IMPLEMENTATION OF FRACTAL IMAGE COMPRESSION ON XPU ARCHITECTURE USING INTEL oneAPI™ APPROACH

MOHAMMAD ADIB BIN MD DAN

UNIVERSITI TEKNOLOGI MALAYSIA

IMPLEMENTATION OF FRACTAL IMAGE COMPRESSION ON XPU ARCHITECTURE USING INTEL oneAPI™ APPROACH

MOHAMMAD ADIB BIN MD DAN

A project report submitted in fulfilment of the
requirements for the award of the degree of
Master of Engineering (Computer and Microelectronic System)

School of Electrical Engineering
Faculty of Engineering
Universiti Teknologi Malaysia

JULY 2022

DEDICATION

This thesis is dedicated to my old self, who taught me that the best kind of knowledge to have is that which is learned for its own sake and never giveup even everything are falling apart. It is also dedicated to my parent, who taught me that even the largest task can be accomplished if it is done one step at a time.

ACKNOWLEDGEMENT

In the name of Allah, the Most Gracious, and the Most Merciful.

Alhamdulillah, all praise to Allah SWT for granting me the health and strength in completing this final year project. In preparing this thesis, I was engaged with many parties, technical leads, academicians, and the Intel developer community. They have contributed to my understanding and thoughts. In particular, I wish to express my sincere appreciation to my supervisor, Dr. Abdul-Malik Haider Yusef Saad, for his encouragement, guidance, advice, and motivation. Without their continued support and guidance, this thesis would not have been the same as presented here.

I am also indebted to Intel Corporation especially Programmable Solution Group (PSG Penang) for funding my study. Academicians at UTM also deserve special thanks for their assistance in supplying the relevant literature and directions.

My fellow postgraduate friends especially Nadia, Syasya, Asyikin, few of my Intel colleagues Faaruuq and Eddy for their strong support through various stages along the master journey. My sincere appreciation also extends to Intel oneAPI developer team located in Santa Clara and my manager for the help on this project.

Lastly, my sincere appreciation and thank you go to my parent and my family who always become my pillar of strength, and give endless encouragement and support to complete this final year project and masters program.

ABSTRACT

Images are stored and processed on computers as collections of bits representing pixels or points forming the picture elements. Fractal Image Compression (FIC) is based on the search for self-similarity in the image, and it can provide a high compression rate to minimize the usage of memory. However, FIC Algorithm techniques take a long time to encode an image. It requires performing an enormous number of matching operations. To speed up the process, multiple improvements in terms of hardware and software have been done. This paper proposes another approach to support flexibility and portability for FIC implementation. Nowadays, there are diverse methods of fractal image compression. Most of the methods establish a commitment between fast coding, image quality, and compression rate. Nevertheless, these methods are difficult to be implemented due to several limitations. Thus, we will develop and implement FIC Algorithm on CPU, GPU, and FPGA based on a single source code. In this work, the implementation of the FIC Algorithm on XPU is using oneAPI™ base toolkit and its library. Furthermore, the framework was developed using the Data-Parallel C++ programming language (DPC++) and executed on diverse heterogeneous hardware architectures such as CPU, GPU, and FPGA. This approach achieves 52 times execution time speed-up between CPU and GPU implementation and significant improvement between targeted XPU architecture.

ABSTRAK

Imej disimpan dan diproses pada komputer sebagai koleksi bit yang mewakili piksel atau titik yang membentuk elemen gambar. Pemampatan Imej Fraktal (PIF) adalah berdasarkan pencarian persamaan diri dalam imej, dan ia boleh memberikan kadar mampatan yang tinggi untuk meminimumkan penggunaan memori. Walau bagaimanapun, teknik Algoritma FIC mengambil masa yang lama untuk mengekod imej. Ia memerlukan melaksanakan sejumlah besar operasi pemadanan. Untuk mempercepatkan proses, pelbagai penambahbaikan dari segi perkakasan dan perisian telah dilakukan. Kertas kerja ini mencadangkan pendekatan lain untuk menyokong fleksibiliti dan mudah alih untuk pelaksanaan PIF. Pada masa kini, terdapat pelbagai kaedah pemampatan imej fraktal. Kebanyakan kaedah mewujudkan komitmen antara pengekodan pantas, kualiti imej dan kadar mampatan. Namun begitu, kaedah ini sukar dilaksanakan kerana beberapa batasan. Oleh itu, kami akan membangunkan dan melaksanakan Algoritma Pemampatan Imej Fraktal pada CPU, GPU dan FPGA berdasarkan kod sumber tunggal. Dalam kerja ini, pelaksanaan Algoritma FIC pada XPU menggunakan kit alat asas oneAPI™ dan perpustakaannya. Tambahan pula, rangka kerja telah dibangunkan menggunakan bahasa pengaturcaraan Data-Parallel C++ (DPC++) dan dilaksanakan pada seni bina perkakasan heterogen yang pelbagai seperti CPU, GPU dan FPGA. Pendekatan ini mencapai 52 kali percepatan masa pelaksanaan antara pelaksanaan CPU dan GPU dan peningkatan ketara antara seni bina XPU yang disasarkan.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

