Software-based Ultrasound Phase Rotation Beamforming on Multi-core DSP

Jieming Ma

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Electrical Engineering

University of Washington
2012

Committee:
Yongmin Kim
Linda Shapiro

Program Authorized to Offer Degree:
Department of Electrical Engineering

University of Washington

Abstract

Software-based Ultrasound Phase Rotation Beamforming on Multi-core DSP

Jieming Ma

Chair of the Supervisory Committee:
Professor Yongmin Kim
Departments of Electrical Engineering and Bioengineering

Phase rotation beamforming (PRBF) is a commonly-used digital receive beamforming technique. However, due to its high computational requirement, it has traditionally been supported by hardwired architectures (e.g., application-specific integrated circuits (ASICs) or more recently field-programmable gate arrays (FPGAs)). In this thesis, we investigate the feasibility of supporting software-based PRBF on a multi-core DSP. To alleviate the high computing requirement, the analog front-end (AFE) chips integrating quadrature demodulation in addition to analog-to-digital conversion were defined and used. With these new AFE chips, only delay alignment and phase rotation need to be performed by DSP, substantially reducing the computational load. We implemented the delay alignment and phase rotation modules on a Texas Instruments C6678 DSP with 8 cores. With a sampling rate of 40 MHz and 2:1 decimation, it takes 200 µs to generate one scanline (2048 samples/scanline) on two cores. With 4 cores, it can support beamforming for 64 channels with 10k scanlines/s, e.g., 200 scanlines/frame at 50 frames/s. The remaining 4 cores can work on back-end processing tasks and applications, e.g., color Doppler or ultrasound elastography. One DSP being able to handle both beamforming and back-end processing could lead to low-power and low-cost ultrasound machines, benefiting ultrasound imaging in general, particularly portable ultrasound machines.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Table number Page

ACKNOWLEDGEMENTS

The work described in this thesis would not have been possible without guidance and help from many people. Most of all, I would like to express my sincere appreciation to my advisor Dr. Yongmin Kim for providing me with the opportunity to work on this project. His dedication to perfection and professionalism has led to the research in this thesis, and will continue to be a lasting inspiration for me.

I would like to thank Dr. Kerem Karadayi for his help and support. I am grateful for his valuable advice for the system design and implementation.

I would like to thank Dr. Murtaza Ali of Texas Instruments for his support and providing insight and experience throughout this project.

I would like to thank Si Luo for his support and encouragement for the work in this thesis.

I would also like to thank Canxing Xu and Cheoljin Lee for helpful discussion. I am grateful to the rest of the Image Computing Systems Laboratory (ICSL) for their support and encouragement.

DEDICATION

To my husband and parents.

Chapter 1: Introduction

1.1 Ultrasound system

Ultrasound imaging is an important medical imaging modality for routine clinical use in hospital. It transmits high-frequency sound waves and generates images using the returned echoes that are produced at tissue interfaces where changes in acoustic impedance occur. Compared to other cross-sectional imaging modalities, e.g., CT, MRI and PET, ultrasound has the advantages of being portable and low cost.

Ultrasound machines have three subsystems: the transducer and its cable assembly, the front-end electronics, and the back-end processor with display.[1] Figure 1.1 shows a high-level block diagram of an ultrasound system. An ultrasound transducer with piezoelectric elements converts electrical pulses from the transmitter into an acoustic beam. The acoustic beam propagates into the body, and echoes generated at tissue interfaces with different acoustic impedances are reflected back to the transducer and converted into electrical signals. Each transducer element is connected to an analog-to-digital converter (ADC) where analog signals are digitized. Afterwards, digital beamforming is performed. In the beamformer, demodulation is applied to remove the carrier frequency of received ultrasound signals to extract the complex baseband data (i.e., in-phase (I) and quadrature (Q) components). The baseband data are then passed onto the back-end of an ultrasound system to perform signal and image processing for various ultrasound modes (e.g., B-mode, color Doppler, spectral Doppler and elastography) before display.

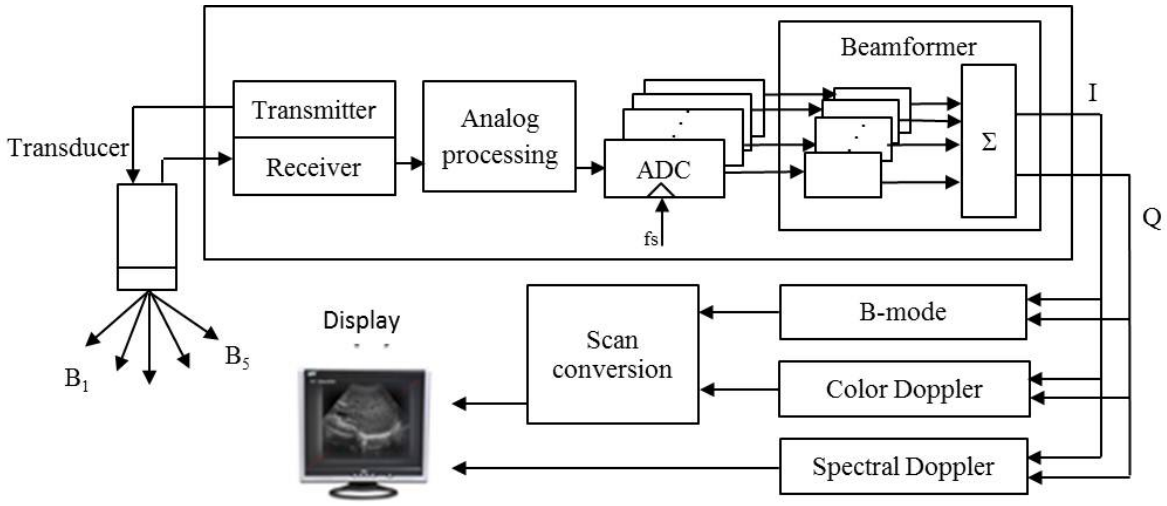


Figure 1.1 Block diagram of a medical ultrasound machine.

1.2 Digital beamforming in ultrasound

One of the most critical components of an ultrasound machine is the beamformer. Beamforming is a common signal-processing technique used in various applications such as wireless communications, radar, and sonar. In medical ultrasound, beamforming is used to focus the reflected echo signals from different tissue structures in the region of interest. In the receive beamformer, focusing is achieved by aligning echo signals arriving at different elements using appropriate delays so that an isophase plane is created. These aligned echoes are then summed up coherently.[2] The basic concept of receive beamforming is illustrated in Fig. 1.2. Ultrasound echoes arrive at different transducer elements at different times determined by the distance between the target in the tissue and the position of a transducer element. As the distance from the target to the center element is shorter, the center element receives the echo sooner than the off-center elements. By applying appropriate delay times, the echoes can be aligned up before summation. Digital beamforming is essential for achieving good image quality by increasing signal-to-noise-ratio (SNR), improving spatial resolution and reducing sidelobe artifacts.[3]

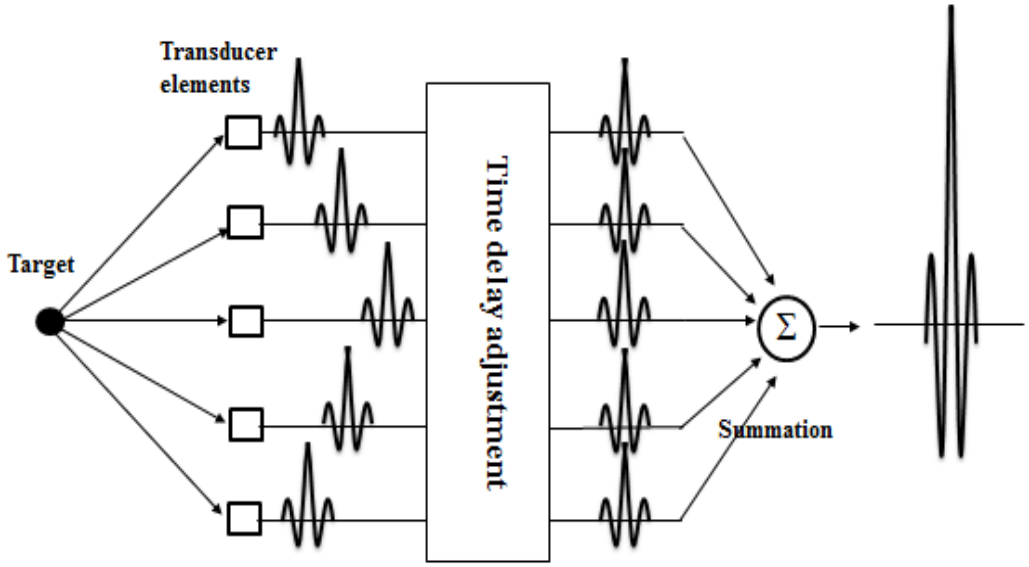


Figure 1.2 Simplified schematic diagram with 5 channels to illustrate the principle of receive beamforming. An appropriate delay is applied to each channel before all the channels are summed.

