Reconfigurable Multiple Operation Array

Humberto Calderon and Stamatis Vassiliadis

Computer Engineering Laboratory,
Electrical Engineering Dept., EEMCS, TU Delft, The Netherlands
Email:{H.Calderon,S.Vassiliadis}@ewi.tudelft.nl,
WWW home page: http://ce.et.tudelft.nl

Abstract. In this paper, we investigate the collapsing of eight multi-operand addition related operations into a single and common (3:2)counter array. We consider for this unit multiplication in integer and fractional representations, the Sum of Absolute Differences (SAD) in unsigned, signed magnitude and two's complement notation. Furthermore, the unit also incorporates a Multiply-Accumulation unit (MAC) for two's complement notation. The proposed multiple operation unit was constructed around 10 element arrays that can be reduced using well known counter techniques, which are feed with the necessary data to perform the proposed eight operations. It is estimated that 6/8 of the basic (3:2)counter array is shared by the operations. The obtained results of the presented unit indicates that is capable of processing a 4x4 SAD macro-block in 36.35 ns and takes 30.43 ns to process the rest of the operations using a VIRTEX II PRO xc2vp100-7ff1696 FPGA device.

1 Introduction: The need for reconfigurability

The need of multimedia Instruction Set Architectures (ISA) extensions with high performance processing and flexibility characteristics are potentially met with the use of reconfigurable technologies[7]. The new emerging capabilities in Reconfigurable Computing (RC) are letting us to dynamically reconfigure a portion of a FPGA. Reconfigurable fabrics can be used to support a common and basic logic blocks intended to be used in several operand addition related operations. The common blocks can be configured in advance; therefore, the hardware differences needed for performing a particular operation will be reconfigured partially based on the hardware differences between the common basic array and the new needed functionalities, instead of programming totally the new entire desired operation [10]. This work presents the collapsing of eight multi-operand addition related operations into the common hardware suitable to be implemented into a VLSI as a run time configurable unit and also over a reconfigurable technology as a reconfigurable run time unit. The multiple operation array has the following embedded units and features:

– A 16 x 16 bit multiplier for integer and fractional representations with universal notations [1].

[1] Universal notation, in the context of this article, assumes operands and results to be in unsigned, sign magnitude and two's complement notations

- A 4 x 4 picture elements concurrent SAD macro-block in universal notation.
- The Multiply-Accumulation Unit (MAC) for two's complement notation.
- A performance of 35.6 ns for 4x4 SAD macro-block and 30.43 ns for the rest of 7 operations.

The paper is organized as follows. Section 2 outlines the Reconfigurable Multiple Operation Array organization. Section 3 presents the experimental results of the mapped unit, as well as other comparison units in terms of area used and time delay. Finally, the article is concluded in section 4.

2 Reconfigurable Multiple Operation Array

This section begins presenting a background and relevant work; consequently, a general array description of the Reconfigurable Multiple Operation Array is described. Finally, a complete description of the equations set for the construction and reproduction of the proposed unit are shown.

2.1 Background and related work

Motion estimation techniques divide an image frame for its processing in macro-blocks of $n * n$ picture elements (pels). The processing establishes if there is a difference between two image blocks using the Sum of Absolute Differences (SAD) operation, establishing the pels differences between two chosen frames. Equation 1 states the metric to evaluate the searched block.

$$\sum_{j=1}^{16}\sum_{i=1}^{16}|IN1(x+i,y+j)-IN2((x+r)+i,(y+s)+j)| \tag{1}$$

where, the duple (x, y) represents the position of the current block, and the pair (r, s) denotes the displacement of $IN2$, relative to reference block $IN1$. Different investigations including a previous author's work have been proposed to speed up the critical SAD kernel [3],[4],[5]. The processing requires that SAD input terms received by the multiple operation array have to be ordered previously, in order to compute correctly the operation; therefore, the absolute operation $| IN1 - IN2 |$ can be substituted, with $IN1 - IN2$ or $IN2 - IN1$ depending whether $IN1$ or $IN2$ is the smallest and thus obtaining a positive result. As is suggested in [5] we can make this operation inverting one of the operands and computing the carry out of the addition of both operands as stated by the following equations:

$$\overline{IN1} + IN2 \geq 2^i - 1 \tag{2}$$

therefore

$$IN2 > IN1 \tag{3}$$

means checking whether the addition of the bit inverted *IN1* and the operand *IN2* produces a carry out. The outcome determines which one is the smallest, depending on the existence or not of the carry output. Consequently, a simple logic can be used to correct the terms and feed the array. Regarding universal units capable to work with universal notations, the reader is referred to view the predecessor unit in [6] and a recent reintroduction in [2]. An extra row of

(3:2)counter for MAC operation is used into the Reconfigurable Multiple Operation Array as a technique for accelerating the processing, a detailed description of this kind of approach units can be seen in [12].

