Avaliação do processador SAR *Range-Doppler* para operação em tempo real em um computador de uso geral

Sérgio Trofino[1,2], David Fernandes[2] e Roberto d'Amore[2]
[1]Mectron – Avenida Brigadeiro Faria Lima, 1399 – Parque Flamboyant – São José dos Campos – SP – CEP 12227-000
[2]Instituto Tecnológico de Aeronáutica ITA - Praça Mal Eduardo Gomes, 50 – Vila das Acácias – São José dos Campos – SP – CEP 12.228-900

***Resumo* — Este trabalho apresenta a análise e a realização de um processador SAR *Range-Doppler*, para operação em tempo real, em um computador de uso geral - PC. O algoritmo foi codificado em linguagem C e os resultados obtidos mostraram que é possível realizar o processamento deste algoritmo em tempo real, para uma dada geometria de aquisição dos dados, sem a necessidade de um *hardware* dedicado. Apenas as facilidades do Sistema Operacional e um PC permitem a realização do processador.**

***Palavras-Chave* — Radar de Abertura Sintética, tempo real, Range-Doppler.**

I. INTRODUÇÃO

Radares de Abertura Sintética (SAR - do inglês *Synthetic Aperture Radar*) são comuns na observação de áreas de conflito, avaliação de extensão de desastres naturais e outras aplicações de sensoriamento remoto. Isto se deve, dentre outros fatores, o fato do SAR ser um sensor ativo, que pode ser operado em variadas condições climáticas e de luminosidade [1]. Principalmente em aplicações militares envolvendo comando e controle (C2) tem-se como requisito a visualização em tempo real de áreas de interesse. Porém, a geração de imagens SAR em tempo real é custosa em termos de processamento devido à alta quantidade de dados a ser processado [2].

Uma grande vantagem do SAR é a sua capacidade de sintetizar, na direção de voo do sensor, uma antena de grande comprimento efetivo, gerando como consequência imagens de alta resolução. Como desvantagem, podemos citar a grande demanda necessária de processamento para gerar uma imagem em tempo real.

Dentre os modos possíveis de operação de um SAR, este trabalho considera o *stripmap mode*, onde uma antena fixa à plataforma móvel ilumina uma faixa no solo a medida que a plataforma se desloca [3].

Muitos estudos tratam do problema da formação da imagem SAR a partir dos dados brutos (sinais ecos que são recebidos pela antena do sensor Radar) considerando a utilização de *hardwares* dedicados compostos por arranjos de DSPs (*Digital Signal Processor*) [4], FPGAs (*Field-programmable gate array*) [5], uma combinação de ambos [6], [7], GPUs (do inglês *Graphics Processing Unit*) [8], [9], entre outros. Os trabalhos, em linhas gerais, exploram o paralelismo das arquiteturas dos algoritmos de formação de imagens para conseguirem o processamento em tempo real.

Dado o aumento do poder de processamento de PCs devido ao maior número de núcleos processadores, aumento da capacidade e da velocidade das memórias, a implementação de um processador SAR em um PC pode ser considerada. Os benefícios são vários: desenvolvimento mais simples, menor custo em termos de homem/hora, disponibilidade de interfaces e *drivers* prontos para troca de dados (ETHERNET, USB, etc), hardware integrado (placa de vídeo, placa de redes, etc), *softwares* para auxilio no desenvolvimento (*softwares* com aplicativos matemáticos, monitoramento do sistema), gerenciamento do processamento por núcleo, dentre outros.

Este trabalho descreve, para uma dada geometria de aquisição de dados, um processador SAR em tempo real, onde por tempo real entende-se que o tempo de processamento e exibição de um quadro da imagem capturada pelo sensor SAR é menor que a taxa de chegada de novos dados. Deste modo, os dados não são acumulados e não ocorre um atraso crescente entre a recepção de dados e a saída (exibição do quadro). Para isto, é necessário receber e processar um bloco bidimensional (alcance x azimute) de dados e exibi-los num tempo menor que a recepção e o processamento de um novo bloco de dados, de mesma dimensão do anterior. O processamento SAR, escolhido como sendo o algoritmo *Range-Doppler*, [10], [3], consiste basicamente na aplicação de FFT (*Fast Fourier Transform*) nos dados e na multiplicação complexa do resultado por uma função de referência. Este processador foi realizado em um *notebook* com processador Intel Core i3, sistema operacional Linux 2.6.35.22, distribuição Ubuntu 10.10, 3GB de memória RAM. O algoritmo *Range Doppler* foi codificado em linguagem C, usando como compilador o GCC, já disponível no sistema operacional.

