FPGA IMPLEMENTATION OF DIGITAL FILTERS

Chi-Jui Chou, Satish Mohanakrishnan, Joseph B. Evans
Telecommunications & Information Sciences Laboratory
Department of Electrical & Computer Engineering
University of Kansas
Lawrence, KS 66045-2228

ABSTRACT

Digital filtering algorithms are most commonly implemented using general purpose digital signal processing chips for audio applications, or special purpose digital filtering chips and application-specific integrated circuits (ASICs) for higher rates. This paper describes an approach to the implementation of digital filter algorithms based on field programmable gate arrays (FPGAs). The advantages of the FPGA approach to digital filter implementation include higher sampling rates than are available from traditional DSP chips, lower costs than an ASIC for moderate volume applications, and more flexibility than the alternate approaches. Since many current FPGA architectures are in-system programmable, the configuration of the device may be changed to implement different functionality if required. Our examples illustrate that the FPGA approach is both flexible and provides performance comparable or superior to traditional approaches.

1. INTRODUCTION

The most common approaches to the implementation of digital filtering algorithms are general purpose digital signal processing chips for audio applications, or special purpose digital filtering chips and application-specific integrated circuits (ASICs) for higher rates [9, 14]. This paper describes an approach to the implementation of digital filter algorithms on field programmable gate arrays (FPGAs).

Recent advances in FPGA technology have enabled these devices to be applied to a variety of applications traditionally reserved for ASICs. FPGAs are well suited to datapath designs, such as those encountered in digital filtering applications. The density of the new programmable devices is such that a non-trivial number of arithmetic operations such as those encountered in digital filtering may be implemented on a single device. The advantages of the FPGA approach to digital filter implementation include higher sampling rates than are available from traditional DSP chips, lower costs than an ASIC for moderate volume applications, and more flexibility than the alternate approaches. In particular, multiple multiply-accumulate (MAC) units may be implemented on a single FPGA, which provides comparable performance to general-purpose architectures which have a single MAC unit. Further, since many current FPGA architectures are in-system programmable, the configuration of the device may be changed to implement alternate filtering operations, such as lattice filters and gradient-based adaptive filters, or entirely different functionality.

This research is partially supported by the Kansas Technology Enterprise Corporation through the Center for Excellence in Computer-Aided Systems Engineering and by the University of Kansas General Research allocation 3626-20-0038.

2. BACKGROUND

Research on digital filter implementation has concentrated on custom implementation using various VLSI technologies. The architecture of these filters has been largely determined by the target applications of the particular implementations. Several widely used digital signal processors such as the Texas Instruments TMS320, Motorola 56000, and Analog Devices ADSP-2100 families have been designed to efficiently implement filtering operations at audio rates. These devices are extremely flexible, but are limited in performance. High performance designs for filtering at sampling rates above 100 MHz have also been demonstrated using CMOS [3, 4, 6, 8, 9, 14, 17, 19, 20, 21] and BiCMOS [8, 20, 22] technologies, using approaches ranging from full custom to traditional factory-configured gate arrays. These efforts have produced high performance designs for specific application domains.

There are several potential shortcomings of the custom VLSI approach, although it does promise the best performance and efficiency for the specific application for which a particular design is intended. The most obvious problem is the lack of flexibility in the custom approach. Custom devices are often suited only for use in a particular application, and can not be easily reconfigured for other operations even within that same domain. Another problem which the custom VLSI approach often imposes is a lack of adaptability once a device is in use within a system. Typical custom approaches do not allow the function of a device to be modified within the system, for purposes such as correcting faults, for example. Although these problems can be overcome with sufficient forethought, the costs in performance, implementation complexity, and additional design time often preclude flexible solutions.

Lack of flexibility can forestall the cost-effective evaluation of exotic algorithms in a high performance real-time environment. Only high volume applications or extremely critical low volume applications can justify the expense of developing a full custom solution. There are a variety of algorithms which are not within the performance envelope of general purpose processors, and which are not sufficiently commonplace or well-understood to justify implementation in a full custom design. These algorithms cannot be evaluated with the traditional approaches, thus limiting innovation.

Field programmable gate arrays (FPGAs) can be used to alleviate some of the problems with the custom approach. FPGAs are programmable logic devices which bear a significant resemblance to traditional custom gate arrays. While there are a variety of approaches to FPGA implementation, some of the more popular series consist of an array of arbitrarily programmable function

blocks, with configurable routing resources which are used to interconnect these blocks. Many of the most popular FPGAs are in-system programmable, which allows the modification of the operation of the device through simple reprogramming.

The primary limitations of FPGAs are related to the overhead imposed by programmability. In particular, the density of the devices is only now reaching the level necessary to implement complete modules of reasonable complexity. Other difficulties associated with the devices result from the constraints imposed by the architecture, such as limitations on the logic functions which may be implemented in each logic block, and routing delays in the array. Many of these difficulties can be overcome by careful design.

Due to ever-increasing integrated circuit fabrication capabilities, the future of FPGA technology promises both higher densities and higher speeds. Many FPGA families are based on memory technology, so the improvements in those areas should correlate with FPGA evolution. The expanded use of FPGAs in a variety of challenging application domains is thus likely.

FPGAs are well suited for the implementation of fixed-point digital signal processing algorithms. The advantages of DSP on FPGAs are primarily related to the additional flexibility provided by FPGA reconfigurability. Not only can high-performance systems be implemented relatively inexpensively, but the design and test cycle can be completed rapidly due to the elimination of the integrated circuit fabrication delays. The new approach also allows adapting the functions to account for unforeseen requirements. The problems of DSP on FPGAs are related to the density and routing constraints imposed by the FPGA architectures. In particular, the number of logic gates which may be implemented on an single device, and hence the number of arithmetic units, is still limited, and the routing between modules on an array imposes the critical delay limitations.

