A High Performance Multiply-Accumulate Unit with Double Carry-Save Scheme for 6-Input LUT Based Reconfigurable Systems

Ugur Cini[1], Olcay Kurt[2]

[1] Dept. of Electrical & Electronics Engineering
Trakya University, Turkey
ugurcini@trakya.edu.tr
[2] Institute of Natural and Applied Sciences
Trakya University, Turkey
olcaykurt@trakya.edu.tr

Abstract

Redundant number systems provide carry-propagation free arithmetic, so that faster arithmetic circuits can be designed. In this work, an alternative redundant arithmetic based fused multiply-accumulate (MAC) unit is designed especially suitable for 6-input look-up-table (LUT) based FPGAs. By employing only (6, 3) counters in the partial product reduction and accumulate operations, least amount of logic depth is provided which results as high performance without any pipeline requirement in the system. The proposed MAC unit has 16x16 input with sign extended 40-bit output. The MAC unit is compared to conventional redundant carry-save and various standard MAC architectures. The proposed structure provides highest performance among the structures that have been compared.

1. Introduction

Multiply-accumulate (MAC) units are extensively used in mathematical operations such as matrix multiplication, convolution, filtering operations etc. Therefore performance analysis of MAC units is critical for improvement of overall performance of a digital system [1-4]. Reconfigurable systems, i.e. Field Programmable Gate Arrays (FPGAs) are extensively used in signal processing applications. In this work, a high throughput and low latency MAC unit is designed for high performance 6-input look-up table (LUT) based FPGAs. More than a decade, high performance FPGAs, such as Stratix Families of Altera™ and Virtex Families of Xilinx™ are built using 6-input LUT elements [5,6]. Higher input LUT structures results in faster logic since more complex building blocks can be mapped into less logic cascades.

Redundant number systems are popular because of having constant delay for the addition operations independent of digit size of the input operands [7-9]. In general, there are two ways of building carry-free arithmetic. One is signed-digit arithmetic; the other is carry-save arithmetic. There are numerous hardware implementations employing carry-free arithmetic for fast digital arithmetic operations [7-9]. Carry-save arithmetic operations have similar performance compared to signed-digit systems since both of them are based on redundant arithmetic operations. Moreover, carry-save arithmetic can be built using conventional adder elements. Carry-save adder trees are most popular for partial product reduction in multiplication [9, 10]. Moreover, they can also be used for redundant arithmetic structures to develop carry-free arithmetic. In this work, a carry-save operation based MAC unit is generated employing double carry-save encoding at the output so that 6-input LUT structures are fully utilized for best performance. In conventional carry-save arithmetic, the resulting redundant output is based on one carry, and one sum bit for each digit, i.e. each digit has the value set of (0, 1, 2) [7, 11]. In the presented implementation, each output digit is represented as three bits, i.e., each digit has the value set of (0, 1, 2, 3). The proposed representation is advantageous for the 6-input LUT based FPGAs, since the arithmetic operations are based on (6, 3) counters for the double-carry save arithmetic. Whenever 6-input LUT devices are considered, (6, 3) counters are recommended for the partial product reduction, since (6, 3) counters have a single atomic delay for the 6-input LUT based systems [12]. The reason is that, each parallel input have 6 inputs in a (6, 3) counter, which can be synthesized in a single LUT delay. Another well-known partial product reduction technique is employing (4, 2) compressors, which is one of the most efficient VLSI design blocks. However, in 6-input LUT based designs, (4, 2) compressors cannot be synthesized in a single level cell. An alternative partial product reduction technique is signed–digit arithmetic [1, 3]. In signed-digit addition, two signed-digit partial products are reduced to a single one, i.e. halving the partial product count, similar to (6, 3) counters and (4, 2) compressors, which also halve the partial products. However, the signed-digit implementation needs more than single LUT delay, since the addition operation is composed of two cascaded operations as interim sum and finalized sum [1, 3, 13].

In this work, (6, 3) counters are both used in the partial product reduction together with accumulate operation. As a result, the carry-propagation does not occur in the multiply-add operations. It should also be noted that, in this work, double-carry save encoding for MAC unit generation using (6, 3) counters is proposed for the first time in the literature. The output is always kept in double carry save format, i.e. the result is kept in three output bits for each digit. As a result, redundant MAC output is generated without any pipeline stage with highest speed compared to any of the non-pipelined MAC unit architecture. Whenever conventional binary result, i.e. conventional 2's complement output is required, a three operand adder is used for the computation of the result.

