

AN OPTIMIZED SQUARE ROOT ALGORITHM FOR IMPLEMENTATION IN FPGA HARDWARE

Tole Sutikno

Department of Electrical Engineering, Universitas Ahmad Dahlan (UAD)
Jln. Prof. Soepomo, Janturan, Yogyakarta 55164, Telp. +62-274-379418, Fax. +62-274-564604
e-mail: tole@ee.uad.ac.id

Abstrak

Makalah ini menyajikan metode perhitungan digit-demi-digit yang dioptimalkan untuk menyelesaikan perhitungan akar kuadrat yang rumit dalam perangkat keras, sebagai algoritma sederhana yang diusulkan untuk implementasi pada field programmable gate array (FPGA). Prinsip utama dari metode ini adalah operasi-operasi penggeseran dua-bit dan pengurang-multipleks, untuk mendapatkan implementasi yang lebih sederhana dan perhitungan yang lebih cepat. Algoritma ini telah digunakan untuk implementasi akar kuadrat biner tak bertanda 32-bit dan 64-bit berbasis FPGA secara sukses. Hasil penelitian menunjukkan bahwa metode yang diusulkan paling efisien sumber daya perangkat keras, bila dibandingkan metode lainnya. Selain itu, strategi ini dapat dengan mudah dikembangkan untuk implementasi akar kuadrat yang lebih besar.

Kata kunci: perhitungan angka-oleh-digit, FPGA, Square Root

Abstract

This paper presents an optimized digit-by-digit calculation method to solve complicated square root calculation in hardware, as a proposed simple algorithm for implementation in field programmable gate array (FPGA). The main principle of proposed method is two-bit shifting and subtracting-multiplexing operations, in order to achieve a simpler implementation and faster calculation. The proposed algorithm has conducted to implement FPGA based unsigned 32-bit and 64-bit binary square root successfully. The results have shown that proposed method is most efficient of hardware resource compare to other methods. In addition, the strategy can be expanded to larger number easily.

Keywords: digit-by-digit calculation, FPGA, Square Root

1. INTRODUCTION

It is well-known that the direct torque control method (DTC) for AC motors has simple structure and good behaviors such as fast torque response, no requirements for PWM pulse generation, no requirements for coordinate transformation, no position encoder and current regulators [1-7].

The DTC algorithm is usually implemented by serial calculations based on a Microcontroller or Digital Signal Processing (DSP) [8-11]. These are truly software-based platform and not adequate to implement a control methods which require very high speed response. As suitable solution, it is proposed FPGA to support execution very fast tasks [12-14]. However, it is not easy to implement DTC in FPGA hardware. One of problem has been addressed mainly in complicated square root calculation. It is hard to implement on FPGA [15-17].

There many algorithms has proposed to solve square root, such as Rough estimation [18], Babylonian method [19], exponential identity [20], Taylor-Series Expansion Algorithm [21], Newton-Raphson method [22-24], and sequential algorithm (digit-by-digit calculation method) [25-29]. Nevertheless, the methods above usually do not focus to solve square root problem in DTC implementation based on FPGA. This paper proposes digit-by-digit calculation method as a simple strategy to solve complicated square root. The proposed implementation strategy is different compared to strategies in [25-29]. An optimization is also done by eliminates circuitry

that is not needed. It is addressed to support DTC implementation in FPGA hardware, and it hopes that it gives rise simpler implementation and faster calculation.

2. DIGIT-BY-DIGIT CALCULATION METHOD

In digit-by-digit calculation method, the each digit of the square root is found in a sequence where it only one digit of the square root is generated at each iteration [29]. It has several advantages, such as: every digit of the root found is known to be correct and it will not have to be changed later; if the square root has to expand, it will terminate after the last digit is found; and the algorithm works for any number base (of course the process depends on number base).

