2012 2nd International Conference on Industrial Technology and Management (ICITM 2012)
IPCSIT vol. 49 (2012) © (2012) IACSIT Press, Singapore
DOI: 10.7763/IPCSIT.2012.V49.51

A Novel Pipelined Multiplier Using Divide and Conquer Algorithm

Wenbing Jin [1 2 +], Bo Zhu[2] and Xuanya Li [1]

[1] School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China
[2] North Automatic Control Technology Institute, Taiyuan, China

Abstract. Multiplier is one of the most important components in the modern processor, but it is extensively implemented by Modified Booth Encoding (MBE) algorithm and compressed tree architecture, both of which were proposed many years ago. A novel pipelined multiplier using Divide and Conquers (D&C) algorithm is proposed in this work. Firstly a deductive process in binary is offered to prove the D&C algorithm by means of the reduction of general multiplication's complexity. Then an example of typical 32-bit multiplication is taken to illustrate the division procedure from 32-bit to 8-bit, which is aimed to reduce the elementary multiplications in light of D&C algorithm. Finally a 32-bit pipelined multiplier using D&C algorithm is constructed and implemented in Xilinx FPGA. Post simulation after synthesis certifies the performance of the designed multiplier is higher than that of array or parallel one with MBE algorithm.

Keywords: Divide and conquer; multiplier; pipeline

1. Introduction

Multiplier is one of the most important arithmetic components in modern processor and it's located in the critical path of data processing. Up to date processor's performance on computing is based on the throughout time that multiplier occupies for its operation. As a result, dedicated multipliers are now extensively embedded in many processors, especially in DSP chips, of which high performance computing ability is essentially required.

Multiplication is usually performed by repetitive addition of partial products, which are generated by the multiplicand and one bit of multiplier according to the bit order. Array multiplier accumulates all the partial products in sequence slowly. To boost the multiplication procedure, Modified Booth Encoding (MBE) algorithm and compressed tree were proposed later in 1950s [1]. So far there are many preferable multipliers in literature, but most of them employ MBE algorithm and compressed tree structure, which were something originally proposed more than fifty years ago, very little effort has been spent on revolutionary algorithm or architecture in the latest ten years [2].

A novel pipelined multiplier architecture which we propose in the work is based on the application of Divide and Conquer (D&C) algorithm, with which the complexity of multiplication is degraded in a large scale. As a result, the speed of the proposed multiplier, in company with pipelined structure for partial products accumulation, boosts a lot; meanwhile the hardware budget for the multiplier is reduced proportionally.

The rest of this paper is organized as follows: Section 2 introduces some multipliers presented before. Section 3 describes the methodology for degrade the complexity of general multiplication by means of divide and conquer algorithm, and finally a 32-bit pipelined multiplier with this algorithm is designed. Section 4 introduces the implement on FPGA and simulation results for the designed multiplier. There is an analysis for timing issue in section 5. At last, some conclusions are drawn in section 6.

+ Corresponding author.
E-mail address: wenbbo@bit.edu.cn

2. Related Work

Array multiplier is simple with regular structure, but it's slow in process due to a number of stages for partial products accumulation [3, 4]. In 1950, Booth described a technique for multiplication of binary numbers [5], which was modified by many researchers later in the past 60 years [6, 7]. MBE is the most popular algorithm to reduce the number of partial products, but the reduction depends on the bits scanned each time. On the other hand, Wallace introduced compressed tree architecture for parallel accumulation of all the partial products [2], which was constructed by a set of counters and aimed to speed up the accumulation procedure for multiplication. Later, combined multiplier architectures with MBE algorithm and compressed tree were often studied with adjusted parameters for different objectives and applications [8]. Pipelined multiplier was first proposed in 1964, which is now proven to be efficient in promoting the clock rate by means of partitioning combinational logic circuits into stages with timing latches inserted in respectively [9].

3. Novel Multiplier Design

3.1. Methodology for degrade the complexity of general multiplication

Divide and conquer algorithm repeatedly breaks down a complicated problem into two or more sub-problems of the same type recursively until these broken sub-problems become simple enough to be solved directly. The solutions to sub-problems are then combined to give a solution to the original problem [10]. This technique is the basis of many efficient algorithms for kinds of problems including multiplication of large numbers and computing the discrete Fourier transform, etc. First of all, we now degrade the complexity of general multiplication by numbers based on 2 in terms of D&C algorithm.

