FORMAÇÃO DE IMAGENS POR ULTRASSOM EM TEMPO REAL UTILIZANDO UM DSP *MULTICORE*

J. D. Medeiros Jr.[*,**], J. C. Cazarin Filho[*,**], R. C. Fernandes[*,**], T. M. Machado[*], J. E. Bertuzzo[*], E. T. Costa[**,***] e H. J. Onisto[*]

*Departamento de Hardware e Serviços, DHS/Instituto de Pesquisas Eldorado, Campinas, Brasil
**Departamento de Engenharia Biomédica, DEB/FEEC/UNICAMP, Campinas, Brasil
***Centro de Engenharia Biomédica, CEB/UNICAMP, Campinas, Brasil
e-mail: {johannes.medeiros, haroldo.onisto}@eldorado.org.br

Resumo: Uma plataforma de pesquisa onde algoritmos de processamento possam ser modificados é de grande relevância para o estudo de técnicas de formação de imagem por ultrassom. Levando isso em consideração, são apresentados neste artigo uma arquitetura de *software* embarcado, os mecanismos de processamento e os resultados da execução desses algoritmos de processamento de sinais e imagens em um DSP *multicore* como componente de uma plataforma de pesquisa, de modo a obter imagens por ultrassom em tempo real, com tempo de execução com comportamento linear e com *frame rate* de até 128 *frames* por segundo.
Palavras-chave: ultrassom**,** plataforma de pesquisa, DSP, multicore, imagem em tempo real.

Abstract: *Research platforms where processing algorithms can be modified are very important to the study of ultrasound imaging formation techniques. Considering that, in this article we present an embedded software architecture, the processing flow and the execution results achieved running those processing algorithms in a multicore DSP that is part of a research platform, producing real-time ultrasound images with linear time execution behavior and a frame rate of up to 128 frames per second.*
Keywords: ultrasound, research platform, DSP, multicore, real time imaging.

Introdução

Há, na indústria, diversos equipamentos de diagnóstico por imagem de ultrassom, com finalidade de uso em medicina. No entanto, esses aparelhos não possibilitam aos pesquisadores alterar os algoritmos de processamento que darão forma às imagens. Outros equipamentos desenvolvidos para pesquisa possibilitam captura de dados brutos ou até mesmo a incorporação de novos algoritmos de processamento. Contudo, esse processamento muitas vezes é feito de maneira *off-line* posteriormente [1, 2].

Tendo como objetivo preencher essa lacuna para os pesquisadores brasileiros que fazem investigações em tecnologias de imagens por ultrassom com uma plataforma de pesquisa que possibilite processamento configurável e tenha formação de imagens em tempo real, foi desenvolvido um protótipo de uma plataforma tecnológica de imagens por ultrassom capaz de produzir imagens Modo-B. Esse desenvolvimento foi realizado pelo Instituto de Pesquisas Eldorado em uma parceria com as universidades brasileiras integrantes da Rede Brasileira de Técnicas de Ultrassom. Além da UNICAMP, coordenadora do projeto da Plataforma Tecnológica, e do Instituto Eldorado, integrador do conhecimento e desenvolvedor do protótipo, integram a rede a UTFPR, a UFRJ, a USP, a UFSCar e o INCOR/HCFMUSP. A Plataforma conterá ainda todos os modos de operação Doppler bem como os modos M e Elastográfico, atualmente em desenvolvimento, além dos aplicativos típicos dos equipamentos de ultrassonografia convencionais.

Existem diversos modos de se realizar o processamento de sinais e imagens em um equipamento de ultrassom, por exemplo, via FPGAs [3], GPUs [4] ou mesmo *software* [5, 6]. Outra forma de realizar essas tarefas é utilizando um *Digital Signal Processor* (DSP) [3, 7].

Nesse trabalho, descreve-se parte do processamento *mid-end* e *back-end* para a formação de imagem por ultrassom Modo-B, utilizando um DSP *multicore*. São descritos a arquitetura do *software* embarcado no dispositivo e os algoritmos utilizados no processamento, bem como os resultados de tempo de execução.

