ISSN (Online) 2321-2004
ISSN (Print) 2321-5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 3, Issue 2, February 2015

Fast Multiplication Based on Different Compressors

Shalu George[1], Jinu Isaac Kuruvilla[2]

Student, VLSI and Embedded System, Mangalam College of Engineering, Kottayam, India [1]

Asst. Professor, Electronics and Communication, Mangalam College of Engineering, Kottayam, India [2]

Abstract: In many of digital systems like graphic processors, digital signal processors fast parallel multiplication using adder trees are present. To speed up the computation like addition is very important. This paper presents different approaches to the efficient implementation of compress tree adders on FPGAs. Through a fair comparison we present a proper compressor selection approach to get minimum XOR delay. This paper will help to choose a proper compressor for fast multiplication. This approach is defined in parameterizable HDL code, which makes it compatible with any FPGA family.

Keywords: Redundant adder, Carry Save addition, multi-operand addition.

I. INTRODUCTION

Addition is one of the basic process in large computations. Different types of adders are available for large addition. The delay of addition will determine the overall performance of the system. Most of the time consumed by adder is due to carry propagation. So to reduce the delay some fast adder structures are present which will calculate the result in reduced time. Large addition processes are commonly seen in multipliers for reducing the partial products. Compressors are used to reduce the partial products of multiplication during an addition process.

ADDER STRUCTURES

Adders are used in different aspects. It is generally recognized that most of the time consumed by adders is due to carry propagation, so to reduce the propagation time different binary adder schemes are used each have their own characters, such as area and energy dissipation. So to choose a adder with specific requirement and constraint is important. Here the function of some commonly used adders is given.

A.REDUNDANT ADDERS

In redundant adders there is no carry propagation is required ie, independent of numbers of bits of the adders. The main aim of the redundant adder is to reduce the addition time. But this adder has some disadvantages, the increase of the number of bits needed for representation of a number and some of operations can't be performed in redundant numbers such as magnitude comparison or sign detection.

B.CARRY SAVE ADDERS

Carry-save adder (CSA) and full adder have same circuit, as show in Figure 1.

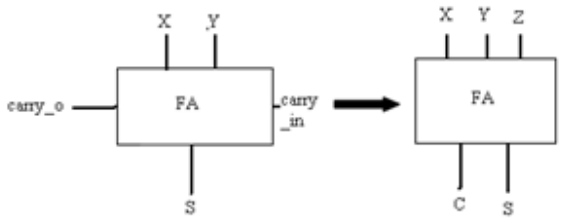

Fig1: Function of Carry-save adder

The carry in signal is considered as an input of the CSA, and the carry out signal is considered as an output of the CSA. Figure 2 show the CSA compute flow and Table 1 will show the CSA working.

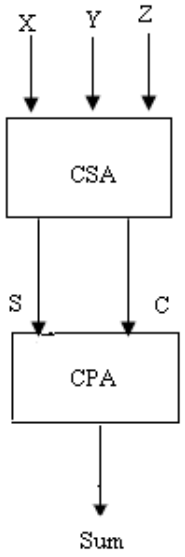

Fig 2: CSA computation

X:	1	0	0	1	1	
Y:	1	1	0	0	1	
Z: +	0	1	0	1	1	
S:	0	0	0	0	1	+
C:	1	1	0	1	1	
sum:	1	1	0	1	1	1

Table 1: CSA Computation

ISSN (Online) 2321-2004
ISSN (Print) 2321-5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 3, Issue 2, February 2015

The computation can be in two steps, first we compute S and C using a CSA, and then we use CPA to compute the total sum. Here the carry signal and the sum signal can be computed independently to get only two *n* -bits numbers. A CPA is used for the last step computation and the carry propagation exists only in the last step.

II. MULTI-OPERAND ADDITION

For adding several operands we have adder tree, such as Wallace tree, Dadda tree, and carry save adder tree and so on. In this paper, carry save adder tree structure is used. There are two methods reduction by rows and reduction by columns, carry save adder tree belong to first method which consist of modules to reduce the rows are called adders and reduce the columns are called counters.

