High Speed Reconfigurable FPGA Based Digital Filter

Navaid Z. Rizvi[1], Raaziyah Shamim[2], Rajesh Mishra[1], and Sandeep Sharma[1]

[1] School of Information and Communication Technology, Gautam Buddha University, India
[2] Department of Electronics and Communication Engineering, JIIT, Noida, India
{navaid,sandeepsharma}@gbu.ac.in, raaziyah.shamim@jiit.ac.in

Abstract. Digital Finite Impulse Response filters are essential building blocks in most Digital Signal Processing (DSP) systems. A large application area is telecommunication, where filters are needed in receivers and transmitters, and an increasing portion of the signal processing is done digitally. However, power dissipation of the digital parts can be a limiting factor, especially in portable, battery operated devices. Scaling of the feature sizes and supply voltages naturally helps to reduce power. For a certain technology, there are still many kinds of architectural and implementation approaches available to the designer. In this paper, a reconfigurable FPGA based pipelined FIR filter is implemented and analyzed. This realized FIR filter is compared for area, power dissipation and data processing rate (throughput). Simulation and compilation of the VHDL code written for the implementation of FIR filters is done using Mentor Graphics ModelSim. For the synthesis targeting to FPGA Xilinx Virtex II Pro XP2VP30 device Xilinx ISE Design Suite 10.1 tool is used. Power estimation is done using Xilinx Xpower tool. FPGA implementation of FIR filter model with respect to power, silicon area, and data processing rate (throughput) is analysed.before and after the abstract. This document is in the required format.

Keywords: DSP, FPGA, FIR Filter, FSM.

1 Introduction

Frequency selective digital filters are required in most DSP (digital signal processing) applications. A typical example is mobile communications where hand-held, battery supplied, devices, such as cellular phones, are used. To obtain a long uptime between recharges of the battery for cellular phones, low power consumption is required. Due to the requirements on high data rates in many communication systems, the corresponding subsystems and circuits must have a high throughput as well. Since significant parts of such communication systems are customer products that are produced in large quantities and are sold at low prices, efficient, fast, and reliable design methods as well as low cost circuit implementations are required. The need for miniaturization of systems also requires subsystems with low power consumption. For such integrated systems, the heat dissipation and the cooling becomes a problem. Low power consumption and circuit area are therefore key design constraints [1].The output data rate of the system also plays a major role in system design. Generally, a tradeoff between different design constraints has to be done to get optimal FIR filter.

In this paper, a reconfigurable FPGA based pipelined [2] FIR filter is implemented and evaluated for different design constraints such as circuit area, power dissipation and data processing rate. The results are compared with non-pipelined FIR filter.

2 FIR Filter Architecture

The architectures for the FIR filter implementation used are [4]:

- 1-MAC without pipeline FIR filter model
- 1-MAC with pipeline FIR filter model

The design requirements of both the filter implementations are that they must finish 64 tap calculations within time period of 700ns.The description of the two FIR filter architectures implemented is given in the following sections:

2.1 1-MAC without Pipeline FIR Filter Model

The 1-MAC without pipeline FIR filter model implemented comprises of no pipeline as well as no parallel mechanism. The hardware structure of this model is shown in figure 1. There are four main components in this architecture control which are the central control unit, coefficient address RAM, sample RAM and multiplier.

