ASIC Implementation of a High Speed, Low Area Reconfigurable Decimation Filter

Ashish Raman
Department of ECE
Dr B R Ambedkar NIT
Jalandhar- 144011. India

Vignesh.V
Department of ECE
Dr B R Ambedkar NIT
Jalandhar- 144011. India

Deepti Kakkar
Department of ECE
Dr B R Ambedkar NIT
Jalandhar- 144011. India

ABSTRACT
Decimation filter is used to reduce the sampling rate for succeeding stages of an oversampling ADC. The speed of a successive-approximation ADC predominately depends on the decimator speed. This necessitates a need to design a high speed decimation filter to improve the overall system performance. A reconfigurable architecture is applied for the design of decimator to serve this purpose. Results show that delay of a Reconfigurable Decimator is reduced by 29.74% compared to a Normal Decimation filter. The Number of Slices and 4 Input LUTs are reduced by 6% and 7% each, which reduces the Area.

Keywords
oversampling, successive approximation ADC, reconfigurable architecture.

1. INTRODUCTION
When oversampling, far more samples are taken to interpret the desired frequencies than are necessary. A decimation filter which succeeds an oversampling ADC, filters off undesirable noise signal and steps up the resolution of the output. To design a high-speed, low area successive approximation system; a high-speed, low area decimator is essential. Reconfigurable architecture is exploited to develop this decimation filter.

2. DSP with FPGA
In electronic system design, the main attraction of microprocessors/microcontrollers is that it substantially decreases the risk of system development by reducing design complexity [3]. As the hardware is fixed, all of the design effort can be focused on developing the *code* which makes the hardware work to the needed system specification [2]. This situation has been complemented by the development of efficient software compilers which have largely removed the need for designer to create assembly language; to some extent, this can free the designer from having a detailed knowledge of the microprocessor architecture (although many practitioners would argue that this is essential to produce *good* code).

This concept has grown in popularity and embedded microprocessor courses are now essential parts of any electrical/electronic or computer engineering degree course.

A lot of this process has been down to the software developer's ability to exploit underlying processor architecture, the Von Neumann architecture. However, this advantage has also been the limiting factor in its application to the topic of this text, namely digital signal processing (DSP). In the Von Neumann architecture, operations are processed sequentially, which allows relative Straight forward interpretation of the hardware for programming purposes; however, this severely limits the performance in DSP applications which exhibit typically, high levels of parallelism and in which, the operations are highly data independent – allowing for optimizations to be applied.

This limitation is overcome in FPGAs as they allow what can be considered to be a second level of programmability, namely programming of the underlying processor architecture. By creating architecture that best meets the algorithmic requirements, high levels of performance in terms of area, speed and power can be achieved. In high volumes, ASIC implementations have resulted in the most cost effective, fastest and lowest energy solutions. However, increasing mask costs and impact of ―right first time system realization have made the FPGA, a much more attractive alternative. In this sense, FPGAs capture the performance aspects offered by ASIC implementation, but with the advantage of programmability usually associated with programmable processors. Thus, FPGA solutions have emerged which currently offer several hundreds of giga operations per second (GOPS) on a single FPGA for some DSP applications which is at least an order of magnitude better performance than microprocessors.

3. DIGITAL LOW PASS FILTER
Digital filters are used extensively in all areas of electronic industry. This is because Digital filters have the potential to attain much better signal to noise ratios than analog filters and at each intermediate stage the analog filter adds more noise to the signal, the digital filter performs noiseless mathematical operations at each intermediate step in the transform. Digital filters have emerged as a strong option for removing noise, shaping spectrum, and minimizing inter-symbol interference in communication architectures. These filters have become popular because their precise reproducibility allows design engineers to achieve performance levels that are difficult to obtain with analog filters.

The digital filter uses the oversampling and averaging algorithm to process the signal into higher resolutions [1]. The output of the digital filter and the decimation filter stages directly affects the resolution and output data rate of the ADC.

