POZNAN UNIVERSITY OF TECHNOLOGY ACADEMIC JOURNALS
No Electrical Engineering 2012

Robert SMYK
Maciej CZYŻAK
Gdansk University of Technology
Faculty of Electrical and Control Engineering
G. Narutowicza 11/12
80-233 Gdańsk

DESIGN AND REALIZATION OF TWO-OPERAND MODULAR ADDERS IN THE FPGA

The paper presents the structure of modular adders implemented in the Xilinx environment with the use of the Virtex 6 family. Two types of adders are considered, one is for 5-bit moduli and the other is for 6-bit moduli. Their structures have been designed and the experimental results have been shown.

1. INTRODUCTION

Two-Operand Modular Adders (*TOMA*) are the basic block in many Digital Signal Processing (*DSP*) applications, especially in these that make use of the Residue Number System (*RNS*) [1,2]. They are used in modulo generators and in the realization such algorithms asthe *FIR* and *FFT*. In the *DSP* applications based on the *RNS* two main types of the *TOMA* are used. The first category are adders for small moduli of 5–7 bits binary length. The second category are the modular adders used in the output converters and scalers with a range of 30 – 60 bits. The separate category are the *TOMA*s that are used for special moduli types akin to the 2^N-1, 2^N+1 moduli type. In this paper the *TOMA*s belonging to the first category are considered. The specific realizations of the *TOMA*s in the *FPGA* environment will be considered and their properties analyzed. The structures for five bits and six bits operands will be presented.

The *TOMA*s have been presented byBanerji [3], Soderstrand [4], Bayoumi and Jullien [5], Dugdale [6], and Piestrak [7]. More recently Hiasat [8] has presented a new structure of the modular adder.

2. STRUCTURES OF TWO OPERAND MODULAR ADDERS

The *TOMA* performs the following operation:

$$r = |A + B|_m \tag{1}$$

The *TOMA* can be implemented using the combinatoral logic or look-up table approach or combination of both. The form of realisation of the *TOMA* depends upon the number range of the modulus m and the technology. For the smaller m the combinatorial or the table-based approach or mixed approach can be applied. For large operands rather the combinatorial approach is preferred. The basic structure of the combinatorial *TOMA* is shown in Fig. 1.

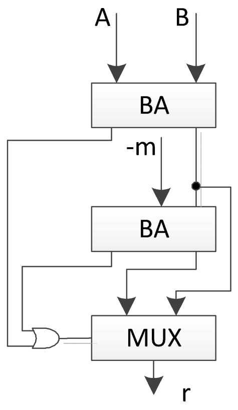

Fig. 1. Modular adder proposed by Bayoumi.and Jullien[5]

Such adder computes in the first stage the binary sum of operands and in the second stage *–m* is subtracted using two's complement. The outputs of the individual binary adders control the output multiplexer, that chooses the correct result.

The second configuration of the modular adder, given by Piestrak [7], is depicted in Fig. 2. In this adder, first $a + b - m$ is computed using the Carry-Save Adder (*CSA*). In the second stage two Binary Adders (*BA*) operate in parallel, one computes $a + b$ and the other the sum of the *CSA* output carry and save vectors. This *TOMA* has a smaller delay than that proposed by Bayoumi and Jullien [5] because the *BA*s work in parallel. Recently a *TOMA* has been proposed by Hiasat (Fig. 3) which makes use of the Carry-Look Ahead (*CLA*) addition. In this *TOMA* initially the Half-Adder-Like (*HAL*) and Semi-HAL cells are applied that are used to obtain propagate and generate signals, and then the output carry is computed that is used to choose the correct output result. The Hiasat *TOMA* requires the smallest hardware amount but has a specific structure for each modulus that makes that for various m each *TOMA* has to be designed individually that increases the design and implementation effort.

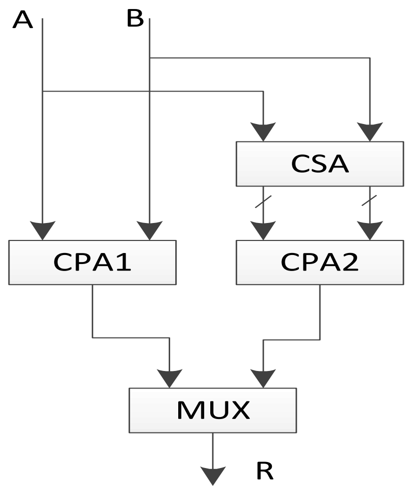

Fig. 2. Modular adder proposed by Piestrak [7].

