A Superpipelined CORDIC Unit

Michael Fitzharris[1], Jeremy R. Johnson[12], Prawat Nagvajara[1], Servesh Tiwari[2]
mrf26@drexel.edu, jjohnson@cs.drexel.edu, prawat@cbis.ece.drexel.edu, st86@drexel.edu
Department of Electrical and Computer Engineering[1]
Department of Computer Science[2]
Drexel University
Philadelphia, PA

Introduction

Some applications, such as power system analysis, require many trigonometric computations. In power system analysis an iterative solver using Newton-Raphson is commonly used to solve the load flow equations, which is an essential and time-consuming computation in many power system applications. The Jacobian matrix, which is used each iteration, is constructed using numerous sine and cosine computations. In such applications, substantial speedup can be obtained by utilizing specialized hardware to compute sine and cosine.

The CORDIC method provides an iterative scheme, consisting of simple addition and binary shift operations, to compute trigonometric values to any desired precision. The iterations in the CORDIC method can be pipelined to devise an efficient hardware unit that is capable of computing one sine and cosine every clock cycle. The number of stages in the pipeline depends on the number of iterations and hence the desired accuracy. This leads to a very long pipeline, which provides efficient computation provided there is a stream of sine and cosine computations. In such cases, an order of magnitude improvement in performance was obtained using CORDIC unit implemented in FPGA (field programmable gate array) compared to a software solution using a Pentium IV processor.

Load Flow Computation

Load flow computation involves solving a set of nonlinear equations using Newton-Raphson iteration technique [1]. For each bus ***i*** in a power system, the power in that bus is a function of the voltage magnitude and phase angle of all the buses in the system and the real and imaginary power is given by the system of equations

$$P_i(\mathbf{x}) = \sum_{k=1}^{n} |V_i||V_k|[G_{ik}\cos(\theta_i-\theta_k)+B_{ik}\sin(\theta_i-\theta_k)]$$

$$Q_i(\mathbf{x}) = \sum_{k=1}^{n} |V_i||V_k|[G_{ik}\sin(\theta_i-\theta_k)-B_{ik}\cos(\theta_i-\theta_k)]$$

Power flow solution via Newton method involves iterating the following equation

$$-J\,\Delta x = f(x),$$

until f(x) = 0 is satisfied, where the Jacobian, J, is the first order partial derivative matrix, Δx is a vector of the change in the voltage magnitude and phase angle for the current iteration, and f(x) is a vector representing the real and imaginary power mismatch. The Jacobian is a large sparse matrix involving terms with sines and cosines.

The bulk of the time for Newton-Raphson [2] is devoted to solving for Δx each iteration. Solving for Δx is typically done by factoring, J, a large sparse matrix, into lower and upper triangular factors and performing forward and backward substitution. In previous work we have shown how to reduce the time for the LU factorization using special-purpose hardware implemented with FPGA [3,4]. However, this solution requires the update to the Jacobian to be performed in software and communication to/from the FPGA can be a bottleneck. Constructing the Jacobian on the FPGA eliminates this bottleneck, but requires the hardware to support trigonometric computations.

The CORDIC Algorithm

CORDIC [5], "COordinate Rotation DIgital Computer," provides an iterative scheme for computing trigonometric functions, which obtains one bit of accuracy per iteration. The CORDIC algorithm works by successively rotating a two-dimensional vector through smaller and smaller angles (micro-rotations) that converge on the desired angle. The vector is initialized so that the final coordinates contain the sine and cosine of the specified angle. Moreover, the micro-rotations are chosen so that the computation involves only shifts and adds. At the ith iteration, the coordinates (x,y) of the vector are updated using the equations

$$x = x \pm 2^i y$$

$$y = y \pm 2^i x,$$

where the signs are chosen depending on whether the micro-rotation needs to be performed clockwise or counter-clockwise.

Because CORDIC can be performed using shifts and adds it has traditionally been used when a hardware multiplier is unavailable or when hardware resources are at a premium, and has been widely used on FPGAs [6]. When multipliers are available CORDIC is not generally considered the most efficient approach for trigonometric computation. However, when the stages in the CORDIC algorithm are unrolled and pipelined, CORDIC can provide a high-performance method for trigonometric calculation.

CORDIC Unit

The CORDIC algorithm is typically implemented as an iterative process, where the time taken is proportional to the number of iterations, which is proportional to the desired accuracy. To produce a precision of approximately four decimal places for sine and cosine calculations, the CORDIC unit requires 22 stages. If there are many sine and cosine computations to perform, then the stages in the CORDIC algorithm can be unrolled, and this leads to a speed versus area tradeoff in a hardware implementation [7]. In our implementation, the algorithm was completely unrolled, leading to a 22-stage pipeline. Three additional stages were used to convert from floating-point to fixed-point. Floating-point computation was required to interface with the rest of the load flow computation.