Phase rotation beamforming (PRBF) is a commonly-used digital receive beamforming technique, which performs coarse delay adjustment by integer sample shifting followed by fine delay adjustment by phase rotation.[4]

1.3 Research goal

The purpose of this thesis is to explore the feasibility of supporting software-based PRBF on a multi-core DSP.

PRBF requires high computing power. Therefore, it has traditionally been supported by hardwired architectures (e.g., application-specific integrated circuits (ASICs) or more recently field-programmable gate arrays (FPGAs) [5]). Programmable processors, e.g., digital signal processors (DSPs), have been used for back-end signal and image processing in ultrasound systems for some time [6-8], but not for front-end beamforming.

However, with a trend towards using more software in ultrasound imaging systems, some attempts have been made to support software-based beamforming using DSPs. Sohn et al.[9] presented a system using 16 ADSP-TS201 DSPs for beamforming. Chen et al.[10] proposed an implementation of digital beamforming on Am2045 massively parallel processor array (MPPA). On the other hand, some software-based approaches based on simplified algorithms were also attempted to reduce the burden in computation. A pixel-based beamforming algorithm was introduced.[11] Also, we proposed a two-stage demodulation phase rotation beamforming method to significantly reduce the amount of computation. However, it incurred some image quality degradation.[12]

One future direction with next-generation AFEs is for more integration of both analog and digital functionalities. The highly-integrated new AFE chip, which includes analog-to-digital converter, quadrature demodulation and decimation, would be beneficial to many systems. This highly-integrated AFE chip could substantially reduce the computation requirement in beamforming. At the same time, with the programmable processors going multi-core, more computational power is now available, and this trend will continue.

With the development of new analog front-end (AFE) chips and multi-core DSPs, it might be possible to support both full (not simplified) PRBF and back-end processing on the same multi-core DSP, which could lead to a lower-cost ultrasound machine that consumes much less power. On the other hand, it might also be possible to support PRBF for high-end machine by using one or more DSPs.

In this thesis, we proposed a new PRBF system architecture with highly integrated AFEs, and implemented it on a multi-core DSP (TMS320C6678). The performance was assessed by the code composer studio (CCS) cycle approximate simulator, and possible application scenarios have been explored.

1.4 Thesis organization

Chapter 2 describes the architecture of TMS320C6678 multi-core DSP.

Chapter 3 describes phase rotation beamforming. Computational requirement for each stage was estimated.

Chapter 4 presents the new PRBF structure. The algorithm was mapped on TMS320C6678 DSP. The performance was evaluated by the CCS cycle approximate simulator.

Chapter 5 explores the application scenarios. One scenario is where a multi-core DSP could support both front-end beamforming and back-end processing, which would lead to a compact ultrasound machine.

Finally in Chapter 6, we summarize our findings.

Chapter 2: TMS320C6678 multi-core DSP architecture

2.1 DSP architecture

C6678 is a new-generation DSP from Texas Instruments with 8 cores running at a clock frequency of up to 1.25 GHz.[13]

Figure 2.1 gives a simplified block diagram of C6678 DSP. 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 two general-purpose register files (A and B) each contain 32 32-bit registers for a total of 64 registers. Each of the eight functional units is capable of executing one instruction every clock cycle. The M unit performs all multiply operations. In addition to multiplying two real numbers, the M unit can also perform a complex multiplication every clock cycle, which enables us to perform phase rotation for one sample using the M unit in one clock cycle. The S and L units perform a general set of arithmetic, logical, and branch functions. The D unit loads data from memory to the register and stores results from the register into memory.

2.2 Memory description

Since the performance of processors has improved at a much faster pace than that of memory, there is a performance gap between core and memory speed. High-speed memory is more expensive than slow memory. To solve this problem, a memory hierarchy has been used. A fast but small memory is placed close to the core that can be accessed without stalls. The next lower memory levels are larger but slower. Typically, the data from lower-level memory are first cached into the higher-level small and fast memory so that the average memory access time will be close to the access time of the fastest memory.

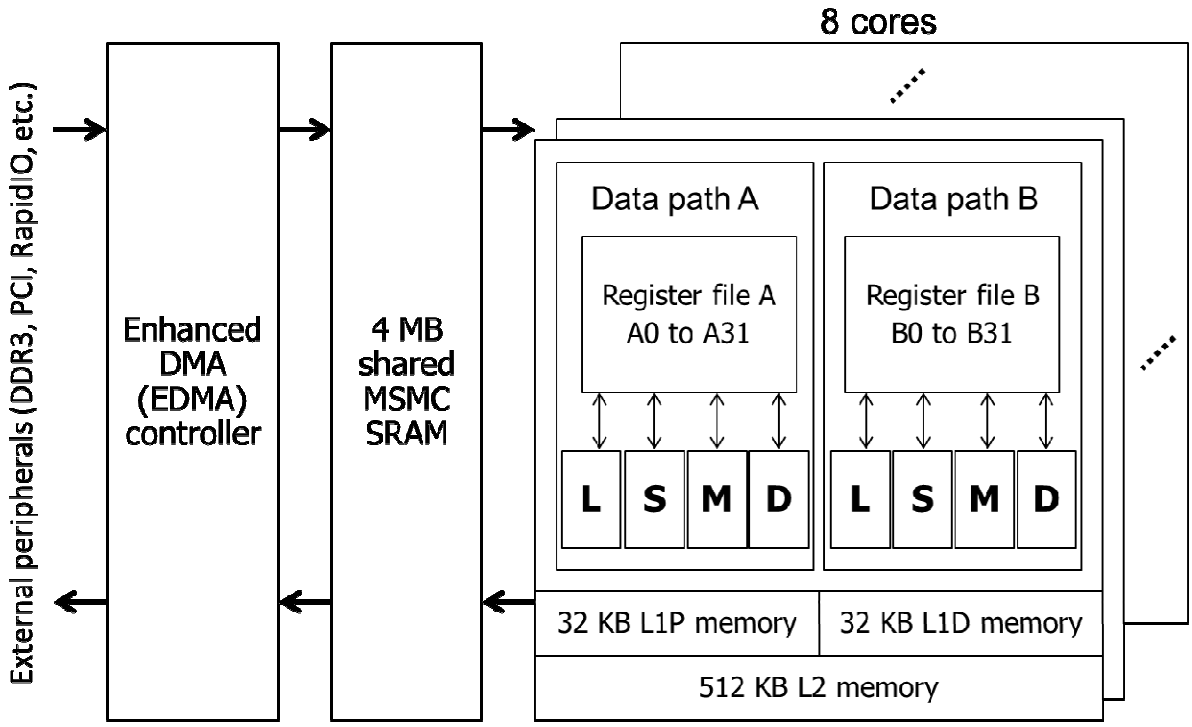


Figure 2.1 Simplified block diagram of Texas Instruments C6678.

In DSP C6678, the memory architecture consists of a two-level internal cache-based memory architecture and external memory, shown as Fig. 2.2.[14,15] Each core has 32-kbyte level-1 program (L1P) memory, 32-kbyte level-1 data (L1D) memory, and 512-kbyte local level-2 (L2) memory. L1P, L1D and L2 can be configured as cache and/or addressable SRAM. Level-1 memory can be accessed by the core without stalls. For each L1D and L1P, the size of cache is user-configurable, and can be set to 4k, 8k, 16k, or 32k bytes. In addition, the C6678 DSP has 4-Mbyte multi-core shared memory (MSM). MSMC SRAM can be configured as shared L2 and/or shared L3 memory.

