A Methodology for Architecture Synthesis of Cascaded IIR Filters on TLU FPGAs

G.N. Rathna, Dept. of Elec. Engg.
S.K. Nandy, CAD Lab, SERC
K. Parthasarathy, Dept. of Elec. Engg.
Indian Institute of Science
Bangalore, 560 012

Abstract

In this paper, we propose an architecture synthesis methodology to realize cascaded Infinite Impulse Response (IIR) filter in Table Look Up (TLU) Field Programmable Gate Arrays (FPGA). The synthesis procedure involves a systematic transformation of the Dependance Graph (DG) corresponding to the cascaded IIR filter to a Pipelined Fixed Full Size Array (PFFSA). We offer an implementation of a cascaded 8th order IIR filters on Xilinx XC3090 FPGA devices.

1 Introduction

Infinite Impulse Response (IIR) and Finite Impulse Response (FIR) digital filters are perhaps the two most fundamental tools in the digital signal processing (DSP) applications. High speed applications like video, radar and image processing require high-sample rates while some applications like speech and communication require slow or moderate sample rates. When speed is the primary goal, dedicated hardware is required for implementation of filter taps. This enforces the ideal coefficients to be represented as a set of finite precision coefficients. This finite representation is a source of error in the frequency spectrum. Such errors are critical in certain filter applications, because this error may cause inexact pole-zero cancellation.

In the direct form representation of the IIR filters [1,2], quantization of one pole disturbs the configuration of other poles which may lead to unstable filter. One method of reducing this instability is by increasing the wordsize of the internal arithmetic unit [3] or by representing the filter in the cascaded form. Each stage in the cascaded form representation is a second order structure. Since each pair of complex conjugate poles is realized independent of all the other poles, (previous stage or post stage poles) the cascade structure is generally less sensitive to coefficient quantization than the direct form. Further, the simulatiom of such systems can be restricted to individual stages of the cascaded structure. In this paper, we address the issues related to synthesis of VLSI architectures for cascaded IIR filters.

VLSI solutions offer the advantages of encapsulating complex systems in a single chip with enhanced performance of the overall system. Realizing complex system on silicon is to a large extent determined by the system architecture. A high performance VLSI system architecture must ensure regular structure with localized communication. These considerations favour implementations which feature arrays of identical or easily programmable Processing Elements (PE) with localized interconnections for reduced communication cost. The regularity in the cascaded stages of the IIR filter makes it an ideal candidate for VLSI implementation.

2 IIR filters

We give below a few definitions and introduce terms to be used in the synthesis of cascaded IIR filters. We also assume that the system clock (T_{clock}) matches the data rate (T_{data}). i.e., $T_{clock} = T_{data}$

Latency of the filter L is defined as the delay through the longest combinational path (τ_c) between any two input and output registers.

Throughput of the filter $F = 1/T_{data}$

We consider a case where , T_{data} is much higher than the τ_c (i.e., $T_{data} > \tau_c$). Clearly, the hardware that implements the combinational logic is underutilized, which can be enhanced by a factor $\lfloor T_{data}/\tau_c \rfloor$ by suitably multiplexing and pipelining the hardware. Achieving this high performance implementation in recursive (Ex. IIR filters) systems is a challenge, since the recursion or the internal feedback counters restricts any improvement in performance. This is because the latency associated with the feed back loop in recursive systems limits the pipelining and/or parallel processing. In non-recursive (Ex.FIR filters) systems, latches can be placed across any feed-forward cutset without changing the transfer function and achieve the desired level of pipelining. However, recursive systems cannot be pipelined at any arbitrary level by simply inserting latches, since the pipelining latches would change the number of delay operators in the loop, and hence the transfer function of the implementation. In this paper, we discuss the architectural synthesis of cascaded IIR filters.