Because of the constraints imposed by FPGAs, implementation of digital filter algorithms through this medium must initially focus on efficient structures which possess low complexity [2]. Concurrent design of efficient digital filter algorithms and FPGA implementations is necessary to take full advantage of the new capabilities.

In this particular work, Xilinx XC4000-series FPGAs were used to implement various digital filter algorithms and evaluate their performance. A Xilinx XC4000 consists of an array of configurable logic blocks (CLBs), each of which has several inputs (F1-F4, G1-G4) and outputs (X,Y and XQ,YQ). Each CLB can contain both random logic and synchronous elements. In addition to the general-purpose logic functions, each CLB also contains special fast carry logic for addition operations. The XC4000-series contains both local and global routing resources. The local resources allow extremely low delay interconnection of CLBs within the same neighborhood, as well as more extended connection through the use of switching matrices. The global resources provide for the low-delay distribution of signals that are used at widely-spaced points in the array. The speed of a particular application is highly dependent on routing in the Xilinx FPGAs. The XC4000 family includes parts ranging from 8 by 8 CLB arrays to 24 by 24 CLB arrays. All of these devices are in-system programmable. Low power versions of many of these parts are also available.

3. MULTIPLY-ACCUMULATE UNITS

Several authors [1, 11, 12, 13] have identified the multiply-accumulate (MAC) operation as the kernel of various digital signal processing algorithms. A variety of approaches to the implementation of the multiplication and addition portions of the MAC function are possible [7, 10]. This work will focus on the realization of multiplication using an array approach and addition using ripple carry methods, although other methods are equally applicable to the FPGA domain.

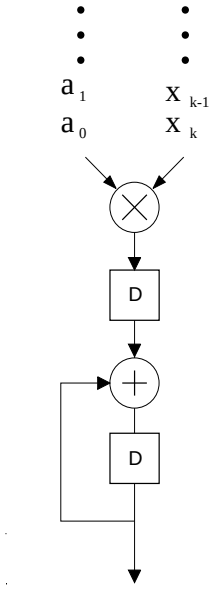

Figure 1. Basic Structure of the MAC Unit

The structure of a MAC unit is illustrated in Figure 1. The MAC unit presented in this section consists of an 8-bit by 8-bit combinatorial array multiplier and a 16-bit accumulator. These word sizes were chosen to balance the size of the implementation, which is limited by the FPGA density, against the numerical precision. Larger word sizes are possible if the number of MAC units per chips is reduced. The increase in density of FPGAs in the future will certainly expand the design space available to the designer, and make such constraints less severe.

3.1. Implementation of Multiplier

The combinatorial multiplier uses one CLB per partial product bit. A 2-input AND gate generates each partial product, but additional circuitry is required to add together all partial products of equal weight. The total number of CLBs used for the multiplier in this case is 64 and the basic cell structure is illustrated in Figure 2.

Each cell is configured as a full adder (except for the type A cell). This full adder accepts a sum and a carry from a previous operation of equal weight, as shown in Figure 2, and the logical AND of the inputs x_i and a_i. The sum and carry generated by the adder are then sent to the CLBs of proper weight as shown in Figure 3.

The multiplier has been configured to perform multiplication of signed numbers in two's complement notation. The small circles in the figure indicate negative inputs or outputs; such bits have to be subtracted rather than being added. The cells in the leftmost column of the array only AND their two inputs and generate the product. If one of the two inputs has a negative weight, then the output will have a negative weight. The conventional 1-bit full adder assumes positive weights on all of its 3 inputs and 2 outputs. Such an adder can be generalized to four types of adder cells by attaching positive and negative weights to the input/output pins as discussed in [7]. Figure 4 lists the logic symbols for the four types of generalized full adders.

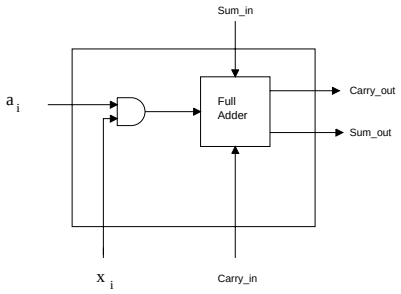

Figure 2. Cell Structure of Combinatorial Array Multiplier

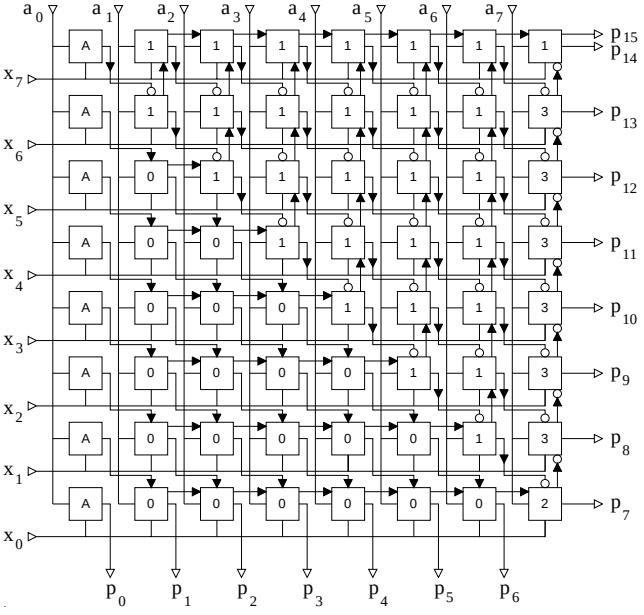

Figure 3. Combinatorial Array Multiplier Block Diagram

Type	Logic Symbol	Operation
Type A Cell	X, A, S, Y	(-X) * Y = (-S)
Type 0 Full Adder	Z, X, 0, C, S, Y	X + Y + Z = C S
Type 1 Full Adder	Z, X, 1, C, S, Y	X + Y + - Z = C (-S)
Type 2 Full Adder	Z, X, 2, C, S, Y	- X - Y + Z = (-C) S
Type 3 Full Adder	Z, X, 3, C, S, Y	- X - Y + - Z = (-C) (-S)

