XXVII Seminarium

ZASTOSOWANIE KOMPUTERÓW W NAUCE I TECHNICE 2017

Oddział Gdański PTETiS

RNS/TCS CONVERTER DESIGN USING HIGH-LEVEL SYNTHESIS IN FPGA

Robert SMYK[1], **Maciej CZYŻAK**[2]

1. Politechnika Gdańska
   tel.: 583471332 e-mail: robert.smyk@pg.edu.pl
2. Politechnika Gdańska
   tel.: 583471502 e-mail: maciej.czyzak@pg.edu.pl

Abstract: An experimental high-level synthesis (HLS) of the residue number system (RNS) to two's-complement system (TCS) converter in the Vivado Xilinx FPGA environment is shown. The assumed approach makes use of the Chinese Remainder Theorem I (CRT I). The HLS simplifies and accelerates the design and implementation process, moreover the HLS synthesized architecture requires less hardware by about 20% but the operational frequency is smaller by 30% than that for the VHDL designed converter.

Keywords: Residue Number System (RNS), Two's-complement system (TCS), Chinese Remainder Theorem I (CRT I), FPGA.

1. INTRODUCTION

The Residue Number Systems (RNS)[1-3] are non-weighted number systems that allow to replace selected arithmetic operations in a large integer ring by a set of equivalent and independent operations in small integer rings. This pertains to addition, subtraction and multiplication which can be performed without carries between the digits of the number. Other operations such as conversion to weighted systems, scaling, division, sign determination, magnitude comparison are generally difficult. Thus the practical use of the RNS is limited to the areas where independent operations dominate.

In this paper we consider the synthesis of RNS/TCS converters. In the literature several residue-to-binary converters have been shown [4-9]. In this work we analyze the FPGA realization of the RNS/TCS high-speed converter based on the CRT I for the RNS system base consisting of five-bit moduli. The aim of the work was to examine the effectiveness of the HLS [10] when used for the converter design. The HLS, also known as the behavioural synthesis, is an approach which automatically translates the behavioural design description in C/C++ languages into register transfer level (RTL) description. The HLS approach considerably reduces the development time but the resulting architecture may require more hardware resources than these required for the VHDL or Verilog designs. The classical approaches to design are viable for less complex architectures whereas the HLS allows to speed up the design process of complex architectures. A considerable advantage when using of C/C++ for hardware description, is due to the fact that the straightforward description of complex algorithms becomes possible. The main task of translation of the algorithm into a hardware structure in the FPGA is performed by the synthesizer. Moreover, the testing procedures of the implementation can be conducted at the high level of abstraction.

We have developed the implementation of the RNS/TCS architecture based on CRT I using the HLS in the Xilinx Vivado environment and we have compared the synthesis results including the hardware size and delays with the previous VHDL implementation. In Section 2 we review the RNS, in Section 3 we present the RNS/TCS converter algorithm based on the CRT I, in Section 4 the converter design and analysis of its properties, and in Section 5 we present the HLS design of the converter.

2. THE RESIDUE NUMBER SYSTEM

The RNS is a non-positional number system in which numbers are represented by n-tuples consisting of numbers being the residues with respect to the members of the set called the RNS base, $B=\{m_1,m_2,...,m_n\}$ where m_i, $i=1,2,3,...,n$, are nonnegative integers termed the moduli. The number range M of the system is equal to $M=\prod_{i=1}^{n} m_i$. If the moduli are pairwise relatively prime, every integer from the number range M can be represented in the unique manner. The residue operations of addition, subtraction and multiplication can be performed independently on the residue digits of operands. The mapping from the RNS to a weighted system can be performed using the Chinese Remainder Theorem [1,3], mixed-radix conversion or the core function [8]. The more complete introduction to the RNS has been presented, for example, in [1-3].

3. CRT CONVERTER ALGORITHM

The aim of the algorithm is to convert the number given in the residue form, *i.e.* as a vector of residues into a weighted number system representation. The latter can be the representation in the natural binary system or two's complement system when signed numbers are processed. This conversion can be performed in many ways. The basic algorithm, already shown in [1], was the use of the mixed radix conversion (MRC). In its basic form it is a serial process but it does not require the cumbersome modulo M operation. The MRC has been considered by many authors, for example in [3]. There also exists the parallel MRC form given in [12]. However, it seems that the most general approach is the CRT [9]. However, its effectiveness depends

on the binary size of the RNS moduli and the efficient realization of modulo M reduction.

