International Journal of Computer Engineering and Information Technology

VOL. 9, NO. 9, September 2017, 198–202
Available online at: www.ijceit.org
E-ISSN 2412-8856 (Online)

Implementation of the Bio Heat Transfer Equation on BEECube FPGA Platform

Idir Mellal[1], Aziz Oukaira[2], Emmanuel Kengne[3] and Ahmed Lakhssassi[4]

[1, 2, 3, 4] LIMA Laboratory, University of Quebec at Outaouais, Gatineau, Quebec, 18X 3X7, Canada

[1]idir.mellal@gmail.com

ABSTRACT

The Hardware Implementation of the physical models offers an outstanding opportunity for engineers in computational computing techniques. Contrary to the software implementation, the physical hardware implementations present the advantage of speed up the computation with inexpensive and practical way. The Finite Difference Method (FDM) is one the most common numerical method used to solve Electromagnetic and Heat transfer problems. In this paper, we present a FPGA-based implementation of the Bio Heat Equation (BHT). We then show the architecture of the proposed model to be implemented in the FPGA platform BEECube. The results of the implementation and simulation are reported and discussed.

Keywords: *FPGA, BEECube, Bio Heat Equation, FDM.*

1. INTRODUCTION

The Hardware Implementation of the physical models offers significant opportunity for engineers. Contrary to the software implementation, the Hardware models present real-time answers and faster response. It can speed up the computation of different algorithms by two digits. However, the massive resources needed for computing simultaneously all the nodes are the biggest challenge for hardware designers. In fact, the management of the memories communication and the enormous number of operations simultaneously presented a big challenge for the emergence of this technique [1].

In the few last decades, with the technological boom, many devices with high performance and small size have seen the day. Among these devices, the Field Programmable Gate Array (FPGA) which is a re-programmable device equipped with a different I/O puts and other internal and external devices. We can find an on-chip memories, registers and arithmetic modules that can be used to shorten the time of computation. Also, some external components, off-chip components integrated on the board and connected to the chip for eventual use. To speed up the computation and take advantages of the computational techniques, researchers have developed many computational platforms. The BEECube is one theses platforms used for parallel and real-time computing. Also, it is designed for accelerated computation, emulation, and prototyping of systems [2].

The BHT equation governing the temperature diffusion in biological tissue was proposed by Pennes [3]. It has been modified and improved by many researchers [4]. For x ϵ Ω and t > 0, the Pennes equation is presented as follow:

$$\rho c \frac{\partial T}{\partial t} = k\Delta T - \rho_b c_b \omega (T - T_a) + Q_m + Q_r \quad (1)$$

where ρ, c, and k are, respectively, the density (kg/m3), the specific heat (J/kg.K), and the thermal conductivity (W/m.k) of the tissue. T is the local tissue temperature T(x, t) in (K) with 0≤x≤L. x=0 corresponds to the skin surface, and x=L is the inner boundary. Ta the arterial blood temperature (K). ρ_b and c_b are the density and specific heat of the blood, respectively. The perfusion rate is represented by ω (l/s). Qm is the metabolic heat production per volume (W/m3). Qr stands for the deposited energy per volume. The left-hand side of Eq. (1) refers to the stored energy. The first term on the right-hand side represents the energy diffusion within the tissue; the second term describes the temperature exchange between the blood and the surrounding tissue, due to blood convection.

The initial and boundary conditions governing the equation are given as:

The initial condition:

$$T(x,t)\big|_{t=0} = 37°C \quad (2)$$

The boundary conditions:

To simplify the architecture of the model for hardware implementation, we suggest that the temperature is constant at x=0.

$$T(x,t)\big|_{x=0} = 37°C \quad (3)$$

For x=L, we assumed that the temperature is equal to the adjacent point. The condition chosen will be translated numerically as a temperature shift of the adjacent point of x=L:

$$\left.\frac{\partial T(x,t)}{\partial x}\right|_{x=L} = 0 \qquad (4)$$

2. METHODOLOGY

2.1 FDM Approximation of the BHT equation

The FDM was proposed for the first time by Yee in 1969 [5] for Electromagnetic computation. His paper did not have a significant impact till the beginning of the technological boom in computer speed and memory capacity. The FDM requires a lot of parallel computing and a large memory space. Therefore, the FDM has seen its vast utilization after the 1990s with the technological advances [6-8]. To implement the model on the silicon, we must discretize the 1D Pennes' equation. We firstly discretize the main equation (1), then we go ahead with the initial condition (2), and the boundary conditions (3-4). The proposed approximation in this work is based on one-dimension (1D) Pennes equation (1) for Ω = [0, L]. We used the Finite Difference Method- Forward Time Centered Space (FDM-FTCS) method to approximate the solution. The initial value (2) and the boundary conditions (3-4) are also discretized. To realize the space mesh, we used Δx in a way that L= M*Δx where "M" is an integer chosen according to the size of the space mesh and the accuracy wanted. To describe any point, we use the symbol "i" with $0 \le i \le M$. For the time meshing, we used "n" to define any time state with $0 \le n \le N$ where "N" is an integer which refers to the end of the loop. If we consider tf the final time (end of the loop) then $t_f = N*\Delta t$. Fig. 1 shows a 1D FDM representation. Fig. 2 shows the structure of the structural diagram.

