IMPLEMENTATION OF DIGITAL FILTER USING FPGA

Rakhi Thakur[1], Swati Thakur[2]

Lecturer in Department of Electronics, KNPC, Jabalur Madhya Pradesh, India[1]

M.Tech Student, Department of Electronics, SRIST, Jabalur Madhya Pradesh, India[2]

ABSTRACT

Digital filters are used extensively in all areas of electronic industry. This is because Digital filters have the potential to attain much better signal to noise ratio than analog filters and at each intermediate stage of the analog filter adds more noise to the signal. The digital filter performs noiseless mathematical operations at each intermediate step in the transform so these filters have become popular because their precise reproducibility allows design engineers to achieve performance levels that are difficult to obtain with analog filters.

Keywords: Finite Impulse Response, Infinite Impulse Response, Field Programmable Gate Array.

I INTRODUCTION

FIR and IIR filters are the two common filter forms. A drawback of IIR filters is that the closed-form IIR designs are preliminary limited to low pass, band pass, and high pass filters, etc. Furthermore, these designs generally disregard the phase response of the filter. Compare to IIR filers, FIR filters can have precise linear phase. Also, in the case of FIR filters, closed-form design equations do not exist and the design problem for FIR filters is much more under control than the IIR design problem because there is an optimality theorem for FIR filters that is meaningful in a wide range of practical situations [1]. The creation and analysis of representative data can be a complex task. Most of the filter algorithms require Multiplication and addition in real-time. The unit carrying out this function is called **MAC** (multiply accumulate). Depends on how good the **MAC** is, the better **MAC** the better performance can be obtained. Once a correct filter response has been determined and a coefficient table has been generated, the second step is to design the hardware architecture [3].

II DESIGN AND IMPLEMENTATION OF DIGITAL FIR FILTER

MATLAB combines the high-level, mathematical language with an extensive set of pre-defined functions to assist in the creation and analysis of filter data. Toolbox are available for designing filter response and generating coefficient tables, each with varying levels of sophistication. Graphical filter design tools provide selections for specifying pass band, filter order, and design methods, as well as provide plots of the response of the filter to various standard forms of inputs. Three choices of technology exist for the implementation of filter algorithms. These are:

1. Programmable DSP chips

2. ASICs and
3. FPGAs.

At the heart of the filter algorithm is the multiply-accumulate operation Programmable DSP chips typically have only one MAC unit that can perform one MAC in less than a clock cycle. DSP processors or programmable DSP chips are flexible, but they might not be fast enough. ASICs can have multiple dedicated MACs that perform DSP functions in parallel. But, they have high cost for low volume production. FPGAs have been praised for their ability to implement filters since the introduction of DSP architectures, which can be efficiently, realized using dedicated DSP resources on these devices. More than 500 dedicated multiply-accumulate blocks are now available, making them exceptionally well suited for high-performance, high-order filtering applications that benefit from a parallel, non-resource shared hardware architecture. In this particular paper, FPGA has been chosen as the implementation tool [7].

To program FPGA, hardware description language is needed. VHDL or verilog synthesis offers an easy way to target a model towards different implementation. High-performance, high-order filtering applications, that are able to exploit dedicated multiplier or DSP blocks, often turns to FPGAs for a solution. When targeting an FPGA device with dedicated DSP blocks capable of supporting cascaded “multiply-add” operations such as the Xilinx Virtex 4, highest performance is achieved using a transposed” architecture.

III IP GENERATORS

When implementing the actual filter on an FPGA, the designer is presented with a fundamental choice of whether to use an IP core or design a custom implementation. Both options have merits and limitations. FIR filter IP cores are readily available from multiple sources with the most common forms being technology-specific enlists, synthesizable RTL, and synthesizable MATLAB. All these generators can construct a filter using a pre-defined coefficient table. IP is typically the most cost effective and offers the best results but provides the fewest options.

