INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 4.295
(Volume 3, Issue 6)
Available online at www.ijariit.com

A High Performance Jacobi Iterative Solver

Chaitanya Nutalapati
PG Scholar, 2nd M.Tech (VLSI & ES)
QIS Institute of Technology, Ongole, Andhra Pradesh
chaitanyan15@gmail.com

Vijay Kumar Yarasi
M.Tech, Assistant Professor
QIS Institute of Technology, Ongole, Andhra Pradesh
vijayyarasi@gmail.com

Abstract: Jacobi solver is one of the most efficient to solve large linear system of equations. As this method involves more number of iterations, it takes a longer time for giving the solution in VLSI. This paper presents an implementation of Jacobi solver in FPGA, in which, the various blocks of Jacobi solver are implemented with Kogge-Stone adder (KSA) where ever addition is required. This resulted in enhanced performance at the cost of little increase in area. The improvement in performance is a decrease in the delay of 67.7% in the minimum period of delay, which is a substantial improvement. As compared to the Ripple carry addition (RCA), addition implementation in the KSA is found to give an area savings of 21.21% in terms of slices, an area increase of 56.60% in terms of flip-flops and an increase of 49.04% in terms of 4 input LUTs. Hence this design can be implemented in places where high performance is of primary concern. The implementation and simulation have been performed in VHDL in Xilinx 14.7 targeted to FPGA.

Keywords: Jacobi Solver, Kogge-Stone adder (KSA), Field Programmable Gate Array (FPGA).

1. INTRODUCTION

There are two methods for solving large linear system of equations, direct method and iterative method. Generally the direct method is more preferable [2]. The iterative method can be further classified into two. The first one is stationary iterative method in which, the coefficients will be iteration independent [3]. The second one is non- stationary iterative method, in which; the coefficients will be iteration dependent. The information of the current iteration step is used for the succeeding iteration step [3,4]. Sparse linear systems can be solved by three iterative methods

- The Jacobi method
- The Gauss-Seidel method
- The Successive over relaxation method (SOR)

Among the above three methods, the Jacobi method is discussed here. Consider the following linear equation having the form

$$Ax = b \qquad (1)$$

In this equation, *A* is the sparse co-efficient matrix and b and x are the real n vectors. b is a known real vector and x is an unknown real vector. The memory and time cost involved in solving these equations by iterative methods is almost linear with the size of the matrix A [5]. The performance of these methods is improved with the usage of preconditioner [6]. The preconditioner modifies the original system as follows.

$$M^{-1}Ax = M^{-1}b \qquad (2)$$

In the above equation M is the pre-conditioner.

2. METHOD ANALYSIS

Jacobi Method: It is an important method for solving diagonally dominant and stable large number of linear equations. This is a stationary iterative method. In this method every variable is locally solved. Every of iteration corresponds to other variables [4]. It is easily implementable but convergence is very slow [7]. This method is also called as method of simultaneous iteration.

© 2017, www.IJARIIT.com All Rights Reserved

In this method the value of x^{k+1} is dependent on the value of x^k in which K denotes the number of iterations. The i^{th} iteration of equation (1) can be expressed as

$$\sum_{k=0}^{n} a_{ij}\, x_j = b_i \qquad (3)$$

Being other entities of *x* remain fixed, the calculation of x_i gives the following equations.

$$x_i^{k+1} = \frac{1}{a_{ij}}(b_i - \Sigma_{j\neq 1}\, a_{ij} x_j^k \qquad (4)$$

A is rewritten as *A=D+A–D*. In this, *D* = diag (a_{11}, a_{22} ..... a_{nn}).

Substituting this in (1) gives

$$Dx + (A - D)x = b \qquad (5)$$

$$x = D^{-1}(D - A)x + D^{-1}b \qquad (6)$$

The matrix representation of the above is

$$x_{k+1} = D^{-1}(\mathrm{D} - \mathrm{A})x_k + D^{-1}b \qquad (7)$$

3. EXISTING METHOD

Finding the value of the variable in Jacobi method involves the implementation of the equation (4). The VLSI implementation of this equation gives the architecture in fig.1.

This architecture is implemented with RCA (Ripple carry adder) where ever addition is implemented.