Control component also comprises Finite State Machine (FSM). The function of control unit is to receive the start of the signal of the filter from outside. The data address from the control component is received from coefficient address RAM component and this component also reads out the sample data address and coefficient data from block RAM component, and in turn sends sample data address to the sample RAM component. The function of sample RAM component is to generate sample data in accordance to the address. After this the sample data and coefficient data access the MAC component. Then the process of multiplication of the sample data and coefficient data with MAC component take place and this multiplication process has been repeated exactly the same time as number of taps. Once the control component has been activated, the sending of read address to the coefficient address RAM component does not take place every clock cycle. The sending of second address only take place after the processing of last calculation and then the result is stored in the register of MAC component. This implies that mostly in the clock cycles the most component remains in idle state and earlier process results always remains in the hardware. In this model there is only one data channel between the components. This indicates that for every clock cycle only one sample data and one coefficient data can access the MAC component. This model only comprise of one multiplier accumulator component. The central control unit, implemented, utilizes eight states. The idle state is initial state. After reset signal turns to 1 the FSM turns to reset which in turn assigns in initial value to output. The arrival of new sample set the start coefficient RAM signal active. After the start and FIFO signal have been set to 1, the FSM goes to loadSample state, which makes the reading of new sample from outside and storing it in sample RAM component. After two clock cycle now FSM goes to step1 state, which makes sending of one address to the coefficient address RAM component. In the next clock cycle the reading of coefficient and sample address by the coefficient address RAM component take place and sample RAM component waits for state step1 and step2. In state step1 and step2 enable signal for MAC component has been turn off. In state step3 enable signal to MAC component has been turn on which begins the process of input coefficient and sample data from it. Till the last tap not reached, FSM continues to go to state step1 and processing of new coefficient and sample continues. After reaching the last tap, state waitresult arrived, which implies end of processing period**.** Now the FSM goes to state waitResult for 3 clock cycles and this has been to compromise delay in the MAC component. Finally the last state writeResultToFIFO arise which in return sends the final accumulated result to output register.

2.2 1-MAC Pipeline FIR Filter Model

The 1-MAC pipeline FIR filter model contains pipeline but no parallel mechanism. The hardware structure is depicted in figure 2. The basic principle for this model has been that with the initialization of FIR filter every clock cycle control component sends read addresses to coefficient address RAM component, which in turn discarded the waiting time for the output from previous data. The next read address follows the last read address in the coming clock cycle which implies read address updated continuously every clock cycle till the final stage reached. The data transmission between the components has been continuous. This model contains one MAC component. The basic function and structure of the sampleRAM Coefficient Address RAM and multiplier accumulator components used in this model is the same as for sample RAM component in 1-MAC without pipeline FIR filter model.

The central control unit, implemented by [3], utilizes seven states in this model. After reset signal turns to value 1, the FSM goes to idle state which has been the initial state. In this state, all the initial values have been assigned. After the arrival of new sample, the input signals start and fifoEmpty have been set to value 1. Due to impact of these signals, the FSM changes its state to loadSample state. In loadSample state, new sample has been read from DSP processor and stored into sample block RAM component. Due to pipeline1 state the control component emits read address continuously followed by every clock cycle. These results in forming data flow through address bus which flows through the control component, coefficient address RAM component and sampleRAM component. This state repeats until the proposed tap number processed. After this state, there have been read addresses (as the number of taps) in coefficient address RAM component. The pipeline2 state helps in achieving the delay processing caused from MAC processing. With consideration of delay, the MAC component multiplies the coefficient and sample, and accumulated them from the fifth clock cycle of pipeline1 state till the last cycle of pipeline2 state. The MAC signal is active throughout this both states. When last coefficient data and sample have been sent to MAC, FSM goes waitResult state. Duration of this state has been equal to the delay in the multiplier accumulator component. The last state has been writeResultToFIFO state and then FSM turns to the first state idle until the control signal will reset to 0 value.