IP generators provide an excellent choice when time-to-market or ease-of-use demands prevail. The general purpose nature of these programmable IP generators, however, seldom yields the optimal hardware for a specific application. Implementing hardware design in Field Programmable Gate Arrays (FPGAs) is a formidable task. There is more than one way to implement the digital FIR filter. Based on the design specification, careful choice of implementation method and tools can save a lot of time and work [6]. MatLab is an excellent tool to design filters. There are toolboxes available to generate VHDL descriptions of the filters which reduce dramatically the time required to generate a solution. Time can be spent evaluating different implementation alternatives. Proper choice of computation algorithms can improve the FPGA architecture to make it efficient in terms of speed and/or area [8].

IV DESIGN SPECIFICATIONS

The objective of system is to provide a hardware platform to test FIR filters that have been generated using MATLab and verilog. The system performs the following functions:

1. Communicate with a PC using a standard RS-232 serial interface.
2. Receive a data file from the PC. Data will be received in binary form so they can be applied to the FIR filter without further transformation. For this exercise consider that the FIR filter receives 8-bit samples.
3. The outputs of the FIR filter must be returned to the PC where they will be stored in a file. Again assume data from the FIR filter have to be sent the outputs in binary form (without transformation).
4. To send and receive the files from/to the PC a program like "HyperTerminal" can be used. Configure the port as follows:

   Bits per second: 9600

   Data bits: 8

   Parity: None

   Stop bits: 2

   Flow control: None

The HyperTerminal programs are set up now and ready to communicate with the Spartan board. We can type a few keys on one hyper terminal and observe the observe characters appearing on other hyper terminal. Note that the received words are stored in the FIFO buffer and data are read only when the FIFO buffer fills with 256 words [2]. The overview of the system is shown in figure 1.RST is an Input and it is asynchronous signal that initializes all internal pointers and output registers.

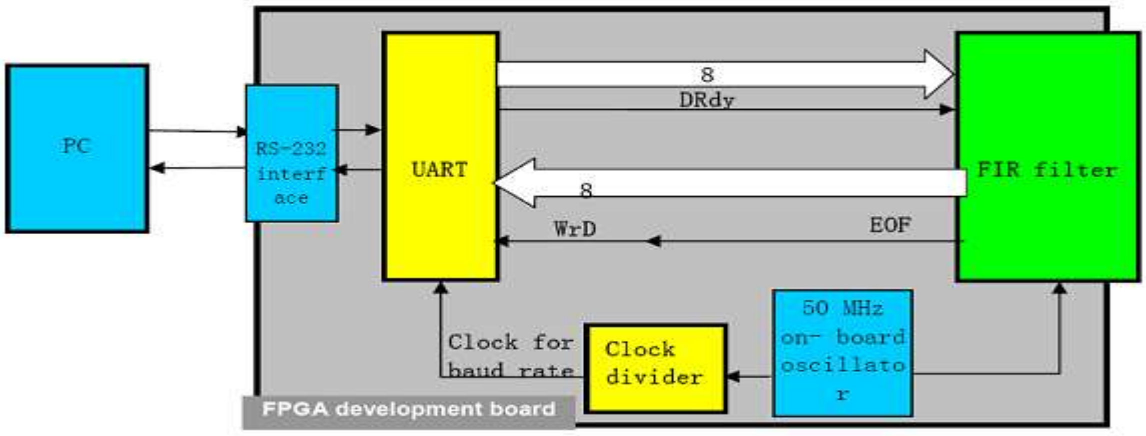

Figure 1: Overview of the system

In above figure, there are two blocks that do not have to be designed: the RS-232 interface, it is provided as part of the development board, and the FIR filter, it is implemented using MATLab SYSGEN. The blocks that have to be developed are the Universal Asynchronous Receiver-Transmitter (UART), is a circuit that sends parallel data through a serial line. UARTs are frequently used in conjunction with the RS-232 standard, which specifies the electrical, mechanical, functional, and procedural characteristics of two data communication equipment.

