FPGA ARCHITECTURE OPTIMIZATION USING REDUNDANT ADDERS

MS. C. SAROJA, DR. K. KANTHAMMA

Student¹, Associate Professor², Dept. of E.C.E, Srinivasa Ramanujan Institute of Technology, Ananthapuramu, Andhrapradesh, India.
Email: 124g1d5715@srit.ac.in, kanthamma.ece@srit.ac.in

Abstract – Here we present different approaches to the efficient implementation of generic carry-save compressor trees on FPGAs. They present a fast critical path, independent of bit width, with practically no area overhead compared to CPA trees. Along with the classic carry-save compressor tree, here we present a novel linear array structure, which efficiently uses the fast carry-chain resources. This approach is defined in a parameterizable HDL code based on CPAs, which makes it compatible with any FPGA family or vendor. In this paper we are verifying results in SPARTAN 3 FPGA family.

Keywords – carry save adders, compressor trees, redundant adder, and multioperand addition, synthesis of compressor trees.

1. INTRODUCTION

The use of Field Programmable Gate Arrays (FPGAs) to implement digital circuits has been growing in recent years. In addition to their reconfiguration capabilities, modern FPGAs allow high parallel computing. FPGAs achieve speedups of two orders of magnitude over a general-purpose processor for arithmetic intensive algorithms. These kinds of devices are increasingly selected as the target technology for many applications, especially in digital signal processing, cryptography and much more.

The typical structure of an FPGA device is a matrix of configurable logic elements (LEs), each one surrounded by interconnection resources. In general, each configurable element is basically composed of one or several n-input lookup tables (N-LUT) and flip-flops. However, in modern FPGA architectures, the array of LEs has been augmented by including specialized circuitry, such as dedicated multipliers, block RAM, and so on.

One of these resources is the carry-chain system, which is used to improve the implementation of carry propagate adders (CPAs). It mainly consists of additional specialized logic to deal with the carry signals, and specific fast routing lines between consecutive LEs, as shown in Fig. 1. This resource is presented in most current FPGA devices from low-cost ones to high-end families, and it accelerates the carry propagation by more than one order of magnitude compared to its implementation using general resources. Apart from the CPA implementation, many studies have demonstrated the importance of using this resource to achieve designs with better performance and/or less area requirements, and even for implementing non-arithmetic circuits [1], [2].

Multioperand addition [3] appears in many algorithms, such as multiplication filters and others. To achieve efficient implementations of this operation, redundant adders are extensively used redundant representation reduces the addition time by limiting the length of the carry-propagation chains. The most usual representations are carry-save [4] (CS) and signed-digit (SD). A CS adder (CSA) [5], adds three numbers using an array of Full-Adders (FAs), but without propagating the carries. In this case, the FA is usually known as a 3:2 counter. The result is a CS number, which is composed of a sum-word and a carry-word. Therefore, the CS result is obtained without any carry propagation in the time taken by only one FA. The addition of two CS numbers requires an array of 4:2 compressors, which can be implemented by two 3:2 counters. The conversion to non redundant representation is achieved by adding the sum and carry word in a conventional CPA.

Fig. 1. General scheme of dedicated carry-chain resources included in Modern FPGA devices.

In this paper, we study the efficient implementation of compressor trees [6] in modern FPGAs by using their fast carry resources [7]. Our approaches strongly reduce delay and they generally present no area overhead compared to a
CPA tree. Moreover, they could be defined at a high level based on an array of standard CPAs.

2. CS COMPRESSOR TREES ON FPGAs

In this section, we present different approaches to efficiently map CS compressor trees on FPGA devices. In addition, approximate area and delay analysis are conducted for the general case. Let us consider a generic compressor tree of Nop input operands with N bit width each. We also assume the same bit width for input and output operands. Thus, input operands should have previously been zero or sign extended to guarantee that no overflow occurs.

3. REGULAR CARRY SAVE COMPRESSOR TREE DESIGN

The classic design of a multioperand CS compressor tree attempts to reduce the number of levels in its structure. The 3:2 counter or the 4:2 compressor are the most widely known building blocks to implement it. We select a 4:2 compressor as the basic building block, because it could be efficiently implemented on Xilinx FPGAs. The implementation of a generic CS compressor tree requires \( \lceil \frac{N_{op}}{2} \rceil - 1 \) 4:2 compressors (because each one eliminates two signals), whereas a carry-propagate tree uses Nop – 1 CPAs.

The speed of a compressor tree is determined by the number of levels required. In this case, because each level halves the number of input signals, the critical path delay (D) is approximately

\[
L_{4:2} = \lceil \log_2 (N_{op}) \rceil - 1
\]

\[
D \approx L_{4:2} \cdot d_{4:2}
\]

Where \( L_{4:2} \) is the number of levels of the compressor tree and \( d_{4:2} \) is the delay of a 4:2 compressor level (including routing).