Generally, the digital filter is a finite impulse response (FIR) filter that essentially implements a weighted average on the digital output of the previous stage. A first-order FIR filter is actually an averaging machine.

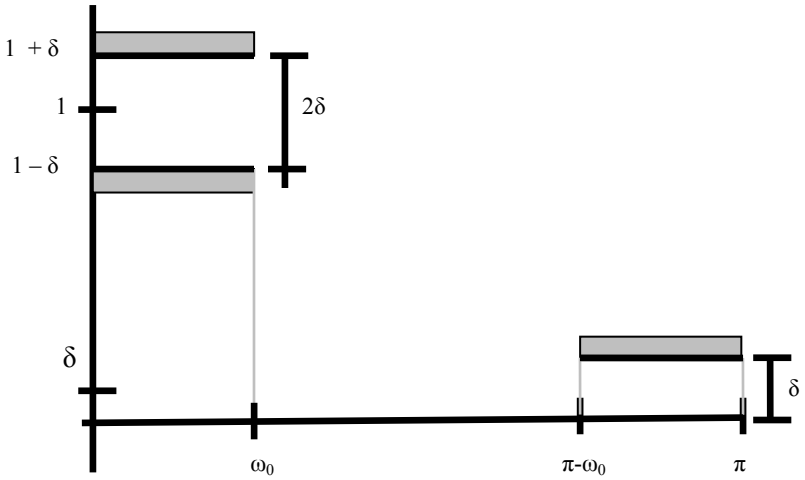

Figure 1: 2-fold decimation filter specification

The first-order FIR filter makes use of a moving average process. This averaging process reduces the uncertainty level in the output signal mathematically, but it does take time to acquire the samples. Theoretically, if you acquire 4 samples, you can change the digital output with two possibilities (0 and 1) to an output that has four possibilities (00, 01, 10, and 11). This is possible through oversampling mechanism and averaging process.

For instance, if 4 bits are acquired, the possibilities after averaging process are 0, 0.25, 0.5 and 0.75. Each oversample by a factor of four gives a 6db (or 1-bit) improvement in the SNR of ADC. So theoretically, given a 1-bit converter, a 2-bit converter can be mathematically realized with 4^1 averaged samples. A 3-bit converter can be realized with 4^2 (or 16) averaged samples. A 4-bit converter can be realized with 4^3 (or 64) averaged samples.

4. DECIMATION FILTER

A Decimator is employed to reduce the sampling rate for the stages succeeding the oversampling ADC to flow at slower rates. A decimation-by-two filter specification is shown in Fig. 1, where ω_0 is the passband frequency, and δ is the maximum ripple/alias magnitude. This specification limits the magnitude of error in the passband and attenuates the stopband to reduce aliasing of unwanted higher frequencies.

The aliasing caused by decimation can be described by

$$Y_D(z) = \frac{1}{M}\sum_{k=0}^{M-1} X\left(z^{1/M}\, W^k\right)$$

Where $W = e^{-j2\pi/M}$, and M is the degree of decimation [5]. The aliasing to the passband from high frequencies during decimation are at multiples of 2π/M. Decimators are M-tap linear-phase FIR filters which are devised by iteratively solving systems of linear inequalities.

The ripple magnitude, δ, is selected to preserve the desired resolution in passband after decimation. If a resolution of B bits is wanted, the passband should not vary by more than $\pm 2^{-(B+1)}$ to allow deviations of less than half an LSB. Both the passband-ripple and the aliasing from stopband induce deviance in the passband which has to be attenuated. Let δ $= 2^{-(B+1)}/M$ ensures that the combination of the passband ripple and stopband aliasing does not exceed half an LSB.

When considering multiple-stage filters, δ must be further constrained due to the additive effects of the filters in series. If $\delta << 1$, then the worst-case effect of the passband-ripple is to add the previous error,

$e = (1 + \partial_1)\cdot(1 + \delta_2) - 1 = \delta_1 + \delta_2 + \delta_1 \cdot \delta_2 \approx \delta_1 + \delta_2$ The aliasing error because of each back-to-back stage is also linear, so the sum of the magnitudes of all the errors in the poly-stage filter has to be less than $2^{-(B+1)}$.