The value of a nonnegative integer N, with the use of the CRT, is given by the formula

$$N = \left| \sum_{j=1}^{n} N_j \right|_M , \quad (1)$$

with

$$N_j = M_j \cdot \left| M_j^{-1} \cdot |N|_{m_j} \right|_{m_j} , \quad (2)$$

$$M_j = M / m_j , \quad (3)$$

and

$$\left| M_j \cdot M_j^{-1} \right|_{m_j} = 1 . \quad (4)$$

M_j^{-1} is called the multiplicative inverse of M_j modulo m_j, and exists if $\gcd(M_j, m_j) = 1$, $j = 1,2,...,n$. For a signed number X, if M is even, $X = N$ for $N < M/2$, and $X = N - M$ for $N \geq M/2$. If M is odd, $X = N$ for $N < (M-1)/2$, and $X = N - M$ if $N \geq (M-1)/2$.

The realization of (1) requires in the first stage the computation of orthogonal projections by (2). N_j can be precomputed and stored in look-up tables addressed by $n_j = |N|_{m_j}$. This approach is practical when the binary size of the modulus m_j is relatively small and does not exceed 10 bits. This also determines the maximum binary size of moduli. Theoretically, m_j can have the greater binary size but the look-up table size grows exponentially making such RNS base less practical and makes the required hardware amount unacceptable. Moreover, the use of large memories reduces the attainable pipelining rate.

Once the orthogonal projections are obtained they have to be added and their sum has to be reduced modulo *M*. As the use of the tree of large two-operand binary adders is impractical the carry save addition is a viable alternative. The sum may have the dynamic range equal to $n \cdot (M-1)$ but it is known that when the sum is smaller than $2M$ modulo M reduction is easy because it requires only the subtraction of M and the choice of the difference or the input number. Hence it is advantageous to reduce the numbers at the output of the CSA tree to the interval $[0,2M)$ and then perform modulo M reduction. One technique to do this is to use the partitioning of the carry-save representation at the output at the carry-save adder tree and reducing modulo *M* the number represented by the high-order bits as shown in [11]. Carry and save representations are divided into the high and low order segment in such a manner that the sum of the low order segments is smaller than M. The numbers represented jointly by high order segments are reduced modulo M. The sum of two low-order segments and the residue modulo M of the number represented by the sum high order segments are added using the carry save adder and thus we receive two-operands with the sum belonging to $[0,2M)$. Then we perform the carry-save addition of their sum and $-M$. We get two operand pairs of which one is the correct sum and the other the negative number. Each pair is added in the parallel two-operand binary adders and at the output the nonnegative sum is selected using a multiplexer.

As the total number of high-order bits for larger *n* may require the memory of the size that excludes the high-speed pipelined operation, a modification based on the initial summing of the high-order bits at the output of the *CSA* using the small binary adder [9],[11]. This modification allows to limit the size of the memory used as the modulo M generator addressed by high-order bits. In this work an attempt is shown aiming at the pipelined realization of the converter architecture presented in [9]. Once the conversion from RNS to the binary system is performed, the sign of the RNS number has to be determined. The number *N* represents a negative number if it exceeds the half of the dynamic range (*M*/2 or (*M*-1)/2). So thus it is enough to perform subtraction *N-M/2* (N-(*M*-1)/2) and examine the difference. If the difference is negative the TCS number equal to *N-M*.

4. RNS/TCS CONVERTER DESIGN

The general architecture of the converter from [7] is shown in Fig. 1

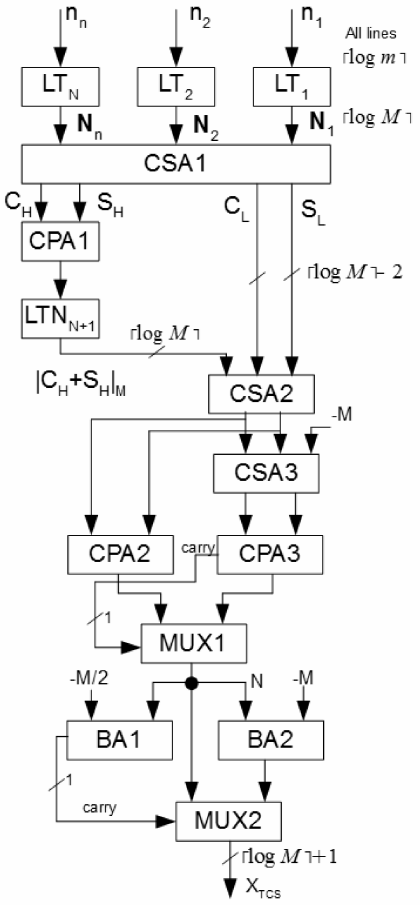

Fig. 1. RNS/TCS converter based on CRT I

We first shortly review the principle of converter operation. The orthogonal projections are calculated at the design stage and stored in LUTs. During converter operation they are obtained by the look-up and their sum modulo *M* is determined by using a multi-operand modulo *M* adder. The orthogonal projections are added in the *n*-operand carry-save adder tree (*CSA*1). The sum of projections can be expressed as $\left| \sum_{j=1}^{n} N_j \right|_M = C^{(1)} + S^{(1)}$. The partition of the CSA tree outputs can be represented as $C^{(1)} = C_H + C_L$ and

