IMPLEMENTING DIGITAL FILTER USING PROGRAMMABLE LOGIC DEVICE FOR EDUCATIONAL AIMS [+]

تنفيذ مرشح رقمي باستخدام الدوائر المنطقية القابلة للبرمجة لأهداف تعليمية

Bashar Seddik Mohamad-Ali [*] **Ziyad Khalf Farej** [**]

Abstract:

Implementing digital filters mathematically, and verifying their functions, is an easy task with the help of many available CAD software such as MATLAB. But implementing hardware digital filters with fixed function IC's and verifying their function is a difficult task particularly for under graduate students in electrical and electronic engineering. However, with the emerging of Programmable Logic Devices (PLD's), and Field Programmable Gate Array (FPGA) devices, implementing hardware digital filters became an easy task. This paper shows how a programmable hardware first order filters can be implemented, synthesized, and simulated using (MAX+PLUS II, ver.10) development software. MATLAB filter design tool were used to determine the filter coefficients, and then these coefficients are fed to the PLD inputs, to get the desired filter specifications. The response of the implemented filter in the PLD device has been verified by comparing it to the response of MATLAB filter of the same type.

المستخلص:

إن تنفيذ المرشحات الرقمية والتحقق من صحتها باستخدام برامجيات التصميم بمساعدة الحاسوب (CAD) هدف سهل، لكن عملية تنفيذ هذه المرشحات باستخدام الدوائر المنطقية المتكاملة ذات الدالة الثابتة، والتحقق من صحتها هي هدف ليس بالسهل خصوصا لطلبة الدراسات الجامعية الأولية في اختصاص الهندسة الكهربائية والالكترونية. إن ظهور ألدوائر المنطقية القابلة للبرمجة (PLD) ومصفوفة البوابات ذات المجال القابل للبرمجة (FPGA) سهل على الطالب عملية تنفيذ المرشحات الرقمية. هذا البحث يوضح عملية تنفيذ وتحليل ومحاكاة مرشح رقمي من الدرجة الاولى في الدوائر المنطقية القابلة للبرمجة باستخدام برنامج التطوير (MAX+PLUS II ver.10). تم الاستعانة ببرنامج (MATLAB) لحساب معاملات المرشح وفقا للمواصفات المطلوبة، ثم تم اختيار قيم هذه المعاملات في مداخل الدائرة المنطقية القابلة للبرمجة (PLD) للحصول على المرشح بالمواصفات المطلوبة. تم التأكد من صحة مخرجات المرشح المنفذ بواسطة الدائرة المنطقية القابلة للبرمجة بمقارنتها بمخرجات نفس نوع المرشح المصمم بواسطة برنامج (MATLAB).

Introduction:

Implementing a digital filter can be either using simulation software (running on general purpose processors), or using hardware devices. The first approach can not be used in real time systems due to its slow speed, but it is flexible. While in the second approach (using the conventional hardware devices), high sampling rate (high performance) can be reached, but

[+] **Received on 4/5/2008 ,Accepted on 30/9/2009 .**
[*]**Lecturer / Technical College/ Mosul**
[**]**Assistant Lecturer/ Technical College/Mosul**

they are expensive and not flexible. The approaches to the hardware implementation of digital filtering algorithms are either general purpose digital signal processing chips for audio applications, or special purpose digital filtering chips and application-specific integrated circuits (ASICs) for higher rates [1, 2]. However, recent advances in the PLD and FPGA devices technology have made PLDs more superior, and attractive for implementing digital filters [3, 4, 5, 6]. PLD and FPGA implementation approach has both flexibility of software, and performance of ASICs. Moreover, reconfigurable PLDs have opened a new era in practical training, due to the ease of design and implementation in comparison with fixed functions devices. PLD devices can be reconfigured times and times again to implement different logic designs as required in practical training. The later merit of PLDs has encouraged authors to use PLD devices for educational aid (for example in computer architecture [7]). In this work, a Complex PLD is used to implement an Infinite Impulse Response (IIR) digital filter, for educational aid in practical training. An IIR filter circuit has been decomposed into two basic sections; the poles circuit, and the zeros circuit. Filter coefficients were determined using MATLAB digital filter design tool, and then these coefficients were fed to a Complex PLD device to get the corresponding filter characteristics. The implemented filter can be tested by applying a sampled signal, and watching the filter response. The filter response can be also compared with the response of the MATLAB designed filter.

Digital filter basics:

There are two basic types of digital filters, Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters. The general form of the digital filter equation is [8]:

$$Y(n) = \sum_{i=0}^{N} b_i X(n-i) + \sum_{i=1}^{N} a_i Y(n-i) \quad \ldots (1)$$

where y(n) is the current filter output, the y(n-i)'s are previous filter outputs, the x(n-i)'s are current or previous filter inputs, the ai's are the filter's feedback coefficients corresponding to the poles of the filter, , the bi's are the filter's feed forward coefficients corresponding to the zeros of the filter, and N is the filter's order. IIR filters have one or more nonzero feedback coefficients. That is, as a result of the feedback term, if the filter has one or more poles, once the filter has been excited with an impulse there is always an output. FIR filters have no non-zero feedback coefficient. That is, the filter has only zeros, and once it has been excited with an impulse, the output is present for only a finite (N) number of computational cycles.
A general filter transfer function can be derived as follows;

Substituting for $X(n-i) = Z^{-i} X(n)$, and $Y(n-i) = Z^{-i} Y(n)$, equation (1) can be rewritten as

$$Y(n) = \sum_{i=0}^{N} b_i Z^{-i} X(n) + \sum_{i=1}^{N} a_i Z^{-i} Y(n) \quad \ldots (2)$$

Hence;

$$Y(n)(1 - \sum_{i=1}^{N} a_i Z^{-i}) = X(n)(b_0 + \sum_{i=1}^{N} b_i Z^{-i}) \quad \ldots (3)$$

Figure 1 shows common IIR architecture, which is a direct implementation of equation (3).