Materiais e métodos

Transdutores Utilizados – O projeto da plataforma tecnológica prevê que ela seja capaz de utilizar diversos tipos de transdutores, como linear, convexo e *phased array*. Contudo, durante o desenvolvimento inicial da plataforma foram utilizados somente dois transdutores: um transdutor linear de 128 elementos e frequência central de 7 MHz com largura de banda de 4 MHz (modelo PH7L95B, BroadSound, Taiwan) e um transdutor *phased array* de 64 elementos e frequência central de 3 MHz com largura de banda de 2 MHz (modelo AT3P42A, BroadSound, Taiwan).

Arquitetura do Software Embarcado – O DSP utilizado foi um TMS320C6678 da Texas Instruments™, que possui oito núcleos para processamento, funcionando com *clock* de 1,0 GHz. Desses, utilizou-se quatro nú-

cleos para realizar o processamento de sinais e de imagens, seguindo uma arquitetura mestre-escravo. Assim, um núcleo é o mestre e os outros três são escravos. A arquitetura flexível permite expansão para outros modos de imagem, bastando utilizar mais núcleos escravos.

O núcleo Mestre é responsável pela aquisição de linhas e controle do processamento realizado pelos escravos. A comunicação entre o mestre e os escravos é feita por meio de fila de mensagens. Através das mensagens são passados todos os dados necessários para realizar o processamento, incluindo entradas e saídas dos algoritmos de processamento e os parâmetros de configuração.

A configuração que dá origem ao processamento é definida pelo usuário, quando esse escolhe o protocolo de interesse. Essa interface é feita através de um PC. Já o processamento de sinais e imagens é dividido entre FPGA e DSP, de tal modo que as linhas de entrada no DSP já passaram pelo processo de *beamforming* e detecção de envelope, realizados em FPGA (Figura 1).

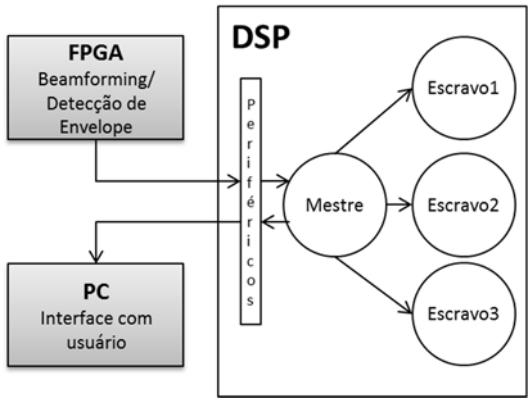

Figura 1: Diagrama simplificado da arquitetura de processamento da plataforma

Para realizar o processamento, o núcleo Mestre envia as linhas recebidas para o núcleo Escravo1 e recebe as linhas processadas. O Mestre então monta a matriz, iniciando a etapa matricial, desempenhada pelos outros dois núcleos escravos. Essas tarefas são realizadas de maneira paralela utilizando *double buffers*. Na Tabela 1 são listadas as tarefas de cada um dos núcleos.

Tabela 1: Tarefas executadas pelos núcleos do DSP

Núcleo	Tarefas
Mestre	Recepção de linhas, montagem de matriz e controle do processamento
Escravo1	Processamento de linha
Escravo2	Processamento matricial
Escravo3	Codificação de frame B

Como se nota na Figura 1 e na Tabela 1, somente o núcleo definido como Mestre tem interface com os periféricos do DSP, já o processamento de sinal para geração da imagem final é totalmente feito pelos escravos. Há também uma camada de abstração de hardware (não mostrada na figura) que permite o uso de periféricos (por exemplo, os utilizados para comunicação externa).

Descrição do Processamento – O processamento no DSP é iniciado após a detecção de envoltória, realizada em FPGA. O sinal em banda base após a detecção de envelope é amostrado a 8,0 MHz, taxa suficiente para as larguras de banda dos transdutores utilizados, que são menores ou iguais a 4,0 MHz.

Sabendo que ocorre aquisição de sinal durante todo o tempo de propagação da onda de ultrassom, desde a emissão até o retorno do eco da profundidade máxima, a quantidade de pontos obtida é dada pela Eq. 1:

$$N = \frac{2.d.fs}{c} \quad (1)$$

onde N é o número de pontos de cada linha, d é a profundidade máxima da imagem, fs representa a frequência de amostragem do sinal e c é a velocidade de propagação do som no meio.

