Digital Down Converter Optimization

Joe Gerhardt and Saiyu Ren
Department of Electrical Engineering, Wright State University, USA
{gerhardtjoe@gmail.com, saiyu.ren@wright.edu}

***Abstract* – A Digital Down Converter (DDC) using four times intermediate frequency ($4f_{IF}$) sampling is analyzed to simplify the required hardware. The DDC is controlled by a two bit counter. One counter bit controls the mixer, implemented as a two input multiplexer, while the other bit controls data flow in direct or transposed form finite impulse response (FIR) filters. The resulting FIR filters significantly reduce the number of multipliers and adders required while still allowing additional filter reduction techniques to be applied.**

I. Introduction

The digital down converter (DDC) is an architecture that shifts an intermediate frequency input signal down the spectrum to a lower frequency, or to baseband. The input comes from an analog-to-digital converter (ADC) and is centered at the intermediate frequency, f_{IF}. It is particularly useful because once the signal is shifted down the spectrum and filtered for reshaping and anti-aliasing purposes, the sampling rate can be reduced so that intensive digital signal processing (DSP) algorithms can process the data [1].

The general form of a DDC, shown in Fig. 1, requires a multiplication for the mixer stage and many more in certain filter implementations. Multipliers are very costly hardware: large area, limited speed, and high power consumption, especially in high-rate implementations. Furthermore, better filter performance requires even more hardware, usually due to increased filter length. For these reasons, hardware-efficient DDC architectures are extensively pursued. This paper proposes a major hardware reduction in the filter stage which is achievable because of the chosen mixer implementation.

II. Background of DDC Implementations and Reductions

A. Mixer Stage

One way to implement the digital mixer is by generating the sine and cosine values using a direct digital synthesizer (DDS), which operates as the local oscillator (LO). Without the need for a changing LO frequency, f_{LO}, the DDS is not a very efficient choice for implementing the LO.

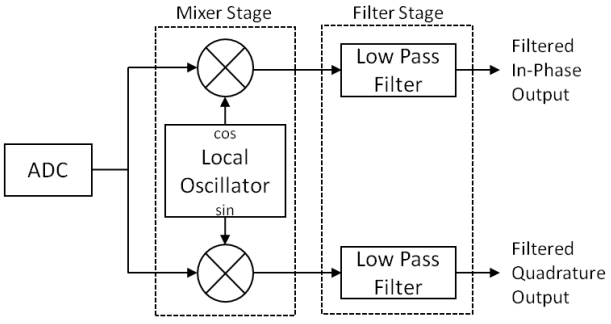

Fig. 1. General DDC architecture

To save area, the LO may be implemented as a lookup table. The number of unique sample values per period determines the size of the lookup table. Using a lookup table costs less hardware than the DDS, but the mixer still requires a multiplier. Sampling the LO at $4f_{IF}$ results in the four values illustrated in Fig. 2 being outputted for every period of f_{IF}.

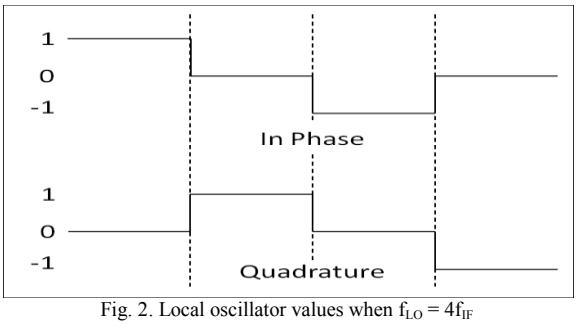

Fig. 2. Local oscillator values when $f_{LO} = 4f_{IF}$

It can be seen that the mixer output for either I (in phase) or Q (quadrature) signal will simply be zero or ±X(t), where X(t) is the input to the mixer from the ADC at time t in two's complement representation. Furthermore, when the in-phase LO output is non-zero, the quadrature LO output is zero, and vice versa. Fig. 3(a) shows the mixer with the LO implemented as outputting each value from memory.

References [2,3,4,5] use $4f_{IF}$ sampling, and [2] implements the mixer as a simple multiplexer. This implementation is possible because, besides control logic, zero and X(t) require no additional logic to output; -X(t) simply requires a two's complement conversion. Thus, a multiplierless mixer architecture can be realized using a multiplexer controlled by a two bit counter. The most significant bit, CNT1, of the counter will be a square wave operating at two times f_{IF}. The least significant bit, CNT0, of the counter will be a square wave operating at four times f_{IF}. Odd-numbered clock cycles (first, third, etc.) will have CNT0 equal to '0', and even-numbered clock cycles (second, fourth, etc.) will have CNT0 equal to '1'. With this implementation, shown in Fig. 3(b), CNT1 and CNT0 are used as control logic to avoid the use of a multiplier in the mixer stage.

