ISSN 1112-9867

Special Issue

Available online at http://www.jfas.info

HIGH SPEED NUMERICAL INTEGRATION ALGORITHM USING FPGA

F. N. A. Razak[1], M. S. A. Talip[1,*], M. F. M. Yakub[2], A. S. M. Khairudin[1], T. F. T. M. N. Izam[1] and F. H. K. Zaman[3]

[1]Department of Electrical Engineering, Faculty of Engineering, University of Malaya, 50603 Kuala Lumpur, Malaysia

[2]Department of Electronic Systems Engineering, Malaysia-Japan International Institute of Technology, Universiti Teknologi Malaysia, 54100 Kuala Lumpur, Malaysia

[3]Department of Computer Engineering, Universiti Teknologi MARA, 40450 Shah Alam, Selangor, Malaysia

Published online: 05 October 2017

ABSTRACT

Conventionally, numerical integration algorithm is executed in software and time consuming to accomplish. Field Programmable Gate Arrays (FPGAs) can be used as a much faster, very efficient and reliable alternative to implement the numerical integration algorithm. This paper proposed a hardware implementation of four numerical integration algorithms using FPGA. The computation is based on Left Riemann Sum (LRS), Right Riemann Sum (RRS), Middle Riemann Sum (MRS) and Trapezoidal Sum (TS) algorithms. The system performance is evaluated based on target chip Altera Cyclone IV FPGA in the metrics of resources utilization, clock latency, execution time, power consumption and computational error compared to the other algorithms. The result also shows execution time of the FPGA are much faster compared to the software implementation.

Keywords: numerical integration algorithm; FPGA; Riemann sum; trapezoidal sum.

Author Correspondence, e-mail: sofian_abutalip@um.edu.my

doi: http://dx.doi.org/10.4314/jfas.v9i4s.7

1. INTRODUCTION

Recently, the High Performance Computing (HPC) system is the fastest growing area of computing in industries. It is related to the used of parallel processing for any complex and intensive application program to run faster, efficient and reliable. Numerical integration method which have long been used in the area of computing, is used in order to find the approximate solutions of a definite integral by using the software implementation. In other words, the integral can be known as the data obtained by sampling. Nowadays, many applications such as in the field of probability has used this function to describe the data sets. The numerical integration is an approximation, so error analysis is a very important thing. In order to achieve a better performance in implement numerical integration and since the high speed and accurate arithmetic is absolutely essential, the hardware oriented solution is developed upon the setback trending [1].

Since a high level of hardware programmability is provided, the Field Programmable Gate Arrays (FPGAs) is the platform of choice for the implementation. It can perform custom hardware functionality using the logic blocks and programmable routing resources. FPGA is a reconfigurable hardware, which has the same software flexibility running on a processor but not limited to the number of cores. The performance of the FPGA is not affected by another processing since it is a parallel processing. The FPGA chip used in this project was an Altera Cyclone IV FPGA chip which was realized using Quartus II software to design the desired circuits in Verilog Hardware Description Language (HDL). The desired circuits were designed based on four integration algorithms; Left Riemann Sum (LRS), Middle Riemann Sum (MRS), Right Riemann Sum (RRS) and Trapezoidal Sum (TS). The quality for implementation of different algorithms on a single chip was evaluated using various metrics performance and were compared to software implementation like MATLAB and C++.

In previous studies, the software implementation did not meet the desired evaluations for its sequential structure. Sometimes, there will be difficulties to compute the definite integral since it is for immobile embedded applications. Therefore, many applications can take advantages from the speed performance, area, power efficiency and memory resources when using the reconfigurable hardware. The computation also can be implemented anywhere and

anytime as long as the hardware is presented.

2. RELATED WORKS

Numerical simulation has been used in wide range of applications [6, 8]. FPGA in an attractive platform for the computation from electronics applications to engine system [7]. A hardware structure for numerical integration can be obtained by basic mapping of Trapezoidal sum on FIR structure. Parallel structure can be obtained by converting a single-input single-output system into a multiple-input multiple-output system. For example, in a 4-parallel structure, the critical path does not change, but with four samples are processed in a clock cycle, the sampling frequency is four times compared to the original frequency. In a large input word length, the multiplication unit need to break into smaller units. This is to make sure that the sampling frequency is increased, hence increasing the number of clock cycles. It shows that this approach will reduce power dissipation by reducing the value of voltage [2].

