FPGA-implementation of a Holographic Pattern-matching Algorithm

András Kiss, Zoltán Nagy,
Péter Szolgay and Tamás Roska
Computer and Automation Research Institute
Hungarian Academy of Sciences
and Dept. Information Technology
Pázmány Péter Catholic University
Budapest, Hungary

György Csaba
and Xiaobo Sharon Hu
and Wolfgang Porod
Dept. of Electrical Engineering
University of Notre Dame, Notre Dame, IN, USA

***Abstract*—In this paper, we demonstrate the FPGA implementation of a massively parallel, non-Boolean pattern-matching algorithm. The algorithm is based on the concepts of optical computing: quasi-optical wave equations are solved numerically, using FPGA-accelerated hardware. The FPGA-based wave-equation solver is very well parallelizable, so the resulting pattern-matching algorithm will also be amenable to mega-core architectures.**

Keywords*—*FPGA simulation, Wave equation, Mega-core systems

I. Introduction

It is expected that thousand-core or even mega-core chips will become ubiquitous in the coming years or decades [1] [2]. It is increasingly challenging to realize large, high performance computing cores, mitigate memory bottlenecks and manage power dissipation, in a complex processor core. It is much more economical to replicate and interconnect relatively simple cores. Off the shelf FPGA systems can readily realize thousand-core systems. However, it is not at all known how the computational power of such many-core systems can be harnessed. Most practically important algorithm are single-threaded and challenging to parallelize to just a few computing cores. There are a few special-purpose computing tasks (such as the ones in scientific computing) which can be efficiently paralellized.

In this paper we point out that a special-purpose computing algorithm, which solves the two-dimensional wave equation, can, in fact serve as a computational engine of a pattern-matching algorithm. Pattern matching itself is a basis of widely used image processing / recognition algorithms. Since the solution of the wave equation is well paralellizable, the pattern-matching algorithm itself will be possible to distribute to a large number of computing cores.

Section II introduces 'holography on a chip' - a method we devise to use on-chip wave propagation to perform computational primitives. Section III shows how optical equations can be emulated by digital hardware. The transformation of the wave equations onto the emulated digital CNN hardware is shown in Section IV. In Section V we describe the details of the implemented solver architecture. Finally, in Section VI the results are introduced.

II. Basics holographic computing

Holographic computing is inspired by optical signal processing algorithms [4], [3], [5], albeit we use only digital electronic hardware. The principle of pattern matching is shown in Fig. 1a). A two-dimensional medium is assumed, which can support the propagation of waves. For now we refer to the waves in an abstract sense, i.e. an excitation that shows interference phenomenon. The nature of the waves will be specified in the coming sessions.

In Fig. 1a) the waves are generated at the bottom and the distribution of the sources is defined by an input pattern. The waves then travel through a diffraction grating (scatterer) before they arrive to the top plane.

The grating is chosen in a way that it focuses light to a certain point on the top plane. In the simplest case, the scatterer can be a series of slits (holes) that lets the waves going through only if they constructively interfere at the chosen focal point. For a certain 'original pattern' there is only one matching grating that creates a sharp focus. In the special case when the sources generate a plane wave, then the scatterer is the well-known Fresnel zone plate in optics [8].

A particular interference pattern is shown in Fig. 1b) - for details, see [10].

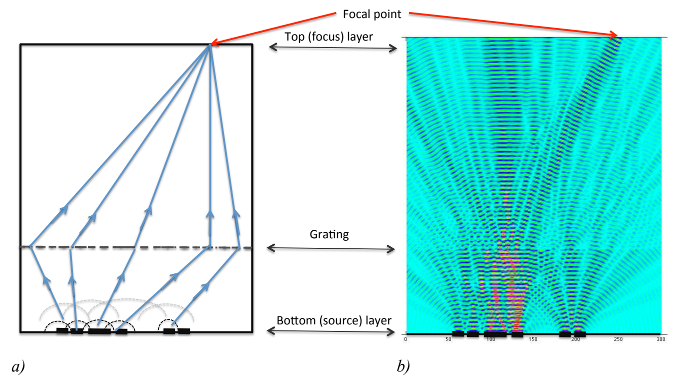

Fig. 1. Panel a) is a sketch of a holographic pattern matching algorithm - for detailed explanation see text. Panel b) shows the calculation of the interference pattern for a particular input, showing how a focal point emerges.

