International Journal of Electrical and Power Engineering 1 (5): 524-529, 2007
ISSN: 1990-7958

Implementation of Division and Square Root Using XSG for FPGA-Based Vector Control Drives

Jean-Gabriel Mailloux, Stéphane Simard and Rachid Beguenane
Groupe ERMETIS, Département des Sciences appliquées, Université du Québec à Chicoutimi,
Chicoutimi (QC), G7H 5B1, Canada

Abstract: The present study deals with hardware implementation of vector control for AC motors using XSG tool. Rotor flux estimation is done through a mathematical formula necessitating two divisions and one square root extraction. These are implemented using non restoring algorithms. FPGA implementations of such operators are considerably smaller compared to the division and the square root derived from XSG Core Library. More importantly, the corresponding modules exhibit a moderate operating frequency of more than 100MHz and present an infinite precision which matches with the idealized floating point Simulink model.

Key words: Vector control, rotor flux, non-restoring algorithm, division, square root, xilinx system generator

INTRODUCTION

To hardcode modern control laws of complex mechatronic systems, Hardware Description Languages (HDL), such as Verilog and VHDL, are not suitable for the designers who are unfamiliar with. A design and verification tool, such as Xilinx System Generator (XSG) for DSP design under Simulink is ideal to tackle the problems within the algorithm design stage by providing the designer an early proof-of-concept of the hard coded control algorithms. In fact XSG takes the graphical algorithmic approach and extend it to FPGA development by using dedicated Simulink Blockset (Xillinx, 2006). The knowledge of HDL, necessary to program FPGA devices, is then not required for DSP designers. This is because the XSG tool can automatically translate the algorithm into FPGA resource and loaded onto the FPGA device.

In the present study, XSG is used to prototype a complex control algorithm which is the vector control, a well known control strategy for AC drives. The use of XSG in electromechanical control systems has been subject of few papers. The authors of Ricci and Le-Huy (2003) have already modelled the widely used DTC algorithm, whereas study (Vasarhelyi *et al*., 2005) has presented the FPGA prototyping of a vector control system for a tandem converter fed induction motor. The divisions and square roots are often involved in many control schemes for electrical drives. For example, the vector control scheme contains a rotating rotor flux analyser. The computation of its modulus and its two axial components involve two divisions and one square root. To rapidly prototype the algorithm, the XSG based FPGA implementation of vector control is first done using IP cores for divisions and square root, leading to a huge logic utilization. Then modules for square root and division, based on non-restoring algorithms, are specifically designed for the estimation of rotor flux.

The hardware implementation of division and square root extraction, either for VLSI or FPGA, is hard given the complexity of the algorithms involved in their computation. This field has been the subject of many investigations in order to implement various algorithms, such as Newton Raphson, SRT-Redundant, Non Redundant, Restoring and Non-Restoring (Kabuo *et al*., 1994; Birman *et al*., 1990; Bannur and Verma, 1985; O'Leary, 1994; Harris *et al*., 1997; Sorkin, 2006; Deschamps, 2006). The Non-Restoring algorithms, for division and square root extraction, are considered as a better compromise in terms of complexity and precision (Li and Chu, 1997; Diromsopa *et al*., 2001; Marnane *et al*., 1997).

The present study is aiming to implement a fixed point division and square root modules using Xilinx System Generator (XSG) blocks. The designed modules are applied to the XSG-based vector control design replacing though the IP cores. A performance comparison, in terms of area and speed, between the two scenarios will be deeply discussed. In brief study reminds the

Corresponding Author: Rachid Beguenane, Groupe ERMETIS, Département des Sciences appliquées, Université du Québec à Chicoutimi, Chicoutimi (QC), G7H 5B1, Canada

theoretical background, respectively of vector control and non restoring algorithms for the division and square root operators, used for a fixed point FPGA implementation. The implementation results in the context of vector control for AC drives and a comparison in the case of using IP cores provided within XSG environment will be performed.

INDUCTION MOTOR VECTOR CONTROL SCHEME

The precise control of AC motors, such as the induction motor, needs an independent control of the stator current components that produce the required magnetizing flux and the electromechanical torque, in the same way as with the DC motor. This can be handled through sophisticated control algorithms, such as the vector control. This has several variants and rotor flux orientation control scheme, shown in Fig. 1, is one of them. This is derived from the electromechanical model of the induction motor in an adequate reference frame (d, q), known as Park domain. The mathematical description and the corresponding vector control algorithm can be found in many references (Beguenane *et al*., 2006).