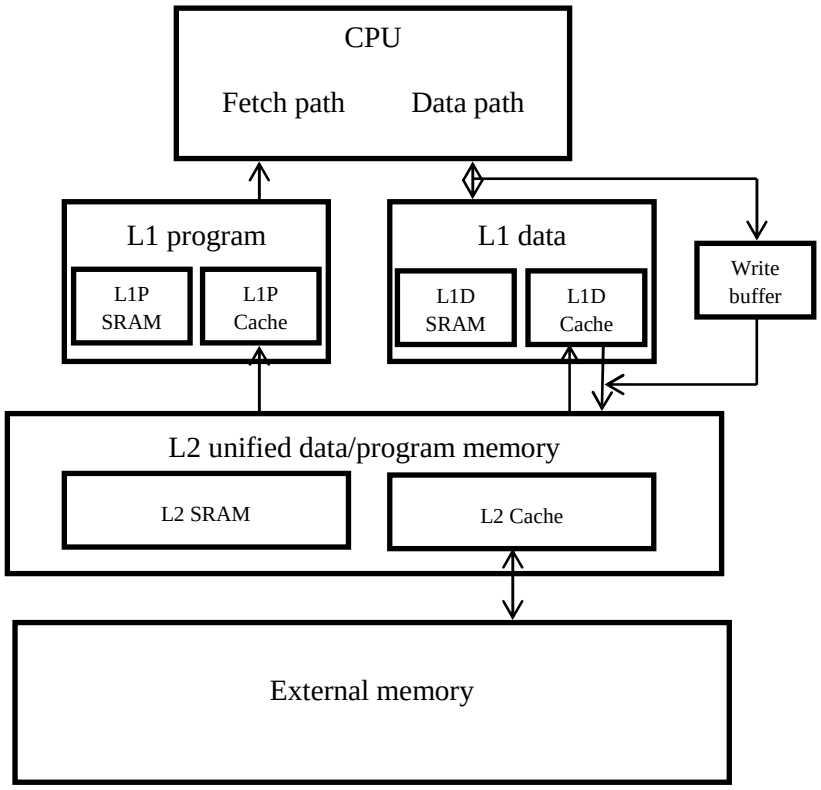


Figure 2.2 Cache memory architecture.

2.3 Internal direct memory access (IDMA) controller

In DSP C6678, each core has a programmable internal direct memory access controller that allows fast block transfer between any two local memory locations. Local memory locations include L1P, L1D, and L2 memories, or in the external peripheral configuration memory. Typically, IDMA is used to transfer between slower level-2 memory and faster level-1 memory. IDMA can provide lower latency because the transfers take place in the background, concurrently with DSP operations.[15,16]

Figure 2.3 illustrates the double-buffering technique using IDMA. The processor allocates four buffers, two for input blocks (ping_in_buffer and pong_in_buffer) and two for output blocks (ping_out_buffer and pong_out_buffer). While the core processor is processing data from the pong_in_buffer and stores the result in pong_out_buffer, the IDMA controller moves the previously computed output in the ping_out_buffer to L2 memory, and brings the

next input block from L2 memory to the ping_in_buffer. When the computation and data transfer are complete, the core processor switches to use the ping buffers and the IDMA controller works on the pong buffers.

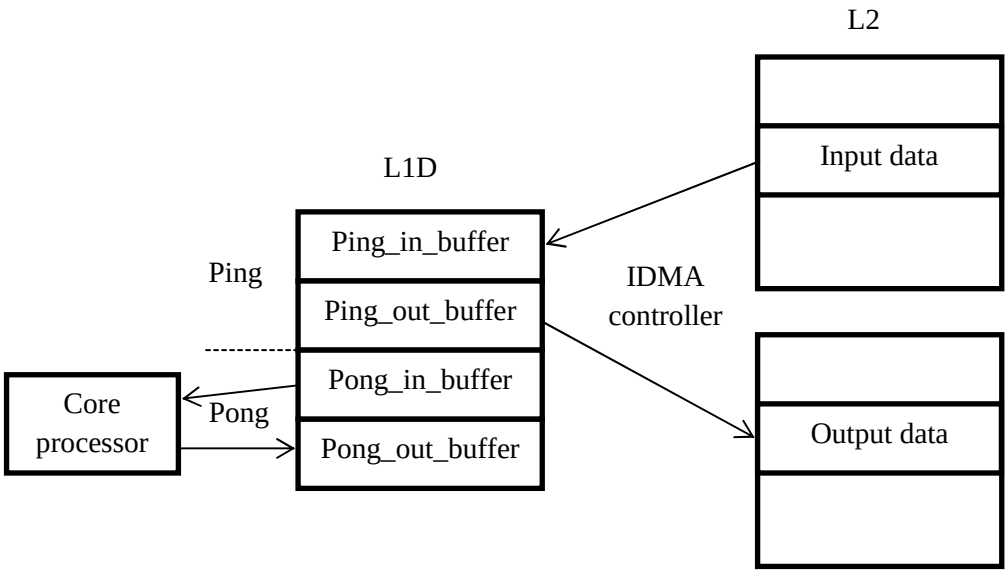


Figure 2.3 Double-buffering technique using IDMA. The IDMA controller loads and stores data using the ping buffers while CPU processes data using the pong buffers.

Chapter 3: Phase rotation beamforming

3.1 Phase error in the quadrature sampling as applied to ultrasound systems

Phase rotation beamforming is commonly used in ultrasound machines, which eliminates the error in phase to improve image quality. The phase error is introduced during quadrature demodulation, which is derived as follows.

The received RF signal at the *j*th element of the transducer array can be described as[4]

$$x_j(t-\tau_j) = A_{jI}(t-\tau_j)cos\omega_0(t-\tau_j) - A_{jQ}(t-\tau_j)sin\omega_0(t-\tau_j) \tag{3.1}$$

where A_{jI}, A_{jQ} are the in-phase and quadrature components of the signal received at the *j*th element, and τ_j is the delay time at the *j*th element. After quadrature demodulation, the signals are

$$x_j(t-\tau_j)cos\omega_r t = \frac{1}{2}A_{jI}(t-\tau_j)cos((\omega_0+\omega_r)t-\omega_0\tau_j) + \frac{1}{2}A_{jI}(t-\tau_j)cos((\omega_0-\omega_r)t-\omega_0\tau_j) - \frac{1}{2}A_{jQ}(t-\tau_j)\sin((\omega_0+\omega_r)t-\omega_0\tau_j) - \frac{1}{2}A_{jQ}(t-\tau_j)\sin((\omega_0-\omega_r)t-\omega_0\tau_j)$$

$$x_j(t-\tau_j)sin\omega_r t = \frac{1}{2}A_{jI}(t-\tau_j)sin((\omega_0+\omega_r)t-\omega_0\tau_j) - \frac{1}{2}A_{jI}(t-\tau_j)sin((\omega_0-\omega_r)t-\omega_0\tau_j) + \frac{1}{2}A_{jQ}(t-\tau_j)\cos((\omega_0+\omega_r)t-\omega_0\tau_j) - \frac{1}{2}A_{jQ}(t-\tau_j)\cos((\omega_0-\omega_r)t-\omega_0\tau_j) \tag{3.2}$$

where ω_r is the reference frequency used in demodulation.

The low-pass filter removes high frequency components, resulting in

$$\bar{I}_j = \frac{1}{2}A_{jI}(t-\tau_j)cos((\omega_0-\omega_r)t-\omega_0\tau_j) - \frac{1}{2}A_{jQ}(t-\tau_j)\sin((\omega_0-\omega_r)t-\omega_0\tau_j)$$

$$\bar{Q}_j = -\frac{1}{2}A_{jI}(t-\tau_j)sin((\omega_0-\omega_r)t-\omega_0\tau_j) - \frac{1}{2}A_{jQ}(t-\tau_j)\cos((\omega_0-\omega_r)t-\omega_0\tau_j) \tag{3.3}$$

To obtain the kth sample point for the center element, $t = (k+m_{jk})T_s$, where m_{jk} is the delay difference between the jth element and the center element for the kth sample point, and $T_s$is the sampling interval. Therefore,

$$\tilde{I}_{jk} = \frac{1}{2}A_{jI}(kT_s - (\tau_j - m_{jk}T_s))\cos((\omega_0 - \omega_r)kT_s - \omega_0(\tau_j - m_{jk}T_s) - \omega_r m_{jk}T_s)) - \frac{1}{2}A_{jQ}(kT_s - (\tau_j - m_{jk}T_s))\sin((\omega_0 - \omega_r)kT_s - \omega_0(\tau_j - m_{jk}T_s) - \omega_r m_{jk}T_s))$$

$$\tilde{Q}_{jk} = -\frac{1}{2}A_{jI}(kT_s - (\tau_j - m_{jk}T_s))sin((\omega_0 - \omega_r)kT_s - \omega_0(\tau_j - m_{jk}T_s) - \omega_r m_{jk}T_s)) - \frac{1}{2}A_{jQ}(kT_s - (\tau_j - m_{jk}T_s))\cos((\omega_0 - \omega_r)kT_s - \omega_0(\tau_j - m_{jk}T_s) - \omega_r m_{jk}T_s)) \quad (3.4)$$

An appropriate integer delay can be chosen such that $\tau_j - m_{jk}T_s \approx 0$.

