Novel Block-Formulation and Area-Delay-Efficient Reconfigurable Interpolation Filter Architecture for Multi-Standard SDR Applications

Basant Kumar Mohanty, Senior Member, IEEE

Abstract—A poly-phase based interpolation filter computation involves an input-matrix and coefficient-matrix of size $(P \times M)$ each, where $P$ is the up-sampling factor and $M = N/P$, $N$ is the filter length. The input-matrix and the coefficient-matrix resizes when $P$ changes. In this paper, we made an analysis of interpolation filter computation for different up-sampling factors to identify redundant computations and removed those by reusing partial results. Reuse of partial results eliminates the necessity of matrix resizing in interpolation filter computation. A novel block-formation is presented to share the partial results for parallel computation of filter outputs of different up-sampling factors. Using the proposed block formulation, a parallel multiplier-based reconfigurable architecture is derived for interpolation filter. The most remarkable aspect of the proposed architecture is that, it does not require reconfiguration to compute filter outputs of an interpolation filter for different up-sampling factor. The proposed structure has regular data-flow and it has no overhead complexity for its reconfigurable feature unlike the existing structures. Besides, the proposed structure has significantly less register complexity than the existing structure and its register complexity is independent of the block-size. Moreover, the proposed structure can support higher input-sampling frequency than the existing structure. ASIC synthesis result shows that the proposed structure for block-size 4, filter length 32, and up-sampling factor 8, involves 13.6 times more area and offers 245 times higher maximum input-sampling frequency compared with the existing multiplier-less structure. It involves 18.6 times less area-delay-product (ADP) and 9.5 times less energy per output (EPO) than the existing multiplier-less structure.

Index Terms—Interpolation filter, reconfigurable, digital-up converter, architecture, VLSI.

I. INTRODUCTION

INTERPOLATOR is used in digital signal processing (DSP) systems to increase the sampling rate digitally, and it comprises of an up-sampler and an anti-imaging (interpolation) filter. The up-sampler change the sampling rate of base-band signal, while the interpolation filter suppress the undesired interference effect resulted due to up-sampling [1] the base-band signal. Pulse shaping filters like root raised cosine (RRC) filter is commonly used as interpolation filter due to its high inter-symbol interference (ISI) rejection ratio and high bandwidth limitation criteria [2]. Interpolation filter has a different coefficient-vector for different up-sampling factors of a base-band signal.

Software defined radio (SDR) technology enables for digital implementation of wide band trans-receivers of multi-standard wireless communications [3]. A multi-standard SDR system involve interpolators with interpolation filters with different coefficients, filter length and up-sampling factors to meet the stringent specifications of different communication standards [3]. For example: Universal Mobile Telecommunication Standard (UMTS) uses interpolators with interpolation factors (4, 8 and 16), and filter lengths (25, 49, 97), respectively. When these interpolators are implemented individually in a hardwired circuit, then a huge amount of resource is spent. A reconfigurable finite impulse response (FIR) interpolation filter is the most appropriate for a resource and power constrained multi-standard SDR receiver which could support different up-sampling factors as well as filter specifications.

During last decade, several multiplier and multiplier-less designs have been suggested for efficient hardware realization of reconfigurable FIR filters and filter-banks for SDR channelization [4]–[16]. But, we do not find much work on reconfigurable interpolation filter architecture except a few. A single rate (fixed up-sampling factor) FIR interpolation filter can be implementation using a FIR filter. This could be the reason for non availability of any specific design in the literature for reconfigurable FIR interpolation filter. However, a single rate interpolation filter operate at $P$ times higher sampling rate than the input sampling frequency and requires $N$ filter parameters to compute each output, where $P$ is the up-sampling factor. On the other hand, a poly-phase based multi-rate interpolation filter operates at the input sampling rate and compute $P$ outputs using $P$ sub-filters each having $\lceil N/P \rceil$ filter parameters [1]. Therefore, a multi-rate interpolation filter structure is more hardware efficient than the single rate interpolation filter structure. The existing reconfigurable FIR filter structures are efficient for channelizer, but they do not offer an efficient computing structure for reconfigurable interpolation filter.

A few multiplier-less designs are proposed for interpolation filter [17], [18], [20]. Symmetric property of pulse shaping filter and a LUT decomposition scheme are used in [17] to reduce the area complexity of 1:4 interpolation filter. In [18], symmetries of coefficient and LUT are used along with LUT sharing of in-phase and quadrature-phase filters to save LUT words which offers a significant saving in area complexity of the interpolation filter. Both these designs implement the in-phase and quadrature-phase pulse shaping filters of QPSK modulator. They can not be reconfigured for up-sampling factor other than 4 and for different filter specifications. A distributed arithmetic (DA)-based reconfigurable FIR interpolation filter architecture is proposed in [19]. The DA-LUT stores partial results of all the sub-filter outputs of interpolation filter with three different interpolation factors. Therefore, the structure requires a large size DA-LUT which
is not favorable for single chip realization. Recently, Hatai et al. [20] have proposed a reconfigurable FIR interpolation filter design similar to [19] using LUT-less DA technique to reduce the area complexity. Coefficient-vector of the desired interpolation filter is selected using an array of multiplexers. The structure uses AND-gates, multiplexer and adders to implement the DA-LUT and computes a sub-filter output of the interpolation filter in bit-serial manner. It involves less area than the previous structures and supports base-band signal of low-sampling rates. Besides, the structure has a large overhead complexity (in terms of multiplexer and registers) for its reconfigurable feature.

