Cluster Computing (2020) 23:1505–1516
https://doi.org/10.1007/s10586-019-03011-2

Comprehensive regression-based model to predict performance of general-purpose graphics processing unit

Mohammad Hossein Shafiabadi[1] · Hossein Pedram[2] · Midia Reshadi[1] · Akram Reza[3]

Received: 5 June 2019 / Revised: 10 October 2019 / Accepted: 29 October 2019 / Published online: 25 November 2019

Abstract
Recently, the use of graphics processors has been significantly increased in fast and accurate scientific calculations. These processors provide a heterogeneous design space, and make designers capable of performing more accurate designs with higher efficiency. In this paper, a regression model is proposed to predict the performance of various applications on general-purpose graphics processors units. We present the main challenges for predicting the efficiency of graphics processing units (GPUs) based on simulation experiments. Also, we build the regression statistical inference from the result of the simulation, which predicts the efficiency of GPUs in various performances with approximately 7% of error of measurement. We have used AMD Southern Island and SDK 2.5, OpenCL which are both based on OpenCL. The first version of the design is built by very large design space, approximately about 17 billion points, from which 8000 points were randomly chosen, and the performance of graphic processors was calculated based on the results of the simulation. The model of non-linear regression is capable of predicting the performance of graphics process with the average error rate of 7%.

Keywords General-purpose graphics processors unit · Non-linear regression model · Performance prediction · Validation

1 Introduction

Nowadays, the use of accelerators, such as graphics processing units (GPUs) has been increased in quick and accurate scientific calculations [1–3]. Fast and efficient GPU enables computers to generate more accurate results in a shorter time. Moreover, GPU supports a variety of different levels of parallelism contracts with many-core processor [4, 5].

Many-core and multi-threads graphic processors support a variety of different levels of data parallelism. These processors support up to 6 times more memory bandwidth, and 21 times higher throughput than the central processing unit (CPU) [6–9], however, there are several challenges to use them. Determining the performance, power consumption and throughput processors are some of those challenges that keep the users and designers unaware of the result of their work. There are several other challenges that arouse before using GPU, such as the lack of automatic graphics processors simulation tools for users and designers to implement the design before the final product, and to be aware of approximate result and being capable of validating them. It's also noteworthy that the heterogeneous many core architecture is very complicated, and finding convenient and efficient design space is difficult [10–16].

✉ Hossein Pedram
Pedram@aut.ac.ir

Mohammad Hossein Shafiabadi
Shafiabadi@iiau.ac.ir

Midia Reshadi
reshadi@srbiau.ac.ir

Akram Reza
a.reza@qodsiau.ac.ir

[1] Department of Computer Engineering, Islamic Azad University, Science and Research Branch, Tehran, Iran

[2] Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran

[3] Department of Computer Engineering, Islamic Azad University, Shahr-e-Qods Branch, Tehran, Iran

A proper configuration and design space require careful search, analysis, and evaluation of performance metrics across the design space. Simulation-based methods have more accuracy with lower speed; however, analytical methods based on the architecture and the relationship between the model parameters are very fast with less accuracy. There are methods which are known as experimental methods. In these methods, some points are selected from the whole space design that undergone a simulation process, and as the final step, the model will be extracted based on the results. These methods are both accurate as simulation models, and fast as analytical methods [17–20].

The process of modeling a plan in various fields such as performance, power, and area makes the designer aware of considering the compromise between all effective parameters of all the design points. Designer calculates all the design parameters based on the different values of the cost function or objective function. Therefore, they define the relationship between output parameters, and then expand it to other uses. Therefore, it seems easy, but by considering the numerous parameters in design space, a large space which grows exponentially is very difficult, and time consuming. Design space in different sizes has different levels of abstraction. Moving from system-level to circuit-level increases the number of parameters those results in a larger design space [21–23]. This problem has led designers to work on abstraction levels for their systems and search design space at this level. At the system level design, modeling and performance evaluation begin in early design stage and represents system-level models which are applied to behavioral characteristics, the relationship between architecture and architectural applications, the initial estimate of efficiency [16, 24], power consumption [2, 25, 26], and pay the cost of the project [27].

Modeling at higher levels makes it simple that increases the speed of simulation and evaluation [28, 29]. System designer should pass different levels of abstraction for final implementation. The higher level models enable designers to search large design space. Additionally, accurate models at lower levels in the next stages shed light on more details; therefore more accuracy is obtained [6].

In this paper, regression statistical inference method is used to predict the efficiency of graphic processors [13, 30]. In this regard, firstly, through Unified at Random method, 8000 samples are selected from the very large design space, each of which includes hardware and software parameters that affect the performance of graphics processors. Some of these parameters are given in Tables 1 and 3. Then, through Multi2sim simulator, each simulation sample is performed, and the GPU performance is calculated and recorded.

The obtained data set is analyzed using the regression method. In this analysis, efficiency is used as the dependent parameter. In addition, software and hardware parameters are used as independent parameters. Next, a mathematical model is obtained to calculate the performance of graphic processors.

The main purpose of this work is to provide a mathematical model based on the statistical deduction method, which can be used to predict the graphics processor with proper accuracy, and in a shorter time before designing the performance. Our main contributions in this paper are:

- Predict performance of GPU,
- Made a non-linear regression model,
- Consider software parameter in generic public model.

The rest of the paper is as follows: Sect. 2 will present works which are previously carried out. In Sect. 3, an experimentally model will be explained. Finally, the generated results of the model will be shown in Sect. 4.

2 Related work

In this section, some related studies are discussed briefly which contain modeling performance and power in type of computers, and so discussed some important features and challenges.

