Reconfigurable Gate Array Architectures for Real Time Digital Signal Processing

Geoff Liersch
lurch@ee.latrobe.edu.au

Chris Dick
chd@ee.latrobe.edu.au

Department of Electronic Engineering
La Trobe University Bundoora Victoria 3083 Australia

Abstract

The number of usable gates in field programmable gate array (FPGA) logic has recently reached a level which allows their use as computational structures in digital signal processing (DSP) applications. This paper reports on the development of a scalable computing architecture based on Xilinx XC4010 FPGAs for real-time digital signal processing. The architectural requirements for efficient implementation of common DSP algorithms on FPGA platforms are considered. An analysis of the implementation and performance of a high-bandwidth finite impulse response (FIR) filter is presented. A second design using data requantization and spectral shaping to acheive higher order filters is also described.

1 Introduction

Many demanding digital signal processing (DSP) algorithms require a massive amount of computational operations. Often it is desirable to perform these algorithms in real-time or near real-time. Conventional VLSI DSP computers can be too slow to perform this processing. One solution is to build a more homogeneous computing machine in which memory and processing elements are combined.

Development of FPGAs (field programmable gate arrays) has drawn a lot of attention in the last few years, particularly with the introduction of fast CMOS reprogrammable logic devices by Xilinx. The logic capacity of these devices has just crossed the threshold that will allow them to perform complex tasks, as opposed to their conventional application as support logic. The devices can be characterized to perform some function by being electrically loaded. This can be done as often as required, and the time needed to load the configuration is relatively short - only a few milliseconds are required to re-program the chip and assign it a new character and functionality.

Recent work by a group from Paris Research Laboratories of Digital Equipment Corporation using field programmable gate arrays as co-processors has produced some impressive results. A prototype board called Perle-0 utilizing an array of 5 x 5 early generation FPGA chips was built [3]. The group has targeted ten different algorithms from a range of fields including arithmetic, algebra, geometry, physics, biology, audio and video which clearly show the power of this concept [4]. Results show that the FPGA solution can outperform Cray II and Cyber 170/750 by as much as 16 times at tasks such as long multiplication.

This paper describes a "smart memory" architecture for a computer system that allows fast implementation of digital signal processing algorithms. The smart memory system allows computation to be performed by elements that are part of the memory system itself, rather than by the CPU as in conventional computers. The computational elements in the smart memory are implemented using FPGAs which can be dynamically reconfigured by a controlling CPU. By utilising the reprogrammability of the Xilinx FPGA device, the architecture of the processing element can be customised for maximum performance on a particular DSP algorithm. Based on this concept, an initial adaptive computing platform consisting of a four node FPGA system has been developed.

Linear filtering operations are the first DSP algorithm group to be targeted at the FPGA computing platform. An analysis of of the implementation of a finite impulse response (FIR) filter on an FPGA architecture is given. Experimental performance results for the multi-tap FIR filter implemented on the reconfigurable gate array architecture are also presented. A second filter design using data requantization and spectral shaping to reduce the arithmetic complexity is also described.

1058-6393/95 $4.00 © 1995 IEEE

2 System Architecture

The prototype FPGA computing platform consists of a four node system wire-wrapped on a 6U eurocard. Central to each processing node is the XC4010 FPGA device. This device is one of the largest members of the 4000 series family, a third generation FPGA device manufactured by Xilinx. The XC4010 consists of a 20-by-20 array of configurable logic blocks (CLB). Each CLB contains a pair of flip-flops and two independent 4-input function generators for implementing user's logic. In addition, the CLB may be optionally configured to be two 16-by-1 or one 32-by-1 internal RAM or ROM. Programmable interconnect resources provide routing paths between CLBs and Input/Output Blocks (IOB). The IOBs interface between the package pins and the internal signal lines.

