A Standard Cell Hardware Implementation for Finite-Difference Time Domain (FDTD) Calculation

L. Verducci (*), P. Placidi (*), P. Ciampolini (**), A. Scorzoni (*), and L. Roselli (*).

(*) DIEI, University of Perugia, via G. Duranti 93, 06125 Perugia (Italy),
Tel: +39-075-5853644, Fax: +39-075-5853654, E-mail: *verducci@diei.unipg.it.*

(**) DII, University of Parma, Parco Area delle Scienze 181/A, 43100 Parma (Italy).

Abstract — **Several inherent characteristics make the Finite - Difference Time Domain (FDTD) algorithm almost ideal for the analysis of a wide class of microwave and high-frequency circuits as testified by the great number of papers appeared in the last two decades and by the presence of many software packages on the present market. The application of the FDTD method to practical, three-dimensional problems, however, is often limited by the demand of very large computational resources. In this paper, the architecture of a digital system, dedicated to the solution of the 3D FDTD algorithm and based on a custom VLSI chip, which implements the "field-update" engine, is described. The system is conceived as a PCB module communicating with a host personal computer via a PCI bus and accommodating dedicated synchronous DRAM banks as well. Expectations are that significant speed-up, with respect to state-of-the-art software implementations of the FDTD algorithm, can be achieved.**

I. Introduction

The Finite – Difference Time Domain (FDTD) method has become one of the most widely used computational techniques for the full-wave analysis of electromagnetic phenomena [1] – [2]. However, its application to practical three-dimensional problems, is often limited by the demand of very large computational resources. Moreover, the actual performance which can be achieved by means of high-level programming is very much dependent on the operating system, the language compiler and the program structure [3].

Thanks to microelectronic technologies inexpensive RAMs and high-speed processors have become available allowing engineers to face practical applications of the FDTD method. In fact, the FDTD algorithm, taking advantage of simplicity and symmetry of discretized Maxwell's equations, exhibits some features which make a hardware implementation appealing. A VLSI based architecture has been proposed in [4] and in this study the feasibility of a chip design which would directly and exclusively compute the FDTD cell field expression has been demonstrated. Another approach has been reported in [5] for the one-dimensional FDTD algorithm. The computational speed is extremely high and not related to the number of the cells. Nevertheless the presented design is not economically or technologically feasible on a large scale.

In this paper, we describe a custom VLSI chip implementing the "field-update" engine of the FDTD algorithm on a 3D domain. This system is conceived as a module communicating with a host personal computer via a PCI bus. As described in Section 2, a virtual implementation of the system has been carried out by specifying the whole architecture through the VHDL [6] hardware description language: in particular, the physical implementation of the custom Floating Point Unit (FPU) has been carried out by mapping the network on a commercial technology. Finally, Section 3 describes how the adopted standard cell design allows for reliable timing characterization of the circuit.

II. FDTD Algorithm Implementation

Maxwell's equations in an isotropic medium can be written as follows:

$$\begin{aligned} &\frac{\partial \vec{B}}{\partial t}+\nabla\times\vec{E}=0 \\ &\frac{\partial \vec{D}}{\partial t}-\nabla\times\vec{H}=-\vec{J} \\ &\vec{B}=\mu\vec{H} \\ &\vec{D}=\varepsilon\vec{E} \end{aligned} \qquad (1)$$

where the symbols have their usual meaning, and J, μ, and ε, are assumed to be a given functions of space and time. According to the FDTD discretization method, the propagation domain is divided into cells, each cell being independently characterized in terms of material properties. Electromagnetic field components are mapped at cell edges as shown in Fig. 1, and Eqs. (1) are

0-7803-7695-1/03/$17.00 © 2003 IEEE

discretized, in both time and space, at each cell, using the (central) finite difference method.

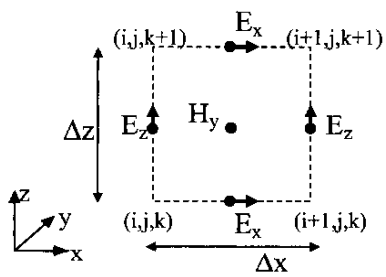

Fig. 1. Mapping of field components on Yee's mesh.