Lee et al. [12], proposed using non-linear regression (Spline) for modeling efficiency and power superscalar microprocessors in the microarchitecture level. Extracted model is evaluated and improved. In this design, sampling from design space was used in order to simulate using uniform random method, also interaction between parameters was considered, and fitness cubic spline function was used.

Joseph et al. [24] proposed a linear regression method regardless of the behavior of program for extracting superscalar CPU model. In this study, regression coefficients for superscalar microarchitecture parameters are extracted by simulating a program with a limited amount of points in the design space.

Wu and Lee [31] provided a regression-based model to analyze both hardware and software of a system that includes and identifies the behavior of the application, and it is also capable of defining the amount and the method for independent micro-architecture. They claim that the regression model has been improved by:

- Variables of model must be properly chosen.
- Non-linear parameters model on a spline.
- Dependency between parameters are extracted revelatory and applied to model. In this study, after building system model, genetic algorithms are used to update it.

Table 1 Design space parameters

Unit	Parameter	Default	Range	|Si|
Device	NumComputeUnits	32	4::2X::64	5
ComputeUnit	NumWavefront	4	4::2X::64	5
	MaxWorkGroupsPerWavefront	10	4::2X::32	4
	MaxWavefrontsPerWavefront	10	8::2X::64	4
	NumVectorRegisters	64k	8K::2X::64K	4
	NumScalarRegisters	2K	1K::2X::8K	4
FrontEnd	FetchLatency	5	1::2X::8	4
	FetchWidth	4	1::2X::8	4
	FetchBufferSize	10	1::2X::16	5
	IssueWidth	5	1::2X::16	5
SIMDUnit	NumSIMDLanes	16	4::2X::32	4
	Width	1	1::2X::8	4
	IssueBufferSize	1	1::2X::16	5
	DecodeLatency	1	1::2X::8	4
	DecodeWidth	1	1::2X::8	4
	DecodeBufferSize	1	1::2X::16	5
	ReadExecWriteLatency	8	1::2X::8	4
	ReadExecWriteBufferSize	2	1::2X::16	5

Baghsorkhi et al. [7] have provided an analytical model to predict the performance efficiency of general-purpose graphics architecture processors (GPU). The plan runs using CUDA architecture on NVIDIA GPUs.

Kothapalli et al. [10] have provided an analytical model to predict the efficiency performance of CUDA-based GPUs. In this model, several factors including scheduling, memory hierarchy, and the executive stream pipeline are considered, which are effective in calculating performance of graphics processors.

Huang et al. [32] proposed the cumulative model to predict both power-consumption and performance of GPU architecture. The authors suggested that it is a tough challenge to investigate the increasing number of computational cores in graphic processors, increasing parallel data, finding both the perfect performance combined, and perfect power consumption. Increasing the number of cores increases performance computing, but it will also increase the power consumption of the system. In addition, by developing technology, the scalability, and minimizing dimension of micro-architecture, various problems such as increased static power consumption, and the effect of temperature on power consumption will occur.

Dubach et al. [18] proposed a new model to predict the performance of graphic processors. Since the graphic processors architecture, memory of CPU, and GPU are separated, in this project, in addition to the processor execution time, the data transmission rate (transmission time) between the CPU and GPU is intended one of the factors affecting the performance of the GPU many core processors. They argue that in some applications, the processor executive time is overshadowed by the data transfer rate between CPU and GPU memories. Providers of this model have used an automatic tool, called GROPHECY to implement their plan. GROPHECY is an automatic tool that has received high-level language code Skeleton and produces all solutions to achieve optimal execution time.

Jia et al. [33] introduced a framework to explore the GPU performance space called Stargazer. This model uses GP-GPUSim simulator and works with stepwise regression method. In this framework, patterns have been chosen randomly from GPU design space and simulated. Then estimated performance is extracted by regression method.

Kerr et al. [19, 20] also introduced a framework called Eiger to analyze applications performance on the GPUs. In this framework, first program compiles with instrumentation instruction and runs to extract a set of information of performance counter. Also the program is analyzed statically to make a series of independent parameters of the microarchitecture. The calculated values are given to analysis phase to decrease their dimensions by PCA method. Collected information is given to model making unit which can produce the effective model using both linear and non-linear methods.

Joseph et al. [24] proposed a model based on regression to estimate processors efficiency. The method used in this model is analytical tasks to dope the efficiency of the processors. He used the IPC as the assessment parameter of processor's efficiency, he also used SPEC CPU2006 benchmark programs for model assessment. The average considered error for this model is 10%.

Jia et al. [29] offered a method based on regression for designing graphic processor's state space using random sampling, after the samples were chosen a model based on regression will be presented.

Ahmed et al. [34] used machine learning supported by predictive analytics. They said early prediction and proper treatments can possibly stop, or slow the progression of this chronic disease to end-stage, where dialysis or kidney transplantation is the only way to save patient's life. Therefor predictive method is used for early diagnosis of sicknesses.

Khalaf et al. [35], presented a novel application of machine learning algorithms including Neural Network architecture for the prediction of flood severity.

According to reviewed papers, simulation based methods are slow but they have high precision and the statistical methods are fast but they have low precisions. In this paper, we purpose new simulation and statistical method to predict performance of GPU.

3 Evaluation method

Before the extraction of regression model to predict the performance of the graphics processor, first, we must select different configuration of the entire design space, in order to obtain the results of their simulation. The results of these simulations are analyzed statistically, which are obtained by the intended regression model.

3.1 AMD Southern Island

We use Multi2Sim which is a multi-threaded and many-core processors simulation tool that supports graphic processors as well. This tool supports OpenCL-based applications now. So, it was chosen to research hypotheses of AMD Southern Island architecture, which is based OpenCL [16, 21]. In following, design space, and the parameters limits of this processor will be explained.

