ABSTRACT

This paper describes that the propagation delay of 8-bit vedic multiplier is reduced when compared with conventional multiplier like array multiplier, booth multiplier, wallance multiplier. In our design we use barrel shifter which requires only one clock cycle for 'n' number of shifts. Also in this multiplier architecture comparator is used to reduce the complexity. The design is implemented in Xilinx simulator and verified using ISE simulator. The design is implemented on Xilinx Spartan-6 family. The propagation delay will taken from synthesis report and static timing report. The design achieve propagation delay 6.807 ns using barrel shifter in base selection module and multiplier.

Keywords: Vedic Formulas, Nikhilam Sutra, Barrel Shifter, Base Selection Module, Propagation Delay, Power Index Determinant.

1. INTRODUCTION

Arithmetic operations like addition, subtraction and multiplication are essential in different digital circuits to boost the process of computation. Vedic mathematics is the great technique for arithmetic operations. As compared to conventional techniques for multiplication gives significant amount of delay in hardware implementation of n-bit multiplier. This delay degrades the performance of the multiplier.

In this work our aim is to reduce the propagation delay of vedic multiplier using barrel shifter. The “Nikhilam Sutra”[1] implemented is modified. By using the barrel shifter in[1] the delay will reduce when compared with conventional multipliers.

2. VEDIC SUTRAS

“Vedic Mathematics” refers to a technique of calculation based on a set of 16 Sutras. Vedic sutras are the gift of ancient Indian mathematics. For large number of mathematical operations they apply. By using these sutras saves a lot of time compared to conventional computations. The faster processing speed is major improvements in processor technologies. The Vedic mathematics technique is totally different. Many architectures of multiplier has been reported but the performance of multiplier was improved in proposed design. The architecture in [1] is changed using barrel shifter so significant amount of clock cycles are reduced so speed increases. The performance of the proposed multiplier is compared with the previously implemented multipliers.

“Vedic mathematics” is comprised of sixteen simple mathematical formulae from the Vedas [5].

1. Ekadhikena Purvena
2. Nikhilam navatascaramam Dasatah
3. Urdhva - tiryagbhyam
4. Paravartya Yojayet
5. Sunyam Samya Samuccaye
6. Anurupy - Sunyamanyat
7. Sankalana - Vyavakalanabhyam
8. Puranapuranabhyam
9. Calana - Kalanabhyam
10. Ekanyunena Purvena
11. Anurupyena
12. Adyamadyenanta - mantyena
13. Yavadunam Tavadunikrtya Varganca Yojayet
14. Antyayor Dasakepi
15. Antyayoreva

2.1 "Urdhva-tiryakbyham " Sutra

The meaning of this sutra is “Vertically and crosswise” and it is applicable to all the multiplication operations.
Fig. 1 Multiplication procedure using “Urdhva-tiryakbyham” sutra

Fig. 1 represents the general multiplication procedure of the 4x4 multiplication. This process is called as array multiplication technique. It is an efficient multiplication technique when the multiplier and multiplicand lengths are small, for the larger length multiplication this technique is not good because a large amount of propagation delays are involved in these cases. To overcome this problem we are describing Nikhilam sutra for calculating the multiplication of two larger numbers.

3. MULTIPLIER ARCHITECTURE DESIGN

Consider two n bit numbers X and Y. k₁ and k₂ are the exponent of X and Y. X and Y can be represented as:

\[ X = 2^{k_1} \pm Z_1 \]  \hspace{1cm} (1)
\[ Y = 2^{k_2} \pm Z_2 \]  \hspace{1cm} (2)

Where Z₁ and Z₂ are residue part.

For the fast multiplication using Nikhilam sutra the bases of the multiplicand and the multiplier should be same for that equation (2) is multiplied by \( 2^{k_1-k_2} \) thus the equation (2) becomes as:

\[ Y \times 2^{k_1-k_2} = 2^{k_1} \pm Z_2 2^{k_1-k_2} \]  \hspace{1cm} (3)

\[ X \times Y \times 2^{k_1-k_2} = (2^{k_1} \pm Z_1)(2^{k_1} \pm Z_2 2^{k_1-k_2}) \]  \hspace{1cm} (4)
\[ = 2^{2k_1} \pm Z_1 Z_2 + 2^{k_1-k_2}Z_2 Z_2 2^{k_1-k_2} \]  \hspace{1cm} (5)
\[ = 2^{k_1} (2^{k_1} \pm Z_1 Z_2 2^{k_1-k_2}) \pm Z_1 Z_2 2^{k_1-k_2} \]  \hspace{1cm} (6)
\[ = 2^{k_1} (X \pm Z_2 2^{k_1-k_2}) \pm Z_1 Z_2 2^{k_1-k_2} \]  \hspace{1cm} (7)

\[ P = XY = 2^{k_1}X \pm Z_1 Z_2 2^{k_1-k_2} \pm Z_1 Z_2 2^{k_1-k_2} \]  \hspace{1cm} (8)

The hardware implementation of the above expression is partitioned into three blocks.

1. Base Selection Module
2. Power index Determinant Module

The base selection module (BSM) is used to select the maximum base with respect to the input numbers. The second sub-module power index determinant (PID) is used to extract the power index of k₁ and k₂. The urdhwa tiryakbyham multiplication is used for multiplying residue parts Z₁ and Z₂.

3.1 Base selection module

The base selection module consists power index determinant (PID) as the sub-module along with barrel shifter, adder, average determinant, comparator and multiplexer.

