

High Speed Digital Filtering on SRAM-Based FPGAs

A. Giri V. Visvanathan S.K. Nandy S.K. Ghoshal

Indian Institute of Science
Bangalore 560 012
INDIA

Abstract

Field Programmable Gate Arrays represent a very promising technology that attempts to provide the benefits of custom VLSI with low turn-around time. However, the applicability of this technology for serious DSP applications, that are compute-intensive and typically demand high throughput, has not yet been fully explored. As a case study, real time digital FIR filters of both the constant and programmable coefficient types are considered. Two filters, one of each type, have been successfully implemented in SRAM-based FPGA parts. In the process of design, several generic issues surfaced: the most important among them being efficient utilisation of routing resources. As ever-improving process technology enables higher and higher speeds of operation, density seems to be the more important deciding factor for the feasibility of such applications on FPGAs.

1 Introduction

Programmability as a convenience to the hardware designer has evolved with time. It has given rise to several levels of compromise between fast implementation of logic and its performance. The ultimate in flexibility in terms of field programmability and/or reprogrammability is offered by Field Programmable Gate Arrays (FPGAs). It combines the logic integration benefits of custom VLSI with the design, production and time to market advantages of standard products. FPGAs can be designed, verified and if necessary redesigned in dramatically less time than masked ASICs. However, they have yet to achieve the performance and densities of standard gate arrays.

The development of suitable architectures for real-time VLSI implementations of compute-intensive signal processing applications is a topic of considerable interest. However, to date little work has been reported on the suitability of FPGAs for this application [1]. It is therefore worthwhile to explore the feasibility of arithmetic type of applications on FPGAs. Since digital filters cover a wide area in this field, these were chosen for case studies. Both constant coefficient

and programmable coefficient type FIR filters were implemented successfully. The design issues that came up are reported here.

The FPGA part used was XC3090PG175 –the largest available part in the 2nd generation of FPGAs from XILINX [2]. The major portion of the total chip area consists of an array of the key functional elements called Configurable Logic Blocks (CLBs). A CLB can implement two independent functions of four variables or a single function of five variables. Both the outputs can be registered. Configurable Input/Output Blocks (IOBs) are arranged along the periphery. Spread all over the chip are the routing resources of basically two types. Local interconnects are dedicated for communication among neighboring CLBs and feature least delay. Global interconnects span the entire area of the chip, but they are considerably slower.

The function generator block is basically a multiplexer fed by a look-up table –the size of which determines the maximum number of variables. Any function of the input variables can be implemented by a CLB. So the depth of logic gets redefined in terms of CLBs, rather than logic gates. Logic partitioning should take care of this fact.

For arithmetic applications, the basic building block is a full adder which operates on three inputs and has two outputs. So the granularity offered by this family of FPGAs matches the application. Also, for high throughput applications bit-level pipelining is often necessary. The presence of two storage elements in each CLB offers the possibility of bit-level pipelining.

One of the key design issues concerns multipliers, which is an essential building block of digital filters and often becomes a deciding factor in terms of the density. However, there are techniques to avoid the use of multipliers for constant coefficient filters. Using Canonical Signed-Digit (CSD) representation [3], multiplications are realised with minimal hardware through the use of a few shift-and-add's. This made designing of an entire 11-tap FIR filter possible on a single FPGA part. In Section 2 we describe the design issues that were handled in this design.

Since actual multipliers are essential for programmable coefficient filters, maximal reuse of hardware was attempted by condensing as much of an entire filter as possible in a single FPGA part. This makes use of a novel architectural concept called

This work was supported in part by the Dept. of Electronics (DOE), Govt. of India under the "LSI/VLSI Design Centre" project and by UNDP and DOE under the KBCS project.

A. Giri is now with Texas Instruments (India) Pvt. Ltd., Bangalore 560 052, India.

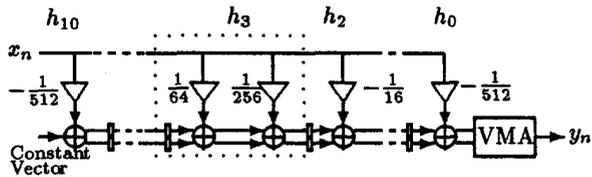


Figure 1: The $z/\sin(z)$ FIR filter

pipelined clustering [4]. In Section 3 we describe the implementation of such an architecture. In our discussions of both the designs, we highlight the design considerations that have an impact on the implementation of high-speed arithmetic in general. We conclude the paper in Section 4 by providing some insights into the suitability of SRAM-based table look-up FPGAs for compute-intensive applications.

