Design of a Costas Loop Down Converter

Mike Roddewig & Seyed A. Zekavat
Dept. of Electrical and Computer Engineering,
Michigan Technological University,
Houghton, MI 49931
Email: {mrroddew,rezaz}@mtu.edu

Saeid Nooshabadi
Dept. of Information and Communications
Gwangju Institute of Science and Technology
Republic of Korea, 500-712
Email: saeid@gist.ac.kr

***Abstract*—This paper presents the the design of a digital down converter (DDC) using a digital Costas loop in a field programmable gate array (FPGA). To reduce the power dissipation we use the CIC filter to perform an efficient decimation, and then follow it with a finite impulse response (FIR) compensation filter that runs at a reduced sampling rate. The final results and performance measures are quantified and discussed. The BER performance of the Costas loop on both floating and fixed point implementations are identical.**

***Index Terms*—Digital down converter, Costas loop, binary phase shift keying**

I. Introduction

Costas loop is a phase-locked loop used for carrier phase recovery for Binary phase shift keying (BPSK). It provides an excellent performance for BPSK, which in terms of noise immunity per unit bandwidth, is one of the most efficient binary data modulation techniques. loop [1], [2] This paper will focus on the implementation of the Costas loop digital down converter. It describes the process of designing and building an all-digital FPGA implementation of the classic Costas loop for a wireless system. This implementation will be used in the base station receiver and converts the digital intermediate frequency (IF) from the analog-to-digital (ADC) to baseband.

Similar works have detailed the implementation of a Costas loop in FPGA [3], [4]. In [4] a Costas loop implementation using direct frequency synthesis that employs the Coordinate rotation digital computer (CORDIC) algorithm [5] is presented. A CORDIC algorithm is commonly used when no hardware multipliers are available, as it relies only on simple operations such as addition and subtraction [6]. However, modern FPGAs contain numerous embedded multipliers [7]. Therefore, we chose to employ multipliers in the implementation of a direct digital synthesizer or numerically-controlled oscillator (NCO). The work in [3] describe an improved implementation of the classic Costas loop that uses decimation to reduce the sampling rate of the arms after the low pass filtering and a loop filter that is implemented entirely without multipliers.

In this paper, we improve the implementation of the Costas loop using the cascaded integrator comb (CIC) filter, first described in [8]. Rather than performing the low pass filtering prior to the decimation, as done in [3], we use the CIC filter to perform an efficient decimation, and then follow it with a finite impulse response (FIR) compensation filter that runs at a reduced sampling rate and, in cascade with the CIC filter, has excellent characteristics.

II. Digital Costas Loop Theory

The general theory behind the operation of the Costas Loop has already been well described in the literature [1], [2] and we will not discuss it here. However, the implementation of the Costas Loop in FPGA, particularly of the interface of the phase error signal and the NCO has not been well covered.

The output frequency f_0 of an NCO is expressed as [9],

$$f_0 = \frac{\Delta\theta \cdot f_s}{2\pi \cdot 2^N} \tag{1}$$

where f_s is the sampling frequency, N is the word–length of the phase accumulator, and $\Delta\theta$ is an input phase increment. The discrete phase increment $\Delta\theta[n]$ from a given error at the phase detector output $e[n]$ after the the loop filter and a center frequency f_c of the carrier signal is

$$\Delta\theta[n] = 2\pi \cdot 2^N \left(\frac{f_c}{f_s} + e[n]\right) \tag{2}$$

We can represent the phase accumulator as a summation of the phase increment values mod $\left(2\pi \cdot 2^N\right)$. That is:

$$\theta[n] = \sum_{i=0}^{n} \Delta\theta[i] \pmod{2\pi \cdot 2^N} \tag{3}$$

And, the outputs of the NCO are

$$\text{NCO Cos}[n] = \cos\left(\frac{\theta[n]}{2^N}\right) \tag{4}$$

$$\text{NCO Sin}[n] = \sin\left(\frac{\theta[n]}{2^N}\right) \tag{5}$$

Substituting (2) into (3) and simplifying yields

$$\theta[n] = 2\pi \cdot 2^N \left[n\left(\frac{f_c}{f_s}\right) + \sum_{i=0}^{n} e[i]\right] \pmod{2\pi \cdot 2^N} \tag{6}$$

