Application Report
SPRAB11–November 2008

Efficient Implementation of Ultrasound Color Doppler Algorithms on Texas Instruments' C64x™ Platforms

Udayan Dasgupta

ABSTRACT

DSP chips are gaining importance in ultrasound applications as the need for portability and low power grows. One of the more computationally demanding applications for ultrasound involves estimating blood flow characteristics using Doppler techniques. The color Doppler ultrasound mode is used to diagnose many conditions like blood clots, valve defects, and blocked arteries. This application report looks at mapping typical color Doppler algorithms onto Texas Instruments' high performance C64x+™ core. The algorithms include RF demodulation, wall filtering and flow power, velocity, and turbulence estimation. This document starts with a general technique for analyzing algorithm complexity on the C64x+ architecture, then applies this technique to Doppler processing algorithms, explains their mapping to the C64x architecture, and finally compares these estimates to actual implementations. Based on these implementations, it will be shown that these algorithms can run on TI's C64x-based DSPs using a fraction of the available processing power.

Contents

1 Background 2
2 Color Doppler Mode 2
3 Method for Estimating Complexity 3
4 RF Demodulator 3
5 Wall Filter 7
6 Flow Power Estimator 9
7 Flow Parameter Estimator 11
8 Conclusions 12
9 References 13

List of Figures

1 Receive Path of Typical Ultrasound System in Color Doppler Mode 2
2 DSP Architecture of TI C64+ Platform 3
3 Mapping RF Demodulation (mixing) to TI C64+ Architecture 5
4 Mapping RF Demodulation (decimation) to TI C64+ Architecture 6
5 Mapping Wall Filtering to TI C64+ Architecture 8
6 Mapping Flow Power Estimation to TI C64+ Architecture 10
7 Mapping Correlation and Power Computations to TI C64+ Architecture 12

List of Tables

1 Complexity Analysis for Mixing Operation 5
2 Complexity Analysis for Decimation Operation 7
3 Complexity Analysis for Wall Filtering Operation 9
4 Complexity Analysis for Flow Power Operation 10
5 Complexity Analysis for Correlation and Power Calculations 12

1 Background

DSP chips are gaining importance in medical imaging applications as the push for portability and low power grows. Ultrasound is one such imaging modality where DSP use has been instrumental in driving down system costs and fueling innovative applications. The most common operating modes of ultrasound systems are:

- B (Brightness) mode: Transmits sound waves of a particular frequency into the body of interest and records the received echoes as a function of time and position. The brightness of each point on the display represents the amplitude of the sampled signal.
- Color Doppler mode: Similar to B-mode, but transmits multiple pulses (ensemble) and uses the relative time between received echoes to estimate blood flow characteristics. The most common flow parameters that are estimated include flow power, direction, velocity and turbulence.

This application report starts off with a general introduction of the color Doppler mode and briefly introduces the algorithms in Section 2. Section 3 explains a general method of estimating algorithm complexity on TI's high performance C64x+ DSP [1] core. Section 4 to Section 7 provides details about each of the algorithms including algorithmic and implementation details, complexity estimates, and results. Finally, Section 8 presents the conclusions from this work.

2 Color Doppler Mode

The receive path of a typical ultrasound system is shown in Figure 1. This application report focuses on algorithms used in the color Doppler mode, which are highlighted in the figure. After passing through the receiver front end and beamformer, the data is processed by the B-mode imaging blocks and the color Doppler imaging blocks. The echo outputs from the B-mode blocks and a combination of scan-converted versions of velocity, turbulence, and power estimates from the color Doppler imaging block are merged in the tissue flow decision block, before going through final image processing and display.

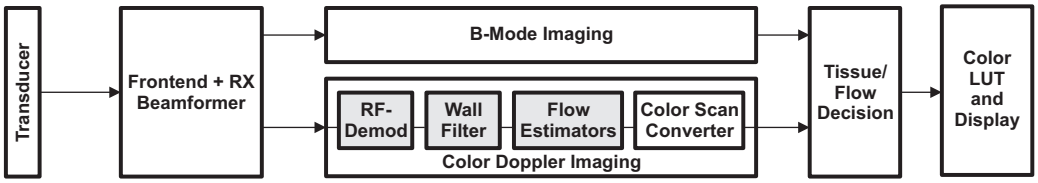

Figure 1. Receive Path of Typical Ultrasound System in Color Doppler Mode

This application report looks at efficient implementation of the following typical color Doppler algorithms.

- RF demodulation consisting of mixing, filtering, and decimation of echo data
- Wall filtering using a matrix initialized form of IIR filters [5] of demodulated data
- Color flow estimator [6] that estimates velocity, turbulence, and power together
- Flow power estimator that estimates the power of multiple sets (ensembles) of points. Note that this is also one of the flow parameters that are estimated in algorithm-3, but a separate implementation is considered since this parameter may be desired in other contexts as well. For example, power estimates before and after wall filtering can provide an estimate of the clutter power.