Later in this paper, filter architecture is changed so that zero is no longer a necessary input to the mixer multiplexer. Instead, the filter uses CNT0 to control data to achieve the equivalent function. Fig. 3(c) shows the mixer implemented without zero as an input.

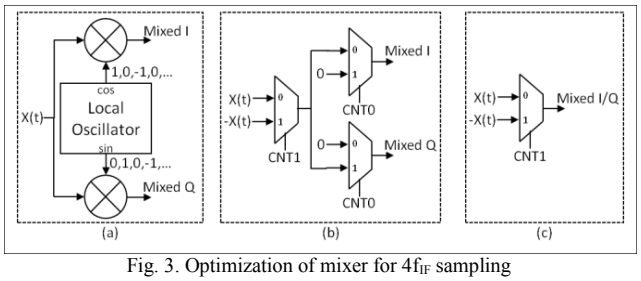

Fig. 3. Optimization of mixer for $4f_{IF}$ sampling

B. Filter Stage

For I/Q mixing, to maintain the phase relationship between in-phase and quadrature signals, any operation performed on the in-phase signal must also be performed on the quadrature signal, including filtering. Conventional DDC architectures use duplicate filters, one for each I/Q signal, to ensure the same operations are performed on each signal, doubling the required hardware for filtering the mixer's output.

Depending on the desired spurious-free dynamic range (SFDR) and transition width, the filter can demand a large amount of hardware resources. Table I shows estimated required hardware based on Simulink's Low Pass Filter block using a single-rate, equiripple design with a 1 dB passband ripple and 12 MHz passband edge.

TABLE I. HARDWARE REQUIREMENTS FOR LOW PASS FILTER

Transition Width (MHz)/ Stopband Attenuation (dB)	Hardware Required		
	Adders	*Multipliers*	*States*
3 / 50	63	64	63
3 / 60	73	74	73
5 / 50	38	39	38
5 / 60	45	46	45

As can be seen in Table I, designing an efficient digital filter that meets the performance requirements is paramount to saving hardware. However, regardless of technique, the filters will account for the majority of the DDC hardware. This paper focuses on reducing hardware in direct and transposed form finite impulse response (FIR) filters.

Reference [2] reduces hardware in its filters by extracting filter blocks that are common to different stages. Reference [6] reduces hardware in a transposed FIR filter design by applying a novel subexpression elimination algorithm to optimize the multiplier block, which replaces multipliers with adders due to multiplication by a constant (each new input X(t)). However, the filter hardware can be further reduced, in both examples, due to the chosen mixer implementation.

This paper introduces a very significant reduction of hardware in the filter stage by analyzing the computations of each block in the filter and removing excess blocks due to unnecessary computations and storage. Reference [5] makes a similar reduction, but uses an interpolated FIR filter implementation which increases the number of delay elements, whereas the proposed design is given as direct/transposed form FIR and does not introduce extra delays. The design in [5] also uses only half of the coefficients for either I or Q channel, whereas the proposed design uses all of the coefficients for both channels. Furthermore, this reduction technique does not interfere with many other hardware reductions, including those in [2] and [6]. Direct and transposed form FIR filter design reductions will be discussed, and the maintainability of aforementioned existing reductions will be explained.

III. PROPOSED METHOD OF HARDWARE REDUCTION

A. Direct Form FIR Reduction

Consider a direct form FIR filter implementation of a low pass filter (LPF) with four taps for simplicity, shown in Fig. 4. Due to sampling f_{LO} at $4f_{IF}$, many zeros are inputted to the filter, and the filter hardware may be reduced as a result.

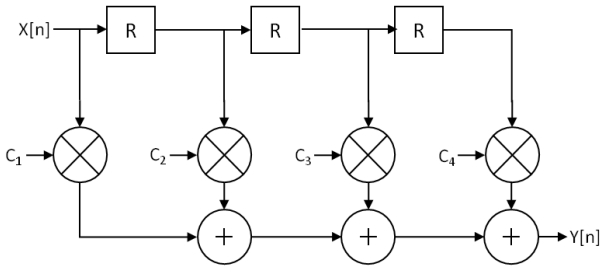

Fig. 4. Direct FIR filter with four taps.