And then substituting (6) into (4) and (5) yields the output of the NCO at a given sample n as

$$\text{NCO Cos}[n] = \cos\left(2\pi\left[n\frac{f_c}{f_s} + \sum_{i=0}^{n} \phi[i]\right]\right) \tag{7}$$

$$\text{NCO Sin}[n] = \cos\left(2\pi\left[n\frac{f_c}{f_s} + \sum_{i=0}^{n} \phi[i]\right]\right) \tag{8}$$

TABLE I
SIGNAL PARAMETERS

Modulation	BPSK
Carrier Frequency	70 MHz
Bit Rate	1 MHz
ADC Sampling Frequency	40 MHz
Post-ADC Carrier Frequency	10 MHz
Post-Decimation Sampling Frequency	10 MHz

The error signal at the output of the loop filter is given as [1]

$$e(t) = A^2(t)\sin 2\left(\phi - \hat{\phi}\right) \tag{9}$$

where $A(t)$ is the amplitude of the incoming carrier signal, ϕ is the phase of the NCO output, and $\hat{\phi}$ is the phase of the carrier signal. Normalizing $A[n-1] = \pm 1$ and absorbing it in the loop gain the equivalent expression in the digital domain is

$$\begin{aligned} e[n] &= A^2[n-1]\sin 2\left(\phi[n-1] - \hat{\phi}[n-1]\right) \\ &= \sin 2\left(\phi[n-1] - \hat{\phi}[n-1]\right) \end{aligned} \tag{10}$$

When the Costas loop locks, the difference between the phase of the NCO and the carrier is small [1] and thus

$$\sin 2\left(\phi[n-1] - \hat{\phi}[n-1]\right) \approx 2\left(\phi[n-1] - \hat{\phi}[n-1]\right) \tag{11}$$

Using this approximation, the phase error in (10) simplify to,

$$e[n] = 2\left(\phi[n-1] - \hat{\phi}[n-1]\right) \tag{12}$$

Thus, the input phase to the NCO is the difference of the carrier phase and the NCO phase of the previous sample and the loop will correct the phase error.

III. BASIC DESIGN

The initial model was designed and tested in Simulink, a software package from The Mathworks for modeling, simulating, and implementing dynamic systems. The parameters of the carrier signal are given in Table I.

Figure 1 shows the initial model in Simulink that assumes a pre analog to digital converter (ADC) image rejection filtering in analog domain with 20MHz pass band centered at 70 MHz.

A. Loop Filter Design

The loop filter was implemented entirely without multiplications and uses only binary shifts. In binary, a right shift by n represents an equivalent multiplication by 2^{-n}. The original loop filter was designed to be a first order Butterworth infinite impulse response (IIR) filter with a passband of 1 MHz. The original transfer function is given as

$$H(z) = 0.245 \cdot \frac{1 + z^{-1}}{1 - 0.509z^{-1}} \tag{13}$$

In the loop filter on FPGA $0.245 \approx 2^{-2}$ which is equivalent to a right shift by two and $-0.509 \approx -2^{-1}$ which is equivalent to a right shift by one followed by a negation. Figure 2 plots the frequency response of the original loop filter and the loop filter implemented with shifts. Figure 3 is the loop filter as implemented on FPGA.