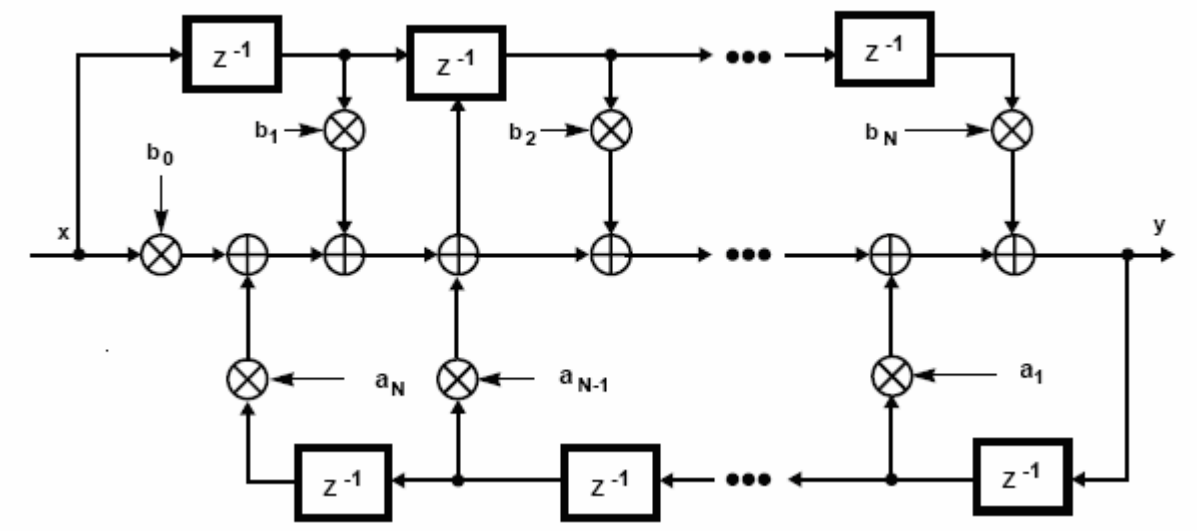

Figure 1 Common IIR filter architecture

Finally the general transfer function of an IIR filter will be;

$$H(z) = \frac{Y(n)}{X(n)} = \frac{b_0 + \sum_{i=1}^{N} b_i Z^{-i}}{1 - \sum_{i=1}^{N} a_i Z^{-i}} \quad \ldots \quad (4)$$

Implementing 1st order IIR filter:

For the purpose of this work, a first order IIR filter has been implemented on a CPLD (ATF1508AS of Atmel). Implementing a digital filter on a PLD can be either in a bit serial (pipelining) method [3, 6], or in bit-parallel method [5]. While the bit-serial method does not consume large number of resources, it requires more clock cycles to produce a single sample. However, the bit-parallel method consumes more resources, but it has high performance speed. In this work the bit-parallel method has been used since it is simpler to implement and to understand as educational aid. The general transfer function (equation 4) of 1st order IIR filter can be written as;

$$H(z) = \frac{b_0 + b_1 Z^{-1}}{1 - a_1 Z^{-1}} \quad \ldots \quad (5)$$

Multiplying both nominator and denominator by Z;

$$H(z) = \frac{b_0 Z + b_1}{Z - a_1} \quad \ldots \quad (6)$$

This equation indicates that the filter has a single pole at $Z = a_1$, and a single zero at $Z=-b_1/b_0$ in Z-plane. Figure 2 shows a direct implementation for equation (6). This IIR filter circuit can be divided into two sections for the purpose of education.

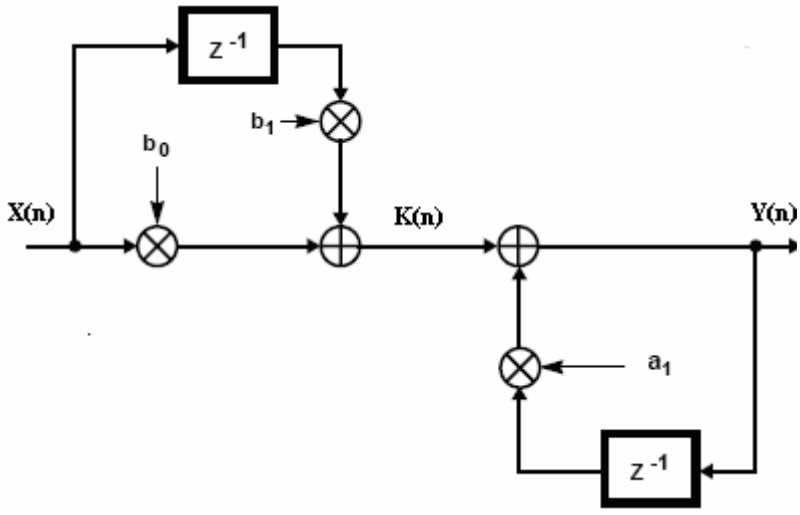

Figure 2 First order IIR filter

The first section is the zeros circuit, which is responsible for the zeros in the Z-plane, and its output is given by;

$$K(n) = X(n)(b_0 + b_1 Z^{-1}) \quad \ldots (7)$$

Hence

$$\frac{K(n)}{X(n)} = b_0 + b_1 Z^{-1} \quad \ldots (8)$$

Where X(n) is the input signal samples to the filter.
The second section is the poles circuit, which has an output given by;

$$Y(n) = a_1 Z^{-1} Y(n) + K(n) \quad \ldots (9)$$

Hence

$$\frac{Y(n)}{K(n)} = \frac{1}{1 - a_1 Z^{-1}} \quad \ldots (10)$$

Multiplying equation (8) by (10);

$$\frac{Y(n)}{X(n)} = H(z) = \frac{b_0 + b_1 Z^{-1}}{1 - a_1 Z^{-1}} \quad \ldots (11)$$

This is the same as equation (5). Consider the case where $b_0 = b_1 = b$ so that the zero in Z-plane will be at Z= -1. In this case the zeros circuit is more simplified to have only one multiplier (as shown in Figure 3).

