THE ANNALS OF "DUNĂREA DE JOS" UNIVERSITY OF GALATI
FASCICLE III, 2018, VOL. 41, NO.1, ISSN 2344-4738, ISSN-L 1221-454X
ELECTROTECHNICS, ELECTRONICS, AUTOMATIC CONTROL, INFORMATICS

HARDWARE IMPLEMENTATION OF A PIPELINED COMPUTATIONAL UNIT FOR N-BODY SIMULATION

Ovidiu Panait

Student, Department of Computer and Information Technology, The University "Dunărea de Jos" of Galati, Romania
ovpanait@gmail.com

Abstract: This paper presents a pipelined FPGA implementation of an engine that computes Newtonian gravitational forces. The module can be incorporated in a large-scale N-body simulation as the primary component used for computing the interaction between bodies. It uses 64-bit floating point arithmetic and relies on the speed provided by the "Fast Inverse Square" root algorithm. The design was implemented and tested on an Altera DE10-Lite FPGA.

Keywords: FPGA, pipeline, N-body simulation, floating point, reconfigurable platform

1. INTRODUCTION

An "N-body simulation" involves computing the evolution of a system composed of N bodies, where each object interacts continuously with the others. Such simulations have extensive use in various applications, such as the formation and evolution of planetary systems, protein folding, and computer graphics global illumination.

A specific type of N-body simulation is the "Newtonian N-body problem" which involves computing the positions of N bodies in space, at equal discrete time intervals, assuming that the bodies interact through gravitational forces only.

Due to the fact that this problem is very demanding in terms of computational resources and floating-point operations, custom computing machines and GPUs are more suitable for high-performance designs (Che *et al.*, 2008; Tsoi *et al.*, 2010; Jones *et al.*, 2010).

Extensive research has been done in the area of FPGA-based solutions for the astrophysical many-body simulation (Sano, *et al.,* 2017; Kim, *et al.,* 2007; Gothandaraman, *et al.,* 2006; Phillips, *et al.,* 2006; Lienhart, *et al.,* 2002). Custom boards such as the PROGRAPE-3 have reached speeds up to 40 GFLOPS (Nakasato and Hamada 2007), while maintaining the specialized pipeline structure flexible.

In the context of many-body simulations, Graphical Processing Units are also good candidates for implementing high-performance and efficient computations (Bédorf, *et al.,* 2012; Jetley *et al.,* 2010, Harris, 2005; Nyland, *et al.,* 2007; Hamada and Iitaka, 2007). In terms of processing power and cost, GPU-based platforms seem to outperform FPGAs, but the performance per Watt figure in O(N^2) gravitational N-body simulations on FPGA systems proved to be 15 times higher (Hamada, *et al.,* 2009).

The solution presented in this paper takes the custom hardware implementation approach on the FPGA,

making the design flexible, high-performance and at the same time keeping the power consumption low. The pipelined architecture computes the interaction between bodies with the increased accuracy provided by the IEEE 754 double-precision floating point operations and the speed of the "Fast inverse square root" algorithm. The implementation and testing was done on an Altera DE10-Lite hardware design platform build around the MAX 10 FPGA chip with 50K logical elements available.

2. IMPLEMENTATION

The force vector on body i, caused by the gravitational attraction of body j is

$$(1)\ f_{ij} = G\frac{m_i m_j}{\| r_{ij} \|^2} \cdot \frac{r_{ij}}{\| r_{ij} \|}$$

where m_i and m_j are the masses of body i and j and $r_{ij} = x_j - x_i$ is the vector from body i to body j.

According to the superposition principle, the total force F_i on body i, caused by the interaction with the other N – 1 bodies is

$$(2)\ F_i = \sum_{\substack{1 \le j \le N \\ j \ne i}} f_{ij}$$

The design assumes a 2D simulation, therefore the last stage of the pipeline will provide the X and Y components of the gravitational force:

$$(3)\ f_{ij_x} = G\frac{m_i m_j}{\sqrt{r^3}} \cdot (x_j - x_i)$$

$$(4)\ f_{ij_y} = G\frac{m_i m_j}{\sqrt{r^3}} \cdot (y_j - y_i)$$

Where (x_i, y_i) and (x_j, y_j) are the positions of the i and j bodies, and r is

$$(5)\ r = (x_j - x_i)^2 + (y_j - y_i)^2$$

These results could be subsequently used to compute the acceleration a_i needed to integrate over time and update the positions and velocities of particle I:

$$(6)\ a_i = \frac{F_i}{m_i}$$

where F_i is the total force F_i on body i, caused by the interaction with the other N – 1 bodies:

$$(7)\ F_i = Gm_i \sum_{1 \le j \le N} \frac{m_j r_{ij}}{\| r_{ij} \|^3}$$

2.1. Fast Inverse Square Root

A faster alternative to the computationally expensive floating point technique of computing the reciprocal of a square root is the "Fast inverse square root" algorithm (Lomont, 2003). It was originally designed for 32-bit floating point arithmetic, but the magic value for 64-bit floating point operations was later determined to be 0x5FE6EB50C7B537A9 (Robertson, 2009).