FIC	-	Fractal Image Compression
DPC++	-	Data-Parallel C++
API	-	Application Programming Interface
OpenMP	-	Open Multi-Processing
OpenCL	-	Open Computing Language
XPU	-	CPU, GPU, FPGA, and ASIC
CPU	-	Centre Processing Unit
GPU	-	Graphic Processing Unit
FPGA	-	Field Programmable Gate Array
ASIC	-	Application Specific Integrated Circuit
ISO	-	International Organisation for Standardization
SOC	-	System-On-Chip
GPGPU	-	General Processing using Graphic Processing Unit
PSNR	-	Peak-Signal-to-Noise-Ratio
OpenCV	-	Open Computer Vision
USM	-	Unified Shared Memory
DCT	-	Discrete Cosine Transform
CAT	-	Contractive Affine Transformation
SSH	-	Secure Shell
GUI	-	Graphical User Interface
ND	-	Dimension Work-Function
SIMD	-	Single Instruction/Multiple Data

LIST OF SYMBOLS

B	-	Brightness
S	-	Contrast
dB	-	Decibel
R_i	-	Range block pixel
D_i	-	Domain block pixel

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Problem Background

Image compression is critical for storing and transmitting visual data, which is the foundation for video and other multimedia applications. Images size grows rapidly as image quality and resolution ratio improves. Thus, this issue becomes a bottleneck in real-world applications. The increase in image size necessitates more storage space and bandwidth for transmission. In image processing, the question of how to reduce image size has become a hot topic[1], [2].

Moreover, Image compression and encoding may be accomplished using a variety of traditional known methods and standards. Classic image compression and encoding methods include Huffman Coding[3], Discrete Cosine Transformation (DCT)[4], wavelet image coding[5], and so on. Examples of these methods are BIG, H.263, JPEG, and MPEG. To compress images, the above-mentioned traditional approaches investigate the connection between pixels to reduce spatial or spectral redundancy[6]. The compression ratio, on the other hand, has now reached the bottleneck. As a result, new visual information redundancy reduction methods are required to achieve larger compression ratios[7].

Thus, the Fractal Image Compression (FIC) was introduced to tackle the compression method improvement. FIC is based on self-similarity between small and large parts of the image. Because the method of FIC is just storing the quantization parameters of Contractive Affine Transformation (CAT), it can reach a higher compression ratio in this case[8]. Eventually, the FIC method has drawbacks due to its high computational time and high resources needed. Over the years, many

enhancements, and improvements in terms of Software and Hardware to tackle the drawback of FIC. Thus this, paper we will present another improvement in terms of portability of software and hardware of FIC algorithm implementation.

1.2 Problem Statement

Based on the problem background discussed in the previous section. In this section, we will discuss and list the problem statement of the research. Firstly, FIC is a complex and intensive algorithm this is because the computational burden increase as the image size increase. For example, the $M \ x \ M$ size of the image, the time complexity of the FIC algorithm is approximately $O(n^4)$. Moreover, the FIC problem due to its computational burden increase. Although many methods proved to improve the coding time. However, the runtime performance is still very low even with a search-less approach.

Moreover, in modern hardware design with the presence of the heterogeneous hardware architecture. Due to its complex and intensive algorithm, that results in the adoption of heterogeneous parallelism of diverse hardware architecture such as CPU, GPU, FPGA, and ASIC known as XPU. This led to a surging interest in the exploitation of multiple hardware architectures instead of the CPU. Fourth is because of current development work of FIC improvement is bound to specific Architecture. For example, developers had to rely on either proprietary architecture-specific solutions like CUDA that only allow using NVIDIA hardware or low-level cross-architecture solution that complicate development like OpenCL.

Lastly, over the years, many adoptions of FIC algorithms across a diverse architecture but the developer still have the limitation of implementing this because of the lack of software framework, portability, and availability. The limitation of the common software stack for programming diverse hardware architecture causes developers had to use vendor-specific programming platforms and Application

Programming Interface (API) that gating the improvement of FIC computation across diverse hardware architectures.

