DESIGNING A NOVEL ARCHITECTURE FOR WT MULTIPLIER TO IMPROVE ITS PERFORMANCE

1G. Srilakshmi, 2M. Sivakumar, 3KSrinivas Rao 4J.E.N.Abilash
1,2,3Dept. of ECE K L University, 4Dept. of ECE, Swarnandhra College of Engineering and technology, narsapur

ABSTRACT: Digital communication mainly depends on signal processing in which it high speed calculations like FIR filter responses etc which in this regard, for these calculations, adders and multipliers plays a key role. So, we are proposing a design of novel architecture of Wallace tree multiplier which is comprising of approximate adders for low area and power consumption. Here the proposed method reduces the number of partial products by changing their logic implementation. Addition logic was modified by introducing three modified adders called Approximate half adder (AHA), Approximate full adder (AFA), Approximate Compressor adder (ACA- 4:2). We have utilized adders for implementing the design for 8-bit multiplier and observed the Power savings at 76% to 71% as well as area is reduced from 64% to 53% and Mean error rate is as low as 7.3% to 0.04%. Modified architecture is implemented for low different applications like FIR filter design either 1-D or 2-Dimensional applications. The proposed multiplier achieves the good results for image compression applications.

Keywords: Approximate adders, Image compressor, mean error rate, low power multiplier, Wallace tree multiplier.

1. Introduction

In image processing applications, video processing, speech processing, data processing application etc requires the processing with large amounts of data manipulations either arithmetic or logical operations in this regard, mainly arithmetic applications like adders and multipliers plays a key role. But all applications does not requires the exact original outputs which all consumes more power and area, so in this perception a new model adders are been proposed so that the area and power along with error rate will be reduced. Modified Half Adder (MHA), Modified Full Adder (MFA), Modified Compressor Adder (MCA).

The previous methods of reductions of area and power mainly depend on truncation multiplier. Mainly truncation multiplier uses the concept of replacing number of bits values’0’and finally a predefined error is added. The drawback in concept is it that LSB made’0’s are multiplied with MSB’s at second number (multiplicand) then resultant value are reduced to ‘0’ in place of ‘1’ in MSB positions, which results to generate wrong values. In some cases when in the place of original zero values, some non zero values are resulting. This concept is suitable for some values only but for higher values results high error rate.

Later enhancements lead to a new methodology called Error Tolerant Adders (ETA) has been developed in which for example adding a two 8 bits. Different steps following as they are as follows explained as below:-

1) Divide the bits into two groups in which group is depend on error tolerance for example is For 4 bit in 8 bit adder then make the 8 it as 2 equal values.
2) Right side addition is as use as ripple carry adder (4 bit RCA) where as the left side addition is different.
3) Left side addition is different. It is going in left direction if any carry is observed then the addition with stops by replacing all remaining bits as ‘1’.

The main drawback in this method is for LSB part in the place of “1000” there may be chance of getting “1111” which it set having a difference of error (7) which is very high error.

Different methods have been proposed in the old methods that applying these approximate adders directly on the architecture. So it can reduce the levels of reduction stages and area. But our proposed architecture is not only applying them directly we are changing the partial products into another format by introducing two new terms called Ai and Bi. Arithmetic circuits like half adders, full adders and 4:2 compressors are replaced by modern architectures. In some cases the calculations of multiplication using our proposed method is generating the differences to the original output that is named as Error Distance (ED) using proposed approximate adders we are going to reduce ED compared to the previous
works. Along with ED calculation general errors and peak signal to noise ratio (PSNR) is also done and proposed method was shown.

The rest of the paper is organized as follows. Section II details the original architecture of Wallace tree multiplier, Section III explains above the proposed methods, Section IV deals with more elaborate analysis of how to use the approximate adders for reduction in stages. Section V gives the clear picture about results and compression of proposed multiplier with existing multipliers.

Section I: Wallace tree multiplier.

WTM is one of the most important multiplier which uses two types of adders called half adders and full adders. It follows mainly steps in calculating the final output and steps are following as

- Calculation of partial products and arrange in 9 column use depending on order.
- If two elements exist in the same column use half adder and if any three elements are there use full adder.
- Follow the above procedure until the last stage is consisting of only two elements and the last stage simple Ripple Carry Adder is used for final addition.

The drawback in applying the proposed approximate adders like half adder and full adder directly in Wallace Tree Multiplier results a high error distance and the output values are not accurate.