The pattern recognition algorithm works by simply using different wave source distributions (test patterns) in the bottom of the structure. If the source distribution is similar to the original pattern, then the focal point of the waves will be sharp. If the test pattern is dissimilar, then constructive interference will not occur. The intensity of the waves at the focal point measures the similarity of the test pattern to the original pattern.

This algorithm allows frameless, analog pattern matching operation: the input at the bottom pattern can continuously change, while the output (the wave intensity of the focal point) can continuously be read out.

III. Wave propagation in a cellular circuit

Here we show how one can construct digital hardware to create interference phenomena similar to optical interference and use this hardware in a computing system. One way to do that is to simulate electromagnetic wave propagation using a circuit.

We choose transversal electric (TE) modes of electromagnetic wave propagation, a particular solution of Maxwell's equation [6]. Using a rectangular grid with Δx and Δy spacings, Maxwell's equations in 2 dimensions take the form:

$$\begin{aligned} \frac{\partial H_x}{\partial t} &= \frac{-1}{\mu}\frac{\partial E_z}{\partial y} \\ \frac{\partial H_y}{\partial t} &= \frac{1}{\mu}\frac{\partial E_z}{\partial x} \\ \frac{\partial E_z}{\partial t} &= \frac{1}{\epsilon}\frac{\partial H_y}{\partial x} - \frac{1}{\epsilon}\frac{\partial H_x}{\partial y} \end{aligned} \tag{1}$$

where H_x, H_y and E_z are the in-plane (x, y) components of the magnetic field and the out of plane (z) component of the electric field. Here we simply consider them as variables describing the wave propagation.

In order to model these waves on digital hardware, we discretize the computational domain by using a Δx and Δy spacing. On each lattice point the dynamics of H_x, H_y and E_z are described by the ordinary differential equations:

$$\begin{aligned} \frac{dH_x^{(i,j)}}{dt} &= \frac{-1}{\mu}\frac{E_z^{(i,j+1)} - E_z^{(i,j-1)}}{2\Delta y} \\ \frac{dH_y^{(i,j)}}{dt} &= \frac{1}{\mu}\frac{E_z^{(i+1,j)} - E_z^{(i-1,j)}}{2\Delta x} \\ \frac{dE_z^{(i,j)}}{dt} &= \frac{1}{\epsilon}\frac{H_y^{(i+1,j)} - H_y^{(i-1,j)}}{2\Delta x} \\ & \quad -\frac{1}{\epsilon}\frac{\partial H_x^{(i,j+1)} - H_x^{(i,j-1)}}{2\Delta y}, \end{aligned} \tag{2}$$

where the indices (i, j) denote the i-th and j-th cell of the lattice. If the calculation domain is discretized to $N \times M$ cells, then obviously $i = 1..N$, $j = 1..M$.

The time-continous, spatially discretized system of Eqs. 2 can be directly simulated by analog Cellular Nonlinear Networks [7]. In order to model this system by digital hardware, one should make one step further to discretize Eqs. 2 in time as well. The simplest approach is to use Euler method: $H_x^{(i,j)}(t+\Delta t) = H_x^{(i,j)}(t)+\Delta t\times\mathrm{r.h.s}$, etc, where $\mathrm{r.h.s}$ denotes the right-hand side of the corresponding equation in Eqs. 2.

The emulation of the TE waves becomes a digital numerical problem, solvable by either software, multi-core CPUs, GPUs or FPGAs. Due to the nearest-neighbor coupling, the simultaneous time evolution of the cell updates and the lack of memory operations solution of the wave problem is straightforward to parallelize. In fact, one can view every lattice point (described by Eqs. 2) as an elementary processor, interacting with its nearest neighbors and calculating the wave amplitudes at this particular lattice point. Alternatively, one can calculate the cell updates 'row by row', and possibly working on a larger number of rows in parallel - we follow this route in the remaining of the paper.

