*International Journal of Engineering & Technology*, 7 (4.10) (2018) 1027-1028

International Journal of Engineering & Technology

*Website: www.sciencepubco.com/index.php/IJET*

***Research paper***

# An Improved Multiplication Algorithm

**S Subha***

*SITE, Vellore Institute of Technology, Vellore,T.Nadu, India*
**Corresponding author E-mail: ssubha@vit.ac.in*

**Abstract**

Multiplication is commonly used arithmetic operation in computers. An algorithm is proposed to perform multiplication of positive numbers using law of indices. The two inputs A and B are represented as binary numbers. Iteratively, the number A is multiplied with the bits in B. The coefficients of the partial products are accumulated. These coefficients are represented as binary numbers, the result coefficients are calculated. The proposed algorithm is simulated using Quartus 2 tool for two four bit inputs. An improvement in timing by 18% with comparable power consumption and increased area is observed for input of four bits. The proposed algorithm can be extended for n-bit inputs.

***Keywords***: *multiplication, Binarynumberrepresentation, Lawofindices, Performance*

## 1. Introduction

Addition, subtraction, multiplication and division are basic operations of numbers. [1]. Multiplication finds vast application as in DCT, matrix operations and various numerical calculations. Algorithms to multiply two binary numbers are proposed in literature [2, 3, 4, 5, 6, 7]. This paper proposes multiplication algorithm for sign magnitude numbers taking the magnitude only. The two numbers are viewed as polynomial in binary number system. The two numbers A and B are inspected for presence of one in various positions. The coefficients of the resulting product is calculated by iterating through B for A. The coefficients of the result thus obtained are represented in binary notation, the exponents of the resultant number are adjusted to obtain the result. The proposed model is simulated in Quartus2 toolkit. A timing improvement of 18% with increase in area and comparable power consumption is obtained compared with the multiplication algorithm present in the tool.
The rest of paper is organized as follows. Mathematical background is in section 2, proposed algorithm in section 3, simulations in section 4, conclusion in section 5 followed by references.

## 2. Mathematical Background

Consider binary number of four bits $b_3b_2b_1b_0$, $b_i \in \{0, 1\}$. Its numerical value is $b_3 2^3 + b_2 2^2 + b_1 2^1 + b_0 2^0$. When two binary numbers of four bits are multiplied, the result contains maximum exponent of six. This concept can be extended to n-bit numbers. When two n-bit binary, n>1, numbers are multiplied, the result has maximum exponent of 2n. The coefficient of terms is given by the following equations when the input is all ones. The least significant bit is numbered as one and most significant bit is numbered 2n.

Coefficient of term $2^j$ is (j+1) if $j \le n$ (1)

Coefficient of term $2^j$ is (2n- j) if $j > n$ (2)

The maximum coefficient value is (n) (3)

The number of binary digits required to express the maximum coefficient value is $\log_2 n+1$. For the product of two n-bit binary numbers , n=1, the result has maximum coefficient of one and minimum coefficient of zero. The number of bits required to express the maximum coefficient is one in this case.

## 3. Proposed Model

Consider two four bit binary numbers A and B. Let A= $(a_3, a_2, a_1, a_0)$, B=$(b_3, b_2, b_1, b_0)$. The result = $(r_7, r_6, r_5, r_4, r_3, r_2.r_1, r_0)$ where $r_i \in \{0,1\}$. The array C=$(c_7, c_6, c_{5,} c_4, c_3, c_2, c_1, c_0)$ stores intermediate results where $c_i$ is decimal number.

1. Start
2. For i= 0 to 3 do steps 3-6
3. If (a[i] != 0) do steps 4-6
4. For j=0 to 3 do steps 5-6
5. If (b[j] != 0) do steps 6
6. Assign c[i+j] = c[i+j] + 1
7. For k=0 to 7 do steps 8-10
8. If (c[k] != 0) do steps 9-10
9. Express c[k] in binary as P= $p_2 p_1 p_0$
10. Let c[k] = 0
10. If $(p_2 = 1)$ then c[k+2] = c[k+2] + 1. If c[k+2] = 2 then c[k+3] = c[k+3] + 1, c[k+2] = 0