A. CARRY SAVE ADDER TREE

The carry save adder tree can be used to add three operands in two's complement representation and produce a result as the sum of two vectors. A 3-to-2 reduction is called [3:2] adder, and using this tree, we can use a [*p* :2] adder to reduce *p* bit-vectors to 2 bit-vectors using CSAs. We can use [3:2] adders to reduce the rows and get 2 bit vectors. No propagation of the carries is required except on the last two rows which result in a speed up of the computation. From Figure 3, the number of input vectors was reduced by the rows. Finally, we should estimate the numbers of levels of the CSA tree as

$$level \approx \frac{\log\left(\frac{k}{2}\right)}{\log\left(\frac{3}{2}\right)}$$

where *k* is the number of input operands.

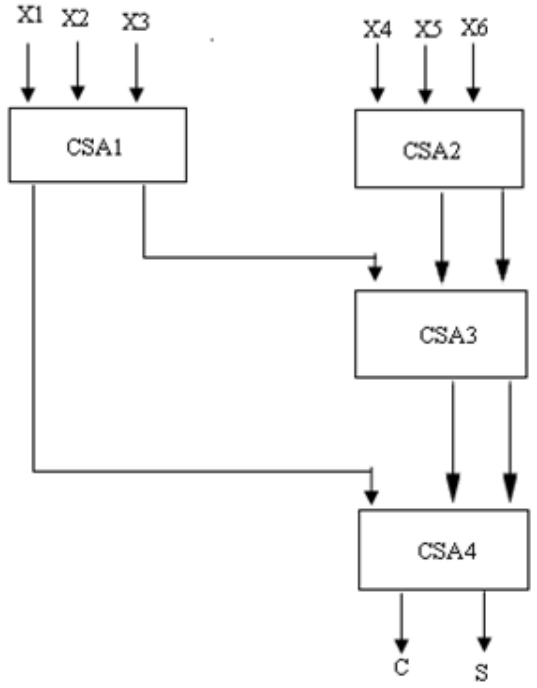

Fig 3: Reduction by rows

B. LINEAR ARRAY STRUCTURE

In the previous approach, specialized carry resources are only used in the design of a single 4:2 compressor, but these resources have not been considered in the design of the whole compressor tree structure. To optimize the use of the carry resources, we propose a compressor tree structure similar to the classic linear array of CSAs. However, in our case, given the two output words of each adder (sum-word and carry-word), only the carry-word is connected from each CSA to the next, whereas the sum words are connected to lower levels of the array. Fig.4 shows an example for a 5:2 compressor tree designed using the proposed linear structure, where all lines are N bit width buses, and carry signal are correctly shifted. For the CSA, we have to distinguish between the regular inputs (A and B) and the carry input (Ci in the figure), whereas the dashed line between the carry input and output represents the fast carry resources.

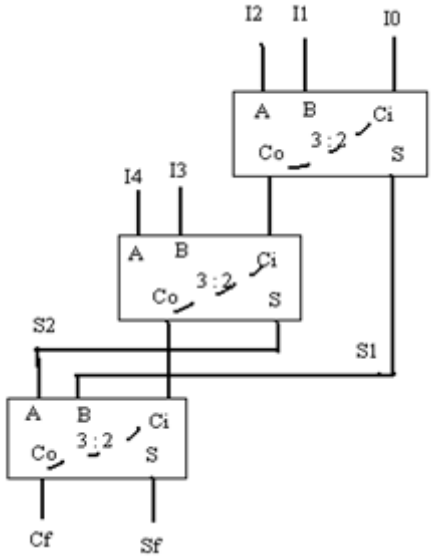

Fig 4: N-bit width 5:2 linear array compressor tree

With the exception of the first CSA, where Ci is used to introduce an input operand, on each CSA Ci is connected to the carry output (Co) of the previous CSA, as shown in Fig. 4.Thus, the whole carry-chain is preserved from the input to the output of the compressor tree (from I0 to Cf). First, the two regular inputs on each CSA are used to add all the input operands (Ii). When all the input operands have been introduced in the array, the partial sum-words (Si) previously generated are then added in order (i.e., the first generated partial sums are added first) as shown in Fig. 4. In this way, we maximize the overlap between propagation through regular signals and carry-chains.

