Software and Hardware Implementation of IIR Based on Matlab&Acceldsp

Chen Hongyan
Department of Computer Science and Technology,
Tongji University,
Shanghai, China
eller2009@163.com

Wang Lisheng
Department of ComputerScience and Technology,
Tongji University,
Shanghai, China
lishwang@tongji.edu.cn

Abstract-**Since the IIR design is very difficult and time-consuming by using the hardware description language directly, this paper presents a simpler and more efficient method for IIR digital filter design. This design is based on Batterworth analog filter, using the impulse response invariant method. The design steps are: design non-quantitative IIR filter using Matlab; convert the designed IIR filter of M language into hardware description language using AccelDSP; then verify and synthesize the IIR design using ISE 10.1 of Xilinx. The design is verified by timing simulation and can be implemented on FPGA directly. The key of this design is to analyze the IIR digital filter algorithm, then describe it through the synthesizable m language and finally convert the m program to RTL design by using AccelDSP. The aim of this paper is to simplify the IIR digital filter design.**

Keywords-IIR digital filter, Butterworth analog filter, AccelDSP, Matlab, ISE.

I. INTRODUCTION

In digital signal processing, digital filtering occupies an extremely important position. Digital filtering is a basic processing algorithm in the application of the voice and image processing, pattern recognition and spectral analysis. There are many advantages in many signal processing applications by using digital filters instead of analog filters. It is easy to realize different magnitude and phase frequency characteristics by digital filter, overcoming the voltage drift, temperature drift and noise problems which are the performance-related with analog filters [1].

Digital filter is divided into infinite impulse response filter (IIR) and finite impulse response filter (FIR). Compared with the FIR, IIR can use a lower order to obtain high selectivity with small storage units used and the signals delay small. The design of IIR can use the results and design ideas of analog filter design. The workload of IIR design is relatively smaller than FIR. In the same of gate level and in the same clock speed, IIR can provide a better band attenuation than FIR. In short, the design of IIR is more economic and effective. Under the normal circumstance, n-order IIR has the same performance with the 2n-order FIR. Practically, 20-order Butterworth IIR filter can achieve approximate ideal linear phase [1] [2].

IIR unit response is infinite, which is the same with the analog filter. So, the key of IIR design is converting H(s) to H(z), namely the discretization of analog filter. This paper presents the design of IIR digital filter by using the design idea of Butterworth analog filter, synthesizing the algorithm described in M language into the algorithm written in hardware description language [1].

II. FUNDAMENTAL OF IIR DIGITAL FILTER

Digital filter is a discrete time systems of filtering function, discrete systems can be divided into types, recursive and non-recursive. Generally, IIR is recursive, while the FIR is non-recursive. Since the input and output signals of digital filter are both discrete-time signal and sequence, the concepts, methods and conclusions of discrete-time systems are all applicable to digital filters.

IIR is a recursive causal linear time-invariant system, the differential equation of IIR digital filter is as follows:

$$y(n) = \sum_{i=0}^{M} b_i x(n-i) + \sum_{j=1}^{N} a_j y(n-j)$$

Where N (>=1) represents the order of the filter, and the ai and bi terms represent the coefficients of the IIR digital filter, which consequently determine the filter characteristics [10]. From the up equation, we can know the output of IIR is equal to the linear combination of each delayed input and output signal. So the IIR is a feedback system. Generally, IIR system can be described by the basic devices: adders, multipliers and delay units. The typical structure of IIR can be divided into three types: direct, cascade and parallel [1].

The Z transfer function of the differential equation is as described below:

$$Y(z) = \sum_{i=0}^{M} b_i z^{-i} X(z) + \sum_{j=1}^{N} a_j z^{-j} Y(z)$$

Then, the system function of IIR can be given by:

$$H(z) = Y(z)/X(z) = \left(\sum_{i=0}^{M} b_i z^{-i}\right) / \left(1 - \sum_{j=1}^{N} a_j z^{-j}\right)$$

There are two Implementation methods of digital filter, software implementation and hardware implementation. This paper firstly realize the IIR by using software simulation Matlab, then convert high-level language programs to

hardware description language programs through synthesis tool AccelDSP. This design can improve the efficiency of digital filter design and narrow the difference between the software and hardware programs [2].

