IJCAT International Journal of Computing and Technology, Volume 1, Issue 6, July 2014
ISSN : 2348 - 6090
www.IJCAT.org

A Survey on Power Reduction Techniques in FIR Filter

[1] Pooja Madhumatke, [2] Shubhangi Borkar, [3] Dinesh Katole

[1, 2] Department of Computer Science & Engineering, RTMNU, Nagpur Institute of Technology
Nagpur, Maharashtra 440013, India

[3] Department of Electronics & Telecommunication, RTMNU, Nagpur Institute of Technology
Nagpur, Maharashtra 440013, India

Abstract - There are different parameters need to be focused while designing a VLSI circuit. Some of them are power, area, and speed. Hence these can be referred as challenging problems. Out of these, power dissipation is a critical parameter in modern VLSI design field. Multiplication occurs frequently in finite impulse response (FIR) filters, fast Fourier transforms, discrete cosine transforms, convolution, and to save significant power consumption of a VLSI design, it is a good direction to reduce its dynamic power that is the major part of total power dissipation. This paper summarizes and examines techniques which are involved in multipliers. It broadly covers Booth multipliers, Wallace tree multipliers and Distributed arithmetic Multipliers.

Keywords **- Distributed Arithmetic (DA), FIR filter, Look up table (LUT), Spartan- 3E FPGA.**

1. Introduction

Finite impulse response (FIR) filters are widely used in various DSP applications. In some applications, the FIR filter circuit must be able to operate at high sample rates, while in other applications, the FIR filter circuit must be a low-power circuit operating at moderate sample rates. The low-power or low-area techniques developed specifically for digital filters can be found in. Parallel (or block) processing can be applied to digital FIR filters to either increase the effective throughput or reduce the power consumption of the original filter. While sequential FIR filter implementation has been given extensive consideration, very little work has been done that deals directly with reducing the hardware complexity or power consumption of parallel FIR filters [1].

Traditionally, the application of parallel processing to an FIR filter involves the replication of the hardware units that exist in the original filter. The topology of the multiplier circuit also affects the resultant power consumption. Choosing multipliers with more hardware breadth rather than depth would not only reduce the delay, but also the total power consumption [2]. A lot of design methods of low power digital FIR filter are proposed, for example, in [3] they present a method implementing fir filters using just registered address and hardwired shifts. They extensively use a modified common sub expression elimination algorithm to reduce the number of adders.

Multipliers play an important part in today's digital signal processing (DSP) systems. Examples of their use occur in implementations of recursive and transverse filters, discrete Fourier transforms, correlation, range measurement and in most of these cases it is enough with a multiplier unit design for specific purpose. Multipliers have large area, long latency and consume considerable power. Therefore, low-power multiplier design has been an important part in low-power VLSI system design. The main research hypothesis of this work is that high-level optimization of multiplier designs produces more power-efficient solutions than optimization only at low levels. Specifically, we consider how to optimize the internal algorithm and architecture of multipliers and how to control active multiplier resource to match external data characteristics. The primary objective is power reduction with small area and delay overhead. By using new algorithms or architectures, it is even possible to achieve both power reduction and area/delay reduction, which is strength of high-level optimization. This paper summarizes the approaches which works on the parameters mentioned above. Initially the basic principle of FIR technique is discussed and then the methods of implementation of it are described.

2. FIR Filter Theory

Digital filters are typically used to modify or alter the attributes of a signal in the time or frequency domain. The most common digital filter is the linear time-invariant (LTI) filter. An LTI interacts with its input signal through a process called linear convolution, denoted by y = f * x

where f is the filter's impulse response, x is the input signal, and y is the convolved output. The linear convolution process is formally defined by:

$Y[n] = x[n] * f[n] = \Sigma_{k=0} x[n]f[n-k] = \Sigma_{k=0} f[k]x[n-k]$ (1)

LTI digital filters are generally classified as being finite impulse response (i.e., FIR), or infinite impulse response (i.e., IIR). As the name implies, an FIR filter consists of a finite number of sample values, reducing the above convolution sum to a finite sum per output sample instant. An FIR with constant coefficients is an LTI digital filter. The output of an FIR of order or length L, to an input time-series x[n], is given by a finite version of the convolution sum given in equation, namely

$$y[n]=x[n]*f[n]=\sum_{k=0}^{L-1} f[k]x[n-k] \quad (2)$$

Where f [0] ≠ 0 through f [L-1] ≠ 0 are the filter's L coefficients. They also correspond to the FIR's impulse response. For LTI systems it is sometimes more convenient to express in the z-domain with

$Y(z) = F(z) X(z)$ (3)

Where F (z) is the FIR's transfer function defined in the z-domain by

$$F(z)=\sum_{k=0} f[z]z^{-k} \quad (4)$$

The Lth-order LTI FIR filter is graphically interpreted in Fig.1. It can be seen to consist of a collection of a "tapped delay line," adders, and multipliers. One of the operands presented to each multiplier is an FIR coefficient, often referred to as a "tap weight" for obvious reasons.

