Efficient FIR filter design using reconfigurable multipliers for VLSI Applications

Mahejabeen[1], Baswaraj Gadgay[2]

[1]Department .VLSI And Embedded Systems VTU CPGS RO Kalaburgi, 585102, India
[2]Professor and Regional Director (I/c), VTU CPGS RO Kalaburgi, 585102, India
mahejabeen800@gmail.com, baswaraj_gadgay@vtu.ac.in

Abstract—**Hardware consumes most of the resources specifically for multiplication process compared to other ALU operations such as addition and subtraction. The most common latency factors like area, power and performance delay are generally controlled by conventional methods such as Array multiplier, Wallace tree and modified Wallace tree multiplier. These factors are important for system performance. The pipelining concepts enhance the performance evaluation. In this paper different multipliers like Wallace tree, LUT and Approximate multipliers are designed and implemented in an 8 – tap FIR filter. It is observed that approximate multiplier is significant for a variable co-efficient and LUT multiplier for fixed co-efficient.**

Keywords— **FIR Filter, Approximate Multiplier, Lookup table, APC, partial products, Odd multiple storage, ADP.**

1. Introduction

In digital systems High speed Multiplication is a primary requirement in performing the digital computations. Advanced multipliers that are popular for its high speed computations and fast performance has a greater impact in the field of technology. A very first column compression multiplier was introduced in 1964 by Wallace where it reduces the partial product of N rows by combining into sets of three different row sets. Later in 1965, a change in Wallace tree is made by the Dadda thus reducing the maximum critical path delay of the multiplier. Thereby Dadda multiplier is slightly fast than the Wallace multiplier also, the hardware requirement of the Dadda multiplier is lesser than the Wallace multiplier [1].
Many of the real time applications doesn't need of an exact solution but rather an acceptable ones. By relaxing the numerical equivalence between the specification and implementation of such applications, approximate computing promises significant energy efficiently improvements [2].

In Arithmetic and logical systems multipliers uses Lookup-tables (LUT) as a memory for different computations. The use of LUTs will be significant if different techniques such as Anti-symmetric product coding (APC) and Odd multiple storage (OMS) techniques used that in turn reduces the size to one-fourth and delay [3]. These calculations have a significant impact in designing of different applications.
Evolution in the semiconductor technologies has raised to a different development of energy efficient design techniques where the energy consumption of machines is rapidly growing in order to process increasing data information. The various Approximated computing techniques have been presented that significantly reduces the delay and size drastically [2].A finite period of the impulse response signal has an efficient use in the emerging signal processing nowadays. All modern electronic systems used in Biomedical applications, Signal processing applications, Mobile applications – DSP applications, the digital Multiplication and Filtering plays a crucial role in the design. Digital filters are an important block in processing DSP systems. It has a significant impact in suppression of unwanted signals and Noise reduction.

$$Y(n) = \sum_{k=0}^{n-1} h(k) * x(n-k)$$

Here, x (n) represents the input sample, y (n) represents the output sample and h(k) represent the filter Co-efficient. The order of the filter is represented by N as each value of the output is weighted sum of the input values. The term x (n-k) represents the delay in the input for multiplication process referred as taps that defines the order of the filter in different forms.

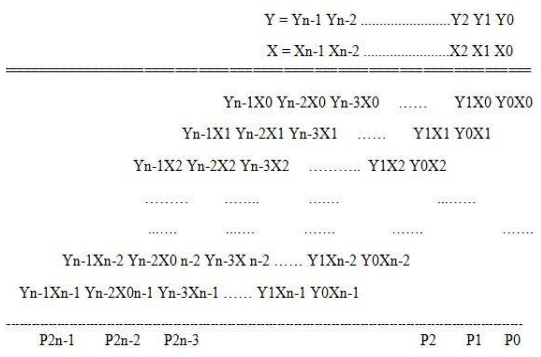
2. FIR FILTER STRUCTURE

Multipliers, adders and delay elements are the basic building blocks of an FIR filter. The multiplication process should be fast enough that the overall throughput should not suffer. Basic adders are being used in combination of multipliers and delay elements to store the sample value as memory for a sample clock cycle [2]. The critical path of the data broadcast FIR filter can be reduced by introducing the pipelining structure. Filter can be further reduced by transposition of the FIR filter where a pipeline Structure is not significant.
A 4-tap FIR filter is constructed where Wallace tree, LUT and Approximate multipliers are been used with the delay elements and an adders.

