A New Algorithm for Designing Square Root Calculators based on FPGA with Pipeline Technology

Xiumin Wang, Yang Zhang, Qiang Ye, Shihua Yang
College of Information Engineering
China Jiliang University
Hangzhou , China
e-mail: wxm6341@163.com

***Abstract*—After analyzing the advantages and disadvantages of all the general algorithms adopted now in designing square root calculators on FPGA chips with pipeline technology, a new algorithm is proposed based on the relationship between increment and remainder and then its principle is analyzed in details in the paper. The algorithm is realized on the Quartus2 development platform with Verilog HDL language and the flow chart which implements the algorithm is given .The simulation results show that it is characterized by occupying less resource and processing in a faster speed as well as being easier to implement. Therefore it is an effective algorithm for implementing square root calculators on commonly-used FPGA chips with pipeline technology.**

Keywords-pipeline ;FPGA; square root; Verilog HDL; algorithm

I. INTRODUCTION

Nowadays most pipeline technology based square root calculators implemented on FPGA chips adopt the Non-Restoring Square Root Algorithm[1]. Although the algorithm proposed in [2] costs less resource when compared with others, its processing speed is low. It will take 36 clock cycles to output the first 16 bit calculation result even with its best cost/performance evaluation solution 2.5s-Root when processing a 32 bit floating point number Although the algorithm proposed in [3] has a high sped processing speed, when the calculation result is 24 bit it takes only 15 clock cycles to output the first one, it consumes too many CLB blocks .Therefore it is not suitable for general FPGA which doesn't have much such resource.. The shortcoming of the algorithm proposed in[4] is that it needs too much pipeline stages. When processing double precision floating point numbers with 55 bits mantissa, 55 stages is needed. Though the number of stages can be reduced in order to save the chip area by combing server stages into one , the highest working frequency of the circuit will also reduce as a result. To solve the shortcomings of the above algorithm this paper proposes a new algorithm. Square root calculators with pipeline technology based on this algorithm have the following characters as well as the advantage all the above algorithm have. Firstly it has a good generality. i.e. it can easily be implemented on a general-use FPGA since it doesn't need any special hardware structure. Secondly when the input is a $2n$(n=1,2,3……) bit integer , it takes only n clock cycles to output the first result. Thirdly it needs fewer pipeline stages compared with others. Besides each stage has almost the same latency so it can take full use of the advantages of pipeline technology. In a word the algorithm has a better integrated performance compared with other algorithm when take all the following 3 factors into consideration: generality, processing speed and resource consumed

II. THE PRINCIPLE OF THE ALGORITHM

The maximum of a $2n$ bit (n=1,2,3……)binary integer number is $2^{2n}-1$ and the maximum of the square of a n bit binary is $2^{2n}-2^{n+1}+1<2^{2n}$ i.e. $(2^n-1)^2<2^{2n}$,while the minimum of the square of a $(n+1)$ bit binary is $2^{2n}>2^{2n}-1$ .so the square root of any binary number no more than $2n$ bit is no more than n bit. Since a $(2n-1)$ bit binary can be considered as a $2n$ bit one with the most significant bit been set to 0,so we take the $2n$ bit binary as an example to illustrate the principle of the algorithm.

Suppose $a[2n-1:0]$ is the $2n$ bit input radicand, $b[n-1:0]$ is its square root, $t3[n-1:0]$ is used to store the intermediate calculation result of b , both are initially set to 0. $num0[2n-1:0]$ stores the partial remainder of each stage which is initially set to a . $num1[2n-1:0]$ is the increment of the square of $t3$ when it has a certain increment. The whole process can be divided into n steps. Each step performs the same operation i.e. to judge whether $t3$ can have an increment of 2^{n-k} where k =1,2,……n indicating that this is the k th stage. From the formula $(a+b)^2=a^2+b(b+2a)$ where a and b are two arbitrary numbers, we can deduce that when $t3$ has an increment of 2^{n-k} ,its square will have an increment of $num1=2^{n-k}\times(2^{n-k}+2\times t3)$ correspondingly. Judge the relationship between the size of $num0$ and $num1$. If $num0\geq num1$ it indicates that if $t3$ has an increment of 2^{n-k} its square will not exceed the input radicand, so update $t3$ and $num0$ according to the formula $t3=t3+2^{n-k}$ and $num0=num0-num1$ respectively. Otherwise it indicates that the square root of a is less than $t3+2^{n-k}$,so $num0$, $t3$ should be kept unchanged in this stage. The following $(k+1)$ th stage is to judge whether $t3$ can have an increment of 2^{n-k-1} .When the n th stage finishes, the value of $t3$ is the required square root.