From Fig. 1 one can notice the vector control scheme presenting many blocks, each performing one particular task. For instance, the rotor flux estimator block evaluates the magnetizing flux using one of the exiting techniques such as Kalman Filter, MRAS, etc. The straightforward and less complicated method is the one derived from the electromechanical model of induction motor. In fact the modulus and the orientation of rotor flux can be computed as follows:

$$\Psi_r = \sqrt{\Psi_{r\alpha}^2 + \Psi_{r\beta}^2}$$

$$\cos\theta = \frac{\Psi_{r\alpha}}{\Psi_r};\ \sin\theta = \frac{\Psi_{r\beta}}{\Psi_r}$$

With

$$\Psi_{r\alpha} = \frac{L_r}{M}\left(\Psi_{s\alpha} - \sigma L_s i_{s\alpha}\right)$$

$$\Psi_{r\beta} = \frac{L_r}{M}\left(\Psi_{s\beta} - \sigma L_s i_{s\beta}\right)$$

$$\Psi_{s\alpha} = \int\left(u_{s\alpha} - R_s i_{s\alpha}\right)dt$$

$$\Psi_{s\beta} = \int\left(u_{s\beta} - R_s i_{s\beta}\right)dt$$

and

Ψ_r : Rotor flux modulus

$\Psi_{r\alpha}$, $\Psi_{r\beta}$: Components of rotor flux in (α, β) stationary frame

$\mu_{s\alpha}$, $\mu_{s\beta}$: Components of stator voltage in (α, β) stationary frame

$i_{s\alpha}$, $i_{s\beta}$: Components of stator current in (α, β) stationary frame

R_s, L_s : Stator resistance and inductance

M : Mutual inductance

σ : Leakage coefficient of the motor

Vector control scheme is fully implemented using XSG under Simulink environment as illustrated in Fig. 2. The computation of rotor flux necessitates two divisions and one square root. For a rapid validation of the algorithm, these operations have been implemented using a CORDIC IP cores provided within XSG interface. This leads to a massive logic in terms of LUTs and FFs. In order to reduce the consumption, the non-restoring division and square root algorithms have been coded in XSG.