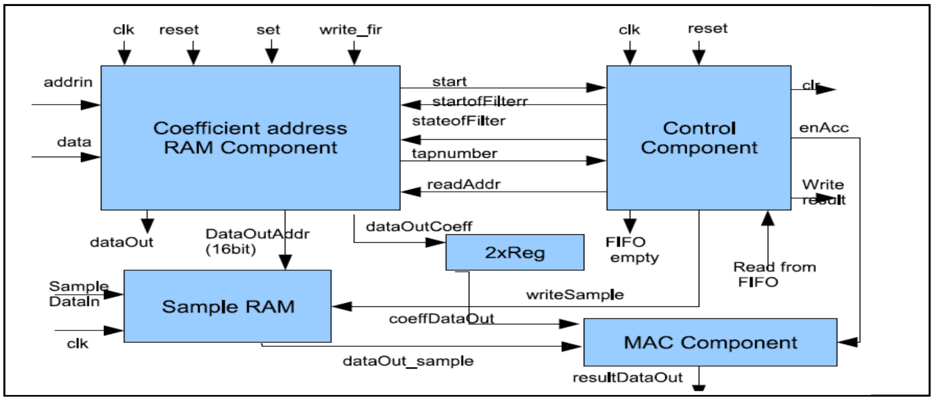

Fig. 1. Hardware architecture of 1-MAC without pipeline FIR filter model

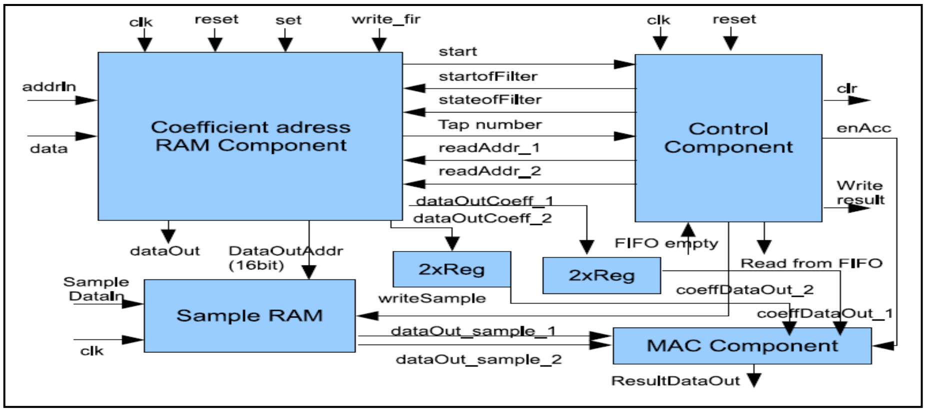

Fig. 2. Hardware architecture of 1-MAC pipeline FIR filter model

3 Simulation and Synthesis of FIR Filters

Firstly, the VHDL code for a particular design, the develop logic is written and tested. Simulation and Compilation is done using Mentor Graphics ModelSim PE Student Edition 6.4a. ModelSim [5] is a simulation and a debugging tool for VHDL, Verilog, and other mixed-language designs from Mentor Graphics [4]. It includes a simulator for VHDL code with complete debugging environment, a source code viewer/editor, waveform viewer, design structure browser, list window, and a host for other features designed to enhance productivity. The design in this paper is targeted for a Xilinx Virtex II Pro XP2VP30 FPGA [6].The Xilinx Virtex-II Pro is a 130nm CMOS nine-layer (copper) FPGA device. It has embedded blocks, consists of a two-dimensional array of configurable logic blocks (CLBs), and programmable interconnect resources. Each CLB has four identical sub-blocks, called slices. A slice comprises two four-input LUTs (whose function are used to map four-input Boolean logic), two Flip

Flops, gates (two AND2, one OR2, two XOR2) which are used to implement arithmetic functions such as carry chains, multiplexers, inverters, and buffers. After synthesis four multiplexers are used as mapped multiplexers and other multiplexers as configurable multiplexers for routing inside the slice. The inverters and the buffers are needed to implement the different clock types allowed in the FPGA. The four slices of one CLB share a common programmable I/O crossbar for communicating with a switch matrix (to realize communication with others CLBs and to route a signal). A CLB, together with its associated switch matrix, is called tile. The Virtex-II Pro contains platform FPGAs for designs that are based on IP cores and customized modules. The family incorporates multi-gigabit transceivers and PowerPC CPU blocks in Virtex-II Pro Series FPGA architecture. It empowers complete solutions for telecommunication, wireless, networking, video, and DSP applications. Here the Block selected RAM and memory modules contribute around 18 Kb storage elements of True Dual Port RAM. The RocketIO Transceiver is a Full-Duplex Serial Transceiver (SERDES) capable of Baud Rates from 600 Mb/s to 3.125 Gb/s. The Select RAM+ Memory module provides upto 1378 Kb of distributed SelectRAM + resources. The Arithmetic functions are performed from 18-bit x 18-bit multiplier blocks. This FPGA contains High-Performance Clock Management Circuitry.