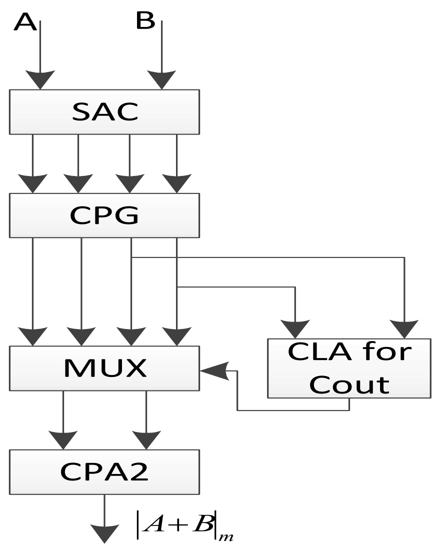

Fig. 3. Modular adder proposed by Hiasat [8]

The exemplary realisations of the table-based and the hybrid structures are presented in Fig. 4a and 4b. In the structures given below the ROM as the table is used but any form of the table can be used. In the first structure the output result is obtained by the look-up, using concatenated representations of the input operands as the table address. For the moduli with a binary length $b = \lceil \log m \rceil$, $2b$-bit address is required, hence $ROM(2^{2b} \times b)$ is needed. This solution is viable only for small moduli because the table size grows exponentially. In addition, such solution severally limits the pipelining rate. In the structure of Fig. 4b the operands are added first in binary that allows to limit the required *ROM* address to $b+1$-bits, hence only $ROM(2^{b+1} \times b)$ is needed.

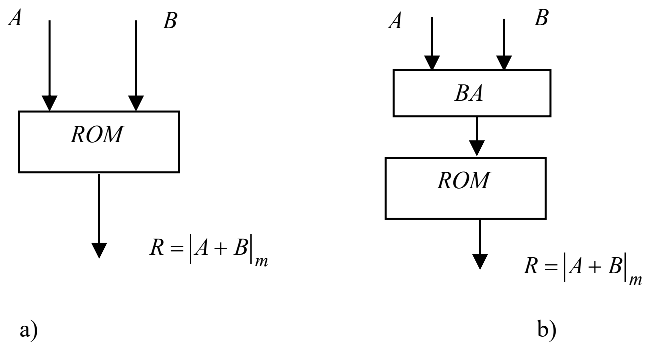

Fig. 4. Exemplary realisations of table-based (a) and hybrid (b) modular adders

In certain hardware technologies such as the Xilinx *FPGA* [9] environment the small *RAMs* with a size $(2^k \times 1)$, where is $k = 4,...,6$ are available. So instead of the *ROM*s the *RAM*s can be applied. This makes possible the table-based approach but with the application of segmentation of operands in such a way that the disposable *RAMs* are used.

3. TWO OPERAND MODULAR ADDER BASED ON RAMs FOR FIVE-BIT OPERANDS

The structure of the five-bit modular adder using *RAM*s is shown in Fig. 5.
The adder computes $R = |A + B|_m$. Let

$$A \leftrightarrow \mathcal{A} = (a_4, a_3, a_2, a_1, a_0)$$
$$B \leftrightarrow \mathcal{B} = (b_4, b_3, b_2, b_1, b_0)$$

Assume that the $RAM(2^6 \times 4)$, $RAM(2^5 \times 3)$ and $RAM(2^6 \times 5)$ are used. Each RAM is implemented using $LUTs(2^6 \times 1)$ and $LUTs(2^5 \times 1)$available in the Xilinx Configurable Logic Block (*CLB*).

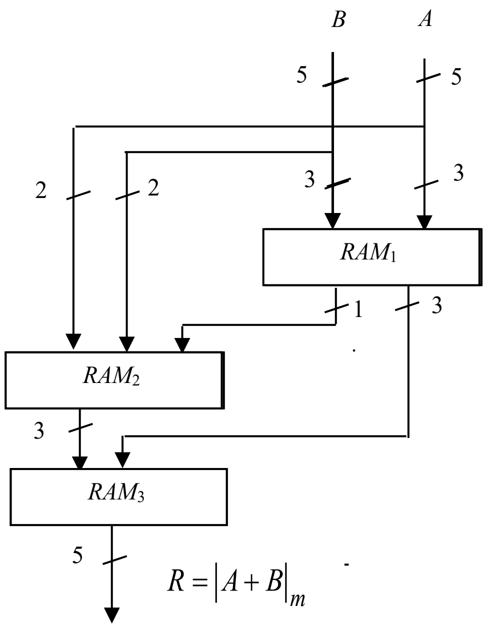

Fig. 5. Five-bit modular adder using $RAM(2^6 \times 4)$, $RAM(2^5 \times 3)$ and $RAM(2^6 \times 5)$