The AMD Southern Islands series of processors (SI-GPU) implements a parallel microarchitecture that provides an excellent platform for both computer graphics applications and general-purpose data parallel computer applications. Any data parallel application that requires high bandwidth or significant computational requirements is a candidate for acceleration on SI-GPU devices.

Figure 1 shows a block diagram of the SI-GPU.

The SI-GPU consists of a command processor that communicates with the host and schedule on chip workloads. The ultra-threaded dispatch processor accepts commands from the command processor and distributes work across the array of compute units. Each compute unit contains instruction logic (fetch, buffer, decode, issue), scalar and vector ALU units with registers, a high-bandwidth, low latency shared memory, and a read/write L1 cache equipped with general load/store/atomic and texture addressing/filtering capabilities. The compute units are supported by a multi-banked read/write L2 memory cache, a global shared memory, and memory controllers to support the data accessibility necessary to support kernel execution.

A compute unit is the basic unit of computation, and different Southern Island products have varying numbers of compute units. Each compute unit contains:

- Scalar ALU and scalar GPRs.
- Four SIMDs, each consisting of a vector ALU and vector GPRs.
- Local memory (local data store, or LDS).
- Read/write access to vector memory through a Level-1 cache.
- Instruction cache, which is shared by four CUs.
- Constant cache, which is shared by four CUs.

Table 1 shows some of primary space chosen for the GPU to CPU–GPU platform modeling. All of these parameters are divided to nine main section (Device, ComputeUnit, FrontEnd, SIMDUnit, ScalarUnit, BranchUnit, LDSUnit, VectorMemUnit, LDS), and their limits and changes are given in column Range. Column | Si | shows the number of changes of all the parameters. If values of this column are multiplied together, the entire design space can be obtained for a program that will be a very large number.

$$\text{Design space} = \prod_{i=1}^{N} |S_i|. \tag{1}$$

3.2 Benchmark application

OpenCL SDK 2.5 was chosen to simulate, and to make modeling program which includes 23 different programs in different fields. The list is given in Table 1.

The Cartesian multiply values of column S_i produce the number of points obtained state space for each benchmark program which is shown in Table 2 that will be 2.01761 e + 38 a very large number, if it is multiplied by the number of applications, the number will be 4.64051 e + 39. It's clear that simulation of all these design space points to analyze the performance of graphic processors is so time-consuming and difficult. Since today's hardware designers are searching after the ways to reduce design time, so other ways must be tried. Employing statistical analysis methods on experiment-based will be very useful. In this plan, it was used to create a model to define the performance of GPU based on experimental methods of statistical analysis.

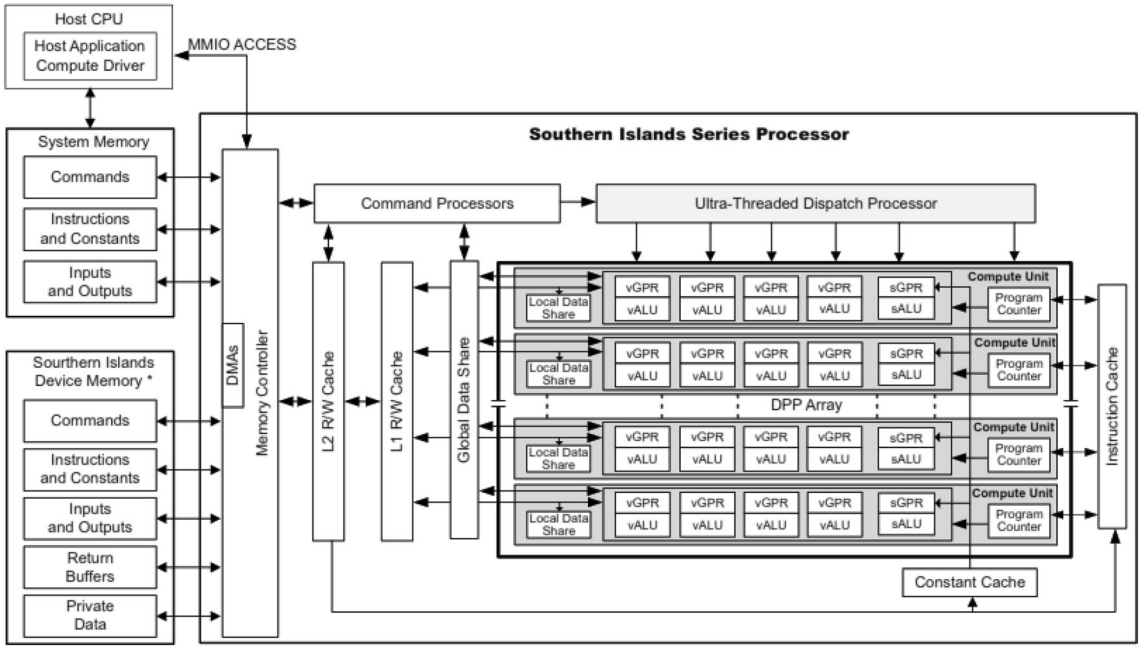

Fig. 1 AMD Southern Islands series block diagram [36]

Table 2 OpenCL SDK 2.5 benchmarks

Row	Program	Row	Program	Row	Program
1	BinarySearch	9	FFT	17	RadixSort
2	BinomialOption	10	FloydWarshall	18	Reduction
3	BitonicSort	11	Histogram	19	RecursiveGaussian
4	BlackScholes	12	MatrixMultiplication	20	ScanLargeArrays
5	DCT	13	MatrixTranspose	21	SimpleConvolution
6	DwtHaar1D	14	MersenneTwister	22	SobelFilter
7	EigenValue	15	PrefixSum	23	URNG
8	FastWalshTransform	16	QuasiRandomSequence		

3.3 Extracting the model

In this design space, considering the software parameters, there are about 5 billion points that should be evaluated. Obtaining the regression model for these number of points is very complicated. We decided that 5000 different configuration and simulation methods were chosen from all the design space points by the UAR method, and also 5000 points were chosen by Multi2Sim for each program which is simulated on Table 2, and the results were recorded. After execute each of simulation steps, IPC[1] is saved. IPC is parameter to predict performance of hardware and software. Then to build the basic model regression, 19 programs were selected between 23 benchmark programs, which obtained a regression model based on their data to predict graphics processor performance, and 4 other programs were used to validate the created model.

