Nigerian Journal of Technology (NIJOTECH)
Vol. 41, No. 1, January, 2022, pp.131 –139
www.nijotech.com

Print ISSN: 0331-8443
Electronic ISSN: 2467-8821
http://dx.doi.org/10.4314/njt.v41i1.17

Power Performance Comparison Analysis of FPGA Using Riemann Definite Integral Equation Solver

M. Adamu[1, *], A. I. Audu[2], C. U. Ngene[3]

[1]*Department of Computer Science, College of Education and Legal Studies, Nguru, Yobe State, NIGERIA*
[2,3]*Department of Computer Engineering, University of Maiduguri, Maiduguri, Borno State, NIGERIA*

Abstract
Definite integral equation solvers were hitherto developed using software which take long time to generate results. The complexity of designs and verifications, due to advancement in calculus and linear algebra, data dependency in designs, and the need for parallelism in models, necessitated the need for faster, accurate, and reliable hardware-based solvers. This will reduce latency and excessive memory requirement in generating values of definite integrals. The Field Programmable Gate Array (FPGA)-Based Definite Integral Solver developed in this research is able to solve definite integral equations with less resource utilization (1.08% look up tables (LUT), 0.07% flip flops (FF), 10.81% digital signal processing (DSP) and 23.86% input-output (IO)) and hence minimum switching activities at lower power (Signals: 0.421W, Logic: 0.345W, Digital Signal Processing: 1.904W and Input/Output: 0.015W) that translates to fast and accurate results generation. The power and resource utilization evaluation on the solver and the performance comparison with similar works showed that excessive memory consumption and resource utilization has been reduced significantly by the design.

Keywords: Field Programmable Gate Array, FPGA, Definite Integral Equation Solver and Mid Riemann Sum.

1.0 INTRODUCTION

Deriving analytical solutions to definite integrals might be difficult or impossible at times. The numerical integration technique facilitates in dealing with such situations successfully [1]. The elliptical integral has been shown to be challenging to solve analytically. An elliptical integral in mathematics is any function f(x) expressed as:

$$f(x) = \int_{c}^{x} R\left(t, \sqrt{P(t)}\right) dt \quad) \qquad (1)$$

c is a constant, and P is a polynomial of degree 3 or 4 with no repeated roots. R is a logical function of the two parameters it receives, and x and t are the two arguments of the function. In general, it's difficult to express integrals like this in terms of simple functions. In this circumstance, numerical integration techniques are quite valuable, and they typically produce precise results. Integrals that are notoriously difficult to solve, such as elliptic integrals, have a wide range of applications in astrophysics and engineering [1].

Problems are primarily described by mathematical models in all engineering professions, which use equations to represent the relationships between different parts of the problems, usually through integration [2]. Heat, wave, gas, electric current, electromagnetic fields, conductivity, vibrations in materials such as bridges and structures, and many more basic phenomena occur all the time in engineering. By integrating both sides of the function from x=a to x=b [3], basic issues attempt to maximize or minimize a quantity such as cost or profit, or the surface area of an object, or the distance a projectile can travel. Much more sophisticated integration approaches can be enabled by a computer with a flexible programming language [4]. Definite integration is a fundamental mathematical idea with several applications in domains like physics and engineering [5]. Divergent integrals and integral operators with divergent kernels are extensively used in mathematics, applied science, and engineering [6].

*Corresponding author (**Tel:** +234 (0) 8066276806)
Email addresses: msmajemusa4@gmail.comb (M. Adamu), lm324fairchild@yahoo.com (A.I. Audu), ngene@unimaid.edu.ng (C. U. Ngene)

The bulk of numerical integration algorithms are software-based and take a long time to perform and produce results. Field Programmable Gate Arrays (FPGAs) can be utilized to build numerical integration algorithms that are faster, more efficient, and more reliable [7]. As a result, utilizing numerical integration techniques with FPGAs will result in a definite integral solver that is considerably faster, more efficient, and more trustworthy. In this work, the speed and precision of FPGA technology will be utilized appropriately in the design and development of the solution. The solution will take advantage of FPGAs' exceptional capabilities in order to add an efficient and technologically advanced device capable of performing definite integration with high speed and accuracy to the repository of knowledge.

The technical need to depart from the traditional approach of numerical integration algorithms being executed in software, which is time consuming, provided the technical need to embark on the design and development of a Field Programmable Gate Array (FPGA) Based Design of Riemann Summation Definite Integral Equation Solver, which will be of improved performance in terms of resource utilization in generating values of definite integrals, when compared to available few previous versions. This study uses a Xilinx 7-series device, Riemann Summation, and VHDL as the synthesis language to assess the performance of a developed FPGA-based Definite Integral Equation Solver.

