HAL
open science

A Heterogeneous Cluster with Reconfigurable Accelerator for Energy Efficient Near-Sensor Data Analytics

Satyajit Das, Kevin Martin, Philippe Coussy, Davide Rossi

To cite this version:

Satyajit Das, Kevin Martin, Philippe Coussy, Davide Rossi. A Heterogeneous Cluster with Reconfigurable Accelerator for Energy Efficient Near-Sensor Data Analytics. International Symposium on Circuits and Systems (ISCAS), May 2018, Florence, Italy. hal-01827425

HAL Id: hal-01827425

https://hal.science/hal-01827425v1

Submitted on 2 Jul 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Heterogeneous Cluster with Reconfigurable Accelerator for Energy Efficient Near-Sensor Data Analytics

Satyajit Das*†, Kevin J. M. Martin*, Philippe Coussy*, and Davide Rossi†‡

*Univ. Bretagne-Sud, UMR 6285, Lab-STICC, F-56100 Lorient, France, [firstname].[lastname]@univ-ubs.fr,
†DEI, University of Bologna, Italy, [firstname].[lastname]@unibo.it
‡Integrated Systems Laboratory, ETH Zurich, Switzerland, [first-initial][last name]@iis.ee.ethz.ch

***Abstract*—IoT end-nodes require high performance and extreme energy efficiency to cope with complex near-sensor data analytics algorithms. Processing on multiple programmable processors operating in near-threshold is emerging as a promising solution to exploit the energy boost given by low-voltage operation, while recovering the related frequency degradation with parallelism. In this work, we present a heterogeneous cluster architecture extending a traditional parallel processor cluster with a reconfigurable Integrated Programmable Array (IPA) accelerator. While programmable processors guarantee programming legacy to easily manage peripherals, radio software stacks as well as the global program flow, offloading data-intensive and control-intensive kernels to the IPA leads to much higher system level performance and energy-efficiency. Experimental results show that the proposed heterogeneous cluster outperforms an 8-core homogeneous architecture by up to 4.8x in performance and 4.5x in energy efficiency when executing a mix of control-intensive and data-intensive kernels typical of near-sensor data analytics applications.**

I. Introduction

High performance and extreme energy efficiency are strict requirements for many deeply embedded near-sensor processing applications such as wireless sensor networks, end-nodes of the Internet of Things (IoT) and wearables. One of the most traditional approaches to improve energy efficiency of deeply embedded computing systems is achieved exploiting architectural heterogeneity by coupling general-purpose processors with application- or domain-specific accelerators in a single computing fabric [1][11]. On the other hand, most recent ultra-low power designs exploit multiple homogeneous programmable processors operating in near-threshold [10]. Such an approach, which joins parallelism with low-voltage computing, is emerging as an attractive way to join performance scalability with high energy efficiency.

In this paper, we present a heterogeneous architecture which integrates a near-threshold tightly-coupled cluster of processors [10] augmented with the Integrated Programmable Array (IPA) presented in [3]. This approach joins the programming legacy of instruction processors with the flexible performance and efficiency boost of Coarse Grain Reconfigurable Arrays [4] (CGRA). A similar approach has been adopted in [5], which introduced an ultra-low power heterogeneous system featuring a Single Instruction Multiple Data (SIMD) CGRA as reconfigurable accelerator for bio-signal analysis. With respect to this domain-specific architecture, where the computational kernels must be mapped manually on the CGRA, the system proposed in our work is meant for general-purpose near-sensor data analytics, also relying on an automated compilation flow that allows to generate the configuration bitstream for the CGRA starting from a general-purpose ANSI-C code [2].

We synthesized the architecture in a 28nm FD-SOI technology, and we carried out a quantitative exploration combining physical synthesis results (i.e. frequency, area, power) and benchmarking of a set of signal processing kernels typical of end-nodes IoT applications. Two interesting findings of our exploration show that (1) the performance of the IPA is much less sensitive to memory bandwidth than parallel processor clusters and that (2) the simpler nature of its architecture allows the IPA to run twice as fast as the rest of the system. Exploiting these two features of our architecture, we show that the heterogeneous cluster achieves significant performance and energy improvement for both compute and control intensive benchmarks with respect to the 8 core homogeneous cluster, achieving up to 4.8× speed-up (with a minimum of 1× and an average of 1.79×) and up to 4.4× (with a minimum of 1× and an average of 2.24×) better energy efficiency.