An interpolation filter of up-sampling factor \( P \) and filter length \( N \) involves an input-matrix of size \((P \times M)\), where \([M = N/P]\). Input-matrix size of the interpolation filter changes for different filter-lengths and up-sampling factors. As a consequence of this, the reconfigurable interpolation filter architecture has irregular data-flow and the structure is not suitable for hardware realization. This problem is partially solved by assuming a constant filter length (equal to the length of the largest size filter) for the reconfigurable interpolation filter, where smaller size filters are realized using the same structure by zero padding. With this assumption, the input-matrix of an reconfigurable interpolation filter only re-sizes when \( P \) changes. For example: the size of the input-matrix of an interpolation filter for filter length \( N \) = 16 and up-sampling factors \( P \) = 2, 4 and 8, are respectively, \((2 \times 8)\), \((4 \times 4)\), and \((8 \times 2)\). Due to re-size of the input-matrix, the number of sub-filters of an interpolation filter changes with the column size \( P \) of the input-matrix where the length of each sub-filter is equal to the size of the row \( N/P \). The number of sub-filters required for reconfigurable interpolation filter for full-parallel realization is equal to the highest up-sampling factor. When a full-parallel reconfigurable interpolation structure is configured for lowest up-sampling factor, then maximum number of sub-filters remain unused in the parallel structure. Similarly, when the full-parallel structure is configured for highest up-sampling factor, then hardware resource of each sub-filter are partially utilized. Therefore, a full-parallel reconfigurable interpolation filter structure has a low hardware utilization efficiency (HUE). Computation of \( P \) sub-filters can be folded into a single sub-filter since they have identical computation. A folded reconfigurable interpolation filter structure has better HUE than the full-parallel structure. It involves only one sub-filter irrespective of up-sampling factor and computes one sub-filter output per cycle [20]. Therefore, the existing reconfigurable interpolation filter architectures uses a folded structure instead of a full-parallel structure. But, the folded interpolation filter structure has one major problem. Its output sampling frequency remain constant irrespective of the up-sampling factor and the input sampling frequency decreases inversely with the up-sampling factor. In other word, the folded structure does not increase the output sampling rate for higher up-sampling factor. Instead of that it decreases the input sampling rate by \( P \) times. Therefore, the folded reconfigurable interpolation filter architectures are suitable for base-band signals of low sampling frequency. On the other hand, the parallel interpolation filter structure increase the output sampling rate proportionately with \( P \), but the structure is not hardware efficient.

We observe that the existing reconfigurable interpolation filter architecture is derived using a straight-forward FIR filter design of [11]. Since, the reconfigurable interpolation filter has a different data-flow than the reconfigurable FIR filter, a different design approach need to be considered for interpolation filter. We do not find any such design approach in the literature. The interpolation filter algorithm need to be reformulated taking the data-flow of reconfigurable filter into consideration. Keeping this in mind, in Section II, we made an analysis on interpolation filter computation for different up-sampling factors to identify redundant computations. The key contribution of this paper are:

- Reuse of partial results in reconfigurable interpolation filter.
- A novel block-formulation is presented for efficient realization of reconfigurable interpolation filter.
- Parallel inner-product computations are performed using array-multiplication and addition to facilitate reuse of partial results in an interpolation filter.
- A parallel reconfigurable architecture is presented for area-delay and power efficient realization of interpolation filters.

The rest of this paper is organized as follows: Analysis of interpolation filter computation is presented in Section II. Block formulation of reconfigurable interpolation filter architecture is presented in Section III. Proposed architecture is presented in Section IV. Hardware complexity and performance comparison is presented in Section V. Conclusion is presented in Section VI.

II. ANALYSIS OF INTERPOLATION FILTER COMPUTATION

The interpolation filter is comprised of an up-sampler and a pulse shaping filter (PSF). Suppose an input signal \( x(n) \) is up-sampled by factor \( P \). The up-sampled sequence \( \{u(m) = x(n)\} \) for \( m = nP \), otherwise \( u(m) = 0 \) is filtered by the PSF of length \( N \). The output of an interpolation filter is computed by the relation

\[
y(m) = \sum_{i=0}^{N-1} h(i) \cdot x(m - i)
\]  

Using poly-phase decomposition, (1) can be expressed as

\[
y(Pn - p) = \sum_{i=0}^{M-1} h(Pi + p) \cdot x(n - Pi - p)
\]  

where \( m = Pn, 0 \leq p \leq P - 1 \), and \( N = MP \).

An interpolation filter of up-sampling factor \( P \) produce \( P \) outputs for each sample of \( x(n) \). Using poly-phase decomposition of (1), \( P \) outputs of the interpolation filter are computed by \( P \) parallel sub-filters of length \( M \) each. Computation of \( P \) parallel sub-filters can be expressed in matrix form as

\[
y_n = X_n \oplus H
\]  

\[
y_n = X_n \oplus H
\]  

SUBMITTED FOR IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-I, REGULAR PAPERS 2
where, $\oplus$ denotes the inner-product of $i$-th row of $X_n$ with $i$-th row of $H$. The input-matrix $X_n$, weight-matrix $H$ and output-vector $y_n$ are defined as

$$X_n = \begin{bmatrix}
x(n) & x(n-P) & \ldots & x(n-N+P) \\
x(n-1) & x(n-P-1) & \ldots & x(n-N+P-1) \\
\vdots & \vdots & \ddots & \vdots \\
x(n-P+1) & x(n-2P+1) & \ldots & x(n-N+1)
\end{bmatrix}$$

$$H = \begin{bmatrix}
h(0) & h(P) & \ldots & h(N-P) \\
h(1) & h(P+1) & \ldots & h(N-P+1) \\
\vdots & \vdots & \ddots & \vdots \\
h(P-1) & h(2P-1) & \ldots & h(N-1)
\end{bmatrix}$$

$$y_n = [y(Pn), y(Pn-1), \ldots, y(Pn-P+1)]^T$$

Matrix-vector product of (3) can otherwise be expressed as:

$$y_n = [s_{n,0} \ s_{n,1} \ \ldots \ s_{n,M-1}] \odot [c_0 \ c_1 \ \ldots \ c_{M-1}]$$

where, $\odot$ denotes the vector operation $\sum_{i=0}^{M-1} s_{n,i} \odot c_i$, and $\odot$ denotes the array-multiplication. $s_{n,i}$ and $c_i$ are the $i$-th column-vector of $X_n$ and $H$, respectively.

The array-multiplication of $X$ with $H$ i.e., $(X \odot H)$ produces $N$ partial results ($u_i$), and they are added separately to compute $P$ sub-filter outputs, where $u_i = x(n-i) \cdot h(i)$ for $(0 \leq i \leq N-1)$. In a particular computation cycle, $X$ does not change its value, but change its size for different $P$. Similarly, $H$ does not change its value for a particular PSF but change its size for different $P$. Interestingly, both $X$ and $H$ re-sizes in the same manner when $P$ changes. Therefore, $N$ partial values obtained from $(X \odot H)$ are remain unchanged when $P$ changes. However, these $N$ partial results are added in different sets to produce the sub-filter outputs of interpolation filter of different $P$. To illustrate this we have considered a PSF of length $N = 8$ for two different up-sampling factors $P = 2$, and 4. The input-matrix and coefficient-matrix of the PSF for $P = 2$ and 4, and the corresponding partial results of $(X \odot H)$ are shown in Fig.1. As shown in Fig.1, $(X \odot H)$ for $P = 2$ and 4, produce two sets of 8 partial results each, and these two sets are identical. Out of the two sets, one set of partial values form a redundant set. Computation of the redundant set could be avoided and partial results of one set can be reused for other. In general, out of $q$ sets of partial results corresponding to $q$ different up-sampling factors, $(q-1)$ sets are redundant. These redundant sets can be avoided by reusing partial values of one set for $(P-1)$ times. Generation of redundant sets (of partial results) involve matrix resizing and array multiplication of re-sized matrices. Resizing the matrices $X$ and $H$ in a hardware structure is resource and time consuming. Reuse of partial values eliminates the necessity of matrix resizing. Also, it offers a saving of $(q-1)N$ multiplications when filter computation is performed for $q$ different up-sampling factors. Therefore, reuse of partial results not only makes the reconfigurable interpolation filter architecture simple, it also favors parallel computation of interpolation filter outputs of different up-sampling factors without performing extra computation.

