# MapReduce Accelerator for Embedded Applications

Mihaela Maliţa
Computer Science Department
Saint Anselm College, Manchester, NH

Gheorghe M. Ştefan
Electronic Devices, Circuits and Architectures Department
Politehnica University of Bucharest, Romania

***Abstract*—There are large classes of embedded applications involving tightly interleaved *complex* and *intense* computations. The solution we propose segregates the complex from the intense in a many-core centered engine. We base our approach on a map-reduce *abstract machine model* suggested by Kleene's *mathematical model of computation*. An actual *MapReduce Accelerator* is described and is evaluated for various application domains. Results, based on ASIC and FPGA implementations, show $> 10\times$ improvements in area & energy use.**

***Index Terms*—many-core, accelerators-based architecture, embedded computation, MapReduce, computation model.**

## I. Introduction

The current embedded accelerators are specific accelerators (for graphics, video, SDR, ...) or *add hoc* structured multi- or many-core engines. Our proposal is to use a many-core approach to provide a general purpose parallel engine able to perform efficiently all forms of intense computations requested in the embedded domain. The criteria for evaluating such an architecture are: the use of power ($GOPS/Watt$) and the use of area ($GOPS/mm^2$). According to the evaluations made for few applications, our approach provides for both, energy and area, more than $10\times$ improvements. The *MapReduce Accelerator* solution is compared with the ARM processors, the most used general purpose embedded processors.

The second section explains the reason for the use of a MapReduce architecture for the intense part of the embedded computation. The third section describes the one-chip implementation of the MapReduce Accelerator's structure. The fourth section reviews few classes of applications, already investigated, and shows the improved use of area and power compared with mono-core embedded computation.

## II. Why Map-Reduce?

In order to develop a general purpose parallel accelerator we propose a three-step approach: (1) consider a mathematical parallel model of computation (answering the question: *what is* parallel computation?), (2) define an abstract machine model (which is about *how* the structure of a parallel machine is organized), and (3) design a parallel architecture (which provides the functional *interface* between the physical structure and the informational structure used to program the parallel machine).

Instead of building *ad hoc* parallel engines, putting together Turing-based machines, we started from Stephen Kleene's mathematical model of computation [4] which provides a genuine model for parallel computation, just as Turing's model did for the sequential computation. We already proved that only the first out of the three Kleene's rules – the *composition* – is independent [6]. Therefore, in defining real abstract machine models, the composition rule, expressed as

$$f(x) = g(h_1(x), h_2(x), \ldots h_p(x))$$

where: $x = \{x_1, x_2, \ldots x_p\}$, is the only one to be considered.

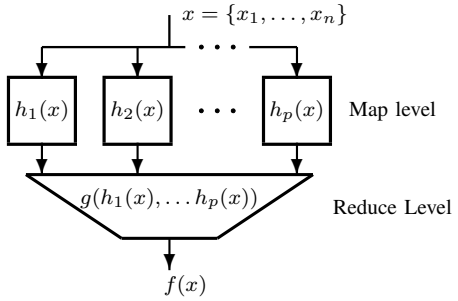


Fig. 1. **The circuit structure associated to composition.**

The *abstract parallel machine model* results as a two-level construct suggested in Fig. 1. The ***map level*** works on correspondences between functions or vector of functions and variables or vector of variables. Three mappings result: function to vector (*data-parallel*), variable to vector of functions (*speculative-parallel*), vector of functions to vector of variables (*thread-parallel*). The ***reduce level*** computes a vector to value function.

The users image of the architecture consists of two arrays: the one-dimension array of the external memory and the two-dimension array of the internal memory (a vector memory) distributed along the cells at the map level (in cell $i$ are stored all the $i$-th components of the vectors). The operations on the internal two-dimension memory are predicated operations on vectors. A Boolean vector is used for predicated operations. The accelerator performs the following types of operations:

```
scalarVect | BooleanVect <= OP(vect, vect)
scalarVect | BooleanVect <= OP(vect, scal)
scal | Boolean <= OP(vect)
```

The predicated operations perform a "spatial if-then-else" along the cells of the map level. The form:

```
where (BooleanVect) OP1(...);
   elseWhere        OP2(...);
endWhere
```

stands for: `OP1` is done in cells where `BooleanVect = 1`, while `OP2` is done in cells where `BooleanVect = 0`

## III. MapReduce One-Chip Engine

An actual implementation of the abstract machine defined above is in Fig. 2 [5], where PROCESSOR runs the complex part of the program and controls the execution of the intense

part on the MapReduce ACCELERATOR. Each cell in MAP and CONTROLLER execute the same ISA. The program is organized in pairs of instructions, one for CONTROLLER, one for the cells in MAP. The local memory in cells contains, if needed, code for the *thread-parallel* mode.