If we define

$$B_{jI}(k) = \frac{1}{2}A_{jI}(kT_s - (\tau_j - m_{jk}T_s))$$

$$B_{jQ}(k) = \frac{1}{2}A_{jQ}(kT_s - (\tau_j - m_{jk}T_s))$$

$$\theta = (\omega_0 - \omega_r)kT_s - \omega_0(\tau_j - m_{jk}T_s) \approx (\omega_0 - \omega_r)kT_s$$

$$\Psi_{jk} = -\omega_r m_{jk}T_s \quad (3.5)$$

Then,

$$\tilde{I}_{jk} = B_{jI}(k)\cos(\theta + \Psi_{jk}) - B_{jQ}(k)\sin(\theta + \Psi_{jk})$$

$$\tilde{Q}_{jk} = -B_{jI}(k)sin(\theta + \Psi_{jk}) - B_{jQ}(k)cos(\theta + \Psi_{jk}) \quad (3.6)$$

Ψ_{jk} is the unwanted phase term. Phase rotation beamforming eliminates this error in phase to improve the image quality, which can be done as follows,

$$I_{jk} = \tilde{I}_{jk}\cos\Psi_{jk} - \tilde{Q}_{jk}\sin\Psi_{jk} = B_{jI}(k)\cos\theta - B_{jQ}(k)\sin\theta$$

$$Q_{jk} = \tilde{I}_{jk}\sin\Psi_{jk} + \tilde{Q}_{jk}\cos\Psi_{jk} = -B_{jI}(k)\sin\theta - B_{jQ}(k)cos\theta \quad (3.7)$$

3.2 PRBF block diagram

The block diagram of a digital phase rotation beamformer is shown in Fig. 3.1, which consists of mixing, lowpass filtering (LPF), decimation, integer delay, phase rotation and

summation operations. After analog echo signals are sampled by the ADCs, mixer and LPFs remove the carrier signal to produce the complex baseband data. After demodulation and decimation, the integer delay component selects a proper sample depending on the delay for each channel. Following that, phase rotation is performed to remove phase error. After that, appropriate samples from all the channels are summed together to form one focused sample. Figure 3.2 illustrates components in a phase rotator.

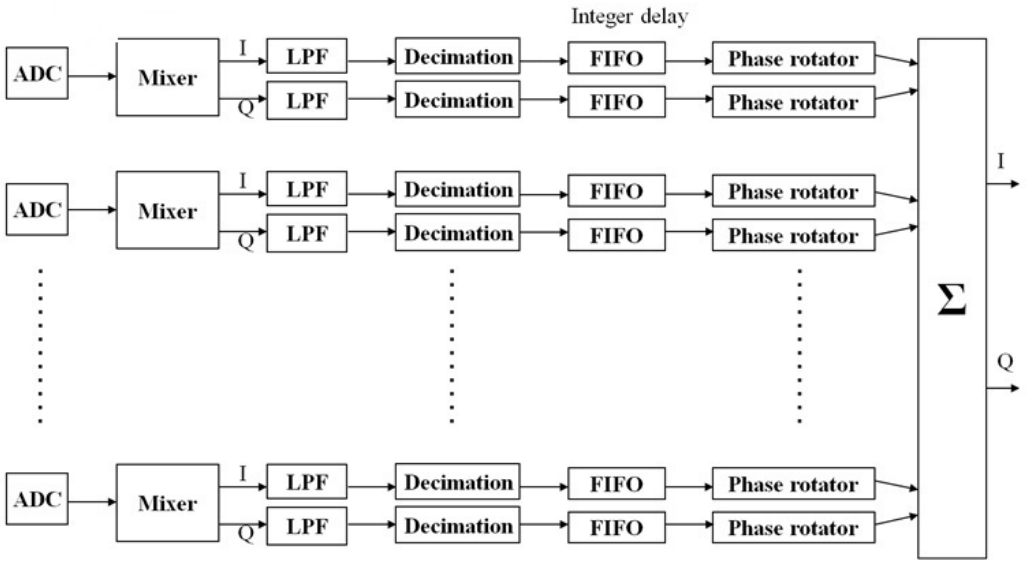


Figure 3.1 Block diagram of phase rotation beamforming.

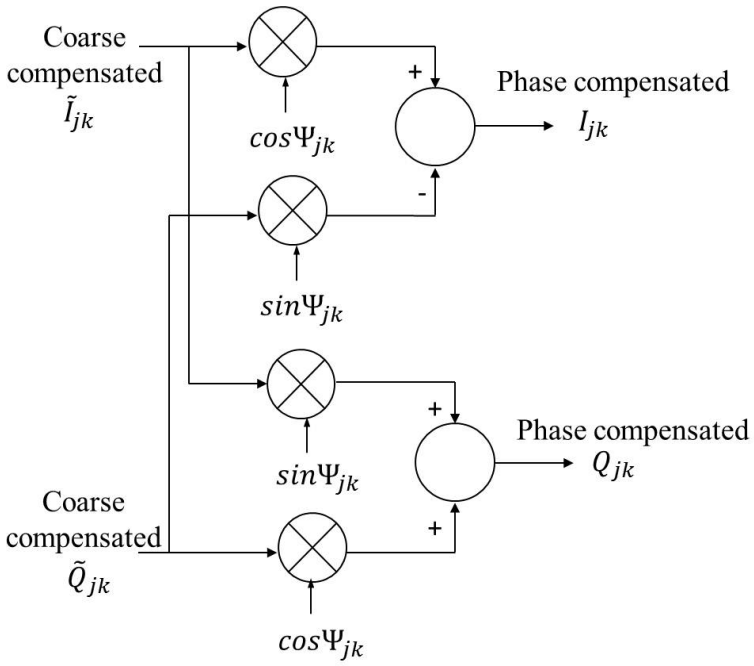


Figure 3.2 Block diagram of a phase rotator.

3.3 Computational requirements

Computational load for each stage in Fig. 3.1 (in terms of multiplications and additions) is listed in Table 3.1 where C is the number of channels and m is the demodulation filter tap size.[12] The percentage of operations needed by each stage to the total number of operations was calculated when the number of channels is 64 and the filter tap size is 32. The table indicates that the demodulation filtering is the most computationally expensive function, taking 91% and 95% of the total multiplications and additions in the PRBF, respectively. Filtering in demodulation is computationally demanding since it is applied to each one of the pre-beamformed channels.

Table 3.1 Computational load for each stage of PRBF and its percentage to the total number of operations when the number of channels (C) is 64 and the demodulation filter tap size (m) is 32

	Multiplications		Additions	
	Number		Number	
Mixing	2C	3%	0	0%
Demodulation filter	2mC	91%	2(m-1)C	95%
Phase rotator	4C	6%	2C	3%
Sum	0	0%	C-1	2%
Total	(2m+6)C	100%	(2m+1)C-1	100%

Chapter 4: Software-based phase rotation beamforming

4.1 New PRBF structure with integrated analog front-end

Demodulation filtering in PRBF is the most computationally expensive function. To alleviate this high computing requirement, the AFE chips integrating ADC and quadrature demodulation were used. By utilizing such AFE chips, only delay alignment and phase rotation need to be performed by DSP, substantially reducing the computational load.

The flow diagram of our implementation of the PRBF algorithm is shown in Fig. 4.1. For our implementation in this thesis, we assume that demodulation and decimation have been performed in the new AFE chips. Complex baseband data (i.e., in-phase (I) and quadrature (Q) components) from AFEs are passed onto delay modules, which select appropriate samples for each channel based on the delay determined by the distance between the target and the transducer element. The delay module is followed by a phase rotator, which compensates the phase shifting in I and Q signals so that they are properly aligned for the summation stage. Then, a summation across all the channels is performed to produce beamformed I and Q signals for back-end processing.

