On the use of Signed Digit Arithmetic for the new 6-Inputs LUT based FPGAs

G.C. Cardarilli†, S. Pontarelli‡, M. Re†, A. Salsano†
{pontarelli, salsano}@ing.uniroma2.it,
{marco.re, g.cardarilli}@ieee.org
†University of Rome "Tor Vergata", Department of Electronic Engineering
Via del Politecnico 1, 00191, Rome, ITALY
‡(ASI) Italian Space Agency, Viale Liegi, 26 00198 Rome, ITALY

***Abstract*— In this paper the use of Signed Digit (SD) Arithmetic to better exploit some of the architectural characteristic of the last generation FPGAs is presented. The implementation of Radix-4 SD adders, multipliers and Finite Impulse Response (FIR) filters has been carried out to demonstrate that the use of this number system representation optimally fits the 6-input LUT Logic Elements (LEs) of the newest FPGAs architectures. Comparisons of implementations of the same circuits by using 4-input LUT and 6-input LUT based FPGAs have been carried out showing that Radix-4 SD arithmetic is very efficiently implemented in the last generation FPGAS.**

I. Introduction

The silicon integrated circuits trend has been characterized by a steady reduction in the feature size combined with a steady rise in density and speed [1]. This trend made possible the release, in 1985, of the first FPGA, the Xilinx XC2064 chip, with its 1,000 gates of complexity [2]. In the following twenty years, the FPGAs architectures evolved in a very fast way. The major evolution has been related to the structure of the interconnect, the topology of the LE, and the introduction of full custom processing elements such as multipliers, hardware processor cores and high speed multi standard I/O blocks. On the other hand, when the FPGA architecture change, also the synthesis algorithms must be modified in order to guarantee an optimum mapping on the available resources. One of the major changes in the architecture of the last generation FPGAs is the use of 6-input LUTs as LEs core [3]. In this paper it is shown how to exploit this characteristic by using Radix-4 SD representation for the implementation of different arithmetic operators and basic DSP blocks. The implementation results show that by using 6-input LUTs a more efficient implementation of Radix-4 SD arithmetic is obtained. This means smaller reconfiguration time (in case of partial reconfiguration) and a more efficient use of the interconnect resources. The paper is organized as follows: in Section II a background on the SD arithmetic representation is given, while the characteristic of 6-input LUT based implementations of basic SD arithmetic operators are discussed in Section III. In Section IV implementation results for Radix-4 SD adders, multipliers and FIR filters are illustrated and compared with the same results obtained by using the standard two complement binary representation (TCS) on the same hardware platform. The conclusions are drawn in Section IV.

II. Signed Digit Representation Background

The general theory of the SD representation is illustrated in many books ([4], [5]). In this section its basic elements are shown. In a Radix-r SD representation, a number x is represented by the equation

$$x = \sum_{i=0}^{n-1} x_i r^i \tag{1}$$

Where, differently from the canonical r radix polynomial representation, the digit set is $x_i \in \{-a, \ldots, -1, 0, 1, \ldots, a\}$, with $\left\lceil \frac{r-1}{2} \right\rceil \leq a \leq r-1$.

The original motivation for introducing the SD representation was to eliminate carry propagation in addition and subtraction [4]. In fact, given two SD operands x and y, the addition is obtained by

$$w_i = x_i + y_i - rc_i \tag{2}$$

978-1-4244-2182-4/08/$25.00 ©2008 IEEE.

$$z_i = w_i + c_{i-1} \quad (3)$$

where

$$c_i = \begin{cases} 1\, if\, (x_i + y_i) \geq a \\ -1\, if\, (x_i + y_i) \leq -a \\ 0\, if\, |x_i + y_i| < a \end{cases} \quad (4)$$

being $w_i \in \{-a+1, \ldots, -1, 0, 1, \ldots, a-1\}$ an auxiliary variable. A carry-free adder is implemented in SD by using a block (ADD1) for the implementation of equations (2) and (4) and a block (ADD2) for equation (3). The complete adder is obtained by connecting ADD1 and ADD2 as shown in Fig. 1.

