Implementation of the Multiplierless Parallel Multiprocessor Interpolating Filter

M. Ashrafand* and R. Mahmud**

* Associate Professor, National University of Sciences and Technology (NUST), College of EME, Peshawar Road Rawalpindi, Pakistan, e-mail: mashraf@ceme.edu.pk, e-mail: ashrafa_2000@yahoo.com

** *Assistant Professor, National University of Sciences and Technology (NUST), PNEC, PNS JAUHAR, Karachi, Pakistan,* email: riaz@pnec.edu.pk, email: riaz71@hotmail.com

***Abstract* — A fast processing approach is presented to convert the digital data gathered at lower sampling-rate into high sampling rate by utilizing the sampling rate changes algorithms using interpolation. A high speed low complexity interpolation filter is implemented at a very low hardware cost using the polyphase form, distributed arithmetic and lookup-tables. The proposed algorithm is used to processes the input data in parallel using a few shift and add operations, which make it very fast. The method allows for the multiplication of the original information stored during digitization of the analog signal at lower computational cost. Therefore, it is an interesting alternative for more sophisticated methods of performance enhancement of sampled analogue signals.**

***Index Terms* — Interpolation; reconstruction; lookup table, shift and add operations, multiplierless**

I. Introduction

Sampling rate changes are useful in many applications, such as interconnecting digital processing systems operating at different rates. High sampling rate helps to alleviate the need for high-quality analog post-filter required at the output of the staircase Digital to Analog Converter (DAC) for the reconstruction of signals. It is used to enhance the quality of the output for simulation, monitoring and control.

The paper presents a simple, very fast approach to convert the digital data gathered at lower sampling-rate into high sampling rate by utilizing the sampling rate changes algorithms.

Polyphase decomposition of a sequence by multiplier free interpolator using minimax and least squares approaches are already presented [1]. Poly phase methods are also used to save power dissipation and hardware complexity [2]. A VLSI implementation of the sample rate converter algorithm has already been presented in which filter coefficients for the multiply accumulate engine are generated using coefficient-interpolation block [3].

Tapped delay lines are used in conventional FIR filters to create the Z^{-1} (delay) terms in the Z-transform. These delay lines are implemented using a one-dimensional array or FIFO in DSP memory. The proposed method uses the delay lines but the combination of parallel processing and use of lookup tables make the processing very fast by lowering the power consumption [4].

Interpolation technique is utilized to place the additional samples between the known basic set of samples (BSS). The process of calculation of the samples can be improved by means of using a bank of polyphase subfilters. Using these, interpolation is performed at reduced computational cost as compared with the cost of the direct form of the interpolation filter. The computational cost can be reduced substantially because each of the subfilters can work independently, thus it is possible to use them like a bank of parallel working filters. The problem of noncausality of the interpolation filter is solved by L×M samples delaying in processing.

A FIR filter using the transversal computation structure [5] adopts a polyphase structure to effectively pipeline the input data streams across a register chain prior to performing the main filtering operation. Despite the simplicity of the structure, it requires a prohibitively large number of registers and incurs area overhead due to the added complexity involved with the pipeline structure [6]. An alternative low cost FIR filter structure suitable for high speed filtering is presented.

To reconstruct the analog signal from sampled values, DAC is used to generate an analogue staircase output. These rectangular pulses are sent through a lowpass filter to finally reconstruct the original signal. Theoretically, the sampling of the original signal, followed by reconstruction using DAC and ideal lowpass filter will perfectly reconstruct the original signal. There will in practice be small errors because it is impossible to construct perfect filters however, it is possible to obtain a reconstructed signal with a very small error.

The filter can be used to identify some of the samples corrupted with noise provided that the signal is oversampled and it contains redundant information. Hence, by calculating each sample value using the neighboring samples and comparing the actual value with the calculated value. The affected sampled can be replaced with the calculated samples (by means of interpolation based on neighboring samples), one can achieve good restoration results. Using this interpolator one can reduce the time of detection and correction of the affected samples

As the coefficients of the interpolation filter are constant and already known we can implement it at a low hardware cost using bit-plane-structures, lookup-table multipliers (LUTMULT) or distributed arithmetic (DA)[7] instead of conventional hardware multipliers.