Regarding the area, the implementation of a generic compressor tree based on N bit width CSAs requires Nop _ 2 of these elements (because each CSA eliminates one input signal) Therefore, considering that a CSA could be implemented using the same number of resources as a binary CPA, the proposed linear array, the 4:2 compressor tree, have approximately the same hardware cost.

In relation to the delay analysis, from a classic point of view our compressor tree has Nop-2 levels. This is much

ISSN (Online) 2321-2004
ISSN (Print) 2321-5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 3, Issue 2, February 2015

more than a classic Wallace tree structure and, thus, a longer critical path. Nevertheless, because we are targeting an FPGA implementation, we temporarily assume that there is no delay for the carry-chain path. Under this assumption, the carry signal connections could be eliminated from the critical path analysis and our linear array could be represented as a hypothetical tree. To compute the number of effective time levels (ETL) of this hypothetical tree, each CSA is considered a 2:1 adder, except for the first, which is considered a 3:1 adder. Thus, the first level of adders is formed by the first CSAs (which correspond to partial addition of the input operands). This first ETL produces partial sum-words that are added to a second level of CSAs (together with the last input operand if Nop is even) and so on, in such a way that each ETL of CSAs halves the number of inputs to the next level. Therefore, the total ETLs in this hypothetical tree are

$$L = \lceil \log_2(N_{op} - 1) \rceil,$$

The delay of this tree is approximately L times the delay of a single ETL. However, the delay of the carry-chain is comparatively low, but not null. Let us consider just two global values for the delay: dcarry, which is the delay for the path between the carry inputs (Ci) of two consecutive CSAs and dsum, which is the delay from one general input of a CSA (A or B) to a general input of a directly connected CSA, i.e., the time taken by the data to go from an ETL to the next one. Even under this simplified scenario, it is unfeasible to obtain a general analytical expression for the delay of our compressor tree structure. On each ETL, the propagation through carry-chains and the general paths are overlapped and this overlap depends on multiple factors.

First, it depends on the relative relationship between the values of dcarry and dsum (which is associated with the FPGA family used). Second, it depends on the number of operands that affect both the delay of the carry-chain of each ETL and the internal structure of the hypothetical tree. Even though the former could be expressed as an analytical formula, the latter cannot be expressed in this way (especially when Nop-1 is not a power of two). However, it is possible to bound the critical path delay by considering two extreme options.

One extreme situation occurs when the delay of the whole carry-chain corresponding to each ETL (dcarry _ the number of CSAs of the ETL) is always greater than the delay from an ETL to the next one (dsum). In this case, the timing behaviour corresponds to a linear array and the critical path is represented in Fig. 4. Initially, the first carry out signal is generated from I1, I2, I3 in the first CSA and then the carry signal is propagated through the whole carry-chain until the output. Thus, the delay of the critical path has two components corresponding to the generation of the first carry signal and the propagation through the carry-chain. If we characterize the delay from a general input to the carry output in the first CSA (including later routing) as dsum, then the estimated lower bound for the delay of the compressor tree is dcarry:

$$D_{low} \approx d_{sum} + (N_{op} - 3) \cdot d_{carry}.$$

III.COMPRESSORS

The conventional adders are the chain of Full adders which generates carries and sum at each level. There was a delay while generating the Final MSB bits of result. During multiplication, Booth's Algorithm or any conventional/Modified approach can be used. But during the partial product addition, the conventional adders are not enough to reach the time constraints. The normal adder structure for n bit addition is shown in figure 5 which adds two numbers with n number of bits but the output is not valid till the last MSB bit not appears. The carry travels through the adder to adder. This generates a delay which is consuming for carry propagation and ultimately efficiency of total circuit gets decreases.

