ISSN: 2229-6948(ONLINE)
DOI: 10.21917/ijct.2014.0140

AREA EFFICIENT FRACTIONAL SAMPLE RATE CONVERSION ARCHITECTURE FOR SOFTWARE DEFINED RADIOS

Latha Sahukar[1] and M. Madhavi Latha[2]
[1]*Department of Electronics and Communication Engineering, Aurora's Technological and Research Institute, India*
E-mail: lathasahukar@gmail.com
[2]*Department of Electronics and Communication Engineering, JNTUH College of Engineering Hyderabad, India*
E-mail: mlmakkena@yahoo.com

Abstract
The modern software defined radios (SDRs) use complex signal processing algorithms to realize efficient wireless communication schemes. Several such algorithms require a specific symbol to sample ratio to be maintained. In this context the fractional rate converter (FRC) becomes a crucial block in the receiver part of SDR. The paper presents an area optimized dynamic FRC block, for low power SDR applications. The limitations of conventional cascaded interpolator and decimator architecture for FRC are also presented. Extending the SINC function interpolation based architecture; towards high area optimization and providing run time configuration with time register are presented. The area and speed analysis are carried with Xilinx FPGA synthesis tools. Only 15% area occupancy with maximum clock speed of 133 MHz are reported on Spartan-6 Lx45 Field Programmable Gate Array (FPGA).

Keywords:
Decimation, Interpolation, Sample Rate Conversion, Fractional Rate Conversion

1. INTRODUCTION

1.1 SAMPLE RATE CONVERSION (SRC)

The sample rate conversion (SRC) block in a digital signal processing (DSP) system computes the samples at a new rate using the samples available at different sampling rate. The SRC blocks, used in several applications include the following.

- Immediately after the Analog to Digital Converter (ADC) to reduce the sampling rate as per the bandwidth
- Just before the Digital to Analog Converter (DAC) to increase the sampling rate to match the DAC specifications
- In demodulators to achieve fixed samples per symbol
- In multi-rate filters

The software defined radio (SDR) uses reconfigurable architecture to adopt different modulation or demodulation schemes with dynamically selectable parameters [1], [9] such as carrier frequency, bandwidth, channel coding etc. As the SDRs are required to provide high level flexibility to handle bandwidths and modulation schemes, the SRC become crucial block. The filtering tasks and control logic that are required to deliver a high-performance and flexible up-conversion system that is typically required at the physical layer in a modern adhoc networks is given in [1].

The SRC can be achieved by either time domain and frequency domain techniques. The core benefits of frequency domain SRC are described for wide band applications in [2]. In recent days the research towards power and speed optimized SRC implementation is gaining importance. The architectures presented in [3] utilize the Farrow based finite impulse response (FIR) filters with parallel hardware approach for high speed and increased throughput for SDR applications.

The work given in [4] presents Design Approach of Low Power VLSI using Multirate digital signal processing system which includes sampling rate conversion. However only the context of integer rate conversion are discussed here.

In paper [5] properties of the SRC algorithm based on fractional delay (FD) filters have been presented. The dissimilarities of different methods of the FD filter design have been analyzed using the overall filter or the overall window. Based on the observed properties the classification of FD filter design methods into three categories have been proposed; optimal fractional filter design, offset window method and polyphase decomposition.

A parallel processing SRC structure is proposed in [6] to achieve high speed data transmission for multiband OFDM based SDR systems. The proposed technique derives an impulse response matrix from the sequential SRC structure, which in turn is calculated from a block of input samples which has less complexity than conventional Farrow structure.

The optimum and minimum order structures is proposed in [7] for sampling rate conversion from 44.1 KHz compact disc (CD) to 48 KHz digital audio tape (DAT). The proposed techniques apply multistage up-sampling and down-sampling technique to obtain the optimum or minimum order configurations.

Towards the high speed SRC systems the Cascaded Integrator and comb filters become more suitable option due to the requirement of only addition being carried out in the higher clock domain. The work given in [8] utilizes signed digit (SD) algorithm to incorporate the key features of the conventional number system with a signed digit (SD) to improve the addition time with high power constraints in an optimized fashion. A sharpening polynomial to improve the stop-band characteristic and the grading technique compensated for the pass-band characteristic in the CIC integer rate conversion architecture is used in [9].

However this architecture cannot handle fractional rate conversion requirements hence shall result in limited configurability in SDR based architectures.

The applications of SRC are spread across communication, DSP, RADAR, SONAR and audio applications [10]. Most of the algorithms used in these applications use multi-rate signal processing techniques. As the efficient implementation of multi rate systems on FPGA heavily depends on architecture level optimization of SRC modules [11], the present work is useful in majority of these applications. The work presented in this paper proposes an area efficient fractional sample rate conversion (FRC) scheme which finds potential applications in different communication applications.

1.2 FREQUENCY DOMAIN TECHNIQUES AND FRACTION DELAY FILTERS

The basic frequency domain based SRC [12] scheme uses Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform (IFFT) blocks. The advantage of achieving the frequency shift property with rotation of FFT bin values is presented in [2] for addressing wide band signal handling challenges. The conventional complex NCO multiplication is achieved with direct spectrum rotation and various possibilities for frequency domain filtering are discussed.

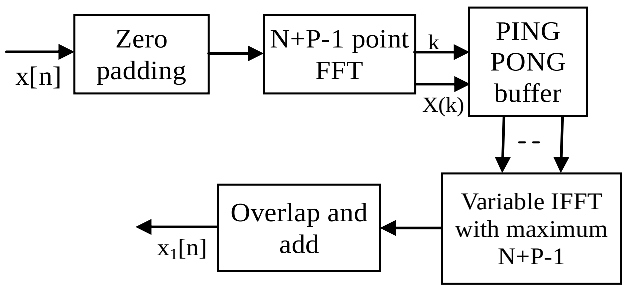

Fig.1. Block diagram of frequency domain SRC

The overlap and add method at the output of IFFT is employed to reconstruct the time domain signal. The Fig.1 shows the high level block diagram of SRC module of architecture implemented in [2].

This technique for fractional sample rate conversion based on an iterative sinc method is given in [13]. The proposed algorithm is evaluated against the Farrow resampler, and performance simulation targeting different signal-to-noise ratios. The architecture is implemented in 65 nm CMOS technology, and synthesis results claim that the ISRC requires at least 23% less silicon area, compared to a Farrow filter with similar performance.