$$\begin{bmatrix}
u_0 & u_2 & u_4 & u_6 \\
u_1 & u_3 & u_5 & u_7
\end{bmatrix} = \begin{bmatrix}
x(n) & x(n-2) & x(n-4) & x(n-6) \\
x(n-1) & x(n-3) & x(n-5) & x(n-7)
\end{bmatrix} \odot \begin{bmatrix}
h(0) & h(2) & h(4) & h(6) \\
h(1) & h(3) & h(5) & h(7)
\end{bmatrix}$$  

$$\begin{bmatrix}
u_0 & u_2 & u_4 & u_6 \\
u_1 & u_3 & u_5 & u_7
\end{bmatrix} = \begin{bmatrix}
x(n) & x(n-1) & x(n-3) & x(n-7) \\
x(n-2) & x(n-4) & x(n-6) & x(n-8)
\end{bmatrix} \odot \begin{bmatrix}
h(0) & h(4) \\
h(1) & h(5) \\
h(2) & h(6) \\
h(3) & h(7)
\end{bmatrix}$$

Fig. 1. (a) The $n$-th cycle input-matrix $(X)$, coefficient-matrix $(H)$ and partial values $(u_i)$ of $(X \odot H)$ for $P = 2$ and $N = 8$, where $u_i = x(n-i) \cdot h(i)$ for $(0 \leq i \leq 7)$. (b) $X$, $H$ and partial values $(u_i)$ of $(X \odot H)$ for $P = 4$ and $N = 8$.

Fig. 2. Full-parallel reconfigurable interpolation filter structure where partial results are shared

A reconfigurable interpolation filter structure is shown in Fig.2, where the partial results are shared for different $P$. The partial result generation unit produces $N$ partial values of $(X \odot H)$. These partial results are distributed into $P$ groups of $M$ partial results each, where $M = N/P$, and re-group these partial results when $P$ changes. The data-selector unit select partial results from the partial result generation unit and distribute them into $P$ groups. Finally, each group of partial results are added in separate adder tree (AT) to compute a sub-filter output, and $P$ ATs are required to compute $P$ sub-filter outputs corresponding to up-sampling factor $P$. The reconfigurable adder unit uses separate sets of ATs for different values of $P$. The reconfigurable adder unit is configured to select a set of ATs corresponding to a particular $P$. The complexity of the reconfigurable adder unit is $(qN - \beta)$ and the complexity of data selector unit is $N(q-1) \times 2:1$ MUXes, where $\beta = p_1 + p_2 + p_3 + \ldots + p_q$, $p_i$ (for $1 \leq i \leq q$) represent $q$ different up-sampling factors of an interpolation filter. For example: $N = 16$, $p_1 = 2$, $p_2 = 4$ and $p_3 = 8$, the adder unit involves 34 adders and the data selector unit involves 32 number of $2:1$ MUXes. Since, reuse of partial result favors parallel computation of interpolation filter outputs for different up-sampling factors, the data-selector unit can be avoided in the reconfigurable architecture without any extra cost. Overall,
a parallel reconfigurable architecture can be designed using the partial result generation unit and the reconfigurable adder unit. We find that using the block-processing scheme the reconfigurable adder unit can be replaced by a fixed adder unit comprising of \( N \) adders. Besides, the block-processing scheme offers register sharing as well [22]. Therefore, the area delay efficiency of the reconfigurable interpolation filter could be improved significantly using the block-processing scheme. Keeping this in mind, we have presented a novel block processing scheme in Section III for the reconfigurable interpolation filter.

### III. Block-Formulation of FIR Interpolation Filter

Let us consider an FIR interpolation filter of up-sampling factor \( P \) processes a block of \( L \) input samples and generates \( P \) filter output blocks of size \( L \) each in every cycle. Computation of the \( k \)-th cycle filter outputs is represented in matrix form as:

\[
\begin{align*}
\begin{bmatrix}
y_{k,0}^1 & y_{k,1}^1 & \cdots & y_{k,L-1}^1
\end{bmatrix}^T &=
\begin{bmatrix}
s_{k,0}^0 & s_{k,1}^0 & \cdots & s_{k,M-1}^0
\end{bmatrix}
\begin{bmatrix}
c_0 & c_1 & \cdots & c_{M-1}
\end{bmatrix}
\begin{bmatrix}
y_{k,0} & y_{k,1} & \cdots & y_{k,L-1}
\end{bmatrix}
\end{align*}
\]

where

\[
s_{k,i} = [x(kL - l - Pi), \ldots, x(kL - l - Pi + P + 1)]^T
\]

\[
y_{k}^i = [y(PkL - lP), \ldots, y(PkL - lP - 1)]^T
\]

#### A. Analysis of Block Computation of Interpolation Filter

Substituting block-size \( L = 4 \), up-sampling factor \( P = 2 \) and filter-length \( N = 8 \) in (5), we have

\[
\begin{align*}
\begin{bmatrix}
y_{k}^0
y_{k}^1
y_{k}^2
y_{k}^3
\end{bmatrix} &=
\begin{bmatrix}
s_{k,0}^0 & s_{k,1}^0 & s_{k,2}^0 & s_{k,3}^0
s_{k,0}^1 & s_{k,1}^1 & s_{k,2}^1 & s_{k,3}^1
s_{k,0}^2 & s_{k,1}^2 & s_{k,2}^2 & s_{k,3}^2
s_{k,0}^3 & s_{k,1}^3 & s_{k,2}^3 & s_{k,3}^3
\end{bmatrix}
\begin{bmatrix}
c_0 & c_1 & c_2 & c_3
\end{bmatrix}
\begin{bmatrix}
y_{k,0} & y_{k,1} & y_{k,2} & y_{k,3}
\end{bmatrix}
\end{align*}
\]

where

\[
s_{k,i} = [x(4kL - l - 2i), x(4kL - l - 2i + 1)]
\]

\[
y_{k} = [y(8kL - 2l), y(8kL - 2l - 1)]
\]

\[
c_i = [h(2i), h(2i + 1)]
\]