Historically, the FIR filter is also known by the name "transversal filter," suggesting its "tapped delay line" structure [4].

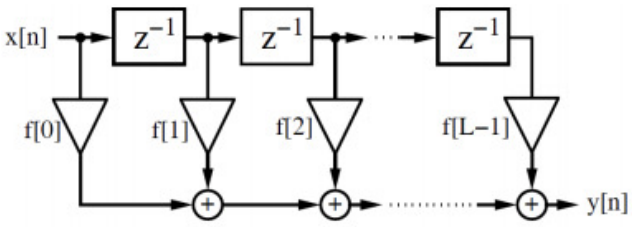

Fig. 1 FIR Filter in Transposed Structure

3. Distributed Algorithm

Distributed arithmetic (DA) is an important FPGA technology.

It is extensively used in computing the sum of products,

$$y[n] = < c, x > = \sum_{n=0}^{N-1} c[n]x[n] \quad (5)$$

DA system, assumes that the variable $x[n]$ is represented by-

$$x[n] = \sum_{b=0}^{B-1} x_b[n] \times 2^b, x_b[n] \in [0,1] \quad (6)$$

If $c[n]$ is the known coefficients of the FIR filter, then output of FIR filter in bit level form is:

$$y = \sum_{n=0}^{N-1} c[n] \times \sum_{b=0}^{B-1} x_b[n] \times 2^b \quad (7)$$

In distributed arithmetic form-

$$y = \sum_{b=0}^{B-1} 2^b \times \sum_{n=0}^{N-1} f(c[n], x_b[n]) \quad (8)$$

In Eq. (8) second summation term realizing as one LUT. The use of this LUT or ROM eliminates the multipliers [6]. For signed 2's complement number output of FIR filter can be computed as-

$$y = -2^B \times f(c[n], x_B[n]) + \sum_{b=0}^{B-1} 2^b \times \sum_{n=0}^{N-1} f(c[n], x_b[n]) \quad (9)$$

Where B represents the total number of bits used. Fig 2 shows the Distributed architecture for FIR filter and different with the MAC architecture.

When $x[n] < 0$, Binary representation of the input is [7],

$$x[n] = -x_b[0] + \sum_{b=1}^{B-1} x_b[n] 2^{-b} \quad (10)$$

The output in distributed arithmetic form-

$$y = -(\sum_{n=0}^{N-1} c[n] x_b[0]) + \sum_{b=1}^{B-1} (\sum_{n=0}^{N-1} c[n] x_b[n]) 2^{-b} \quad (11)$$

If the number of coefficients N is too large to implement the full word with a single LUT (Input LUT bit width = number of coefficients), then partial tables can be added to the results. If pipeline registers are also added, then this modification will not reduce the speed, but can dramatically reduce the size of the design [5].

3.1 Parallel Distributed Arithmetic Architecture

A basic DA architecture, for a length Nth sum-of-product computation, accepts one bit from each of N words. If two bits per word are accepted, then the computational speed can be essentially improved. The maximum speed can be achieved with the fully pipelined word-parallel

architecture as shown in Fig 3. For maximum speed, a separate ROM (with identical content) for each bit vector x_b[n] should be provided [11].

4. Booth Algorithm

Booth's algorithm involves repeatedly adding one of two predetermined values A and S to a product P, then performing a rightward arithmetic shift on P. Let m and r be the multiplicand and multiplier, respectively; and let x and y represent the number of bits in m and r. [8]

1. Determine the values of A and S, and the initial value of P. All of these numbers should have a length equal to (x + y + 1). (a) A: Fill the most significant (leftmost) bits with the value of m. Fill the remaining (y + 1) bits with zeros. (b) S: Fill the most significant bits with the value of (-m) in two's complement notation. Fill the remaining (y + 1) bits with zeros. (c) P: Fill the most significant x bits with zeros. To the right of this, append the value of r. Fill the least significant (rightmost) bit with a zero.