This IIR design is based on the theory of Butterworth analogy filter. With maximal flat amplitude characteristics in the passband, Butterworth analog filter decreases monotonically with the frequency increasing in the positive frequency range. The amplitude square function is as follows: $|Ha|^2=1/(1+(j\Omega/j\Omega c)2N)$,where N is the order of Butterworth analog filter. The N is greater, the approximation of stopband and passband is the better and transition belt is steeper [1].

The IIR design steps based on Butterworth analogy filter are as follows:

a) Translate the required characteristic frequency parameters of the IIR into the parameters of a low-pass analog filter;

b) Obtain the transfer function Hp(z) of low-pass analog filter through the analog approximation method;

c) Obtain the corresponding digital low-pass system function Hp(z) by Hp(s), through the mapping relation of s plane and z plane;

d) Obtain the system function H (z) of IIR by Hp(z), through the frequency transformation in the digital domain.

In this paper, IIR digital filter is designed based on the analogy filter through mapping the analogy filter to the digital filter. As a "sample" of the digital filter, the analogy filter should meet the performance requirements of the digital filter. The main mapping methods are: Impulse response invariant method; Bilinear transform invariant method and Step response method. In this design, the Impulse response invariant method is used [1] [9].

Impulse response invariant method is to make the unit impulse response sequence h(n) of the digital filter imitate the unit impulse response of the analog filter h(t). h(t) is sampled with the equal intervals, making h(n)=h(nT),where T is the sampling cycle.

H(s) is the Laplace transform of h(t) and H(z) is the Z transform of h(n).The relation of H(z) and H(s) is as follows:

$$H(z)|_{z=e^{kT}} = (1/T)\sum_{k=-\infty}^{\infty} H(s - j(2\pi/T)k)$$

From the above formula, we can know Impulse response invariant method is transforming the s plane of analog filters into z plane of digital filters.

III. SOFTWARE DESIGN OF IIR DIGITAL FILTER

There are many methods for designing the IIR, such as the following types:

a) Through mapping the analogy to digital based on the sophisticated analog filters (Butterworth, chebyshev and Elliptic [1]);

b) Through zero-pole trial and error method or through amplitude squared error minimization in the frequency domain or time domain method;

c) Computer-aided design method.

This paper design IIR utilizing the Impulse response invariant method. Firstly, excogitate a Butterworth analog filter prototype equivalent to IIR; secondly, map the Butterworth analog filter to the IIR. Since there are a lot of simple and ready-made formulas for analog filter and design parameters have been tabulated, the software design IIR digital filter is becoming simpler and more accurate by using this method.

Matlab Signal Processing Toolbox provides the functions about the Butterworth filter design, such as buttap, buttord, butter. Function [z,p,k]=buttap(n) can design the n-order normalized Butterworth analog low-pass filter prototype that can also be designed by the Matlab FDAToolg [2]. According to the following procedures, the IIR design can be completed. The software design procedures of IIR are presented in Figure 1.

The software design procedures of IIR are as follows:

a) Confirm every digital low-pass filter specification(passband cutoff frequency ωp, passband attenuation αp, stopband cutoff frequency ωs, stopband attenuation αs);

b) Transform the digital low-pass filter specifications into the analog low-pass filter specifications. Design the Butterworth low-pass filter prototype by using Matlab FDAToolg and functions;

c) Obtain the system function H(z) by transforming the s plane of analog filters into z plane of digital filters , confirming the filter coefficients {an} and {bn} of IIR.

The following is the core code of the IIR digital filter design which is transformed by the Butterworth analog low-pass prototype.

```
%Design the Butterworth analog low-
pass prototype
Function[A,B]=butter_iir(wp,ws,Rp,As)
 [n,wn]=buttord(wp,ws,Rp,As,'s');
 [z,p,k]=buttap(n);
 [b,a]=zp2tf(z,p,k);
      [b1,a1]=lp2lp(b,a,wn);
%Obtain the filter coefficients {az} and
{bz} of IIR.
[bz,az]=impinvar(b1,a1);
```

Toward running the upward code on Matlab, the coefficients {az} and {bz} of IIR digital filter can be received. The coefficients {az} and {bz} are the key of the next design procedure, which are used for the convolution with the input signals. Through the Convolution transform, the output signal of IIR digit filter can be acquired.