The implementation consists of the following blocks.

a) Multipliers - 6
b) Adders - 4
c) Divisor - 1
d) Subtractors - 2

All the above blocks are designed using 64 - bit IEEE 754 floating point standard [1].

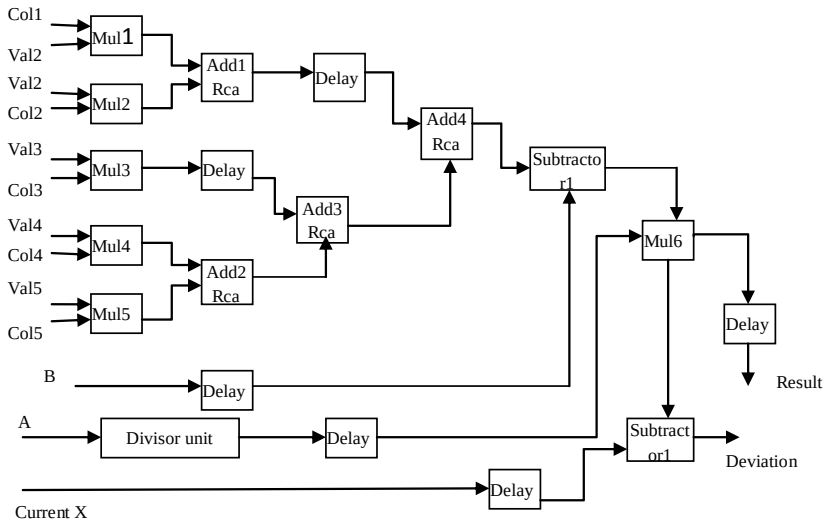

Figure.1: Jacobi Solver using RCA

The equation (4) is implemented using the architecture given in fig. 1. This architecture is divided into two modules.

a) Jacobian Calculation module
b) Error module

© 2017, www.IJARIIT.com All Rights Reserved

The Jacobian calculation module calculates the Jacobi value by solving the equation (4). Delay units constructed with flip-flops are used to introduce the required amount of delays in the path of values so that they arrive the calculation submodules at the same time for calculation [8-10].

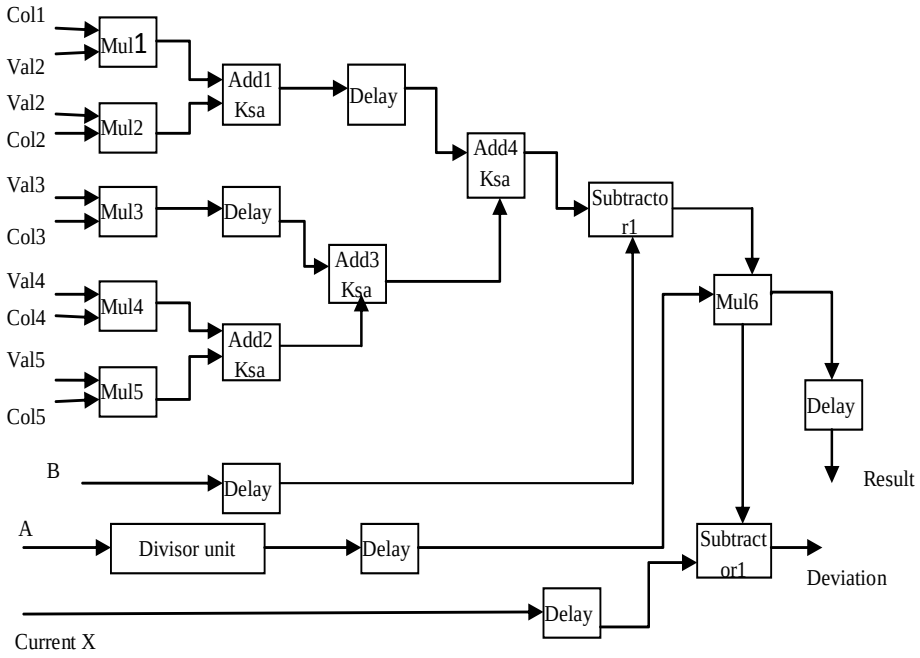

Figure.2: Jacobi solver using KSA

In order to calculate the error, two numbers in the 64 bit floating point format are passed each clock cycle. The first number passed tracks the largest of the values since the last reset. The deviation is also calculated and it is passed through. The latest value of it is stalked in the module [10]. A start signal starts the floating point divider which divides the latest division value with largest value of B. On completion of this calculation, the computed error value is arrived. This value is compared with desired error value. When the computed error is smaller compared to the desired error, the stop flag in the module is asserted high indicating no need of continual of iteration. Else, the stop flag in the module is set low to continue the iteration.