As described in [14] the coherent resampling (CR) algorithm can greatly reduce the DFT leakage as well as the errors of DFT-based measurements. This technique can also improve the accuracy of frequency estimation systems. The algorithm uses extended Kalman filter for instantaneous frequency tracking and fractional B-spline resampler for signal approximation. The CR algorithm is a software counter part of synchronous sampling, i.e., sampling, synchronized with fundamental frequency of the signal.

The OFDM signaling, resampling for time-scaling compensation is investigated and shown to be a reduced-rank signal processing strategy in [15], for multiscale–multilag signals occurring in underwater acoustic communications.

The architecture presented in [16] allows the same RF front end and digitization to be used for many waveforms and symbol rates, but requires the demodulator to generate symbol based samples from the asynchronous input samples. A method for jointly removing timing delay and timing drift is also proposed.

A variable fractional delay (VFD) filter is widely used in applications such as symbol timing recovery [17], arbitrary sampling rate conversion and echo cancellation.

The interpolator and control mechanism that maintains the proper interpolation ratio through the range of frequency tolerance and slow drift of the input and output sampling clocks is given in [18].

The design of batch and streaming resamplers with arbitrary sample rate change, prescribed accuracy and known exact delay are described in [19]. The main concern is in applying these resamplers in digital radar systems to provide exact delay through the resampler implementation.

Section 2 details the issues in handling fractional rate conversion by conventional cascaded interpolation and decimation stages. It also illustrates the high level block diagram of typical integer rate conversion modules of any DSP system. Section 3 explains the principle of fractional rate conversion using sinc based virtual DAC and ADC method. Section 4 provides the architecture details and optimization techniques for FPGA implementation of proposed FRC. Section 5 has the simulation, synthesis and onchip verification results. Section 6 concludes the work.

2. INTEGER RATE CONVERSION (IRC) AND FRACTIONAL RATE CONVERSION (FRC)

2.1 INTEGER RATE CONVERSION ARCHITECTURES

The SRC architectures for decimation or interpolation by integer factor are realized with digital filters in combination with the multi clock domain register blocks. These blocks become directly suitable for streaming purpose applications as they maintain samples guaranteed in real time for next blocks in signal processing chain. For example in a decimator block with decimation by integer value D, the output gets one sample for every D clock cycles with respect to the input clock period. Similarly for integer interpolator, for every input sample, $(I-1)$ additional samples are filled-in within each clock period.

The Fig.2 has the block diagrams for decimation and interpolation SRC blocks. The functional block diagrams illustrate that both these SRC architectures involve low pass filters. However there are several area and speed optimized architectures for efficient implementation of these blocks on FPGAs/ASICs, which are not discussed here.

The decimator block filters the inputs signal by fs/D cutoff frequency and then selects one sample for every D samples. The filter ensures that the $(D–1)$ fs/D to fs band is attenuated to the stop band attenuation level, so that it doesn't introduce distortion when it gets aliased to the 0 to fs/D band after decimation. Similarly the Interpolator block need to filter out the images of the signal after adding the $I-1$ zeros in between the original input samples. It is to be noted that the required filtering in both types of SRC are finite impulse response (FIR) type filters. Hence the choice of higher filter order and higher word lengths allows ideal characteristics for resulting SRC implementation.

Let $x(n)$ be input signal with impulse response $h(n)$, the frequency domain equivalent of decimated output, is, thus, equal to the sum of the down sampled and the aliased components, divided by the decimation factor.

$$V(k) = X(k)H(k) \qquad k = 0,1,\ldots N\text{-}1 \tag{1}$$

$$Y(k) = 1/D\sum_{i=0}^{D-1} v\left[k + \frac{Ni}{D}\right] \quad k = 0,1,2\ldots\frac{N}{D}-1 \tag{2}$$

Here, $V(k)$ and $Y(k)$ are considered as the filtered and decimated outputs in frequency domain and D as the decimation factor.

Interpolation process contains insertion of $(I-1)$ zeros between successive samples of input signal and filtering with impulse response $h(n)$ for an interpolation factor of I. In frequency domain the same can be denoted as below.

$$V(W_y) = X(IW_y) \tag{3}$$

$$Y(W_y) = V(W_y)H(W_y) \tag{4}$$

where, ω_x and ω_y are the continuous frequency variables of $x(n)$ and $y(n)$ respectively. If the frequency variable ω_x varies from 0 to $2\pi/I$, then ω_y varies from 0 to 2π. The spectrum $V(\omega_y)$ contains $(I-1)$ images of spectrum $X(\omega_x)$ along with actual spectrum of $x(n)$. In general, this can be expressed

$$V\left[k + \frac{iN}{I}\right] = X(k) \tag{5}$$

where, $k = 0,1,..N/I$ -1 and $i = 0, 1, 2,\ldots I-1$.

The discrete equivalent of filtered output spectrum is simply obtained by multiplying this spectrum with that of $H(\omega_y)$.

$$Y(k) = V(k)H(k) \tag{6}$$

where, $k = 0$ to $N - 1$.

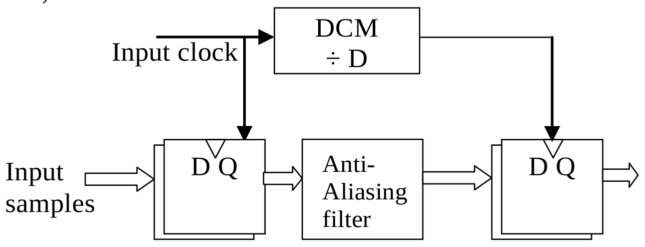

Fig.2(a)

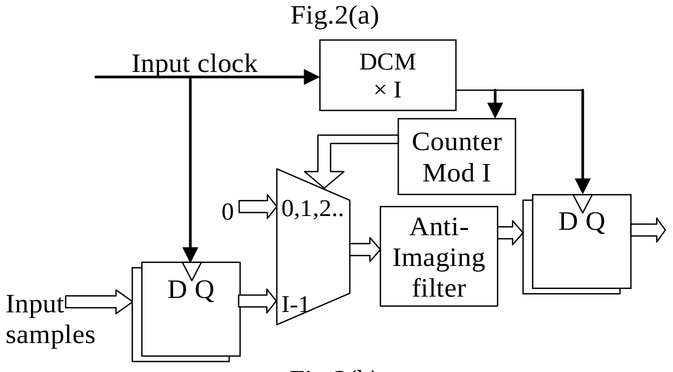

Fig.2(b)

Fig.2. Generalized rate conversion architectures for (a). Decimation, (b). Interpolation

In Fig.2(a) the digital clock manager (DCM) generates the decimated clock, which has output clock period equal to D times input clock's period. The input sample is registered by the input register on rising edge of input clock. The anti-aliasing filter performs low pass filtering as explain above. The filtered output is registered with respect to output clock.