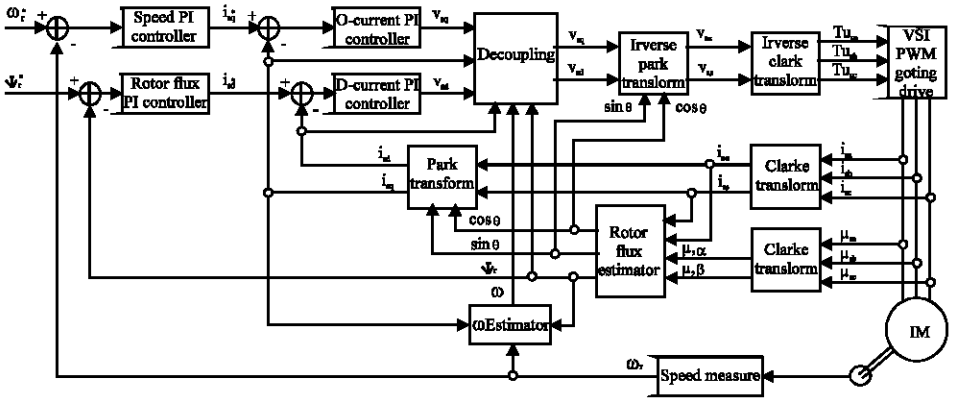

Fig. 1: Rotor flux oriented control scheme

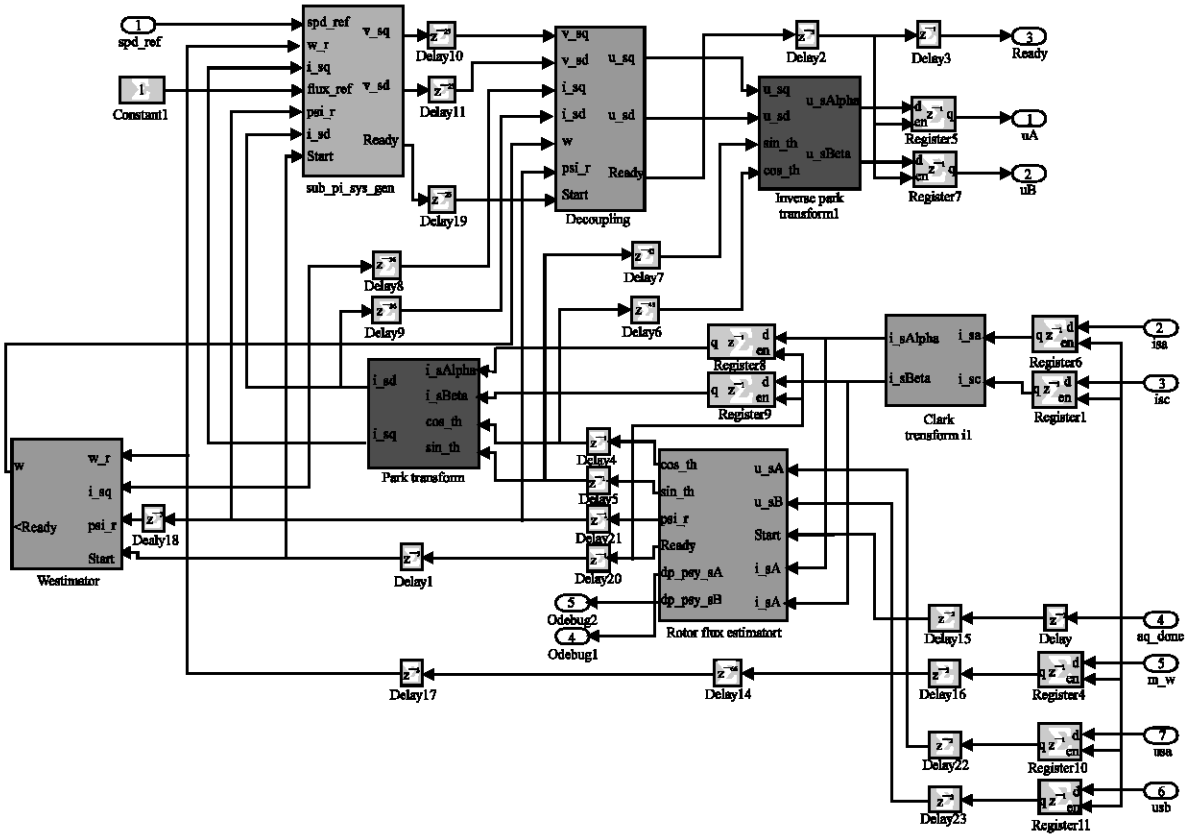

Fig. 2: Vector control coded in XSG

IMPLEMENTATION OF NON RESTORING DIVISION AND SQUARE ROOT USING XSG

Restoring and non-restoring algorithms are the basic digit recurrence schemes for division and square root extraction. The restoring algorithm uses an additional step to conditionally restore the partial remainder to its previous value when the difference is negative. Non-restoring algorithms always store the difference in the partial remainder register, eventually correcting the remainder, if necessary, by an addition in the next step. It is then the preferred algorithm to implement the division and square root operations in FPGA.

Non-restoring division: Binary division reduces to subtracting a set of numbers, each being 0 or a shifted version of the divisor Y, from the dividend X. The main equation for division is

$$X = (Y \cdot Q) + R$$

with Q and R being respectively the quotient and the remainder. This equation, along with the condition $R < Y$,

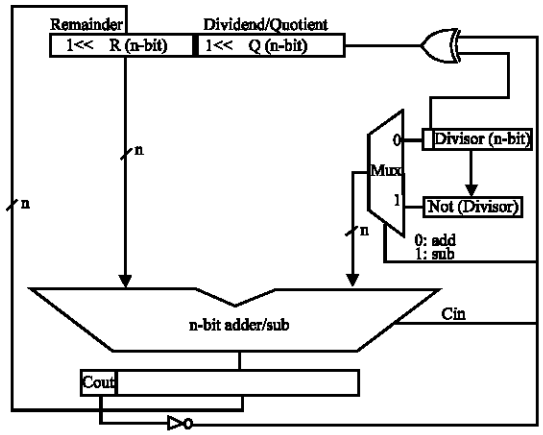

Fig. 3: Iterative non-restoring divider

completely defines unsigned integer division and the expressions for the quotient and the remainder can be derived as follows

$$Q = X / Y,\ R = X - (Y \cdot Q)$$

To obtain a fractional quotient, non restoring division can be performed iteratively by a sequence of additions and subtractions and shifts as illustrated in Fig. 3.