2. Determine the two least significant (rightmost) bits of P. (a) If they are 01, find the value of P + A. Ignore any overflow.
(b) If they are 10, find the value of P + S. Ignore any overflow.
(c) If they are 00, do nothing. Use P directly in the next step.
(d) If they are 11, do nothing. Use P directly in the next step.

3. Arithmetically shift the value obtained in the 2nd step by a single place to the right. Let P now equal this new value.

4. Repeat steps 2 and 3 until they have been done y times.

5. Drop the least significant (rightmost) bit from P. This is the product of m and r.

5. Wallace Tree Multiplier

A Wallace tree is an efficient hardware implementation of a digital circuit that multiplies two integers. The WT multiplier sums up all the bits of the same weights in a merged tree rather than completely adding the partial products in pairs. Full adder (FA) and Half adder (HA) cells are used to add three or two equally weighted bits respectively to produce two bits: the sum bit with a weight equal to that of the operands and the carry bit with a weight equal to one more than that of the operands. The height of the WT is reduced by a factor of 3:2, whenever a FA is used. The final tree is composed of as many levels of FA and HA cells as are necessary to reduce the height of the tree to 2. The hardware synthesis process for a WT multiplier mainly consists of two steps. The first step is to arrange the partial product bits as the initial WT structure, as shown in Fig. 2 for the case of a 4x4 multiplier with operands (a3; a2; a1; a0) and (b3; b2; b1; b0). Secondly, a series of FA and HA transformations are applied on the WT structure until the tree height is reduced to 2. At this point, any n-bit conventional adder may be used to add the remaining two n-bit rows of the tree to get the final multiplication result.

6. Implementation and Results

To evaluate the performance of the Distributed Arithmetic serial and parallel scheme for symmetric FIR filters are implemented and synthesized using Xilinx ISE 10.1 Target as a Spartan 3E (Xc3s100c-5vq100) FPGA device and the results are compared to conventional FIR filter. ISE design software offers a complete design suit based programmable logic devices on Xilinx ISE. The design can be simulated and synthesized in the form of schematic or HDL entry on Xilinx ISE platform. Spartan3E FPGA can be programming directly from Xilinx ISE in configuration logic blocks interconnected with switching matrix. Spartan 3E has a microblaz DSP processor of 325 MHz operating frequency, so that DSP design can be implemented for less resources, high speed and low power. The designed FIR filter is programmed in verilog HDL language [9]. The proposed design is implemented for small memory location LUT and also for large memory location LUT to analyze the performance of the proposed design for speed and area parameters. In the present work, the proposed design is analyzed through 3-tap and 16 –tap DA FIR filters.

