Implementation of Folded FIR Filter Direct Form using VHDL Programming

K. Subramanian
Department of Electronics and Communication Systems
Karpagam University, Coimbatore, Tamilnadu, India

Abstract- *The design of folded finite-impulse response (FIR) filters based on pipelined multiplier arrays is presented in this paper. The design is considered at the internal delays and bit-level of the pipelined multiplier array are fully exploited in order to reduce hardware complexity and power consumption, direct FIR filter forms is considered. The proposed schemes are compared as to the aspect of hardware complexity with a straightforward implementation of a folded FIR filter based on the pipelined Wallace Tree multiplier. The comparison reveals that the proposed schemes require less hardware. Finally, efficient implementation of folded FIR filter circuits is presented when constraints in area, power consumption and clock frequency.*

Keywords- Filter, Folded, Direct, FIR, Adder, Multiplier, Pipeline, Array

I. INTRODUCTION

Nowadays, digital signal processing (DSP) is used in a wide variety of real-time applications and is playing an important role in the digital revolution. Finite-impulse response (FIR) digital filters are the most fundamental DSP components. FIR filters have the advantage of easy implementation and stability but the large number of filter taps leads to excessive hardware complexity. Power consumption is another important factor in DSP circuits, especially when used in mobile communication systems.

Folding techniques have been proposed as a means of reducing hardware when the processing throughput required by the application is less than the throughput at which the circuit can operate. FIR filters are ideal candidates for folding since they are essentially a repetition of multiplications. Reconfigurable folded direct form FIR filter architecture has been presented in but not detailed implementation is given, although the bit-level design can lead to hardware efficient circuits. An important issue in synchronous designs is the avoidance of problems related to clock-skew.

A number of asynchronous architectures have been presented in the bibliography as a solution to this problem at the expense of extra hardware. A significant advantage of the folded FIR architectures is that they lead to reduced hardware in comparison with the corresponding unfolded schemes and the clock-skew problem does not exist. Thus there is no need to resort to hardware expensive asynchronous solutions.

In this paper, we propose a folded scheme based on pipelined multiplier arrays for the implementation of FIR filters in direct form. Multiplier arrays have canonic structure and can achieve high throughput since they can be pipelined at the bit-level. We suggest the exploitation of the internal pipelining delays of the array for storing filter coefficients and data words. The result is significant hardware reduction. The Wallace Tree multiplier pipelined at the bit-level has lower latency than the array multipliers but its does not allow the internal pipelining delays to be exploited. The design of direct form is considered. Detailed designed circuits based on possible combinations of filter forms and array types are presented, comparing the results from the aspect of hardware complexity. A way to combine the merits of folded and unfolded filters is to cascade a number of folded FIR filter units.

The partially folded filter is an intermediate form between the folded and unfolded form of a filter, featuring higher throughput than the folded and requiring less hardware than the unfolded. In this work the effect of the number of cascaded filter units in the area, power consumption and operation frequency has been studied. A technique for the optimal selection of the number of cascaded filter stages is presented which takes into account the design limitations of the requested filter.

II. VLSI

VLSI stands for "Very Large Scale Integration". This is the field which involves packing more and more logic devices into smaller and smaller areas. Thanks to VLSI, circuits that would have taken boardfuls of space can now be put into a small space few millimeters across! This has opened up a big opportunity to do things that were not possible before. VLSI circuits are everywhere computer, car, brand new state-of-the-art digital camera, the cell-phones. All this involves a lot of expertise on many fronts within the same field.

VLSI has been around for a long time, there is nothing new about it, but as a side effect of advances in the world of computers, there has been a dramatic proliferation of tools that can be used to design VLSI circuits. Alongside, obeying Moore's law, the capability of an IC has increased exponentially over the years, in terms of computation power, utilization of available area, yield. The combined effect of these two advances is that people can now put diverse functionality into the IC's, opening up new frontiers. Examples are embedded systems, where intelligent devices are put inside everyday objects, and ubiquitous computing where small computing devices proliferate to such an extent that even the shoes you wear may actually do something useful like monitoring your heartbeats! These two fields are kind a related, and getting into their description can easily lead to another article.

