad (4)$$

where $N_W N_{OPS} = N_W(N_{MUL} + N_{ADD}) = 2N_W IF$ is the number of operations executed by one lane in II_W.

In a system where the computation latency is the dominant term, $II_W \simeq N_W II_{IF}$ and therefore $Th_L \simeq (N_{OPS}F_{CK})/II_{IF}$, which further simplifies to $Th_L \simeq 2F_{CK}$ (i.e. two operations per clock cycle, one multiplication and one addition).

3) Energy Model: The model integrates the energy spent (i) to access DRAM and SRAM, (ii) to transfer data through the NoC, (iii) by the PE array. The efficiency [GOPS/W] is the ratio between the number of computations performed and the sum of the three energy contributions, which we briefly analyze in the following.
DRAM: The energy consumed during DRAM access is:

$$E_{DRAM} = P_{DRAM} T_{EXE} = P_{DRAM} NOPS/Th_S \quad (5)$$

P_{DRAM} is the DRAM power consumption and is proportional to the number of channels N_{MC} and the actual bandwidth (BW_{DRAM}/N_{MC} =96 Gbps); P_{DRAM} is calculated through a regression model we built using characterization data taken from the datasheet of a commercial low-power DDR memory chip (Micron Technology Inc. LPDDR4X, Z00M 4-Gb die).
NoC and PE: power and energy consumption are collected from post-layout simulations. The Noc and PE are designed with a commercial CMOS 28nm FDSOI design kit. Simulations under realistic workload are used to extract the internal signal statistics which are then back-annotated in a signoff static-timing/power engine for power estimation.

V. Pareto Analysis

The analytical model coded in Matlab is used for an exhaustive design-space exploration. The feasible solutions are projected into the sub-space defined by throughput and energy efficiency. Finally, dominant points are identified and the pareto curves drawn. We chose AlexNET as representative workload.
Concerning the 2-mm^2 case, the number of resulting implementations is 4.1 millions and the CPU time is 75 minutes on an 8-core Intel Core i7-6700, 3.40GHz, 8-MB cache. When considering the 10-mm^2, we constrained the exploration by down-sampling the SRAM percentage $S_{\%}$ values (from 0 to 80% in 20% steps), the NoC bit-width values (from 64 to 256 in three power-of-two steps), and the aspect-ratio of the NoC mesh (N_{PEc} cannot be less than 5 times N_{PEr}). With such restriction, we obtain enough points to build statistically significant Pareto curves; the CPU time is 135 minutes.
Figs. 2(a)-(b) show the results of an exploration for $A_{MAX} = 2\,\text{mm}^2$. In the top plot, the DRAM energy is omitted (as done for simplicity in most of the previous works), whereas the bottom one takes it into consideration. Each solution is associated with an n-tuple with the corresponding values of the n design variables. The Pareto frontier is labeled with a subset of the n-tuple: the triple size of the PE array (N_{PEr}, N_{PEc}), the total area, the SRAM percentage $S_{\%}$, and the voltage/frequency pair index VF (from 1 to 16).
At a first glance, the plots clearly highlight the huge impact of the external memory: the energy drained due to DRAM accesses decreases the efficiency by one order of magnitude. This is somewhat expected. A more accurate, yet interesting analysis concerns the shape of the Pareto curves: *Pareto-points substantially drift in the design space*. When the DRAM energy is ignored, Fig. 2(a), the most energy-efficient points do not use SRAM (i.e., $S_{\%} = 0$) and do not fully use the available area (i.e., $< 2\,\text{mm}^2$). More area and more SRAM tend to be eaten by Pareto points with increasing throughput. In Fig. 2(b), instead, the area is almost fully utilized and the SRAM uses between 10 and 30% of this area for all the Pareto-points. What

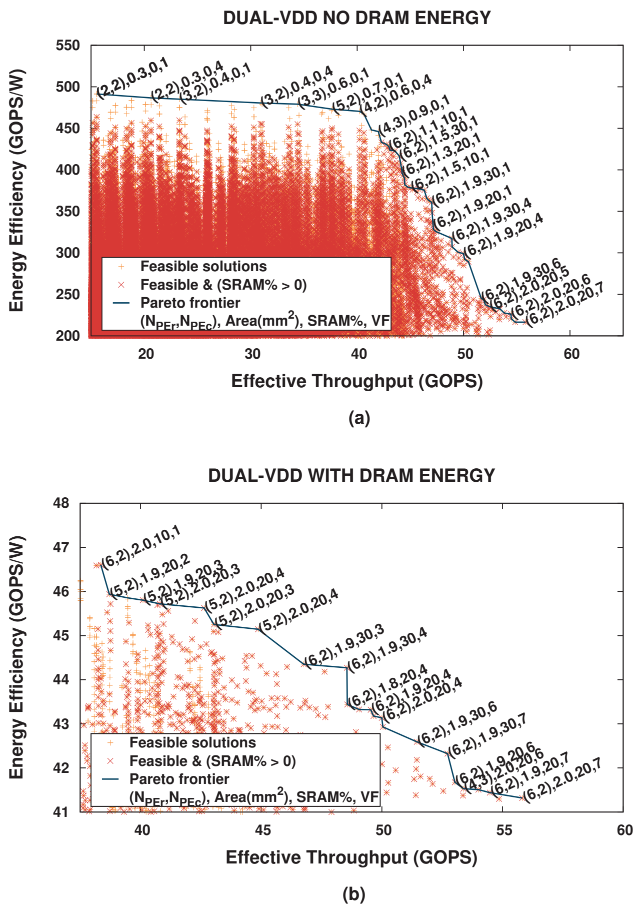

Figure 2. Feasible and Pareto solutions for $A_{MAX} = 2\,\text{mm}^2$, obtained (a) ignoring the DRAM energy and (b) including it.