II. Background

This section presents the background technology used to design the heterogeneous reconfigurable cluster described in this work.

A. PULP Cluster Architecture

The PULP cluster features 8 32-bit RISC-V cores based on a four pipeline stages micro-architecture optimized for energy-efficient operation [6] sharing a 64KB multi-banked scratchpad memory through a low-latency interconnect [8]. The ISA of the cores is extended with instructions targeting energy efficient digital signal processing such as hardware loops, load/store with pre/post increment, SIMD operations. The cores share a 4KB private instruction cache to boost performance and energy efficiency for tightly coupled clusters of processors typically relying on data parallel computational models [7]. Off-cluster data transfers are managed by a lightweight multi-channel DMA optimized for energy-efficient operation [9]. Both the (I$) and DMA are connected to an AXI4 cluster bus.

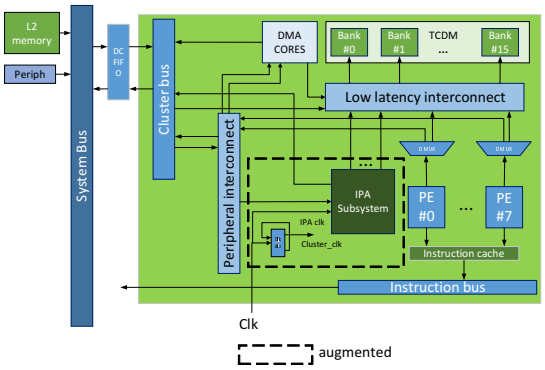

Fig. 1. PULP heterogeneous cluster augmented with IPA subsystem

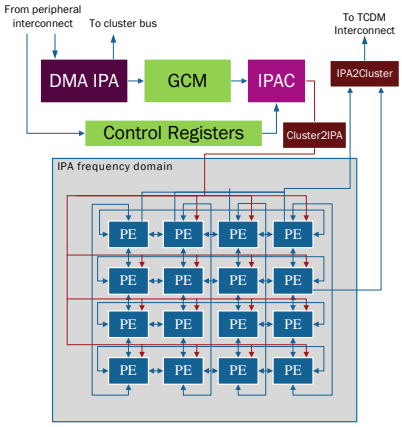

Fig. 2. Block diagram of the IPA subsystem.

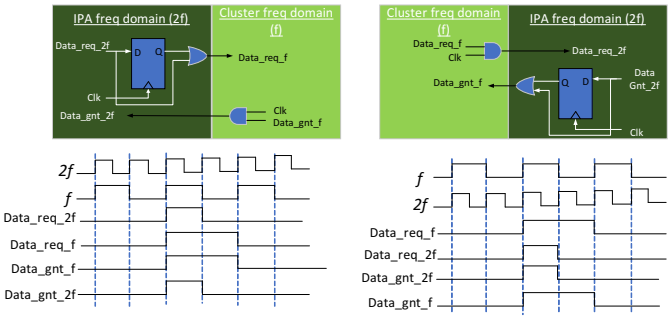

Fig. 3. Synchronous interface for reliable data transfer between the two clock domains.

A peripheral interconnect is used to communicate with on-cluster peripherals such as a timer, a hardware synchronizer and other memory mapped peripherals such as application-specific accelerators. To operate at the best operating point for a given workload the cluster can be integrated in an independent voltage and frequency domain, featuring dual-clock FIFOs and level shifters at its boundary.

B. IPA Architecture

The IPA consists of an array of 16 PEs communicating through a 2D torus interconnect [2]. Each PE can perform 32-bit ALU operations (both arithmetic and logical), 16-bit $\times$ 16-bit $\rightarrow$ 32-bit multiplications and control flow operations such as branches. The functional units of each PE features two input operands coming from the neighbouring PEs or the internal register files. The PEs also include an instruction register file which stores the program, a regular register file to store temporary variables and a constant register file to store immediates. To reduce dynamic power consumption in idle mode, each PE contains a tiny Power Management Unit (PMU) which clock gates the PEs when idle [3]. A parametric number of PEs can be augmented with a load-store unit employing the same request-grant protocol of the PULP low-latency interconnect [8], which allows to communicate with a multi-banked shared memory. The configuration of the array is generated automatically by a compilation flow which starts from a ANSI-C code and generates the configuration bitstream for the IPA [2].

