POZNAN UNIVERSITY OF TECHNOLOGY ACADEMIC JOURNALS
No Electrical Engineering 2012

Maciej CZYŻAK
Robert SMYK
Gdansk University of Technology

REALIZATION OF MULTI-OPERAND MODULAR ADDERS IN THE FPGA TECHNOLOGY

The paper presents the design and realization of the Multi-Operand Modular Adder (*MOMA*) structures in the Xilinx *FPGA* environment with the use of the Virtex 6 technology. The design is based on the *LUT*s(2^6x 1) that simulate small *RAM*s that serve as the main component for the look-up realization of addition and modulo generation. In this paper the *MOMA*s for modular addition of five-bit operands are shown. In the paper first the general structures of the *MOMA*s are considered and next two approaches to the multi-operand modulo addition are examined. Both approaches make use of the four-operand *MOMA*s, in the first approach, the four-operand *MOMA* is based on the two-operand modular adders, whereas in the second approach first the four operand binary addition is performed followed by the modulo reduction. The implementation of both *MOMA* types is shown and analyzed with respect to hardware amount and pipelining frequency,

1. INTRODUCTION

Since their introduction at the beginning of 90's the Field Programmable Gate Arrays (*FPGA*) have become in many cases an alternative to the Application Specific Integration Circuits (*ASIC*). The advantage of the *FPGA* over the *ASIC*s is due to the fact that there is no need to design specfic *VLSI* chip which i time-consumming and expensive. Currently the *FPGA*s are known that contain about $6 \cdot 10^9$ transistors and the clock frequencies above 1 GHz. The more advanced *FPGA*s are composed of the Configurable Logic Blocks (*CLB*), *RAM*s, *DSP* blocks and transceivers. The *CLB*s contain smaller parts termed slices that in turn contain small *RAM*s $(2^k \times 1)$ or $(2^{k-1} \times 2)$, where k=4,5.6, latches, multiplexers and logic gates that allow to construct fast carry-chains for implementation of addition. The fact that the *FPGA*s can be programmed in software have changed the attitude towards less conventional arithmetics such as residue arithmetic [1], [2], that can be used for implementation of the digital signal processing algorithms.The realization of the most common *DSP* algorithms such as *FFT* and *FIR* require, along with the modular multiplication, the multi-operand modulo addition.Such addition can be represented as

$$|X|_A = \left| \sum_{i=1}^{N} X_i \right|_A \tag{1}$$

It is assumed that X_i are binary numbers $X_i \leq A-1$, with the binary length $b = \lceil \log_2 A \rceil$. In the *DSP* algorithms that are typically three types of operands, one is for small operands of the length equal to the length of the Residue Number System (*RNS*) moduli m_i, with $m_i \leq 5,6$. Such operands come up in binary/residue converters and when adding residues in residue channels of the *FFT* or *FIR* processors[3]. The second type are the operands of 8-15 bits that are resudues of the special class of moduli akin to $2^l - 1$class. The third type are the operands of 30-50 bits that appear in *RNS/B* converters. In this work we take into consideration the operands of the first group. In general, there are two approches to perform (1). The first is the multi-operand addition with the result in the standard binary form or in the carry-save form. This is followed by the modulo A operation. The second approach is the use of the interleaving of the binary addition for the group of operands and modulo generation and the succeding modulo addition for the reduced number of operands. In this work we shall consider the multi-operand modular addition where Two-Operand Modulo Adder (*TOMA*) is the basic buliding block and such in which the Four-Operand Modulo Adder(*FOMA*) is the basic block. The hardware amount and delay for two *MOMA*s based on *LUT*s implemented using Xilinx Virtex 6 family are analyzed.

2. MULTI-OPERAND MODULO ADDER BASED ON THE TOMA

In this subchapter we shall analyse the *MOMA* structure in the form of the tree structure based on *TOMA*s. First we shall consider the structure of the *TOMA* .

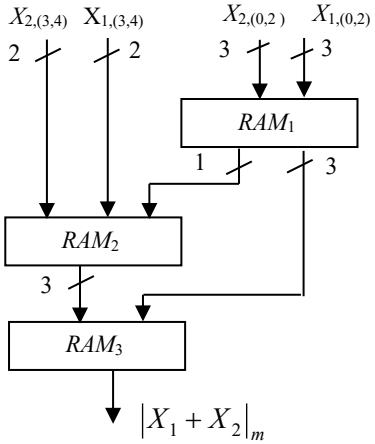

Fig. 2 5-bit *TOMA* based on *RAM*s

MOST WIEDZY Downloaded from mostwiedzy.pl

This structure of the *TOMA* is based on small *RAM*s $(2^6 \times 1)$termed *LUT*s in Xilinx FPGA. The *TOMA* based on the *LUT*s is presented in detail in the companion paper [4] . In the *TOMA* from Fig.2 RAM_1 implements three-bit binary addition of low-order bits, generates 3-bit sum and 1-bit carry that enters the RAM_2 that realizes two-bit addition for high-order bits with the input carry. The formal description is given in [4]. The number of outputs of the *RAM* is equal to the number of the *LUT*s needed to implement the given *RAM*.
In the following we shall analyze hardware amount of the *TOMA* of Fig. 2.
Denote as A_{LUT} the hardware amount of the *LUT* (*RAM* $(2^k \times 1)$). Hence we have for the 5-bit *TOMA* of Fig.1

