Software-based Ultrasound Beamforming on Multi-core DSPs

Kerem Karadayi[1], Cheoljin Lee[1], and Yongmin Kim[1,2]

Image Computing Systems Laboratory
Departments of Electrical Engineering[1] and Bioengineering[2]
University of Washington
Seattle, WA 98195

1. Introduction

Ultrasound imaging is an important diagnostic tool in clinical medicine. It displays images in real time and is considered safe, noninvasive and less costly than other cross-sectional imaging modalities, e.g., X-ray CT, MRI and PET. In modern ultrasound machines, digital beamforming (focusing) is widely used. It is essential for attaining good image quality by increasing signal-to-noise ratio (SNR), improving spatial resolution and reducing sidelobe artifacts [1].

Digital beamforming is the most compute-intensive block in modern ultrasound systems. As a result, it has traditionally been supported by hardwired architectures (e.g., with application-specific integrated circuits (ASICs)). Due to the recent trend of programmable processors going multi-core, it might be possible to support this compute-intensive digital beamforming in software on these multi-core architectures.

In this study, we investigate the feasibility of supporting digital beamforming on one of Texas Instruments' (TI) C66x multi-core processors, i.e., C6678 [2].

2. Overview of Digital Beamforming in Ultrasound

Figure 1 shows the high-level block diagram of an ultrasound system. An ultrasound transducer with many piezoelectric elements converts electrical pulses from the transmitter into an acoustic beam. This acoustic beam propagates into the body, where echoes from interfaces between tissues with different acoustic impedances are reflected back to the transducer and converted into electrical signals by multiple transducer elements. Each transducer element is connected to an analog-to-digital converter (ADC) followed by a digital beamforming channel. After analog signal conditioning, signals from transducer elements are digitized by ADCs, so that digital beamforming can be performed. Following beamforming, demodulation is applied to remove the carrier frequency of the received ultrasound data to extract the complex baseband data (i.e., in-phase (I) and quadrature (Q) components). The baseband data are passed onto the back-end of an ultrasound system where signal and image processing for various ultrasound modes (e.g., B-mode, color Doppler, spectral Doppler and elastography) are performed before display.

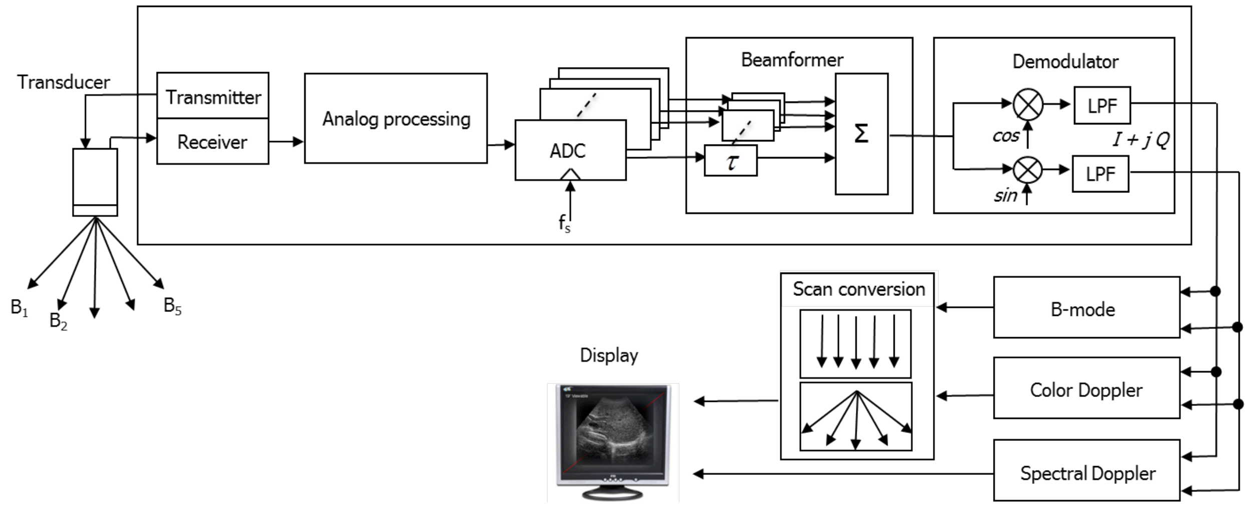


Figure 1. Block diagram of a medical ultrasound machine.

Figure 2 is a schematic diagram showing the principle of ultrasound beamforming. Echoes arrive at transducer elements at different times depending on the location of a target point and the element's position on the transducer. In Fig. 2, the center element receives the echo from the target point sooner than off-center elements, and the two outermost elements receive the echo last. By applying a proper time delay to each channel (e.g., τ for the center element), the received echoes are aligned properly before they are summed up coherently. As a result, higher SNR and better spatial resolution are obtained, leading to improved image quality.

For high-quality beamforming, ultrasound data need to have a fine temporal resolution, which requires very high-frequency ADCs. For example, in case of a 2-ns delay resolution in beamforming, a 500-MHz ADC per channel is required. Modern ultrasound machines have 32 to 256 channels. Because of the impracticality of this requirement, a beamforming algorithm is used to attain a fine delay resolution (before summing up) via interpolation of ultrasound data sampled at a lower clock frequency. This is called interpolation beamforming (IBF).

Figure 3 shows the flow diagram of the IBF algorithm. Once analog echo signals are sampled by the ADCs (e.g., at 40 MHz), the IBF algorithm performs interpolation (upsampling and lowpass filtering) to increase the time delay resolution required in beamforming (e.g., 8× upsampling would result in a temporal resolution of 3.125 ns in case the ADC sampling frequency is 40 MHz). After interpolation, the samples would be at a finer time resolution, and a proper sample is selected depending on the delay for each channel. After the appropriate samples are selected from all the channels, they are summed together to form one focused RF sample. Since the focused RF samples still contain the carrier signal, they are demodulated to produce the complex baseband (I/Q) data for back-end processing.