Non-restoring square root: Every pair of most significant bits of the unsigned 2n-bit radicand D = ($d_{2n-1}...d_0$), corresponds one bit of n-bit square root Q = ($q_{n-1}...q_0$). The square root algorithm starts by initializing the quotient Q = ($q_{n-1}...q_0$) to (0...0). Iteratively from j = n-1 down to 0, the remainder R is computed and shifted adequately as illustrated in the scheme of Fig. 4. In case of R≥0, the square root bit q_j is set to 1, otherwise it is reset to 0.

Implementation with XSG: The non-restoring division and square root algorithms were optimally and synchronously coded in XSG under Simulink environment using embedded blocs such as adders, muxes, inverters, etc. The implementations are fully parameterized with the widths declared as generic parameters of the XSG modules. This permits to select an arbitrary precision and word lengths for different operators of the division and square root. The mapping onto FPGA is done through ISE tool and the designs are analyzed in order to define their performances in terms of area usage and speed. Figure 5 and 6 illustrate the XSG codes of the division and square root for a particular case of 32-bits operators with 14 bits for binary point.

ANALYSIS AND IMPLEMENTATION RESULTS

To evaluate the precision of the proposed division and square root modules relatively to CORDIC IP Cores from XSG interface, Fig. 7 and 8 shows the computed rotor flux with respect to the idealized Simulink model for each case. The test is conducted under a complete closed loop where the motor and its power drive are simulated using SimPowerSystem. The rotor flux is controlled to have a reference value of 1Wb, under various speed profiles and loads.