<u>Algorithm</u>:

Input: $A, \lceil \log_2 A \rceil \le 5$ and $B, \lceil \log_2 B \rceil \le 5$, $\lceil \log_2 m \rceil \le 5$
Output: $R = |A+B|_m$

Step1. Decompose *A* and *B* as follows:
$\mathcal{A}_H = (a_4, a_3)$ and $\mathcal{A}_L = (a_2, a_1, a_0)$
$\mathcal{B}_H = (b_4, b_3)$ and $\mathcal{B}_L = (b_2, b_1, b_0)$

Step 2. Construct the vector V_1:
$V_1 = (a_2, a_1, a_0, b_2, b_1, b_0)$

Step 3. Compute by look-up:using *RAM*1

$$S_1 = \sum (a_i + b_i) \cdot 2^i \leftrightarrow (s_3^{(1)}, s_2^{(1)}, s_1^{(1)}, s_0^{(1)})$$

Step 4. Construct the vector V_2:

$$V_2 = (a_4, a_3, b_4, b_3, s_3^{(1)})$$

Step 5. Compute by look-up:using *RAM*2

$$S_2 = (a_4 + b_4) \cdot 2^4 + (a_3 + b_3 + s_3^{(1)}) \cdot 2^3 \leftrightarrow (s_5^{(2)}, s_4^{(2)}, s_3^{(2)})$$

Step 6. Construct the vector V_3:

$$V_3 = (s_5^{(2)}, s_4^{(2)}, s_3^{(2)}, s_2^{(1)}, s_1^{(1)}, s_0^{(1)})$$

Step 7. Compute by look-up using *RAM*3

$$|r|_m = \left| \sum_{i=3}^{5} s_i^{(2)} + \sum_{i=0}^{2} s_i^{(1)} \right|_m \leftrightarrow (r_4, r_3, r_2, r_1, r_0)$$

We consider now the structure of the six-bit *TOMA* . It may have the form as in Fig 6.

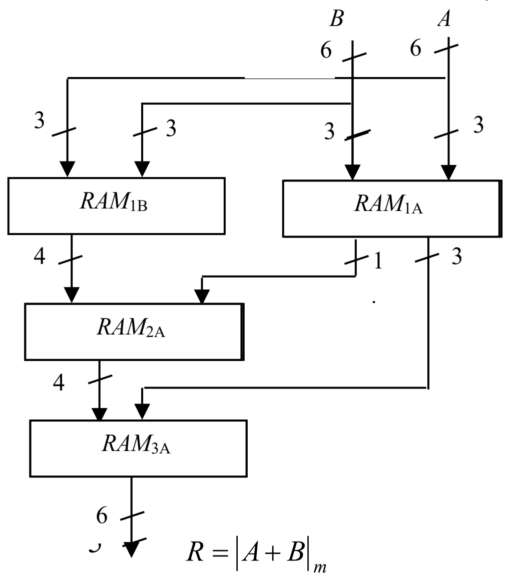

Fig. 6. Six-bit *TOMA* using $RAM(2^6 \times 4)$, $RAM(2^5 \times 4)$ and $RAM(2^7 \times 6)$

4. EXPERIMENTAL RESULTS

The basic element in the Xilinx environment is the *CLB*. This block in Virtex 6 consists of two slices, with each of them containing six $\left(2^6 \times 1\right)$ *LUT*s and additional logic circuitry. The configuration of the *CLB* has been modified by Xilinx in comparison with the previous versions, by limiting of the number of slices and increasing the number of *LUT*s in the individual slice. This was made because the wires inside the slice have the smaller delay than those between slices or the *CLB*s and in this case more wiring can be slice level.

Five-bit modular adder implemented in Virtex 6 requires 12 *LUTs* $\left(2^6 \times 1\right)$ (Fig. 5). In stage 1 four $\left(2^6 \times 1\right)$ *LUT*s are used. In stage 2 four *LUT*s $\left(2^5 \times 1\right)$ are needed, but $\left(2^6 \times 1\right)$ *LUT*s are used, because in this case the *LUT* $\left(2^6 \times 1\right)$ cannot be used as a two $\left(2^5 \times 1\right)$ *LUT*s. This results from the fact that *LUT* $\left(2^6 \times 1\right)$ can be used as two *LUT*s $\left(2^5 \times 1\right)$, when they have common inputs.