Similarly, substituting \( L = 4 \), \( P = 4 \) and \( N = 8 \) in (5), we have

\[
\begin{align*}
\begin{bmatrix}
y_{k}^0
y_{k}^1
y_{k}^2
y_{k}^3
\end{bmatrix} &=
\begin{bmatrix}
r_{k,0}^0 & r_{k,1}^0 & r_{k,2}^0 & r_{k,3}^0
r_{k,0}^1 & r_{k,1}^1 & r_{k,2}^1 & r_{k,3}^1
r_{k,0}^2 & r_{k,1}^2 & r_{k,2}^2 & r_{k,3}^2
r_{k,0}^3 & r_{k,1}^3 & r_{k,2}^3 & r_{k,3}^3
\end{bmatrix}
\begin{bmatrix}
d_0 & d_1
\end{bmatrix}
\begin{bmatrix}
v_{k,0} & v_{k,1} & v_{k,2} & v_{k,3}
\end{bmatrix}
\end{align*}
\]

where

\[
r_{k,i}^j = [x(4(4kL - l - 2i) + 4l), \ldots, x(4(4kL - l - 2i + 3))]
\]

\[
y_{k}^j = [y(16(4kL - l - 2i) + 4l), \ldots, y(16(4kL - l - 2i + 3))]
\]

\[
d_i = [h(2i), h(2i + 1)]
\]

Substituting (9a) in (9b), and (7c) in (9c), we have

\[
r_{k,i}^j = \begin{bmatrix} s_{k,2i+1}^j & s_{k,2i+2}^j \end{bmatrix}^T
\]

\[
d_i = [c_{2i}, c_{2i+1}]^T
\]

Rewriting (9b) in split form as

\[
y_{k}^j = [y_{k}^j, y_{k}^{j+1}]^T
\]

where

\[
y_{k}^j = [y(16(4kL - l - 2j) + 4l), y(16(4kL - l - 2j + 1))]^T
\]

for \( j = 0, 1 \) and \( l = 0, 1, 2, 3 \).

#### B. Modified Block Formulation for Reconfigurable Interpolation Filter Architecture

Column-vectors of input-matrix of (6) are derived from two input data-blocks \( x_k = \{x(4k), x(4k + 1), x(4k + 2), x(4k + 3)\} \) and \( x_{k-1} = \{x(4k - 4), x(4k - 5), x(4k - 6), x(4k - 7)\} \) corresponding to clock cycles \( k \) and \( (k-1) \), respectively. From the current input-block \( x_k \), a pair of input-vectors \( s_{k,0}^i \) and \( s_{k,1}^i \) are derived, while \( s_{k,2}^i \) and \( s_{k,3}^i \) are derived from the most recent past input-block \( x_{k-1} \). To incorporate these information, a different notation is used to represent the input-vectors of (6). The input-vector \( s_{k,2i+1}^j \) (for \( L = 4, P = 2 \) and \( 0 \leq l \leq L - 1, 0 \leq j \leq (N/4) - 1, i = 0, 1 \) is denoted as:

\[
v_{k-3,j,i}^l = s_{k,2i+1}^j
\]
Using (13), input-vectors \( \{ s_{k,0}^l, s_{k,1}^l, s_{k,2}^l, s_{k,3}^l \} \) are represented as \( \{ v_{k,0}^l, v_{k,1}^l, v_{k,-1,0}^l, v_{k,-1,1}^l \} \), respectively. Substituting (13) in (6) and expressed in decomposed form as:

\[
\begin{bmatrix}
    y_k^0 \\
    y_k^1 \\
    y_k^2 \\
    y_k^3
\end{bmatrix}
= \begin{bmatrix}
    v_{k,0}^0 \\
    v_{k,0}^1 \\
    v_{k,0}^2 \\
    v_{k,0}^3
\end{bmatrix}
\otimes
\begin{bmatrix}
    c_0 & c_2 \\
    c_0 & c_2 \\
    c_0 & c_2 \\
    c_0 & c_2
\end{bmatrix}
\]

\[
+ \begin{bmatrix}
    v_{k,1}^0 \\
    v_{k,1}^1 \\
    v_{k,1}^2 \\
    v_{k,1}^3
\end{bmatrix}
\otimes
\begin{bmatrix}
    c_1 & c_3 \\
    c_1 & c_3 \\
    c_1 & c_3 \\
    c_1 & c_3
\end{bmatrix}
\]  

(14)

Similarly, substituting (13) in (12) and expressed in split form as:

\[
\begin{bmatrix}
    y_k^{00} \\
    y_k^{10} \\
    y_k^{20} \\
    y_k^{30}
\end{bmatrix}
= \begin{bmatrix}
    v_{k,0}^0 \\
    v_{k,0}^1 \\
    v_{k,0}^2 \\
    v_{k,0}^3
\end{bmatrix}
\otimes
\begin{bmatrix}
    c_0 & c_2 \\
    c_0 & c_2 \\
    c_0 & c_2 \\
    c_0 & c_2
\end{bmatrix}
\]

(15a)

\[
+ \begin{bmatrix}
    v_{k,1}^0 \\
    v_{k,1}^1 \\
    v_{k,1}^2 \\
    v_{k,1}^3
\end{bmatrix}
\otimes
\begin{bmatrix}
    c_1 & c_3 \\
    c_1 & c_3 \\
    c_1 & c_3 \\
    c_1 & c_3
\end{bmatrix}
\]  

(15b)

Using the proposed block formulation, partial results of a set of interpolation filters with selected up-sampling factors can be reused. A set of interpolation filters must have a minimum filter length to take advantage of the proposed scheme. However, the minimum filter length is not a binding factor and it could be relaxed by zero padding. Table I lists different sets of interpolation filters according to their up-sampling factors and the minimum required filter length for each set. The input block size could be equal to the up-sampling factor of a given set.

