ISSN: 0974-6471, Vol. 15, No. (1-2-3) 2022, Pg. 21-26

Oriental Journal of Computer Science and Technology

www.computerscijournal.org

OpenCL Altera SDK v.14.0 vs. v. 13.1 Benchmarks Study

ABEDALMUHDI ALMOMANY* and AMIN JARRAH

Computer Engineering Department, Yarmouk University, Irbid, Jordan.

Abstract

Altera SDK for OpenCL allows programmers to write a simple code in OpenCL and abstracts all Field programmable gate array (FPGA) design complexity. The kernels are synthesized to equivalent circuits using the FPGA hardware recourses Adaptive logic modules (ALMs), DSPs and Memory blocks. In this study, we developed a set of fifteen different benchmarks, each of which has its own characteristics. Benchmarks include with/without loop unrolling, have/have not atomic operations, have one/multiple kernels per single file, and in addition to one/more of these characteristics are combined. Altera OpenCL v14.0 adds more features compared with previous versions. A set of parameters chosen to compare the two OpenCL SDK versions Logic utilization (in ALMs), total registers, RAM Blocks, total block memory bits, and clock frequency.

Article History
Received: 19 October 2022
Accepted: 21 November 2022

Keywords
ALM; Altera; AOC; CPU; DE5; FPGA; GPU; OpenCL.

Introduction

OpenCL stands for Open Computing Language, which is an open framework for parallel programming executed across heterogeneous platforms CPUs, GPUs and DSPs.[1] OpenCL programming model consists of two programs; first, host program, which is usually written in C/C++, and it is responsible for loading the OpenCL programs, memory management, data transfer and errors checking.[2] Second program is the device code, which is written in OpenCL, and can be run on the available devices such as GPUs, DSPs, or FPGAs.

In OpenCL, kernel could be executed by a large number of work-items (threads). Work-items are organized in one, two or three dimensions, and are divided into blocks which can be multi-dimensions. Each block is called a workgroup. The size of a workgroup can be up to 1024 or 2048 work-items depending on device capability. All work-items inside the workgroup can be synchronized using barrier. However, synchronization cannot be between workgroups, and they could be executed in any order.[7]

The Altera SDK for OpenCL allows the programmer to implement parallel algorithms on FPGA with a high level of hardware abstraction. The Altera offline compiler (AOC) is used to generate the Altera executable file, which can be run on the FPGA (DE5 in this study. Each kernel is synthesized to an equivalent circuit on the FPGA board, and

CONTACT Abedalmuhdi Almomany emomani@yu.edu.jo Computer Engineering Department, Yarmouk University, Irbid, Jordan.

Doi: http://dx.doi.org/10.13005/ojcst15.010203.03

each circuit contains a set of hardware recourses. FPGAs implement parallel algorithm using pipelining architecture where input data passes through a sequence of stages.[4,5] FPGA main resources include Adaptive logic modules (ALM), digital signal processing (DSP) and memory blocks.

AOC is used to create a hardware configuration file. Some parameters can be combined for the optimization purpose. Compilation process is very length, which can range anywhere between minutes and several days. In the set of benchmarks here, the compilation time ranges between one hour and few minutes up to six hours and few minutes.

The Altera SDK 14.0 has been developed to include new features, such as supporting hard floating points, channel extensions, supporting new types (float 3) and other features.[5] Our motivation behind this study is to shows how these new features could affect the performance by compiling and running set of benchmarks.

FPGAs are widley used to improve the performance in several scientific applications.[7-26] The FPGA device used here is Stratix V ALM is developed to implement most of function efficiently. Each ALM contains a look-up table (eight inputs), two dedicated adders and four dedicated registers. The LUT can implement any 6-input logic functions and a number of 7-input functions. It can also be used as two separated LUTs for efficient using. The block diagram for ALM is shown in figure-1.[6]

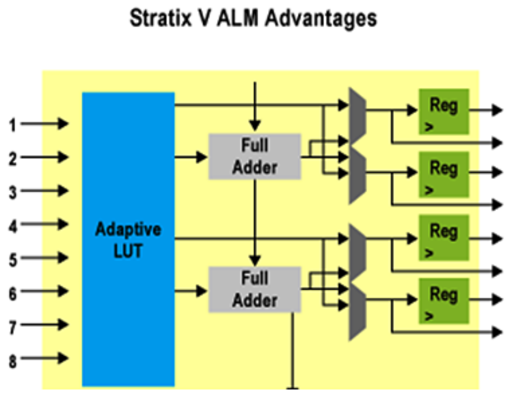

Fig. 1: Adaptive logic module (ALM) block diagram[6]