3. Proposed Architecture

Our Proposed method mainly focuses on logical implementation of product terms along its generation is explained as follows in following steps briefly:-

- Calculation of normal partial products.
- Alterations of Partial products to altered partial product terms called Ai and Bi continued by reduction.
- Find stage is comprised of normal ripple carry addition. To generate the final output

An example of multiplication of 8 bit is consider like two 8 bit numbers L and M. The partial product terms are calculated by using simple logic gate between two bits

\[ F_{p,q} = L_p \times M_q \]

According to the bit values i.e. mansLp and Mq will have a combination of both ones in only one case the statistical analysis proved mean probability of generating \( F_{p,q} \) as ‘1’ is 0.25. In the above diagram of partial products the column 3 to column 11 are replaced by altered partial products. The altered partial products are generated by the equators

\[ A_{p,q} = F_{p,q} \text{ or } F_{q,p}, B_{p,q} = F_{p,q} \text{ and } F_{q,p} \quad (1) \]

The probability of generating the second element is \( B_{p,q} \) is 0.0625 which is less than 0.25 for generating \( F_{p,q} \) because it is product of two terms with probability of 0.25.

4. Modified Wallace Tree Architecture

\[ F_{p,q} \\ F_{0,7} F_{0,5} F_{0,1} F_{0,9} F_{0,3} F_{0,1} F_{0,7} F_{0,9} \\ F_{0,7} F_{0,5} F_{0,1} F_{0,9} F_{0,3} F_{0,1} F_{0,7} F_{0,9} \\ F_{0,7} F_{0,5} F_{0,1} F_{0,9} F_{0,3} F_{0,1} F_{0,7} F_{0,9} \\ F_{0,7} F_{0,5} F_{0,1} F_{0,9} F_{0,3} F_{0,1} F_{0,7} F_{0,9} \\ F_{0,7} F_{0,5} F_{0,1} F_{0,9} F_{0,3} F_{0,1} F_{0,7} F_{0,9} \\ F_{0,7} F_{0,5} F_{0,1} F_{0,9} F_{0,3} F_{0,1} F_{0,7} F_{0,9} \\ F_{0,7} F_{0,5} F_{0,1} F_{0,9} F_{0,3} F_{0,1} F_{0,7} F_{0,9} \\ F_{0,7} F_{0,5} F_{0,1} F_{0,9} F_{0,3} F_{0,1} F_{0,7} F_{0,9} \]
Similarly for the first term \( A_{p,q} \) the probability will be 
\[
0.25 + 0.1875 + 0.1875 = 0.4375
\]
which is higher than the probability of generating ‘1’ in \( B_{p,q} \). So these altered partial products can be given to approximate adder.

**Generation of \( B_{p,q} \) term:**

The term \( B_{p,q} \) is generated in column wise. In each column every term will have the probability of generating ‘1’ is 0.0625. If a column consists of more than one ‘Bi’ element it will decreases. Like it in a column with 2 generate signals probability of all numbers ‘0’ is \((1-K)^2\) only one element being ‘1’ is \(2K(1-K)\) probability of all ‘1’ is \(K/2\), where \(K\) is 0.0625.

So depending on the above considerations the accumulation of column use altered partial product terms can be given to ‘or’ to generate reduced signals so that the error can also be reduced.

**5. Approximation For Partial Products**

The addition of remaining partial products with the probability of 0.25 for \( F_{p,q} \) and 0.4385 for \( A_{p,q} \) are to be generated by the approximate arithmetic circuits. Approximate Half adders are the modified version of half adders. Similarly approximate full is the successor of compressor (4:2) in which the general compressor is having 4 inputs and generate 3 different outputs i.e. two carries of one order and one sum bit of another order. In approximate compressor four inputs are added to generate only two bits.

In general an Exclusive-OR gate requires 2 AND gates, 2 NOT gates and one OR gate. So in our proposed methods of designing AHA we are replacing the XOR gate for sum generation with only OR gate so that the remaining gates are calculated. It is explained in equations as follows:-

\[
H.A \quad (\text{Half Adder}) \quad \begin{align*}
S &= A \text{ XOR } B \quad (A \text{ bar AND } B) \text{ OR } (A \text{ AND } B \text{ bar}) \\
C &= A \text{ AND } B
\end{align*}
\]

\[
AHA \quad (\text{Approximate Half Adder}) \quad \begin{align*}
S &= A \text{ OR } B \\
C &= A \text{ AND } B
\end{align*}
\]

The above modified equation is generating one single bit error and this case only effected only once but it reduces the maximum power is consumed by 2 AND gates and 2 NOT gates. The detailed bit wise analysis is given in the above Table no II.

Approximate half adders are also designed taking the advantage of above concept of replacing XOR gate with OR gate. Here in full adder the generation of sum, 2 XOR gates are required so in the modification one XOR gate is removed, and replaced by simple OR gate. Similarly in case generation full adder uses three AND gates and 2 OR gates are required but in the modified full adder only 2 AND gates and 1 OR gate is used. So 1 XOR (2 AND, 2 NOT) and 1 OR gate.