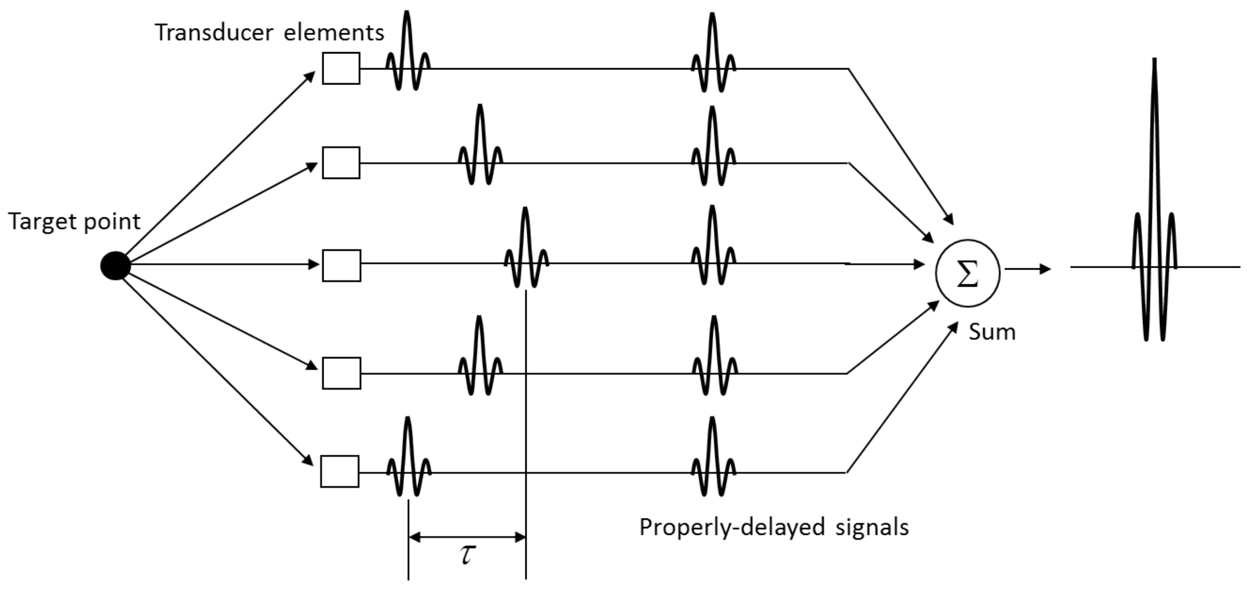


Figure 2. Simplified schematic diagram with 5 channels illustrating the principle of beamforming, which is to apply an appropriate delay to each channel before all the channels are summed. Analog processing and analog-to-digital converter after each transducer element are not shown here.

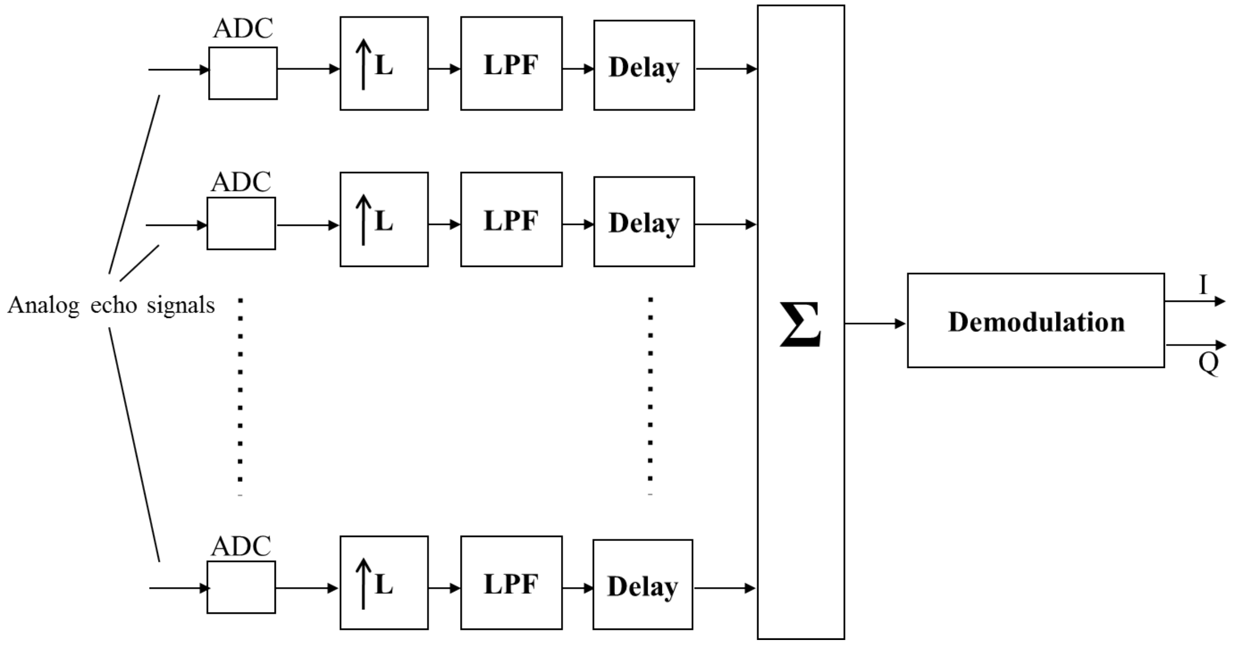


Figure 3. Interpolation beamforming (IBF) block diagram. L is the upsampling ratio.

3. TMS320C6678 Multi-core DSP Architecture

C6678 is a new-generation multi-core DSP from TI [2]. As shown in Fig. 4, it has 8 cores, where cores run at a clock frequency of up to 1.25 GHz. Each core is a very-long-instruction-word (VLIW) processor with two data paths, each with four functional units (L, S, M and D).

The L, S and D units accommodate 32- or 64-bit integer and partitioned arithmetic and logical operations. In addition, the D unit handles loads and stores of 8-, 16-, 32- or 64-bit data. The M unit supports various multiply and inner-product operations as well as some bitwise operations.

Each core has 32-kbyte level-1 program (L1P) memory and 32-kbyte level-1 data (L1D) memory, both of which can be configured as cache and/or addressable SRAM. In addition, each core has 512 kbytes of local level-2 (L2) memory. Similar to L1 memory, L2 can be configured as cache and/or addressable SRAM. Also, C6678 DSP has separate 4-Mbyte MSMC SRAM that is shared by all 8 cores.