O trabalho realizado está dividido do seguinte modo: no Capítulo 2 apresenta-se a fundamentação teórica do processamento SAR e faz-se um levantamento de algumas arquiteturas de processamento utilizadas; no Capítulo 3 tem-se a proposta deste trabalho, sua realização e as considerações assumidas para sua execução; no Capítulo 4 mostram-se os resultados obtidos; e no Capítulo 5 tem-se a conclusão do trabalho.

II. PROCESSAMENTO DE IMAGENS SAR

Considera-se que o sinal transmitido por um SAR é um *chirp* (pulso modulado linearmente em frequência), representado pela envoltória complexa dada por:

$$\tilde{s}(t) = K_0 rect\left[\frac{t}{T_p}\right]\exp\{j\pi\gamma_R t^2\} \qquad (1)$$

onde K_0 é uma constante real positiva, *rect*[.] um pulso retangular com duração T_p, t é o tempo e γ_R a taxa de variação da frequência instantânea.

A transmissão de sucessivos sinais *chirp*, à medida que o sensor SAR se desloca, e a recepção dos sinais ecos retroespalhados pela cena iluminada dão origem aos dados brutos que, em termos espectrais, podem ser representados como [11]:

$$H(\xi,\eta) = \iint \gamma(a,r)\exp\left\{-j\frac{4\pi}{\lambda}r\right\} \times G(\xi,\eta;r)\exp\{-j(2\pi\xi a + 2\pi\eta r\}dadr \quad (2)$$

onde $\gamma(a,r)$ é a refletividade complexa da cena nas coordenadas (a, r), a é a variável independente azimute (direção de deslocamento do sensor), r é a variável independente alcance ou *range* (direção de propagação da onda transmitida pelo Radar), $G(\xi,\eta;r)$ é a transformada dupla de Fourier da função espalhamento pontual $\tilde{g}(a'-a,r'-r)$, que representa o sinal eco no plano (a',r') relativo a um alvo pontual na posição (a,r) [11]. $G(\xi,\eta;r)$ é dado por:

$$G(\xi,\eta;r) = rect\left[\frac{\xi}{2\Delta\xi}\right] rect\left[\frac{\eta}{2\Delta\eta}\right]\exp\{-j2n\Psi(\xi,\eta;r)\} \quad (3)$$

com (ξ,n) representando as dimensões espectrais azimutal e radial, respectivamente, na faixa $-\Delta\xi \le \xi \le \Delta\xi = 1/L_a$, $-\Delta\eta \le \eta \le \Delta\eta = (\gamma_R T_p)/c_0$, sendo L_a o comprimento efetivo da antena, c_0 a velocidade da luz no vácuo e

$$\Psi(\xi,\eta;r) = \frac{\eta^2 c_0 T_p}{8\Delta\eta} - \left(\eta+\frac{2}{\lambda}\right)r + r\sqrt{\left(\eta+\frac{2}{\lambda}\right)^2 - \xi^2} \quad (4)$$

O processador SAR que gera a imagem estimará a refletividade complexa da cena através da equação:

$$\hat{\gamma}(a,r) = \int_{-\Delta\xi'}^{\Delta\xi'}\int_{-\Delta\eta'}^{\Delta\eta'} \iint H(\xi,\eta)G^*(\xi,\eta;r_0) \times \exp\{-j2\pi\xi a\}\exp\{-j2\pi\eta r\}d\xi d\eta \quad (5)$$

que representa a Transformada de Fourier inversa do produto $H(\xi,\eta)$ por $G^*(\xi,\eta;r_0)rect\left[\xi/(2\Delta\xi')\right]rect\left[\eta/(2\Delta\eta')\right]$, onde $\Delta\xi' \le \Delta\xi$ e $\Delta\eta' \le \Delta\eta$ regulam a resolução final da imagem.

