Massively Parallel Processor Array for Mid-/Back-end Ultrasound Signal Processing

Dean N. Truong
Dept. of Electrical and Computer Engineering
University of California, Davis
hottruong@ucdavis.edu

Bevan M. Baas
Dept. of Electrical and Computer Engineering
University of California, Davis
bbaas@ucdavis.edu

Abstract— **Ultrasound remains a popular imaging modality due to its mobility and cost-effectiveness. As general purpose computing and DSPs are entering an era of multi-core architectures, the potential for parallel performance gains are significant when used properly. This work explores the possibility of using a massively parallel processor array to meet real-time throughputs for mid-/back-end ultrasound processing. A many-core array of simple DSP cores, shared memories, and an FFT processor is shown to dissipate 87.79 mW for B-mode, 33.20 mW for color flow, and 29.24 mW for spectral doppler, while achieving a frame rate of 37.6 fps for B-mode and 12.5 fps for color flow.**

I. Introduction

The clinical use of ultrasound is increasingly commonplace with a given estimate of 250 Million exams done worldwide—over three times greater than the next often used imaging modality (X-ray CT) [1]. The choices of transducers and digital processing algorithms, as well as a wider range of design requirements such as adaptive beamforming, automated 3D real-time ultrasound and enhanced mobility, create a design environment for a wider array of possibilities. Thus, future ultrasound systems will have reprogrammable and reconfigurable electronics at all levels including the analog front end, beamforming, image processing, and scan conversion. To meet these challenges multi-processor DSPs have been touted as a viable solution to achieve real-time multimodal ultrasound with the capability of being programmable [2], [3]. However, next generation computation will be guided by many-core chip architectures and more integrated electronics, which creates the foundation to push what is often thought of as research into the hands of clinicians.

The potential of many-core parallelism in ultrasound was demonstrated using Ambric, Inc.'s massively parallel processor array (MPPA) with ultrasound beamforming [4]. Digital beamformers are typically implemented on FPGAs to meet both reconfiguration and processing requirements, but lack software programmability. Ambric's MIMD architecture provided the flexibility and it's 336 DSP core 2D mesh let the application scale to most any typically sized transducer array. A 64-channel test platform was built using Ambric's solution [5].

Academic research has also delved into the many-core question early on with MIT's 16-core RAW [6], the polymorphic TRIPs architecture from UT Austin [7], and the 36-core AsAP from UC Davis [8]. The second generation AsAP chip (**As**ynchronous **A**rray of simple **P**rocessors) is a 167-core computational platform fabricated in 65 nm. In addition to its 164 homogeneous DSP processors, it includes three shared memory blocks and three dedicated purpose processors for process heavy tasks such as video motion estimation, Viterbi decoding, and FFT [9]. To the best of our knowledge, these research chips have not attempted to tackle biomedical imaging.

In this paper we will describe a B-mode and color flow application implemented on a many-core architecture based on the 167-core computational platform. Like many other DSP applications, ultrasound signal and image processing kernels tend to have small instruction memory footprints and are naturally task parallel. Because the platform is globally asynchronous and locally synchronous (GALS), and can support dynamic voltage and frequency scaling (DVFS), per-core voltage and frequency selection will be used to save power and energy.

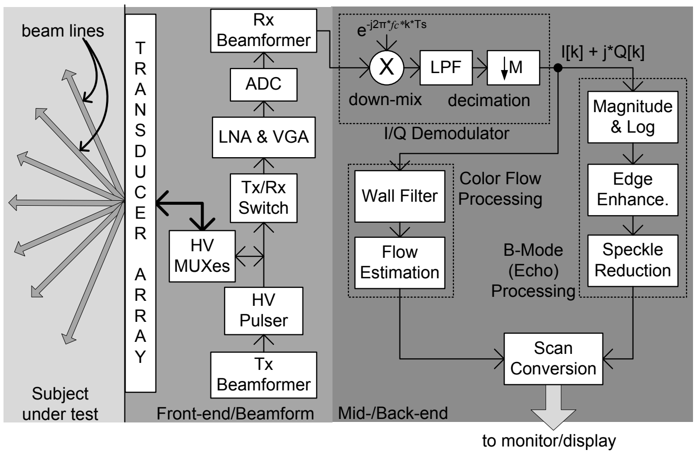

Fig. 1. Ultrasound system block diagram.

