EFFICIENT ULTRASOUND ECHO PROCESSING ALGORITHMS FOR IMPLENTATION WITH TMS320C64x DSPs

Hak-Yeol Sohn, Hyun-Chul Kim, Dong-Hoon Han, Tai-Kyong Song

Center for Medical Solutions Reserch/Electronic Eng., Sogang Univ., Seoul, Korea

tksong@sogang.ac.kr

Abstract: DSP (digital signal processor)-based ultrasound system has the advantage of easy reconfiguration and use of previous design. These offer reduced development cost and time. However since the ultrasound imaging system requires high data rate and huge amount of computations compared to the commercial DSP, the implementation of ultrasound imaging system based on a DSP is challenging. In this paper, we implemented the echo processor among many functional blocks in ultrasound imaging system, and proposed the new algorithm optimized to the DSP architecture in order to enhance the system performance.

Introduction

Conventional ultrasound imaging system is divided into front-end and back-end segments. The front-end segment is composed of an analog transmitter, an analog receiver and a beamformer. The back-end segment is composed of an echo processor and a DSC (digital scan converter). Especially, the back-end segment is suitable for implementation with a DSP because it contains many one dimensional signal processing blocks. The new algorithm can be easily applied to these blocks.

However, since the data rate and computational power of back-end segment is higher than those of the commercial DSP. We must design an efficient DSP system using many DSPs for realtime processing of signal processing tasks in back-end segment. This would result in a large increase in the development time and the overall system cost. In this paper, we proposed new algorithms to optimally implement the echo processor using minimum number of DSPs.

In the next section, each functional block of the echo processor is programmed with TMS320C64x to evaluate its processing time. The echo processor shown in figure 1 can be divided into two sections: the functional blocks of the first half of the echo processor contains DC cancel FIR filter, TGC (time gain correction), quadrature demodulator and decimation filter and the second half of the echo processor contains magnitude calculator, log compressor, zone blending, edge enhancement FIR filter, nonlinear filter and lateral FIR filter.

First, we will propose efficient methods to reduce the computation time of most time-consuming blocks of eaccho processor sections. Secondly, the performance of the software echo processor using the proposed methods will be compared with the that of the conventional echo processor. Finally, the poposed method is verified by comparing tha images produced by both methods..

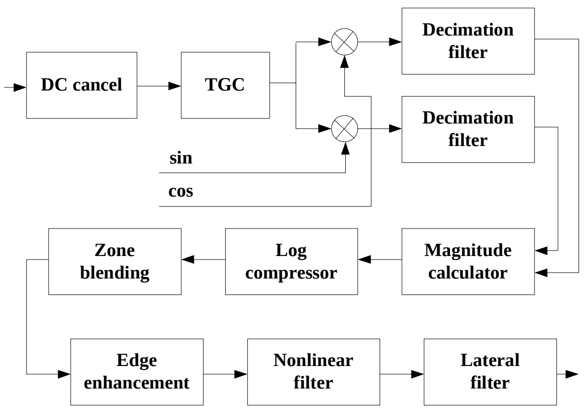


Figure 1: Conventional echo processor

DSP implementation of echo processor

Each functional block in the echo processor was programmed in assembly or C language and various algorithms were searched to program each functional block efficiently. Especially, decimation filter required the largest processing time and hence was efficiently programmed using linear assembler for C64x. DC cancel filter was also programmed in assembly language.Other assembly coding? All other functional blocks were programmed in C language. The results are summarized in table 1, which shows the number of cycles required for each functional block to process one scan line data consisting of 7134 rf samples . The input rate of the rf samples to the echo processor block is 40MHz throughout this paper.

Table 1: Processing time of each functional block in the first half of the echo processor

Function	Cycles/scanline
DC cancel filter	121,349
TGC	66,598
Quadrature demodulator	12,296
Decimation filter	334,328
Total cycles	536,571

Table 2 shows the number of cycles required for each functional blocks of 2nd half of echo processor to process 1024 inphase and quadrature data pairs per scanline from the decimation block. The magnitude calculator and log compression blocks contain the transcendental functions such as squared root and log functions, respectively. Such functions take too much time to calculate directly using MAC (Multiply and Accumulation) operations. Instead, we investigated other methods for programming square root function, including Newton Raphson algorithm. Among them, CORDIC rotator algorithm, requiring only simple bit shift and addition/subtraction, was used to calculate the magnitude and the result is shown in table 2. The log compression block was implemented using LUT (look-up table) reference method instead of typical Taylor series expansion method.