However, the redundant to standard 2's complement binary conversion is only needed after all multiply-add operations are finished. As an example, in a 100-tap FIR filter implementation, redundant to binary conversion is needed once in 100 multiply-add operation. The proposed system provides a high throughput with minimum delay of the system.

The designed MAC unit is synthesized for Altera^{TM}'s Stratix-III FPGA and compared to conventional MAC units built using various software multipliers, hardware multipliers, pipelined multipliers etc. The hardware cost, speed and throughput metrics and dynamic power requirements are analyzed.

2. Redundant MAC Architecture

In the proposed MAC unit, after the partial product generation using well-known radix-4 modified Booth encoding, multiply and add operations are unified under the (6, 3) counter tree arrays. Counters are generally used for the reduction of number of the addition operands. Fig. 1(a) and 1(b) show the single bit and multi-bit (6, 3) counter schemes, respectively. In Fig. 1(a), six of the input operands are added up to have output values between 0 and 6, i.e. $(000)_2$ – $(110)_2$. In Fig. 1(b), six of eight-bit binary inputs operands to be added are fed into (6, 3) array vertically and the number of the input operands to be added are reduced from six to three. The advantages of the usage of (6, 3) counters and efficient implementation techniques for the 6-input LUT based FPGAs are explained in [12]. Since each (6, 3) counter for each bit array requires 6-inputs, any (6, 3) counter array can be synthesized within a single LUT delay.

Table 1. Various reduction operator metrics

	Binary signed-digit		(4,2) compressor		(6,3) counter	
Bit-width	Delay (ns)	Area (LUT)	Delay (ns)	Area (LUT)	Delay (ns)	Area (LUT)
16-bit	2,77	80	1,84	48	1,15	48
24-bit	3,45	120	1,91	72	1,10	72
32-bit	3,27	160	1,95	96	1,15	96
64-bit	3,19	320	1,93	192	1,15	192

Generally, a carry-save reduction tree is not preferred in the partial product reduction in FPGA systems, since they require more area than carry-propagate schemes [15, 16]. Moreover, fast carry chains provide efficient performance which makes carry-propagate schemes more popular [15, 16] for general applications. However, carry-propagate schemes have linear dependency on bit-width, and, as the bit-width grows, the carry-propagate scheme becomes inefficient. Carry-save scheme doubles area requirement compared do carry-propagate schemes [16]. If area requirement is not main concern, redundant reduction trees, such as (4, 2) compressors, binary-signed digit adder trees, counter trees can be used in FPGA multipliers. Table 1 shows performance characteristics and area requirements of various multi-operand reduction operators. Among various reduction operators, (6, 3) counters give best performance for the 6-input LUT based FPGAs as shown in Table 1. According to Table 1, (4, 2) compressors and (6, 3) counters require same amount of resources where (6, 3) counters are much faster. Binary signed-digit scheme require more area and the delay of the operator is more compared to (4, 2) compressor and (6, 3) counter array. Among the compared redundant operators; (6, 3) counter array, (4, 2) reduction array and binary signed digit operators all halve the input operands, i.e., all of them have similar functionality for arithmetic efficiency regardless of performance concern.

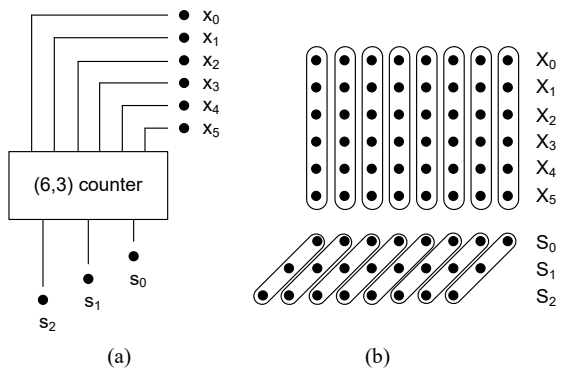

Fig. 2. (6, 3) counter: (a) Single-bit; (b) multi-bit representation with vertical line reduction

In [11], a regular carry-save scheme is proposed suitable for multiply-add operations without carry propagation. The redundant number format in [11] consists of two components, where in the proposed double carry-save scheme the redundant output consists of three components. A MAC unit consists of a multiplier path, a registered accumulate output. For the multiplication of the proposed architecture, standard Booth encoding scheme is employed with sign extension to 40-bit outputs. The sign extension is made due to the fact that the