After obtaining the regression model, to verify the accuracy of the model, three criteria R_squared (R^2), MAE, RMSE were calculated, which respectively represent the average of errors root, errors mean, and prediction accuracy of model. $\mathbf{R^2}$ is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model. The $\mathbf{R^2}$ formula is calculated by dividing the sum of the first errors by the sum of the second errors and subtracting the derivation from 1. Here's what the R^2 equation looks like. Keep in mind that this is the very last step in calculating the R^2 for a set of data point. **RMSE**[2] and **MAE**[3] are both used to evaluate models. MAE gives equal weight to all errors, while RMSE gives extra weight to large errors.

[1] Instruction per cycle.

[2] Root mean squared error.

[3] Mean absolute error.

3.4 Statistical analysis

Sampling of the entire condition space results in 50 hardware parameters as their simulation results. The data set contains $19 \times 300 = 5700$ records. This information was analyzed using R, and as the result regression model was obtained.

The results of the initial analysis were not desirable, because the linear regression model was created to predict the application's performance of the graphic processors, it seemed regression equation should be a non-linear equation. Several measure were taken to improve the model. First, data was normalized using COXBOX functions, then the relationship between the predictor parameters and output were analyzed, and effective parameters were preserved. Then the relationship between each remaining predictor parameters and output were examined, and the type of relationship was defined. It became clear that some parameters don't have linear relationship with output. Therefore, their relationship was considered as a non-linear relationship with output, and polynomial equation was obtained from regression equation. The results were improved so much. Figure 1 shows an example of the relationship between one of the parameters with output.

To apply the possible relations between the parameters and the model, the relationship between each predictor parameters was studied, after defining the relationship between output parameters. Then applications were analyzed to improve the model. Some software parameters were added to the model to find out the effect of applications in the model. Table 3 shows software parameters which were extracted from applications like Binary Search application.

These parameters were collected by running application on base configuration called Baseline, and then they were normalized by COXBOX functions, and were added to the data set. All of the above procedures were done on new data sets to obtain the final model. In the next section, the results of the obtained model will be examined.

Table 3 Software parameter

Section	Parameter	Binary search
CPU	Integer	4,411,272
	Logic	72,095
	FloatingPoint	5789
	Memory	3,198,354
	Control	1,073,788
Program	ScalarALUInstructions	37
	ScalarMemInstructions	33
	BranchInstructions	5
	VectorALUInstructions	46
	LDSInstructions	0
	VectorMemInstructions	7

4 Experimental results

As shown in Table 4 and Fig. 2, prediction accuracy was very low, and the average error was high as well, while the model was created linearly using the hardware parameters. Finally, the model was changed to non-linear, and the relationship between predictor parameters were added with outputs to the model, and software parameters were also added to the model. Accuracy was reasonable, and the average error was also reduced. The results shown in row 9 are the result of regression non-linear model that contains hardware and software parameters. The predicted accuracy of the model is about 95.5% that predicts the results by an average error of about 7%.

4.1 Validation of the model

Until now, we found a model that predicts percentage of the graphics processors performance and the low error rate it has. The model should be evaluated to complete the task. There are various methods to evaluate the model; in this section two methods will be described:

(1) The first method was evaluated on outside of the sampled data, but on programs that were used to build the model.
(2) The second method was evaluated on programs that are not used to create the model.

4.1.1 First method: using efficient programs to create the model

In this paper, the programs which were used to build the model were also employed to evaluate it. Note that, we chose 300 samples of the entire design space and built the model. We chose 30 samples randomly from same space to evaluate the model in a way that those points were not a part of the model, then we merged them to obtain a data set to evaluate it. Then we evaluated the model with predict command in R and we calculated the three parameters, R^2, MAE, RMSE. Table 5 shows the results of this process.

As shown in Table 5, the evaluation accuracy of this method is about 88% and the average error is 0.12 and squared error is 0.09 which is appropriate to evaluate the model.

Table 4 Results of regression model

Row	Model	R^2	MAE	RMSE	
1	Reg	0.06568	0.77653	0.91064	Only hardware parameter
2	nor_reg	0.07675	0.29077	0.29067	
3	opt_normal_reg	0.09457	0.29152	0.29108	
4	poly_opt_nor_reg	0.10543	0.28931	0.28809	
5	poly_int_nor_reg	0.14562	0.28849	0.28743	
6	SW_nor_reg	0.59456	0.20248	0.19567	Hardware and software parameter
7	SW_opt_normal_reg	0.59319	0.20288	0.19597	
8	SW_poly_opt_nor_reg	0.92139	0.08934	0.10595	
9	SW_poly_int_nor_reg	0.9546	0.078	0.0803	