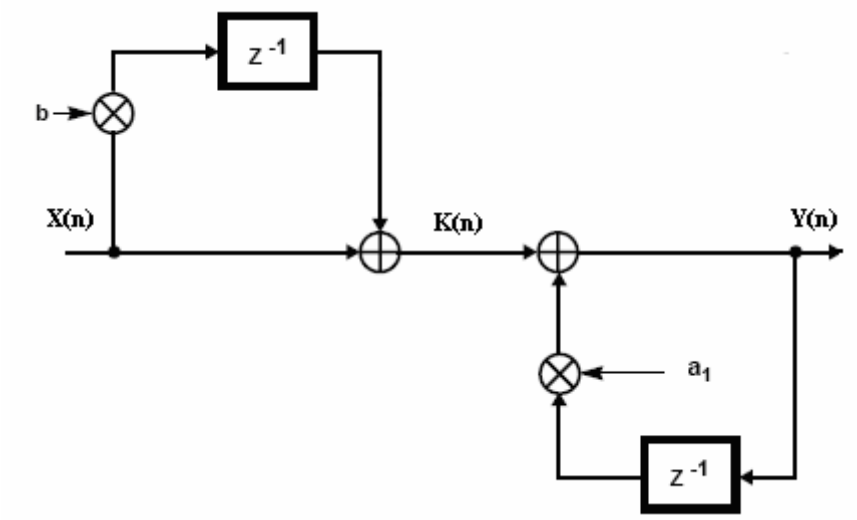

Figure 3 Simplified first order IIR filter

This simple circuit of digital filter can be used to analyze the digital filter for educational aid. Coefficients a_1, and b can be determined using MATLAB filter design tool box from the desired filter characteristics (low pass or high pass, sampling frequency, cutoff frequency, …etc). The pole and the zero positions on z-plane can be noticed by the student in the digital tool box. Then the determined coefficients can be applied to the CPLD inputs in binary form to get the desired hardware digital filter characteristics. The student can be familiar with the followings through this design cycle;

1. The effect of changing the ratio of the cutoff frequency to the sampling frequency (Fc/Fs), on (a, and b) coefficients.
2. The relation between (a, and b) coefficients with the positions of the poles and zeros on z-plane.
3. Finally the effect of the positions of poles and zeros in z-plane with the type of filter (high pass, low pass, band pass).

CPLD realization of the IIR filter :

Each section (the zeros and the poles circuits) has been implemented on a separate CPLD device, so that the output of the zeros section, and the poles section can be noticed by the student (as binary numbers). The filter circuit shown in (Fig. 3) was realized, and synthesized using (MAX+PLUS II ver. 10) software of Altera.

1. Zeros circuit realization

The zeros circuit inputs are 8-bit binary vector X[7..0] (Fig. 4), while its outputs are 8-bit binary vector K[7..0]. The coefficient (b) is a 4-bit binary positive number a[3..0] with a range of (0 to 15). Since the used multiplier does not support signed numbers, an absolute function was used to complement the negative number before multiplier, and an add/subtract circuit was used to convert the output of multiplier back to a negative number (two's complement). A second add/subtract circuit was used to add the delayed input to the present input. The latter add/subtract circuit is controlled through (AS) input, to obtain a positive or negative coefficient of the delayed input. Notice that the multiplier output is a 12-bit number bX[11..0], while only the higher 8-bit of this number is fed to the add/subtract circuit. This

arrangement is to change the range of (a) coefficient to be 16 binary weight values (from 0 to 0.9375), which correspond to (0.0000 to 0.1111) in binary.