Operation:

Consider an ‘n’ bit binary number X, and it can be represented as:

\[ X = \sum_{i=0}^{n-1} x_i 2^i \]
\[ x_i \in \{0,1\} \]

Then values of X must lie in the range \( 2^{n-1} \leq X < 2^n \). Consider the average of the range is equal to A.

\[ A = \frac{2^{n-1} + 2^n}{2} \]

The aim of the BSM is to select the desired base by using below formula.
If $X > A$ then Base = $2^n$
If $X \leq A$ then Base = $2^{n-1}$

The Block level architecture of BSM is shown in Figure 2. BSM consists of three main subsections: (i) Power index Determinant (PID), (ii) Average Determinant and (iii) Comparator. ‘$n$’ number bit from input $X$ is fed to the PID block. The maximum power of $X$ is extracted at the output which is again fed to barrel shifter and the adder block. The second input to the barrel shifter is the $(n+1)$ bit representation of decimal ‘1’. If the maximum power of $X$ from the PID unit is $(n-1)$ then the output of the barrel shifter is $2^{n-1}$. The adder unit is needed to increment the value of the maximum power of $X$ by ‘1’. The second barrel shifter is needed to generate the value of $2^n$. Here $n$ is the incremented value taken from the adder block. The Average Determinant unit is required to compute the average of $(2^{n-1} + 2^n)$. The Comparator compares the actual input with the average value of $(2^{n-1} + 2^n)$. If the input is greater than the average then $2^n$ is selected as the required base. If the input is less than the mean then $2^{n-1}$ is selected as the base. The select input to the multiplexer block is taken from the output of the comparator.

3.2 Power index determinant

Figure 3 represent the block diagram of PID. The input number is fed to the shifter which will shift the input bits by one clock cycle. The shifter pin is assigned to shifter to check whether the number is to be shifted or not. In this power index determinant (PID) the sequential searching has been employed to search for first ‘1’ in the input number starting from MSB. If the search bit is ‘0’ then the counter value will decrement up to the detection of input search bit is ‘1’. Now the output of the decrementer is the required power index of the input number.

4. MULTIPLIER ARCHITECTURE

The base selection module and the power index determinant form subpart of multiplier architecture. The architecture computes the mathematical expression in equation (8).

As shown in figure 4 at First step suppose P and Q two 8-bit input numbers are given input to the comparator to search greater (suppose X) and smaller (suppose Y) numbers. Now these two input numbers (X and Y) are fed to the base selection module (BSM) to select proper base corresponding to the input numbers. As discussed in nikhilam sutra that if the selected base is nearer to the given number then multiplication of the residual parts ($Z_1 \times Z_2$) can be easier to compute. The outputs of base selection module (BSM) and the input numbers ‘X’ and ‘Y’ are fed to the subtractors. The subtractor blocks are required to extract the residual parts $Z_1$ and $Z_2$. The inputs to the power index determinant (PID) are from base selection module (BSM) of respective input numbers. The sub-section of power index determinant (PID) is used to extract the power ($k_1$ and $k_2$) of the base and
followed by subtractor to calculate the value of \((k_1-k_2)\). The outputs of subtractor are fed to the urdhwatyakhyam multiplication block that feeds the input to the second adder/subtractor. The output of the subtractor\((k_1-k_2)\) and \(Z_2\) fed the input to the barrel shifter to calculate the value of \((Z_2\times2^{k_1-k_2})\). The input number ‘X’ and the output of barrel shifter\((Z_2\times2^{k_1-k_2})\) are given to first adder/subtractor block to calculate the value of \((X \pm Z_2\times2^{k_1-k_2})\). The output of adder/subtractor block is applied to the second barrel shifter to compute the value of \(2^{k_2}\times(X \pm Z_2\times2^{k_1-k_2})\). The output of multiplier\((Z_1Z_2)\) and output of second barrel shifter\((2^{k_2}\times(X \pm Z_2\times2^{k_1-k_2}))\) are fed to the second adder/subtractor block to compute the value of \((2^{k_2}(X \pm Z_2\times2^{k_1-k_2}) \pm Z_1Z_2)\). This is the final result of multiplier and of equation (8).

Fig.4 Multiplier Architecture

5. SIMULATION RESULTS AND DESIGN ANALYSIS

This multiplier architecture is compared with 4-bit adder based\(^6\) & compressor based architecture\(^2\) for propagation delay from result it is clear that around 45% improvement is come in this design. The design was implemented in Xilinx Spartan-6 family xc6s1x75T-3 fg676 FPGA.

<table>
<thead>
<tr>
<th>Name of Multiplier</th>
<th>Array Multiplier</th>
<th>Booth Multiplier</th>
<th>Proposed Multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay (ns)</td>
<td>47</td>
<td>117</td>
<td>27</td>
</tr>
</tbody>
</table>

Table 1. Comparison of Multiplier w.r.t delay\(^3\)

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Using 4-bit Adder</th>
<th>Using Compressor</th>
<th>Using Barrel Shifter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay (ns)</td>
<td>17.768</td>
<td>14.41</td>
<td>6.807</td>
</tr>
</tbody>
</table>

Table 2. Comparison of various multiplier architecture w.r.t delay

Fig.5 Simulation Result of Multiplier Architecture

Fig.6 RTL schematic of multiplier Architecture
CONCLUSION
In our design, efforts have been made to reduce the propagation delay and may be achieve an improvement in the reduction of delay with 45% when compared to architecture using 4-bit adder and compressor based architecture. The high speed implementation of such a multiplier has wide range of applications in image processing, arithmetic logic unit and VLSI signal processing. The future scope of this particular work can be extended in design of ALU’s in RISC processor.

ACKNOWLEDGEMENTS
I am thankful to my parents, my brother, my special friends to motivate me and to increase my confidence. My special thanks goes to my friend Mr. Rajan Shah for his huge support. I am thankful to Altaf Darvadiya sir to guide me.

REFERENCES