The coefficients of the FIR filter are described by fixed-point binary numbers in the 2's complement format. In this format, b_s represents the sign of the number, and b_1 through b_B represent the digits of the binary number

$$x = b_s \cdot b_1 b_2 \cdots b_B$$

with the corresponding decimal value of [5]

$$x = -b_s + \sum_{i=1}^{B} b_i \cdot 2^{-i}$$

Round-off quantization noise is an issue in filter design. In FIR filters these errors cumulate as truncated numbers and are added together. A simple method of round-off is truncation to zero. The error caused by truncation is $-2^{-B} < e[n] \leq 0$ for each B-bit adder [6]. Given A to be the number of truncated values added in the filter (all truncated values, including those in the multipliers), the possible error becomes $-A.2^{-B} < e[n] \leq 0$. If each value in the filter has B_v bits, and the most significant of the B_v bits define the final B_d-bit output resolution of the filter, then letting $A < 2^{Bv - Bd}$, confines the error to less than one LSB of the final output [6]. Extending each value in the filter by $\log_2(A)$ bits helps reduce the errors due to truncation.

Reconfigurable decimator is designed to give output with a 14-bit resolution at 250 kHz. Each filter is designed to have a minimum number of taps so the total error remains within a 14-bit resolution using linear programming methods. For a successive approximation filter, using three 2-fold

decimators require the least number of multiples per second.

5. DECIMATOR USING MATLAB FDA TOOL

The filter coefficients were generated using the Matlab FDA tool for which the frequency response is as shown in Fig.2. The filter coefficients were later taken with fixed-point number representation system. The coefficients were later quantized to obtain only in integer format, which is multiplied, and then the response is taken. Filter response is taken by giving the specifications as cutoff frequency.

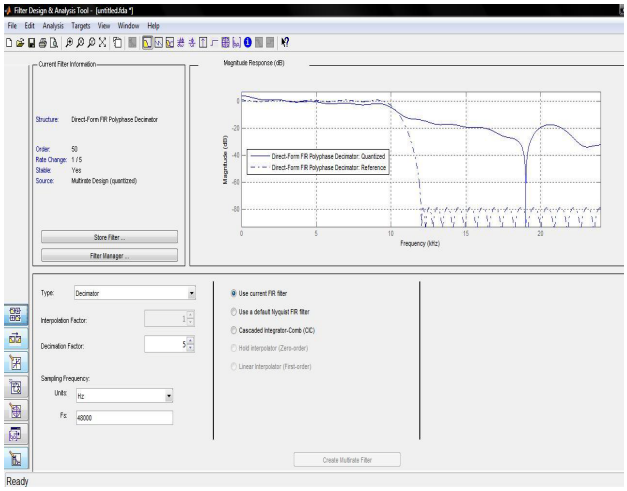

Figure 2: Frequency response of the filter generated using Matlab FDA Tool

6. RECONFIGURABLE ARCHITECTURE

Speed of a decimation filter depends on the rate reduction or decimation factor. By making the Decimation Factor Reconfigurable, the speed of the filter can be minimized further. The decimation factor is taken upto 16 in this paper.

A Control Block shown in Fig. 3 controls the reconfigurability of the Decimation Filter. The Decimation Factor is controlled by the signal we give in the select line pin.

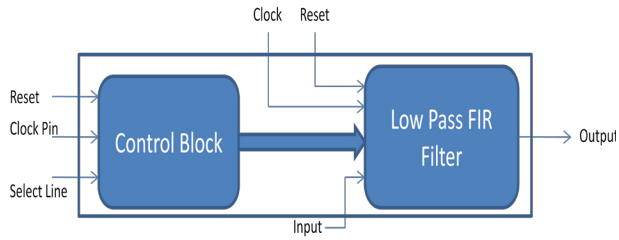

Figure 3: Reconfigurable Architecture

7. RESULTS AND DISCUSSIONS

From the table shown below it's clear that Reconfigurable Decimator architecture has less delay and area. Delay and Area calculation is done using Xilinx ISE 10.1. Fig. 8 & 10 shows the simulation result of Reconfigurable Decimator. This architecture is implemented into Spartan 3E, Device-XC3S250E, package – FT256, speed-5

