DOI: 10.5433/1679-0375.2011v32n1p115

Implementação eficiente de filtros adaptativos utilizando a plataforma TMS320C6713

Efficient implementation of adaptive filters using TMS320C6713 DSP platform

Diogo Kaoru Takayama[1] ; Rodger Vitória Pereira[2] ; Valter Luís Arlindo de Camargo[3] ; Taufik Abrão[4]

Resumo

Este artigo apresenta uma metodologia para o desenvolvimento acelerado de soluções em aplicações envolvendo filtragem adaptativa a partir do uso das tecnologias Matlab/Simulink, Code Composer Studio e do kit de desenvolvimento DSK 6713 para DSP. O propósito desta metodologia é fornecer um método eficiente e rápido para implementar e testar filtros adaptativos em DSPs. A metodologia apresentada aqui consiste de ferramentas essenciais ao projeto, simulação e implementação de filtros adaptativos. Outro benefício importante é que evita o trabalho de baixo nível em hardware que pode ser tedioso e consumir um tempo excessivo de projeto. Um exemplo de aplicação é apresentado neste artigo para demonstrar a factibilidade da metodologia proposta. Em seguida, a metodologia de projeto é empregada na obtenção de um filtro adaptativo em DSP. Descreve-se em detalhes a seqüência de etapas para esta implementação, incluindo o procedimento de transformação do código Matlab para o código DSP. Os resultados são então analisados em termos de precisão/exatidão e velocidade de convergência do filtro implementado. O kit de desenvolvimento TMS320C6713 da empresa Spectrum Digital Inc, o qual inclui um processador digital de sinais com operações em ponto flutuante, foi empregado na implementação do filtro adaptativo.

Palavras-chave: Filtragem adaptativa. Aplicações de tempo real. Implementação de filtros adaptativos em DSP. Simulink. DSK6713.

Abstract

This paper presents a methodology for accelerated development of solution associated to adaptive filtering using Matlab/Simulink, Code Composer Studio and DSK 6713 digital signal processing (DSP) technologies. The purpose of this methodology is to provide an efficient and rapid method to develop and test adaptive filters in DSPs. The methodology development represents a very important tool for the engineer in charge of design-simulation-implementation of adaptive filters. Another important benefit is that it avoids low level hardware work that can be tedious and time consuming. An example of application is presented in this paper in order to illustrate the feasibility of this methodology. The methodology is applied in order to implement an adaptive filter in the DSP platform. The project steps are discussed in details, including the methods of transforming the Matlab code into DSP code. The results are analyzed in terms of accuracy and convergence speed. The TMS320C6713 development kit supplied by Spectrum Digital Inc., that includes a float point DSP, was used to implement the adaptive filter.

Keywords: Adaptive filtering. Real-time applications. Adaptive filtering implementation in DSP. Simulink. DSK6713.

[1] Mestre pelo Programa de Mestrado Acadêmico em Engenharia Elétrica da Universidade Estadual de Londrina PPG-EE-UEL, Londrina, PR, 86051-990, Brasil, Fone: (55) 433371-4789; Fax: (55) 433371-4790

[2] Mestre pelo Programa de Mestrado Acadêmico em Engenharia Elétrica da Universidade Estadual de Londrina PPG-EE-UEL, Londrina, PR, 86051-990, Brasil, Fone: (55) 433371-4789; Fax: (55) 433371-4790

[3] Mestre pelo Programa de Mestrado Acadêmico em Engenharia Elétrica da Universidade Estadual de Londrina PPG-EE-UEL, Londrina, PR, 86051-990, Brasil, Fone: (55) 433371-4789; Fax: (55) 433371-4790

[4] Professor Associado do Depto de Eng. Elétrica da Universidade Estudual de Londrina DEEL-UEL}. E-mail: taufik@uel.br taufik@pq.cnpq.br HP: www.uel.br/pessoal/taufik

Introdução

O processamento de sinais é essencial na implementação de funcionalidades de sistemas eletro-eletrônicos. Atualmente, a maior parte deste processamento tem sido realizado em sua versão digital justamente por suas inúmeras vantagens sobre o processamento analógico de sinais. Para isto, são empregados processadores com finalidade específica para o processamento digital de sinais, comumente conhecidos como DSP (digital signal processing). Os DSPs são organizados em plataformas capacidade de processamento da ordem de GFlops/seg e em expansão, com funcionalidades extras, tais como recursos adicionais de memória, entradas e saídas com conversores analógico-digital (ADC) e digital-analógico (DAC), respectivamente, cada vez mais rápidos e precisos etc.

