POZNAN UNIVERSITY OF TECHNOLOGY ACADEMIC JOURNALS
No 76 Electrical Engineering 2013

Robert SMYK*
Zenon ULMAN*
Maciej CZYŻAK*

PIPELINED DIVISION OF SIGNED NUMBERS WITH THE USE OF RESIDUE ARITHMETIC FOR SMALL NUMBER RANGE WITH THE PROGRAMMABLE GATE ARRAY

In this work an architecture of the pipelined signed residue divider for the small number range is presented. Its operation is based on reciprocal calculation and multiplication by the dividend. The divisor in the signed binary form is used to compute the approximated reciprocal in the residue form by the table look-up. In order to limit the look-up table address an algoritm based on segmentation of the divisor into two segments is used. The approximate reciprocal transformed to residue representation with the proper sign is stored in look-up tables. During operation it is multiplied by the dividend in the residue form and subsequently scaled. The pipelined realization of the divider in the *FPGA* environment is also shown.

1. INTRODUCTION

In the digital signal processing the division is usually performed when the quotient of two signals has to be determined, for example, in computation of the phase shift before arctangent calculation. The residue arithmetic [1, 2, 3] is a tool that can be used for realization of *DSP* algorithms due to its decompositional properties with respect to addition, subtraction and especially to multiplication because multiplication in one large integer ring can be replaced by a set of multiplications performed in small integer rings in parallel. The other important feature is the possibility to decompose the complex multiplication of Gaussian numbers in similar manner as that for integers using derivative systems such as the *MQRNS* (Modified Quadratic Residue Number System) [4]. However, other operations in residue arithmetic such as reverse conversion, sign detection, magnitude comparison, scaling and division are difficult. The division of residue numbers can be carried out by converting them to a weighted system, performing division and converting back to the residue form. However, division of residue numbers partly or fully in residue arithmetic can be more effective. The algorithms

* Gdańsk University of Technology.

of residue division belong mainly to a group of subtractive [5, 6, 7] or multiplicative [8, 9, 10] algorithms. The multiplicative algorithms compute, using the Mixed-Radix Conversion (*MRC*) [1], the reciprocal of the divisor which is subsequently multiplied by the dividend. Also two algorithms were presented [11, 12], where the *MRC*, sign detection, overflow detection are not needed but in the former the conversion of the divisor and dividend to the binary system is necessary. They have better time-hardware complexity, however, they are iterative what makes them not suitable for pipelined processing. The algorithm based on iterative reciprocal computation was given in [13]. In this work an architecture of the pipelined residue divider of signed number is shown. The implementation uses a non-iterative residue multiplicative division algorithm. The approximate reciprocal of the divisor is computed by the look-up with the use of the algorithm from [14] based on segmentation of the divisor in the binary form in two segments that address the look-up tables. In this way the size of look-up tables is reduced. The algorithm has fixed division time. It is assumed that the architecture will use 6-bit look-up tables available in the Xilinx *FPGA*. The algorithm permits to implement the division for signed 12-bit numbers with the maximum division error smaller than 2.

2. THE RESIDUE NUMBER SYSTEM(*RNS*)

The *RNS* with the base $B=\{m_1,m_2,..,m_p\}$, where m_j, j=1,2,..,p, are named moduli and the number range $M=\prod_{j=1}^{p} m_j$, allows to represent the nonnegative integer N from $[0,M\text{-}1]$ by the digit vector $\left(|N|_{m_1},|N|_{m_2},...,|N|_{m_p}\right)=\left(n_1,n_2....,n_p\right)$, where $|N|_{m_j}$ is the least nonnegative residue from division of N by m_j, $j=1,2,3,...,p$. This representation is one-to-one correspondence if the moduli are pairwise relatively prime. In such a case there is a unique mapping given by the Chinese Remainder Theorem [1-2]. The main advantage of the *RNS* is due to the fact that addition, subtraction and multiplication of two *RNS* numbers can be performed independently on the corresponding pairs of residues. For the numbers with the sign denoted as X, if M is even, $X=N$ for $N<M/2$, and $X=N-M$, if $N\geq M/2$. If M is odd, $X=N$ for $N<(M-1)/2$, and $X=N-M$, if $N\geq(M-1)/2$. As the multiplication of signed numbers is used in the method of division presented below, we shall illustrate it with an example.