Strictly looking holographic algorithms assume that waves propagate in a boundary-free region, so radiating boundary conditions must be implemented for Eq. 2. This can be done by the introduction of perfectly matched layers as described in [6].

A Runge Kutta method significantly improves the accuracy of the solution of Eq. 2, resulting in also a much more complex cell structure.

IV. Emulated-Digital CNN Implementation of the Wave Equations

The Falcon architecture is an emulated digital CNN processor [9], which can solve the discretized version of the governing equations of the wave propagation. On this architecture the flexibility of simulators and computational power of analog architectures are mixed. Not only the size of templates and the computational precision can be configured but space-variant and non-linear templates can also be used.

The Maxwell equations are solved by a modified Falcon processor array in which the arithmetic unit was redesigned according to the discretized governing equations. Since each CNN cell has only one real output value, three layers are required to represent the variables Hx, Hy and Ez as can be seen on Figure 2.

In the first-order case the CNN templates acting on the different layers can easily be taken from (2). Equations (3)-(6) show templates, in which cells of different layers are connected to the cell at position (i, j). $A_{1a}^{E_z(H_x)}$ means that layer H_x has an effect on E_z layer. The other templates are defined by the same way.

$$A_{1a}^{E_z(H_x)} = \frac{\Delta t}{2\epsilon\Delta x}\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{3}$$

$$A_{1b}^{E_z(H_y)} = \frac{\Delta t}{2\epsilon\Delta y}\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{4}$$

$$A_{2}^{H_x(E_z)} = -\frac{\Delta t}{2\mu\Delta y}\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \tag{5}$$

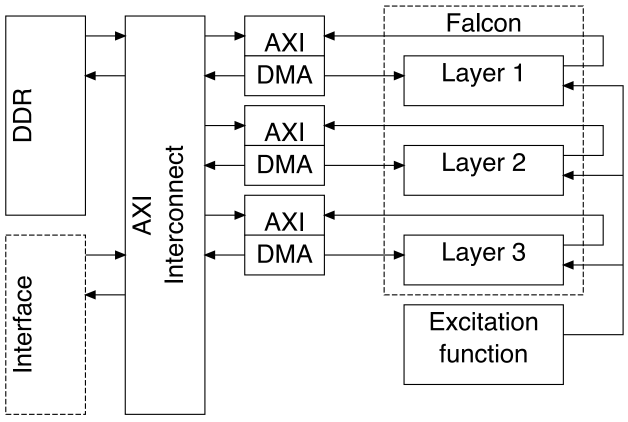

Fig. 2. Block diagram of the architecture of the first order scheme. Three layers are required for computing the variables and an arithmetic unit is needed for computing the excitation function.

$$A_3^{H_y(E_z)} = \frac{\Delta t}{2\mu\Delta x} \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{6}$$

A 2 row belt of the input image has to be stored on-chip in order to efficiently work the arithmetic unit. A mixer unit is needed for every layer in order to generate the proper 3 $\times$ 3 neighbor of a computed cell (see Figure 3).

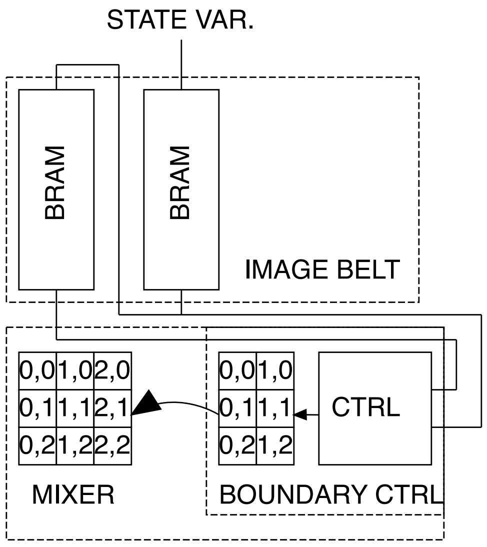

Fig. 3. Block diagram of the mixer unit. The three layers require the on-chip store of a belt from the computational domain and a mixer unit for generating the neighborhood of the computing cell.

