A High Performance Multi-rate Reconfigurable VLSI Architecture for LMS Adaptive Filter Using Low complexity Distributed Arithmetic

Mr. Mohammad shaffi[1]

[1]*Research Scholar, Department of Instrument Technology,*

Andhra University, Visakhapatnam, Andhra Pradesh, India

Mrs. Angajala Kamala Kumari[2]

[2]*Assistant Professor, Department of Instrument Technology,*

Andhra University, Visakhapatnam, Andhra Pradesh, India

ABSTRACT:- *A high performance multi-rate reconfigurable VLSI architecture for LMS Adaptive filter using low complexity distributed arithmetic is presented. In Multi-rate Signal Processing studies used in Digital Signal Processing systems include sample rate conversion. This technique is used for systems with different input and output sample rates, but may also be used to implement systems with equal input and output rates. We are going to study the architecture of Interpolator and Decimator with low complexity, LMS adaptive filter with distributive arithmetic which is based on storing possible filter partial products in look-up-table fallowed by shift accumulator unit. The proposed technique employs RAM based LUT for storing offset binary coding combination of input samples and filter weights.*

KEY WORDS— Interpolator, decimator, LMS(Least Mean Square),DA(Distribution Arithmetic).

I. INTRODUCTION

Recently, there has been rapid progress in the area of multi-rate digital signal processing. In multi-rate systems, decimation and interpolation filters are the most important building blocks. A great amount of literature deals with design of LMS Adaptive Filter, decimator and interpolator**.** Adaptive filters are widely used in many signal processing applications such as echo and noise cancelation etc. The weights of FIR filter are updated using LMS algorithm due to its simplicity and satisfactory convergence**.** Since the speed of processing time and the silicon area are the crucial factors in the VLSI implementation a scalable implementation scheme to flexibly and efficiently implement the multi-rate LMS Adaptive filters is presented in this paper.

Croisier *et al.* [6] introduced an efficient multiplier less approach known as distributed arithmetic (DA). It consists of a look-up table (LUT) and a shift-accumulate (SA) unit. Unlike MAC-based adaptive filter, it is more effective technique for realizing large tap-sized filters. This is due to pre-computation of filter partial products and storing them in LUT. As a result, it occupies relatively smaller area as compared with MAC based design.

One of the most important applications of multi-rate systems is sub-band coding **(SBC).** Since it was introduced by Croisier *et al.* in 1976, sub-band coding is so far one of the most effective coding approaches for video and audio applications. Because sub-band coding needs filter banks to split the input signal aside from the coding mechanism, the filter banks are the kernel of this coding scheme. For sub-band coding systems,, implementation of their filter banks is the most important task. Since filter banks usually deal with large amount of

data, high speed computing hardware is indispensable. In order to achieve both high performance and low complexity, by employing the scalable implementation scheme, we propose an efficient design technique suitable for all types of sub-band filter banks i.e LMS Adaptive Filter.

In Digital Signal Processing System sometimes it becomes necessary to convert the data to a new rate to make it easier to process or to achieve compatibility with another system. Therefore Multi-rate Signal Processing is used which is defined as the discrete time system that process data at more than one sampling rate to perform the desire digital operations.

In this work, in general we are going to present a systematic approach for the low-power design of a LMS Adaptive Filter based on the multi-rate approach. The direct implementation of the system transfer function ***H (z)*** has the constraint that the speed of the processing elements must be as fast as the input data rate. It cannot compensate the speed penalty under low supply voltage. On the other hand, the multi-rate system will require only low-speed processing elements at half of the original clock rate to maintain the same throughput. Therefore, the processing elements can be operated at a lower supply voltage to reduce the power dissipation and the data throughput rate is not degraded by the lowered voltage. As a result, the multi-rate implementation can provides a direct and efficient way to compensate the speed penalty in low-power designs at the algorithmic/ architectural level**.**

Guo *et al.* Proposed a new strategy using a single LUT to store the filter partial products [8]. However, it requires complex external weight updating block. Surya *et al.* proposed a novel approach using offset binary coding (OBC) scheme to update the filter weights[9]. Due to OBC combinations of input samples and filter weights, the throughput achieved was several folds. On the other hand, the critical path is increased significantly due to rotation of physical addresses. In order to eliminate the rotation of physical addresses, we employed random access memory (RAM) based LUT which has both read and write operations. By extending the idea of RAM based LUT, a new strategy for updating the filter weights is derived. It is based on storing the OBC combinations of input samples in weight updating LUT which does not require physical rotation of addresses. This helps to save chip area, logic complexity and power consumption.

II. DESIGN ASPECTS

The section will discuss the techniques used for realizing LMS adaptive filters, Interpolator and Decimator

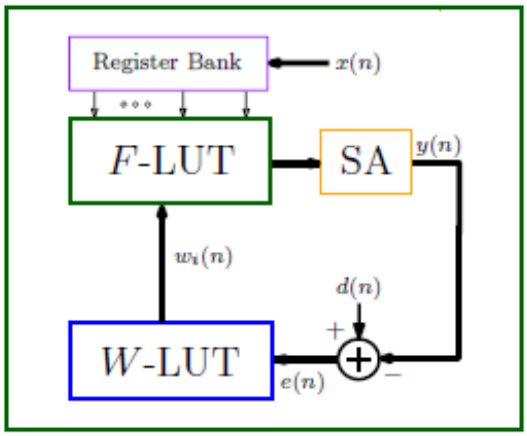

Fig. 1(a) Block diagram of DA based LMS adaptive filter.