Figure 4. Structure of Generalized Full Adder Cells

The Boolean equations governing the Type 0 and 3 full adders are

$$\begin{aligned} S &= \overline{X}\,\overline{Y}Z + \overline{X}Y\overline{Z} + X\overline{Y}\,\overline{Z} + XYZ, \\ C &= XY + YZ + XZ, \end{aligned} \tag{1}$$

and those for the Type 1 and 2 adders are

$$\begin{aligned} S &= \overline{X}\,\overline{Y}Z + \overline{X}Y\overline{Z} + X\overline{Y}\,\overline{Z} + XYZ, \\ C &= XY + X\overline{Z} + Y\overline{Z}. \end{aligned} \tag{2}$$

Type 0 and Type 3 full adders are characterized by the same pair of logic equations, identical to that of the conventional 1-bit full adder (Type 0). This is because a Type 3 full adder can be obtained from a Type 0 full adder by negating all of the input and output values and vice versa. A similar relationship can be established between Type 1 and Type 2 full adders.

For Type 0, 1, 2, and 3 full adders, the two independent 4-bit functions were used to generate the sum and carry outputs. We can easily include the AND gate in the CLB just by replacing, for example, X with (x_i AND a_i) when configuring the CLB. The horizontal inputs (x_i, a_i) can use the horizontal longlines which are associated with each row for distribution of the signal with a very short routing delay. Other interconnections can be made using the single-length or double-length lines via Programmable Interconnection Points (PIP) or switching matrices.

3.2. Adder Implementation

In the XC4000 series, each CLB includes high-speed carry logic that can be activated by configuration. The two 4-input function generators may be configured as a 2-bit adder with built-in hidden carry that can be expanded to any length. The 16-bit adder in our MAC unit, which uses the dedicated carry logic, requires nine

CLBs. The middle 14 bits use 7 CLBs, one CLB is used for the MSB, and one is used for the LSB of the adder. For each CLB in the middle section, the F function is used for lower-order bit and the G function is used for higher-order bit. Obviously, we need to use the G function for the LSB bit and F function for the MSB bit.

In the case of the LSB CLB, two values must be input on the G1 and G4 pins. The carry signal enters on the F1 pin, propagates through the G carry logic, and exits on the COUT pin. The F function of this CLB is not used and can be used for other purposes. For the middle CLBs, the logic is configured to perform a 2-bit addition of A+B in both the F and G functions, with the lower-order A and B inputs on the F1 and F2 pins, and the higher-order A and B inputs on the G1 and G4 pins. The carry signal enters on the CIN pin, propagates through the F and G carry logic, and exits on the COUT pin. For the MSB CLB, the two values must be input on F1 and F2 pins. The carry signal enters on the CIN pin, propagates through the F carry logic, and exits on the COUT pin. The G function generator of this CLB is used to access the carry out signal or calculate a two's complement overflow.

The limitation of using this built-in carry logic is that the carry out (COUT) pin of a CLB can only be connected to the carry in (CIN) pin of the CLBs above or below. Thus the adder using fast carry logic can only be configured vertically in the array.

The dedicated carry circuitry greatly increases the efficiency and performance of adders. Conventional methods for improving performance such as carry generate/propagate are not useful even at 16-bit level, and are of marginal benefit at longer wordlengths. In our case, the 16-bit adder has a combinatorial delay of only 20.5 ns.

3.3. MAC Implementation

We use the most significant 8 output bits of the multiplier as the input to the low order bits of the adder. The 8-bit input of the adder is sign-extended and added with previous outputs using two's complement addition.

The basic structure of the MAC unit can use pipeline registers between the multiplier and accumulator to increase the throughput. The flip-flops in the CLBs are used as pipeline registers and hence no additional CLBs are needed.

The layout of a single MAC unit on an XC4000-series part is shown in Figure 5.

The performance of the MAC unit with an 8-bit by 8-bit multiply and 16 bit accumulator is determined by the speed of the multiplier. The worst case multiplier delay reported is approaching 100 ns. The MAC unit can thus support a clock speed better than 10 MHz. With the use of the horizontal longlines to distribute the critical path signals, the speed can be further improved, although this may restrict the use of the MAC unit in various system configurations. The implementation of a MAC unit on an XC4000-series part requires 73 CLBs.

4. FIR FILTERS

4.1. Filter Structures

The transfer function of an N tap FIR filter is given by

$$H(z) = a_0 + a_1 z^{-1} + \ldots + a_{N-1} z^{-(N-1)}. \quad (3)$$

This structure can be realized in many ways, such as the canonical form, pipelined form, and inverted form as depicted in Figure 6.