A key requirement for a successful signal processing engine is a large bandwidth between the processor and data memory. The adaptive computing platform uses a distributed memory architecture to achieve this large processor-to-memory bandwidth. Each FPGA processor on the system is interfaced to two banks of 256k by 8 or 16 bit memory. These memory banks can be independantly addressed to allow for concurrent memory accesses. Communication between nodes is achieved using Xilinx I/O lines connected in a mesh network topology. The prototype FPGA system includes an external bus interface allowing the configuration and control of the processing nodes by an IBM-PC. Data transfer between the IBM-PC and the processing nodes can also be achieved using this external interface. A second external bus interface is provided to enable real time data aquisition options. The initial interface option to be developed in the Signal Processing Laboratory at La Trobe University is an Analog I/O module. This module contains stereo 16 bit 64 times oversampling converters operating at a 48kHz sampling rate. A frame-grabber I/O module for capture of real time image data is also under construction.

3 Bit Serial FIR Implementaton

The first DSP algorithm considered for implementing on the FPGA computing platform was a real time audio bandwidth FIR filter. A length M FIR filter is described by the difference equation:

$$y_n = \sum_{k=0}^{M-1} a_k x_{n-k} \qquad n = 0, 1, \ldots \tag{1}$$

where y_n is the filter output, x_n are the input samples and the a_k are the filter coefficients.

While there are many methods for implementing a FIR filter, we concentrated on the tapped delay line realisation. The signal flow graph for the standard FIR filter is illustrated in Figure(1).

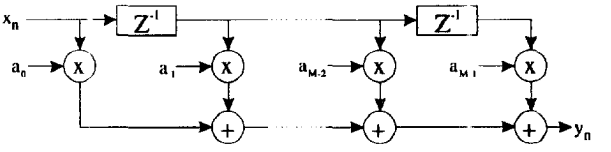

Figure 1: Tapped delay line realization of a FIR filter.

Calculation of each output sample requires M multiplications and $M-1$ additions. For VLSI DSP processors, with similar execution times for multiplication and addition operations, the total number of operations per output sample is $2M-1$. The maximum number of taps that can be acheived with a VLSI DSP implementation of a FIR filter is therefore limited by the speed of the arithmetic operations and the memory-processor bandwidth.

Using an FPGA architecture to implement a FIR filter allows the execution of multiple multiplication and addition operations in parallel. With intelligent use of the internal ram available on the FPGA device, the effective memory bandwith can also be increased. The initial design utilised a bit serial architecture to implement a FIR filter. A bit serial approach provides a method for concurrent arithmetic operations while still allowing the DSP algorithm to be efficiently mapped onto the target FPGA architecture. Bit serial achitecures also have the advantage that they are very routable in FPGA designs which lead to a higher device utilisation.

The FIRST [2] programming language was used to implement the FIR filter design. The FIRST language was conceived to allow DSP algorithms to be easily specified and implemented in VLSI technology. The specification of the this language also includes a default bit serial library of commonly used operations in DSP algorithms such as addition and multiplication. TRANS [1] is a FIRST language compiler that targets FPGA devices. TRANS compiles or translates signal processing algorithms specified in the FIRST language into a netlist format that is suitable for placing and routing using Xilinx software development tools. The TRANS software package also provides a revised version of the FIRST library, called DFIRST, with arithmetic elements more suited to FPGA implementation.

Implementation of an FIR filter utilises only two DFIRST library elements: multiplication and addition. A DFIRST bit serial addition (ADD) can be

implemented in one half of a Xilinx CLB. Several DFIRST primitives exist for the implementation of a bit serial multiplication. Each implementation provides a trade off between size, speed, latency and input word format. For the FIR filter implementation with fixed coefficients, the most efficient implementation in terms of speed is the SHIFTMULT primitive. This primitive allows for the construction of smaller multipliers by using the Canonic Signed Digit (CSD) representation of the fixed coefficient values. Due to the implementation of the SHIFTMULT primitive, the size and latency of the resulting multiplier is dependent on the value of the CSD coefficient. In the worst case, the SHIFTMULT element uses 30 Xilinx CLBs.