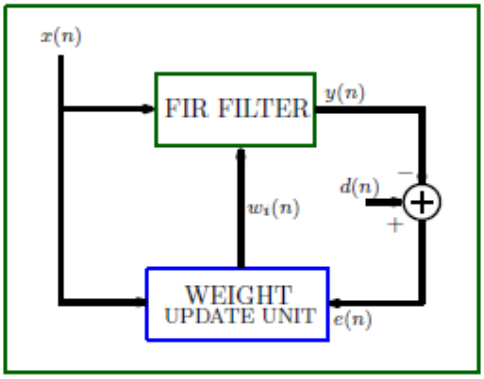

Fig. 1(b) Block diagram of MAC based LMS adaptive filter.

A. MAC based LMS adaptive filter

The block diagram of a N-tap MAC based LMS adaptive filter is shown in Fig. 1(b). It process an input sample (n) and produces the output signal (n) according to

$$y(n) = \sum_{i=0}^{N-1} w_i(n)x(n-i)$$

where $wi(n)$ denotes the filter weights at time instant n with $i \in [0, N-1]$. Using (n), an error signal (n) is computed from desired signal (n) as follows

$e(n) = d(n) - y(n)$

The filter weights are updated based on LMS criterion for next iteration as per

$$w_i(n+1) = w_i(n) + \mu e(n)x(n-i)$$

The number of additions and multiplications for realizing the adaptive filter are $2N + 1$ and $2N$ respectively. Due to multipliers, the area occupied will be very large.

B. DA based LMS adaptive filter

DA based LMS adaptive filter are mainly classified as twos complement and offset binary coding (OBC) schemes. A typical block diagram of DA based LMS adaptive filter is shown in Fig. 1(a). It is similar to MAC-based filter but requires a register bank and shift-accumulate (SA) units. In addition, the filtering and weight updating units are replaced by filtering LUT (F-LUT) and weight updating LUT (W-LUT).

1) ***Twos Complement Scheme:*** This technique requires 2's complement representation of input samples ($n-i$), according to

$$x(n-i) = x_{n-i} = -x_{i,B-1} + \sum_{j=1}^{B-1} x_{i,j}2^{-j}$$

By substituting above equation in first equation and rearranging, we get

$$y(n) = \sum_{j=0}^{B-1} b_{B-1-j}2^{-j}$$

where,

$$b_{B-1-j} = \sum_{i=0}^{N-1} w_i x_{i,B-1-j} 2^{-j}$$

$$b_{B-1} = -\sum_{j=0}^{B-1} w_i x_{B-1}$$

It can be noted from that the term $bB-1-j$ represents filter partial products, which could take $2N$ possible combinations due to input samples are bit-sliced as $Xi,B-1-j \in [0, 1]$. Hence, the term $bB-1-j$ can be pre-computed and stored in a LUT whose address bits are least significant bits (LSBs) of registers present in the register bank. Depending on the bit-slices $Xi,-1-j$ for $i \in [0,N-1]$, the partial products are accessed from LUT and undergoes SA operation for B clock cycles to produce the output.

*2) **Offset binary coding (OBC) Scheme**:* In twos complement scheme, the number of partial products to be stored in LUT whose complexity grows exponentially with filter tap-size (N). In order to reduce the LUT size by half, offset binary coding (OBC) scheme can be used [3]. This scheme exploits the redundancy between input samples and their twos complement. It requires $(n-i)$ to be coded as $xn-m = (1/2)[xn-i - (-xn-i)]$. By using x(n-i), this can also be re-written as

$$x_{n-i} = \frac{1}{2}\Bigg[-(x_{i,B-1} - \overline{x}_{i,B-1}) + \sum_{j=1}^{B-1}(x_{i,B-1-j} - \overline{x}_{i,B-1-j})2^{-j} - 2^{-(B-1)}\Bigg]$$

Let

$$p_{i,j} = \begin{cases} -(x_{i,B-1} - \overline{w}_{i,B-1}), & \text{if } j = B-1 \\ x_{i,B-1-j} - \overline{w}_{i,B-1} & \text{otherwise} \end{cases}$$

Therefore,

$$y(n) = \sum_{j=0}^{B-1}(\sum_{i=0}^{N-1}\frac{1}{2}w_i p_{i,B-1-j})2^{-j} + p_{initial}2^{-(B-1)}$$

$$y(n) = \sum_{j=0}^{B-1} c_{B-1-j}2^{-j} + p_{initial}2^{-(B-1)}$$

It is important to note from above equation that the term $CB-1-j$ represents the partial products. Importantly, the number of $cB-1-j$ terms to be stored in LUT are reduced by half. This is due to $xi,-1-j \in [-1, 1]\ \forall i$ which makes the other half are mirror image. This reduction in LUT size comes at the expense of few XOR gates for sign-reversal operation and hardware cost for $Pinitial$ term, as shown in Fig. 2(b).

Multi-rate filters include both interpolation and decimation filters. Interpolation increases the sample rate by inserting zero-valued samples between the original samples, while decimation discards samples to decrease the sample rate. The FIR Compiler automatically creates interpolation and decimation filters using poly-phase decomposition. Poly-phase filters simplify the overall system design and also reduce the number of computations per cycle required by the hardware.

***Interpolation filters*:-**

An interpolation filter increases the output sample rate by a factor of *I* through the insertion of *I-1* zeros between input samples, a process known as zero padding. Poly-phase decomposition reduces the number of operations per clock cycle by ignoring the zeros that are padded in between the original input samples. Poly-phase interpolation filters provide both speed and area optimization because each poly-phase filter runs at the input data rate for maximum throughput.