The algorithm starts with a very good first approximation of the inverse square root of the input, then it runs one iteration of Newton's method, to increase accuracy.

In this case, this algorithm is used to compute the inverse square root of r^3:

```
float Q_rsqrt(float r_cube)
        {
        threehalfs = 1.5F;
        x2 = r_cube * 0.5F;
        y  = 0x5FE6EB50C7B537A9 - (r_cube >> 1);
        y  = y * (threehalfs - ( x2 * y * y ));
        return y;
        }
```

2.2. FPGA implementation and testing

The force module was synthetized and tested on an Altera De10-Lite board, along with an UART module, for testing purposes. The design is clocked from the onboard 50MHz oscillator and makes use of approximately 27 thousand logical elements from the 50 thousand available.

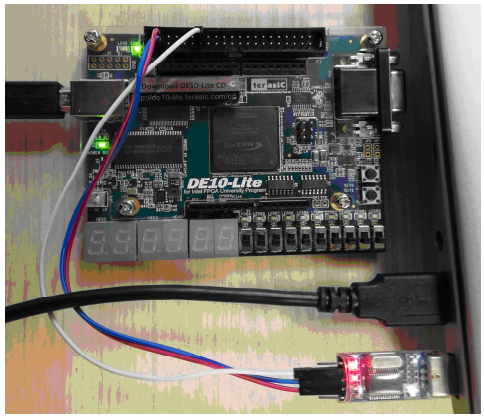

Fig.1. Hardware setup

Family	MAX 10
Device	10M50DAF484C6GES
Timing Models	Preliminary
Total logic elements	27,577 / 49,760 (55 %)
Total registers	2352
Total pins	49 / 360 (14 %)
Total virtual pins	0
Total memory bits	0 / 1,677,312 (0 %)
Embedded Multiplier 9-bit elements	198 / 288 (69 %)
Total PLLs	0 / 4 (0 %)
UFM blocks	0 / 1 (0 %)
ADC blocks	0 / 2 (0 %)

Fig.2. Resource utilization (Quartus II)

The force pipeline is fed through the serial port of a computer running Linux, with the particles' (x, y) positions in 64-bit floating point representation. The output values are the gravitational force components on the X, respectively Y axis, which are transferred from the FPGA back to the computer by the UART module. An USB to serial converter was used for the two-way communication.

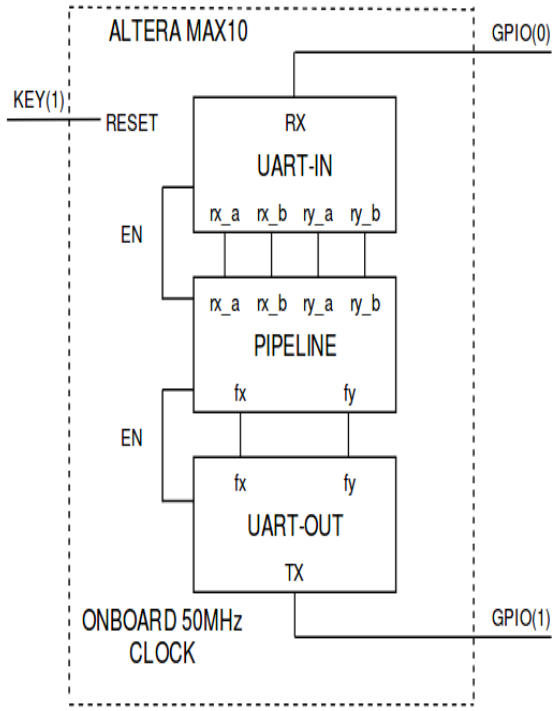

Fig.3. Test setup

64-bit floating point addition and multiplication blocks have been implemented in hardware.

The force pipeline consists of 10 stages, each stage being executed in one clock cycle and having its own multiplier/adder blocks incorporated. Because of the local buffering done by the registers present in each stage, the pipeline could be theoretically fed with a new particle at each clock cycle, without the risk of structural/data hazards.

In the current implementation, on-chip resources (embedded multipliers and logical elements) have been traded in favor of speed and accuracy. The high number of embedded multipliers used in the design (198 – 69% of the total available) is due to the increased accuracy provided by the IEEE 754 double-precision binary format. However, for some multi-body simulation applications, this level of precision may be overkill. Because the modules responsible for performing floating point arithmetic were written with re-usability in mind, using the VHDL "generic" parameters, much lower resource utilization could potentially be achieved, by reducing the mantissa size.

The test scenario involved transmitting the particles' positions from a computer, through a serial connection. However, in order to fully take advantage of the pipelined architecture, a capable particle controller should be synthesized alongside the force module. In a real-world large scale numerical N-body simulation, the controller should be able to feed the pipeline at a considerably high rate, which might pose some challenges considering the fact that the MAX10 FPGA chip is relatively small in terms of resources.