3 Method for Estimating Complexity

This section outlines a simple *best-case* complexity analysis for these routines in order to establish a lower bound for the complexity (DSP cycles) on the C64x+ core, as shown in Figure 2. Although all the work is with respect to C64x+ cores, the general methodology carries over to other very long instruction word (VLIW) architectures as well.

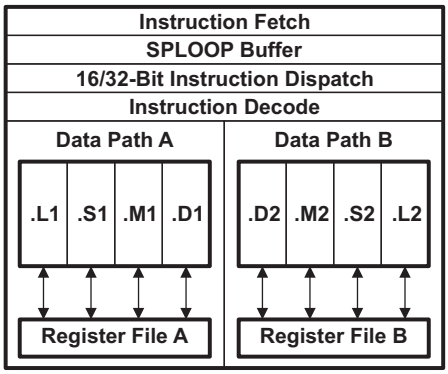

Figure 2. DSP Architecture of TI C64+ Platform

During every clock cycle in the TI C64x+ platform, the instruction fetch, dispatch, and decode units deliver instructions to the eight functional units that reside in data paths A and B, as shown in Figure 2. Each data path contains four functional units (L, S, M, and D) and 32 general-purpose registers. For more details about the architecture, instructions, and mapping of instructions to functional units, see the *TMS320C64x/C64x+ DSP CPU and Instruction Set Reference Guide* (SPRU732) [1]. Unless otherwise noted, every unit produces a result at each clock cycle when pipelined perfectly.

This approach estimates the complexity of the loop kernels that dominate performance and are expected to take the maximum cycles of these algorithms. It is *best-case* in that it does not attempt to estimate the overheads and is a first order approximation of the complexity. Overheads include epilog and prolog of loop kernels, outer loop operations in nested-loop implementations, etc. To achieve these estimates, the implementation may need to change the algorithm data path, without affecting performance, to get the lowest complexity on the C64x+ core. To arrive at these estimates, the most efficient data flow and CPU instructions are estimated for each of the algorithmic operations. Then, the various units where these instructions could be mapped to are considered and, finally, the cycles on the unit that are most loaded are used as the estimate of the algorithm's complexity. Note that perfect pipelining performance is assumed. More details on the optimization process can be found in [2], [3], [4].

In the next four sections, complexity estimates obtained using this approach are compared to implementation results for the typical color Doppler algorithms mentioned earlier. Also the fixed point implementations will be compared to their floating point counterparts in terms of normalized mean square error (NMSE) to ensure that performance remains acceptable. The NMSE is computed by normalizing the mean square error with the average of the floating point reference output.

4 RF Demodulator

4.1 Algorithm Description

In color-flow mode, the input to this module is multiple ensembles of scan lines. Assume that there are *B* scan line sets and each of the scan line sets consists of *N* scan lines. Therefore, the ensemble size is *N*. Each scan line consists of *T* RF data samples. In this block, each scan line is processed independently by first *mixing* it with sinusoids to produce in-phase (I) and quadrature (Q) components, followed by low pass filtering (LPF) using a finite impulse response (FIR) filter to prevent aliasing, and finally decimating the filtered output by a factor, *S*. Assume that the FIR filter uses (*L*) taps. Each output scan-line from this function consists of *D* (=*T*/*S*) decimated points.

The operations in this module are summarized by the following equations. For a given transducer center frequency (f_c) and front-end sampling frequency f_s, the beamformed RF data for each scan line, R_T, is first mixed to create in-phase and quadrature components of the mixed output vector, MI_T.

$$Re\{m_t\} = r_t \cdot cos\left(\frac{2\pi f_C t}{f_S}\right), \qquad 0 \le t < T-1, \tag{1}$$

$$Im\{m_t\} = r_t \cdot sin\left(\frac{2\pi f_C t}{f_S}\right), \qquad 0 \le t < T-1, \tag{2}$$

Note that throughout this application report, lower-case letters are used as indexes into quantities whose maximum values are indicated by their upper-case counterparts, e.g., *t* is an index into the *T* RF points of each scan line. Also upper case letters denote the matrices and vectors *(R, M)*, while their lower case counterparts (*r* ,*m*) denote the individual elements in these matrices/vectors. The subscripts used with matrices denote their dimensions; whereas, subscripts used with elements denote their positions in the matrix/vector, i.e., r_t denotes the t^{th} element in the *R* vector.