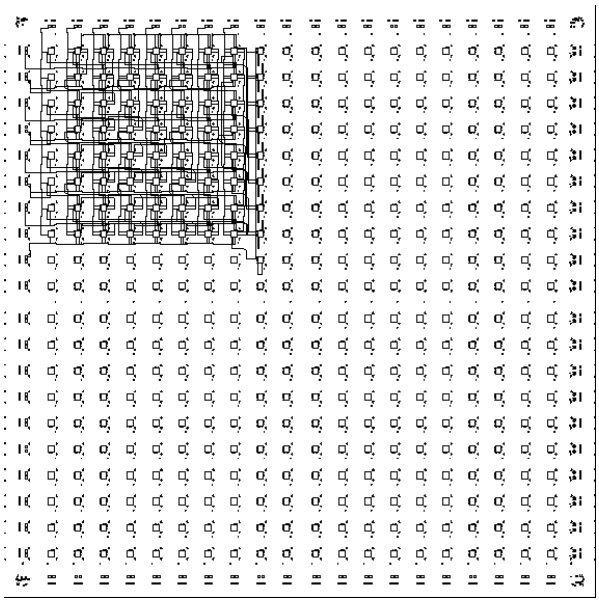

Figure 5. Layout of Single MAC Unit on XC4010 FPGA

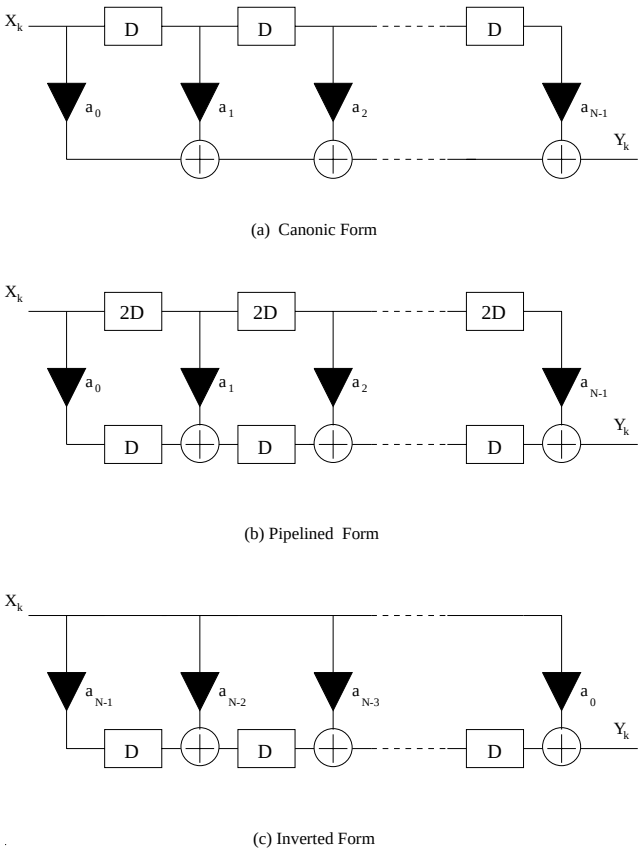

Figure 6. Various Realizations of FIR Filters

4.2. High Performance Filters on FPGAs

The inverted form shown in Figure 6(c) is well-suited for achieving a high sampling rate even for higher order filters. This is possible because the throughput does not depend strongly on the number of taps due to extensive pipelining. The fact that the multipliers occupy a large area, however, might render the implementation of higher order filters impractical.

It has been shown in [2] that a high performance FIR filter with substantial number of taps can be implemented on FPGAs by approximating the filter coefficients to a sum or difference of two power-of-two terms. Implementation of digital filters may be simplified by using only a limited number of power-of-two terms so that only a small number of shift and add operations is required. A variety of techniques have been proposed [15, 16] to minimize the deterioration of the frequency response due to these constraints. Such coefficient optimization techniques yield performance sufficient for most practical applications.

4.3. Moderate Performance Filters on FPGAs

When the size of the chip is a constraint, the arithmetic resources need to be shared at the expense of speed. The structure shown in Figure 7 is suitable for sharing of arithmetic resources. This is a multiply/accumulate (MAC) unit with four multipliers and an adder tree. The inputs and the corresponding filter coefficients are fed to the MAC unit as shown in Figure 7. With the insertion of pipeline registers, the clock speed is increased. The delay in the multiplier is greater than that in the adder and hence the clock frequency is dependent on the delay in the multiplier. As there are four multipliers in this MAC unit, summation of four terms is computed every clock cycle. Hence a four tap filter can be made to operate at a sampling rate equal to the clock rate, and an eight tap filter to operate at a sampling rate half that of the clock rate.

In general, if there are M multipliers in a chip and if the delay in the multiplier is T sec, then an N tap filter can operate at a maximum sampling frequency f_s given by

$$f_s = \frac{1}{T\lceil N/M \rceil}. \tag{4}$$

An implementation based on the multiple-input MAC unit, as shown in Figure 7, was used to evaluate this moderate performance approach to the realization of a filter with an arbitrary number of taps. The placement of the MAC unit on a Xilinx XC4010 is shown in Figure 8. The four multipliers are arranged in the four corners of the 20 by 20 array of CLBs to reduce the delay from the input pins to the multipliers. Inputs to the multipliers are fed in at right angles, as explained previously, and the arrays are oriented in such a way that the routing delays from pads are minimized. For ease of understanding, the most significant bit (M), intermediate bit (I), and least significant bit (L) of the output of each multiplier are marked in the Figure. The four adders were arranged vertically to exploit the dedicated carry logic supported by the XC4000 series. The size of the chip limited the number of multipliers to four. Four columns of CLBs were left for the adders. The three intermediate adders were provided with the required number of bits, that is, 16 bits, 16 bits, and 17 bits, respectively. The adders were arranged in two columns as shown in Figure 8. This leaves two full columns capable of supporting more than 70 bits for the final adder and provides sufficient intermediate word width protection for most applications.

The routing between the arithmetic elements is not critical because the delay in the multiplier is 100 ns and that in the adder is 22.5 ns (for a 16 bit adder) which allows routing delays to be as large as 75 ns without affecting the clock speed. A clock of 10 MHz is used for the pipeline registers through the global clock buffer.