II. Overview

A. System Architecture

An ultrasound system generally consists of a transducer, beamformer, mid-/back-end processing, and scan converter. The transducer transmits the ultrasonic waves and then waits to receive the echoes (i.e. the transceiver); the beamformer controls the direction of the transmitting wave and the 'listening' direction of received echoes; the mid-/back-end processing converts the raw echo signals into anatomical and physiological data, and the scan converter maps the data to the target video/image display. This system architecture is shown in Fig. 1.

The analog front end consists of high-voltage muxes and ultrasound pulsers, as well as transmit/receive switches and low noise amplifiers (LNAs), variable gain amplifiers (VGAs), and high speed ADCs [3], [10]. The digital beamformer can be built with FPGAs and/or Ambric's MPPA, the scan conversion is done using a graphics processor, while the mid-/back-end processing is done using our proposed many-core architecture. Additionally, a control processor can be used to reconfigure control signals and LUTs to every system block.

B. Chip Architecture

The key requirements for ultrasound signal and image processing centers around RF demodulation, 1D and 2D filtering, and Doppler-based imaging, which requires autocorrelation algorithms and the

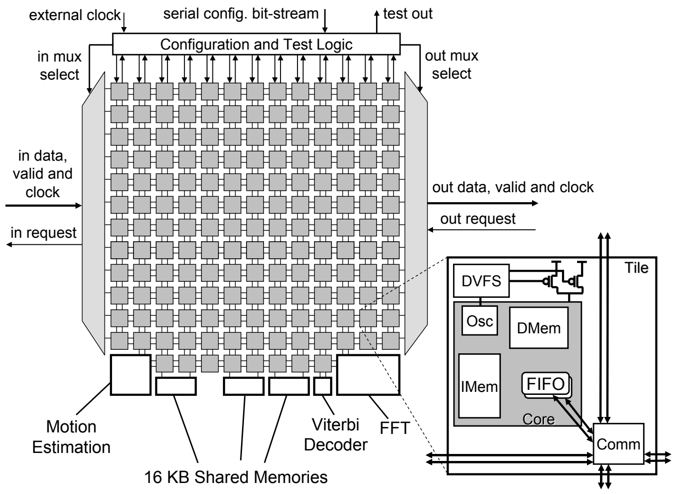

Fig. 2. Block diagram of the 167-processor computational array.

FFT. The basic chip architecture is shown in Fig. 2. The kernels are mapped onto the array of simple DSP cores, each core consisting of a 128-word local instruction memory (IMem), and 128-word by 16-bit local data memory (DMem). In order to facilitate cross-frequency domain communication, two 64-entry FIFOs are provided. In addition, 16 KB shared memory modules are used to temporarily store processed data (usually around several beam lines—spatially and/or temporally) when the filtering becomes 2D.

Once the receive beamformer forms a data point that sample is sent to the chip. Thus, the I/O can be a simple parallel bus of 16-bits since the typical dynamic range acceptable for ultrasound (that is achievable by an ADC) is around 12-bits. Given a modest I/O speed of 250 MHz (for single ended CMOS), the ideal number of bytes per second at the I/O is 500 MB/sec. Typically the sampling is chosen to be at least four times the carrier to preserve echo information [5]. Since the normal diagnostic range for the ultrasound carrier frequency is between 1 to 20 MHz, the I/O bandwidth will be between 4 to 80 MSamples/sec, accordingly (assuming 4× sampling rate). Thus, the worst case bytes per second will be 120 MB/sec with a 12-bit data word, which is more than doable with our modest I/O. The output bandwidth will be lower because of decimation in the RF demodulation step on the initial beamformed data stream.