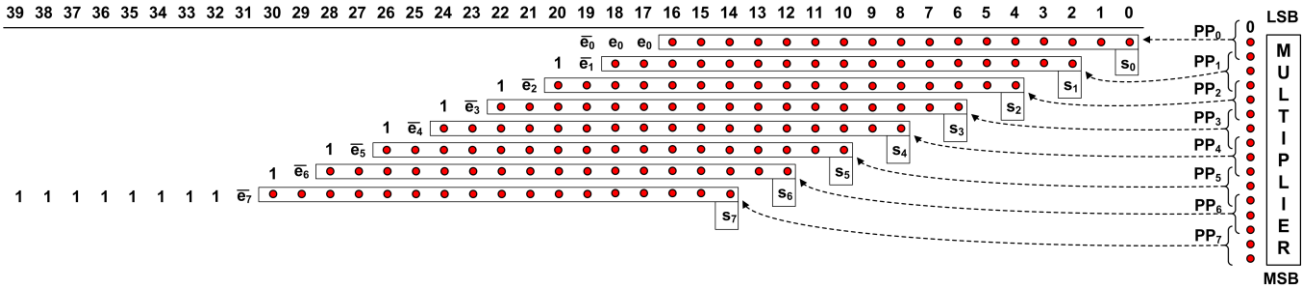

Fig. 2. Modified Booth encoding scheme with sign extension to 40-bits

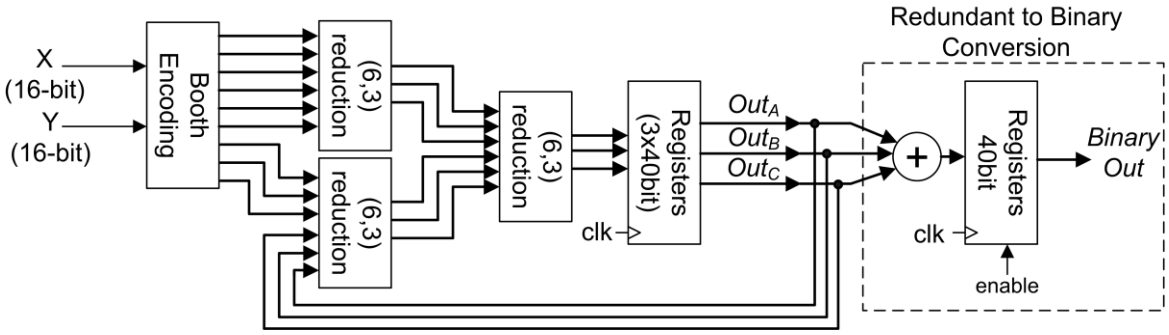

Fig. 3. Proposed multiply-accumulate architecture

accumulate output should not overflow after multiple multiply-accumulate operations. The radix-4 modified Booth encoding with sign extension is depicted in Fig. 2. Here, each line represents generated partial product. The partial products are generated according to the rules shown in Table 2. Here, each partial product is generated according to the multiplier bits (x_{2i+1} x_{2i+1} x_{2i+1}). The e_i in each line is equivalent to the sign bit of the related partial product. s_i in each line is 1 if the sign of Y is inverted, it is 0 if the sign of Y is positive, which is generated according to Table 2.The last partial product of the Booth encoding stage (i.e. 8th line in Booth encoding output) is padded with 1's to extend the multiplier output from 16x16 = 32 bits to 40 bits in order not to overflow the output after multiple multiply-accumulate operations. 40-bit output with 32-bit+8-bit sign extension provides versatile structure so that overflow does not exist for recursive multiply-add operation. The details of the Booth encoding scheme is straightforward and explained in [14].

Although Booth encoding halves the number of partial products, i.e. there exists 8 partial product, s_7 of the last line results as another operand to be added together. As a result, the output of Booth encoding results as 9 operands to be added. Empty lines for the each addition operands are padded with 0's for easy hardware implementation, i.e, there exists 40-bits of 9 rows for the partial product reduction scheme.

The proposed MAC architecture is shown in Fig. 3. Here, the thick lines represent 40-bit operands. The output of the MAC consists as a composition of three outputs, i.e. each digit has the value set of (0, 1, 2 3). For the accumulate operation, the three bit output is fed back to the (6, 3) counter tree. By using the proposed architecture, the critical path for each multiply-add operation consist of one Booth encoding stage and two (6, 3) counter stages. Since both Booth encoding and (6, 3) counter stages can be implemented with a cost of single LUT logic depth, total system critical path is only 3 LUT delays. As a result, a high throughput with minimum register delay can be achieved. As shown in Fig. 3, the redundant multiply accumulate operation does not have an explicit pipeline stage. Even with no pipeline in the operation, the structure operates at high speed as contrast to conventional multiply-accumulate structures. The only clock delay appears for the redundant to normal binary conversion stage which is simply a three operand adder. The redundant to binary conversion stage is shown inside the dotted box in Fig. 3. Normal binary conversion is not required frequently in most of the digital signal processing applications. As an example, for a 100-tap finite impulse response filter (FIR) stage, redundant to binary conversion is needed after every 100 multiply-add operations. The condition is true for other digital signal processing applications such as matrix multiplication, general convolution operations etc.