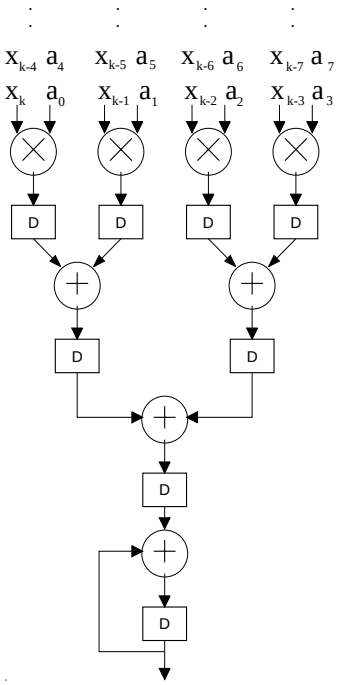

Figure 7. FIR Filter Realization Using MAC with Four Multipliers

Additional support chips are required to synchronize the inputs to the multipliers. There are a few CLBs left in the array after the implementation of multipliers and adders, which can be used to implement the logic for interfacing to these other devices.

As the MAC unit has 4 multipliers and the delay in the multiplier is 100 ns, an N tap filter with these word sizes can be operated with sampling rates of 40/N MHz, where N is a multiple of 4. For example, a 32 tap filter can support a sampling rate of 1.25 MHz.

5. IIR FILTERS

Our implementations of multiply-accumulate units indicate that the larger FPGAs can easily support a general purpose second order IIR filter with reasonable word sizes at moderate to high sampling rates. Designs which exploit the FPGAs reconfigurability can be used to attain even higher densities and speeds.

5.1. IIR Filter Structure

The transfer function of an Nth order IIR filter is given by

$$H(z) = \frac{b_0 + b_1 z^{-1} + \ldots + b_{N-1} z^{-(N-1)}}{(1 - a_1 z^{-1} - a_2 z^{-2} - \ldots - a_{N-1} z^{-(N-1)}}. \tag{5}$$

Some of the realizations possible are direct form I and direct form II, as discussed in [18]. For reducing the delay in the paths between registers, however, the realization shown in Figure 9 is used in this paper. This realization, like direct form II, is a cascade of an autoregressive (AR) filter and a moving average (MA) filter, but with a pipeline register in between. The delay elements are also rearranged in such a way that there is only one path with a multiplier and two adders. The others have only one multiplier and one or no adder. This realization allows easier placement of

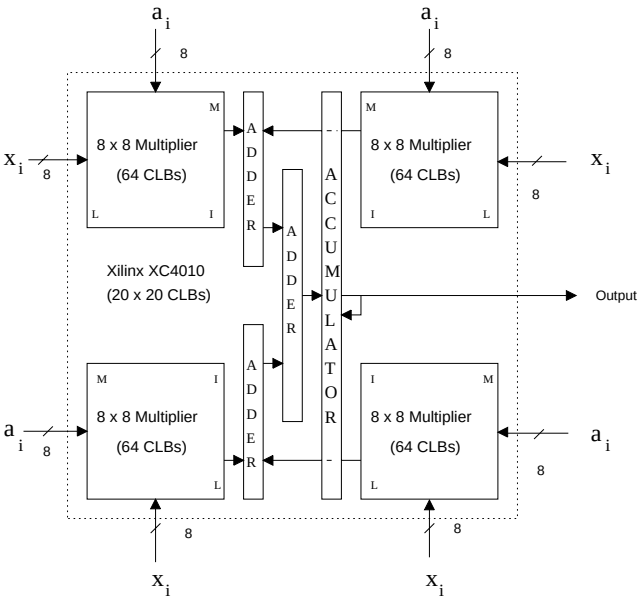

Figure 8. Placement of MAC with Four Multipliers on XC4000-Series FPGA

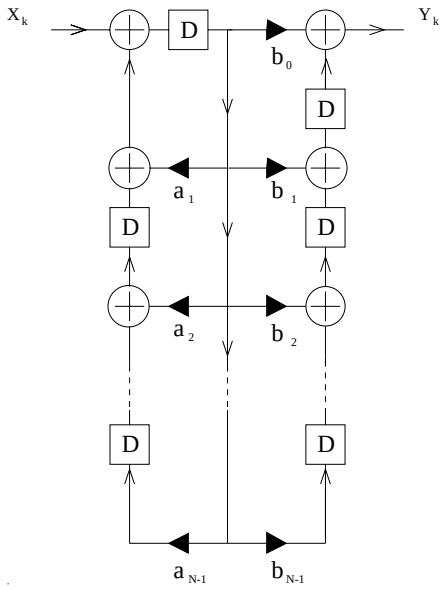

Figure 9. Modified Canonical Form Realization of IIR Filters

the multipliers and adders in the array of CLBs to achieve minimal routing delays.

5.2. General IIR Filter Implementation on FPGAs

Second order IIR filters with general purpose multipliers, which can take coefficients as inputs from outside the chip, can be used as building blocks for cascade or parallel realizations of higher order IIR filters. We will show that an XC4013 can support a general purpose second order IIR filter at moderately high sampling rates.

The first term in the denominator of the transfer function may be scaled according to the number of bits in the coefficients for fixed point implementation. This implies that a scaling module is needed before the pipeline register between the AR and MA sections shown in Figure 9. This divider can be implemented with a shifter without considerably increasing the area and delay by constraining this coefficient to be equal to the nearest allowable power-of-two number during the discrete space optimization of the quantized coefficients.

In this implementation, the multipliers have 8 bit inputs and the adders have 16 bit inputs. The 16 bit output of the multipliers is fed to the adders and the most significant 8 bits of the output of the adders are fed back to the multipliers. The two adders in the autoregressive part of the filter referred to in Figure 9 are implemented as a cascade of two dedicated carry logic adders. This adder has a total combinatorial delay of 23 ns (2.5 + 20.5), which is close to that of a single dedicated carry logic adder. Thus the dominant delay is not necessarily in the logic path with two adders as it may seem, but could very well be in other paths with one multiplier and adder, unless placement is carefully considered.