2.0 LITERATURE REVIEW

[8], in the paper, "A Study of Definite Integrals Using Parseval's Identity," looked into two sorts of definite integrals. Parseval's identity was used to determine the infinite series expressions of the two forms of definite integrals. Two instances were used for practical calculations. The study's research method was to use hand computations to identify solutions, which were then validated using Maple. The study approach allows for the detection of math errors as well as the alteration of pre-existing thought processes. When compared to Maple's result, Parseval's identity method yielded good results.

The Five Columns Rule in Solving Definite Integration by Parts by Transformation of Integral Limits, published by [9], proposes an alternate approach for solving definite integration by parts. It uses a tabular integration by parts approach with integral limit modifications to solve integrals. The results revealed that the part-by-part integration formula takes on a new shape following transformation. The final solutions acquired by this method and those obtained by the traditional method in calculus textbooks were found to be identical.

In the paper by [10], titled: "Definite integrals using the method of brackets," they proposed a novel heuristic method for assessing definite integrals. The researchers feel the method is a potent method of integration, despite the fact that it is heuristic and lacks a rigorous explanation. It's relatively easy to work with definite integrals and generate outcomes from them. Evaluating a definite integral is reduced to solving a linear system of equations with method of brackets and origin in methods developed for the assessment of Feynman diagrams.

In their publication, "Solution of Definite Integrals Using Functional Link Artificial Neural Networks," [1], presented a new approach for solving definite integrals using feedforward artificial neural networks. The researchers developed a neural network that can be used as a revolutionary numerical method in place of established methods. By minimizing a well-constructed error function, the learning algorithm was employed to solve the definite integrals. In comparison to existing numerical approaches, the results demonstrated that the algorithm is effective and exact, and that it is well-suited for applications that demand the integration of higher order polynomials.

In [7], four numerical integration algorithms were built on hardware using FPGA. The Left Riemann Sum (LRS), Right Riemann Sum (RRS), Middle Riemann Sum (MRS), and Trapezoidal Sum (TS) algorithms were used in the computation. The system performance was measured using the target chip Altera cyclone IV FPGA in terms of resource utilization, clock latency, execution time, power consumption, and computational error in comparison to other algorithms. According to the data, the FPGA implementation is much faster than the software version.

Integration algorithms and universal FIR filters were linked one-to-one by [11]. By translating the integrating algorithm onto FIR structures, the trapezoidal rule for numerical integration was applied in this study. This relationship was used by the Integration method to generate a structure. Critical route delays, on the other hand, are typical in such systems, restricting sampling/throughput rates. Concurrency was used to solve the problem at numerous stages along the method. The effects of pipelined and parallel structures on speed and power metrics were studied separately. The data paths inside the structure were altered as a result of these architectural changes, allowing it to run at higher throughput rates and/or with less power usage.

The idea in [12] provides a systematic method for hardware implementation of the basic operators of fractional calculus (fractional integrator and derivative) using a field programmable gate array (FPGA) in the

LabVIEW environment, based on the Grunwald–Letnikov notion. The fractional order integrator and derivative of sinusoid and square waveform signals were simulated, with results from hardware implementation. Fractional order calculus has been discovered to be useful in system modeling. Fractional calculus modeling is far more accurate than integer order modeling.

3.0 METHODOLOGY

The design and development of a Definite Integral Equation Solver using a Xilinx 7-series device and Very High Speed Integrated Circuit Hardware Description Language (VHDL) as the synthesis language begins with:

i. Design and development of an arithmetic and logic unit (ALU) that will perform numerical operations on targets operands. These operations include integer addition, subtraction, multiplication, and division.

ii. Design and development of a definite integral equation solver based on Riemann summation using the ALU in (i) above.

iii. And finally, simulating the designs in (i), (ii) above and evaluating the performance of the definite integral solver developed in (ii) above using an example problem.