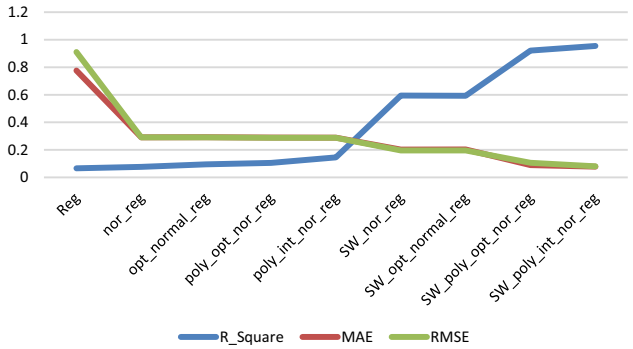

Fig. 2 Comparison of the results of the presented regression model

Table 5 Evaluation result for first method

	R^2	MAE	RMSE
Validation with same program (30)	0.88985	0.12329	0.09459

4.1.2 Second method: using the applications not affecting the modeling

In this method, programs which did not affect the model creation will be used to evaluate the model. OpenCL SDK 2.5 benchmark program had 23 different programs from which 19 programs were used to build the model. Four programs (histogram, Recursive Gaussian, SobelFilter and URNG) were not used to build the model. In this method, Histogram program was chosen to evaluate the model as the benchmark program. This means that from entire available design space were sampling with a different amount of points than three parameters, R^2, MAE, RMSE, were calculated. Table 6 shows this step results.

As it can be seen in the table, whenever the number of samples is increasing, the model has higher accuracy evaluation, and the average error is less. Finally, we calculated the average evaluation of all conditions which can be concluded nearly 85% of evaluation accuracy, and mean error is 0.1, and the mean square error is approximately 0.097. The following charts show that the greater number of samples will increase evaluation accuracy. In the following, all of the above steps were done on three other applications, Recursive Gaussian, SobelFilter and URNG, and the mean of fitting accuracy of the model, and the average error and the average mean square error were calculated for them.

In Table 7 and Fig. 3, the average results of the above programs are shown, so we wind up the model has been fitted with an accuracy of 0.88% and average error of about 0.09.

Table 6 Evaluation results for **Histogram** program

	R^2	MAE	RMSE
Other program (50)	0.74129	0.12765	0.11439
Other program (100)	0.77072	0.11652	0.10863
Other program (200)	0.79172	0.10672	0.10254
Other program (300)	0.83871	0.10256	0.10125
Other program (400)	0.84200	0.10199	0.09894
Other program (500)	0.88122	0.09913	0.09543
Other program (600)	0.90201	0.09543	0.09322
Other program (700)	0.90349	0.09623	0.09032
Other program (800)	0.92189	0.09357	0.08787
Other program (1000)	0.92361	0.09176	0.08542
Average	0.85167	0.10316	0.09780

Table 7 Average result of four programs

Program	R^2	MAE	RMSE
Histogram	0.851665	0.1031554	0.0978016
Gaussian Recursive	0.851196	0.0985546	0.1095403
Filter Sobel	0.888396	0.102611	0.089097
URNG	0.918042	0.0941421	0.082048
Average	0.877325	0.0996158	0.0946217

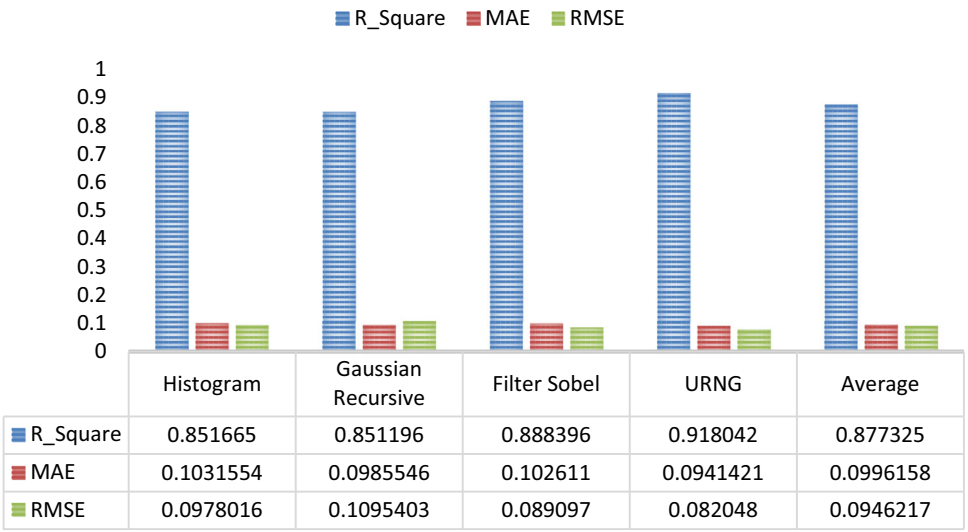

Fig. 3 Average result of four programs

	Histogram	Gaussian Recursive	Filter Sobel	URNG	Average
R_Square	0.851665	0.851196	0.888396	0.918042	0.877325
MAE	0.1031554	0.0985546	0.102611	0.0941421	0.0996158
RMSE	0.0978016	0.1095403	0.089097	0.082048	0.0946217

Table 8 Evaluation four program simultaneous

	R^2	MAE	RMSE
Other program (50)	0.601147	0.091613	0.086123
Other program (100)	0.818172	0.091351	0.0856735
Other program (200)	0.877857	0.091342	0.0856786
Other program (300)	0.885588	0.091459	0.0858692

In the following of the second method, from the above programs, we chose randomly 50, 100, 200 and 300 samples respectively and merged them together. The result from four files with the amount of 200, 400, 800 and 1200 samples were built, and files were validated, Table 8 shows the results of this step. As shown in Table 8 and previous tables, greater number of samples caused higher accuracy of built model. At this step, also the model has been fitted with an accuracy of 0.88%, and average error of about 0.08.

5 Results statistics analysis

We discussed the validity of the results in the previous section. In this section, we are going to proof the results through the statistics procedure.

5.1 Significant tests in regression model

In the multiple regressions, it is necessary to do two tests to make sure that the variables are significant. First test shows that the regression equation is significant. Second test shows each independent variable coefficients are significant.