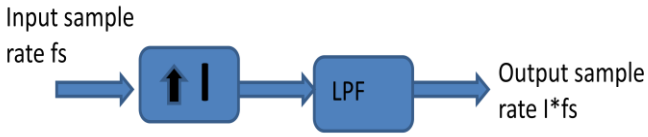

Fig 2.(a) Poly-phase interpolation block diagram

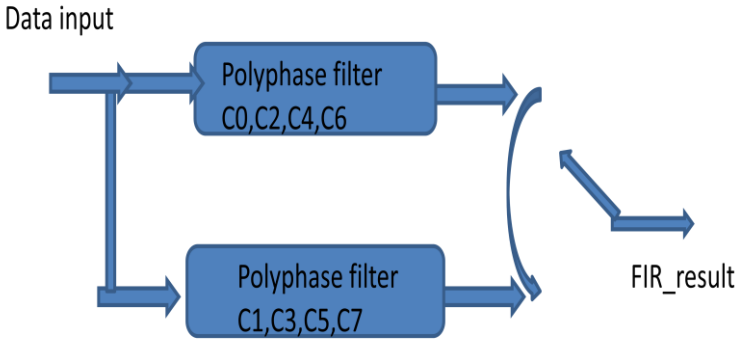

Fig 2.(b) poly-phase decomposition for interpolation filters

***Decimation filters*:-**

A decimation filter decreases the output sample rate by a factor of *D* by keeping only every *D*-th input sample. Poly-phase decomposition reduces the number of computations per cycle by ignoring the input data samples that are discarded during down sampling. Poly-phase decimation filters provide speed optimization because each poly-phase filter runs at the output data rate.

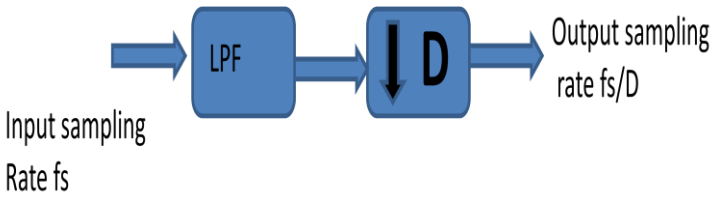

Fig 3.(a) Poly-phase decimation block diagram

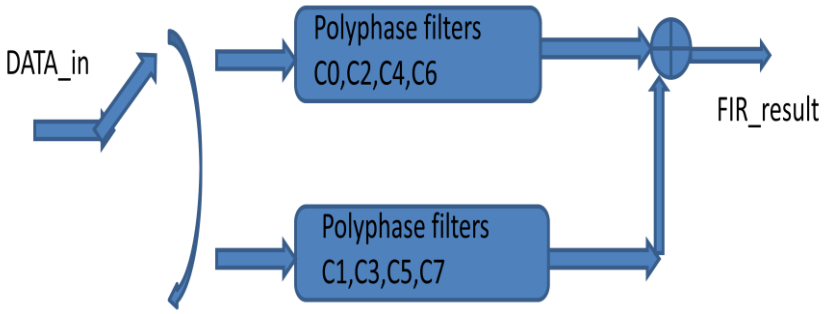

Fig 3.(b) poly-phase decomposition for decimation filters

III. LOGIC BLOCK DIAGNOSIS

Main Concept:

The block diagram of proposed DA based LMS adaptive filter is similar to shown in Fig.1(a). The *F*-LUT and *W*-LUT stores the OBC combinations filter weights and input samples respectively. More importantly, the LUTs are random-access memory (RAM) type. For simplicity, consider a four tap DA based adaptive filter with contents of *F*-LUT and *W*-LUT at time instant *n* are shown in Fig. 2.

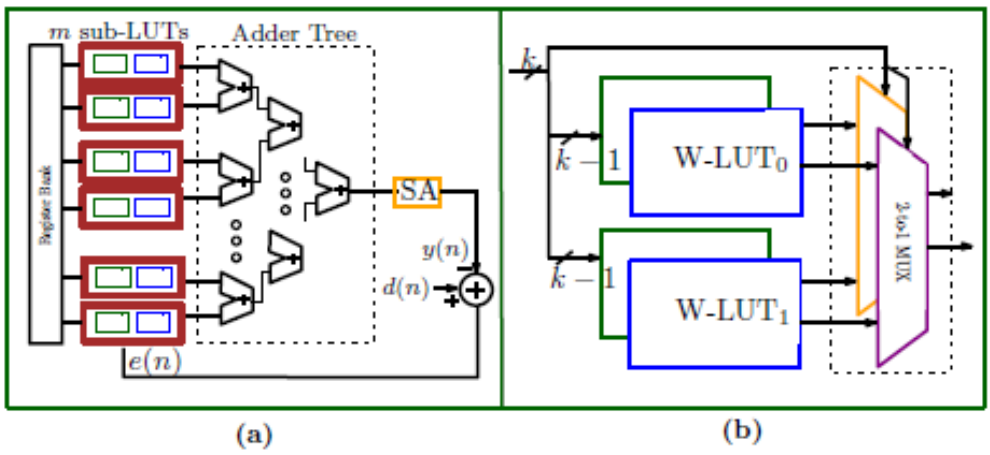

Fig. 4: Techniques for reducing LUT access time, using (a) adder tree (b) multiplexers.