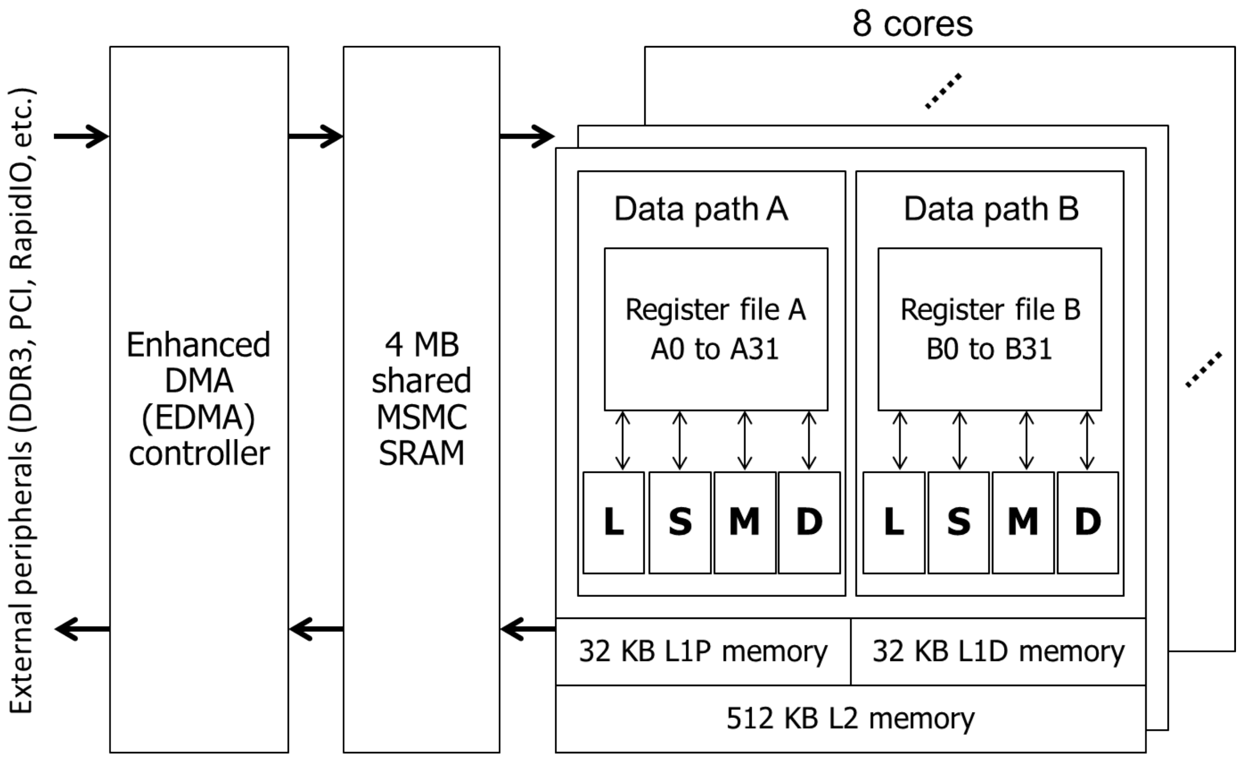
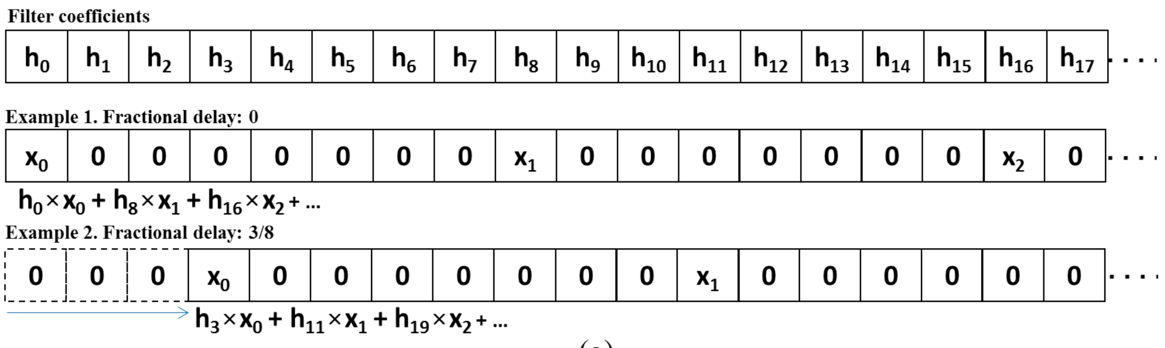

8 cores
External peripherals (DDR3, PCI, RapidIO, etc.)
Enhanced DMA (EDMA) controller
4 MB shared MSMC SRAM
Data path A
Data path B
Register file A A0 to A31
Register file B B0 to B31
L
S
M
D
L
S
M
D
32 KB L1P memory
32 KB L1D memory
512 KB L2 memory

Figure 4. Simplified block diagram of Texas Instruments C6678.

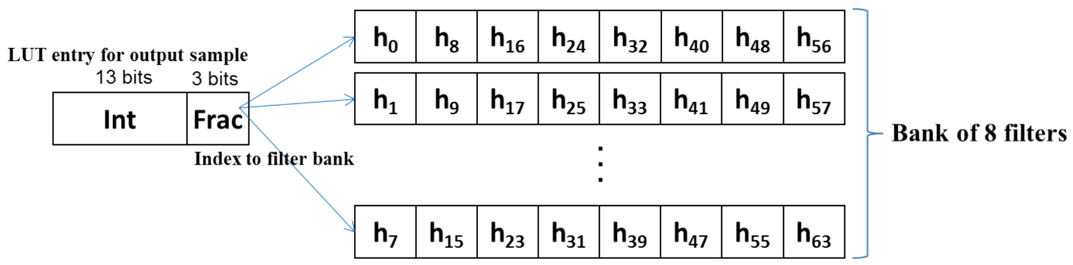
Each core has a programmable internal direct memory access (IDMA) controller that allows data transfers between L2 and L1 memory concurrent with CPU processing [3]. In addition, there is a separate programmable enhanced direct memory access (EDMA) controller shared among cores, which allows data transfers across external memory, memory-mapped peripherals and on-chip memory. Programmers can use IDMA and EDMA controllers to hide data transfers behind computation via double buffering to improve performance [4].