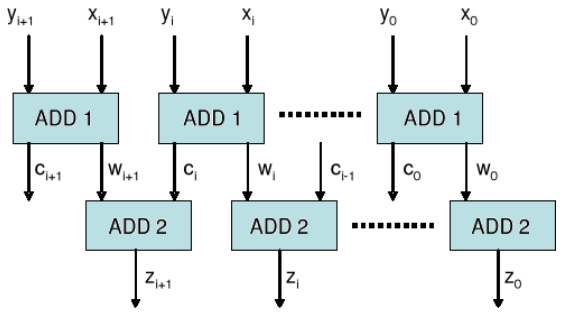

Fig. 1. Signed Digit Adder Architecture

III. Radix-4 Implementation of Basic Arithmetic Operators

In Radix-4 SD number system, the input digits x_i and y_i and the outputs digits w_i belong to the range [-3,3] and consequently three bits are used for their representation, while the c_i belong to the range [-1,1]. Therefore the computation of w_i requires three six input logic functions, while the computation of c_i requires two logic functions. The entire ADD1 block is represented by five six inputs logic functions and therefore, it can be implemented by using five 6-input LUTs. For the ADD2 block, the number of inputs bits is five (three bits for w_i and two for c_{i-1}) while the number of outputs bits is 3. This block is implemented by using three 5-input LUTs in parallel. It is important to mention that in the Xilinx Virtex V FPGAs a 6-input LUT can be also configured as a dual 5-input LUT enhancing the utilization of this block. Consequently, the ADD2 block is implemented by a 6-input LUT configured as a dual 5-input LUT and a 6-input LUT. The entire SD adder is therefore implemented by two levels of LUTs (the working frequency is 550 MHz, i.e. the maximum frequency of Xilinx Virtex V FPGAs [1]). The number of 6-input LUTs to implement the SD adder is $7N$, where N is the number of signed digits used for the operands representation.

The shift and add implementation of the multiplication by a constant (i.e. $y = k \cdot x$) has been extended to the signed digit representation in [5]. The multiplication by the radix r is performed by simply shifting the digits of x, while the multiplication by $-r$ is obtained by shifting and inverting its digits. To minimize the number of adders required for the implementation of the constant multiplier, the number of non-zero digit must be minimized by factorizing k (see [6] for details). In the following example the implementation of a Radix-4 constant multiplier with $k = 75$ is shown. The direct implementation is obtained starting from the following factorization $75 \cdot x = (64 + 16 - 4 - 1) \cdot x = 64x + 16x - 4x - x$, by using three adders. On the other hand, if k is factorized as $75 = 15 \cdot 5 = (16 - 1) \cdot (4 + 1)$, and by introducing the auxiliary variable b the multiplication requires only 2 adders, in fact

$$b = 5 \cdot x = 4x + x$$
$$c = 15 \cdot b = 16b - b$$

In Fig. 2 the direct and factorized implementations for the constant multiplier are shown.

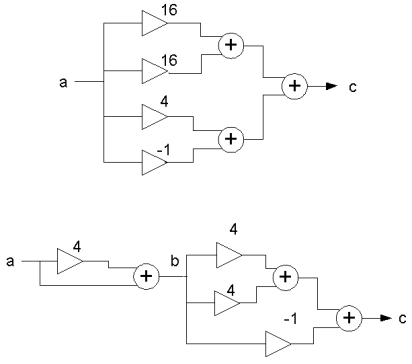

Fig. 2. Example of a Radix-4 Signed Digit constant multiplier ($k = 75$)

IV. Results and Comparisons

In this Section the results of the implementation of Radix-4 SD arithmetic basic operators and DSP blocks using as a target device a Virtex V FPGA [3] are presented. To compare these results with respect to those obtained by using older FPGAs families, the same implementations have been carried out also

for Virtex IV and Virtex II FPGAs (both based on 4-input LUT LE). The implementation of the same arithmetic blocks, by using TCS representation, has been also carried out. To ensure a fair comparison of the synthesis results, the placement and routing parameters have been set to obtain the best results in terms of speed. The following blocks have been implemented and analyzed in the following subsections

- SD adders with different number of digits,
- Constant multipliers with different constant factors k,
- A 16 taps Low-Pass FIR filter.