1.3 Research Objectives

The research objective for this project is:

I. To implement Fractal Image Compression Algorithm using oneAPI™ approach.
II. To design the oneAPI™ Data-Parallel C++ framework for Fractal Image Compression Algorithm across XPU architecture.
III. To explore the portability of oneAPI™ framework toward multiple XPU architectures.

1.4 Research Scopes

The research scope for this project is:

I. This project will implement and use the conventional Fractal Image Compression (FIC) Algorithm.
II. Only chosen loop available in the FIC algorithm will be implemented using Intel® oneAPI™ Data-parallel C++ (DPC++) direct programming function and Intel® oneAPI™ oneTBB library with Unified Shared Memory (USM) allocation strategy.
III. In terms of software and API, the code will be written in Intel® oneAPI™ Data-parallel C++ and using a build-in Intel® oneAPI™ Base toolkit and library on the Microsoft Visual Studio platform and Intel® DevCloud™ platform.
IV. The code will execute and test on hardware such as Intel® Core™ & Xeon™ CPU, Intel® iRIS® Xe™ MAX & Xeon™ Gold integrated GPU, and Intel® STRATiX™ 10 FPGA Emulation Platform.

V. Performance indicators will be discussed in terms of kernel computational runtime, Peak signal-to-noise ratio (PSNR), Compression rate, and Analysis from the Intel VTune profiler.

1.5 Thesis Outline

This thesis consists of five chapters. Chapter 1 discusses the research introduction, problem statement, objectives, and scope of this project. The main focus of this project is to implement the FIC Algorithm on XPU using the oneAPI approach.

Chapter 2, will be discussed the theory of FIC, drawbacks of the Algorithm, oneAPI programming model and toolkit, Data-parallel C++ (DPC++), and the related work based on software and hardware approach. Moreover, also point out the research gap in this chapter.

In Chapter 3, the techniques and methodology throughout the project are discussed. The method of coding the FIC algorithm using DPC++ will be discussed in detail. Moreover, the implementation of the oneAPI base toolkit is also discussed in Chapter 3. Lastly, the benchmarking of technique and implementation of the oneAPI approach is also will be discussed in detail.

All results and discussion for this project will be presented in the next chapter, Chapter 4. The faced problem and gating solution to overcome the problems will be discussed in this chapter. The novelty of the results and findings will be mentioned in this chapter as well. Lastly, Chapter 5 will brief on the expected outcome of this project within the time allocated.

REFERENCES

[1] A. Wakatani, “Improvement of adaptive fractal image coding on GPUs,” *Digest of Technical Papers - IEEE International Conference on Consumer Electronics*, pp. 255–256, 2012, doi: 10.1109/ICCE.2012.6161854.

[2] A. Wakatani, “Preliminary implementation of two parallel programs for fractal image coding on GPUs,” *Digest of Technical Papers - IEEE International Conference on Consumer Electronics*, pp. 333–334, 2011, doi: 10.1109/ICCE.2011.5722612.

[3] D. A. Huffman, “A Method for the Construction of Minimum-Redundancy Codes,” *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952, doi: 10.1109/JRPROC.1952.273898.

[4] “Discrete Cosine Transform: Algorithms, Advantages, Applications - K. Ramamohan Rao, P. Yip - Google Books.” https://books.google.com.my/books?hl=en&lr=&id=fWviBQAAQBAJ&oi=fnd&pg=PP1&dq=K.+R.+Rao+and+P.+Yip,+Discrete+Cosine+Transform:+Algorithms,+Advantages,+Applications+(Academic+Press,+Cambridge,+MA,+2014).&ots=1tKMym1hyB&sig=kudDCMAFFD78uonrAak9VX0b9Co&redir_esc=y#v=onepage&q=K.%20R.%20Rao%20and%20P.%20Yip%2C%20Discrete%20Cosine%20Transform%3A%20Algorithms%2C%20Advantages%2C%20Applications%20(Academic%20Press%2C%20Cambridge%2C%20MA%2C%202014).&f=false (accessed Jan. 24, 2022).

[5] C. Heil and D. F. Walnut, “Fundamental papers in wavelet theory,” *Fundamental Papers in Wavelet Theory*, pp. 1–878, Jan. 2009, doi: 10.1515/9781400827268/HTML.