Com a frequência de amostragem igual a 8,0 MHz, a velocidade de propagação do som no meio de 1540 m/s e cinco valores de profundidade máxima (de 4 a 20 cm, em intervalos de 4 cm), são obtidos os valores de números de pontos apresentados na Tabela 2.

Contudo, os valores calculados através da Eq. 1 são valores baseados somente na propagação da onda ultrassônica, sem considerar as especificidades do dispositivo onde o processamento é realizado.

O DSP utilizado possui uma linha de *cache* com tamanho de 128 *bytes* e a execução realizada no dispositivo tende a ser mais rápida caso o vetor utilizado tenha tamanho múltiplo do tamanho da linha de *cache* [8]. Como cada ponto ocupa 2 *bytes*, a quantidade de pontos utilizada é um múltiplo de 64, próximo ao valor calculado baseado somente na propagação da onda. Esses valores também são mostrados na Tabela 2.

Tabela 2: Número de pontos das linhas processadas pelo DSP calculado baseado na profundidade da imagem gerada (N – Profundidade) e calculado com base no tamanho da linha de *cache* do DSP (N – DSP).

d (cm)	N – Profundidade	N – DSP
4	416	448
8	831	832
12	1247	1216
16	1662	1664
20	2078	2048

Já o número de linhas (*scanlines*) varia conforme o transdutor e a configuração utilizada. Para os transdutores disponíveis, que possuem no máximo 128 elementos, determinou-se que para o transdutor linear o número de linhas seria sempre 128, já para o transdutor *phased array* seria duas vezes o valor do ângulo de abertura mais 1, limitado a 127 linhas. Por exemplo, com uma abertura de 30º serão obtidas 61 linhas, para 90º, 127 linhas. Esses valores devem ser alterados caso sejam utilizados transdutores com um maior número de elementos.

O algoritmo de processamento de linha executado pelo Escravo1 realiza a compressão logarítmica das linhas recebidas do Mestre, que por sua vez as obtém da FPGA. O processamento é feito através de uma função que recebe uma entrada de 16 bits com sinal e, utilizando um *look up table* com valores de logaritmo, converte a entrada para uma saída de 8 bits sem sinal, utilizando o valor de *dynamic range* especificado pelo usuário do aparelho e devolve a saída para o Mestre. Essa função foi criada em linguagem de programação C, utilizando geração automática de código a partir do MATLAB®.

Por sua vez, o núcleo Escravo2 é responsável pelo processamento matricial dos dados e consequente formação da imagem final de ultrassom. Esse núcleo recebe como entrada do Mestre um vetor de 8 bits sem sinal, que representa uma matriz com as saídas da compressão logarítmica. Essa matriz passa pelos algoritmos de filtro, *scan conversion* e transposição de matriz.

Utilizou-se um filtro de mediana realizado utilizando a biblioteca ImgLib® da Texas Instruments™, com uma função que filtra uma linha individualmente. Isso é feito um número de vezes igual ao número de linhas usadas.

O *scan conversion* realiza a conversão da matriz filtrada, para o tamanho da matriz (linhas e colunas) que será recebida pelo PC, além de transformar o sistema de coordenadas quando as imagens geradas pelos transdutores utilizados são curvadas. Esse algoritmo é baseado na técnica de interpolação *nearest neighbor* e recebe, além da entrada e dos parâmetros de configuração, uma tabela com índices que indicam o mapeamento ponto a ponto da matriz de entrada para a matriz de saída.

O tamanho da imagem de saída do *scan conversion* depende do transdutor utilizado. Para o transdutor linear ela é de 1024 x 128 e para o *phased array* é de 512 x 256, totalizando, para ambos os casos, 128 KB. Esses valores foram escolhidos considerando as dimensões dos transdutores disponíveis.

O processamento no Escravo2 é finalizado com a transposição da matriz de saída do *scan conversion* para tornar a imagem adequada para visualização. As funções de *scan conversion* e transposição de matriz também foram geradas automaticamente pelo MATLAB®.