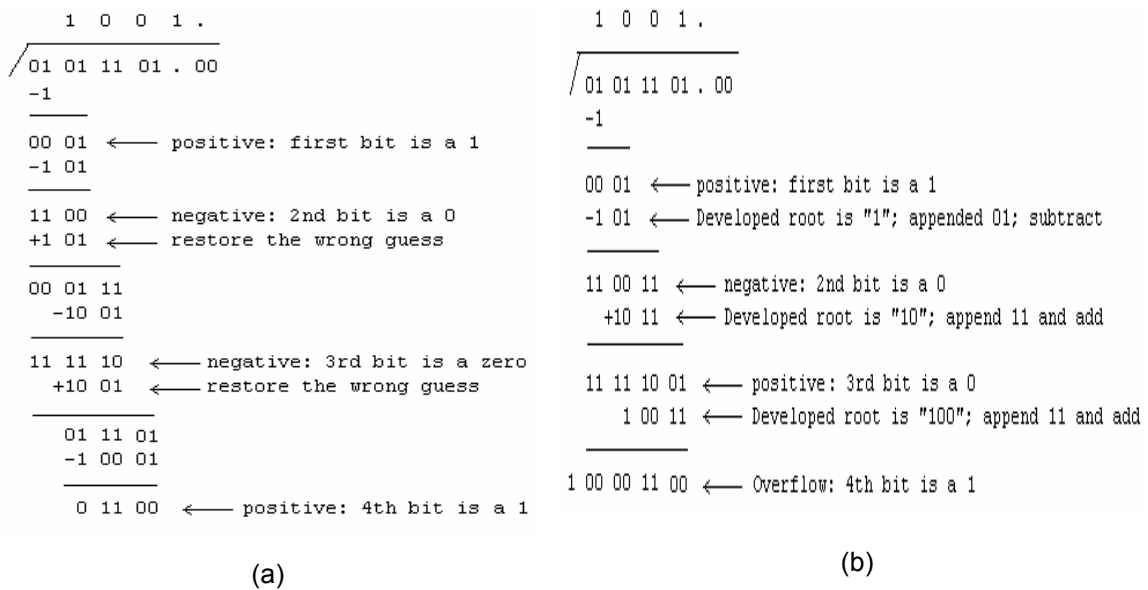


Figure 1. The example of digit-by-digit calculation to solve square root: (a) restoring algorithm; (b) non restoring algorithm

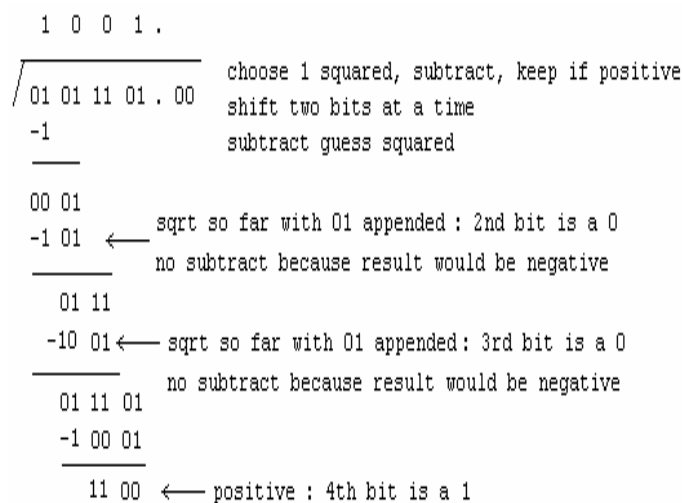


Figure 2. The example of using modified non restoring digit-by-digit calculation algorithm to solve square root

In general, this method can be divided in two classes, i.e. restoring and non restoring digit-by-digit algorithm [29]. In restoring algorithm, the procedure is composed by taking the square root obtained so far, appending 01 to it and subtracting it, properly shifted, from the current remainder. The 0 in 01 corresponds to multiplying by 2; the 1 is a new guess bit. The new root bit developed is truly 1, if the resulting remainder is positive, and vice versa is 0, which the remainder must be restored by adding the quantity just subtracted. It is different, in non restoring algorithm does not restore the subtraction if the result was negative. Instead, it appends a 11 to the root developed so far and on the next iteration it performs an addition. If the addition causes an overflow, then on the next iteration you go back to the subtraction mode [30]. The Figure 1 is the example gives to take the binary square root of 01011101 (equivalent with 93 decimal).