The difference equation that identifies an IIR filter is given by

$$y(n) = \sum_{k=0}^{M} b_k x(n-k) + \sum_{k=1}^{N} a_k y(n-k) \qquad (1)$$

0-8186-4990-9/94 $3.00

where b_k & a_k are the coefficients of the filter and $x(n)$ & $y(n)$ are the inputs and outputs respectively. M & N represents the number of zeroes and poles of the filter respectively.

The rational system function for an IIR filter is given by

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}} \quad (2)$$

The direct form representation of equation (2) is as shown in Fig 1. By factoring the numerator and denominator of (2), we get

$$H(z) = A \frac{\prod_{k=1}^{M_1}(1 - g_k z^{-1}) \prod_{k=1}^{M_2}(1 - h_k z^{-1})(1 - h_k^* z^{-1})}{\prod_{k=1}^{N_1}(1 - c_k z^{-1}) \prod_{k=1}^{N_2}(1 - d_k z^{-1})(1 - d_K^* z^{-1})} \quad (3)$$

where $M = M_1 + M_2$ & $N = N_1 + N_2$.

The first order factors represent the real zeros at g_k and real poles at c_k. The second order form represents the complex conjugate pairs of zeros at h_k & h_k^* and poles at d_k & d_k^*. A cascaded structure is obtained by combining pairs of real factors and complex conjugate pairs into second order factors and is given by the expression,

$$H(z) = \prod_{k=1}^{N_s} \frac{b_{0k} + b_{1k} z^{-1} + b_{2k} z^{-2}}{1 - a_{1k} z^{-1} - a_{2k} z^{-2}} \quad (4)$$

Where $N_s = \lfloor (N+1)/2 \rfloor$ & N is the order of the filter. M & N are assumed to be equal and even. If one of them is odd, then the second order section in (4) will be zero in the last product. Fig 2 shows the cascade structure for 8th order filter, i.e., $N = 8$ and $N_s = 4$.

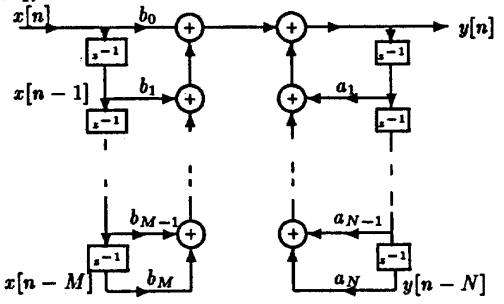


Fig. 1. Block diagram representation for a general Nth - order difference equation.

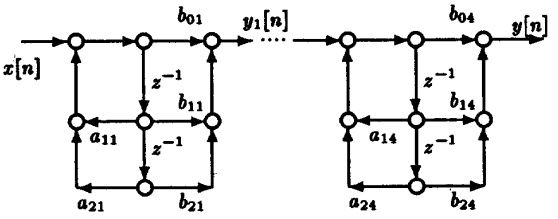


Fig. 2. Cascade structure for a 8th order system

The difference equation of one stage of equation (4) is given by

$$y_1(n) = b_{00}x(n) + b_{10}x(n-1) + b_{20}x(n-2) + a_{10}y_1(n-1) + a_{20}y_1(n-2) \quad (5)$$

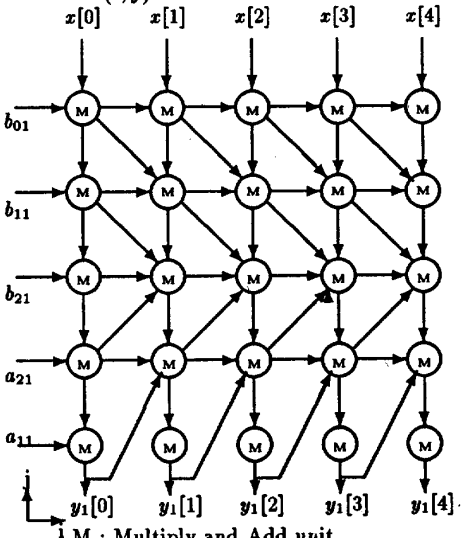
3 IIR Filter Architecture

The architecture of the cascaded IIR filter is derived from the system equation of a cascade stage of Fig. 2. In the following, we introduce additional terms that are used to describe the architecture synthesis procedure for a cascade section of the IIR filter.