2.2 The array description

The proposed unit has been constructed around a rectangular array that can be reduced using (3:2)counters; all ten operational fields of the array presented in figure 1(a), called sections in the paper, receive the integer numbers $X(i)$ and $Y(i)$, the fractional numbers $A(i)$ and $B(i)$, (all represented with 16 bits), the $W(i)$ summand represented with 32 bits for MAC operation as well as the 32 SAD terms depicted in equation 4 for the computation of a 4x4 macro-block:

$$I_{(j,i)} = I_{(j,15)} I_{(j,14)} \cdot \cdots \cdot I_{(j,1)} I_{(j,0)} \quad \forall \;\; 1 \leq j \leq 32 \tag{4}$$

where the index j states the 32 inputs; and i is used to denote the positional weight of the data bits in each input.
The multiplication related operations of the proposed unit requires the partial product creation as stated by the following equations: $\forall \quad 0 \;\leq j \leq \; 15$ while $\forall \quad 0 \;\leq i \leq \; 15$

$$Z_{(j,i)} = X_{(i)} \cdot Y_{(j)} \tag{5}$$

$$F_{(j,i)} = A_{(i)} \cdot B_{(j)} \tag{6}$$

All these data, feeds the (3:2)counters through 3 inputs $I1(j)(i)$, $I2(j)(i)$ and $I3(j)(i)$; and produces two outputs $S(j)(i)$ and $C(j)(i)$. The basic layout topology of the (3:2)counter is presented on figure 1(b) and this detail is replicated in all sections of the array; nevertheless, in the limit of both sides, left and right, the carry input $I3(j)(i)$ uses an additional multiplexor which is in charge of propagating or inhibiting the carry out of the (3:2)counters of the right side, and introduces a Hot One (HO) or a zero for SAD processing. The multiplexor presented in the aforementioned figure 1(b), represented by the bar, has a signal e to control the data related operations and reconfiguring in this way the operation being computed by the array. Furthermore has to be noticed that the first row of both sides, the left and the right uses three multiplexors instead of one as is described further (see equations, section 2.3).

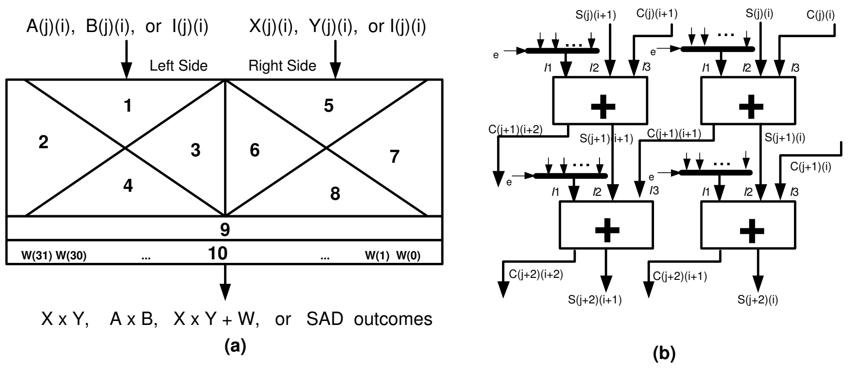

Fig. 1. The Reconfigurable Multiple Operation Array Scheme

Regarding the use of these sections, depicted in figure 1(a), (3:2)counters of sections 5, 6, 3 and 4 process the integer partial products described by equation 5. Fractional processing is achieved with sections 1, 3, 6 and 8; furthermore, sections 1 and 2 as well as 2 and 4 process the sign extensions, and zeros given the universal characteristic to the multiplier processing for integers and fractional numbers respectively. Concerning SAD processing, two main sections of the array are been used, the left and right side. The left side utilizes the sections 1, 2, 3 and 4 and the right side uses section 5, 6, 7, 8. Additionally, section 9 as part of the multiplier array, is used to add the last partial products; and section 10 receives the W(i) addend for MAC processing. Table 1 summarizes the terms received by $I1$ in each one of the main sections through the 8 to 1 multiplexor. From the table is evident that we process numbers of two complement representation with sign extension, and signed magnitude numbers are processed like positive numbers, making the sign bit zero and updating the result with the XOR of multiplicand and multiplier signs.

Table 1. Sections of the Reconfigurable Multiple Operation Array

	SAD	Unsigned Integer	Signed Integer	Two's Integer	MAC Two's	Unsigned Fractional	Signed Fractional	Two's Fractional
Section 1	$I_{(j,i)}$	0	0	$Z_{(j,15)}$	$Z_{(j,15)}$	$F_{(j,i)}$	$F_{(j,i)}$	$F_{(j,i)}$
Section 2	$I_{(j,i)}$	0	0	$Z_{(j,15)}$	$Z_{(j,15)}$	0	0	$F_{(j,15)}$
Section 3	$I_{(j,i)}$	$Z_{(j,i)}$	$Z_{(j,i)}$	$Z_{(j,i)}$	$Z_{(j,i)}$	$F_{(j,i)}$	$F_{(j,i)}$	$F_{(j,i)}$
Section 4	$I_{(j,i)}$	0	0	$Z_{(j,15)}$	$Z_{(j,15)}$	0	0	$F_{(j,15)}$
Section 5	$I_{(j,i)}$	$Z_{(j,i)}$	$Z_{(j,i)}$	$Z_{(j,i)}$	$Z_{(j,i)}$	0	0	0
Section 6	$I_{(j,i)}$	$Z_{(j,i)}$	$Z_{(j,i)}$	$Z_{(j,i)}$	$Z_{(j,i)}$	$F_{(j,i)}$	$F_{(j,i)}$	$F_{(j,i)}$
Section 7	$I_{(j,i)}$	0	0	0	0	0	0	0
Section 8	$I_{(j,i)}$	0	0	0	0	$F_{(j,i)}$	$F_{(j,i)}$	$F_{(j,i)}$
Section 9	0	$Z_{(15,i)}$	$Z_{(15,i)}$	$Z_{(15,i)}$	$Z_{(15,i)}$	0	0	0
Section 10	0	0	0	0	$W_{(i)}$	0	0	0