A little different than non restoring digit-by-digit algorithm in Figure 1 (b), a modification as shown on Figure 2 can be conducted to give simpler implementation and faster calculation. In this modification, it only uses subtract operation and append 01, while add operation and append 11 is not used. This paper adopts this modification to implement unsigned 64-bit binary square root based on FPGA.

3. PROPOSED SQUARE ROOT ALGORITHM

Samavi, et al. [29] has improved classical non-restoring digit-by-digit square root circuit by eliminate redundant blocks. Their circuit is referred to as the reduced area non restoring circuit. However, it still based on constant digit of 01 or 11 and add-subtract as the main building block (still refer to Figure 1 b). This paper offers a simple alternative solution that it only uses subtracts operation and appends 01. As consequent, the subtract-multiplex is used as the main building block (refer to Figure 2). The principle of proposed algorithm can be described as shown in Figure 3.

- Step 0. Start
- Step 1. Initialization radicand (the n-bit number will be squared root), quotient (the result of squared root), and remainder. To calculate square root of a 2n bit number, it needs n stage pipelines to implement the proposed algorithm.
- Step 2. Beginning at the binary point, divide the radicand into groups of two digits in both direction.
- Step 3. Beginning on the left (most significant bit), select the first group of one or two digit (If n is odd then the first groups is one digit, and vice versa)
- Step 4. Choose 1 squared, and then subtract.
Fist developed root is "1" if the result of subtract is positive, and vice versa is "0"
- Step 5. Shift two bits, subtract guess squared with append 01.
Nth-bit squared is "1" if the result of subtract is positive, and Because of subtract operation is done
else
Nth-bit squared is "0", and not subtract
- Step 6. Go to step 5 until end group of two digits
- Step 7. End

Figure 3. The principle of proposed algorithm to solve square root

A simple hardware implementation of the non-restoring digit-by-digit algorithm for unsigned 6-bit square root by an array structure is shown in Figure 4. The radicand is P (P5,P4,P3,P2,P1,P0), U (U2,U1,U0) as quotient and R (R4,R3,R2,R1,R0) as remainder. It can be shown that the implementation needs 3 stage pipelines. The main building blocks of the array are blocks called as *controlled subtract-multiplex (CSM)*. Figure 5 present the details of a CSM. Input of the building block is x,y,b and u, and as output is bo (borrow) and d (result). If $u=0$, then $d \leq x-y-b$ else $d \leq x$.

The generalization of simple implementation of the non-restoring digit-by-digit algorithm for unsigned n-bit square root by an array structure is shown in Figure 6. Each row (stage) of

the circuit in Figure 6 executes one-iteration of the non-restoring digit-by-digit square root algorithm, where it only uses subtracts operation and appends 01.

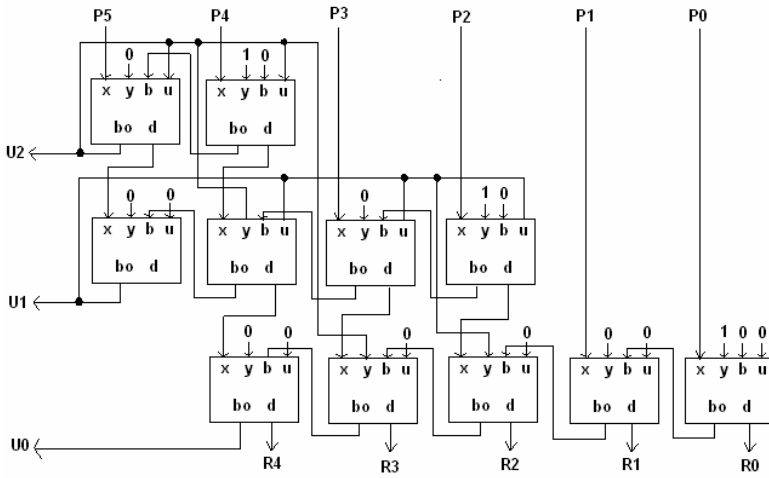


Figure 4. A simple hardware implementation of the non-restoring digit-by-digit algorithm for unsigned 6-bit square root