Note that, *F*-LUT stores the OBC combinations of filter weights whereas *W*- LUT stores OBC combinations of input samples except for recent input sample (*n*). Each entry of *F*-LUT at time instant *n* are addressed by index *a* as follows

$$F_a(n) = \frac{1}{2}[w_0(n) + \sum_{i=1}^{N-1} w_i(n)(-1)^{q^a_{N-1-i}+1}]$$

where, q^a_l is *lth* bit in *N*-bit representation (q^a_l) of address *a*. That is,

$$a = \sum_{i=0}^{N-2} q^a_i 2^i$$

In similar manner, the entry of *W*-LUT at time instant *n* with input samples ($n - i$) can also be given as

$$W_a(n) = \frac{1}{2}[x(n) + \sum_{i=1}^{N-1} x(n-i)(-1)^{q^a_{N-1-i}+1}]$$

where the recent input sample $x(n)$ is not stored in LUT, therefore, $Wa(n)$ of W-LUT are also addressed by a as follows

$$W_a(n) = \frac{1}{2}[\sum_{i=1}^{N-1} x(n-i)(-1)^{q^a_{N-1-i}+1}]$$

It can be observed that W-LUT stores the OBC combinations (upper half) of input samples except for the recent sample (n). The update scheme for W-LUT in the proposed implementation from time n to n+1 is explained as follows. The subtraction of content at first address location of W-LUT from oldest sample (n−3) would result in a term independent of (n − 3) sample followed by the addition of (n) sample. Let the contents of W-LUT at time instant n is denoted by (n). Therefore, the contents of W-LUT at time instant n can also be expressed as

$$W_i(n+1) = \frac{1}{2}[(-1)^{i\%2^{N-1}} x(n) + (-1)^{i+1\%2} x(n-N+1) + W_i(n)]$$

It is important to note that when the recent sample (n) has arrived, its right-shifted version *i.e.*, 1/2(n) is either added or subtracted. Similarly, the right-shifted version of (n − 3) sample *i.e.*, 12(n−3) is either subtracted or added. Note that *pinitial* term in (8) can be stored in a register and concurrently updated in every iteration with the arrival of recent sample (n).

loop
  $y(n) = \sum_{j=0}^{B-1} c_{B-1-j} 2^{-j}$
  for $a = 0$ to $2^{N-1} - 1$ **do**
    if $a \bmod(2^{N-1}) == 0$ **then**
      if $a \bmod(2) == 0$ **then**
        $W_a \leftarrow W_a - x(n-N+1) + x(n)$
      else
        $W_a \leftarrow W_a + x(n-N+1) + x(n)$
      end if
    else
      if $a \bmod(2^{N-1}) == 1$ **then**
        $W_a \leftarrow W_a - x(n-N+1) - x(n)$
      else
        $W_a \leftarrow W_a + x(n-N+1) - x(n)$
      end if
    end if
  end for
  $e(n) \leftarrow d(n) - y(n)$
  for $a = 0$ to $2^{N-1} - 1$ **do**
  $F_a(n+1) \leftarrow F_a(n) + \mu e(n)\{W_a(n) + x(n)\}$
  end for
  return $y(n)$
  $n \leftarrow n + 1$
end loop

Fig. 5: Algorithm explaining proposed DA based adaptive filter.

The access time of *F*-LUT and *W*-LUT can be reduced by splitting a large *N*-tap filter into *m* small sub filters with tap size *k* such that $N = m \times k$ as shown in Fig. 5(a). It can also be stated as follows

$$c_{B-1-j} = \frac{1}{2} \sum_{l=0}^{m-1} \sum_{s=l}^{k-1} w_s p_{s,j}$$

where, $0 \le s \le l(k-1)$ and $0 \le l \le m$. It is clear from (16) that an adder tree is required of depth $\log 2m$. One can also reduced the access time of LUT by splitting into two small multiplexed LUTs. The algorithm of proposed filter is illustrated in Fig. 5.

An interpolation filter increases the output sample rate by a factor of *I* through the insertion of *I-1* zeros between input samples, a process known as zero padding. Poly-phase decomposition reduces the number of operations per clock cycle by ignoring the zeros that are padded in between the original input samples. Poly-phase interpolation filters provide both speed and area optimization because each poly-phase filter runs at the input data rate for maximum throughput.

A decimation filter decreases the output sample rate by a factor of *D* by keeping only every *D*-th input sample. Poly-phase decomposition reduces the number of computations per cycle by ignoring the input data samples that are discarded during down sampling. Poly-phase decimation filters provide speed optimization because each poly-phase filter runs at the output data rate.

The drawback of using interpolator and decimator is complexity of the circuit is increased if we use normal FIR filters.

To overcome this draw back we go for low complexity FIR filters such as FIR filter design by using PSM(programmable shift method) architecture .By doing so the overall complexity of the Interpolator and decimator circuit is reduced.

B. Implementation and Experimental Results:

Multi-rate Signal Processing studies Digital Signal Processing systems which include sample rate conversion. This technique is used for systems with different and equal input and output sample rates.

Multi-rate filters are required when changing the sampling rate of a data path in a system. Multi-rate filters include both interpolation and decimation filters. Interpolation increases the sample rate by inserting zero-valued samples between the original samples, while decimation discards samples to decrease the sample rate. The FIR Compiler automatically creates interpolation and decimation filters using poly-phase decomposition. Poly-phase filters simplify the overall system design and also reduce the number of computations per cycle required by the hardware.