The mixer unit provides the input image, where the templates can operate with. In this case the arithmetic unit is optimized to the template function in order to save some FPGA resources and gain computing speed (see Figure 4).

Due to the relative small amount of excitation point (compared to the grid size) (e.g.: Figure 1), the excitation function in every iteration step can be computed separately from the governing equations.

V. Properties of the FPGA implementation

The architecture was developed on a Xilinx XC7Z020 FPGA [11], which takes place on a Zedboard. The FPGA contains 85K programmable Logic Cells, 220 18 $\times$ 25 bit programmable DSP slices and 140 36Kb BlockRAMs. There is an 512MB on-board DDR3 memory, which can store the initial data and the result of one iteration and furthermore the exciting function of the excited points.

The on-board memory and the concept of processing the cells row-by-row makes it possible to compute on large 1000 $\times$ 1000 or even larger grid arrays. It is also possible to implement an array of processing element, where each column of processor elements can work on a stripe of the grid image and each consecutive row of the processors can compute the next iteration step of the algorithm.

The whole system relies on the AMBA (Advanced Microcontroller Bus Architecture) AXI system [12], which is not a bus as its name suggest but a point-to point structure between the communicating elements. This makes it possible later to move the implementation into a larger FPGA. The results of one iteration step of each layer are stored on the on-chip memory via DMA transactions. For uploading the initial data, the excitation function and other parameters and for downloading the computational results a host interface is needed.

The operators of the arithmetic unit are XILINX Floating Point operators with single precision and AXI bus support. The operators improved with the AXI bus makes it possible to change the arithmetic unit easily and therefore the arithmetic unit is optimized according to the sparse and symmetrical templates (Equations (3), (5) and (6)).

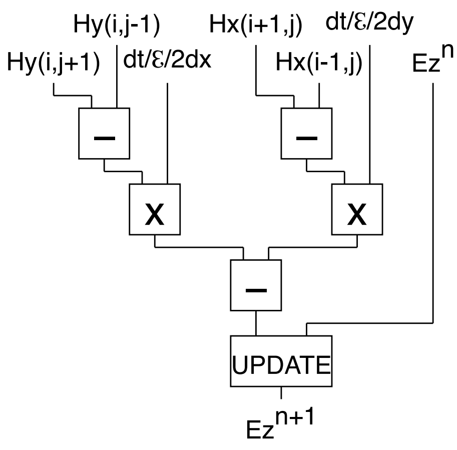

Fig. 4. Part of the arithmetic unit. Computing one iteration step of the E_z variable.

The precision of the operators are overestimated, because preliminary simulations showed that less precision is enough, but the optimal bit width is not yet determined.

TABLE I. COMPARISON OF DIFFERENT HARDWARE IMPLEMENTATIONS

Implementation	Speed (MHz)	Processor Cores (usable/all)	Computation time (sec)	Speedup
g++ compiled Intel Core I7	2800	1 (4)	7.5	0.37
g++ compiled AMD Opteron	2300	1 (4)	14.85	0.185
icc compiled AMD Opteron	2300	4	2.75	1
Xilinx XC7Z020	289	3	0.58	4.77
Xilinx XC6VSX475T	380	20	0.066	41.8
Xilinx XC7VX980T	380	42	0.031	87.78

TABLE II. NUMBER OF RESOURCES IN DIFFERENT FPGAS

Resources	# of slices	# of BRAM 36k	# of DSP48E
Xilinx XC7Z020	13300	140	220
Xilinx XC6VSX475T	74400	1064	2016
Xilinx XC7VX980T	153000	1500	3600

A large amount of space can be saved if the excitation function is periodic. Other trick can be applied if we choose Δt or other parameters to be integer power of two because the multiplication in this case can be done by shifts so we can eliminate several multipliers from the hardware and additionally the area requirements will be greatly reduced.