Assim, considerando um alvo pontual com refletividade complexa γ_0 nas coordenadas $(\bar{a},\bar{r})$ definido por:

$$\gamma(a,r) = \gamma_0\delta(a-\bar{a}, r-\bar{r}) \quad (6)$$

obtem-se, com (6) em (5), a imagem SAR do alvo pontual (função espalhamento pontual) localizado em $(\bar{a},\bar{r})$:

$$\hat{\gamma} = (a,r) = 4\gamma\Delta\xi'\Delta\eta'\exp\left\{-j\frac{4\pi}{\lambda}\bar{r}\right\} \times \frac{sen[2\pi\Delta\xi'(a-\bar{a})]}{2\pi\Delta\xi'(a-\bar{a})}\frac{sen[2\pi\Delta\eta'(r-\bar{r})]}{2\pi\Delta\eta'(r-\bar{r})} \quad (7)$$

Processamento SAR é um problema bidimensional. O sensor SAR coleta a energia dispersada na reflexão do sinal por um alvo e a focaliza em um único pixel na imagem. Em *range*, ou distância, o espalhamento do sinal tem a duração T_p do pulso transmitido e em azimute o espalhamento tem a duração do tempo de iluminação de um alvo pontual pelo feixe da antena, sendo, portanto variante com o alcance. Na imagem SAR ocorre a focalização e o espalhamento, o qual, inicialmente, é muito grande (centenas de metros), podendo ficar reduzido a alguns metros $1/\Delta\xi'$ em azimute e $1/\Delta\eta'$ em alcance, como pode ser constatado em (7).

Um efeito que deve ser destacado, que será compensado no processamento SAR, é o chamado *Range Cell Migration* (RCM). O RCM provoca um acoplamento entre a dimensão azimutal e em alcance, que pode ser notada no terceiro termo da expressão (4), onde são mesclados η, ξ e r. O RCM é devido ao fato da distância do Radar a um alvo na cena ser variante com o tempo, devido o deslocamento do sensor SAR. Esse efeito pode ser observado na Fig. 1, onde se mostra os sucessivos sinais ecos de um alvo pontual com diferentes atrasos.

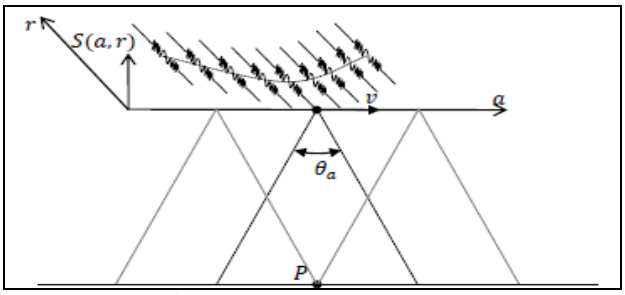

Fig. 1. Migração da célula de resolução com o deslocamento do sensor SAR [8].

O problema do processamento de imagens SAR em tempo real é tratado com o uso de arquiteturas que empregam DSPs operando em conjunto [1], [4], [7], [12], [13]. Estes trabalhos têm em comum o uso do DSP *ADSP-TS201S* ou do *ADSP-TS101S*, onde são combinados em uma placa para atingir a capacidade computacional necessária.

Dentre os algoritmos existentes empregados no processamento de imagens SAR em tempo real, tem-se, por exemplo, o *Chirp Scaling* [14], o k-ω [15] e o *Range-Doppler*. Nesse trabalho, por suas vantagens, optou-se pelo algoritmo *Range-Doppler*, sendo este um dos mais utilizados [8].

O algoritmo *Range-Doppler* executa a compressão em *range* do sinal, a correção de migração em *range* e a compressão azimutal, conforme a representação na Fig. 2.

