Using Run-time Reconfiguration for Energy Savings in Parallel Data Processing

Madhura Purnaprajna, Christopher Pohl, Mario Porrmann, Ulrich Rueckert
Systems and Circuit Technology, Heinz Nixdorf Institute,
University of Paderborn, Germany
{madhurap, pohl, mario, rueckert}@hni.upb.de

***Abstract*—Parallelism and adaptability are two distinct architectural design considerations in embedded processors. Multi-core processors accelerate application execution on account of their inherent parallelism and run-time reconfiguration capabilities add adaptability during infield deployment. To benefit from both these features, a reconfigurable multiprocessor architecture – *QuadroCore* has been developed. A novel reconfiguration mechanism has been incorporated that provides fast run-time adaptability in a 4-processor cluster. In this paper, this scheme of reconfiguration has been used to save energy when using *QuadroCore* for data-parallel applications. As a proof of concept, a data-intensive neural network application called Self-organising Maps has been implemented on *QuadroCore*. Via reconfiguration, energy reduction of up to 30% has been observed for an implementation in UMC's 90nm standard cell technology.**

I. Introduction and Related Work

In embedded processing stringent performance and cost budgets necessitate both high-speed operations and low energy consumption. Huge design costs often restrict designing new processors for every new application. To shorten design & verification times and avoid fabrication costs, off-the-shelf products such as processors and FPGAs have been deployed. Easy programmability is an advantage in processors. In FPGAs the advantage of abundant parallel resources enables design acceleration. In addition, FPGAs provide flexibility and adaptability to map user-defined applications. Dynamic reconfiguration provides additional support to introduce design modifications during infield deployment. However, the dense interconnection network and the SRAM-based LUTs make present day FPGAs require relatively high power and area. High reconfiguration time, lack of dedicated tools and methods to automate reconfiguration are the main drawbacks in using FPGA-based run-time reconfiguration schemes. As an alternative, our approach is to merge parallelism in multi-core processors and dynamic reconfigurability in an existing cluster of parallel processors. The reconfiguration scheme introduces flexibility to adapt according to application-specific characteristics, viz., computation requirements, amount of communication, and frequency of synchronisation. This approach combines the advantages of parallel processing in multiprocessors and reconfigurability for application-specific adaptability. Furthermore, in FPGAs run-time reconfiguration has primarily been used as a mechanism for space multiplexing. In this paper, we present a novel reconfiguration mechanism to achieve energy savings, when using data-parallel applications.

Embedded applications such as video & image processing, and audio applications are characterised by large amount of data processing. For parallel data processing, multi-core, SIMD, and vector processors have been used frequently in literature. Multi-core architectures such as Ambric [1], Picochip [2] have been used for data-parallel processing. In multi-core processors, introducing additional cores provides additional resources to enhance the application performance. However, energy savings in parallel processing is entirely dependent on the ratio of the parallelisable component in the application. In the case of the Vector Thread processor [3], a control processor is responsible for broadcasting atomic instruction blocks to virtual processors, which decode and execute instructions locally. This allows parallelising parts of the application that can be executed simultaneously. However, the individual processing elements perform their own instruction fetch and decode operations independently. Instruction fetches translate to processor initiated memory transactions, which directly influence the power dissipated by the memory. As power consumed by memory is substantial, methods to minimise the contribution of memory power is of importance. Reducing memory transactions directly corresponds to reduction in system power consumption. Tensilica's Xenergy [4] approaches energy optimisation by customising the cache sizes and cache policies, since the cache itself contributes to a significant amount of power. Our approach is to introduce an energy-efficient SIMD mode of operation, which can be used for data-parallel applications when mapped onto parallel processors. In this SIMD mode one of the processors takes over the job of instruction fetch and decode, and all the decoded instructions are forwarded to all the processors, thus executing a single instruction stream. Additionally, a quick reconfiguration scheme helps reverting to the autonomous MIMD mode, where processors operate on their independent instruction and data streams. The possibility of switching between these two modes of operation enables using our reconfigurable multiprocessor *QuadroCore* for data-parallel applications even in the presence of intermediate data dependent control stages. In [5], a mixed-mode SIMD/MIMD reconfigurable processor called XC core has been designed for multi-media applications. Reconfiguration in XC core is a choice of operating the processing unit as a single processor in MIMD mode, or reconfiguring it to operate as 4 SIMD processing elements. In comparison to the above-mentioned

