Design of 64 bit High Speed Vedic Multiplier

Ila Chaudhary, Deepika Kularia

ABSTRACT: A multiplier is one of the key hardware blocks in most digital signal processing (DSP) systems. Typical DSP applications where a multiplier plays an important role include digital filtering, digital communications and spectral analysis. Many current DSP applications are targeted at portable, battery-operated systems, so that power dissipation becomes one of the primary design constraints. Since multipliers are rather complex circuits and must typically operate at a high system clock rate, reducing the delay of a multiplier is an essential part of satisfying the overall design. This paper puts forward a high speed multiplier, which is efficient in terms of speed, making use of UrdhvaTiryagbhyam, a sutra from Vedic Maths for multiplication and half adder for addition of partial products. The code is written in VHDL and results shows that multiplier implemented using Vedic multiplication is efficient in terms of area and speed compared to its implementation using Array and Booth multiplier architectures.


I. INTRODUCTION

Multiplication is an important fundamental function in arithmetic operations. Multiplication-based operations such as Multiply and Accumulate (MAC) and inner product are among some of the frequently used computation Intensive Arithmetic Functions (CIAF) currently implemented in many Digital Signal Processing (DSP) applications such as convolution, Fast Fourier Transform (FFT), filtering and in microprocessors in its arithmetic and logic unit. Multiplication can be implemented using several algorithms such as: array, Booth, modified Booth algorithms.

Array multiplier is well known due to its regular structure. Multiplier circuit is based on add and shift algorithm. Each partial product is generated by the multiplication of the multiplicand with one multiplier bit. The partial product are shifted according to their bit orders and then added.

Booth Multipliers is a powerful algorithm for signed-number multiplication, which treats both positive and negative numbers uniformly. This method that will reduce the number of multiplicand multiples. For a given range of numbers to berepresented, a higher representation radix leads to fewer digits.

The partial-sum adders can also be rearranged in a tree like fashion, reducing both the critical path and the number of adder cells needed. The presented structure is called the Wallace tree multiplier. The tree multiplier realizes substantial hardware savings for larger multipliers. The propagation delay is reduced as well. In fact, it can be shown that the propagation delay through the tree is equal to O (log3/2(N)). While substantially faster than the carry-save structure for large multiplier word lengths, the Wallace multiplier has the disadvantage of being very irregular, which complicates the task of an efficient layout design.

II. LITERATURE SURVEY

Rapidly growing technology has raised demands for fast and efficient real time digital signal processing applications. Multiplication is one of the primary arithmetic operations every application demands. A large number of multiplier designs have been developed to enhance their speed. Active research over decades has lead to the emergence of Vedic Multipliers as one of the fastest and low power multiplier over traditional array and booth multipliers. Honey DurgaTiwari.et.al talked about designing a multiplier and square architecture is based on algorithm of ancient Indian
Vedic Mathematics, for low power and high speed applications. They explained Urdhvatiryakbhyam and Nikhilam algorithm and found that Urdhvatiryakbhyam is applicable to all cases of multiplication but due to its structure, it suffers from a high carry propagation delay in case of multiplication of large numbers. This problem has been solved by introducing Nikhilam Sutra which reduces the multiplication of two large numbers to the multiplication of two small numbers.

Prof J M Rudagil.et.al. designed a multiplier using vedic mathematics. They explained Urdhvatiryakbhyam and found that it is efficient Vedic multiplier with high speed, low power and consuming little bit wide area was designed. It was also found that the multiplier based on vedic sutras had execution delay of almost half of that of binary multiplier.

Sree Nivas A.et.al. presented a technique that modifies the architecture of the Vedic multiplier by using some existing methods in order to reduce power. They explained Nikhilam sutra and double base number system. Nikhilam sutra method is not valid for negative numbers. They found that Vedic Multiplier without any Modification has high power consumption. Vedic Multiplier with modified Two’s complement block has less power consumption with cost of delay and area.

III. VEDIC MATHEMATICS

Vedic mathematics - a gift given to this world by the ancient sages of India. A system which is far simpler and more enjoyable than modern mathematics. The word “Vedic” is derived from the word “Veda” which means the store-house of all knowledge[1]. Vedic math was rediscovered from the ancient Indian scriptures between 1911 and 1918 by Sri Bharati Krishna Tirthaji (1884-1960), a scholar of sanskrit, Mathematics, History and Philosophy. It is part of four Vedas (books of wisdom). It is part of Sthapatya- Veda (book on civil engineering and architecture), which is an upa-veda (supplement) of Atharva Veda[2]. Vedic mathematics is mainly based on 16 Sutras dealing with various branches of mathematics like arithmetic, algebra, geometry etc. These Sutras along with their brief meanings are enlisted below alphabetically[3].