In [2] pressed that the complexity of a problem will not allow mapping onto specific FPGA. Thus, an analysis on the arithmetic unit precision can solve the computational problem to allow mapping to smaller FPGAs and have high accuracy. Numerical algorithm can be implemented using fixed-point or floating point arithmetic with different precision. During FPGA computations, 64-bit floating point numbers are used to reach an appropriate accuracy but consumes more computing power. Significant speedup can be achieved by decreasing the state precision, which makes it possible mapping to some particularly complex problems onto an FPGA. For problems without analytic solution, the reduced precision results can be compared to the 64-bit floating point reference precision.

In [3] mentioned that the time and power consumption had improved when reducing the FPGA resources. By using the FPGA, a SIMD architecture was designed in a floating point precision. The reduction of the resources had allocated the operation cores in a shared operation block. The pipelined architecture was able to execute an instruction per cycle and this had done by a compiler. The compiler had reorganized the instructions to exploit the architecture in maximum. The language used is a C programming on a Xilinx FPGA chip.

This had proved that the pipelined structure had minimized the loss in scalability when implemented on the FPGA chip.

In [4] proposed a hardware implementation of ANN-based chaotic generator in FPGA. The chaotic generator was reconstructed by the Feed Forward Neural Network (FFNN), which was created using MATLAB. Meanwhile, the hardware was implemented on a Xilinx Virtex 6 chip. Its architecture was presented in VHDL. All parameters were related to a single precision floating point number format. The design performance was achieved successfully when compared to the outputs of the chaotic generator. The accurateness of the design also was very high with low errors.

The parallel implementation on CPU, GPU and FPGA were compared by [5]. They have proved that between the three implementations of Jacobi algorithm for matrix, the FPGA design gives the best task performance but the GPU had a superior scalability, while CPU was a poor computing platform. A systolic array structure was designed to support various matrix sizes on the FPGA implementation and thus, gives the best computing performance of matrices compared the other two [9-10].

3. NUMERICAL ALGORITHM INTEGRATION

Numerical integration algorithm is an approximation of differential equation and a computation of a definite integral, $\int_a^b f(x)\,dx$. There are four integration algorithms being used as a method for the numerical integration approximated to a desired precision such as LRS, MRS, RRS and TS. The best method is a systematic technique with less arithmetic operations involved resulting less error for small evaluations.

3.1. Riemann Sum

Riemann Sum consists of area summation for all vertical rectangles of height *f(x)* with base dx under a curve. The function f is approximated by the value at the end point of each rectangle with the dx, which the height is *f(a+i*dx)*. The base dx is equal to the interval of lower a and upper b limits, which is then divided with the total number of equal rectangles n.

$$\Delta x = \frac{b-a}{n} \quad (1)$$

Fig. 1 shows a linear graph of LRS. The value of i represents the red dotted from the left end point of each rectangle, which is from 0 to $n-1$ where $n=5$.

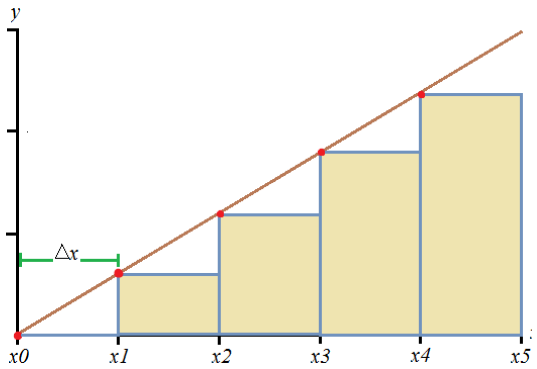

Fig.1. Left Riemann Sum (LRS)

Area of LRS $= \sum_{n=0}^{n-1} y(a+i*dx)dx$

$\approx [y(a)+y(a+dx)+y(a+2*dx)+\cdots+y(b-dx)]dx$ (2)

Area of LRS from Fig. 1

$\approx [y(a)+y(a+dx)+y(a+2*dx)+y(a+3*dx)+y(b-dx)]dx$ (3)

Fig. 2 shows a linear graph of RRS. The value of i represents the red dotted from the right end point of each rectangle which is from 1 to n where $n=5$ only for this case.