II. SAMPLING AND RECONSTRUCTION

A Effect Of Sampling

In sampling process high frequency components are generated which appear in periodical fashion that is every frequency component of the original signal is periodically replaced over entire frequency axis [8]. The frequency spectrum of a continuous analog signal is shown in Fig 1.a while ideal sampler is shown in Fig-1.b.

B Signal Reconstruction

Since spectrum of sampled signal consists of baseband spectrum and spectral *images* shifted at multiples of F_s, reconstruction means isolating the baseband image as shown in Fig-1.c. To reconstruct the sampled values, DAC generates an analogue staircase output waveform. The reconstructor does not completely eliminate the replicated spectral images. An additional lowpass post filter, called an anti-image post filter, may be used to remove the surviving spectral replicas [9].

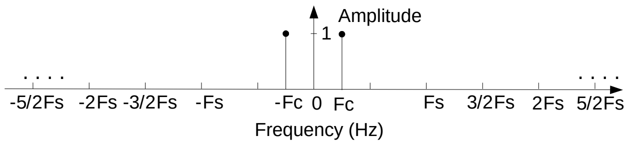


Figure 1.a. The frequency spectrum of continuous signal

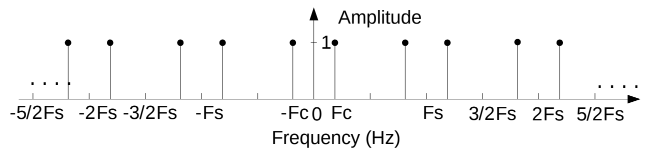


Figure 1.b. The frequency spectrum of ideal sampler output

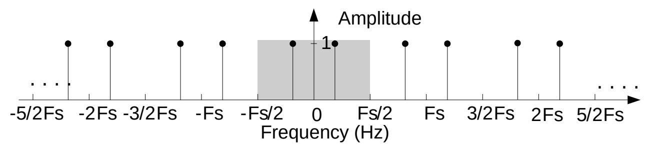


Figure 1.c. Isolation of baseband using Ideal Lowpass filter

C Interpolation

It is used to increase sampling rate by inserting additional samples of the signal between the original ones. An FIR digital filter calculates the inserted samples. The process of increasing the sampling rate by the factor of L= 4, that is L fold over sampling of the signal, as illustrated in Fig.2.

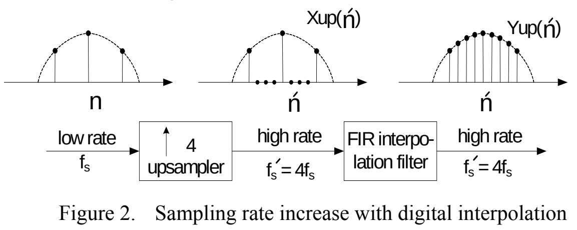


Figure 2. Sampling rate increase with digital interpolation

Consider an analog signal x(t) and its discrete time signal x(n), n = 0,1,2, - -, N -1 (N – the number of samples of the signal). The first stage of the L- fold over sampling of x(n) is to insert L-1, zero samples for every low-rate sample (the L fold upsampler). The resultant signal x_{up}(n is connected with the signal x(n) by the relationship [9]:

$$xup(n') = \begin{cases} x(n), & \text{If} \quad n' = nL \\ 0, & \text{If} \quad n' = nL+i \end{cases} \quad (1)$$

Where i = 1,2, - - -,L-1,

To reduce the computational cost by a factor of L, the high rate interpolating FIR filter is replaced by slow FIR subfilters, known as polyphase filters. Each polyphase filter has to be operated at low sampling rate Fs at a factor of 1/L as compares to the high rate single interpolator.

D Interpolation Filter

The filters can be implemented in the following two forms.