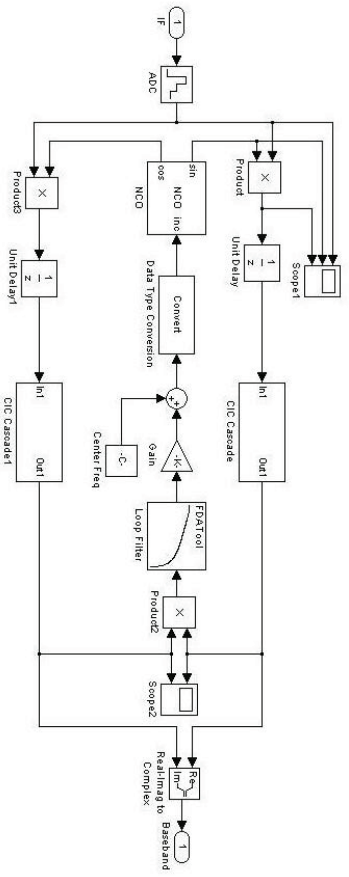


Fig. 1. Simulink Model of the Costas Loop

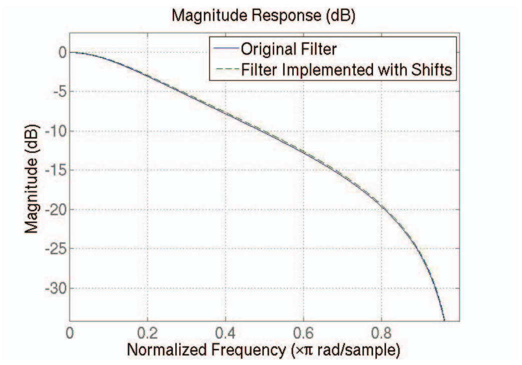
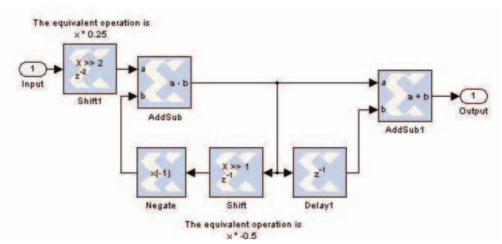


Fig. 2. Loop Filter Frequency Response Comparison



The equivalent operation is x * 0.25
Input
Shift1
AddSub
a - b
AddSub1
a + b
Output
Negate
Shift
Delay1
The equivalent operation is x * -0.5

Fig. 3. Loop Filter in FPGA

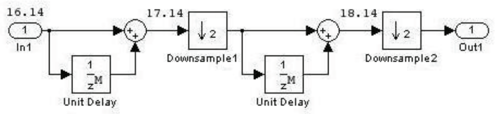


Fig. 4. First order decimate by four CIC filter

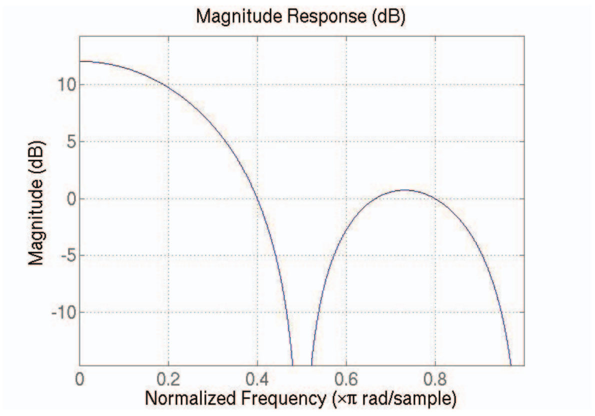


Fig. 5. CIC Filter Frequency Response

B. CIC Filter and Compensator Design

With a bit rate of 1 MHz, We desired to reduce the sampling rate of 40 MHz to 10 MHz, a decimation by four. We decided to use a CIC filter to perform the decimation. CIC filters are a class of light-weight narrowband lowpass filters (requiring no multipliers) that are commonly used in decimation and interpolation [8]. Traditional CIC filters have a word–length growth problem where each integrator must add an additional NR bits, where N is the differential delay, and R is the sample rate change. The work in [10] describes a technique that eliminates the recursive integrator structures of the traditional CIC filter and reduces the word–length growth rate to M bits per stage, where M is the order of the filter. A stage is defined as a filter followed by a decimation . Using this technique, we designed the first (M=1) order CIC filter in Figure 4 that performs a decimation by four. The frequency response of the CIC filter is plotted in Figure 5.

To compensate for the sloping passband and extremely high DC gain of 12.5 dB, we designed a 12-tap FIR compensation filter to follow the CIC filter with a normalized passband of 0.25. Its frequency response is plotted in Figure 6 and the combined response of the two filters is plotted in Figure 7. The passband ripple of the filter cascade is about 3 dB.

The average lock time of the Costas loop, computed over 10 trials, where the VCO starting phase/frequency and the modulation data were randomized, is 17.2 symbol times, 7.2 times more than that of the floating point model, (at 1 MHz symbol rate).