2.3 The Array Construction equations description

The multiple operation array is conformed by 32 columns and 16 rows of (3:2)counters, giving a total of 496 (3:2)counters, see figure 2. It can be mentioned that figure 2 contains all the details from figure 1(a). The notation used for representing (3:2)counters gives us information of the different kind of data received by these core logic blocks. The first columns of sections 3 and 7 (columns 0 and 16, see figures 1(a) and 2) from row 1 to 8, are used to introduce the Hot One (HO), the rest of the column introduces zero, this data is necessary for SAD calculation due to $A - B = A + \overline{B} + 1$, also column 16 inhibits the carry propagation from the right side. It should also be noted that the last row of this figure, symbolized with a plus sign + represents the final adder used to calculate outcome values of the (3:2)counter array. The remainder of the section describes in a detailed way the equations for the 10 element arrays of (3:2) counter subsections, detailing for all cases the input $I1$ and also the other equations for inputs $I2$ and $I3$ when the application is different to the functionality presented in figure 1(b).

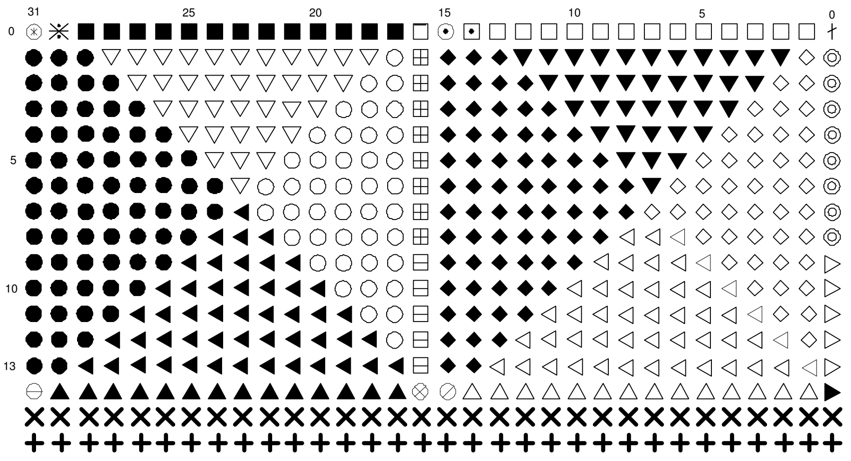

Fig. 2. The Reconfigurable Multiple Operation Array Organization

Section 1 The section is divided into four subsections; the first three are embedded into the first row.

□ **:** $I1_{(0,16)} = I_{(1,0)} \cdot e_0 + Z_{(1,15)} \cdot e_1 + Z_{(1,15)} \cdot e_2 + Z_{(1,15)} \cdot e_3 + Z_{(1,15)} \cdot e_4 + F_{(15,1)} \cdot e_5 + F_{(15,1)} \cdot e_6 + F_{(15,1)} \cdot e_7$

$I2_{(0,16)} = I_{(2,0)} \cdot e_0 + 0 \cdot e_1 + 0 \cdot e_2 + Z_{(0,15)} \cdot e_3 + Z_{(0,15)} \cdot e_4 + F_{(14,2)} \cdot e_5 + F_{(14,2)} \cdot e_6 + F_{(14,2)} \cdot e_7$
$I3_{(0,16)} = I_{(3,0)} \cdot e_0 + 0 \cdot e_1 + 0 \cdot e_2 + 0 \cdot e_3 + 0 \cdot e_4 + 0 \cdot e_5 + 0 \cdot e_6 + 0 \cdot e_7$
■ **:** $\forall \quad 1 \leq i \leq 13$

$I1_{(0,i+16)} = I_{(1,i)} \cdot e_0 + 0 \cdot e_1 + 0 \cdot e_2 + Z_{(0,15)} \cdot e_3 + Z_{(0,15)} \cdot e_4 + F_{(15,i+1)} \cdot e_5 + F_{(15,i+1)} \cdot e_6 + F_{(15,i+1)} \cdot e_7$

$I2_{(0,i+16)} = I_{(2,i)} \cdot e_0 + 0 \cdot e_1 + 0 \cdot e_2 + Z_{(1,15)} \cdot e_3 + Z_{(1,15)} \cdot e_4 + F_{(14,i+2)} \cdot e_5 + F_{(14,i+2)} \cdot e_6 + F_{(14,i+2)} \cdot e_7$

$I3_{(0,i+16)} = I_{(3,i)} e_0 + 0 \cdot e_1 + 0 \cdot e_2 + 0 \cdot e_3 + 0 \cdot e_4 + 0 \cdot e_5 + 0 \cdot e_6 + 0 \cdot e_7$