SIMD architectures, in *QuadroCore* the instruction fetch and decode is performed by a single processor. This results in a significant reduction in the memory power introduced via reduction in processor initiated instruction memory transactions. As a proof of concept for the architecture, a data-parallel application with intermediate stages of data-dependent control flow, called Self-Organizing Maps (SOM) has been evaluated. The SOM algorithm is representative of data-parallel scientific applications that can be implemented using this scheme. In Section II, the architectural details of our QuadroCore processor and the reconfiguration mechanism is presented. Section III describes the details of the SOM algorithm and its implementation on QuadroCore. A range of experiments was conducted to validate our approach. A performance comparison has been made to Tensilica's configurable processors, these results are included in Section IV. Finally, Section V provides the conclusions and future work.

II. QuadroCore Architecture and the Reconfiguration Mechanism

The QuadroCore architecture represents a MIMD (Multiple Instruction Multiple Data) processor cluster, comprising four 32-bit RISC processors called NCore, [6]. The four processors operate autonomously with their own local data and instruction memory, as shown in Figure 1. Each of the four processors has its own 16×32 bit local register file and 32KB instruction and data memory. Quick exchange of register contents between the four processors is enabled via a 32×32 bit shared register file. For larger data sharing, the processors have access to an external memory via arbitration over a shared bus. The architecture itself is scalable, i.e., multiple such QuadroCores can be interconnected for higher computational demands. The inter-cluster communication is possible via a network-on-chip with packet-based communication described also detailed in [6]. In this paper, we present the architectural features in QuadroCore that enables energy savings.

Analysing the power distribution in the QuadroCore, it has been observed that nearly 80% of the total power is drawn by memory. Thus, the power consumed by all the four processor cores approximately equates to the power dissipated by a single memory unit. Memory power is determined by the read and write operations initiated by the processor. This necessitates new strategies to improve the processor-memory interaction to reduce the entire system power. Hence, a reconfigurable interconnect has been introduced between the decode & execute stages of all the processors. This reconfigurable interconnect enables altering the control path of the individual processors. Via reconfiguration during run-time, the control path can be reconfigured such that only one processor performs the fetch & decode for all the processors and forwards the decoded instruction to all the participating processors. During run-time a special *reconfiguration* instruction configures the control path of the processors. This special power-saving mode of operation is called the SIMD mode. Via clock gating, the unused instruction fetch & decodes stages for other processors are disabled, resulting in zero processor-to-instruction memory transactions. In the SIMD mode, the processors execute instructions using the data in the local register files, as the data path remains unchanged. The main difference between conventional SIMD and the SIMD mode in QuadroCore is that only one processor performs the fetch & decode operation, all instructions within the base instruction set architecture can be used as SIMD instructions. Additionally, no special vector registers or operators are required for this SIMD mode. When using QuadroCore for applications such as search algorithms, sorting methods etc., it is typical to expect intermediate stages of data dependent control. In presence of such stages, reconfiguring the control path again permits switching back to the default MIMD mode, where the processors operate autonomously using their local instructions. The reconfiguration time is very small, a single clock cycle is sufficient to switch modes. Further, the reconfiguration instructions are embedded in the instruction stream, aiding automatic generation and management of reconfiguration. Further, to ease the inter-processor communication and synchronisation, additional reconfiguration capabilities have been integrated into QuadroCore. The associated compilation tool flow for these modes in QuadroCore is detailed in [7].