$S^{(1)} = S_H + S_L$, C_H and S_H are the numbers represented by the high-order bits of $C^{(1)}$ and $S^{(1)}$, and C_L, S_L by the low-order bits. The partitions are done so that $C_L + S_L < M$. This gives in effect the reduction of the sum to $[0,2M)$. The addition of MSB's, represented by $S^{(2)} = C_H + S_H$, is carried out in the small binary adder CPA1 and the modulo M reduction is performed by mapping $S^{(3)} = \left|S^{(2)}\right|_M$, with the use of the LT_{N+1} and subsequently the addition $S = S^{(3)} + C_L + S_L$ is carried out by the CSA2. Denote the carry and save representations received at the CSA2 output as N_1 and N_2 respectively. Next CSA3 is used to compute $N_1 + N_2 - M$. Finally, CSA2 and CSA3 output vectors are added in two parallel binary carry-propagate adders CPA2 and CPA3. If the CPA3 output sum is non negative, it is selected by the MUX as the correct result else the CPA2 output is chosen. In order to determine the sign of N we compute the difference $N-M/2$ ($N-(M-1)/2$) using the BA1. In parallel we perform subtraction $X_D = N - M$ (BA2). If the output sum of the BA1 is negative then X_D is selected else the original N is chosen.

5. RNS/TCS CONVERTER HLS DESIGN

The description of the converter structure in the HLS leads to the modified CRT converter presented in Fig. 2. Firstly the CSA1 adder is replaced by the multi-operand binary adder (mBA). As there is only one sum vector, it is divided into two segments S_H and S_L instead of four as in Fig 1. Also the final two-operand modulo M addition slightly modified.

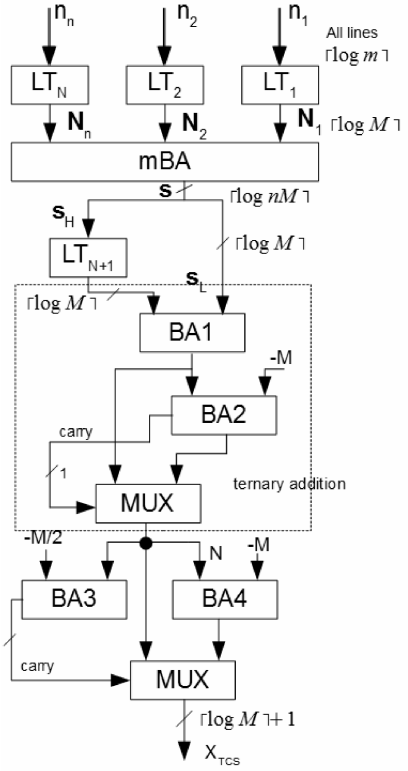

Fig. 2. Modified RNS/TCS converter based on CRT I for HLS

In the case of the HLS ternary addition is implemented using two binary adders in series that introduces the considerable delay. In order to reduce this delay, we can perform this addition by simulating in the HLS code the single CSA adder and apply two binary adders in parallel as in Fig. 3.

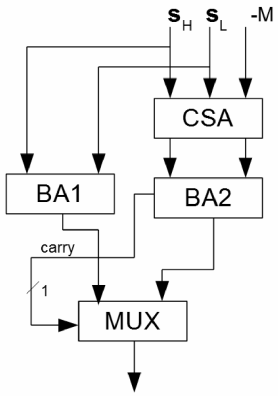

Fig. 3. Modulo reduction from [0,2*M*) to [0,*M*)

The C/C++ version used in the HLS design calls for several elements that are specific to the synthesis process. The most important element are integer types with a controllable binary length. They allow you to select the proper binary lengths of the input of the circuit, the intermediate results, and output. These integer types are implemented by using parametrized types as shown in Fig. 4. For example, the programmatic construction `typedef ap_int<36> int36` introduces the new signed integer type `int36` by using the generic type `ap_int`. This type represents integer variables of the binary lengths equal to 36 bits.

```
#include "ap_int.h"

typedef ap_int<36> int36;
typedef ap_uint<36> uint36;
typedef ap_uint<37> uint37;
typedef ap_uint<38> uint38;
typedef ap_int<38> int38;
typedef ap_int<39> int39;
typedef ap_uint<42> uint42;
typedef ap_int<42> int42;

typedef ap_uint<5> uint5;
typedef ap_uint<6> uint6;
```