### C. For Interpolation Filter P = 2, N = 16 and L = 4

\[
\begin{bmatrix}
    y_k^0 \\
    y_k^1 \\
    y_k^2 \\
    y_k^3
\end{bmatrix}
= \begin{bmatrix}
    v_{k,0}^0 \\
    v_{k,0}^1 \\
    v_{k,0}^2 \\
    v_{k,0}^3
\end{bmatrix}
\otimes
\begin{bmatrix}
    c_0 & c_4 \\
    c_0 & c_4 \\
    c_0 & c_4 \\
    c_0 & c_4
\end{bmatrix}
\]

\[
+ \begin{bmatrix}
    v_{k,-1,0}^0 \\
    v_{k,-1,0}^1 \\
    v_{k,-1,0}^2 \\
    v_{k,-1,0}^3
\end{bmatrix}
\otimes
\begin{bmatrix}
    c_2 & c_6 \\
    c_2 & c_6 \\
    c_2 & c_6 \\
    c_2 & c_6
\end{bmatrix}
\]  

(16)

### D. For Interpolation Filter P = 4, N = 16 and L = 4

\[
\begin{bmatrix}
    y_k^{00} \\
    y_k^{10} \\
    y_k^{20} \\
    y_k^{30}
\end{bmatrix}
= \begin{bmatrix}
    v_{k,0}^0 \\
    v_{k,0}^1 \\
    v_{k,0}^2 \\
    v_{k,0}^3
\end{bmatrix}
\otimes
\begin{bmatrix}
    c_0 & c_4 \\
    c_0 & c_4 \\
    c_0 & c_4 \\
    c_0 & c_4
\end{bmatrix}
\]

\[
+ \begin{bmatrix}
    v_{k,-1,0}^0 \\
    v_{k,-1,0}^1 \\
    v_{k,-1,0}^2 \\
    v_{k,-1,0}^3
\end{bmatrix}
\otimes
\begin{bmatrix}
    c_2 & c_6 \\
    c_2 & c_6 \\
    c_2 & c_6 \\
    c_2 & c_6
\end{bmatrix}
\]  

(17a)

\[
\begin{bmatrix}
    y_k^{01} \\
    y_k^{11} \\
    y_k^{21} \\
    y_k^{31}
\end{bmatrix}
= \begin{bmatrix}
    v_{k,1}^0 \\
    v_{k,1}^1 \\
    v_{k,1}^2 \\
    v_{k,1}^3
\end{bmatrix}
\otimes
\begin{bmatrix}
    c_1 & c_5 \\
    c_1 & c_5 \\
    c_1 & c_5 \\
    c_1 & c_5
\end{bmatrix}
\]

\[
+ \begin{bmatrix}
    v_{k,-1,1}^0 \\
    v_{k,-1,1}^1 \\
    v_{k,-1,1}^2 \\
    v_{k,-1,1}^3
\end{bmatrix}
\otimes
\begin{bmatrix}
    c_3 & c_7 \\
    c_3 & c_7 \\
    c_3 & c_7 \\
    c_3 & c_7
\end{bmatrix}
\]  

(17b)

where, \( y_{k}^{ij} = \{ y(16k - 8l - 2j), y(16k - 8l - 2j - 1) \} \) for \( 0 \leq l \leq 3 \), and \( 0 \leq j \leq 1 \).
E. For Interpolation Filter \( P = 8, N = 16 \) and \( L = 4 \)

\[
\begin{align*}
\begin{bmatrix}
y_{000}^k \\
y_{100}^k \\
y_{200}^k \\
y_{300}^k \\
y_{010}^k \\
y_{110}^k \\
y_{210}^k \\
y_{310}^k \\
y_{001}^k \\
y_{101}^k \\
y_{201}^k \\
y_{301}^k \\
y_{011}^k \\
y_{111}^k \\
y_{211}^k \\
y_{311}^k
\end{bmatrix}
&= \begin{bmatrix}
v_0^k,0 \\
v_1^k,0 \\
v_2^k,0 \\
v_3^k,0 \\
v_0^{k-1},0 \\
v_1^{k-1},0 \\
v_2^{k-1},0 \\
v_3^{k-1},0 \\
v_0^k,1 \\
v_1^k,1 \\
v_2^k,1 \\
v_3^k,1 \\
v_0^{k-1},1 \\
v_1^{k-1},1 \\
v_2^{k-1},1 \\
v_3^{k-1},1
\end{bmatrix} \otimes 
\begin{bmatrix}
c_0 \\
c_0 \\
c_0 \\
c_0 \\
c_2 \\
c_2 \\
c_2 \\
c_2 \\
c_1 \\
c_1 \\
c_1 \\
c_1 \\
c_3 \\
c_3 \\
c_3 \\
c_3
\end{bmatrix}
\end{align*}
\]  
(18a)

where, \( y_{ij}^{(l)} = \{y(32j - 8l - 4i - 2j), y(32j - 8l - 4i - 2j - 1)\} \) for \( 0 \leq l \leq 3, 0 \leq i \leq 1 \) and \( 0 \leq j \leq 1 \).

Block formulations for other sets of up-sampling factors of Table I can also be derived in the similar form as (16), (17) and (18). Using the modified block formulation, we have derived a parallel reconfigurable architecture for interpolation filter. The proposed architecture has many interesting features. These are summarized here.

1) It computes filter output of multiple up-sampling factors in parallel without using any additional resource or computation time.
2) It does not require reconfiguration to compute filter outputs of the interpolation filter for different up-sampling factors. A different coefficient-vector is selected from the coefficient storage unit to configure the proposed architecture for a different filter specification.
3) It does not involve any overhead complexity for its reconfigurable feature unlike the existing structures.
4) It has the same register complexity as the conventional parallel interpolation filter structure and its register complexity is independent of the block-size.

IV. PROPOSED RECONFIGURABLE ARCHITECTURE

Computation of (16), (17) and (18) are mapped to a parallel architecture for realization of an interpolation filter of up-sampling factors \( p_1 = 2, p_2 = 4 \) and \( p_3 = 8 \) (IF2, IF4 and IF8). The proposed reconfigurable architecture is shown in Fig.3. It has three main units, (i) coefficient selection unit (CSU), (ii) input-vector generation unit (VGU), and (iii) arithmetic-unit. The CSU is comprised of \( N \) number of J:1 MUXes or \( N \) number of ROM LUTs of depth \( J \) words each, where \( N \) is the filter length and \( J \) is the number of interpolation filters of different coefficient vector to be realized in the reconfigurable architecture. The MUX-based CSU is used for \( J \leq 4 \) to avoid longer critical path delay due to MUX, and the ROM-based CSU is preferred for \( J > 4 \). The required coefficient-vector of a particular interpolation filter is selected in one cycle from the CSU.