Expermintal Setup Environments

- Linux 2.6.32-504.1.3.el6.x86_64
- Altera SDK , 64-Bit Offline Compiler Ver. 14.0
- Altera SDK , 64-Bit Offline Compiler Ver. 13.1
- gcc version 4.4.7
- DE5 Board (StratixV ,Dev 5SGXEA7N2F45C2)

Experiment and Results Discussion

Several studies handel the issue of comparing different compilers.[3,4] To compare the two Altera SDK versions, a set of fifteen benchmarks were developed for comparison purpose. These benchmarks are varied in their characteristics as follows none, one, or more atomic operations, with/without loop unrolling, single/multiple kernels per file.

The benchmarks written can be classified as pure memory access, where the whole kernel is written using reads or writes memory operations. The read/ write operations could be atomic or non-atomic, using same or different atomic operation. “atomic add” and atomic exchange are used in this study. The other class is consisted of a set of arithmetic operations on floating points. These operations include four main operations (addition, subtraction, division, and multiplication). The OpenCL kernels can repeat the same code many times, where loop unrolling is used in some kernels. The same kernels run again but without loop unrolling in other benchmarks. The last thing tested using theses benchmarks is repeating the same kernel in the file up to seven times, or using more than one kernel with different characteristics. In summary, a set of fifteen benchmarks summaries all of the above attributes. A set of parameters are concerned here: logic utilization in ALMs, RAM blocks, total memory bits, clock frequency, total registers and compile time. Other parameters might be added here are size of configuration and backup files created. Our results show that the size of the files created by Altera 14.0 is less by 400MBs

The FPGA device used in the experiments contains 234,720 ALMs, 256 DSP Blocks, 52,428,800 block memory bits and 2,560 RAM Blocks.

Table 1: Altera 13.0 Benchmarks results

Altera 13.1	Bench1	Bench2	Bench3	Bench4	Bench5	Bench6	Bench7	Bench8	Bench9	Bench10	Benchll	Benchl2	Bench13	Benchl4	Benchl5
Logic utilzation (ALMS)	70%	66%	66%	67%	70%	70%	66%	16%	25%	65%	38%	20%	23%	26%	29%
Total registers	317785	298505	298246	300935	318041	318501	300312	53893	86153	305756	143413	72397	84439	97485	106650
RAM Blocks Percentage	71.3%	64.3%	64.3%	65.4%	72%	71.3%	64.3%	11%	17.9%	18.6%	23.5%	14.1%	11.4 %	20%	21.7%
Total Block memory Bits	12%	11%	11%	11%	12%	12%	12%	3%	4%	18%	5%	3%	3%	4%	4%
Actual Clock frequencey	189	203	200	193	194	193	195	305	246	187	206	211	185	194	203
compile Time in minuites	310	296	290	303	305	299	294	63	100	344	153	88	101	112	114

Table 2: Altera 14.0 Benchmarks results

Altera 14.0	Bench1	Bench2	Bench3	Bench4	Bench5	Bench6	Bench7	Bench8	Bench9	Bench10	Benchll	Benchl2	Bench13	Benchl4	Benchl5
Logic utilzation (ALMS)	59%	59%	59%	59%	59%	59%	59%	18%	26%	69%	40%	20%	21%	23%	24%
Total registers	253721	241166	241166	247225	255229	255229	242248	55886	90962	308050	160268	71354	71532	79783	88015
RAM Blocks Percentage	54.7%	37.6%	37.6%	42.7.%	56.9%	56.9%	38.8.%	11%	17.1%	74%	21.6%	14.7%	16.7%	18.4%	20%
Total Block memory Bits	10%	9%	9%	10%	11%	11%	9%	3%	3%	18%	5%	3%	5%	6%	6%
Actual Clock frequencey	171.	186.	186.	197.	174.	174.	204.	266.	220.	185.	21113	236.	241.	226.	231.
compile Time. minuites	262	256.	257.	249.	257	250	247.	69.	92.	338.	148.	83	85.	125.	99.