[6] “Fractal Geometry: Mathematical Foundations and Applications - Kenneth Falconer - Google Books.” https://books.google.com.my/books?hl=en&lr=&id=JXnGzv7X6wcC&oi=fnd&pg=PR5&dq=K.+Falconer,+Fractal+Geometry:+Mathematical+Foundations+and+Applications+(John+Wiley+%26+Sons,+NY,+2004).&ots=FdWMdwwB0K&sig=3g4_DdTTJDnRX1iuDOilQ4F-

Ny8&redir_esc=y#v=onepage&q=K.%20Falconer%2C%20Fractal%20Geometry%3A%20Mathematical%20Foundations%20and%20Applications%20(John%20Wiley%20%26%20Sons%2C%20NY%2C%202004).&f=false (accessed Jan. 24, 2022).

[7] A. Wakatani, “Implementation of fractal image coding for GPGPU systems and its power-aware evaluation,” *SysCon 2012 - 2012 IEEE International Systems Conference, Proceedings*, pp. 64–68, 2012, doi: 10.1109/SYSCON.2012.6189434.

[8] A. M. H. Y. Saad, M. Z. Abdullah, A. M. Alduais Nayef, and A. S. H. Abdul-Qawy, “An Improved Full-search Fractal Image Compression Method with Dynamic Search Approach,” *Proceedings - 10th IEEE International Conference on Control System, Computing and Engineering, ICCSCE 2020*, pp. 15–18, Aug. 2020, doi: 10.1109/ICCSCE50387.2020.9204926.

[9] “An Introduction to Fractal Image Compression Literature Number: BPRA065 Texas Instruments Europe,” 1997.

[10] M. Joshi, A. K. Agarwal, and B. Gupta, “Fractal image compression and its techniques: A review,” in *Advances in Intelligent Systems and Computing*, 2019, vol. 742, pp. 235–243. doi: 10.1007/978-981-13-0589-4_22.

[11] E. Zhao and D. Liu, “Fractal Image Compression Methods: A Review *,” 2005.

[12] D. Sophin Seeli and M. K. Jeyakumar, “A Study on Fractal Image Compression using Soft Computing Techniques,” 2012. [Online]. Available: www.IJCSI.org

[13] M. Galabov, “Fractal image compression,” in *Proceedings of the 4th international conference conference on Computer systems and technologies e-Learning - CompSysTech ’03*, 2003, pp. 320–326. doi: 10.1145/973620.973673.

[14] Y. Fisher, Ed., *Fractal Image Compression*. New York, NY: Springer New York, 1995. doi: 10.1007/978-1-4612-2472-3.

[15] IEEE Electrical Insulation Society Staff, *2013 18th Asia and South Pacific Design Automation Conference*.

[16] A. M. H. Y. Saad and M. Z. Abdullah, “High-speed fractal image compression featuring deep data pipelining strategy,” *IEEE Access*, vol. 6, pp. 71389–71403, 2018, doi: 10.1109/ACCESS.2018.2880480.

[17] A. M. H. Y. Saad and M. Z. Abdullah, “High-speed implementation of fractal image compression in low cost FPGA,” *Microprocessors and Microsystems*, vol. 47, pp. 429–440, Nov. 2016, doi: 10.1016/j.micpro.2016.08.004.

[18] T. Nguyen, “Implementation of Fractal image compression on FPGA,” *2012 Fourth International Conference on Communications and Electronics (ICCE)*, Jan. 2012, Accessed: Dec. 31, 2021. [Online]. Available: https://www.academia.edu/29936487/Implementation_of_Fractal_image_compression_on_FPGA

[19] M. Joshi, A. K. Agarwal, and B. Gupta, “Fractal Image Compression and Its Techniques: A Review,” *Advances in Intelligent Systems and Computing*, vol. 742, pp. 235–243, 2019, doi: 10.1007/978-981-13-0589-4_22.

[20] S. Bhavani and K. G. Thanushkodi, “Comparison of fractal coding methods for medical image compression,” *IET Image Processing*, vol. 7, no. 7, pp. 686–693, 2013, doi: 10.1049/IET-IPR.2012.0041.

[21] A. M. H. Y. Saad, M. Z. Abdullah, A. M. Alduais Nayef, and A. S. H. Abdul-Qawy, “An Improved Full-search Fractal Image Compression Method with Dynamic Search Approach,” *Proceedings - 10th IEEE International Conference on Control System, Computing and Engineering, ICCSCE 2020*, pp. 15–18, Aug. 2020, doi: 10.1109/ICCSCE50387.2020.9204926.

[22] A. M. H. Y. Saad and M. Z. Abdullah, “High-speed fractal image compression featuring deep data pipelining strategy,” *IEEE Access*, vol. 6, pp. 71389–71403, 2018, doi: 10.1109/ACCESS.2018.2880480.