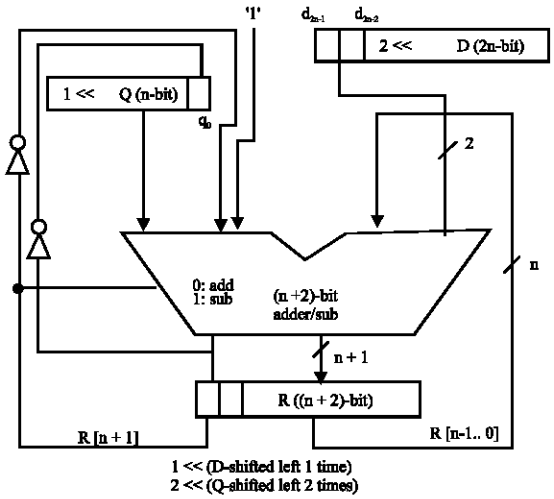

Fig. 4: Iterative non-restoring SQRT

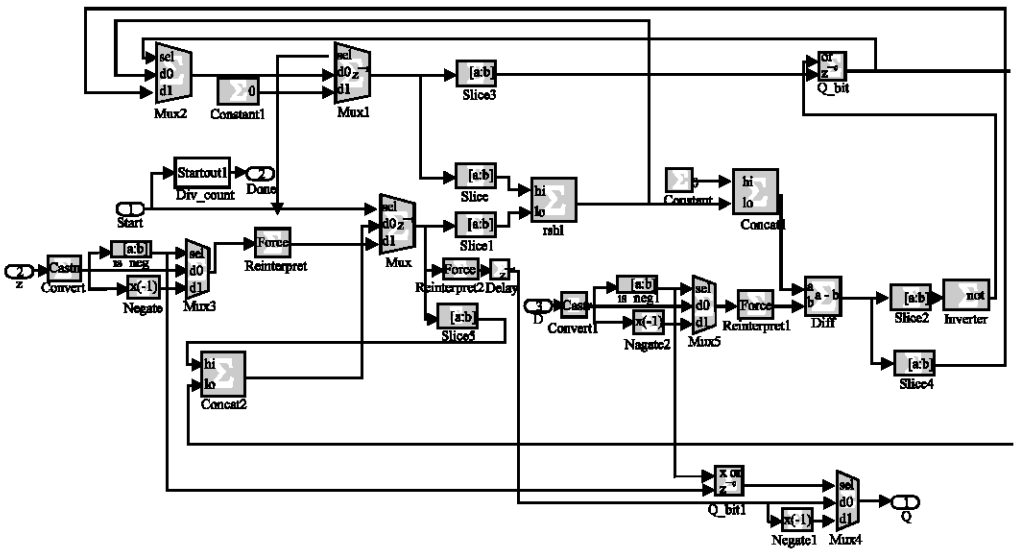

Fig. 5: XSG code for parameterized non restoring division

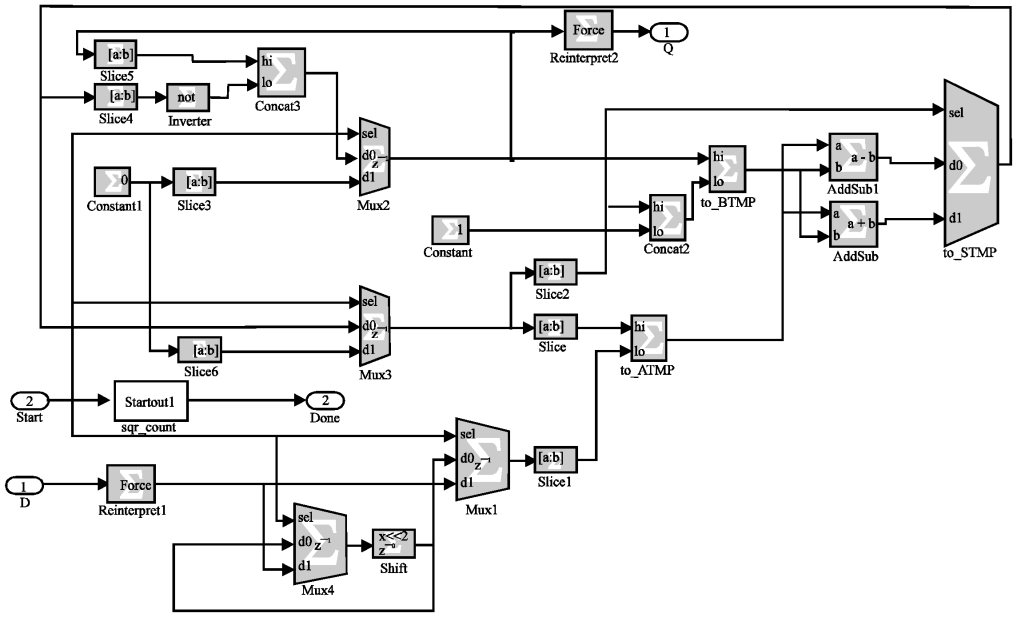

Fig. 6: XSG code for parameterized non restoring SQRT

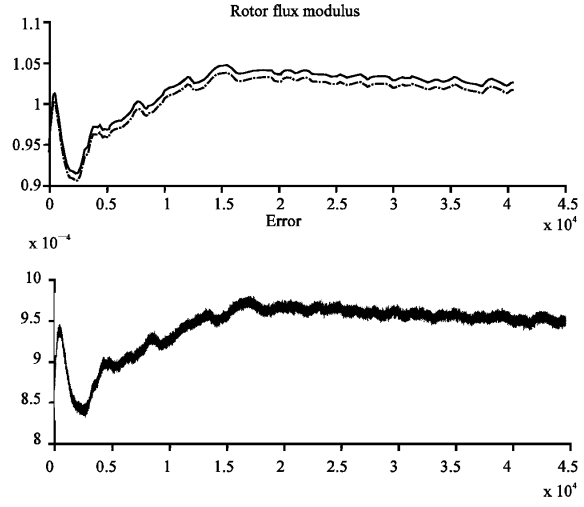

Fig. 7: Rotor Flux computed with CORDIC IP Cores vs. a floating point value from Simulink model

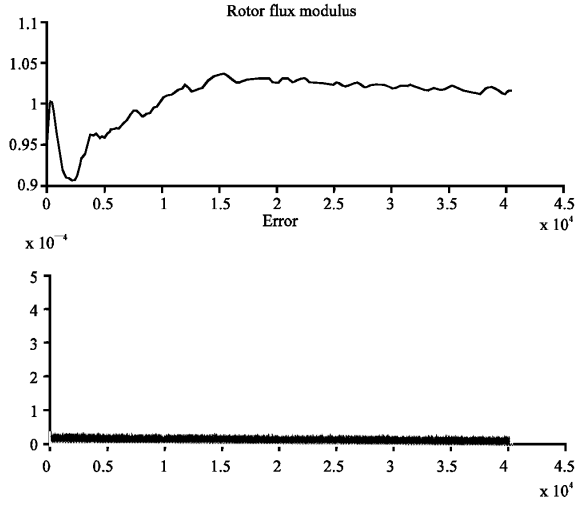

Fig. 8: Rotor Flux computed with non restoring operators vs. a floating point value from Simulink model

One can notice, for a particular precision of 32-bit, with the proposed modules the error is nil whereas it is about 1% when IP Cores are used.

Synthesis was targeted to evaluate the FPGA resources. The synthesis results are gathered in Table 1 for the division module and in Table 2 for the square root. In each case, the maximal operating frequency is given.