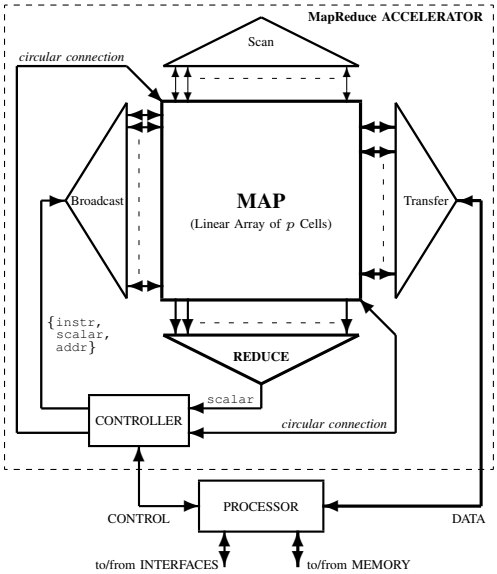


Fig. 2. **The MapReduce Accelerator**

Because each cell has its $m$-word memory, the data in MAP represents an array of $m$ $p$-scalar horizontal vectors:

$$v_1 =< x_{11}, \ldots, x_{1p} >$$
$$v_2 =< x_{21}, \ldots, x_{2p} >$$
$$\ldots$$
$$v_m =< x_{m1}, \ldots, x_{mp} >$$

where: each "column", $< x_{1i}, \ldots, x_{mi} >$, is the "vertical" vector of scalars associated to the computational cell $c_i$.

## IV. Applications

The embedded MapReduce architecture is cycle accurate evaluated on a FPGA prototype described in [3], where PROCESSOR is an ARM Cortex 9 and MapReduce ACCELERATOR consists of 128 16-bit cells engine. On the other hand, for evaluating the area and energy efficiency we synthesized an ARM Cortex 9 core and a MapReduce ACCELERATOR for 64 16-bit cells, proving that the two engines have the same area and consume the same energy working at the same frequency.

### A. Pure SIMD Applications

When data consists of independent vertical vectors, the system works in SIMD mode with $100\%$ degree of parallelism.

*1) Encryption:* the AES algorithm on *Cortex A9* is done in 173 cycle/byte, while on a 64-cell MapReduce engine is 2.1 cycle/byte. Results the use of area & power is improved $173/2.1 = 82$ times [2]. The acceleration is supra-linear because the data transfer is transparent to the computation.

*2) FFT:* because for FFT 32-bit computation is requested, a 32 32-bit cell engine is used [2]. It has the size and the power consumption of a *Cortex A9* core. The improvement in area and power use is $19\times$. Because the accelerator does not have hardware multipliers in its cells, the multiplication is performed in 10 cycles, resulting less than $32\times$ improvement.

### B. SPMD Applications

*1) Frame Rate Conversion:* the BA1024 chip, centered on a 1024 16-bit cells MapReduce accelerator implemented in $90nm$ and running at 250 MHz, was programmed to perform in real time the frame rate conversion for HDTV.

*2) Sorting:* the Batcher sort algorithm implemented on a 64 16-bit cell accelerator provide $> 64\times$ improvement [2].

*3) MIMD-Like Operation:* in [7] the cells in MAP are serial connected to perform a pipe of functions on a stream of data.

### C. MapReduce Applications

*1) SIFT Algorithm:* the SIFT algorithm has a version for SAD, and another for SSD. For SAD [3] the acceleration is $26\times$; for SSD it is $42\times$. A MapReduce engine of 128 cells was compared with a dual ARM Cortex 9 with Neons (equivalent with 8 SIMD cores). Thus, the acceleration is supra-linear.

*2) Neural Network:* the NN computation consists in matrix-vector multiplication, i.e., in dot-product, a typical map-reduce operation. In [1] is proved that the fastest specialized chip in 2007, Hitachi WSI, is $17\times$ slower than the solution with our MapReduce engine with 1024 16-bit cells ($90nm$ version).

## V. Conclusion

The MapReduce Accelerator provides, for some computationally intense applications, more than $10\times$ improvement in the area and energy use by strongly segregating the complex part from the intense part of the computation [5]. The MapReduce abstract model originated in Kleene's mathematical model provides the foundation for developing a parallel environment for embedded computation.


## Acknowledgment

The authors got support from the development team of the first versions of the MapReduce Accelerator: E. Altieri, F. Ho, B. Mîţu, M. Stoian, D. Thiebaut, T. Thomson, D. Tomescu. Special thanks for V. Codreanu for the synthesis of the ARM core and the MapReduce Accelerator.


## References


[1] R. Andonie, M. Maliţa, "The ConnexArray$^{TM}$ as a Neural Network Accelerator", *Proceedings of the 3rd IASTED Inter. Conf. COMPUTATIONAL INTELLIGENCE* July 2-4, 2007, Canada, pp 163-167.

[2] C. Bîra, L. Gugu, M. Maliţa, G. Ştefan: "Maximizing the SIMD Behavior in SPMD Engines", in *Proc. of the World Congress on Engineering and Computer Science 2013* Vol I 2013, San Francisco,pp 156-161.

[3] C. Bîra, R. Hobincu, L. Petrică, V. Codreanu, S. Coţofană, "Energy-Efficient Computation of L1 and L2 Norms on a FPGA SIMD Accelerator, with Applications to Visual Search", *18th International Conference on Circuits, Systems, Communications and Computers*, Greece, 2014.

[4] S. C. Kleene, "General Recursive Functions of Natural Numbers", in *Math. Ann*., 112, 1936.

[5] G. Ştefan, "One-chip TeraArchitecture", *Proceedings of the 8th Applications and Principles of Information Science Conference*. Okinawa, 2009.

[6] G. Ştefan, M. Maliţa, "Can One-Chip Parallel Computing Be Liberated From Ad Hoc Solutions? A Computation Model Based Approach and Its Implementation", *18th Inter. Conf. on Ciruits, Systems, Communications and Computers*, Santorini, July 17-21, 2014, 582-597.

[7] D. Thiebaut, M. Maliţa, "Pipelining the Connex Array", *BARC 07*, Boston University, MA, Jan 26, 2007,pp. 13-17.