[23] R. Hamzaoui and D. Saupe, “Fractal image compression,” *Document and Image Compression*, pp. 145–175, Jan. 2006, doi: 10.1142/S0218348X94000442.

[24] Md. E. Haque, A. al Kaisan, M. R. Saniat, and A. Rahman, “GPU Accelerated Fractal Image Compression for Medical Imaging in Parallel Computing Platform,” Apr. 2014, Accessed: Jan. 02, 2022. [Online]. Available: https://arxiv.org/abs/1404.0774v1

[25] P. Palazzari, M. Coli, and G. Lulli, “Massively parallel processing approach to fractal image compression with near-optimal coefficient quantization,” *Journal of Systems Architecture*, vol. 45, no. 10, pp. 765–779, 1999, doi: 10.1016/S1383-7621(98)00037-X.

[26] E. W. Jacobs, Y. Fisher, and R. D. Boss, “Image compression: A study of the iterated transform method,” *Signal Processing*, vol. 29, no. 3, pp. 251–263, 1992, doi: 10.1016/0165-1684(92)90085-B.

[27] U. Erra, “Toward Real Time Fractal Image Compression Using Graphics Hardware,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3804 LNCS, pp. 723–728, 2005, doi: 10.1007/11595755_92.

[28] J. Reinders, B. Ashbaugh, J. Brodman, M. Kinsner, J. Pennycook, and X. Tian, “Data Parallel C++,” *Data Parallel C++*, 2021, doi: 10.1007/978-1-4842-5574-2.

[29] “SYCL Overview - The Khronos Group Inc.” https://www.khronos.org/sycl/ (accessed Jan. 02, 2022).

[30] Intel Corporation, “Intel ® oneAPI Programming Guide,” 2020. [Online]. Available: www.intel.com

[31] Intel Corporation, “oneAPI Specification Release 1.1-rev-1 Intel,” 2020.

[32] “Introduction — oneAPI Specification 1.1-rev-1 documentation.” https://spec.oneapi.io/versions/latest/introduction.html (accessed Jan. 01, 2022).

[33] “oneAPI Programming Model | oneAPI.” https://www.oneapi.io/ (accessed Jan. 01, 2022).

[34] R. Nozal and J. L. Bosque, “Exploiting co-execution with oneAPI: heterogeneity from a modern perspective,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12820 LNCS, pp. 501–516, Jun. 2021, doi: 10.1007/978-3-030-85665-6_31.

[35] E. Marinelli and R. Appuswamy, “XJoin: Portable, parallel hash join across diverse XPU architectures with oneAPI; XJoin: Portable, parallel hash join across diverse XPU architectures with oneAPI,” *Proceedings of the 17th International Workshop on Data Management on New Hardware (DaMoN 2021)*, 2021, doi: 10.1145/3465998.

[36] J. Reinders, B. Ashbaugh, J. Brodman, M. Kinsner, J. Pennycook, and X. Tian, *Data Parallel C++*. Apress, 2021. doi: 10.1007/978-1-4842-5574-2.

[37] “Software Architecture — oneAPI Specification 1.1-rev-1 documentation.” https://spec.oneapi.io/versions/latest/architecture.html (accessed Jan. 01, 2022).

[38] “Data Parallel C++ Language.” https://www.intel.com/content/www/us/en/developer/tools/oneapi/data-parallel-c-plus-plus.html#gs.l1tf7j (accessed Jan. 02, 2022).

[39] J. Reinders, B. Ashbaugh, J. Brodman, M. Kinsner, J. Pennycook, and X. Tian, “Unified Shared Memory,” *Data Parallel C++*, pp. 149–171, 2021, doi: 10.1007/978-1-4842-5574-2_6.

[40] J. Reinders, M. Voss, P. Reble, and R. Asenjo-Plaza, “C++ for Heterogeneous Programming: oneAPI (DPC++ and oneTBB),” Nov. 2020, Accessed: Jan. 01, 2022. [Online]. Available: https://riuma.uma.es/xmlui/handle/10630/20404

[41] K. Mani Chandy and C. Kesselman, “Compositional C++: Compositional parallel programming,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 757 LNCS, pp. 124–144, 1992, doi: 10.1007/3-540-57502-2_44.

[42] W. Yong, Z. Yongfa, W. Scott, Y. Wang, X. Qing, and W. Chen, “Developing medical ultrasound imaging application across GPU, FPGA, and CPU using oneAPI,” *ACM International Conference Proceeding Series*, Apr. 2021, doi: 10.1145/3456669.3456680.