4 Results and Discussions

4.1 Simulation Results for Control Component Used for without Pipeline Mechanism

The testbench of Control component used for without Pipeline mechanism includes starting with start single assigning to value of 1. At 1530 ns the fifoempty has been assigned to 0 value which marks change of idle to loadSample state. After this FSM goes through state step1, waitForCoeff1, step2, and step3. The four taps proccessing implies FSM to process four times. Now the FSM reached to states waitResult and writeResultToFIFO. After crossing these two states one complete cycle is finished. The function of ennAcc signal is to active the MAC component. The simulation result is depicted in figure 3.The output indicates correctness in functionality.

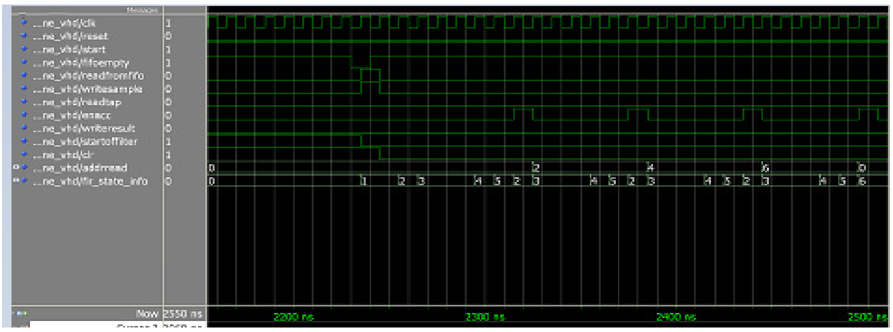

Fig. 3. Simulation waveform for Control component used without Pipeline mechanism

4.2 Simulation Results for Control Component Used for Pipeline Mechanism

Pipeline mechanism only involves the states - idle, loadsample, waitResult and, writeResultToFIFO without pipeline mechanism. The testbench includes starting of start signal. At 2230 ns fifoEmpty signal is assigned value 0, FSM changes its state from idle to loadsample1. After this it passes the state of loadsample1, pipeline1, pipeline2, waitResult and writeResulttoFIFO. The pipeline1 state repeats for 32 times in between 2260 ns to 2580 ns. This state also results in addread to read address from value 0 to 31 in increasing order. At loadsample state readFromFIFO and writeSample signal are also active. The simulation result is depicted in figure 4.

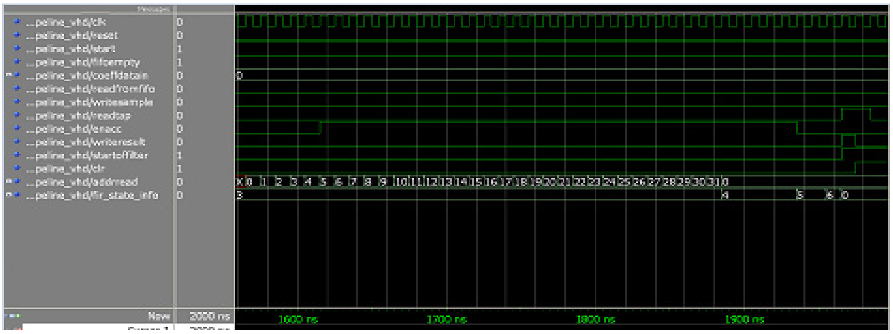
Messages
...peline_vhd/clk 0
...peline_vhd/reset 0
...peline_vhd/start 1
...peline_vhd/fifoempty 1
...peline_vhd/coeffdatain 0
...peline_vhd/readfromfifo 0
...peline_vhd/writesample 0
...peline_vhd/readtap 0
...peline_vhd/enacc 0
...peline_vhd/writeresult 0
...peline_vhd/startoffilter 1
...peline_vhd/clr 1
...peline_vhd/addrread 0
...peline_vhd/fir_state_info 0
X 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 0
3 4 5 6 0
Now 2000 ns
1600 ns
1700 ns
1800 ns
1900 ns