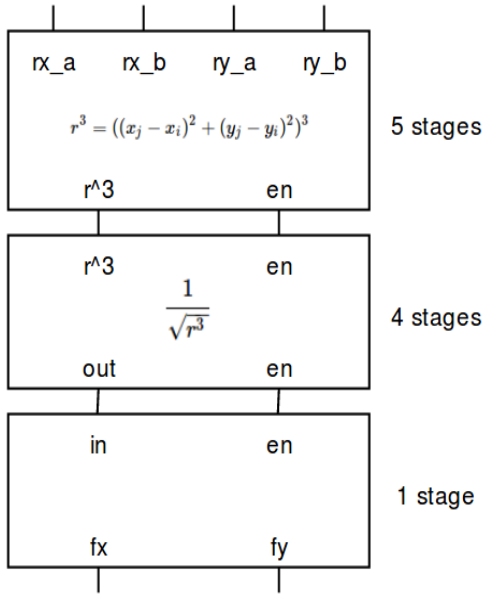

Fig.4. Pipeline block diagram.

The code was written in VHDL93, is open-source and can be found at

https://github.com/ovpanait/nbody-pipeline, on git branch PIPEF.

3. CONCLUSION

This paper presented a hardware implementation of a pipelined gravitational force engine on an Altera MAX10 based development board. This particular design proved that the "Fast inverse square root" algorithm and 64-bit floating point arithmetic blocks, along with a pipelined architecture provide a flexible, high-performance and highly accurate way to compute the interactions between objects in the context of many-body simulations.

4. REFERENCES

Bédorf, J., Gaburov, E., & Zwart, S. P. (2012). A sparse octree gravitational N-body code that runs entirely on the GPU processor. Journal of Computational Physics, 231(7), 2825-2839.

Che, S., Li, J., Sheaffer, J. W., Skadron, K., & Lach, J. (2008, June). Accelerating compute-intensive applications with GPUs and FPGAs. In Application Specific Processors, 2008. SASP 2008. Symposium on (pp. 101-107). IEEE.

Gothandaraman, A., Warren, G. L., Peterson, G. D., & Harrison, R. J. (2006, November). Reconfigurable accelerator for quantum Monte Carlo simulations in N-body systems. In Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, SC (p. 177).

Hamada, T., & Iitaka, T. (2007). The chamomile scheme: An optimized algorithm for n-body simulations on programmable graphics processing units. *arXiv preprint astro-ph/0703100*.

Hamada, T., Benkrid, K., Nitadori, K., & Taiji, M. (2009, July). A comparative study on ASIC, FPGAs, GPUs and general purpose processors in the O (N^ 2) gravitational N-body simulation. In *Adaptive Hardware and Systems, 2009. AHS 2009. NASA/ESA Conference on* (pp. 447-452). IEEE.

Harris, M. (2005). Gpgpu: General-purpose computation on gpus. *SIGGRAPH 2005 GPGPU COURSE*, 1-51.

Jetley, P., Wesolowski, L., Gioachin, F., Kalé, L. V., & Quinn, T. R. (2010, November). Scaling hierarchical N-body simulations on GPU clusters. In Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (pp. 1-11). IEEE Computer Society.

Jones, D. H., Powell, A., Bouganis, C. S., & Cheung, P. Y. (2010, August). GPU versus FPGA for high productivity computing. In Field Programmable Logic and Applications (FPL), 2010 International Conference on (pp. 119-124). IEEE.

Kim, J. S., Mangalagiri, P., Irick, K., Kandemir, M., Narayanan, V., Sobti, K., ... & Sun, X. (2007, August). TANOR: A tool for accelerating N-body simulations on reconfigurable platform. In *Field Programmable Logic and Applications, 2007. FPL 2007. International Conference on* (pp. 68-73). IEEE.

Lienhart, G., Kugel, A., & Manner, R. (2002). Using floating-point arithmetic on FPGAs to accelerate scientific N-body simulations. In *Field-Programmable Custom Computing Machines, 2002. Proceedings. 10th Annual IEEE Symposium on* (pp. 182-191). IEEE.

Lomont, C. (2003). Fast inverse square root. Tech-315 nical Report, 32.

Nakasato, Naohito & Hamada, Tsuyoshi. (2007). Astrophysical Simulations with Reconfigurable Hardware Accelerator. 347-351. 10.1007/978-4-431-49022-7_70.

Nyland, L., Harris, M., & Prins, J. (2007). Fast n-body simulation with cuda. *GPU gems*, *3*(1), 677-696.

Panait, O. (2018) Full project with source code: https://github.com/ovpanait/nbody-pipeline, accessed May, 2018.

Phillips, J., Areno, M., Eames, B., & Dasu, A. (2006). An fpga-based dynamic load-balancing processor architecture for solving n-body problems. In Proceedings of the 10th Annual High Performance Embedded Computing Workshop.

Robertson, M. (2012). *A brief history of invsqrt*. Department of Computer Science & Applied Statistics.

Sano, Kentaro & Abiko, Shin & Ueno, Tomohiro. (2017). FPGA-based Stream Computing for High-Performance N-Body Simulation using Floating-Point DSP Blocks. 10.1145/3120895.3120909.

Tsoi, K. H., & Luk, W. (2010, February). Axel: a heterogeneous cluster with FPGAs and GPUs. In Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays (pp. 115-124). ACM.