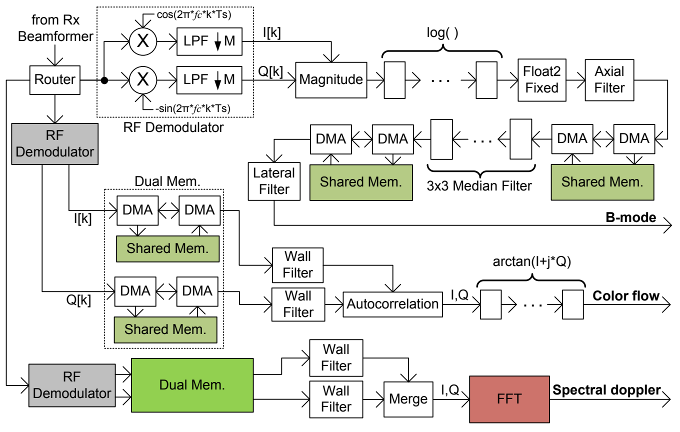

Fig. 3. Mid-/Back-end implementation block diagram.

C. Mid-/Back-end Processing

Figure 3 shows the main blocks and subblocks in mid-/back-end ultrasound processing. Each of the algorithms will be described below.

1) RF Demodulation: The first basic block before any further processing is the RF demodulation, which mixes the received beam-formed sample with the carrier frequency (the original ultrasound transducer transmitted frequency) to extract the real and imaginary, or I and Q components. Low pass filtering (LPF) and decimation is done to eliminate the negative frequency spectra and to reduce the number of samples, respectively. The polyphase filtering technique is employed to reduce the number of computation steps by combining the LPF and decimation step, which eliminates needless LPF processing on generating samples that will eventually get discarded in decimation.

2) B-mode Processing: In B-mode (or pulse-echo) processing the first steps is a magnitude calculation of the demodulated signal and a logarithmic compression, which helps to convert the dynamic range into 8-bits for final display onto a monitor. The logarithm can be done using a CORDIC algorithm. Filtering is then used on both dimensions of the image. Axial and lateral filters are FIR filters that can be done using one processor. Median filtering will require some processor pipelining (see Sec. II-D) to speed up processing, but any median filter dimension larger than 5×5 will require a histogram-based algorithm that gives O(N) versus O(N^2) for a $N \times N$ filter. Both median and lateral filters require temporary storage of beam line data (spatially) for filtering across multiple beam lines.

3) Color Flow Processing: Color flow imaging takes multiple pulse-echo acquisitions along a select set of beam lines. In other words, color flow images are constructed from repeated snapshots of a subset of the whole B-mode image. In each snapshot only objects that are moving will form changes in amplitude and phase. By using a subtraction of images over time only the signals that come from moving objects remain from the difference. An arctangent is done on the difference to recover the phase information, i.e. the average velocity over a given region of interest. This process is called *autocorrelation*, which is also commonly used in radar to track moving targets.

Color flow processing requires the most amount of temporary data storage due to the fact that it needs data from previous acquisitions (temporal). Thus, after RF demodulation, the data of the past beam lines are stored before any processing is done. Otherwise, the signal processing kernels are no more demanding than in B-mode: FIR (Wall) filters and CORDIC for arctangent.

4) Spectral Doppler Processing: In spectral Doppler processing continous wave (requires analog beamforming [10]) and/or pulsed Doppler data is filtered and then sent to an FFT. A small area of tissue is targeted and a beamformed wave is transmitted, and over time a set of received data is processed using a 128 or 256-point FFT. For our chip architecture it will be trivial to do FFTs because of the built-in dedicated FFT processor [11].

D. Task Parallelization

Many tasks, such as those based on CORDIC, can be done using one processor, but execution takes many cycles per sample due to a costly iterative loop. As a result, a single loop in one processor requires a higher than necessary operating frequency in order to meet real-time requirements. Alternatively, it is possible to do a 'loop unroll' and replicate the kernel over a pipelined set of processors in order to increase throughput and reduce the minimum operating frequency. This is not beneficial from a latency standpoint, but from

Algorithm 1 CORDIC Arctangent Kernel

```
Require: Get initial values of $I$ and $Q$ from autocorrelation.
  $\theta = 0$
  for $n = 0$ to $N - 1$ do
    $I_{shifted} = I >> n$
    $Q_{shifted} = Q >> n$
    if $Q < 0$ then
      $I_{new} = I - Q_{shifted}$
      $Q_{new} = Q + I_{shifted}$
      $\theta_{new} = \theta - \arctan(2^{-n})$
    else
      $I_{new} = I + Q_{shifted}$
      $Q_{new} = Q - I_{shifted}$
      $\theta_{new} = \theta + \arctan(2^{-n})$
    end if
    $I = I_{new}$
    $Q = Q_{new}$
    $\theta = \theta_{new}$
  end for
```

a power density standpoint we rather have a distribution of many low power cores than a few high power cores.