The multiplication algorithm for N bit multiplicand is as below:

X = Xn-1 Xn-2..................... X2X1 X0

Y= Yn-1 Yn-2 ....Y2Y1 Y0

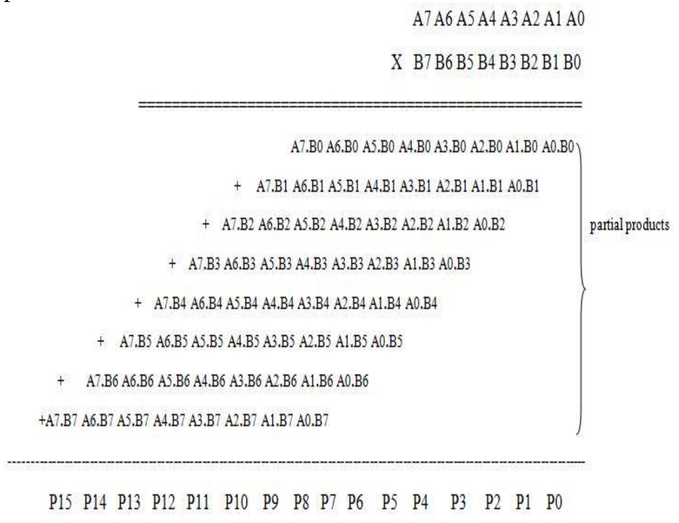


```
                              Y = Yn-1 Yn-2 .......................Y2 Y1 Y0
                              X = Xn-1 Xn-2 ......................X2 X1 X0
=====================================================================
                           Yn-1X0 Yn-2X0 Yn-3X0   ......   Y1X0 Y0X0
                      Yn-1X1 Yn-2X1 Yn-3X1   ......   Y1X1 Y0X1
                 Yn-1X2 Yn-2X2 Yn-3X2   ..........  Y1X2 Y0X2
                   .........   ........    .......        .........
                   .......     .......    .......    .......      .......
        Yn-1Xn-2 Yn-2X0 n-2 Yn-3X n-2 ...... Y1Xn-2 Y0Xn-2
   Yn-1Xn-1 Yn-2X0n-1 Yn-3Xn-1 ...... Y1Xn-1 Y0Xn-1
---------------------------------------------------------------------
     P2n-1    P2n-2    P2n-3                          P2    P1    P0
```

Fig1: Basic structure of Multiplication algorithm

Partial products are generated using AND gates. If N-bits Multiplicand and M-bits multipliers are considered then N*M partial products is obtained.

The multiplication of two 8-bit numbers A and B generates 16 product terms.



```
                                              A7 A6 A5 A4 A3 A2 A1 A0
                                          X  B7 B6 B5 B4 B3 B2 B1 B0
        ==============================================================
                         A7.B0 A6.B0 A5.B0 A4.B0 A3.B0 A2.B0 A1.B0 A0.B0
                    +  A7.B1 A6.B1 A5.B1 A4.B1 A3.B1 A2.B1 A1.B1 A0.B1
                 +  A7.B2 A6.B2 A5.B2 A4.B2 A3.B2 A2.B2 A1.B2 A0.B2
              +  A7.B3 A6.B3 A5.B3 A4.B3 A3.B3 A2.B3 A1.B3 A0.B3      partial products
           +  A7.B4 A6.B4 A5.B4 A4.B4 A3.B4 A2.B4 A1.B4 A0.B4
        +  A7.B5 A6.B5 A5.B5 A4.B5 A3.B5 A2.B5 A1.B5 A0.B5
     +  A7.B6 A6.B6 A5.B6 A4.B6 A3.B6 A2.B6 A1.B6 A0.B6
  +A7.B7 A6.B7 A5.B7 A4.B7 A3.B7 A2.B7 A1.B7 A0.B7
------------------------------------------------------------------------
  P15  P14  P13  P12  P11  P10  P9  P8  P7  P6  P5  P4  P3  P2  P1  P0
```

Fig2: 8X8 Multiplier Structure

The equation for the addition is: P (M + N) = A (m) B (n) $=\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} a_i b_j 2^{i+j}$

I. WALLACE TREE MULTIPLIER

A Wallace tree is an efficient hardware implementation of digital circuits.

The basic steps include:

1. Multiply that is AND each bit of one of the arguments, by each bit of the other, depending on position of the multiplied bits.
2. Reduce the number of partial products to two by layers of full and half adders.
3. Grouping the wires and adding them with the conventional adder.