These mixed outputs are then passed through a filter, F, and down-sampled in a single step, to generate the decimated output vector E_D.

$$Re\{e_d\} = \sum_{l=0}^{L} Re\{m_{t-l}\}, \qquad 0 \le d \le D-1, t = Sd, \tag{3}$$

$$Im\{e_d\} = \sum_{l=0}^{L} Im\{m_{t-l}\}, \qquad 0 \le d \le D-1, t = Sd \tag{4}$$

For implementation purposes, it has been assumed that the input *R* consists of 32-bit signed integers, while *M* and *E* consist of 16-bit signed integers. The sine, cosine values, and filter coefficients are assumed to be 16-bit signed integers.

4.2 Mapping to TI C64+ Core

Two main kernels have been studied using the methodology outlined in Section 3: mixing and decimation. It can be shown that the mixing operation can be efficiently carried out using reasonably small sine and cosine tables (< 4 KB), instead of computing these values on the fly. The data flow and intrinsics used for the mixing are summarized in Figure 3. Note that sine and cosine tables are interlaced and that table indexing has been omitted from the figure for simplicity.

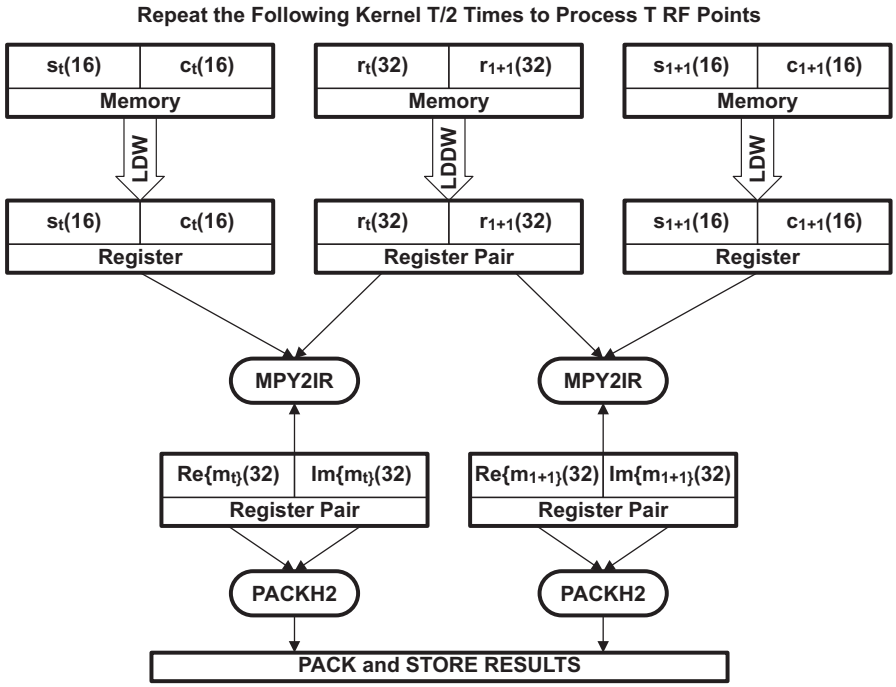

Figure 3. Mapping RF Demodulation (mixing) to TI C64+ Architecture

Mapping of these instructions to the CPU units and calculating the cycle-usage yields the results in Table 1. It can be seen that the D units would take the maximum number of cycles for this algorithm. Therefore, this kernel can be considered to be *load-limited* and would need approximately *T* cycles for processing *T* RF points, achieving a best-case performance of 1 pipelined kernel cycle-per-output point. Note that during the implementation phase, additional tricks like further loop unrolling may be needed to improve pipelined performance and achieve the ideal MIPs benchmark.

Table 1. Complexity Analysis for Mixing Operation

Operation	Cycles	C64x+ Instruction	CPU Units
Loads – Input	(T/2)/2	LDDW	D1/D2
Loads – Sine/Cosine	T/2	LDW	D1/D2
Multiplies	(2T/2)/2	MPY2IR	M1/M2
Adds (Table indexing)	2T/4	ADD	L1/L2
Shift	T/2	SHR	S1/S2
Modulus (Table indexing)	T/4	AND	L,S
Stores	(2T/4)/2	STDW	D1/D2

The mixing kernel is typically followed by the decimation kernel, which does the anti-aliasing filtering apart from reducing the sampling rate by a factor of *S*. An efficient way to implement this is the well known polyphase implementation [7], which is conceptually similar to clocking the mixed input samples into the two filters, one for the real part and one for the imaginary parts, every cycle of the up-sampled clock, but clocking out a complex output only every S^{th} clock cycle every clock in the down-sampled domain. This block takes *T* RF input samples and outputs *D (=T/S)* decimated samples. The data flow and intrinsics used for the decimation kernel are summarized in Figure 4. Note that since the filtering operation occurs in the down-sampled domain, it runs for only *D*, not *T*, points.