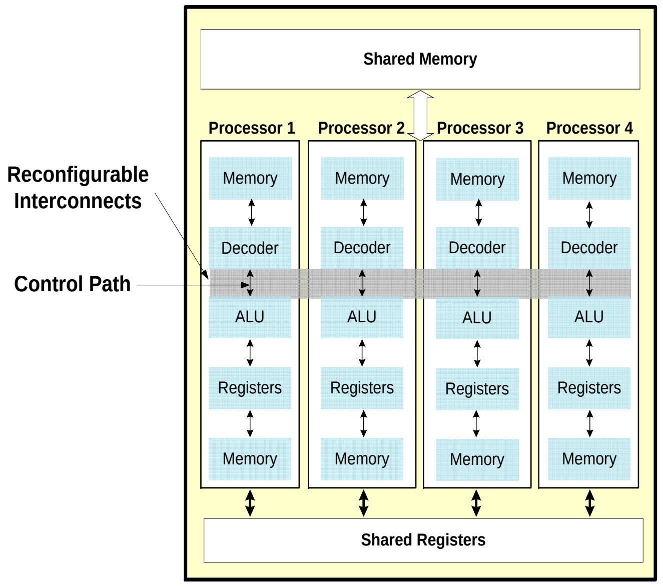


Fig. 1. QuadroCore Reconfigurable Multiprocessor

Figure 2 shows the control path, when QuadroCore operates in the SIMD mode. As seen, a single instruction fetch in SIMD mode replaces 4 instruction fetches in the MIMD mode. Hence, the SIMD mode of operation results in power savings in comparison to MIMD mode, but it does so at the cost of additional clock cycles required for all the processors to operate in synchronous mode. An additional advantage of the SIMD mode is the reduced code size for the slave processors, since in the SIMD mode the program resides only in the master processor.

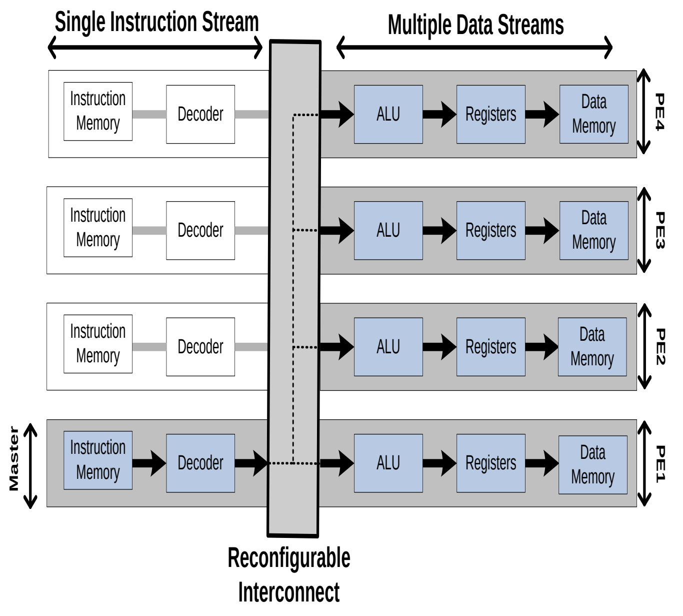


Fig. 2. Control and Data Flow in SIMD Mode

A. *Design Flow*

Figure 3 shows the methodology incorporated to optimize the given application for both time and power. The architecture was synthesized for UMC's 90nm standard cell technology. Further, the gate-level netlist running the application was simulated to record the application specific switching activity, which directly influences the dynamic power of the reconfigurable multiprocessor. This switching activity was fed to the power analysis tool from Synopsys to analyse the resulting power. With the same netlist, modifications were introduced in the application described in C, such that the SIMD mode of operation could be enabled wherever possible. The modifications are only MIMD to SIMD mode switches after the processors are synchronized and are executing the same code, without any input data-dependence in the control flow. In doing so, it had to be ensured that the master processor operated normally and the rest of the processors executed the same instructions as the master processor.

Partitioning the application onto the four processors is simplified for data parallel applications, since a single program resides on all the processors, when operating in MIMD mode. Tasks or functions that are same to all the processors, with no data dependent control, are executed in SIMD mode, by inserting a MIMD−SIMD mode switch. In case of data-dependent control, reconfiguration instructions are inserted to enable a SIMD−MIMD switch. Since, the reconfiguration instruction itself has an execution time of one clock cycle; the overhead of switching to and from SIMD mode is minimal. To ensure a seamless mode switch, the processors need to be synchronized before switching to the SIMD mode. The synchronization is achieved via a fast barrier synchronization, described in [7]. The synchronization process itself is achieved in a single clock cycle, after all the participating processors encounter the barrier instruction. Thus, for applications with data-dependent control, merging of blocks executed in SIMD or MIMD is ensured to minimize SIMD−MIMD mode switches. The following section describes the application scenario for evaluating the proposed scheme.