The initial value is chosen as the largest value such thatdj <min (n1, n2), where n1and n2 are the number of bits in the input multiplication and multiplier. The lesser of the two bit lengths will be the maximum height of each column of weights after the first stage of multiplication.

For each stage j of the reduction, the goal of the algorithm is the reduce the height of each column so that it is less than or equal to the value of d_j reduce each column starting at the lowest-weight column, C_0according to it:

1. If height $(c_i) \leq d_j$ the column does not require reduction, move to column c_{i+1}
2. If height $(c_i) = d_j + 1$ add the top two elements in a half-adder, placing the result at the bottom of the column c_{i+1} and the carry at the top of column then move to column c_{i+1}.
3. Else, add the top three elements in a full-adder, placing the result at the bottom of the column c_{i+1} and the carry at the top of column, restart ci.

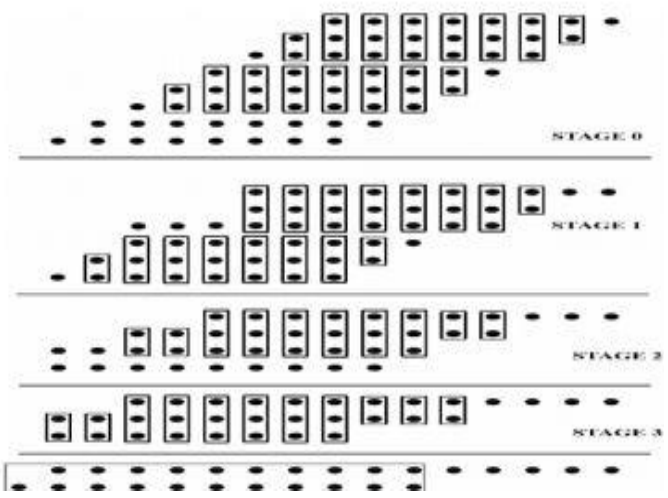


Fig3: Conventional Wallace tree Multiplier

II. APPROXIMATE MULTIPLIER

Multiplier implementation comprises of generation of partial products, partial products reduction tree and a vector merge addition to produce the product from sum and carry rows generated from the reduction tree. Although it's a fact that it consumes more power in processing the partial products reduction tree, thereby approximation s applied in reduction tree stage.

The accumulation of generate signals is performed column wise as each element has a probability of 1/16 of being one, two elements being in the same column even decreases. The probability of error (P_{err}) while using OR gate for reducing the generated signals is high. As the number of generated signal increase, the P_{err} probability increases linearly.

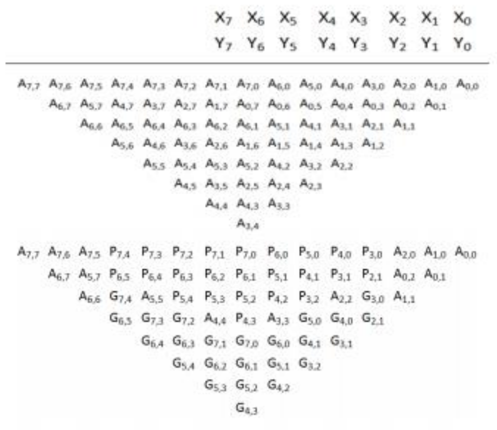


Fig4: Transformation of altered partial products

Thereby the error also rises, to prevent this total generated signals grouped by OR gate is to be placed low. Approximate multiplier handled in such a way that the absolute difference between the actual output and approximated output is always maintained as one. Hence carry outputs are approximated only for the cases where sum is to be approximated. In adders ex-or gates tend to contribute to large area and delay.

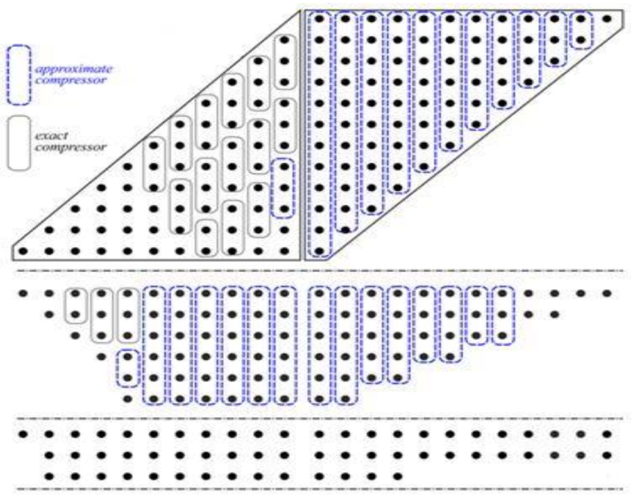


Fig5: Structure of Approximate Multiplier

This Approximate circuit are used in (n-1) least significant columns. This computation returns an inaccurate results rather than a guaranteed one. It is based on the exact computation that requires large amount of latency, allowing all the bounded approximation that provides a disproportionate gain in performance still achieving the acceptable result accuracy. The key requirement in this computing is that approximation can be introduced only in non-critical data that can lead to disastrous consequences.