In Fig.2(b) the DCM generates interpolated clock, with output clock frequency equals to I times of input clock frequency. In other words the clock period gets reduced by factor of I from input to output. The input register loads the value on rising edge of input clock. Since in one cycle interval of input clock the output clock makes I number of cycles. The input register holds same value for I number of output clock periods. The Mod-I counter runs as per output clock and generates count value between 0 to $I-1$. As a result, the multiplexer output generates a signal pattern with *I-1* zeros inserted between each two adjacent input samples. The anti imaging filter produces output at output clock rate. The output register loads the filter output on rising edge of output clock.

2.2 LIMITATIONS OF CASCADED I AND D STAGES

To achieve a fractional rate conversion (FRC) one approach could be to implement the required fraction with cascaded I and D stages.

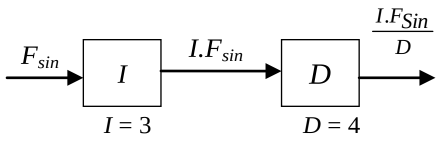

Fig.3(a)

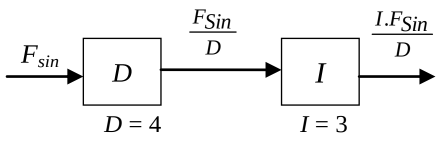

Fig.3(b)

Fig.3. Two possible schemes of cascaded interpolator with $I = 3$ and decimator $D = 4$ stages

The Fig.3(a) shows the cascaded interpolator followed by decimator to achieve FRC of 0.75. The Fig.3(b) shows other possible combination decimator followed by interpolator.

The resulting sampling rates in both cases are indicated. It is to be observed that in case of Fig.3(a), the sampling rate at the output of interpolator stage is I times that of input sampling rate F_{sin}. Hence the decimator stage must be able to handle the sample coming at this higher sampling rate. In situation where higher precision SRC is required in the required fraction I/D, both the I and D value will be higher. Then realizing the I stage with such large I value becomes practically impossible.

Where, as in Fig.3(b), the sampling rate at the output of D stage is only $\frac{F_{Sin}}{D}$, this can be easily handled by I stage. However this approach has got major drawback that the information bandwidth is only preserved up to $\frac{F_{Sin}}{2D}$ after decimation stage. Even after interpolation the sampling clock increases to $\frac{I.F_{Sin}}{D}$ but the information bandwidth remains same. Hence this scheme is suitable only if the maximum information bandwidth requirement is less than this value.

The role of filter in the efficient sampling rate converter is twofold: it acts as the anti-imaging filter $H_I(z)$, and also as the anti-aliasing filter $H_D(z)$. For the adequate removal of images, the stop-band edge frequency of the low-pass filter must be below π/L, and avoiding of aliasing requires the stop-band edge below π/M. The stop-band edge frequency at ω_s, which is given by,

$$W_S = \min\left(\frac{\pi}{I}, \frac{\pi}{D}\right) \tag{7}$$

Choosing ω_s according to Eq.(7) ensures the elimination of imaging which appears in interpolation, and at the same time ensures the suppression of aliasing that may be caused by decimation. Hence, the ideal specifications for the magnitude response of filter are given by,

$$\left|H\left(e^{jw}\right)\right| = \begin{cases} L, & |w| \le \min\left(\frac{\pi}{I}, \frac{\pi}{D}\right) \\ 0, & otherwise \end{cases} \tag{8}$$

It is important to observe that for a large L or M, filters with very narrow pass bands are required.

However here an asynchronous memory based FIFO would be required to read the samples at the required output sampling rate. Based on these limitations, it can be concluded that the cascaded I and D architecture based SRC cannot suit for high precision FRC.

2.3 RATE CONVERSION SEEN AS INTERPOLATION PROBLEM (FOR FRC)

Instead of treating the FRC as low pass filtering problem, it can be attempted as interpolation problem. The input samples are treated as sample values available at T_s intervals of time, and based on the required FRC, the new sample values will be computed with interpolation. The interpolation method for a given signal $x(t)$ is illustrated in Fig.4.

Interpolator computes the new sample values $y(l)$ at arbitrary points $t_l = (n_l + \mu_l)T_{in}$. The output sample time is determined by the fractional interval or delay μ_l € [0 1) and the integer index n_l. The l^{th} output sample $y(l) = ya(tl)$ as $t_l = (n_l+\mu_l)T_{in}$. The interpolation filter calculates the y(l) according to the convolution after knowing n_l and μ_l parameters as given in Eq.(9).

$$y(l) = \sum_{k=-N/2}^{\frac{N}{2}-1} x(n_l - k)h(k, \mu_l) \tag{9}$$

where, N is the length and $h(k, \mu_l)$ is the impulse response of the interpolation filter.

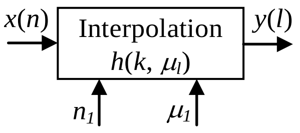

Fig.4(a)

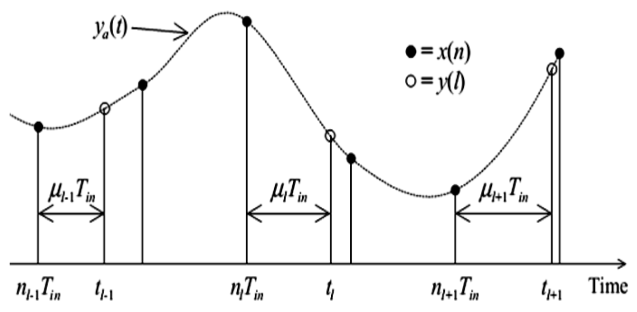

Fig.4(b)

Fig.4(a). input and outputs of interpolator, (b) Interpolation in time domain, computing $y(l)$ from $x(n)$

For the interpretation of convolution sum, first we have to find the existing sample preceding the new sample instant t_l, denoted by n_l. Based on the location of this sample, existing samples located at $n = n_l–N/2+1$, $n_l–N/2+2,\cdots$, $n_l+N/2$ are used. Second, the distance between t_l and n_l. T_{in} is measured as a fraction of T_{in}, giving the fractional interval μ_l. The coefficient values for the above convolution are determined by using μ_l.