III. Heterogeneous Cluster Architecture

In this work, the PULP cluster is extended with the Integrated Programmable Array accelerator, as shown in Fig. 1. Fig. 2 shows a detailed block diagram of the subsystem embedding the IPA array. The IPA array is configured through a global context memory (GCM), responsible for storing locally the configuration bitstream of the PEs. The GCM is connected through a DMA-capable AXI-4 port to the cluster bus, enabling pre-fetching of IPA contexts from L2 memory. The GCM is considered twice the size of the configuration bitstream of the IPA in the worst case. In this way, it is possible to employ a double-buffering mechanism and load a new bitstream from the L2 to the GCM when the current one is being loaded on the array, completely hiding time for reconfiguration. More details on the structure of the IPA array bitstream can be found in [3].

A set of memory mapped control registers allow to load a new context to the IPA array, trigger the execution of a kernel and synchronize with the other processors in the cluster.

As opposed to many CGRA architectures, the IPA can access a multi-banked shared memory through 8 master ports connected to the low-latency interconnect. This eases data sharing with the other processors of the cluster, following the computational model described in [1]. The optimal number of port has been chosen to optimize the trade-off between the size of the interconnect and the bandwidth requirements of the IPA. Following the analysis conducted in [2], which shows that the IPA can operate 2$\times$ faster than the processors, we have extended the architecture of the cluster in a way that the IPA can work at twice the frequency of rest of the cluster. This approach allows to operate each component in the cluster at the optimal frequency, without paying the overheads of dual-clock FIFOs, requiring a significant amount of logic and synchronization overhead. On the contrary, the hardware support for the dual-frequency mode includes a clock divider to generate the two different edge aligned clocks, and two modules needed to adapt the request-grant protocol of the low-latency interconnect [8] to deal with the frequency domain crossing, as shown in Fig. 3.

IV. Software Infrastructure

To offload jobs to the IPA and synchronize the execution, the cores access the *control registers* of the IPA, by memory mapped operations. The control registers are composed of a

TABLE I. LIST OF APIS FOR CONTROLLING IPA

Function	Description
void load_data_l2totcdm (int DMA_CORE_ID, int size, unsigned int l2_addr, unsigned int tcdm_addr)	Writes data from L2 memory to the TCDM banks through DMA_CORE
void load_context_l2togcm (int DMA_IPA_ID, int size, unsigned int l2_addr, unsigned int gcm_addr)	Writes context from L2 memory to the GCM through DMA_IPA
int ipa_start_execution ()	Initiate IPA execution by writing in the command register
void ipa_check_status(in id)	Core synchronization
void free_ipa (int id)	Release IPA

command register and a *status register*. We designed a simple Application Programming Interface (APIs) to perform the offload and synchronize tasks with the IPA. The main functions are described in Table I. Before execution starts in the IPA accelerator, the cores load the corresponding context and data from the L2 memory to the GCM and L1 memory, programming the system DMA and the IPA DMA, respectively. The context for the IPA consists of instructions and constants for the PEs, generated by the compilation flow proposed in [2]. The functions *load_data_l2totcdm* and *load_context_l2togcm* contain a set of routines to write data and context into TCDM and GCM respectively. The *ipa_start_execution* writes *execute* command into the *command register* of the IPA. The completion of the execution is notified by updating the *status register*. The core is synchronized with the IPA execution by calling the *ipa_check_status* function, which checks for the updates in the status register.

V. EXPERIMENTAL RESULTS

In this section we present the implementation results of the heterogeneous PULP cluster. The three possible modes considered in these comparisons are: (a) single-core: running applications in a single core, (b) ipa: running applications in the IPA where the core takes part in offloading only, (c) multi-core: running applications in parallel cores. All the benchmarks are coded in fully portable C, using the OpenMP programming model to express parallelism for PULP. In these benchmarks, matrix multiplication, convolution, FFT, FIR, separable filter, sobel filter are broadly used in near sensor image and multimedia applications. Sampling scheduler of the sensors strongly depends on computation of the greatest common divisor (GCD). Feature extraction in sensor networks widely uses CORDIC.