The VGU receives one input-block in each cycle and generates \( (N/p_1) \) input-vectors of size \((Lp_1)\) each in parallel, where \( p_1 \) is the smallest up-sampling factor from a set of \( q \) different up-sampling factors to be realized in the reconfigurable architecture. Internal structure of the VGU is shown in Fig.4 for \( p_1 = 2, N = 16 \) and \( L = 4 \). It is comprised of \( (N - 1) \) registers. The VGU receives a block of input in every cycle and produces 8 data-vectors \( \{s_{k,0}, s_{k,1}, s_{k-1,0}, s_{k-1,1}, s_{k-2,0}, s_{k-2,1}, s_{k-3,0}, s_{k-3,1}\} \), where \( s_{k-i,j} = v_k^{0-i-j}, v_k^{0-i-j}, v_{k-1-i-j}, v_{k-1-i-j}, v_{k-2-i-j}, v_{k-2-i-j}, v_{k-3-i-j}, v_{k-3-i-j} \), and \( v_{k-j,i} = s_{k-2j+i} \). Input-samples \( \{x(4k), x(4k-1), x(4k-2), x(4k-3)\} \) and 15 recent past samples \( \{x(4k-4), x(4k-5), \cdots x(4k-17), x(4k-18)\} \) are distributed in overlap manner to obtain all the required data-vectors.

Fig. 3. Proposed reconfigurable architecture for a set of interpolation filters of up-sampling factors \( p_1 = 2, p_2 = 4, p_3 = 8 \), filter length \( N = 16 \) and input-block size \( L = 4 \).

The structure of AU is shown in Fig.5 for a set of interpolation filters IF2, IF4 and IF8 with up-sampling factor \( p_1 = 2, p_2 = 4, p_3 = 8 \), block-size \( L = 4 \) and filter length \( N = 16 \). It is comprised of \( [(N/p_1) = 8] \) multiplier units (MUs) and \( [(N/p_1 - 1) = 7] \) adder-units (ADU). Each MU receives an \((Lp_1)\)-point input-vector \( s_{k-j,i} \) from the VGU and a short \( p_1 \)-point coefficient-vector \( c_{n,m} \) from the CSU, and calculates one partial filter output-vector \( z_{k,m} \).
of size \((N/p_1)\), for \(0 \leq j \leq L - 1\), \(0 \leq i \leq p_1 - 1\), and \(0 \leq m \leq Lp_1 - 1\). Internal structure of the MU and ADU are shown in Fig.6. As shown in Fig.5, partial output-vectors \((z_{k,0}, z_{k,1}), (z_{k,1}, z_{k,5}), (z_{k,2}, z_{k,6})\) and \((z_{k,0}, z_{k,4})\) are added in four separate ADUs \((ADU_1, ADU_2, ADU_3, ADU_4)\) to compute filter output-blocks \(\{Y_{k,0} \ldots Y_{k,10}\}\) of IF\(_8\), where \(Y_{k,j} = [y_{k,j}^{(0)}, y_{k,j}^{(1)}, y_{k,j}^{(2)}, y_{k,j}^{(3)}]\), and \(y_{k,j}^{(i)} = \{(y(32k - 8i - 4j - 2), y(32k - 8i - 4j - 2j - 3))\), for \(0 \leq l \leq 3\), \(i = 0, 1\), and \(j = 0, 1\). Interestingly, the output-vectors \((Y_{k,0}^{(0)}, Y_{k,1}^{(1)}, Y_{k,10}^{(0)}, Y_{k,11}^{(1)})\) also represent the partial filter output of IF\(_4\), and these partial output-vectors are added in \(ADU_5\) and \(ADU_6\) to obtain the complete filter output vectors \((Y_{k}^{(0)}, Y_{k}^{(1)})\) of IF\(_4\), where \(Y_{k} = [y_{k}^{(0)}, y_{k}^{(1)}, y_{k}^{(2)}, y_{k}^{(3)}]\), and \(y_{k}^{(j)} = \{(y(16k - 8m - 2j), y(32k - 8l - 2j - 1))\), for \(0 \leq j \leq 3\) and \(j = 0, 1\). Similarly, the output-vectors \((Y_{k}^{(0)}, Y_{k}^{(1)})\) also represent the partial filter outputs of IF\(_2\), and they are added in \(ADU_7\) to obtain the complete filter output vector \((Y_{k})\) of IF\(_2\), where \(Y_{k} = [y_{k}^{(0)}, y_{k}^{(1)}, y_{k}^{(2)}, y_{k}^{(3)}]\), and \(y_{k}^{(j)} = (y(8k - 2l), y(8k - 2l - 1))\), for \(0 \leq l \leq 3\).

V. HARDWARE-TIME COMPLEXITIES

The proposed architecture consists of one CSU, one VGU and one AU. The CSU is comprised of either \(N\) number of \(J : 1\) MUXes or the same number of ROM LUTs of depth \(J\) words each. The CSU complexity of all the reconfigurable FIR structures is the same for a given value of \(J\). Therefore, we have excluded CSU of all the competing designs for performance comparison. The VGU is comprised of \((N - 1)\) registers, where the AU is comprised of \((N/p_1)\) MUs and \([(N/p_1) - 1]\) ADUs. Each MU is further comprised of \((Lp_1)\) adders. The AU involves \(NL\) multipliers and \(L(N - p_1)\) adders. Therefore, the core of the proposed reconfigurable interpolation filter architecture involves \(NL\) multipliers, \(L(N - p_1)\) adders and \((N - 1)\) registers. The proposed architecture is configured for a specific pulse shaping filter and compute filter outputs corresponding to all the up-sampling factors of a given set irrespective of the choice. In every cycle, it computes \(Lp_1, Lp_2, Lp_3\) parallel filter outputs corresponding to the up-sampling factors \(p_1, p_2, p_3\), respectively. Therefore, it computes \(L(p_1 + p_2 + p_3)\) filter outputs in every cycle, where a cycle period \(T = T_M + T_A + T_{FA}(\log_2(N/p_1) - 1)\), where \(T_M, T_A,\) and \(T_{FA}\) are respectively, one multiplier delay, one adder delay, and one full-adder delay. It is interesting to note that, the hardware complexity of the proposed structure is independent of up-sampling factor and compute filter outputs of all up-sampling factors.