The floorplan is illustrated in Figure 10. The placement was done carefully to reduce the routing delays. Three 8 by 8 multiplier units are arranged at the top of the array of 24 by 24 CLBs and two multipliers below them. The placement of the multipliers corresponding to a_2 and b_2 is not critical because there are no adders in their logic paths. The adders are arranged vertically to make use of the dedicated carry logic. The scaling block is implemented with a shifter whose shift is controlled by a 3 bit input (S). This shifter is implemented in two stages, the first stage providing shifts of any one of 0, 1, 2, or 4 shifts and the second stage providing a shift of 0 or 3 stages. The first stage requires 8 CLBs and the second stage requires 4 CLBs.

The worst case delay in any logic path was found to be less than 145 ns. Thus using this approach, the larger parts in this series can easily support a general purpose second order IIR filter with the indicated word sizes at sampling rates approaching 7 MHz.

5.3. Dedicated IIR Filter Implementation on FPGAs

Dedicated IIR sections have "hardwired" coefficients that are programmed when the array is configured. In binary multiplication, each partial product is a shifted version of the multiplicand if the corresponding multiplier bit is a one, and a zero if the corresponding bit is a zero. This zero term need not be computed and a row of adders in the multiplier array can be eliminated, so that higher densities can be achieved. We will show that a single XC4013 can support two dedicated second order IIR filters using this approach.

In order to evaluate the practicality of implementing several dedicated second order sections on a single FPGA, a typical low pass IIR filter was designed as a cascade of two second order sections, and was implemented on a Xilinx XC4013. As the coefficients have only a small number of non-trivial bits, the multiplications can be realized using shift and add technique. Thus for the filter,

$$H(z) = \frac{6 - 10z^{-1} + 5z^{-2}}{64 + 113z^{-1} - 50z^{-2}} \cdot \frac{3 + z^{-1}}{64 + 105z^{-1} - 45z^{-2}}, \quad (6)$$

the first second order section needs 12 columns and the second one needs 11 columns. This was implemented in a single XC4013 chip. The placement of the key modules is given in Figure 11. Addition of N inputs is performed by $N - 1$ stages of dedicated carry logic (DC) adder/subtractors. The dotted lines represent horizontal longlines and the shaded triangles, the shifts. Blocks which have registers are shaded. The shift for the scaling block and the shifts needed for the multiplication tend to cancel each other, thus shifting in the scaling block is absorbed into the shift of the multiplier units. Negative coefficients were handled by using the dedicated carry subtractors; these columns are represented in the layout by the small circles at the inputs to the adders. The

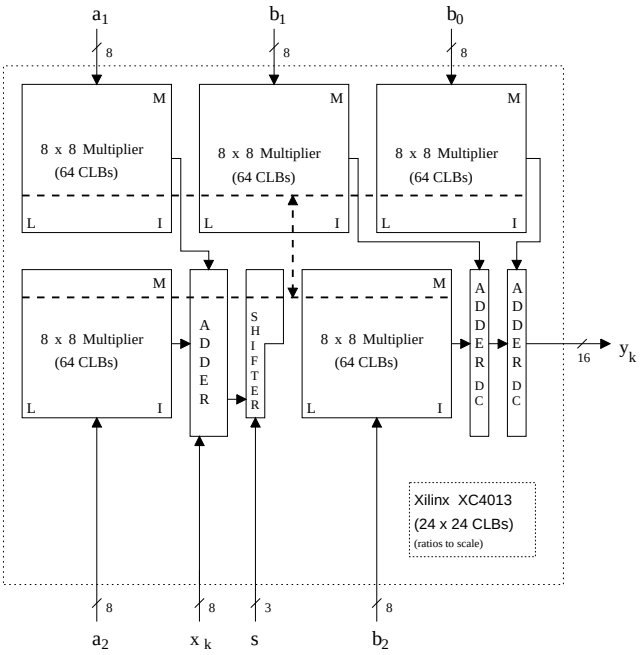

Figure 10. Placement of Modified Canonical Form IIR Realization on XC4013

numerous shifts encountered in the shift and add approach will very easily exhaust the routing resources. We have provided an empty column between every stage, which frees up additional routing resources. As shown in Figure 10, one second order section is placed on top of the other, for reasons discussed above, though the 24 columns available in the XC4013 might seem to satisfy the requirements for the two sections, which total only 23 columns. The adders in this implementation use 22 bits, and the most significant 14 bits of the output of the dedicated carry adders are fed to the shift and add blocks.

The sampling rate achieved with this configuration was more than 10 MHz.

A number of other example fixed-point filters were designed to evaluate the utility of this approach; in all the cases studied, it was practical to implement two IIR sections on a single chip.

6. PIPELINED MAC UNITS

It has been mentioned that the delay in the multiplier poses a major limitation on the maximum sampling rate that can be attained. Array multipliers can be configured to allow a pipelined mode of operation, where the execution of separate multiplications overlaps. If this mode of operation is applied, the long delay associated with the carry propagating addition performed in the last row of the array multiplier can be minimized, since it determines the throughput of the pipeline. This approach has been shown to yield extremely high speed custom implementations [5]. With this more aggressive pipelining, a MAC unit which operates at rates approaching 100 MHz can be implemented on the XC4000-series FPGAs, thus providing a building block for high sampling rate filters. The pipelined MAC units can be applied to high performance FIR and IIR filter structures, as well as other signal processing algorithms which can tolerate the pipeline delay.