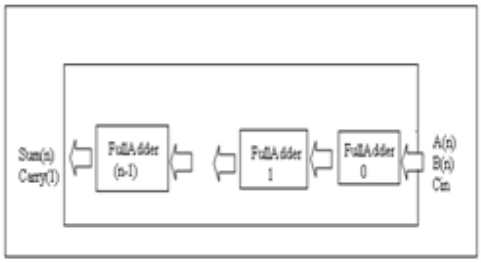

Fig 5: Conventional Adder structure for n bit addition generating n bit sum and 1 bit carry

A. 5:2 COMPRESSOR

A simple implementation of the (5, 2) compressor is to cascade three (3, 2) full adders in a hierarchical structure, as shown in Figure 6. The block diagram of a (5:2) compressor shown in Figure 8 has seven inputs and four outputs. Five of the inputs are the primary inputs I1, I2, I3, I4 and I5 and two other inputs, Cinl and Cin2. The architecture is connected in such a way that five of the inputs come from the same bit position of the weight j while other two inputs (Cinl and Cin2) are fed from the neigh-boring position j-1(known as carry-in). The outputs of 5:2 compressor consists of one bit in the position j (sum) and two bits in the position j+1 (cout1, cout2, carry).

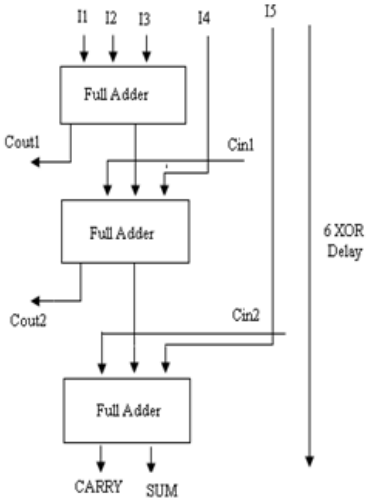

Fig 6: 5:2 Compressor general structure

This architecture has a critical path delay of 6 XOR gates. The alternative implementation shows that this design has

ISSN (Online) 2321-2004
ISSN (Print) 2321-5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 3, Issue 2, February 2015

a critical path delay of 4XOR + 1MUX unlike the conventional implementation with a delay of 5XOR.