⁎ **:** $I1_{(0,30)} = I_{(1,14)} \cdot e_0 + 0 \cdot e_1 + 0 \cdot e_2 + Z_{(1,15)} \cdot e_3 + Z_{(1,15)} \cdot e_4 + F_{(15,1)} \cdot e_5 + F_{(15,1)} \cdot e_6 + F_{(15,1)} \cdot e_7$

$I2_{(0,30)} = I_{(2,14)} \cdot e_0 + 0 \cdot e_1 + 0 \cdot e_2 + Z_{(0,15)} \cdot e_3 + Z_{(0,15)} \cdot e_4 + 0 \cdot e_5 + 0 \cdot e_6 + F_{(14,15)} \cdot e_7$
$I3_{(0,30)} = I_{(3,14)} \cdot e_0 + 0 \cdot e_1 + 0 \cdot e_2 + 0 \cdot e_3 + 0 \cdot e_4 + 0 \cdot e_5 + 0 \cdot e_6 + 0 \cdot e_7$

⊛ **:** $I1_{(0,31)} = I_{(1,15)} \cdot e_0 + 0 \cdot e_1 + 0 \cdot e_2 + Z_{(1,15)} \cdot e_3 + Z_{(1,15)} \cdot e_4 + 0 \cdot e_5 + 0 \cdot e_6 + F_{(15,1)} \cdot e_7$

$I2_{(0,31)} = I_{(2,15)} \cdot e_0 + 0 \cdot e_1 + 0 \cdot e_2 + Z_{(0,15)} \cdot e_3 + Z_{(0,15)} \cdot e_4 + 0 \cdot e_5 + 0 \cdot e_6 + F_{(14,15)} \cdot e_7$
$I3_{(0,31)} = I_{(3,15)} \cdot e_0 + 0 \cdot e_1 + 0 \cdot e_2 + 0 \cdot e_3 + 0 \cdot e_4 + 0 \cdot e_5 + 0 \cdot e_6 + 0 \cdot e_7$
▽ **:** Let $n = 2$, $m = 12$; $\forall \quad 1 \leq j \leq 6$ and; $\forall \quad n \leq i \leq m$,with $m = m - 1$; and ;$n = n + 1$; for each column

$I1_{(j,i+16)} = I_{(j+3,i)} \cdot e_0 + 0 \cdot e_1 + 0 \cdot e_2 + Z_{(j+1,15)} \cdot e_3 + Z_{(j+1,15)} \cdot e_4 + F_{(14-j,i+3)} \cdot e_5 + F_{(14-j,i+3)} \cdot e_6 + F_{(14-j,i+3)} \cdot e_7$

Section 2 The section has the two representative equations:
● : Let $n = 13$; $\forall \ 1 \leq j \leq 6$ and $\forall \ n \leq i \leq 15$, with $n = n - 1$

$$I1_{(j,i+16)} = I_{(j+3,i)} \cdot e_0 + 0 \cdot e_1 + 0 \cdot e_2 + Z_{(j+1,15)} \cdot e_3 + Z_{(j+1,15)} \cdot e_4 + 0 \cdot e_5 + 0 \cdot e_6 + F_{(14-j,15)} \cdot e_7$$

● : Let $n = 8$; $\forall \ 7 \leq j \leq 13$ and $\forall \ n \leq i \leq 15$, with $n = n + 1$
$I1_{(j,i+16)} = I_{(j+3,i)} \cdot e_0 + 0 \cdot e_1 + 0 \cdot e_2 + Z_{(j+1,15)} \cdot e_3 + Z_{(j+1,15)} \cdot e_4 + 0 \cdot e_5 + 0 \cdot e_6 + F_{(14-j,15)} \cdot e_7$

Section 3 The section is subdivided into four main parts:
⊞ : $\forall \ 1 \leq j \leq 8$

$$I1_{(j,16)} = I_{(j+3,0)} \cdot e_0 + Z_{(j+1,15-j)} \cdot e_1 + Z_{(j+1,15-j)} \cdot e_2 + +Z_{(j,16-j)} \cdot e_3 + Z_{(j,16-j)} \cdot e_4 + F_{(14-j,j+2)} \cdot e_5 + F_{(14-j,j+2)} \cdot e_6 + F_{(14-j,j+2)} \cdot e_7$$

$$I3_{(j,16)} = 1 \cdot e_0 + C_{(j-1,i+15)} \cdot e_1 + C_{(j,i+16)} \cdot e_2 + C_{(j,i+16)} \cdot e_3 + C_{(j,i+16)} \cdot e_4 + C_{(j,i+16)} \cdot e_5 + C_{(j,i+16)} \cdot e_6 + C_{(j,i+16)} \cdot e_7$$

⊟ : $\forall \ 9 \leq j \leq 13$

$$I1_{(j,16)} = I_{(j+3,0)} \cdot e_0 + Z_{(j+1,15-j)} \cdot e_1 + Z_{(j+1,15-j)} \cdot e_2 + Z_{(j,15-j)} \cdot e_3 + Z_{(j,15-j)} \cdot e_4 + F_{(14-j,j+2)} \cdot e_5 + F_{(14-j,j+2)} \cdot e_6 + F_{(14-j,j+2)} \cdot e_7$$

$$I3_{(j,16)} = I3_{(j,16)} = 1 \cdot e_0 + C_{(j,i+16)} \cdot e_1 + C_{(j,i+16)} \cdot e_2 + C_{(j,i+16)} \cdot e_3 + C_{(j,i+16)} \cdot e_4 + C_{(j,i+16)} \cdot e_5 + C_{(j,i+16)} \cdot e_6 + C_{(j,i+16)} \cdot e_7$$