978-0-7695-3745-0/09 $25.00 © 2009 IEEE
DOI 10.1109/HIS.2009.27

We can deduce that in the k th stage $num0$ is less than 2^{2n+2-k} i.e. the bits from $num0[2n-1]$ to $num0[2n-k+2]$ of $num0$ all become zeros ,otherwise from the formula $2^{n+1-k}\times(2\times\sum_{i=n-1}^{k+1}b_k\times 2^k+2^{k+1})<2^{n+1-k}\times 2^{n+1}=2^{2n-k+2}\le num0$ we can conclude that there must be an error in the previous stages where $t3$ could have had an increment of 2^{j+1} where j is an integer larger or equal to k . Besides it is obvious that the width of $num1=2^{n-k}\times(2^{n-k}+2\times t3)$ is less than $2n-k+2$,and the bits from $num1[n-k-1]$ to $num0[0]$ are all zeros. Considering these two points we only need to take the bits $num1[(2n-k):(n-k)]$ and $num0[[2n-k+1):(n-k)]$ into consideration during the k th stage which can be stored in a $(n+2)$ bit variance $t0$ and a $(n+1)$ bit variance $t1$ respectively, A $(n+1)$ -bit variance $t2$ is used to store $t0[n:0]$ at the end of each stage. $t0[n+1]$ isn't saved because it has already become zero at that time. When the next stage begins $t2$ is assigned to $t0[n+1:1]$,and $a[n-k]$ is assigned to $t0[0]$ .When the n th stage finishes , the value of $t3$ is the required square root. and $t2$ is the remainder.

III. Implementation of the Algorithm

When processing a $2n$ bit number, we need a n stage pipeline to realize the algorithm. The following variances are used in the design: n bit variance $t10$, n bit variances from $t30$ to $t3(n-2)$, n -bit variances from $t20$ to $t2(n-1)$, $(n+2)$ bit variances from $t01$ to $t0(n-1)$, the width of the variances from $r1$ to $r(n-1)$ which are used to store the unprocessed bits of the input radicand in each stage are from $(n-1)$ bit to 1bit in the order. The variances from $t00$ to $t0(n-1)$ are used to store the bits which should be considered in each stage of the partial remainder $num0$. Variances from $t10$ to $t1(n-1)$ store the correspondent bits of the increment $num1$. Variances from $t20$ to $t2(n-1)$ store the value of the variances from $t01$ to $t0(n-1)$ at the end of each stage respectively.. Variances from $t30$ to $t3(n-2)$ store the value of the partial square root calculated in each stage. The whole process is described below, while the flow chart is shown in Fig 1.

A. The First Stage

First assign $a[2n-1:n-1]$ to $t00$, store $a[n-2:0]$ in $r1$,and assign the value 2^{n-1} to $t10$ .Then judge whether $t00$ is no less than $t10$,if it is so then assign the value 2^{n-1} and the result of $t00-t01$ to $t30$ and $t20$ respectively, otherwise assign 0 to $t30$ .and $t00[n:0]$ to $t20$ directly.

B. The Second Stage

First assign $t20$ to $t01[n+1:1]$ and $r1[n-2]$ to $t01[0]$ and update $t11$ according to the formula $t11=2\times t30+2^{n-2}$. Then judge whether $t01$ is no less than $t11$,if it is so ,then update $t31$ and $t21$ according to the formula $t31=t30+2^{n-2}$ and $t21=t01-t11$ respectively. Otherwise assign $t30$ to $t31$ and $t20[n:0]$ to $t21$ directly. Finally assign $r1[n-3:0]$ to $r2$.