The Design will begin by designing an arithmetic and logic unit (ALU) that will perform numerical operations (operations include integer addition, subtraction, multiplication, and division) on targets operands, followed by a definite integral solver based on Riemann summation using the ALU developed and then finally simulate the above two designs and evaluate the performance of the definite integral solver using an example problem. The workflow of the methodology is indicated below:

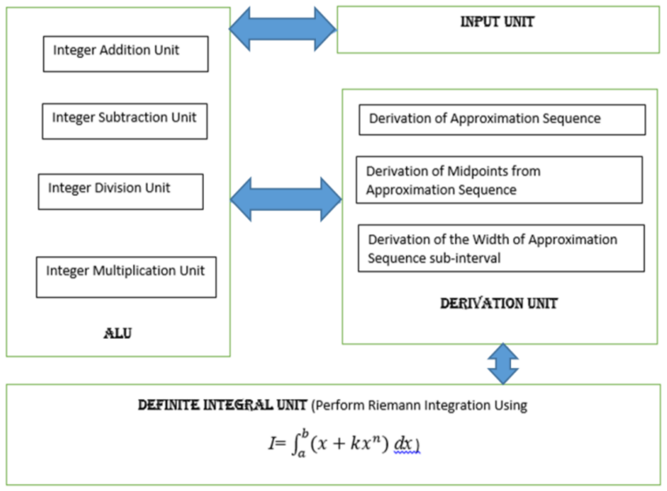

INPUT UNIT
Integer Addition Unit
Integer Subtraction Unit
Integer Division Unit
Integer Multiplication Unit
ALU
Derivation of Approximation Sequence
Derivation of Midpoints from Approximation Sequence
Derivation of the Width of Approximation Sequence sub-interval
DERIVATION UNIT
DEFINITE INTEGRAL UNIT (Perform Riemann Integration Using
$I= \int_a^b (x + kx^n)\, dx$)

Figure 1: Workflow for the Methodology

3.1 Performing Definite Integration

To perform definite integration as shown in Figure 1, the first step is the identification of the equation parameters which will be passed to the solver. As an example, considering the following definite integral equation I with a constant 16 and a variable x having power of 2 and assuming an upper limit of 4 and lower limit of -1 as shown below:

$$I = \int_{-1}^{4} (16 - x^2)\, dx \tag{2}$$

The equation parameters can be extracted as:

i. Upper and lower limits of integration are 4 and -1, respectively

ii. Equation constant is 16

iii. Variable in the equation x

iv. Power of variable in the equation 2

When these parameters are passed to the definite integral solver, it will determine the difference between the upper and lower limits of the integral in equation 2, as shown below:

$$\alpha = 4 - (-1) = 4 + 1 = 5.$$

The solver will make this computation by calling on the integer subtraction service offered by the ALU as shown in Figure 1.

The value of n is the number of sub-intervals between [a, b], and it can be mathematically expressed as:

$$[x_0, x_1][x_1, x_2], \dots, [x_{n-1}, x_n] \tag{3}$$

For any sub-interval $[x_{i-1}, x_i]$ under consideration, its width can be determined as follows:

$$\Delta x_i = x_i - x_{i-1} \tag{4}$$

The operation in (4) is also performed by the integer subtraction unit shown in Figure 1. The next step is the derivation of the midpoints M from the approximation sequence in (3). This is achieved by using the following relationship:

$$M_i = \frac{\Delta x_i}{2} = \frac{x_i - x_{i-1}}{2} \tag{5}$$

To perform the operation in (5), the definite integral solver requests two services from the Arithmetic unit i.e. the integer subtraction and the integer division. Each of the value M_i obtained is substituted into the algebraic expression in the integral to obtain the corresponding values of the function $f(\xi_i)$. For the relationship in (2), it holds that $f(\xi_i) = 16 - M_i^2$.

The final step is the determination of the Riemann sum, and which is done using the relationship:

$$I = \int_a^b f(x)dx = \sum_{i=n}^{N} f(\xi_i)\Delta x_i \qquad (6)$$

In most cases, $\Delta x = \Delta x_1 = \Delta x_2 = \cdots = \Delta x_n$. Hence, equation (6) can be written as:

$$I = \int_a^b f(x)dx = \sum_{i=n}^{N} f(\xi_i)\,\Delta x = \Delta x \sum_{i=n}^{N} f(\xi_i) \qquad (7)$$

The relationship in (7) will request two services from the ALU i.e. integer multiplication, and integer addition to generate the value of the definite integral.

3.2 Target device and reasons for the choice

The target device for the definite integral solver will be the Kintex-7 FPGA. The reason for the choice of this device is as follows:

i. It is a robust device
ii. It has low power consumption
iii. It is highly reconfigurable

Since this device is a 7-series device, Xilinx VIVADO will be the development environment and the language of choice for the implementation is VHDL. The hardware description of the design using VHDL will be verified for accuracy and errors using VIVADO.