Table 1: Delay comparisons for Reconfigurable and Normal Architectures

Type of Architecture	**Delay (ns)**
Reconfigurable Decimator	28.470
Normal Decimator generated by FDAtool	36.937

Table 2: Area comparisons for Reconfigurable and Normal Architectures

Type of Architecture	**Number of Slices**	**Number of 4 input LUTs**
Normal Decimation Filter	563 out of 2448 22%	1051 out of 4896 21%
Reconfigurable Decimation Filter	392 out of 2448 16%	731 out of 4896 14%

7.1 Speed

As seen in Table 1, the speed of reconfigurable decimator is much higher than a normal one. The values were calculated for a decimation factor 4. Fig. 7 & 8 shows the results obtained.

7.2 Area

By using Reconfigurable Architecture the area is reduced as shown in Table 2 compared to Normal Decimator. Screen shots are shown in Fig 9 & 10.

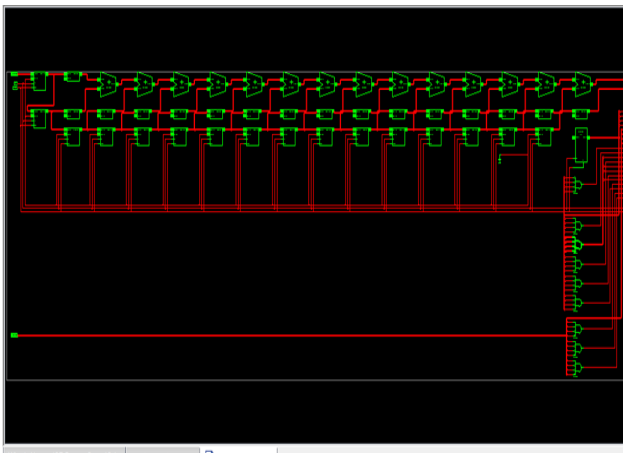

Figure 4: RTL Design of Reconfigurable Decimator

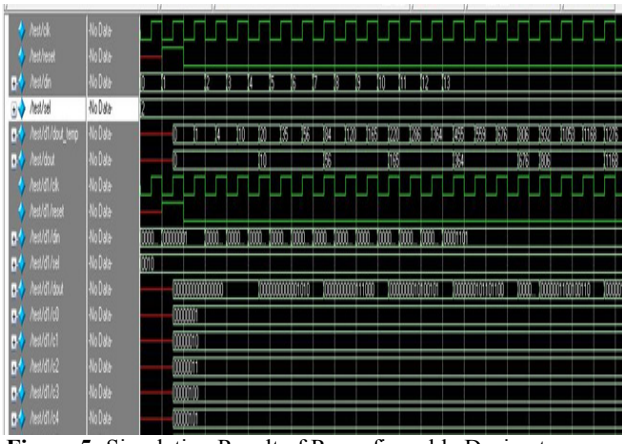

Figure 5: Simulation Result of Reconfigurable Decimator with Decimation Factor 2

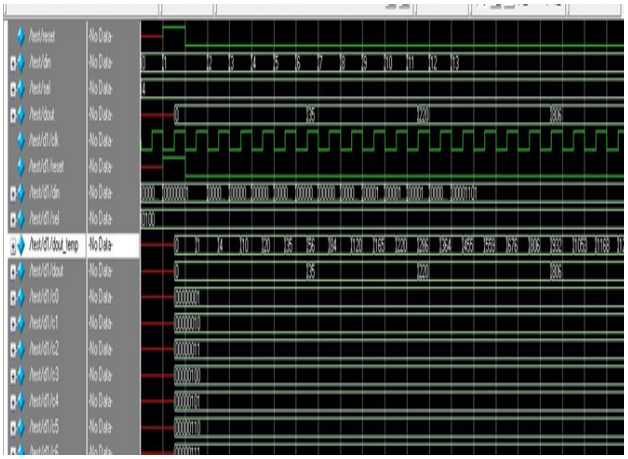

Figure 6: Simulation Result of Reconfigurable Decimator with Decimation Factor 4

```
reset                              | IBUF              | 273   |
-----------------------------------+-------------------+-------+

Timing Summary:
---------------
Speed Grade: -5

   Minimum period: 36.937ns (Maximum Frequency: 27.073MHz)
   Minimum input arrival time before clock: 2.727ns
   Maximum output required time after clock: 4.040ns
   Maximum combinational path delay: No path found

Timing Detail:
--------------
All values displayed in nanoseconds (ns)

=========================================================================
```

Figure 7: Delay calculation of Normal Decimator using Xilinx ISE 10.1

```
Clock Information:
------------------
-----------------------------------+-------------------+-------+
Clock Signal                       | Clock buffer(FF name) | Load  |
-----------------------------------+-------------------+-------+
clk                                | BUFGP             | 150   |
-----------------------------------+-------------------+-------+

Asynchronous Control Signals Information:
-----------------------------------------
No asynchronous control signals found in this design

Timing Summary:
---------------
Speed Grade: -5