The 25-stage pipeline was synthesized on a Statix FPGA (EP1S25) [8] at a maximum frequency of 122 MHz. Each stage can be further divided to separate the most extensive logic and provide for even more parallelism. This method is considered a Super-Pipelined CORDIC unit and forces the entire pipeline to double in length. The benefit of the super-pipelined architecture is realized by synthesis results which record a maximum frequency of approximately 166 MHz; a 36% increase in frequency over the previous architecture.

Performance

The Super-Pipelined CORDIC unit was used in the construction of the Jacobian matrices arising in load flow computation. Performance data was obtained for industry standard benchmarks in Table 1 [9]. The construction of the Jacobian for 1648 and 7917 bus systems required 6852 and 33,945 trigonometric computations.

Table 1: System Data

Name	Source	PV Buses	PQ Buses	Y-bus Non Zeros	Jacobian Non Zeros
1648 Bus	PSS/E	312	1335	6852	22232
7917 Bus	PSS/E	1324	6592	33945	113316

Table 2 shows the timing data comparing sine and cosine computations using the FPGA CORDIC unit compared to software computation on a 2.8 GHz Pentium IV processor. The CORDIC unit obtained a 20x speedup for the trigonometric computations; however, when the time for DMA transfer between the FPGA (Tsunami board) and host PC using a 66 MHz PCI bus is included, the speedup gain was only a factor of two. Note that the PCI bus performance was significantly degraded due the use of the Avalon bus, which is only 32 bits wide. This two-fold reduction in the time for trigonometric computation only led to a 10% reduction in the time to compute the Jacobian.

Table 2: Timing Analysis

	Time (msec) SW	Time (msec) HW	Speedup (SW/HW)
Trig. VS. Trig.			
1648 Bus	2.1212	0.1042	20.3570
7917 Bus	10.197	0.5147	19.8115
Trig. VS. Trig. + DMA			
1648 Bus	2.1212	1.0758	1.9717
7917 Bus	10.197	5.2727	1.9339
Jac. VS Jac.			
1648 Bus	12.8788	11.8182	1.0897
7917 Bus	67.4242	62.0076	1.0874

The timing data in Table 2 shows that an FPGA CORDIC approach to trigonometric computation can lead to a significant improvement over computation on a Pentium; however, the combined software/hardware architecture does not lead to a significant reduction in the time to construct the Jacobian matrices in our load flow application. This is primarily due to the communication overhead and the limitations of the PCI interface that was used. For the performance gain provided by CORDIC to be fully exploited, an architecture must be designed which allows for the necessary data to reside local to the hardware unit or to be accessible at much faster communication speeds. Two possible approaches are being investigated. The first uses embedded processors enhanced with the CORDIC unit and additional floating-point processors and the second uses a complete hardware solution to Jacobian construction.

References

[1] A. R. Bergen, V. Vittal, Power Systems Analysis, 2nd Edition, Prentice Hall, 2000.

[2] Feng Tu and A. J. Flueck, "A Message-Passing Distributed Memory Parallel Powerflow Algorithm," Power Engineering Society Winter Meeting, 2002. IEEE, Volume 1 (2002), pp. 211 – 216.

[3] J. Johnson, P. Vachranukunkiet, S. Tiwari, P. Nagvajara, C. Nwankpa, "Performance Analysis of Load Flow Computation Using FPGA," In Proceedings of the 15th Power Systems Computation Conference, 2005.

[4] J. Johnson, P. Nagvajara, C. Nwankpa, "Sparse Linear Solver for Power System Analysis using FPGA", In Proceedings of the Eighth Annual Workshop on High Performance Embedded Computing (HPEC 2004), 2004.

[5] J. E. Volder, "The CORDIC Trigonometric Computing Technique", IRE Transactions on Electronic Computers, V. EC-8, No. 3, pp. 330-334, 1959.

[6] Ray Andraka, "A Survey of CORDIC Algorithms for FPGA Based Computers," in Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays (FPGA '98), Feb. 22-24, 1998, Monterey, CA. pp. 191-200.

[7] S. Wang and V. Piuri, "A Unified View of CORDIC Processor Design," Application Specific Processors, Edited by Earl E. Swartzlander, Jr., Ch. 5, pp. 121-160, Kluwer Academic Press, November 1996.

[8] http://www.altera.com.

[9] PSS/E. The Shaw Power Group Inc 2005. http://www.shawgrp.com