3.3 Problem Addressed

In the quest to solve different challenges and problems in human society, scientists and engineers make a lot of decisions based on the results obtained from precise calculations made in designs and models. With advances in calculus and linear algebra, large volume of data processing having a lot of dependencies and the need to improve performance of complex models through parallel implementations, there is need for deviation from the software based definite integral equation solvers that take long time to generate results to FPGA based solvers (Razak et al, 2017). This research is aimed at developing a solver that can leverage on the advanced capabilities of FPGAs, such as low resource and power utilization and precision, and the circuit optimization capabilities of VHDL, in order to add an efficient and technologically advanced device capable of performing definite integration with accuracy and low latency when compared to the available few, to the repository of knowledge.

4.0 RESULT AND DISCUSSION

The performance of the definite integral solver is presented in a progressive approach i.e. the analysis is based on the outcome of the results as they are produced by the definite integral solver and its units. The first part of the analysis tests the different units of the Arithmetic and Logical Unit (ALU). The functions in the ALU unit are displayed in the progressive results generated in the process of solving definite integral equation by the integral solver.

4.1 The Arithmetic and Logical Unit

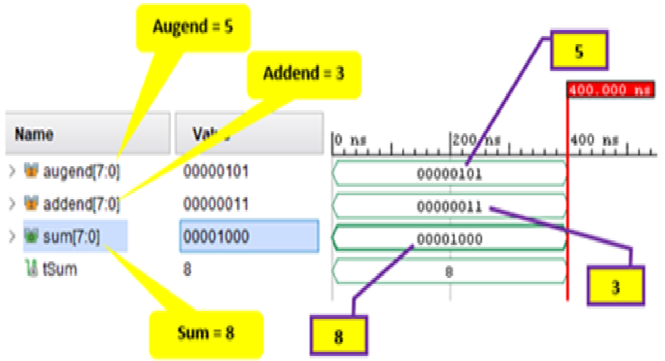
The first part of the analysis begins by testing the different units of the Arithmetic and Logical Unit (ALU): the integer addition unit, integer subtraction unit, integer multiplication unit and the integer division unit. The four units together form the Arithmetic and Logical Unit (ALU).

4.1.1 The Integer Addition Unit

The integer addition unit is tested by providing two input values, the augend=5 and the addend =3. The unit successfully produced an accurate result of the addition of the two integers, which is equal to 8. The image in Figure 2 and Table 1 show the binary equivalent of the inputs as 0000 0101 representing 5, 0000 0011 representing 3, and 0000 1000 representing the result of the addition, which is 8. The integer addition unit has been tested, and found to be functional and performing according to expectation.

Table 1: Binary and Decimal Input and Output Values of the Integer Addition Unit

System interface		Binary Representation	Decimal Representation
Input	Augend	0000 0101	5
	Addend	0000 0011	3
Output	Sum	0000 1000	8



Name	Value
augend[7:0]	00000101
addend[7:0]	00000011
sum[7:0]	00001000
tSum	8

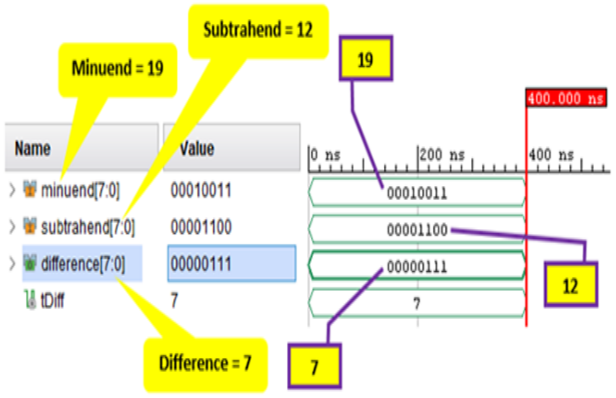
Figure 2: Integer addition unit

4.1.2 The Integer Subtraction Unit

The integer subtraction unit is tested by providing two input values, the minuend=19 and the subtrahend =12. The unit successfully produced an accurate result of the subtraction of the two integers, which is equal to 7. The image in Figure 3 and Table 2 show the binary equivalent of the inputs as 0001 0011 representing 19, 0000 1100 representing 12, and 0000 0111 representing the result of the subtraction, which is 7. The integer subtraction unit has been tested, and found to be functional and performing according to expectation.