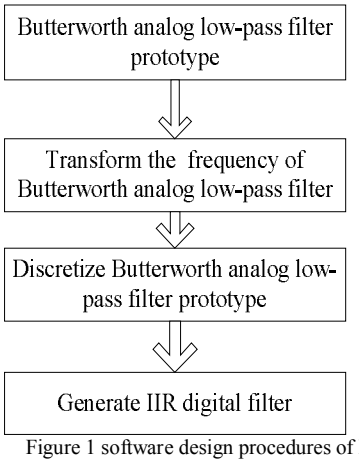

Figure 1 software design procedures of IIR

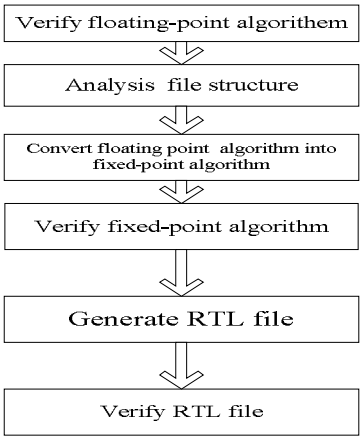

Figure 2 Design processes of AccelDSP

IV. HARDWARE MODULAR OF IIR DIGITAL FILTER

The better implementation way of digital signal processing algorithms is describing the algorithms by using high-level language. And then convert the high-level language into the hardware description language (Verilog or VHDL) automatically. This method can improve the efficiency of design development. This way can be realized through the two tools: AccelDSP Synthesis Tool and System Generator of Xilinx. AccelDSP Synthesis Tool can transform the M language into hardware description language (Verilog or VHDL); while the System Generation Tool is based on the Simulink. The module designed by the System Generation Tool can be used to construct larger systems that can be converted into the actual circuit by using the Xilinx ISE [6].

The M programs under AccelDSP Synthesis Tool are named the "synthesized" M programs, which should observe the following rules so as to convert the M programs into the hardware circuits [3]. The M programs of AccelDSP programming rules are including:

a) Every design should contain a top-level M script and at least a implementation function;

b) The implementation functions should be contained in an independent M file and called by the M script;

c) The M script must contain loop functions where the implementation functions are be called, imitating the data flow of the real hardware.

Although the Matlab Signal Processing Toolbox provides the functions about the IIR digital filter design, but these functions can't be synthesized. So the M script and M file in this IIR design must be rewritten in this IIR design [3].

The main work of this paper is to overcome the difficulty of IIR design by using the hardware description language directly. According to the M programs rules of AccelDSP, at least an M script and an M file for calling by the M script should been programmed in this design. The Design processes of AccelDSP are showed in Figure 2 [5].

AccelDSP synthesis tool is used to transform a MATLAB floating-point design into a hardware module that can be implemented in a Xilinx FPGA. The AccelDSP Synthesis Tool features an easy-to-use Graphical User Interface that controls an integrated environment with other design tools such as MATLAB, Xilinx ISE tools, and other industry-standard HDL simulators and logic synthesizers [6].

In this paper, hardware modular of IIR digital filter is implemented by AccelDSP. While AccelDSP Synthesis Tool is started, it will connect the Matlab automatically. AccelDSP Synthesis Tool can not only transform M programs into the hardware design automatically, but also convert M programs into the modules of System Generator, so as to establish a greater system design.

The specific operation steps of AccelDSP are as follows:

a) Reads and analyzes the Matlab floating-point design;

b) Automatically creates an equivalent MATLAB fixed-point design;

c) Invokes a Matlab simulation to verify the fixed-point design;

d) Provides you with the power to quickly explore design trade-offs of algorithms that are optimized for the target FPGA architectures;

e) Creates a synthesizable RTL HDL model and its corresponding testbench to ensure bit-true, cycle accurate design verification;

f) Provides scripts that invoke and control down-stream tools such as HDL simulators, RTL logic synthesizers, and Xilinx ISE implementation tools [6].

The input signal of this IIR design is random noise coupled with a sine function, the core code of the IIR input signal design is shown below:

```
%Define the constants of the input signals
MM = 200; Fs= 40000;SS= 1000;
% Define the sine function
data = 5 * sin( 2 * pi * [1:MM] / (Fs/SS))+2;
% Define the noise
rand('state',0);
noise = 2*(rand(1,MM)-0.5);
% the input signals can be received
indata = data + noise
```