Table 2. Radix-4 modified booth encoding

x_{2i+1}	x_{2i}	x_{2i-1}	Partial Product	Booth Selector Output
0	0	0	0	0
0	0	1	Y	y_j
0	1	0	Y	y_j
0	1	1	$2Y$	y_{j-1}
1	0	0	$-2Y$	$\overline{y_{j-1}}$
1	0	1	$-Y$	$\overline{y_j}$
1	1	0	$-Y$	$\overline{y_j}$
1	1	1	-0	0

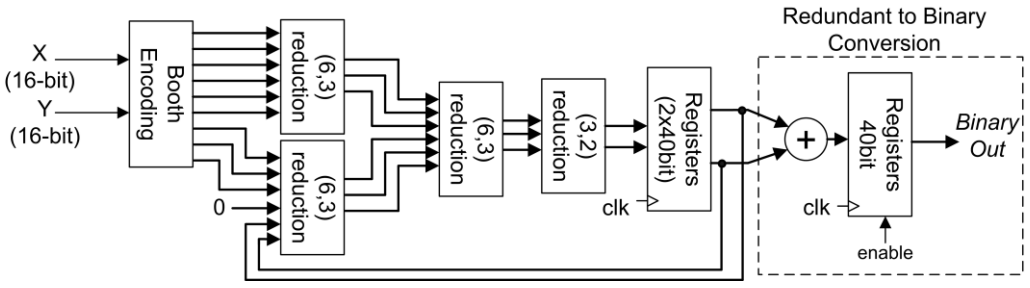

Fig. 4. Redundant MAC architecture based on conventional redundant CSA structure

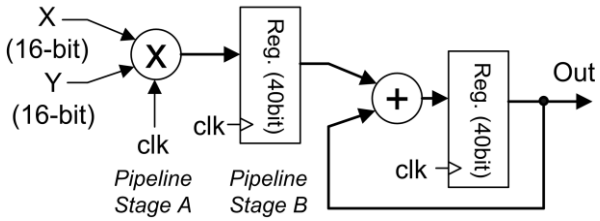

Fig. 5. Conventional MAC unit with various pipeline combinations

3. Results and Discussions

For performance and resource requirement comparison, redundant carry-save-adder (CSA) redundant standard output encoding as (C, S) structure is also designed and compared with the proposed double carry-save structure as depicted in Fig. 4. Here, each output digit is encoded as two bits (C, S) as explained in detail in [11]. The standard redundant CSA scheme is employed by adding an extra (3, 2) reduction stage to our proposed MAC unit which is shown in Fig. 4. Moreover, to compare other standard structures, conventional multiply-accumulate units are also constructed using: MAC units with hardware and software multipliers as depicted in Fig. 5, where software based multipliers are automatically synthesized using Altera's Megafunctions Library. Fig. 5. shows a generic pipelined MAC unit model with various pipeline stages which is built for comparison with the proposed structure. The first pipeline stage appears at the multiplier, which is represented as stage A. The second stage is the pipeline stage between the multiplier and the adder operation. There does not exist a bit-level pipeline stage for the add operator, since fast carry chains provide sufficiently fast operation for 40-bit addition. According to the given pipeline stages, performance metrics are measured and tabulated in Table 3. The first 5 rows are non-redundant standard binary MAC unit implementations, i.e. different configurations of Fig. 5. Resource usage is determined by Adaptive LUT units (ALUT) and register count (Reg.). If the multiplier inside the MAC unit synthesized using logic elements, the structure is named as soft multiplier and, if the multiplier is synthesized using hardwired multiplication units, the structure is named as hardware multiplier.

If hardware multiplier is used in standard configuration, high performance is achieved. However, the proposed structure is still faster. The proposed structure has no pipeline; however, it may require a single extra clock delay for the redundant output to binary conversion, so that the clock delay is represented as (1+1) in Table 3. The benchmark circuits are built by employing shallow pipeline stages and the proposed scheme is faster than pipelined multiply-add units. Hardware multiplier based MAC unit performance is close to the proposed scheme; however, the proposed scheme is still 10% faster. As a result, double carry-save redundant representation provides best performance metrics among the carry-propagate multipliers and hardware multiplier based implementations. Moreover, the proposed structure has 8% better speed performance and requires 18 % less LUT requirement compared to the conventional redundant CSA (Fig. 4) architecture. However, double carry-save encoding requires more registers compared to standard redundant CSA implementation of Fig. 5.