Example 1. Multiplication of signed numbers in residue arithmetic.
Let B={32,31,29,27,25,23), we have M=446623200 and let

$$z_1 = 35 \leftrightarrow \{3, 4, 6, 8, 10, 12\}$$

$$z_2 = -70 \rightarrow M - 70 = 446623130 \leftrightarrow \{26, 23, 17, 11, 5, 22\}$$

We want to obtain the product $P = z_1 \cdot z_2$ .By performing the multiplications in the individual rings we obtain the residues of P

$$P = \left(\left|3 \cdot 26\right|_{32}, \left|4 \cdot 23\right|_{31}, \left|6 \cdot 17\right|_{29}, \left|8 \cdot 11\right|_{27}, \left|10 \cdot 5\right|_{25}, \left|12 \cdot 22\right|_{23}\right) = (14,30,15,7,0,11)$$

These residues are the residues of the number M-2450=46620750, that represents the product P in the M ring.

3. DIVISION ALGORITHM

In the residue division algorithm we have to find an integer $\widetilde{Q}$ that approximates $Q = X / Y$ with the maximum acceptable division error, $\varepsilon_{div}^{\max}$. The reciprocal of the divisor has to be determined with such accuracy that after multiplication by dividend X, the resulting division error is smaller than the assumed maximum acceptable error. The additional requirement imposed on the algorithm may be the use of small tables for the reciprocal computation. In the algorithm initially m-bit divisor ($m \le 12$) is decomposed into m-k bit segment and k-bit segment with k not exceeding 7 bits. For computation of the divisor by look-up such segmentation allows to use smaller look-up tables than in the case when the look-up table is addressed with the full representation of the divisor. The reciprocal R can be decomposed into two parts in the following manner

$$R = \frac{1}{Y} = \frac{1}{a+b} = \frac{1}{a} - \frac{b}{a \cdot (a+b)}. \tag{1}$$

The transformation of (1) into the form that allows to use small look-up tables was presented in [14] along the reciprocal computation algorithm. In the following a short review is provided. It is seen that the computation of $1/(a \cdot (a+b))$ requires m-bit address, in order to replace it with $\lceil \log_2 a \rceil$-bit address we may try to replace b by a suitably chosen constant, K that leads to

$$\frac{b}{a \cdot (a+b)} \cong \frac{b}{a \cdot (a+K)}, \tag{2}$$

and in effect we obtain the following reciprocal approximation

$$\widetilde{R} = \frac{1}{a} - \frac{b}{a \cdot (a+K)}, \tag{3}$$

where $a = \lfloor Y / 2^k \rfloor$ and $b = |Y|_{2^k}$. We see that $1/a$ can be computed by the look-up using m-k bits. For $a > 0$ we replace b by K that approximates b in $[0, 2^k - 1]$. Remark that for a=0, $1/b$ can be looked up using k-bits.

The reciprocal approximation error resulting from using K instead of b is expressed as [14]

$$R - \widetilde{R} = \varepsilon(a, b, K) = \frac{b \cdot (K - b)}{a \cdot (a + b) \cdot (a + K)}. \tag{4}$$

As K has to approximate b, it should belong to $[0, b_k]$, where $b_k = 2^k - 1$ is the end of the interval. It is evident that $\varepsilon(a, b, K)$ is maximal with respect to a when $a = a_{min}$ and with respect to b when $b = b_k$ or for certain $b = b_{\max}$. Using (4) and $a = a_{min}$, the maximum division error for the maximum dividend, $X_{\max}$ can be written as

$$\varepsilon_{div}^{max} = \varepsilon(a_{min}, b, K) \cdot X_{max}. \tag{5}$$

The extreme of (4) with respect to $b_{\max}$ is obtained as

$$b_{\max} = -a + \sqrt{a \cdot (a + K)} \tag{6}$$

Using this $b_{\max}$, we want to equalize the division error, for b_k and $b_{\max}$

$$-\varepsilon\ (a_{\min}, b_{\max}, K) = \varepsilon\ (a_{\min}, b_k, K) \tag{7}$$