Referring to a given electric-field component (for instance E_x) the update equation reads:

$$E_x\Big|_{i+\frac{1}{2},j,k}^{n+1} = E_x\Big|_{i+\frac{1}{2},j,k}^{n} + k_1\left(H_z\Big|_{i+\frac{1}{2},j+\frac{1}{2},k}^{n+\frac{1}{2}} - H_z\Big|_{i+\frac{1}{2},j-\frac{1}{2},k}^{n+\frac{1}{2}}\right) + k_2\left(H_y\Big|_{i+\frac{1}{2},j,k-\frac{1}{2}}^{n+\frac{1}{2}} - H_y\Big|_{i+\frac{1}{2},j,k+\frac{1}{2}}^{n+\frac{1}{2}}\right) \quad (2)$$

Here, the standard Yee's [1] notation has been adopted.

In Eq. (2), k_1 and k_2 depend only on the actual material, on the discretization mesh-size and on the time step adopted.

The same discretization procedure is applied to all components of magnetic and electric field at each cell, this eventually resulting in a large set of algebraic linear equations. However, thanks to the so-called "leapfrog" scheme, a computationally-effective way to face the solution of such an algebraic system can be found, which makes inversion of huge, sparse matrices unnecessary.

Basically, such a scheme involves evaluation of time derivatives of electric and magnetic-field components at alternate time intervals: according to such a scheme, field updates depend only on quantities computed at previous time iterations, so that update equations pertaining to a given time step can be decoupled and independently solved (i.e., sequentially, in any arbitrary order). Six scalar equations per cell are eventually obtained, which share the same structure:

$$y = a + k_1(b-c) + k_2(d-e) \quad (3)$$

In the above equation, y represents the field component to be updated, a is its current value (i.e., that computed at the previous time step), b, c, d and e are known field components (i.e., previously computed at neighboring grid points).

II. System Architecture Description

A high level architecture block scheme of the designed system is reported in Fig. 2.

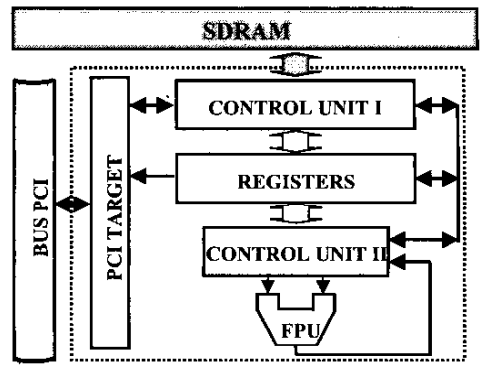

Fig. 2. High level system block scheme.

The most important blocks are: a Floating Point Unit (FPU), some internal register banks and a distributed control unit. The architecture is conceived with the goal of maximizing the exploitation of the FDTD symmetries. To achieve such goal, a custom organization of the data stored in the SDRAM is necessary. Referring to the space coordinates adopted in Fig.1, all the field components (e.g. E_x) sharing X and Z coordinates are stored in the same SDRAM row. With such data organization it is possible to implement SDRAM burst read and write operations, allowing to optimize FPU pipeline performance. Since the FDTD algorithm can be applied to very big discretized volumes (more than 10^6 cells in some cases), several SDRAM banks are necessary to store all needed data. In our system, each SDRAM is assigned to a specific kind of data (exactly x, y and z electromagnetic field components, material related coefficients, electromagnetic sources, SDRAM pointers). Every SDRAM module communicate with the core system by a dedicated data and address bus, bringing the total number of used busses to six [7].

Data flow between SDRAM and FPU during field update cycles is managed by a distributed control unit. In order to achieve maximum parallelism, each SDRAM is managed by an independent control unit sub-block.

System functionality will be described referring to its behavior in a practical example, considering the updating procedure of an E_x component burst. All inherent data flow will be discussed step by step.

First of all, SDRAM burst addresses must be provided from the SDRAM POINTERS control to all the other SDRAM control sub-blocks. Therefore, 1 or 2 of 32-bit addresses (depending on the field component to be updated) are read from SDRAM pointers and then stored

in some dedicated SDRAM control registers (R_1, R_2, R_3, R_4 in Fig.3).