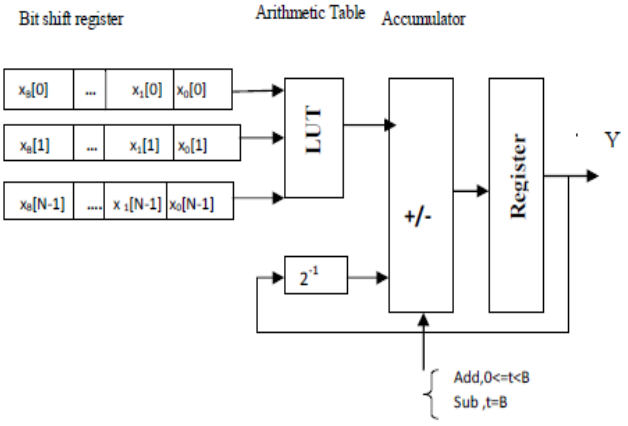

Fig. 2 DA Architecture

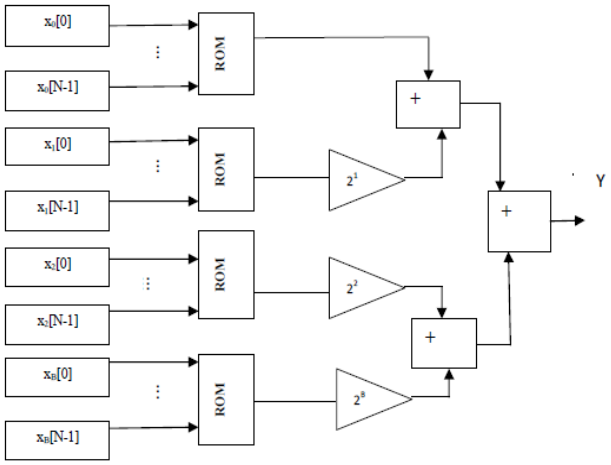

Fig. 3 Parallel DA Architecture

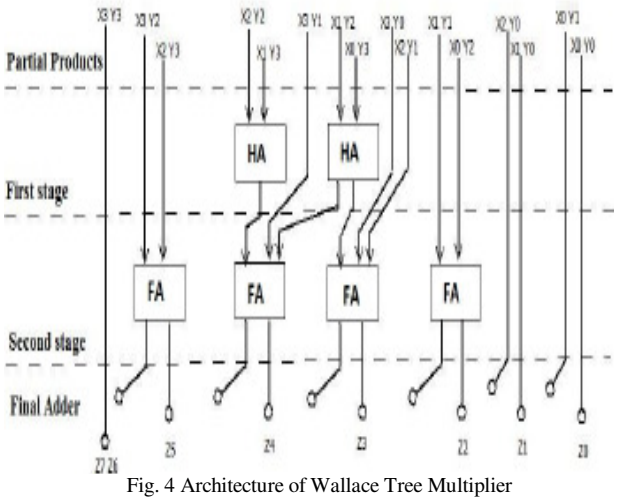

Fig. 4 Architecture of Wallace Tree Multiplier

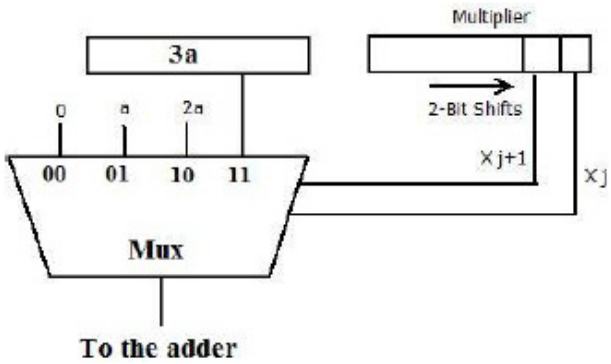

Fig. 5 Booth's Multiplier of Radix 2

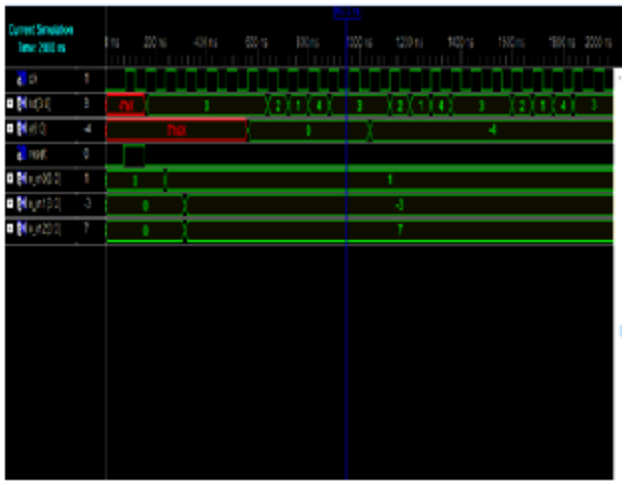

Fig. 6 Simulation Result of serial DA Filter

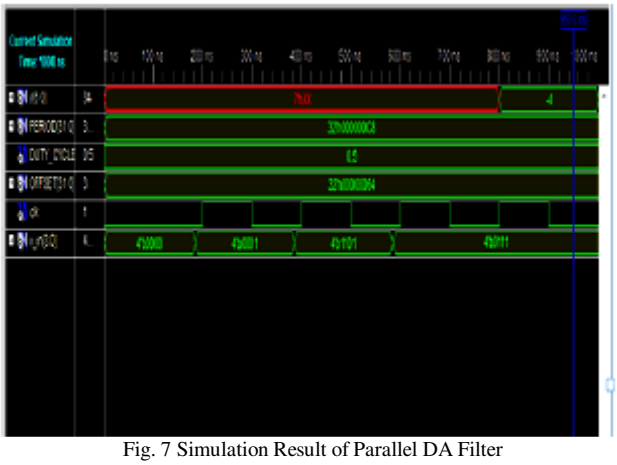

Fig. 7 Simulation Result of Parallel DA Filter

The implementation results of 3-tap and 16-tap FIR filter after applying the distributed arithmetic algorithm as shown in Table 1. The 3-tap parallel DA FIR filter take high speed and lowest power dissipation in comparison to serial DA FIR filter and conventional FIR filter as shown in Table 1. For small tap filters, the serial DA algorithm saves 50 % of the area and cost in comparison to the conventional design techniques. The speed is approximately 2 times for serial DA and 3 times in parallel DA is achieved and very less power is consumed in comparison to simple FIR filter.