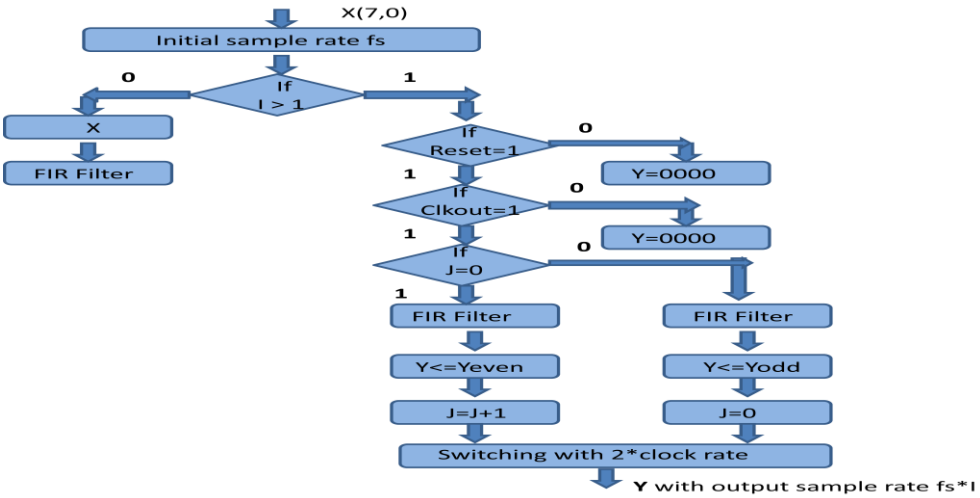

Fig 6.ASM chart of Interpolator Top-order module

The above ASM chart represents top-order module of the Interpolator. The ASM chart describes the sequence of events as well as the timing relationship between the states of a sequential controller and the events that occur while going from one state to the next.

Interpolation module is explained in five stages.Stage1 represents the sample rate check operation i.e sampling rate at input module is compared with sampling rate of data (fs).stage2 checks the scaling factor I if I = 1 then sampling rate remains as it is and the input message X is given as input to the filter or else we check reset condition.stage3 check the reset condition of circuit which is an active low enable, if reset = 0 then the output the circuit is y=00000 or else we check for output clock enable i.e clkout which is also an active high enable, if clkout=0 then the output of the circuit is y=00000 or else we perform poly-phase decomposition by checking for j=0.stage4 if j=0 filtering is done to the even coefficients and the output of the filter is y=yeven or else the filtering is done to the odd coefficients and the output of the filter is y=yodd.stage5 now we operate the output switch of the circuit with double the clock frequency so now the output of the interpolator is double the sampling rate at the input.

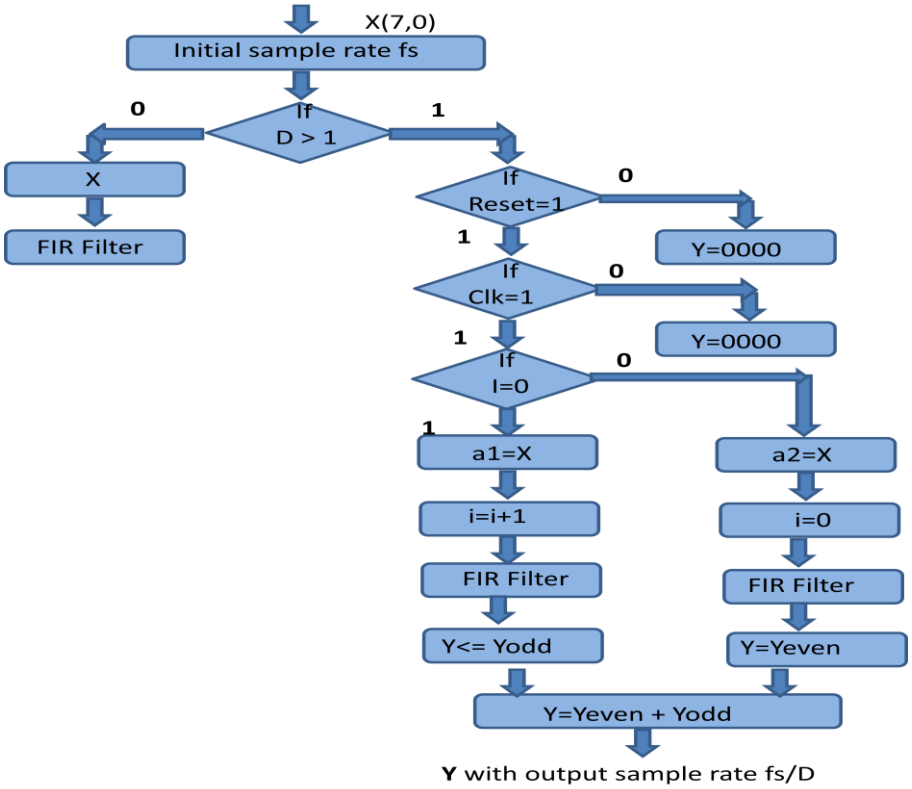

Fig 7.ASM chart of Decimator Top-order module

Figure 7 represents the ASM chart of top-order module of the Decimator. Decimator module is explained in five stages.Stage1 represents the sample rate check operation i.e sampling rate at output module is compared with sampling rate of data (fs).stage2 checks the scaling factor D if I = 1 then sampling rate remains as it is and the input message X is given as input to the filter or else we check reset condition.stage3 check the reset condition of circuit which is an active low enable, if reset = 0 then the output the circuit is y=00000 or else we check for clock enable i.e clk which is also an active high enable, if clk=0 then the output of the circuit is y=00000 or else we perform poly-phase decomposition by checking for i=0.stage4 if i=0 then X is given as input to the firfil1 and the output of the firfil is yodd or else X is given as input to the firfil2 and the output of the firfil is yeven.stage5 hear we summate all the outputs of y i.e yeven and yodd and so now the output of the decimator is half the sampling rate at the input.