(7) using (4) can be written in the following form

$$-\frac{b_{\max} \cdot (b_{\max} - K)}{a + b_{\max}} = \frac{b_k \cdot (b_k - K)}{a + b_k} \tag{8}$$

Inserting (6) into (8), we obtain the equation for K that allows to determine K that provides the fulfilment of (8)

$$(1 + \frac{4 \cdot b_k}{a_{\min}} + \frac{4 \cdot b_k^2}{a_{\min}^2}) \cdot K^2 + (\frac{2 \cdot b_k^2}{a_{\min}} - \frac{4 \cdot b_k^3}{a_{\min}^2} + 4 \cdot b_k) \cdot K - 4 \cdot b_k^2 - 4 \cdot \frac{b_k^3}{a_{\min}} + \frac{b_k^4}{a_{\min}^2} = 0 \tag{9}$$

Sample solutions of (9) are given in Example 2.

Example 2. Assume the length of the divisor Y equal to m=12 bits and the lengths of a and b equal to 6 bits. We have $a_{\min} = 64$ and $b_k = 63$. The coefficients of the quadratic equation (9) are $A = 8.81$, $B = 131.84$, $C = -27658.0$. Moreover, we have $b_{\max} = 21.05$ and the optimum $K = 49.07$. In effect we obtain for $X_{\max} = 2^{12} - 1$ and for $b_{\max}$, the maximum division error equal to $\varepsilon(K, b_{\max}) \cdot X_{\max} = 3.92$ and for b_k to $\varepsilon(K, b_k) \cdot X_{\max} = -3.92$.

In order to reduce this error we may increase the length of a to 7 bits and shorten b to 5 bits. We then have $a_{\min} = 128$ and $b_k = 31$. The coefficients of (9) are $A = 2.2$, $B = 131.74$, $C = -4718.6$, and the optimum $K = 25.21$ and $b_{\max} = 12.03$. In effect we have for $X_{\max}$, $\varepsilon(K, b_{\max}) \cdot X_{\max} = 0.236$ and $\varepsilon(K, b_k) \cdot X_{\max} = -0.236$.

4. HARDWARE REALIZATION

Now we shall consider the realization of the divider with the use of residue arithmetic. Such realization requires the transformation of the approximated reciprocal values to integers. This transformation is done by the multiplication by a constant K_s and rounding off the result. After transforming of (3) to integers we get

$$\left\{\frac{1}{Y}\cdot K_s\right\} \cong \left\{\frac{1}{a}\cdot K_s\right\} - b\cdot\left\{\frac{1}{a\cdot(a+K)}\cdot K_s\right\} \tag{10}$$

where $\{\cdot\}$ denotes rounding off to nearest integer. K_s in (10) should give the appropriate dynamic number range to represent the both terms and provide for the allowable error value that arises after multiplication of the round-off error of the second term by b. The maximum value of this error should not cause the unacceptable division error. The upper bound of this error is reached for $b_{\max}$ and $X_{\max}$ and maximal value of a, $a_{\max c}$ for which the compensation of the reciprocal approximation error is still needed. It is easy to verify that for the considered number range of division of 2^{12}, we have $a_{\max c} = 2^8$. The error of the second term of (10) has to fulfill the following condition

$$b_{\max}\cdot\left(\frac{1}{a_{\max c}\cdot(a_{\max c}+K)}\cdot K_s - \left\{\frac{1}{a_{\max c}\cdot(a_{\max c}+K)}\cdot K_s\right\}\right) < 0.5\cdot K_s. \tag{11}$$

Representing the second term in (11) as $1/(a\cdot(a+K)) + \varepsilon_r$ where ε_r is the rounding error, we obtain the bound on ε_r to limit the error of transformation to integers to 0.5

$$\varepsilon_r < \frac{0.5\cdot K_s}{b_{max}\cdot X_{max}}, \tag{12}$$

moreover, K_s has to fulfill the condition

$$K_s > 2\cdot a_{\max c}\cdot(a_{\max c}+K), \tag{13}$$

For example, for $a_{\max c} = 256$ we have $K_s > 156195.84$, that gives after inserting into (12) $\varepsilon_r < 0.3072$. We may avoid the round-off error, by assumming K_s as the multiple of the $a_{\min} \cdot (a_{\min} + K) = 7236.48$, for example, 159202.56, that slighly extends the error bound. However, there can be additional requirements imposed on K_s, because certain values may facilitate the design of the scaling circuit that performs scaling after division.