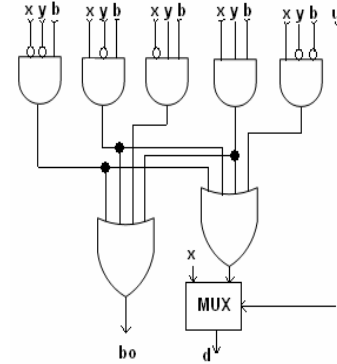


Figure 5. Internal structure of a CSM block

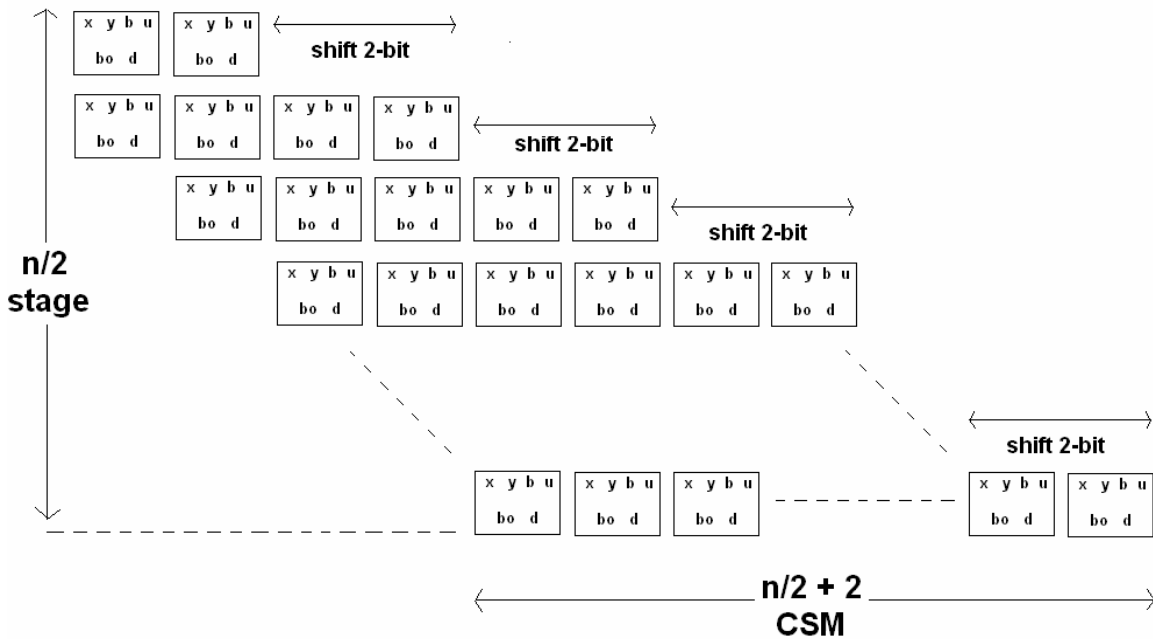


Figure 6. A simple hardware implementation of the non-restoring digit-by-digit algorithm for unsigned n-bit square root

To be optimizer hardware resource saving of the implementation above, specialized entities can be created as building block components. It will eliminate circuitry that is not needed. As example, the implementation in Figure 6 for unsigned 6-bit square root can be optimized become as shown in Figure 7 (in this case, the remainder is ignored, because in the DTC drive, it is not required). The specialized entities A, B, C, D and E are minimized CSM when input ybu=100, yu=00, u=0, yu=10, and y=0 respectively, and the remainder is ignored.

The generalization of optimized simple implementation of the non-restoring digit-by-digit algorithm for unsigned n-bit square root is shown in Figure 8.