First part **∘ :** Let $m = 1$; $\forall \ 1 \leq j \leq 6$ and $\forall \ 1 \leq i \leq m$, with $m = m + 1$

$$I1_{(j,i+16)} = I_{(j+3,i)} \cdot e_0 + Z_{(j+1,i+14)} \cdot e_1 + Z_{(j+1,i+14)} \cdot e_2 + Z_{(j+1,i+14)} \cdot e_3 + Z_{(j+1,i+14)} \cdot e_4 + F_{(14-j,i+3)} \cdot e_5 + F_{(14-j,i+3)} \cdot e_6 + F_{(14-j,i+3)} \cdot e_7$$

Second part **∘ :** Let $m = 6$; $\forall \ 7 \leq j \leq 12$ and $\forall \ 1 \leq i \leq m$, with $m = m - 1$

$$I1_{(j,i+16)} = I_{(j+3,i)} \cdot e_0 + Z_{(j+1,i+15-j)} \cdot e_1 + Z_{(j+1,i+15-j)} \cdot e_2 + Z_{(j+1,i+15-j)} \cdot e_3 + Z_{(j+1,i+15-j)} \cdot e_4 + F_{(14-j,i+j+2)} \cdot e_5 + F_{(14-j,i+j+2)} \cdot e_6 + F_{(14-j,i+j+2)} \cdot e_7$$

Section 4 ◀ : Let $n = 7$, $m = 7$; $\forall \ 7 \leq j \leq 13$ and $\forall \ n \leq i \leq m$; with $n = n - 1$ and $m = m + 1$

$$IN1_{(j,i+16)} = I_{(j+3,i)} \cdot e_0 + Z_{(j+1,15-j+i)} \cdot e_1 + Z_{(j+1,15-j+i)} \cdot e_2 + Z_{(j+1,15-j+i)} \cdot e_3 + Z_{(j+16,15-j+i)} \cdot e_4 + 0 \cdot e_5 + 0 \cdot e_6 + F_{(14-j,15)} \cdot e_7$$

Section 5 The section is divided into four subsections; the first three are embedded in the first row.
∤ : $I1_{(0,0)} = I_{(17,0)} \cdot e_0 + Z_{(0,0)} \cdot e_1 + Z_{(0,0)} \cdot e_2 + Z_{(0,0)} \cdot e_3 + Z_{(0,0)} \cdot e_4 + 0 \cdot e_5 + 0 \cdot e_6 + 0 \cdot e_7$
$I2_{(0,0)} = I_{(18,0)} \cdot e_0 + 0 \cdot e_1 + 0 \cdot e_2 + 0 \cdot e_3 + 0 \cdot e_4 + 0 \cdot e_5 + 0 \cdot e_6 + 0 \cdot e_7$
$I3_{(0,0)} = I_{(19,0)} \cdot e_0 + 0 \cdot e_1 + 0 \cdot e_2 + 0 \cdot e_3 + 0 \cdot e_4 + 0 \cdot e_5 + 0 \cdot e_6 + 0 \cdot e_7$
□ :$\forall \ 1 \leq i \leq 13$
$I1_{(0,i)} = I_{(17,i)} \cdot e_0 + Z_{(0,i)} \cdot e_1 + Z_{(0,i)} \cdot e_2 + Z_{(0,i)} \cdot e_3 + Z_{(0,i)} \cdot e_4 + 0 \cdot e_5 + 0 \cdot e_6 + 0 \cdot e_7$

$I2_{(0,i)} = I_{(18,i)}\cdot e_0 + Z_{(1,i-1)}\cdot e_1 + Z_{(1,i-1)}\cdot e_2 + Z_{(1,i-1)}\cdot e_3 + Z_{(1,i-1)}\cdot e_4 + 0\cdot e_5 + 0\cdot e_6 + 0\cdot e_7$
$I3_{(0,i)} = I_{(19,i)} e_0 + 0\cdot e_1 + 0\cdot e_2 + 0\cdot e_3 + 0\cdot e_4 + 0\cdot e_5 + 0\cdot e_6 + 0\cdot e_7$

$\boxdot$ **:** $I1_{(0,14)} = I_{(17,14)}\cdot e_0 + Z_{(0,14)}\cdot e_1 + Z_{(0,14)}\cdot e_2 + Z_{(0,14)}\cdot e_3 + Z_{(0,14)}\cdot e_4 + F_{(14,0)}\cdot e_5 + F_{(14,0)}\cdot e_6 + F_{(14,0)}\cdot e_7$

$I2_{(0,14)} = I_{(18,14)}\cdot e_0 + Z_{(1,13)}\cdot e_1 + Z_{(1,13)}\cdot e_2 + Z_{(1,13)}\cdot e_3 + Z_{(1,13)}\cdot e_4 + 0\cdot e_5 + 0\cdot e_6 + 0\cdot e_7$
$I3_{(0,14)} = I_{(19,14)}\cdot e_0 + 0\cdot e_1 + 0\cdot e_2 + 0\cdot e_3 + 0\cdot e_4 + 0\cdot e_5 + 0\cdot e_6 + 0\cdot e_7$