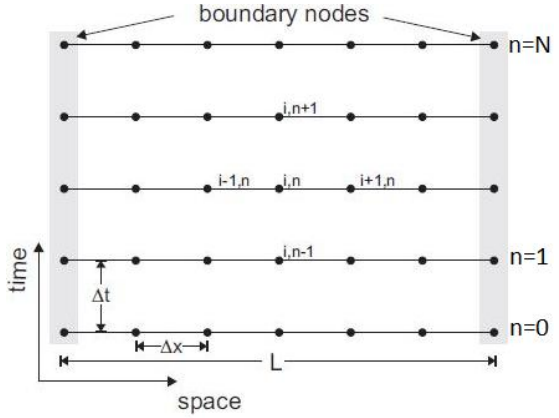

*Fig. 1. Proposed beam former. 1D FDM time-space meshing with L= M*Δx and $t_f = N*\Delta t$. The point (i, n) and its four neighbors represented.*

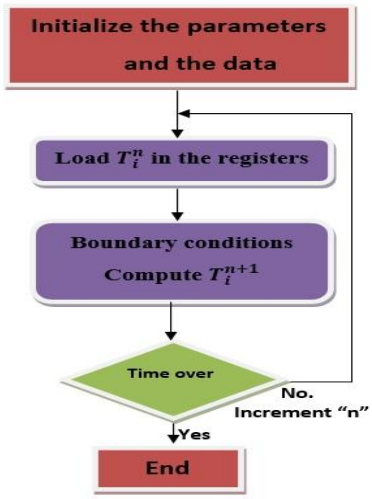

Fig. 2 Structural diagram of 1D FDM. At each step 'n', the core computes the value of the "i+1" nodes. The values of the neighbors of the point 'i' at the previous state 'n-1' are stored in the registers and used to update each node. The process is stopped when the stop condition is reached.

Using the FDM-FTCS approximation of the Pennes equation, we can find the value of the temperature at the next state "n+1" at any point "i". The algorithm will use the known values of the same point "i" and its two direct neighbors ("i-1" and "i+1") at the present state "n". Equation (5) shows the value of the temperature of any point "i" at the next state (n+1).

$$T_i^{n+1} = aT_i^n + b(T_{i-1}^n + T_{i+1}^n) + \frac{c_b \rho_b w T_a}{\rho c}\Delta t + \frac{Q_m}{\rho c}\Delta t + \frac{Q_r}{\rho c}\Delta t \qquad (5)$$

where

$$a = 1 - \frac{2k}{\rho c}\frac{\Delta t}{\Delta x^2} - \frac{c_b \rho_b w}{\rho c}\Delta t$$

and

$$b = \frac{k}{\rho c}\frac{\Delta t}{\Delta x^2}$$

For the same, the boundaries conditions (BC) (3-4) were discretized.

The initial condition (2) governing the model implies that the temperature all over the geometry is constant and equals to 37^0C, T(x, t)|t=0= 37^0C. The discretization form is

$$T_i^0 = 37°C$$

The left boundary condition (3), x = 0, was taken as it is a constant T(x, t)|x=0=37^0C. Consequently, the discretized form was written as:

$$T_0^{n+1} = 37°C$$

The right boundary condition (4), x = L, was considered as like the previous temperature of the left neighbor of the last point "M". The reason to do this is to simplify the

hardware implementation. In fact, by using this condition, we will need just one register to implement the condition by doing a right shift of the temperature. Thus, the discretization of the condition (4) was given as:

$$T_M^{n+1} = T_{M-1}^n$$

2.2 Prototype Model

To verify the functionality and the feasibility of such an approach, we proposed a FPGA hardware implementation. The BEECube platform was chosen which is a multi FPGAs platform for hardware emulation and rapid prototyping. It includes four Xilinx Virtex-5 FPGAs interconnected with 16 GB DDR2 DRAM per FPGA and many other peripherals [2, 9]. The BEECube platform combines between Software and Hardware solution to have better performances for emulation and rapid prototyping of the most complex architectures. Many other platforms exist, and most of them are employed to prototype and realize a parallel computing with low cost, less time, and high performances. These platforms are highly used in biomedical science [1, 2, 10-12].