is instead common to both cases is that the voltage-frequency pair gets larger with the throughput (i.e., the PEs work at higher frequency and thus higher voltage to achieve higher throughput), while it decreases with the energy efficiency. This confirms higher energy efficiency is obtained at lower voltages approaching the near-threshold region.

Concerning power-management, we notice that the points obtained with Dual-V_{dd} dominate the Single-V_{dd} ones for increasing values of the throughput and of the VF code, whereas at high energy efficiency (and so at a very low voltage) the difference is negligible.

For $A_{MAX} = 10\,\text{mm}^2$ similar comments do hold, in particular those concerning the impact of the DRAM. The collected results reveals that the most important difference is the higher value of $S_{\%}$ in the Pareto solutions, which ranges between 40 and 80% in the $10 mm^2$ case. That means a large portion of the layout is taken by the SRAM; indeed, investing area for additional PEs would not be an efficient choice: at some point the throughput gets limited by the memory bandwidth, regardless the number of PEs. Only at the highest throughput some area can be fruitfully used for more SRAM buffering, which has the effect of decreasing accesses to the DRAM, and therefore, of saving more energy.

For what concerns the other design variables listed in Sec. IV, the following trends can be inferred:

- The Pareto solutions tend to maximize the number of memory controllers and so the memory bandwidth. In our experiments we rarely noticed optimal solutions with $N_{MC} < 4$.
- The input buffers are larger for the solutions obtained with a $2\,\text{mm}^2$ area: 128 and 256 words compared to 64 for the $10\ mm^2$ case. The larger buffers are used by solutions with higher energy efficiency, whereas the solutions with higher throughput use smaller buffers.
- All the Pareto solutions use a double buffer for the IFs (DB_{IF}=1) and do not use it for the weights (DB_W=0). The reason is that using a double buffer for weights would improve only marginally the throughput of a lane, while significantly reducing the memory space for storing IFs.

The above rules can improve the quality of design and speed-up convergence to optimal solutions.

VI. Summary

This paper introduced an analytical model through which we quantified the figures of merit of a novel deep-learning accelerator. The latter, called *Loc-1*, is an architectural template with all the key characteristics that most of the existing implementations have in common.

VII. Acknowledgment

This work was supported by Compagnia di San Paolo, MACLoC project, 2016 Call for Joint Projects for the Internationalization of Research.

References

[1] Y. LeCun *et al.*, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[2] V. Gokhale *et al.*, “A 240 g-ops/s mobile coprocessor for deep neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 682–687.
[3] L. Cavigelli and L. Benini, “A 803 gop/s/w convolutional network accelerator,” *IEEE Trans. Circuits Syst. Video Technol.*, 2016.
[4] Y.-H. Chen *et al.*, “Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks,” *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2017.
[5] A. Dundar *et al.*, “Accelerating deep neural networks on mobile processor with embedded programmable logic,” in *Neural information processing systems conference (NIPS)*, 2013.
[6] M. Sankaradas *et al.*, “A massively parallel coprocessor for convolutional neural networks,” in *2009 IEEE 20th Int. Conf. on Application-specific Systems, Architectures and Processors (ASAP)*. IEEE, 2009, pp. 53–60.
[7] M. Peemen *et al.*, “Memory-centric accelerator design for convolutional neural networks,” in *2013 IEEE 31st Int. Conf. on Computer Design (ICCD)*. IEEE, 2013, pp. 13–19.
[8] C. Zhang *et al.*, “Optimizing fpga-based accelerator design for deep convolutional neural networks,” in *Proc. 2015 ACM/SIGDA Int. Symp. on Field-Programmable Gate Arrays*. ACM, 2015, pp. 161–170.
[9] B. Reagen *et al.*, “A case for efficient accelerator design space exploration via bayesian optimization,” in *2017 IEEE/ACM Int. Symp. on Low Power Electronics and Design (ISLPED)*. IEEE, 2017, pp. 1–6.
[10] D. Bertozzi *et al.*, “Noc synthesis flow for customized domain specific multiprocessor systems-on-chip,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 2, pp. 113–129, 2005.
[11] V. Peluso *et al.*, “Ultra-fine grain vdd-hopping for energy-efficient multi-processor socs,” in *IFIP/IEEE Int. Conf. on Very Large Scale Integration (VLSI-SoC)*, 2016, pp. 1–6.
[12] Y. Jia *et al.*, “Caffe: Convolutional architecture for fast feature embedding,” *arXiv preprint arXiv:1408.5093*, 2014.