From the data given in tables 1 and 2, total number of cycles to process all the functional blocks of the echo processor was estimated to be 591,499. Since internal clock period of TMS320C64x DSP is 1ns, the echo processor can be programmed to support PRF up to 3.4KHz using two DSPs.

Table 2: Processing time of each functional block of the second of the conventional echo processor

Function	Cycles/scanline
Magnitude calculator	17,934
Log compressor	8,441
Zone blending	571
Edge enhancement filter	11,310
Nonlinear filter	13,316
Lateral filter	3,356
Total cycles	54,928

Table 1 shows that decimation block takes up more than 60% of the total processing time of the 1st half of the echo processor. In the 2nd half, magnitude calculator and log compressor blocks require about 50% of the computation time. To reduce the pocessing time, we devised efficient algorithms for those blocks, which can be programmed to run much faster on the target DSP.

Figure 2 show the conventional decimation filter architecture, in which upsampling is performed prior to decimation filter to obtain finer decimation ratio. Conventionally, the length of the decimation filter in figure 2 increases in proportion of M. That is, it is given by $K \times M$. This implies that when K=16, 368 filter taps are required for M=23. Therefore, direct implementation of the decimation filter requires such a large instruction cycles as shown in table 1.

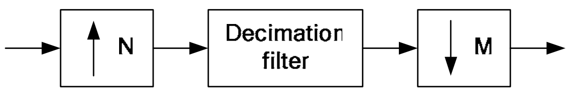


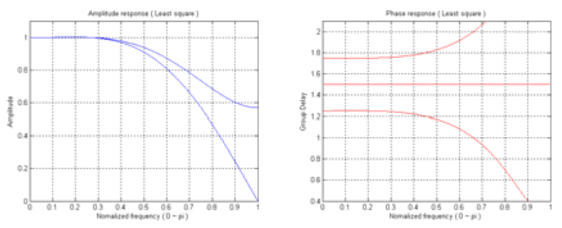
Figure 2: Conventional decimation filter

If the decimation filter is implemented with the polyphase structure, its length can be reduced to K regardless of M. However, polyphase implementation of the decimation block increases the overhead such as in data manipulation and program flow control. Such overheads tend to reduce the efficiency of software pipelining and parallel execution of multiple instructions, thereby increasing the overall processing time.

Table 3: Illustration of input output relations for various values of M

M	y(1)	y(2)	y(3)	y(4)	y(5)
5	x(1)	x(2.25)	x(3.5)	x(4.75)	x(6)
6	x(1)	x(2.5)	x(4)	x(5.5)	x(7)
7	x(1)	x(2.75)	x(4.5)	x(6.25)	x(8)
8	x(1)	x(3)	x(5)	x(7)	x(9)
9	x(1)	x(3.25)	x(6.5)	x(7.75)	x(10)

To reduce the computational complexity of the decimation filter, we used a fractional delay (FD) filter-based decimation filter. With only a few taps, fractional delay filter can provide samples positioned in fine intervals between actual samples. When K=4, the decimation block produces output samples as shown in table 3 for M values. In this case, the FD filter should provide delays of 0.25, 0.5, and 0.75. Figure 3 shows the characteristics of the 4-tap fractional delay filters used in this paper.



(a) (b)

Figure 3: Property of a 4-tap FD filter: (a) amplitude response and (b) Phase response

Since the input signals of the decimation are outputs of IQ demodulation mixers, they contain harmonic components. In the direct structure (figure 2), the decimation filter is designed to remove the harmonic components completely. FD filters, however, do not have sharp magnitude response as shown in figure 3. Therefore, the decimation filter block proposed in this paper consists of two decimation blocks separated by a conventional LPF as shown in figure 4.