A compressão em *range* consiste na transformada de domínio do sinal, aplicando-se a FFT em cada linha de *range* recebida, e a multiplicação deste resultado por uma função de referência $rect\left[\eta/(2\Delta\eta')\right]\exp\{j2\pi\eta^2 c_0 T_p/(8\Delta\eta')\}$, que pode ser a FFT de uma réplica do sinal transmitido e a aplicação da FFT inversa no sinal. A correção de migração em *range* é

realizada aplicando-se uma FFT na direção azimutal e interpolando-se o sinal comprimido na direção do *range* (para se ter mais precisão nos deslocamentos a serem realizados, a fim de se compensar o RCM). Por fim, a compressão em azimute consiste na multiplicação do sinal por uma função de referência do sinal azimutal (8):

$$rect\left[\xi/(2\Delta\xi')\right]\exp\left\{j2\pi\left[-(2/\lambda)r+r\sqrt{(2/\lambda)^2-\xi^2}\right]\right\} \quad (8)$$

e a aplicação da FFT inversa no sinal. Nota-se que, após a correção da RCM, o parâmetro *r* na função de referência azimutal é conhecido e é constante, em função da linha em *range* que está sendo feita a focagem na direção azimutal.

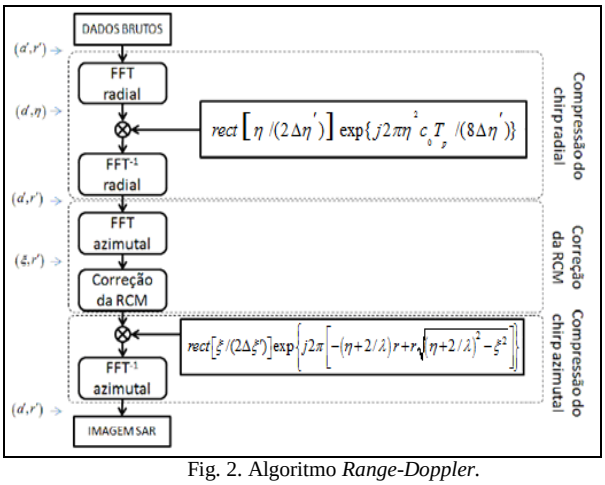

Fig. 2. Algoritmo *Range-Doppler*.

Apesar das diferenças entre o *Chirp Scaling* e o *Range-Doppler*, podemos comparar as partes em comum, como, por exemplo, FFT em range e FFT em azimute, as quais demandam um tempo de processamento elevado, conforme é possível verificar nos tempos de execução apresentados em [4].

Em [4], o algoritmo *Chirp Scaling* é usado no processador SAR e uma FFT de 1024 pontos é executada em 15,7μs no ADSP-201S. O algoritmo empregado possui passos para correção de movimento, correção de fase, entre outros. Em [4] o sensor está instalado em uma aeronave, executa-se o processador SAR em tempo real e se justifica o uso do algoritmo *Chirp Scaling* através das Equações (8) e (9). Liu et al. [4] afirmam que o algoritmo *Range-Doppler* é apropriado se for possível ignorar a migração em *range* e o Chirp *Scaling* é mais apropriado quando a migração em *range* não pode ser ignorada. Apesar desta consideração, na referência não se define os limites para considerar que estes valores podem ser ignorados.

$$M_r = \frac{\lambda^2 r}{32\delta_A{}^2}, \quad r = r_{\min}, r_{\max} \quad (8)$$

$$\Delta M = M_{r_{\max}} - M_{r_{\min}} = \frac{\lambda^2 (r_{\max} - r_{\min})}{32\delta_A{}^2} \quad (9)$$

onde M_r é o máximo deslocamento da RCM de um alvo a uma distância *r* da trajetória do sensor SAR, ΔM é a diferença do máximo desvio da RCM dos alvos no extremo da faixa imageada, δ_A é resolução azimutal, r_{min} é a menor distância da cena ao Radar (near range) e r_{min} é a maior distância da cena ao Radar (far range).

Em [9], tem-se a comparação do desempenho do algoritmo *Range-Doppler* executado em dois *hardwares* distintos: uma GPU de 240 núcleos e uma CPU com 4 núcleos. Uma imagem de 4096X4096 de tamanho é processada em 127.129 segundos na CPU, enquanto que na GPU a mesma imagem é processada em 0.832 segundos.

Uma outra implementação de processador SAR em tempo real empregando GPUs é relatada em [8]. Nesta proposta foi empregada uma GPU com 192 núcleos para realizar a compressão em azimute e uma CPU para a compressão em range. O algoritmo do processador SAR é o mesmo utilizado neste trabalho, permitindo, assim, uma comparação de desempenho das soluções empregadas.