The hardware implementation of the FDM algorithm is a hot issue. In fact, because of the high parallelism of the FDM and the enormous number of computing operations, the hardware implementation of this kind of algorithm is so difficult to implement on a single chip. The management of memories and the updating of each point in the grid is a big challenge that reduces the speed of execution and increases the used resources. In consequence, the hardware implementation of the FDM was a tough task for many years. The first hardware implementation was realized by Marek and al. [13] in 1992. However, the lack of powerful computer and large memory limited the performance of his architecture. In 2002, Schneider and al. [14-16] and Placidi et al. [17] proved the feasibility and the efficiency of the hardware implementation of the FDM.

2.3 Presentation of the Platform

The BEECube platform is a multi FPGA platform designed for parallel computing and complex design prototyping and emulation. It is composed of a Hardware platform associated with a software infrastructure. The hardware platform includes all the four FPGAs Virtex-5 with 64 GB of DRAM, several I/O ports and other components. The software infrastructure includes distinct layers starting from the Operating System ACE that runs the platform to the gateware used to program the FPGAs. Fig. 3 shows the different blocks of the BEECube platforms.

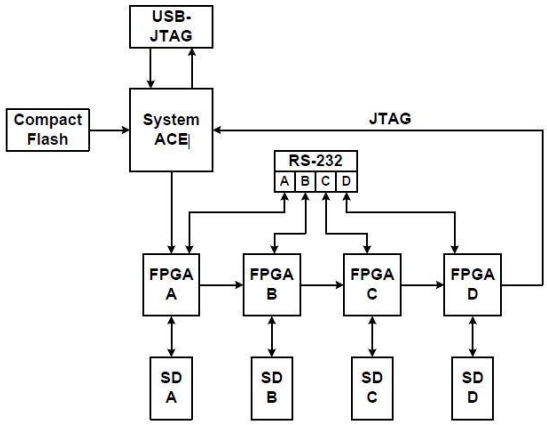

Fig. 3. The internal architecture of the BEEcube Platform.

Each FPGA is connected to a multitude of I/O ports, external memories, other blocks like an external clock. Fig. 4. shows the FPGA connections.

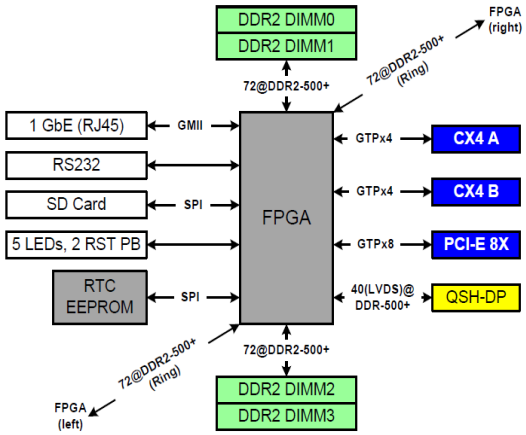

Fig. 4. FPGA Connections. Each chip can use the internal resources and the external resources available on the platform.

The BEECube Platform Studio (BPS) is hardware/software tool used by the BEECube platform to automatically accelerate the implementation by generating all the necessary files needed for a rapid and fruitful implementation. It combines the top-level System Generator of Xilinx ISE which uses Matlab Simulink and the BEECube software tools for custom designs. This combination approach has proved its success and efficiency for complex algorithms and parallel computation [2, 9]. The combination of the software libraries available from Xilinx in Matlab Simulink and the BPS customized components and the hardware component available on the platform is a big opportunity for complex designs. Fig. 5 shows a different state of a BEECube design and the interpellated tools.

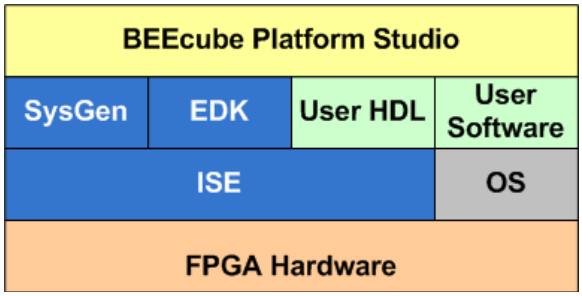

Fig. 5. Several blocks of the BEECube at a different level. Before BPS generates the bit file, many tools have been used following the usual step of the design flow.