IV.SIMULATION RESULTS

In order to validate the proposed scheme, a four-tap LMS adaptive filter was simulated in VHDL. For simplicity, the design in [6] is referred as DA0, the first and second designs in [7] are referred as DA1 and DA2, respectively and; the design in [12] is referred as DA3. Throughput of an adaptive filter [6] is defined as

$$\text{Throughput} = \frac{1}{\text{Critical path x Number of clock cycles}}$$

It depends on the number of clock cycles involved in *F*-LUT, *W*-LUT, SA unit and error computation, with critical path = 1/clock rate. In general, the update of LUT takes the longest time, thus limiting the system throughput. The critical path of DA0 and DA3 schemes also depend on the rotation of physical addresses which introduces significant delay. On the other hand, the DA1 and DA2 schemes do not require physical rotation of addresses due to complex external weight updating block. Thus, DA1 and DA2 schemes have smaller critical path as compared to DA0 and DA3 schemes. Similar to DA3, the proposed design has also less LUT access time since two small sized LUTs (LUT0 and LUT1) are used. Hence, the number of clock cycles required for the proposed and DA3 schemes are same. However, the proposed design does not have address rotation circuitry makes the operation of filter faster as compared to DA3 scheme. In order to compare the throughput of proposed design with that of DA3 scheme, we have defined a figure of merit called as clock speed-up, according to,

$$\frac{T_R + 2T_M + 2T_A}{T_R + (k+1)T_M + 2T_A}$$

It is the ratio of critical paths of proposed design to DA3 scheme It is clear from above equation that when taps of sub-filter increases, it decreases the relative critical path. But, the proposed filter still performs faster operation as compared to DA3. Note that DA3 is similar to DA0 since it also involves address rotation circuitry. In order to estimate the total number of clock cycles required for the proposed design, it is necessary to estimate the time required in terms of clock cycles for different filter components. The clock cycles required to update *F*-LUT contents are $2k/2$ whereas *W*-LUT requires $\max(B, 2k/2-1)$ clock cycles, since the operation of SA unit is performed in parallel with *W*-LUT. Apart from that, one more clock cycle is required for computing the error signal (n). Using decomposition of large order filter into sub-order filters, which requires an adder tree that would take $\log 2m$ clock cycles. Hence, the total number of clock cycles for the proposed design are $\max(B, 2k/2 - 1) + 2k/2 + \log 2m + 1$. Interestingly, when the tap-size of sub-filter increases, the proposed implementation is found to be more advantageous due to clock speed-up gain as compared to existing schemes.

TABLE I: General Comparison of Time, Hardware and Memory Complexities of Different Schemes

Design	Throughput	Adders (per cycle)	Registers	SH	Memory
DA_0 [6]	$1/[m_0(T_R+(k-1).T_M+T_A)]$	$m.(2^{k-1}+2^k)+m.B-1$	$m.(1+k)+2$	m	$2m.(2^k-1)$
DA_1 [7]	$1/[m_1(T_R+T_A)]$	$m.(2^{k-1}+k)+m.B-1$	$m.(2+2k)+1$	$m.k$	$m.(2^k-1)$
DA_2 [7]	$1/[m_1(T_R+T_A+T_M)]$	$m.(2^{k-1}+k+1)+m.B+1$	$m.(2+2k)+1$	$m.k$	$m.2^{k-1}$
DA_3 [12]	$1/[m_2(T_R+(k+1).T_M+2T_A)]$	$m.(2^{k/2+1}+2^{k/2+2}+2)+m.B+1$	$m.(4+k)+2$	m	$m.2^k$
Proposed	$1/[m_2(T_R+2.T_M+2T_A)]$	$m.(2^{k/2+1}+2^{k/2+2}+2)+m.B+1$	$m.(4+k)+2$	m	$m.2^k$

$N = mxk$, $m_0=2^k+\max(B,2^{k-1})+\log_2 m$, $m_1=2^{k-1}+\log_2 N+W+1$, $m_2=2^{k/2}+\max(W,2^{(k/2)-1})+\log_2 m+1$, T_R = LUT access time, T_M = 2-to-1 multiplexer delay and T_A = adder delay. Note that, the scheme in [9] is based on delayed LMS for pipelined realization of adaptive filters. In addition to above listed complexities, the proposed design does not require address rotation circuitry while the designs DA_0 and DA_3 have address rotation circuitry.

However, the complexity due to rotation of physical address is reduced, unlike DA0 and DA3 scheme. Further, the size of LUTs for DA0, DA1, DA2 and DA3 schemes are $m.(2k+1-2)$, $m.(2k-1)$, $m.2k-1$ and $m.2k$, respectively. Interestingly, the proposed scheme has same LUT size as that of DA1 and DA3 scheme. Moreover, DA1 and DA2 schemes have a low memory requirement, updating of filter weights operation requires more hardware complexity. In contrast, DA2 and DA3 schemes have extra storage registers and adder units for pre-computation of certain terms with minimum memory complexity.