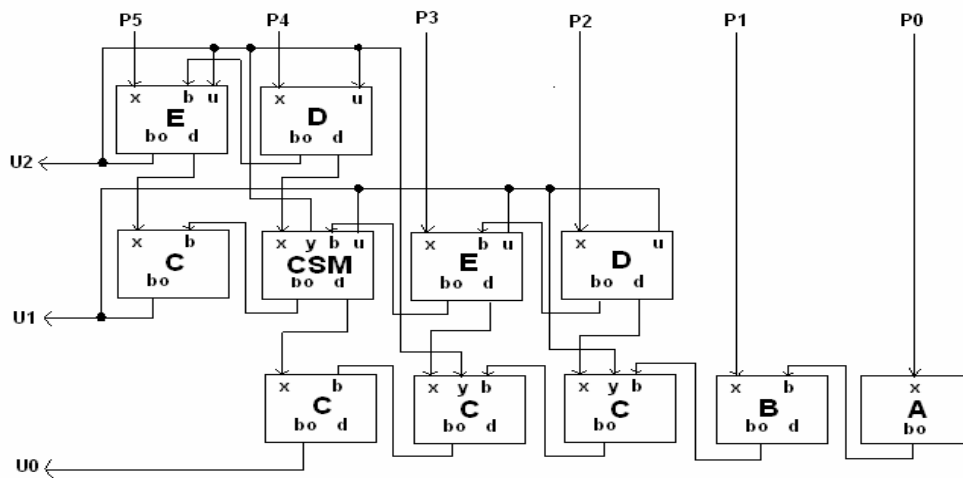


Figure 7. Optimized simple hardware implementation of the non-restoring digit-by-digit algorithm for unsigned 6-bit square root

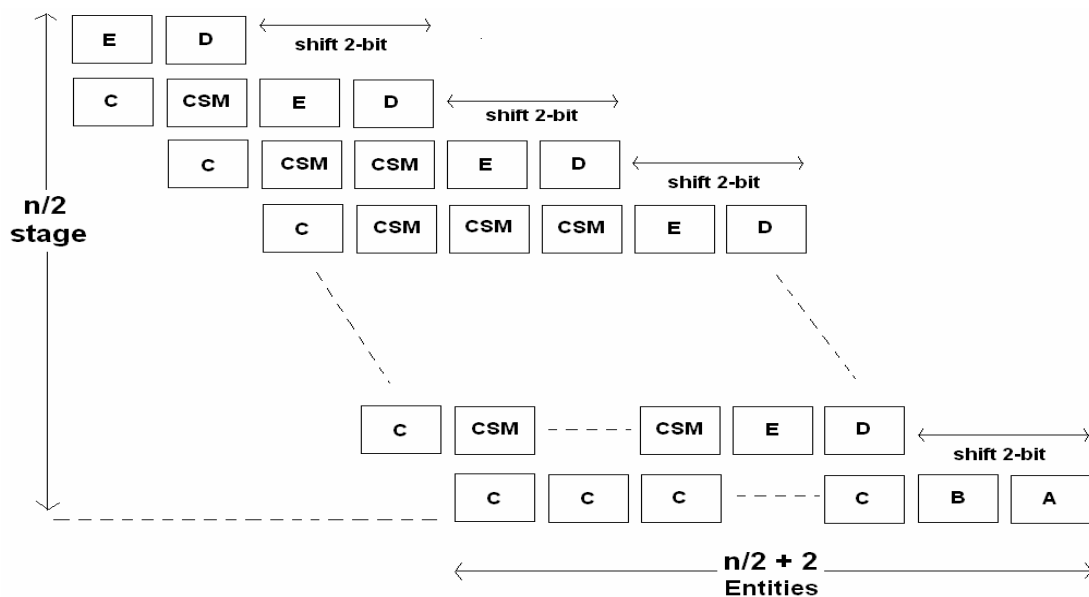


Figure 8. Optimized simple hardware implementation of the non-restoring digit-by-digit algorithm for unsigned n-bit square root

4. RESULTS AND ANALYSIS

In the previous sections, optimized simple hardware implementation method of the non-restoring digit-by-digit algorithm for square root and the difficult task in DTC to calculate square root were explained. In this section, simulation results of 32-bit and 64-bit square root based on Altera APEX 20KE FPGA by using method above are presented, as shown in Figure 9. In this simulation, P is radicand and U is quotient. The results showed that the implementation has succeeded and worked properly.