The provided IP libraries by BPS and SysGen/Simulink make the tasks easy and less complex for designers. They can choose and change the design easily with more freedom and flexibility. The engineers can change or simplify the architecture in less time and less cost [9]. These libraries are composed of a different regular and customized component. Xilinx blocks offer many components for DSP, arithmetic, control and logic designs. Furthermore, BPS offers some essential tools for an optimal use the BEECube resources and optimizes the architecture. We can find a Share FIFO, a Shared BRAM, a Software registers and High-speed interfaces.

3. RESULTS AND DISCUSSION

3.1 The Hardware Implementation

The architecture proposed in this model uses a simple component. It contains among other adders, multipliers, and registers. The computation of each point is made by a computational module called Processing Element (PE). Each PE computes the values of a specific node "i". Fig. 6 shows the architecture of one PE.

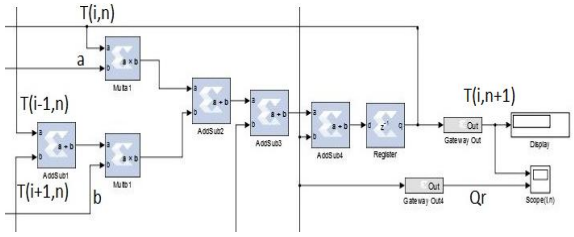

Fig. 6. PE's structural diagram of one PE. The value of T(i, n)

We can see how the PE uses the actual value of the point 'i' and the values of its neighbors 'i-1' and 'i+1' to compute the next value of the node 'i'.
The whole architecture is implemented on the BEECube. We divided the entire algorithm into four parts. Each one is implemented on a single FPGA. To update the peripheral nodes, each FPGA needs to communicate with its direct neighbors. So, each FPGA send the values of its peripheral node and receive the values of the neighbor nodes. Fig. 7. Shows the internal repartition and the blocks of the architecture.

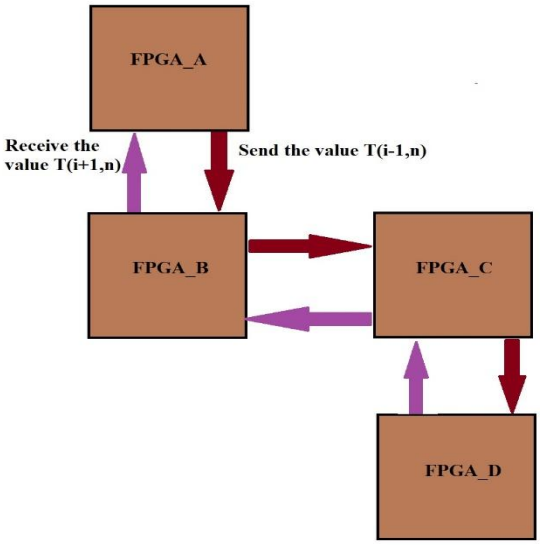

Fig. 7. Architecture and communication of the model in the BEECube platform.

3.2 Simulation and Results

The results of the simulation show the expected answer of the model. We used a pulse generator to produce a pulse as a heat source. The characteristics are inspired in the literature [18]. Figure. 8. shows the answer of the system for an impulse.

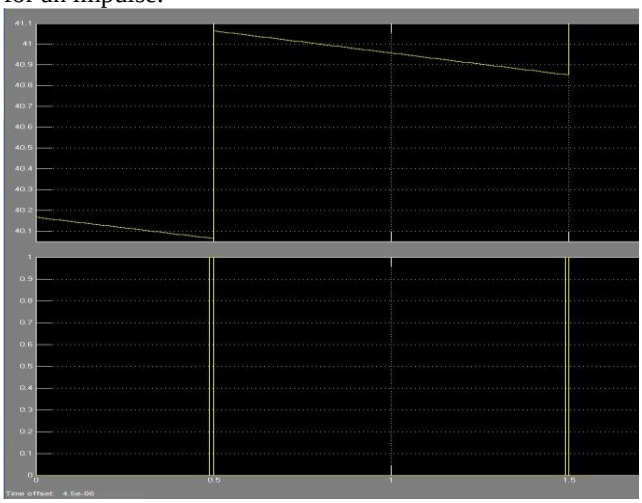

Fig. 8. Simulation of the model and results for successive pulses.

To program the FPGAs, we used a Xilinx iMPACT tool. We make an assignment for each FPGA device separately; then we program them. Figure. 9. Shows a view of the

tool with the four FPGAs and the associated memory to save the program.