1) Direct form: The interpolation filter operates at the fast rate *fs*, with a cutoff frequency F_c equal to the low rate Nyquist frequency. Impulse response coefficients of the ideal L-fold interpolation filter are obtained from the equation [9], given below.

$$d\ (k') = \frac{\sin(\pi k'/L)}{(\pi k'/L)}, \quad -LM \le k' \le LM \quad (2)$$

or

$$h(k') = d(k' - LM) = \frac{\sin(\pi(k' - LM)/L)}{\pi(k' - LM)/L} \quad (3)$$

Where k′ = 0, 1, ..., N′ -1

The FIR approximation to the ideal interpolator is obtained by truncating d(k') to the finite length, say N = 2LM+1 and a casual filter is obtained by delaying it L×M samples. The output of the interpolation filter is obtained by convolving the upsampled input $x_{up}(n')$ with the impulse response d(k′) For example using d(k′) one can get:

$$y_{up}(nL+i) = \sum_{k'=-LM}^{LM} d(k')x_{up}(nL+i-k') \quad (4)$$

Where i = 0, 1, ..., L -1

2) *Polyphase form:* Using the above equation the coefficients of the impulse response for ideal FIR filter are calculated. The computational cost of direct form interpolator is $2L^2M$ and the cost of polyphase filter is $2LM$ for L interpolated values. The computational cost is L times less in case of polyphase filter.

$$y_{up}(nL+i) = \sum_{k'=-LM}^{LM} \sum_{j=0}^{j=L-1} d(kL+j) x_{up}(nL+i-kL-j) \quad (5)$$

Defining ith polyphase filter by *di(k) = d(KL+i), M ≤ k ≤ M-1* and by algebraic manipulation, the above equation may be simplified into the form

$$y_i(n) = \sum_{k'=-M}^{M-1} d_i(k) x(n-k), \; i = 0,\, 1,\, \ldots,\, L\text{-}1 \quad (6)$$

Where

$$d_i(k) = h(i,k) = \sin\frac{(\pi(kL+i)/L)}{\pi(kL+i)/L} \quad (7)$$

III Implementation Scheme Of The Parallel Multiprocessor Interpolating FIR Filter using multiplier and adder

Equation (7) is used to calculate impulse response for the ith subfilter and L is total number of subfilters. There are total N (0,1, ….., m) coefficients and the value of k represents the kth coefficient. Equation (6) is used to generate missing samples calculated by the subfilters. Block diagram of a polyphase interpolating filter is shown below in Fig 3. Each subfilter is consists of a processor that has N multipliers and (N – 1) adders to process the data. Each subfilter is implemented in parallel and processes the input data independently. The interpolated output data is achieved by collecting output of individual subfilters and arranging it into proper sequence.

The polyphase FIR filter can be implemented by a lookup tables containing all possible pre-calculated filter outputs. These are tabulated in memory for any input transition patterns which requires a very large memory. In this way the subfilter can be implemented using a ROM or Dual RAM in case it is to be implemented using FPGA.

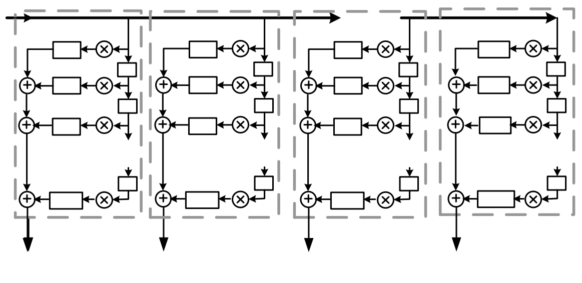


Figure 3. The parallel multiprocessor interpolating filter with multiplier and adder

IV. Multiplication Using Lookup Tables

The size of the lookup table can be reduced by using Quarter Squares technique and speed of the multiplication can be increased by using nibble based lookup table as explained below.

A *Bit multiplication*

An 8-bit multiplication can be perform by successive addition or binary shifting and accumulation, or by concatenating the two 8-bit values to give a 16-bit index into a massive 65,536 entry lookup table.

We can reduce the size of the lookup table to just 511 entries by using Quarter Squares by using the equation (8) given below

$$a \times b = \frac{(a+b)^2 - (a-b)^2}{4} \quad (8)$$

We can express the above equation in terms of function *f*(*n*) as given below

$$a \times b = \frac{(a+b)^2}{4} - \frac{(a-b)^2}{4} = f(a+b) - f(a-b) \quad (9)$$

Where $f(n) = \frac{n^2}{4}$

A lookup table of indexed n can easily represent the function itself.

The whole lookup table necessary to implement an 8-bit by 8-bit multiplication can then be held in just 511 entries. The maximum value that needs to be held is for an index of 511, and that table entry will have a maximum value of 255x255 (65,025), which can be held in a 16-bit word.