Interpolation filters can be designed using many types of mathematical functions, among which polynomials are most efficient technique for hardware implementation. Simplest polynomial interpolation is linear interpolation which has the degree of one and interpolates between two samples and filter has two coefficients for this interpolation. The popularly used interpolation techniques are listed below.

- Linear interpolator
- Cubic interpoloator
- Sinc interpolator

Linear interpolator equation can be written as given in Eq.(10).

$$y[l] = x[m_k+1]\mu_k+x[m_k](1-\mu_k) \tag{10}$$

Linear interpolator can be considered as a filter with 2 coefficients μ_k and μ_{k+1}.

In cubic interpolator the $y[k]$ is evaluated using nested evaluation. Two types of cubic interpolators are available for implementation.

- Cubic Lagrange Interpolant
- Cubic B-spline Interpolant

Cubic Lagrange Interpolant equation can be written as given in Eq.(11)

$$g(t) = \begin{cases} 1 - \frac{1}{2}t - |t|^2 + \frac{1}{2}|t|^3, & |t| < 1 \\ 1 - \frac{11}{6}|t| + |t|^2 - \frac{1}{6}|t|^3, & 1 \le |t| < 2 \\ 0, & |t| \ge 2 \end{cases} \tag{11}$$

The most commonly used B-spline in practice is the cubic B-spline, equation can be written as given in Eq.(12)

$$g(t) = \begin{cases} \frac{2}{3} - \frac{1}{2}|t|^2(2-|t|), & |t| < 1 \\ \frac{1}{6}(2-|t|)^3, & 1 \le |t| < 2 \\ 0, & |t| \ge 2 \end{cases} \tag{12}$$

The time domain and frequency domain characteristics of these interpolators are given in Fig.5. The Sinc interpolator which is explained in next section is also illustrated here for comparison.

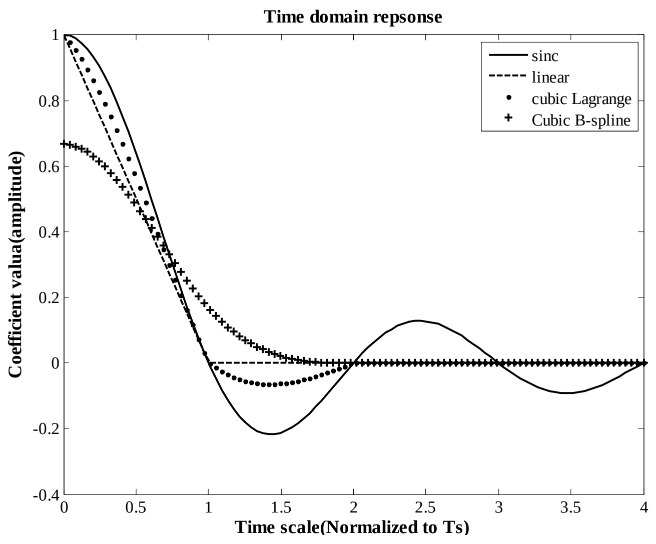

Fig.5(a). Coefficient value (amplitude)

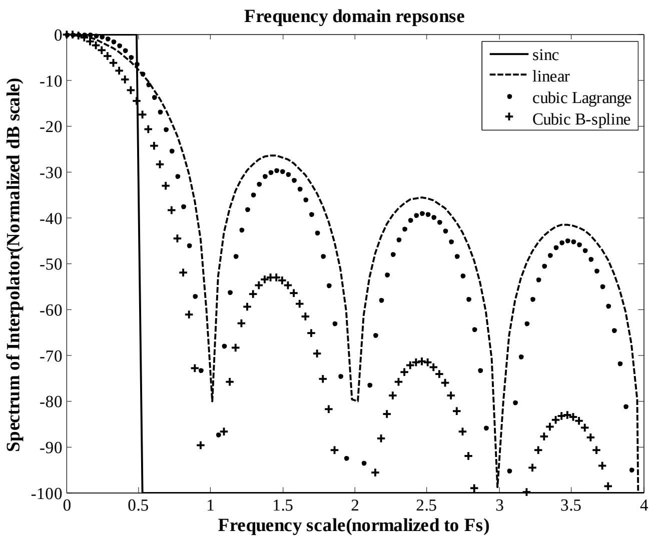

Fig.5(b). Spectrum of interpolator (normalized dB scale)

Fig.5(a) Time domain, (b) frequency domain characteristics of interpolators

It is to be observed that linear and basic form of cubic interpolators cannot meet the wide band modern SDR communication requirements which demand higher accuracies and sharp spectral characteristics. However some of these architectures find applications, where a reasonable accuracy is sufficient.

In this paper with simulation it is illustrated that, the sinc interpolator with multiple zero crossing based architecture having sufficient coefficients in each zero crossing interval can give very precise interpolation. The details are presented in further sections.

3. FRC WITH VIRTUAL DAC AND ADC

The sinc interpolator is equivalent to adopting a virtual DAC whose conversion rate is equal to input sampling rate followed by a virtual ADC whose sampling rate is equal to desired new sampling rate.

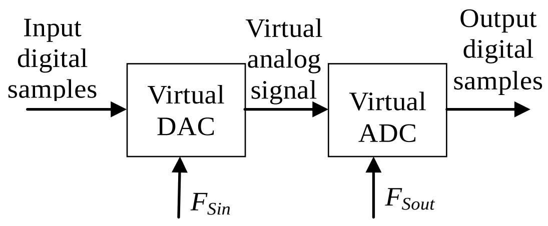

Fig.6. SINC interpolation equivalent to virtual DAC followed by ADC

The virtual DAC implements low pass filter to only allow the spectrum around zero frequency and reject other images of spectrum which are around $\pm F_s$. In ideal low pass filter, to maintain rectangular window type frequency domain characteristics, the filter requires infinite length coefficients forming sinc function. The time domain filtering is convolution of these filter coefficients with digital sample values, i.e. the weighted sinc functions at each sample instant with weight equal to sample value are added to obtain the virtual DAC output.