A. Implementation

The cluster consists of 8 cores featuring 4 kB of shared I$, one IPA with 16 PEs and a GCM of 4KB, while the TCDM is composed of 16 banks of 4 kB each, leading to an overall TCDM size of 64 kB. These architectural parameters were chosen to fit the constraints of the wide range of signal processing benchmarks presented in this paper. The SoC was synthesized with Synopsys Design Compiler 2013.12-SP3 on a STMicroelectronics 28nm UTBB FD-SOI technology library. Since, the achievable frequency of the PEs in the IPA is higher than the RI5KY cores used in the cluster, the IPA is clocked at 100 MHz, while the rest of the cluster runs at 50 MHz (in the SS, 0.6V, $-40°$C corner). Synopsys PrimePower 2013.12-SP3 was used for timing and power analysis at the supply voltage of 0.6V, $25°$C temperature, in typical process conditions. Table II presents the area information of the components in the cluster. Although, the total area of the IPA with 16 PEs is almost similar to the area of the 8 cores combined, the area occupied by the GCM is much less than the total cache memory, which in turn provides better area efficiency while running applications in IPA.

TABLE II. SYNTHESIZED AREA INFORMATION FOR THE PULP HETEROGENEOUS CLUSTER

Components	Area (μm^2)		% of cluster area
CORES		160,352	18
ICACHE		190,089	22
DMA_CORE		41,406	5
IPA	**Total**	156,323	18
	IPAC	861	
	GCM_INTCNCT	359	
	PE ARRAY	154,515	
	OTHERS	588	
DMA_IPA		32,636	4
GCM		18,704	2
TCDM		149,638	17
CLUSTER_INTCNCT		63,126	7
CLUSTER_PERIPHERALS		21,610	2
OTHERS		37,932	4
Total		871,816	100

B. Performance and Energy Efficiency

Table III reports the execution time in nano seconds for different benchmarks running on a single-core, on 8 cores and on the IPA. The IPA execution time includes the time taken for loading the context into the PEs. Comparing to the performance of execution in single-core, the accelerator achieves a maximum of $8\times$ (with a minimum of $2.49\times$ and an average of $5.4\times$) speed-up. The control intensive kernel like GCD does not exhibit parallelism, hence parallel software execution does improve performance of the homogeneous cluster. On the other hand, the execution on the IPA improves the performance by almost $5\times$, exploiting also instruction-level parallelism rather than data-level parallelism only. The performance gain in the accelerator for the compute intensive kernels like matrix multiplication, convolution, FIR and separable filters is limited if compared to the performance of parallel-cores. However, the relatively small performance gain compared to the parallel cluster is compensated by the gain in energy consumption (Table V) due to the simpler nature of the compute units of the IPA with respect to full processors, to the smaller number of power-hungry load/store operations (Table VI), and to the fine-grained power management architecture that allows clock gate the inactive PEs during execution (Table V).

Table IV presents the performance improvement of the IPA when moving from iso-frequency to the $2\times$ frequency domain execution in the IPA. This shows that, although there is a reduction of memory bandwidth (see loss due to additional stalls column in Table IV), since the TCDM operates at the same frequency as the rest of the cluster (i.e. half frequency w.r.t. the IPA array), an average of $1.82\times$ speed-up (with maximum of $1.92\times$ and a minimum of $1.73\times$) can be achieved with this dual-frequency cluster architecture.

The power consumption profiles for the different modes of execution presented in Fig. 4, which shows the percentage of contribution by the several components in the cluster.