Example 3. Realization of division for three divisor values 127, 191 and 319 with *a*=64, 128, 256, respectively, and maximum of *b*=63, for which the highest level of error compensation is needed and the error due to round-off of the second term in (10) may reach its maximum.

First we shall we consider $Y_1 = 127$. We have *a*=64 and *b*=63.
$a \cdot (a + K) = 64 \cdot (64 + 49.07) = 7236.48$, and we will adopt $K_s = 159712$.
Such choice of $K_s = 32 \cdot 29 \cdot 23 \cdot 7$ results from the requirements of scaling after division, scaling becomes more simple when the scaling factor is a product of the moduli of the *RNS* base.

Using (10) we get

$$\left\{\frac{1}{Y_1} \cdot K_s\right\} = \left\{\frac{159712}{64}\right\} - 63 \cdot \left\{\frac{159712}{7236.48}\right\} = \{2495.5\} - 63 \cdot \{22.07\} = 1110$$

We obtain the approximate quotient as

$$\widetilde{Q}_1 = \left\{\frac{1}{Y_1} \cdot K_s\right\} \cdot X_{\max} / K_s = \frac{1110 \cdot 4095}{159712} = \{28.46\} = 28$$

whereas $Q_1 = \dfrac{X_{\max}}{Y_1} = 32.34$.

In the second case we shall consider *a* in the middle of its interval.
$Y_2 = 193$. We have *a*=128 and *b*=63.
Here $a \cdot (a + K) = 128 \cdot (128 + 49.07) = 22664.96$ .We get

$$\left\{\frac{1}{Y_2} \cdot K_s\right\} = \left\{\frac{159712}{128}\right\} - 63 \cdot \left\{\frac{159712}{22664.96}\right\} = \{1247.5\} - 63 \cdot \{7.04\} = 807$$

We obtain the approximate quotient as

$$\widetilde{Q}_2 = \left\{\left\{\frac{1}{Y_2} \cdot K_s\right\} \cdot X_{\max} / K_s\right\} = \left\{\frac{807 \cdot 4095}{159712}\right\} = \{20.69\} = 21$$

whereas $Q_2 = \frac{X_{\max}}{Y_2} = 21.39$.

Finally we consider the division for $Y_3 = 319$, where *a*=256, that means that it reaches the end point of interval in which the reciprocal approximation error is compensated. We have *a*=256 and *b*=63. Here
$a \cdot (a + K) = 256 \cdot (256 + 49.07) = 78097.92$.

$$\left\{\frac{1}{Y_3} \cdot K_s\right\} = \left\{\frac{159712}{256}\right\} - 63 \cdot \left\{\frac{159712}{78097.92}\right\} = \{623.87\} - 63 \cdot \{2.04\} = 498$$

We obtain the approximate quotient as

$$\widetilde{Q}_3 = \left\{\frac{1}{Y_3} \cdot K_s\right\} \cdot X_{\max} / K_s = \frac{498 \cdot 4095}{159712} = \{12.76\} = 13$$

whereas $Q = \frac{X_{\max}}{Y_3} = 12.83$.

We can estimate the required number range by (14).

$$M = \left(\left\lceil \frac{1}{a_{\min}} \cdot K_s \right\rceil - b_{\max} \cdot \left\lfloor \frac{K_s}{a_{\max c} \cdot (a_{\max c} + K)} \right\rfloor \right) \tag{14}$$

In our case we have $a_{\min} = 64$, $b_{max} = 63$, $K_s = 159712$ and $K = 49.03$. The dynamic range of the first term in (14) is equal to 2495.5 and of the second term 441. Finally we may estimate the require dynamic range as
$M_D = \lceil log_2(2495.5 \cdot 4095 \rceil = \lceil log_2 10219072.5 \rceil = 24$ bits.