Fig. 4. Simulation waveform for Control component used for without Pipeline mechanism

4.3 FIR Filter Model - Simulation Results

For both the filters, the testbench is written considering one sample reads for the input sample file every 700ns. The function of the main model is to process the data and emit them to 32 bit port resultdataout. For every data processing period there is only one new sample to read. Four taps are processed per period. The simulation waveforms are shown in figure 5 and figure 6.

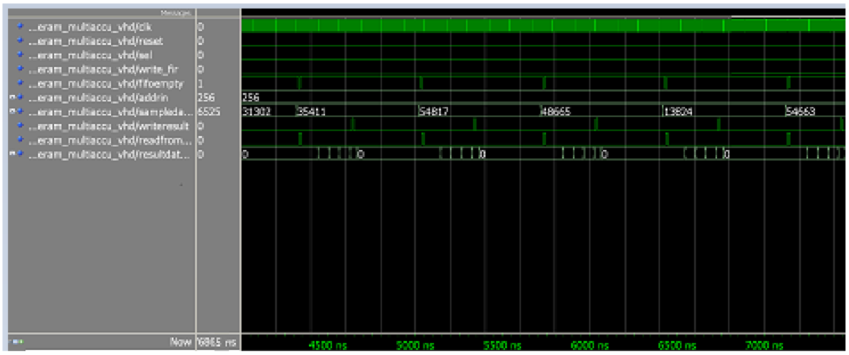
Messages
...eram_multiaccu_vhd/clk 0
...eram_multiaccu_vhd/reset 0
...eram_multiaccu_vhd/sel 0
...eram_multiaccu_vhd/write_fir 0
...eram_multiaccu_vhd/fifoempty 1
...eram_multiaccu_vhd/addrin 256
...eram_multiaccu_vhd/sampleda... 6525
...eram_multiaccu_vhd/writeresult 0
...eram_multiaccu_vhd/readfrom... 0
...eram_multiaccu_vhd/resultdat... 0
256
31302 35411 54817 48665 13824 54563
0 0 0 0 0
Now 6865 ns
4500 ns
5000 ns
5500 ns
6000 ns
6500 ns
7000 ns

Fig. 5. Simulation waveform for 1-MAC without pipeline FIR filter model

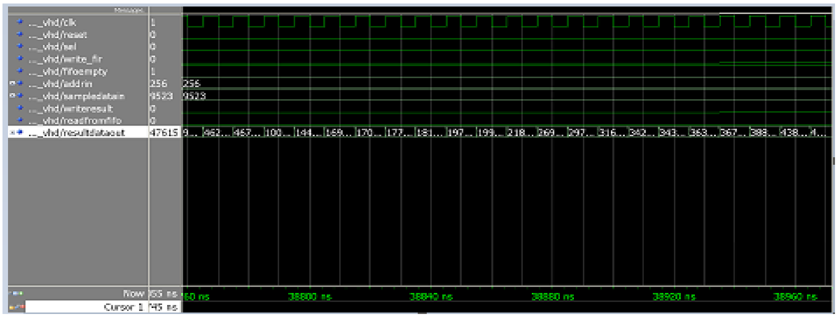

Fig. 6. Simulation waveform for 1-MAC with pipeline FIR filter model

4.4 Synthesis Results Targeted to FPGA for Pipeline and Non-pipeline FIR Filter Models

For The synthesis report, device utilization report in percentage and timing report for pipeline and without pipeline FIR filter models are depicted in table 1, table 2, and table 3 respectively.