[43] “DPC++ — oneAPI Specification 1.1-provisional-rev-1 documentation.” https://spec.oneapi.io/versions/1.1-provisional-rev-1/elements/dpcpp/source/index.html (accessed Jan. 02, 2022).

[44] “oneDPL — oneAPI Specification 1.1-provisional-rev-1 documentation.” https://spec.oneapi.io/versions/1.1-provisional-rev-1/elements/oneDPL/source/index.html (accessed Jan. 02, 2022).

[45] N. Alqudami and S.-D. Kim, “Adaptive discrete cosine transform-based image compression method on a heterogeneous system platform using Open Computing Language,” *https://doi.org/10.1117/1.JEI.23.6.061110*, vol. 23, no. 6, p. 061110, Sep. 2014, doi: 10.1117/1.JEI.23.6.061110.

[46] “An Introduction to Fractal Image Compression Literature Number: BPRA065 Texas Instruments Europe,” 1997.

[47] A. Öztürk, “A Fast Fractal Image Compression Algorithm Based on a Simple Similarity Measure The Implementation of a Parallel Data Processing Platform on Linux based PC Clusters View project Augmented Reality with VR Glasses View project,” 2011. [Online]. Available: https://www.researchgate.net/publication/236658356

[48] H. Cao and X. Q. Gu, “Implement research of fractal image encoding based on OpenMP parallelization model,” in *2011 International Conference on Electric Information and Control Engineering, ICEICE 2011 - Proceedings*, 2011, pp. 462–465. doi: 10.1109/ICEICE.2011.5777994.

[49] H. Cao and X. J. Gu, “OpenMP parallelization of Jacquin fractal image encoding,” *2010 International Conference on E-Product E-Service and E-Entertainment, ICEEE2010*, 2010, doi: 10.1109/ICEEE.2010.5661366.

[50] “OpenMP Parallelization of Jacquin Fractal Image Encoding | IEEE Conference Publication | IEEE Xplore.” https://ieeexplore.ieee.org/abstract/document/5661366?casa_token=vtjvFua5hOMAAAAA:ML5XLUMbQNUxPuabE716tY-QOQW70RfZqnMTppNUZ6fiNRxQLRVt-mv1KrumVR_HpHDNpO-p4A (accessed Dec. 30, 2021).

[51] A. Wakatani, “Improvement of adaptive fractal image coding on GPUs,” *Digest of Technical Papers - IEEE International Conference on Consumer Electronics*, pp. 255–256, 2012, doi: 10.1109/ICCE.2012.6161854.

[52] S. N. Khan and N. Akhtar, “Parallelization of Fractal Image Compression Over CUDA,” *undefined*, vol. 150 LNEE, pp. 375–382, 2013, doi: 10.1007/978-1-4614-3363-7_42.

[53] N. Alqudami and S.-D. Kim, “OpenCL-based optimization methods for utilizing forward DCT and quantization of image compression on a heterogeneous platform”, doi: 10.1007/s11554-015-0507-5.

[54] D. J. Jackson, H. Ren, X. Wu, and K. G. Ricks, “A hardware architecture for real-time image compression using a searchless fractal image coding method,” *Journal of Real-Time Image Processing*, vol. 1, no. 3, pp. 225–237, Apr. 2007, doi: 10.1007/S11554-007-0024-2/TABLES/3.

[55] D. Vidya, R. Parthasarathy, T. C. Bina, and N. G. Swaroopa, “Architecture for fractal image compression,” *Journal of Systems Architecture*, vol. 46, no. 14, pp. 1275–1291, Dec. 2000, doi: 10.1016/S1383-7621(00)00018-7.

[56] S. Sharma, V. Chaurasia, and M. K. Gupta, “Review of Hardware on Fractal Image Compression”.

[57] D. Vidya, R. Parthasarathy, T. C. Bina, and N. G. Swaroopa, “Architecture for fractal image compression,” *Journal of Systems Architecture*, vol. 46, no. 14, pp. 1275–1291, Dec. 2000, doi: 10.1016/S1383-7621(00)00018-7.

[58] K. Belloulata and J. Konrad, “Fractal image compression with region-based functionality,” *IEEE Transactions on Image Processing*, vol. 11, no. 4, pp. 351–362, Apr. 2002, doi: 10.1109/TIP.2002.999669.