$\odot$ **:** $I1_{(0,15)} = I_{(17,15)}\cdot e_0 + Z_{(0,15)}\cdot e_1 + Z_{(0,15)}\cdot e_2 + Z_{(0,15)}\cdot e_3 + Z_{(0,15)}\cdot e_4 + F_{(15,0)}\cdot e_5 + F_{(15,0)}\cdot e_6 + F_{(15,0)}\cdot e_7$

$I2_{(0,15)} = I_{(18,15)}\cdot e_0 + Z_{(1,14)}\cdot e_1 + Z_{(1,14)}\cdot e_2 + Z_{(1,14)}\cdot e_3 + Z_{(1,14)}\cdot e_4 + F_{(14,1)}\cdot e_5 + F_{(14,1)}\cdot e_6 + F_{(14,1)}\cdot e_7$

$I3_{(0,15)} = I_{(19,15)}\cdot e_0 + 0\cdot e_1 + 0\cdot e_2 + HO\cdot e_3 + HO\cdot e_4 + 0\cdot e_5 + 0\cdot e_6 + HO\cdot e_7$
∇ **:** Let $n = 2$, $m = 12$; $\forall \quad 1 \le j \le 6$ and $\forall \quad n \le i \le m$,with $m = m - 1$; and $n = n + 1$;

$I1_{(j,i+16)} = I_{(j+19,i)}\cdot e_0 + Z_{(j+1,i-j-1)}\cdot e_1 + Z_{(j+1,i-j-1)}\cdot e_2 + Z_{(j+1,i-j-1)}\cdot e_3 + Z_{(j+1,i-j-1)}\cdot e_4 + 0\cdot e_5 + 0\cdot e_6 + 0\cdot e_7$

Section 6 The section has the two representative equations:
$\bullet$ **:** Let $n = 13$; $\forall \quad 1 \le j \le 6$ and $\forall \quad n \le i \le 15$, with $n = n - 1$

$I1_{(j,i)} = I_{(j+19,i)}\cdot e_0 + Z_{(j+1,i-j-1)}\cdot e_1 + Z_{(j+1,i-j-1)}\cdot e_2 + Z_{(j+1,i-j-1)}\cdot e_3 + Z_{(j+1,i-j-1)}\cdot e_4 + F_{(14-j,i-14+j)}\cdot e_5 + F_{(14-j,i-14+j)}\cdot e_6 + F_{(14-j,i-14+j)}\cdot e_7$

$\bullet$ **:** Let $n = 8$; $\forall \quad 7 \le j \le 13$ and $\forall \quad n \le i \le 15$, with $n = n - 1$

$I1_{(j,i)} = I_{(j+19,i)}\cdot e_0 + Z_{(j+1,i+j-1)}\cdot e_1 + Z_{(j+1,i+j-1)}\cdot e_2 + Z_{(j+1,i+j-1)}\cdot e_3 + Z_{(j+1,i+j-1)}\cdot e_4 + F_{(14-j,i-14+j)}\cdot e_5 + F_{(14-j,i-14+j)}\cdot e_6 + F_{(14-j,i-14+j)}\cdot e_7$

Section 7 The section is conformed by three subsections:
$\circledcirc$ **:** $\forall \quad 1 \le j \le 8$,
$I1_{(j,0)} = I_{(j+19,i)}\cdot e_0 + 0\cdot e_1 + 0\cdot e_2 + 0\cdot e_3 + 0\cdot e_4 + 0\cdot e_5 + 0\cdot e_6 + 0\cdot e_7$
$I3_{(j,0)} = 1\cdot e_0 + 0\cdot e_1 + 0\cdot e_2 + 0\cdot e_3 + 0\cdot e_4 + 0\cdot e_5 + 0\cdot e_6 + 0\cdot e_7$
$\rhd$ **:** Let $n = 1$, $m = 6$, $\forall \quad 9 \le j \le 13$, with $m = m - 1$
$I1_{(j,0)} = I_{(j+19,i)}\cdot e_0 + 0\cdot e_1 + 0\cdot e_2 + 0\cdot e_3 + 0\cdot e_4 + 0\cdot e_5 + 0\cdot e_6 + 0\cdot e_7$
$I3_{(0,i)} = 0$
$\Diamond$ **:**Let $m = 1$; $\forall \quad 1 \le j \le 7$ and $\forall \quad 1 \le i \le m$, with $m = m + 1$
$I1_{(j,i+16)} = I_{(j+19,i)}\cdot e_0 + 0\cdot e_1 + 0\cdot e_2 + 0\cdot e_3 + 0\cdot e_4 + 0\cdot e_5 + 0\cdot e_6 + 0\cdot e_7$
$\Diamond$ **:** Let $m = 6$; $\forall \quad 8 \le j \le 12$ and $\forall \quad 1 \le i \le m$, with $m = m - 1$
$I1_{(j,i)} = I_{(j+19,i)}\cdot e_0 + 0\cdot e_1 + 0\cdot e_2 + 0\cdot e_3 + 0\cdot e_4 + 0\cdot e_5 + 0\cdot e_6 + 0\cdot e_7$