3. IMPLEMENTATION IN AN FIR FILTER

Different Multipliers are been implemented in an FIR Filter and it is observed that Approximate multiplier plays a significant role in terms of size and power. Whereas, Wallace tree multiplier woks more likely similar to conventional multiplier consuming a huge number of Lookup Tables and less reliable performance with increased delay. It is also observed that both Dadda and Wallace tree multiplier has a common bounded IOBs.

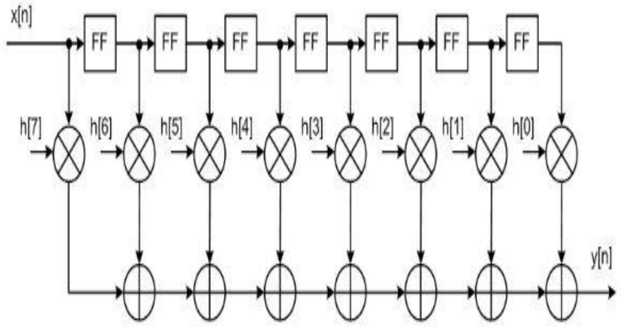
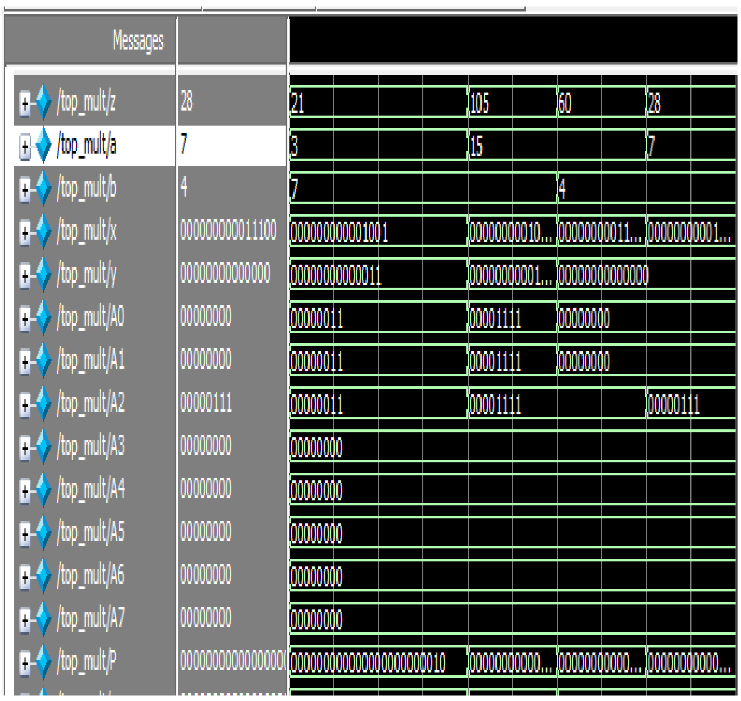


Fig6: 8 tap FIR Filter

4. RESULT AND DISCUSSION

A conventional Wallace tree and approximate multipliers are designed for a variable co-efficient. Although Lookup table Multiplier is designed for fixed co-efficient of multiplicand where the co-efficient factors are predefined within the LUTs. All the three multipliers are implemented in 8-tap FIR filter and it is observed that, since the LUT multiplier is for fixed co-efficient the delay and size of the filter is very less. But, if the same is being designed for variable co-efficient then the latency will be much higher than the Wallace tree and approximate multipliers. Although results show that even for variable co-efficient the approximate multiplier is significant with its latency factors.



```
Messages
/top_mult/z    28                  21                       105            60             28
/top_mult/a    7                   3                        15                            7
/top_mult/b    4                   7                                       4
/top_mult/x    00000000011100      0000000000001001         00000000010... 00000000011... 00000000001...
/top_mult/y    00000000000000      0000000000011            00000000001... 0000000000000
/top_mult/A0   00000000            00000011                 00001111       00000000
/top_mult/A1   00000000            00000011                 00001111       00000000
/top_mult/A2   00000111            00000011                 00001111                      00000111
/top_mult/A3   00000000            00000000
/top_mult/A4   00000000            00000000
/top_mult/A5   00000000            00000000
/top_mult/A6   00000000            00000000
/top_mult/A7   00000000            00000000
/top_mult/P    0000000000000000    0000000000000000000010   00000000000... 00000000000... 00000000000...
```