Fig. 4. Arbitrary types used in the description of the structure and synthesis of the TCS/RNS converter

The implementation of the CRT I (1) requires in the first stage the computations of orthogonal projections (2) that depend only on residue n_j, the other factors in (2) are constants. In this case we assume the same RNS base as for the VHDL realization, *i.e.* $B = \{32, 31, 29, 27, 25, 23, 19, 17\}$ (Fig 5), that gives the RNS dynamic range of 37.07 bits. In the HLS realization we simulate look-up tables of the first layer in Fig. 2 by precomputing N_j and placing them as initial values of the array. The array for each modulus containing the projections will be addressed in the program using the respective residue n_j. The exemplary array of projections for m_1=17 is shown in Fig. 4. In the hardware structure the content of this array becomes the content of the look-up table addressed by the residues modulo m_1=17.

```
uint38 Pr_m1[17]= {
0,67886726400,135773452800,
59400885600,127287612000,50915044800,
118801771200,42429204000,110315930400,
33943363200,101830089600, 25457522400,
93344248800, 16971681600, 84858408000,
8485840800,76372567200};
```

Fig. 5. The exemplary array of projections for the modulus m_1=17 in RNS/TCS converter

The high-level description of the RNS/TCS converter based on the CRT I is shown in Fig. 6. Instead of the carry-save addition performed in the original version of algorithm a series of two-operand additions is used (lines 7-10). The output sum SPr is divided into low-order segment SPr_Low and high-order segment SPr_High by using appropriate binary masks (lines 11-12). The binary number SPr_High in the line 13 is used for indexing the array containing the reduced modulo M numbers represented by the bits of the high-order segment. In the lines 15-21 the modulo reduction from [0,2M) to [0,M) is done and then the sign is retrieved and the proper value is selected.

```
1  uint38 M = 144259293600;
2  uint42 SPr = 0;//sum of all projections
3  uint37 SPr_Low;//LSB 37 bits of SPr
4  uint5 SPr_High; //MSB 5 bits of SPr
5  int39 S1, S, X_out=0, X_M_2;
6
7  SPr = Pr_m1[r[0]] + Pr_m2[r[1]];
8  SPr = SPr + Pr_m3[r[2]] + Pr_m4[r[3]];
9  SPr = SPr + Pr_m5[r[4]] + Pr_m6[r[5]];
10 SPr = SPr + Pr_m7[r[6]] + Pr_m8[r[7]];
11 SPr_High = SPr (41, 37);//MSB bits grouping
12 SPr_Low = SPr (36, 0);//LSB bits grouping
13 S = modMreduction[SPr_High];
14 S = S + SPr_Low;
15 S1 = S - M;
16 if (S1 >= 0)
17     X_out = S1;
18 else
19     X_out = S;
20
21 X_M_2=X_out-M/2;
22
23 if(X_M_2>0)
24 {
25   X_out=X_out-M;
26 }
27 return X_out;
```

Fig. 6. High level description of the RNS/TCS converter based on CRT I.

The HLS synthesis results in Xilinx Vivado FPGA environment is shown in Figure 6.

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	564
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	15	-	150	50
Multiplexer	-	-	-	10
Register	-	-	268	-
Total	15	0	418	624
Available	264	288	32640	32640
Utilization (%)	5	0	1	1

Fig. 7. Synthesis results of the RNS/TCS converter using CRT I in Xilinx Vivado HLS environment

Previously the converter architecture in the form Fig. 1 has been implemented [9] using the structural approach in VHDL and the Xilinx Virtex XC5VSX50T for the RNS base $B = \{32, 31, 29, 27, 25, 23, 19, 17\}$. The look-up tables have been implemented by using 38 1-bit LUTs for each projection. The eight-operand CSA tree has been realized as two parallel 4-operand carry save-adders in the first layer with the successive 4-operand carry-save adder. As the CPA adders the standard 39-bit Xilinx adders have been used. It can be seen from Table 1 that the hardware amount of the HLS realization is smaller from about 20% for LUTs and 30% for FFs. The smaller maximum frequency of operation of the HLS designed converter can be attributed to the method of automatic multi operand adder construction by the HLS synthesizer. Several experiments have been carried out to increase the operation frequency by inserting pipeline directive inside addition but it has not led to the increase of operational frequency.