Due to the nature of implementation of the SHIFTMULT primitive, a delayed version of the input signal is provided at the output. This allows the storage or delay elements of the FIR filter to be incorporated into the multiplication stage. With latency of the multiplier dependent on the value of the CSD coefficient, additional bit serial delays must be included in each tap of the filter to align the serial bit streams for the addition process. Figure(2) shows the revised filter structure. Note that the data paths shown represent serial bit streams and that all the adders and multipliers are operating concurrently.

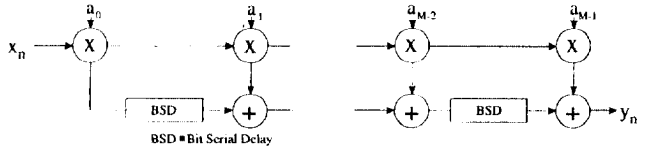

Figure 2: FIRST realization of FIR system.

Using the FIRST programming language a 13 tap FIR filter was constructed. The filter was implemented on a single node of the FPGA adaptive computing platform. The resulting design, including interface logic to the Analog I/O module, utilised 398 of the 400 CLBs available on the Xilinx XC4010, or around 30 CLBs per tap. With the 16 bit A/D converter operating at 48kHz the bit clock speed for the filter was set at 768kHz. While the timing analysis package by Xilinx reported the maximum operating bit clock speed to be in excess of 26 MHz, limitations of the Analog I/O interface did not allow for this to be tested.

4 Reduced Arithmetic FIR Filter

Results from the bit serial implementation of the FIR filter clearly show that unlike DSP processors, multiplications are much more costly than additions in an FPGA architecture. To construct higher order filters, the complexity, and hence size, of the multiplier must be reduced. If the input data to the filter can be requantized to a smaller number of bits, the number of CLBs used in the multiplication stage can be reduced considerably. One such method for requantizing data to implement digital filters is outlined in [5]. The standard method for requantizing data is to use a linear or uniform quantization stage where only the top Q bits of the data word are saved. By using such a method of requantization the dynamic range of the input signal is reduced at the rate of 6 dB per bit of the data word. The reduction in dynamic range of the input signal is due to the requantization stage raising the noise floor uniformly across the spectrum of the signal. In the case of digital filtering, we are only interested in the spectral region specified by the output bandwidth of the filter. By using the information about the bandwith of the filter we can requantize the data to a smaller number of bits while retaining the dynamic range of the input signal within the spectrum of interest. This can be achieved by spectrally shaping the noise generated by the requantization process to lie in the band of frequencies to be rejected by the filtering operation. This requantizer due to harris [6] is shown in Figure(3).

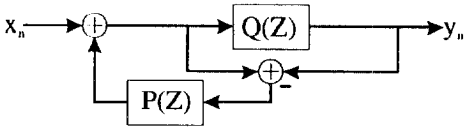

Figure 3: Error feedback requantizer.

Data requantization and spectral shaping of noise provides a method to reduce the number of bits in the data word without loosing dynamic range in the spectrum of interest. This can be used to advantage in the implementation of efficient FIR filters. Consider the case when an P bit input word is requantized to Q bits. From the FIR filter structure in Figure(1), it can be seen that with the requantization process the delayed input samples are also reduced to Q bit values. If we perform the filtering using the tapped delay line structure, reduced bit length multipliers can be used. With one Q bit input, the size of the multiplier can be reduced from P-by-P to P-by-Q with considerable cost savings. Furthermore, for small values of Q, the

multiplication units can be replaced with more area efficient add/subtract stages.