A simple example can be shown with the CORDIC arctangent kernel. The computational core of the loop is described in Algorithm 1. The kernel consists of shifts and add/subtracts. In a serial version of this algorithm, the loop's kernel is iterated N times, where N is generally the number of bits in the fixed point format; in other words, the iterations required depends on the numerical precision required. For each iteration, θ is updated by $\pm\arctan(2^{-n})$, where $n \in 0, 1, \ldots, N-1$. The decision whether to add or subtract comes from the value of Q[1] from the autocorrelation processor (see Fig. 3). For every subsequent iteration, a new Q is computed (Q_{new}) and then a if/else decision is made where I, Q, and θ are updated based on one of two sets of rules. For a parallel pipeline implementation the kernel essentially stays the same, except that I, Q, and θ are received from the previous processor, while I_{new}, Q_{new}, and θ_{new} are sent to the next processor (rather than being written/read to/from registers).

For our implementation we use fixed-point 3.13 format, which requires 14 look-up-table entries from $\arctan(2^0)$ to $\arctan(2^{-13})$, and thus $N = 14$. Notice that if we use a smaller fixed-point format (ultrasound processing typically require only 8 bits of information) power dissipation for CORDIC arctangent will decrease linearly, e.g. $N_{new}/N_{old} = 6/14$ for a 3.5 versus 3.13 format case, as the frequency for a single processor implementation will decrease by the number of iterations. Similarly, the number of processors will decrease by the same ratio in a pipelined implementation. (Of course, other algorithms will have different tradeoff functions.)

III. Results

The results of an ultrasound application implemented on the many-core architecture is summarized in Table I. We have assumed that the ultrasound pulses are carried on a 10 MHz frequency, the decimation factor is four, sampling is done on 80 MSample/sec ADCs, the pulse repetition frequency is 19.25 kHz, there are 512 B-mode beam lines, 192 color flow beam lines, and an ensemble size of 8 (more than 1024 samples per beam line for both modes). These characteristics are relatively high-end [12], [13]. In addition, the (unweighted) median filter size is 3×3, the axial filter has 15 taps, the lateral filter has 5 taps, and the wall filters have 5 taps.

[1]The numerator in $\arctan(\frac{Q}{I})$.

Note that there is an extremely low need for local data storage per DSP processor—only large memories are required to store multiple beam lines of data. Instruction memory usage is generally modest, with only a few even going beyond half of the IMem's 128 word capacity. This shows that the processor granularity is well suited to the parallelism inherent in mid-/back-end ultrasound.

In total, there are 77 DSPs, 6 shared memories, and one FFT processor for the targeted ultrasound mid-/back-end applications. The table shows that most processors require very few cycles per FIFO read (*CPR*) and cycles per FIFO write (*CPW*), which helps reduce their minimum operating frequencies in order to meet a throughput requirement. For instance, given that the input throughput is 80 MSamples/sec., and that the *mixer* processor of the RF-demodulator takes four cycles to process each sample, the *mixer* operates at 320 MHz (see Table I). Therefore, by using each processor's *CPR* and *CPW* we can determine the optimal frequency required by each processor to avoid data starvation or data abundance—both of which causes FIFO empty/full states to occur. Energy efficient operation occurs when the mismatch in *CPR* and *CPW* of two communicating blocks is eliminated with optimal frequency selection.

