dx.doi.org/10.7437/NT2236-7640.2011.01.002
Notas Técnicas, v. 1, n. 1, 2011

Desenvolvimento de Instrumentos Científicos em Hardware Reconfigurável.

M. Bochner,* Marcelo P. de Albuquerque, L.C. Resende, and R.A.A. Nunes
Coordenação de Atividades Técnicas (CAT), Centro Brasileiro de Pesquisas Físicas (CBPF), Rua Dr. Xavier Sigaud, 150, Urca, Rio de Janeiro, Brasil

J.L.G. Alfonso
Instituto de Física da Universidade Federal do Espírito Santos (UFES), Av. Fernando Ferrari 514, Goiabeiras, Vitória

Resumo:
Esta Nota Técnica apresenta as principais características da computação por hardware reconfigurável incentivando a adoção de FPGAs, em especial da família Stratix II disponível no KIT de desenvolvimento da ALTERA EP2S60, para prototipagem do instrumento. É apresentada também a forma de utilização deste kit da ALTERA utilizando blocos de funções para prototipagem, por meio do software Quartus II e funções MEX. Por fim é apresentada uma implementação na FPGA de um método de detecção sensível a fase (PSD – *Phase Sensitivity Detection*), com a finalidade de introduzir sua aplicação em medidas de módulo e fase entre dois sinais.

Abstract:
This Technical Note presents the main characteristics of computation based on reconfigurable hardware, encouraging the adoption of FPGAs, especially Stratix II family which is available in ALTERA EP2S60 development Kit, for the prototyping of the instrument. It is also shown how to use this ALTERA Kit by using function block for prototyping via the Quartus II software and MEX function. Finally, we present an implementation on FPGA of a Phase Sensitivity Detection (PSD) method, aiming to introduce its application in magnitude and phase measures between two signals.

Keywords: Computação Reconfigurável, Lock-in, Detecção sensível à fase (PSD), Processamento digital de sinais.

1. COMPUTAÇÃO RECONFIGURÁVEL

Dispositivos que executem grande quantidade de funções diferentes podem ser caracterizados como de uso geral. Os processadores Pentium da INTEL ou Power PC da Motorola são exemplos de microprocessadores de uso geral, encontrados na maioria dos computadores pessoais que podem executar um grande número de funções. Entretanto, quanto ao desempenho, a velocidade desses dispositivos pode ser lenta comparada aos dispositivos com aplicações específicas (ASIC) que executam apenas um conjunto limitado de instruções, atingindo melhores resultados com menos recursos de hardware, mas inflexíveis quanto a modificação das suas funções depois de implementadas no "chip".

Uma terceira opção, com grande capacidade e velocidade e que permite modificação do hardware durante o uso são as FPGAs (*Field Programmable gate Array*), que consistem de um *array* de blocos lógicos configuráveis cujas funções e conexões entre os blocos podem ser alterados através de sinais de configuração enviados ao dispositivo.

A computação reconfigurável [1] [2] [3], ao invés de computar uma função seqüencial no tempo, computa paralelamente através de blocos funcionais configurados no espaço.

1.1. Principais características do Hardware Reconfigurável

- Capacidade de se mudar o circuito dentro de um *chip*.
- A Computação Reconfigurável (CR) preenche o *gap* entre hardware e software, alcançando mais performance enquanto mantêm um alto grau de flexibilidade (Figura 1 e Figura 2).

2. POR QUE USAR FPGA?

Os algoritmos para Processamento de Sinais Digitais ("DSP") têm sido tradicionalmente implementados utilizando-se circuitos integrados dedicados ("ASIC) ou Processadores Digitais Programáveis ("DSP – *Digital Signal Processors* ") [4]. Porém, com a introdução das "FPGAs" houve um significativo avanço em direção a computação reconfigurável para processamento de sinais digitais. O paralelismo dos dados encontrado nas funções de processamento de sinais associado ao paralelismo do hardware nas FPGAs, oferece uma alternativa que combina funcionalidade do hardware fixo e a flexibilidade dos dispositivos programáveis por software.

Ao contrário dos Processadores Digitais Programáveis, as FPGAs permitem tamanho da palavra não padronizado e total ou parcial processamento paralelo promovendo assim redução no espaço de memória ocupada e melhoria na transmissão dos dados. Adicionalmente, plataformas de emulação podem oferecer prototipação em tempo-real de lógica que permite a depuração e otimização do sistema em desenvolvimento em um ambiente semelhante ao dispositivo alvo.