O uso de DSPs pode ser encontrado em praticamente todas as áreas. A cada dia o mercado oferece um novo produto que utiliza esta tecnologia. O novo paradigma surgiu graças à evolução da micro-eletrônica, possibilitando aplicações com DSPs que antes eram caras e difíceis de implementar. Assim, o uso de DSPs para aplicações de consumo em massa é uma realidade.

Sistemas eletro-eletrônicos que utilizam processamento de sinais frequentemente necessitam de soluções adaptativas. Uma primeira abordagem não sistemática na busca de soluções pode ser obtida via simulação. As simulações são feitas, na maioria das vezes, via tentativa e erro, onde se espera identificar um algoritmo que resolva o problema sugerido com certa qualidade. Os tempos envolvidos em simulações computacionais empregando-se simuladores matemáticos, tais como o Matlab, são algumas vezes extraordinariamente longos (KEHTARNAVAZ; KIM; PANAHI, 2004). Quando o algoritmo é destinado a aplicações de tempo real não se pode dispor de tais tempos de processamento. Assim, um compromisso entre desempenho e custo computacional deve ser atingido. Na maioria das vezes as soluções são implementadas em processadores digitais de sinais com finalidades específicas.

Com o uso cada vez maior dos DSPs em diversos campos de aplicações da eletrônica, a exigência por processadores digitais de sinais de alto desempenho tem aumentado rapidamente nos últimos anos. Muitas empresas estão atualmente desenvolvendo pesquisas em implementação de DSPs com alto poder de processamento para aplicações de tempo real.

O desenvolvimento de aplicações de tempo real difere significativamente dos outros tipos de aplicações (GAN, 2002), (SPANIAS et al., 2006), (GALANIS; PAPAZACHARIAS; ZIGOURIS, 2002), pois introduz novas exigências, especialmente relativas ao tempo de execução, as quais não podem ser efetivamente avaliadas nas simulações computacionais. O fator dominante para este tipo de aplicação é a restrição do tempo envolvido, de tal sorte que o processamento de uma amostra do sinal, via de regra, deva ser realizado antes da chegada de outra (GAN, 2002).

O desafio real no projeto de filtros adaptativos reside no algoritmo utilizado. Este deve possuir as seguintes características:

1. complexidade adequada, com a possibilidade de implementação em tempo real utilizando os DSPs atuais.

2. velocidade de convergência apreciável, i.e., capacidade de adaptação dos coeficientes em uma escala temporal compatível com as exigências da aplicação.

3. desempenho adequado.

Adicionalmente, em aplicações práticas, espera-se que o algoritmo empregado no projeto de filtros permita a rápida prototipagem, permitindo assim reduzir o tempo total de projeto (OROFINO, 2003), (GAN et al., 2000).

Na última década, inúmeras iniciativas visando a melhoria nas ferramentas e metodologia de projeto

empregando técnicas de processamento digital de sinais (DSP) têm sido feitas tanto por parte de fabricantes de processadores de sinais, quanto no ambiente da industria e no mundo acadêmico. Tal melhoria na metodologia de projeto tem sido atingida através da introdução de técnicas de projeto combinando e integrando diferentes conceitos, ambientes e ferramentas de projeto (TEXAS INSTRUMENTS, 2002a), (TEXAS INSTRUMENTS, 2002b), (WRIGHT; WELCH; GOMES, 1999), (FERZLI; KARAM, 2006), (CHACON; VALENZUELA, 2004), (GONZALEZ; WOODS, 2002).

Este trabalho está focado no desenvolvimento de uma metodologia de projeto e implementação de filtros adaptativos utilizando o Simulink/Matlab da Mathworks, o Code Composer Studio (CCS) e a plataforma de desenvolvimento DSK6713 da empresa Texas Instruments. Observe-se ainda que aplicações com processadores digitais de sinais (DSP) não se limitam a filtragem. A metodologia de projeto proposta permite é facilmente aplicável na análise de freqüências, geração de sinais, demodulação, detecção de sinais, entre outros. Uma metodologia de projeto similar a desenvolvida neste trabalho, porém voltada para aplicações de detecção de sinais em sistemas de comunicação de múltiplo acesso é reportada em (MUSSI; RIBEIRO; ABRÃO, 2010).