2 Constant coefficient FIR filter

The constant coefficient FIR filter chosen for the case study corrects the $\sin(z)/z$ distortion introduced by D/A converters incorporating zero order hold. This filter with a transfer function that approximates $z/\sin(z)$, is used just before the D/A converter to pre-distort the digital samples. It is a linear phase filter having 11 taps. So there are 6 different coefficients. The fact that constant coefficients can be represented in CSD form reduces the hardware requirement by a great extent. The use of approximated values of the coefficients does not distort the transfer function much [3]. By representing the 11 coefficients $h_0 - h_{10}$ in CSD form, all multiplications are replaced by at most 2 additions/subtractions of shifted data words (see Fig 1).

Since each CLB of the FPGA can implement two independent functions of up to four input variables, it is appropriate in this application to implement a full adder in one CLB. Further, since the two outputs can be registered, the carry-save transpose form of an FIR filter is the natural choice for implementation.

Linear phase filters feature symmetric coefficients. This can save the amount of hardware by a factor of 2 when actual multipliers have to be used. But in this case, where multiplications are replaced by a small number of shift-and-add operations, the saving is not much. On the other hand, the design becomes irregular and consequently demands lot of routing. Since the penalty for irregular designs is severe in terms of speed, this does not provide an attractive alternative.

In this design, the input data word is broadcast vertically down. Each row of CLBs is used as a 3→2 compressor (see Fig 2). The multiplication by the CSD digits were implemented by shifting the input data word by an appropriate number of places. Each row of CLBs merges this shifted data word with carry and sum word coming from the previous row and produces output in carry-save form. Following the CSD representation given in [3] there are a total of 16 rows to implement multiplication by 11 coefficients.

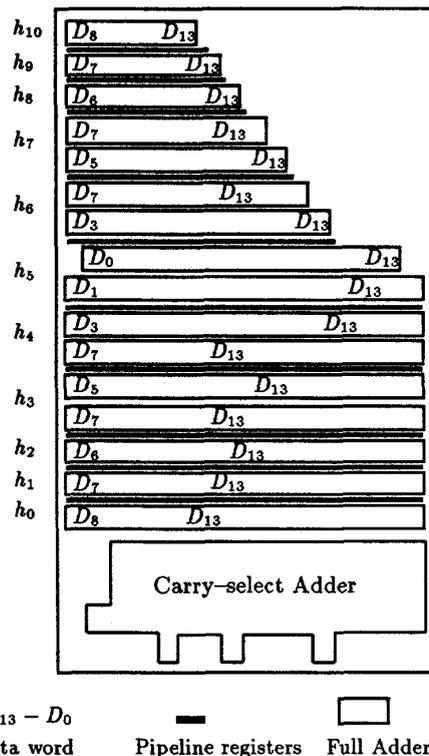


Figure 2: The floorplan of $z/\sin(z)$ FIR filter

2.1 Routing Resources and Pipelining

The architecture of the specified FPGA supports efficient data flow from left to right rather than the reverse. By 'efficient' it is meant that this type of data flow uses less routing resource and costs less delay. Consequently, the implemented array is the reflected version of the normal one, so that the LSB is in the left most position. The respective shifted words were placed one below the other in such a way that the digits of the same modified weight were aligned in a column. This was to ensure that most of the data flow was among the adjacent CLBs. Thus, use of local interconnects was maximised. The arrangement is illustrated in Fig 2.

The finest grain of pipelining that is possible is at the CLB level. Here since CLBs are used as full adders, this offers the potential of bit-level pipelining. However, the routing delay associated with the broadcast of the input data word bits was the limiting factor in terms of speed. Therefore, pipelining was maintained at the level of two CLBs as dictated by the filter architecture (see Fig 1) and bit-level pipelining was not attempted. In 2's complement arithmetic, the shift and add requires a wide broadcast of the MSB of the data word. This only worsens the broadcast problem. This is taken care of by the use of a constant vector as suggested in [3] (see Fig 1).

Chip used	XC 3090PG175-70
Input word length	14 bits
Output word length	14 bits
Internal word length	15 bits
Maximum latch to latch delay	55 ns
Clock frequency	18 MHz
Latency	1 clock cycle
Hardware utilisation	75% of 320 CLBs
Configuration Program size	13208 bytes

Table 1: Features of the $z/\sin(z)$ filter

The final stage of the filter is the vector merge adder (VMA). This was chosen to be of the carry-select type. It was carefully partitioned and pipelined by one stage to match the maximum latch to latch delay with that in the actual filter part. The latency introduced by the adder was thus 1 clock cycle.

