An approach to finding an optimal FPGA for memory intensive problems

Stefan Pijetlović, *Student Member, IEEE,* Miloš Subotić, *Student Member, IEEE,* Nebojša Pjevalica, *Member, IEEE*

***Abstract* — Memory intensive problems are a group of problems which require a lot more access and work with the memory in regard to computing problems. One example of such is a multidimensional (2D and 3D) FDTD algorithm which served as a motivation for this paper. To cope with the specifics of memory intensive problems, the most common solutions include invoking software parallelism and/or making improvements to the current hardware or creating a completely new one. The principle chosen by the authors was to use the programmability of an FPGA in order to achieve maximal memory bandwidth with a lower price and power consumption. To realize such a specific architecture an appropriate chip is required. The goal of this paper is to describe a script used to find an optimal FPGA based on the search criteria – highest achievable memory bandwidth per price.**

***Key words* — FPGA, FDTD, memory intensive problems, component selection, hardware design**

I. Introduction

FDTD (Finite-difference time-domain) is a powerful algorithm for the modeling of electromagnetic field. It provides a direct time-domain solution of Maxwell's Equations in differential form by discretizing both the physical region and time interval using a uniform grid. Even though the method was introduced in the year 1966 [1], it was not widely used until the last decade due to limited computing resources [2]. Its areas of application include mine detection [3], microwave tomography [4], electromagnetic compatibility, antenna design and more. FDTD is a memory intensive problem and as such has sparked an interest to many authors in the past. To improve overall performance, the algorithm has already been successfully implemented on multi-core architectures [5]. Additional improvements were made by using GPUs [6] for computing, adding additional FPGAs into the mix as well as using fixed point arithmetic [7].

One of the issues with these platforms is that they are compute oriented – they can do multiple computing operations per one memory load resulting in a lot of time spent idle waiting for the much slower memory. The amount of data required for FDTD is huge, so misses in the cache are bound to happen because the entire problem simply cannot efficiently fit into the cache of existing architectures.

The idea behind this paper is to find an alternative efficient solution with minimized time spent idle during the computation process. Less idle time means less power consumption and further reduction of the overall cost. Existing available architectures are predominantly compute oriented so when coping with memory intensive problems, their full potential is quite suppressed. On the other hand, FPGAs could be utilized and configured in a "smart" way which could ensure better memory management.

As far as pricing goes, there are a lot of possible options on the market. What is certain is that both the CPUs and GPUs require more silicon surface. Also, when taking into consideration the long term costs, the CPUs and GPUs generally use much more power than an FPGA. In case of a dedicated hardware, after a period of several years the costs of electric power are no longer insignificant.

II. Solution

Memory bandwidth is the rate at which data can be read or stored by the processor from/to the memory, usually measured in bytes per second. It is the product of the following parameters: memory clock frequency, number of transfers per clock, width of the bus and the number of buses. For example, a standard DDR3 DIMM (dual in-line memory module) has the following characteristics: I/O bus clock ranging from 400 to 1066 MHz, 2 transfers per clock (one on the rising and one on the falling edge), has interface width of 8 bytes and has 1 interface, achieving maximum theoretical bandwidth ranging from 6.4 to over 17 GB/s. For the purpose of this paper, only DDR3 DIMMs and DDR3 chips were taken into consideration. The main reason behind this decision is the lack of support for DDR4 modules (on the FPGA side) as well as better performance of DDR3 modules compared to DDR2. As a side note, DDR3 DIMMs are preferable due to their advantage of being ubiquitous and easy accessible on the market when compared to DDR3 chips and SRAM. The challenge was finding an FPGA which could make use of the biggest amount of DDR3 DIMMs while at the same time costing the least amount of money.

The starting point for solving this problem was using Octopart [8], a search engine for various electronic components. Its database consists of hundreds of distributors, thousands of manufacturers as well as a huge number of available components. The search results for the query FPGA contains over 300.000 results. Even for a rather specific query such as Kintex (one of the FPGA chip families), there are

Stefan Pijetlović – RT-RK Institute for Computer Based Systems, Bajči Žilinskog bb, 21000 Novi Sad, Serbia (e-mail: stefan.pijetlovic@rt-rk.com).

Miloš Subotić – RT-RK Institute for Computer Based Systems, Radnička 30, 21000 Novi Sad, Serbia (e-mail: milos.subotic@rt-rk.com).

