Fractional Sample Rate Conversion for Multi-Standard Radio Digital Front Ends

Gaelen Pereira and You Je Jung

Abstract—In order to support multiple wireless standards, it is desirable that radios have a reconfigurable front end that can select different frequency bands of interest. This flexibility is provided by pushing the ADC further up the receive chain, so that most of the processing is done in the Digital Front End (DFE). Fractional Sample Rate Conversion (FSRC) is an important component of DFEs since different sampling rates need to be supported for each standard while the ADC has a fixed clock rate. The aim of this project is to investigate the feasibility of different FSRC architectures, and to analyze the tradeoffs each architecture presents in the energy – delay – area space. Traditional architectures for sample rate conversion are throughput limited, while alternate architectures like time varying CIC filters and polynomial interpolation are area limited. However, architecture techniques can be used to trade performance for area efficiency and the use of pipelining and folding in this context is presented.

I. INTRODUCTION

Digital Front Ends (DFEs) are the key to realizing a radio that is truly flexible with respect to the bandwidth of communication. Moreover, as different communication standards call for varying bandwidths and therefore different sampling rates, an important component of a DFE is Fractional Sample Rate Conversion (FSRC). This is necessary because the Analog to Digital conversion generally takes place at a fixed clock rate as close to the antenna as possible. Ideally, fractional sample rate conversion can be realized by passing the signal through a Digital to Analog converter (DAC), filtering the analog signal to avoid subsequent aliasing and then resampling the signal at the desired sample rate. However, this cascade of the DAC and band-limiting filter can be realized by a digital filter if the ratio of sampling rates is a rational factor [1]. This is illustrated in Figure 1. Here the equivalent operation in the digital domain is interpolation by an integer factor followed by decimation by an integer factor, such that the ratio of the two is the rational factor by which the sample rate has to be converted. The interpolation operation requires an anti imaging filter at the output, and the decimation operation requires an anti aliasing filter at the input. Cascaded Integrator Comb (CIC) filters, which have been used for both anti-aliasing and anti-imaging filtering are a good candidate for these filters. Moreover, CIC filters are multiplier free and don’t require coefficient storage, therefore can be helpful to reduce the complexity of hardware implementation.

However, the problem in the architecture from Figure 1 is that the intermediate sampling rates that are required can be extremely high, especially in the context of mobile communications where the bandwidth to be digitized is extremely large. This means that the input sampling rate is already high, and on top of that the signal is generally oversampled for noise shaping purposes. Therefore, such an approach to fractional sample rate conversion is not feasible, and alternate architectures need to be explored.

One way to realize sample rate conversion is by using a time varying CIC filter [1]. Since interpolation involves only zero padding, a time varying filter can be used that eliminates the need to up-sample the input signal. While this reduces the burden on the CIC filter in terms of sampling rate, it introduces complex circuitry that entails an area cost.

Another way to implement fractional sample rate conversion is by using a polynomial interpolation filter [3]. By providing a fractional delay of the digital signal, the new sampling frequency can be generated without up-sampling the input signal. However there is an area cost associated with this implementation as well.

The aim of this project is to analyze the feasibility of each of these architectures for fractional sample rate conversion. Direct fractional sample rate conversion is infeasible due to high sampling rate requirement. The two techniques – using time varying CIC filters and polynomial interpolation can help solve this performance issue, but have complex circuitry associated with them. Therefore, architecture techniques can be used to trade performance for area / energy efficiency. Pipelining is used to reduce the critical delay path, while folding is used to reduce the area of these architectures. Section II presents the three architectures being compared, Section III presents the architecture techniques used to optimize the architectures, Section IV presents results and Section V concludes the report.

\[
\begin{align*}
\text{Interpolation by } L & \quad \text{Decimation by } M \\
\text{up-sampler} & \quad \text{down-sampler} \\
\text{anti-imaging filter} & \quad \text{anti aliasing filter}
\end{align*}
\]

Figure 1. Sample Rate Conversion with arbitrary rational factor L/M [1]

II. CANDIDATE ARCHITECTURES

A. Traditional sample rate converter (Arch 1)

Traditional methods for sample rate conversion by a rational factor L/M first up-sample the signal by L and then down-sample it by M. This is shown in Figure 2(a), where D represents a differentiator and I represents an integrator. The front part of this structure is a traditional interpolation filter with 2 stages and the latter part is a decimation filter with 2 stages. By cascading these two filters, fractional sample rate
conversion can be realized. This architecture might lead to extremely high sampling rates after the interpolation stage. Moreover, this problem is further complicated when fractional sample rate conversion with a factor close to 1 is desired. For example, sample rate conversion from 10.1 MHz to 10 MHz would require an up-sampling by a factor of 100 and subsequent down-sampling by a factor of 101. The filter in the intermediate stage will have to work at a clock rate of 1.01 GHz, which is problematic since the feedback loop in the integrator limits the maximum clock frequency.