O fluxo de desenvolvimento da FPGA é um processo interativo e requer muitas etapas intermediárias. Existem diversas ferramentas de automação de projetos (EDA – *electronic design automation*) para projetos com FPGA que são oferecidos tanto pelos fornecedores da FPGA como fornecedores independentes. Ambientes de desenvolvimento completos são oferecidos pela Xilinx (ISE), Altera (Quartus II) e Mentor Graphics (FPGA Advantage), que se integram com

*Electronic address: mauricio@cbpf.br

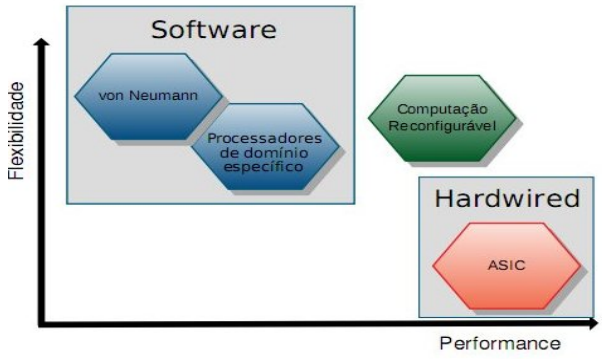


Figura 1: Gráfico mostrando a posição da computação reconfigurável na contradição entre desempenho e flexibilidade.

Figura 2: Computação reconfigurável combina vantagens principais de computadores de uso geral e de circuitos integrados orientados à aplicação (ASIC) e permite eliminar as suas desvantagens principais.

algumas ferramentas EDA de fornecedores independentes. De modo a reduzir o tempo de desenvolvimento, o projeto com DSPs tem girado em torno do uso de núcleos de propriedade intelectual (IP – *Intelectual Property*) e modelos customizáveis para funções DSP básicas.

Uma tendência entre fornecedores como Xilinx, Altera, AccelChip e Synplicity, são soluções de desenvolvimento baseadas na integração Matlab/Simulink que contêm bibliotecas de blocos DSP parametrizáveis, em ponto fixo. Isto habilita o projetista a integrar o seu modelo em um sistema completo que permite uma verificação dinâmica.

3. SOLUÇÕES EM KIT DE DESENVOLVIMENTO

A ALTERA fornece uma Ferramenta de desenvolvimento de DSP (digital signal processing) que conecta o Quartus II e as ferramentas MATLAB/SIMULINK (Math-works). Encurta ciclos do projeto de DSP, ajuda a criar representação de hardware de um projeto DSP em um ambiente de desenvolvimento algoritmo-amigável. As funções do MATLAB e os blocos do Simulink podem ser combinados com blocos do DSP Builder e funções de MegaCore do IP (propriedade intelectual). Ferramentas de desenvolvimento da Altera, incluindo o DSP Builder, SOPC Builder, e Quartus II, fornecem uma plataforma detalhada do projeto. Permite o aproveitamento dos benefícios da lógica programável, ao construir um co-processor FPGA (para acelerar a performance de um outro sistema) ou uma arquitetura de hardware DSP dedicada (Figura 3).

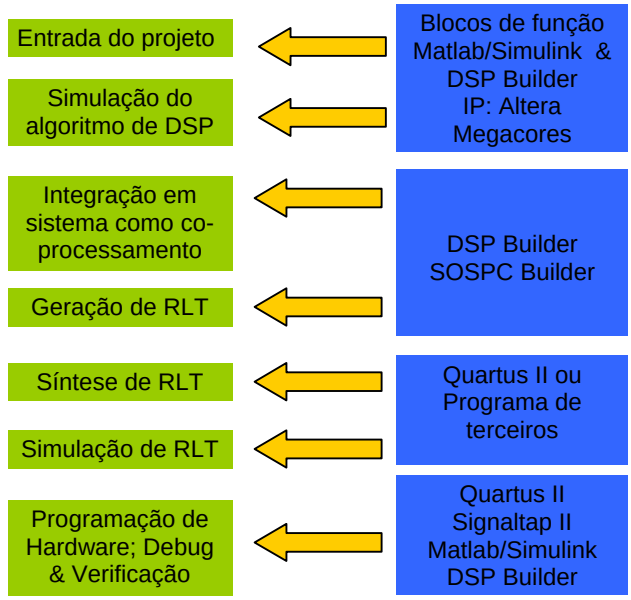


Figura 3: Visão geral do Fluxo de design DSP (Prototipagem).

4. FUNDAMENTOS DA UTILIZAÇÃO DO KIT ALTERA EP2S60

Para obter os benefícios citados, sugerimos o KIT ALTERA EP2S60, com a FPGA Stratix II. Além de funções lógicas também dispõe de funções para processamento digital de sinais (DSP) em uma funcional placa de desenvolvimento da ALTERA que disponibiliza hardware suficiente para se implementar um protótipo e softwares em uma funcional IDE (*Integrated Development Envirnment*).

4.1. Prototipagem Utilizando Blocos de Funções

A programação para o kit EP2S60 utiliza o MATLAB, Simulink e a biblioteca de funções da Altera para DSPs [5]. Essa biblioteca utilizada para criar modelos no Simulink foi

instalada a partir do software DSPBuilder, permitindo que um programa utilize os blocos programáveis compatíveis com o kit EP2S60 por meio de um subgrupo específico para o DSP em questão (Figura 4).