A saída final, após todo o processamento realizado, é enviada para o Mestre que então recruta o Escravo3, para que este codifique a imagem em caracteres para posterior envio ao PC e a devolva para o Mestre.

Testes – Para avaliar o tempo de execução dos algoritmos foram calculadas as médias do tempo de processamento de 1000 execuções em cada escravo.

No Escravo1 foram anotados os tempos de execução em função da profundidade da imagem e, consequentemente, do número de pontos das linhas. O objetivo para o algoritmo de compressão logarítmica é que ele tenha um tempo de execução linear e inferior ao tempo necessário para a propagação do sinal de ultrassom naquela profundidade.

No núcleo Escravo2 foram variados o tipo de transdutor, o número de linhas e o número de pontos de cada linha, bem como os tempos de processamento foram calculados em função do tamanho do vetor de entrada, que é dado pelo produto entre o número de linhas e o número de pontos da linha, independentemente do transdutor utilizado. Para o processamento matricial, espera-se que o tempo de execução tenha característica linear e que possibilite um *frame rate* superior a 60 *frames* por segundo (fps), valor encontrado em equipamentos comerciais.

No Escravo3, a codificação utiliza sempre a mesma quantidade de dados, não importando o número de linhas, a profundidade ou tipo de transdutor. Nesse caso espera-se que o tempo de execução seja aproximadamente constante e suficiente para obter 60 fps.

Também foram geradas imagens para avaliar qualitativamente o processamento. Para isso foi utilizado um *phantom* fetal (modelo 068, CIRS Inc – Norfolk, EUA) e os dois transdutores já mencionados. Esse *phantom* simula um feto com 21 semanas e possui algumas estruturas, dentre elas, ventrículos cerebrais.

Resultados

Processamento no Escravo1 – A regressão linear feita para obter uma equação que determine o tempo de processamento (em µs), necessário para realizar a compressão logarítmica de uma linha, tem coeficiente de determinação $R^2 = 0{,}9998$ e é dada pela Eq. 2.

$$t = 7{,}12.d + 4{,}50 \qquad (2)$$

onde *d* é a profundidade máxima da imagem (em cm).

Na Tabela 3 é mostrado um comparativo entre o tempo necessário para a propagação da onda de ultrassom (t Ultrassom) e o tempo de processamento de uma linha (t Comp. Log.), ambos variando com a profundidade (d).

Tabela 3: Valores de tempo de propagação da onda de ultrassom e tempo de processamento da compressão logarítmica (média ± desvio padrão)

d (cm)	t Ultrassom (µs)	t Comp. Log. (µs)
4	51,9	32,5 ± 0,09
8	103,9	61,5 ± 0,13
12	155,8	90,4 ± 0,14
16	207,8	119,1 ± 0,19
20	259,8	146,2 ± 0,18

Considerando o maior número possível de linhas utilizadas (128) e a maior profundidade (20 cm) teríamos que o tempo de um *frame* é de 18,7 ms (128 x 146,2 µs) que corresponde a um *frame rate* de 53,4 fps.

Processamento no Escravo2 – Na Tabela 4, são mostrados os tempos máximos médios (obtidos com 1000 medidas) de execução dos três algoritmos de processamento executados nesse núcleo, bem como o tempo total de processamento dele, tanto com transdutor *phased array* quanto com transdutor linear.

Tabela 4: Tempo médio máximo de processamento das funções efetuadas no Escravo2 (média ± desvio padrão)

Algoritmo	t Phased (µs)	t Linear (µs)
Median Filter	1000,1 ± 10,88	996,1 ± 12,82
Scan Conversion	821,4 ± 1,17	648,2 ± 1,51
Transposição	2357,6 ± 0,73	2389,2 ± 1,83
Total	4180,0 ± 10,76	4033,5 ± 14,45

Como pode ser visto na Tabela 4, o tempo máximo de processamento no Escravo2 é de 4,2 ms, que equivale a um *frame rate* de 238 fps. Já a regressão linear do tempo (em µs) para realizar o processamento matricial em função do número de pontos da matriz de entrada (*M*) com um transdutor *phased array* é mostrada na Eq. 3 (R^2 = 0,995) e na Eq. 4 para o linear (R^2 = 0,9985):

$$t = 4{,}7.10^{-3}.M + 2947{,}8 \quad (3)$$

$$t = 3{,}7.10^{-3}.M + 3024 \quad (4)$$

Na Figura 2 é apresentada uma imagem do *phantom* fetal obtida com o transdutor *phased array*. Somente é mostrada a imagem obtida com esse transdutor porque ele possui menor frequência central e pode realizar imagens de profundidades maiores do que o linear.

