The Design and Realization of a Divider with Adjustable Precision in Fingerprint Enhancement

Weina Zhou*† ,Shuai Wang* and Xiaoyang Zeng*
*State Key Lab of ASIC &System, Fudan University, Shanghai
†Shanghai Maritime University, Shanghai
E-mail: 081021041@fudan.edu.cn Tel: +86-021-51355322

***Abstract*— The normalization of the fingerprint is an essential and important step in fingerprint enhancement, and its implementation needs some high speed dividers of different lengths and precisions. However, traditional division algorithm used many subtractions to get the result, and fix-point operations should also be implemented when either the divisor or the dividend is decimal, which both make the computation time consuming. A modified divider with adjustable precision and length based on Verilog is proposed in this paper. It can process both integer and decimal with any given precision conveniently, and its pipeline structure is also very useful to execute serials output of fingerprint data, which is very attractive to the normalization of the fingerprint. The paper described its operation and the data structure in detail, and the simulation result indicates that the velocity of the divider is quicker than other dividers, and it decreases the computation times efficiently.**

I. INTRODUCTION

As a fundamental module of electronic technology field, divider has been broadly applied in many kinds of electronic and circuit design. Today, with the development of microelectronic technology and VLSI, it even becomes an indispensable component in ICs, FPGA and other programmable logic devices' designs, and is one of the most resource consumption parts in them.

In intelligent recognition fields, like fingerprint enhancement, divider is also a fundamental and important component. In fingerprint normalization, one of its five main steps of fingerprint enhancement [1], dividers with different lengths and precisions are needed in different procedure of the whole realization. If these dividers are implemented by traditional method, all the dividends and divisors should be fix-pointed respectively and their speed will be limited, thus greatly affect the efficiency of normalization. In that case, a high speed and precision adjustable divider is necessary in fingerprint normalization.

Generally, there are two kinds of methods to realize the divider: hardware and software. Compared to software, divider realized in hardware has the advantage of high speed, but it still consumes huge power and area. Optimizing the algorithm of divider is an efficient method to improve it, for they can increase the work frequency of the operation, improve the flexibility of the design, and promote the design performance as a whole. Some researches have been done in these years for finding the efficient and practical algorithm.

Minus [2] 、SRT algorithm [3-4] are commonly traditional divider algorithms, but their speed is still not very high. In papers [5-6], improved algorithms have been proposed to modify the minus algorithms, and they greatly decreased the working cycles. But when either the dividend or the divisor is decimal, it is need to change the decimal to an integer by a fixed-point implementation at advance, which is very troublesome and not mentioned in detail by these papers unfortunately. Another widespread divide algorithm is SRT algorithm. SRT is a linear convergent algorithm with high performance, and is able to get several bits of quotient by executing elementary operation, and it can even get the exact quotient and remainder at the same time. The main operation of SRT is to realize the minus circularly, so the efficient approach to improve its speed is to decrease its circular times. Increasing the bit number processed in an elementary operation can greatly raise its speed, but which will increase the complexity of the system as well. In this paper, we proposed a modified divider algorithm, which can directly process the decimal at any desired precision as easy as integer, and compared to [5-6], it needs fewer clock cycles. And its work frequency reaches 242.1MHz, which is higher than the divider with SRT algorithm proposed in paper [4], although its architecture is much simpler. In addition, its pipeline architecture is also very practical in serially outputting the normalized fingerprint data.

The outline of the paper is arranged as follows: section II describes the proposed divider and its architecture. While section III introduces the application of it in the normalization module, section IV gives out the experimental results, and compares it with other methods describe in papers [4-5]. Then the last section makes a conclusion.

II. THE PROPOSED DEVIDER

We introduce the new divider from three sides: the initialization of the dividend and the divisor, the divide procedure and its structure.

This paper is supported by shanghai leading academic discipline project (S30602), innovation program of shanghai municipal education commission (09ZZ164/ 09YZ247), and science & technology program of shanghai maritime (20090131).