4. PROPOSED METHOD

In the proposed method, the KSA (Kogge-Stone adder) is implemented in place of RCA in the existed method. The KSA is a parallel prefix form of carry look ahead adder (CLA) [13], which is efficient and faster. The implementation of Jacobi solver with KSA is as shown in fig.2.

The KSA has the shortest critical path among all the tree adders with little drawback of large area [13]. Because of larger number of iterations that are required in the computation of Jacobians of large system of linear equations, the high performance is of prime importance than the area. Hence the KSA implementation of Jacobian solver is proved superior in performance as compared to RCA implementation. With KSA implementation, the performance of Jacobi solver is improved by 67.7%. This is due to the small logic depth of KSA, which is log_2n where n is the bit length of each number to be added in contrast with the logic depth n in case of RCA. Also it is observed that the proposed method is found to have an area savings of 21.21% in terms of slices, an area increase of 56.60% in terms of flip-flops and an increase of 49.04% in terms of 4 input LUTs.

5. RESULTS

The Jacobi solver is implemented using both the RCA and KSA. The RTL diagram obtained after synthesis is shown in fig.3.The Implementation results are compared in tables I and II.

© 2017, www.IJARIIT.com All Rights Reserved

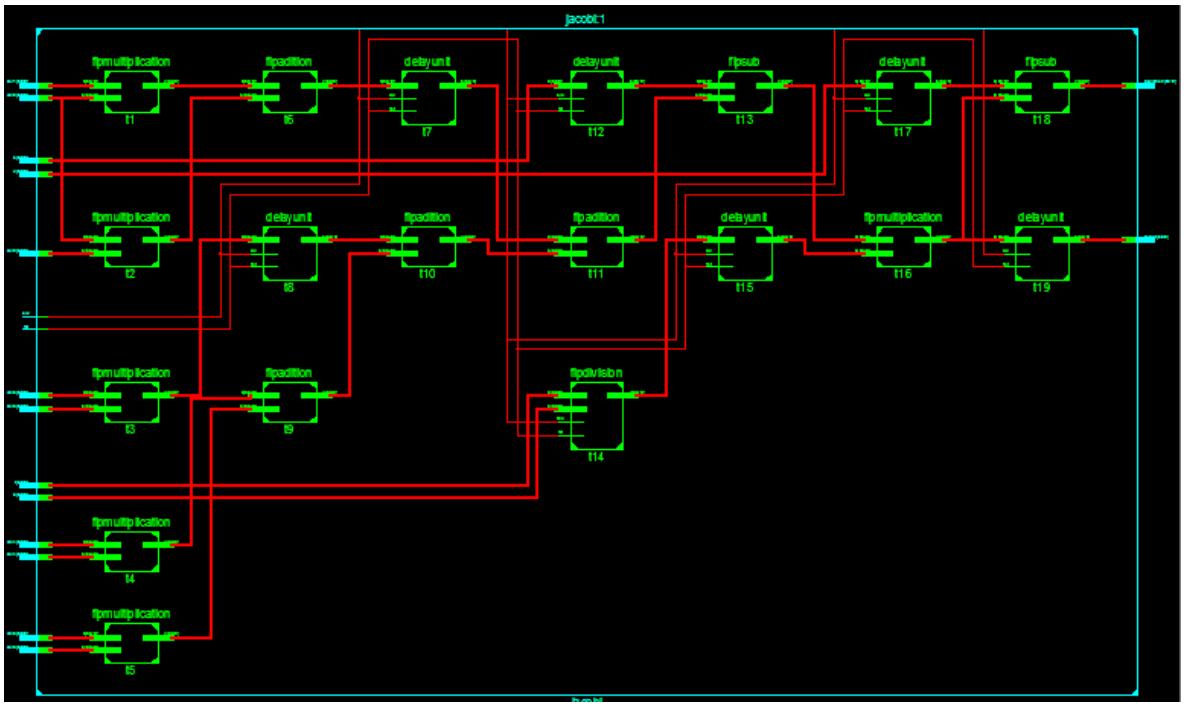

Figure.3: RTL Synthesis Diagram of Jacobi Solver

TABLE I: Timing Summary

Timing Parameter	RCA Implementation	KSA Implementation	Delay Improvement
Minimum Period	176.325ns (Maximum Frequency: 5.67MHz)	56.948ns (Maximum Frequency: 17.560MHz)	67.7%
Minimum input arrival time before clock	284.691ns	133.535ns	53%
Maximum output required time after clock	244.162ns	133.720ns	45%

TABLE II: Area Summary

Parameter	RCA Implementation	KSA Implementation	Area savings
Number of slices used	27183	21417	21.21%
Number of slice Flip-flops	454	711	-56.60%
Number of 4 input LUTs	25916	38626	-49.04%
Number of bonded IOBs	962	962	0
Number of GCLKs	1	1	0

© 2017, www.IJARIIT.com All Rights Reserved

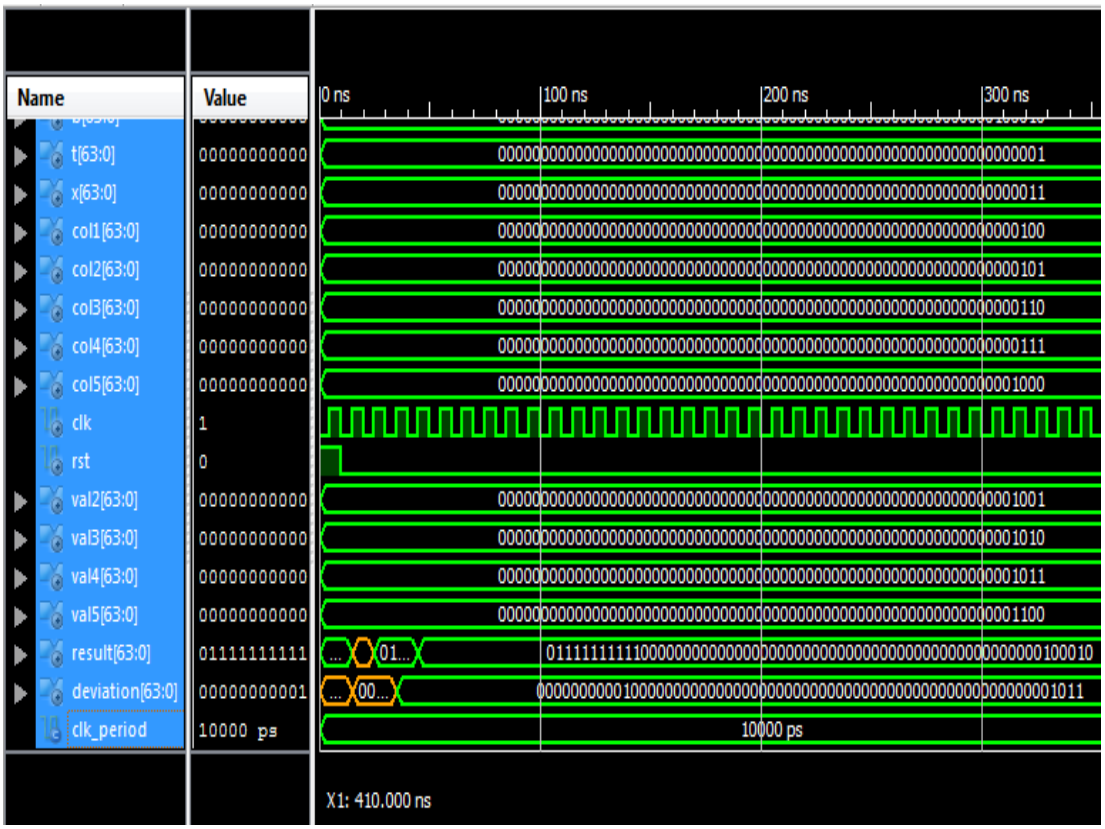

Fig.4: Simulation Test Bench Wave Forms of Jacobi Solver Using Ksa

6. CONCLUSION AND FUTURE SCOPE

The jacobi solver is implemented in VHDL in Xilinx targeted to Spartan-3E FPGA. As high performance is required the KSA has been implemented wherever addition is needed. The improvement in performance is observed to be 67.7% at the cost of little extra area. Hence wherever high performance is of primary requirement, there the KSA implementation of Jacobi solver can be used. More and more high performance computing methods need to be developed in order to solve the jacobian in still lesser time.