Figura 2: Imagem de *phantom* fetal obtida na plataforma com um transdutor *phased array*

Processamento no Escravo3 – O tempo de execução da codificação tem variação entre o menor e o maior valor medidos de 0,04 % e igual a 7,8 ms, que equivale a um *frame rate* de 128 fps.

Discussão

Os resultados apresentados mostram que todos os algoritmos têm tempo de execução linear. Isso é importante para que o tempo de processamento não cresça exponencialmente, inviabilizando a aplicação em tempo real. Nota-se também que o algoritmo de processamento de linha tem sempre tempo de execução menor do que o tempo necessário para a propagação da onda de ultrassom, portanto o fator limitante do *frame rate* são as características físicas da geração da imagem.

Em relação ao processamento matricial, o limitante é o algoritmo de codificação da imagem. Ainda assim, seria possível obter um *frame rate* de 128 fps.

Em relação à imagem gerada, não foi feita nenhuma análise quantitativa da qualidade da imagem, mas percebe-se claramente a morfologia do crânio do feto.

Conclusão

O processamento de dados para formação de imagens de ultrassom utilizando algoritmos gerados automaticamente a partir de MATLAB® e executados em um DSP *multicore* mostrou-se capaz de gerar imagens modo B em tempo real e de maneira previsível.

Agradecimentos

Os autores agradecem ao Ministério da Saúde e à FINEP pelo financiamento do projeto.

Referências

[1] The Verasonics Ultrasound System. Verasonics. [internet]. 2012 [cited 24 Jul 2014] Available from: www.verasonics.com/uploads/files/pdf/VerasonicsFeaturesSummary.pdf.

[2] Assef AA, Maia JM, Gewehr PM, Gamba HR, Costa ET, Button VLSN. Sistema para Geração, Aquisição e Processamento de Sinais de Ultra-som. Revista Controle & Automação. 2009; 20(2):145-55.

[3] Schneider FK, Agarwal A, Yoo YM, Fukuoka T, Kim Y. A Fully Programmable Computing Architecture for Medical Ultrasound Machines. IEEE Transactions on Information Technology in Biomedicine. 2010; 14(2):538-40.

[4] Chiu HCT, Zhang L, Cheung DKH, Hu C, Shung KK, Yu ACH. Design of a Programmable Micro-Ultrasound Research Platform. In: Proceedings of IEEE International Ultrasonics Symposium; 2010 Oct 11-14; San Diego, USA. 2010. p. 1980-3.

[5] Shamdasani V, Kim Y. Programmable Low-cost Ultrasound Machine. In: Proceedings of the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society; 2003 Sep 17-21; Cancún, México. 2003. p. 1164-7.

[6] Lewandowski, M. Hardware-Software Partitioning of Digital Signal Processing in Ultrasound Medical Devices: a Case Study. In: M. Tanabe, editor. Ultrasound Imaging. InTech [internet]. 2011 [cited 24 Jul 2014]. p. 1-16. Available from: www.intechopen.com/books/ultrasound-imaging.

[7] Pailoor R, Pradhan D. Digital Signal Processor (DSP) for Portable Ultrasound. Application Report SPRAB18A, Texas Instruments [internet]. 2008 [cited 24 Jul 2014]. Available from: www.ti.com/lit/an/sprab18a/sprab18a.pdf.

[8] TMS320C66x DSP Cache User Guide SPRUGY8, Texas Instruments [internet]. 2010 [cited 24 Jul 2014]. Available from: www.ti.com/lit/ug/sprugy8/sprugy8.pdf.