4.1 FPGA implementation

Data requantization and spectral shaping of quantization noise can be used to implement more area efficient FIR filters in FPGA architectures. The data requantization method requires that the bandwidth of interest be a small percentage of the available spectrum. This requirement is necessary to allow the quantization noise to be shaped into the out of band spectrum. From the results obtained in the FIRST FIR filter implementation, it is evident that the potential speed available in the Xilinx FPGA is not being utilised. The 16 bit FIR filter implementation only required a bit clock speed of 768kHz while the maximum bit clock speed was estimated to be in excess of 26Mhz. The additional speed available in the FPGA can be used to upsample the input data to a higher rate. This oversampling of the input data stream provides the additional bandwith required by the data requantization technique of FIR filtering. The new FIR filtering process is indicated in Figure(4).

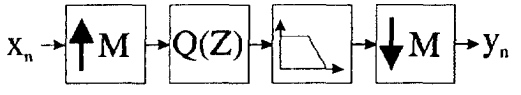

Figure 4: Filtering operation using an error feedback requantizer.

Utilizing the maximum clock speed of the FPGA, input data to the filtering process can be interpolated by a factor as large as 32. By using the additional bandwith provided by the interpolation process the upsampled data can be requantized to a small number of bits using spectral shaping of the quantization noise. The required filtering operation can now be performed on the requantized data stream using reduced bit length multipliers. Data from the filtering process is decimated back to the original sample rate for output.

In order to assess the performance of implementing the modified FIR filter process in an FPGA architecture consider a single bit requantizer with output values of +1 and -1 and a system wordlength of 16 bits. The standard tapped delay line multiply and accumulate operation has now been reduced to the summation of filter coefficient values, where the sign of the delayed input sample is used to determine if the corresponding filter coefficient is added or subtracted. While the input data has been reduced to a single bit, the accumulation data path and hence output data is calculated to the accuracy of the system wordlength of 16 bits. Again we have chosen to implement the FIR filter using a bit serial approach. Figure(5) shows the bit serial implementation of a FIR filter tap with a single bit input stream.

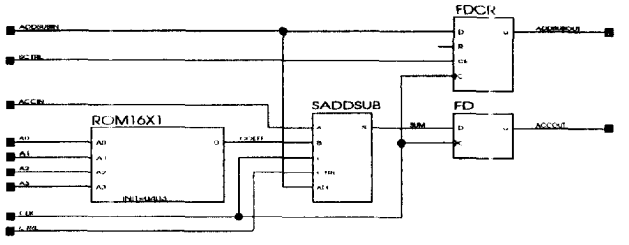

Figure 5: Single Bit FIR filter tap.

Each tap of the bit serial FIR filter can be decomposed into three fundamental building blocks, coefficient storage, a serial add/subtract unit and flip-flops as delay elements. FPGA filter implementations allow coefficients to be stored on-chip or off-chip. Given the performance penalties for external memory accesses, the off-chip solution is inadequate for high order filter implementations. On-chip storage of filter coefficients can be achieved using flip-flops in the FPGA architecture at the cost of one CLB per two bits of storage or 8 CLBs per coefficient. This storage method is necessary when concurrent access to muliple bits of the filter coefficients is required. In a bit serial implementation, concurrent filter coefficient accesses are limited to a single bit. This allows the coefficients to be stored in CLBs configured as ROM's at the cost of one CLB per 32 bits of storage or a half a CLB per coefficient.

The serial add/subtract unit consists of a 1 bit full adder/subtractor with a flip-flop for carry feedback. This unit requires an external control line which is active for the first bit time of each data word to clear the input carry. The serial add/subtract unit is responsible for combining previously accumulated data and the current filter coefficient. Control of the serial add/subtract function is provided by the single delayed input bit. For correct operation of the add/subtract unit, the input bit must remain stable for one system wordlength. The input bit must also be stored to allow it to be passed to the next filter tap. Both these functions can be performed using a single flop-flop with a clock enable that is active for one bit time each system word length. Output from the serial add/subtract unit is delayed by a single bit time to allow for pipeline operation of the addition process. Explanation of the pipeline process can be best