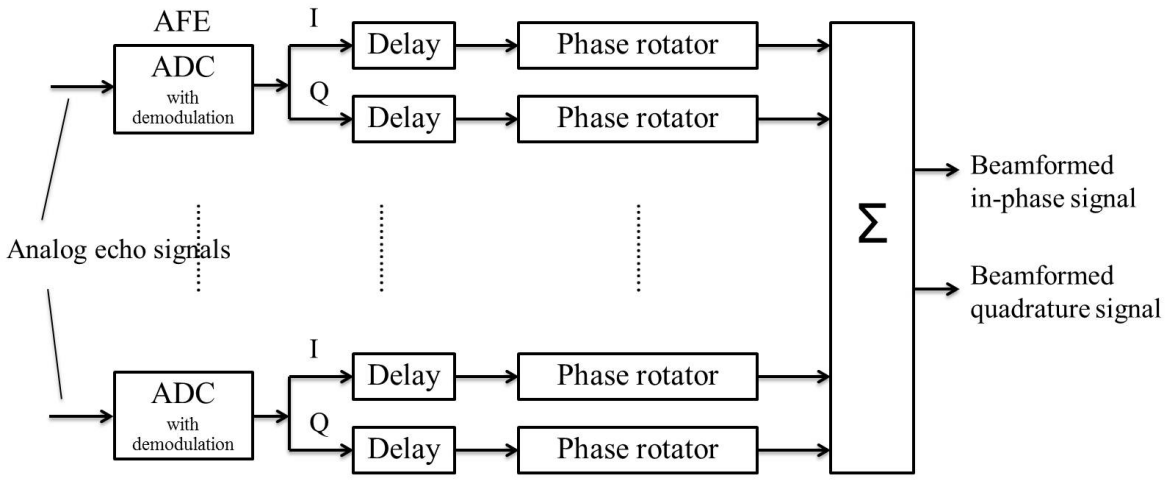


Figure 4.1 Block diagram of a phase rotation beamformer assuming quadrature demodulation is performed in the AFE chips.

4.2 Mapping of PRBF algorithm on TMS320C6678 DSP

The phase rotation beamforming algorithm described above was implemented on C6678. To save the computational power, delay and phase rotation parameters were pre-computed and stored in lookup tables (LUT). To reduce the LUT size and cache misses due to loading the LUT, all these parameters were combined into one LUT by using differential delay addressing, shown in Fig. 4.2. The first entry in the LUT contains the absolute address for the first sample. Starting from the second entry, sine and cosine values for phase rotation and a differential delay address are combined. The least significant bits of two 16-bit fields are extracted to form a 2-bit differential delay address, while the upper 15 bits are used for a phase rotation parameter. Once the LUT is loaded, sine and cosine values for phase rotation can be extracted, and the absolute address for each sample can be calculated.

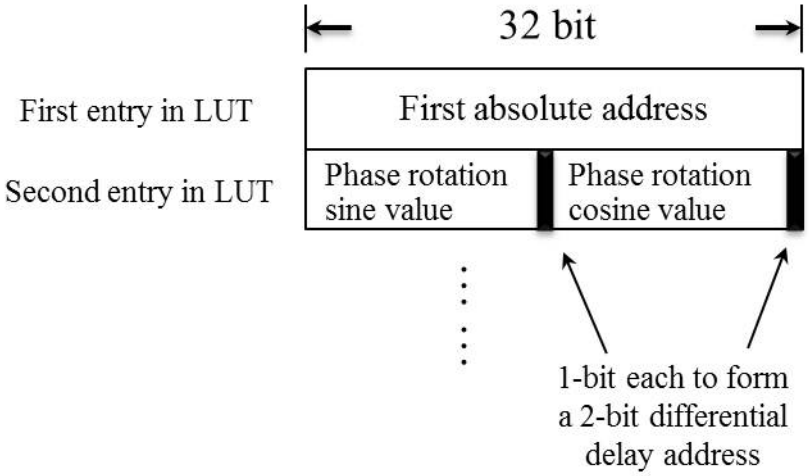


Figure 4.2 Structure of LUT by using differential delay addressing.

Figure 4.3 shows the pseudo-code for the PRBF tight loop for 64 channels and 2048 samples per scanline. After initializing the partial sum array to zero, the tight loop computes the absolute address and load properly-delayed demodulated IQ data, and then performs phase rotation to eliminate the error in phase, and accumulates it into the partial sum array. This occurs for every sample in each channel (n=0 to 2047), and then for every channel (m=0 to 63). Once all 64 channels have been processed, one focused scanline is generated.

```
for(n=0; n<2048; n++)
{
        partial_sum[n] = 0; // initialize the partial sum array to zero
}

for(m=0; m<64; m++) // for each channel m
{
        load first_index;
        previous_index = first_index;
        for(n=0; n<2048; n++) // for each output sample n in channel m
        {
                load LUT_entry[m,n];
                extract two msb15(LUT_entry[m,n]) => phase_rotation_parameters;
                extract two lsb(LUT_entry[m,n]) => differential_delay_index;
                absolute_index = differential_delay_index + previous_index;
                previous_index = absolute_index;

                load demodulated_IQ[m][absolute_index] => input_samples;
                intermediateIQ = complex_multiplication(input_samples, phase_rotation_parameters);
                partial_sum[n] = partial_sum[n] + intermediateIQ;
        }
}

for(n=0; n<2048; n++)
{
        output[n] = partial_sum[n];
}
```

Figure 4.3 Pseudo-code for the PRBF tight loop for 64 channels and 2048 samples per scanline.

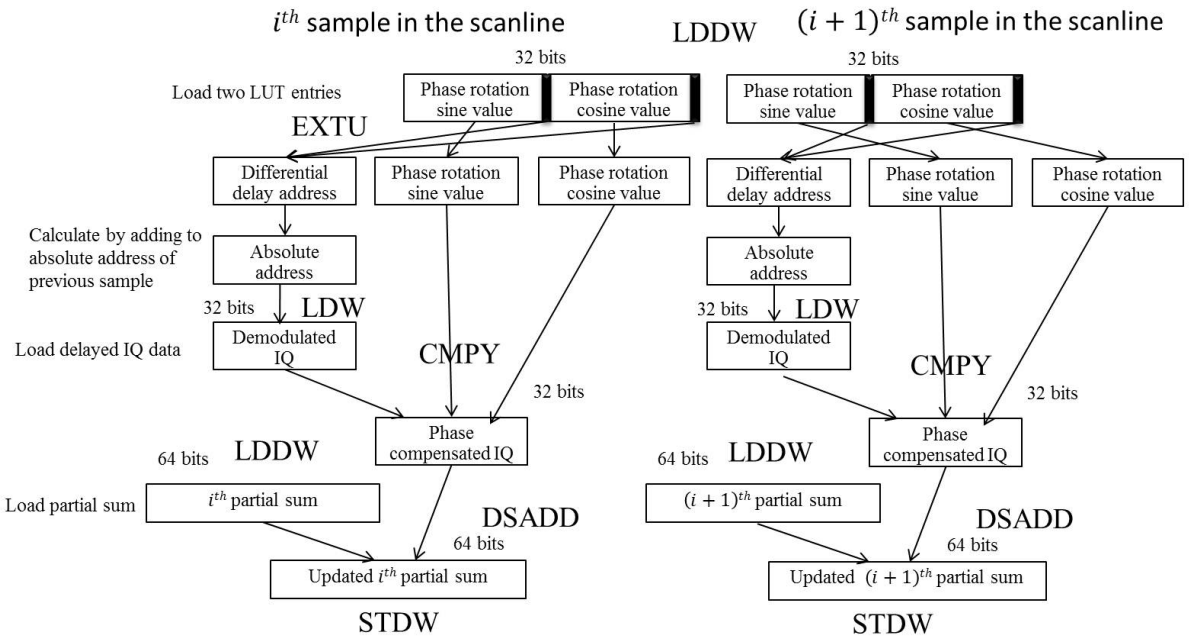


Figure 4.4 Mapping of PRBF on C6678 using C6678 instruction set.

Figure 4.4 shows how the tight loop of PRBF was mapped on C6678 using the C6678 instruction set.[17] The beamforming is performed in the axial direction. Two LUT entries for two consecutive samples in the axial direction are loaded, and each contains a differential delay address and phase rotation parameters. The absolute address can be calculated by adding the differential delay address with the absolute address of the previous sample. The first absolute address is contained in the LUT. With the calculated absolute address, the appropriate sample is loaded. Phase compensation is performed by using the CMPY instruction. Then, the partial sum for the current sample is loaded and updated.