In reality no rounding error appears in the multiplication that is introduced by an integer division by four within the lookup table values.

B *Nibble based lookup table*

Another way to speed up multiplication, using lookup tables, additions and shifting, which are all usually efficiently implemented is to take individual groups of nibbles (4-bits) from the numbers to be multiplied and use those.

Every byte has two nibbles, the top four bits and the bottom four bits.

Consider the following example of multiplying two 2-digit numbers together

$$ab \times cd = \mathrm{a} \times \mathrm{c} \times 100 + \mathrm{a} \times \mathrm{d} \times 10 + \mathrm{b} \times \mathrm{c} \times 10 + \mathrm{b} \times \mathrm{d} \quad (10)$$

The function *f*(*x*,*y*) can be used to perform a 4-bit by 4-bit multiply, using 8-bit, 30 bytes lookup table.

$$ab \times cd = f(a,c) \times 100 + f(a,d) \times 10 + f(b,c) \times 10 + f(b,d) \quad (11)$$

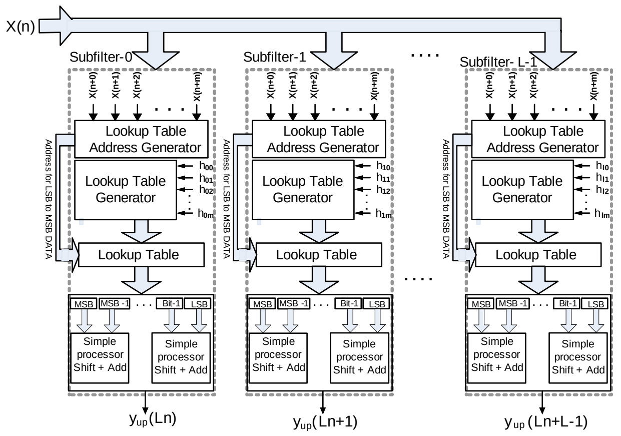
Where $f(\mathrm{x},\mathrm{y}) = x \times y$

The reduction in the size of the lookup table does reduce the execution speed due to the process of de-nibbling the numbers, nibble multiply, shift and accumulate operations.

The above process can be implemented efficiently within FPGA where parallel processing for the manipulation of nibble data is often very easily achievable.

V Proposed Implementation Scheme Of The Parallel Multiprocessor Interpolating Fir Filter

Block diagram of a low cost interpolating filter is shown below in Fig.4. It consists of lookup table generator, lookup table address generator, lookup table and parallel processing unit to do all required computation to obtain the required output. The coefficients of ideal interpolation filter are already known. The lookup tables can be generated for any number of coefficients (N) outside and can be imported in the proposed multiprocessing system. The lookup table generator section may be removed from the block diagram. Each subfilter contains small parallel processors which are consists of shifter and adder units to process the data. The main processor synchronizes all the small processors and feed data at low sampling rate to each subfilter. Each small processor is implemented in parallel and processes the input data independently. The output data, from each subfilter is arranged to get over sampled interpolated data having L times higher sampling rate by the main processor.



Figurev 4. Implementation of the proposed efficient parallel multiprocessor interpolating filter

A Lookup Table Generation

As the coefficients of the subfilters are already known, the tables can be generated and made readily available. Table-1 gives the coefficient of the subfilters (subfilter-0 to subfilter-3) for M = 1, 2 and 3 and the corresponding number of coefficients N = 2, 4 and 6. The 'Look up' table is generated using the algorithm shown below in.Fig-5. The number of address bits is equal to the number of the coefficients of the subfilter. If address bit is logic-1 then the value of the corresponding coefficient is added. A lookup table for Am = 3 (4 Bit Address for 4 coefficients of subfilter-0) is shown for elaboration, where h_{00}, h_{01}, h_{02} and h_{03} are the coefficient of the subfilter-0 and 16 table entries corresponding to the 4 coefficients. Similarly tables can be generated for each subfilter having any number of coefficients. Lookup table entries for each subfilter having N coefficients are 2^N which are generated using corresponding coefficients as shown in Fig.5.