Table 2: Binary and Decimal Input and Output Values of the Integer Subtraction Unit

System interface		Binary Representation	Decimal Representation
Input	Minuend	0001 0011	19
	Subtrahend	0000 1100	12
Output	Sum	0000 0100	7



Name	Value
minuend[7:0]	00010011
subtrahend[7:0]	00001100
difference[7:0]	00000111
tDiff	7

Figure 3: Integer subtraction unit

4.1.3 The Integer Multiplicaton Unit

The integer multiplication unit is tested by providing two input values, the multiplicand=21 and the multiplier =8. The unit successfully produced an accurate result of the multiplication of the two integers, which is equal to 168. The image in Figure 4 and Table 3 show the binary equivalent of the inputs as 0000 0000 0001 0101 representing 21, 0000 0000 0000 1000 representing 8, and the output 0000 0000 1010 1000 representing the result of the multplication, which is 168. The integer multiplication unit has been tested, and found to be functional and performing according to expectation.

Table 3: Binary and Decimal Input and Output Values of the Integer Multiplicaton Unit

System interface	Binary Representation		Decimal Representation
Input	Multiplicand	0001 0101	21
	Multiplier	0000 1000	8
Output	Sum	0000 0000 1010 1000	168

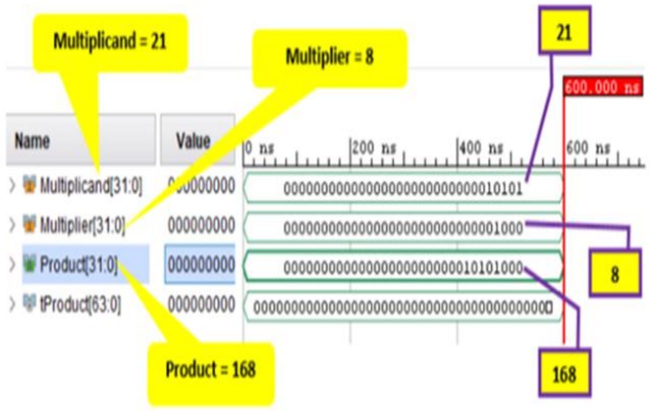


Figure 4: Integer multiplication unit.

4.1.4 *The Integer Division Unit*

The integer division unit is tested by providing two input values, the dividend =32 and the divisor =12. The unit successfully produced an accurate result of the division of the two integers, which is equal to 4. The image in Figure 5 and the Table 4 show the binary equivalents of the inputs as 0000 0000 0010 0000 representing 32, 0000 0000 0000 1000 representing 8, and the output of 0000 0000 0000 0100 representing the result of the division, which is 4, while 0000 0000 0000 0000 represent zero (0) remainder. The integer division unit has been tested, and found to be functional and performing according to expectation.

Table 4: Binary and Decimal Input and Output Values of the Integer Division Unit

System interface		Binary Representation	Decimal Representation
Input	Dividend	0000 0000 0010 0000	32
	Divisor	0000 0000 0000 1000	8
Output	Quotient	0000 0000 0000 0100	4
	Remainder	0000 0000 0000 0000	0

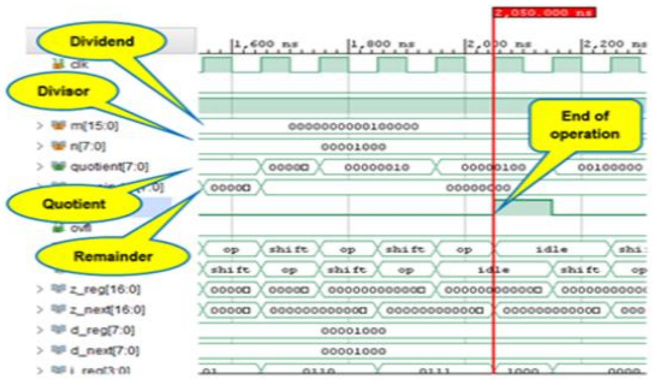


Figure 5: Integer Division Unit

4.2 *Evaluating the developed Solver Using an Equation of x with a coefficient and a constant factor*

Let consider a definite integral equation:

$$I = \int_{-3}^{5} (1 + 2x^2)dx \qquad (8)$$

Using the relationship$\Delta = \varphi_{Upper} - \varphi_{Lower}$, the interval Δ of the integral is computed as follows:

$$\Delta = \varphi_{Upper} - \varphi_{Lower} = 5 - -3 = 5 + 3 = 8 \qquad (9)$$

The integral solver obtained the same result. This is shown in figure 6.

Figure 6: Integral interval

Using the relationship in (10), the midpoint factor $\alpha$was determined, and the integral solver computed the same value as shown in Figure 7.