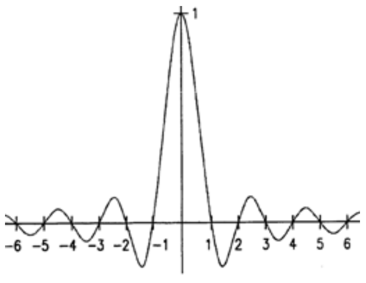

Fig.7(a)

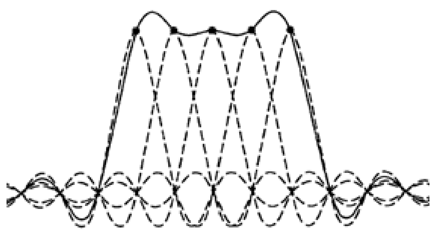

Fig.7(b)

Fig.7(a). Filter coefficients for DAC, (b). Virtual DAC output for input with 5 non-zero samples with value 1

The Fig.7 illustrates the virtual DAC computational requirement for an input signal with 5 consecutive nonzero samples with value 1. The reconstructed analog signal, showing the rectangular pulse can be observed. The contribution of signal value at a given time is more (higher weightage by sinc function) for the nearby digital sample values and less for the samples which are away from the instant where value is computed.

The analog signal $x(t)$ input to the ADC is given in Eq.(13)

$$(t) = \sum_{m=-\infty}^{\infty} x(mT_{in}) \sin c\left(\frac{t}{T_{in}} - m\right)$$

$$(t) = \sum_{m=-\infty}^{\infty} x_d[m] \sin c\left(\frac{t}{T_{in}} - m\right) \tag{13}$$

Here $x_d[m] = x(mT_{in})$ denotes the discrete-time sampled signal obtained from the ADC.

To alter the sampling interval to some other output value, say T_{out}, we would like to generate the discrete-time sequence $y_d[n] = x((n+\varepsilon)T_{out}$, where ε is some fractional sampling offset parameter which satisfies $0 \leq \varepsilon < 1$.

$$y_d[n] = \sum_{m=-\infty}^{\infty} x_d[m] \sin c\left(\left(\frac{F_{in}}{F_{out}}\right)(n+\varepsilon) - m\right)$$

$$y_d[n] = \sum_{m=-\infty}^{\infty} x_d[m] \sin c(\rho(n+\varepsilon) - m) \tag{14}$$

Here, $F_{in} = \frac{1}{T_{in}}$ and $F_{out} = \frac{1}{T_{out}}$ denote, respectively, the input and output sampling rates, whereas $\rho = \frac{F_{in}}{F_{out}}$ denotes the sampling rate conversion factor.

As the practical implementation cannot take infinite length sinc function, based on the expected accuracy levels the length of sinc function can be selected. The number of zero crossings N_z up to the sinc function selected is usually considered as the design parameter. In that case N_z number of samples on either side are sufficient for estimating the value for the required time. In digital implementation, the number of samples stored between two crossings decide the time resolution accuracy during FRC. For example if 1024 samples between two zero crossings are stored in ROM then it allows 1024 time steps between two sample intervals. Hence in the resampling factor upto 10 fractional bits, the rate conversion can be achieved with this ROM.

The virtual DAC is implemented by storing sinc function in ROM up to few zero crossings. The virtual ADC is the estimation of the digital sample value by taking the weighted (from *sinc* function) sum of adjacent samples.

The following equation describes the practically implemented logic with windowed finite length *sinc* function $\hat{g}$, where N_z is the selected number of zero crossings over which the non zero *sinc* function is considered.

$$y_d[n] = \sum_{m=-Nz}^{+Nz} C_d[m]\hat{g}(\rho(n+\varepsilon) - m) \tag{15}$$

With the proposed SINC based interpolation the algorithm for fractional rate conversion is summarized below.

Algorithm Configuration parameters

a) Number of zero crossings for the SINC function (N_z)
b) Length of the ROM with sampled SINC values (L)
c) Time register size (MQN)
d) Input sample Buffer size

Algorithm runtime I/Os

a) Input - Time register value based on the required FRC rate
b) Input - Input samples as per input clock domain
c) Output - output samples as per the output clock domain

Algorithm

a) Load the input samples in input sample buffer (true asynchronous dual port RAM) with input clock and control signals
b) Increment the time register for every clock cycle
c) Generate the address for read port of input sample buffer by considering the integer part of the time register
d) Read ($2N_z$) values from sample buffer
e) Generate the address for ROMs with SINC function as per the fractional part of time register
f) Multiply the $2N_z$ ROM outputs with input samples and compute the output sample
g) Generate the output value and enable in output clock domain

4. RTL IMPLEMENTATION

The block diagram of Register Transfer Level (RTL) implementation of the proposed FRC block is show in Fig.8. VHDL is used for this implementation. Only synthesizable constructs of VHDL are used to ensure that the module works on FPGA.

The input sample memory allows samples to be buffered before using them for fractional rate conversion. This is implemented with true asynchronous FIFO in FPGA to enable the FRC between two independent clock domains. The input clock can be from any different source. All the computations are performed with respect to output clock domain. However the required FRC factor needs to be given in terms of input sample writing clock.

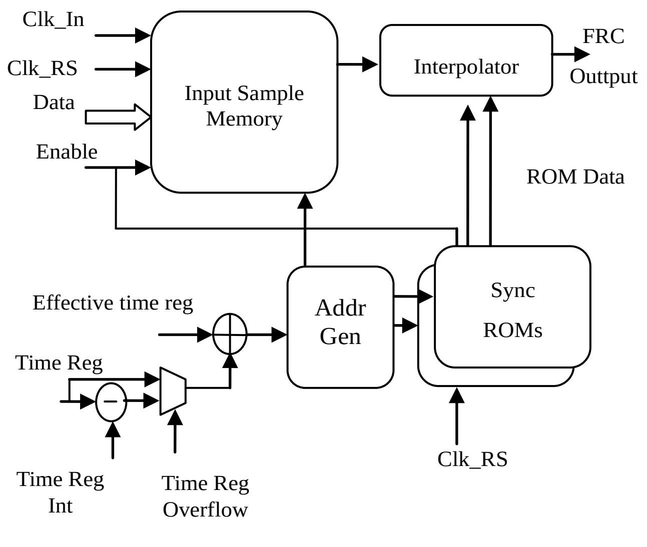

Fig.8. RTL block diagram for proposed FRC

The FRC factor will be specified by a ratio of required clock period to input clock period.

$$FRC\_factor = \frac{Desired\ sample\ period}{FIFO\ writing\ clock\ period} \qquad (16)$$

The time register is loaded with this value represented in MQN unsigned fixed point format. The M-N bits indicate the integer part to decide the maximum decimation possible and LSB N bits describe the fractional part of FRC_factor. The present implementation uses 24Q16 format. The loaded FRC_factor value is incremented after every output sample computation. The integer part is used to offset the read address pointer in the input sample memory and fractional part is used for sinc interpolation. The accumulated result of fractional part is compared with 1 and to detect overflow. The overflow will result in one additional increment of read address pointer of input sample memory.