4.3 Data flow optimization

To reduce the penalty caused by cache misses in loading LUT, the double-buffering technique and IDMA are used to transfer LUT data while CPU is computing with previously-loaded data. Part of the L1D memory is configured as SRAM with a half for the ping buffers and another half for the pong buffers. While the core processor is working on delay and phase compensation by using the data from the pong buffer, the IDMA controller brings the

next LUT block from local L2 memory to the ping buffer. When the computation and data transfer are complete, the core processor switches to use the ping buffer and the IDMA controller works on the pong buffer.

4.4 Performance

We programmed the PRBF algorithm in the code composer studio and measured the computation time using the cycle approximate simulator. Table 4.1 shows the ideal performance and compiler output of our implementation. The compiler output is the same as the ideal performance of 1.75 cycles/IQ output. Base on this performance, assuming a sampling rate of 40 MHz, 2:1 decimation and a clock frequency of 1 GHz, it takes 229.4 µs to beamform one scanline for 64 channels and 2048 samples/scanline (2048 beamformed samples × 1.75 cycles × 64 channels × 1 ns/cycle) on a single core, which means 4.4k scanlines per second can be beamformed.

As the tight loop needs to load the demodulated IQ data (4 bytes/sample), LUT (4 bytes/sample) and partial sum values (8 bytes/sample), a large number of cache misses would be expected, which could cause severe performance degradation. To reduce the number of cache misses, we configured L1D as 8 kbytes of data cache and 24 kbytes of SRAM. 24-kbyte SRAM was divided into two parts; 16-kbyte SRAM to keep a partial sum array and 8-kbyte SRAM for IDMA double-buffering. In addition, for 64 channels and 2048 samples/scanline, the size of LUT is 512 kbytes. It is not feasible to keep the whole LUT in local L2 since the size of local L2 memory is 512 kbytes, which needs to be used for other purposes. To fit the LUT into local L2 so that it can be transferred independently by IDMA, we used each core to process the data for 32 channels. The data flow on each core is shown in Fig. 4.5. Therefore, 2 cores are needed to beamform one scanline for 64 channels and 2048 samples/scanline. With this configuration, the total CPU cycles for the tight loop from the cycle approximate simulator is 174,417 cycles, corresponding to 174 µs to beamform one scanline with 2 cores at the clock frequency of 1 GHz, shown in Table 4.2.

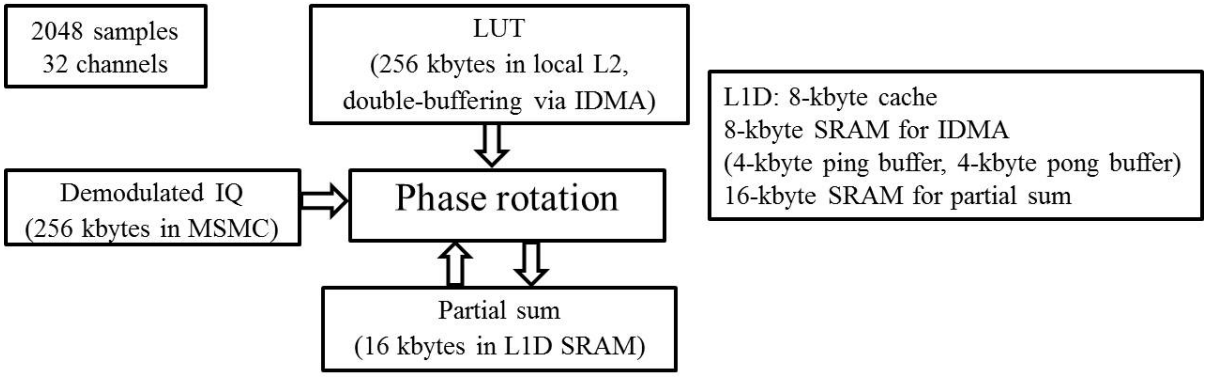


Figure 4.5 Data flow of PRBF with 32 channels and 2048 samples/scanline on one core.

Table 4.1 Ideal performance and compiler output

	Ideal performance	Compiler output
Performance (cycles per IQ output)	1.75	1.75

Table 4.2 Cycle approximate simulator profiling output and multi-core performance assuming 1-GHz clock frequency

	Profiling output	Multi-core performance (including multi-core partitioning overhead)
Performance (per scanline with 2 cores)	174 µs	200 µs

After one core finishes the processing, the half partial sum results are kept in MSMC, which would be loaded by the other core that works on the second 32-channel data to produce the final results. Taking this partitioning overhead and communication overhead among cores into account, it takes 200 µs to generate one scanline (2048 samples/scanline) with two cores at the clock frequency of 1 GHz, shown in Table 4.2. Therefore, with 4 cores, it can support beamforming of 64 channels with 10k scanlines/s, e.g., 200 scanlines/frame at 50 frames/s. If we increase the clock frequency to 1.25 GHz, 12.5k scanlines can be beamformed per second with 4 cores. The remaining 4 cores can work on other tasks. One

possibility is to run back-end processing and applications, e.g., color Doppler or ultrasound elastography.

Figure 4.6 gives an example timing diagram with 4 cores, assuming one scanline is acquired every 100 µs. When the first 32-channel data are available in MSMC, core 0 at 50 µs starts processing the first 32 channels for scanline 0. 50 µs later, core 1 starts beamforming the second 32-channel data for scanline 0. The scanline 0 is generated at 300 µs. After that, each scanline is completed every 100 µs. Figure 4.7 shows the simplified high-level architecture of the front-end. The ADC and demodulation are performed in AFEs, and then I and Q data are streamed into the DSP.

To show the beamforming quality, a phantom study was performed. A phantom (Model RMI 403GS LE Multipurpose Phantom, Gammex Inc., Middleton, WI) was used. An ultrasound machine (Verasonics Research System, Verasonics, Inc., Redmond, WA) was used to acquire pre-beamformed data. An ATL L7-4 linear array transducer with a center frequency of 5 MHz and a pitch of 0.298 mm was used. Figure 4.8(a) shows the phantom image beamformed using our implementation with a dynamic range of 55 dB. For comparison, we also implemented PRBF by using floating-point Matlab (The MathWorks Inc., Natick, MA). The beamformed image using Matlab is shown in Fig. 4.8(b). To evaluate the image quality, the root mean square error (RMSE) between two methods was calculated as follows,

$$RMSE(x,y) = \sqrt{\frac{\sum_i \sum_j |x(i,j) - y(i,j)|^2}{N}} \tag{4.1}$$

where x, y represent an 8-bit pixel value in B-mode images generated by our implementation and by Matlab, respectively, i, j are pixel indices, N is the number of pixels. The result is 1.69. Besides, the normalized magnitudes around the bright dot are shown in Fig. 4.9. The difference between the two methods is insignificant. Thus, there is no noticeable image quality degradation with our implementation.

4.5 Discussion

These results were obtained by using the cycle approximate simulator since we did not have access to the hardware for this study. Besides, the overhead for streaming the demodulated IQ data via Hyperlink into the MSMC SRAM of C6678 has not been included.

In this implementation, we did not take advantage of the symmetry in the LUT. When there is no beam steering, delay values are symmetric around the middle of the aperture, as shown in Fig. 1.2. 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. Therefore, phase rotation parameters and differential delay address are symmetric. Based on this observation, the LUT size could be reduced in half. In this case, splitting channels across cores is not necessary. However, since beam steering is used in modern ultrasound systems (e.g., for oblique sectors or spatial compounding), we used the original LUT size to keep the flexibility of adding beam steering in the future.