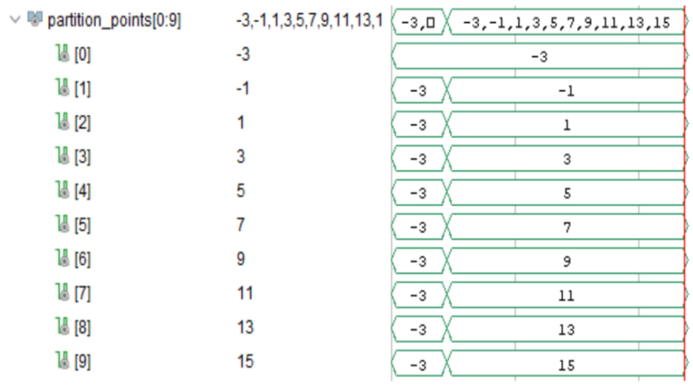
$$\propto = \frac{\Delta}{n} = \frac{8}{4} = 2 \qquad (10)$$

Figure 7: Midpoint factor

From the relationship in (11),

$$\begin{gathered} P1 = \psi_{Lower} \\ P2 = P1 + \alpha \qquad (11) \\ P3 = P2 + \alpha \\ Pn = Pn - 1 + \alpha) \end{gathered}$$

The partition points were determined by the integral solver as shown in figure 8.



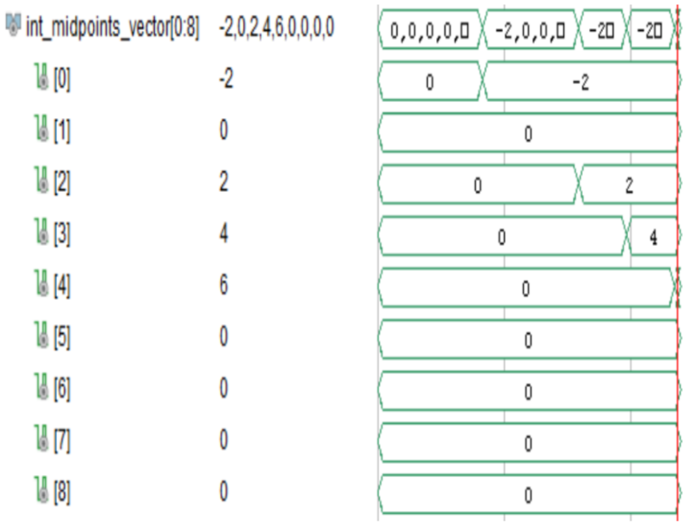
```
partition_points[0:9]   -3,-1,1,3,5,7,9,11,13,1   -3,□   -3,-1,1,3,5,7,9,11,13,15
  [0]   -3                                        -3
  [1]   -1                                 -3     -1
  [2]   1                                  -3     1
  [3]   3                                  -3     3
  [4]   5                                  -3     5
  [5]   7                                  -3     7
  [6]   9                                  -3     9
  [7]   11                                 -3     11
  [8]   13                                 -3     13
  [9]   15                                 -3     15
```

Figure 8: Partition points

Using the relationship in (12) below,

$$\begin{pmatrix} M_1 = \dfrac{P_1 + P_1}{2} \\ M_1 = \dfrac{P_2 + P_3}{2} \\ M_1 = \dfrac{P_{n-1} + P_n}{2} \end{pmatrix}$$

The midpoints of the integral subintervals were computed as shown in figure 9.



```
int_midpoints_vector[0:8]   -2,0,2,4,6,0,0,0,0   0,0,0,0,□   -2,0,0,□   -2□   -2□
  [0]   -2          0      -2
  [1]   0           0
  [2]   2           0      2
  [3]   4           0      4
  [4]   6           0
  [5]   0           0
  [6]   0           0
  [7]   0           0
  [8]   0           0
```

Figure 9: Midpoints

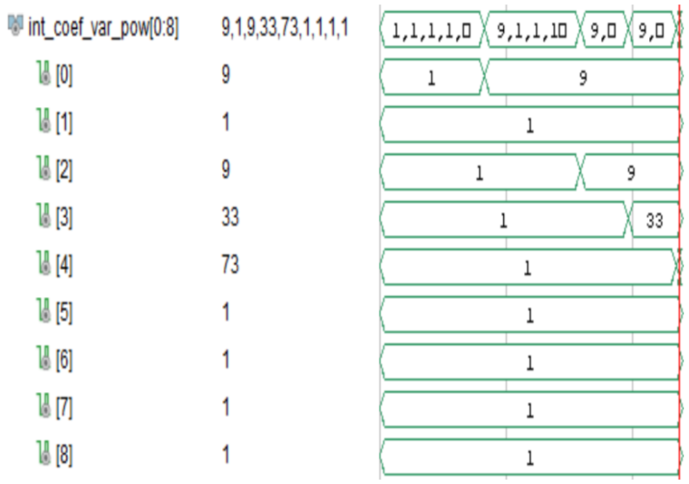
The values of zero for other values of the midpoint subintervals in figure 9, occurred because the integral solver determined that it is not necessary to compute those values for the equation being solved.
From the relationship in (13) below:

$$I_n = \sum_{k=1}^{n} f(\xi_k)\Delta x \qquad (13)$$

The values of the function $f(\xi k)$ are determined as follows:

$$\begin{gathered} f(\xi_1) = f(-2) = 1 + 2.(-2)^2 = 9 \\ f(\xi_2) = f(0) = 1 + 2.(0)^2 = 1 \\ f(\xi_3) = f(2) = 1 + 2.(2)^2 = 9 \qquad (14) \\ f(\xi_4) = f(4) = 1 + 2.(4)^2 = 33 \end{gathered}$$

The integral solver computed the values of $f(\xi k)$ in (14) using the same approach. These values are shown in figure 10.



```
int_coef_var_pow[0:8]   9,1,9,33,73,1,1,1,1   1,1,1,1,□   9,1,1,1□   9,□   9,□
  [0]   9      1      9
  [1]   1      1
  [2]   9      1      9
  [3]   33     1      33
  [4]   73     1
  [5]   1      1
  [6]   1      1
  [7]   1      1
  [8]   1      1
```

Figure 10: Variable Power

For a midpoint number value of $n = 4$, the relationship in (13) is determined as:

$$I_4 = \sum_{k=1}^{4} f(\xi_k)\Delta x = 2X(9 + 1 + 9 + 33) = 104 \qquad (15)$$

The computation of the result in (XV) by the integral solver is analyzed as follows based on Definition 1(β_1) and Definition 2(β_2) as:

$$\beta_1 = \sum_{k=1}^{4} n_k = 52 \qquad (16)$$

The integral solver arrived at the same result after the computation as shown in figure 11.

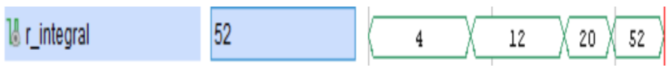


Figure 11: Summation

The final result of the integration was determined by the integral solver using the relationship in (13). This is shown in Figure 12.

$$\beta_2 = 2\sum_{k=1}^{1} n_k = 104 \qquad (17)$$

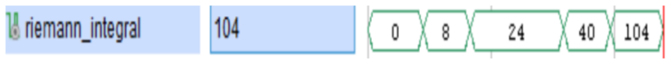


Figure 12: Integral value

4.3 Power Estimation and Resource Utilization Evaluation and performance comparison

The design like any other design in hardware uses up resources on the target hardware for which it is implemented. It is against this backdrop that power consumption analysis was performed in order to estimate how much power is used up by the processor during execution. Using the Xilinx Power Estimator, the dynamic power and static power for the design were estimated as shown in Figure 13 where 95% (2.684W) of the power was dynamic power while 5% (0.143W) of the power was static power.

The dynamic power is the power used up as a result of switching activities by the processor while executing the definite integral function. The static power is the power used up as a result of the nature of the silicon used in the fabrication of the hardware device itself using CMOS (complementary metal oxide semiconductor) technology.

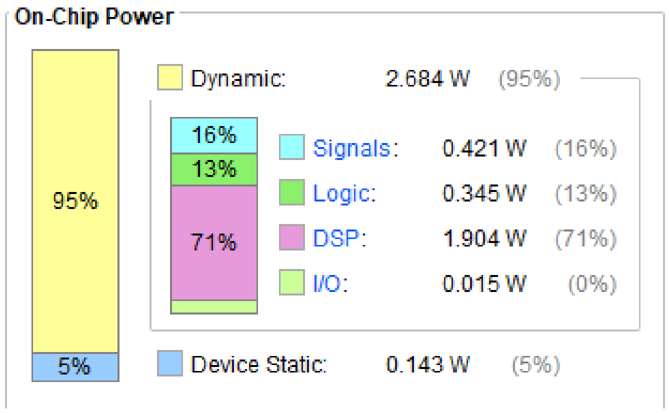
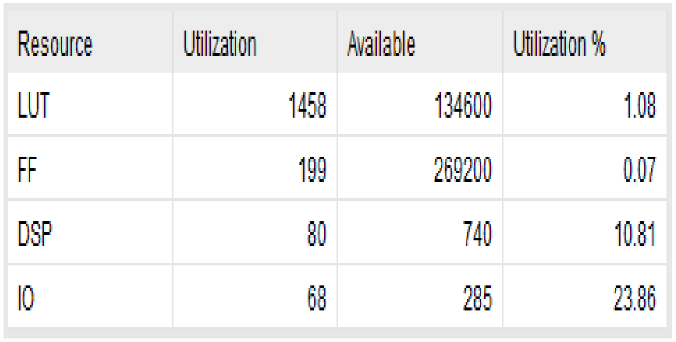