Let X and Y be two n-digit binary integers, X is the multiplicand and Y is the multiplier. Below are the expressions given for X and Y:

$$X = a_{n-1}\,2^{n-1} + a_{n-2}\,2^{n-2} + \cdots + a_1\,2^1 + a_0\,2^0$$

$$= \sum_0^{n-1} a_i$$

Where: $a_i = 0$ or 1, $i = 0, 1, 2 \cdots n-1$.

$$Y = b_{n-1}\,2^{n-1} + b_{n-2}\,2^{n-2} + \cdots + b_1\,2^1 + b_0\,2^0$$

$$= \sum_0^{n-1} b_i$$

Where: $b_i = 0$ or 1, $i = 0, 1, 2 \cdots n-1$.

Traditional method for multiplication of X by Y is to multiply each bit in X and Y from the least significant bit to the most significant bit, and accumulate all partial products properly shifted. Time complexity for the long multiplication above is traditionally O (n^2).

Now we employ divide and conquer algorithm to split X and Y into two segments with the same digits. In convenience of discuss, we assume that n is a power of two, i.e. $n = 2^r$, then we can say that:

$$X = a_{n-1}\,2^{n-1} + a_{n-2}\,2^{n-2} + \cdots + a_1\,2^1 + a_0\,2^0$$

$$= (a_{n-1}\,2^{n-1} + a_{n-2}\,2^{n-2} + \cdots + a_{n/2}\,2^{n/2}) +$$

$$\cdots + (a_{n/2-1}\,2^{n/2-1} + \cdots + a_1\,2^1 + a_0\,2^0)$$

$$= (a_{n-1}\,2^{n/2-1} + a_{n-2}\,2^{n/2-2} + \cdots + a_{n/2}\,2^0)\;2^{n/2} +$$

$$\cdots + (a_{n/2-1}\,2^{n/2-1} + \cdots + a_1\,2^1 + a_0\,2^0)$$

$$= K\,2^{n/2} + L$$

Where: $K = a_{n-1}\,2^{n/2-1} + a_{n-2}\,2^{n/2-2} + \cdots + a_{n/2}\,2^0$

$$L = a_{n/2-1}\,2^{n/2-1} + \cdots + a_1\,2^1 + a_0\,2^0$$

$$Y = b_{n-1}\,2^{n-1} + b_{n-2}\,2^{n-2} + \cdots + b_1\,2^1 + b_0\,2^0$$

$$= (b_{n-1}\,2^{n-1} + b_{n-2}\,2^{n-2} + \cdots + b_{n/2}\,2^{n/2}) +$$

$$\cdots + (b_{n/2-1}\,2^{n/2-1} + \cdots + b_1\,2^1 + b_0\,2^0)$$

$$= (b_{n-1}\,2^{n/2-1} + b_{n-2}\,2^{n/2-2} + \cdots + b_{n/2}\,2^0)\,2^{n/2} +$$

$$\cdots + (b_{n/2-1}\,2^{n/2-1} + \cdots + b_1\,2^1 + b_0\,2^0)$$
$$= M\,2^{n/2} + N$$

Where: $M = b_{n-1}\,2^{n/2-1} + b_{n-2}\,2^{n/2-2} + \cdots + b_{n/2}\,2^0$

$$N = b_{n/2-1}\,2^{n/2-1} + \cdots + b_1\,2^1 + b_0\,2^0$$

And we then obtain:

$$X \times Y = (K\,2^{n/2} + L) \times (M\,2^{n/2} + N)$$
$$= KM\,2^n + (KN + LM)\,2^{n/2} + LN$$
$$= KM\,2^n + [(K + L) \times (M + N) - KM - LN]\,2^{n/2} + LN$$

Here K, M, L and N are all binary integers with $n/2$ digits, but $(K + L)$ and $(M + N)$ maybe binary integers with $(n/2 + 1)$ or $n/2$ digits. On the complex conditions, given that $(K + L)$ and $(M + N)$ are both binary integers with $(n/2 + 1)$ digits, we can gain as follows:

$$(K + L) \times (M + N) = (2^{n/2} + P) \times (2^{n/2} + Q)$$
$$= 2^n + (P + Q)\,2^{n/2} + PQ$$