4. Mapping of IBF Algorithm on C6678 DSP

In mapping the IBF algorithm on C6678, we used a 64-tap filter to support 8× interpolation, which was implemented based on polyphase decomposition. Figure 5(a) shows the filter coefficients (h_i, where i = 0, 1, ..., 63) and the input samples (x_i, where i = 0, 1, 2, ...) aligned appropriately depending on a fractional delay. Upsampling starts with inserting seven 0s between consecutive input samples before filtering. We do not have to explicitly perform multiplications between the filter coefficients and these inserted 0s. Furthermore, the coefficients used in multiplications with the original (no 0s inserted) samples are determined by the fractional delay amount in increments of 1/8th of the original sampling period (e.g., 3.125 ns in case of 40-MHz ADC). The polyphase decomposition method utilizes a filter bank design based upon the number of filter taps and the upsampling factor, as shown in Fig. 5(b). For each output in filtering, only the original input samples are used with an 8-tap filter selected from the filter bank. This approach helps reduce the memory/register usage as well as the number of multiplications.


Filter coefficients
h0 h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 ....
Example 1. Fractional delay: 0
x0 0 0 0 0 0 0 0 x1 0 0 0 0 0 0 0 x2 0 ....
h0×x0 + h8×x1 + h16×x2 + ...
Example 2. Fractional delay: 3/8
0 0 0 x0 0 0 0 0 0 0 0 x1 0 0 0 0 0 0 ....
h3×x0 + h11×x1 + h19×x2 + ...

(a)


LUT entry for output sample
13 bits
3 bits
Int
Frac
Index to filter bank
h0 h8 h16 h24 h32 h40 h48 h56
h1 h9 h17 h25 h33 h41 h49 h57
h7 h15 h23 h31 h39 h47 h55 h63
Bank of 8 filters

(b)

Figure 5. Polyphase decomposition in case of an upsampling ratio of 8 where (a) the input samples are aligned according to a fractional delay and (b) a specific filter is selected by the 3-bit fractional delay. Because of the causality of the interpolation filter (which results in a group delay of 31.5 (63/2) 1/8th samples), the output is centered at the middle of the original 64-tap filter. This offset is incorporated into the calculation of integer and fractional delay amounts of each LUT entry.

Because determining the delay values on-the-fly tends to be expensive due to the requirement of trigonometric and square-root operations, we used a pre-computed LUT to reduce the amount of computation during beamforming. Each LUT entry consists of integer and fractional delay parts as shown in Fig. 5(b). The integer part represents a gross delay in the original ADC time resolution (e.g., 25 ns in case of 40 MHz), and the fractional part represents a fine delay within one sample period. This approach, however, requires additional memory to store the LUT. For example, if 2 bytes per entry are used as shown in Fig. 5(b), a total of 512 kbytes would be needed to support 64 channels and 4096 beamformed samples per scanline in case of a convex or linear array transducer. If a phased array transducer is used, it will require a different LUT per channel for each scanline. These LUTs may be loaded from memory or computed whenever a new probe is connected and a new set of imaging parameters (i.e., sector angle and scanline density) are selected by the operator.

Figure 6 is the pseudo-code for the IBF tight loop. After initializing the partial sum array to zero, the tight loop kernel computes one properly-delayed output sample and accumulates it into the partial sum array. This occurs for every sample in each channel (n = 0 to 4095), and then for every channel (m = 0 to 63). Once all 64 channels have been processed, the final partial sum array becomes a focused scanline.

```
for (n=0;n<4096;n++)
        partial_sum[n] = 0; //Initialize the partial sum array to zero

for (m=0;m<64;m++) //for each channel m
{
        for (n=0;n<4096;n++) // for each output sample n in channel m
        {
                load LUT_entry[m, n];
                input_index = msb13(LUT_entry[m, n]);
                filter_index = lsb3(LUT_entry[m, n]);
                load input_array[m][input_index:input_index+7] => input_samples; //load 8 input samples
                load filter_bank[filter_index][0:7] => filter_coeffs; //load 8 filter coefficients

                interpolated_sample = inner_product(filter_coeffs, input_samples);
                partial_sum[n] =  partial_sum[n] + interpolated_sample;
        }
}

for (n = 0; n< 4096; n++) // for each focused RF sample n in a scanline
{
        output[n] = partial_sum[n];
}
```

Figure 6. Pseudo-code for the IBF tight loop for 64 channels and 4096 samples per channel with 8× interpolation.

Figure 7 shows how the tight loop in Fig. 6 is implemented using the C6678 instruction set [5], where each iteration of the tight loop processes two output samples. The integer part and fractional part of each LUT entry are used as an input address for input channel data and a row index for the filter bank, respectively. Since the required 8 input samples may not necessarily be aligned on a 64-bit (8-byte) boundary in memory, a non-aligned load instruction, LDNDW, is used. By using DDOTP4H (two parallel inner-product operations, each between four input channel data and 4 filter coefficients) followed by ADD, interpolation filtering is performed. DDOTP4H can complete these inner-product operations with a single-cycle throughput. Since each inner product multiplies and adds 4 pairs of 16-bit values, 8-tap filtering can be completed with one DDOTP4H instruction and one ADD. Two output samples (X_{curr1} and X_{curr2}) computed this way are then accumulated into the corresponding partial sum using a DADD instruction. DDOTP4H and DADD are two of C66x's new instructions.