Examining both tables, it is clear that the Altera SDK 14.0 shows better optimization of resources, and requires less compilation time. On the other hand, the clock frequency may not be enhanced, but may be decreased. Dividing the values in Table II by the corresponding values in Table I and averaging each row will generate the results shown in Table III. This gives a comparison between the two versions considering the parameters mentioned above. Taking the average effects of all parameters, every parameter is normalized to the Altera SDK 13.1 corresponding parameter.

Table 3: Comparison Results

	Logic Utlization	Total Registers	Ram blocks	Total block memory bits	Clock Frequencey	Compile time in minuites
Altera 13.1	1	1	1	1	1	1
Altera 14.0	0.9	0.86	0.74	0.96	0.99	0.89

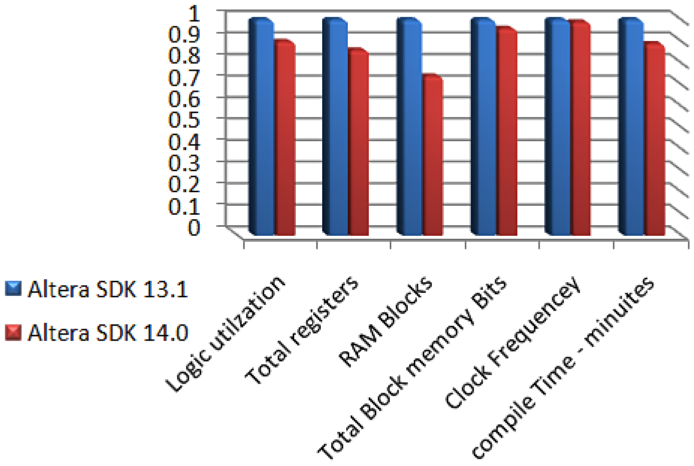

Fig. 2 :Altera SDK 14.0 vs. Altera SDK 13.1 Comparison results

Conclusion

Our study shows that using the Altera SDK 14.0 for the previous benchmarks provides better recourses utilization. We need fewer resources compared to the Altera SDK 13.0. Although the clock speed may decrease or increase, the changes is insignificant. We recommend using Altera SDK14.0 instead of Altera SDK 13.0. In future paper, the comparison will handle the most recent Intel FPGA compilers.

Acknowledgement

Nill

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

Conflict of Interest

This manuscript has not been submitted to, nor is under review at, another journal or other publishing venue. The authors certify that they have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants, participation in speakers' bureaus, membership, employment, consultancies, stock ownership, or other equity interest, and expert testimony or patent-licensing arrangements), or non-fi nancial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