Through the above code we can get the IIR input signals, the IIR output signals can be obtained through calling the m file-iir function in the loop function of the m script-iir_script. The iir function realizes that the output signal can be received by entering the input signal into the IIR digital filter. The core code of iir function is shown below.

```
function [y] = iir(indata)
%define the coefficients {an} and {bn} of IIR digital filter
coeff_a=[1.0000 0.2635 0.1241 -0.0215 0.0026];
coeff_b=[0.0000 0.7685 0.5876 0.0428];
p = length(coeff_a);
q = length(coeff_b);
% define and initialize the variable quantities
      persistent x_n, y_n;
if isempty(x_n)
    x_n = zeros(q,1);
    y_n = zeros(p,1);
     end
% Obtain the output signal of IIR
    x_n(2:q) = x_n(1:q-1);
    x_n(1) = sample;
    y = coeff_b*x_n - coeff_a*y_n;
    y_n(2:p) = y_n(1:p-1);
y_n(1) = y;
```

To imitate the flow of data in the actual circuit, there should be a script function programed. The iir function should be called in the loop function of iir_script. The core code of iir_script is shown below.

```
% define and initialize the constants
MM = 200; Fs= 40000; SS= 1000;
% define the input signals of IIR
data =5 * sin( 2 * pi * [1:MM] / (Fs/SS));
rand('state',0);
noise = 2*(rand(1,MM)-0.5);
indata = data + noise;
% call iir function for every input signal
for n = 1:MM
  outdata(n) = iir(indata(n));
end

% Draw input response
figure(1);subplot(2,1,1);plot(data);
axis([1 MM -6 6]);
title(['Input = ',num2str(SS),' Hz']);
subplot(2,1,2);plot(noise);axis([1 MM -6 6]);
title('Noise');
% Draw output response
figure(2);subplot(2,1,1);plot(indata);
axis([1 MM -6 6]);
title('Combined Input');
subplot(2,1,2);plot(outdata);
axis([1 MM -6 6]);
  title('Filtered Output');
```

The hardware modular of IIR digital filter can be completed according to the AccelDSP operation flow, converting the float-point algorithm into the gate-level circuit. The concrete steps are as follows:

a) New a project in AccelDSP and load iir function and iir_script under this new project;

b) According the operation flow of AccelDSP, complete the following steps: verify float-point programs; analyse the struct of the files; quantification; transform float-point into fixed-point; verify the fixed-pointed programs; generate the RTL files;

c) There are two kinds hardware description languages of RTL files generated with the testbench files of each generated RTL file. The generated RTL files can be simulated by using the modelsim or the ISE simulator.

The AccelDSP tool in this paper is served as a link of the hardware design and software design of IIR digital filter, narrowing the differences between them. The M language programs can be transformed into the VerilogHDL or VHDL source code using AccelDSP tool. VerilogHDL or VHDL source code can be runned on the related hardware.

AccelDSP can complete the transformation of floating-point arithmetic into fixed-point algorithm automatically. It can also implement every operation step of FPGA implementation needed, such as RTL code generation, RTL verification, gate-level verification and the hardware verification in the In-circuit. With AccelDSP analyzing the dynamic simulation results, it is easy to confirm the correct word length and calibration. Users can adjust the length of word, calibration, saturation dynamically. The AccelDSP provide the underflow and overflow of data report, fixed-point report, generating RTL report, verifying fixed-point report, verifying RTL report and so on. From these reports, the errors can be easily founded and modified quickly.

V. HARDWARE IMPLEMENTATION OF IIR DIGITAL FILTER

Although AccelDSP can complete all of operation steps which are needed in the FPGA implementation, it is needed to set the running environment of this IIR digital filter design by using ISE 10.1 of Xilinx for the verification on the specific FPGA.

The Integrated Software Environment (ISE) is the Xilinx design software suite that allows users and designers to take their design from design entry through Xilinx device programming. The ISE Project Navigator manages and processes your design through ISE design flow, as show in Figure 3 [7].

To verify the correctness of the hardware modular of IIR digital filter, the generated RTL files from AccelDSP can be debugged and runned on ISE 10.1 of Xilinx. The design and verification flow of ISE10.1 is shown in Figure 3 [4][7][8].