While the 167-core chip can only select between two Vdds, as an experiment Table I shows an ideal DVFS where each processor can select an arbitrary voltage and the resulting average power at that voltage. This shows the extreme energy and power savings limit of our many-core design. With a total average power dissipation of 150.23 mW with all three modes concurrently running 100% of time, the average energy per frame for B-mode at 37.6 frames per second and color flow at 12.5 frames per second is 2.33 mJ/frame and 2.66 mJ/frame, respectively. In comparison, a two voltage operation with $VddHigh = 0.8\ V$ and $VddLow = 0.67\ V$ static assignments results in a total average power of 160.07 mW (a 6.5% increase), with B-mode, color flow, and spectral doppler dissipating 96 mW (2.55 mJ/frame—9.4% increase), 34 mW (2.72 mJ/frame—2.3% increase), and 30 mW, respectively. Thus, even though the majority of processors can operate efficiently using a static assignment, an adaptive DVFS can still bring in gains if implemented with low overhead circuits and algorithms [14].

IV. Conclusion

Many-core processing is becoming the norm to achieve performance gains that satisfies future needs while meeting the power density, energy, and heat requirements. Ultrasound is an example medical imaging application that can benefit from many-core processing, which has the potential to have markedly improved performance over conventional multi-core or multi-chip DSPs. By exploiting task parallelism and using an array of frequency independent simple cores, ultrasound signal and image processing can meet real-time throughput requirements while maintaining low power. However, as multi-mode (duplexing) ultrasound and improved multi-beam technologies become available, bandwidth as well as latency will become larger issues, and future ultrasound applications will further challenge system designers.

References

[1] T. L. Szabo, *Diagnostic Ultrasound Imaging: Inside Out*, 1st ed. London: Elsevier, Inc., 2004.

[2] Y. Kim, J. Kim, C. Basoglu, and T. Winter, "Programmable ultrasound imaging using multimedia technologies: a next-generation ultrasound machine," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 1, no. 1, pp. 19–29, march 1997.

TABLE I

Operating points and performance of all DSP processors, shared memory elements, and dedicated purpose processors used for B-mode, Color Flow, and Spectral Doppler ultrasound Mid-/Back-end applications. Note that shared memories and dedicated purpose processes are fixed at 1.3 V.

Processor(s) or Memory	CPR (cycles)	CPW (cycles)	IMEM (instr.)	DMEM (words)	Optimal Frequency (MHz)	Optimal Vdd (Volts)	× (units)	Average Power (mW)
B-mode								
Mixer	4	4	5	5	320	0.8	2	12.36
LPF/Decimate	4	16	22	16	320	0.8	2	12.36
Magnitude	5	5	6	1	100	0.7	1	1.54
Log (PreP1)	8	8	17	3	160	0.7	1	2.47
Log (PreP2)	10.5	5.25	21	3	210	0.75	1	3.63
Log (Core 1)	4.5	3.6	18	5	180	0.7	1	3.12
Log (Core 2 to 13)	3.6	3.6	18	5	180	0.7	12	37.38
Log (Core 14)	3	3.75	15	3	150	0.7	1	2.31
Log (Core 15)	3	4	11	1	120	0.7	1	1.85
Log (PostP)	6	12	32	4	180	0.7	1	3.12
float2fixed	3.5	7	8	3	70	0.7	1	0.81
Axial Filter	31	31	45	16	310	0.8	1	4.02
DMA Write (Med)	2.16	2.16	104	15	21.6	0.67	1	0.23
Memory (Med)	1	1	N/A	N/A	10	1.3	1	0.03
DMA Read (Med)	2.45	2.45	55	10	24.5	0.67	1	0.26
Median 0 to 3	2.56	2.56	7	1	25.6	0.67	4	1.10
Median 4	2.56	2.875	7	1	25.6	0.67	1	0.28
Median 5	2.875	3.833	7	1	25.6	0.67	1	0.28
Median 6	3.167	4.75	7	1	21.2	0.67	1	0.23
Median 7	3.75	7.5	7	1	16.8	0.67	1	0.18
Median 8	5.5	11	6	1	12.3	0.67	1	0.13
DMA Write (Lat)	2.75	2.75	69	10	3.1	0.67	1	0.03
Memory (Lat)	1	1	N/A	N/A	1.2	1.3	1	0.01
DMA Read (Lat)	2.45	2.45	35	10	2.8	0.67	1	0.03
Lateral Filter	1.6	8	9	7	1.9	0.67	1	0.02
Subtotal							41	87.79
Color Flow								
Mixer	4	4	5	5	320	0.8	2	12.36
LPF/Decimate	4	16	22	16	320	0.8	2	12.36
DMA Write	2.75	2.75	69	10	55	0.7	2	1.70
Memory	1	1	N/A	N/A	20	1.3	2	0.14
DMA Read	2.41	2.41	35	10	48.2	0.7	2	1.49
Wall Filter	1.6	8	9	7	32	0.67	2	0.92
Autocorrelation	7	7	15	5	28	0.67	1	0.40
Arctan (PreP)	1.5	1	6	0	6	0.67	1	0.09
Arctan (Core)	2.75	2.75	12	5	16.5	0.67	14	3.32
Arctan (PostP)	5	10	8	1	30	0.67	1	0.43
Subtotal							29	33.20
Spectral Doppler								
Mixer	4	4	5	5	320	0.8	2	12.36
LPF/Decimate	4	16	22	16	320	0.8	2	12.36
DMA Write	2.75	2.75	69	10	55	0.7	2	1.70
Memory	1	1	N/A	N/A	20	1.3	2	0.14
DMA Read	2.41	2.41	35	10	48.2	0.7	2	1.49
Wall Filter	1.6	8	9	7	32	0.67	2	0.92
Merge	2	1	3	0	8	0.67	1	0.11
FFT	0.5	0.5	N/A	N/A	4	1.3	1	0.16
Subtotal							14	29.24
Total							**84**	**150.23**