Dependence Graph (DG) : A DG is a data flow graph(DFG) that shows the dependence of the computations in an algorithm.

Shift-Invariant DG : A DG is a shift-invariant, if the dependence arcs corresponding to all nodes in the index space are independent of their positions. If the existance of an edge $I_1 \rightarrow I_1 + e$ with i delays implies the existance of an edge $I_1 + I_2 \rightarrow I_1 + I_2 + e$ with the same number of i delays for all indexes I_2 permissible within the index set of the DG, then the DG is regular.

Consider a two-dimensional DG for one stage of the cascaded IIR filter (5) as shown in Fig. 3. A computation in the DG is represented by an Index Vector $c = (i, j)^T$.



M : Multiply and Add unit

Fig. 3. DG of IIR filter for one - stage.

If each node in the DG is assigned to one Processing Element (PE), we obtain a trivial implementation of an IIR filter on a 2-d processor array. This results in a very inefficient utilization of the PEs, since each PE is active only for a small fraction of the computation time. In order to improve PE utilization, the nodes of the DG are projected onto Virtual Full Size PE-Array (VFSA) such that $p^T\vec{d} = 0$, where p is the processor basis and $\vec{d}$ is the projection direction [4].

Consider a projection vector $\vec{d} = [1,0]^T$ to project the node of the DG onto the VFSA. Physically this corresponds to the projection of all operations located parallel to j-axis (i.e., all operations with the same i-index values but different j-index values) onto the same processor. This is shown in Fig. 4.

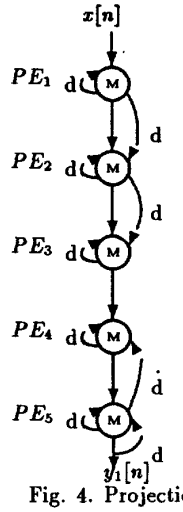


Fig. 4. Projection of DG to VFSA with $\vec{d} = [1,0]^T$

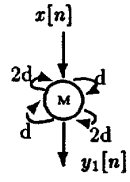


Fig. 5. A PE in the FFSA.

The resulting VFSA has 5 PE's and hence the throughput of the VFSA, $F = 1/5\tau_{PE(VFSA)}$, where $\tau_{PE(VFSA)}$ is the delay through each PE in the VFSA. Since we consider synthesis of cascaded IIR filters, we will restrict the architecture synthesis approach to cases when $5\tau_{PE(VFSA)} < T_{data}$. Thus the data rate is slower than the computation time through a PE in VFSA. This implies that the PE in VFSA is underutilized. A further folding of the PEs results in a single PE as shown in Fig. 5. The four stages of the cascaded filter of Fig. 2 now corresponds to a linear array of four PEs which forms a Fixed Full Size PE-Array (FFSA) as shown in Fig. 6. Thus a PE in FFSA represents a stage of the original cascaded filter. The critical path of FFSA comprises of 4 PEs and corresponding throughput $F = 1/4\tau_{PE(FFSA)}$,where $\tau_{PE(FFSA)}$ is the delay through a PE in FFSA. Further by introducing four delay registers at the input of the system and retiming [5] the PEs in FFSA, we obtain a pipeline system as shown in Fig. 7. The throughput of the resulted pipelined FFSA (PFFSA) is $F = 1/\tau_{PE(PFFSA)}$, where $\tau_{PE(PFFSA)}$ is the delay through a PE in PFFSA. Since a PE in PFFSA performs rhe computation of 5 multipliers and 5 adders, functionally, it can be represented as $F_{PFFSA} = f(i_1, i_2, .., i_5, o)$, where $i_1, i_2, .., i_5$ are the five inputs to the Multiply/Add unit of the PE in PFFSA and o is its output.