The features of the design are summarised in Table 1. The data word and the output word are 14 bits wide while the internal arithmetic was of 15 bits. The worst case latch to latch delay was 55ns which translates to approximately 18 MHz. The estimate is for a chip with a 70 MHz toggle rate [2]. The chip 3090 is available with 150 MHz as toggle rate and also the identical chip is available in the 3100 series which is fabricated in reduced feature size technology and is advertised to be twice as fast as the corresponding 3000 series chips. Hence the same design should be capable of running several times faster.

Recently, implementation of the above filter in a $1\mu\text{m}$ p-well CMOS process was reported in [3]. The 14000-transistor chip takes up $3.45\text{ mm} \times 4.25\text{ mm}$ on silicon. It operates at 200 MHz and has a total latency of 20 clock cycles.

3 Programmable coefficient FIR filter

In a programmable coefficient filter, multipliers are indispensable. Since multipliers demand a lot of resources, maximal reuse of hardware was essential for efficient design. A concept called pipelined clustering [4] facilitates this by using a clock frequency that is multiple of the data-arrival rate. For an FIR filter, this means time-multiplexing multiple number of taps (multiply-add modules) on a single physical multiply-accumulate module.

Pipelined clustering transforms an FIR filter to that shown in Fig 3. Local interconnections between neighbouring taps give rise to feedback loops with proper number of synchronisation delays. The communication between neighbouring clusters takes place via the accumulation path which also needs proper number of delays to retain the functionality. Multiplexers in the cluster take care of the scheduling of the coefficients and accumulation of the result. The switching instances of the multiplexers and the number of synchronisation delays in the accumulation path and the feedback path have been derived in [5].

It turned out that the size of the multiplexers was one of the deciding factor of the maximum clustering possible. This is typical of the design issues in this

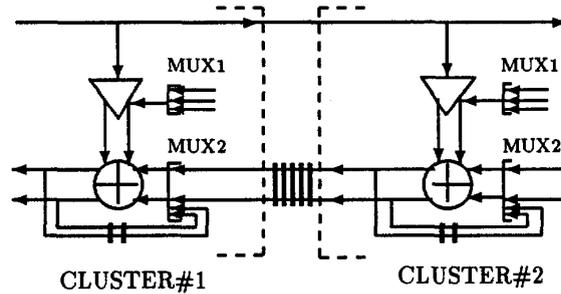


Figure 3: 3→1 clusters

technology. Since a CLB can perform any function of 5 variables, 3:1 is the maximum multiplexing it could do (3 input and 2 select bits). Anything more than that would increase the depth of logic by one more CLB. Thus, initial estimates of both the resource and the speed were in favour of a 3 → 1 cluster. The proper number of delays in the feedback and the accumulation path are 2 and 5 respectively.

A pipelined carry-save array multiplier forms the heart of the design. One input to the multiplier is 8-bit data coming from outside. Another input is one of the three 12-bit coefficients corresponding to three successive taps. The multiplier implements modified Bough-Wooley algorithm for multiplying 2's complement numbers[6]. Each of the CLBs that made up the multiplier was a gated full-adder in the sense that, each CLB generated the partial product bit from the multiplier and multiplicand bits and merged it with the incoming carry and sum bits. For the same reasons as in the previous design, the multiplier was not pipelined at the bit level. Instead, the carry and sum bits were registered after every 2 CLBs. The multiplier produces the result in carry-save form. A 4→2 compressor [7] merges this with the incoming, accumulated data.

For an N-tap filter, $\lceil \frac{N}{3} \rceil$ clusters are to be cascaded with a vector merge adder at the end. Since each cluster was implemented on a different chip, it was absolutely necessary to make the clusters systolic so that, when multiple clusters are cascaded together to build an entire filter there was no speed penalty.

Delay elements were necessary in both the broadcast and the accumulation paths to make the design systolic. Since algorithmic delays are available in the accumulation path only, retiming is applied to get some of these delays on the broadcast path. Since the clusters are time-variant, retiming requires that the switching instances of the multiplexers be modified properly[8].

The final placement of the delay elements is shown in Fig 4. However, this way, the switching instances of the multiplexers get rotated by $2 \bmod 3 = 2$ clock cycles. This resulted in three different designs—different only in the switching instances of the multiplexers. For SRAM based FPGAs this only implies that three slightly different configuration programs have to be created for otherwise identical designs.