The filter's output, Y_n, can be expressed as (1), where n is the current clock cycle and C_i denotes coefficient i. The output of the mixer for odd-numbered cycles will be zero for the mixed Q signal, while the output of the mixer for even-numbered cycles will be zero for the mixed I signal. Thus, (2) shows the in-phase filter output during odd-numbered cycles as well as the quadrature filter output during even-numbered cycles. Equation (3) shows the in-phase filter output during even-numbered cycles as well as the quadrature filter output during odd-numbered cycles. Thus, the output for an FIR filter of even-numbered length, M, is given in (4) and (5). Equations (4) and (5) alternate each clock cycle to simulate zeros propagating through the filter; this means that X_n in (4) will be X_{n-1} in (5) due to the new clock cycle and mixer output.

$$Y_n = X_n*C_1 + X_{n-1}*C_2 + X_{n-2}*C_3 + X_{n-3}*C_4 \quad (1)$$

$$Y_n = X_n*C_1+0*C_2+X_{n-2}*C_3+0*C_4 = X_n*C_1 + X_{n-2}*C_3 \quad (2)$$

$$Y_n = 0*C_1+X_{n-1}*C_2+0*C_3+X_{n-3}*C_4 = X_{n-1}*C_2 + X_{n-3}*C_4 \quad (3)$$

$$Y_n = X_n*C_1 + X_{n-2}*C_3 + \ldots + X_{n-(M-2)}*C_{M-1} \quad (4)$$

$$Y_n = X_{n-1}*C_2 + X_{n-3}*C_4 + \ldots + X_{n-(M-1)}*C_M \quad (5)$$

The non-zero values of X are simply switching coefficients for multiplications. If the multipliers and adders are reused to process the previous X with a different coefficient during clock cycles when X is zero, filter hardware can be greatly reduced. This is achieved by clocking in only non-zero inputs to the filter and switching coefficients for multiplication depending on the value of CNT0. Finally, the reduced hardware

equivalent of the direct form FIR filter is shown in Fig. 5. For the in-phase filter, the enable pin (En) for the registers will be CNT0 inverted (CNT0_BAR), and the select line (S) will be CNT0. For the quadrature filter, enable will be CNT0, and the select line will be CNT0_BAR. The differences of enable pins and select lines between filters is reflective of the odd-even relationship of non-zero values for in-phase and quadrature signals.

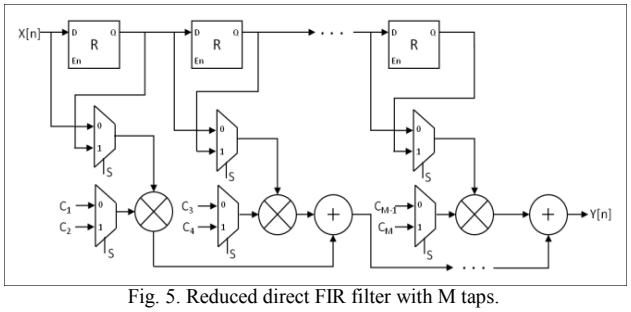

Fig. 5. Reduced direct FIR filter with M taps.

B. *Transposed Form FIR Reduction*

Now consider a transposed form FIR filter implementation of the LPF, again with four taps for simplicity, shown in Fig. 6. Equations (1), (2), (3), (4), and (5) are true for the transposed form FIR filter as well. The reduction of the hardware can be done differently, however, due to the placement of the registers.

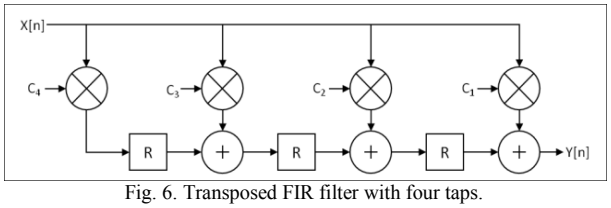

Fig. 6. Transposed FIR filter with four taps.

Recall that the mixer output will be zero for the mixed Q signal when CNT0 is '0', while the output of the mixer will be zero for the mixed I signal when CNT0 is '1'. This results in all multipliers in one filter multiplying by zero while the other filter multiplies non-zero values. Instead, the multipliers can be shared between the in-phase and quadrature filters by adding an enable feature to the registers for each filter. The multipliers will multiply every Mixed I/Q value from Fig. 3(c) by the coefficients and use CNT0 to output the products to the appropriate filter. When CNT0 is '0', the multiplier outputs will go to the in-phase filter, and when CNT0 is '1', the multiplier outputs will go to the quadrature filter. Sharing the multipliers halves the required multipliers.