Em [15] e [16] usa-se uma arquitetura com diversos núcleos em paralelo controlados por um FPGA. Esses núcleos são formados por um DSP denominado HiPAR-DSP 16 desenvolvido especialmente para processamento de imagens que utilizam FFT. O algoritmo utilizado é o ω-k, um filtro bidimensional que também executa FFT em *range* e em azimute. No processamento são executadas 4096 FFTs de 4096 em 629.15ms [15].

O ponto comum de todos os trabalhos é a execução do algoritmo em diversos núcleos, paralelizando o processamento. Também podemos destacar a necessidade de uma grande quantidade de núcleos de processamento, principalmente no caso da GPU [9], para se alcançar o desempenho desejado. Além disto, alguns *hardwares* acabam consumindo muita potência e se tornam pesados. Em [12] implementou-se em C++ e *Assembly* o algoritmo *Range Doppler* em um hardware com mais de 80 DSPs, pesando 35 kg e consumindo de 800 a 1000 Watts. Observa-se que, com um consumo desta ordem, o conjunto de baterias para alimentar este equipamento inviabiliza o seu uso como equipamento portátil.

O algoritmo utilizado neste trabalho é apresentado na Fig. 2 e é, basicamente, o mesmo utilizado em [8] e [1].

III. ABORDAGEM

Para analisar o resultado do processamento de uma imagem SAR em tempo real, é necessário definir o cenário de captura da imagem. Foi escolhido um cenário simulado com três refletores pontuais dispostos eqüidistantes em *range* e em azimute em um fundo não reflexivo conforme a disposição mostrada na Fig. 3. Numa imagem real, cada refletor seria como um pequeno objeto na cena imageada que reflete a maior parte da energia do sinal recebida em direção ao sensor, resultando em um ponto brilhante na imagem. Os parâmetros do processador SAR são apresentados na Tabela I.

Os dados do cenário e dos parâmetros adotados são baseados em uma aplicação real e são gerados por um aplicativo de modo que estes dados alimentem o processador SAR, simulando uma captura real de um sensor. Em [8], usa-se um cenário parecido. Com esta escolha, podemos fazer uma melhor comparação de desempenho com [8], uma vez que ele apresenta dados de tempo de execução e medida de qualidade do resultado obtido.

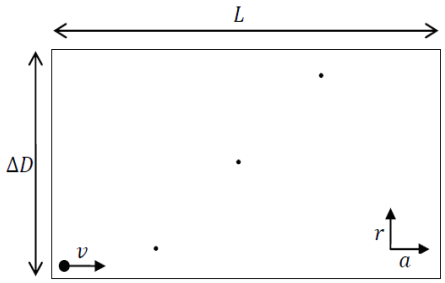

Fig. 3. Cenário considerado para simulação de dados brutos.

TABELA I PARÂMETROS DE PROCESSADOR SAR CONSIDERADO

Velocidade da plataforma	v = 200 m/s
Altura da plataforma	h = 4000 m
Resolução azimutal e em range	δA = δB = 1.0 m
Largura de banda azimutal (total)	Ba = 400 Hz
Abertura azimutal da antena	θa=6.88 o
Comprimento de onda	λ = 0.03 m
Largura da faixa imageada	ΔD = 2000 m
Comprimento da faixa imageada	L = 2000m
Distância mínima da faixa imageada em range	Rmim = 6000 m
Distância máxima da faixa imageada em range	Rmax = 7200 m
Largura da faixa imageada em range	ΔR = 1200 m
Largura do pulso	Tp = 0.667 us
Largura de banda em range	Br = 150 MHz
Chirp rate	γ ≈ 2.25 x 1014
Pixel-spacing em range	dR = 0.5m
Pixel-spacing em azimute	dA = 0.125m
Número de amostrar em range	N = 2048
Freqüência de amostragem	fp ≈ 300 MHz
Pulse Repetition Frequency	PRF = 1600 Hz

O tempo para executar o algoritmo do processador SAR em tempo real depende da PRF, pois é nessa frequência que chegam os sinais ecos, no caso com 2048 amostras. No cenário proposto, tem-se que o tempo para execução de 8192 linhas em azimute deve ser de 8192/PRF. Como PRF = 1600Hz, o tempo de execução deve ser menor que 5,12 segundos.