Este artigo está organizado da seguinte forma. Na seção II é feita uma revisão da literatura sobre filtragem adaptativa, princípios de processamento digital de sinais em tempo real, arquiteturas, algoritmos e considerações sobre implementações. Na seção III é discutida a metodologia de projeto proposta e revisados os blocos fundamentais de processamento em tempo real utilizados nesta metodologia. Tendo em vista comprovar a validade e eficiência da metodologia de projeto proposta, na seção IV é descrita a implementação de um filtro adaptativo digital utilizando a plataforma de hardware DSK6713, com avaliação dos resultados numéricos obtidos. Por fim, as principais conclusões deste trabalho são apresentadas na seção V.

Filtragem Adaptativa

Na filtragem convencional do tipo resposta impulsiva finita (FIR) ou do tipo resposta impulsiva infinita (IIR), geralmente assume-se que os parâmetros do processo estocástico de entrada são conhecidos, tais parâmetros determinam as características do sinal de entrada. No entanto, em muitos problemas práticos, pelo menos alguns parâmetros de entrada podem mudar, ou ainda, o sistema pode não conhecer exatamente o comportamento destes parâmetros. Nestes casos é altamente desejável um filtro que possua o atributo de auto-aprendizado, de maneira que este possa ajustar-se automaticamente de acordo com cada situação.

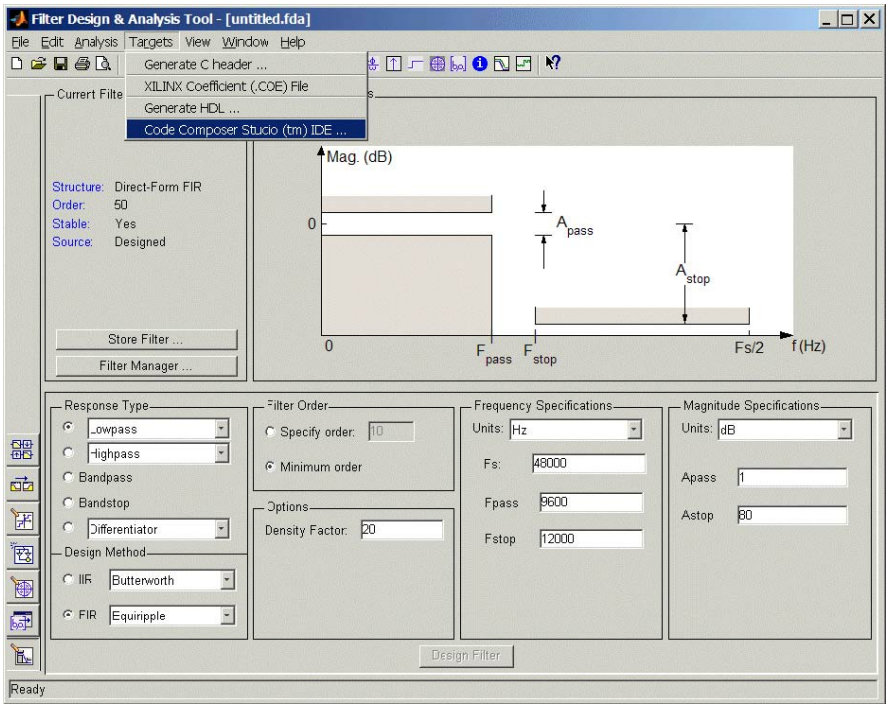
Filtros adaptativos são interessantes nos casos onde algumas das características da aplicação não são conhecidas. Desta maneira, pode-se dizer que os filtros adaptativos são mais adequados para aplicações em que as condições do sinal de entrada ou os parâmetros do sistema variam lentamente e o filtro seja capaz de se auto-ajustar para compensar essas mudanças. O algoritmo clássico LMS (Least Mean Squares) constitui uma das técnicas mais conhecidas para a implementação de um filtro adaptativo. O LMS utiliza um algoritmo de gradiente estocástico para adaptar a carga de um filtro. Esta adaptação consiste no ajuste contínuo (e adaptativo) dos valores dos coeficientes do filtro, tendo como métrica a minimização do erro no sentido da média dos quadrados.