We see that after scaling the binary size of quotient obtained from this residue channel will not exceed 7 bits.

The *RNS* base has been chosen as $B = \{32, 31, 29, 23, 21\}$ with *M*=13894944. and, given above, $K_s = 32 \cdot 31 \cdot 23 \cdot 7 = 159712$.

For the *RNS* architecture we assume that $-2048 \le X \le 2047$, and has the residue representation $X \leftrightarrow (x_1, x_2, x_3, x_4, x_5)$, where $x_j = |X|_{m_j}$, *j*=1,2,...,5. and Y is represented in 12-bit signed binary form.

$$\left|\widetilde{Q} \cdot K_s\right|_m = \left\| \left\{\frac{1}{a} \cdot K_s\right\} \right|_m - |b|_m \cdot \left(\left| \left\{ \frac{1}{a \cdot (a + K)} \cdot K_s \right\} \right|_{mm} \right)_m \tag{15}$$

In Fig.1 an architecture that implements (15) is depicted.

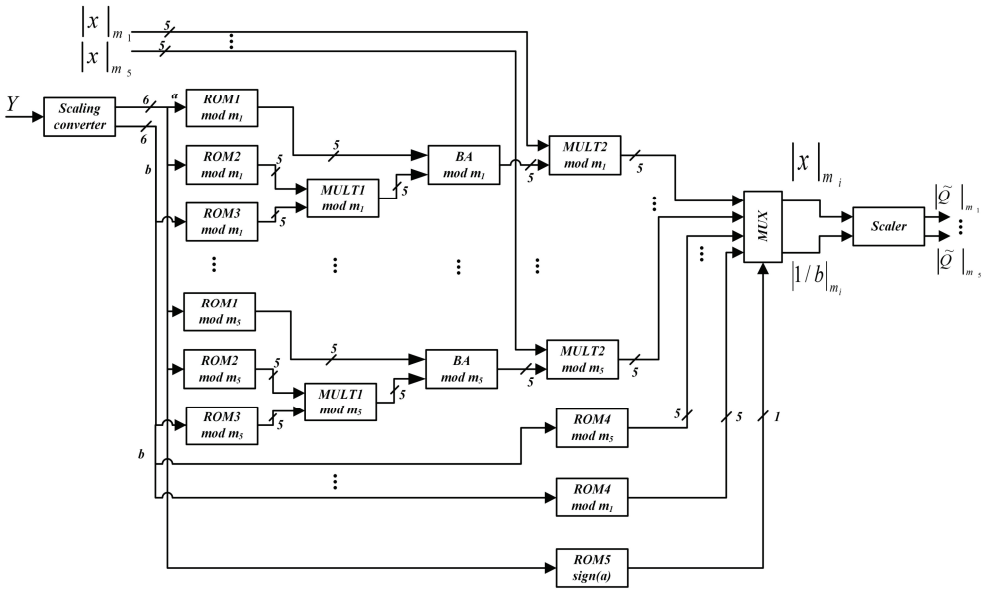

Fig. 1. The architecture of the residue divider

The dividend X is represented at the input as $\left(|X|_{m_1},|X|_{m2},...,|X|_{m_6}\right)$. The scaling converter scales Y to the range $[-2^{11},2^{11}-1]$ and outputs 12-bit binary word where the most significant bit is the sign bit, the next 5 bits form operand a and six least significant bits represent operand b. For each residue channel the same configuration of components are used. *ROM1* mod m_i compute $\left|\left\{\frac{k_s}{a}\right\}\right|_{m_i}$, *ROM2* mod m_i compute $\left|\left\{\frac{1}{a\cdot(a+K)}\cdot K_s\right\}\right|_{m_i}$ and *ROM3* mod m_i compute $|b|_{m_i}$,. In the next stage the multiplication is performed (*MULT1* mod m_i) and in the following stage the subtraction is performed (*BA* mod m_i). In the final stage the obtained residues are scaled by K_s. If a=0 the *ROM*4 mod m_i are applied that compute $\left|\left\{\frac{K_s}{b}\right\}\right|_{m_i}$ and *ROM*5 detects the sign. The outputs of these circuits are multiplexed with these obtained from (15). In this simplified divider architecture there is no divisor zero detection.