Nebojša Pjevalica – University of Novi Sad, Faculty of Technical Sciences, Computing and Control Engineering Dept. Trg Dositeja Obradovića 6, 21000 Novi Sad, Serbia (e-mail: pjeva@uns.ac.rs).

several thousand components. Not the mention that the variations in the price range as well as performance and availability are also huge.

When choosing a component, there are certain criteria which are more important than the others. Considering memory intensive problems, of highest interest is the maximum achievable memory bandwidth per price. This parameter can be obtained by counting the number of pins which could be connected to a DDR3 DIMM, taking into account their speed and their layout as well. The goal is to find a chip with the best ratio of price and matching pins. For example, a certain chip can be cheaper but have only enough pins for 1 module, however if there is a chip which has enough pins for 2 modules and does not cost twice as much, its ratio is better.

For a group of pins to be compatible with the DDR3 module, certain conditions have to be fulfilled. 10 pins are needed per 1 byte group (8 are used for 8 data bits, and 2 are used for data strobe signals). Each bank should have 4 byte groups, and there should be the highest number of banks possible in the same column (to be on the same side of the FPGA). 3 byte groups are needed for address and control signals, disregarding the data width, so 1 bank is enough for 1 byte of data. 2 banks allow 4 bytes of transfer (leaving 1 bank unused) because the 3 byte groups from bank 1 can be used to address the data from this bank as well. 3 banks allow 8 bytes which is desired due to the matching interface width with the standard DDR3 DIMM. This is shown in figure 1. The Xilinx software does not allow connecting banks from different columns to the same DDR3 DIMM. This is related to the routing problems which can occur.

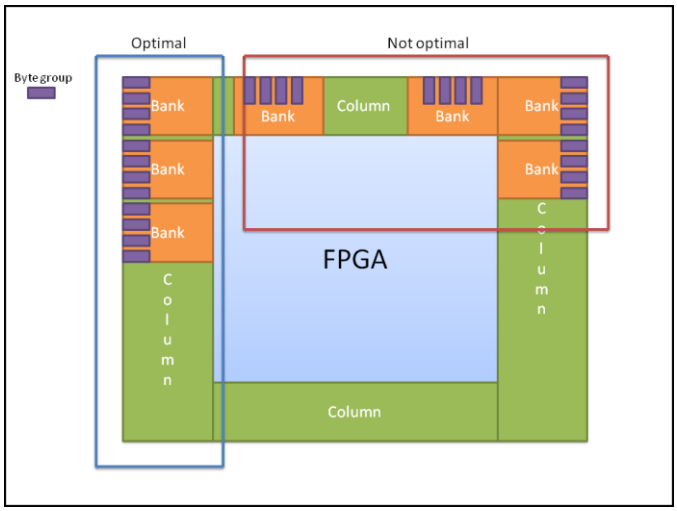

Figure 1 – Optimal and sub-optimal pin layout

To solve the bandwidth per price issue, a script was written. Its first task is to read from the database created from the manufacturer's data-sheets and make the corresponding queries. At the moment of writing of this paper, all the information came from Xilinx due to the author's familiarity with it. The database has all the necessary information about a large number of FPGAs such as name, package, family, speed, number of logic circuits, number of PCI buses, DSPs, RAM blocks and many more. Queries are generated from these data and sent to Octopart over the Web. The results are first placed in the main table and the unavailable products (those out of stock) are removed. Afterwards, for the remaining FPGAs (over 170 in total) in the table, other information is analyzed.

The next most significant factor is the pin layout as well as the type of these pins. High performance (HP) pins are better than high range (HR) pins when it comes to memory bandwidth, and faster FPGAs can work together with DDR3 modules on a higher clock frequency. All this information is acquired from the pin layout files which are also available from Xilinx.

The next step is to find groups of 3 banks on the same side of the chip. This is important due to the compatibility with the standard DDR3 interfaces as mentioned before. If there are only 1 or 2 banks available, then the DDR3 chips have to be used. They have a few drawbacks when compared to DDR3 DIMMs. DDR3 chips have a worse price / bandwidth ratio than the modules (due to the popularity and mass production of DDR3 DIMMs they have become a commodity) and are generally harder to solder than the DDR3 DIMM module socket.