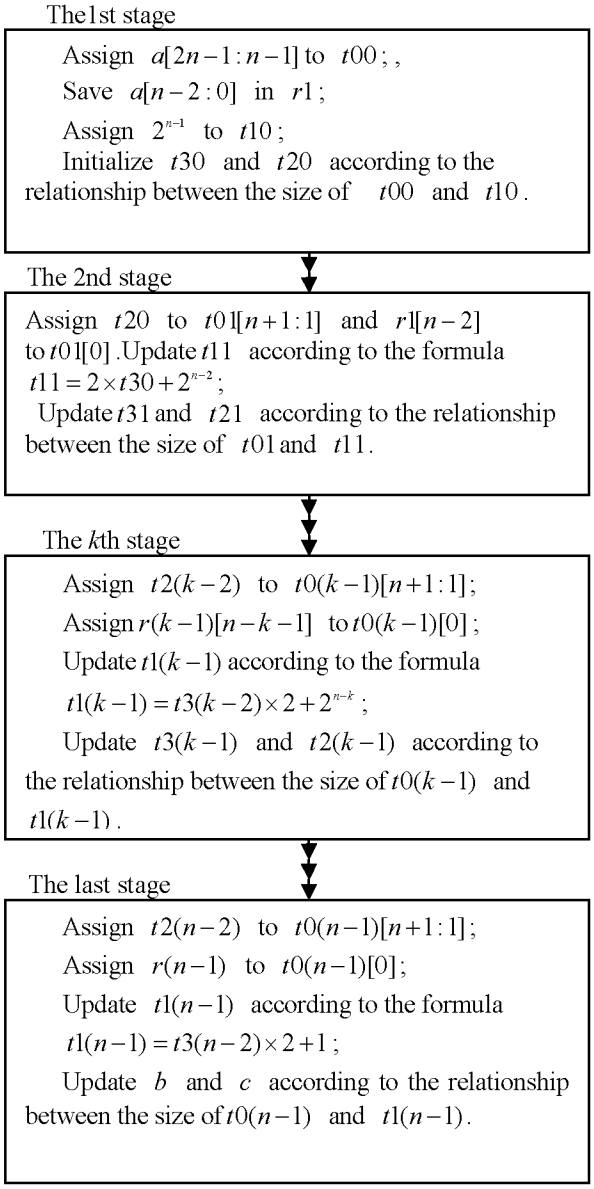

Figure 1. Flow chart of the process

C. The k-th Stage

First assign $t2(k-2)$ to $t0(k-1)[n+1:1]$ and $r(k-1)[n-k-1]$ to $t0(k-1)[0]$ and update $t1(k-1)$ according to the formula $t1(k-1)=t3(k-2)\times 2+2^{n-k}$ .Then judge whether $t0(k-1)$ is no less than $t1(k-1)$ .If it is so, then update $t3(k-1)$ and $t2(k-1)$ according to the formula $t3(k-1)=t3(k-2)+2^{n-k}$ and $t2(k-1)=t0(k-1)-t1(k-1)$ respectively. Otherwise assign $t3(k-2)$ to $t3(k-1)$ and $t0(k-1)[k:0]$ to $t2(k-1)$ directly. Finally assign $r(k)[n-1-k:0]$ to. $r(k+1)$.

D. The Last Stage(The nth Stage)

First assign $t2(n-2)$ to $t0(n-1)[n+1:1]$ and $r(n-1)$ to $t0(n-1)[0]$ and update $t1(n-1)$ according to the formula $t1(n-1)=t3(n-2)\times 2+1$ .Then judge whether $t0(n-1)$ is no less than $t1(n-1)$,if it is so then update the final square root b and remainder c according to the formula $b=t3(n-2)+1$ and $c=t0(n-1)-t1(n-1)$ respectively. Otherwise assign $t3(n-2)$ and $t0(n-1)[n:0]$ to b and c directly.

IV. Simulation Result Analysis

The algorithm described above can be applied to design any $2n$ bit square root calculators. A 8bit and 16bit square root calculators are designed in this paper. The source code is written in Verilog HDL language and then complied and simulated on the integrated CPLD/FPGA development platform Quartus2. EP1C3T100C6 is chosen as the target device for both designs The simulation results of the 8bit and 16bit square root calculators are shown in Fig 2, Fig3 and Fig 4 respectively.

From Fig.3 we can see that when processing 16-bit binary integers, the root and remainder of the first input data are output during the 8^{tth} clock cycle after which the calculator output a result every clock cycle. The square root and the remainder of the k-th (k =1,2, $\cdots\cdots$) input data will be output during the ($k+7$)th clock cycle. In one test a group of data $2^{2m}-1$ are input where m is set to 8, 7, 6, 5, 4, 3, 2, 1 during the 1st.2nd ,3rd ,4th ,5th ,6th 7th 8th clock cycle respectively. From the waveform in Fig.4 we can see that the output square root is $2^{m}-1$ and the remainder is $2^{m+1}-2$ .Since $2^{2m}-1=(2^{m}-1)^{2}+(2^{m+1}-2)$, the result is perfectly right .