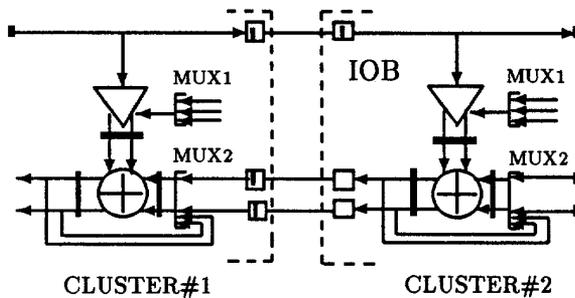


Figure 4: Systolic clusters

Chip	3090PG175-70
No of taps	3
Input word length	8 bits
Output word length	12 bits
Maximum latch to latch delay	35 ns
Clock frequency	29 MHz
Data rate	9.66 Million samples/sec
Hardware utilisation	66% of 320 CLBs
Configuration Program size	14028 bytes

Table 2: Features of the 3→1 cluster

Each of the clusters is complete in itself along with the required control circuitry to take care of coefficient loading and to operate as part of a filter. At a clock speed of 29 MHz, each 3→1 cluster works as 3 consecutive taps of a filter for a data-rate of 9.66 Million samples per second. The features are summarised in Table 2.

4 Summary and Conclusions

We have presented two digital filters; each designed in a single XILINX 3090PG175-70 chip. The first one is an 11-tap constant coefficient FIR filter, which is estimated to run at a worst case clock frequency of 18 MHz. The second design is a systolic module for programmable coefficient FIR filters. Through the use of the technique of pipelined clustering three filter taps are provided through appropriate multiplexing and synchronisation of a single pipelined multiply-add module. At an estimated worst case clock frequency of 29 MHz, the cluster can support a data rate of 9.66 Million samples per second.

The speeds are, however, worst case predicted values for a slow 3090 chip. As already pointed out, a speed-up of several times should be possible for the same designs due to the availability of the same chip in the faster 3100 series.

Effective use of available routing resources is critical for high speed applications. The general purpose routing resource performs decidedly slower than dedicated local interconnects. So local interconnects should be used as often as possible. Also, since these dedicated paths cannot be used for other than particular point-to-point connections, demand on general purpose routing resource becomes much higher if these

local interconnects are not used. This may result in a shortage of routing resources. Hence, a regular design is crucial for both high speed operation and maximum hardware utilisation.

A design involving local interconnects can be suitably pipelined to increase throughput. However, mainly due to the dominance of routing delay, CLB-level pipelining (synonymous to bit-level pipelining) is not worthwhile. It turns out that registering the result after every two CLBs results in balanced delays.

In sum, SRAM-based FPGAs appear suitable for high speed DSP, provided a regular architecture is worked out and routing resources are used judiciously. However, density seems to be the more important deciding factor for the implementability of complex arithmetic algorithms.

References

- [1] Marcelo Alves de Barros and Mohamed Akil, "Study and Implementation of a Real Time 3×3 Programmable Convolver with Reconfigurable Technology", *Proc. Euro ASIC 1992*, pp. 392 - 395.
- [2] The Programmable Gate Array Data Book, Xilinx Inc., 2100 Logic Drive, San Jose, CA 95124, USA, 1992.
- [3] Thu-ji Lin and Henry Samueli, "A 200-MHz CMOS $x/\sin(x)$ Digital Filter for Compensating D/A Converter Frequency Response Distortion", *IEEE Journal of Solid State Circuits*, vol 26, No. 9, pp. 1278 - 1285, September 1991.
- [4] V. Visvanathan, N. Mohanty and S. Ramanathan, "An Area-Efficient Architecture for Real Time VLSI Finite Impulse Response Filters", *Proc. Int. Conf. VLSI Design*, Bombay, January 1993, pp. 166 - 171.
- [5] N. Mohanty, S. Ramanathan and V. Visvanathan, "A new systolic architecture for real-time VLSI infinite impulse response filters", *Proc. Signal Processing and Communications*, Bangalore, January 1993, pp. 190 - 194.
- [6] Mehdi Hatamian and Glenn L. Cash, "A 70-MHz 8-bit×8-bit Parallel Pipelined Multiplier in 2.5- μ m CMOS", *IEEE Journal of Solid State Circuits*, vol. SC-21, No. 4, pp. 505 - 513, August 1986.
- [7] Paul J. Song and Giovanni De Michelli, "Circuit and Architecture Trade-offs for High-Speed Multiplication", *IEEE Journal of Solid State Circuits*, vol. 26, No. 9, pp. 1184 - 1198, September 1991.
- [8] V. Visvanathan and S. Ramanathan, "A Generalized Retiming Theorem for Periodic Digital Systems", *Technical Report*, CAD Laboratory, SERC, Indian Institute of Science, 1993.