A. The Initialization of the Dividend and the Divisor

The precision of the proposed divider can be adjusted easily, and it is realized by simple preprocess of changing the bits width of the dividend and divisor instead of fix point operation. The preprocessing, which we can also call as initialization, can be divided into three steps. First, determine the bit numbers of the integer and decimal part of the quotient, thus fixed the precision of the result. Second, enlarge the dividend by 2^n, n is the sum of the bit numbers of the decimal part of divisor and quotient. Third, enlarged the divisor by 2^m, m is the sum of the bit numbers of the quotient (including integer and decimal) and the decimal part of the divisor. The procedure can be explained by the example shown as below:

$$\text{aaaa / bb = cc.cccc} \quad (1)$$

"a", "b", "c" represent a bit of the dividend, the divisor and the quotient respectively. They can either be "1" or "0", and the same character in different position can represent different digits in practice. In this example, the divider is supposed as an integer of 4 bits, divisor as an integer of 2 bits, and the decimal bit numbers of both of them are zeros. The bit numbers of the integer and decimal part of the quotient are determined as 2 and 4 as shown in formula (1). So, in the preprocess, the dividend will be enlarged by 2^4, because there are 4 bits in the decimal part of the divisor and the quotient (0 bit in divisor and 4 bits in quotient). And the divisor will be enlarged by 2^6 (the sum of the bit numbers of the quotient and the decimal part of the divisor). So after initialization, the dividend is aaaa0000 and divisor is bbbb000000.

When the dividend or the divisor is not integer, they can be processed as the same. As the example shown in formula (2), the divisor has a two-bit decimal, and after initialization, the dividend will be aaaa000000, and the divisor will be ccdd000000.

$$\text{aaaa / cc.dd = ee.ffff} \quad (2)$$

In this way, the fixed point implementation of the dividend and divisor can be replaced by simple bit adding operation.

B. The Divide Procedure

The whole divide procedure can be described as follows:

1. Right shift the divisor with one bit after initializing the dividend and the divisor.
2. Compare the dividend with the divisor and right shift the divisor again. If the dividend is the bigger one, left shift the quotient and change the last bit of the quotient to digit "1", and replace the dividend with the difference getting by subtracting the divisor from dividend. In other case, left shift the quotient and change the last bit of the quotient as "0" only.
3. Repeat the step 2 until divisor changes to the 2^K times of initial divisor. K is the decimal's bit number of initial divisor. Then we will finally get the quotient of demanded precision.

Taking 128/11 as an example, and the procedure can be described in Table I.

The bit numbers of the integer and decimal part of the quotient are set as 4 and 7.

TABLE I
THE PROCEDURE OF THE DIVIDER

	Dividend	Divisor	Quotient
	1000,0000	1011	aaaa.aaaaaaa
initilize	100,0000,0000,0000	101,1000,0000,0000	
step1	100,0000,0000,0000	10,1100,0000,0000	1
step2	1,0100,0000,0000	1,0110,0000,0000	0
step3	1,0100,0000,0000	1011,0000,0000	1
step4	1001,0000,0000	101,1000,0000	1
step5	11,1000,0000	10,1100,0000	1
step6	1100,0000	1,0110,0000	0
step7	1100,0000	1011,0000	1
step8	1,0000	101,1000	0
step9	1,0000	10,1100	0
step10	1,0000	1,0110	0
step11	1,0000	1011	1
result			1011.1010001

Here, "aaaa.aaaaaaa" represents the bit numbers of the integer and decimal of the determined quotient are 4 and 7. The divisor will right shift a bit every step until it is the same as it initially is. And the dividend will be minus by divisor in every step if it is bigger than the divisor, or it will hold the same. The bit of the quotient is also determined by the comparison of the dividend and the divisor, it will be 1 when the dividend is bigger than the divisor, or vice versa. And the final result is obtained by serially joining these bits in order.

Carefully analysis the procedure, we will notice that the cycle number the procedure spent lies on the bit numbers of the quotient. the more bit numbers of the quotient the more clock cycle it needs, for we will get a bit of the quotient each clock. However, in other papers like [5-6], they usually need 1-10 clocks to get a digit in decimal number.

C. The Data Structure of the Divider

The data structure of the divider is shown in Fig.1. At the beginning, dividend_in and divisor_in are the dividend and divisor after initialization respectively and the result_in is set as 0. The shifter_right can right shift the divisor by one bit every clock, and then compare it with the dividend_in. The result will determine whether to do the minus between them and whether the quotient should be left shift by 1 or 0.