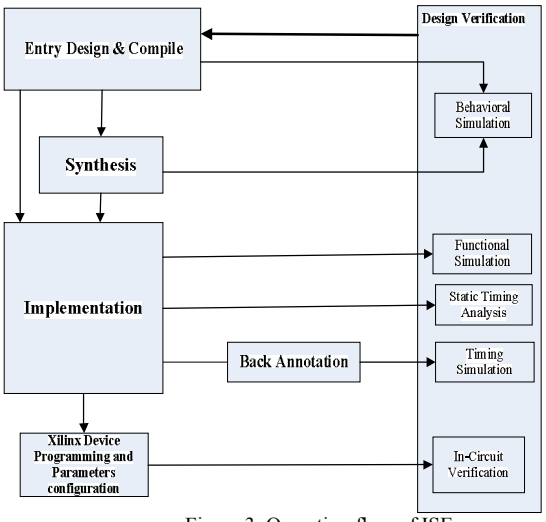

Figure 3. Operation flow of ISE

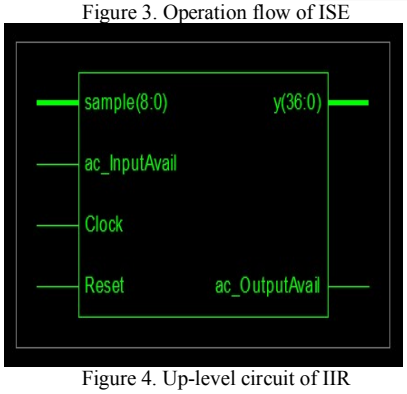

Figure 4. Up-level circuit of IIR

According to Operation flow the Fig 3, the specific steps of this design are as follows [4] [8]:

a) New a project in ISE and load the generated RTL files(VHDL or Verilog)of the AccelDSP in the new project;

b) Compile the RTL files of the IIR design and correct all of the errors;

c) Synthesis the IIR design and make behavior simulation, the up-level circuit of IIR is shown in Figure 4;

d) Implementation the IIR design through the following operations: functional simulation; static timing analysis; timing simulation;

e) Xilinx device programming and parameters configuration;

f) Generate netlist and bitstream files and download them into the FPGA;

g) Verify the In-circuit design of IIR on FPGA.

Through the above operations, the IIR digital filter design can be completed. Through the verification , it can be concluded that the In-circuit design of IIR can be runned correctly and effectively on FPGA.

VI. CONCLUSION

The aim of this paper is to raise a simple and efficient design method of the IIR filter. From the above discussion, the complete operation flow of the IIR design can be concluded. Firstly, the software design of IIR is completed by Matlab; secondly, the hardware modular of IIR digital filter is achieved by using AccelDSP; finally, hardware modular of IIR can be transformed into the hardware circuit by using ISE, and then it can be realized on FPGA. The innovation of this paper lies in designing and implementing the IIR filter with AccelDSP. This design method overcomes the difficulty of the IIR filter design with hardware description language directly. This design method has a lot advantages, such as with higher sampling frequency and real-time than the IIR design on the traditional DSP chip and with less time, less money and more flexible than the IIR design on the large ASIC.

REFERENCES

[1] Liu Hai Tang, ”Signal and Systems”[M], Xi’an: Xi'an Jiaotong University Press, July 2009, ISBN: 9787560509709.

[2] Xv Mingyuan,Liu zhenli,”MATLAB Simulation of the Application in Signal Processing”[M], Xi'an University of Electronic Science and Technology Press, Nov. 2007, ISBN: 9787560619040.

[3] Yuan Jiangnan, Tang Biyu,Chen Huihuang. “AccelDSP Based Adaptive Filter Design”[J], Journal of Xiamen University (Natural Science), Mar.2010,Vol.49,No.2.

[4] Tian Yun, Xv Wenbo.”Xilinx FPGA”[M].,Beijing: Tsinghua University Press, Nov.2008, ISBN:9787302184256.

[5] Landry, R., Jr.; Calmettes, V.; Robin, E.”High speed IIR filter for XILINX FPGA”[C]. 1998.

[6] XILINX. “AccelDSP synthesis tool (user guide) ”[0L]. Xilinx Inc., 2008.

[7] XLINX. “XLINX ISE 10.1 user guide ” [OL].Xilinx Inc.,2008.

[8] XILINX. ”Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet, 4.5 edition”[OL]. Xilinx Inc., 2005.

[9] Zhang Ke , Wu Binbin,Zhang Wei and SuhHeejong .”The application of the IIR filters based on FPGA in the DTV field”[C]. 2009.

[10] Zhenbin Gao , Xiangye Zeng , Jingyi Wang , Jianfei Liu. ”FPGA implementation of adaptive IIR filters with particle swarm optimiz ation algorithm”[C].2008