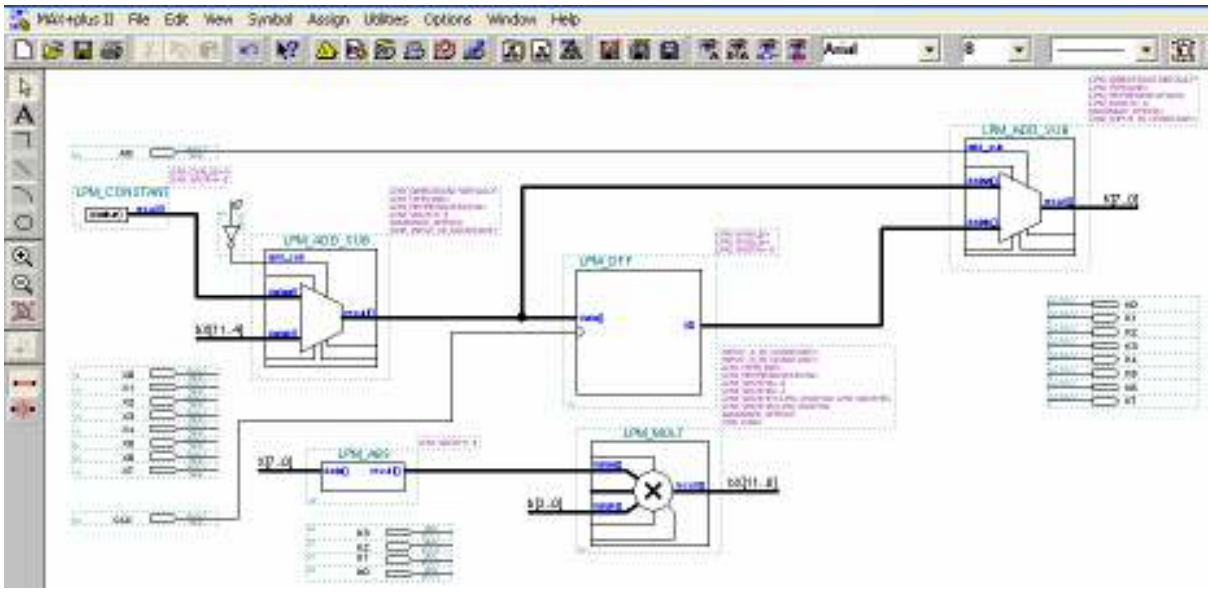

Figure 4 Schematic capture design of the zeros circuit

2. Poles circuit realization

The poles circuit inputs is an 8-bit binary vector K[7..0], which is the output of the zeros circuit (as shown in Fig. 5). The output of this circuit should be the 8-bit binary vector Y[7..0] (before the delay circuit), but the actual output of the filter is being taken from the output of the delay circuit O[7..0], so that signals fluctuation in the add⁄subtract circuit output due to signals skew is not shown in the filter output. The a_1 coefficient is a 4-bit binary positive number, which has 16 binary weight numbers in the range of (0 to 0.9375). Negative coefficient can be applied by controlling the add⁄subtract circuit to be a subtract circuit through AS2 input. An absolute circuit together with the add⁄subtract control line have been used to coupe with negative numbers which may be fed to the multiplier input.

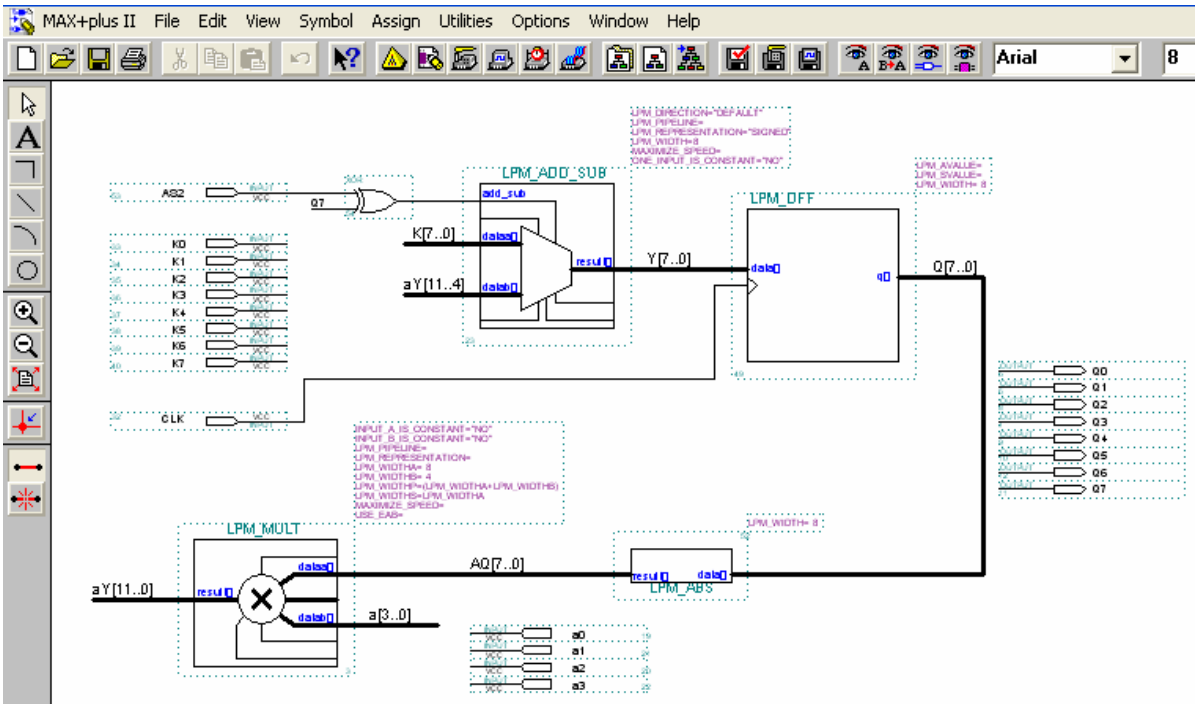

Figure 5 Schematic capture design of the poles circuit

Testing the CPLD filter:

Once the filter coefficients are determined using MATLAB filter design tool, these coefficients are rounded to the nearest binary weight number, then the equivalent binary number is applied to the CPLD coefficients inputs b[3..0], and a[3..0]. At this point the CPLD is configured to the desired filter specifications, which are selected in the MATLAB filter design tool. The CPLD filter can now accept a sampled input signals (in binary) at its inputs X[7..0], and can give the corresponding filter response in its outputs O[7..0]. The CPLD filter was tested by comparing its unit step, impulse, and sine wave responses with the corresponding responses obtained by MATLAB commands.

1. Impulse and step responses

Table 1 shows the impulse response (for five clock pulses) of butterworth low pass filter with Fc / Fs = 0.25 in MATLAB calculations, and in the designed CPLD. An error of 1.58% is shown due to the available binary coefficients of only four digits fraction. Table 2 compares the step response of the same filter in MATLAB calculations, and the designed CPLD. Nearly the same error% is obtained due to the same reason mentioned above. These responses can be easily calculated by the student. Notice that for the above filter only the zeros section of the filter is effective, since the coefficient (a = 0).