Os blocos têm variáveis específicas próprias, não sendo compatíveis com a maioria dos outros blocos do Simulink.

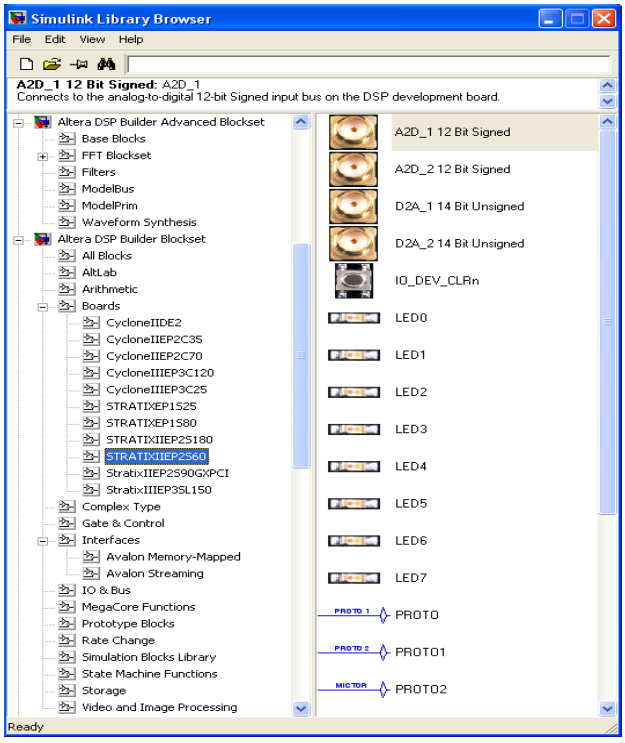


Figura 4: Biblioteca da Altera para uso de diversos DSPs, inclusive do EP2S60, instalada a partir do DSPBuilder e funções do MegaCore.

Possuindo uma cor azul, se destacam das outras funções disponíveis no MATLAB (Figura 5). Estes blocos trabalham em conjunto com o compilador Quartus II que é apresentado com mais detalhes a seguir.

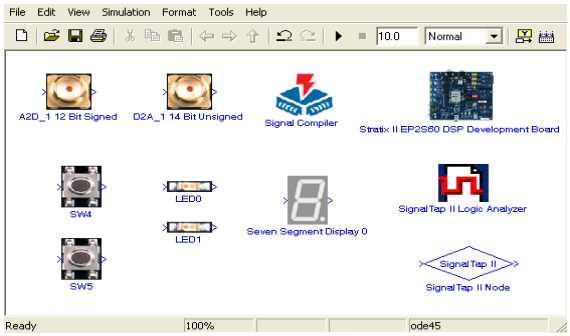


Figura 5: Principais blocos utilizados na programação do kit EP2S60.

Ao abrir a biblioteca instalada pelo DSPBuilder, podemos escolher o modelo de nosso kit e arrastar para dentro do arquivo do Simulink os blocos desejados. Desta forma conseguimos montar facilmente algoritmos que se tornam visualmente fáceis de entender, pois dentro desta biblioteca encontram-se as principais funções da placa, como os LEDs, os botões de interrupção e os conectores para os conversores AD e DA.

Uma importante função utilizada na programação do DSP EP2S60 dentro do ambiente do Simulink, é o SignalTap. Esta ferramenta trabalha em conjunto com o compilador Quartus II para capturar dados em diversos pontos do modelo trabalhado. O SignalTap® II Embedded Logic Analyzer é, como o próprio nome já sugere, um analisador lógico que ajuda a estratificar o design feito na FPGA, capturando sinais e dados internos do modelo feito no Simulink, sem que haja a necessidade de equipamentos externos de medição nem a utilização de pinos I/O no circuito do DSP.

Uma das vantagens de se utilizar o SignalTap é a possibilidade de se capturar dados a partir de diversos clocks diferentes ao mesmo tempo. Ao possuir uma interface rápida e fácil com o MATLAB/Simulink, o SignalTap se tornou uma importante ferramenta para captura dos sinais neste projeto. Ao adicionarmos um "node" dentro do modelo a ser trabalhado, podemos configurá-lo de forma a nos fornecer os dados que passam por aquele nó específico do circuito. Podemos adicionar quantos "nodes" forem necessários ao projeto, a fim de obter todos os sinais desejados (Figura 6).