DEALING WITH VLSI CIRCUITS

Digital VLSI circuits are predominantly CMOS based. The way normal blocks like latches and gates are implemented is different from what students have seen so far, but the behavior remains the same. All the miniaturization involves new things to consider. A lot of thought has to go into actual implementations as well as design. Let us look at some of the factors involved.

1. Circuit Delays: Large complicated circuits running at very high frequencies have one big problem to tackle - the problem of delays in propagation of signals through gates and wires. even for areas a few micrometers across!

2. Power. Another effect of high operation frequencies is increased consumption of power. This has two-fold effect - devices consume batteries faster, and heat dissipation increases. Coupled with the fact that surface areas have decreased, heat poses a major threat to the stability of the circuit itself.

3. Layout: Laying out the circuit components is task common to all branches of electronics. What’s so special in our case is that there are many possible ways to do this; there can be multiple layers of different materials on the same silicon, there can be different arrangements of the smaller parts for the same component. The power dissipation and speed in a circuit present a trade-off; if we try to optimize on one, the other is affected. The choice between the two is determined by the way we chose the layout the circuit components. Layout can also affect the fabrication of VLSI chips, making it either easy or difficult to implement the components on the silicon.

THE VLSI DESIGN PROCESS

A typical digital design flow is as follows:

→Specification
→Architecture
→RTL Coding
→RTL Verification
→Synthesis
→Backend

Tape Out to Foundry to get end product

All modern digital designs start with a designer writing a hardware description of the IC (using HDL or Hardware Description Language) in Verilog/VHDL. A Verilog or VHDL program essentially describes the hardware (logic gates, Flip-Flops, counters etc) and the interconnection of the circuit blocks and the functionality. Various CAD tools are available to synthesize a circuit based on the HDL. Without going into details, we can say that the VHDL can be called as the "C" of the VLSI industry. VHDL stands for "VHSIC Hardware Definition Language", where VHSIC stands for "Very High Speed Integrated Circuit". This language is used to design the circuits at a high-level, in two ways. It can either be a behavioral description, which describes what the circuit is supposed to do, or a structural description, which describes what the circuit is made of. There are other languages for describing circuits, such as Verilog, which work in a similar fashion. Both forms of description are then used to generate a very low-level description that actually spells out how all this is to be fabricated on the silicon chips. This will result in the manufacture of the intended IC.

ADVANTAGES OF VLSI IMPLEMENTATION OF FIR FILTER

The proposed schemes are compared as to the aspect of hardware complexity with a straightforward implementation of a folded FIR filter based on the pipelined Wallace Tree multiplier. The comparison reveals that the proposed schemes require 20%–30% less hardware. Finally, efficient implementation of folded FIR filter circuits is presented when constraints in area, power consumption and clock frequency are given.

Low power consumption because each of the devices consumes only a little amount of power. In a switching circuit most of the power is consumed switching the charge on the capacitors in that connect the switches to each other. Less testing is required. If you built the same circuit out of discrete ICs or other DSPs, each IC has to be tested (before you use it) for the many different ways it could be used in different applications. for 10000 ICs this is a lot of testing. In a VLSI the components are dedicated to a single use. Further, most are located in the middle of the VLSI and there is no access to them for testing. All you can test is the function the entire circuit was designed for reliability. Over time, we have found that the reliability of an IC is a function of how many connections it has to the outside world. So if the function is constructed with many smaller ICs connected together, then there are many connections, and the reliability is lower. The VLSI has fewer connections, and higher reliability.

III. FIR FILTER

In signal processing, a filter removes unwanted parts of the signal, such as random noise, or extracts the useful parts of the signal, such as the components lying within a certain frequency range.

In signal processing, there are many instances in which an input signal to a system contains extra unnecessary content or additional noise, which can degrade the quality of the desired portion. In such cases we may remove or filter out the useless samples. For example, in the case of the telephone system, there is no reason to transmit very high frequencies since most speech falls within the band of 400 to 3,400 Hz. Therefore, in this case, all frequencies above and below that band are filtered out. The frequency band between 400 and 3,400 Hz, which isn't filtered out, is known as the pass band, and the frequency band that is blocked out is known as the stop band.

FIR, Finite Impulse Response, filters are one of the primary types of filters used in Digital Signal Processing. FIR filters are said to be finite because they do not have any feedback. Therefore, if you send an impulse through the system (a single spike) then the output will invariably become zero as soon as the impulse runs through the filter.