TABLE II: Comparison of LEs for sub-filter tap-size $k = 4, 8$ with tap-size $N = 16, 32$ and 64

Design	Filter Taps (N)					
	16		32		64	
	$k=4$	$k=8$	$k=4$	$k=8$	$k=4$	$k=8$
DA_0 [6]	915	798	1429	1073	2426	1624
DA_1 [7]	845	588	1289	812	2134	1245
DA_2 [7]	803	512	1193	724	1816	1078
DA_3 [12]	712	467	1023	602	1357	905
Proposed	663	372	981	492	1203	788

TABLE III: Comparison of power (in mW) consumption for sub-filter tap-size $k = 4, 8$ with tap-size $N = 16, 32$ and 64

Design	Filter Taps (N)					
	16		32		64	
	$k=4$	$k=8$	$k=4$	$k=8$	$k=4$	$k=8$
DA_0 [6]	15.83	11.87	19.82	14.94	37.15	24.95
DA_1 [7]	14.15	8.65	17.65	11.45	32.81	17.93
DA_2 [7]	12.56	8.54	15.88	10.43	26.77	16.88
DA_3 [12]	12.23	8.34	14.63	9.56	21.56	15.87
Proposed	11.57	7.81	13.27	8.31	18.78	13.23

Interpolator:-

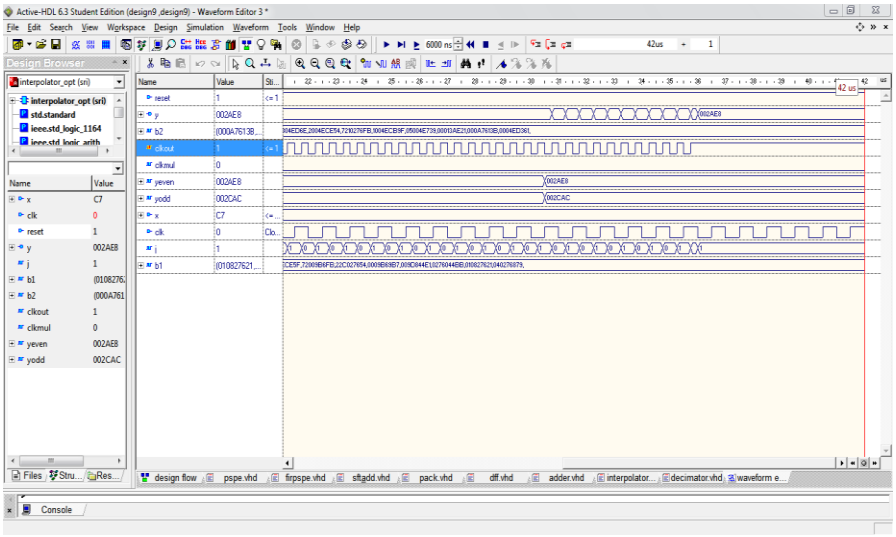

Fig 8.Simulation results of interpolator Top-order module

We can achieve multi-rate sampling by using Interpolator at input side but the drawback of this technique is complexity is increased, to reduce this complexity we use low complexity FIR filters. complexity is reduced by designing FIR filter by using shift and add module along with programmable shifting method.

Decimator:-

The same complexity persists in the decimator as well and this is also reduced by using same technique.

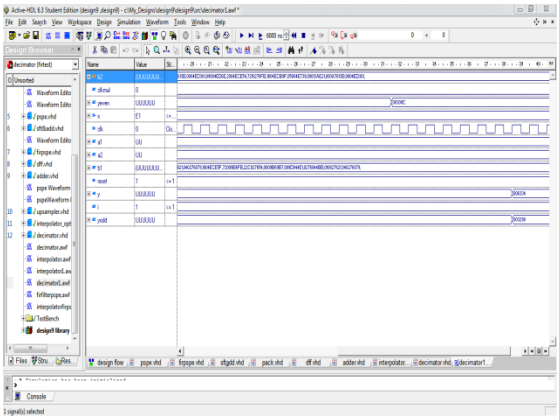

Fig 9.Simulation results of decimator Top-order module

FPGA REALIZATION

The designed system is targeted on to Xilinx Virtex FPGA device belonging to virtex6v family with a speed grade of -1. It is observed that above 50% area for the targeted FPGA is covered for the implementation of this system. The CLB'S are connected in cascade manner to obtain the functionality for the designed system.

A. Synthesis Report

Synthesis is a process of constructing a gate level net list from a model of a circuit described in VHDL Figure Depict the synthesis report of multiple fault diagnosis modules in Xilinx ISE 14.1 Environment for VHDL source code.

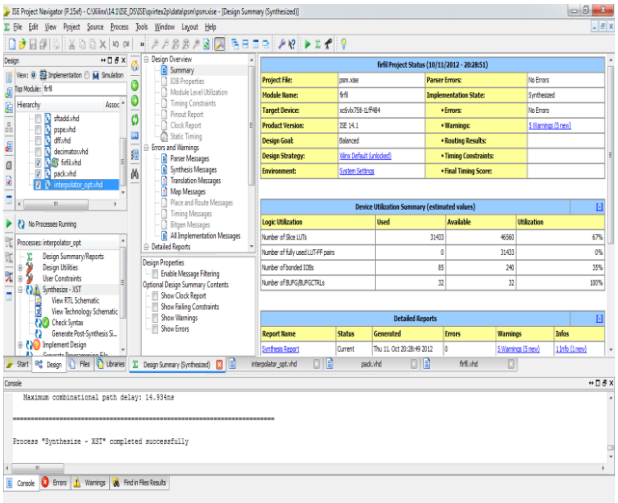

Fig 10. synthesis report of interpolator and decimator

The above figure shows the device summary of the multiple fault diagnosis top order module which represents the summary of the used 6301 IO Buffers, 512 flipflops ,LUTs ,buffers, inverters etc.

The synthesis result for the proposed algorithm is presented:

```
=====================================
*                Design Summary                *
=====================================
Top Level Output File Name         : firfil.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                      : 43803
#      GND                 : 65
#      INV                 : 128
#      LUT1                : 6
#      LUT2                 : 3202
#      LUT3                 : 12801
#      LUT4                 : 15296
#      MULT_AND             : 448
#      MUXCY                : 3850
#      MUXF5                : 4160
#      XORCY                : 3847
# FlipFlops/Latches         : 512
#      FD                  : 512
# Clock Buffers             : 32
#      BUFGP                : 32
# IO Buffers                : 6301
#      IBUF                : 4712
#      OBUF                : 1589
```

From the result it is observed that logical counts of 43803 Basic Element Logic (BEL) is required for the realization of DST processor, and the total memory usage of 254952 kilobytes.