A. SD Adder Comparisons

Different Radix-4 SD adders have been implemented by using the ADD1 and ADD2 blocks as described in Section II. In particular, adders with 8, 16, 32 and 64 digits, corresponding to a TCS representation of 16, 32, 64, 128 bits have been implemented. The implementation results for the Virtex V FPGA are illustrated in Table I, in conjunction with the results obtained for an high speed TCS implementation. In this case the architectural choice has been left to the synthesizer that has been configured to obtain a maximum speed implementation. The maximum working frequency for the SD and TCS adders versus the operands wordlentgth is shown in Fig. 3.

N. bit Digit	Freq.(SD) [MHz]	Freq.(TCS) [MHz]
16 bits 8 digits	580	425
32 bits 16 digits	580	425
64 bits 32 digits	580	334
128 bits 64 digits	580	232

TABLE I

MAXIMUM FREQUENCY FOR SD AND TCS ADDERS (6-LUT FPGA)

As shown in Table I the SD adder maximum frequency is independent of the operand wordlength. Since the maximum frequency matches the maximum clock frequency allowed for the Virtex V devices [3], these is the best architectural solution for fast adder implementations using this device. The TCS speed performance are shown in the second column of Tab. 1 and exhibits a more complex behavior

- the adder is slower with respect to the SD also for very small wordlengths

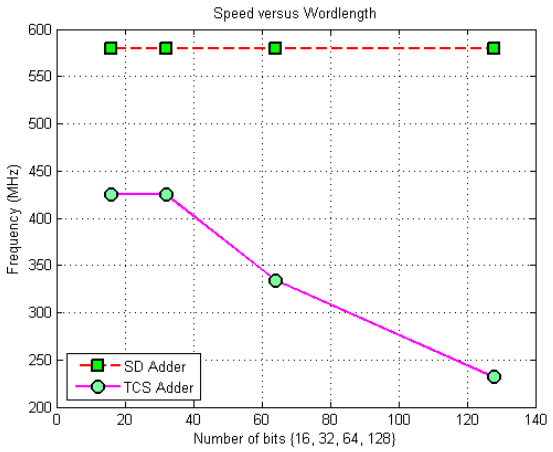

Fig. 3. Maximum frequency for SD and TCS adders versus the operands wordlength

- adders with wordlengths up to 32 bits achieve the same maximum frequency. This is related to the architecture of the Virtex V FPGAs LEs that uses fast carry chains from LE to LE.
- For wordlengths greater than 32 bits the maximum frequency decreases.

In Table II, the speed gain for a SD adder with respect to the TCS counterpart is shown. This comparison has been carried out for different Xilinx FPGA families. Also for the Virtex II the SD adders are faster than those based on TCS arithmetic but in the Virtex 5 generation this gain increases up to 40%. This gain is maintained even for small wordlength SD adders (for example in the 8 digits case). In Table III the area used for the implementation of the two adder architectures for different FPGAs families is shown. The area has been measured in terms of number of slices and the last column shows that the use of 6-input LUT based FPGAs permits to implement Radix-4 SD arithmetic by consuming less resources.

FPGA Family	Speed Gain [%]
Virtex II (4-LUT)	10
Virtex IV (4-LUT)	20
Virtex V (6-LUT)	40

TABLE II

SPEED GAIN OF A A 8 DIGIT SD ADDER FOR DIFFERENT FPGAS FAMILIES

B. Constant Multiplication Comparisons

In this section the implementation of the constant multiplier is illustrated. In particular Table IV shows the result for the two implementations by varying

FPGA Family	Area(SD)	Area(TCS)	Area(SD)/Area(TCS)
Virtex 5	448	238	1.88
Virtex 4	352	136	2.58
Virtex II	586	238	2.46