Area of RRS $= \sum_{n=1}^{n} y(a+i*dx)dx$

$\approx [y(a+dx)+y(a+2*dx)+\cdots+y(b)]dx$ (4)

Area of RRS from Fig. 2

$\approx [y(a+dx)+y(a+2*dx)+y(a+3*dx)+y(a+4*dx)+y(b)]dx$ (5)

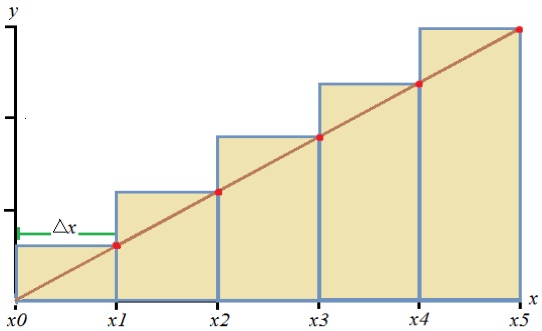

Fig.2. Right Riemann Sum (RRS)

Fig. 3 shows a linear graph of MRS. The function f is approximated at the middle point of each rectangle which initially gives $f\left(a+\left(\frac{dx}{2}\right)\right)$, $f\left(a+\left(3*\frac{dx}{2}\right)\right)$ and so on until $f\left(b-\left(\frac{dx}{2}\right)\right)$.

Area of MRS $= \sum_{n=0}^{n} y(a+i*dx)dx$

$$\approx [y(a + dx/2) + y(a + 3 * dx/2) + \cdots + y(b - dx/2)]dx \quad (6)$$

Area of MRS from Fig. 3

$$\approx \left[y\left(a + \frac{dx}{2}\right) + y\left(a + 3 * \frac{dx}{2}\right) + y\left(a + 5 * \frac{dx}{2}\right) + y\left(a + 7 * \frac{dx}{2}\right) + y\left(b - \frac{dx}{2}\right)\right] dx \quad (7)$$

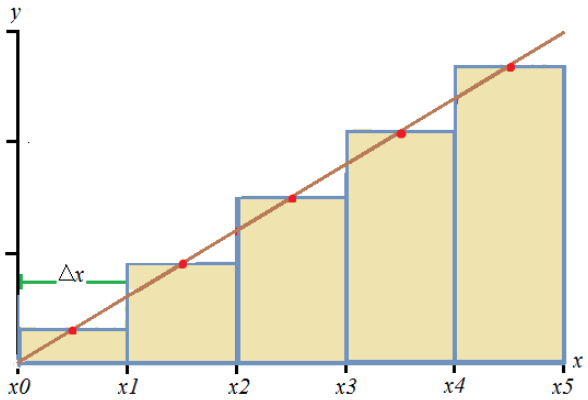

Fig.3. Middle Riemann Sum (MRS)

3.2. Trapezoidal Sum

As we all know, the area of a trapezium is equal to the average length of its parallel sides multiply by the distance between them. So, adding all the area of trapezoids will give the approximate area under the curve. If the number of trapezoid is large, they almost cover all the area under the curve which gives a better approximation.

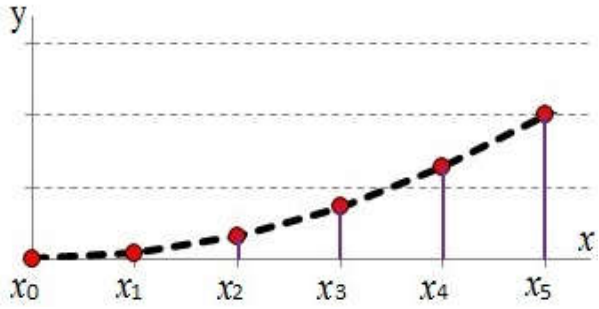

Fig.4. Trapezoidal Sum

Area of TS $= \int_a^b y(x)\, dx$

$$\approx dx[\frac{y(a)}{2} + y(a + dx) + y(a + 2 * dx) + \cdots + y(a + (n - 1)dx) + \frac{y(b)}{2}] \quad (8)$$

Area of TS from Fig. 4

$$\approx \Delta x[\frac{y(a)}{2} + y(a + \Delta x) + y(a + 2\Delta x) + y(a + 3\Delta x) + y(a + 4\Delta x) + \frac{y(b)}{2}] \quad (9)$$

4. DESIGN AND IMPLEMENTATION

In product development [11] process, it starts with defining the specification of the product. The designed products are developed and simulated using to determine the required specification on the designs. A prototype is implemented and thoroughly tested whether the design meets the requirement or not.