7. ACKNOWLEDGEMENTS

The authors hereby acknowledge their gratitude to the management of QIS Institute of Technology, Ongole to have provided laboratory facility for implementing this project.

REFERENCES

1. M. Tamuli, S. Debnath, A. Ray and S. Majumdar, "Implementation of Jacobi iterative solver in verilog HDL," *2016 2nd International Conference on Control, Instrumentation, Energy & Communication (CIEC)*, Kolkata, 2016, pp. 103-105.
2. Jamil Noreen," A Comparison of Direct and Indirect Solvers for Linear Systems of Equations". Int. J. Emerg. Sci., 2(2), 310-321, June 2012 ISSN: 2222-4254.
3. J. Foertsch, J. Johnson and P. Nagvajara, "Jacobi load flow accelerator using FPGA," *Proceedings of the 37th Annual North American Power Symposium*, 2005, pp. 448-454.
4. Cryer, "The solution of a quadratic programming problem using systematic over relaxation", SIAM Journal on Control, vol. 9,no. 3,pp. 385-392, 1971.
5. G. R. Morris, L. Zhuo and V. K. Prasanna, "High-performance FPGA-based general reduction methods," *13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'05)*, 2005, pp. 323-324.
6. Safaa J. Kasbah, Ramzi A. Haraty, and Issam W. Damaj. "The successive over relaxation method in reconfigurable hardware", Proceedings of the IAENG International MultiConference of Engineers and Computer Scientists. Hong Kong, China. March 2007.
7. I. Bravo, P. Jimenez, M. Mazo, J. L. Lazaro and A. Gardel, "Implementation in Fpgas of Jacobi Method to Solve the Eigen value and Eigen vector Problem," *2006 International Conference on Field Programmable Logic and Applications*, Madrid, 2006, pp. 1-4.
8. Safaa J. Kasbah, Ramzi A. Haraty, and Issam W. Damaj. "Reconfigurable Hardware Implementation of the Successive Over-relaxation Method". Advances in Industrial Engineering and Operations Research. Lecture Notes in Electrical Engineering. Volume 5, pp. 453-466. 2008.
9. Safaa J. Kasbah and Issam W. Damaj, "The Jacobi Method in Reconfigurable Hardware", Proceedings of the World Congress on Engineering 2007 Vol II WCE 2007, July 2 - 4, 2007, London, U.K.

© 2017, www.IJARIIT.com All Rights Reserved

10. John P. Morrison, Padraig O'Dowd and Philip D. Healy, "An Investigation into the Applicability of Distributed FPGAs to High Performance Computing", Department of Computer Science, University College Cork, Ireland, In Proceedings of the International Symposium on Parallel and Distributed Computing, pages 171.179, Iasi, Romania, July 2002.
11. S. Debnath, M. Tamuli, A. Ray and G. Trivedi, "A review on accelerating scientific computations using the Conjugate Gradient method," *2015 International Conference on Electronic Design, Computer Networks & Automated Verification (EDCAV)*, Shillong, 2015, pp. 150-153.
12. M. Tamuli, S. Debnath, A. K. Ray, S. Majumder, "A Review on Jacobi Iterative Solver and its Hardware based performance analysis", proceedings of 1st International Conference on Power, Dielectric and Energy Management at NERIST, 10-11 January 2015 (ICPDEN 2015), organized by NERIST Dept of EE with IEEE Kolkata chapter.
13. P. M. Kogge and H. S. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations," in *IEEE Transactions on Computers*, vol. C-22, no. 8, pp. 786-793, Aug. 1973. doi: 10.1109/TC.1973.5009159

© 2017, www.IJARIIT.com All Rights Reserved