B. Time varying CIC filter implementation [1] (Arch 2)

The fundamental idea behind the time varying CIC filter implementation is that it is possible to realize the interpolation followed by the filtering without up-sampling the signal. This is possible since up-sampling merely involves zero padding, therefore the states of the integrator \((z_1, \ldots, z_0)\) can be calculated at any time instant without computing all the intermediate states. The architecture is presented in Figure 3, where \(D\) represents a differentiator, \(I_i\) represents the product of an integrator and delay. The output can be computed in terms of the input and the integrator states using the equations below:

\[
\begin{align*}
\begin{bmatrix}
z_1(k+1) \\
\vdots \\
z_N(k+1)
\end{bmatrix} &= \begin{bmatrix}
1 & 0 & \cdots & 0 \\
\vdots & \ddots & \ddots & \vdots \\
L_{N,1} & \cdots & L_{N,N-1} & 1
\end{bmatrix} \begin{bmatrix}
z_1(k) \\
\vdots \\
z_N(k)
\end{bmatrix} + x(k) \\
y(m) &= \begin{bmatrix}
\rho_{N-1}(m) \\
\vdots \\
\rho_1(m)
\end{bmatrix} \begin{bmatrix}
z_1(k) \\
\vdots \\
z_N(k)
\end{bmatrix} + x(k)
\end{align*}
\]

The factors \(L_{n,j}\) and \(\rho_1(m)\) can be computed as

\[
L_{n,j} = \frac{(N-j)!(L-1+i-j)!}{(N-i)!(i-j)!(L-1)!}
\]

\[
\rho_1(m) = \prod_{h=1}^{i} \rho(m) + h
\]

The main implication from the hardware perspective is that \(L_{n,j}\) can be implemented using a constant gain, while \(\rho_1(m)\) needs to be calculated in real time using multipliers. Moreover, \(L_{n,j}\) depends on the factor by which sample rate conversion is desired. Also, the number of integrator stages and real time multipliers is \(O(N)\) for an \(N^{th}\) order filter.

C. Polynomial interpolation implementation [3] (Arch 3)

This technique involves digital interpolation to re-sample the incoming signal, as shown in Figure 4.

The interpolator processes the incoming input samples at \(\text{Clk}_{\text{old}}\) to compute the output samples at \(\text{Clk}_{\text{new}}\). The 3\(^{rd}\) order Taylor series polynomial is used to interpolate the output samples, according to the equation below:

\[
f(t + \alpha) = f(t) + \alpha f'(t) + \frac{\alpha^2}{2!} f''(t) + \frac{\alpha^3}{3!} f'''(t)
\]

Typically, 3\(^{rd}\) order is sufficient for 2-3 dB of noise figure specifications, so it is used [4]. To realize the 3\(^{rd}\) order Taylor series polynomial, the modified Farrow structure is used which is proposed by C. Pun [5]. This structure is applied transposed, and multiplier-less techniques are applied to optimize the original Farrow structure. The resulting architecture is shown in Figure 5. This FIR filter consists of 8 taps, and therefore requires 32 coefficients. These coefficients are implemented using the sum of powers of two and is optimized for the minimum number of adders needed or the multiplier blocks [5]. By implementing this structure, a phase delay can be generated at every cycle by successive increments of \(\alpha\) and this function can generate a new output sampling frequency that is equal to \(f_m / (1 + \alpha f_m)\).
III. ARCHITECTURE OPTIMIZATIONS

A. Pipelining /Retiming

Due to the complex nature of logic in Arch 2 and Arch 3, the critical delay path can be extremely long in each of these implementations. However, the aim of using these alternate methods for fractional sample rate conversion is to enable a higher clock rate. Therefore, pipelining is essential to reduce the critical delay and improve the performance of each of these architectures.

B. Folding

Since the area cost of Arch 2 and Arch 3 are limiting factors, it is desirable to trade performance for area efficiency. Folding is an ideal candidate for this purpose. Moreover, the main area cost in Arch 2 comes from the multipliers that have to be implemented for every integrator stage, and the number of integrator stages increases with the filter order linearly. Folding enables these multipliers to be shared among the stages, and the saving in area could be substantial for higher order filters since the folding is equal to the order of the filters being used. An interesting point to note is that by folding, the bottleneck in sampling rate can be shifted form the up-sampling factor of the fractional sample rate converter to the filter order. This implies that higher sampling rates can support only lower order filters, which is a reasonable tradeoff since the filter requirements reduce with oversampling. Folding for Arch 2 and Arch 3 are depicted in Figure 6.

CIC architecture and the traditional architecture. The spectrum at the outputs of these architectures is depicted in Figure 7. The sampling rate is down-converted to 400MHz, while the input sinusoid is still visible at 10MHz. This verifies the functionality of the two architectures. However, the traditional sample rate converter performs better by about 20dB in terms of suppression of unwanted spurs in the spectrum.