Statistical significance the relation or difference between variables is analyzed using probability. The difference or relation between two variables can be random. Therefore, it's necessary to check the variable's difference or relation according to their significance level. The minimum acceptable level is 95% probability, which means if a difference or relation was observed between two variables, and the statistics calculations showed the relation or difference, it would be accidental to 5%, and would not be 95%, and influenced from experiment, the mentioned relation or difference would be significant.

5.2 Regression equation significance test

In a multiple regression equation, if there was no relation between the dependent variable and the independent variable, all of the independent variables coefficients in the equation must be zero. Thus, we can test the significance of the regression equation. In this design we will use the variance analysis and F statistics to check the significance of the regression equation.

In the F statistics, the zero hypothesis (H_0), is a hypothesis which must be tested and will present the lack of difference and similarity in the statistical population. The substitute hypothesis (H_1), is exactly opposite of zero hypothesis.

- Regression equation is not significant $(H_0) = \beta_1 = \beta_2 = \cdots = \beta_k = 0.$
- Regression equation is significant $(H_1) = \ni \beta_i \neq 0 \therefore i = 1 \; to \; k.$

The $F^* = \frac{MSR}{MSE}$ is the statistics of the test,F^* will be compared with the result value from fisher distribution table. If $F^* \leq F(1-\propto, p-1, n-p)$ then the zero assumption of the regression equation significance will be

Table 9 F-statistic results

Variables of model	DF	Sum_Sq	Mean_Sq	F_value	P-value	Significant level
poly(NumComputeUnits, 3)	3	90	30	324.73	< 2e−16	***
poly(FetchBufferSize, 3)	3	2	1	5.76	0.00063	***
poly(IssueBufferSize, 2)	2	9	5	51.03	< 2e−16	***
poly(DecodeLatency, 2)	2	3	2	16.55	6.80e−08	***
ALULatency	1	8	8	85.06	< 2e−16	***
ExecLatency	1	2	2	16.58	4.70E−05	***
poly(Integer, 3)	3	1275	425	4599.52	< 2e−16	***
poly(Logic, 3	3	432	144	1558.83	< 2e−16	***
poly(FloatingPoint, 3)	3	657	219	2368.15	< 2e−16	***
poly(Memory, 3)	3	659	220	2378	< 2e−16	***
poly(ScalarALUInstructions, 3)	3	346	115	1248.68	< 2e−16	***
poly(ScalarMemInstructions, 3)	3	129	43	464.98	< 2e−16	***
NumComputeUnits:FetchBufferSize	1	1	1	5.91	0.01513	*
NumComputeUnits:IssueBufferSize	1	3	3	35.09	3.30E−09	***
NumComputeUnits:DecodeLatency	1	1	1	11.89	0.00057	***
ALULatency:NumComputeUnits	1	0	0	0.63	0.42729	
ExecLatency:NumComputeUnits	1	0	0	0.33	0.56647	
FetchBufferSize:IssueBufferSize	1	3	3	33.37	8.00E−09	***
FetchBufferSize:DecodeLatency	1	0	0	1.07	0.30194	
ALULatency:FetchBufferSize	1	0	0	0.81	0.36744	
ExecLatency:FetchBufferSize	1	0	0	0.12	0.72826	
IssueBufferSize:DecodeLatency	1	0	0	0.51	0.47306	
ALULatency:IssueBufferSize	1	0	0	0.03	0.87019	
ExecLatency:IssueBufferSize	1	0	0	3.19	0.07419	.
ALULatency:DecodeLatency	1	0	0	0.42	0.51763	
ExecLatency:DecodeLatency	1	0	0	0.03	0.87389	
ALULatency:ExecLatency	1	0	0	1.86	0.0903	.

F-statistic: 871 on 45 and 5654 DF, p-value < 2e−16

Significance codes: 0 '***', 0.001 '**', 0.01 '*', 0.05, '.' 0.1, ' ' 1

accepted. In other words, it is safe to claim that the presented model from available independent variables are able to describe independent variable changes to $(1-\propto)$ %. If $F^* \geq F(1-\propto,\ p-1,\ n-p)$. On the other hand, the zero hypotheses are not significant, so the presented model cannot be suitable for the independent variable. The results of F statistics are shown in the below table. We can use P-value comparison, to examine (H_0). The P-value in statistics is one of the most important statistical inferences which show us the significant difference between models. In fact, the P-value is the minimum amount for $\propto$ which allows us to accept or decline (H_0) according to results obtained in the statistics examination.

The significant amount of statistics in the model above is 871 which will be compared with the F-statistic value in Table 9. According to P-value which is $2e^{-16}$ and less than 0.05 the (H_0) will be declined which shows the model is 95% significant and is able to estimate the applied programs efficiency on graphic processors.

5.3 Coefficients significance test

After the regression equation significance test, it is necessary to check the necessity of each estimating variables in the model. The purpose of this test in the current significance is to determine whether the calculated coefficient is zero. The t test can be used to analyze the average in the one-group univariate or 2-group multivariate researches. The hypothesis of the mentioned test is as follows.

- Population coefficient is zero $(H_0):\ \beta = 0$.
- Population coefficient is not zero $(H_0):\ \beta \neq 0$.