The six-bit modular adder of Fig. 6 uses also four $\left(2^6 \times 1\right)$ *LUT*s, whereas in the second and third stage it requires four $\left(2^7 \times 1\right)$ *LUT*s and six $\left(2^6 \times 1\right)$ *LUT*s, respectively. $\left(2^7 \times 1\right)$ *LUT*s are implemented by parallel connection of two $\left(2^6 \times 1\right)$ *LUT*s with the multiplexed output controlled by the seventh bit.

The delay verification in the design system has been performed. The values of the delay indicate that such configurations of the *TOMA*s do not severely limit the maximum pipelining frequency of the system equal to 550MHz. Two techniques of synthesis have been applied. The first was direct programming of *LUT*s and the second was the use of arrays in the synthesis process.

As the effect of synthesis the following results have been obtained for five-bit and six-bit *TOMA*.

a) five-bit*TOMA* with the direct programming of *LUT*s

Timing Summary:
 minimum period: 2.985*ns* (maximum frequency: 335.008MHz)
 minimum input arrival time before clock: 0.413*ns*
 maximum output required time after clock: 0.777ns
Cell usage *LUT*5: 3, *LUT*6: 9

b) five-bit *TOMA* with the use of the automatic synthesis:

Selected Device : 6vlx240tff1156-1
Cell usage *LUT2*: 1, *LUT3*: 1, *LUT4*: 1, *LUT5*: 2, *LUT6*: 7

Timing Summary:
minimum period: 1.695ns (maximum frequency: 589.970MHz)
minimum input arrival time before clock: 0.686ns
maximum output required time after clock: 0.777ns

The six-bit *TOMA* has been implemented using indirect programming of *LUT*s with the application of the arrays in the VHDL instead of directly programming *LUT*s, the automatic synthesis performed by the Xilinx design system has been applied. The following results for six-bit *TOMA* have been obtained:
Timing Summary:
minimum period: 1.695*ns* (maximum frequency: 589.970MHz)
minimum input arrival time before clock: 0.686ns
maximum output required time after clock: 0.777ns

Cell Usage: *LUT*2: 4, *LUT*3: 4, *LUT*4: 3, *LUT*6: 15

It is seen that the automatic synthesis requires more *LUT*s than direct synthesis but the pipelining frequency is greater.

5. CONCLUSIONS

The paper presents the derivation and experimental realization of the *TOMA* for 5- and 6-bit operands in Xilinx *FPGA* environment using theVirtex 6 family. The detailed analysis and careful realization is very important because such adders serve as building blocks of the large multioperand adders and directly influence effectivness of such adders. The obtained results have shown that the if the *MOMA* will be used as the basic block of a large MOMA it is worthwhile to try both synthesis methods and choose the structure that fulfills the design goals.

REFERENCES

[1] Szabo N., Tanaka R., Residue Arithmetic and Its Applications to Computer Technology, McGraw-Hill, New York, 1967.

2] Soderstrand M., Jenkins M., Jullien G., Taylor F., Residue Number System Arithmetic: Modern Applications in Digital Signal Processing, IEEE Press, 1986.

[3] Banerji D., A Novel Implementation Method for Addition and Subtraction in Residue Number Systems, IEEE Transactions on Computers, Volume 23, Number 1, Pages 106-109, 1974.

[4] Soderstrand M., A New Hardware Implementation of Modulo Adders for Residue Number Systems, Proc. IEEE 26th Midwest Symp. Circuits and Systems, Pages 412-415, August 1983.

[5] Bayoumi M., Jullien G., A VLSI Implementation of Residue Adders, IEEE Trans. Circuits and Systems, Volume 34, Pages 284-288, March 1987.

[6] Dugdale M., VLSI Implementation of Residue Adders Based Based On Binary Adders, IEEE Trans. Circuits and Systems II: Analog and Digital Signal Processing, Volume 39, Pages 325-329, May 1992.

[7] Piestrak S., Design of Residue Generators and Multioperand Modular Adders Using Carry-Save Adders, IEEE Trans. Computers, Volume 43, Nomber 1, Pages 68-77, January 1994.

[8] Hiasat A. A., High-Speed and Reduced-Area Modular Adder Structures for RNS, IEEE Trans. Computers, Volume 51, No. 1, Pages 84-89, January 2002.

[9] www.xilinx.com (available online)