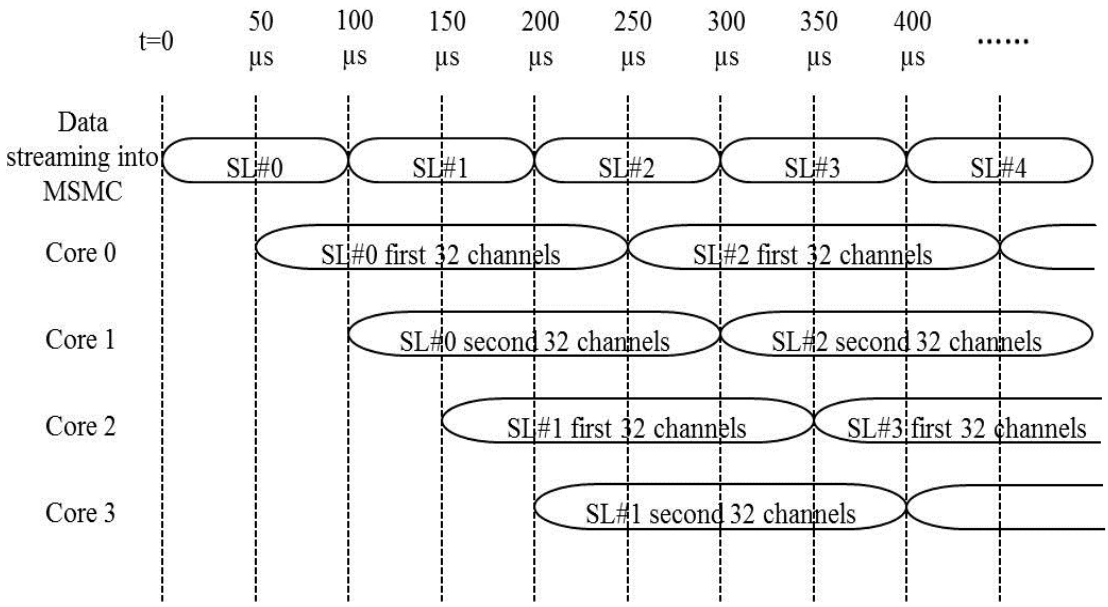


Figure 4.6 Example timing diagram with 4 cores, assuming one scanline is acquired every 100 µs and it takes 200 µs for each core to process 32-channel data.

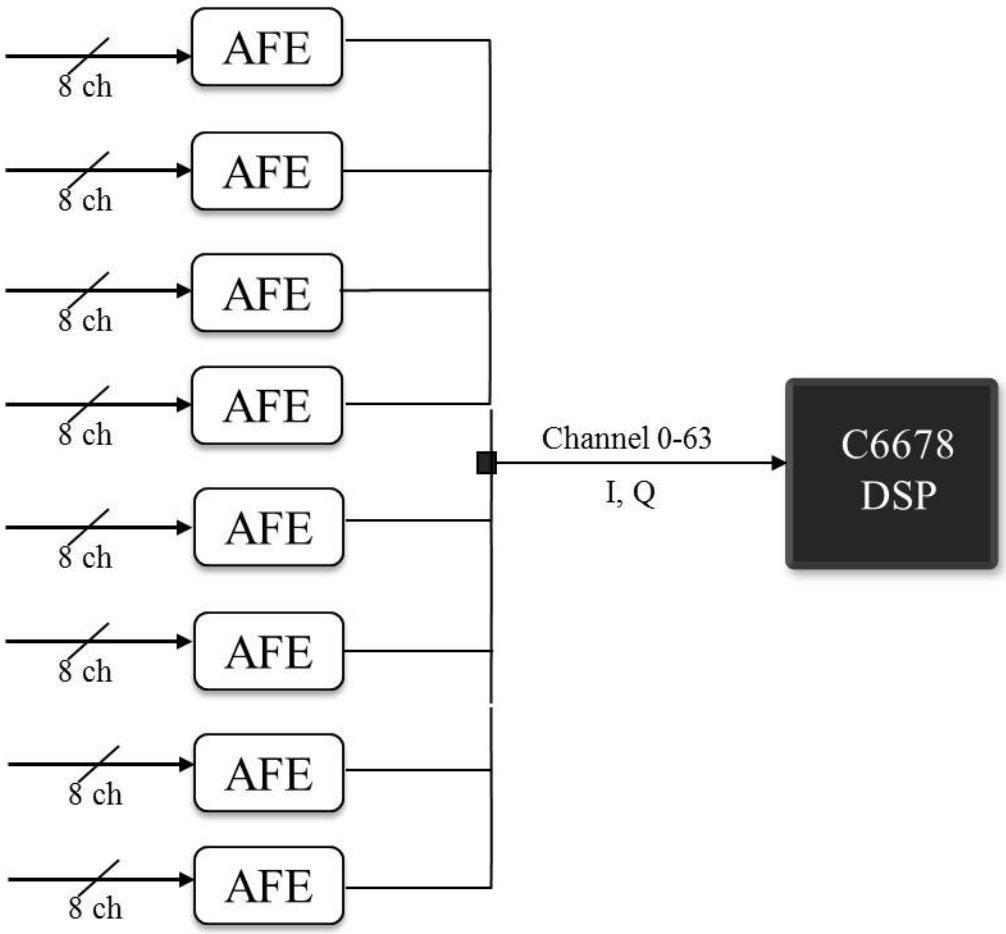
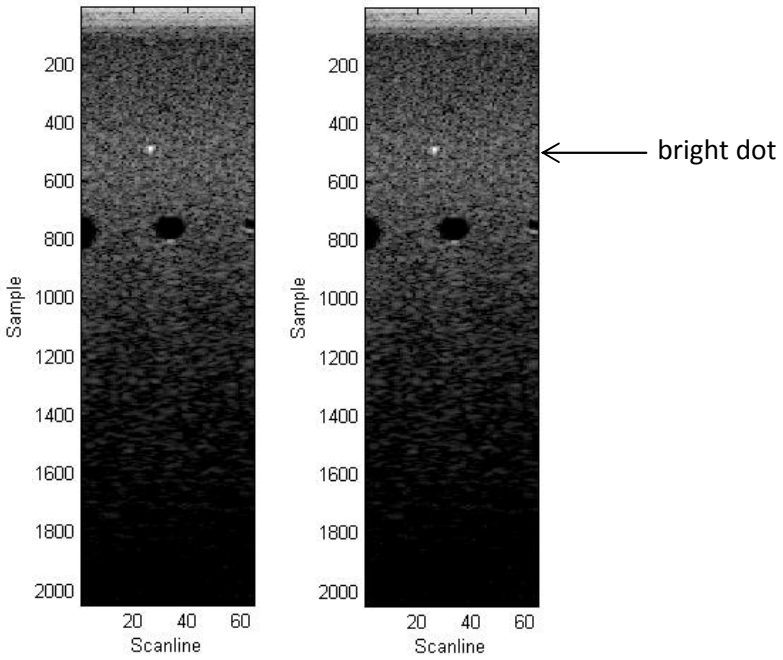
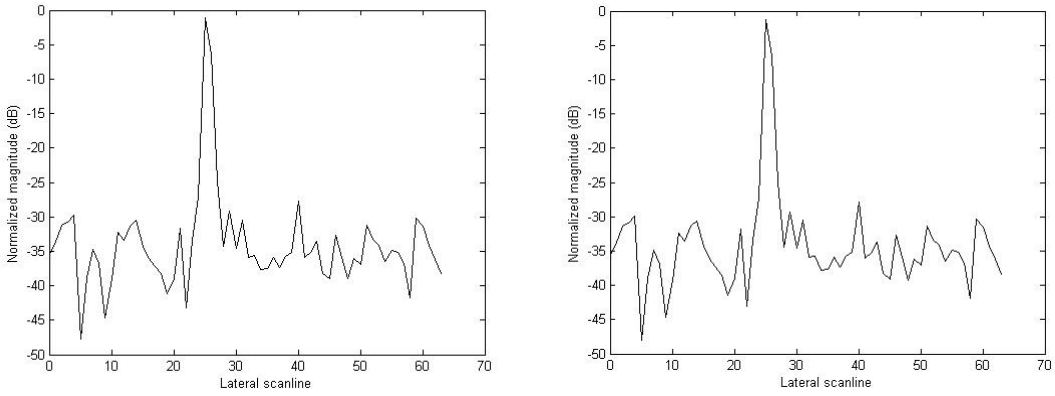


Figure 4.7 Simplified high-level block diagram of the front-end. The ADC and demodulation are performed in AFEs, and then I and Q data are streamed into the DSP.