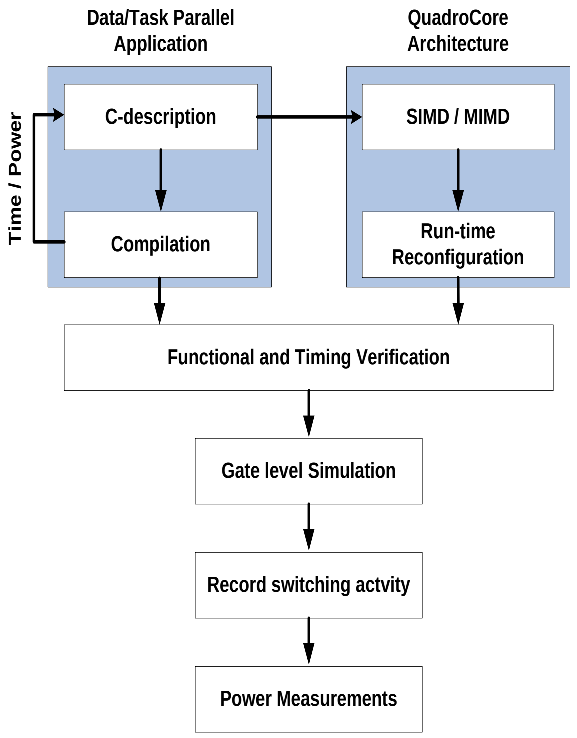


Fig. 3. Design Flow for Time and Power Optimization

III. Application Scenario

Self-organizing maps (SOM) are a special type of artificial neural networks (ANN) that have proven to be very effective for data analysis and exploration of high dimensional datasets [8]. The SOM is based on an n-dimensional (usually with $n = 2$) grid of such neurons (i.e., processing elements) that adapted to the input data-set $X \subset \mathbb{R}^m$. This results in a dimensionality reduction $\mathbb{R}^m \rightarrow \mathbb{R}^n$ where $m \geq n$, and hence resulting in a simplified representation of the original data-set. This representation can be visualised and analysed further. The algorithm is divided into three steps:

1) Initialization: The weight vectors $\overrightarrow{m_i} \in \mathbb{R}^m$ of all neurons N_i need to be initialised with random values

$$\forall_{i,j}\ m_{i,j} = rand[0...1], \tag{1}$$

where j denotes the components of the vector $\overrightarrow{m_i}$.

2) Best-match search: A vector $\overrightarrow{x}(t)$ is randomly selected from X and the distance between $\overrightarrow{x}(t)$ and all $\overrightarrow{m}_i$ is calculated. The neuron with the smallest distance to the input is called the best-match (BM).

$$\|\overrightarrow{x} - \overrightarrow{m}_{bm}\| = \min_{\forall_i} \|\overrightarrow{x} - \overrightarrow{m}_i\| \tag{2}$$

3) Adaptation: The weight vectors $\overrightarrow{m}_i$ are adjusted towards the input $\overrightarrow{m_i}$ based on the so called neighbourhood function h_{ci}

$$\forall_i \; \overrightarrow{m}_i(t+1) = \overrightarrow{m}_i(t) - |\overrightarrow{m}_i(t) - \overrightarrow{x}(t)| \cdot h_{ci}, \quad (3)$$

$$\text{where } h_{ci} = h_{ci}(t, \|N_i - N_{BM}\|) \quad (4)$$

h_{ci} is a function that decreases both in time and distance from the best-match; often a Gauss-kernel is chosen.

Steps two and three are repeated for all vectors $\overrightarrow{x} \in X$, where each iteration for all vectors is called one epoch. Depending on the requirements of the dataset, several such epochs may be required to form a properly organized map.

A. Runtime and Parallelism

The runtime of the SOM algorithm depends on the number of neurons, the number of vectors, the dimension of the vectors and the number of epochs:

$$T \propto |N||X| \dim(\overrightarrow{x}) \cdot \#epochs \quad (5)$$

The inherent nature of the algorithm suggests the exploitation of parallelism. All $|N|$ neurons do the same calculations when finding the BM or adapting to a new vector, therefore up to $|N|$ processing elements (PEs) can execute in parallel. The only necessary sequential step occurs when searching for the BM and communicating its position to all the PEs for adaptation. In general, the ratio of the sequential to parallel portion of code is very small portion of the code [9], which makes SOM ideal for implementation on parallel processors. For the QuadroCore, the execution time T_{Quadro} is governed by the following equation:

$$T_{Quadro} \propto \left(\left\lceil \frac{|N|}{|PE|} \right\rceil \dim(\overrightarrow{x}) + t_{gBM} \right) |X| \cdot \#epochs, \quad (6)$$

Where, PE is 4 for our QuadroCore processor and t_{gBM} is the time (in clock cycles) during which the processors synchronize, share their local BMs, and find a global BM.