If $(p_1 = 1)$ then c[k+1] = c[k+1] + 1. If c[k+1] = 2 then c[k+2] = c[k+2] + 1, c[k+1] = 0

If $(p_0 = 1)$ then c[k] = c[k} + 1. c[k] = 2 then c[k+1] = c[k+1] + 1, c[k] = 0
11. For k=0 to 7 do step 12
12. If ( c[k] != 0) result[k] = 1
13 Stop
The computational complexity of the above algorithm is of the order of the square of input number of bits. The above algorithm can be extended for finding the product of two n-bit numbers in which case, the time complexity is $O(n^2)$. The above algorithm does not produce partial products.
For the algorithm proposed for four bit input, the result has maximum exponent of six. According to (3) the maximum coefficient is (3+1) = 4. To represent 4 in binary notation we need three bits.

## 4. Simulation

Experiments using Verilog code of the proposed model is performed in Quartus 2 toolkit with the parameters given in Table 1. The proposed model is compared with the in-built multiplication algorithm supported by the tool. The area, power consumed and timing results are shown in Table 2. As seen from Table 2 there is performance improvement of 18% with increase in area and comparable power consumption.

**Table 1:** Simulation Parameters

| Parameter | Value |
|---|---|
| Processor family | CycloneII |
| Device | Auto selection by fitter |
| Package | FBGA |
| Pin count | 484 |
| Speed grade | Fastest |

**Table 2:** Simulation Results

| Description | Traditional | Proposed |
|---|---|---|
| Area (#slices/total #slices) | 31/14448 | 42/14448 |
| Power | 68.57mW | 68.60mW |
| Time | 8.118ns | 6.598ns |

**Example**: Consider the product of 1010 and 1001. The following gives the steps in simulating the algorithm given in section 3.

1. A = 1010 B = 1001 C is matrix of dimension 8x1. Result is matrix of dimension 8x1
2. Taking A i = 3, j = 3, c[6] = 1
   i = 3, j = 0, c[3] = 1
   i=1, j = 3, c[4] = 1
   i=1, j = 0, c[1] = 1
   The C matrix is [0,1, 0, 1, 1, 0, 1, 0]
3. Representing elements of C in binary the following is obtained
   C[6] = 001, c[6] = 1
   C[4] = 001, c[4] = 1
   C[3] = 001, c[3] = 1
   C[1] = 001, c[1] = 1
4. Assiging to Result matrix, the result is [0, 1, 0, 1, 1, 0, 1, 0] which gives the answer as 01011010.
5. Stop

## 5. Conclusion

A multiplication algorithm to find the product of two four bit binary numbers is proposed in this paper. The algorithm finds the coefficients of the product. It expresses the coefficients in binary notation, calculates the coefficients of the result based on the binary number. Experiments using Verilog code and Quartus 2 toolkit are conducted for the proposed model. It is compared with the multiplication algorithm supported by the Quartus 2 toolkit. A performance improvement of 18% is observed with increase in area and comparable power consumption.

## References


[1] Israel Koren(1993), Computer Arithmetic Algorithms, Prentice Hall, NJ
[2] Jongsu Park, San Kim, Yong-Surk Lee (2004), A Low-Power Booth Multiplier Using Novel Data Partition Method, Proceedings of AP-ASIC , pp. 54-57
[3] Nan-Ying Shen, OscalT.C. Chen (2002), Low Power Multipliers by Minimizing Switching Activities of Partial Products, Proceedings of ISCAS, pp. IV-93-IV-96
[4] Philip.E.Madrid, Brian Millar, Earl E Swartzlander Jr. (1993), Modified Booth Algorithm for High Radix Floating point Multiplication , IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 1, pp.164-167
[5] RajendraKatti (1994), A Modified Booth Algorithm for High Radix Fixed Point Multiplication, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 2, pp, 522-524
[6] Sandy Wang, Yi-Wen Wu, OscalT.C.Chen, Ruey-Lian Ma (2000), Low Power Multipliers by Minimizing Inter-Data Switching Activities, Proceedings of IEEE Midwest Symposium on Circuits and Systems Vol. I, pp.89-92
[7] S Subha, R Sakthivel (2016) A Power Saving Multiplication Algorithm, IJAER, Vol. 11, pp. 6200-6203