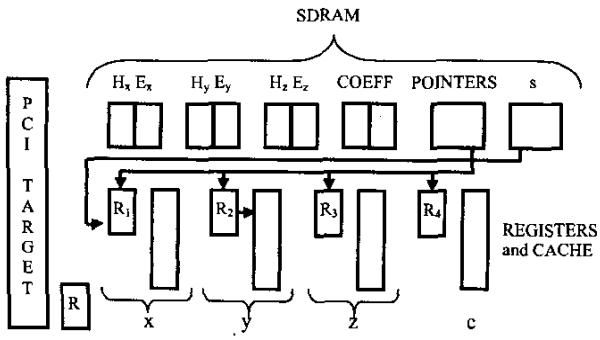

Fig. 3. Communication of burst addresses from POINTER control.

Before requesting burst addresses, the X control checks if there are any stimulus or observation points within the burst to be updated. At the same time, the Y, Z and COEFF controls execute their first burst read operations and store the burst in their own register bank memory (REGISTER and CACHE in Fig.3). After checking stimulus and observation points, also the X control executes a burst read operation and stores the burst in its registers bank (Fig. 4).

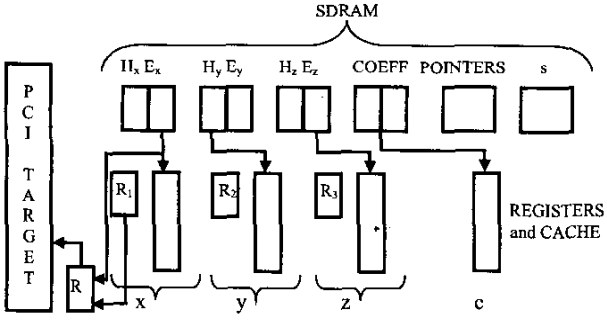

Fig. 4. First data burst read operation.

Then the Y, Z and COEFF controls execute the second SDRAM burst read and provide the FPU inputs with these data. The latter reach the inputs of the FPU aligned with the data stored in the internal registers during the previous read operation. While the FPU is evaluating the updated E_x burst, the X control stores these results in the same address where the burst was first read. Should any stimulus point be present during updated field burst storing, a new excitation field value will take its place (Fig. 5).

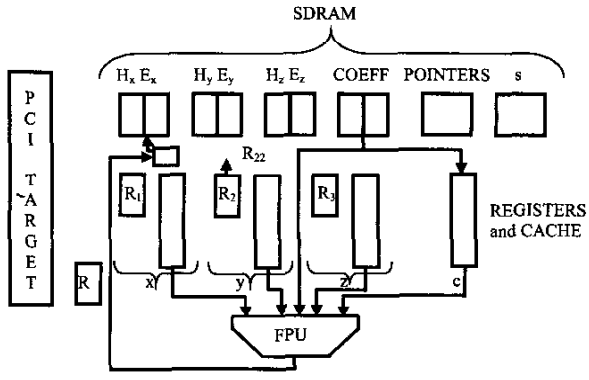

Fig. 5. Updated burst calculation and its storage

At the end, the X, Y, Z and COEFF controls ask for information to another block (called UPDATING control) about the next operation to be executed. At this point all the operations restart repeatedly for a new burst updating.

III. Simulation Results

System architecture design has been developed using Register Transfer Level (RTL) Hardware Description Language (VHDL).

The whole system has been tested by means of simulations performed with a standard commercial VHDL simulator. SDRAM VHDL models are provided by the SDRAM vendor. To verify the behavior of the overall system, test bench vectors have been produced. In fact, during the initialization phase of the algorithm, all the SDRAM banks have to be filled with data organized as described before.

To validate the accuracy of the results supplied by the system, a comparison with a MATLAB software simulator has been carried out. An electric resonator with a discretized structure of 5x5x10 cells including a point of observation has been simulated. It should be underlined that the dimension of the simulated structure is not limited by the characteristic of the system. In fact, with the SDRAM used, it is possible to simulate a structure featuring up to 80x80x128 cells. Furthermore, it is possible to interface the system with double capacity memories keeping the system almost unchanged (only 1 memory address for every SDRAM should be added up).

In Fig. 6 the simulation results for the E_x field component at coordinates (3,3,6) have been reported.

About 160 time steps have been simulated. System results perfectly match those obtained with the MATLAB simulator.