References

1. Almomany, A., Al-Omari, A. M., Jarrah, A., Tawalbeh, M., & Alqudah, A. (2020). An OpenCL-based parallel acceleration of a Sobel edge detection algorithm Using Intel FPGA technology. South African Computer Journal, 32(1), 3-26.
2. Abedalmuhdi, A., Wells, B. E., & Nishikawa, K. I. (2017, April). Efficient particle-grid space interpolation of an FPGA-accelerated particle-in-cell plasma simulation. In 2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM) (pp. 76-79). IEEE
3. Almomany, A., Alquraan, A., & Balachandran, L. (2014). GCC vs. ICC comparison using PARSEC Benchmarks. International Journal of Innovative Technology and Exploring Engineering, 4(7).
4. Hiasat, R. H., &Almomani, A. A. (2013). Real Time Radio Frequency Identification Vehicles Data Logger Traffic Management System
5. Almomany, A., Ayyad, W. R., & Jarrah, A. (2022). Optimized implementation of an improved KNN classification algorithm using Intel FPGA platform: Covid-19 case study. Journal of King Saud University-Computer and Information Sciences
6. http://www.altera.com/devices/fpga/stratix-fpgas/about/fpga-architecture/stx-architecture.html
7. Almomany, A. M. (2017). Efficient openCL-based particle-in-cell simulation of auroral plasma phenomena within a commodity spatially reconfigurable computing environment
8. Nishikawa, K., Almomany, A., & Wells, B. (2016, April). Two-dimensional PIC simulations of double layers in the upward current region of the aurora with quasi-dipole magnetic fields. In EGU General Assembly Conference Abstracts (pp. EPSC2016-3037)
9. Jarrah, A., Almomany, A., Alsobeh, A. M., & Alqudah, E. (2021). High-performance implementation of wideband coherent signal-subspace (CSS)-Based DOA algorithm on FPGA. Journal of Circuits, Systems and Computers, 30(11), 2150196
10. Jarrah, A., Haymoor, Z. S., Al-Masri, H. M., & Almomany, A. (2022). High-Performance Implementation of Power Components on FPGA Platform. *Journal of Electrical Engineering & Technology,* 17(3), 1555-1571
11. Almomany, A., Sewell, S., Wells, B. E., & Nishikawa, K. I. (2017). A study of V-shaped potential formation using two-dimensional particle-in-cell simulations. Physics of Plasmas, 24(5), 052305.
12. Almomany, A., Jarrah, A., & Al Assaf, A. (2022). FCM Clustering Approach Optimization Using Parallel High-Speed Intel FPGA Technology. Journal of Electrical and Computer Engineering, 2022
13. Almomany, A., Al-Omari, A. M., Jarrah, A., & Tawalbeh, M. (2020). Discovering regulatory motifs of genetic networks using the indexing-tree based algorithm: a parallel implementation. Engineering Computations
14. Jarrah, A., Al-Tamimi, A. K., &Albashir, T. (2018). Optimized parallel implementation of extended Kalman filter using FPGA. Journal of Circuits, Systems and Computers, 27(01), 1850009
15. Al Bataineh, A., Kaur, D., & Jarrah, A. (2018, July). Enhancing the parallelization of backpropagation neural network algorithm for implementation on fpga platform. In *NAECON 2018-IEEE National Aerospace and Electronics Conference* (pp. 192-196). IEEE
16. Jarrah, A., Jamali, M. M., & Hosseini, S. S. S. (2014, June). Optimized FPGA based implementation of particle filter for tracking applications. In *NAECON 2014-IEEE* National *Aerospace and Electronics Conference* (pp. 233-236). IEEE
17. Alqudah, E., & Jarrah, A. (2020). Parallel implementation of genetic algorithm on FPGA using Vivado high level synthesis. *International Journal of Bio-Inspired Computation,* 15(2), 90-99.
18. Jarrah, A., & Jamali, M. M. (2014, November). Optimized FPGA based implementation of discrete wavelet transform. In 2014 48th *Asilomar Conference on Signals, Systems*

and Computers (pp. 1839-1842). IEEE.

19. Jarrah, A., & Jamali, M. (2013). Software tool for efficient FPGA design of direct data domain approach for space-time adaptive processing. *Electronics letters,* 49(13), 789-791.
20. Al Bataineh, A., & Jarrah, A. (2022). High performance implementation of neural networks learning using swarm optimization algorithms for EEG classification based on brain wave data. *International Journal of Applied Metaheuristic Computing* (IJAMC), 13(1), 1-17.
21. Jarrah, A. A., & Jamali, M. M. (2016). FPGA based architecture of extensive cancellation algorithm (ECA) for passive bistatic radar (PBR). *Microprocessors and Microsystems*, 41, 56-66.
22. Jarrah, A., & Jamali, M. M. (2015). Reconfigurable FPGA/GPU-based architecture of block compressive sampling matching pursuit algorithm. *Journal of Circuits, Systems and Computers,* 24(04), 1550055.
23. Al Bataineh, A., Jarrah, A., & Kaur, D. (2019). High-speed FPGA-based of the particle swarm optimization using HLS tool. *International Journal of Advanced Computer Science and Applications,* 10(5)
24. Jarrah, A., Haddad, B., Al-Jarrah, M. A., &Obeidat, M. B. (2017). Optimized parallel architecture of evolutionary neural network for mass spectrometry data processing. *International Journal of Modeling, Simulation, and Scientific Computing,* 8(01), 1750016.
25. Jarrah, A., Jamali, M. M., Hosseini, S. S. S., Astola, J., & Gabbouj, M. (2015). Parralelization of non-linear & non-Gaussian Bayesian state estimators (Particle filters). In 2015 23rd *European Signal Processing Conference (EUSIPCO)* (pp. 2506-2510). IEEE.
26. Jarrah, A., & Jamali, M. M. (2013, November). Software tool for FPGA based MIMO radar applications. In 2013 *Asilomar Conference on Signals, Systems and Computers* (pp. 1792-1795). IEEE