Como exemplos de aplicação que requerem filtragem adaptativa, podemos citar: decodificação de canal de transmissão, detecção de sinais em presença de ruídos aleatórios, cancelamento de eco, reconhecimento de padrões de imagens e reconhecimento de padrões de voz, dentre outras.

Projeto de Filtros em Ambiente MatLab

O simulador matemático Matlab oferece uma série de pacotes e facilidades de programação em alto nível. Tais facilidades, distribuídas por diversos toolbox, auxiliam o projetista no desenvolvimento e testes dos algoritmos. Como exemplo, pode-se citar a interface gráfica para processamento digital de sinais (SPTool) e a ferramenta para o projeto de filtros digitais (FDATool), Figura 1. Estas são ferramentas poderosas para projetos, análise, quantização, teste e implementação de filtros, sendo empregadas na metodogia de projeto discutida nas seções subsequentes deste trabalho.

Figura 1. Tela do FDATool



A partir da comercialização da versão 7.0 do simulador matemático Matlab da Mathworks Inc. foi introduzida a biblioteca de interface para a plataforma de desenvolvimento em DSP denominada DSK6713 da Texas Instruments. Tal biblioteca permite a elaboração e desenvolvimento do bloco funcional no próprio simulador matemático e posteriormente a transferência do código para a plataforma DSP DSK6713. Por sua vez, existem muitos algoritmos para DSP disponíveis nos pacotes aplicativos (toolbox) do Matlab que podem servir de base para a concepção de outros. Para ilustrar a eficácia do método, nas seções III-H e IV é feita uma análise do projeto de um filtro adaptativo a partir do ambiente Matlab; neste caso, a programação em linguagem C é utilizada para implementar o algoritmo de processamento de digitais sinais no kit DSK6713.

Metodologia de Projeto

Nesta seção, descreve-se a metodologia de projeto de filtros adaptativos proposta neste trabalho. O processo geral de desenvolvimento de aplicativos em DSP é ilustrado na Figura 2. Os principais componentes deste processo são: DSK TMS320C6713, Matlab/Simulink e o Code Composer Studio (CCS).

Figura 2. Esquema do processo geral de desenvolvimento

Assim, o modelo do filtro a ser executado no DSP deve ser inicialmente implementado através dos diagramas de blocos do Simulink. O diagrama de blocos do Simulink é convertido para código em linguagem C utilizando o recurso Real-Time Workshop do Simulink. Este código C é integrado ao projeto CCS através de uma ligação do Matlab para o CCS. O software de edição Code Composer Studio compila, adiciona as bibliotecas necessárias e converte o código C em um arquivo executável que é então descarregado no módulo DSK6713. Neste ponto, o algoritmo está pronto para ser testado no DSK. Uma informação pode ser enviada ou recebida entre o Matlab e o DSK através dos canais RTDX. Em resumo, um processo típico de implementação de um algoritmo na plataforma DSK envolve os seguintes passos:

1. Construção do modelo do algoritmo no Simulink a ser convertido em código para a placa de desenvolvimento DSK.

2. Inclusão dos blocos específicos do kit para o modelo, tais como blocos ADC e DAC.

3. Configuração de cada um dos blocos com os parâmetros desejados.

4. Configuração das opções da placa de desenvolvimento, tais como mapa de segmentos de memória, alocação de área para código e dados e outras configurações necessárias.

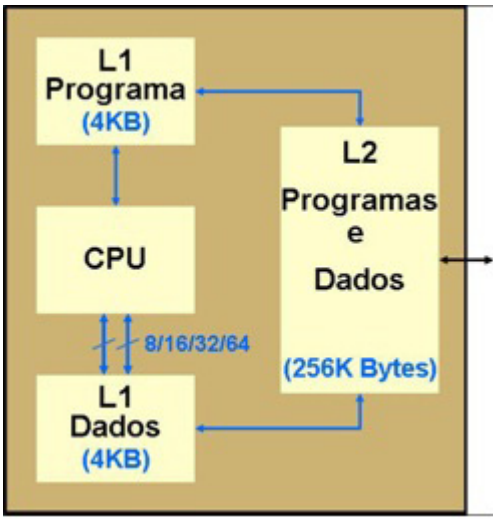
5. Envio do modelo para o Code Composer Studio.