TABLE-1
Impulse Response For M = 3 and Lfold = 4

h	0	1	2	3	4	5
0	0	0	0	1	0	0
1	0.0818	-0.1286	0.3001	0.9003	-0.1801	0.1
2	0.1273	-0.2122	0.6366	0.6366	-0.2122	0.1273
3	0.1	-0.1801	0.9003	0.3001	-0.1286	0.0818

TABLE-2
Lookup Table for Subfilter-0

Address	Data
0000	0
0001	h_{00}
0010	h_{01}
0011	$h_{01}+h_{00}$
0100	h_{02}
0101	$h_{02}+h_{00}$
0110	$h_{02}+h_{01}$
0111	$h_{02}+h_{01}+h_{00}$
1000	h_{03}
1001	$h_{03}+h_{00}$
1010	$h_{03}+h_{01}$
1011	$h_{03}+h_{01}+h_{00}$
1100	$h_{03}+h_{02}$
1101	$h_{03}+h_{02}+h_{00}$
1110	$h_{03}+h_{02}+h_{01}$
1111	$h_{03}+h_{02}+h_{01}+h_{00}$

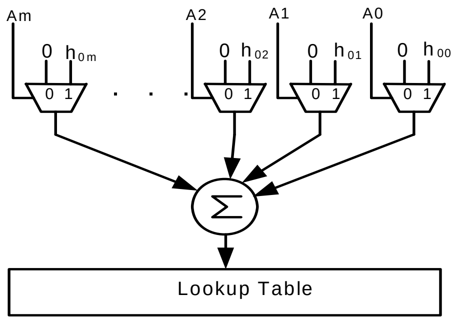


Figure 5. Algorithm for the lookup table generation

B Lookup Table Address Generation

The graphical representation of the algorithm is shown below in Fig.6. The binary values of the required samples X(n+0) to X(n+m) are used to generate the address to get output data from the lookup table as shown in Fig.6. The bit D0 of X(n+0) corresponds to Ao, the bit D0 of X(n+1) corresponds to A1 and similarly the bit D0 of X(n+m) corresponds to Am. As explained above D0 of each sample value is used to generate address Am to Ao for LSB. Where Am is the MSB of the address and number of address bits is equal to the number of filter coefficients being used for each subfilter. The bit D1 of each sample value is used to generate address A0 to Am for Bit-1. Similarly address is generated upto MSB location.

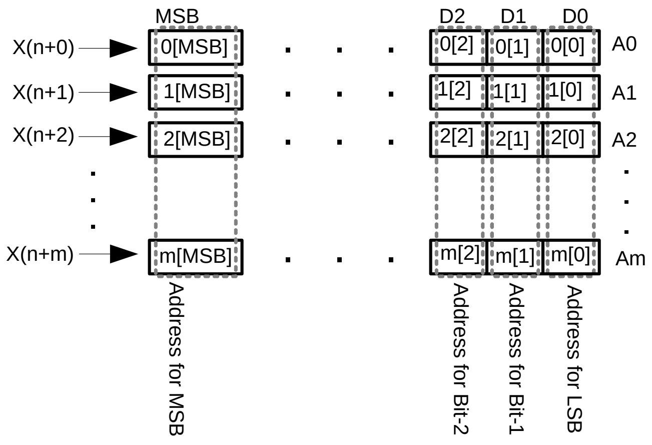


Figure 6. Algorithm for the lookup address generation generator.

D Parallel Processing of the Data

The number of locations for the parallel processing unit is equivalent to the width (word length - no. of bits) of the sampled data.

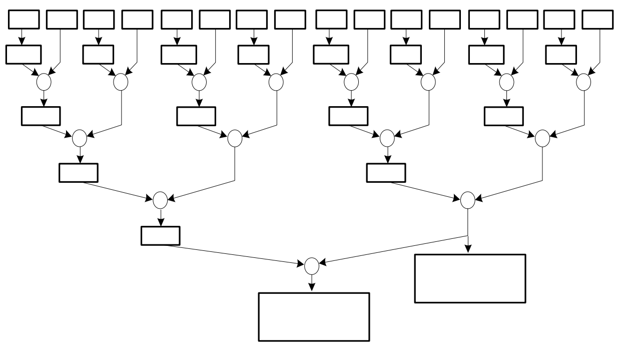


Figure 7. Algorithm for parallel processing of the data using shift and add operations.