6.1. Structure of the Pipelined MAC Unit

The structure of the pipelined MAC unit is shown in Figure 12. The basic cells shown here are identical to that in the unpipelined

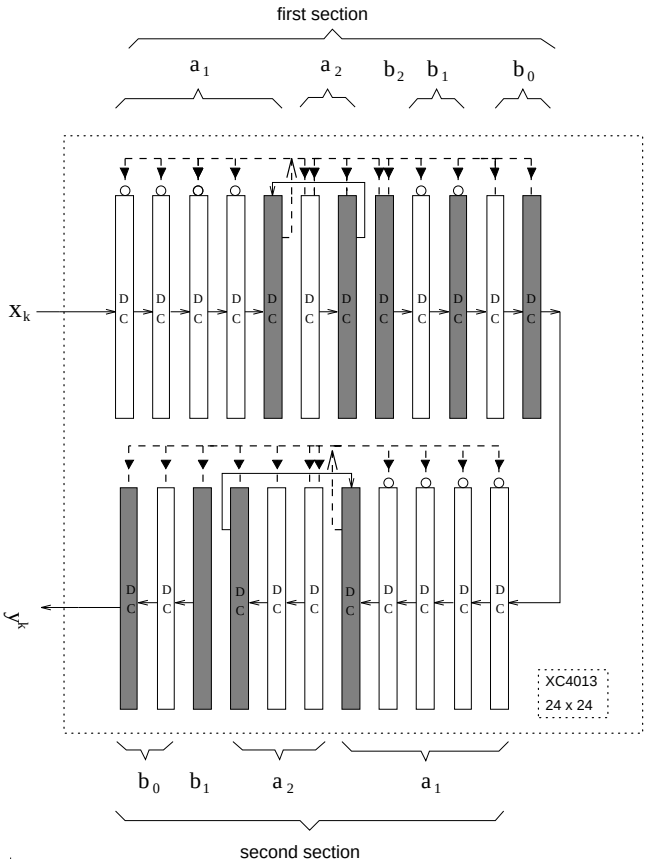

Figure 11. Placement of Two Dedicated Second Order IIR Filters on XC4013

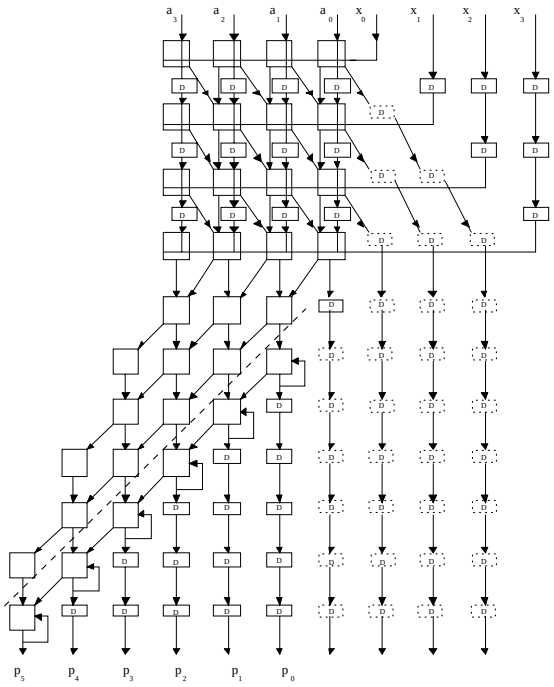

Figure 12. Structure of the Pipelined Multiply/Accumulate Unit

MAC unit except that these cells include pipeline registers. Registers are needed to propagate the multiplier and multiplicand bits to their destination and also to propagate the product bits that have been completed, which is done in parallel with the generation of new product bits.

For an N by N bit multiplier, the carry propagate adder is replaced with N rows of half adders with pipeline registers between the rows. This allows a carry propagation of only one position between any two consecutive rows. The clock speed depends only on the delay in the cells of the multiplier.

If multiple tap filters are to be realized, the adder needs to accumulate the result by getting feedback from the past output. By introducing a set of full adders immediately below the diagonal of the array and feeding back the outputs of the full adders to their inputs through a single register, a bit-level pipelined multiply-accumulate unit is formed. Figure 12 shows a 4 by 4 multiplier and 6 bit accumulator with the 6 most significant bits of the multiplier output being fed back for accumulation; the dashed line denotes the diagonal of the carry-save section. As the 4 least significant bits are discarded, the delay elements shown in the dotted squares need not be implemented. After the required number of multiplications and accumulations are made, the output should be clocked out and the accumulator reset.

The structure of FIR filters facilitates their implementation using the pipelined MAC unit. The filter coefficients and the corresponding delayed inputs are fed to the multiplier in synchronized data streams, with their arrivals corresponding to the basic clock rate. An N tap filter requires $N + 1$ clock cycles to complete the computation of one output.

6.2. FPGA Implementation

Unlike the unpipelined MAC unit, the routing delay is very critical in the pipelined MAC. This is because it takes 3.0 ns for the output of the pipeline register to stabilize after it gets the clock, the output is then routed, and finally there is a 4.5 ns delay in the next CLB.

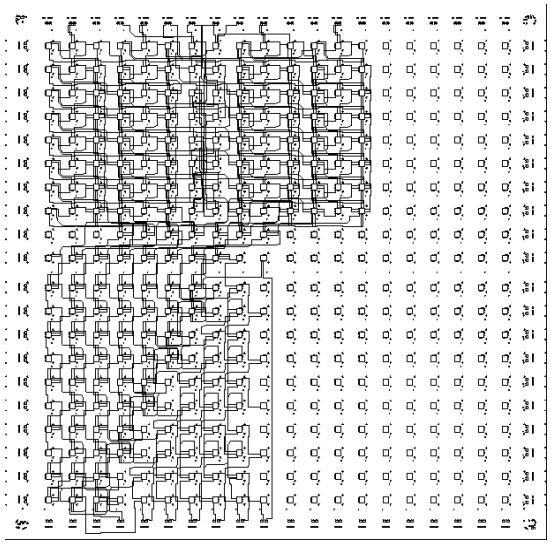

Figure 13. Layout of the Pipelined MAC Unit on XC4010

Thus there is a minimum delay of 7.5 ns and unless proper care is exercised, the routing delays may dominate. There is a delay of 3.0 ns from the pad to Direct In input which will add to the routing delay when the coefficients are distributed to the cells. Some CLBs were used as registers between the input pads and the cells; thus this delay was prevented from reducing the clock speed.