Utilizando-se os valores da Tabela I e considerando um limiar de 10% para desprezar um valor, aplicando-se as Equações (8) e (9), verifica-se que a diferença do máximo desvio da RCM é menor que 10% da resolução azimutal, enquanto que a migração em *range* é maior que 15%. Assim, podemos desprezar a diferença de migração em *range*, mas não a migração em range, o que justifica a escolha do *Range-Doppler* para este trabalho [4].

A Fig. 4 apresenta o aplicativo desenvolvido neste trabalho.

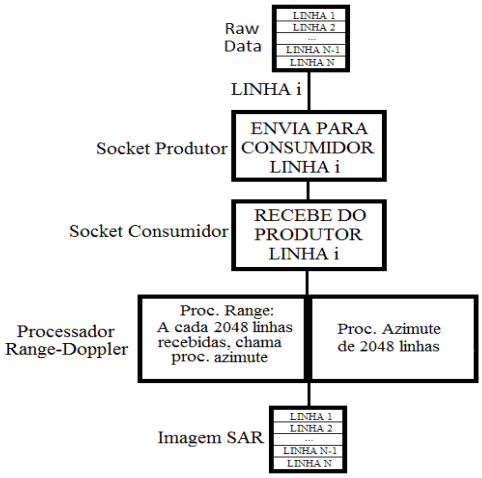

Fig. 4. Ilustração da estrutura do programa codificado.

Codificou-se um aplicativo para gerar o cenário de teste, dado o tamanho da cena, a posição dos alvos pontuais. A saída do aplicativo, uma matriz de valores complexos, foi transmitida por *socket TCP/IP* para outro aplicativo, o qual ficou responsável pela execução do processador SAR. Este aplicativo contém duas *threads*, uma que executa o processamento em *range* e outra em azimute.

A imagem - o resultado do processamento - foi escrita em um arquivo texto e interpretada via um aplicativo matemático (IDL) que dispõe de ferramentas próprias para exibição da imagem.

Como a FFT é uma das funções mais críticas deste processamento e não é o foco deste trabalho desenvolver um algoritmo de FFT, usou-se a biblioteca FFTW, desenvolvida e distribuída pelo MIT [17], para executar estes cálculos.

IV. RESULTADOS

A Fig. 5 apresenta os passos do processamento da imagem SAR desde os ecos até a imagem processada. Temos o dado bruto (a), a compressão em *range* (b), a correção de migração em *range* (c), e a imagem final (d). A Fig. 6 apresenta a saída do processador SAR para o cenário com três refletores. A Fig. 6 teve suas cores invertidas para melhorar a visualização, além do que seus alvos foram circundados com a mesma finalidade.

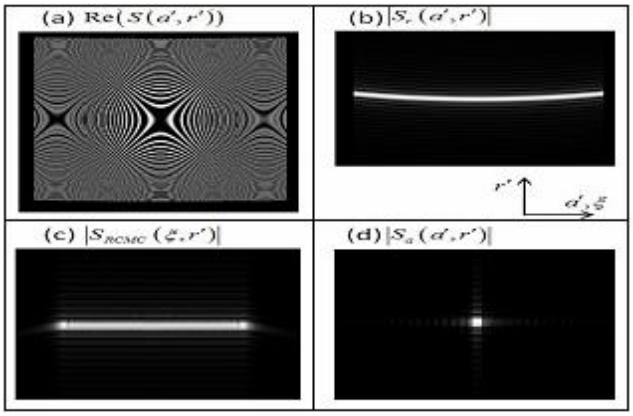

Fig. 5. Dado bruto gerado para teste do processador SAR considerando um refletor.