For the polynomial interpolation structure, sample rate conversion by a factor of 4/5 is implemented using 3\textsuperscript{rd} order Taylor series, and the performance is compared to the traditional sample rate converter in Figure 8. The output sample rate is 800MHz, and the traditional sample rate converter performs better by about 25dB. One might also observe that the time varying CIC architecture from Figure 7 performs only about 5dB better than the polynomial interpolation architecture, and thus it is fair to compare the hardware complexity of the two architectures.

IV. RESULTS

In this section we first present a comparison of the three architectures for fractional sample rate conversion discussed in this report, and analyze the tradeoffs in the energy – delay – area space. Secondly, we demonstrate how architecture techniques can be used to optimize the time varying architecture and polynomial interpolation architecture. Each of the architectures is implemented in Simulink using the Symphony HLS blockset, and verified to be bit true cycle accurate. The results are obtained by using the Cadence RC compiler to estimate the area, power and maximum frequency for 65nm ASIC implementation.

A. Performance Comparison

The performance of the two alternate architectures is compared to that of traditional fractional sample rate conversion. Each of the systems is simulated using an input sinusoid of frequency 10mHz, and the input sampling rate is 1 Hz. The input is passed through a quantizer with 16 bits.

Fractional sample rate conversion by a factor of 2/5 is implemented using 2\textsuperscript{nd} order CIC filters for the time varying...
known that the filter requirements for Arch 3, wherever possible, retiming was done to reduce the power and area of both these architectures. It is shown that it is possible to equally distribute the path delays. The resulting impact on the power – delay – area tradeoff is not as much as Arch 2.

C. Architectural improvements

The results obtained from pipelining Arch 2 and Arch 3 are presented in Table 2. Wherever possible, retiming was done and in other cases the architecture was pipelined in order to equally distribute the path delays. The resulting impact on the power – delay – area tradeoff can be observed from Figure 9 and Figure 10. For Arch 3, pipelining improves the power substantially due to a significant reduction in the critical path delay. However, this limits the amount of area reduction that can be achieved by transistor scaling due to the presence of pipeline registers. For Arch 2, this cross-over is observed in both the power – delay and area – delay performance, since the reduction in the critical path delay is not as much as Arch 2.

Table 2. Pipelining / Retiming results

<table>
<thead>
<tr>
<th></th>
<th>Arch 2 (Time Var. CIC)</th>
<th>Arch 3 (Poly Interp.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original t\text{crit}</td>
<td>(7t_{\text{add}}+3t_{\text{mul}}+2t_{\text{max}})</td>
<td>(5t_{\text{add}}+5t_{\text{mul}}+t_{\text{shift}})</td>
</tr>
<tr>
<td>Pipeline reg.</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>Latency</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>New t\text{crit}</td>
<td>(4t_{\text{add}}+3t_{\text{mul}}+2t_{\text{max}})</td>
<td>(3t_{\text{add}}+t_{\text{mul}}+t_{\text{shift}})</td>
</tr>
</tbody>
</table>

Table 3. Maximum clock frequency for sample rate conversion arch

<table>
<thead>
<tr>
<th></th>
<th>Arch 2 (Time Var. CIC)</th>
<th>Arch 3 (Poly Interp.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>200 MHz</td>
<td>166 MHz</td>
</tr>
<tr>
<td>Pipelined</td>
<td>500 MHz</td>
<td>333 MHz</td>
</tr>
<tr>
<td>Folded</td>
<td>96 MHz</td>
<td>150 MHz</td>
</tr>
</tbody>
</table>

The main architecture optimization in this report is the use of folding to reduce the number of multipliers. For Arch 2, this leads to a reduction of 2 multipliers and an adder, while in Arch 3 this leads to a reduction of 2 multipliers and 2 adders. Therefore, the power and area of both these architectures reduces significantly when folded. The key point here is that for an Nth order filter / Taylor series, it is possible to reduce the number of multipliers by a factor of N-1. Therefore, the savings are larger as the filter order increases. However, folding limits the clock frequency achievable since the folded section needs to operate at an up-sampled rate. Since the increase in clock frequency is proportional to the order of the filter, this can be compensated for by using a lower order filter. This is reasonable since it is well known that the filter requirements reduce with the oversampling ratio. The maximum clock frequency achievable using different architecture techniques are summarized in Table 3. As a side note, it is also found that the output amplitude is attenuated by 10dB due to the folding technique. The details are omitted due to space constraints.

V. CONCLUSION

Fractional Sample Rate Conversion is a critical component of Digital Front Ends for Multi-Standard Radios. Traditional architectures for sample rate conversion are limited in the maximum clock frequency by the up-sampling factor and are unable to realize sample rate conversion by factors close to 1. Two alternate architectures – time varying CIC filters and polynomial interpolation have been compared in this report in the power – delay – area space. Since these filters are limited by the power and area cost, pipelining and folding have been used to achieve a better power – delay – area tradeoff. Moreover, by using folding it is shown that it is possible to tradeoff the maximum clock rate with the filter order.

REFERENCES