Table 1. Timing Report

	Without pipeline	With Pipeline
Minimum period	7.210ns	6.747ns
Minimum input arrival time before clock	4.061ns	4.043ns
Maximum output required time after clock	5.246ns	5.246ns
Maximum combinational path delay	6.654ns	6.654ns
Maximum Frequency	138.706MHz	148.225MHz

Table 2. Device Utilization Report

FPGA Resources	Without Pipeline	Pipeline
Slices	1%	1%
Slices Flip Flops	1%	1%
4 input LUTs	1%	1%
IOBs	44%	44%
BRAM	14%	3%
MULT18x18s	0.74%	0.74%
GCLKs	6%	6%

Table 3. Synthesis Report

FPGA Resources	Without pipeline	Pipeline
RAMs	5	5
-64x16 bit single port RAM	1	1
-64x32 bit dual port RAM	4	4
Multipliers	1	1
16x16 bit multiplier	1	1
Adders/Subtractors	5	5
14-bit addsub	1	1
14-bit subtractor	2	2
3-bit adder	1	1
6-bit adder	1	1
Counters	1	1
14-bit up counter	1	1
Accumulators	1	1
33-bit up accumulator	1	1
Registers	63	63
1 bit register	39	38
14 bit register	2	2
16 bit register	6	7
3 bit register	1	1
32 bit register	11	11
5 bit register	1	1
6 bit register	3	3
14 bit comparator less	1	1
3 bit comparator less	1	2
6 bit comparator less	1	1
32 bit 4 to 1 MUX	1	1

4.5 Power Estimation Results

The power estimation [7] results targeted to FPGA for the two FIR filter models are depicted in table 4. The table shows power (in watts) consumed by clock, logic, signals, IOs as well as Total Quiescent, Total Dynamic and Total Power dissipated by the Control components in the two models. The throughput (time per sample) at 100 MHz clock frequency is calculated to be 70ns/sample for without pipeline and 12.1 ns/sample for with pipeline FIR filter model. Thus, adding pipeline mechanism has increased the data processing rate by five to six times. There is very little difference in the FPGA resources (i.e. area) consumed between the two filter architectures. Adding pipelinism has reduced power dissipation by some amount. The results clearly emphasize that FIR filter with pipeline is a better choice for FPGA filter implementation than FIR filter without pipeline.

5 Conclusion

Single MAC FIR filter models with and without pipeline mechanisms are implemented in VHDL, simulated and synthesised. The two architectures are evaluated for resources (area) consumed, power dissipation and data processing rate. The filter architecture employing pipeline mechanism outperforms the one without pipeline mechanism.

References

1. Khorbotly, S., Carletta, J.E., Veillette, R.J.: A Methodology for Implementing Pipelined Fixed-Point Infinite Impulse Response Filters. In: 41st Southeastern Symposium on System Theory, March 15-17, University of Tennessee Space Institute Tullahoma, TN (2009)
2. Shaw, A., Ahmed, M.: Pipelined recursive digital filters: a general look-ahead scheme and optimal approximation. IEEE Trans. On Circuits and Systems II: Analog & Digital Signal Processing 46(11), 1415–1420 (1999)
3. Wei, C.-H., Hsiao, H.-C., Tsai, S.-W.: FPGA Implementation of FIR Filter with smallest Processor. IEEE (2005)
4. Pirsch, P.: Architectures for Digital Signal Processing. John Wiley & Sons, Chichester (1998)
5. ModelSim User Manual, Software Version 6.4a, Mentor Graphics Corporation (2008)
6. Xilinx. Vitex-II Pro and Virtex-II Pro X FPGA User‘s Guide, Xilinx, Inc. (2007)
7. Xilinx. Xilinx Power Estimator User Guide, Xilinx, Inc. (2007)
8. Xilinx, Jiang, Z., Willson, A.N.: Efficient digital filtering architectures using pipelining/interleaving. IEEE Trans. Circuits Systems–II 44, 110–118 (1997)