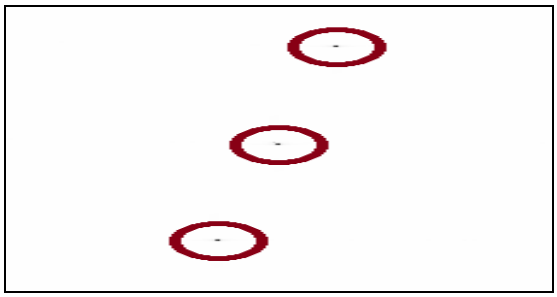

Fig. 6. Parte real da imagem SAR com os alvos circundados para destaque.

O *hardware* utilizado, assim como as ferramentas de análise, são relacionados na Tabela II.

A Tabela III apresenta uma comparação entre os resultados alcançados neste trabalho com outros trabalhos. Entende-se por tempo limite o tempo máximo para execução do processador SAR, ou seja, o tempo entre o início da aquisição do dado do sensor até a exibição da imagem de modo que não haja acúmulo de dados no tempo. Por tempo

alcançado, entende-se como o tempo que o algoritmo especificado foi executado no *hardware* empregado em cada trabalho.

TABELA II HARDWARE E FERRAMENTAS UTILIZADAS

Processador	Intel Core i3 350 2.27 GHz
Memória	1 slot 2GiB-64bits 1067 MHz
	1 slot 1GiB-64bits 1067 MHz
SO	Ubuntu 10.10 32bits - Desktop
Compilador	gnu-gcc com openmp
Uso da CPU	gnome-system-monitor

TABELA III TABELA COM RESULTADOS PARA COMPARAÇÃO DESTE TRABALHO COM A LITERATURA

	[4] (DSP)	[16] (DSP)	[8] (GPU)	[8] (PC)	Este (PC)
Algoritmo	*Chirp-Scaling*	ωk	*Range-Doppler*	*Range-Doppler*	*Range-Doppler*
Tempo limite	4.00 seg	13.00 seg	10.00 seg	10.00 seg	5.12 seg
Tempo alcançado	1.25 seg	12.97 seg	2.08 seg	20.19 seg	3.07 seg

A Tabela III mostra que em termos de tempo de processamento as abordagens [4] e [8] (GPU) são as mais rápidas, entretanto o processador implementado neste trabalho tem uma complexidade de Hardware e de Software de desenvolvimento muito menor, pois utiliza um PC de uso geral. Comparando-se a solução em PC de [8], o processador desenvolvido neste trabalho foi mais rápido.

O tempo de processamento do processador utilizado pode ainda ser reduzido otimizando-se o algoritmo da FFT para o hardware utilizado, bem como explorando-se os recursos do próprio hardware e do compilador.

V. CONCLUSÃO

É possível codificar um processador SAR *Range-Doppler* em tempo real utilizando-se de um computador de uso geral. Pelos resultados alcançados neste trabalho, para o cenário considerado, percebe-se que há uma margem de quase 40% entre o tempo limite para processamento e o tempo alcançado.

Vale ressaltar ainda que o computador utilizado não está entre os computadores pessoais com maior desempenho da atualidade e que computadores com mais poder de processamento não possuem um custo muito mais elevado, o que reforça ainda mais a proposta que um sistema completo para processamento SAR em tempo real se faz possível com o uso de um PC.

Diferentemente das implementações utilizando FPGAs e DSPs, utilizando-se um PC para realizar o processamento SAR, trabalha-se com a vantagem ter um Sistema Operacional executando, o qual trás ferramentas gráficas e bibliotecas prontas. Além disto, o desenvolvimento se dá numa arquitetura amigável, sem necessidade de importação de componentes eletrônicos, fabricação e montagem de placas de circuito impresso, simplicidade na substituição de equipamento avariado, facilidade na atualização de hardware e software, etc.

Vale ressaltar que o uso de um PC em ambiente aeronáutico ou espacial implica em certificação. Este requisito encarecerá a solução, mas um estudo de custo/benefício pode ser feito comparando o tempo e o custo do desenvolvimento de um hardware dedicado para uso espacial e/ou aeronáutico com o tempo e o custo de se desenvolver em um PC.

VI. AGRADECIMENTOS

Os autores agradecem à FINEP, que deu suporte a este estudo, por meio do projeto FINEP 0043/10 AEROSAR (ITA-FUNDEP).