achieved with a simple example. Consider the case of a three tap filter where the n^{th} input sample is S_n. The timing of the filter pipeline is shown in Figure(6) The figure shows that each tap requires 16 bit times to add or subtract its filter coefficient to the accumulator data stream and the addition process has a single bit latency. For correct filter operation each tap requires two control signals and a 4 bit address for the coefficient ROM. It is evident from the pipeline operation that the control signals for each tap can be generated by delaying those signals from its neighbour. If the control lines were generated using this method an additional 2 flip-flops per filter tap would be required. By extrapolating the pipeline operation for a larger number of filter taps we can see that the control signals become repetitive for taps which are multiples of the system word length. The control lines can therefore be added at the cost of additional routing complexity. The problem of address line generation for the filter coefficients can also be solved with careful examination of the pipeline operation. The address lines can be shared by all filter taps if we skew adjacent filter coefficients by a single bit during placement in their ROMs.

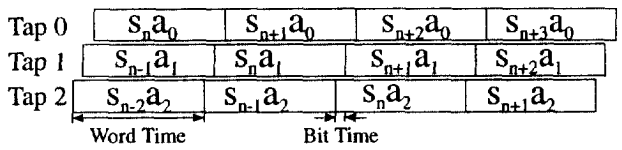

Figure 6: 3 Tap FIR Filter Pipeline Operation.

The resulting FIR filter can be implemented in Xilinx FPGA technology using only two CLBs per tap. This can be decomposed into one half a CLB for coefficient storage, one half a CLB for the serial add/subtract unit and one CLB for delay elements. This method of FIR filter implementaion can be modified for a larger number of quantized bits. For a two bit quantized data stream with levels of +1, 0, and -1 an additional multiplexor is required. The multiplexor is used on the input of the serial add/subtract unit to select between the coefficient data and data of zero. The multiplexor is controlled by the extra bit in the quantized data stream. An additional flip-flop is also required for storage of this additional data bit. This modification can be achieved at the cost of one half a CLB per tap. A similar method can be employed to increase the requantized data stream to three bits by realising that the coefficient can be delayed by a single bit to achieve signal levels of +2 and -2.

5 Summary and Conclusions

The implementation of a FPGA computing platform for the real-time implementation of common DSP algorithms has been described. The design and implementation of bit serial FIR filter on the computing platform has been presented. The resulting filter was acheived at the cost of approximately 30 CLBs per tap. In order to acheive higher order filters a second method of FIR filter implementation using data requantization and reduced arithmetic was investigated. The design resulted in a filter implementation with as little as 2 CLBs per tap. The perfomance increase comes at the expense of a higher system bit clock rate and additional processing before the filtering stage.

Acknowledgements

We gratefully acknowledge fred harris for suggesting the method for implementing the reduce arithmetic FIR filter and his help on the requantization stage. Thanks to Peter Graumann for releasing the TRANS software tools and providing backup support. The FPGA computing platform was constructed with support by Xilinx under their University Program.

References

[1] Peter Graumann and Laurence Turner, *TRANS User's Guide*, Department of Electrical Engineering, University of Calgary, Internal Report, 1993.

[2] P. Denyer and D. Renshaw, *VLSI Signal Processing: A Bit Serial Approach*, Addinson-Wesley, 1985.

[3] P. Bertin, D. Roncin, J.Vuillemin, "Introduction to Programmable Active Memories", *PRL Report-3*, Digital Equipment Corp., Paris Research Laboratory, France 1989.

[4] P. Bertin, D. Roncin, J.Vuillemin, "Programmable Active Memories a Performance Assessment", *Proceedings of the ACM-FPGA Workshop*, February 17-18, 1992 Berkeley, California.

[5] f. harris, Tony Constantinides "On the Requantization of Data to Implement Digital Filters with Reduced Resolution Arithmetic", *International Conference on Signal Processing and its Applications (ISSPA)*, Gold Coast, Australia, 27-29 Aug 1990.

[6] f. harris, personal communication, August 1994.