A. PERFORMANCE COMPARISON

A few reconfigurable interpolation filter architectures are proposed in the literature. Recently, a multiplier-less reconfigurable architecture is proposed in [20]. We have extracted an equivalent multiplier-based design from the multiplier-less...
TABLE II

<table>
<thead>
<tr>
<th>Structures</th>
<th>Multiplier</th>
<th>Adder</th>
<th>ROM (word)</th>
<th>Register</th>
<th>MUX/DMUX</th>
<th>AND-gate (word)</th>
<th>CP</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>Son et al [18]</td>
<td>$N/P$</td>
<td></td>
<td>$2(N/2P-1)$</td>
<td>$N+1$</td>
<td>$P$</td>
<td>$0$</td>
<td>$T_1$</td>
<td>$2/(wT_1)$</td>
</tr>
<tr>
<td>Hatai et al [20] (multiplier)</td>
<td>$N/p_1$</td>
<td>$N/p_1$</td>
<td>$0$</td>
<td>$N\alpha/p_3+1/p_1$</td>
<td>$N\alpha/p_3$</td>
<td>$N/p_1$</td>
<td>$T_2$</td>
<td>$1/(wT_1)$</td>
</tr>
<tr>
<td>Hatai et al [20] (multiplier)</td>
<td>$N/p_1$</td>
<td>$(N/p_1)-1$</td>
<td>$0$</td>
<td>$N\alpha/p_3$</td>
<td>$N\alpha-4/p_3$</td>
<td>$0$</td>
<td>$T_3$</td>
<td>$1/T_2$</td>
</tr>
<tr>
<td>Proposed (p_1)</td>
<td>$NL$</td>
<td>$L(N-p_1)$</td>
<td>$0$</td>
<td>$N-1$</td>
<td>$0$</td>
<td>$0$</td>
<td>$T_4$</td>
<td>$Lp_1/T_3$</td>
</tr>
<tr>
<td>Proposed (p_2)</td>
<td>$NL$</td>
<td>$L(N-p_1)$</td>
<td>$0$</td>
<td>$N-1$</td>
<td>$0$</td>
<td>$0$</td>
<td>$T_4$</td>
<td>$Lp_2/T_3$</td>
</tr>
<tr>
<td>Proposed (p_3)</td>
<td>$NL$</td>
<td>$L(N-p_1)$</td>
<td>$0$</td>
<td>$N-1$</td>
<td>$0$</td>
<td>$0$</td>
<td>$T_4$</td>
<td>$Lp_3/T_3$</td>
</tr>
</tbody>
</table>

\[ \alpha = 2p_1 + p_2 + p_3, T_1 = T_A + 3T_{MX} + T_{MR}, T_2 = 2T_A + T_{MX} + T_{AND}, T_3 = T_M + T_A + T_{FA}(\log_2(N/p_1)-1) + T_{MX}, T_4 = T_M + T_A + T_{FA}(\log_2(N/p_1)-1) + T_{MX} \]

The proposed design involves $NL$ multipliers instead of $(N/p_1)$ AND-gates (word-level), nearly $(L/p_1)$ times more adders and $(\alpha/p_3)$ times less registers than those of the multiplier-based design of [20], and it offers more than $wL$ times higher throughput, where $w$ is the input bit-width. Compared with [18], the proposed structure involves nearly $NL$ multipliers and adders instead of $(2(N/2P-1))$ ROM words and offers $(wL/2)$ times more throughput rate. However, the proposed structure can be reconfigured for three different up-sampling factors where the structure of [18] is for up-sampling factor 4 only.

We can find from Table II that the existing reconfigurable interpolation filter architectures have constant throughput rate (output sampling rate) where the throughput rate of the proposed design increases proportionately with the up-sampling factor. This is mainly due to the fact that the existing design [20] calculates only one sub-filter output in each cycle. A poly-phase structure of an interpolation filter (with up-sampling factor $P$) has $P$ number of sub-filters. Therefore, the multiplier-based design of [20] takes $P$ cycles to compute a set of $P$ sub-filter outputs corresponding to each input sample, where the multiplier-less design takes $Pw$ cycles to compute the same set of $P$ sub-filter outputs. Therefore, the maximum input sampling rate (MISR) and output sampling rate (OSR) of multiplier-less and multiplier-based structures of [20] are $\{(1/wPT),(1/wT)\}$, and $\{(1/PT),(1/T)\}$, respectively, where $T$ is the cycle period. However, the proposed design computes $LP$ sub-filter outputs corresponding to a block of $L$ input samples in each cycle. The MISR and OSR of the proposed design is $(L/T)$ and $(LP/T)$, respectively. In other word, the input sampling rate of the proposed design is independent of $P$, while it decreases proportionately with $P$ in case of existing structure [20]. The MISR of the proposed design can be increased flexibly by changing the input-block size where in case of [20], it is solely depend on the cycle period. The proposed design has higher MISR than the existing design [20]. It does not involves multipliers unlike the existing design to reconfigure the structure for different up-sampling factors. The register complexity of the proposed structure is independent of input blocks-size and up-sampling factor where it increases proportionately with $\alpha$ in existing design. The register and MUX saving offers a significant reduction of area and power consumption in the proposed design.

We have made a theoretical estimate of hardware and time complexities of the proposed design and the similar reconfigurable architectures of [20] for filter length $N = 32$, up-sampling factors ($p_1 = 2, p_2 = 4$, and $p_3 = 8$), and input bit-width $w = 12$. The estimated values are listed in Table III for comparison. As shown in Table III, the proposed structure for block-size 4 involves 8 times more multipliers and adders, nearly 2 times less registers than those of the multiplier-based design of [20] and it supports 8 times, 16 times and 32 times higher input sampling rate than the design of [20] for $p_1 = 2, p_2 = 4,$ and $p_3 = 8$, respectively. Compared with the multiplier-less design of [20], the proposed one involves 128 multipliers against 16 AND-gates, 7 times more adders, nearly 2.6 times less registers, and it has 96 times, 192 times and 384 times higher input sampling rate than the multiplier-less design of [20] for $p_1 = 2, p_2 = 4,$ and $p_3 = 8$, respectively.

**B. Synthesis Results**

To validate the proposed design, we have coded it in VHDL for filter length $N = 16$ and $32$ with block-size $4$. We have also coded both the multiplier and multiplier-less designs of [20] for the same filter lengths. We have considered input signal width $w = 12$ and used a post-truncated fixed-width Booth multiplier. All the designs are synthesized by Synopsys Design Compiler using TMSC 65nm CMOS library. The results obtained from synthesis reports generated by Design Compiler are listed in Table IV.
As shown Table IV, multiplier-less structure of [20] has the lowest minimum clock period (MCP) than the multiplier-based structures due to absence of multiplier in the critical path. The MCP of proposed structure and the multiplier-based structure of [20] are equal or marginally differ due to $T_{MX}$. In spite of 8 times higher multiplier complexity than the multiplier-based structure of [20] the proposed structure involves 5.95 times more area on average for different filter lengths. This is mainly due to the register saving. The propose structure dissipates nearly 31 times and 24 times more power than the multiplier-based structure of [20] for filter length 16 and 32, respectively. The higher power dissipation is due to extra arithmetic (multiplier and adders) complexity and higher input sampling frequency than the existing structure of [20].