A seguir são discutidas em detalhes as características dos diversos blocos componentes da metodologia de projeto.

TMS320C6713

O processador digital de sinais utilizado na implementação de filtros adaptativos é o TMS320C6713, fabricando pela Texas Instruments Inc., o qual possui arquitetura em ponto flutuante, capacidade de processamento simultâneo de 8 instruções a cada ciclo de relógio e desempenho equivalente a 1800 MIPS\ footnote{Milhões de informação por segundo.}. Ainda é capaz de realizar duas operações de multiplicação e acumulação (MAC) por ciclo de relógio. Também dispõe de 264kB de memória interna, das quais 4kB é alocada para a memória cache tipo L1P (Level 1 Program cache) e L1D (Level 1 data cache) e o restante é alocado para a memória denominada L2 que compartilha o espaço de programa e de dados, conforme esquematizado na Figura 3.

Figura 3. Memória Interna do DSP TMS320C6713



Kit de Desenvolvimento DSK6713

O kit DSK6713 constitui o hardware da plataforma escolhida para a execução das aplicações de processamento digitais de sinais em tempo real analisadas aqui. Trata-se de uma plataforma de desenvolvimento DSP produzida pela empresa Spectrum Digital Inc., conforme mostrado nas Figuras 4 e 5. A placa inclui um processador digital de sinais TMS320C6713 operando a uma frequência de relógio de 225 MHz, além de um codificador/decodificador (CODEC) com conversor analógico para digital (ADC) e digital para analógico (DAC) estéreos de 16 bits, cujas freqüências de amostragem variam entre 8 e 96 ksps.

A placa é normalmente conectada a um computador para o desenvolvimento de aplicações. A placa da plataforma original apresenta restrição de velocidade, que podem ser amenizada através da utilização de placas de expansão acopláveis à plataforma de desenvolvimento original de modo a possibilitar até mesmo o processamento de sinais de vídeo. O kit inclui um emulador JTAG, melhorando o desempenho de comunicação. A placa orginal contém 16MB de memória SDRAM e 256kB de memória flash. Também está disponível na própria placa original uma interface denominada EMIP (External memory interface), dois barramentos i2c (Inter-Integrated Circuit), dois temporizadores, uma GPIO (General Purpose Input and Output), uma interface HPI (Host-Port Interface), e um bloco de memória de acesso direto EDMA (Enhanced Direct Memory Access) com 16 canais independentes. A Figura 4 representa uma foto da placa original do kit DSK6713, na qual podem ser identificados os principais componentes e blocos funcionais descritos na Figura 5.

Figura 4. Foto do kit TMS320C6713 DSK - Spectrum Digital Inc.

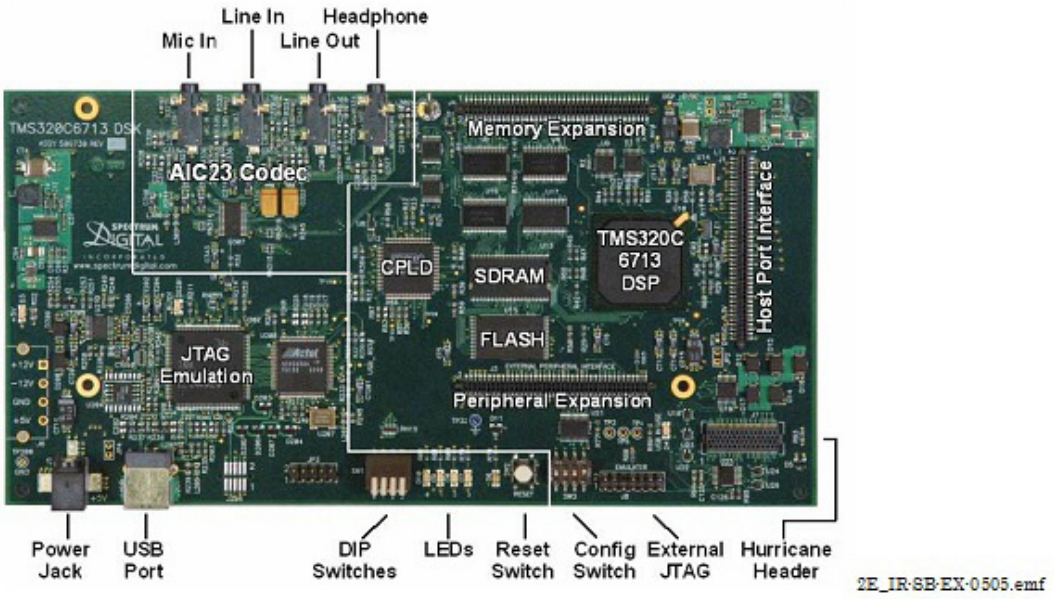
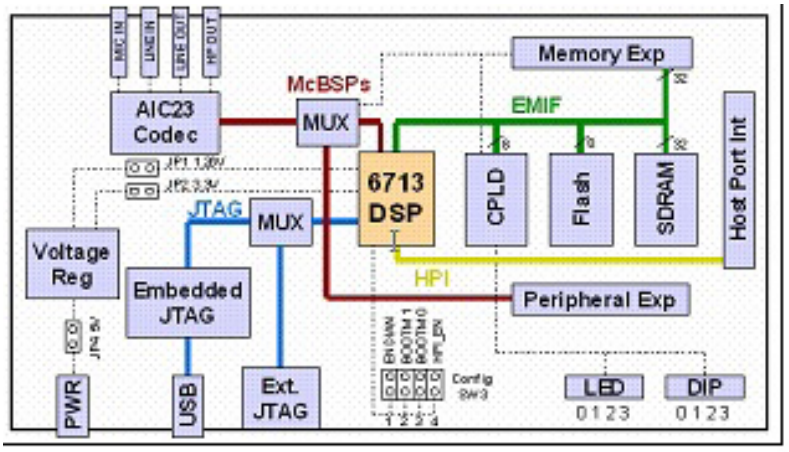