B. Implementation

As a neuron-parallel approach is selected, the functionality of the neurons is represented by the functions *find local BM*, *find global BM*, and *adapt Map*. Since *find local BM* and *adapt Map* perform exactly the same operations on different data (cf. equation 2, function-level parallelism), they can be executed in SIMD mode (see Figure 4), which is done by inserting a reconfiguration instruction (function called `reconfig(mode)`). After the BM calculation is done, each processor sends its local BM data to the shared register file (function called `send(data, location)`).

The following synchronization steps between *find local BM* and *adapt Map* are data-dependent (a search operation), thus performed by one processor only (MIMD). Therefore, another reconfiguration instruction is inserted and the local BM data from all the processors is received from the shared register file `receive(data, location)`) to calculate a global BM. For adaptation (cf. equation 3), all processors enter SIMD mode again, as they share the same instruction stream. Finding the local best matches and adapting the map are the dominating functions, most frequently executed. They are executed on all the processors simultaneously in SIMD mode, resulting in significant power savings and an increase in code density when using the proposed approach, cf. Section IV.

C. Applicability to other algorithms

The mapping of the SOM algorithm to QuadroCore design flow is typical of data-parallel applications, where large amount of data needs to be processed. The algorithm is composed of compute intensive functions, which are mapped to several processors executed in parallel. A producer-consumer relationship (e.g. *find local BM* and *adapt Map* functions) is made possible via a message passing interface using the shared register file. Function-level parallelism uses the available spatial parallelism of multi-processors, which is independent of the input data. Further, the proposed concept of reconfiguration during run-time to save power is applied when all processors operate on independent data, but perform the same operations using the SIMD mode. Thus, reconfiguration is infrequent and at function-level granularity.

Since mapping an application to this architecture does not involve rewriting legacy code, portability is made possible by inserting mode-specific directives in code. The code modifications involve identifying regions of data-dependant control, for regions where the control path can be determined only during compile time. Thus, in case of input data dependencies, processors operate on their individual instruction stream in the default MIMD mode. Reconfiguration is managed via high-level language support, reducing design and developmental costs.

IV. Experiments and Results

The QuadroCore architecture comprising four processors and their local memories was synthesized in UMC's 90nm standard cell technology with a timing constraint of 200 MHz. Power reports were obtained after back-annotating the switching activity report generated using gate-level simulation. The core architecture has a power dissipation of 0.05 mW/MHz (excluding memories). For comparison, a single instance of a 90 nm, 32-bit Xtensa processor from Tensilica exhibits a power dissipation of 0.032 mW/MHz and the MIPS 34K processor exhibits a power dissipation of 0.56 mW/MHz, as reported in [10].

A. Power Distribution

The plot shown in Figure 5 shows the variation in power distribution within the QuadroCore cluster. Comparing the power consumption of the processor core itself to the power consumed by the local instruction and data memory, it can be seen that nearly 80% of the total power is drawn by memory, which is mainly determined by the processor initiated memory transactions. This implies that the total power for all the four processors approximately equates to the power dissipated by a single memory unit. Hence, optimizations in terms of power, for the processor itself can only make a