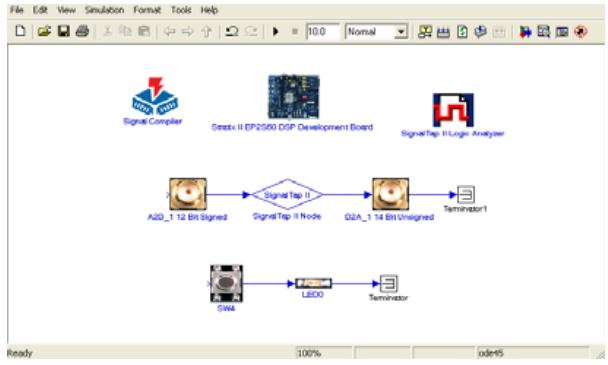


Figura 6: Configuração básica de demonstração de como utilizar os blocos de programação do kit de desenvolvimento. O sinal entra no conversor A2D de 12 bits, passando pelo "node" de leitura de sinal do SignalTap e retorna para a saída pelo conversor D2A de 14 bits.

O *SignalTap* necessita de um bloco de ativação chamada de *SignalTap Logic Analyzer*, que faz a comunicação de cada node com o *Quartus* II, direcionando e agrupando cada bit do sinal analisado em uma única variável. Uma vez posicionado dentro do design, este "node" agora possui uma seqüência de numeração, de acordo com a ordem o qual foi alocado, conforme Figura 7. Dentro do bloco de compilação do design, o *SignalCompiler*, podemos configurar a quantidade de amostras que desejamos capturar.

Detalhes sobre o *SignalCompiler* são mostrados na Figura 8. Uma vez feita a configuração e ativação do *SignalTap*,

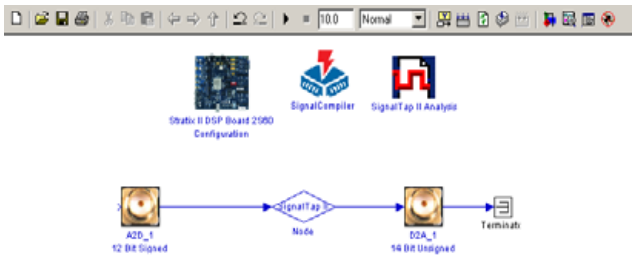


Figura 7: Esquemático mostrando o posicionamento de um "node" dentro do design feito no Simulink.

compilamos o projeto, gerando os arquivos a serem utilizados dentro do *Quartus* II para captura dos sinais.

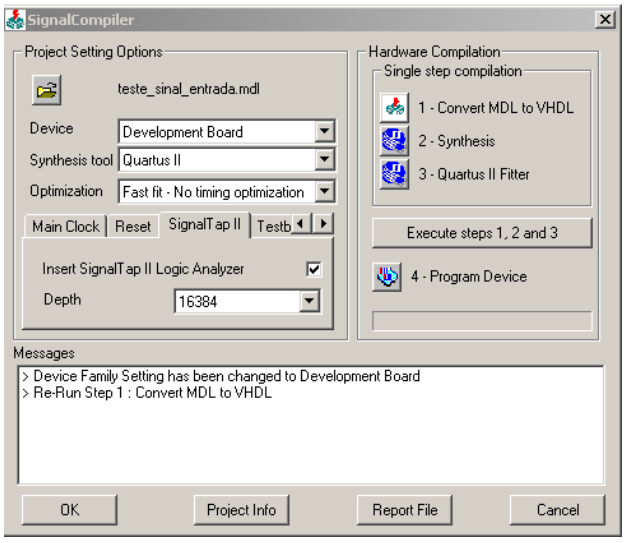


Figura 8: Janela do SignalCompiler dentro do Simulink, mostrando a opção de escolha da quantidade de amostras a serem capturadas pelos "nodes".

4.2. QUARTUS II

O Quartus II é um ambiente de desenvolvimento integrado (IDE) feito para trabalhar com os DSPs da Altera. Este compilador é muito poderoso, fornecendo ao usuário a possibilidade de programar cada função dentro da FPGA, pino a pino, posição a posição da memória. Ao iniciar o programa, a primeira etapa a ser feita é abrir o projeto criado através do *SignalCompiler*, dentro do *Simulink*. Selecionamos o arquivo de extensão .QPF" (*Quartus Project File*). Este arquivo é indexado a todos os arquivos secundários necessários para a programação e captura dos sinais no DSP. Segue a seguir uma lista de todos os arquivos utilizados na programação do kit EP2S60 e sua funcionalidade:

- ".QPF" – arquivo de projeto que indexa todos os arquivos da compilação
- ".STP" – arquivo de dados que direciona o posicionamento dos "nodes" dentro do design, fornecendo ao compilador a indicação da quantidade de bits de cada nó.
- ".SOF" – ponteiro para a interface de comunicação JTAG.

Ao carregarmos o projeto criado pelo SignalCompiler, dentro do Simulink, abre-se automaticamente o arquivo de dados do SignalTap (.STP). Após isso, escolhemos a interface JTAG que iremos utilizar, conforme mostrado na Figura 9, e carregamos o arquivo .SOF correspondente, que também fora gerado anteriormente.