A UART includes a transmitter and a receiver. The transmitter is essentially a special shift register that loads data in parallel and then shifts it out bit by bit at a specific rate. The receiver, on the other hand, shifts in data bit by bit and then reassembles the data. The serial line is 1 when it is idle. The transmission starts with a start bit, which is 0, followed by data bits the number of data are 8. The number of stop bits that are used are 2. Here the LSB of the data is transmitted first [4].

FIFO:

The Xilinx LogiCORE IP FIFO Generator is a fully verified first-in first-out (FIFO) memory queue for applications requiring in-order storage and retrieval. The core provides an optimized solution for all FIFO configurations and delivers maximum performance (up to 500 MHz) while utilizing minimum resources. Delivered through the Xilinx CORE Generator™ software, the structure can be customized by the user including the width, depth, status flags, memory type, and the write/read port aspect ratios.

INTERFACE CIRCUIT

The design uses a FIFO buffer. The FIFO buffer provides more buffering space and further reduces the chance of data over run. We can adjust the desired number of words in FIFO to accommodate the processing need of the main system. The main system obtains the data from FIFO's read port. After retrieving 256 words, it asserts a signal of the FIFO one clock cycle to remove the corresponding item. The prog_empty signal of the FIFO can be used to indicate whether any received data word is available. A data-overrun error occurs when a new data word arrives and the FIFO is full [5].

The design of a UART transmitting system is similar to that of the receiver. It consists of a UART transmitter, baud rate generator, and a wrapper. The UART transmitter is essentially a shift register that shifts out data bits at a specific rate. The rate can be controlled by one-clock-cycle enable ticks generated by the baud rate generator. Because no oversampling is involved, the frequency of the ticks is slower than that of the UART receiver. FPGA designing is a synchronous design the reset from the push button on FPGA is synchronized first and to reduce the noise debouncing circuit is added and the active reset(if held high continously for 1000ns a pulse of 300ns is generated) is given as an input to receiver, FIFO and transmitter. Here synchronizer is for receiver hence it is having the clock frequency as 50 MHz. If it is for a transmitter than the clock frequency is being doubled and it will be 100 MHz. As reciever uses 50MHz and transmitter uses 100 MHz frequencies, in order to generate and synchronize both the clock domains, DCM is used.

DESIGN SPECIFICATION OF FIR FILTERS

Low-pass FIR filter has been implemented with

- Response type: Low pass;
- Design method: Equi ripple;
- Density factor: 20;
- Filter order: order 5;
- Hardware architecture: Direct form;
- Sampling frequency: 48000Hz;
- Pass band frequency: 9600Hz;
- Stop band frequency: 12000Hz;
- Input data length: 8 bits;

• Output data length: 8 bits;

HARDWARE UTILIZATION

Total memory usage in Proposed Algorithm is around 240 MB. Design summary of circuit is shown in Table 1 and Table 2.

Table 1: Design Summary of the system

Logic utilization	Used	Available	Utilization
Number of slices	2782	42176	6%
Number of DSP48s	52	160	32%
Number of LUTs	5113	84352	6%
Number of Bonded IOBs	48	576	8%
Number of GCLKs	1	32	3%
Number of Slice Flip Flops:	4701	84352	6%

Table 2:Design Summary of the FIR Filter

S.No.	Specification	Value
1	Minimum period:	2.832ns (Maximum Frequency: 353.107MHz)
2	Minimum input arrival time before clock:	1.306ns
3	Maximum output time after clock:	3.793ns
4	Maximum combinational path delay:	Maximum combinational path delay:
5	Clock period:	2.832ns (frequency: 353.107MHz)
6	Total number of paths / destination ports	40654 / 18972
7	Number of failed paths / ports	0 (0.00%) / 0 (0.00%)\

TIMING WAVEFORM

Minimum period required for FIR filter is 2.832ns and Maximum Frequency is 353MHz. Figure 2 shows the simulation waveform of FIR Filter.