4. LINEAR ARRAY STRUCTURE

To optimize the use of the carry resources, we propose a compressor tree structure similar to the classic linear array of CSAs. For the CSA, we have to distinguish between the regular inputs (A and B) and the carry input (C_i) in the figure), whereas the dashed line between the carry input and output represents the fast carry resources [4]. With the exception of the first CSA, where C_i is used to introduce an input operand, on each CSA C_i is connected to the carry output (Co) of the previous CSA, as shown in Fig. 2.

Fig.3 (where the carry-chain is represented in gray). 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 \( \lceil \frac{N_{op} - 1}{2} \rceil \) CSAs (which correspond to partial addition of the input operands).

![Fig. 2. CS 9:2 compressor tree based on a linear array of CSAs.](image)

![Fig. 3. Time model of the proposed CS 9:2 compressor tree.](image)

This first ETL produces \( \lceil \frac{N_{op} - 1}{2} \rceil \) 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

\[
L = \lceil \log_2 (N_{op} - 1) \rceil
\]

and the delay of this tree is approximately L times the delay of a single ETL.

5. Pipelining

One of the main advantages of CS compressor trees is that they are very suitable for pipelining. An M-stage pipeline compressor tree [8] of L levels of compressors is implemented simply by introducing one level of registers for each L=M levels of compressors. However, the proposed designs are defined using an array of CPAs, which complicates the introduction of registers in the linear array compressor tree.
6. 4:2 COMPRESSOR

The 4:2 compressor structure actually compresses five partial products bits into three \([1, 2, 3]\). The architecture is connected in such a way that four of the inputs are coming from the same bit position of the weight \(j\) while one bit is fed from the neighbouring position \(j-1\) (known as carry-in). The outputs of 4:2 compressor consists of one bit in the position \(j\) and two bits in the position \(j+1\). Fig. 4 shows the block diagram of 4-2 compressor. A 4-2 compressor can also be built using 3-2 compressors. It consists of two 3-2 compressors (full adders) in series and involves a critical path of 4 XOR delays as shown in Fig 5.

![Fig 4. Block Diagram of 4:2 Compressor](image1)

![Fig 5. 4:2 Compressor Design using Full Adders](image2)

7. 5:2 COMPRESSOR

The block diagram of a (5:2) compressor shown in Fig.6 has five inputs and two outputs. Five of the inputs are the primary inputs \(I_0, I_1, I_2, I_3, I_4\) and \(C_f, S_f\) are the outputs.

8. 5:3 COMPRESSOR

The block diagram of 5:3 compressor consisting of \(A, B, C\) as primary inputs and two carry signals \(cA, cB\). The outputs are two carry signals and sum.

9. 6:2 COMPRESSOR

Fig.8 shows the block diagram of 6:2 compressor tree [9] and fig.9 shows the architecture of 6:2 compressor tree.
Which consists of four 3:2 counters and one 2:2 counter. Here six are primary inputs and two are carry inputs of rank 0. The out1 and out0 are the two primary outputs. Two carry outputs are of rank 1 and rank2.

10. 7-2 COMPRESSOR

Fig.10 shows the block diagram of 7:2 compressor tree and fig.11 shows the architecture of 7:2 compressor tree. This consists of five 3:2 counters. Here seven are primary inputs and two are carry inputs of rank 0. The out1 and out0 are the two primary outputs. Two carry outputs are of rank1 and rank2.

11. 3-2 COMPRESSOR

Fig.12. 3:2 compressor block diagram

Fig.13. 3:2 compressor architecture

12. Simulation Results

Fig.14 shows the simulation results of 9:2 compressor tree designed by using of 3:2 compressor trees.

Fig.15 shows the simulation results of 4:2 compressor tree designed for 16 bit inputs.
Fig. 15. 4:2 compressor tree with 16 bit inputs

Fig. 16 shows the simulation results of 5:2 compressor tree with 16 bits as input.

Fig. 16. 5:2 compressor tree with 16 bit inputs

Fig. 17 shows the simulation results of 5:3 compressor tree with 16 bits as inputs.

Fig. 17. 5:3 compressor tree with 16 bit inputs

Fig. 18 shows the simulation results of 6:2 compressor tree with 16 bits as inputs.

Fig. 18. 6:2 compressor tree with 16 bit inputs

Fig. 19 shows the simulation results of 7:2 compressor tree with 16 bits as inputs.

Fig. 19. 7:2 compressor tree with 16 bit inputs

Fig. 20 shows the simulation results of 3:2 compressor tree with 16 bits as inputs.

Fig. 20. 3:2 compressor tree with 16 bit inputs

In the entire compressor trees 16 bits are taken as inputs and in some compressor trees redundant bits are added additionally. We can implement the number of inputs up to 64 bits [10].
13. CONCLUSION

Efficiently implementing CS compressor trees on FPGA, in terms of area and speed, is made possible by using the specialized carry-chains. While comparing among 3:2 and 4:2 and 5:2 compressor trees (single carry propagation) 4:2 is having reduced delay among these. Comparing 5:3, 6:2 and 7:2 (multiple carry propagation) for 16 bit input data 6:2 have reduced delay. The comparisons are made in fig.21 and fig.22. We can implement the number of bits up to 64 further.

14. REFERENCES