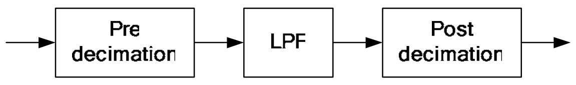


Figure 4: The proposed decimation filter

The pre-decimation is performed by using a 2-tab FD filter in conjunction with the following LPF. As illustrated in figure 5(a), the input signals from the quadrature mixers have down-converted base-band component and harmonic components, centered at 2f0, denoted by bold line. The 2-tap FD filter has a role to apply delays to the base band component of figure 5(a), which are required to down samplie the signals so that the pre decimated signals have a spectrum as shown in figure 5(b). Since only the base band components require accurate pre-decimation delays, 2-tap FD filter is enough to serve as pre-dicimation filter. Now, the purpose of LPF is to remove the harmonic components shown in figure 5(b). To do this, we used a FIR LPF with 25 taps which has a magnitude response represented by dotted line in figure 5(c). Since the data rate was reduced by the pre-decimation block, this LPF has a low computation complexity. Finally, post decimation filter is used to generate 1024 delayed samples as described in table 3 as examples at the aimed data rate.

The proposed algorithm was programmed preliminary, with no optimization efforts, in C language to find that the clock cycles to process the decimation block was reduced greatly from 334,328 to 63,017.

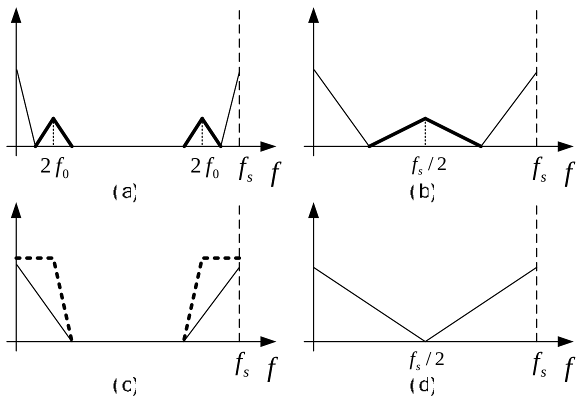


Figure 5: Frequency spectrums of (a) input signal, (b) pre-dicimation output, (c) LPF output, and (d) post-decimation output of the proposed decimation filter block.

To reduce the toal number of clock cycles in table 2, magnitude calculator and log compressor are combined as

$$y = \log(\sqrt{i^2+q^2}) = \frac{1}{2}\log(i^2+q^2) \qquad (1)$$

and LUT is used to compute the logarithm. A poblem with this approach is that i^2+q^2 is 31-bit wide, requiring a LUT of 4Gbytes. Such a LUT is too big and should be implemented in external memories. Furtheremore, LUT requires random memory access, which is very slow compared to the direct internal memory access. Consequently, direct LUT method is not suitable for our purpose.

To implement the LUT approach more efficiently, we modified equation (1) as

$$\frac{1}{2}\log[i^2\{1+(q/i)^2\}] = \log(i) + \frac{1}{2}\log\{1+(q/i)^2\} \qquad (2)$$

and used the following approximation:

$$\frac{1}{i} = \frac{d\ln(i)}{di} \approx \frac{\ln(i)-\ln(i-1)}{i-(i-1)} = \ln(i)-\ln(i-1) \qquad (3)$$

Equation (2) can be calculated by using one LUT. The logarithm of *i* and $1+(q/i)^2$ can be computed using the same LUT, which is only 64Kbytes, if $i \geq q$. One can exchange *i* with q in equation (2) when $i < q$. A LUT of this size can be located in the 1MB internal memory of TMS320C64x.

Since division operation takes tens of cycles in TMS320C64x, however, calculation of $1+(q/i)^2$ takes more cycles than magnitude calculation. The approximation in equation (3) eliminates the division operation. As a result, the magnitude calculation and log compression blocks can be computed as

$$y = \log(i) + \frac{1}{2}\log[2^{14} + \{q(\log(i)-\log(i-1)) \times \frac{18864}{2^6}\}^2] - 2\log 7 \qquad (4)$$

The proposed algorithm was programmed with C language and tested on the TMS320C64x DSP. The results showed that the magnitude calculator and log compressor can be computed in only 11,602 cycles as opposed to 26,375 cycles in table 2 for a scanline consisting of 1024 samples.