We have estimated the MISR using MCP and the formula given in Table III. The estimated values are listed in Table V. As shown in Table V, the MISR of [20] is decreasing with the up-sampling factor, while in case of the proposed structure, it is independent of up-sampling factor and increases proportionately with the input block-size. The multiplier-less design of [20] performs filter computation bit-serially and it has the lowest MISR than others. From Table III and Table V, we can find that the proposed structure involves 5.95 times more area, and it has 8.3 times, 16.7 times and 33 times higher MISR than those of the multiplier-based design of [20] for up-sampling factor 2, 4 and 8, respectively, on average for different filter sizes. Compared with the multiplier-less design of [20], the proposed one involves 13.6 more area and offers 62 times, 124 times and 245 times more MISR for up-sampling factor 2, 3 and 8 on average for different filter sizes.

As shown Table IV, multiplier-less structure of [20] has the lowest minimum clock period (MCP) than the multiplier-based structures due to absence of multiplier in the critical path. The MCP of proposed structure and the multiplier-based structure of [20] are equal or marginally differ due to $T_{MX}$. In spite of 8 times higher multiplier complexity than the multiplier-based structure of [20] the proposed structure involves 5.95 times more area on average for different filter lengths. This is mainly due to the register saving. The propose structure dissipates nearly 31 times and 24 times more power than the multiplier-based structure of [20] for filter length 16 and 32, respectively. The higher power dissipation is due to extra arithmetic (multiplier and adders) complexity and higher input sampling frequency than the existing structure of [20].

We have estimated area-delay-product (ADP) and energy

### Table III

<table>
<thead>
<tr>
<th>Structures</th>
<th>Filter length (N)</th>
<th>up-sampling (P)</th>
<th>MULT</th>
<th>ADD</th>
<th>REG</th>
<th>MUX</th>
<th>AND gate</th>
<th>CP</th>
<th>MISR ($f_o$)</th>
<th>MOSR ($f_o$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hatai et al [20] (multiplier-less)</td>
<td>32</td>
<td>2</td>
<td>0</td>
<td>16</td>
<td>80</td>
<td>48</td>
<td>16</td>
<td>$T_1 = 2T_A + 3T_{FA} + T_{MX} + T_{AND}$</td>
<td>$1/(12T_1)$</td>
<td>$1/(12T_1)$</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>4</td>
<td>16</td>
<td>15</td>
<td>64</td>
<td>48</td>
<td>0</td>
<td>$T_2 = T_M + T_A + 3T_{FA} + T_{MX}$</td>
<td>$1/(4T_2)$</td>
<td>$1/T_2$</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>8</td>
<td>128</td>
<td>112</td>
<td>31</td>
<td>0</td>
<td>0</td>
<td>$T_3 = T_M + T_A + 3T_{FA}$</td>
<td>$4/T_3$</td>
<td>$8/T_3$</td>
</tr>
</tbody>
</table>

### Table IV

<table>
<thead>
<tr>
<th>Structures</th>
<th>Filter length (N)</th>
<th>MCP (ns)</th>
<th>Area ($\mu$m$^2$)</th>
<th>Power (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hatai [20] (multiplier-less)</td>
<td>16</td>
<td>0.76</td>
<td>8810.28</td>
<td>2.8519</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>0.85</td>
<td>18175.68</td>
<td>4.8844</td>
</tr>
<tr>
<td>Hatai [20] (multiplier)</td>
<td>16</td>
<td>1.20</td>
<td>20893.68</td>
<td>7.9023</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>1.40</td>
<td>39803.4</td>
<td>14.3661</td>
</tr>
<tr>
<td>Proposed (L = 4)</td>
<td>16</td>
<td>1.12</td>
<td>127936.8</td>
<td>89.6727</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>1.38</td>
<td>230512.68</td>
<td>121.3106</td>
</tr>
</tbody>
</table>

### Table V

<table>
<thead>
<tr>
<th>Structures</th>
<th>Filter length (N)</th>
<th>Max. input-sampling freq. (MHz)</th>
<th>$p_1 = 2$</th>
<th>$p_2 = 4$</th>
<th>$p_3 = 8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hatai [20] (multiplier-less)</td>
<td>16</td>
<td>54.82</td>
<td>27.41</td>
<td>13.75</td>
<td></td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>49.01</td>
<td>24.50</td>
<td>12.25</td>
<td></td>
</tr>
<tr>
<td>Hatai [20] (multiplier)</td>
<td>16</td>
<td>416.67</td>
<td>208.38</td>
<td>104.19</td>
<td></td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>357.14</td>
<td>178.57</td>
<td>89.28</td>
<td></td>
</tr>
<tr>
<td>Proposed (L = 4)</td>
<td>16</td>
<td>3571.43</td>
<td>3571.43</td>
<td>3571.43</td>
<td></td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>2898.55</td>
<td>2898.55</td>
<td>2898.55</td>
<td></td>
</tr>
</tbody>
</table>

#### Fig. 7. Comparison of area-delay-product (ADP).

#### Fig. 8. Comparison of energy per output (EPO).
per output (EPO) of all the designs. The estimated ADP and EPO values are shown in Fig.7 and Fig.8 for N = 32. As shown in these figures, the proposed structure has significantly less ADP and EPO than the existing structure. It has 5.6 times less ADP and 3.9 times less EPO than those of multiplier-based design of [20] for up-sampling factor 8. Compared with the multiplier-less design of [20], the proposed design has 18.6 times less ADP and 9.5 times less EPO.

VI. CONCLUSION

In this paper, we made an analysis of interpolation filter computation for different up-sampling factors to identify redundant computations and removed those by reusing partial results. Reuse of partial results eliminates the necessity of matrix resizing in interpolation filter computation. A novel block-formulation is presented to share the partial results for parallel computation of filter outputs of different up-sampling factors. Using the proposed block formulation, a parallel reconfigurable architecture is derived for interpolation filter. The most remarkable aspect of the proposed architecture is that, it does not require reconfiguration to compute filter outputs of an interpolation filter for different up-sampling factors. The proposed structure has regular data-flow and it has no overhead complexity for its reconfigurable feature unlike the existing structures. Besides, the proposed structure has significantly less register complexity than the existing structure and its register complexity is independent of the block-size. Moreover, the proposed structure can support higher input-sampling frequency than the existing structure. ASIC synthesis result shows that the proposed structure for block-size 4, filter length 32, and up-sampling factor 8, involves 13.6 times more area and offers 245 times higher maximum input-sampling frequency compared with the existing multiplier-less structure [20]. It involves 18.6 times less ADP and 9.5 times less EPO than the existing structure of [20]. Due to less ADP, EPO and high input sampling frequency, the proposed reconfigurable architecture is a good candidate for efficient realization of digitally up-converters for multi-standard SDR applications.

REFERENCES