   Minimum period: 28.470ns (Maximum Frequency: 35.120MHz)
   Minimum input arrival time before clock: 7.379ns
   Maximum output required time after clock: 4.370ns
   Maximum combinational path delay: No path found
```

Figure 8: Delay calculation of a Reconfigurable Decimator using Xilinx ISE 10.1

```
#      FDCE                          : 273
# Clock Buffers                      : 1
#      BUFGP                         : 1
# IO Buffers                         : 51
#      IBUF                          : 18
#      OBUF                          : 33
# MULTs                              : 9
#      MULT18X18SIO                  : 9
=========================================================================

Device utilization summary:
---------------------------

Selected Device : 3s250eft256-5

 Number of Slices:                      563  out of   2448
 Number of Slice Flip Flops:            273  out of   4896
 Number of 4 input LUTs:               1051  out of   4896
 Number of IOs:                          52
 Number of bonded IOBs:                  52  out of    172
 Number of MULT18X18SIOs:                 9  out of     12
 Number of GCLKs:                         1  out of     24

---------------------------
Partition Resource Summary:
---------------------------
```

Figure 9: Area measurement using of a Normal Decimator using Xilinx ISE 10.1

```
#      XORCY                         : 121
# FlipFlops/Latches                  : 150
#      FDR                           : 134
#      FDRE                          : 16
# Clock Buffers                      : 1
#      BUFGP                         : 1
# IO Buffers                         : 29
#      IBUF                          : 13
#      OBUF                          : 16
=========================================================================

Device utilization summary:
---------------------------

Selected Device : 3s250eft256-5

 Number of Slices:                      392  out of   2448    16%
 Number of Slice Flip Flops:            150  out of   4896     3%
 Number of 4 input LUTs:                731  out of   4896    14%
 Number of IOs:                          30
 Number of bonded IOBs:                  30  out of    172    17%
 Number of GCLKs:                         1  out of     24     4%

---------------------------
Partition Resource Summary:
---------------------------
```