REFERÊNCIAS

[1] Chen, L., Long, T., 2008, "The Research and Implementation of Spaceborne SAR Real-Time Quick Look Imaging System", Congress on Image and Signal Processing - CISP '08, Sanya, China, pp. 587-591.

[2] Sun, Z., Liu, X., Ji, Z., 2008, "The Design of SAR Signal Processor's Data Compress System in FPGA", International Symposium on Intelligent Information Technology Application Workshops – IITAW '08, Shanghai, China, pp. 769-772.

[3] Dastgir, N., 2007, "Processing SAR Data using Range Doppler and Chirp Scaling Algorithms", Master's of Science Thesis in Geodesy Report No. 3096 – TRITA-GIT 07-005.

[4] Liu, J. et al, 2007, "Airborne C-SAR (Synthetic Aperture Radar) Real-time Imaging System", 1st Asian and Pacific Conference on Synthetic Aperture Radar – APSAR 2007, Huangshan, China, pp. 675-679.

[5] Franceschetti, G., et al, 1998, "A VLSI Architecture for Real Time Processing of One-bit Coded SAR Signals", URSI International Symposium on Signals, Systems, and Eletrocnics – ISSSE 98, Pisa, Italy, pp. 282-286.

[6] Langemeyer, S., et al, 2005, "Architecture of a Flexible On-Board Real-Time SAR-Processor", IEEE International Geoscience and Remote Sensing Symposium – IGARSS '05, Vol. 3, Seoul, Korea, pp. 1746-1749.

[7] Zhihong, F., Jijin, X., 2009, "A Miniature Implementation of Air-born SAR Real-Time Processing", 2nd Asian-Pacific Conference on Synthetic Aperture Radar – APSAR 2009, Xian - Shaanxi, China, pp. 939-942.

[8] Guarita, F. C., 2011, "Avaliação da Arquitetura CUDA para Síntese de Imagens SAR", Dissertação de Mestrado em Engenharia Aeronáutica – Instituto Tecnológico de Aeronáutica, São José dos Campos, Brazil.

[9] Ning, X., et al, 2011, "Multiple-GPU Accelerated Range-Doppler Algorithm for Synthetic Aperture Radar Imaging", IEEE Radar Conference - RADAR, Kansas City – MO, USA, pp. 698-701.

[10] Cumming, I.G., Bennett, J., 1979, "Digital Processing of Seasat SAR Data", IEEE International Conference on Acoustics, Speech and Signal Processing – ICASSP 1979, Richmond, B.C., Canada, Vol. 4, pp. 710-718.

[11] Franceschetti, G.; Lanari, R. Synthetic aperture radar processing. New York: CRC Press, 1999. ISBN: 0849378990 ISBN-13: 9780849378997.

[12] Desai, N. M., et al, 2008, "Near Real Time SAR Processors for ISRO's Multi-Mode RISAT-I and DMSAR", 7th European Conference on Synthetic Aperture Radar - EUSAR, Friedrichshafen, Germany, pp. 1-4.

[13] Su, T., He, X., Wu, S., 2006, "Opttimizing and Implementing the Fast Algorithm for Real Time SAR Imaging", International Conference on Radar – CIE '06, Shanghai, China, pp. 1-4.

[14] Cumming, I. G.; Wong, F. H. Digital Processing of synthetic aperture radar data: algorithms and implementations. Norwood: Artech House, 2005. ISBN: 1-58053-058-3.

[15] Simon-Klar, C., et al, 2002, "A Multi DSP Board for Real Time SAR Processing using the HiPAR-DSP 16", IEEE International Geoscience and Remote Sensing Symposium – IGARSS '02, Vol. 5, Toronto, Canada, pp. 2750-2752.

[16] Friebe, L., et al, 2001, "A Compact Real-Time SAR Processing System using the highly Parallel HiPAR-DSP 16", IEEE International Geoscience and Remote Sensing Symposium – IGARSS '01, Vol. 5, Sydney, Australia, pp. 2307-2309.

[17] Frigo, M., Johnson, S.G., 1998, "FFTW: an adaptive software architecture for the FFT", IEEE International Conference on Acoustics, Speech and Signal Processing, Seattle, WA, USA, Vol. 3, pp. 1381-1384.