To test these hypothesizes; t-value statistics is used. If 95% of statistic significant was lesser than the t obtained

Table 10 t-value results

Variables of models	t-value	P-value	Significant level
poly(NumComputeUnits, 3)	49.76	< 2e−16	***
poly(FetchBufferSize, 3)	2.81	0.0049	**
poly(IssueBufferSize, 2)	2.32	0.02021	*
poly(DecodeLatency, 2)	− 3.3	9.60E−04	***
ALULatency	− 0.46	0.64673	
ExecLatency	0.59	5.53E−01	
poly(Integer, 3)	29.47	< 2e−16	***
poly(Logic, 3)	− 46.23	< 2e−16	***
poly(FloatingPoint, 3)	− 48.92	< 2e−16	***
poly(Memory, 3)	52.86	< 2e−16	***
poly(ScalarALUInstructions, 3)	− 37.77	< 2e−16	***
poly(ScalarMemInstructions, 3)	− 24.61	< 2e−16	***
NumComputeUnits:FetchBufferSize	3.58	0.00035	***
NumComputeUnits:IssueBufferSize	5.98	2.40E−09	***
NumComputeUnits:DecodeLatency	3.37	0.00076	***
ALULatency:NumComputeUnits	0.96	0.33709	
ExecLatency:NumComputeUnits	0.67	0.50393	
FetchBufferSize:IssueBufferSize	− 5.81	6.60E−09	
FetchBufferSize:DecodeLatency	− 1.11	0.26736	
ALULatency:FetchBufferSize	− 0.95	0.34328	
ExecLatency:FetchBufferSize	− 0.75	0.45085	
IssueBufferSize:DecodeLatency	− 0.71	0.47823	
ALULatency:IssueBufferSize	− 0.17	0.86446	
ExecLatency:IssueBufferSize	− 1.76	0.0782	.
ALULatency:DecodeLatency	0.59	0.55642	
ExecLatency:DecodeLatency	− 0.03	0.97459	
ALULatency:ExecLatency	− 1.69	0.0903	.

Significance codes: 0 '***', 0.001 '**', 0.01 '*', 0.05, '.' 0.1, ' ' 1

from the table with the same degree of freedom, the (H_0). is accepted, unless it's declined. Accepting the (H_0) shows the insignificancy of the coefficient and declining it expresses the (H_0). significance.

The test's statistic related to β_k is mentioned in the t-value column in Table 10. By using P-value and comparing it with 0.05, it is possible to decide about accepting or declining the zero hypotheses. The independent variables which are more than 0.05 are marked with another color in the significance table. Therefore, we can conclude that there is no significant relation between these dependent and independent variables. After determining significance of the estimating coefficients in the model, their positive or negative influence can be specifying using t-value.

6 Conclusion and future work

In this paper a non-linear model based on regression was presented to predict the performance of general-purpose graphic processors. In this regard, OpenCL SDK 2.5 application Programs were evaluated on it. In this model, We used IPC parameter to predict performance of graphic processor unit. First, 5000 different configuration and simulation methods were chosen from all the design space points by the UAR method and also 5000 points were chosen by Multi2Sim for each program which is simulated, and the IPC were recorded. Then we used R tools to make regression model. This model predict efficiency programs about 95% accuracy with error rate about 7%. In the evaluation section of the model, the proposed model predict other programs performance with 88% accuracy, and the error rate about 0.1. In the future work, we will use data mining approach to improve accuracy, and reduce error rate. Also we will add CPU parameters to make the regression model to improve accuracy.