B. RTL View

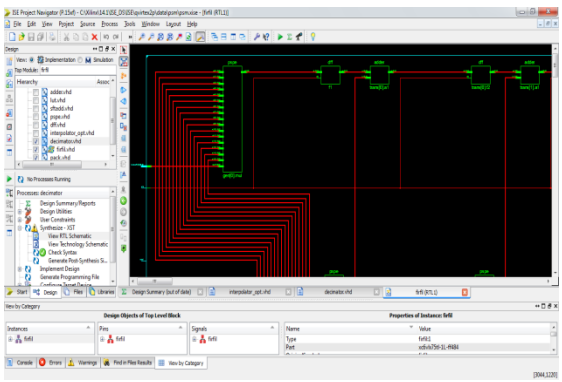

Fig 11.Shows the RTL schematic view of interpolator and decimator

The above figure shows the RTL schematic of the Interpolator and decimator RTL is an acronym for *register transfer level*. This implies that the source code VHDL/Verilog HDL describes how data is transformed as it is passed from register to register

C. Technology Schematic

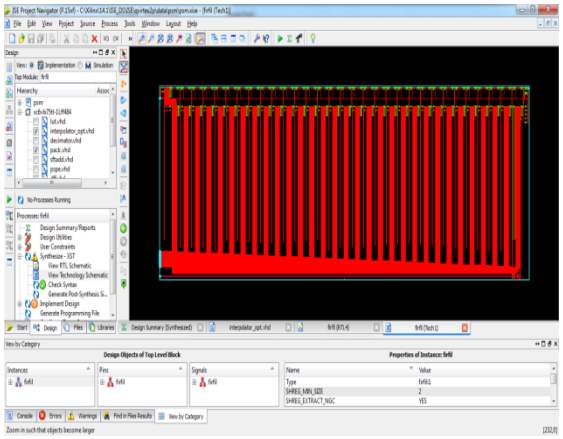

Fig 12 shows the Technical schematic view of interpolator and decimator

D. Implementation Observations

The implementation of proposed algorithm is illustrated in various pictorial views obtained during the process of realization i.e., Fig.10 Represents the Synthesis is a process of constructing a gate level net list from a model of a circuit described in VHDL. Fig.11 shows the RTL views of existing and proposed algorithms.Fig.12 shows the one of the technical schematic of targeted FPGA.

V. RESULT AND DISCUSSION

In this paper, we have presented a new high performance DA based LMS adaptive filter with multiple sampling rates by using interpolator and decimator. The proposed design uses W-LUT for storing OBC combinations of input signal.

The technique in this paper is also able to reduce the complexity by using decimator filter and interpolator filter circuits in DSP processing where multiple sampling rates are required.

Advantages:

The advantages of this work is that it Reduced complexity which in turn reduces the area, Power dissipation ,gives Higher throughput rate, Higher processing speed, Fast Computation, LFSR can rapidly transmit a sequence that indicates high-precision relative time offsets and many more. Hence the proposed design can be best suited for high performance adaptive filtering applications.

REFERENCES

[1] Dr. K. Babulu and Mohammad Shaffi," FPGA Implementation of Multi-Rate Reconfigurable Architectute with Low complexity FIR Filteers", IJETAE, vol 2,Issue 10,Sep 2012.

[2] Mohd Tasleem Khan and Shaik Rafi Ahamed,"A New Performance VLSI Architecture for LMS Adaptive Filter using Distributed Arithmetic" IEEE Computer Society Annual Symposium on VLSI,2017.

[3] R. Mahesh and A. P. Vinod, "A new common subexpression elimination algorithm for realizing low complexity higher order digital filters," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 2, pp.217–219, Feb. 2008.

[4] R. Mahesh and A. P. Vinod, " New Reconfigurable Architectures for Implementing FIR Filters with Low Complexity" *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 2, Feb. 2010.

[5] Altera corporation design for polyphase interpolation and decimation filters *FIR Compiler MegaCore Function User Guide*.

[6] A. Croisier, D. Esteban, M. Levilion, and V. Riso, "Digital filter for PCM encoded signals," Dec. 4 1973, uS Patent 3,777,130.
[7] D. J. Allred, H. Yoo, V. Krishnan, W. Huang, and D. V. Anderson, "LMS adaptive filters using distributed arithmetic for high throughput," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, no. 7,pp. 1327–1337, 2005.
[8] R. Guo and L. S. DeBrunner, "Two high-performance adaptive filter implementation schemes using distributed arithmetic," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 58, no. 9, pp. 600–604, 2011.
[9] M. Surya Prakash and R. A. Shaik, "High performance architecture for LMS based adaptive filter using distributed arithmetic."
[10] T. Saramaki, "A class of linear-phase FIR filters for decimation, interpolation, and narrow-band filtering,'' *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 32, pp. 1023-1036, Oct. 1984.

Authors Profile:

Mohammad Shaffi pursuing Ph.D in the Department of Instrument Technology, Andhra University, Andhra Pradesh, India. His research areas include VLSI Design, Embedded System Design and Image Processing.

Dr. Angajala Kamala Kumari, Assistant Professor, Department of Instrument Technology, Andhra University. Her research areas are Doppler Sodar, VLSI, Digital systems.