$$A_{TOMA1} = A_{RAM1} + A_{RAM2} + A_{RAM3} = 4 \cdot A_{LUT} + 3 \cdot A_{LUT} + 5 \cdot A_{LUT} = 12\, A_{LUT} \quad (2)$$

The delay is equal to $t_D = 3 \cdot t_{LUT}$. Such description of delay symbolises only the number of stages of the *LUT*s in the structure and not the real delay because this depends not only on the *LUT* delay but also on internal connections of the *LUT*s within the slice and switching matrix. Although, the *LUT* outputs are registered but the internal structure may influence the attainable pipelining frequency.

Now we may consider the *MOMA* tree structure based on the *TOMA*. It is known that for *n* operands *n-1 TOMA*s are needed. Such *MOMA* has the known form of *n*-1 *TOMA*s as in Fig. 1 for *n*=8.

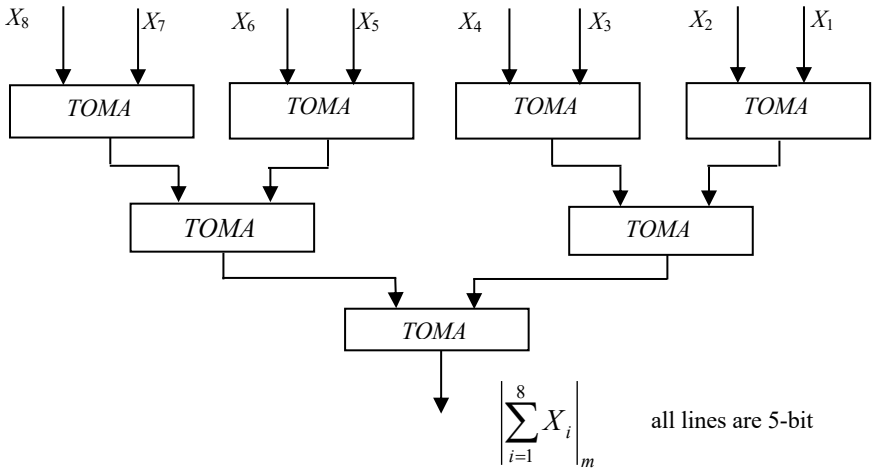

Fig.1 Multi-operand adder for *n*=8 based on the *TOMA*

Hence the *MOMA* based on the *TOMA* requires

MOST WIEDZY Downloaded from mostwiedzy.pl

$$A_{MOMA(n)} = 12 \cdot n \cdot A_{LUT} \tag{3}$$

$$t_{MOMA(n)} = 3 \cdot n \cdot t_{LUT} \tag{4}$$

2. FOUR-OPERAND MODULO ADDER (FOMA)

In this subchapter we shall consider four-operand modulo *m* adder being a series connection of a block of two 5-bit binary adders working in parallel and the modulo *m* generator. It will be examined if such configuration is more effective for *MOMA*s with greater number of operands than the structure based on *TOMA*s. It is assumed that *RAM*s $(2^k \times 1)$,with k=5,6,7 can be used. in the Xilinx environment and Virtex 6 family . The *RAM*s $(2^7 \times 1)$ can configured as two *RAM*s operating in parallel with output multiplexer controlled by the most significant bit of the argument. Such solution does not contribute substantially to the delay. The FOMA is shown in Fig. 3 and the *FOMA* operation is given in *Algorithm 1*.