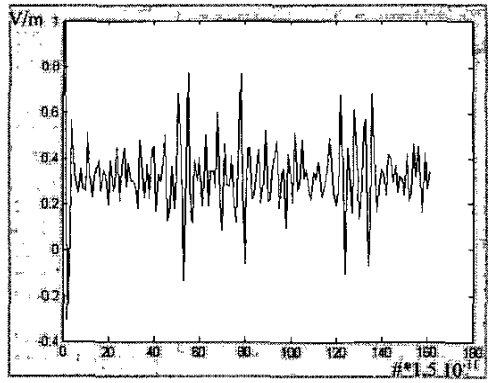

Fig. 6. Time evolution of the E_x field component amplitude.

Time performance has been estimated by looking at the time required to compute a given amount of time iterations.

For instance, using an Intel Pentium IV Personal Computer equipped with 1Gb SDRAM, a MATLAB simulation featuring 30000 time steps takes 135 s. From the system simulations results, 1 time step is computed in 40 μs, so the total simulation time for 30000 time steps is 1.2 s. It is important to point out that the MATLAB simulation time is very much dependent on the characteristics of the Personal computer, the kind of simulator and the operating system. In [7] the comparison with a FORTRAN simulation resulted in a time saving factor of about 5. In the present work that estimation has been substantially confirmed. An interesting feature of the designed system is that, when simulating a 5x5x20 cells structure, the simulations time has a 20% increase only, to be compared with a two fold increase of a software simulator. This is because the FPU pipeline achieves its best yield with long burst data. Another very important system feature is that just using higher system clock (i.e. using for example DDR SDRAM memories) system performances dramatically increases This is not the same for commercial general purpose systems (doubling system clocks does not halve the FDTD simulations time).

IV .CONCLUSIONS

In this article, a custom architecture for the FDTD algorithm has been proposed. All the system is described by means of RTL VHDL language, thus making the whole design (synthesis, placement and routing) independent of the chosen digital hardware. During the development of the VHDL system, in order to verify the quality of the RTL code, digital synthesis has been performed for every system block, obtaining satisfactory results with UMC VST 0.18μm technology and even with UMC VST 0.25μm. By adopting a standard cell design style, reliable timing characterization of the circuit can be obtained by means of simulation, and the overall performance of the system can be thus estimated. The system clock is set at 200 Mhz (the same of the SDRAM block, which represents the bottleneck of the system). Behavioral simulations have been issued in order to evaluate the system performance and to compare it with software FDTD simulators. Results obtained are very encouraging. The system provides results with the same accuracy but much faster than standard software simulators.

In light of the obtained results, custom hardware for the solution of FDTD algorithm appears to be an appealing approach in order to keep costs reasonably low and to achieve a good increase in terms of time saving.

ACKNOWLEDGEMENT

The authors would like to thank Marco Picchiarelli and Roberto Vincenti Gatti, Dipartimento di Ingegneria Elettronica e dell'Informazione (DIEI), University of Perugia (Italy) for their support and helpful discussions.

REFERENCES

[1] K. S. Yee, "Numerical solution of initial boundary value problems involving Maxwell's equation in isotropic media", *IEEE Trans. Antennas Propagat.*, vol. 14, pp.302-307, 1966.

[2] P. Ciampolini, P. Mezzanotte, L. Roselli, and R. Sorrentino, "Efficient simulation of high-speed digital circuits using time adaptive FD-TD technique", *25th European Microwave Conference*, (Bologna, IT), pp.636-640, sep 1995.

[3] S. D. Gedney, "Finite-Difference Time-Domain Analysis of Microwave Circuit Device on High Performance Vector/Parallel Computers", *Time-Domain Methods For Microwave Structures, IEEE Press*, pp.344-344, 1997.

[4] J. R. Marek, M. A Mealic, A. J. Terzuoli, "A Dedicated VLSI Architecture for Finite-Difference Time Domain", *8th Annual Review of Progress in Applied Computational Electromagnetic*, March 16-20, 1992, Monterey (CA).

[5] R. N. Schneider, M. M. Okoniewski, L. E. Turner, "Custom Hardware Implementation of the Finite-Difference Time-Domain (FDTD) Method", *International Microwave Symposium 2002*, June 2-7 2002, Scattle

[6] "IEEE Standard VHDL Reference Manual", *IEEE Standard 1076, 1987.*

[7] P. Placidi, L. Verducci, G. Matrella, L. Roselli and P. Ciampolini., "A custom VLSI architecture for the solution of FDTD equations", *IEICE Transactions*, VOL-E85, No.3, March 2002.