Table1 Comparison of impulse response in both MATLAB and CPLD

Clock	Normalized MATLAB Response a = 0.000094, b = 0.49995	CPLD Response a = 0.0000, b = 0.5 Impulse magnitude= 63	Normalized CPLD Response	Error %
0	0.5000	31	0.4921	1.58
1	0.5000	31	0.4921	1.58
2	0.0000	0	0	0
3	0.0000	0	0	0
4	0.0000	0	0	0

Table2 Comparison of step response in both MATLAB and CPLD

Clock	Normalized MATLAB Response a = 0.000094, b = 0.49995	CPLD Response a = 0.0000, b = 0.5 Step magnitude= 63	Normalized CPLD Response	Error %
0	0.5000	31	0.4921	1.58
1	0.9999	62	0.9841	1.58
2	1.0000	62	0.9841	1.59
3	1.0000	62	0.9841	1.59
4	1.0000	62	0.9841	1.59

2. The sine wave response

The sine wave response has been tested by comparing it to the MATLAB response of a filter which has the following specifications;

Type : low pass butterworth,
Cutoff to sampling frequency ratio Fc⁄Fs = 0.15,
MATLAB filter coefficients a = 0.3249, b = 0.3375,
CPLD filter coefficient a = b = 0.3125 (nearest binary weight) = 0.0101 (in binary).

The maximum sample magnitude of the sine wave which is applied to the CPLD, is taken as 3F (in hexadecimal) = 63 (in decimal). The time responses of a sine wave at three different frequencies (0.025*Fs, 0.154*Fs, 0.25*Fs) have been calculated in MATLAB filter design tool, then they are compared to the simulated CPLD response. Figure 6 shows the simulated CPLD response of (41) samples of one cycle sine wave, which gives a frequency of (1⁄40)*Fs = 0.025*Fs. Figure 7 shows a comparison between MATLAB, and CPLD responses at this frequency . Figures 8, and 9 show the simulated CPLD response and a comparison between MATLAB, and CPLD responses of sine waves at 0.154*Fs, and 0.25*Fs respectively. It is clear from the mentioned figures that there is always an error between the MATLAB and CPLD responses. This error is due to the rounding of the MATLAB coefficients to the nearest binary weighted numbers in CPLD. Also it is clear that as the frequency of the input sine wave is increased, the response of the low pass filter get smaller in magnitude.

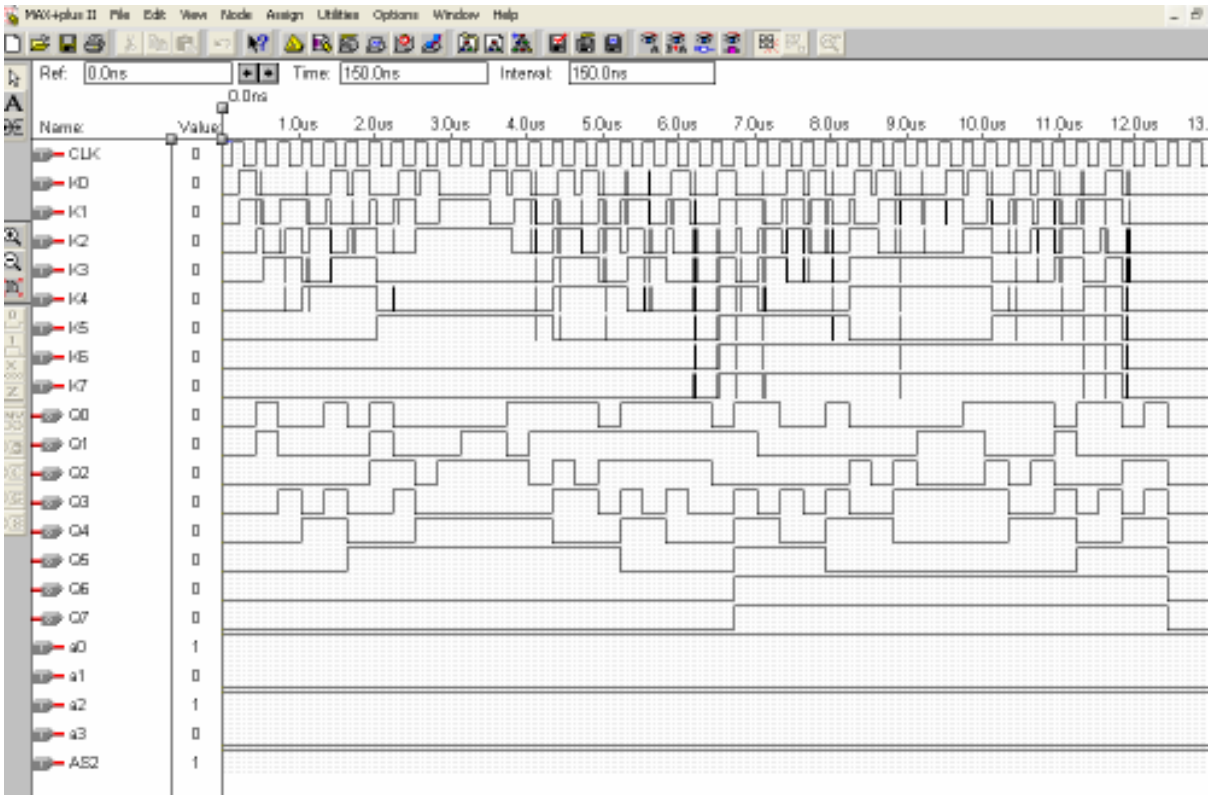

Figure 6 CPLD simulated low pass filter response of sine wave at frequency = 0.025 Fs