Section 8 $\lhd$ **:** Let $n = 7$, $m = 8$; $\forall \quad 9 \le j \le 13$ and $\forall \quad n \le i \le m$, with $n = n - 1$ while $m = m + 1$

$I1_{(j,i)} = I_{(j+17,i)}\cdot e_0 + 0\cdot e_1 + 0\cdot e_2 + 0\cdot e_3 + 0\cdot e_4 + F_{(16-j,16-j-i)}\cdot e_5 + F_{(16-j,16-j-i)}\cdot e_6 + F_{(16-j,16-j-i)}\cdot e_7$

Section 9 The section is divided into six subsections.
▶ **:** $I1_{(14,0)} = 0 \cdot e_0 + 0 \cdot e_1 + 0 \cdot e_2 + 0 \cdot e_3 + 0 \cdot e_4 + F_{(0,0)} \cdot e_5 + F_{(0,0)} \cdot e_6 + F_{(0,0)} \cdot e_7$
△ **:** $\forall \quad 1 \leq i \leq 14,$

$$I1_{(14,i)} = 0 \cdot e_0 + 0 \cdot e_1 + 0 \cdot e_2 + 0 \cdot e_3 + 0 \cdot e_4 + F_{(0,i)} \cdot e_5 + F_{(0,i)} \cdot e_6 + F_{(0,i)} \cdot e_7$$

⊘ **:** $I1_{(14,15)} = 0 \cdot e_0 + Z_{(15,0)} \cdot e_1 + Z_{(15,0)} \cdot e_2 + Z_{(15,0)} \cdot e_3 + Z_{(15,0)} \cdot e_4 + F_{(0,15)} \cdot e_5 + F_{(0,15)} \cdot e_6 + F_{(0,15)} \cdot e_7$
⊗ **:** $I1_{(14,16)} = 0 \cdot e_0 + Z_{(15,1)} \cdot e_1 + Z_{(15,1)} \cdot e_2 + Z_{(15,1)} \cdot e_3 + Z_{(15,1)} \cdot e_4 + F_{(0,15)} \cdot e_5 + F_{(0,15)} \cdot e_6 + F_{(0,15)} \cdot e_7$
▲ **:** $\forall \quad 1 \leq i \leq 14$

$$I1_{(14,i+16)} = 0 \cdot e_0 + Z_{(15,i+1)} \cdot e_1 + Z_{(15,i+1)} \cdot e_2 + Z_{(15,i+1)} \cdot e_3 + Z_{(15,i+1)} \cdot e_4 + 0 \cdot e_5 + 0 \cdot e_6 + F_{(0,15)} \cdot e_7$$

⊖ **:** $I1_{(14,31)} = 0 \cdot e_0 + 0 \cdot e_1 + 0 \cdot e_2 + Z_{(15,15)} \cdot e_3 + Z_{(15,15)} \cdot e_4 + 0 \cdot e_5 + 0 \cdot e_6 + F_{(0,15)} \cdot e_7$

Section 10 $\forall \quad 0 \leq i \leq 31$; with $n = n - 1$ while $m = m + 1$
$I1_{(j,i)} = 0 \cdot e_0 + 0 \cdot e_1 + 0 \cdot e_2 + 0 \cdot e_3 + W(i) \cdot e_4 + 0 \cdot e_5 + 0 \cdot e_6 + 0 \cdot e_7$

3 Experimental Results and Analysis

The Reconfigurable Multiple Operation Array with all necessary control logic and the Carry Unit were described using VHDL, synthesized and proved with ISE 5.2i Xilinx environment [11], for the VIRTEX II PRO xc2vp100-7ff1696 FPGA device. Additionally, all operations such us: unsigned multiplier, signed magnitude multiplier, two's complement multiplier, rectangular SAD unit (half of the rectangular array), and MAC unit were implemented individually without the overhead logic. Furthermore, previous presented units like the Universal SAD-Multiplier array (U-SAD-M) [2] for integer numbers and Baugh and Wooley (B&W) signed multiplier [1] were synthesized with the same tool in order to have more comparison parameters with our new proposal. Table 2 summarizes the performance in terms of time delay of those structures.

The additional logic introduced into the core array reduces the performance of the functionalities, as can be seen in table 2. The extra delay in terms of time is between 12 % for a 16 bits MAC operand in the proposed unit, and up to 50 % for a 32 bits MAC over the previous single functionality units like Baugh-Wooley or a simple integer U-SAD-M unit. This extra time delay diminishes to 27 % when the fast carry logic provided into the FPGAs Xilinx is used [9] in the final adder. The additional extra delays of the proposed unit compared with the previous ones are due two principal factors: the first one is related to the multiplexor used to feed the data into the input $I1$, which presents a constant delay for all the logic blocks, and the extra multiplexor introduced in the array to separate logically the right and left sides, given the possibility of propagating or inhibiting the carry and also force a Hot One or a zero for SAD processing.
An acceleration into the processing can be achieved using a parallel reduction tree. Instead of using a regular array, a Wallace tree [8] can be implemented in