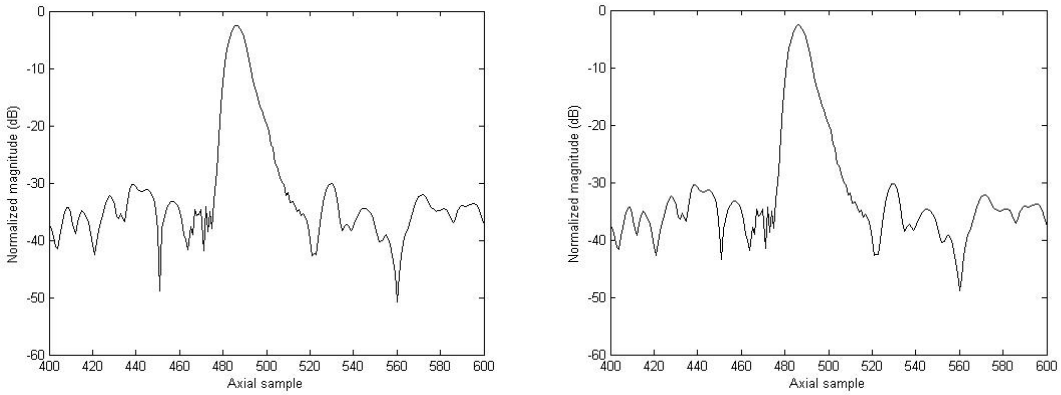


(a) (b)

Figure 4.8 Phantom study result of PRBF with 55 dB dynamic range. Beamformed ultrasound image with (a) our implementation on C6678 and (b) Matlab. The bright dot is pointed out by the arrow.



(a) (b)



(c) (d)

Figure 4.9 Normalized magnitude around the bright dot. (a) Lateral and (c) axial profiles for beamformed data with our implementation on C6678. (b) Lateral and (d) axial profiles for the beamformed data from Matlab.

Chapter 5: Application scenarios

5.1 Clinical application examples

The beamforming performance enabled by this architecture based on new AFEs and DSP can support various clinical applications, as listed in Table 5.1. One example is neck imaging, where the carotid arteries are examined for plaques and blood flow abnormalities. With the AFE chip performing digitization at 40 MHz, demodulation and 2:1 decimation, there are 1299 samples per scanline assuming an imaging depth of 5 cm (40 MHz × 10 cm (round-trip distance) / 1540 m/s (speed of sound in tissue) / 2 (2:1 decimation)). Based on the physical limitation, the maximum number of scanlines that can be produced is 15.4k, which can be supported by our implementation since we can beamform up to 15.7k scanlines/s with 4 cores (10k scanlines/s × 2048 samples/scanline / 1299 samples/scanline). If 308 scanlines are acquired per frame, a frame rate of 50 Hz can be achieved, which is typically sufficient for imaging the carotid artery.

Another example is abdominal imaging, where abdominal organs (e.g., liver and aorta) are examined. In this case, larger imaging depths are required. Assuming AFE chips performing 40-MHz digitization, demodulation, 2:1 decimation and an imaging depth of 20 cm, there will be 5,195 samples per scanline. The maximum number of scanlines that can be acquired at this imaging depth is 3.8k. If 200 scanlines are acquired per frame, a frame rate of 19 frames/s can be achieved. Since abdominal organs do not move fast, this frame rate is typically adequate.

Table 5.1 Example clinical scenarios

	Depth (cm)	Typical frame rate (frames/s)	Number of samples /scanline (40-MHz ADC and 2:1 decimation)	Transducer type
Neck	5	30-50	1,299	Linear
Kidney	12	15-30	3,117	Convex
Liver/Aorta/OB	20	15-30	5,195	Convex

5.2 Multi-core DSP scenarios

As the C6678 DSP has 8 cores, there are 4 spare cores that can be used for additional processing. One possibility is to use the remaining 4 cores for back-end processing. Therefore, one DSP can support both front-end beamforming and back-end processing. Typically, back-end processing, such as B-mode, color Doppler and spectral Doppler, only needs one or two cores. Thus, more sophisticated algorithms might be supported with 4 cores, e.g., ultrasound elastography or speed of sound correction. Also, if all 8-core DSP was used for beamforming, it could support 128 channels with the beamforming throughput of 10k scanlines/s or 64-channel full-rate dual receive beamforming with the throughput of 20k scanlines/s (2048 samples/scanline). With an increased number of cores in the future, the number of scanlines that can be supported per second could increase further, and/or more advanced front and back-end algorithms could be supported in real time.

In addition, this software-based approach could allow reusability and scalability in ultrasound systems. Based on the performance we obtained, we can estimate that PRBF for 64 channels with 10k scanlines/s and 1024 samples/scanline (40-MHz digitization with 4:1 decimation) can be supported on C6678 with 2 cores. Therefore, one 4-core DSP might be sufficient to support both front-end beamforming and back-end processing, which would reduce the cost and power consumption even further. On the other hand, two DSPs might be able to support 256-channel beamforming for high-end ultrasound machine.

Chapter 6: Conclusions

In this thesis, we investigated the feasibility of using one multi-core DSP to support PRBF. Our results show that, when quadrature demodulation is performed in the AFE chips and the AFE output data can be fed to C6678, beamforming for 64 channels with 10k scanlines/s and 2048 samples/scanline can be supported on C6678 with 4 cores assuming a clock frequency of 1 GHz. Therefore, an 8-core DSP can support both front-end and back-end processing, which could benefit portable ultrasound machines. In the future, with higher clock frequency and additional cores, this architecture should be in position to support high-end systems with more channels.

This software-based architecture could be advantageous in achieving shorter time-to-market (e.g., compared to designing the beamformer ASIC chip and making it work), flexibility (different algorithms can be supported) and scalability. This new architecture could lead to low-power and low-cost ultrasound machines, benefiting ultrasound imaging in general, but particularly portable ultrasound machines.

Bibliography

1. *Signal Processing Overview of Ultrasound Systems for Medical Imaging*. Texas Instruments, Literature Number SPRAB12, November 2008.

2. K. E. Thomenius, “Evolution of ultrasound beamformers,” *Ultrasonics Symposium*, vol. 2, pp. 1615-1622, 1996.

3. B. D. Steinberg, "Digital beamforming in ultrasound," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 39, pp. 716-721, 1992.

4. S. H. Chang, S. B. Park and G. H. Cho, “Phase-error-free quadrature sampling technique in the ultrasonic B-scan imaging system and its application to the synthetic focusing system,” *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 40, pp. 216-223, 1993.

5. L. Pelissier, “Ultrasound imaging system,” U.S. Patent 6 558 326, 2002.

6. S. Sikdar, R. Managuli, L. Gong, V. Shamdasani, T. Mitake, T. Hayashi, and Y. Kim, “A single mediaprocessor-based programmable ultrasound system,” *IEEE Transactions on Information Technology in Biomedicine*, vol. 7, pp. 64–70, 2003.

7. V. Shamdasani, R. Managuli, S. Sikdar and Y. Kim, “Ultrasound color-flow imaging on a programmable system,” *IEEE Transactions on Information Technology in Biomedicine*, vol. 8, pp. 191-199, 2004.

8. C. Lee, H. Sohn, D. Han and T. Song, “Real-time implementation of the echo signal processing and digital scan conversion for medical ultrasound imaging with a single TMS320C6416 DSP,” *Proceedings of SPIE*, vol. 6920, pp. 692004-692004-9, 2008.

9. H. Sohn, S. Seo, J. Kim and T. Song, “Software implementation of ultrasound beamforming using ADSP-TS201 DSPs,” *Proceedings of SPIE*, vol. 6920, pp. 69200Z-69200Z-11, 2008.

10. P. Chen, M. Butts and B. Budlong, "Medical ultrasound digital beamforming on a massively parallel processing array platform," *Proceedings of SPIE*, vol. 6920, pp. 692003-692003-9, 2008.

11. Available at http://www.verasonics.com/pdf/verasonics_ultrasound_eng.pdf.

12. F. K. Schneider, Y. M. Yoo, A. Agarwal, L. M. Koh and Y. Kim, “New demodulation filter in digital phase rotation beamforming,” *Ultrasonics*, vol. 44, pp. 265-271, 2006.

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

14. *TMS320C66x DSP Cache User Guide*. Texas Instruments, Literature Number SPRUGY8, November 2010.

15. *TMS320C66x DSP CorePac User Guide*. Texas Instruments, Literature Number SPRUGW0A, November 2010.

16. 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.

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

VITA

Jieming Ma is a graduate student in the Department of Electrical Engineering at the University of Washington where she is currently pursuing a Ph.D. degree. She was born in Shanghai where she earned a BS degree in Electronics Engineering and an MS degree in Information Science and Engineering from Fudan University. She was working as a software engineer at EMC^2 China Research & Development Center for two years. Her research is focused on digital beamforming for ultrasound imaging, biomedical signal and image processing, and machine learning.