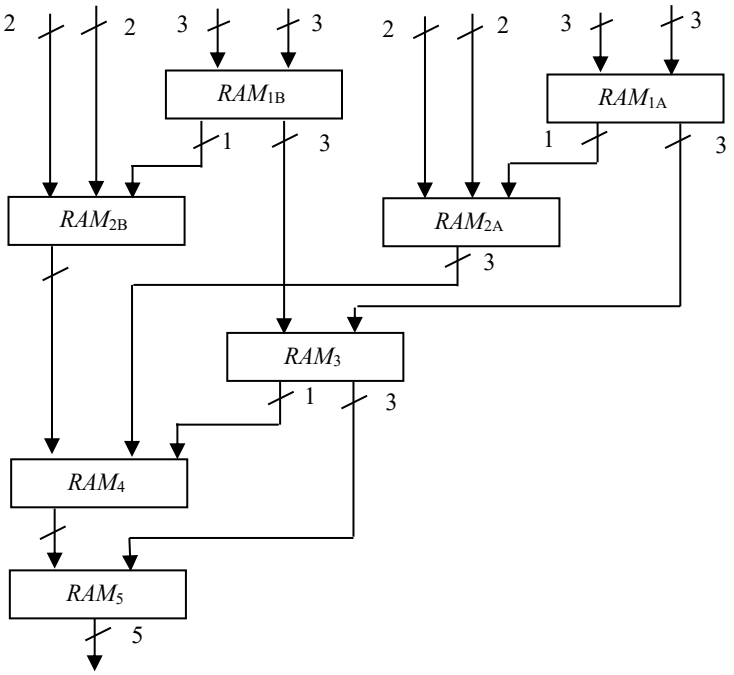

Fig. 2 5-bit FOMA based on *RAM*s

MOST WIEDZY Downloaded from mostwiedzy.pl

Algorithm 1.

Input: 4 5-bit operands X_i, i=0,1,2,3 with $\lceil \log_2 X_i \rceil = 5$.

$$X_i \leftrightarrow X_i = \left(x_4^{(i)}, x_3^{(i)}, x_2^{(i)}, x_1^{(i)}, x_0^{(i)}\right) \tag{5}$$

Output: $r = \left|\sum_{i=0}^{3} X_i\right|_m$

Step 1. Decompose the operands into 2-bit high-order segments and 3-bit low order segments as

$$X_{H,i} = \left(x_4^{(i)}, x_3^{(i)})\right) \text{and } X_{L,i,} = \left(x_2^{(i)}, x_1^{(i)}, x_0^{(i)}\right) \tag{6}$$

Step 2. Compute in parallel by look up (RAM_{1A}, RAM_{1B})

$$S_L^{(01)} = \sum_{j=0}^{2} (x_j^{(0)} + x_j^{(1)}) \cdot 2^j \leftrightarrow S_L^{(01)} = \left(c_3^{(01)}, s_{L,2}^{(01)}, s_{L,1}^{(01)}, s_{L,0}^{(01)}\right) \tag{7a}$$

$$S_L^{(23)} = \sum_{j=0}^{2} (x_j^{(2)} + x_j^{(3)}) \cdot 2^j \leftrightarrow S_L^{(23)} = \left(c_3^{(23)}, s_{L,2}^{(23)}, s_{L,1}^{(23)}, s_{L,0}^{(23)}\right) \tag{7b}$$

Step 3. Construct the address vectors

$$V_H^{(01)} = \left(s_4^{(0)}, s_3^{(0)}, s_3^{(1)}, s_{41}^{(1)}, c_3^{(01)}\right) \tag{8a}$$

$$V_H^{(23)} = \left(s_4^{(2)}, s_3^{(2)}, s_3^{(3)}, s_{41}^{(3)}, c_3^{(23)}\right) \tag{8b}$$

Step 4. Compute in parallel by look up using RAM_{2A} andRAM_{2B}.

$$S_H^{(01)} = \sum_{j=0}^{2} (s_j^{(0)} + s_j^{(1)}) \cdot 2^j + c_3^{(01)} \cdot 2^3 \leftrightarrow S_H^{(01)} = \left(s_{L,5}^{(01)}, s_{L,4}^{(01)}, s_{L,3}^{(01)}\right) \tag{9a}$$

$$S_H^{(23)} = \sum_{j=0}^{2} (s_j^{(2)} + s_j^{(3)}) \cdot 2^j + c_3^{(23)} \cdot 2^3 \leftrightarrow S_H^{(23)} = \left(s_{H,5}^{(23)}, s_{H,4}^{(23)}, s_{H,3}^{(23)}\right) \tag{9b}$$

Step 5.Construct the address vector

$$V_L^{(0123)} = \left(s_{L,2}^{(01)}, s_{L,1}^{(01)}, s_{L,0}^{(01)}, s_{L,2}^{(23)}, s_{L,1}^{(23)}, s_{L,0}^{(23)}\right) \tag{10}$$

MOST WIEDZY Downloaded from mostwiedzy.pl

Step 6. Compute in parallel by look up using RAM_3

$$S_L^{(0123)} = \sum_{j=0}^{2} (s_{L,j}^{(01)} + s_{L,j}^{(23)}) \cdot 2^j \leftrightarrow S_L^{(0123)} = \left(c_3^{(0123)}, s_{L,2}^{(0123)}, s_{L,1}^{(0123)}, s_{L,0}^{(0123)}\right) \quad (11)$$

