Review Study on a Design of Finite Impulse Response Filter

Gopichauhann, Pavan Kumar, Vijaya Kumar

Abstract - This paper is a review study of a design for Finite impulse response (FIR) filter with minimum hardware and software cost with high performance. In the VLSI implementation, the hardware complexity of the FIR filter is directly proportional to the tap-length and the bit-width of input signal. The key observations of this paper is that how the software processing performs with minimum hardware resources, without effecting the performance of original FIR filter.

Key Words - FIR filter, VLSI Implementation, Tap-length, Bit-width

1. Introduction

In digital signal processing applications two primary types of digital filters are used. FIR (Finite Impulse Response) filters and IIR (Infinite Impulse Response) filters. some advantages of FIR filters are

- Phase linearity
- Finite-precision
- Implementation
- Suitable for multi-rate applications

The realization of FIR filters can be done either by software or hardware solutions.

1.2 Contribution

Our major focus of this paper is to implement smallest processor using following instructions 'IDLE', 'LOAD', 'ADD', 'STORE' and 'JUMP' by Using three different blocks in memory 'RAM','ROM','COEFFICIENT' in Verilog HDL Quartus-II tool. The entire paper is organized as follows: Section 2 Implementation structure of FIR filter, Section 3 Introduction to FPGA implementation, Section 4 programming model. Section 5 explore Simulation, Section 6 shows implementation results, Section 7 conclusion.

2. Implementation Structure

Minimal operations as per FIR mathematics analysis are

- Loading
- Storing
- Delay
- Multiplication

The 'delay' operation can be done with 'loading' and 'storing'; the 'multiplication' is inefficient with software processing and hardware resource consuming; therefore it will be placed outside the CPU architecture. Thus, only four type operations remain for the CPU to process.

2.1 FIR Filter

FIR filter is a type of a digital filter. The impulse response is finite because it settles to zero in a finite number of sample intervals. The impulse response of an Nth-order FIR filter lasts for N+ 1 sample, and then dies to zero. Firstly Time Domain input data is transformed into the Frequency Domain using a Fast Fourier Transform (FFT), filtered directly in the Frequency Domain by passing (or) amplifying desired frequencies and zeroing out unwanted ones, and inverse transformed back into the Time Domain using an Inverse Fast Fourier Transform (IFFT).

Our method is designed in Verilog HDL that can be synthesized on a single FPGA; it requires 140 LCs, 95 Registers resources of an Altera Cyclone II EP2C5T series FPGA [2, 5, 6].

System on a chip (SOC) is an integrated circuit that includes a processor, a bus, and other elements on a single monolithic substrate.
A system on a chip may include a configurable logic unit. The configurable logic unit includes a processor, interface, and a programmable logic on the same substrate.

3. FPGA Implementation

Recently there has been an increased interest in the design of DSP systems on Field Programmable Gate arrays (FPGAs). Several implementations of FIR filters have been reported in the literature. All of these implementations use traditional number systems. This paper describes the implementation of a Fermat Number ALU as a processing element (PE) for FIR filters implemented over independent channels. Using Replicated Finite Rings one gains large dynamic range with the advantage of replication of a single modulus ALU. In the VLSI design of a FIR filter, using a CMOS process, based on Replicated Finite Rings is presented. The results shows that this method offers a higher throughput rate and power saving over a comparable binary implementation.

The complete processor was designed in VHDL that can be synthesized on a single FPGA[5,6,7]; it requires 145 LCs, 71 Registers and 256 bits internal memory, less than 3% resources of an Altera Cyclone II EP2C5T series FPGA.

System architecture of FIR processor having address bus, data bus, input port 8-bit address moves to the input port from address bus and data access to the data bus from the input port. Register is 5-bytes address moves to the input port from address bus and data shared between to the data bus and register files. Program rom is 32 bytes. Minimal cpu is 8 bits coefficient table is 4 bytes, multiplier 8x3 bits and output port is 8 bits.

3.1. Smallest Processor

The design of the datapath was divided into two paths, one path for the data and other for the address. The address generation is done with a distinct PC incrementer, which increases the PC by one after the opcode execution except the branch instruction. The execution of branch opcode JB S is dependent on external bit Condition: jumps to the address loaded from 'Data In' (or) remains at the same location. LSB 6 bits are address bit and MSB 2 bits are op-code. Initial state is idle; states are moves to the next state depend on their op-codes. At the idle state data is zero in address and accumulator. If state is load address is Lsb 6 bit data is

$$\text{Acc} = \text{mem} [\text{AAAAAA}]$$

Address is moves to the accumulator from the memory. Wenable and Oenable are high. next state will be addle which is increment by one. If state is load address is lsb 6 bit data is

$$\text{Acc}= \text{acc}+ \text{mem} [\text{AAAAA}]$$

Accumulator is add previous data and present data. Wenable and Oenable are high and next state will be idle which increment is by one. If state is load address is lsb 6 bit data is

$$\text{mem} [\text{AAAAAA}] = \text{acc}$$
Wenable and Oenable are low and next state will be idle which increment is by one. If state is jump address is lsb 6 bit data is Wenable and Oenable are low. Next state will be idle which increment is by one. Data is jump to the address bit from the data in.

4. Programming model

This method uses one byte per instruction format; it is encoded with a two-bit op-code, a 6-bit address/immediate field and enables simpler CPU architecture and bigger code density.