[59] “EBSCOhost | 71826711 | Fast Search Strategies for Fractal Image Compression.” https://web.p.ebscohost.com/abstract?direct=true&profile=ehost&scope=site&authtype=crawler&jrnl=10162364&AN=71826711&h=R%2f%2bT86C6Ke1ybopVamp3pdvXN1FrIR6dy8IRjv9KCRc%2bhQwKaVLA0kqqRqQV%2b0L7cgfx6uOC1h174dUXDKbTQA%3d%3d&crl=c&resultNs=AdminWebAuth&resultLocal=ErrCrlNotAuth&crlhashurl=login.aspx%3fdirect%3dtrue%26profile%3dehost%26scope%3dsite%26authtype%3dcrawler%26jrnl%3d10162364%26AN%3d71826711 (accessed Jan. 01, 2022).

[60] R. Gupta, D. Mehrotra, and R. K. Tyagi, “Computational complexity of fractal image compression algorithm,” *IET Image Processing*, vol. 14, no. 17, pp. 4425–4434, Dec. 2020, doi: 10.1049/IET-IPR.2019.0489/CITE/REFWORKS.

[61] O. A. S. Alkishriwo, “Image compression using adaptive multiresolution image decomposition algorithm,” *IET Image Processing*, vol. 14, no. 14, pp. 3572–3578, Dec. 2020, doi: 10.1049/IET-IPR.2019.1699.

[62] W. Xing-Yuan, Z. Dou-Dou, and W. Na, “Fractal image coding algorithm using particle swarm optimisation and hybrid quadtree partition scheme,” *IET Image Processing*, vol. 9, no. 2, pp. 153–161, Feb. 2015, doi: 10.1049/IET-IPR.2014.0001.

[63] D. Chen and D. Singh, “Fractal video compression in OpenCL: An evaluation of CPUs, GPUs, and FPGAs as acceleration platforms,” *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC*, pp. 297–304, 2013, doi: 10.1109/ASPDAC.2013.6509612.

[64] “(PDF) FRACTALS IMAGE RENDERING AND COMPRESSION USING GPUS.” https://www.researchgate.net/publication/230772743_FRACTALS_IMAGE_

RENDERING_AND_COMPRESSION_USING_GPUS (accessed Dec. 31, 2021).

[65] A. al Sideiri, N. Alzeidi, M. al Hammoshi, M. S. Chauhan, and G. AlFarsi, "CUDA implementation of fractal image compression," *Journal of Real-Time Image Processing*, vol. 17, no. 5, pp. 1375–1387, Oct. 2020, doi: 10.1007/s11554-019-00894-7.

[66] S. Padmavati and V. Meshram, "FPGA Implementation for Fractal Quadtree Image Compression," *International Journal of Computer Sciences and Engineering*, vol. 6, no. 10, pp. 405–409, Oct. 2018, doi: 10.26438/IJCSE/V6I10.405409.

[67] F. Ancarani, A. de Gloria, M. Olivieri, and C. Stazzone, "Design of an ASIC architecture for high speed fractal image compression," *Proceedings of the Annual IEEE International ASIC Conference and Exhibit*, pp. 223–226, 1996, doi: 10.1109/ASIC.1996.551998.

[68] K. P. Acken, M. J. Irwin, and R. M. Owens, "A Parallel ASIC Architecture for Efficient Fractal Image Coding," *Journal of VLSI signal processing systems for signal, image and video technology 1998 19:2*, vol. 19, no. 2, pp. 97–113, Jul. 1998, doi: 10.1023/A:1008005616596.

[69] A. Wakatani, "Preliminary implementation of two parallel programs for fractal image coding on GPUs," *Digest of Technical Papers - IEEE International Conference on Consumer Electronics*, pp. 333–334, 2011, doi: 10.1109/ICCE.2011.5722612.

[70] D. Chen and D. Singh, "Fractal Video Compression in OpenCL," 2013.

[71] W. D. Marsland, "Use of digital computers in basic mathematics courses," *AFIPS Conference Proceedings - 1965 Fall Joint Computer Conference - Computers: Their Impact on Society, AFIPS 1965*, pp. 111–114, Nov. 1965, doi: 10.1145/1464013.1464032.

[72] C. Engelman, "MATHLAB: A program for on-line machine assistance in symbolic computations*," *Proceedings of the November 30--December 1, 1965, fall joint computer conference, part II: computers: their impact on society on XX - AFIPS '65 (Fall, part II)*, doi: 10.1145/1464013.

[73] M. Costanzo, E. Rucci, C. G. Sanchez, and M. Naiouf, "Early Experiences Migrating CUDA codes to oneAPI," May 2021, Accessed: Jan. 01, 2022. [Online]. Available: https://arxiv.org/abs/2105.13489v1

[74] “Optimize Your GPU Applications with the Intel® oneAPI Base Toolkit.” https://www.intel.com/content/www/us/en/developer/tools/oneapi/gpu-optimization-workflow.html#gs.4leqww (accessed Jul. 02, 2022).