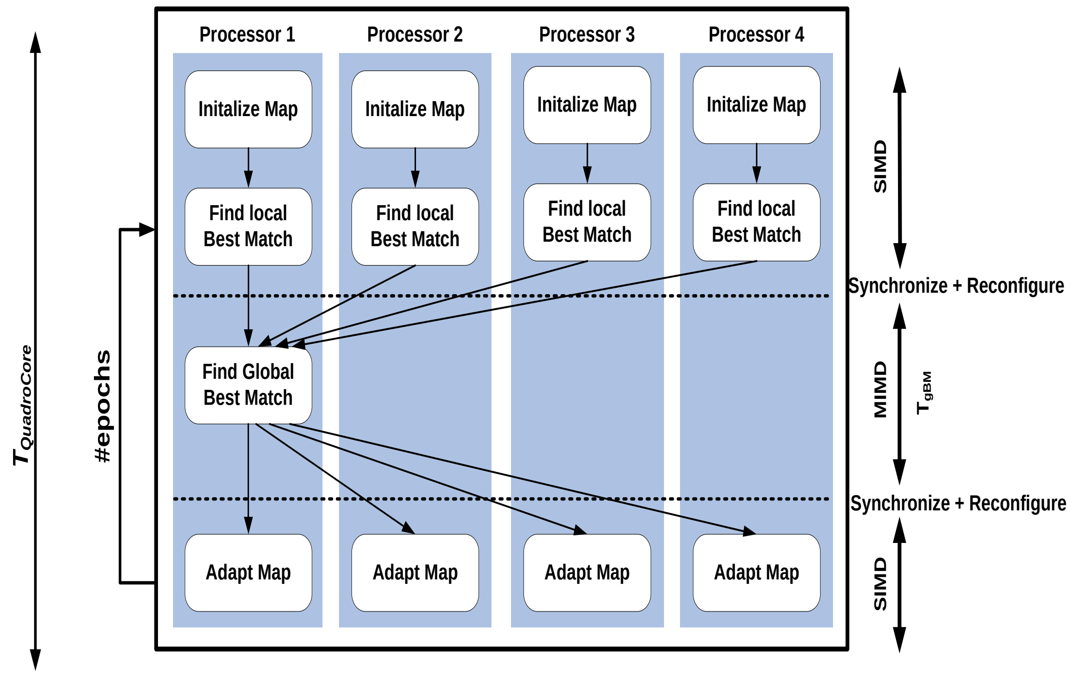


Fig. 4. SOM Application mapping on QuadroCore

low impact in the total power of the system. This entails new strategies to improve the processor-memory interaction in order to optimize the entire system power, as proposed here in this paper. As illustrated in Figure 5, switching to SIMD mode has reduced the power consumed by memory 1, 2, 3 because of the reduced instruction fetches in the respective processors. A 15% reduction in the total power consumed has been observed.

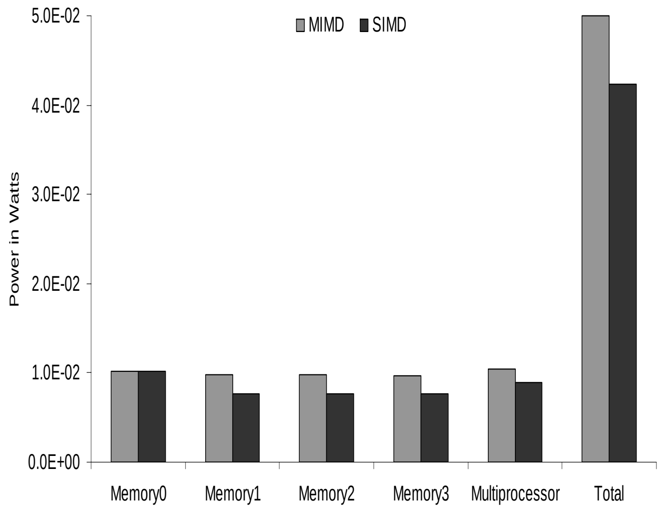


Fig. 5. Comparing power consumed for memory and processors

B. Energy Savings in QuadroCore

Tables I, II, and III compare the performance of the SOM application when mapped to (a) a single processor, (b) QuadroCore operating in MIMD mode, and (c) when operating in the low-power SIMD mode (which includes reconfiguration time to/from MIMD mode). All power measurements include the processor cores and their instruction and data memories respectively. The speedup denotes the improvement in speed of execution on the 4-processor QuadroCore as compared to the time on one NCore processor. Power savings indicate the amount of power saved when operating in the SIMD mode. Energy is calculated for the processor operating at 200 MHz and is the product of the total time required (product of execution cycle and clock period) and the power consumed in the respective modes. Energy savings in MIMD denote the reduction in energy in comparison to a single NCore. Energy savings in SIMD mode imply the energy saved when operating in SIMD mode in comparison to the MIMD mode. Energy saving of up to 30% is observed on account of the reduced instruction memory accesses, which in turn results in switching off unused resources. The reduction in speedup for the SIMD mode is the cost of reconfiguration between the MIMD$-$SIMD in the presence of input data dependent changes in control flow.

C. Variation in Application Parameters

Performance characteristics for variations in application parameters such as, the number of vectors, components, and number of epochs were measured. Gate-level simulations are time-consuming and since the parameters directly influence the simulation time, three indicative results are plotted in Figure 6.