4.1 Instruction set

The four encodable instructions are listed in table I. These instructions were modified from the TB02CPU2 CPU design. However, there is no carry bit generation by the unsigned addition of accumulator with memory content, therefore it has to be checked so that no overflow calculation will occur.

External multiplier uses a special STM instruction processes as the multiplication operation of the FIR. The conditional branch instruction JBS, which checks an external bit condition to decide whether it should stay or jump to specified address, is interesting; and enables the synchronous program execution with external control signal.

The total numbers of instructions for processing 4-tap FIR filter is 23, which will be carried out in 45 clocks for one sample signal[3,4,5]. The input register and output register are clocked at the same time (at address 00) that saves one instruction time. To save the requirement of a data pointer and its management, the input signal is not stored in a ring buffer as in conventional method; instead, the value of x(n-i) will be stored into x(n-i-1) after the readout.

5. Simulation

Simulation is carried out with a simple 4-tap FIR filter built with the designed processor and implemented with sequential method saving hardware resources. The input and output signals to and from the filter are the unsigned 8-bit x [7-0] and the unsigned 8-bit y [7-0].

This filter implements the following FIR equation:

\[ y(n) = x(n)h(n) + x(n-1)h(n-1) + x(n-2)h(n-2) + x(n-3)h(n-3) \]  \( \ldots (1) \)

Where

\[ h(n) = \frac{7}{16}, \]
\[ h(n-1) = \frac{5}{16}, \]
\[ h(n-2) = \frac{3}{16}, \]
\[ h(n-3) = \frac{1}{16} \] .

This equation can be rewritten as:

\[ y(n) = \frac{1}{16} \times 7x(n) + \frac{1}{16} \times 5x(n-1) + \frac{1}{16} \times 3x(n-2) + \frac{1}{16} \times x(n-3) \ldots (2) \]

\[ y(n) = \frac{1}{16} \times (7x(n) + 5x(n-1) + 3x(n-2) + x(n-3)) \ldots (3) \]

The simulation was first done by MATLAB software using equation 2 with floating point arithmetic; this executes the simulation without loosing precisions. Then, simulation with the designed FIR processor on FPGA using equation 1 and was done with fixed-point arithmetic. The equation 2 sums all multiply products then divided by 16, this will make an overflow calculation on the 8 bits ALU[9], therefore each multiply product should divided by 16 (i.e. shifts 4

5.1. Memory map

Three different blocks are used in memory map

- ROM
- RAM
- COEFFICIENT
5.2. ROM

ROM is a read only memory which contain the address bit of 00000000-11111111 base on this memory ROM is enable.

RAM is a random access which contain the address bit of 00100000-00101111 base on this memory ROM is enable.

5.4. COEFFICIENT

Which contain the address bit of 01100000-01100110 base on this memory ROM is enable. These are access base on their address bits.

6. Simulation Results

Data will be taken from the address location in memory map and feed back to the cpu memory to cpu .Memory map contain three different blocks RAM, ROM, COEFFICIENT. address decoder has 6-bit register, enable, utenable, clock, 8-bit data input. depends on address bit blocks are enable to the address decoder. Memory map having X(n) input registers, Y(n) output registers which are access by cpu [9]. To get the multiplier multi-tri state buffer input connect to the output of cpu and output of tri state buffer in connect to the memory map input. Cpu tri-state buffer input is connect to the memory map out put and output is connect as feedback to input of the CPU.

| TABLE II |
| SIMULATION SETUP |
| Implementation | LC Combination | LC Register |
| V1.1 | 145 | 71 |
| V1.2 | 123 | 71 |
| V2.0 | 77 | 57 |
| V2.0 | 140 | 96 |

| TABLE III |
| SIMULATION RESULT |
| Implementation | 9bit DSP block | Total logic elements | Fmax(MHz) |
| V1.1 | 0 | 171 | 79.62 |
| V1.2 | 1 | 141 | 84.32 |
| V2.0 | 0 | 102 | 123.46 |
| V2.0 | 0 | 140 | 98.36 |

From Table II. and Table III. the fmax of the implemented hardware is about 80 MHz that means the input signal can reach up to about 1.7 MHz beyond this limit, more memory or a more advanced structure is needed, this can be solved easily in a FPGA chip.

7. Conclusion

In this paper we facilitate the optimal approach for chip implementation on SoPC systems[8], which offers some advantages over conventional methods. Such as, It can be synthesized and fitted to a small FPGA/CPLD; many identical units can be integrated into one chip for parallel processing, It has the flexibility of easy changing values of coefficients, easy extension to n-tap FIR filter, and easy expansion to outside word using internal tri-state I/O bus, and It can work with high input signal frequency; the fmax of the implemented hardware is about 80 MHz that means the input signal can reach up to about 1.7 MHz beyond this limit, more memory or a more advanced structure is needed, this can be solved easily in a FPGA chip. In implementation, a real band pass FIR filter of 10 coefficients in two processors were developed for voice signal processing with a frequency response from 300Hz to 3.4KHz.

ACKNOLEDGEMENT

This work was fully supported by the National Institute of Technology, Tiruchirappalli. Besides, the authors would thank Faculty of Science and Technology, IFHE University, Hyderabad.

REFERENCES