The architecture has been implemented in the Xilinx environment using the device from the Virtex-6 family. Below the synthesis report is shown. The pipelining rate of 2.74 ns has been attained. It is possible to obtain 1.52 ns that corresponds to 658.610 MHz. The pipelining rate is greater because of reduction of the number of pipeline stages.

Selected Device : 6vcx240tff784-2

Slice Logic Utilization:

Number of slice registers:	443 out of 301440
Number of slice LUTs:	908 out of 150720
Number used as logic:	834 out of 150720
Number used as memory:	74 out of 58400
Number used as SRL:	74

Timing Summary:

Minimum period: 2.747 ns (maximum frequency: 363.980MHz)
Minimum input arrival time before clock: 0.550ns
Maximum output required time after clock: 0.659ns

5. CONCLUSIONS

The paper presents the implementation of the pipelined residue divider for 12-bit number range in the Xilinx *FPGA* environment. The divider makes use of the multiplicative division algorithm with the two-term reciprocal approximation. The residue error belongs to $[-3.92,3.92]$, however for two's complement coding the error is halved. The divider architecture uses 5-bit moduli so that easy implementation is possible as in this environment 6-bit *LUT*s are available. The architecture use neither large memories nor multipliers.

REFERENCES

[1] Szabo N.S., Tanaka R.I.: *Residue Arithmetic and its Applications to Computer Technology*, McGraw-Hill, New York, 1967.
[2] Soderstrand M. et al., *Residue Number System Arithmetic, Modern Applications in Digital Signal Processing*, IEEE Press, NY, 1986.
[3] Omondi A., Premkumar B., *Residue Number Systems: Theory and Implementation*, London, Imperial College Press, 2007.
[4] Jenkins W.K., Krogmeier J.V.: The design of dual-mode complex signal processors based on quadratic modular number codes, IEEE Trans.on Circuits and Systems, Volume 34, Number 4, pp.354-364, 1987.

[5] Keir, Y.A, Cheney P.W., Tanenbaum M.: Division and overflow detection in residue number systems, IRE Trans. Electron. Comput., Volume EC-11, pp. 501-507, 1962.
[6] Kinoshita E., Kosako H., Koyima Y.: General division in symmetric residue number systems, IEEE Trans. on Computers, Volume C-22, pp.134-142, 1973.
[7] Banerji D.K., Cheung T.Y., Ganesan V.: A high speed division method in residue arithmetic, Proc. of 5th IEEE Symp.on Comput. Arithm., pp. 331-342, 1981.
[8] Lin, M. L., Leiss, E., McInnis B.: Division and sign detection algorithms for residue number systems, Comput. Math. Appl. Volume 10, Number4/5, pp. 331-342, 1984.
[9] Chren W.A., Jr.: A new residue number system division algorithm, Comput. Math. Appl., vol.19, Number 7, pp.13-29, 1990.
[10] Lu M, Chiang Jen-Shiun: A novel division algorithm for the residue number system, IEEE Trans. on Comput., Volume C-41, pp.1026-1032, 1992.
[11] Hiasat A.A., Zohdy,H.A.A.: Semi-custom VLSI design and implementation of a new efficient RNS division algorithm, Computer Journal, Volume 42, Number3, pp.232-240, 1999.
[12] Talameh S., Siy P.: Arithmetic division in RNS using Galois field *GF(p)*, Comput. Math. Appl., Volume 39, pp. 227-238, 2000.
[13] Hitz, M.A., Kaltofen, E: Integer division in residue number system, IEEE Trans. on Computers, Volume C-44, pp.983-989, 1995.
[14] Czyzak, M.: Noniterative small range residue division, RADIOELEKTRONIKA 2002, May 14-16, Bratislava, pp.111-114, 2002.