Equation (3) produces approximation error. However, in most cases, the error was negligible.

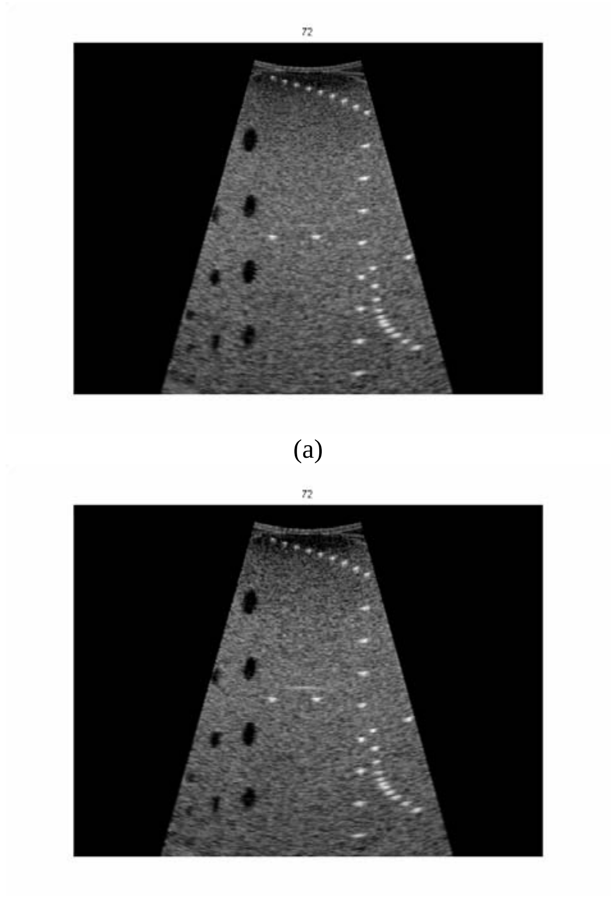
Results

The software echo processor using the proposed algorithms were run on a DSP board using two TMS320C64x processors to test its performance in processing speed and computation error.

As was fore-mentioned, with the proposed algorithm, the processing time for decimation filter block was reduced from 334,328 to 63,017 clock cycles. The magnitude calculation and log compression blocks were also improved in the number of clock cycles from 26,375 to 11,602.

As a result, the highest PRF of the proposed echo processor increases from 3.4KHz to 6.54KHz when two TMS320C64x DSPs are used. This implies that in many cases where high frame rate greater than 20 is not required, the entire echo processor can be implemented with a single DSP.

Finally, figures 6(a) and (b) show images of the echo propcessor output when the conventional method and the proposed method are used, respectively. The two images look almost identical inspite of the errors caused by the approximation in equation (3). The difference between two images were 31.2307dB in PSNR.



(b)

Figure 6: Comparison of ultrasound images obtained with (a) the conventional echo processor and (b) the proposed echo processor

Conclusion

In this paper, we proposed an efficient cho processing algorithms for fast implementation with a commercial DSP TMS320C64x by TI. The most time consuming functions in the two sections of echo processor were computed with newly developed algorithms that can dramatically improve the processing time. The priminary experimental results show that the entire echo processor block can be implemented in software using two DSPs and supports up to PRF as high as 6.54kHz. Since this result was obtained without optimization efforts, the processing time can be further reduced if programmed using linear assembler and other techniques to best utilize the DSP architecture.

Reference

[1] LAAKSO T.I.,VALIMAKI V., (1996) :'Splitting the unit delay[FIR/all pass filters design]', Signal Processing Magagine. IEEE, vol. 13, pp. 30-60, Jan, 1996.
[2] TEXAS INSTRUMENT, INC. (2005): 'TMS320C6000 CPU and Instruction Set Reference Guide'
[3] TEXAS INSTRUMENT, INC. (2005): 'TMS320C6000 Programmers Guide'
[4] H.C KIM, J.H SIM, (2004): 'An optimized software-based echo processing algorithm for small scale ultrasound systems', Ultrasonics symposium, IEEE, vol. 3, p. 2053-2056