Characterizing digital FIR filters

There are a few terms used to describe the behavior and performance of FIR filter including the following:

- Filter Coefficients - The set of constants, also called tap weights, used to multiply against delayed sample values. For an FIR filter, the filter coefficients are, by definition, the impulse response of the filter.
- Impulse Response – A filter's time domain output sequence when the input is an impulse. An impulse is a single unity-valued sample followed and preceded by zero-valued samples. For an FIR filter the impulse response of a FIR filter is the set of filter coefficients.
- Tap – The number of FIR taps, typically N, tells us a couple things about the filter. Most importantly it tells us the amount of memory needed, the number of calculations required, and the amount of "filtering" that it can do. Basically, the more taps in a filter results in better stop band attenuation (less of the part we want filtered out), less rippling (less variations in the pass band), and steeper roll off (a shorter transition between the pass band and the stop band).
- Multiply-Accumulate (MAC) – In the context of FIR Filters, a "MAC" is the operation of multiplying a coefficient by the corresponding delayed data sample and accumulating the result. There is usually one MAC per tap.

Existing FIR Filter Structures

The direct-form and transpose-form structures are most commonly used to implement FIR filters. For certain special filters, recursive implementations require less computation. Lattice and cascade structures are occasionally also used.

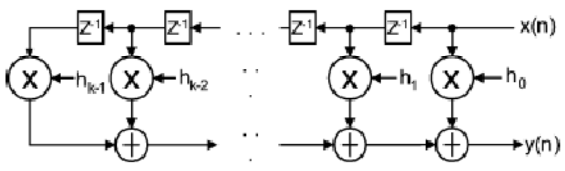

Fig. 3.1 FIR – Direct Form

There are no closed loops (no feedback) in this structure, so it is called a non-recursive structure. Since any FIR filter can be implemented using the direct-form, non-recursive structure, it is always possible to implement an FIR filter non-recursively. However, it is also possible to implement an FIR filter *recursively*, and for some special sets of FIR filter coefficients this is much more efficient.

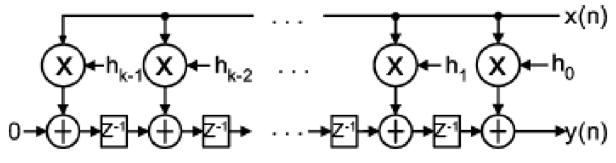

Fig. 3.2 FIR – Transposed Form

The flow-graph-reversal theorem says that if one changes the directions of all the arrows, and inputs at the output and takes the output from the input of a reversed flow-graph, the new system has an identical input-output relationship to the original flow-graph.

IV. PIPELINED FIR FILTER ARCHITECTURES

FOLDED FIR FILTER ARCHITECTURES

Folding allows several almost identical operations performed on a stream of data to be time-multiplexed by the same circuit unit. In order to achieve this, the internal clock of the circuit unit must be a multiple of the input data rate. Folding is mainly used in applications where the required

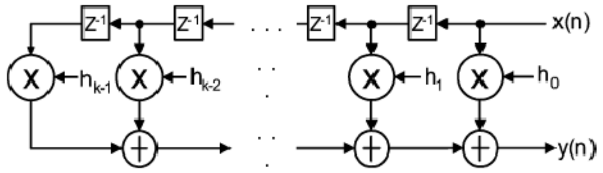

Fig. 4.1. Structure of an unfolded k-tap FIR filter in direct form.

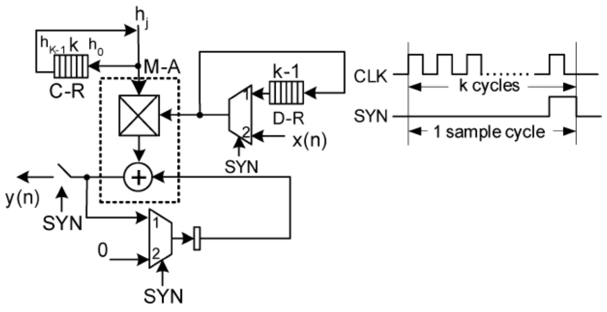

Fig. 4.2. Folded architecture of a k-tap FIR filter in direct form.

throughput is significantly lower compared to the maximum throughput at which the circuit can operate. In the case of the FIR filters, the time-multiplexed operations are the filter tap multiplications and their addition. The circuit unit that performs this operation is a multiplier combined with an adder and from now on it will be referred to as a multiply-add unit (M-A). In Fig. 4.1, the structure of an unfolded -tap FIR filter in direct form is shown.