4. Conclusions

In this paper, alternative redundant carry-save arithmetic, i.e. double carry-save encoding based MAC unit is proposed. The designed MAC unit has high throughput with low clock delay. The structure is advantageous since it is free of carry propagation. Since all of the redundant arithmetic systems are free of carry propagation, the system is also compared to another classical carry-fee MAC unit with carry-save output encoding. Among all, the proposed structure shows best throughput with least amount of clock delay. The proposed structure provides lowest logic depth providing fast multiply-add operation in a single stage without pipeline. The system is especially suitable for 6-input LUT based FPGA structures.

Table 3. Comparison of the MAC units

Structure	Resource Usage	Dynamic Power (mW)	Clock Delay	Speed (MHz)
Soft Multiplier (no pipeline)	258 ALUT 72 Reg.	8	1	124
Soft Multiplier 1-level pipeline: (B =1)	259 ALUT 104 Reg.	8	2	160
Soft Multiplier 2-level pipeline: (A =1; B =1)	263 ALUT 190 Reg.	9	3	210
Hardware Multiplier (no pipeline)	2 DSP Block 40 ALUT 40 Reg.	6.8	1	141
Hardware Multiplier 1-level pipeline: (B =1)	2 DSP Block 40 ALUT 80 Reg.	6.9	2	261
Carry-Save MAC Unit (Fig. 4)	493 ALUT 150 Reg.	12	1+1	265
Proposed (Fig. 3)	**418 ALUT 178 Reg.**	**11.5**	**1+1**	**286**

5. References

[1] K. Parhi, VLSI Digital Signal Processing Systems, John Wiley & Sons, 1999.

[2] W. Kamp, A. Bainbridge-Smith, "Multiply Accumulate Unit Optimised for Fast Dot-Product Evaluation", Int. Conf. on Field-Programmable Technology, pp. 349-352, 2007.

[3] X. Huang, W. Liu, B. Wei, "A High Performance CMOS Redundant Binary Multiplication-and-Accumulation (MAC) Unit", IEEE Trans. Circuits & Syst.-I, Vol. 41 No. 1, pp. 33-44, Jan. 1994.

[4] D. Lee, C. Ryu, K. Kwon, W. Choi, "Design and implementation of 16-bit fixed point digital signal processor", Int. SoC Design Conference, Vol.2 pp 61-64, 2008.

[5] Xilinx Inc., Virtex-6 Family Overview, www.xilinx.com, 2012.

[6] Altera Corp., Stratix III Device Handbook, www.altera.com, 2011.

[7] A. F. Gonzalez and P. Mazumder, "Redundant Arithmetic: Algorithms and Implementations," INTEGRATION, the International VLSI Journal, Vol. 30, Dec. 2000, pp. 13-53.

[8] D. S. Phatak, T. Goff, and I. Koren, "Constant-Time Addition and Simultaneous Format Conversion Based on Redundant Binary Representations", IEEE Trans. Computers, Vol. 50, pp. 1267-1278, Nov. 2001.

[9] M. D. Ercegovac, T. Lang, Digital Arithmetic, Morgan Kaufmann, 2003.

[10] B. Parhami, Algorithms and Design Methods for Digital Computer Arithmetic, Oxford University Press, 2012.

[11] T. G. Noll, "Carry-Save Architectures for High-Speed Digital Signal Processing", Journal of VLSI Signal Processing, vol. 3, 121-140, 1991.

[12] Altera Corp., Advanced Synthesis Cookbook, www.altera.com, 2011.

[13] G. C. Cardarilli, S. Pontarelli, M. Re, A. Salsano, "On the use of Signed Digit Arithmetic for the new 6-Inputs LUT based FPGAs", Int. Conf. on Electronics, Circuits and Systems, ICECS'2008, pp. 602-605, 2008.

[14] N. Weste, D. Harris, CMOS VLSI Design: A Circuits and Systems Perspective, 4th Ed., Addison Wesley, 2010.

[15] W. Jamro, K. Wiatr, "FPGA implementation of addition as a part of the convolution", Euromicro Int. Conf., Sept. 2001.

[16] C. D. Moreno, F. J. Quiles, M. A. Ortiz, M. Brox, J. Hormigo, J. Villababa, E. L. Zapata, "Efficient mapping on FPGA of convolution computation based on combined CSA-CPA Accumulator", Int. Conf. on Electronics, Circuits, and Systems, ICECS 2009, pp. 419-422, 2009.