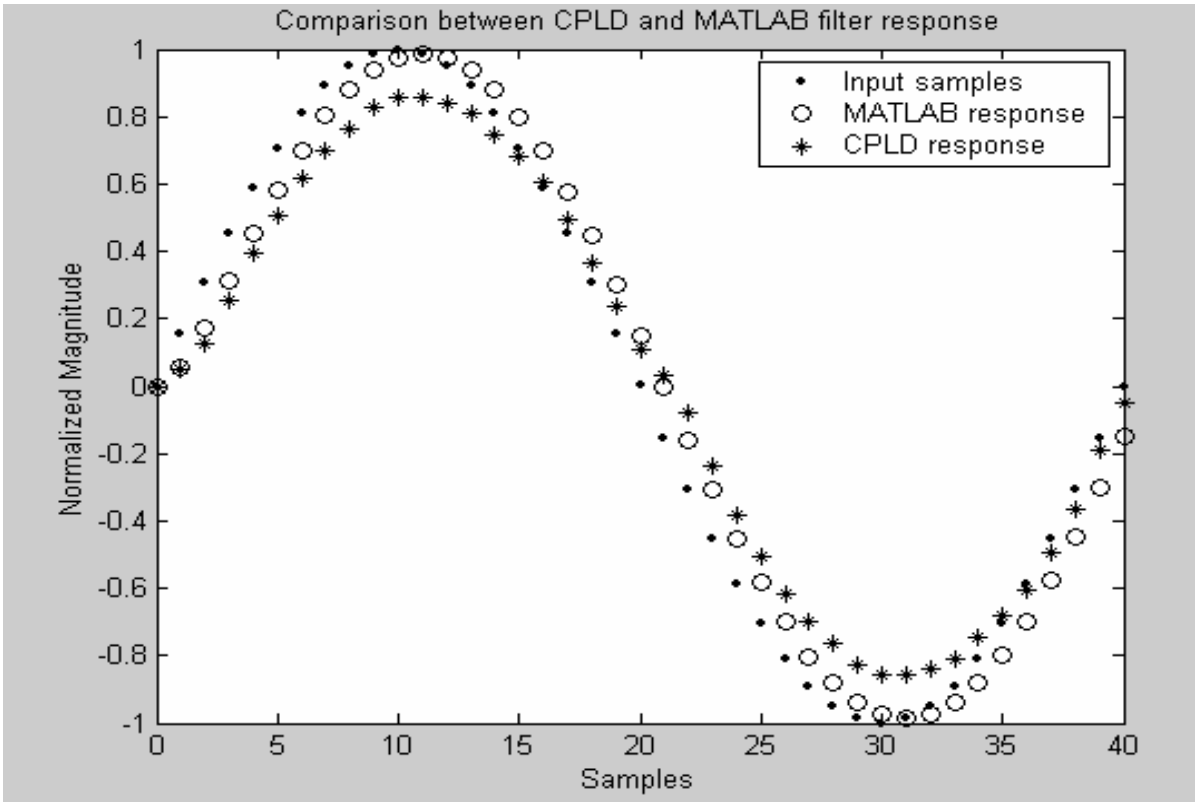

Figure 7 CPLD and MATLAB low pass Filter response of sine wave at frequency = 0.025Fs