Here P and Q are both binary integers with $n/2$ digits, consequently:

$$X \times Y = KM\,2^n + [(K + L) \times (M + N) - KM - LN]\,2^{n/2} + LN$$
$$= KM\,2^n + [(2^n + (P + Q)\,2^{n/2} + PQ) - KM - LN]\,2^{n/2} + LN$$
$$= (KM + P + Q)\,2^n + (PQ - KM - LN)\,2^{n/2} + LN + 2^{3n/2} \quad (1)$$

On the complex conditions mentioned in above equation, calculation for the product of X by Y needs only 3 times of multiplication by two $n/2$ digits binaries and 7 times of addition or subtraction. From above equation we can obtain the time complexity of X by Y:

$$T(n) = \begin{cases} c, & n = 1 \\ 3T\left(\frac{n}{2}\right) + cn, & n > 1 \end{cases}$$

Where $T(n)$ represents time complexity of multiplication by two n-digits, c and cn above represent time complexities for shift, addition and subtraction, which are generally performed by hardware within constant time or time proportional to n. Similarly, we continue to halve K, M, L, N, P and Q, further induce (1) by applying divide and conquer algorithm in terms of $n > 1$, hence the following time complexity holds:

$$T(n) = 3T\left(\frac{n}{2}\right) + cn$$
$$= 3\left[3T\left(\frac{n}{2^2}\right) + c\frac{n}{2}\right] + cn$$
$$= 3^2 T\left(\frac{n}{2^2}\right) + \frac{3}{2}cn + cn$$
$$= 3^2\left[3T\left(\frac{n}{2^3}\right) + \frac{1}{2^2}cn\right] + \frac{3}{2}cn + cn$$
$$= 3^r T\left(\frac{n}{2^r}\right) + \frac{3^{r-1}}{2^{r-1}}cn + \cdots + \frac{3^2}{2^2}cn + \frac{3}{2}cn + cn$$
$$= \frac{cn\left(\left(\frac{3}{2}\right)^{r+1} - 1\right)}{\frac{3}{2} - 1} 3^r c + \frac{3^{r-1}}{2^{r-1}}cn + \cdots + \frac{3^2}{2^2}cn + \frac{3}{2}cn + cn$$
$$= 3^r c + 2\left(\frac{3^r}{2^r} - 1\right)cn$$
$$= c \cdot 3^{\log_2 n} + 2c \cdot 3^{\log_2 n} - 2cn$$
$$= 3c \cdot n^{\log_2 3} - 2cn$$

Inference above utilizes the relations $n = 2^r$ and $3^{\log_2 n} = n^{\log_2 3}$. Since c is constant and cn is linear to n, the time complexity for this algorithm is O ($n^{\log_2 3}$), which is approximately O ($n^{1.59}$).

In general, if we split n-bit binary multiplicand and multiplier into $n/2$-bit in terms of divide and conquer algorithm, the complexity of multiplication is reduced accordingly.

3.2. Division of 32-bit binary integers for multiplication

Now we take a multiplication of two 32-bit binary integers as an example. There are 1024 times of elementary multiplications for two 32-bit binary integers. According to divide and conquer algorithm, we can divide each of 32-bit binary integers into four 8-bit binary integers recursively in steps and multiply them as follows,

Given that 32-bit multiplicand and multiplier are X and Y, $X = A_3A_2A_1A_0$, $Y = B_3B_2B_1B_0$, in which A_i, B_j $(0 \le i, j \le 3)$ are binary 8-bit integers,

$$
\begin{aligned}
X \times Y &= (A_3A_2A_1A_0) \times (B_3B_2B_1B_0) \\
&= (A_3A_2\, 2^{16} + A_1A_0) \times (B_3B_2\, 2^{16} + B_1B_0) \\
&= (A_3A_2 \times B_3B_2)\, 2^{32} + (A_1A_0 \times B_3B_2 \\
&\quad + A_3A_2 \times B_1B_0)\, 2^{16} + A_1A_0 \times B_1B_0 \\
&= (A_3A_2 \times B_3B_2)\, 2^{32} + ((A_3A_2 + A_1A_0)(B_3B_2 \\
&\quad + B_1B_0) - A_3A_2 \times B_3B_2 - A_1A_0 \times B_1B_0)\, 2^{16} \\
&\quad + A_1A_0 \times B_1B_0
\end{aligned}
$$

