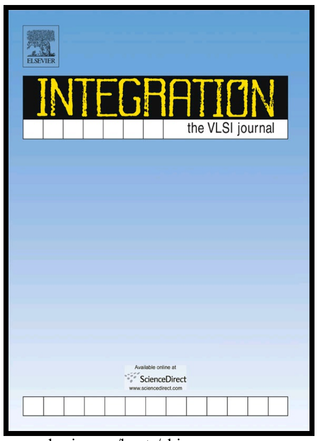
Fast Multiplier Generator for FPGAs with LUT based Partial Product Generation and Column/Row Compression

Ahmet Kakacak, Aydin Emre Guzel, Ozan Cihangir, Sezer Gören, H. Fatih Ugurdag



www.elsevier.com/locate/vlsi

PII: S0167-9260(16)30211-5
DOI: http://dx.doi.org/10.1016/j.vlsi.2016.12.012
Reference: VLSI1288

To appear in: *Integration, the VLSI Journal*

Received date: 21 August 2015
Revised date: 28 October 2016
Accepted date: 18 December 2016

Fast Multiplier Generator for FPGAs with LUT based Partial Product Generation and Column/Row Compression

Ahmet Kakacak[a,b], Aydin Emre Guzel[c], Ozan Cihangir[b], Sezer Gören[d,*], H. Fatih Ugurdag[b,c]

[a]*Dialog Semiconductor A.S., Istanbul, Turkey*
[b]*Dept. of Electrical and Electronics Engineering, Ozyegin University, Istanbul, Turkey*
[c]*Dept. of Computer Science, Ozyegin University, Istanbul, Turkey*
[d]*Dept. of Computer Engineering, Yeditepe University, Istanbul, Turkey*

Abstract

We present a new parallel integer multiplier generator for FPGAs. It combines (i) a new Generalized Parallel Counter (GPC) grouping algorithm for column compression with (ii) a LUT based partial product generation, is (iii) unique as it automatically generates placement pragmas, (iv) uses a ternary adder as a final adder to exploit FPGA's internal carry-chains, and (v) employs a novel GPC based row compression, which aims to reduce the width of the final adder. We wrote Verilog generators for our method as well as one leading work in the literature. For synthesis, we wrote a script that can do "binary search" for the optimum latency. Our extensive implementation results on Xilinx Virtex-6 FPGAs show that we almost always produce circuits with smaller latency (i.e., timing) and Area-Timing Product (ATP) compared to the state-of-the-art in the literature, 10% and 7%, respectively, on the average. We are worse only in 4 cases out of 33 and by 5% in terms of latency but never in ATP. We also offer smaller latency compared to the HDL * operator by 7% on the average at a cost of 14% larger ATP on the average. We are worse in latency in 7 cases out of 33, in all of which synthesis maps * to DSP slices. Binary search improves our results better than the results of the leading work in the literature and * operator. We also include area and energy results on Virtex-6 as well as a limited amount of latency, area, and ATP results on Virtex-5 and Altera Stratix III.

Keywords: Fast Multipliers, FPGA, Look-Up Table, Partial Product Generation, Column Compression Tree, Carry-Save Tree, Generalized Parallel Counter.

1. Introduction

Multiplication operation requires Partial Product Generation (PPG) followed by summation of the partial products. Although the summation can be simply done by a binary tree of two-operand adders, a much more speed-optimized summation, which is yet area-efficient, can be done if Carry-Save Adders (CSAs) are used, which is in fact nothing but a set of parallel full adder cells. A CSA reduces the summation of 3 numbers to summation of 2 numbers (i.e., addition). Hence, successive CSAs can "compress" (i.e., reduce) summation of 3 or more numbers to 2 numbers [1]. The circuit comprised of these successive CSAs is also called Column Compression Tree (CCT). A "final adder" can then add the outputs of the CCT to produce a single number.

The critical path of CCT based approach is approximately $2.5\log_2 n$ logic levels[1], while the critical path of the binary adder-tree approach is $(\log_2 n)^2$, when two n bit numbers are multiplied[2].

The most well-known CCTs are Wallace Tree [2] and Dadda Tree [3], which were introduced in the 1960s. Dadda Tree is very similar to Wallace Tree but it was published a little later than Wallace tree and can be thought of as a refined version. It produces smaller and yet faster circuits in most cases. Dadda introduced each basic unit of the CCT as a "parallel counter", which outputs the number of ones at its input and is shown as (x, z), where x is the number of input bits and z is the number of output bits [4]. Thus full adder cell is a (3,2) counter and half adder is a (2,2) counter. Dadda also introduced higher order counters such as (7,3) and (15,4) in [3]. The definition of counter can be extended such that a counter sums bits that have different weights. For this purpose, counters were generalized in [5], and they are commonly called Generalized Parallel Counter (GPC, see Figure 1).

*Corresponding author. Tel: +90 216–578–1734
Email addresses: ahmet.kakacak@diasemi.com (Ahmet Kakacak), aydin.guzel@ozu.edu.tr (Aydin Emre Guzel), ozan.cihangir@ozu.edu.tr (Ozan Cihangir), sgoren@cse.yeditepe.edu.tr (Sezer Gören), fatih.ugurdag@ozyegin.edu.tr (H. Fatih Ugurdag)

[1]$\log_{1.5} n \simeq 1.5\log_2 n$ for the compression tree, as it compresses 3 bits to 2 at every level, plus $\log_2 n$ for the final adder.

[2]In a binary tree, there are $\log_2 n$ nodes, i.e., adders, each of which is $\log_2 n$ logic levels.