Table 1: Comparison of Serial and Parallel DA

Parameter	Direct FIR Filter	Serial DA FIR Filter	Parallel DA FIR filter
Slices	38	25	26
Slice Flip-flops	32	30	44
4 input LUTs	57	47	30
Delay (ns)	13.116	6.413	4.014
Power (mW)	1.5220	1.3134	1.1965

Table 2: Analysis of Multipliers

Parameters	Array Multipli-er	Radix 2 Multipli-er	Radix 4 Multipli-er	Wallac-e Tree Multip-lier
No. of slices	76	55	14	192
No. of 4 input LUT'S	133	99	24	384
No. of bonded inputs	32	34	27	90
No. of bonded outputs	32	34	27	90
Delay (ns)	32.237	18.726	13.798	286.487
Memory (in kb)	72272	70480	69712	113684
Power evaluated (in Watt)	0.143	0.140	0.143	0.155
Power delay Product	4.609	2.621	1.973	44.405

7. Conclusion

The results were analyzed for 3-tap and 16-tap FIR filter using partitioned input based LUT on Xilinx 10.1i as a target of SPARTAN-3E FPGA device. The speed performance of the Parallel DA FIR Filter was superior in comparison to all other techniques. For small tap filter less area, high speed and low power consumption is achieved after applying the Serial and Parallel DA technique. In large-tap FIR filter, speed of parallel DA FIR design technique become 3 times faster than that of conventional FIR filter. The proposed algorithm for FIR filters is also area efficient since approximately 50% of the area is saved with this technique as compared to conventional FIR filter design. Area efficiency and high speed is achieved with parallel DA technique at very slight cost of power consumption for large tap FIR filter. Since, distributed arithmetic FIR filters are area efficient and contained less delay, so these filters can be used in various applications such as pulse shaping FIR filter in WCDMA system, software.

References

[1] Jin-Gyun Chung, Keshab K. Parhi "Frequency Spectrum Based Low-Area Low-Power Parallel FIR Filter Design" EURASIP Journal on Applied Signal Processing 2002, vol. 31, pp. 944–953.

[2] AHMED F. SHALASH, KESHAB K. PARHI "Power Efficient Folding of Pipelined LMS Adaptive Filters with Applications" Journal of VLSI Signal Processing, pp. 199–213, 2000.

[3] Shahnam Mirzaei, Anup Hosangadi, Ryan Kastner, "FPGA Implementation of High Speed FIR Filters Using Add and Shift Method", IEEE, 2006.

[4] Uwe Meyer-Baese, "Digital Signal with Field Programmable Gate Arrays", Springer-Verlag Berlin Heidelberg 2007.

[5] H. Yoo, and D. Anderson, "Hardware-Efficient Distributed Arithmetic Architecture for High-Order Digital Filters", in Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005, Vol. 5, pp. 125 – 128.

[6] T.Vigneswarn and P.Subbarami Reddy"Design of Digital FIR Filter Based on DDA algorithm" Journal of Applied Science, 2007.

[7] Stanley A. White,"Application of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review" IEEE Acoustic speech signal processing Magazine, July 1989.

[8] Tisserand, "Low-power arithmetic operators," in Low Power Electronics Design, C. Piguet, Ed. CRC Press, Nov. 2004.

[9] Samir Palnitkar, "Verilog HDL A guide to Digital Design and Synthesis" Second Edition-2007.

Pooja Madhumatke has done B. Tech from S.N.D.T University, Mumbai in Electronics and Communication. Currently she is a final year (M.E. student) pursuing her post graduation in the field of Embedded System & Computing from Nagpur University. She has published papers in International Conference and attended an International Conference held in year 2014. Currently she is working for her final year project in the same field of filters.

Prof. Shubhangi Borkar has done Engineering from Nagpur University in Computer Science and Technology. She has also completed her post graduation from Nagpur University. Currently she is working as Assistant Professor in NIT, Nagpur University. She has published several papers in International and National conferences.

Prof. Dinesh Katole has done Engineering from Nagpur University in Electronics and Telecommunication. He has also completed his post graduation from Nagpur University. Currently he is working as

Assistant Professor in NIT, Nagpur University. He has published several papers in International and National conferences. He is also continuing his P.hd program.