In Fig. 4.2, a folded implementation of this filter is given. It consists of an M-A unit that performs one multiplication and addition at each clock cycle.k such operations are required to produce one result at the output. The filter coefficients are stored in a cyclic shift register coefficient register (C-R) in descending order (h_0 at the right end of the register). The k-1 samples of x are stored in the cyclic shift register data register (D-R) in ascending order (the oldest at the left end of the register).

The timing diagram in Fig. 4.2 clarifies the operation of the circuit. The computation cycle lasts k clock cycles. During those k cycles the most recent samples as well as the coefficients are cyclically shifted in order to compute the convolution. The term $x(n-k-1)h_{k-1}$ is computed first and $x(n)h_0$ last. The input 2of both multiplexers shown in Fig. 4.2 is activated when the control signal SYN is high. At the last clock of the operation cycle, when the control signal SYN is high, a new input sample x(n) enters the circuit and replaces the oldest stored input sample x(n-k-1) in D-R. At the same clock, the delay element of the accumulator is cleared while the switch at the output of the circuit closes and the final result is produced. Thus the circuit is of immediate response. The circuit operates at f_c clock frequency. An input sample is processed every clock cycles and therefore the filter operation frequency is $f_s = f_c /k$. The frequency of the control signal SYN is f_s. There is a close correspondence between the folded and the unfolded architectures presented in Figs.4.1 and 4.2. The D-R at the data input corresponds to the delay elements in the data line in Fig.4.1. The C-R corresponds to the data latches where the filter coefficients are stored when the filter is programmed. These latches are not shown in Fig.4.1, but are implied and thus the filter operation frequency $f_s = f_c /k$ is equal to the control signal SYN.In folded schemes, the input/output throughput is $1/T_{MA}k$, where T_{MA} the delay of the M-A unit is and is the number of filter taps. Pipelining the M-A unit increases this throughput at the expense of a number of extra delay elements proportional to the number of the pipelined stages.

PROPOSED FOLDED SCHEMES FOR DIRECT FIR FILTERS

From now on, it is assumed that the M-A unit is internally pipelined at the bit-level in order for the circuit throughput to be maximized. In this section we present the use of the carry propagate and carry–save multiplication array to implement thefolded direct FIR filter architecture shown in Fig.4.2. The internal delays of the multiplier have been drawn separately from the multiplier array, which is denoted by a box with a cross, in order to clearly indicate how they can be used for the implementation of the registers C-R and D-R shown in Fig. 4.2.

The symbols m and b denote the bit-length of x and h. The symbols x and h denote that the two signals enter the circuit in bit-skew and double bit-skew form respectively. The register R_x (shift register for x) corresponds to the delays in the lines of x, R_h (shift register for h) corresponds to the delays in the lines of h and the double shift register CS-R (carry–save shift register) corresponds to the delays in the carry and sum lines of the array. During the operation of the multiplier, successive values of x(n), h_j, sums and carries are shifted inside the corresponding registers.

The above scheme can implement the architecture of Fig. 4.2 by using the register R_x as D-R, the register R_h as C-R and applying the necessary modifications shown in Fig.4.1.1. According to Fig. 2 the lengths of D-R and C-R must be k-1 and k bits respectively. Therefore the registers R_x and R_h are extended by k-2m+1 and k-b+1 bits by connecting additional registers outside the multiplier. This implementation is shown in Fig. 4.1.1.

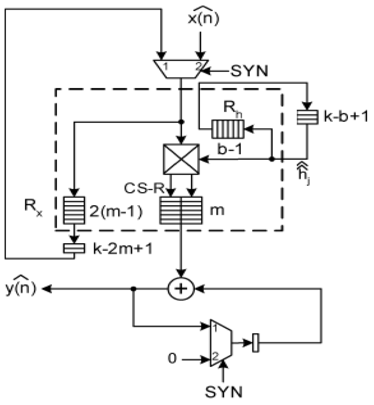

Fig. 4.1.1. Folded k-tap FIR filter in direct form based on a pipelined carry–propagate multiplier.