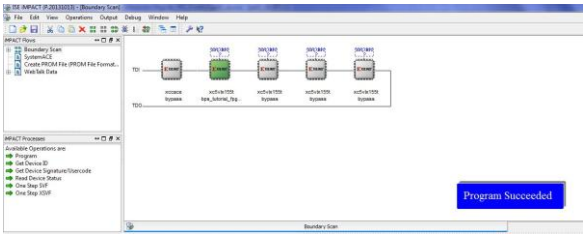

Fig. 9. Programming the fourth FPGA and downloading the Bit file with the ISE iMPACT tool.

Table 1 reports the consumption resources on the platforms.

Table 1: Margin specifications

Types	***Slices***	FFs	LUTs	IOBs	Mults
Numbers	5957	7508	10363	448	160

4. CONCLUSIONS

In this paper, we presented a hardware implementation of a 1D FDM algorithm of the BHT equation. We introduced the computation method FDM and the approximation of the equation. Then we exposed the platform and its various parts. We discussed the advantages of this kind of FPGA platforms for prototyping and emulation. Finally, we showed the architecture and the results.

REFERENCES

[1] G. Chrysos, E. Sotiriades, C. Rousopoulos, A. Dollas, A. Papadopoulos, I. Kirmitzoglou, V. Promponas, T. Theocharides, G. Petihakis, J. Lagnel, Opportunities from the use of FPGAs as platforms for bioinformatics algorithms, Bioinformatics & Bioengineering (BIBE), 2012 IEEE 12th International Conference on, IEEE, 2012, pp. 559-565.

[2] J.D. Davis, C.P. Thacker, C. Chang, C. Thacker, J. Davis, BEE3: Revitalizing computer architecture research, Microsoft Research, (2009).

[3] H.H. Pennes, Analysis of tissue and arterial blood temperatures in the resting human forearm, Journal of applied physiology, 1 (1948) 93-122.

[4] A. Lakhssassi, E. Kengne, H. Semmaoui, Modified pennes' equation modelling bio-heat transfer in living tissues: analytical and numerical analysis, Natural Science, 2 (2010) 1375.

[5] K. Yee, Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media, IEEE Transactions on antennas and propagation, 14 (1966) 302-307.

[6] U.S. Inan, R.A. Marshall, Numerical electromagnetics: the FDTD method, Cambridge University Press2011.

[7] K. Morton, D. Mayers, Numerical solution of partial differential equations, 1998.

[8] J.B. Schneider, Understanding the Finite-Difference Time-Domain Method, (2016).

[9] BEECube Platform Studio Version 4.0 User Manual.

[10] C. Chang, K. Camera, Exploring Field-Programmable Gate Array (FPGA)-Based Emulation Technologies for Accelerating Computer Architecture Development and Evaluation, DTIC Document, 2009.

[11] C. Chang, K. Kuusilinna, B. Richards, A. Chen, N. Chan, R.W. Brodersen, B. Nikolic, Rapid design and analysis of communication systems using the BEE hardware emulation environment, Rapid Systems Prototyping, 2003. Proceedings. 14th IEEE International Workshop on, IEEE, 2003, pp. 148-154.

[12] J. Shalf, D. Quinlan, C. Janssen, Rethinking hardware-software codesign for exascale systems, Computer, 44 (2011) 22-30.

[13] J. Marek, M. Mehalic, A. Terzuoli, A dedicated VLSI architecture for finite-difference time domain calculations, 8th Annual Review of Progress in Applied Computational Electromagmetics, (1992) 546-553.

[14] R.N. Schneider, M.M. Okoniewski, L.E. Turner, Custom hardware implementation of the finite-difference time-domain (FDTD) method, 2002 IEEE MTT-S International Microwave Symposium Digest (Cat. No.02CH37278), 2002, pp. 875-878 vol.872.

[15] R.N. Schneider, M.M. Okoniewski, L.E. Turner, A software-coupled 2D FDTD hardware accelerator [electromagnetic simulation], IEEE Antennas and Propagation Society Symposium, 2004., 2004, pp. 1692-1695 Vol.1692.

[16] R.N. Schneider, L.E. Turner, M.M. Okoniewski, Application of FPGA technology to accelerate the finite-difference time-domain (FDTD) method, Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays, ACM, 2002, pp. 97-105.

[17] P. Placidi, L. Verducci, G. Matrella, L. Roselli, P. Ciampolini, A custom VLSI architecture for the solution of FDTD equations, IEICE Transactions on Electronics, 85 (2002) 572-577.

[18] I. Mellal, E. Kengne, K. El Guemhioui, A. Lakhssassi, 3D Modeling Using the Finite Element Method for Directional Removal of a Cancerous Tumor, Journal of Biomedical Sciences, 5 (2016).