Table 2. Multiple Operation Array Unit and related units - time delay

Unit	Logic Delay (ns)	Wire Delay (ns)	Total Delay (ns)
SAD ‡	13.184	12.006	25.191
Unsigned Multiplier ‡	14.589	14.639	29.282
Two's I ‡	12.595	15.564	28.159
Two's F ‡	15.153	16.825	31.978
Baugh&Wooley ‡	15.555	15.826	31.381
U-SAD-M ‡	15.877	15.741	31.618
U-SAD-M †	14.311	9.568	23.879
MAC ‡	15.062	19.064	34.662
Our Proposal ‡	21.351	26.040	47.391
Our Proposal-16 MAC ‡	16.521	19.065	35.586
Our Proposal †	15.311	15.127	30.438
Carry Unit	2.576	3.338	5.914

RCA final adder: ‡ : LUT based ; † : Based on Xilinx Fast Carry Logic.[9]

order to accelerate the performance of the operations. Considering that $n(h) = \llcorner 3n(h-1)/2 \lrcorner$ quantifies the number of necessary levels h of (3:2)counters for the reduction of n input bits, we estimate that a logic delay has been reduced in 3.22 ns and cut 4.65 ns off in routing delay, accelerating the processing in 7.87ns. This amount of time represents an improvement of a 25.87 % in the total array delay.

Concerning the silicon used by the Reconfigurable Multiple Operation Unit and the other structures, depicted on table 3, the overhead in terms of hardware of the presented unit is considerable and is the paid price for its multi-functional characteristic. Nevertheless, if the eight units are implemented separately we will need two times the hardware resources and we will have one third of additional bandwidth needs in the worst case scenario. We should also consider that 6/8 of the basic logic block array are shared by the eight operations making this chosen operations a good candidates for it's implementation with partial reconfiguration paradigm, based on differences of the functional units.

4 Conclusions

We have presented a novel Reconfigurable Multiple Operation Array organization. The proposed unit can be implemented on a VLSI intended to be used as a run time configurable unit, and it can also be used in a reconfigurable technology as a run time reconfigurable unit. The whole array is configured using multiplexors, which can be replaced with faster connections on a partially reconfigurable environment. Several units are been coded and synthesized to have a wide comparison environment, furthermore, a brief analysis of the obtained results in terms of area used and time delay are presented given a maximum work frequency of 27.5 MHz for the calculus for a 4x4 SAD macro-block and a 32.85 MHz for MAC and the other multiplier operations in a VIRTEX II PRO device using a 3% of the available slices of the chosen FPGA.

Table 3. Multiple Operation Array Unit and related units - hardware Use

Unit	# Slices	# LUTs	# IOBs
SAD ‡	242	421	272
Unsigned Multiplier ‡	300	524	64
Two's I ‡	294	511	64
Two's F ‡	443	770	64
Baugh&Wooley ‡	330	574	65
U-SAD-M ‡	686	1198	322
U-SAD-M †	658	1170	322
MAC ‡	358	622	96
Our Proposal ‡	1373	2492	643
Our Proposal-16 MAC ‡	1360	2465	643
Our Proposal †	1354	2458	643
Carry Unit	35	61	64

RCA final adder: ‡ : LUT based ; † : Based on Xilinx Fast Carry Logic[9].

References

1. C. Baugh and B. Wooley. A two's complement parallel array multiplication algorithm. *IEEE, Transactions on Computers*, pages 1045–1047, Dic. 1973.
2. H. Calderon and S. Vassilidis. Reconfigurable universal sad-multiplier array. *Accepted for publication in: Proccedings of ACM international conference - Computer Frontiers*, May. 2005.
3. D. Guevorkian, A. Launiainen, P. Liuha, and V. Lappalainen. Architectures for the sum of absolute differences operation. *IEEE Workshop on Signal Processing Systems (SPIS'02)*, Oct. 2002.
4. P. Kuhn. Fast mpeg-4 motion estimation: Processor based and flexible vlsi implementations. *Journal of VLSI Signal Processing*, pages 67–92, 1999.
5. S. Vassiliadis, E. Hakkennes, S. Wong, and G. Pechanek. The sum-absolute-difference motion estimation accelerator. *Proceedings of Euromicro Conference, 24th*, pages 559–566, Aug. 1998.
6. S. Vassiliadis, E. Schwarz, and M. Putrino. Quasi-universal vlsi multiplier with signed digit arithmetic. *Proceedings of the 1987 IEEE,Southern Tier Technical Conference*, pages 1–10, Apr. 1987.
7. S. Vassiliadis, S. Wong, G. Gaydadjiev, K. Bertels, G. Kuzmanov, and E. Panainte. The molen polymorphic processor. *IEEE Transactions on Computers*, pages 1363 – 1375, Nov. 2004.
8. C. S. Wallace. A suggestion for a fast multiplier. *IEEE, Transactions on Electronic Computers*, pages 14 – 17, Feb 1964.
9. Xilinx. Virtex ii pro platform fpga handbook. Oct. 2002.
10. Xilinx. Two flows for partial reconfiguration: Module based or difference based. *Application Note:Virtex, Virtex-E, Virtex-II, Virtex II Pro Families,XAPP290*, pages 1 – 28, Nov. 2003.
11. Xilinx. The xilinx software manuals, xilinx 5.2i. *http://www.xilinx.com/support/sw_manuals/xilinx5/index.htm*, 2003.
12. N. Yaday, M. Schulte, and J. Glossner. Parallel saturating fractional arithmetic units. *Proccedings of the Ninth great lakes Symposium on VLSI*, Mar. 1999.