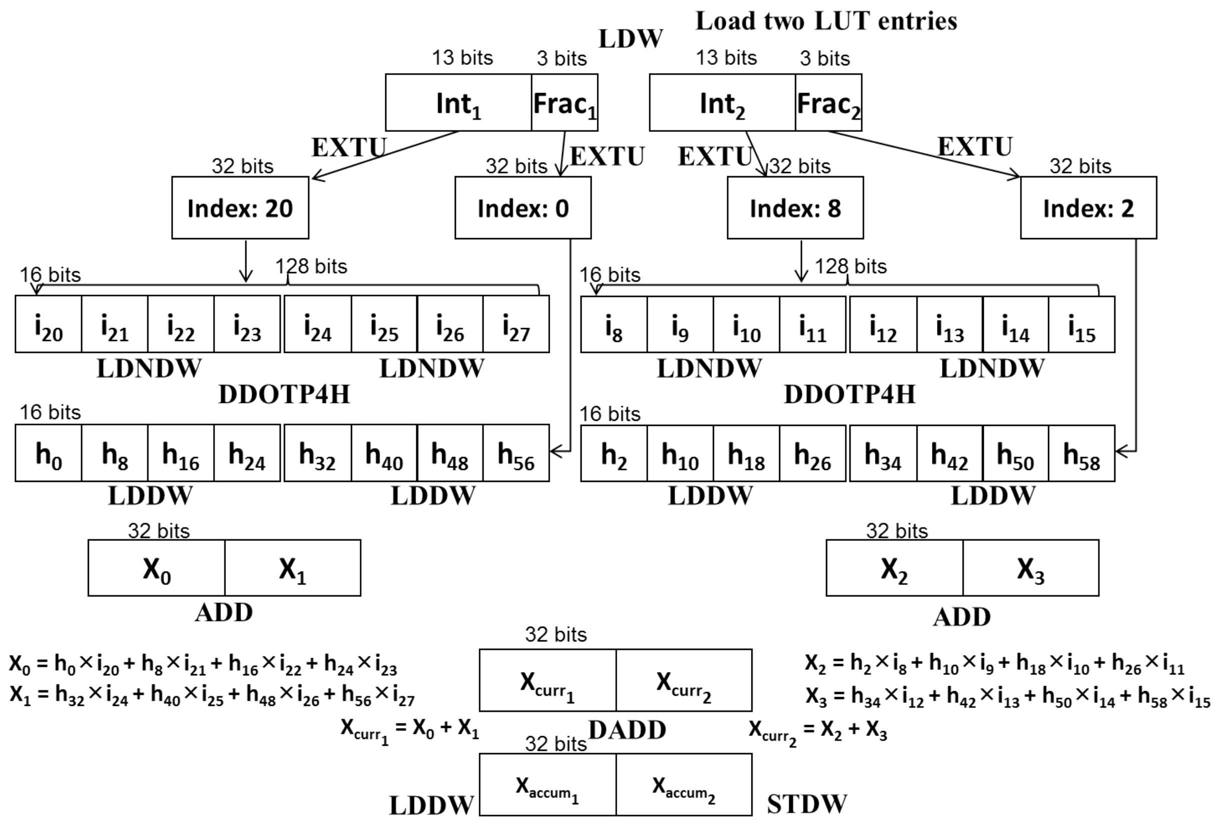

Load two LUT entries
LDW
13 bits
3 bits
Int_1
$Frac_1$
Int_2
$Frac_2$
EXTU
32 bits
Index: 20
Index: 0
Index: 8
Index: 2
16 bits
128 bits
i_{20} i_{21} i_{22} i_{23} i_{24} i_{25} i_{26} i_{27}
i_8 i_9 i_{10} i_{11} i_{12} i_{13} i_{14} i_{15}
LDNDW
DDOTP4H
h_0 h_8 h_{16} h_{24} h_{32} h_{40} h_{48} h_{56}
h_2 h_{10} h_{18} h_{26} h_{34} h_{42} h_{50} h_{58}
LDDW
X_0 X_1
X_2 X_3
ADD
$X_0 = h_0 \times i_{20} + h_8 \times i_{21} + h_{16} \times i_{22} + h_{24} \times i_{23}$
$X_1 = h_{32} \times i_{24} + h_{40} \times i_{25} + h_{48} \times i_{26} + h_{56} \times i_{27}$
$X_2 = h_2 \times i_8 + h_{10} \times i_9 + h_{18} \times i_{10} + h_{26} \times i_{11}$
$X_3 = h_{34} \times i_{12} + h_{42} \times i_{13} + h_{50} \times i_{14} + h_{58} \times i_{15}$
X_{curr_1} X_{curr_2}
$X_{curr_1} = X_0 + X_1$
$X_{curr_2} = X_2 + X_3$
DADD
X_{accum_1} X_{accum_2}
LDDW
STDW

Figure 7. Instruction mapping on C66x.

Table 1 shows how the instructions for the IBF tight loop shown in Fig. 7 are mapped to the L, S, M and D functional units of the C6678 DSP [6]. As can be seen, this tight loop is D-unit-bound, taking 15 cycles per two output samples per data path. Therefore, the ideal performance of the IBF tight loop is 983 kcycles in a single core (7.5 cycles/sample × 4096 output samples × 64 channels / 2 data paths). In Table 1, we counted one LDNDW instruction as 2 cycles on the D unit because this instruction uses the resources for two aligned 64-bit loads simultaneously to extract and combine the necessary data for a single non-aligned address. This would prevent the D unit of the other data path from issuing another load or store in the same cycle.

Once the tight loop was designed as shown in Fig. 7 and its instructions are mapped as in Table 1, we coded it in Code Composer Studio and assessed the computation time including cache misses by running the tight loop on the cycle approximate simulator. We measured a total computation cycles of 1202 kcycles, which includes compiler and function call overhead as well as stalls of 217 kcycles due to L1D and L1P cache misses (with the data and program in L2 SRAM). To reduce stall cycles from cache misses, we attempted to use IDMA to hide data transfers between L2 and L1 memory via double buffering while the IBF tight loop is processing the data in L1D. However, because of IDMA's inability to efficiently manage 3 simultaneous double-buffered data streams needed in the tight loop (the input channel data, LUT and partial sum array), we found cache-based implementations were generally faster for our interpolation beamforming algorithm.

Table 1. Mapping of the IBF tight loop instructions on functional units

L	S	M	D
2 ADD	4 EXTU	2 DDOTP4H	1 LDW
1 DADD	2 ADD		4 LDDW
			4 (8)* LDNDW
			1 LDDW
			1 STDW
3 instructions	**6 instructions**	**2 instructions**	**11 instructions (15 cycles)**

* One LDNDW is counted as 2 cycles.