1) (Anurupye) Shunyamanyat - If one is in ratio. The other is zero.
2) Chalana-Kalanabyham Differences and Similarities.
3) EkadhikinaPurvena - By one more than the previous one.
4) EkanyunenaPurvena - By one less than the previous one.
5) Gunakasamuchyah - The factors of the sum is equal to the sum of the factors.
6) Gunitasamuchyah - The product of the sum is equal to the sum of the product.
7) NikhilamNavatashcaramamDashatah - All from 9 and the last from 10.
8) ParaavartyaYojayet - Transpose and adjust.
9) Puranapuranabhyham - By the completion or non completion.
10) Sankalana-yyavakalanabhyam - By addition and by subtraction.
11) ShesanyankenaChararamena - The remainders by the last digit.
12) ShunyamSaamyasamuccaye - When the sum is the same that sum is zero.
13) Sopaantyadvayamantyam - The ultimate and twice the penultimate.
14) Urdhva-tiryakbhyam - Vertically and crosswise.
15) Vyashitisamanstih - Part and Whole.
16) Yaavadunam - Whatever the extent of its deficiency.

IV. METHODOLOGY

The algorithms and multiplier architecture which were studied are represented below:

4.1 Multiplication method

4.1.1 Urdhva-tiryakbhyam[3]
It is the general formula applicable to all cases of multiplication and also in the division of a large number by another large number. It means “Vertically and cross wise.”
4.1.2 Nikhilam Sutra [7]
Nikhilam Sutra literally means “all from 9 and last from 10”.

\[
\begin{align*}
96 \times 93 & \\
\text{Nearest base} &= 100 \\
96 & \quad (100 - 96) \\
93 & \quad (100 - 93) \\
\end{align*}
\]

4.2 Partial product addition
For addition of partial products various methods used was
1) Carry skip adder[4]
2) Zero padding[5]
3) Ripple carry adder[6]
4) Kogge stone adder[8]
5) Carry look ahead adder[8]

V. PROPOSED MULTIPLIER
Proposed multiplier architecture of 64x64 bit vedic multiplier and major change adopted here is use of half adder for addition of partial products.
Where \( h = \text{previous carry} \)
- \( Z(0) = a_0b_0 \)
- \( Z(1) = a_0b_1 + a_1b_0 \)
- \( Z(2) = a_1b_1 \)

**Figure 3: Addition of partial products with previous carry**

**VI. RESULTS AND STIMULATION**

The VHDL code of 64x64 bit vedic multiplier was synthesized using Xilinx ISE 14.4 on virtex4 family device XC4VLX25 and the results are shown in fig4. Comparison of area and delay is shown in table1. In which vedic multiplier 8X8bit is stimulated on xc3s400-5tql44 of SPARTAN 3 and rest on xc4vlx25-12ff676 of Virtex 4.

Figure 4 and Figure 5 shows device utilization summary and timing details respectively of 64 bit Vedic Multiplier.
Figure 5: Timing Details

Figure 6 shows RTL schematic of 64 bit vedic multiplier.

![RTL schematic of vedic multiplier](image)

Figure 7 shows output result of 64bit inputs where inputs are:
x <= "0000000000000000111111111111111111111111111111111111111111111111"
v <= "0000000000000000000000100000000000000000000000000000000000000000";  
h <= '0';
Figure 7: output stimulation of vedic multiplier

Figure 8 shows output result of 64 bit array multiplier

\[
x \leftarrow "0000000000000000000000000000000000000000000000000000001111111111"; \\
v \leftarrow "000000000000000000000000000000000000000000111111111110000000000"; \\
cin \leftarrow '0';
\]

Figure 9 shows output result of 64 bit booth’s multiplier

\[
x \leftarrow "0111111111111111111111111111111111111111111111111111111111111"; \\
i1 \leftarrow "0000000000000000000000000000001000000000000000000000000000000000";
\]
VII. CONCLUSION

Table 1: comparison table of designed architecture

<table>
<thead>
<tr>
<th>PROGRAM</th>
<th>DELAY</th>
<th>AREA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No. of 4 input LUTs</td>
<td>No. of occupied slices</td>
</tr>
<tr>
<td>Vedic multiplier 8X8 bit[8]</td>
<td>28.669 ns</td>
<td>192/7168</td>
</tr>
<tr>
<td>Array multiplier 64X 64 bit</td>
<td>1525.592 ns</td>
<td>9110/21504</td>
</tr>
<tr>
<td>Booth multiplier 64X 64 bit</td>
<td>138.250 ns</td>
<td>16192/21504</td>
</tr>
<tr>
<td>Vedic multiplier 64X 64 bit</td>
<td>29.967ns</td>
<td>2289/21504</td>
</tr>
</tbody>
</table>

From table 1 we can conclude that Vedic multiplier of 64x64 bit is has less delay and area as compared to Array and Booth multiplier.

REFERENCES