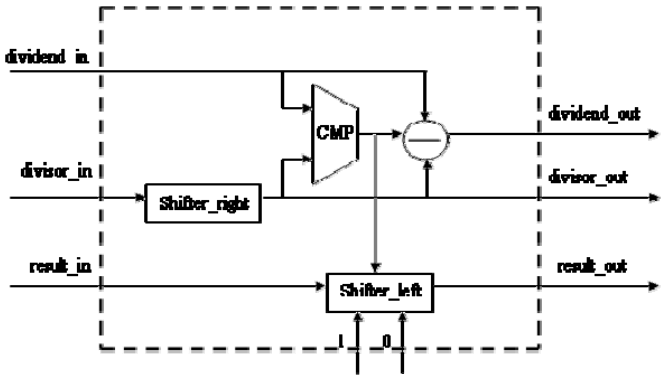

Fig. 1 Data structure of the divide module

Because the operation will repeat in every clock, the dividend_out, divisor_out, and result_out will then output to the dividend_in, divisor_in and the result_in to do the procedure again. And after n clocks (n is the bit number of the quotient), the result_out will output the final result we need. In addition, the subtraction in our structure is realized by a carry-look-ahead synthesis model which makes the operation much quicker.

III. THE APPLICATION OF DIVIDER IN NORMALIZATION

In fingerprint normalization, the normalized image is calculated by formula (3)[1] after getting the mean and variance of the input fingerprint image at advance:

$$N(i,j)=\begin{cases} M_0+\sqrt{\dfrac{VAR_0(I(i,j)-M)^2}{VAR}} & I(i,j)>M \\ M_0-\sqrt{\dfrac{VAR_0(I(i,j)-M)^2}{VAR}} & otherwise \end{cases} \quad (3)$$

In formula (3), $N(i, j)$ represent the normalized gray-level value of pixel in the position (i, j), $I(i, j)$ denotes the gray-level value of pixel (i, j), M and VAR denote the mean and variance of Image I respectively, and M_0 and VAR_0 are respectively the predefined mean and variance values. In fact, formula (3) can be further simplified to formula (4), which could be much clearer about the operation happened in the normalization.

$$N(i,j)=M_0+(I(i,j)-M)*\sqrt{VAR_0}*\frac{1}{\sqrt{VAR}} \quad (4)$$

From the above formulate, we can see that the critical operations of normalization are the divide and square root computation which could also be implemented by several divide operations of different precisions. So divider with adjustable precision is very useful here.

A. The implementation of square root operation

As we know, the principle of the square root operation is just the divide operation, and its precision can be approached by dividers of different precisions.

$$m'=(m+a/m)/2 \quad (5)$$

Formula (5) shows the computation of the square root operation. Here, "a" is the value to be calculated, m is the coarse value of the square root, and m' is the more precise square root. The precision of the square root is just promoted by using several above operations which needs several dividers of different lengths and precisions. We will explain the procedure by computing the square root of number "5" as an example.

$$(2+5/2)/2=2.25 \quad (6)$$

$$(2.25+5/2.25)/2=2.236 \quad (7)$$

$$(2.236+5/2.236)/2=2.2360679 \quad (8)$$

Before the divide operation, we should find an integer which is smaller but mostly approaches $\sqrt{a}$. It is 2 in this example. From formula (6) to (8), the precision we defined is higher and higher, and the adjustable dividers we proposed above can just complete it conveniently. What's more, the final result obtained by (8) also proves its efficiency, for it is very close to the real value.

B. The pipeline structure of the divider

The fingerprint normalization belongs to point operation in image processing, which means that every pixel is implemented by the same normalization operation independently. To increase the efficiency, pipeline structure is very suitable. And the proposed dividers could also be implemented in pipeline structure, which is very useful for the continuous output of the normalized result when the fingerprint data are serially inputted. And after several clocks' delay, we can get the result of a divider every one clock. The data structure of the pipeline divider module is shown in Fig.2.

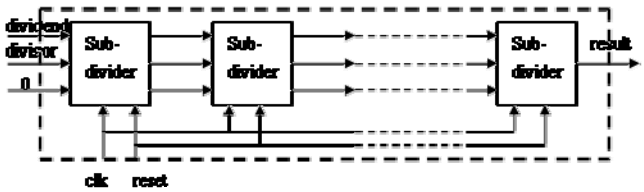

Fig.2 data structure of the pipeline divider module

IV. THE EXPERIMENTAL RESULTS

We realized the divider proposed above in hardware design language - Verilog, and simulated it in Modsim5.5. To prove its efficiency, we chose two typical impactful algorithms proposed in paper [4] and paper [5] to compare with it. So, in this section, two experiments are designed for the test and the same test data are used in them.