Based on these conditions the address is computed for input sample memory and sinc ROM. To enable parallel computation, multiple ROMs are used to store the *sinc* function. The interpolator block performs the multiplication of input samples with sinc function values and computes sum of these products. The output value is produced with enable signal.

To enable high speed implementation pipelining technique is adopted in all stages. As the sinc function is symmetric the sample values of sinc function are stored for only positive side. The negative side sinc values are computed by using same ROMs.

5. SIMULATION AND FPGA VERIFICATION

5.1 SIMULATION WITH MODELSIM

The implemented FRC module is simulated using Mentor's Modelsim simulator and functionality is verified for several signal conditions. The Fig.9 shows the simulation results for sine wave input.

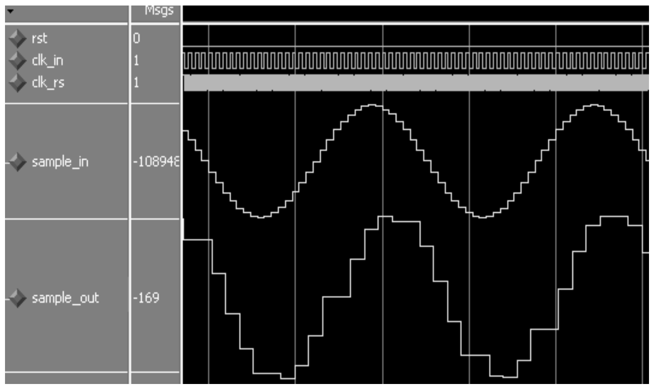

Fig.9. Simulation results for implemented FRC module

The FRC_factor of 2.125 is considered for simulation. The validation of the FRC without any spectral aliasing/imaging is proved through MATLAB analysis of captured results. The details are explained in next section.

5.2 FPGA SYNTHESIS AND TIMING SIMULATION

The FRC module is synthesized with Xilinx ISE 12.3 version for Spartan-6 L×45T FPGA. The Table.1 shows the area occupancy for various Xilinx devices. As the architecture is highly area optimized it can observed that the module occupies less than 15% resources. This capability is essential in SDR applications as the necessity is for complex FRC module having again two instances, one for *I* and other for *Q*. It also shows the device utilization summary produced by Xilinx ISE tool.

Table.1. Device utilization summary of FRC module for Spartan 6 family L×45TFPGA

Project File:	Ise_for_area_and_speed.xise	**Parser Errors:**	No Errors
Module Name:	resampler_top	**Implementation State:**	Synthesized
Target Device:	xc6slx45t-2fgg484	• **Errors:**	No Errors
Product Version:	ISE 12.3	• **Warnings:**	3 Warnings (0 new)
Design Goal:	Balanced	• **Routing Results:**	
Design Strategy:	Xilinix Default(unlocked)	• **Timing Constraints:**	
Environment:	System Settings	• **Final Timing Score:**	

Device Utilization Summary(estimated value)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	700	54576	1%
Number of Slice LUTs	958	27288	3%
Number of fully used LUT-FF pairs	265	1393	19%
Number of bonded IOBs	81	296	27%
Number of Block RAM/FIFO	8	116	6%
Number of BUFG/BUFGCTRLs	2	16	12%
Number of DSP48A1s	8	58	13%

The FRC module is also speed optimized by using pipelining at various stages to achieve clock speeds up to 100 MHz. The Fig.10 below has the snap shot of timing summary generated by Xilinx ISE tool.

```
Timing Summary:
Speed Grade: -3
    Minimum period: 7.465ns (Maximum Frequency:
                                  133.962 MHz)
    Minimum input arrival time before clock: 6.789ns
    Maximum output required time after clock: 5.862ns
    Maximum combinational path delay: 1.222ns
```

Fig.10. The timing summary produced by Xilinx ISE tool

The usable maximum frequency is considered as 100 MHz (10ns period with 2.5ns slack period), which is suitable for all the wide band communication applications. The minimum input arrival time and maximum output required time can be ignored as they correspond to FPGA pin level delays.

5.3 VERIFICATION USING XILINX CHIPSCOPE TOOL

The Xilinx Chipscope tool is used to capture the input and output waveforms of FRC module and then the data is analyzed using MATLAB to verify the results. The relationship between sampling frequency and peak bin number are used to verify the correctness of FRC. The Fig.11 shows the FFT results of both input and output of implemented FRC module. The ratio of peak observed bins is matching with the FRC factor of 2.125.