Totally 5 gates are removed in this proposed approximate full adder. This is explained by following equations

\[
M = R1 \text{ OR } R2 \quad S = M \text{ XOR } R3 \quad C = M \text{ AND } R3
\]

But for general full adder the equations are

\[
S = R1 \text{ XOR } R2 \text{ XOR } R3; \quad C = (R1 \text{ AND } R2) \text{ OR } (R2 \text{ AND } R3) \text{ OR } (R3 \text{ AND } R1)
\]
The detailed bit wise analysis is shown in table III. In some cases like last two rows the error value is generated i.e. (in place of 2, ‘1’ is generator and in place of ‘3’, ‘2’ is generated) so the error is ‘1’ bit less than the specified value.

Compressors are also a combination of two full adders in which these are used to add four bits at a time. So in generally two full adders requires totally 4 XOR gates, 6 AND gates and 4 OR gates. But in the proposed approximate 4:2 compressor only 2 XOR gates are used and 3 AND gates and 4 OR gates are used, important factor in this approach is in the place of ‘2’ carries our method generates only one carry. This will cause errors in 5 places but the power consumption is heavily reduced.

The operation of approximate compressor is given as bellow

\[
M1 = R1 \text{ AND } R2; \quad M2 = R3 \text{ AND } R4; \quad C = M1 \text{ AND } M2
\]

\[
S = (R1 \text{ XOR } R2) \text{ OR } (R3 \text{ XOR } R4) \text{ OR } (M1 \text{ AND } M2)
\]

The bitwise operations are explained in the **TABLE IV**

<table>
<thead>
<tr>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>C</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

6. Architecture-1

**ROUND-I**

![ROUND-I Diagram]

**ROUND-II**

![ROUND-II Diagram]

**ROUND-III**

![ROUND-III Diagram]

7. Architecture-2

**ROUND-I**
8. Results and Comparison

All multipliers have been designed using the Wallace tree architecture. In this we applied for 8-bit multiplier. These architectures are simulated and synthesized by using spartan 3E board of Xilinx FPGA chip XC3S250-5tq144 and power analysis was done. Two different architectures have been proposed in our paper where first design is the approximate adders are used in 3rd column 13th and second method the approximate adders are used from 3rd column to 14th column.

In this analysis is given dearly in table-V. From the report we applied the proposed architecture-1 the conversion was made up to 13th column only. But in the previous method the conversion was made up to 15th column and approximate adders are applied up to 14th column. This results in heavy error rate. We have developed the codes for URDHUA TRIBYAM multiplier of 8-bit, Exact 8-bit multiplier is designed for Wallace tree multiplier. Table V gives the correct information about different analysis like ADP, PDP, DELAY and AREA.

The proposed architecture-1 is having good tolerance value compared to previous works. For example of we consider two digits 255 and 255 the output of an exact multiplier is 65025 but by applying previous architecture. We are getting the output of 49159 where as the proposed architecture-1 results in 50506. Which is very near value to original value.

Similarly the same case is applied for architecture-2 in previous paper the output is 22463 but our proposed architecture-2 is 25993. So this case also proposed architecture is better than previous architecture-1. It has been absorbed that high values of MRE for ACMS.
because of non-zero outputs for inputs with all zeros. This has been resolved in our proposed architecture-1 and architecture-2.

TABLE-VII Ranking of different multiplier architectures depends on their performances

<table>
<thead>
<tr>
<th>Multiplier type</th>
<th>PDP</th>
<th>ADP</th>
<th>MRE</th>
<th>NED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed multipl.-1</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>Multipl.-1[1]</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Proposed multipl.-2</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Multipl.-2[1]</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

Proposed multiplier-1 offers 39% of area savings and 21% of power consumption and proposed multiplier-2 is reducing the area by 43% of old architecture and power 24%. The previous models only covers area 32% for multiplier-1 [1], 19% for SSM [6] and 19% for UDM [7]. But power consumption is very low compared to the multipliers. We have compared all the performances of different adders the proposed multiplier-1 is very good ADP, PDP and NEO. Table-VI gives the total information.

9. Conclusion

In our paper finally we conclude that we have proposed two new approximate multiplier which are generating more near values to the original values. We have taken base from previous papers and usage of same approximate adders like AHA, AFA and compressor (4:2) helps us to get the final outputs. In the models we have modified the approximate adders are used in 3rd column to 13th column 8-bit multiplier architecture-1 and in the second model approximate adders are used from 3rd column to 14th column. We have decreased ADP by 11.9% compare to old multiplier-1 and 12.3% compared to multiplier-2 by proposed architecture-2. Similarly PDP’S also 6.2% reduced by proposed architecture-1 and 5.1% by proposed architecture-2. These two proposed multipliers are reduced the MRE and NED also and can be applied in the placed of old multipliers-1&2.

ACKNOWLEDGEMENT

The authors express sincere gratitude to Department of ECE at K L University for their encouragement during this work. Further, M.sivakumar would like to express his gratitude to DST through grant EEQ/2016/000604.

References

[1] Design of Power and Area Efficient Approximate Multiplier, Suganthi Venkatachalam and Seok-Bum Ko, Senior Member, IEEE


