IMPLEMENTATION AND ANALYSIS OF MULTIPLICATION ALGORITHMS FOR VLSI APPLICATIONS USING FPGA

[1]ANOOP C, [2]ANU CHALIL

[1,2]Electronics and Communication Engineering, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Kerala, India
E-mail: [1]anoopcn336@gmail.com, [2]anuchalil@am.amrita.edu

Abstract— The multiplication algorithms play a vital role in carrying out any mathematical operations. In the present sce- nario where power and area constraints are of utmost impor- tance, the algorithms used determine the efficiency and usability of the device. In this paper various multiplication algorithms have been implemented using FPGA and its different parameters are categorically analyzed.

Index Terms— Montgomery, Multipliers, Wallace Tree, Baugh-Wooley.

I. INTRODUCTION

MAC (Multiply and accumulate) is the core DSP operation in most of the DSP processors. In the paper, different multiplication algorithms have been implemented using verilog and analyzed the hardware output using FPGA. In the case of multiplier our main goal is to achieve scalable and parameterized design for fast prototyping in Field Programmable Gate Arrays (FPGAs). Flexibility of the design and computational latency create a trade-off, therefore this concept is suitable mostly for prototyping and proof-of-concept designs.
As a secondary objective we want to achieve effective utilization of a selected family of FPGAs and apply its specific features. In this way we can analyze suitability of a certain algorithm for the selected FPGA platform. Such approach is particularly appropriate in case the final implementation platform will be the same FPGA family. Flexible and effective design of multiplier would have a chance to offer an universal solution in the applications with different asymmetric algorithms or in similar systems based on the same algebraic operations [1].
The aim to design and implement a multiplier block with a universal interface that could be included in a variety of cryptosystems offering features for changing its configuration parameters e.g. length of the input parameters, computational time and occupied area.

II. BACKGROUND STUDY

Different multiplications algorithms such as

- Booth Multiplier Algorithm
- Wallace tree Multiplier Algorithm
- Baugh-Wooley Multiplier Algorithm
- Montgomery multiplier Algorithm

are analyzed as the background study for this work. There are reconfigurable devices which has hardware architecture with functionality of processing elements. Interconnection between them can be modified after fabrication time. The most known reconfigurable hardware components are FPGAs [2]. The reconfigurable platform makes it possible to remove obsolete algorithms from running systems and provide the new ones, even without hardware update or exchange.

A. FPGA Architecture

The underlying FPGA architecture consists of an array of the smallest programmable units - logic elements (LE) or configurable logic blocks (CLB), and the programmable connection switches [3]. A typical FPGA architecture consists of a high number (hundreds to thousands) of LEs and routing channels with different length/speed. By the LE we understand the smallest functional unit that is addressed by the mapping tools. Typically it consists of a look-up table (LUT) and a register (D flip-flop), what makes possible to implement the combinatorial as well as sequential logic, or a small memory block [5]. Additionally, the FPGA architecture may include special dedicated blocks or building items for other functions e.g. for storing data, computing multiplication and addition, synthesis clock signals. Modern FPGAs provide support for implementation of a wide range of the algorithms from area of signal processing, communication or networking.

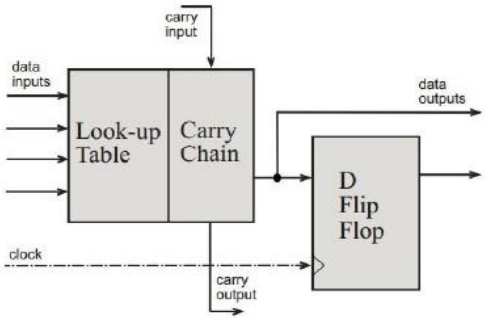

Fig. 1. Typical architecture of the smallest functional unit in a FPGA

B. Multiplication Algorithms

A multiplication algorithm is an algorithm used to multiply two numbers. Based On the number size, different algorithms are being used. Right from the

arrival of decimal system, efficient multiplication algorithms existed.

1) Grid Method: The grid method is an elementary multiple-digit multiplication technique used to teach pupils at elementary school level.

• First both the factors are partitioned into their hundreds, tens and units parts.

• By making use of single stage multiplication, product of the parts are explicitly calculated.

• These contributions are given to a separate addition stage for obtaining the final answer.

This approach of calculation is also referred as the partial products algorithm. The overall gist of this approach is calculation of simple multiplications and usage of addition in final gathering stage. The grid method calculation approach can be applied to factors of any size, even though the number of sub-products becomes cumbersome as the number of digits increases. Its an explicitly useful method for performing multiple-digit multiplications.