TABLE I
PERFORMANCE COMPARISON OF SOM WITH PARAMETERS: VECTORS = 5, EPOCH = 1, COMPONENTS = 5, NEURONS = 16

Operating Mode	Execution Cycles	Speedup	Power(W)	%Power Savings	Energy(μJ)	%Energy Savings
(a) Single Processor	53,870	1	0.018		4.89	
(b) MIMD	15,608	3.45	0.052		4.05	17.05
(c) MIMD−SIMD	15,790	3.41	0.044	14.72	3.50	28.44

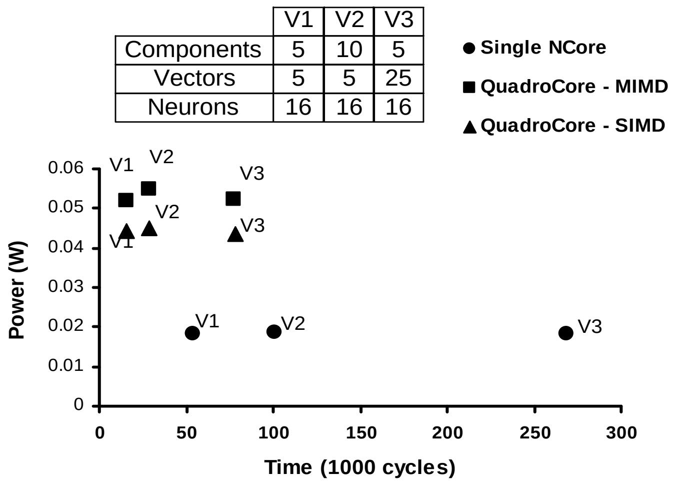
TABLE II
PERFORMANCE COMPARISON OF SOM WITH PARAMETERS: VECTORS = 25 , EPOCH = 1, COMPONENTS = 5, NEURONS = 16

Operating Mode	Execution Cycles	Speedup	Power(W)	%Power Savings	Energy(μJ)	%Energy Savings
(a) Single Processor	268,327	1	0.018		24.40	
(b) MIMD	77,267	3.47	0.052		20.20	17.27
(c) MIMD−SIMD	78,250	3.43	0.043	16.69	17.04	30.20

TABLE III
PERFORMANCE COMPARISON OF SOM WITH PARAMETERS : VECTORS = 5 , EPOCH = 1, COMPONENTS = 10, NEURONS = 16

Operating Mode	Execution Cycles	Speedup	Power(W)	%Power Savings	Energy(μJ)	%Energy Savings
(a) Single Processor	100,719	1	0.019		9.21	
(b) MIMD	28,581	3.52	0.055		7.83	16.45
(c) MIMD−SIMD	28,744	3.51	0.045	17.78	6.47	30.91

As seen in the plot, the speedup was in the range of 3.2 to 3.5, and power savings in the range of 14% to 18% have been observed in the SIMD mode when compared to the MIMD mode. In comparison to a single processor, the corresponding energy savings were in the range of 28 to 30%. On account of switching between SIMD−MIMD modes, the reduction in speed for SIMD operation is in the range of 0.2 to 2%. Apart from power and energy savings, a reduction in code-size of 8% on account of reduced operations in the SIMD mode were also observed.



	V1	V2	V3
Components	5	10	5
Vectors	5	5	25
Neurons	16	16	16

Fig. 6. Impact of Application Parameters on Power and Execution Time

D. Comparing Area, Execution Time and Energy

The plot in Figure 7 compares the variations in area, time, and energy for the SOM algorithm mapped onto a single NCore processor, the QuadroCore operating in MIMD mode, SIMD mode, Tensilica's base processor Xtensa (1K Instruction cache and 1K Data cache), and the same Xtensa base processor augmented with instruction set extensions for the SOM algorithm. For comparison, all the cores were analysed at 90nm technology operating at 200 MHz, with application parameters as indicated in Table I. The size of the points corresponds to a third dimension, which is the gate-count for the processor core (excluding the memory) in each of the cases. QuadroCore outperforms the other processors in terms of clock cycles and energy. However, it has to be noted that a QuadroCore processor has a larger area on account of its four independent data paths as compared to a single data path in NCore and Xtensa processors. Additional control and data paths in QuadroCore explore coarse-grained parallelism and provide a speed advantage, in comparison to fine-grained additional instructions in Xtensa. Additional energy savings are observed in QuadroCore's SIMD mode. An important difference between the reconfigurable QuadroCore and the