V. The Performance of the Algorithm

A. Resource Consumed

The compilation report shows that the 8bit square root calculator only consumes 39 logic elements, while the 16bit calculator consumes 160 logic elements, Every time the width of the data the calculator can process increases 2,the resource it consumes will increase by once .The main reason for this is that a large number of registers have to be added between two adjacent stages to restore the intermediate results[5]

B. The Highest Work Frequency of the System

The timing analysis report shows that the highest work frequency of the 8bit square root calculator is 260Mhz,while the 16bit calculator is 202MHz. In contract when Spartan II XC2S150 is selected as the target device the highest work frequency of the 8bit and16bit calculators based on the algorithm proposed in[6] can only reach 143MHz and 134MHz respectively

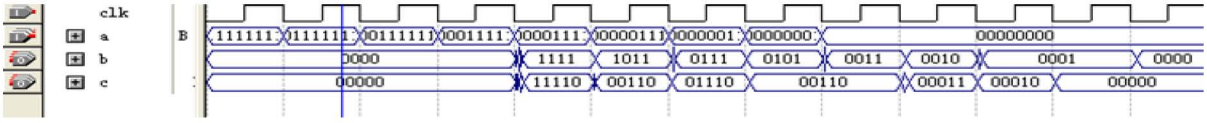

Figure 2. Simulation waveform of the 8bit square root calculator

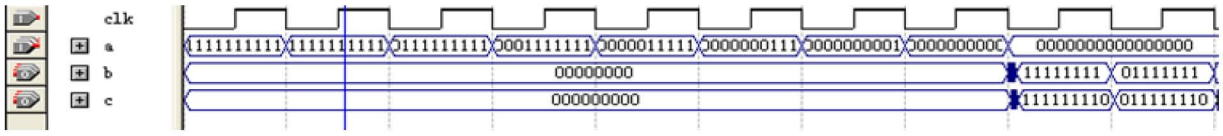

Figure 3. Simulation waveform 1of the 16bit square root calculator

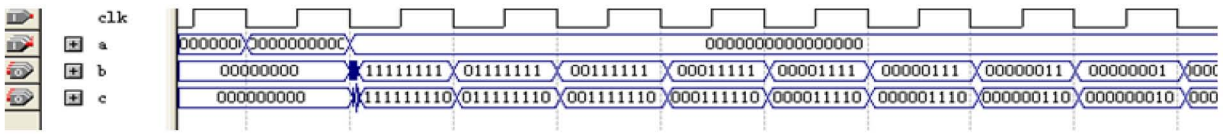

Figure 4. Simulation waveform 2 of the 16bit square root calculator

VI. CONCLUSION

A new square root calculation algorithm of binary integer numbers is proposed which can be implemented more easily on general FPGA with the pipeline technology compared with other algorithm. It has the following advantages: easy to implement ,less resource-consume and high processing speed. Therefore it is an effective algorithm for the implementation of square root calculators with pipeline technology on general FPGA chips.

ACKNOWLEDGMENT

The authors gratefully acknowledge the financial support from the National Natural Science Youth Foundation of China (200802025), Key Special planning Topic by the Higher Education Society of China (08CX06068).

REFERENCES

[1] Li Yamin,Chu Wanming,"Parallel-Array Implementations of a Non-restoring Square Root Algorithm," Computer Design: VLSI in Computers and Processors ,1997,pp:690 - 695

[2] Wanming Chu and Yamin Li.,"Cost/Performance/Tradeoff of N-select Square Root Implementations,"Field-Programmable Custom Computing Machines,11th Annual IEEE Symposium(FCCM2003),pp195 – 203.

[3] Yamin Li and Wanming Chu,"Implementation of Single Precision Floating Point Square Root on FPGAs,"FPGAs for Custom Computing Machines,The 5th Annual IEEE Symposium ,1997,pp226 – 232.

[4] Anuja J. Thakkar, Abdel Ejnioui,"Design and Implementation of Double Precision Floating Point Division and Square Root on FPGAs,"Aerospace Conference, 2006,pp.7.

[5] Wang Jinmiong, Digital System Design & Verilog HDL, 2nd ed, Beijing: Publishing House of Electronics Industry,2006; pp264-267.

[6] Lin Zhimou, Lu Guizhu,"A Square Root Algorithm Based on FPGA,". Journal of Xiamen University(natural science edition),.2006, Vol.45 ,No. 2,pp199-201