References

1. Puttaswamy, K., et al.: System level power-performance trade-offs in embedded systems using voltage and frequency scaling of off-chip buses and memory. In: Proceedings of the 15th International Symposium On System Synthesis. ACM (2002)
2. Meyer, B.H., et al.: Power-performance simulation and design strategies for single-chip heterogeneous multiprocessors. IEEE Trans. Comput. **54**(6), 684–697 (2005)
3. Park, Y.-H., et al.: System-level power-performance trade-offs in bus matrix communication architecture synthesis. In: Hardware/Software Codesign and System Synthesis, 2006. CODES+ISSS'06. Proceedings of the 4th International Conference. IEEE (2006)
4. Top 500.: Available from: https://www.top500.org/lists/2019/06/. Accessed June 2019
5. Green 500.: Available from: https://www.top500.org/green500/lists/2019/06/. Accessed June 2019
6. Thompson, M., et al.: A mixed-level co-simulation method for system-level design space exploration. In: Proceedings of the 2006 IEEE/ACM/IFIP Workshop on Embedded Systems for Real Time Multimedia. IEEE (2006)
7. Baghsorkhi, S.S., et al.: Analytical performance prediction for evaluation and tuning of GPGPU applications. In: Workshop on EPHAM2009, in Conjunction with CGO, Citeseer (2009)
8. McClanahan, C.: History and Evolution of GPU Architecture. A Survey Paper, p. 9 (2010)
9. Power, J., et al.: gem5-gpu: a heterogeneous CPU–GPU simulator. IEEE Comput. Archit. Lett. **14**(1), 34–36 (2015)
10. Kothapalli, K., et al.: A performance prediction model for the CUDA GPGPU platform. In: 2009 International Conference on High Performance Computing (HiPC). IEEE (2009)
11. Hong, S., Kim, H.: An integrated GPU power and performance model. In: ACM SIGARCH Computer Architecture News. ACM (2010)
12. Lee, B.C., Brooks, D.: Applied inference: case studies in microarchitectural design. ACM Trans. Archit. Code Optim. **7**(2), 8 (2010)
13. Schafer, B.C., Wakabayashi, K.: Design space exploration acceleration through operation clustering. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **29**(1), 153–157 (2010)
14. Meng, J., et al.: GROPHECY: GPU performance projection from CPU code skeletons. In: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. ACM (2011)
15. Song, S., et al.: A simplified and accurate model of power-performance efficiency on emergent GPU architectures. In: 2013 IEEE 27th International Symposium on Parallel and Distributed Processing (IPDPS). IEEE (2013)
16. Moren, K., Göhringer, D.: Automatic Mapping for OpenCL-Programs on CPU/GPU Heterogeneous Platforms. Springer, Cham (2018)
17. Azizi, O., et al.: An integrated framework for joint design space exploration of microarchitecture and circuits. In: Design, Automation and Test in Europe Conference and Exhibition (DATE), 2010. IEEE (2010)
18. Dubach, C., et al.: A predictive model for dynamic microarchitectural adaptivity control. In: Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture. IEEE Computer Society (2010)
19. Kerr, A., Diamos, G., Yalamanchili, S.: Modeling GPU–CPU workloads and systems. In: Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units. ACM (2010)
20. Kerr, A., et al.: Eiger: a framework for the automated synthesis of statistical performance models. In: 2012 19th International Conference on High Performance Computing (HiPC). IEEE (2012)
21. Ubal, R., et al.: Multi2Sim: a simulation framework for CPU–GPU computing. In: 2012 21st International Conference on Parallel Architectures and Compilation Techniques (PACT). IEEE (2012)
22. Benatia, A., et al.: BestSF: a sparse meta-format for optimizing SpMV on GPU. ACM Trans. Archit. Code Optim. **15**(3), 29 (2018)
23. Sun, Y., et al.: MGSim+ MGMark: A Framework for Multi-GPU System Research (2018). arXiv preprint arXiv:1811.02884
24. Joseph, P., Vaswani, K., Thazhuthaveetil, M.J.: A predictive performance model for superscalar processors. In: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture. IEEE Computer Society (2006)
25. Hu, L., Che, X., Zheng, S.-Q.: A closer look at GPGPU. ACM Comput. Surv. **48**(4), 60 (2016)
26. Issa, J.: Processor performance modeling using regression method. In: 2016 18th Mediterranean Electrotechnical Conference (MELECON). IEEE (2016)
27. Gianniti, E., Zhang, L., Ardagna, D.: Performance prediction of GPU-based deep learning applications. In: Conference: 2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD) (2018)
28. Mukherjee, R., Rehman, M.S., Kothapalli, K., Narayanan, P.J, Srinathan, K.: Fast, Scalable, and Secure encryption on the GPU (2014)
29. Jia, W., et al.: GPU performance and power tuning using regression trees. ACM Trans. Archit. Code Optim. **12**(2), 13 (2015)
30. Siavashi, A., Momtazpour, M.: GPUCloudSim: an extension of CloudSim for modeling and simulation of GPUs in cloud data centers. J. Supercomput. (2018). https://doi.org/10.1007/s11227-018-2636-7
31. Wu, W., Lee, B.C.: Inferred models for dynamic and sparse hardware–software spaces. In: Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture. IEEE Computer Society (2012)
32. Huang, B., et al.: Development of a GPU-based high-performance radiative transfer model for the Infrared Atmospheric Sounding Interferometer (IASI). J. Comput. Phys. **230**(6), 2207–2221 (2011)
33. Jia, W., Shaw, K.A., Martonosi, M.: Stargazer: automated regression-based GPU design space exploration. In: 2012 IEEE International Symposium on Performance Analysis of Systems & Software. IEEE (2012)
34. Ahmad, L.G., et al.: Using three machine learning techniques for predicting breast cancer recurrence. J. Health Med. Inf. **4**(124), 3 (2013)
35. Khalaf, M., et al.: A data science methodology based on machine learning algorithms for flood severity prediction. In: 2018 IEEE Congress on Evolutionary Computation (CEC) (2018)
36. Reference Guide: Southern Islands Series Instruction Set Architecture.: Rev. 1.0, Aug. 2012. http://developer.amd.com/wordpress/media/2012/10/AMD_Southern_Islands_Instruction_Set_Architecture.pdf

Mohammad Hossein Shafiabadi received his B.S. degree from Shahid Beheshti University in 2001 and M.S. degree from AmirKabir University in 2004 in Computer Engineering. He received his Ph.D. degree from Islamic Azad University. He has served as a faculty member in the Computer Engineering Department in Islamic Azad University EslamShahr Branch since 2005. He teaches courses in computer architecture, Operating systems and Network. His research interests include innovative methods in computer architecture such as Nano circuits, Asynchronous Circuit, management of computer networks, Modeling and Simulation, distributed systems, and Cloud Computing.

Hossein Pedram received his B.S. degree from Sharif University in 1977 and M.S. degree from Ohio State University in 1980 in Electrical Engineering. He received his Ph.D. degree from Washington State University in 1992 in Computer Engineering. Dr Pedram has served as a faculty member in the Computer Engineering Department in Amirkabir University of Technology since 1992. He teaches courses in computer architecture and distributed systems. His research interests include innovative methods in computer architecture such as synchronous circuits, management of computer networks, distributed systems, and robotics. Home page http://ceit.aut.ac.ir/∼pedram.

Midia Reshadi is currently an Assistant Professor in Computer Engineering Department at Science and Research Branch of Azad University since 2010. His research interest is Network-on-chip including performance and cost improvement in topology, routing and application-mapping design levels of various types of NoCs such as 3D, photonic and wireless. Recently, he has started carrying out research in NoC based deep neural network accelerators and Silicon interposer based NoC with his team consists of M.Sc. and Ph.D. students.

Akram Reza received B.Sc. degree in computer hardware engineering from Qazvin Branch, Islamic Azad University, Qazvin, Iran, in 2004. She also received his M.Sc and Ph.D. degrees in Computer Architecture at the Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran, in 2008 and 2014, respectively. She is currently employ in the Department of computer Engineering, Share-e-Qods Branch, Islamic Azad University, Tehran, Iran. Her research focuses on different aspects same as power and thermal management on homogeneous and heterogeneous multiprocessor system, scheduling, allocation and Parallel Systems Architecture. Also she work on chip structure for run deep leaning, machine learning and other artificial intelligence algorithm with high performance on multiprocessors chip.