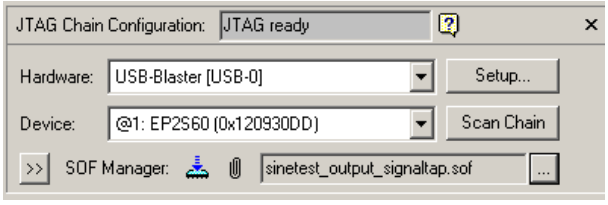


Figura 9: Carregando o arquivo .sof dentro do Quartus II.

O programa automaticamente reconhece o dispositivo a ser programado e deixa-o previamente preparado para a programação no DSP. Sendo assim, podemos compilar o projeto e carregar o código (agora em formato VHDL) para a FPGA. Detalhes são mostrados na Figura 10.

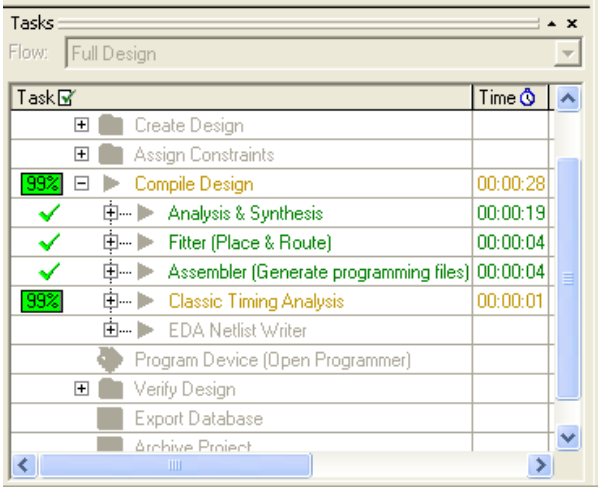


Figura 10: Janela de seqüência de compilação dentro do Quartus II.

Uma vez programado, podemos capturar os sinais correspondentes de cada "node". Após compilar e programar o kit podemos capturar os sinais de acordo com cada "node"

que posicionamos dentro do design. Para isso temos que primeiramente agrupar os bits de cada nó que desejamos fazer a leitura conforme a Figura 11.

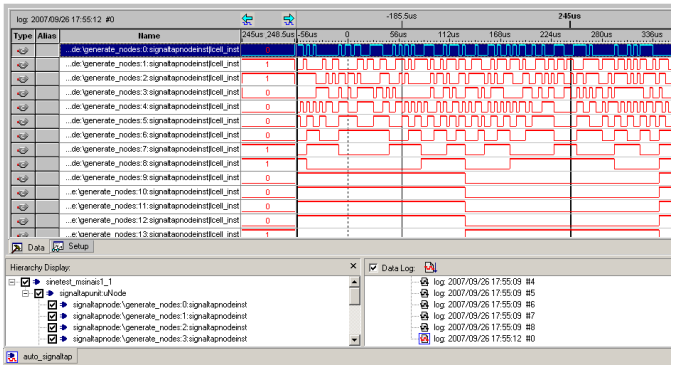


Figura 11: Janela dentro do Quartus II, mostrando os "nodes" com os bits sem agrupamento.

Agrupando todos os bits de cada "node" (Figura 12), criamos uma variável, que pode ser exportada para um arquivo texto, contendo todos os valores já convertidos para o formato *unsigned integer*. A figura 13 mostra um exemplo de um arquivo texto gerado. Com estes valores em mão podemos criar uma variável no Matlab ou em qualquer editor de planilhas, como o Excel, e manipular os dados da forma mais conveniente.