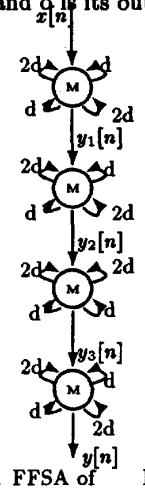


Fig. 6. FFSA of 4 PEs

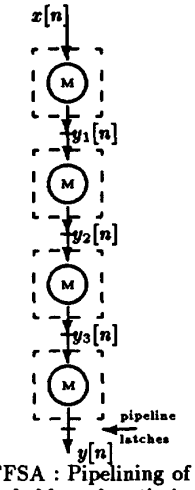


Fig. 7. PFFSA : Pipelining of FFSA with hold up & retiming.

4 VLSI Implementation

The PEs in the PFFSA are regular and pipelined and hence VLSI implementation is feasible. In this section, we present an architectural synthesis methodology of cascaded IIR filters in Table Look Up (TLU) Field Programmable Gate Array (FPGA) technology. The advantage of the FPGA technology is its programmability or reprogrammability in the field [6]. Over a half-dozen of FPGA architectures, based on either Static-RAM (SRAM) or antifuse configuration elements, are now available for designers to choose from. A key aspect of SRAM-based arrays (Eg. Xilinx) is their in-system reprogrammability. The ability to reuse the array definitely benefit prototyping applications because it eliminates waste of improperly programmed one-time programmable parts. Large arrays of RAM-based FPGAs can prototype a gate array or standard-cell-based chip. In particular, the Xilinx (S-RAM based technology) has a basic configurable logic block (CLB). Each CLB contains programmable combinatorial logic and storage registers. The delays through the CLBs are predictable, since the combinatorial logic section of the block is capable of implementing any Boolean function of its input variables defined by the Table Look Up function. Unlike in TLU FPGAs, where the delays are predictable, in antifuse technology, the delays are functions of the depth of the combinational logic. Each CLB in Xilinx FPGAs is a function of four independent input variables and two independent latched (optional) outputs. The function of the CLB can be extended to five variables and two independent outputs. Functionally, a CLB is defined as $F_{CLB} = f(i_1, i_2, .., i_5, o_1, o_2)$, where $i_1, i_2, .., i_5$ are the 5 inputs and $o_1, o_2$are the two latched outputs of the CLB.

In power system applications, particularly in protective relaying where the data has to be filtered, the data rate is around 3 KHz. Since the data rate is suf-

ficiently slow, it turns out that multiple PEs of the VFSA can be folded onto a single PE of the PFFSA and realized in FPGA technology. A PE in the PFFSA comprises a multiply and add unit. This necessiates implementing both multiplier and adder in FPGA technology which is difficult to accomodate in a single XC3090 device, which is a top of the line 3000 series device from Xilinx. Instead, if the coefficients are represented as simple sum and/or differences of powers of 2, then both high speed and low complexity can be achieved at the cost of slight frequency degradation. Therefore, the multiply and add unit in PFFSA uses a two-digit Canonic-Signed-Digit (CSD) [7] code for the representation of the filter coefficients. This reduces the critical path from five multiplication and five additions to ten additions. Thus each filter coefficient h(n) is expressed as a sum or difference of atmost two powers of two, i.e.,

$h(n) = s_1 2^{-i} + s_2 2^{-j}$

Where $s_1, s_2 \in (-1, 0, 1)$ and $i, j \in \{0, 1...9\}$.

Since the coefficients are represented as sums and/or differences, the basic block in a PE of PFFSA is an adder. Thus the delay of the PE is equal to that of a carry-save adder . In Fig. 9, we give the system diagram of an FPGA implementation for a PE in the PFFSA. It consists of I/O Multiplexer, Coefficient Server, and Computation Server.