Table 1. Comparison of synthesis results of RNS/TCS converters in Xilinx Virtex XC5VSX50T

Parameter	LUT	FF	Max. freq.
HLS	624	418	125Mhz
VHDL	802	621	190Mhz

Finally we may remark that the use of the HLS gives better results with respect to the source lines of code (SLOC) metric [13]. The HLS code has the SLOC at the level 170 but for the VHDL code the SLOC at the level 8300. Additionally, the HLS project have got only one subprogram in one compilation unit but the VHDL code has to be placed in more than 20 subprograms, every in one compilation units, because of the required mapping of the hardware structures. The compilation process (C synthesis) allowing verification of the result in the case of HLS takes about 23 sec, but for the VHDL a full hardware synthesis is required that takes about 30 sec (PC with i7 class 4-cores CPU and 32 GB RAM).

4. CONCLUSIONS

The new approach to the RNS/TCS converter design in Xilinx Vivado FPGA environment using the high-level synthesis has been presented. This approach facilitates the design and debugging processes by shortening the whole procedure and doing it more transparent. This is due to the description of the circuit architecture using a specialized version C/C++ language. The performed research indicates that probably, at least for this type of converters the HLS design can be more effective with respect to the hardware amount and less efficient regarding to operational frequency as compared to the VHDL design.

5. REFERENCES

1. Szabo N.S., Tanaka R. J.: Residue Arithmetic and its Applications to Computer Technology, New York, McGraw-Hill, 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. Wang Y.: Residue-to-binary converters based on the new Chinese remainder theorems, IEEE Trans. Circuits

and Systems-II: Analog and Digital Signal Processing, vol. CAS-47, Sept. 2000, pp.197-205.
5. Wang Z., Jullien G.A., Miller W.C.: An improved residue-to-binary converter, IEEE Trans. Circuits Syst.-I: Fundamental Theory and Applications, vol. CAS-47, Sept. 2000, pp. 1437-1440.
6. Meehan S.J., O'Neil S.D., Vaccaro J.J.: An universal input and output converter, IEEE Trans. Circuits Syst., vol. CAS-37, June 1990, pp. 1158-1162.
7. Cardarilli G.C., Re M., Lojacono R.: A systolic architecture for high performance scaled residue to binary conversion, IEEE Trans. Circuits Syst. –I: Fundamental Theory And Applications, vol. CAS-47, October 2000, pp.667-669.
8. Burgess N.: Scaled and unscaled residue number system to binary conversion techniques using the core function, 1997 IEEE Symposium on Computer Arithmetic, pp. 250-257.
9. Czyżak M., Smyk R.: FPGA realization of the high-speed residue-to-binary converter based on Chinese Remainder Theorem, Poznan University of Technology Academic Journals. Electrical Engineering, 2010, no. 63, pp. 197-205.
10. Meeus W, Van Beeck K., Goedemé T., Meel J., Stroobandt D.: An overview of today's high-level synthesis tools, DOI 10.1007/s10617-012-9096-8, Springer, 2012.
11. Piestrak S.J., Design of high-speed residue-to-binary number system converter based on the Chinese Remainder Theorem. In: Proc. of the Int. Conf. on Computer Design ICCD'94, VLSI in Computers and Processors, Cambridge, MA, October 10-12, 1994, pp. 508-511.
12. Huang C. H., A fully parallel mixed-radix conversion algorithm for residue number applications, IEEE Transactions on Computers, vol. C-32, April 1983, pp. 398 – 402.
13. Nguyen V., Deeds-Rubin S., Tan T., Boehm B., A SLOC counting standard, Center for Systems and Software Engineering, University of South. California, COCOMO II Forum, 2007.

WYSOKOPOZIOMOWA SYNTEZA KONWERTERA RNS/U2 W FPGA

W pracy przedstawiono eksperymentalną wysokopoziomową syntezę w FPGA konwertera z systemu resztowego do systemu reprezentacji z uzupełnieniem do 2 (U2). W zastosowanym podejściu wykorzystano algorytm konwersji na bazie chińskiego twierdzenia o resztach (CRT I). Zauważono, że synteza wysokopoziomowa ułatwia proces projektowania oraz zauważalnie skraca czas testowania układu. Zaprojektowana architektura konwertera przy wykorzystaniu syntezy wysokopoziomowej pochłania o około 20% zasobów układu FPGA mniej niż dla konwertera zaprojektowanego przy użyciu języka VHDL, jednak maksymalna częstotliwość pracy jest niższa o około 30%.

Słowa kluczowe: system resztowy, RNS, system z uzupełnieniem do 2, U2, konwerter RNS/U2, chińskie twierdzenie o resztach, FPGA.