Table 1. Implementation environment

Name	Tools
Hardware description language (HDL)	Verilog HDL
Field Programmable Gate Array (FPGA)	Cyclone IV (EP4CE115F29C7)
Synthesis	Quartus II 15.0
Simulation	ModelSim-Altera
Programming	Quartus II Programmer
Library of parameterized module (LPM)	Megawizard Plug-in Manager

Table 1 displays the implementation environment in this project. The hardware platform used is a DE2-115 board with the lowest cost Altera's Cyclone IV E FPGA chip. The implementation will be realized using Quartus II software and the circuits are designed in Verilog HDL language. The FPGA chip is built on an optimized low power process with high functionality and lowest cost without sacrificing its performance. Its resources are shown in Table 2.

Table 2. Cyclone IV E device resources

Resources	EP4CE115
Logic elements (LEs)	114,480
Embedded memory (Kbits)	3,888
Embedded 18x18 multipliers	266
General purpose PLLs	4
Global Clock Networks	20
User I/O Banks	8
Maximum user I/O	528
Data size (bits)	28,571,696

LPMs are needed in order to create architecture independent designs that have high silicon efficiency. The LPM functions are supported by the Quartus II software and consists of 25 functions at the present time. Instead of using own coding, the functions can save time and have more efficient logic synthesis. They are used to simplify the complexity of a design in the FPGA chip.

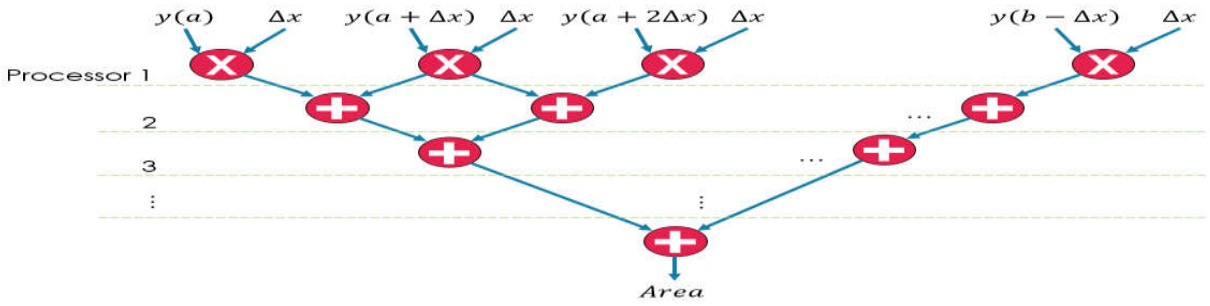

Fig.5. Data flow graph for general circuit design

Fig. 5 shows a data flow graph for a general circuit design that represent the four integration algorithms. In parallel processing, the query is divided into multiple smaller tasks. Each task executes immediately on its own processor and no delay time is involved.

5. PERFORMANCE EVALUATION

We evaluated our system performance based on the following performance metrics: resource utilization, clock latency, execution time, power dissipation and computational error.

5.1. Resource Utilization

Fig. 6 shows the graph of resource utilization. The amount of resources for LRS and RRS were approximately the same for the combinational functions, logic registers and pins. The total logic elements for the LRS was higher than the RRS, but for the total memory bits and the amount of embedded multiplier, the LRS was lower than the RRS. The embedded multiplier for the RRS had the same amount with the MRS. The amount of other resources for the MRS were much higher than the LRS and RRS. Comparing the amount of resources for the four algorithms, the TS had the highest amount of resources compared to others. But the total pin was the same as the MRS. These results showing that the highest amount of resources utilization has more complexity on the design.