After experimenting with multiple configurations, the fastest cache-based performance of 1202 kcycles was achieved with L1D configured as 16 kbytes of SRAM and 16 kbytes of data cache, and the partial sum buffer allocated in L1D SRAM (with the LUT and input channel data in L2 SRAM). With this performance, assuming a clock frequency of 1 GHz, 64-channel beamforming for 3.41 Msamples/s (4096 beamformed samples / (1202 kcycles × 1 ns/cycle)) can be supported by a single core of C6678. If two cores are used to beamform a single scanline, with each core processing 32 out of 64 channels (as discussed in the following section), 6.82 Msamples/s can be beamformed by a pair of cores.

5. System-level Considerations

Figure 8 shows a system-level architecture using 2 C6678 DSPs to beamform 64 parallel receive channels. In this arrangement, the outputs from the upper 32 ADCs are directed into DSP0, while the outputs from the lower 32 ADCs go to DSP1. These channel data can be spooled into a circular buffer allocated in DSP's MSMC SRAM (Fig. 9). Since the scanline #0 data are now distributed among two DSPs (data from the first 32 channels in DSP0 and data from the last 32 channels in DSP1), both DSPs need to work together to beamform scanline #0. DSP0, after generating the partial sum array from the first 32-channel data, needs to pass this partial sum array to DSP1, which will add this partial sum array to its own results from its 32-channel data. In addition, DSP1 could perform demodulation on these beamformed RF samples, followed by transferring the scanline to the back-end processor.

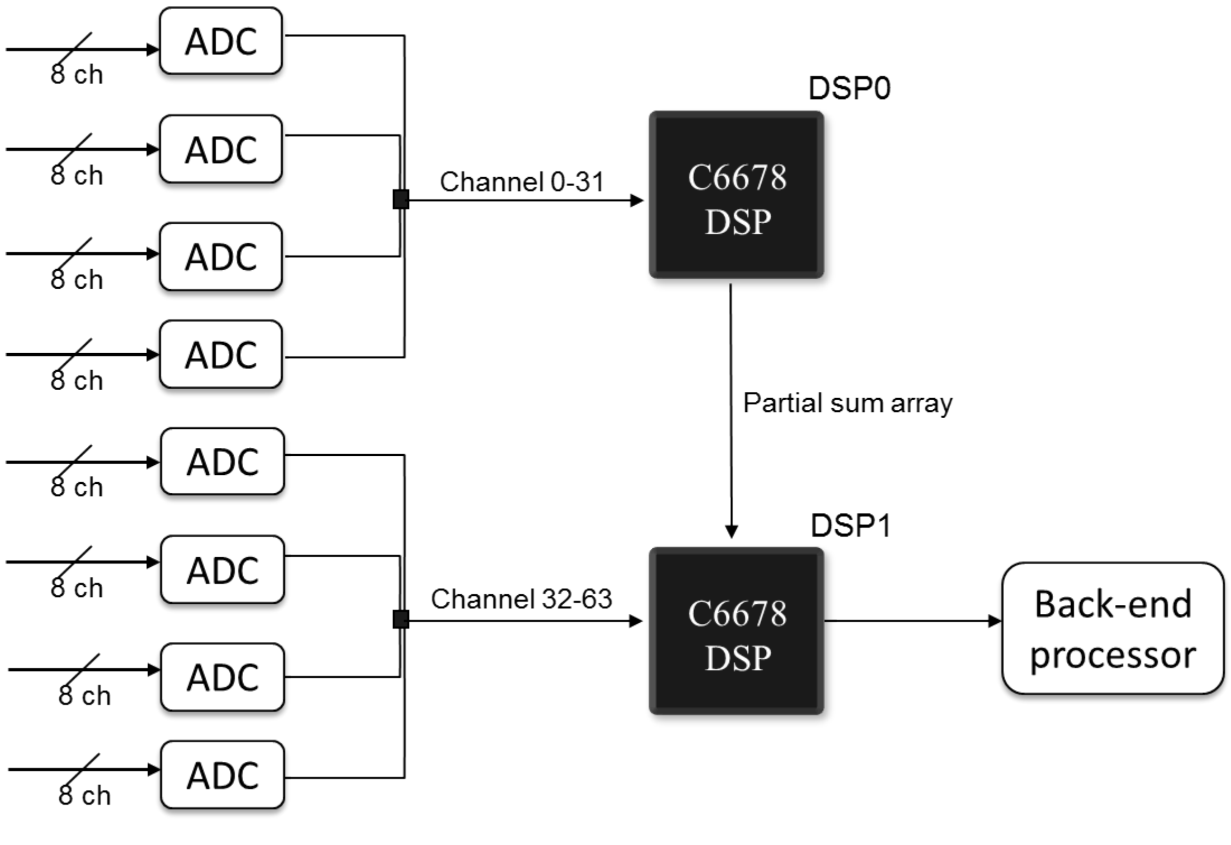


Figure 8. Simplified high-level block diagram of an ultrasound system for IBF using two C6678 DSPs.

Figure 9 shows the use of MSMC SRAM to buffer 32-channel scanline data for each core. Each small rectangular block in the buffer is 256 kbytes (32 channels × 4096 samples × 2 bytes/sample). Once all the data for scanline #0 have been transferred into this buffer, Core 0 can start beamforming with these data immediately, while the subsequent scanlines are being transferred into the following blocks in the circular buffer. Similarly, other cores can start processing as soon as data become available. Therefore, multiple cores work on their own scanlines in a staggered fashion. An example timing diagram is shown in Fig. 10.