Table 1: Synthesis results for FPGA implementation of non-restoring division and CORDIC based division (32-bits, 14 bits for binary point)

Parameter	Non restoring division	CORDIC based division
Slices	124	1806
FFs	96	3132
LUTs	218	2981
Embedded multipliers	0	3
Steps for completion	48	47
Speed in MHz	173	77

Table 2: Synthesis results for FPGA implementation of non-restoring SQRT and CORDIC based SQRT (32-bits, 14 bits for binary point)

Parameter	Non restoring division	CORDIC based division
Slices	60	1377
FFs	82	2289
LUTs	103	2425
Embedded multipliers	0	6
Steps for completion	24	51
Speed in MHz	238	59

CONCLUSION

Some signal digital processing applications exhibit algorithms with one or more fixed-point square roots and/or divisions. In FPGA, integer square rooters and dividers are provided from an IP libraries provided by FPGA vendors. Besides their huge area utilization, they are limited in terms of precision and often the precision of the result don't exceed 32 bits. In the present study, the vector control algorithm for induction motor drives is implemented using XSG under Simulink. The rotor flux, which necessitates a couple of divisions and one square root extraction, is implemented based on a non restoring division and square root algorithms. The detailed study has shown the main advantage of such implementation which resides on its huge logic saving. Moreover, the proposed implementations of such operators lead to an impressive precision and are many times fast compared to the IP Cores.

ACKNOWLEDGEMENT

This research is funded by a grant from the National Sciences and Engineering Research Council of Canada (NSERC). Canadian Microelectronics Corporation (CMC) Microsystems provided development tools and support through the System-on-Chip Research Network (SOCRN) program.

REFERENCES

Bannur, J. and A. Varma, 1985. The VLSI Implementation of a square root. Algorithm. In: IEEE Symp. Computer Arithmetic, IEEE. Comput. Soc. Washington DC, pp: 159-165.

Beguenane, R., J.G. Mailloux, S. Simard and A. Tisserand, 2006. Towards the System-on-Chip Realization of a Sensorless Vector Controller with Microsecond-order Computation Time. In 19th IEEE Canadian Conference on Electrical and Computer Engineering, Ottawa, Canada.

Birman, M. *et al*., 1990. Developing the WTL3170/3171 Sparc Floating-Point Coprocessors. IEEE. Micro, 10: 55-64.

Deschamps, J.P., G.J.A. Bioul and G.D. Sutter, 2006. Synthesis of Arithmetic Circuits. Copyright © John Wiley and Sons, Inc.

Harris, D.L., S.F. Oberman and M.A. Horowitz, 1997. SRT division architectures and implementations. In: Proc. 13th IEEE. Int. Symp. Comput. Arithmetic, pp: 18-25.

Kabuo, H. *et al*., 1994. Accurate rounding scheme for the newton-raphson method using redundant binary representation. IEEE. Trans. Comput., 43: 43-51.

Li, Y. and W. Chu, 1997. Implementation of single precision floating point square root on FPGAs. In: 5th IEEE. Symp. FPGA-Based Custom Comput. Mach. (FCCM), Napa, California, USA., pp: 226-233.

Marnane, W.P., S.J. Bellis and P. Larsson-Edefors, 1997. Bit-serial interleaved high speed division. Elec. Lett., 33: 1124-1125.

O'Leary, J.W., M. Leeser, J. Hickey and M. Aagaard, 1994. Non-restoring integer square root: A Case Study in Design by Principled Optimization. In TPCD, 2nd Int. Conf. Theorem Provers in Circuit Design-Theory, Practice and Experience, Springer-Verlag, London, UK., pp: 52-71.

Piromsopa, K., C. Aporntewan and Chongstitvatana, 2001. An FPGA implementation of a fixed-point square root operation. In: International Symposium on communications and Information Technologies, ISCIT.

Ricci, F. and H. Le-Huy, 2003. Modeling and simulation of FPGA-based variable-speed drives using Simulink. Math. Comput. Simulation, 63: 183-195.

Sorokin, N., 2006. Implementation of high-speed fixed-point dividers on FPGA. JCS and T J., Vol. 6.

Vásárhelyi, J., M. Imecs, C. Szabó and I.I. Incze, 2005. FPGA Implementation vector control of tandem converter fed induction machine. In 6th International Symposium of Hungarian, Researchers on Computational Intelligence, Budapest, Magyar.

Xilinx Inc, 2006. System Generator for DSP. http://www.xilinx.com/ise/optional_prod/system_generator.htm