Based on compilation report, to implement 32-bit and 64-bit square root using optimized simple hardware implementation method of the non-restoring digit-by-digit algorithm are needed

256 and 1023 logic element (LE) respectively. The comparison of results obtained from different implementation method is shown in Table 1.

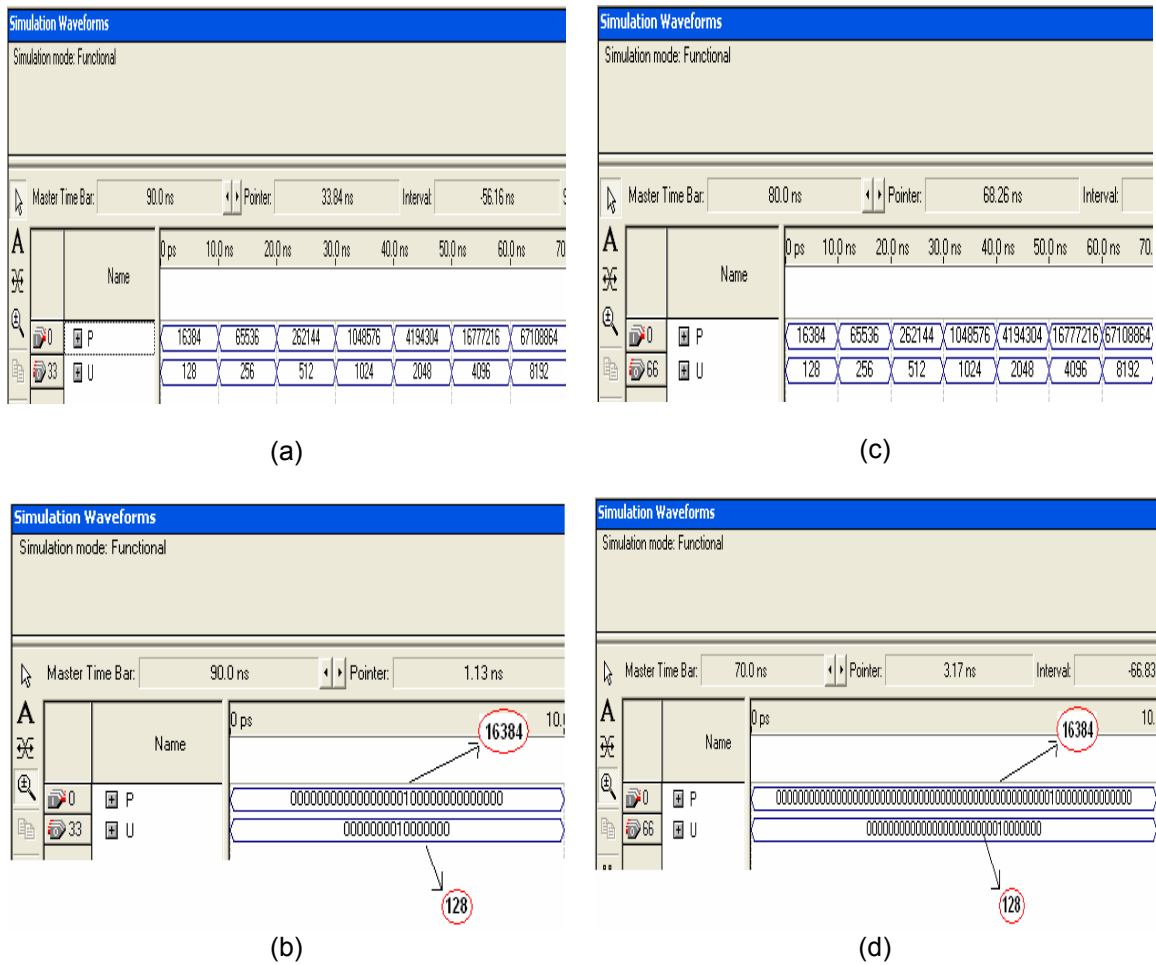


Figure 9. Simulation result of n-bit square root using optimized simple hardware implementation method of the non-restoring digit-by-digit algorithm: (a) 32-bit in decimal display, (b) 32-bit in binary display, (c) 64-bit in decimal display, (d) 64-bit in binary display