2) Long multiplication: A method of multiplying numbers when positional numeral system is used. It is also known as Standard Algorithm. Multiplicand is multiplied with each digit of the multiplier and the properly shifted results obtained are then added up. This method requires memorization of the multiplication table for single digits. This method is usually used for multiplying large numbers in base 10 manually.

3) Lattice multiplication: Lattice, or sieve, multiplication is algorithmically equivalent to long multiplication. Lattice is prepared for guiding the calculation which separates the multiplications from additions. Steps followed:

• In the multiplication phase, pointing to each row and column lattice gets filled with two digit product of the corresponding digits and the tens digit goes to the topleft corner.

• Lattice is added on the diagonals in the addition phase.

• If a carry phase is necessary in the final stage, the answer obtained along the left and bottom sides of the lattice are converted to normal form by carrying ten's digits similar to that done in long addition or multiplication.

4) Peasant or Binary multiplication: For radix 2 operations, long multiplication reduces to a nearly trivial operation. So for each '1' bit in the multiplier, the multiplicand is shifted by an appropriate amount and then the shifted values are then added. Based on the choice of multiplier and computer processor architecture used, coding of this algorithm can be done in a faster way using hardware bit shifts and adds instead of depending on multiplication instructions, when the multiplier is fixed and the number of adds required is small.

This algorithm also known by the name Peasant multiplication is widely used among those who are unschooled and also for those who have not memorized the multiplication tables required by long multiplication.

The main advantages obtained from this method include: it can be taught quickly, no memorization is required, and it can be performed using tokens. Also when compared to long multiplication, this method requires more steps, thereby making it unwieldy when large numbers are involved.

5) Booth Multiplier Algorithm: For signed-number multiplications, this is one among powerful algorithms which consider both positive and negative numbers in a uniform manner. For carrying out standard add-shift operation, each multiplier bit will generate one multiple of the multiplicand needed to obtain the partial product. When the multiplier used is very large, then a large number of multiplicands need to be added. In this case the delay of multiplier is mainly determined by the number of additions required to be performed. Performance can be improved provided the additions are reduced. This method also reduces the count of multiplicand multiples.

TABLE I
EXAMPLE OF BOOTH'S MULTIPLICATIONAL GORITHM:-MULTIPL 7 WITH 3 WITH THE HELP OF BOOTH'S ALGORITHM

AC	Q(R)	Q(n+1)	SC
0000	0011	0	4
1001(SHR)	0011(SHR)	0	4
1100(SHR)	1001(SHR)	1	3
1110	0100	1	2
0101(SHR)	0100(SHR)	0	1
0010(SHR)	1010(SHR)	0	1
0001	0101	0	0

6) Wallace tree Multiplier: A Wallace tree is an efficient hardware implementation of a digital circuit that multiplies 2 integers. Steps involved: In the initial phase,

• Each bit of one of the arguments is multiplied with each bit of the other to yield n2 results. The wires carry different weights based on the position of multiplied bits.

• The counts of the partial products are reduced to two using layers of full adders and half adders.

• The wires in the two numbers are grouped and then added by means of a conventional adder. Then as a part of the next phase,

• If three or more wires are comprised with the same weight, a following layer is added.

• Any three wires with same weight are taken and served as input to a full adder. The result obtained would be an output wire of same weight and an output wire with a higher weight for each three input wires.

• If two wires serving the same weight are left, they are input into a half adder.

• If there is just one wire left, connect it to the next layer. Advantages of Wallace tree are stated below.
• Nominal multiplier delay.
• Logic levels needed to perform the summation is reduced. Disadvantages of Wallace tree
• Layout complexity is much higher.
• Implementation results in creation of irregular wires.

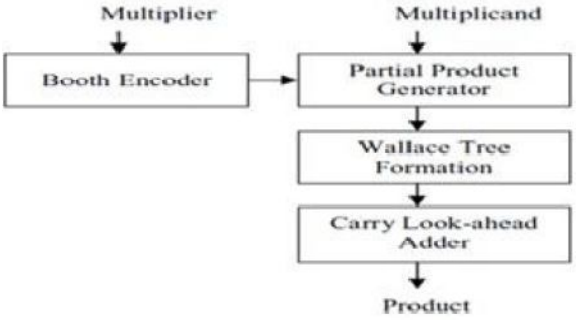

Fig. 2. Block diagram of Wallace tree multiplier

7) Baugh-Wooley Multiplier: For handling sign bits in a efficient way, we use Baugh Wooley Multiplication Algorithm. We utilise this technique in order to design regular multipliers suited for 2s complement numbers.