IV. Conversion to Fixed Point and FPGA Synthesis

After the completion of the Costas loop design that performed satisfactory using the floating point model, we carried out the conversion to the fixed point model that is suitable for implementation on an FPGA platform. Using the Fixed Point Tool, the floating point model was analyzed while operating on a test signal and a size for the number of fractional bits

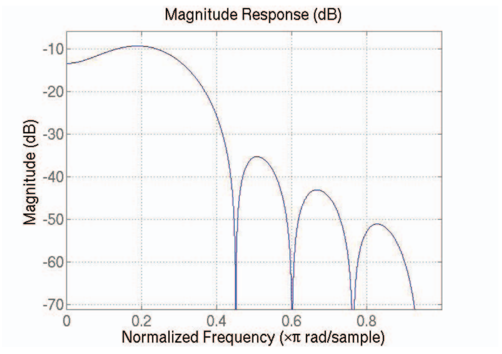


Fig. 6. Compensation Filter Frequency Response

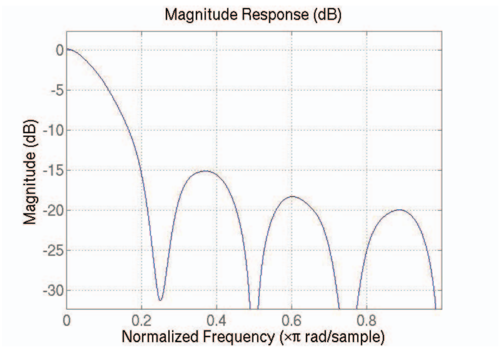


Fig. 7. CIC and Compensation Filter Frequency Response

was determined. After the creation of the fixed point model, it was compared to the floating point version to verify that its performance had not been degraded by a significant amount (some degradation is to be expected).

First, a word–length of 16 was chosen to be used throughout the design. Then, the Fixed Point Tool was run to estimate binary fraction lengths in the model. The Fixed Point Tool records the maximum, minimum, and average values of signals in a model during simulation and then uses this data to estimate the fraction lengths for a chosen word–length. When the model was run with test data, the Fixed Point Tool recommend a fraction length of 14 in the arm multipliers and filters, and a fraction length of 16 for the phase multiplier and the loop filter. After the loop filter the phase error is multiplied by a scaling value causing the length to grow to 32 bits, with 16 fractional that are truncated out. The blocks that convert the phase error into a phase increment for the NCO does not use fractions. Full precision arithmetic is not used in the design, and excess least significant bits (LSBs) are truncated to fit the intermediate data to 16-bit size. The CIC filter is the only occasion where the word–length grows to values 17 and 18 bits as shown in Figure 4.

The fixed point model described above is used as the reference model for synthesis of the design in FPGA. To target the module for FPGA, we chose to use Xilinx System Generator, which provides a Simulink blockset that is then converted to VHDL or Verilog for synthesis and implementation.

The most important consideration in System Generator (and for synthesis in FPGA in general) is the computation time of operations. In Simulink, gains and multiplications take zero time to compute, but in FPGA multipliers are registered and

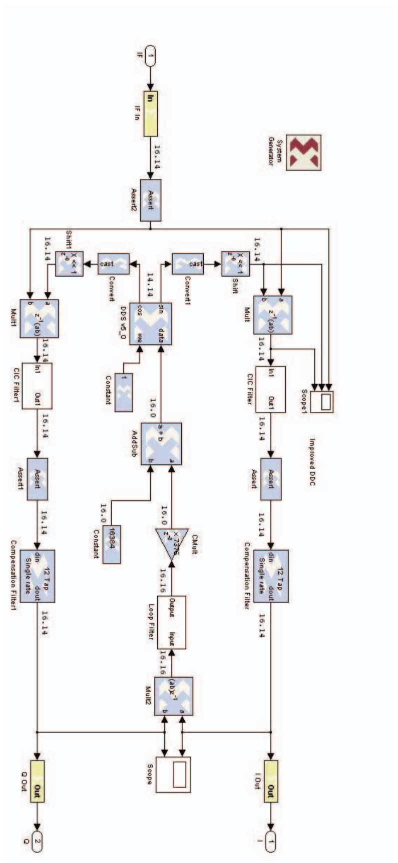


Fig. 8. System Generator DDC

require multiple clock cycles to computer (in our design, one or two clock cycles). Therefore, the FPGA Costas loop takes a longer time to lock because of the added delay in the feedback loop, but otherwise, performance is not affected. Although the sampling rate of the FIR compensation filters is 10 MHz, they run at a clock rate 4 times higher. This is to use the same multiplier to compute the output for several taps of the filter during one sample period of the filter ("folding"). In the final design, we used a folding factor of four, so the 12 taps of the filter only use three multipliers in hardware. Figure 8 shows the completed model. Note that data format for all the blocks in $\{word\text{–}length\}.\{fraction\}$ is are indicated on each block.