The circuit, drawn in detail in Fig. 3.1.2, implements the above filter scheme for k = 8 filter taps. The i-order bit of the input sample x(n), the coefficient $h_j$and y(n) the output is represented as $x_i(n)$,h^i_j and $y_i(n)$, respectively. The same notation is used in all figures in the rest of the project report.

The dashed lines represent the accumulation feedback loop. The feedback lines of x(n) and y(n) are not shown in order the complexity of the schematic to be kept low. In this example the length of R_x is equal to k-2, so one external delay for is required. The length of R_h is b=4 (less than k), so five additional delays in each bit line of h_j have been added on the right side of the multiplier. The external delays are represented with empty boxes and the internal delays with filled boxes.

The diagonal row of cells at the right edge of the array is used for the accumulation of the least significant part of the result. For the accumulation of the most significant part, an additional row of full-adders has been added at the bottom of the array. This row is extended leftwards by $\log_2 k$ full-adders to avoid accumulation overflows.

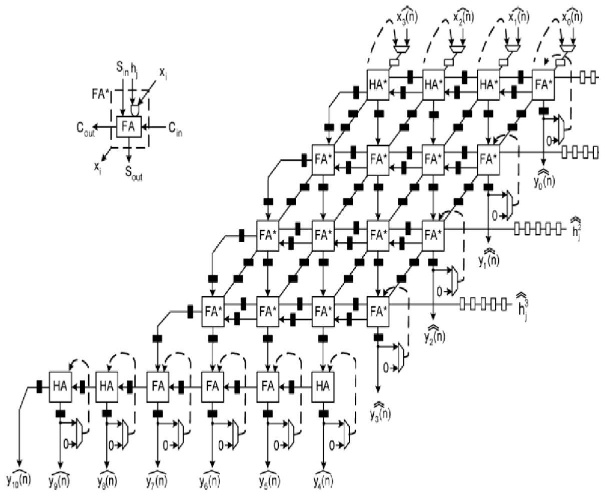

Fig. 4.1.2. Circuit implementation of a folded 8-tap FIR filter in direct form, basedon a pipelined carry–propagate multiplier.