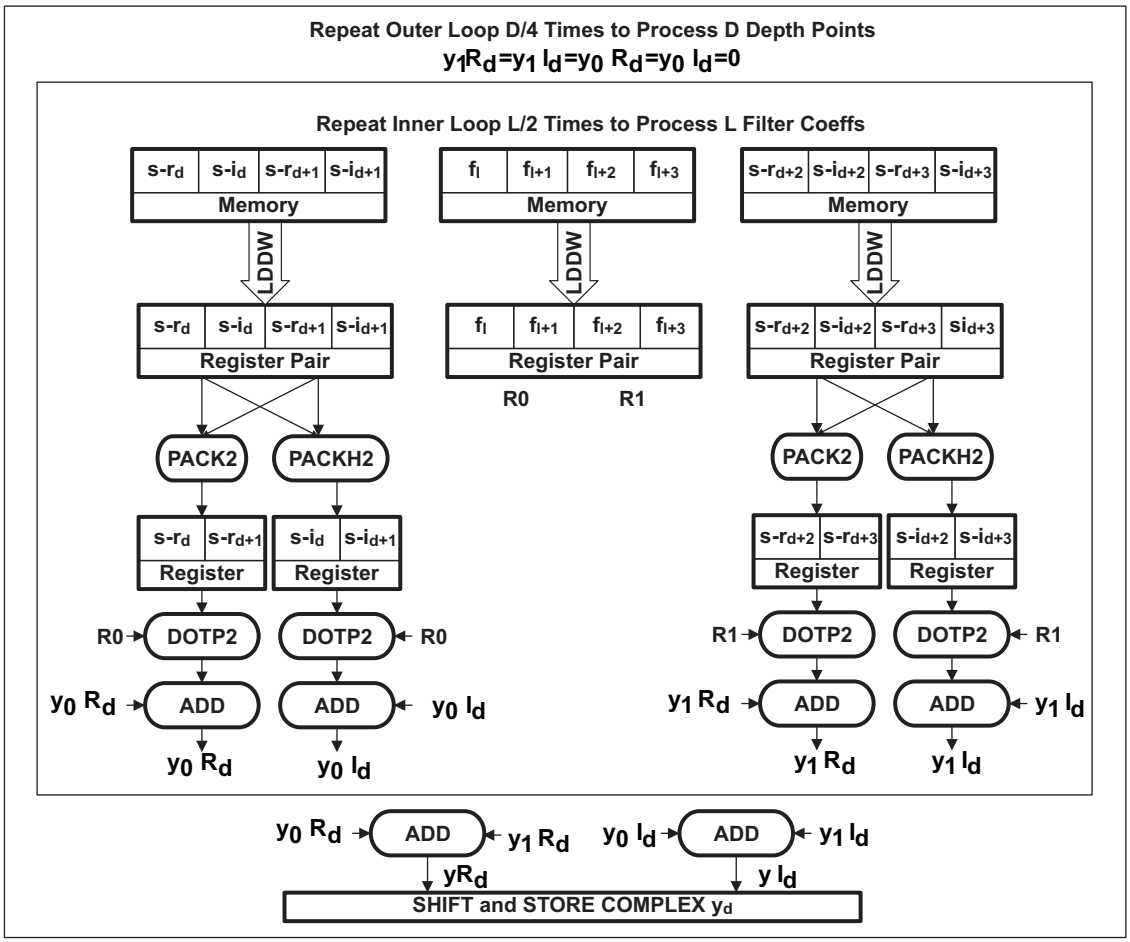

Figure 4. Mapping RF Demodulation (decimation) to TI C64+ Architecture

Table 2 maps the main instructions onto the processor units and shows that the filtering operation would be multiplier-limited since the M units would need *DL/2* cycles to complete its operations while the D, L, and S units would need lesser cycles to complete theirs.

The above analysis ignores the overhead for circular indexing and filter state updates. It is important to note that by allowing this function to accept and return the starting and ending states of the filter, the RF demodulation can be done on portions of scan lines. This feature becomes especially important when working with systems with limited amounts of local memory.

Table 2. Complexity Analysis for Decimation Operation

Operation	Cycles	C64x+ Instruction	CPU Units
Loads – Input	(T/2)/2	LDDW	D1/D2
Loads – Filter	(L/4)/2	LDDW	D1/D2
Multiplies	D*L/2	DOTP2	M1/M2
Adds	D*(L/2-1)	ADD	L1/L2
Stores	(2D/4)/2	STDW	D1/D2

Test code using C and intrinsics was used to achieve pipelined kernel performance of *L/2+S* cycles-per-output point, while maintaining NMSE performance within 1e-6 of the floating-point output.