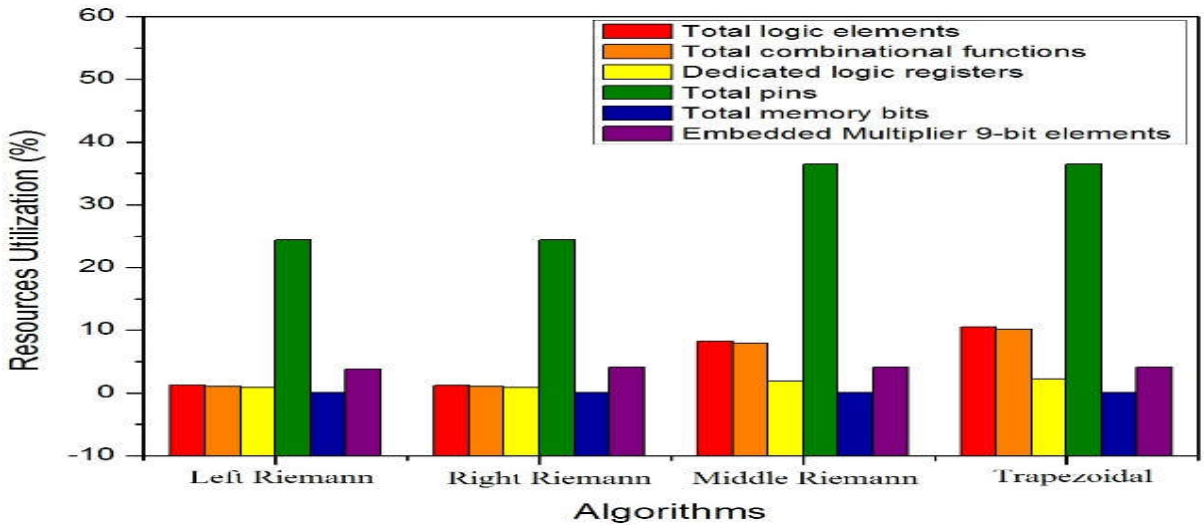

Fig.6. Resource utilization for linear function

5.2. Clock Latency

The time for the complete output to produce is called a latency. The FPGA can be used to reduce the latency. This gives extremely precise timing and very reliable. It has flexibility since no physical changes need to be happened to the device to change its behavior, but has the speed of hardware which is executing at clock rates up to the megahertz range. A graph of latency is shown in Fig. 7. From the graph, the number of clock cycle for the LRS has the same amount as the number of clock cycle for the RRS. But, both has lower amount of clock cycle when compared to the MRS. TS had the highest latency compared to others.