1. I/O Multiplexer An adder in PE of the PFFSA must perform weighted carry save addition of five filter coefficients in CSD representation. This can be represented functionally as $F_A = f(i_1, i_2, .., i_{10}, o_1, o_2)$, where $i_1, i_2, .., i_{10}$ corresponds to the five coefficients in two-digit CSD form and o_1, o_2 corresponds to latched outputs in CSA form. Since a CLB in Xilinx XC3090 can be suitably used to perform the function of a full adder, the above function F_A has to be realized using this full adder. In order to do this, the inputs to the full adder must be suitably multiplexed and the intermediate results of addition has to be stored. These functions are carried out in the I/O Multiplexer unit.

2. Coefficient Server Filter coefficients which are represented in two-digit CSD code must be shifted and provided as operands for the full adder in a PE of the PFFSA. In order to keep the design general, the shift amounts corresponding to the coefficients are kept programmable. To that extent, the coefficient server serves to shift the coefficient by desired amount before presenting it as an operand for the full adder in a PE of the PFFSA. Pipelining is achieved at a very fine level of granularity by latching each stage of the coefficient shifter at the level of a CLB. The shifter implemented has the properties of a barrel shifter and hence it is possible that shifts by variable amount can be realized in constant time.

3. Computation Server This forms the heart of the PE in the PFFSA. It comprises a basic full adder that operates on operands served by the coefficient server. Intermediate results of addition are either stored in the adder and as well as in the I/O multiplexer.

The PE has been implemented on XC3090 device. The longest combinational path between any two registers observed to be 96 ns. This ensures a sustained operation of 10 MHz.

5 Conclusion

In this paper, we presented a novel method of architecture synthesis of cascaded IIR filter in TLU FPGA technology. The synthesis procedure involves deriving the final architecture of the IIR filter through a series of transformations on the data flow graph (DG) of the filter. These transformations ensure that the hardware utilization is maximized for a given data rate and given technology. Since each PE in the PFFSA is systolic and corresponds to a single cascade stage, it is possible to build nth order IIR filters by interconnecting n PEs in a linear array.

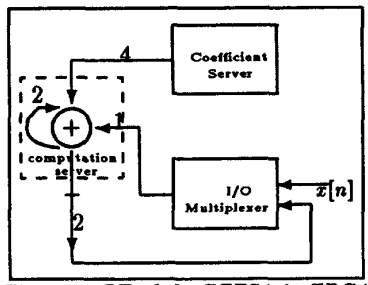


Fig. 9. A PE of the PFFSA in FPGA.

References

[1] K. K. Parhi and D. G. Messershmitt, "Pipelining Using Scattered Look Ahead and Decomposition", *IEEE Trans. on ASSP*, vol. 37, pp. 1099-1117, July '89.

[2] V. Visvanathan et. al., "A New Systolic Architecture for Real-Time VLSI Infinite Impulse Response Filters" *International conference on VLSI Design*, Jan'93.

[3] K. K. Parhi, "Finite Word Effects in Pipelined Recursive Filters", *IEEE Trans. on Signal Processing*, vol. 39, no. 6, June'92.

[4] S. Y. Kung, *VLSI Array Processors*, Prentice Hall, 1988.

[5] Charles E. Leiserson et.al, "Optimizing Synchronous Circuitry by Retiming", *Proceedings of the third Caltech conference on Very large Scale Integration*,pp. 87-116, 1983.

[6] *FPGA Advances Cut Delays, Add Flexibility*, Electronic Design, Oct'92.

[7] Henry Samueli, "The Design of Multiplierless FIR Filters for Compensating D/A Converter Frequency Response Distortion", *IEEE Trans. on Circuits and Systems*, vol. 35, no. 8, Aug '88.

Acknowldgements

The authors acknowledge the helpful discussions and support received from Prof. V Rajaraman, Chairman, Supercomputer Education and Research Center, Indian Institute of Science for this work.