5 Wall Filter

5.1 Algorithm Description

This module does an IIR filtering operation. However, since it needs to filter very short sequences of ensemble size *N*, its transient performance is of primary importance. To gain greater control over its transient performance, a state-space formulation of the IIR filter [5] is used. This filter can be used with different types of initialization, the most common forms being: zero, step, and projection initialization schemes. The basic operation done for filtering for each scan-line set is a multiplication of complex input data matrix, *X*, with real coefficient matrix, *W*, given by,

$$Y_{DxN} = X_{DxN} \cdot W_{NxN}, \tag{5}$$

Both of these input and output matrices consist of *D* rows corresponding to the *D* depth points and *N* columns, corresponding to the *N* ensembles. Note that the matrix *X*, is created from the *N* decimated scan lines (*E*), which forms the ensemble.

As this module works with the decimated output from the previous module, its input and output consist of packed 16-bit real and 16-bit imaginary data. Also the wall filter coefficients are 16-bit integers.

5.2 Mapping to TI C64+ Core

To load filter coefficients efficiently, it is assumed that the filter coefficient matrix is stored column-wise in memory. With this assumption, the coefficients on the same column of *W* would lie on adjacent memory addresses and could be loaded using wide-load instructions for multiplying the rows of the input matrix. Also, it is assumed that the input and output data are arranged ensemble-by-ensemble, (i.e., all the *N* input/output points at a given depth point, d, lie adjacent to each other in memory, before the *N* input/output points of the next depth point).

As shown in the algorithm description, the filtering operation basically consists of a multiplication of a complex matrix with a real matrix. The data flow and intrinsics used for this kernel are summarized in Figure 5. Note that multiple such kernels could be run in parallel to use efficient shifting, packing, and storage instructions in the last stage.

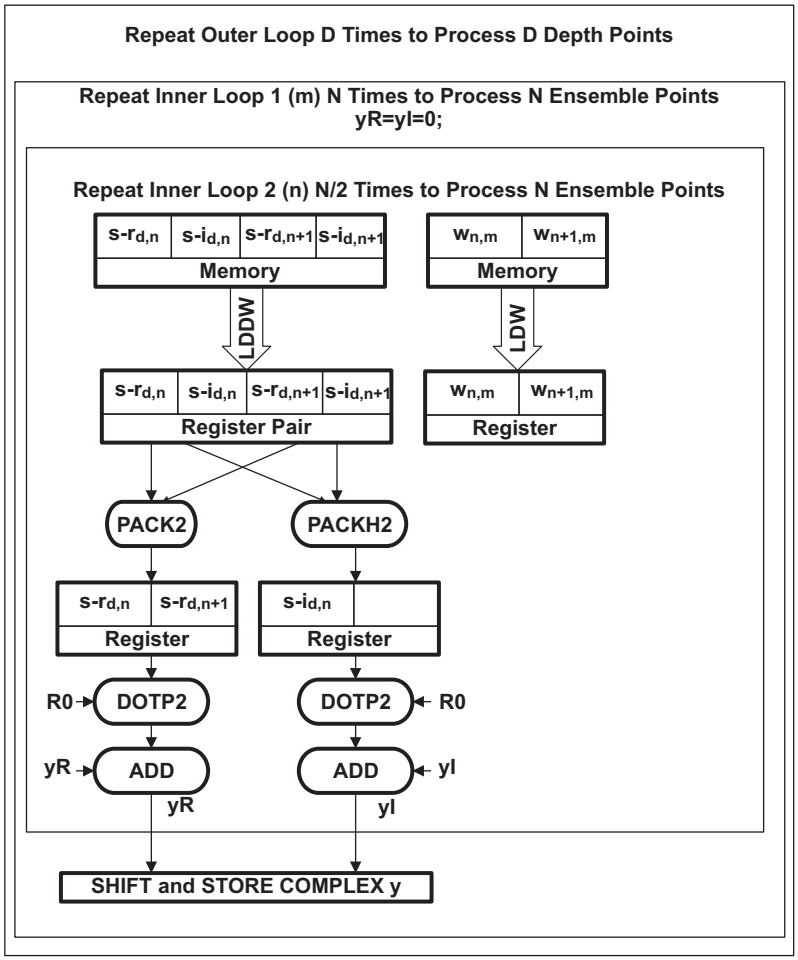

Figure 5. Mapping Wall Filtering to TI C64+ Architecture

Table 3 is obtained by mapping the main instructions onto CPU units. From this analysis, the two M units would be used the most because they would be used for the two sets (real and imaginary) of 16x16 multiplies. To generate each of the *DN* complex output points, *2N* real multiplies are needed for each point, taking $DN^2/2$ cycles. The *N/2* complex points to be added for each output point would also need *N/2* cycles assuming 40-bit additions on L units. Therefore, this algorithm should need $DN^2/2$ cycles for processing *D* decimated points translating to a pipelined performance of *N/2* cycles per output complex point.