Let: $V = A_3A_2 \times B_3B_2 = (A_3\, 2^8 + A_2) \times (B_3\, 2^8 + B_2)$

$$
\begin{aligned}
&= (A_3 \times B_3)\, 2^{16} + (A_3 \times B_2 + A_2 \times B_3)\, 2^8 \\
&\quad + A_2 \times B_2 \\
&= (A_3 \times B_3)\, 2^{16} + [(A_3 + A_2) \times (B_3 + B_2) \\
&\quad - A_3 \times B_3 - A_2 \times B_2]\, 2^8 + A_2 \times B_2 \\
&= (A_3 \times B_3)\, 2^{16} + [(C_1C_0) \times (D_1D_0) - A_3 \times B_3 \\
&\quad - A_2 \times B_2]\, 2^8 + A_2 \times B_2 \\
&= (A_3 \times B_3)\, 2^{16} + [(C_1\, 2^8 + C_0) \times (D_1\, 2^8 + D_0) \\
&\quad - A_3 \times B_3 - A_2 \times B_2]\, 2^8 + A_2 \times B_2 \\
&= (A_3 \times B_3)\, 2^{16} + \{[(C_1 \times D_1)\, 2^{16} + (C_0 \times D_1 \\
&\quad + C_1 \times D_0)\, 2^8 + C_0 \times D_0] - A_3 \times B_3 - A_2 \times B_2\}\, 2^8 \\
&\quad + A_2 \times B_2
\end{aligned}
\qquad (2)
$$

Where C_1, C_0, D_1 and D_0 above are all 8-bit binary integers, but C_1, D_1 in fact have at most the least significant letter only, and the rest bits are all 0. Mean while $C_1C_0 = A_3 + A_2$, $D_1D_0 = B_3 + B_2$. Elementary multiplication times above are less than 209.

Let: $W = A_1A_0 \times B_1B_0$, we induce W in a similar way as (2):

$$
\begin{aligned}
W &= (A_1 \times B_1)\, 2^{16} + \{[(E_1 \times F_1)\, 2^{16} + (E_0 \times F_1 \\
&\quad + E_1 \times F_0)\, 2^8 + E_0 \times F_0] - A_1 \times B_1 - A_0 \times B_0\}\, 2^8 \\
&\quad + A_0 \times B_0
\end{aligned}
\qquad (3)
$$

Where E_1, E_0, F_1 and F_0 are the same as C_1, C_0, D_1 and D_0, and $E_1E_0 = A_1 + A_0$, $F_1F_0 = B_1 + B_0$. Similarly with (2), elementary multiplication times in (3) are less than 209.

Given $U = (A_3A_2 + A_1A_0) \times (B_3B_2 + B_1B_0) = G_2G_1G_0 \times H_2H_1H_0$, where $G_2G_1G_0 = A_3A_2 + A_1A_0$, $H_2H_1H_0 = B_3B_2 + B_1B_0$, G_2, G_1, G_0, H_2, H_1 and H_0 are all binary 8-bit integers, G_2 and H_2 in fact have at most the least significant letter only. The rest bits are all 0.

$$
\begin{aligned}
G_2G_1G_0 \times H_2H_1H_0 &= (G_2\, 2^{16} + G_1G_0)(H_2\, 2^{16} + H_1H_0) \\
&= (G_2 \times H_2)\, 2^{32} + (G_2 \times H_1H_0 + G_1G_0 \times H_2)\, 2^{16} \\
&\quad + G_1G_0 \times H_1H_0 \\
&= (G_2 \times H_2)\, 2^{32} + (G_2 \times H_1H_0 + G_1G_0 \times H_2)\, 2^{16} \\
&\quad + \{(G_1 \times H_1)\, 2^{16} + \{[(I_1 \times J_1)\, 2^{16} + (I_0 \times J_1 \\
&\quad + I_1 \times J_0)\, 2^8 + I_0 \times J_0] - G_1 \times H_1 - G_0 \times H_0\}\, 2^8
\end{aligned}
$$

$$+ G_0 \times H_0\} \quad (4)$$

Where above I_1, I_0, J_1 and J_0 are the same as C_1, C_0, D_1 and D_0, $I_1I_0 = G_1 + G_0$, $J_1J_0 = H_1 + H_0$. Elementary multiplication times in (4) are less than 242.

Now we obtain a new equation:

$$X \times Y = V\,2^{32} + (U - V - W)\,2^{16} + W \quad (5)$$

In a word, after division of 32-bit integers into 8-bit, elementary multiplication times are reduced from 1024 to 660. Meanwhile, variables *V, W* and *U* in (2), (3) and (4) can be generated in parallel. Next we will use (5) to design a pipelined multiplier.