The number of adders can also be reduced because in any one given clock cycle, only half of the coefficients (even or odd-numbered coefficients) are necessary for additions, and during the next clock cycle, the other half will be used. The reduced hardware equivalent of the transposed form FIR filter is shown in Fig. 7. The registers with enable pins prevent values that belong in the opposite filter from being processed. The enable pins and select lines are assigned as previously mentioned in the direct form reduction section.

The transposed form FIR reduction is significant independently of the direct form because it allows for other multiplier reductions such as multiplier block implementation and optimization, discussed in [6].

C. *Maintainability of Existing Reductions*

The removal of common filter blocks in [2] is still possible because the reduced hardware equivalents introduced in this paper do not alter the performances of the filters. The outputs are exactly the same as the full, standard forms. The common filter blocks that [2] removes will simply contain less hardware. Combining the introduced technique with the reduction in [2] will yield an even more hardware-efficient design.

Using the reduced hardware equivalent of the transposed form FIR filter, a multiplier block may still be implemented and optimized by the subexpression elimination algorithm in [6]. Again, the combination of the introduced technique and the reduction in [6] will yield a design with even less required hardware. In this case, an entirely multiplierless architecture for the DDC is possible.

D. *Mixer Reduction Revisited*

Since the reduced filters process zeros without explicitly receiving them as inputs, the mixer no longer needs to output zeros. Instead, the mixer only needs to output ±X(t). Finally, the mixer is a two input multiplexer with positive and negative X(t) for inputs. CNT1 will determine the sign of X(t) for the mixer output, and CNT0 will control the flow of data in each filter to achieve the same overall filter function without the extra hardware.

IV. Extending to Other DDC Applications

The reductions in these proposed designs are based on the general equations (4) and (5). For other DDC applications to use this reduction, the sample rate must be $4f_{IF}$. For filters with an odd number of coefficients, an extra adder and register are

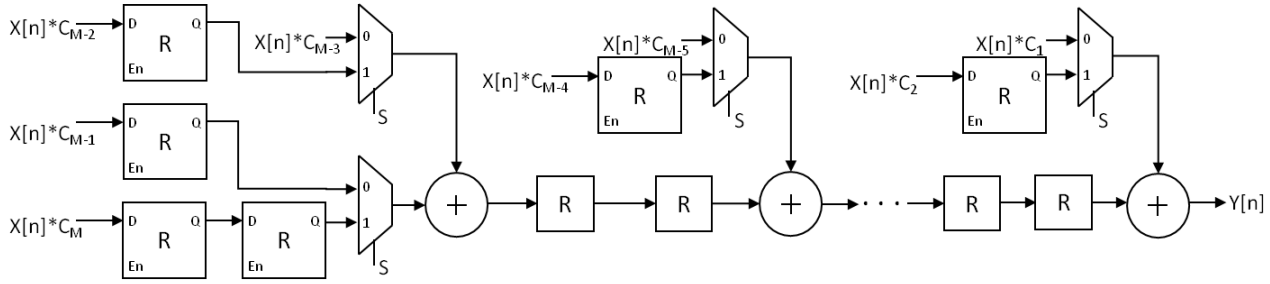

Fig. 7. Reduced transposed FIR filter with M taps.

necessary, but the adder may be shared between I and Q. However, sharing the additional adder requires control logic similar to the rest of the reduced filters. Changing the precision of coefficients, adders, or multipliers can be done independently of the structures depicted in the block diagrams.

Advantageously, scaling the reduced filters is very simple. The collection of blocks in Fig. 5 and Fig. 7 after the ellipses may be replicated as many times as is necessary to achieve the correct filter length (within one tap in the case of odd number of coefficients/taps).

V. Experiment Results

Table II shows the required hardware blocks for the full direct, reduced direct, full transposed, and reduced transposed form FIR filters using 64 taps. As can be seen, the hardware reductions are very significant. Most importantly, the number of multipliers and adders has been significantly reduced.

TABLE II. Reduced FIR Filter Hardware Comparisons

Filter	Hardware Required for 64 Taps			
	Multiplier	***Adder***	***Register***	***Multiplexer***
Full Direct Form	128	126	126	0
Proposed Reduced Direct Form	64	62	64	128
Full Transposed Form	128	126	126	0
Proposed Reduced Transposed Form	64	62	188	64

Input to the DDC for the case included in this report is a sine wave operating at 34 MHz, outputting values eight bits wide to simulate an eight bit ADC output. The input frequency bandwidth is 24 MHz and centered at 26 MHz. The hardware reductions were tested and verified using the aforementioned 64 tap filters. The filter coefficients were determined by applying a hamming window to the ideal coefficients of a low pass filter which were then truncated to twelve bits. The output signal frequency is determined by (6).

$$f_{out} = f_{in} - f_{LO} = 34 \text{ MHz} - 26 \text{ MHz} = 8 \text{ MHz} \qquad (6)$$

All four filters performed identically, demonstrating that the reduced architectures are truly equivalent to the full, standard form architectures. Equation (6) also shows a calculated expected output at 8 MHz which is confirmed in Fig. 8 which shows the FFT result of the DDC in-phase output signal after the proposed reduced direct form filter.