VI. RESULTS AND PERFORMANCE

By using the BlockRAMs of the FPGA, a 200×200 large grid array can be computed. With the OCM (On-Chip Memory) supplemented a bit larger 256×256 array can be updated. By communicating with the 512MB on-board memory (and sacrificing from the speed of the architecture by the memory transaction) it is possible to work with a 6600×6600 large computing space. In this case storing a 6600 wide belt from the image requires 40 BlockRAMs from the FPGA chip resource. That means 3 processor can be implemented on the ZedBoard which can work on successive iterations in one clock cycle.

For testing and performance evaluation purposes a simple initial setup was used. The size of the computational domain is 50×200, there are 20 excitation points which are excited with a sinusoidal function and the iteration number is 50000. Experimental results were compared on different conventional microprocessors (with different compilers) and Xilinx FPGAs and runtimes are summarized in Table I. The achievable performance of the different FPGAs is compared to the performance of the AMD Quad-Core 2.3GHz Opteron processor. Comparison of the required computation time with the above mentioned initial setup show that the computations can be carried out almost 5 times faster with the ZedBoard. Significant saving on computation time can be achieved if larger FPGA is used. Theoretical performance of the largest Virtex 7 FPGA is 87-time faster than the AMD Opteron microprocessor.

The number of resources in different FPGAs are shown in Table II. In this particular problem, the limitations in the FPGA is not the dedicated elements (multiplier, memory) but the number of slices.

VII. CONCLUSION

This paper demonstrates how (1) numerical solutions of the wave equation can be used to power a non-Boolean pattern matching algorithm and (2) how FPGA-based cellular solvers can be used to generate efficient, massively parallel solutions to the wave equations. This offers a new approach to utilize the computing power of large-scale FPGAs or mega-core chips.

We plan to extend the implemented architecture with more precise discretization scheme and with a more complex boundary condition. We also want to investigate other non-Boolean algorithms, and simulate them with FPGA architectures.

ACKNOWLEDGMENT

This research project supported by the OTKA Grant No. K84267 and TAMOP - 4.2.2/B-10 and TAMOP - 4.2.1 B-11. This work was also (partially) supported by the NRI-NSF Nanoelectronics for 2020 and Beyond (NEB) program.

REFERENCES

[1] S. Borkar: Thousand core chips: a technology perspective; Proceedings of the 44th annual Design Automation Conference, 2007

[2] D. Yeh, L.-S. Peh, S. Borkar, J. Darringer, A. Agarwal, W.-M Hwu, "Thousand-Core Chips," IEEE Design and Test of Computers, vol. 25, no. 3, pp. 272-278, May 2008, doi:10.1109/MDT.2008.85

[3] P. Ambs: Optical Computing: A 60-Year Adventure, Advances in Optical Technologies Volume 2010, Article ID 372652

[4] D. G. Feitelson: Optical computing : a survey for computer scientists, MIT Press, 1988

[5] E. M. Leith: The Evolution of Information Optics, IEEE Journal of Selected Topics on Quantum Electronics, 6, 6 2000

[6] J.-P. Berenger, "A perfectly matched layer for the absorption of electromagnetic waves." Journal of computational physics 114, no. 2 (1994): 185-200.

[7] T. Roska, L. O. Chua, D. Wolf, T. Kozek, R. Tetzlaff, F. Puffer: Simulating nonlinear waves and partial differential equations via CNN-Part I: Basic techniques, IEEE Transaction on Circuits and Systems-I, vol. 42, pp. 807-815, 1995.

[8] J. W. Goodman: Introduction to Fourier optics New York : McGraw-Hill 1996

[9] Z. Nagy, P. Szolgay: Configurable Multilayer CNN-UM Emulator on FPGA, IEEE Trans. Circuits and Systems I: Fundamental Theory and Applications, 50 6 2003

[10] G. Csaba, X. He, X. Sharon Hu, A. Papp, Wolfgang Porod: Holographic Algorithms for on-Chip, Non-Boolean Computing to be submitted to IEEE Nano 2013

[11] Xilinx Inc. [Online] http://www.xilinx.com/ Urldate 2013.

[12] ARM AMBA System IP [Online] http://www.arm.com/products/system-ip/amba/ Urldate 2013.