3.3. Design of a 32-bit multiplier

Pipeline is now a widely used approach to improve performance. We implemented a 32-bit multiplier with a 3 stages pipeline in register transfer level (RTL). The functional structure in the proposed multiplier is illustrated in Fig. 1, which is constructed according to (5). Each block in the diagram is a module written in VHDL.

U0: Most top level block covers all modules in Fig.1. Input signals include CLOCK, RESET, 32-bit multiplicand and multiplier. Output signal is 64-bit product C64OUT.

U1: Loading the input operands and generating the intermediate values. 32-bit multiplicand and multiplier are locked in the registers at the rising edge of CLOCK signal after the RESET signal is available, intermediate values including A_3, A_2, A_1, A_0, B_3, B_2, B_1 and B_0 are transmitted to the second level units: U2, U3, U4. There are 9 8-bit parallel multipliers placed in the second level.

U2: Value V calculation. Input signals include A_3, A_2, B_3 and B_2. C_1, C_0, D_1 and D_0 are intermediate values induced by input signals; V is generated and output in the next clock.

U3: Value W calculation. Input signals include A_1, A_0, B_1 and B_0. E_1, E_0, F_1 and F_0 are intermediate values induced by input signals; W is generated and output in the next clock.

U4: Value U calculation. Input signals include A_3, A_2, A_1, A_0, B_3, B_2, B_1 and B_0. G_2, G_1, G_0, H_2, H_1, H_0, I_1, I_0, J_1 and J_0 are intermediate values induced by input signals; U is generated and output in the next clock.

U5: Accumulation for all partial products. All partial products addition and the product output at the third rising edge of clock.

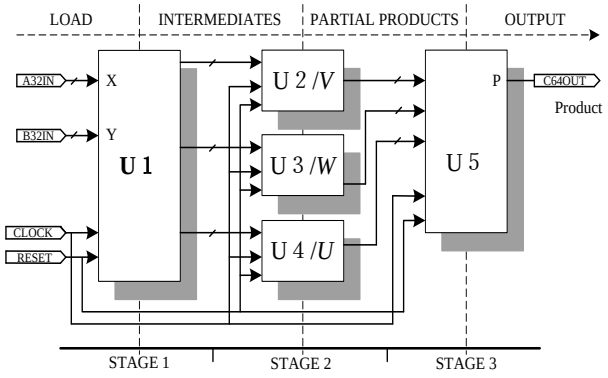

Fig. 1: Block diagram for the multiplier with pipelined stages

4. Simulation and Synthesis

ModelSim SE PLUS 6.0 presented by Mentor Graphics Corporation is adopted to simulate the logic function of the designed multiplier. Fig. 2 illustrates the simulated waveform for the designed 32-bit multiplier. The multiplier enters the working state after the reset signal turns over. Module U1 loads 32-bit multiplicand and multiplier at the first rising edge of clock, divides and outputs all the intermediate values required by the second level units, which include U2, U3 and U4. After the third rising edge of clock, U5 sends out the result of 64-bit product. Totally the 32-bit multiplication operation delays only 3 clocks.

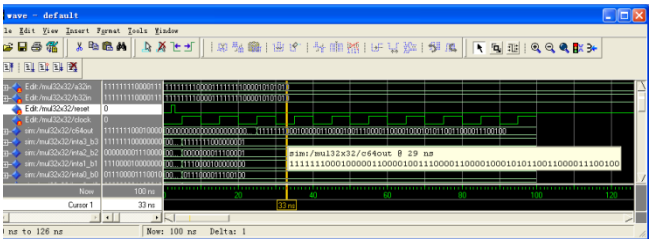

Fig. 2: Simulated waveform for designed multiplier

ISE 11 is a set of FPGA development tools presented by Xilinx Corp. [11]. We use ISE to implement, synthesize and post simulate the proposed design as well as two comparable samples. The target component is XC4VLX15/ff668/-12. Table 1 shows the post-simulated results for the designed multiplier and compared ones after synthesis. As can be seen from the table, the designed multiplier can output the product a little faster than the compared parallel multiplier, and the hardware budget occupied by the former is only two-thirds of the latter one. Array multiplier is too slow to discuss because of its serial addition structure. It should be noted that all the three simulated multipliers are implemented without utilizing any special structures embedded in FPGA chip, such as dedicated multiplier and fast carry chains. Fig. 3 is the post-simulated waveform for designed multiplier after synthesis with appropriate parameters.