Test code using C and intrinsics was used to achieve pipelined kernel performance of *N/2* cycles-per-output point while maintaining performance, which is within a NMSE target of 1e-6 of the floating-point output. Note that this module consists of three nested loops: one running over D and the other two running over N. Such nested loops need to be implemented carefully since only the innermost loop gets pipelined and the overheads of the inner loops get multiplied as many times as the outer loop is run. For such algorithms, the complexity estimate may not be as close as it can be with single loop kernels.

Table 3. Complexity Analysis for Wall Filtering Operation

Operation	Cycles	C64x+ Instruction	CPU Units
Loads – Input	DN/4	LDDW	D1/D2
Loads – Coefficients	$N^2/2$	LDDW	D1/D2
Multiplies	$DN^2/2$	DOTP2	M1/M2
Adds	$DN^2/2$	ADD	L1/L2
Stores	DN/4	STDW	D1/D2

6 Flow Power Estimator

6.1 Algorithm Description

This module is used to compute the power of an ensemble of points. Typically, this function can be used to calculate the input and output power of the wall filter to compute the signal-clutter ratio or it could be used for detecting the presence of blood in color Doppler processing. For a complex input matrix $(X_{DxN} = I_{DxN} + j\ Q_{DxN})$ where all the matrices are of size DxN, the output vector $P_D = \{ p_d \}$ is computed as,

$$p_d = \sum_{n=0}^{N-1} i_{d,n}^2 + q_{d,n}^2, \ldots\ldots\ldots 0 \le d \le (D-1), \tag{6}$$

The input data to this module consist of packed 16-bit real and imaginary data and the output data are 16-bit real numbers. Again, it can be assumed that the input data is arranged ensemble-by-ensemble.

6.2 Mapping to TI C64+ Core

The data flow used for mapping this algorithm to the TI C64+ architecture is given in Figure 6. This mapping shows both the outer loops and the inner loops have been unrolled in parallel so as to facilitate scheduling without imposing large multiplicity constraints on *N*. In addition, Table 4 shows CPU units which contain the various instructions. From Table 4, you can see that this algorithm is multiply-limited and would need *DN/2* cycles to process D decimated points translating to a pipelined performance of *N/2* cycles/output point.

Test code using C and intrinsics was used to achieve pipelined kernel performance of *N/2* cycles-per-output point, matching the benchmarks, while maintaining NMSE performance less than 1e-6 with respect to the floating point output.