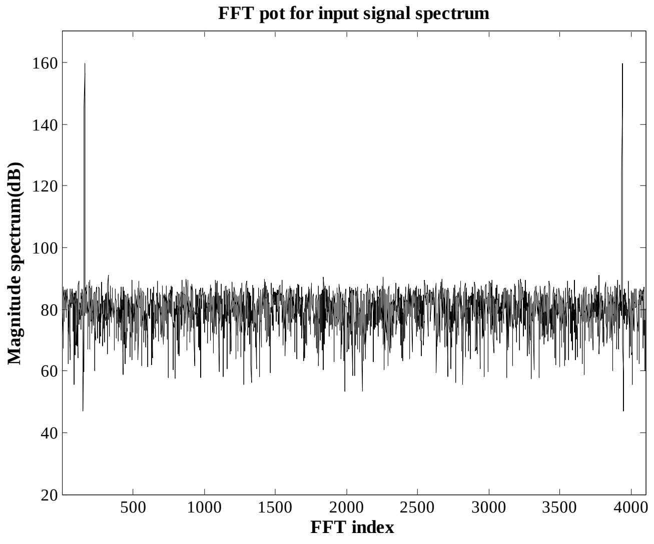

Fig.11(a). FFT pot for input signal spectrum

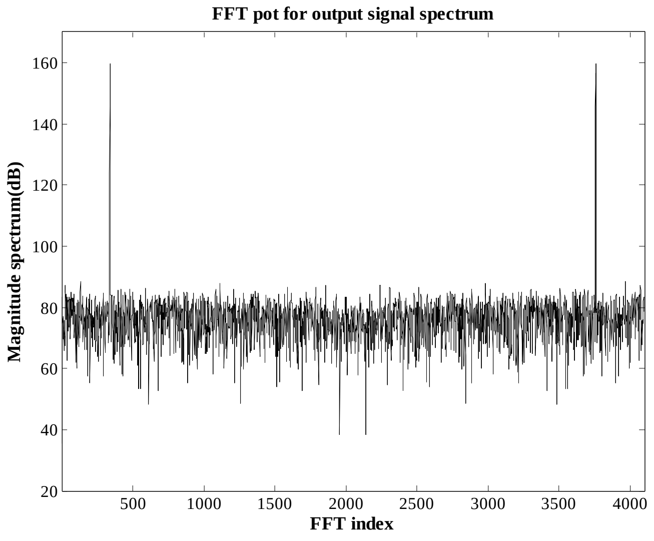

Fig.11(b). FFT pot for output signal spectrum

Fig.11. Matlab's FFT plots for captured chipscope data

5.4 VERIFICATION FOR 16-QAM SIGNALS

To verify the performance of developed FRC on wideband random signals, a 16-QAM signal generated with 100 MHz sampling rate and with symbol rate 40.96 M symbol/sec is applied as input signal to the module. Recorded signal data from signal generator is applied through ROM as test input. The FRC factor of 1.2205 is applied to achieve exactly 2 samples per symbol. The input and output signals of FRC module are captured in chipscope and results are analyzed with MATLAB. The Fig.12 has the time domain and frequency domain representation of input and output signals.