TABLE I: Resources used by three simulated multipliers

a. Time in Table above Represents the Throughput Time for Multiplication.

5. Timing Analysis

Timing analysis is the basic for module optimization as well as for timing constraint and frequency improvement. In order to determine every out-of-order path, ISE static timing analysis tools adopt Brute-force attack algorithm to search for each possible path. We analyzed our designed multiplier, and found the critical path was U1/A_0 to U3/W, which delayed 9.638ns with 13 levels of logic structure in. Meanwhile, the critical path is also one of the additional trees. There are totally three parallel paths for addition tree with little difference in time. We applied timing constraint in our design within 15% margin left. Finally the main frequency was uppermost to 88 MHz. From Fig. 3 we can deduce that designed multiplier can work stable in 50MHz and the 64-bit products stream out in sequence continuously.

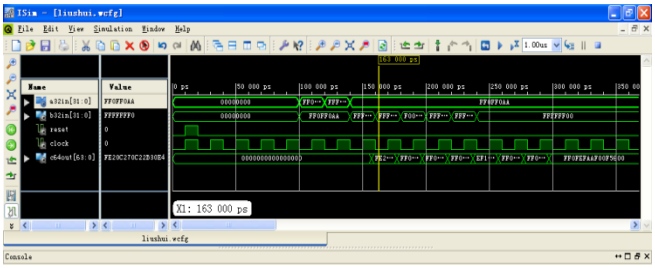

Fig. 3: Post simulated waveform for 32-bit multiplier

6. Summary

The divide and conquer algorithm divides the complex problem into several parts and solves them one by one, which is the effective way for parallel computing as well as some difficult problems. This work makes full use of the D&C algorithm to deal with multiplication. Meanwhile a pipelined multiplier is constructed by inserting latch registers between parts of combinational logic circuits. As a result, the frequency for the designed system increases a lot. The experimental results on FPGA demonstrate that the novel multiplier architecture with D&C algorithm outperforms the others used previously. In a word, the

proposed multiplier has the advantages of array multiplier as well as that of parallel one, and it can be used in many area-limited and power-sensitive applications, of which high performance is required.

7. References

[1] C. S. Wallace, "Suggestions for a fast multiplier," IEEE Trans. Electronic Computers, vol. 13, pp. 114-117, Feb. 1964.

[2] Y. Kanie et al., "4-2 Compressor with complementary pass-transistor logic," IEICE Trans. Electron, vol. E77-c, no. 4, pp. 789-796, Apr. 1994.

[3] Shailesh Shah, "Design and comparison of 32-bit multipliers for various performance measures", MEng. Project Report, Concordia University, January 2000.

[4] O. Salomon, J.-M. Green, and H. Mar, "General algorithms for a simplified addition of 2's complement numbers", IEEE Journal of Solid-state Circuits, V01.30, No.7, pp.839-844, July 1995.

[5] A. D. Booth, "A signed binary multiplication technique," Quart. J. Mech. Appl. Math., vol. 4, no. 2, pp. 236–240, June 1951.

[6] Magnus Sjalander and Per Larsson Edefors. "Multiplication acceleration through twin precision," IEEE Transaction on very large scale integration (VLSI) system. 2009, 17(9):1233-1246

[7] Jin-Hao Tu and Lan-Da Van, "Power-efficient pipelined reconfigurable fixed-width baugh-wooley multiplier." IEEE Transaction on computers, 2009, 58(10):1346-1354

[8] Dadda.L, "Some schemes for parallel multipliers", Alta Frequenza. Vo1.34, pp 349-356, 1965.

[9] Alex Panato, Sandro Silva, Flavio Wagner, Marcelo Johann, Ricardo Reis, Sergio Bampi, "Design of very deep pipelined multipliers for FPGAs", Proceedings of the Design, Automation and Test in Europe Conference and Exhibition Designers' Forum (DATE'04), 2004.

[10] http://en.wikipedia.org/wiki/Divide_and_conquer_algorithm,2011.

[11] Xilinx Inc. Xilinx ISE 11 In-Depth Tutorial. UG695 (v11.2) (C). June 24, 2009. www.xilinx.com