[75] “Model Offloading to a GPU.” https://www.intel.com/content/www/us/en/develop/documentation/advisor-user-guide/top/model-offloading-to-a-gpu.html (accessed Jul. 02, 2022).

[76] “Pipes Extension.” https://www.intel.com/content/www/us/en/develop/documentation/oneapi-fpga-optimization-guide/top/flags-attr-prag-ext/kernel-controls/pipes-extension.html (accessed Jul. 02, 2022).

[77] “Kernel Variables.” https://www.intel.com/content/www/us/en/develop/documentation/oneapi-fpga-optimization-guide/top/flags-attr-prag-ext/kernel-variables.html (accessed Jul. 02, 2022).

[78] “Emulator Environment Variables.” https://www.intel.com/content/www/us/en/develop/documentation/oneapi-programming-guide/top/programming-interface/fpga-flow/emulate-your-design/emulator-environment-variables.html#emulator-environment-variables (accessed Jul. 02, 2022).

[79] “Cygwin.” https://www.cygwin.com/ (accessed Jul. 02, 2022).

[80] “Fix Performance Bottlenecks with Intel® VTune™ Profiler.” https://www.intel.com/content/www/us/en/developer/tools/oneapi/vtune-profiler.html#gs.4msclv (accessed Jul. 02, 2022).

[81] “Analysis System Options.” https://www.intel.com/content/www/us/en/develop/documentation/vtune-help/top/set-up-project/analysis-system/analysis-system-options.html (accessed Jul. 02, 2022).

[82] “Set Up Analysis Target.” https://www.intel.com/content/www/us/en/develop/documentation/vtune-help/top/set-up-analysis-target.html (accessed Jul. 02, 2022).

[83] “The stdlib Chrono Library”.

[84] S. Kwon, J. Byun, and H. W. Park, “Elimination of Race Condition During GPU Acceleration of Probabilistic Height Map,” *Lecture Notes in Networks*

and Systems, vol. 429 LNNS, pp. 313–322, 2022, doi: 10.1007/978-3-030-97672-9_28.

[85] “Intel® oneAPI Base Toolkit - Intel Communities.” https://community.intel.com/t5/Intel-oneAPI-Base-Toolkit/bd-p/oneapi-base-toolkit (accessed Jul. 03, 2022).

[86] “Limitations of the Emulator.” https://www.intel.com/content/www/us/en/develop/documentation/oneapi-programming-guide/top/programming-interface/fpga-flow/emulate-your-design/limitations-of-the-emulator.html (accessed Jul. 03, 2022).

[87] “Discrepancies in Hardware and Emulator Results.” https://www.intel.com/content/www/us/en/develop/documentation/oneapi-programming-guide/top/programming-interface/fpga-flow/emulate-your-design/discrepancies-in-hardware-and-emulator-results.html (accessed Jul. 03, 2022).

[88] “IPC.” https://www.intel.com/content/www/us/en/develop/documentation/vtune-help/top/reference/cpu-metrics-reference/ipc.html (accessed Jul. 03, 2022).

[89] “Memory Bound.” https://www.intel.com/content/www/us/en/develop/documentation/vtune-help/top/reference/cpu-metrics-reference/memory-bound.html (accessed Jul. 03, 2022).

[90] “Microarchitecture Exploration Analysis.” https://www.intel.com/content/www/us/en/develop/documentation/vtune-help/top/analyze-performance/microarchitecture-analysis-group/general-exploration-analysis.html (accessed Jul. 03, 2022).

[91] “Microarchitecture Usage.” https://www.intel.com/content/www/us/en/develop/documentation/vtune-help/top/reference/cpu-metrics-reference/microarchitecture-usage.html (accessed Jul. 03, 2022).

[92] K. O’leary, “Intel® VTune Profiler and Intel® Advisor Overview”.

[93] “CPU Metrics.” https://www.intel.com/content/www/us/en/develop/documentation/vtune-help/top/reference/cpu-metrics-reference.html (accessed Jul. 03, 2022).

[94] “Summary - Hotspots by CPU Usage.” https://www.intel.com/content/www/us/en/develop/documentation/vtune-help/top/reference/user-interface-reference/window-summary/window-summary-hotspots-by-cpu-usage.html#window-summary-hotspots-by-cpu-usage_CPU_USAGE (accessed Jul. 03, 2022).