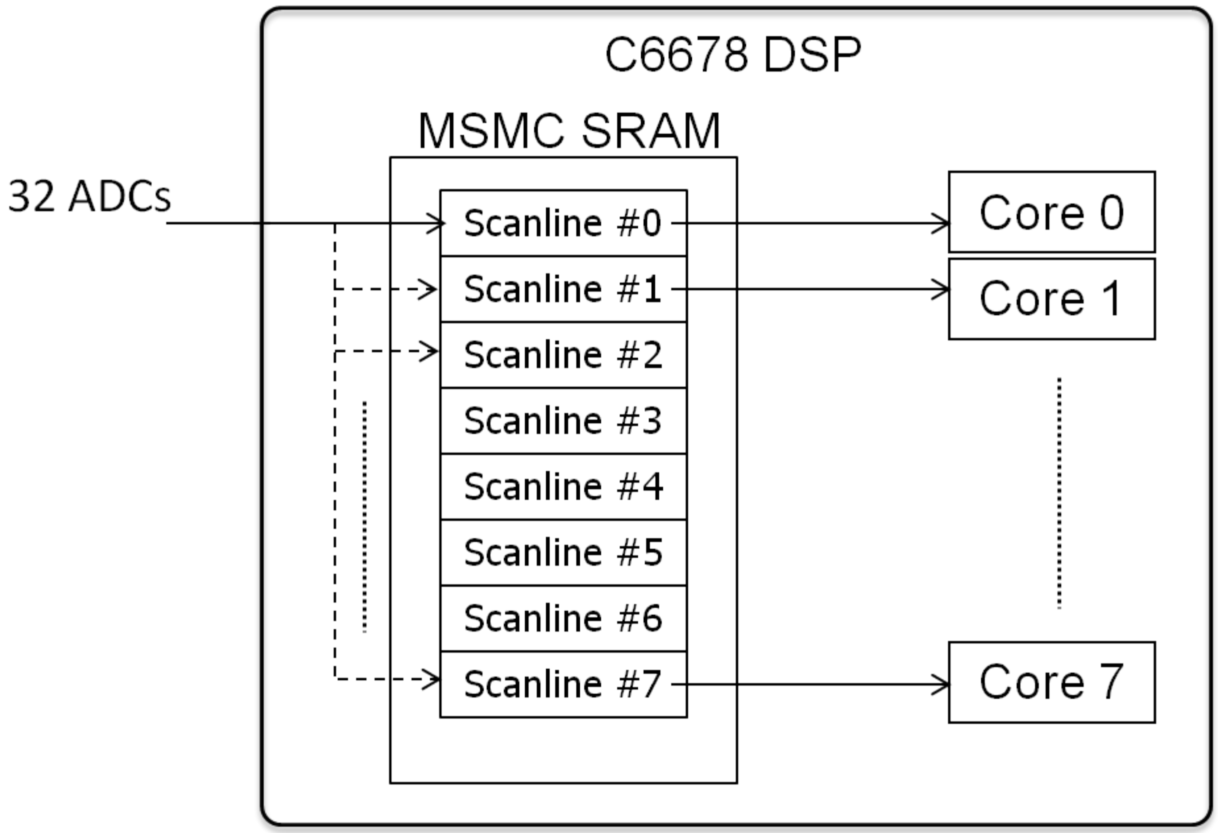


Figure 9. Input data streaming into a DSP. Each DSP processes only one half of the channels for each scanline (i.e., channel 0-31 in DSP0 and channel 32-63 in DSP1).

With this architecture, based on the tight loop computation time we measured in Section 4, 54.5 Msamples/s (6.82 Msamples/s × (8 pairs of cores across 2 DSPs)) can be ideally beamformed. In practice, the number of beamformed samples per second would be less than 54.5 Msamples/s because of the necessary I/O activities and overhead due to partitioning the computation among multiple cores and two DSPs. For example, according to Figs. 9 and 10, all 8 cores need to access MSMC SRAM, which could create access conflicts. We emulated this access to MSMC SRAM by running 8 concurrent threads of the IBF tight loop in SYS/BIOS (TI's real-time operating system) and running on the cycle approximate simulator. It takes 631 kcycles for a single core to produce a partial sum array for 32 channels of one scanline. We estimate that it would take an additional ~16 kcycles for DSP0 to transfer the partial sum array to DSP1. Then, DSP1 needs extra 10 kcycles to produce the beamformed scanline by adding partial sums from two DSPs. Therefore, it would take 657 kcycles to beamform a scanline using 2 cores across 2 C6678 DSPs, leading to a throughput of 49.9 Msamples/s with 2 DSPs assuming that they run at 1 GHz.

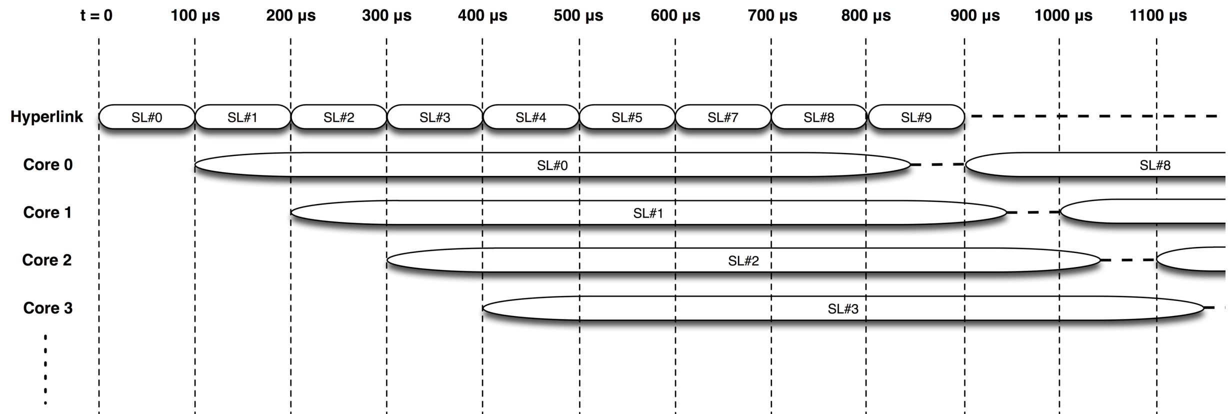


Figure 10. Example timing diagram for input data streaming into MSMC SRAM of a DSP in case one scanline is acquired every 100 µs and assuming that it takes 740 µs for each core to beamform a scanline. Each core starts processing the scanline assigned to it after all the data for that scanline are transferred into MSMC SRAM. In this diagram, it is assumed that data are brought from ADCs into C6678's MSMC SRAM via Hyperlink.

6. Discussion and Conclusion

Our results show that DSP-based ultrasound beamforming is possible. At 1 GHz, 49.9 Msamples/s can be beamformed for 64 channels with 2 C6678s. If we increase the clock frequency to 1.25 GHz, 62.3 Msamples/s could be beamformed. This would make it possible to support ADC sampling frequencies of up to 62.3 MHz, assuming I/O from ADCs to DSPs at this rate can be sustained. On the other hand, if an ADC sampling rate of 40 MHz is used and DSPs run at 1.25 GHz, each DSP would be ~64% loaded, leaving room for additional processing (e.g., demodulation and back-end processing tasks).