[3] M. Ali, D. Magee, and U. Dasgupta, *Signal Processing Overview of Ultrasound Systems for Medical Imaging*, Texas Instruments, Inc., 2008.

[4] P. Chen, M. Butts, and B. Budlong, "Ultrasound imaging and signal processing," in *Proc. SPIE, Medical Imaging: Ultrasound Imaging and Signal Processing*, vol. 6920, 2008, pp. 692 003–1—692 003–9.

[5] C.-H. Hu, F. Zheng, Y. Huang, J. Cannata, K. Shung, and P. Sun, "Design of a 64-channel digital high frequency linear array ultrasound imaging beamformer on a massively parallel processor array," in *Ultrasonics Symposium, 2008. IUS 2008. IEEE*, 2-5 2008, pp. 1266–1269.

[6] E. Waingold *et al.*, "Baring it all to software: Raw machines," *Computer*, vol. 30, no. 9, pp. 86 –93, sep. 1997.

[7] P. Gratz *et al.*, "On-chip interconnection networks of the TRIPS chip," *IEEE Micro*, vol. 27, pp. 41–50, 2007.

[8] Z. Yu *et al.*, "Asap: An asynchronous array of simple processors," *Solid-State Circuits, IEEE Journal of*, vol. 43, no. 3, pp. 695–705, mar. 2008.

[9] D. N. Truong *et al.*, "A 167-processor computational platform in 65 nm cmos," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 44, no. 4, pp. 1130–1144, Apr. 2009.

[10] E. Brunner, *Ultrasound System Considerations and their Impact on Front-End Components*, Analog Devices, Inc., 2002.

[11] A. Jacobson, D. Truong, and B. Baas, "The design of a reconfigurable continuous-flow mixed-radix fft processor," in *ISCAS*, may. 2009, pp. 1133–1136.

[12] H.-C. Kim, H.-Y. Sohn, J.-H. Sim, and T.-K. Song, "An optimized software-based echo-processing algorithm for small scale ultrasound systems," in *Ultrasonics Symposium, 2004 IEEE*, vol. 3, 23-27 2004, pp. 2053 – 2056.

[13] V. Shamdasani, R. Managuli, S. Sikdar, and Y. Kim, "Ultrasound color-flow imaging on a programmable system," *IEEE Trans. Information Technology in Biomedicine*, vol. 8, no. 2, pp. 191–199, june 2004.

[14] D. N. Truong and B. M. Baas, "Circuit modeling for practical many-core architecture design exploration," in *DAC*, jun. 2010, pp. 627–628.