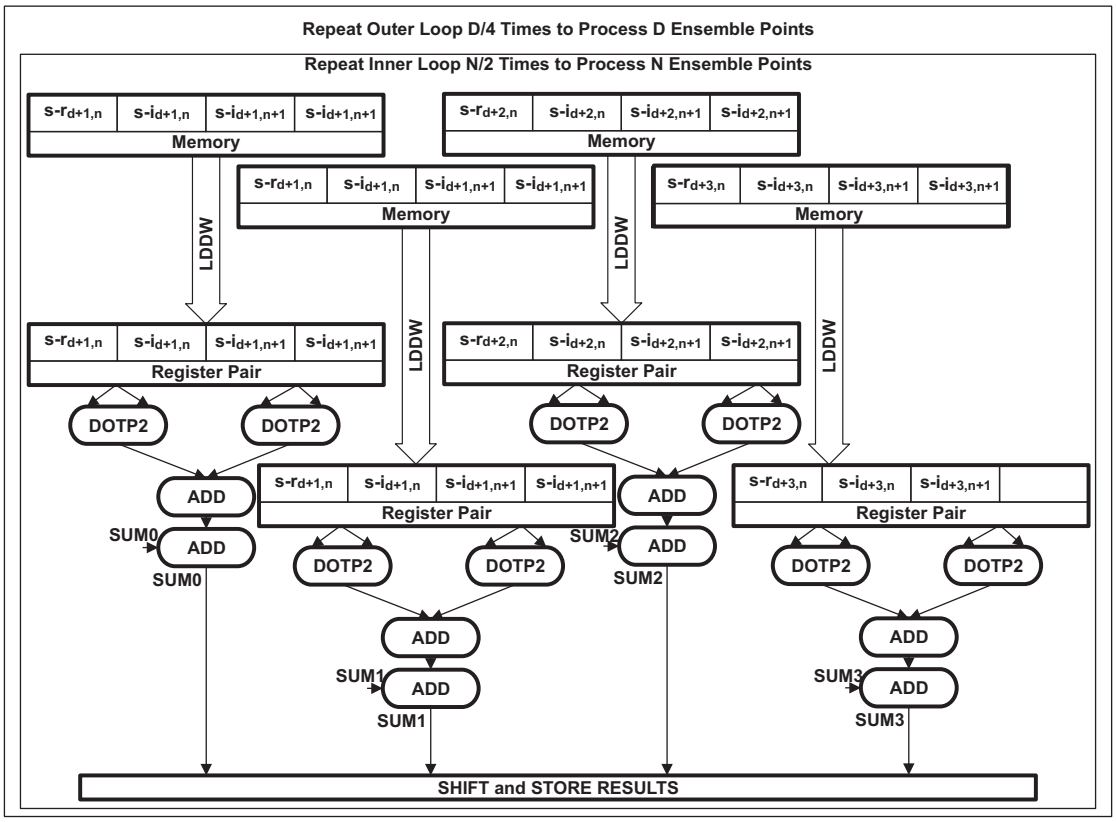

Figure 6. Mapping Flow Power Estimation to TI C64+ Architecture

Table 4. Complexity Analysis for Flow Power Operation

Operation	Cycles	Instruction	CPU Units
Loads – Input	DN/4	LDDW	D1/D2
Multiplies	DN/2	DOTP2	M1/M2
Adds	D(N–1)/2	ADD	L1/L2
Stores	D/8	STDW	D1/D2

7 Flow Parameter Estimator

7.1 Algorithm Description

Based on input, Y, this module estimates blood flow velocity, power, and turbulence using autocorrelation-based techniques. At depth point, *d*, for computing both its velocity, v_d, and turbulence, t_d, estimates, it uses the correlations between adjacent ensemble points, c_d, and ensemble power, p_d, as shown below,

$$p_d = \sum_{n=0}^{N-2} y_{d,n} \cdot y^*_{d,n}, \tag{7}$$

$$c_d = \sum_{n=0}^{N-2} y_{d,n+1} \cdot y^*_{d,n}, \tag{8}$$

$$v_d = tan^{-1}\left\{\frac{Im(c_d)}{Re(c_d)}\right\}, \tag{9}$$

$$t_d = 1 - \left\{\frac{|c_d|}{p_d}\right\}, \tag{10}$$

The module outputs estimates for flow power, velocity, and turbulence. The input, $Y_{DxN} = \{y_{d,n}\}$ consists of packed 16-bit complex values; the power, velocity, and turbulence output consists of 16-bit real data.

7.2 Mapping to TI C64+ Core

For the purposes of this work, the DSP IQMath library [8] is used for the four quadrant inverse tangent (_IQNatan2), division (_IQNdiv), and magnitude (_IQNmag) calculations needed for this algorithm. All these functions use 32-bit inputs to produce 32-bit outputs. Although lower precision may be sufficient for this application, they are not discussed in this document. The correlation outputs are kept at 32-bits and all intermediate sums use 40-bit accumulators.

This module consists of three kernels: one for power and correlation, the second one for velocity, and the last one for turbulence estimates. Figure 7 explains how the correlation and power computation kernel uses the DSP instructions to achieve its goal, and **Table 5** maps the used instructions to CPU units to arrive at an estimate of the complexity of the kernel. As shown in the table, this kernel is multiply-limited and needs a minimum of *3D(N–1)/2* cycles for processing *D* decimated points, The inverse tangent computation for velocity needs ~32 cycles per depth point using a pipelined implementation of _IQNatan2. In addition, the turbulence calculation involves using _IQNmag and _IQNdiv functions, taking ~13 and ~11 cycles, respectively, in pipelined implementations. Therefore, the turbulence calculation is expected to take ~24 cycles per output depth point, yielding a total pipelined performance of *3(N–1)/2 + 56* cycles-per-output point.

Test code using C and intrinsics was used to achieve pipelined kernel performance of *3(N–1)/2 + 51* cycles-per-output point while maintaining NMSE performance less than 1e-6 with respect to the floating point for all the outputs. Note that the kernel cycles are slightly better than expected since the IQMath functions provided better performance than their benchmarks.

Table 5. Complexity Analysis for Correlation and Power Calculations

Operation	Cycles	C64x+ Instruction	CPU Units
Loads	DN/4	LDDW	D1/D2
Multiplies			
• Correlations	D(N–1)	DOTP2, DOTPN2	M1/M2
• Power	D(N–1)/2	DOTP2	M1/M2
Adds			
• Correlations	D(N–2)	ADD	L1/L2
• Power	D(N–2)/2	ADD	L1/L2
Stores	D + D/4	STDW	D1/D2