The beamforming performance of 49.9 Msamples/s with the architecture in Fig. 8 can support various clinical applications, as listed in Table 2. One example is neck imaging, where the carotid arteries are examined for plaques and blood flow abnormalities. Assuming an ADC sampling frequency of 40 MHz and an imaging depth of 5 cm, there are 2598 samples per scanline (40 MHz × 10 cm (round-trip distance) / 1540 m/s (speed of sound in tissue)). At this imaging depth, the maximum number of scanlines that can be beamformed is 19.2k, assuming there is no gap between the completion of previous scanline reception and transmitting an ultrasound beam for the next scanline. If 192 scanlines are acquired per frame, a frame rate of 100 frames/s can be achieved. If the scanline density is doubled (i.e., 384 scanlines), then 50 frames/s can be supported. These numbers are sufficient for imaging the carotid artery.

Another example is abdominal imaging, where abdominal organs (e.g., liver, kidney and aorta) are examined. Larger imaging depths are required in this case. Assuming an ADC frequency of

40 MHz and an imaging depth of 20 cm, there will be 10,390 samples per scanline. The maximum number of scanlines that can be acquired at this imaging depth is 4.8k. If 100 scanlines are acquired per frame, a frame rate of 48 frames/s can be achieved. With 200 scanlines per frame, the frame rate will decrease to 24 frames/s. Since abdominal organs do not move fast, this frame rate is typically sufficient.

Table 2. Example clinical applications

	Depth	Typical frame rate (frames/s)	Number of samples/scanline in case of 40-MHz ADC	Transducer type
Neck imaging	5 cm	30-50	2,598	Linear
Kidney	12 cm	15-30	6,234	Convex
Liver/Aorta/OB	20 cm	15-30	10,390	Convex

These results are obtained using the cycle approximate simulator since the hardware is not yet available to us. We expect the results to vary little with the actual hardware. Furthermore, our analysis does not include the overhead for streaming ultrasound channel data (sampled ADC outputs) via Hyperlink into the MSMC SRAM of C6678 since the simulator does not model this configuration.

It is interesting to note that in Table 1 there are numerous unused instruction slots in the IBF tight loop. The total number of unused instruction slots is 38 (12 for L, 9 for S, 13 for M and 4 for D). Therefore, additional functionality could be supported without increasing the total computation time. As a matter of fact, we were able to add apodization to this IBF tight loop without any increase in the number of cycles by utilizing these unused instruction slots.

One optimization we did not perform in this implementation was to take advantage of the symmetry in the LUT. When there is no beam steering, delay values in the LUT are symmetric around the middle of the aperture, as shown in Fig. 2. For example, in case of a 64-channel system, the same set of delay values would be applied to channels 0 and 63, channels 1 and 62, channels 2 and 61 and so on. Based on this observation, the LUT size could be reduced in half if the channels are split across DSPs so that the channels with the same set of delay values are processed by the same DSP (e.g., channels 0-15 and 48-63 go into DSP0, whereas channels 16-47 go into DSP1). However, since beam steering is used in modern ultrasound systems even with linear and convex transducers (e.g., for oblique sectors or spatial compounding), we kept the original LUT size to keep the flexibility of adding beam steering in the future.

In this paper, we have demonstrated the feasibility of a software-based ultrasound beamformer using multi-core DSPs. This software-based architecture could be advantageous in achieving shorter time-to-market (e.g., compared to designing beamformer ASIC chips), flexibility (different beamforming algorithms can be supported) and reusability. Also, this approach is scalable. For example, a portable, low-cost ultrasound system with 32 channels could be developed using a single C6678, whereas 2~4 DSPs could support a mid-end system with more channels (e.g., 64 - 128). In the past, analog front-end electronics and digital beamformers have been the key determinants in differentiating ultrasound machines by ultrasound manufacturers. With the increasing availability of off-the-shelf components, such as DSPs that are ready for use in beamforming, a paradigm shift in ultrasound system design would take place, where the hardware and ultrasound processing library (beamforming, B-mode, color Doppler, spectral Doppler, elastography, and 3D) become commoditized. In this era, new clinical applications and algorithms, i.e., to support these clinical applications and further improve image quality and diagnostic capability of ultrasound machines, would become more important for engineers/researchers and ultrasound manufacturers.

References

[1] B. D. Steinberg, “Digital beamforming in ultrasound,” *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 39, pp. 716–721, 1992.

[2] *TMS320C6678 Multicore Fixed and Floating-point Digital Signal Processor Data Manual*. Texas Instruments, Literature Number SPRS691, November 2010.

[3] *TMS320C66x DSP CorePac User Guide*. Texas Instruments, Literature Number SPRUGW0A, November 2010.

[4] D. Kim, R. Managuli and Y. Kim, “Data cache and direct memory access in programming mediaprocessors,” *IEEE Micro*, vol. 21, no. 4, pp. 33-42, 2001.

[5] *TMS320C66x DSP CPU and Instruction Set Reference Guide.* Texas Instruments, Literature Number SPRUGH7*,* November 2010.

[6] K. Karadayi, V. Markandey, R. J. Gove and Y. Kim, “Strategies for mapping algorithms to mediaprocessors for high performance,” *IEEE Micro*, vol. 23, no. 4, pp. 58–70, 2003.

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**	
Audio	www.ti.com/audio	Communications and Telecom	www.ti.com/communications
Amplifiers	amplifier.ti.com	Computers and Peripherals	www.ti.com/computers
Data Converters	dataconverter.ti.com	Consumer Electronics	www.ti.com/consumer-apps
DLP® Products	www.dlp.com	Energy and Lighting	www.ti.com/energy
DSP	dsp.ti.com	Industrial	www.ti.com/industrial
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Security	www.ti.com/security
Logic	logic.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Power Mgmt	power.ti.com	Transportation and Automotive	www.ti.com/automotive
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Mobile Processors	www.ti.com/omap		
Wireless Connectivity	www.ti.com/wirelessconnectivity		
		TI E2E Community Home Page	e2e.ti.com

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2011, Texas Instruments Incorporated