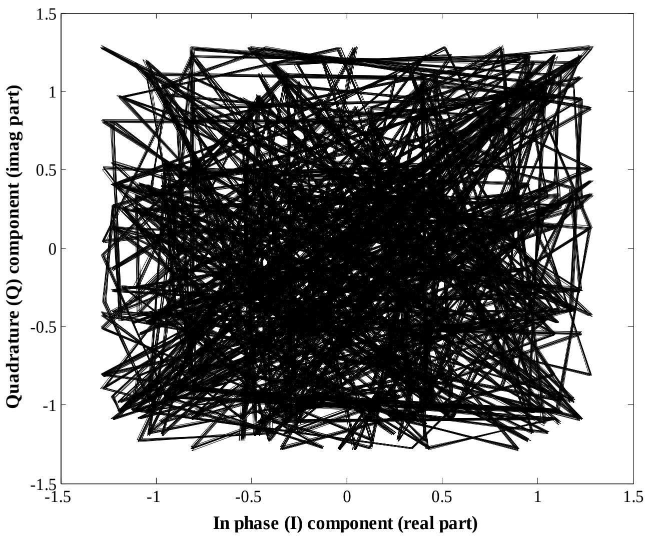

Fig.12(a). Time domain representation of input

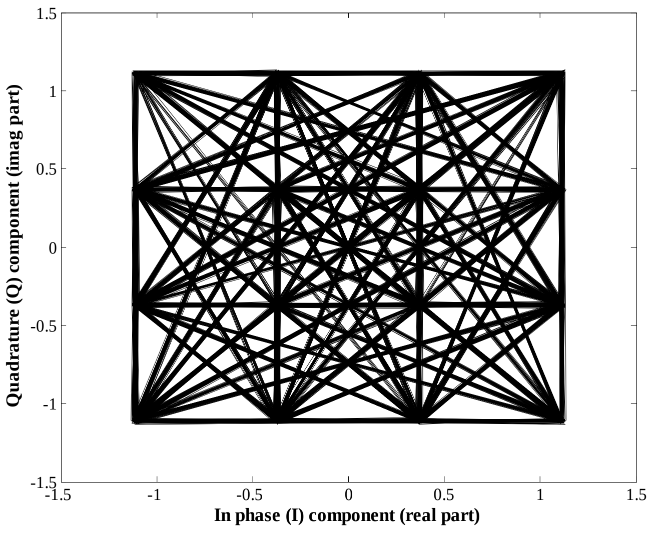

Fig. 12(b). Time domain representation of output of FRC module

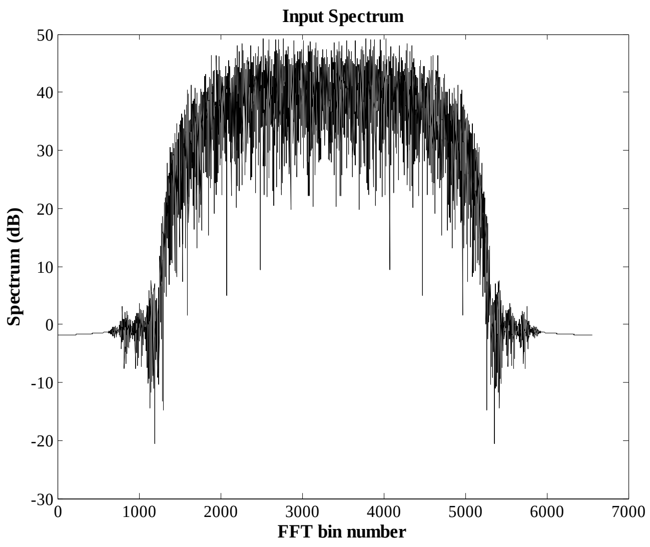

Fig.12(c). Frequency domain representation of input

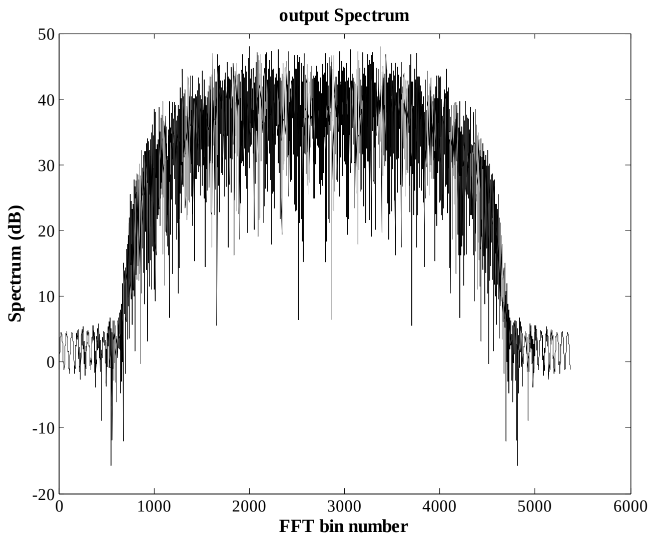

Fig.12(d). Frequency domain representation of output of FRC module

It can be observed that as the selected FRC type is decimation (above 1) the number of output samples is less in comparison with input, and hence the FFT point size is less for output while computing the spectrum. Note that no zero padding is performed while computing FFT. This test case verifies that the FRC module is capable of handling wide band signals and finds application in several emerging wideband communication architectures. As the output sample rate is twice of symbol rate the output I-Q constellation results in16-QAM constellation.

6. CONCLUSION

The necessity of integer and fractional sample rate conversion blocks in software defined radio applications are discussed. The architectural challenges in FRC are presented. The principle of virtual DAC and ADC based rate converter is explained. An area efficient approach with 8 zero crossing long sinc function based rate converter is proposed. The results for high bandwidth signals are verified both at simulation level and hardware level. To verify the performance aspects, the synthesis results are analyzed. Only 15% area occupancy with maximum clock speed of 133 MHz is reported on Spartan-6 L×45 FPGA. The developed FRC is tested for real wideband 16-QAM signals and validated for its time domain and frequency domain characteristics.

ACKNOWLEDGEMENT

The authors would like to acknowledge Aurora's Technological and Research Institute, Hyderabad and Jawaharlal Nehru Technological University, Hyderabad for its support in carrying out the research work.

REFERENCES

[1] Latha Sahukar and M. Madhavi Latha, "FPGA based low power reconfigurable modulator with Digital Up Converter for adhoc networks", *Journal of Telecommunications*, Vol. 11, No. 2, pp. 49-53, 2011.

[2] Latha Sahukar and M. Madhavi Latha, "Area Efficient Architecture for Frequency Domain Multi Channel Digital Down Conversion for Randomly Spaced Signals", *Advances in Intelligent Systems and Computing-Springer Link*, Vol. 177, pp. 233-241, 2013.

[3] J. P. Long and J. A. Torres, "High throughput Farrow re-samplers utilizing reduced complexity FIR filters", *IEEE Military Communications Conference*, pp. 1-6, 2012.

[4] R. M. Rewatkar and S. L. Badjate, "A Design Approach of Low Power VLSI for Downsampler Using Multirate Technique", *IEEE International Conference on Communication Systems and Network Technologies*, pp. 727-731, 2013.

[5] M. Blok, "Fractional delay filter design for sample rate conversion", *IEEE Federated Conference on Computer Science and Information Systems*, pp. 701-706, 2012.

[6] Xiaojing Huang, Jayasri Joseph, Jian A. Zhang and Y. Jay Guo, "Sample rate conversion with parallel processing for high speed multiband OFDM systems", *IEEE Wireless Communications and Networking Conference*, pp. 2754-2759, 2013.

[7] Mahdi Mottaghi-Kashtiban, Saeed Farazi and Mahrokh G. Shayesteh, "The Optimum Configuration for Sampling Rate Conversion from CD to DAT using Multistage Interpolation and Decimation", *12th International Workshop on Systems, Signals and Image Processing*, pp. 101-104, 2005.

[8] V. Awasthi and K. Raj, "Power performance analysis of compensated Cascaded Integrator Comb (CIC) filter in optimum computing", *International Conference on Power and Energy in NERIST*, pp. 1-6, 2012.

[9] Min-Wei Qin, Jia-Mei Feng, Yuan-Cheng Yao and Jian-Hao Hu, "An improved wideband CIC filter design of software radio receivers", *International Journal of Wireless and Mobile Computing*, Vol. 5, No. 3, pp. 263-270, 2012.

[10] S. Cucchi, F. Desinan, G. Parladori and G. Sicuranza, "DSP implementation of arbitrary sampling frequency conversion for high quality sound application", *International Conference on Acoustics, Speech, and Signal Processing*, Vol. 5, pp. 3609-3612, 1991.

[11] Roger Woods, John McAllister, Gaye Lightbody and Ying Yi, "*FPGA-based Implementation of Signal Processing Systems*", John Wiley and Sons, Ltd., Publication, 2008.

[12] Guoan Bi and S. K. Mitra, "FFT-based sampling rate conversion", *IEEE Conference on Industrial Electronics and Applications*, pp. 428-431, 2012.

[13] M. Stala, C. Bilgin, R. Gangarajaiah and J. N. Rodrigues, "Hardware Implementation of an Iterative Sampling Rate Converter for Wireless Communication", *IEEE Global Telecommunications Conference*, pp. 1-6, 2010.

[14] D. Borkowski and A. Bien, "Improvement of Accuracy of Power System Spectral Analysis by Coherent Resampling", *IEEE Transactions on Power Delivery*, Vol. 24, No. 3, pp. 1004-1013, 2009.

[15] S. Yerramalli and U. Mitra, "Optimal Resampling of OFDM Signals for Multiscale-Multilag Underwater

Acoustic Channels", *IEEE Journal of Oceanic Engineering*, Vol. 36, No.1, pp. 126-138, 2011.

[16] G. Maalouli and D. R. Stephens, "Joint, fractional resampler with delay equalization for high synchronization accuracy with a reduced number of samples per symbol", *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 5, pp. V-349-V-352, 2004.

[17] U. Nithirochananont, S. Chivapreecha and K. Dejhan, "An FPGA-based implementation of variable fractional delay filters", *5th International Colloquium on Signal Processing and Its Applications*, pp. 104-107, 2009.

[18] C. Dick and F. Harris, "Options for arbitrary resamplers in FPGA-based modulators", *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers*, Vol. 1, pp. 777-781, 2004.

[19] J. P. Horridge, Frazer and J. Gordon, "Accurate arbitrary resampling with exact delay for radar applications", *International Conference on Radar*, pp. 123-127, 2008.