The last step is to calculate the ratio of price of the FPGA and the maximal theoretical bandwidth. This parameter is created by first trying to connect as many DDR3 DIMMs as possible. Afterwards, when there are no more available groups of 3 banks, the rest of the groups are connected to the smaller, less price efficient DDR3 chips.

III. RESSULTS AND DISCUSSION

After the table has been sorted, the analysis shows that the optimal Xilinx FPGA is the **XC7A15T**, in the **1FGG484** package, Artix-7 familly, distributed by Digi-Key. It is a small FPGA which can host 1 DDR3 DIMM and 1 32 bit DDR interface working on the lowest frequency of 400 MHz. Together, they have a maximal theoretical bandwidth of 9.6 GB/s at the price of 38 $ (at the time this paper was written). The next several best candidates are all from the same family and package and achieve the same bandwidth. They have other better characteristics such as better working temperature range, more available logic but this information is not relevant for the solution of the problem.

The next best FPGA which offers a bigger bandwidth is the Kintex-7 **XC7K160T** from the **FBG676** package. It can use up to 2 DDR3 DIMMs working at a higher frequency than those working with the **XC7A15T**. The maximal bandwidth for this FPGA is 23.4 GB/s but this does come with quite a higher price of 235 $.

An interesting observation can be made at this point. We can see that for 2.43 times bigger bandwidth, we have to pay 6.18 times more, not to mention that this is only the price of the chip itself, not including the other expenses related to the printed circuit board (PCB). The **XC7K160T** being a bigger FPGA requires a more sofisticated PCB which costs more as well. On the other hand, when purchasing more of the same

FPGAs and putting them on a single PCB the price stays roughly the same. This means that 4 smaller and cheaper FPGAs could be used togather to achieve a higher bandwidth than 1 bigger and more expensive chip. To put this into perspective, 4 **XC7A15T**s cost 152 $ and have a theoretical bandwidth of 38.4 GB/s. That means 64% more throughput for 35% less money.

Name	Price [$]	Bandwith [MBps]	Ratio [MBps/$]
XC7A15T-1FGG484C	37,940112	9600	253,0303548
XC7K160T-2FBG676C	234,612	23456	99,97783575
XC7V2000T-1FLG1925C	1085,9184	102400	94,29806144

Table 1 – Summary of the best 3 candidates

As shown in Table 1, we can see that this trend continues as we go further down the table. After the **XC7K160T**, the next big increase in bandwidth comes with the Virtex-7 **XC7V2000T** from the **FLG1925** package. This big FPGA can host up to 8 DDR3 DIMMs working at 800 MHz and achieveing the bandwidth of 102.4 GB/s but at a price of 1086 $, not including the price for the PCB. When compared to the others on this list, Virtex has a slightly worse price / bandwidth ratio than Kintex and more than double worse than the Artix, shown in figure 2. This further suggests that for memory intensive problems, it is more efficient to have several smaller FPGAs than just one big one.

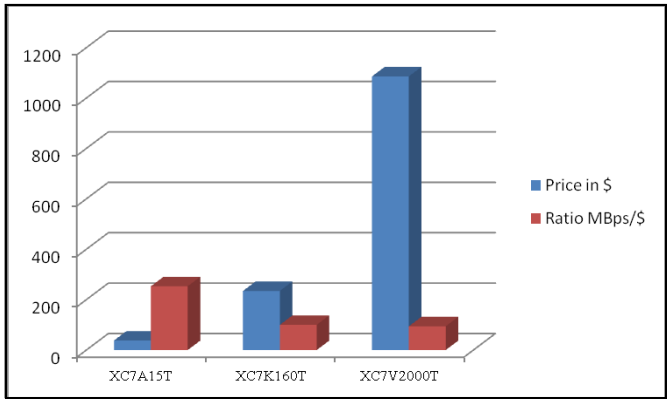

Figure 2 – Bandwidth per price ratio

When comparing GPUs to FPGAs several things have to be taken into consideration. Much like with the DDR3 DIMMs and DDR3 chips relationship, GPUs are a commodity and due to their widespread usage in PCs tend to be more affordable for the amount of computations they can do. FPGAs on the other side are not so widely used. If we approximate GPU's Gflops (floating point operations) with FPGA's GMAC (multiply and accumulate) even the modest GPUs such as a Radeon R7 360 outperforms all of the FPGAs. The main feature which gives the GPUs the edge is the fact that the GPU is an ASIC, having GDDR5 (graphics double data rate type 5 RAM) speed as well as a larger number of available pins. All these factors contribute to the GPUs superior bandwidth. On the flip side, they consume much more power. This should be the main area of concern because the aim is to create a dedicated architecture with a lifetime of several years. After years of work, the price of electricity adds up giving the FPGAs an advantage. However, it would take several decades for the electricity price to become a factor and make the FPGA a better option.