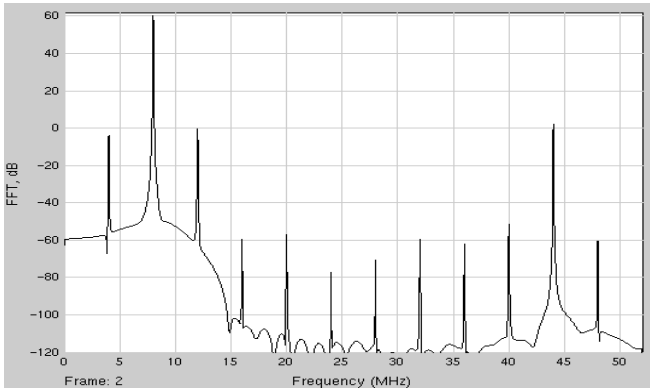

Fig. 8. FFT of in-phase filtered DDC output for f_{in} = 34 MHz.

The worst case, when f_{in} results in f_{out} at the edge of the passband, is shown in Fig. 9. The input for the worst case test was 38 MHz, resulting in an output frequency of 12 MHz, the edge of the 24 MHz bandwidth. Different input frequency signals were tested and verified; the DDC works as expected.

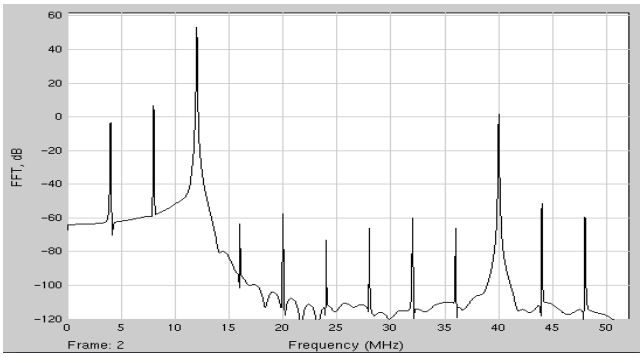

Fig. 9. FFT of in-phase filtered DDC output for worst case, f_{in}=38 MHz and f_{out}=12 MHz.

VI. Conclusion

In this paper, a simplified mixer architecture sampling at four times the intermediate frequency is analyzed to reduce hardware. Reduced architectures for direct and transposed form FIR filters are presented and compared to the full hardware equivalent. The architecture is implemented, tested, and verified using Xilinx Sysgen.

Unlike many other hardware reductions, this hardware reduction may be combined with several other techniques. Using the resulting DDC architecture yields a very hardware efficient design and presents an opportunity for further research in extremely hardware-efficient DDC designs.

References

[1] C. Ruan, J. Hua, Z. Zheng, Y. Wu, and L. Meng, "A Study of Different Matched Filter In Digital Down Converter," *2012 Int. Conf. Systems and Informatics*, Yantai, China, 2012, pp. 2059-2063.

[2] M. Kim and S. Lee, "Design of Dual-Mode Digital Down Converter for WCDMA and cdma2000," *ETRI Journal*, vol. 26, no. 6, pp. 555-559, Dec. 2004.

[3] Saiyu Ren; Billman, S.; Siferd, R., "Multiplier-less Digital Down Converter in 90nm CMOS technology," *Aerospace Electronics Conference (NAECON), Proc. 2011 IEEE Nat.*, pp.316-319, July 2011.

[4] E. S. Malki, K. A. Shehata, and A. H. Madian, "Design of Triple-Mode Digital Down Converter for WCDMA, CDMA2000 and GSM of Software Defined Radio," *2009 Int. Conf. Microelectronics*, Marrakech, Morocco, 2009, pp. 272-275.

[5] S. Jou, S. Wu, C. Wang, "Low-Power Multirate Architecture for IF Digital Frequency Down Converter," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 45, pp. 1487-1494, Nov. 1998.

[6] S. Mirzaei, A. Hosangadi, and R. Kastner, "FPGA Implementation of High Speed FIR Filters Using Add and Shift Method," *Int. Conf. Computer Design*, San Jose, California, 2006, pp. 308-313.