Structure of an unfolded k-tap FIR filter in direct form

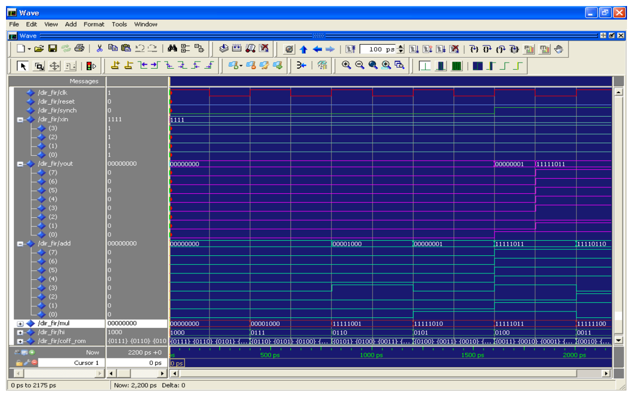

Fig.1 Output Waveform for Direct Form Architecture

Folded architecture of a k-tap FIR filter in direct form

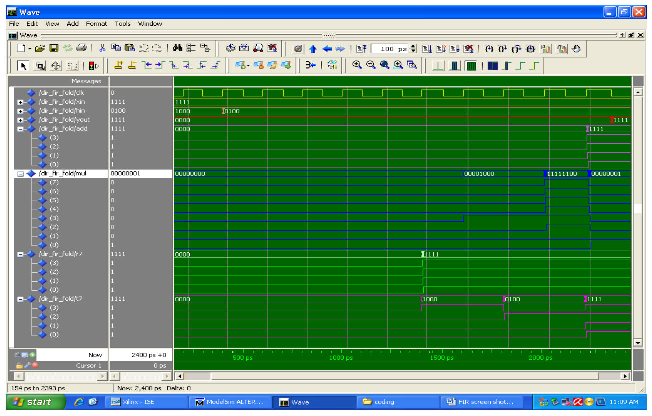

Fig. 2 Output Waveform for Direct Form Folded Architecture

V. CONCLUSION

In this project, high throughput folded FIR filters schemes based on bit-level pipelined multiplier arrays are proposed. The suggested exploitation of the internal pipelining registers of these multiplier arrays results in four schemes that have less hardware than a straightforward implementation based on a Wallace Tree multiplier pipelined also at the bit-level. The proposed folded FIR filters can be cascaded to implement partially folded FIR filters. In this case, the choice of the most suitable of the proposed schemes depends on the number of filter taps and the requirements for the operational frequency and hardware complexity. For an implementation under speed and transistors number constraints the designer can make use of graphs that permit the optimal choice of the degree of folding and the architecture of the proposed schemes.

The above ideas can be also applied to the implementation of folded IIR filters. The operation of all the circuits presented in this project has been verified by extensive simulation in Xilinx Spartron3E.

REFERENCES

[1] M. Kahrs et al., “The past, present and future of audio signal processing,” IEEE Signal Process. Mag., vol. 14, no. 5, pp. 30–57, Sep. 1997.

[2] T. H. Meng, A. C. Hung, E. K. Tsen, and B. M. Gordon, “Low-power signal processing system design for wireless applications,” IEEE Pers. Commun., vol. 5, no. 3, pp. 20–31, Jun. 1998.

[3] B. G. Haskell, P. G. Howard, Y. A. LeCun, A. Puri, J. Ostermann, M. R. Civanlar, L. Rabiner, L. Bottou, and P. Haffner, “Image and video coding–Emerging standards and beyond,” IEEE Trans. Circuits Syst. Video Technol., vol. 8, no. 7, pp. 814–837, Nov. 1998.

[4] P. G. Paulin, C. Liem, T. C. May, and S. Sutarwala, “DSP design Requirements for embedded systems: A telecommunications industrial perspective,” J. VLSI Signal Process., vol. 9, pp. 23–47, Jan. 1995.

[5] A. V. Oppenheim and R. W. Schafer, Discrete-Time Signal Processing. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[6] R. Amirtharajah and A. P. Chandrakasan, “Self-powered low power signal processing,” in IEEE Symp. VLSI Circuits Dig. Tech. Papers, Jun. 1997, pp. 25–26.

[7] K. K. Parhi, C.-Y.Wang, and P. Brown, “Synthesis of control circuits in folded pipelining DSP architectures,” IEEE J. Solid-State Circuits, vol. 27, no. 1, pp. 29–43, Jan. 1992.

[8] K. K. Parhi, “Calculations of minimum number of registers in arbitrary life time chart,” IEEE Trans. Circuits Syst. II, vol. 41, no. 6, pp. 434–436, Jun. 1994.

[9] K. K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation. New York: Wiley-Interscience, 1999.

[10] S. Ramanathan, S. K. Nandy, and V. Visvanathan, “Reconfigurable filter coprocessor architecture for DSP applications,” J. VLSI Signal Process., vol. 26, pp. 333–359, Nov. 2000.

[11] S. Hauck, “Asynchronous design methodologies: an overview,” Proc. IEEE, vol. 83, no. 1, pp. 69–93, Jan. 1995.

[12] W. Kuang, J. S. Yuan, R. F. DeMara, D. Ferguson, and M. Hagedorn, “A delay-insensitive FIR Filter for DSP applications,” in Proc. 9th Annu. NASA Symp. VLSI Design, Albuquerque, NM, Nov. 2000, pp. 135–165.

[13] S. Y. Kung, “On supercomputing with systolic/ wavefront arrays,” Proc. IEEE, vol. 72, no. 7, pp. 867–884, Jul. 1984.

[14] C. R. Baugh and B. Wooley, “A two’s complement parallel array Multiplication algorithm,” IEEE Trans. Computers, vol. C-22, no. 12, pp.

[15] C. S. Wallace, “A suggestion for fast multiplier,” IEEE Trans. Electron. Comput., vol. EC-13, pp. 14–17, Feb. 1964.

[16] N. H. E. Weste and K. Eshraghian, Principles of CMOS VLSI Design, 2nd ed. Burlington, MA: Addison-Wesley, 1993.