Fig7: Implementation of 8-tap FIR Filter using Wallace tree Multiplier

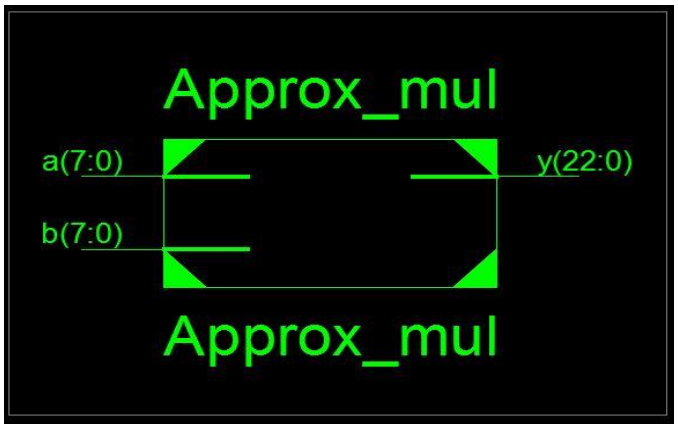


Fig8: Top module of Approximate Multiplier

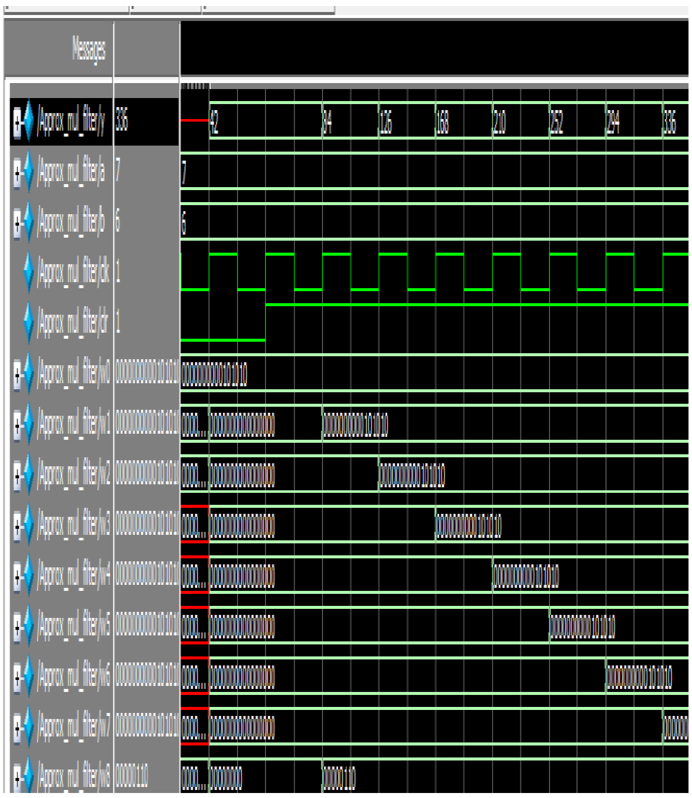


Fig9: Implementation of 8-tap FIR Filter using Approximate Multiplier

REFERENCES

[1] *B. Ramkumar, V. Sreedeep and Harish M Kittur, "A Design Technique for Faster Dadda Multiplier"Member, IEEE.*

[2] *Manish B. TrimaleChilveri, "A review: FIR filter implementation", 2nd IEEE International Conference on Recent trends in Electronics, Information & Communication Technolgy(RTEICT), 2017.*

[3] *Pramod Kumar Meher, "LUT Optimization for Memory-Based Computation", IEEE Transactions on Circuits and Systems II, Volume57, Issue:4, 2010.*

[4] *Mario Osta, Ali Ibrahim, Hussein Chible, Maurizio Valle "Approximate Multipliers Based on Inexact Adders for Energy Efficient Data Processing", NGCAS, IEEE 2017.*

[5] *KokilaBhartijaiswal, Nithishkumar V, Pavithraseshadri, Lakshminarayanan G," adder", International conference on signal processing communication and networking (ICSCN), IEEE, 2015.*