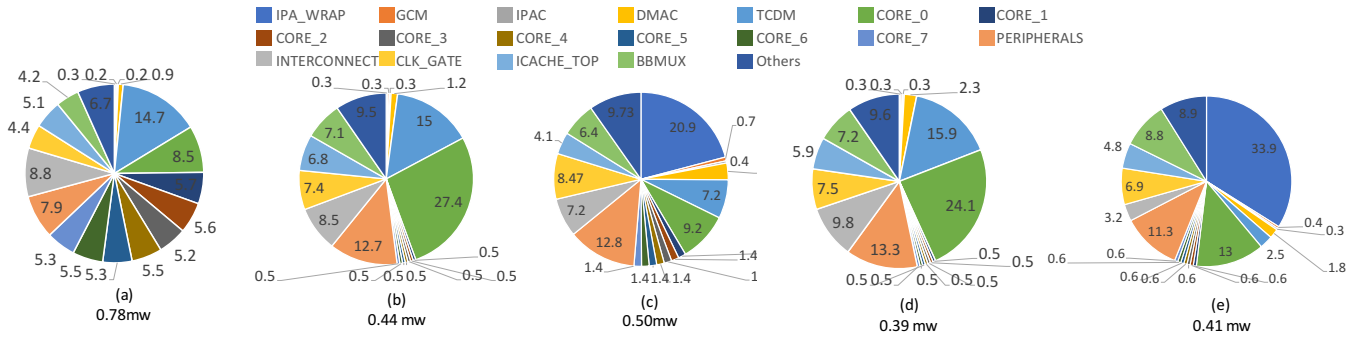

Fig. 4. Power consumption breakdown in percentage: Executing Matrix-Multiplication in (a) Multi-Core; (b) Single-Core; (c) IPA. Executing GCD in (d) Single-Core; (e) IPA.

TABLE III. PERFORMANCE EVALUATION IN EXECUTION TIME (NS) FOR DIFFERENT CONFIGURATION IN THE HETEROGENEOUS PLATFORM

Kernels	Data size (KB)	Single-core (ns)	Multi-core (ns)	Speed-up in multi-core (x)	IPA (ns)	Speed-up in IPA (x)
MatMul	8	3,358,740	435,180	7.72	432,630	7.76
Conv	8	9,733,380	1,520,840	6.4	1,494,860	6.51
FFT	1	767,640	142,720	5.38	94,510	8.12
FIR	0.84	182,500	33,460	5.45	33,410	5.46
Sep Filter	10	39,870,420	6,404,160	6.23	6,334,700	6.29
Sobel Filter	10	117,024,880	40,894,260	2.86	28,865,890	4.05
GCD	0.01	2,951,160	2,951,160	1	61,1300	4.83
Cordic	0.06	9,000	7,000	1.29	3,610	2.49
Manh Dist	8	244,640	164,640	1.49	70,300	3.48

TABLE IV. PERFORMANCE COMPARISON BETWEEN ISO-FREQUENCY AND 2× FREQUENCY EXECUTION IN IPA

Benchmarks	#cycles in iso frequency	#cycles in 2x frequency	Loss due to stalls	overall execution speed-up
MatMul	39,330	43,263	3,933	1.82
Convolution	130,896	149,486	18,590	1.75
FFT	8,182	9,451	1,269	1.73
FIR	3,122	3,341	219	1.87
Separable filter	575,882	633,470	57,588	1.82
Sobel Filter	2,634,172	2,886,589	252,417	1.83
GCD	58,573	61,130	2,557	1.92
Cordic	328	361	33	1.82
ManhDistance	6,391	7,030	639	1.82
Average				1.82

Fig. 4 (a), (b), (c) represents the power breakdown while executing matrix multiplication in multi-core, single-core and IPA respectively, representative for other compute intensive benchmarks. Similarly, in this figure, (d) and (e) present the profiles for executing GCD, a control intensive benchmark, in single-core and IPA respectively. In Fig. 4 (a), (b), (c), the TCDM contributes to 14.7%, 15% and 7.2% in the multi-core, single-core and IPA configurations, respectively. The reduced memory access in IPA execution helps to achieve better energy efficiency. While executing GCD in single-core and the IPA, the TCDM consumed around 15.9% and 2.5% of the total power in the two analysed configurations, respectively. The simpler nature of the compute units, low burden on the TCDM and data exchange through PEs explains the energy gain of 7× in the IPA execution.

TABLE V. ENERGY CONSUMPTION EVALUATION IN μ J FOR DIFFERENT CONFIGURATION IN THE HETEROGENEOUS PLATFORM

Kernels	Single-core	Multi-core	IPA	
			Energy	of Active PEs/cycle
MatMul	1.247	0.313	0.208	58.5
Convolution	2.876	1.095	0.658	59.2
FFT	0.292	0.087	0.042	59.7
FIR	0.08	0.026	0.026	46.1
Separable filter	16.663	4.611	4.28	55.5
Sobel Filter	51.491	29.444	12.701	51.2
GCD	1.151	1.151	0.257	6.25
Cordic	0.004	0.003	0.001	50
ManhDistance	0.1	0.095	0.03	48.5