TABLE III

ADDER AREA (N. OF SLICES) FOR THE SD AND THE TCS ADDER (DIFFERENT FPGAS)

the value of the constant factor k. Also in this case the SD constant multiplier is faster and its maximum frequency is quite independent from the value of k.

k	Freq.(SD) [MHz]	Freq.(TCS)[MHz]
257	318	294
4389	321	294
8546	337	302
15189	275	186

TABLE IV

MAXIMUM SPEED FOR THE SD CONSTANT MULTIPLIER AND THE TCS MULTIPLIER (6-LUT FPGA IMPLEMENTATION)

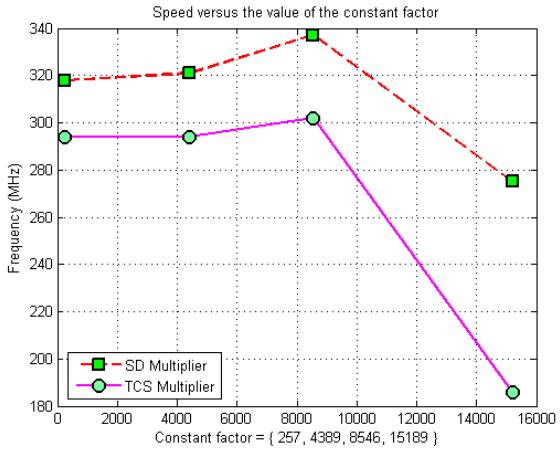

Fig. 4. Maximum frequency as a function of the value of the constant factor k

C. FIR filter comparison

In this experiment, a 16 taps low pass FIR filter in fixed point arithmetic has been implemented. The number of bits used for the input samples and the fixed coefficients is 8, while the number of bits at the multiplier output is 16 (no truncation after multiplication). The filter has been implemented by using SD and TCS arithmetic on a Virtex-5 FPGA. The TCS filter clock frequency is 200 MHz while the SD based filter can run at 293 MHz, with 50% of speed increment. The SD filter uses more LUTs, this depends both on the greater complexity required by the SD implementation and on the in/out conversions blocks (from the TCS representation to the SD one and vice versa). The conversion overhead become negligible as the number of filter taps increments. In fact, in real cases, FIR filters need a high number of taps in order to obtain sufficiently sharp frequency responses.

V. CONCLUSIONS

In this paper the use of SD arithmetic to better exploit some of the architectural characteristic of the last generation FPGAs is presented. In particular Radix-4 SD adders, multipliers and FIR filters have been implemented to demonstrate how the use of SD optimally fits the 6-input LUT based LE of the newest FPGAs architectures (Xilinx Virtex 5). These implementations have been compared in terms of speed and number of LE with those obtained by using 4-input LUT based FPGAs. These comparisons show that the increasing complexity and flexibility of the FPGAs logic elements can reduce the resource utilization, making the SD representation suitable for these architectures if fast processing is needed.

ACKNOWLEDGMENT

The authors would like to thank the University of Rome Tor Vergata Department of Electronics and the Denmark Technical University Department of Informatics for the support. The research work has been also supported by the Otto Mønteds Fond in the context of a Visiting Professor Sponsorship for the years 2007-2008.

REFERENCES

[1] "2007 International Technology Roadmap for Semiconductors", http://public.itrs.net/.
[2] http://www.xilinx.com/company/history.htm#begin
[3] Virtex-5 Family Overview LX, LXT, and SXT Platforms
[4] A. Avizienis, "Signed-Digit Number Representations for Fast Parallel Arithmetic" IRE Trans. Electronic Computers, vol. 10, pp. 389-400, 1961.
[5] M. Ercegovac, T. Lang,"Digital Arithmetic", Morgan Kaufman, 2004.
[6] A.G. Dempster, M.D. Macleod, "Constant integer multiplication using minimum adders" Circuits, Devices and Systems, IEE Proceedings, Volume 141, Issue 5, Oct. 1994 Page(s):407 - 413