The graphical representation of the algorithm is shown in Fig.7. The data is loaded from the lookup table by using the addresses generated in previous section 3.2. The data from Bit-1 is shifted by 1 and added to LSB (Bit-0). It is done for all the consecutive pairs in parallel upto MSB. The result of the 2nd pair is further shifted by 4 and added to the result of the 1st pair. It is repeated for all other consecutive sets of 4 data values. The result of 2nd set of 4 data values is further shifted by 4 and added to the resultant of the 1st set of 4 data values. This makes a set of 8 data values and is the required result if the sampled data width is 8 bits. For 16-bit data width the resultant of the 2nd set of 8 values (Bit-8 to Bit-15) is further shifted by 8 and added to the resultant of 1st set of 8 data values. This makes a set of 16 data values and is the required result, if the sampled data is 16-bits. Similarly one more shift is required if the sampled data width is 32-bits. Only 3,4 and 5 shift and add operations are required for 8-bit, 16 bit and 32 bit wide sampled data.

The computational cost of direct form interpolator is $2L^2M$ and the cost of polyphase filter is 2LM for L interpolated values. Cost of proposed implementation is only four shift and four add operations.

VI SIMULATION USING MATLAB FOR THE PROPOSED POLYPHASE INTERPOLATING FILTER

The methodology for the simulation of the signal processing through polyphase interpolating filters, as described above, is carried out by first determining the coefficients i.e. impulse response of each subfillter. The impulse response of the sinc interpolator is listed in table-1. The software implementation is covered in ensuing paragraph.

The input sinusoidal signal shown in Fig-8(a) is converted into discrete signal by sampling it at Fs/Fa = 4 samples/cycle to generate Basic Set of Samples (BSS) as shown in Fig-8(b). Three subfilters are used to calculate three additional samples to be inserted between the BSS. Taking M = 2, four BSS are used to calculate additional sample by each subfilter. The plot of the calculated interpolated values inserted between BSS is shown in Fig-8(c) and (d). Part (a) and (b). of both the Figures Fig-8 and Fig-9 are same. Fig-9(c) and (d) are plotted using M = 3 and BSS = 6.

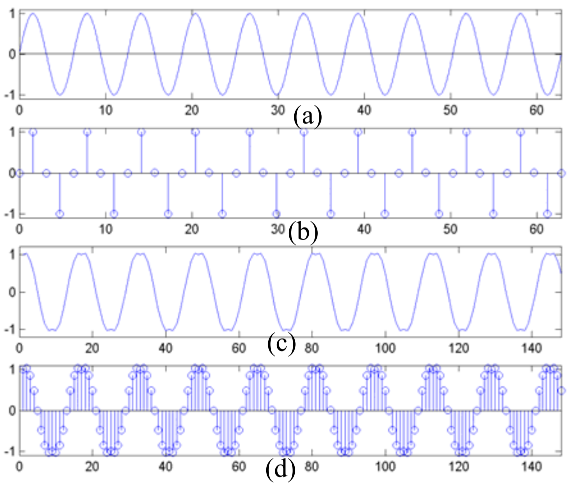


Figure .8 plot to show the reconstructed signal and interpolated high rate samples using Lfold =4, M=2 and N = 17
(a.) original signal, (b) sampled signal, (c) reconstructed signal and (d) high sampling rate interpolated samples values.

The results by the graphical presentation of the reconstructed samples for different coefficients and different number of Basic Set of Samples (BSS) of low rate sampled data are used for interpolation algorithm. For M=2, there is a delay of first two BSS samples which corresponding to $\pi/2$. Three samples are inserted between BSS samples 2 and 3. For M=3, there is a delay of first three BSS samples which corresponding to π. This delay is used to make the system a causal system. The results show that, more the number of BSS and Lfold better is the reconstructed plot.

The computational cost of direct form interpolator is $2L^2M$ and the cost of polyphase filter is 2LM for L interpolated values. The computation cost of the proposed scheme is a few add and shift operations. Hadamard transform has been used for polyphase decomposition of FIR filter and its applications in efficient FIR filter design [1]. The number of multiplications is reduced but its computational cost is still higher than the proposed implementation.