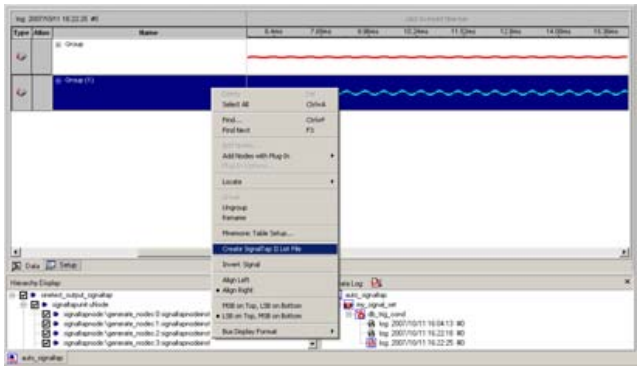


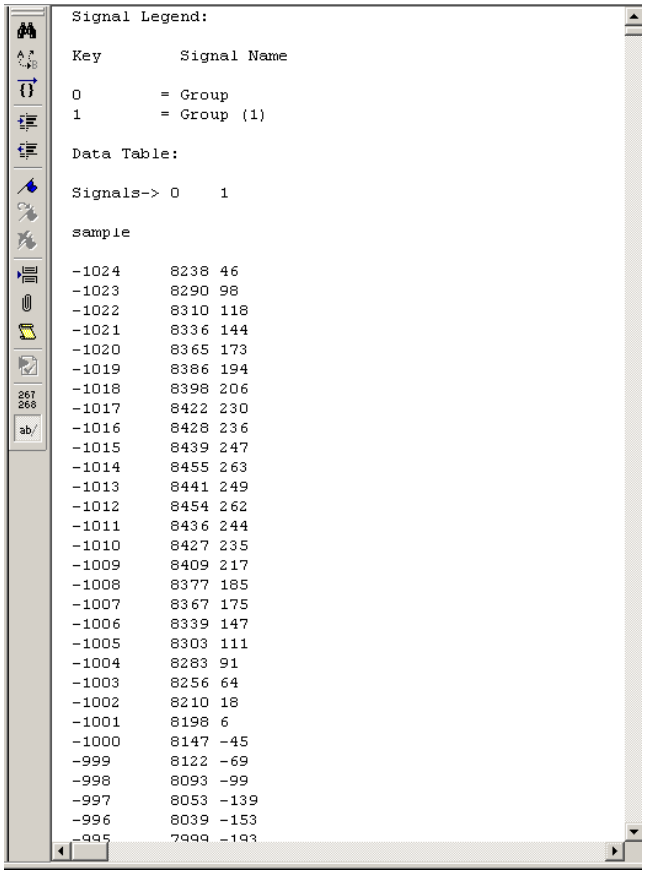
Figura 12: Janela dentro do Quartus II, mostrando os "nodes" agrupados.

4.3. A "MEX-Function"

Uma forma de se evitar este trabalho, que pode se tornar demorado, de acordo com a profundidade de cada grupo (podendo conter até 16 mil pontos), existe uma função que pode ser rodada dentro do MATLAB, na Command Window ou em um arquivo ".m", que se chama MEX-Function. Esta função é instalada dentro do Matlab e trabalha exclusivamente com o SignalTap e o Quartus II. Ao compilarmos o programa dentro do DSP, basta agrupar os "nodes" uma única vez.

Salvando o arquivo .stp alterado (já com os grupos formados), podemos rodar esta MEX-Function, e "chamar" o arquivo .stp diretamente de dentro do MATLAB, sem precisar criar uma tabela com os valores de cada grupo. A função é descrita a seguir:

>>var_signal=alt_signaltap_run('arquivo.stp');



```
Signal Legend:

Key         Signal Name

0         = Group
1         = Group (1)

Data Table:

Signals-> 0    1

sample

-1024     8238 46
-1023     8290 98
-1022     8310 118
-1021     8336 144
-1020     8365 173
-1019     8386 194
-1018     8398 206
-1017     8422 230
-1016     8428 236
-1015     8439 247
-1014     8455 263
-1013     8441 249
-1012     8454 262
-1011     8436 244
-1010     8427 235
-1009     8409 217
-1008     8377 185
-1007     8367 175
-1006     8339 147
-1005     8303 111
-1004     8283 91
-1003     8256 64
-1002     8210 18
-1001     8198 6
-1000     8147 -45
-999      8122 -69
-998      8093 -99
-997      8053 -139
-996      8039 -153
-995      7999 -193
```

Figura 13: Signal Tap List File criado a partir do agrupamento dos bits dos "nodes". A primeira coluna representa o número da amostra. A segunda e terceira colunas representam o valor de cada amostra para o node 0 e node 1 respectivamente.

A variável criada "var_signal" contém n colunas, onde cada coluna representa um agrupamento de bits de cada "node" utilizado dentro do design, ou seja, se utilizamos 4 "nodes" do SignalTap em nosso design, o arquivo conterá 4 colunas. Esta variável ainda conta com m linhas, onde cada linha representa uma amostra capturada. Ou seja, se escolhemos uma precisão de 16384 pontos, a n-ésima coluna conterá 16384 linhas.

5. IMPLEMENTAÇÃO NA FPGA, DE UM PSD (SISTEMA DE DETECÇÃO SENSÍVEL À FASE).

A Figura 14 apresenta o circuito PSD (Sistema de Detecção Sensível a Fase) em quadratura, núcleo do amplificador Lock-In e outros instrumentos de medição síncronos, utilizando o Simulink e as bibliotecas da Altera.

Os conversores A2D_1 e A2D_2 são a entrada do sinal a ser medido *Vin = Acos(wt + θ)* e do sinal de referência *Vref = Acos(wt)* respectivamente. O sinal de referência é defasado através do módulo Delay que atrasa o sinal em uma determinada quantidade de amostragens, de acordo com a frequência deste sinal, para que seja introduzida a defasagem de 90º (Figura 15) necessária à demodulação em quadratura. Os produtos dos sinais em fase e quadratura obtidos (Figura 16 e Figura 17), são então processados de modo a compor o PSD, *U1 e U2,* que corresponde matematicamente à correlação entre os sinais de entrada e referência conforme demonstrado nas equações [6]:

correlação em fase:

$$\Gamma(\varphi) = \frac{1}{T}\int_0^T Vref(t)Vin(t+\varphi)dt$$

$$\Gamma(\varphi) = \frac{AB}{T}\cos(\varphi) = U1$$

correlação em quadratura:

$$\Gamma(\varphi) = \frac{AB}{T}\int_0^T \cos(wt \pm 90º)\cos(wt+\varphi)dt$$

$$\Gamma(\varphi) = \frac{\pm AB}{2}\sin(\varphi) = U2$$

A recuperação do módulo e fase do sinal de entrada é feita de acordo com as equações:

$$Magnitude = \sqrt{U1^2 + U2^2} = \sqrt{(\frac{AB}{2})^2(\cos^2(\varphi) + \sin^2(\varphi))}$$

$$Magnitude = (\frac{A}{\sqrt{2}})(\frac{B}{\sqrt{2}}) = ArmsBrms$$

$$\tan(\varphi) = \frac{U2}{U1} = \frac{(\sin(\varphi))}{(\cos(\varphi))} = \tan(\varphi) \rightarrow \varphi = \arctan(\frac{U2}{U1})$$

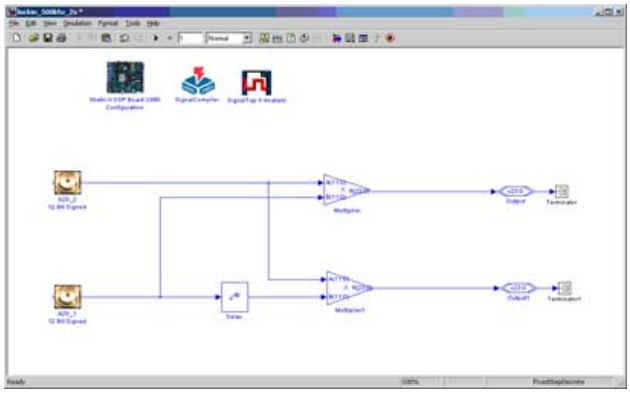


Figura 14: Circuito PSD (Sistema de Detecção Sensível a Fase) utilizando o Simulink e as bibliotecas da Altera.

As saídas podem ser capturadas via signaltapII sendo então processadas no computador ou podem ser configurados pinos de saída da FPGA (Figura 14).

Uma de nossas metas é a implementação das funções matemáticas acima todas em hardware reconfigurável e fornecer nos pinos de saída ou via signaltapII: módulo e fase do sinal de entrada e disponibilizar também *U1 e U2* para processamento externo.

As figuras 15, 16 e 17 foram capturadas em tempo real via interface JTAG, utilizando o recurso signaltapII. O sinal aplicado em fase com a referência. Uma vez que os sinais de entrada e referência estão em fase, os resultados apresentaram-se conforme esperado: o produto em fase sem projeção de componente negativa e o produto em quadratura simetricamente sobre o eixo, sem componente CC.

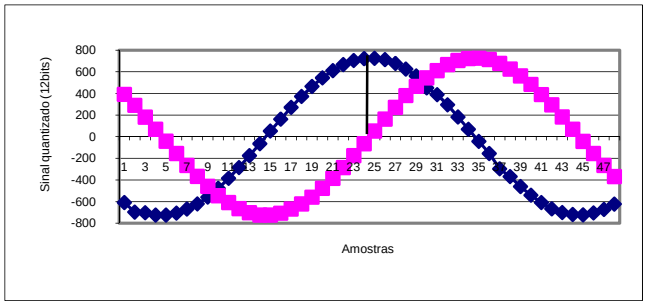


Figura 15: Sinal de referência e sinal de referência gerado com defasagem de 90^0.

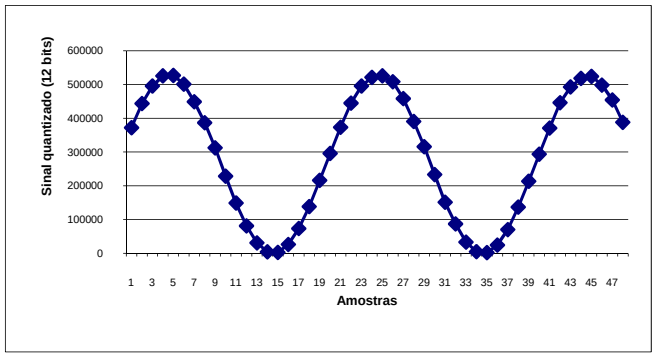


Figura 16: Produto dos sinais de entrada e referencia estando ambos em fase.

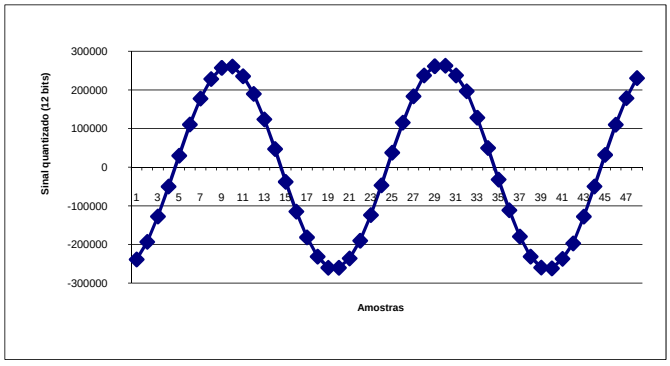


Figura 17: Produto dos sinais de entrada e referencia em quadratura.