A. Compare by the Clock Cycle

A divider with adjustable precision is also proposed in paper [5]. To compare, we used the same dividend (128) and divisor (11) as in paper [5], and the decimal bit number of the quotient is set as 7 in consider that the result precision is centesimal. The divide procedure has been shown in Table I, and the result is obtained as 1011.1010001 after 11 clocks, which is very close to the real value 11.63636…. However, paper [5] needs 12 clocks to complete the computation. The simulation wave is shown in Fig.3.

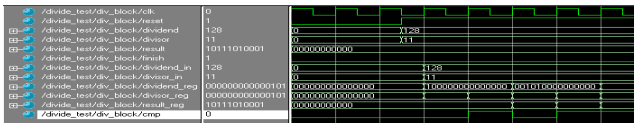

Fig. 3 The simulation waveform with precision of 7.

Although it seems that, there is not much difference between our algorithm and the algorithm proposed in paper [5], after analysis it carefully, we will find out that, in paper [5] the clock cycles it needs is dependant on the value of the quotient, and in our method, it lies on the length of the quotient. As we know, the biggest number that n bits can represent is 2^n-1, and which is always bigger than n except when n=1. So when the quotient is a big value, the number of clock cycles consumed will much more than the cycles we use in the proposed algorithm.

B. Compare by the Speed

The divider is synthesized using Synopsys' design compiler targeting SMIC 0.13um CMOS techniques. With 32-bits-long dividend and divisor, the speed of our divider reaches 242.1MHz, and the area of the divider cost in our method is about 1.85k gates, which is relative small. Comparing with the paper [4], which proposed a modified SRT algorithm with double speed of radix-2, our divider is even much quicker. What's more, its area is also much smaller than other divider with SRT algorithm, although the speed may be smaller than dividers of higher radix.

V. CONCLUSIONS

The paper proposed a divider with adjustable precision, and in it, fixed-point implementation has been replaced by simple shift operation, which makes the decimal be processed as easy as the integer. The divider is very efficient in the normalization procedure of the fingerprint enhancement, and the experiment result shows that it requires relative fewer cycles and has a higher speed compared with other methods.

REFERENCES

[1] L. Hong, Y. F. Wan and A. Jain, "Fingerprint image enhancement: algorithm and performance evaluation", IEEE Transactions on Pattern Analysis and Machine Intelligence, VOL.20, No.8, Aug. 1998

[2] Y.D. Chen, J.L. Qi and J.S.Chen, "The design of 8 division with VHDL", Control & Automatic, vol.22, no.13, pp. 277-278, 2006.

[3] D. Hua, "Study on SRT divider and its algorithm," Computer Engineering and Design, vol. 28, no. 1, pp. 248-249, Jan. 2007.

[4] H. Y. Liu, "Simulating and implementation of high performance division," Instrument Technique and Sensor, Vol.6, pp.38-39,70, Jan. 2006

[5] X. Y. Ye, H. Y. Zhang, D. J. Pi, S. J. Qin, "Design of integer divider with adjustable precision based on Verilog", Modern Electronics Technique, Vol.32, No. 3, pp.146-147, 2009.

[6] W. H. Zhou, "Design of the decimal integer divider per-setting up precision in calculation based on VHDL", Foreign Electronic Measurement Technology, Vol.27, No. 2, pp. 16-18, Feb. 2008.

[7] Takagi, N., Kadowaki, S., and Takagi, K," A hardware algorithm for integer division", 17th IEEE Symposium on Computer Arithmetic, pp.140 – 146, Jun. 2005

[8] Juang, T.-B, S. H. Chen, and S.M. Li, "A novel VLSI iterative divider architecture for fast quotient generation", IEEE International Symposium on Circuits and Systems, pp.3358 – 3361, May 2005

[9] X.S.HUANG., Y.E. Qing, Y.L. QIU, "High Speed Divider and ASIC Implementation", Microelectronics & Computer, Febrary, 2008