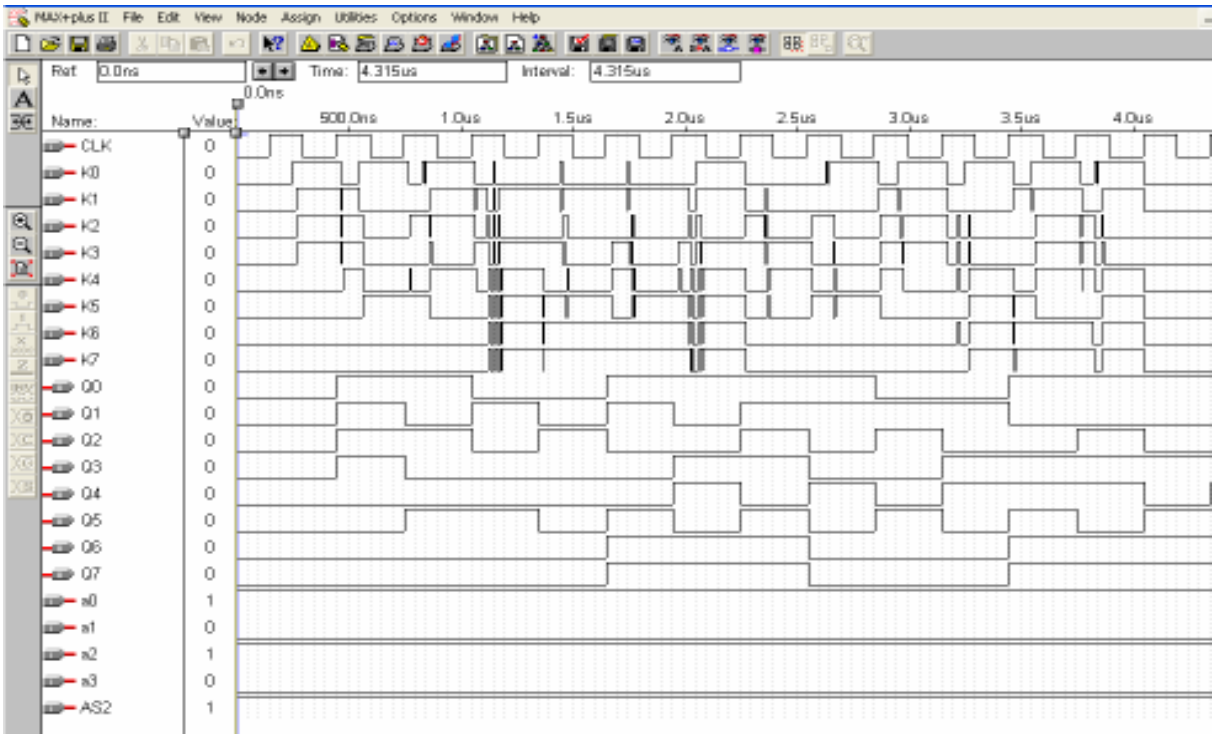

Figure 8 (a) CPLD simulated response at frequency = 0.154Fs

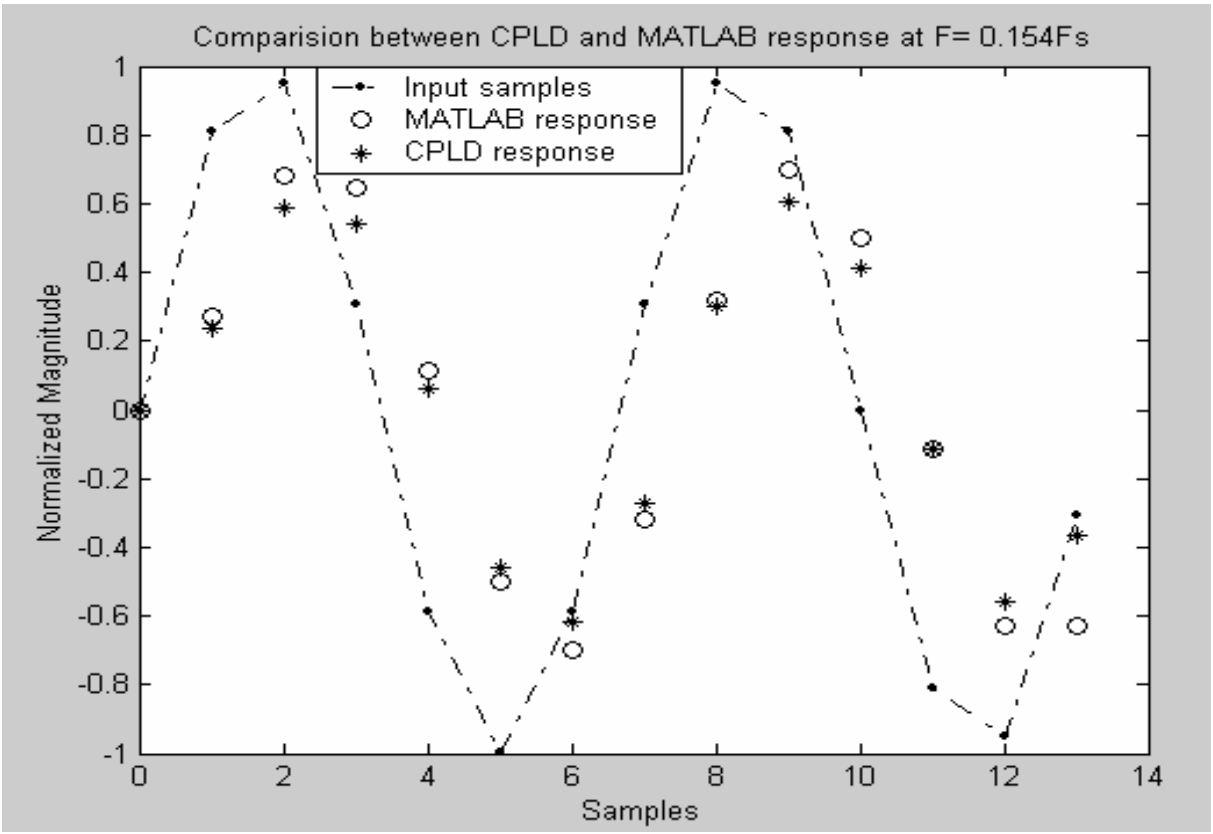

Figure 8 (b) CPLD, and MATLAB low pass Filter response of sine wave at frequency = 0.154Fs