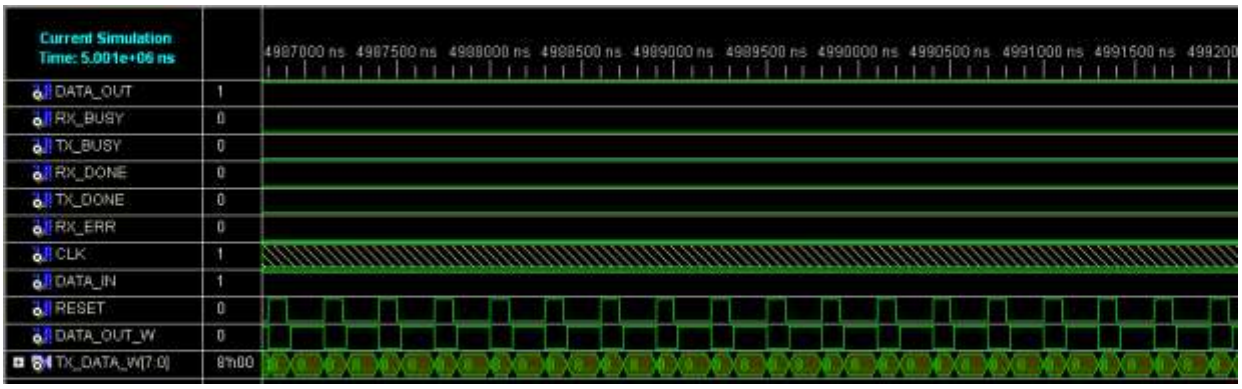

Figure 4: Simulation behavior of the Filter

CONCLUSION

The traditional methodology for designing a filter consists of two phases: system specification and hardware implementation. Both phases require multiple iterations and can involve a four-to-six-week process just to ensure that a single functional block operates to specification within the system. Creating this filter from the ground up, using either VERILOG or other design entry methods, would have required too much development time. MatLab is an excellent tool to design filters. There are toolboxes available to generate VERILOG descriptions of the filters which reduce dramatically the time required to generate a solution.

REFERENCES

[1.] J.G. Proakis and D.G. Manolakis, Digital Signal Processing: Principles, algorithms and Applications. Upper Sad dle river, NJ: Prentice-Hall, 1996.

[2.] http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_4/ug440.pdf3.

[3.] Sheenu Thapar, "A Low Pass FIR Filter Design Using Genetic Algorithm Based Artificia Neural Network "International Journal of Computer Technology and Electronics Engineering (IJCTEE) Volume 2, Issue 4, August 2012, pp. 99-103

[4.] A.Shaw and M.Ahmed, "Pipelined recursive digital filters: a general look ahead scheme and optimal approximation," IEEE Trans. On Circuits and Systems II: Analog & Digital Signal Processing, Vol.46, no.11, Nov. 1999, pp. 1415-1420.

[5.] Chao-Huang Wei, Hsiang-Chieh Hsiao, Su-Wei Tsai "FPGA Implementation of FIR Filter with smallest Processor" IEEE NEWCAS conference, 19-22 June 2005, pp.337-340.

[6.] R. A. Hawley, B. C. Wong, T.J. Lin, J. Laskowski, and H. Samueli, "Design techniques for silicon compiler implementations of high-speed FIR digital filters," IEEE Journal of Solid-State Circuits, vol. 31, May 1996, pp. 656-667.

[7.] A.G.Dempster and M.D.Macleod, "Use of minimum adder multiplier blocks in FIR digital filters," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol.42, no.9, Sep.. 1995, pp. 569-577.

[8.] Hourani, Y. Kim, S. Ocloo, and W. Alexander, "Automated hardware IP generation for digital signal processing applications," in Signals, Systems and Computers, 2006. ACSSC ' 06. Fortieth Asilomar Conference on, (Paci_c Grove, CA, USA), Nov. 2006 pp. 1156-1160.