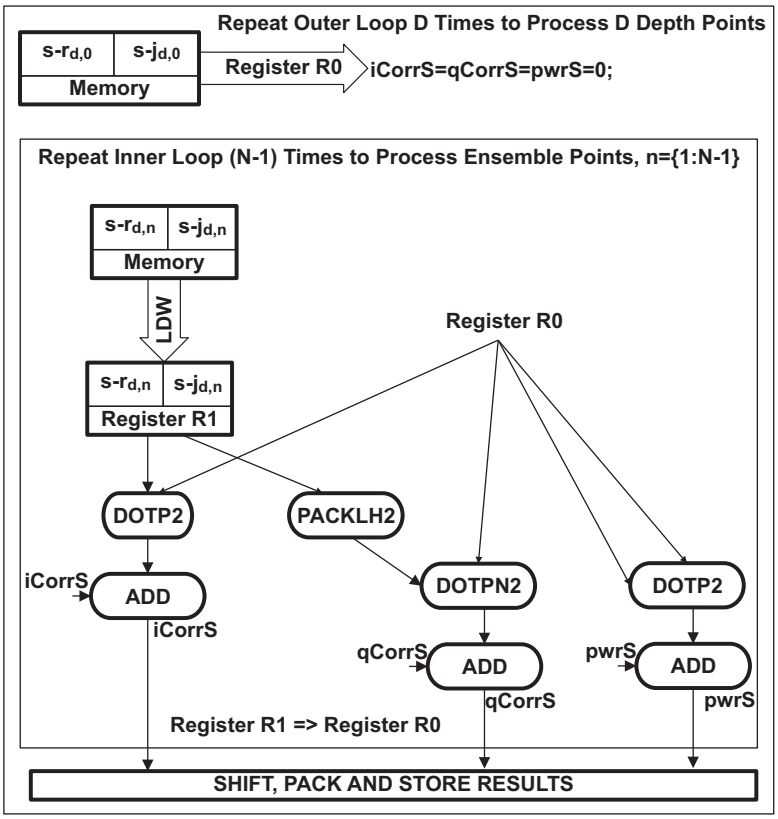

Figure 7. Mapping Correlation and Power Computations to TI C64+ Architecture

8 Conclusions

The commonly used color Doppler algorithms, namely RF demodulation, wall filter, flow power, and flow parameter estimation, can be mapped on to TI's C64x+ core. The complexity estimates (lower bounds) of these algorithms have been estimated for the C64x+ core. The complexity bound is especially tight for simple loops like the correlation/power computation kernel, but is loose for modules with multiple nested loops like the wall filter. For example, for D = 128 and N = 8, the correlation/power implementation takes 1371 cycles versus the benchmark of 1344 cycles, while the wall filter implementation takes about 8.9K cycles versus the benchmark of 4K cycles. However, all the algorithms cycles, consumed by the innermost kernels, exactly matched the theoretical expectations.

It was also seen that for small values of *N*, significant complexity savings can be achieved in the velocity and turbulence estimation loops by using lower precision and complexity variants of algorithms used for computing magnitudes, divisions, and inverses of the tangent.

For a color-flow system with frame rate, F, the RF demodulation, wall filter, and color flow algorithms together would use approximately:

$$\text{complexity}_{\text{cycles}} = O * F * B * D * \left[NS + \frac{NL}{2} + \frac{N^2}{2} + \frac{3(N-1)}{2} + 56 \right] \text{cycles}, \quad (11)$$

where *O* is the overhead factor, which is the difference between the estimates and the implementations. Assuming parameters $B = 64$, $S = 8$, $L = 64$, $D = 128$, $N = 10$, $F = 30$ and $O = 2$, the kernels would take approximately ~250 MHz of processing power, or ~25% of a 1GHz DSP.

Therefore, such implementations of the ultrasound algorithms would use a fraction of the available resources on TI's high performance DSP chips. Developing such DSP-based systems would allow for an overall reduction in system cost with respect to ASICs, while adding the flexibility of a programmable device.

9 References

1. *TMS320C64x/C64x+ DSP CPU and Instruction Set Reference Guide* (SPRU732)
2. *TMS320C6000 Programmers Guide* (SPRU198)
3. *Hand-Tuning Loops and Control Code on the TMS320C6000* (SPRA666)
4. Magee, David P., Matlab Extensions for the Development, Testing and Verification of Real-Time DSP Software, *Proceedings of the 42nd Design Automation Conference*, DAC 2005, San Diego, CA, USA, June 13-17, 2005, pp.603-606.
5. Steinar Bjaxum, Hans Torp and Kjell Kristoffersen, Clutter Filter Design for Ultrasound Color Flow Imaging, *IEEE Trans. on ultrasonics, Ferroelectrics, and Frequency Control*, Vol. 49, No. 2, pp. 204-216, Feb. 2002.
6. C. Kasai, K. Namekawa, A. Koyano, and R. Omoto, Real-Time Two Dimensional Blood Flow Imaging Using an Autocorrelation Technique, *IEEE Trans. Sonics Ultrasonics*, vol. SU-32, pp. 458- 464, 1985.
7. Application to sample-rate alteration and filter banks", *IEEE Trans. Acoust.*, Speech, Signal Processing, Vol. ASSP-24, No. 2, pp. 109-114, Apr. 1976 8.
8. Download: *C64x+ IQMath Library - A Virtual Floating Point Engine* (SPRC542)

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Clocks and Timers	www.ti.com/clocks	Digital Control	www.ti.com/digitalcontrol
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Telephony	www.ti.com/telephony
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265