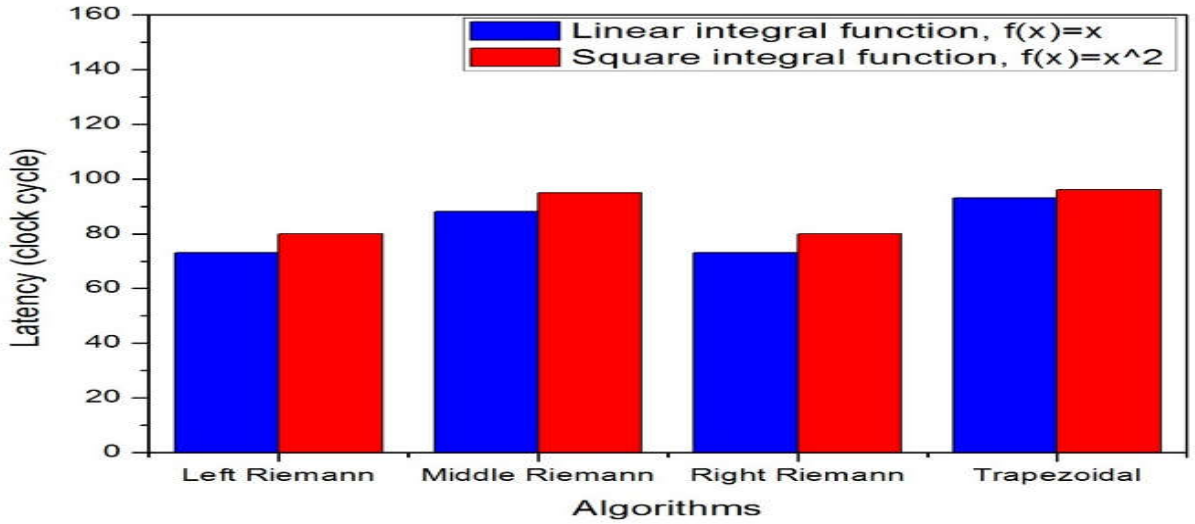

Fig.7. Clock latency

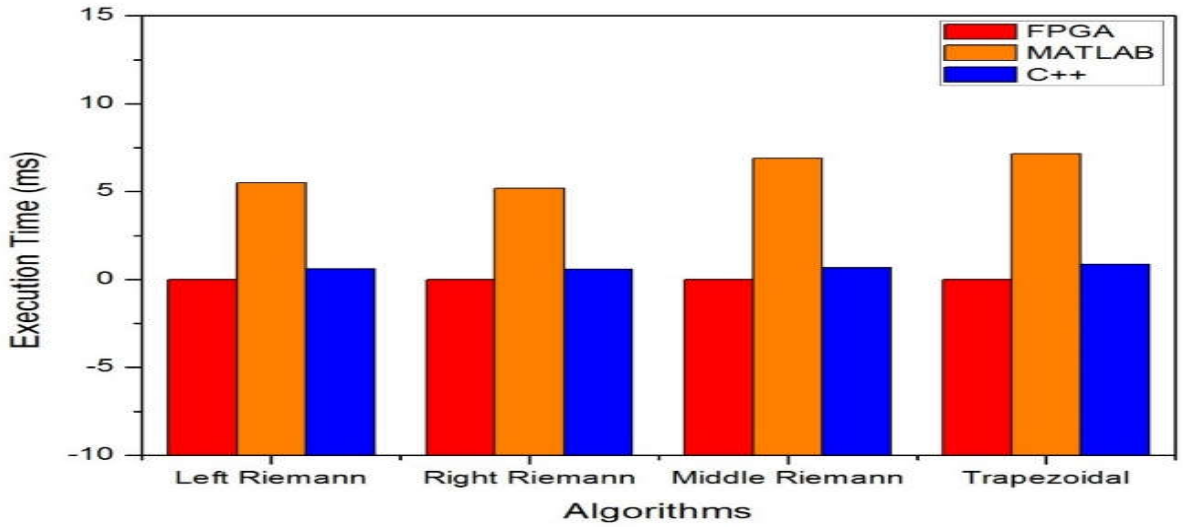

Fig.8. Comparison between FPGA, MATLAB and C++ for execution time

5.3. Execution Time

Execution time for the four algorithms are compared between the FPGA, MATLAB and C++ to show the speed performance on difference computations. In the FPGA's row, it shows that the TS had the slowest execution time compared to the three Riemann Sums. The complexity of a design makes the output to produce slower than the simple design. The MRS was the slowest in the Riemann Sums, whereas LRS and RRS had approximately the same and the fastest execution time when comparing to the other algorithms. The speed performance for the FPGA chip is improved by the factor of 3,500,000 to 3,800,000 times faster than MATLAB and 380,000 to 410,000 times faster than C++. It shows that the speed performance for FPGA is the fastest compared to others, then followed by C++ and MATLAB computations.

5.4. Power Consumption

In estimating the power dissipation accurately for these designs in the chip, a *PowerPlay* Power Analysis tool is used to develop an appropriate power budget. The FPGA's power dissipation can be calculated from the *PowerPlay* Early Power Estimator (EPE) spreadsheet with Microsoft Excel-based. In the *Quartus II* software, a *PowerPlay* Power Analyser (PA) also can be used to calculate more accurate power dissipation.

From Fig. 9, both LRS and RRS produce the same total of power dissipation. Both MRS and TS also give the same total power dissipation, but were higher than the LRS and RRS. Comparing the power dissipation in Fig. 10, the TS had the highest total power dissipation than the total power dissipation for the Riemann Sums. Then, followed by the MRS, LRS and

RRS. When comparing to the EPE results for the four algorithms, the static power dissipation was reduced and the dynamic power dissipation was increased.