Table 1. The comparison of logic element usage

No	Method	LE Usage	
		32-bit square root	64-bit square root
1	Classical-NR	1008	4092
2	Reduced-Area-NR	632	2464
3	Modular-NR	624	2468
4	Simple-X-Module	648	2488
5	Proposed	256	1023

Note: Altera APEX 20KE & Xilinx Virtex-E, 1 LC = 1 LE, and 1 CLB = 4 LE [31]

This comparison of LE or logic cell (LC) usage is listed based on references [29] and [30]. The number of employed LE indicates the size of the implemented circuit “hardware resource”. Table 1 showed that proposed method is most efficient of hardware resource. Based on Figure 8, the strategy is very easy to be expanded for larger number to solve complicated square root problem in FPGA implementation.

5. CONCLUSION

This contribution presented digit-by-digit calculation method as a proposed simple strategy for implementation in field programmable gate array (FPGA) hardware mainly to solve complicated square root. The main principle of proposed method is two-bit shifting and subtracting-multiplexing operations. The proposed strategy has conducted to implement FPGA based unsigned 32 bit and 64-bit binary square root successfully. The results have shown that proposed method is most efficient of hardware resource compare to other methods. The method also can be expanded to larger number easily, to solve complicated square root problem in FPGA implementation.

REFERENCES

- [1]. Takahashi I, Noguchi T. A New Quick-Response and High-Efficiency Control Strategy of an Induction Motor. *IEEE Transactions on Industry Applications*. 1986; IA-22(5): 820-827.
- [2]. Depenbrock M. Direct Self Control (DSC) of Inverter-fed Induction Machine. *IEEE Trans. on Power Electronics*. 1988; 3(4): 420-429.
- [3]. Habetler TG, Profumo F, Pastorelli M, Tolbert LM. Direct Torque Control of Induction Machines Using Space Vector Modulation. *IEEE Transactions on Industry Applications*. 1992; 28(5): 1045-1053.
- [4]. Zhong L, Rahman MF, Hu WY, Lim KW. Analysis of Direct Torque Control in Permanent Magnet Synchronous Motor Drives. *IEEE Transactions on Power Electronics*. 1997; 12(3): 528-536.
- [5]. French C, Acarnley P. Direct Torque Control of Permanent Magnet Drives. *IEEE Transactions on Industry Applications*. 1996; 32(5): 1080-1088.
- [6]. Yong L, Zhu ZQ, Howe D. Direct Torque Control of Brushless DC Drives with Reduced Torque Ripple. *IEEE Transactions on Industry Applications*. 2005; 41(2): 599-608.
- [7]. Yong L, Zhu ZQ, Howe D. Commutation-Torque-Ripple Minimization in Direct-Torque-Controlled PM Brushless DC Drives. *IEEE Transactions on Industry Applications*. 2007; 43(5): 1012-1021.
- [8]. Bose BK, Szczesny PM. A Microcomputer-based Control and Simulation of An Advanced IPM Synchronous Machine Drive System for Electric Vehicle Propulsion. *IEEE Transactions on Industrial Electronics*. 1988; 35(4): 547-559.
- [9]. Lianbing L, Hexu S, Xiaojun W, Yongqing T. A High-Performance Direct Torque Control Based on DSP in Permanent Magnet Synchronous Motor Drive. Proceedings of the 4th World Congress on Intelligent Control and Automation. 2002; 2: 1622-1625.
- [10]. Weijie L. Implementation of Direct Torque Control for Permanent Magnet Synchronous Motor with Space Vector Modulation Based on DSP. 8th International Conference on Signal Processing. 2006; 4: 101-104.
- [11]. Cruz SMA, Toliyat HA, Cardoso AJM. DSP Implementation of The Multiple Reference Frames Theory for The Diagnosis of Stator Faults in A DTC Induction Motor Drive. *IEEE Transactions on Energy Conversion*. 2005; 20(2): 329-335.
- [12]. Monmasson E, Cirstea MN. FPGA Design Methodology for Industrial Control Systems: A Review. *IEEE Transactions on Industrial Electronics*. 2007; 54(4): 1824-1842.
- [13]. Kowalski CT, Lis J, Orłowska-Kowalska T. FPGA Implementation of DTC Control Method for the Induction Motor Drive. The International Conference on Computer as a Tool (EUROCON). 2007:1916-1921.
- [14]. Colli VD, Di Stefano R, Marignetti F, Scarano M. Design of a System-on-Chip PMSM Drive Sensorless Control. IEEE International Symposium on Industrial Electronics (ISIE). 2007: 2386-2391.
- [15]. Yamin L, Wanming C. Implementation of Single Precision Floating Point Square Root on FPGAs. IEEE Symposium on FPGA for Custom Computing Machines. Napa, California, USA. 1997: 226-232.
- [16]. Piromsopa K, Aporntewan C, Chongstitvatana P. An FPGA Implementation of A Fixed-Point Square Root Operation. Int. Symp. on Communications and Information Technology (ISCIT 2001). ChiangMai, Thailand. 2001: 100-102.
- [17]. Lachowicz S, Pfliederer HJ. Fast Evaluation of the Square Root and Other Nonlinear Functions in FPGA. 4th IEEE International Symposium on Electronic Design, Test and Applications (DELTA). 2008: 474-477.