5.1. Aplicação

O processamento do PSD foi utilizado para efetuar medidas experimentais de magnitude e fase.

Com o sistema PSD (Figura 14) implementado na FPGA, aplicamos um sinal, também implementado em hardware (Figura 18), de 200mV defasado de 28.8^0 da referência. O sinal foi criado através de um gerador de rampa de endereços para uma LUT (*look up table*) onde são armazenados os valores das amostras do sinal gerado. O delay, para uma defasagem de 28.8^o é configurado de acordo com a frequência do sinal. O sinal gerado foi utilizado como referência (*Vref* = *Acos(wt)*) e o sinal com atraso foi utilizado como entrada (*Vin* = *Acos(wt* + θ*)*).

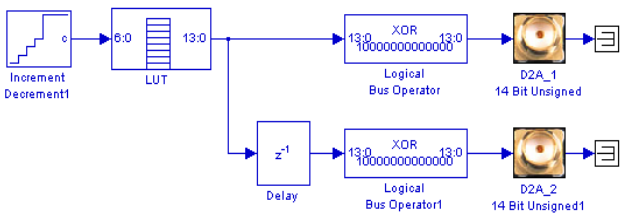


Figura 18: Implementação de um sinal digital com defasagem para as medidas de magnitude e fase. O delay (28.8) é configurado de acordo com a frequência.

O módulo XOR transforma os bits *signed* em bits *unsigned* que são reconhecidos pelos conversores D/A: D2A_1 e D2A_2 utilizados para visualização e depuração do sinal de referência e do sinal de entrada, respectivamente. Os resultado capturados via signaltapII e o processamento efetuado no Matlab, com constante de tempo de 8 ciclos são apresentados na Tabela 1.

Frequência (KHz)	Amostras por ciclo	Magnitude (V)	Fase (º)
500	200	0.20476 (±0.00007)	28.830 (±0.004)
1000	100	0.20350 (±0.00020)	28.821 (±0.006)
2000	50	0.19900 (±0.00100)	30.100 (±0.200)

Tabela 1 – Medidas de magnitude e fase onde é visto o aumento do desvio padrão em função da redução do número de amostras processadas.

6. CONCLUSÃO

Foram apresentadas as ferramentas e principais características da computação por hardware reconfigurável utilizando o KIT de desenvolvimento da ALTERA EP2S60, onde foi configurado na FPGA o hardware para implementação do método de detecção sensível a fase (PSD), que é o núcleo de inúmeros instrumentos de aplicação científica. Como uma primeira aplicação do protótipo (PSD), foram feitas medidas de magnitude e fase (Tabela 1) em sinais gerados pelo próprio sistema, com uma constante de tempo CT=8, ficando patente o hardware implementado na FPGA, minimizando circuitos externos. A precisão da leitura pode ser aumentada com o processamento em maiores constantes de tempo, visando compensar a redução do número de amostras nas frequências mais altas bem como maior atenuação de ruído para sinais aplicados.

[1] Francisco de Souza Junior, O que é Computação Reconfigurável, http://fsjunior.com.br/o-que-é-computação-reconfigurável, última Atualização: 10.01.2001.

[2] Iouliia Skliarova, António B. Ferrari, Introdução à computação reconfigurável, REVISTA DO DETUA, VOL. 2, Nº 6, SETEMBRO 2003, http://www.ieeta.pt/˜iouliia/Papers/2003/1_SF_ETSet2003.pdf

[3] Instituto de Ciências Aplicadas e de Computação - Universidade de São Paulo – São Carlos,SP, Computação Reconfigurável, http://www.icmc.usp.br/˜lcr/pt/comp_reconfiguravel.htm, August 3, 2009

[4] Kevin Banovic, Mohammed A. S. Khalid, and Esam Abdel-Raheem, FPGA-Based Rapid Prototyping of Digital Signal Processing SystemsResearch Centre for Integrated Microsystems Department of Electrical and Computer Engineering University of Windsor, Windsor, Ontario, Canada N9B 3P4 { banovic, mkhalid, eraheemg}@uwindsor.ca, 0-7803-9197-7/05/$20.00 ©2005 IEEE.

[5] Leonardo Rezende,Tese de Mestrado em Instrumentação Científica. Desenvolvimento de um Amplificador Lock-In com DSP operando em altas frequências, CBPF. 2009.

[6] J. G. Proakis and D. G. Manolakis. “Digital Signal Processing: Principles, Algorithms, and Applications”. 3a Edição; Prentice-Hall, ISBN 0133737624; 1996.