The question that arises from this is the following: what can be done to improve the FPGA bandwidth? The answer is some kind of data compression where the FPGA could be very effective. If we reduce the amount of data and not lose out on the precision of the algorithm, we would "improve" the bandwidth 4 times, thus putting the FPGA more in line with the GPU. If this criterion is met, then the FPGA becomes a better solution after 6-8 years which is acceptable and opens up more room for improvement.

IV. CONCLUSION

The main contribution of this work is a strong foundation for future research. As seen from the previous section, the optimal FPGA from Xilinx (when taking into account only the price of the chip) for solving memory intensive problems is **XC7A15T-1FGG484**. However, if more bandwidth is required, it can be obtained using the **XC7K160T** or **XC7V2000T** but at a higher cost. Interesting observation is that with more bandwidth prices of the FPGA rise faster than linear.

If the problem can be geometrically decomposed (such as FDTD), than more computing units could be used simultaneously. Each individual smaller FPGA could run its own simulation so the entire problem would be divided into several smaller parts. These simulations would be running in parallel instead of being time-multiplexed in a larger and more expensive FPGA, thus reducing execution time. With this information, a fitting architecture can be designed using several smaller FPGAs. The benefits of this option are: a bigger bandwidth, less initial costs for the components and a simpler PCB which further reduces the costs. The drawback is more work required to synchronize and coordinate the FPGAs.

Future work branches off in several ways. One obvious path is optimization of the bandwidth. This would include mainly some sort of data compression which is needed in order to reduce the magnitude of data flow. It is mandatory in order to come closer to the GPU margin. When this is achieved, the results have to be compared with the refferent version to make sure that the precission is not lost significally.

The next path is the definition of a new type of architecture which would utilize the possibly large memory bandwidth comprising of several smaller FPGAs and DDR3 DIMMs alongside the optimized bandwidth. This would include the architecture design and implementation. An alternative would be an ASIC design. Its size could be relatively small due to the fact that FDTD computation does not require a big amount of logic. It would, however, utilize many pins in order to use

as many DIMMs as possible. The tranzition from FPGA to ASIC would also be easier than to just start with the ASIC from scratch. Again, after the completion, its results should be compared with the existing solutions in order to find out if this idea is feasable and applicable.

Finally, improvements can be made in the script as well. Different kinds of analysis and conclusions can be made with this script such as finding the optimal FPGA with the DSPs per cost. Other manufacturers such as Altera should be added as well and perhaps even an entire application can be made as a helping tool when searching for optimal components.

ACKNOWLEDGEMENT

This project was partially supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia under the grant TR32029, year 2017.

REFERENCES

[1] K. Yee, "Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media", IEEE Trans. Antennas and Propagation, 16 (1966), pp. 302–307.

[2] W. Chen, P. Kosmas M. Leeser, C. Rappaport, "An FPGA implementation of the two-dimensional finite-difference time-domain (FDTD) algorithm", Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays, ACM, February 2004.

[3] P. Kosmas, Y. Wang, and C. Rappaport, "Three-Dimensional FDTD Model for GPR Detection of Objects Buried in Realistic Dispersive Soil", SPIE Aerosense Conference, Orlando, FL, April 2002, pp.330–338.

[4] S. Noghanian, A. Sabouni, T. Desell, A. Ashtari, *Microwave Tomography - Global Optimization, Parallelization and Performance Evaluation*, Springer 2014.

[5] M. Naylor, P.J. Fox, A.T. Markettos, S.W. Moore, "Managing the FPGA memory wall: Custom computing or vector processing?", Field Programmable Logic and Applications (FPL), 23rd International Conference on (pp. 1-6). IEEE September 2013,.

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

[7] W. Chen, "Acceleration of the 3D FDTD Algorithm in Fixed-point Arithmetic using Reconfigurable Hardware" Ph.D. dissertation, The Department of Electrical and Computer Engineering, Northeastern University, Boston, Massachusetts, August 2007.

[8] Octopart, https://octopart.com, loaded on 5.4.2017.