A MAC unit with an 8 by 8 multiplier and a 12 bit accumulator has been implemented on an XC4010. The pipeline registers in the cells have been incorporated inside the CLBs which are also used for the adders. The distribution of multiplier and multiplicand bits is a major source of delay. The registers used to store multiplier and multiplicand bits were interspersed between the CLBs in each row of the multiplier array to reduce this delay.

The final output is computed in the diagonal full adders, and hence it suffices to reset the registers in these adder cells at the completion of an accumulation cycle. Due to the bit-level pipelining, these registers should be reset one by one starting from the top of the carry save adder stage. This could have been easily implemented by asserting the direct reset input to the registers in the cells one by one through a set of skewing registers, but the 8 ns delay imposed by the direct reset is highly undesirable. This problem was solved by making use of the multiplexing capability of the XC4000 series, whereby the outputs of these diagonal cells are forced to zero when a clear signal is asserted.

By careful routing, the maximum routing delay was kept within 4.6 ns which made the worst case delay in a logic path equal to 12.1 ns. Clock rates of more than 80 MHz can thus be attained.

The layout of the pipelined MAC on an XC4010 FPGA is shown in Figure 13, which suggests that two such MAC units could be accommodated in an XC4013.

7. CONCLUSION

This paper has described an approach to the implementation of digital filter algorithms based on field programmable gate arrays (FPGAs). General purpose DSP implementations often lack the performance necessary for moderate sampling rates, and ASIC approaches are limited in flexibility and may not be cost effective

for many applications. Our examples of FIR and IIR filter implementations illustrate that the FPGA approach is both flexible and provides performance comparable or superior to traditional approaches. Because of the programmability of this technology, the examples in this paper can be extended to provide a variety of other high performance FIR and IIR filter realizations.

REFERENCES

[1] P. R. Cappello, editor. *VLSI Signal Processing*. IEEE Press, 1984.

[2] J. B. Evans. An efficient FIR filter architecture. In *IEEE Int. Symp. Circuits and Syst.*, pages 627–630, May 1993.

[3] J. B. Evans, Y. C. Lim, and B. Liu. A high speed programmable digital FIR filter. In *IEEE Int. Conf. Acoust., Speech, Signal Processing*, Apr 1990.

[4] R. Hartley, P. Corbett, P. Jacob, and S. Karr. A high speed FIR filter designed by compiler. In *IEEE Cust. IC Conf.*, pages 20.2.1–20.2.4, May 1989.

[5] M. Hatamian and G. Cash. A 70-mhz 8-bit x 8-bit parallel pipelined multiplier in 2.5-μm CMOS. *IEEE J. Solid State Circuits*, SC-21:505–513, Aug 1986.

[6] M. Hatamian and S. Rao. A 100 MHz 40-tap programmable FIR filter chip. In *IEEE Int. Symp. Circuits and Syst.*, pages 3053–3056, May 1990.

[7] K. Hwang. *Computer Arithmetic: Principles, Architecture, and Design*. John Wiley & Sons, Inc., 1979.

[8] R. Jain, P. Yang, and T. Yoshino. Firgen: A computer-aided design system for high performance FIR filter integrated circuits. *IEEE Trans. Signal Processing*, 39(7):1655–1668, Jul 1991.

[9] K.-Y. Khoo, A. Kwentus, and A. N. Willson, Jr. An efficient 175 MHz programmable FIR digital filter. In *IEEE Int. Symp. Circuits and Syst.*, pages 72–75, May 1993.

[10] I. Koren. *Computer Arithmetic Algorithms*. Prentice-Hall, 1993.

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

[12] S. Y. Kung, R. E. Owen, and J. G. Nash, editors. *VLSI Signal Processing II*. IEEE Press, 1986.

[13] S. Y. Kung, H. J. Whitehouse, and T. Kailath, editors. *VLSI and Modern Signal Processing*. Prentice-Hall, Inc., 1985.

[14] J. Laskowski and H. Samueli. A 150-Mhz 43-tap half-band FIR digital filter in 1.2-μm CMOS generated by compiler. In *IEEE Cust. IC Conf.*, pages 11.4.1–11.4.4, May 1992.

[15] D. Li, J. Song, and Y. C. Lim. A polynomial-time algorithm for designing digital filters with power-of-two coefficients. In *IEEE Int. Symp. Circuits and Syst.*, pages 84–87, May 1993.

[16] Y. C. Lim and B. Liu. Design of cascade form FIR filters with discrete valued coefficients. *IEEE Trans. Acoust., Speech, Signal Processing*, ASSP-36:1735–1739, Nov 1988.

[17] H. Moscovitz, W. Bullman, J. Carelli, et al. Automatic full-custom design of high performance DSP chips. In *Proc. IEEE Globecom*, 1988.

[18] A. Oppenheim and R. Schafer. *Digital Signal Processing*. Prentice-Hall, Inc., 1975.

[19] P. Ruetz. The architectures and design of a 20-MHz real-time DSP chip set. *IEEE J. Solid State Circuits*, 24(2):338–348, Apr 1989.

[20] P. Yang, T. Yoshino, R. Jain, and W. Gass. A functional silicon compiler for high speed FIR digital filters. In *IEEE Int. Conf. Acoust., Speech, Signal Processing*, pages 1329–1332, Apr 1990.

[21] F. Yassa, , J. Jasica, et al. A silicon compiler for digital signal processing: Methodology, implementation, and applications. *Proc. IEEE*, 75(9):1272–1282, Sep 1987.

[22] T. Yoshino, , R. Jain, et al. A 100-MHz 64-tap FIR digital filter in 0.8 μm BiCMOS gate array. *IEEE J. Solid State Circuits*, 25(6):1494–1501, Dec 1990.