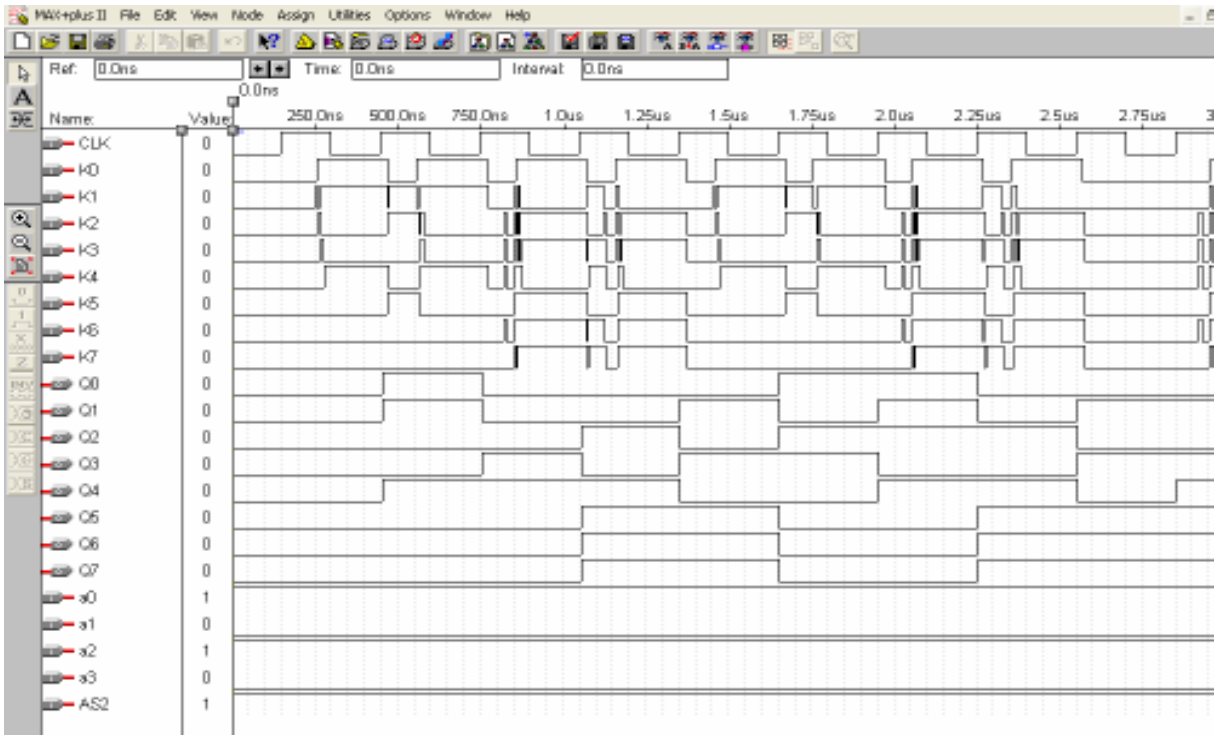

Figure 9 (a) CPLD simulated response at frequency = 0.25Fs

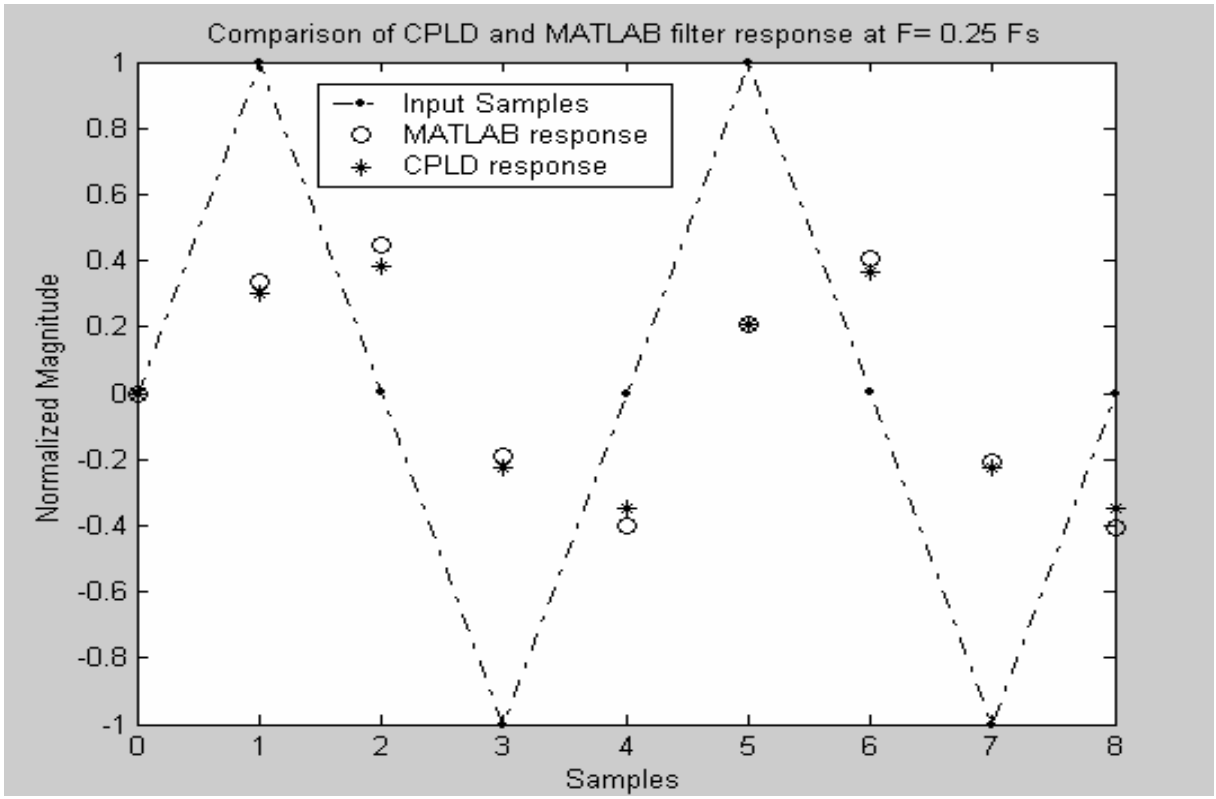

Figure 9 (b) CPLD, and MATLAB low pass Filter response of sine wave at frequency = 0.25Fs

Conclusions:

A CPLD-based 1st order filter has been implemented to help students to understand, and to be familiar with the design, and the function of digital filters. This can be achieved through practicing in the calculation of filter coefficients for certain specifications, and then applying these coefficients to the CPLD design to get the desired filter specifications. The student can also practice calculating the output of the designed filter, which is an accumulated number from the present sample and the previous feedback sample. Highest order filter can be designed using more advanced PLD (FPGA).

References:

1- R. Hartley, P. Corbett, P. Jacob, and S. Karr. "A high speed FIR filter designed by compiler" *IEEE Cust. IC Conf.*, California, pp. 20.2.1–20.2.4, May 1989.

2- K.-Y. Khoo, A. Kwentus, and A. N. Willson, Jr., "An efficient 175MHz programmable FIR digital filter", *IEEE Int. Symp. Circuits and Syst.*, Chicago, pp. 72–75, May 1993.

3-Chi-Jui Chou, Satish Mohanakrishnan, Joseph B. Evans, "FPGA Implementation of Digital Filters", *Proc. Of the fourth Inter. Conf. on signal processing and Technology*, California, pp. 80-88, 1993.

4-Igor Dzukleski, and Cesar Otega-Sanchez, "Reconfigurable FIR Filter in FPGA", *Proc. of Postgraduate Electrical Engineering and computing Symp.*, Perth, Western Australia, pp.149-152, Nov. 2006.

5-Knut Arne Vinger, and Jim Torresen Implementing, "Evolution of FIR-Filters Efficiently in an FPGA", *Proc. of NASA/DoD Conf. on Evolvable Hardware*, Chicago, pp. 26-29, July 2003.

6-Atmel application note, "Implementing Bit-Serial Digital Filters in AT6000 FPGAs", www.atmel.com .

7-M. Holland, J. Harris, S. Hauck, "Harnessing FPGAs for Computer Architecture Education", *Proc. of IEEE Inter. Conf. on microelectronic System Education*, California, pp.12-13, June 2003.

8- B.A. Shenoi, *Introduction to digital signal processing and filter design,* John Wiley & Sons Inc., Newjersey, 2006.