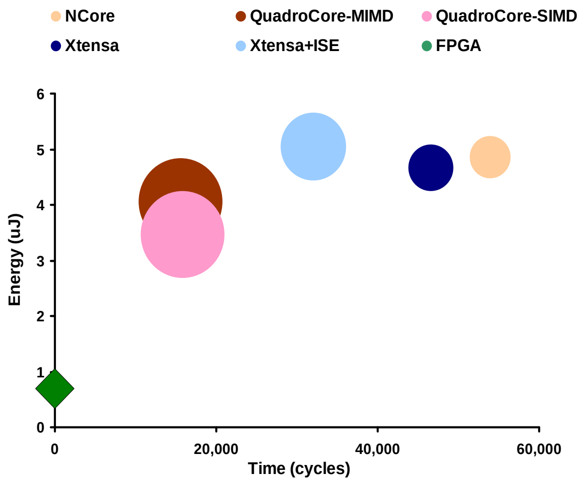


Fig. 7. Comparing Area, Energy and Execution time

configurable Xtensa processors is the run-time adaptability in QuadroCore in comparison to design time configurability (via additional instructions, resources) in Tensilica's processors. For comparison, a custom realisation of the SOM algorithm based on [11], was implemented on Xilinx XCV4LX100, operating at 150 MHz (indicated as a green quadrilateral). This implementation results in a significantly improvement in time and energy. However, the inherent configurability in FPGAs results in a high area, which is excluded in Figure 7.

V. Conclusions and Future Work

Reconfiguration is enabled in the QuadroCore processors by adding a layer of interconnects between the decode & execute units. This architectural modification enables reconfiguring the control path among the processors. This introduces a SIMD mode of operation, which results in achieving energy-efficiency by reducing the instruction memory transactions. The reconfigurable interconnect does not interfere with the original ISA and hence ensures backward compatibility. This feature has been used to achieve energy savings in data parallel applications. Mapping of a data-parallel SOM algorithm has shown to save about 14% to 18% of power and nearly 30% of energy in comparison to a single processor. The energy savings achieved in SIMD in comparison to a fixed MIMD architecture is about 15%. This philosophy can be extended to other applications, which benefit from parallel data processing on multiple processors. As seen, increasing the number of processors results in energy savings even with variations in the problem parameters. The presented scheme of control path reconfiguration is generic and can be extended to other multiprocessor architectures. Ongoing work includes extending this scheme to further reduce power by minimising data memory access by using the reconfigurable interconnect to alter the data path among the processors.

References

[1] M. Butts, "Synchronization through communication in a massively parallel processor array," *IEEE Micro*, pp. 32–40, Sept.-Oct. 2007.

[2] *PC102 Product Brief*, PicoChip Inc, March 2004, available from http://www.picochip.com/.

[3] R. Krashinsky, C. Batten, M. Hampton, S. Gerding, B. Pharris, J. Casper, and K. Asanovic, "The vector-thread architecture," *SIGARCH Comput. Archit. News*, vol. 32, no. 2, p. 52, 2004.

[4] *Optimizing for Energy Using the Xenergy Energy Estimation Tool*, Tensilica Inc., Aug. 2007, available from http://www.tensilica.com.

[5] S. Kyo, T. Koga, L. Hanno, S. Nomoto, and S. Okazaki, "A low-cost mixed-mode parallel processor architecture for embedded systems," in *ICS '07: Proceedings of the 21st annual international conference on Supercomputing*. New York, NY, USA: ACM, 2007, pp. 253–262.

[6] J.-C. Niemann, C. Puttmann, M. Porrmann, and U. Rueckert, "Resource efficiency of the GigaNetIC chip multiprocessor architecture," *Journal of System Architecture*, vol. 53, no. 5-6, pp. 285–299, May 2007, Special issue on Architectural premises for pervasive computing.

[7] M. Hussmann, M. Thies, U. Kastens, M. Purnaprajna, M. Porrmann, and U. Rueckert, "Compiler-driven reconfiguration of multiprocessors," *in Proceedings of the Workshop on Application Specific Processors (WASP) 2007*, pp. 3–10, 2007.

[8] T. Kohonen, *Self-organization and associative memory*. Springer-Verlag New York, Inc. New York, NY, USA, 1989.

[9] M. Porrmann, U. Witkowski, and U. Ruckert, "A massively parallel architecture for self-organizing feature maps," *Neural Networks, IEEE Transactions on*, vol. 14, no. 5, pp. 1110–1121, Sept. 2003.

[10] T. R. Halfhill, "MIPS 74k goes superscalar," *In-Stat Microprocessor Report 2007*, vol. 21, May 2007.

[11] C. Pohl, M. Franzmeier, M. Porrmann, and U. Ruckert, "gNBX-reconfigurable hardware acceleration of self-organizing maps," in *Field-Programmable Technology, 2004. Proceedings. 2004 IEEE International Conference on*, 2004, pp. 97–104.