Figura 5. Diagrama de blocos do kit TMS320C6713 DSK



Code Composer Studio (CCS)

O CCS é um ambiente de desenvolvimento da Texas Instruments para os DSPs de sua fabricação. O CCS é utilizando como uma interface entre o Simulink e o DSK para testar novos algoritmos no DSK.

Simulink

O Simulink, que é uma extensão do simulador matemático Matlab, é utilizado no desenvolvimento de algoritmos para processamento digital de sinais. O Simulink utiliza a abordagem baseada em blocos para projeto e implementação de algoritmos. Um recurso denominado RTW (Real-Time Workshop) converte esses modelos do Simulink em código C/C++ ANSI que pode ser compilado pelo CCS.

O RTW amplia as capacidades do Simulink e automaticamente gera, empacota e compila o código fonte dos modelos do Simulink, tendo em vista a geração de softwares para aplicações em tempo real em vários tipos de processadores. O recurso RTW é a base para o processo de geração de códigos em um ambiente de projeto cuja principal característica consiste na rápida geração de códigos para prototipação e distribuição.

O Simulink disponibiliza uma série de blocos e bibliotecas denominadas ETTI (Embedded Target for Texas Instruments C6000 DSPs). A Figura 6 mostra alguns destes blocos. Estas bibliotecas fornecem as APIs (Application Program Interface) necessárias para o módulo Real-Time Workshop gerar o código especificamente para a plataforma C6000. Os blocos ETTI simplificam a tarefa de desenho e associação funcional entre blocos, já que representam os recursos disponíveis na placa DSK, tais como ADC, DAC, leds etc, conforme exemplo ilustrativo na Figura 7. Estes blocos podem ser utilizados diretamente na construção de uma aplicação. Na geração do arquivo executável, o processo de compilação de código disparado a partir de um comando no CCS. Este código pode então ser carregado na placa de DSP destino onde será executada. Os dados na placa de desenvolvimento são acessíveis no Matlab ou no CCS através de um link ou através do RTDX (Real-Time Data Transfer).