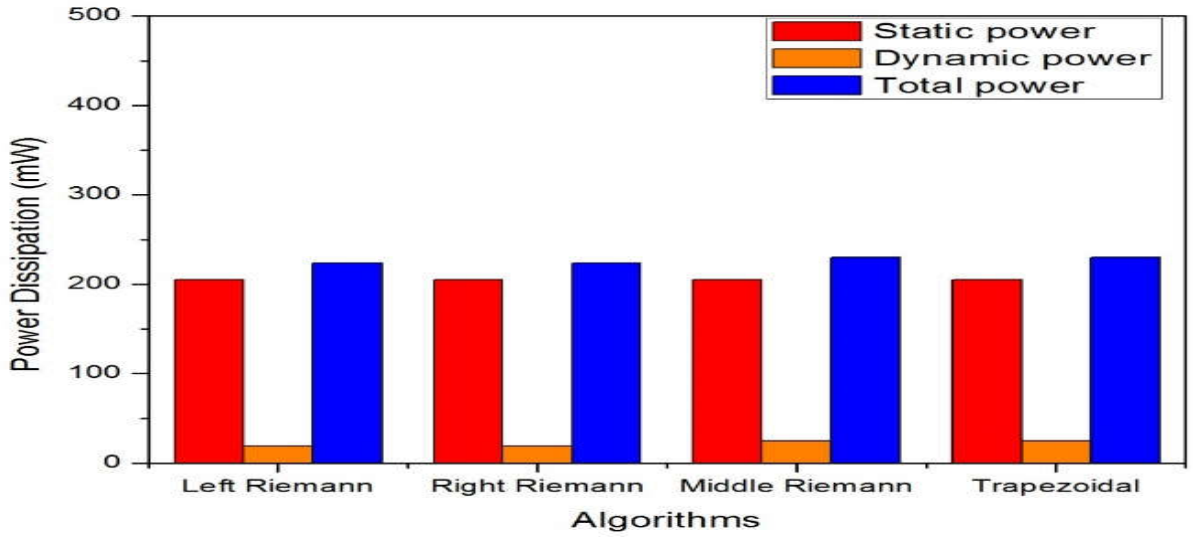

Fig.9. Power estimation

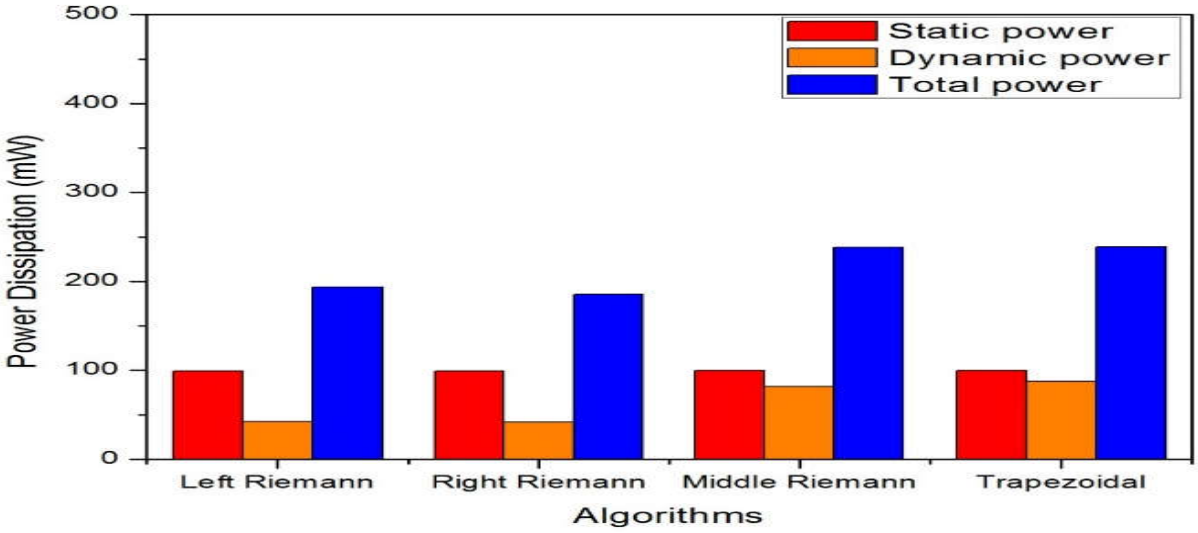

Fig.10. Power dissipation

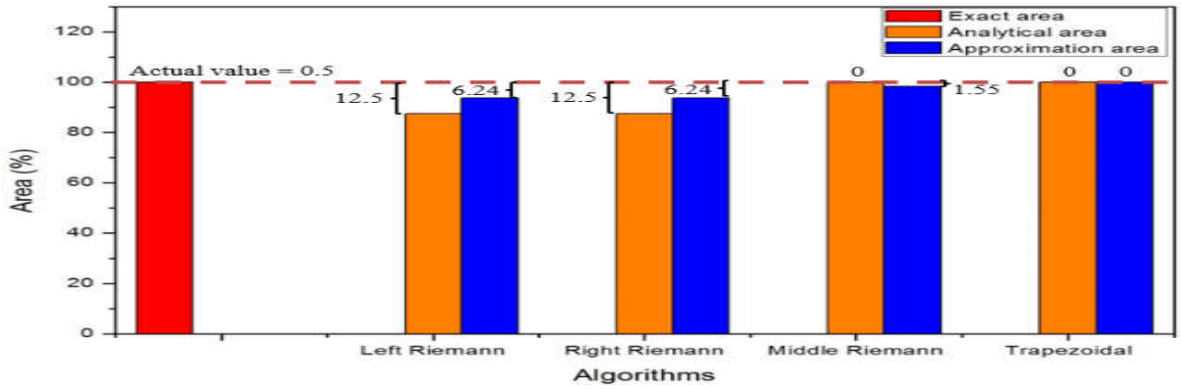

Fig.11. Computational error

The results compiled by each design is studied to analyze the behavior of its approximation error. The input for a and b were key-in with n = 8, where the given n is the maximum length of sub-interval that can fit-in in the desired designs. The fit-in design is due to the used of fixed point binary numbers. The analytical value is an area estimation calculated manually based on Equation (3) for LRS, Equation (5) for RRS, Equation (7) for MRS and Equation (9) for the TS. Meanwhile, the approximation value is taken from the compilation data of functional simulation. The data is represented in a fixed-point number. The error is then calculated and compared between the analytical and the approximation value. This had shown in a graph of Fig. 11. The analytical error for LRS had approximately the same value as the RRS. Meanwhile, the MRS and TS had zero error. The approximation area is taken from the functional simulations representing the fixed point binary numbers. From the result, the LRS and RRS had the highest error compared to others. Meanwhile, the TS produces zero error.