V. Performance and Resource Usage

The bit error rate (BER) performance of the DDC is presented in Figure 9 for the floating point and fixed point models for a BPSK receiver with differential encoding, no channel coding, and perfect synchronization in an AWGN channel. The `berawgn` function in MATLAB is used to generate the floating model[11], [12], [13]. The comparison data for the fixed point models were obtained using the System Generator models, which accurately model the real performance of the design in FPGA.The match between the performance of the floating point and fixed point models is perfect.

The resource usage, power consumption and maximum clock frequency are summarized in Table II . These results are from implementation in a Xilinx Virtex-4 SX35 FPGA using ISE 10.1.

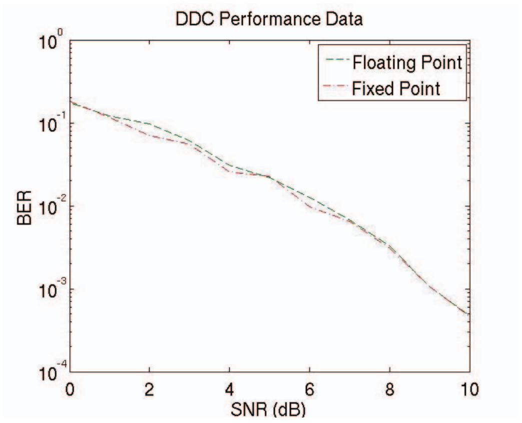


Fig. 9. BER Performance

TABLE II
Resource usage

Word Size	Flip Flops	Look Up Tables	Multipliers DSP48 slices	CLK MHz	Power mW
16	1073	788	9	102(max)	892
				40(used)	330

VI. Conclusion

This paper presented an efficient FPGA implementation of a DDC using a digital Costas loop as part of a wireless system. The final 12-bit and 16-bit fixed point implementations performs as good as floating point model. The FPGA implementation can run at a maximum clock rate of 102 MHz with power dissipation of only 829 mW. It utilizes 1073 flip flops, 788 Look Up Tables and 9 embedded multipliers.

References

[1] J. Proakis, *Digital Communications*, 4th ed. McGraw-Hill, 2001, pp. 356–357.

[2] B. P. Lathi, *Modern Digital and Analog Communication Systems*, 3rd ed. Oxford University Press, 1998, pp. 187–188.

[3] M. Xiao and T. Cheng, "Improved implementation of costas loop for dqpsk receivers using fpga," *Int. Conf. on Comm. Tech. (ICCT)*, pp. 1–4, 2006.

[4] A. Shoari, M. Kamarei, and A. Radmand, "Implementation of costas loop using cordic algorithm for software radio applications," *IEE Proceedings on Communications*, vol. 152, no. 1, pp. 113–118, 2005.

[5] E. Grayver and B. Daneshshrad, "Direct digital frequency synthesis using a modified cordic," in *Proc. of IEEE Int. Symp. on Cir. and Sys. CA*, May 1998, pp. 241–244.

[6] R. Andraka, "A survey of cordic algorithms for fpga based computers," *Proc. of ACM Int. Symp. on Field Programmable Gate Arrays*, pp. 191–200, 1998.

[7] X. Inc., *Virtex-4 FPGA User Guide*.

[8] E. Hogenauer, "An economical class of digital filters for decimation and interpolation," *IEEE Transactions on Acoustics, Speech, and Signal Processing,*, vol. 29, no. 2, pp. 155–162, 1981.

[9] T. M. Inc., *Signal Processing Blockset: NCO*.

[10] R. G. Lyons, *Understanding Digital Signal Processing*, 2nd ed. Prentice Hall PTR, 2004.

[11] K. Cho and D. Yoon, "On the general ber expression of one- and two-dimensional amplitude modulations," *IEEE Trans. on Commun.*, vol. 50, no. 7, 2002.

[12] P. J. Lee, "Computation of the bit error rate of coherent m-ary psk with gray code bit mapping," *IEEE Trans. on Commun.*, vol. 34, no. 5, 1986.

[13] M. K. Simon, "On the bit-error probability of differentially encoded qpsk and offset qpsk in the presence of carrier synchronization," *IEEE Trans. on Commun.*, vol. 54, no. 5, 2006.