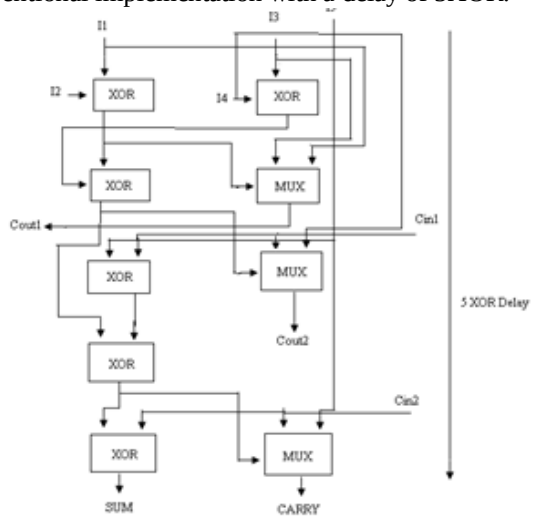

Fig 7: Alternative implementation of 5:2 compressor

IV. SIMULATION AND EXPERIMENTAL RESULTS

A. *LINEAR ARRAY STRUCTURE*

i. *Simulation result*

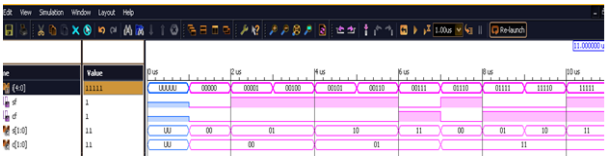

Fig 8:Simulation result of Linear array structure

ii. *Synthesis report*

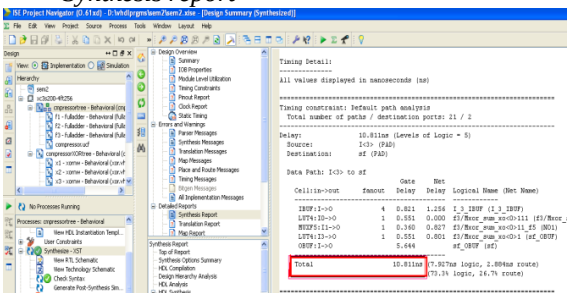

Fig 9: Synthesis report of linear array structure

B. *COMPRESSOR TREE*

i. *Simulation result*

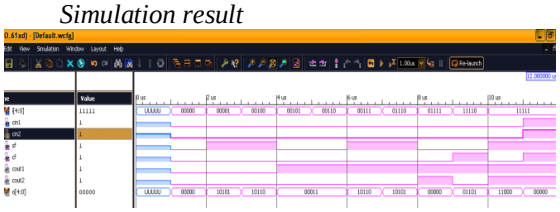

Fig 10: Simulation result of compressor tree

ii. *SYNTHESIS REPORT*

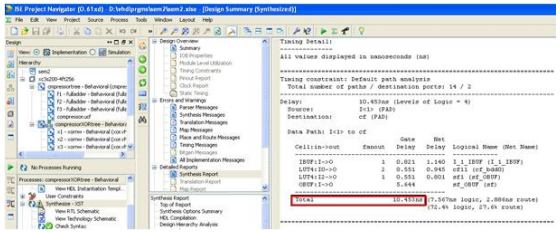

Fig 11: Synthesis report of compressor tree

V. TABULATION RESULT

logic	delay
Linear array structure	10.811ns
Compressor tree	10.453ns

Fig.12.Tabulation results of redundant adder structures

VI. CONCLUSION

Efficiently implementing CS compressor trees on FPGA, in terms of area and speed, is made possible by using the specialized carry-chains of these devices in a novel way. Similar to what happens when using ASIC technology, the proposed compressor trees lead to marked improvements in speed compared to CS linear array tree approaches and, in general, with no additional hardware cost. Furthermore, the proposed high-level definition of CSA arrays facilitates ease-of-use and portability, even in relation to future FPGA architectures, because they will probably remain a key element in the next generations of FPGA. We have compared our architectures, implemented on different FPGA families, to several designs and have provided a qualitative and quantitative study of the benefits of our proposals.

ACKNOWLEDGMENT

I owe my heartfelt gratitude to God almighty for all the blessings showered on me during this endeavour. I take this opportunity to express my sincere gratitude to all the people who have been instrumented in bringing out this work to the correct form. I thank my internal guide **Mr. JINU ISAAC KURUVILLA,** Asst. Professor, Electronics and Communication Engineering Department for his proper guidance and support during the course of this seminar. I also thank the Staff members of Electronics and Communication Engineering Department**,** for their co-operation for the completion of the seminar. Finally I thank my friends, classmates and family for providing me the strength and endurance.

REFERENCES

[1] "*Multioperand Redundant Adders on FPGAs"* Javier Hormigo, Julio Villalba, Member, IEEE, and Emilio L. Zapata IEEE transactions on computers, vol. 62, no. 10, october 2013.

[2] V.G. Oklobdzija, D. Villeger, S. S. Liu, "*A Method for Speed Optimized Partial Product Reduction and Generation of Fast Parallel Multipliers Using and Algorithmic Approach*", IEEE Transaction on Computers, Vol. 45, No 3, March 1996.

[3] Pallavi Devi Gopineedi, Hamid R. Arabnia. 2012. Novel and Efficient 4:2 and 5:2 Compressors with Minimum number of Transistors Designed for Low-Power Operations. Georgia : Athens, 2012.

[4] K. Prasad and K. K. Parhi, "*Low-power 4-2 and 5-2 compressors*," in proc.signals,systems and computers, vol. 1, pp. 129-133, Nov. 2001.