6. CONCLUSION

In this project, hardware implementation of numerical integration algorithm using FPGA is proposed. All the performance metrics evaluation was resulting from the implementation of the four algorithms of numerical integration using Cyclone IV FPGA chip. The desired circuits were designed using the LPM in the Quartus II software to save the designing time. The number of resources increased as the circuit design becomes more complicated. The result shows that the TS had more complicated design compared to others. The FPGA chip also meets the speed performance which was much faster than the software implementation such as C++ and MATLAB and more complicated circuit increases the execution time. The power dissipation was reduced when compared to the early estimation which was before run the compilation. The complexity of the design also effects the power efficiency. More complex on the design causing higher power consumption. Lastly, the computational error for the FPGA computational was approximately ~10%. The error is caused by the used of fixed point binary numbers.

As a future work to further improve and extend this project, there are three recommendations that can be modified. First, the circuit designs can use a floating point number to achieve greater precision and maintain its speed performance. Second, the use of multi-stages pipelined. It is a technique that implements parallelism within a single processor. Lastly, the

used of multi-FPGAs. The multi-FPGA systems contain more than one chip, and are constrained by the inter-chip connection topology.

7. ACKNOWLEDGEMENTS

This work was supported by the Fundamental Research Grant Scheme (FRGS), Grant No. FP042-2014B.

8. REFERENCES

[1] Khurshid B, Mir R N. A hardware intensive approach for efficient implementation of numerical integration for FPGA platforms. In 27th IEEE International Conference on VLSI Design and 13th International Conference on Embedded Systems, 2014, pp. 312-317

[2] Kiss A, Nagy Z, Csik Á, Szolgay P. Examining the accuracy and the precision of PDEs for FPGA computations. In 13th International Workshop on Cellular Nanoscale Networks and their Applications, 2012, pp. 1-5

[3] Carrascosa J C, Mas J R, del Moral B A, Balaguer M, Jiménez A L, del Toro Iniesta J C. SIMD architecture on FPGA for scientific computing aboard a space instrument. Journal of Systems Architecture, 2016, 62:1-11

[4] Alçın M, Pehlivan İ, Koyuncu İ. Hardware design and implementation of a novel ANN-based chaotic generator in FPGA. Optik-International Journal for Light and Electron Optics, 2016, 127(13):5500-5505

[5] Torun M U, Yilmaz O, Akansu A N. FPGA, GPU, and CPU implementations of Jacobi algorithm for eigenanalysis. Journal of Parallel and Distributed Computing, 2016, 96:172-180

[6] Zexi D, Feidan H. Cuckoo search algorithm for solving numerical integration. In IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems, 2015, pp. 1508-1512

[7] Osorio R R. Pipelined FPGA implementation of numerical integration of the Hodgkin-Huxley model. In IEEE 27th International Conference on Application-Specific Systems, Architectures and Processors, 2016, pp. 202-206

[8] Drexler D A, Kovács L. Second-order and implicit methods in numerical integration improve tracking performance of the closed-loop inverse kinematics algorithm. In IEEE

International Conference on Systems, Man, and Cybernetics, 2016, pp. 003362-003367

[9] Ueno T, Sano K, Yamamoto S. Bandwidth compression of floating-point numerical data streams for FPGA-based high-performance computing. ACM Transactions on Reconfigurable Technology and Systems, 2017, 10(3):1-22

[10] Liu S, Han J. Hardware ODE solvers using stochastic circuits. In 54th ACM Annual Design Automation Conference, 2017, pp. 1-6

[11] Fadhlan H K Z, Md. Hazrat A, Amir A S, Zairi I R. Development of mobile face verification based on locally normalized gabor wavelets. International Journal on Advanced Science, Engineering and Information Technology, 2017, 7(3):1026-1031

How to cite this article:
Razak FNA, Talip MSA,. Yakub MFM, Khairudin ASM, Izam TFTMN, Zaman FHK
. High speed numerical integration algorithm using FPGA. J. Fundam. Appl. Sci., 2017, 9*(4S), 131-144.*