-
- [18]. Ercegovic MD. *On Digit-by-Digit Methods for Computing Certain Functions*. Conference Record of the Forty-First Asilomar Conference on in Signals, Systems and Computers (ACSSC). 2007: 338-342.
- [19]. Kosheleva O. *Babylonian Method of Computing The Square Root: Justifications Based on Fuzzy Techniques and on Computational Complexity*. Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS). 2009: 1-6.
- [20]. Ligon WB, Monn IG, Stanzione D, Stivers F, Underwood KD. *Implementation and Analysis of Numerical Components for Reconfigurable Computing*. IEEE Proceedings Aerospace Conference. 1999; 2:325-335.
- [21]. Taek-Jun K, Sondeen J, Draper J. *Floating-Point Division and Square Root Implementation Using A Taylor-Series Expansion Algorithm*. 15th IEEE International Conference on Electronics, Circuits and Systems (ICECS). 2008:702-705.
- [22]. Kabuo H, Taniguchi T, Miyoshi A, Yamashita H, Urano M, Edamatsu H, Kuninobu S. *Accurate Rounding Scheme for The Newton-Raphson Method Using Redundant Binary Representation*. *IEEE Transactions on Computers*. 1994; 43(1): 43-51.
- [23]. Allie M, Lyons R. *A Root of Less Evil [Digital Signal Processing]*. *IEEE Signal Processing Magazine*. 2005; 22(2): 93-96.
- [24]. Liang-Kai W, Schulte MJ. *Decimal Floating-Point Square Root Using Newton-Raphson Iteration*. 16th IEEE International Conference on Application-Specific Systems, Architecture Processors (ASAP). 2005: 309-315.
- [25]. Tchoumatchenko V, Vassileva T, Gurov P. *A FPGA Based Square-Root Coprocessor*. Proceedings of the 22nd EUROMICRO Conference Beyond 2000: Hardware and Software Design Strategies. 1996: 520-525.
- [26]. Takagi N, Takagi K. *A VLSI Algorithm for Integer Square-Rooting*. International Symposium on Intelligent Signal Processing and Communications (ISPACS). 2006: 626-629.
- [27]. Yamin L, Wanming C. *Parallel-Array Implementations of A Non-Restoring Square Root Algorithm*. IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD). 1997: 690-695.
- [28]. Xiumin W, Yang Z, Qiang Y, Shihua Y. *A New Algorithm for Designing Square Root Calculators Based on FPGA with Pipeline Technology*. Ninth International Conference on Hybrid Intelligent Systems (HIS). 2009: 99-102.
- [29]. Samavi S, Sadrabadi A, Fanian A. *Modular Array Structure For Non-Restoring Square Root Circuit*. *Journal of Systems Architecture*. 2008;54(10): 957-966.
- [30]. *Comparing Altera APEX 20KE & Xilinx Virtex-E Logic Densities*. Altera Corporation. 2010.