TABLE VI. COMPARISON BETWEEN TOTAL NUMBER OF MEMORY OPERATIONS EXECUTED

Benchmarks	multi-core	single-core	IPA
MatMul	66,584	66,561	35,032
Convolution	135,280	135,114	75,600
FFT	12,528	11,733	6528
FIR	5,904	5,893	3,990
Separable filter	142,840	142,800	95,200
Sobel Filter	148,240	148,224	120,000
GCD	64,531	64,531	2
Cordic	32	28	15
ManhDistance	2,158	2,049	2,048

VI. CONCLUSION

In this paper, we present a novel approach towards heterogeneous computing, augmenting ultra-low power reconfigurable accelerator in the PULP multi-core cluster. The experiments integrating IPA in the PULP platform suggests that architectural heterogeneity is a powerful approach to improve energy profile of the computing systems. We have presented three possible executions of the benchmarks in the IPA integrated PULP platform. The heterogeneous cluster achieves achieving up to 4.8× speed-up and up to 4.4× better energy efficiency with respect to an 8-core homogeneous cluster.

ACKNOWLEDGEMENTS

This work was funded by the ERC MultiTherman Project (ERC-AdG-291125), and by the OPRECOMP Project funded from the European Unions Horizon 2020 Research and Innovation Programme under grant agreement No. 732631.

REFERENCES

[1] F. Conti, A. Marongiu, and L. Benini. Synthesis-friendly techniques for tightly-coupled integration of hardware accelerators into shared-memory multi-core clusters. In *Proceedings of the Ninth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, CODES+ISSS '13, pages 5:1–5:10, Piscataway, NJ, USA, 2013. IEEE Press.

[2] S. Das, K. J. M. Martin, P. Coussy, D. Rossi, and L. Benini. Efficient mapping of cdfg onto coarse-grained reconfigurable array architectures. In *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 127–132, Jan 2017.

[3] S. Das, D. Rossi, K. J. M. Martin, P. Coussy, and L. Benini. A 142MOPS/mW Integrated Programmable Array Accelerator for Smart Visual processing. In *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*, pages 1–4. IEEE, 2017.

[4] B. De Sutter, P. Raghavan, and A. Lambrechts. Coarse-grained reconfigurable array architectures. In S. S. Bhattacharyya, E. F. Deprettere, R. Leupers, and J. Takala, editors, *Handbook of Signal Processing Systems*, pages 449–484. Springer US, 2010.

[5] L. Duch, S. Basu, R. Braojos, G. Ansaloni, L. Pozzi, and D. Atienza. Heal-wear: An ultra-low power heterogeneous system for bio-signal analysis. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2017.

[6] M. Gautschi, P. D. Schiavone, A. Traber, I. Loi, A. Pullini, D. Rossi, E. Flamand, F. K. Gürkaynak, and L. Benini. Near-threshold risc-v core with dsp extensions for scalable iot endpoint devices. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, PP(99):1–14, 2017.

[7] I. Loi, A. Capotondi, D. Rossi, A. Marongiu, and L. Benini. The quest for energy-efficient i$ design in ultra-low-power clustered many-cores. *IEEE Transactions on Multi-Scale Computing Systems*, PP(99):1–1, 2017.

[8] A. Rahimi, I. Loi, M. R. Kakoee, and L. Benini. A fully-synthesizable single-cycle interconnection network for shared-l1 processor clusters. In *2011 Design, Automation & Test in Europe*, pages 1–6. IEEE, 2011.

[9] D. Rossi, I. Loi, G. Haugou, and L. Benini. Ultra-low-latency lightweight dma for tightly coupled multi-core clusters. In *Proceedings of the 11th ACM Conference on Computing Frontiers*, page 15. ACM, 2014.

[10] D. Rossi, A. Pullini, I. Loi, M. Gautschi, F. K. Gürkaynak, A. Teman, J. Constantin, A. Burg, I. Miro-Panades, E. Beignè, F. Clermidy, P. Flatresse, and L. Benini. Energy-efficient near-threshold parallel computing: The pulpv2 cluster. *IEEE Micro*, 37(5):20–31, September 2017.

[11] M. B. Taylor. Is dark silicon useful? harnessing the four horsemen of the coming dark silicon apocalypse. In *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, pages 1131–1136. IEEE, 2012.