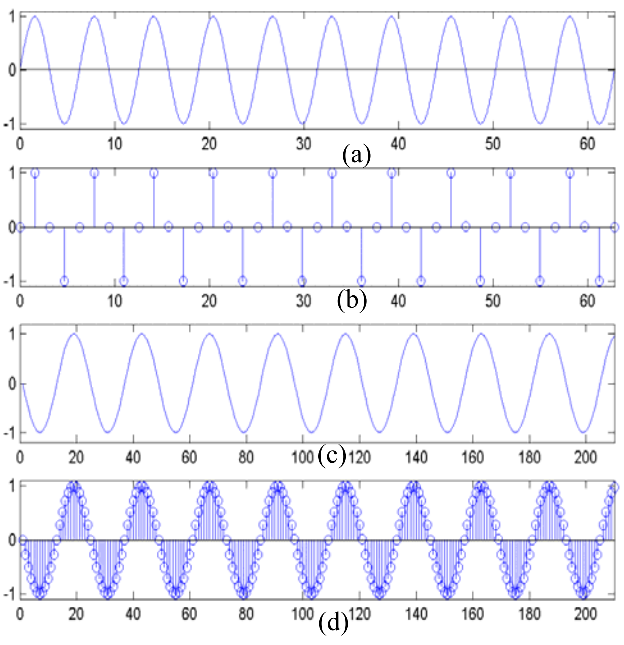


Figure 9. plot to show the reconstructed signal and interpolated high rate samples using Lfold = 6, M = 3 and N = 37

VII. Conclusions

A fast approach to implement parallel multiprocessor interpolating polyphase filter is presented. The tables are generated and made readily available, which are tabulated in the memory. Other discrete interpolating kernels may be implemented by generating tables for them. The implementation is done in parallel and the computation in the memory. Other discrete interpolating kernels may cost in term of Shift and Add operations remains same for any number of filter coefficients (N). Only 3 Shift and 3 add operations are required in series for all the calculations if the width of the sampled data is 8-bit. Similarly for 16-bit wide sampled data, maximum 4 shift and 4 add operations are required in series for all the calculations. The lookup table size is very small as compared with other techniques. The method is an efficient low cost technique and can be implemented on FPGA where lookup tables (LUTs) are the basic logic blocks.

References

1] Sanjit K Mitra, Fellow IEEE, Abhijit Mahalanobis and Tapio Saramaki, "A generalized structureal subband decomposition of FIR filter and its applications in efficient FIR filter design and implementation," IEEE transactions on circuits and systems-II Analog and digital signal processing, Vol 40, pp 363-374, 6 Jun 1993

[2] Shyh-Jye Joe, Kai-yaun jheng, Hsiao-Yun Chen and An-yeu Wu. "A multiplierless Multirate Decimator/Interpolator module generator," IEEE Asia- pacific Conference on advanced system integrated circuits (AP-Asic-2004), pp 58-61, August 4-5, 2004..

3] Robert Adams and Tom Kwan, "A stereo asynchronous digital sample-rate convertor for digital audio," IEEE journal of solid state circuits, Vol 29, NO. 4, pp 481-487, April 1994

[4] Sangyam Hwang, Gunhee Han, Sungho Kang and Jae Seok KLM, A low power implementation scheme of interpolation FIR filter using distributed arithmetic , IEICE Treans, Electroni Vol E 86-C, No. 11 2003, pp 2346-2350

[5] J. Proakis and D. Manolakis, "Digital Signal Processing: Principles, Algorithms, and Applications," Prentice-Hall:Upper Saddle River, New Jersey, 1996.

[6] R. Peterson, R. Ziemer, and D. Borth, "Introduction to Spread-Spectrum Communications," Prentice Hall:Englewood Cliffs, New Jersey, 1995.

[7] L. Mintzer: "FIR Filters with Field-Programmable Gate Arrays," IEEE Journal of VLSI Signal Processing (August 1993) 119{128

[8] L. Phillips and H. T. Nagle, Jr., "Digital Control System Analysis And Design," 3rd edition, Prentice-Hall International, Inc, 1998.

[9] S. J. Orfanidis, "Introduction to signal processing", Prentice Hall Inc., A Simon& Schuster Company, Englewood Cliffs, New Jersey 07632, 1996.