8) Montgomery multiplier: It is a method for performing faster modular multiplication [4], [7]. For the integers A and B, the classical modular multiplication algorithm computes the output as C = AB mod M where M is modulus. One of the broadly utilized algorithms for efficient modular multiplication is Montgomery algorithm. The central idea of the Montgomery Multiplication algorithm is utilize an alternate representation of the singular components of the underlying finite field. Montgomery multiplication of A and B (mod M) denoted as MM (A, B, M) is defined as A.B.2−m mod (M).Where A,B are variables and m denotes size M such that A,B < M < 2m . For cryptographic applications, M is typically a prime number or a result of primes, consequently this condition is easily satisfied [6]. The steps involved in Montgomery multiplication are as shown below.

• The bits xi of X is scanned bitwise from right to left.
• If the i-th bit of X, the value of Y is added to the intermediate result.
• Check whether the least significant bit (LSB) of this intermediate result is high, then additionally the prime M is added in order to make the LSB of the intermediate result to zero.
• Now, the result which is obtained from the above step is divided by 2, which is basically one bit shift to right.
• The process is continued until all the bits of X are scanned.
• The result obtained after the final scanning process is verified if it is more than M or not. If M is greater value, then the prime number is subtracted once for the final result.

III. IMPLEMENTATION AND RESULTS

The major multiplier algorithms mentioned below such as
• Booth Multiplier
• Wallace tree Multiplier
• Baugh-Wooley Multiplier
• Montgomery multiplier

All the algorithms are implemented in verilog and simulated using Questasim tool. After functional verification of the same, multipliers are implemented in FPGA. Synthesis is done using Xilinx ISE tool and the corresponding area and summary reports are compared.

TABLE II
BOOTH MULTIPLIER - SYNTHESIS REPORT

Components/Parameters	Number
4-bit adder	2
3-bit up counter	1
Flip-Flops	13
Multiplexer	2
Speed of operation	252.293MHz

TABLE III
WALLACETREE – SYNTHESIS REPORT

Components/Parameters	Number
Adders/Subtractors	3
Speed of operation	72.06MHz

TABLE IV
MONTGOMER MULTIPLIER - SYNTHESIS REPORT

Components/Parameters	Number
FSMs	1
4-bit subtractor	1
6-bit adder	3
Flip-Flops	23
6-bit comparator	1
1-bit xor2	1
Speed of operation	147.552MHz

TABLE V
BAUGH -WOOLEY - SYNTHESIS REPORT

Components/Parameters	Number
Adders/Subtractors	3
Speed of operation	70.7MHz

CONCLUSION

In this paper, we have done a comparative study of various multiplication algorithms. All four algorithms, Booth Multiplier Algorithm, Wallace tree, Baugh-Wooley, Montgomery multiplier are implemented for four bits in Verilog and simulated in Questasim. Bit file generation was done in Xilinx ISE and the complete analysis was carried out using Virtex-5 FPGA board.

It was found that Booth multiplier algorithm shows maximum speed of operation while Baugh-Wooley and Wallace tree showed least speed of operation. But the advantage of the Baugh-Wooley and Wallace tree is the lesser area and number of slices used for implementation. Whereas in Montgomery and Booth multiplier consumes more area. So it can be considered as a trade-off between area and speed of operation.

REFERENCES

[1] T. Blum and C. Paar, "Montgomery modular exponentiation on reconfigurable hardware" in in Proc. 14th IEEE Symp. On Computer Arithmetic, Year: 1999, Pages: 70-77

[2] Ghoreishi, S. et al.,"High Speed RSA Implementation Based on Modified Booths Technique and Montgomerys Multiplication for FPGA Platform" in Proceedings of the 2009 Second International Conference on Advances in Circuits, Electronics and Micro-electronics, Year: 2009,

[3] Schramm, M., Grzemba, A.,"Reconfigurable Trust for Embedded Computing Platforms" in IEEE Applied Electronics International Conference, Year: 2015,

[4] Drutarovsky, M., Simka, M., and Fischer, V. ,"Comparison of Scalable Montgomery Modular Multiplications Embedded in Reconfigurable Hardware" in Acta Electrotechnica et Informatica, Year: 2006, Pages: 3745

[5] Eldridge, S.E., and Walter, C.D.,"Hardware implementation of Montgomerys modular multiplication algorithm" in IEEE Trans. Comput., Year: 1993, Pages: 693699

[6] Tenca, A. F., Koc, C. K. ,"A scalable architecture for modular multiplication based on Montgomerys algorithm" in IEEE Transactions on Computers, Year: September 2003, Pages: 1215-1221

[7] Tenca, A. F., Koc, C. K. ,"A new RSA cryptosystem hardware design based on Montgomerys algorithm" in IEEE Trans. Circuits and Systems II: Analog and Digital Signal Processing, Year: 1998, vol. 45, Pages: 908913.

★★★