Figure 13: Dynamic and static power estimation for design

The resource utilization by the design shows that 1.08% (1458) LUT (look up tables) programming elements (PEs) were utilized out of a total of 134600 available LUTs on the FPGA. Similarly, 0.07% (199) FF (flip flops) PEs were utilized out of a total of 269200 available FFs; 10.81% (80) DSP (digital signal processing) PEs were utilized out of a total of 740 available DSPs; 23.86% (68) IO (input-output) PEs were utilized out of a total of 285 available IOs. Figure 14 shows this analysis as obtained from the Utilization Report Generator of the Xilinx VIVADO.



Resource	Utilization	Available	Utilization %
LUT	1458	134600	1.08
FF	199	269200	0.07
DSP	80	740	10.81
IO	68	285	23.86

Figure 14: Integral processor resource utilization.

A performance comparison was made between the resource utilization of the developed solver and that of similar works by (Razak *et al.* 2017) and (Rana *et al.* 2016). Table 5 shows this comparison.

Table 5: Performance comparison with related works

Parameters	*Our work*	*(Razak et al. 2017)*	*(Rana et al. 2016)*
LUT (%)	1.08	19.8	13
FF (%)	0.07	13	8
DSP (%)	10.81	20	19
IO (%)	23.86	10	23

It can be seen from Table 5 that the proposed design in this research showed better performance for all the parameters considered except for the IO blocks.

5.0 REFERENCES

[1] Changdar, S. Bhattacharjee, S. “Solution of Definite Integrals using Functional Link Artificial Neural Networks”. arXiv preprint arXiv:1904.09656. 2019 April, 2021.

[2] Zhang, H., Chen, Y., and Nie, X. “Solving the Linear Integral Equations Based on Radial Basis Function Interpolation.” *Journal of Applied Mathematics,* 2014.

[3] Edelstein-keshet, L.“Integral Calculus with Applications to the Life Sciences.” https://docplayer.net/90163421-Integral-calculus-with-applications-to-the-life sciences.html 2015, Accessed on 30/12/2021.

[4] Herceg, D. and Dorde H. “The Definite Integral and Computer." The Teaching of Mathematics.” 22:33–44. http://elib.mi.sanu.ac.rs/files/journals/tm/22/tm1215.pdf 2009, Accessed on 30/12/2021.

[5] Zhao, W., Zhaoning Z., and Zhijian Y. “Midpoint Derivative-Based Trapezoid Rule for the Riemann-Stieltjes Integral.” *Italian Journal of Pure and Applied Mathematics* 33, 2014, pp. 369–376.

[6] Zozulya, V. (2007). “Regularization of the Divergent Integrals. II. Application in Fracture Mechanics.” *Electronic Journal of Boundary Elements* 4(2), 2007, pp. 49–57.

[7] Razak, F. N. A., Talip, M. S. A, Yakub, M. F. M., Khairudin, A. S. M., Izam, T. F. T, and Zaman F. H. K. “High Speed Numerical Integration Algorithm Using FPGA.” *Journal of Fundamental and Applied Sciences*, 9(4S), 2017, pp.131–144.

[8] Yu, C. A Study of Definite Integrals Using Parseval ’ s Identity. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology,* 1(2), 2016, pp.28–32,.

[9] Jandja, M. and Lutfi, M. “The Five Columns Rule in Solving Definite Integration by Parts Through Transformation of Integral Limits.” *Journal of Physics: Conference Series,* 1028(1), 2018.

[10] Gonzalez, I. and Moll. V. H. “Definite Integrals by the Method of Brackets. Part 1.” *Advances in Applied Mathematics* 45(1), 2010, 50–73.

[11] Khurshid, Burhan, and Roohie Naz Mir. "A hardware intensive approach for efficient implementation of numerical integration for FPGA platforms." In 27th International Conference on VLSI Design and 13th International Conference on Embedded Systems, 2014, pp. 312-317.

[12] Rana, K. P. S., Kumar, V. Mittra, N. and Pramanik, N. “Implementation of Fractional Order Integrator/Differentiator on FIeld Programmable Gate Array.” *Alexandria Engineering Journal* 55, , 2016, pp. 1765–1773.