Figure 10: Area measurement using of a Normal Decimator using Xilinx ISE 10.1

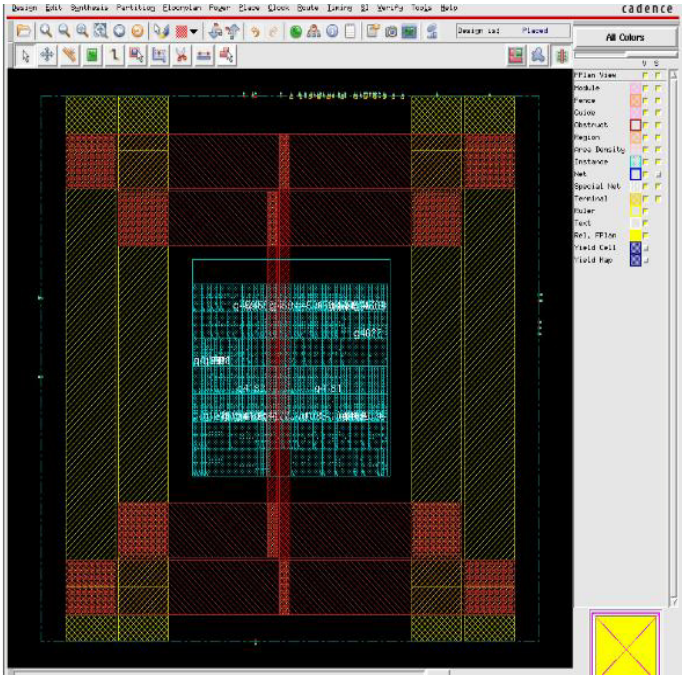

Figure 11: Layout Design of a Reconfigurable Decimation Filter using Cadence Encounter Tool

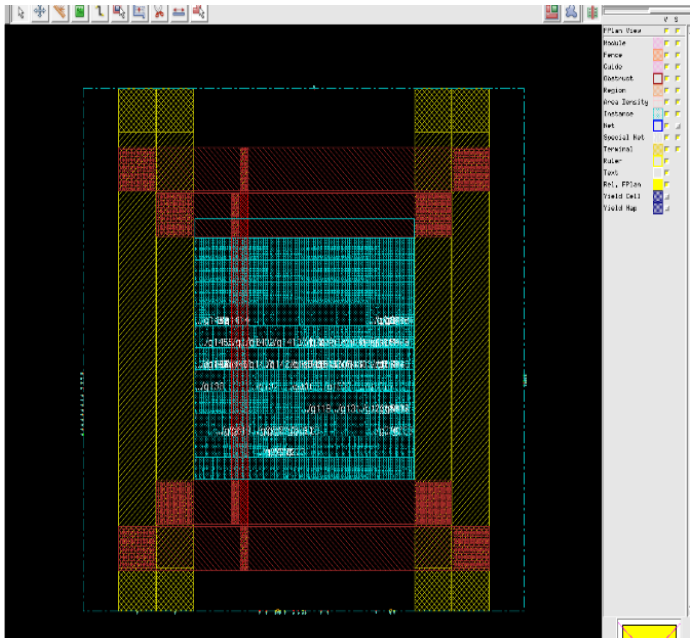

Figure 12: Layout Design of a Normal Decimation Filter using Cadence Encounter Tool

8. CONCLUSION

By using a reconfigurable architecture for a decimator the performance of a decimation filter increases substantially. Since the decimator is a vital part of any oversampling ADC system, the system performance is also increased. The whole system as such, can be used in the receiver end of a software defined radio (SDR) or it can also be applied to design an ADC for Bio-MEMS sensors.

9. REFERENCES

[1] Mohamed Ali Mahdi Eshtawie and Masuri Bin Othman, "An Algorithm proposed for FIR Filter Coefficients Representation", International Journal of Applied mathematics and Computer Sciences, Vol 4 No. 1, 2008, pp24-pp30.

[2] E.Abu-Shama, M.B.Maaz, and M.A.Bayoumi, A fast and low power multiplier architecture. Proceedings of the 39th Midwest Symposium on Circuits and Systems (1997), pp.26–32.

[3] R. Veljanovski, J.Singh and M. Faulkner "DSP and ASIC Implementation of A Channel Filter For A 3G Utra-Tdd System" Personal, Indoor and Mobile Radio Communications. The 13th IEEE International Symposium, 2002 Vol 3, pp 1447-1451

[4] P. P. Vaidyanathan, *Multirate systems and Filter Banks*, Prentice Hall, Inc., 1993.

[5] A. Oppenheim; R. Schafer, *Discrete-Time Signal Processing: 3rd Edition,* Prentice Hall, 2009.

[6] Clive Maxfield "The design warrior's guide to FPGAs: devices, tools and flows".