Step 7.Construct the address vector

$$\mathcal{V}_{\mathcal{H}}^{(0123)} = \left(s_{H,5}^{(01)}, s_{H,4}^{(01)}, s_{H,3}^{(01)}, s_{H,5}^{(23)}, s_{H,4}^{(23)}, s_{H,3}^{(23)}, c_3^{(23)}\right) \quad (12)$$

Step 8. Compute in parallel by look up using RAM_4

$$S_H^{(0123)} = \sum_{j=3}^{5} (s_{H,j}^{(01)} + s_{H,j}^{(23)}) \cdot 2^j + c_3^{(0123)} \cdot 2^3 \leftrightarrow S_H^{(0123)} = \left(s_{H,6}^{(0123)}, s_{H,5}^{(0123)}, s_{H,4}^{(0123)}, s_{H,3}^{(0123)}\right) \quad (13)$$

Step 9. .Construct the address vector

$$\mathcal{V}_{\mathcal{F}}^{(0123)} = \left(s_{H,6}^{(0123)}, s_{H,5}^{(0123)}, s_{H,4}^{(01)}, s_{H,3}^{(01)}, s_{L,2}^{(23)}, s_{L,1}^{(23)}, s_{L,0}^{(23)}\right) \quad (14)$$

Step 10. Compute in parallel by look up using RAM_5

$$R = \left| \sum_{j=3}^{6} S_{H,j}^{(0123)} \cdot 2^j + \sum_{j=0}^{2} S_{K,j}^{(0123)} \cdot 2^j \right|_m \leftrightarrow \mathcal{R} = \left(r_4, r_3, r_2, r_1, r_0\right) \quad (15)$$

The number of 1-bit *LUT*s to construct *TOMA* is equal to . The delay is .
In Table 1 the results of realization of the 8-operand *MOMA* based on seven *TOMA*s in three stages 8 operand *MOMA* based on two *FOMA*s in the first stage and one *TOMA* in the second stage.

Synthesis results in Virtex 6 6vlx240tff784-1:

Eight-operand modular adder based on *TOMA*s:

Timing Summary:

minimum period: 3.011ns (maximum Frequency: 332.116MHz)
minimum input arrival time before clock: 0.413ns
maximum output required time after clock: 0.777ns

Primitive and Black Box Usage: LUT5: 21, LUT6: 63

MOST WIEDZY Downloaded from mostwiedzy.pl

Eight-operand adder based on *FOMA*s and *TOMA*

Timing Summary:
minimum period: 2.985ns (maximum Frequency: 335.008MHz)
minimum input arrival time before clock: 1.766ns
maximum output required time after clock: 0.777ns

Primitive and Black Box Usage: LUT3: 4, LUT5: 32, LUT6: 64

3. CONCLUSIONS

Two ways of design of the multi-operand modular adders in the Xilinx *FPGA* environment with the use of the Virtex 6 have been analyzed. The first technique is based on the tree of *TOMA*s while the second the tree of *FOMA*s with the use of the *TOMA* when needed. It has has been stated that the required hardware amount in terms of the number of *LUT*s is the same for the number of operands being the power of 2, but the delay is smaller for the structure based on *FOMA*s and the difference grows with depth of the tree.

REFERENCES

[1] N.S. Szabo and R.J. Tanaka, *Residue Arithmetic and its Applications to Computer Technology*, New York, McGraw-Hill, 1967.
[2] M .Soderstrand *et al*., *Residue Number System Arithmetic*: Modern Applications in Digital Signal Processing, IEEE Press, NY, 1986.
[3] A. Omondi, B. Premkumar, *Residue Number Systems: Theory and Implementation,* London*,* Imperial College Press, 2007.
[4] S.J.Piestrak, "Design of residue generators and multioperand modulo adders using carry-save adders," IEEE Trans. Comp., vol. 43, pp.68-77, Jan. 1994.
[5] G.Alia, E.Martinelli, Designing multi-operand modular adders, Electronic Letters, vol.32, No. 1, Jan.1996
[6] R.Smyk, M.Czyżak, Design and realization of two-operand modular adders in the FPGA, Poznań University of Technology, Academic Journals, Electrical Engineering (this issue).
[6] G. Alia, E. Martinelli, "VLSI binary-residue converters for pipelined processing," Computer J., vol. 33, no.5, pp. 473-475, 1990.
[7] A.B. Premkumar," A formal framework for conversion from binary to residue numbers," IEEE Trans. Circuits and Systems-II, vol.49, no.2, Feb.2002, pp. 135-144.
[8] M. Czyżak, "High-speed binary-to-residue converter with improved architecture," 27th Int. Conf. on Fundamentals of Electrotechnics and Circuit Theory, Gliwice-Niedzica, May 26-29, 2004, pp. 431-436.

MOST WIEDZY Downloaded from mostwiedzy.pl

MOST WIEDZY Downloaded from mostwiedzy.pl