Figura 6. Opções de pacotes do RTW

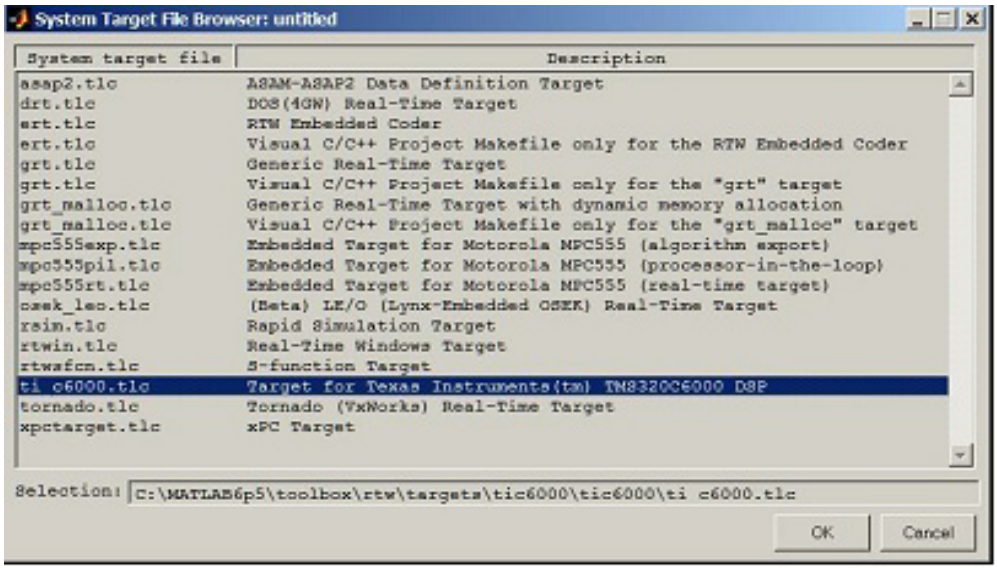
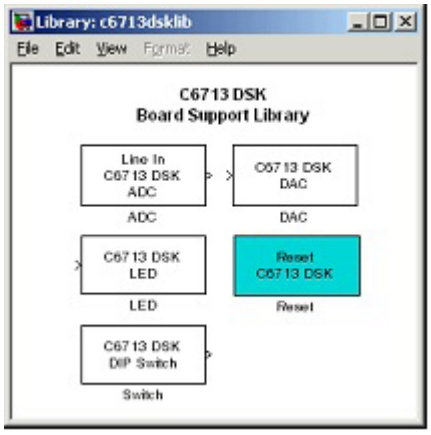
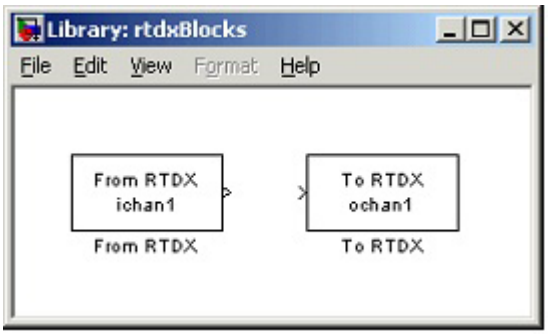


Figura 7. Alguns blocos ilustrativos da biblioteca C6713DSKLIB



Outros blocos incluídos no Simulink como parte do ETTI são os blocos RTDX Instrumentation. Estes blocos oferecem o recurso de envio e recebimento de dados do DSK. A Figura 8 mostra os blocos RTDX.

Figura 8. Biblioteca RTDX



Canais RTDX

A máxima taxa de transferência de dados possível de ser transferida para o DSK utilizando a porta paralela é de 64KBytes/seg, o que pode ser insuficiente para várias aplicações. Para aumentar a velocidade de transmissão é recomendável incluir um emulador JTAG. Um exemplo de código, escrito em linguagem C, para a comunicação entre o Matlab e o CCS através de um canal RTDX é mostrado no Algoritmo 1.

Algoritmo 1 - Comunicação RTDX

```
RTDX_CreateInputChannel(cin);
RTDX_CreateOutputChannel(cout);
...
TARGET_INITIALIZE();
RTDX_enableInput(&cin);
RTDX_enableOutput(&cout);
for(;;)
{
while(!RTDX_read(&cin, input, sizeof(input)));
// Rotina da filtragem aqui.
RTDX_write(&cout, output, sizeof(output));
}
```

No trecho de código descrito no Algoritmo 1, várias APIs (Application Program Interface) que fazem parte da biblioteca do CCS são utilizadas para fazer troca de dados. Primeiramente as APIs "RTDX_CreateInputChannel()" e "RTDX_CreateOutputChannel()" são utilizadas para declarar os canais que serão utilizados como entrada e como saída, respectivamente. Depois, o kit DSK é inicializado através de "TARGET_INITIALIZE()". Ambos os canais

RTDX são habilitados com o comando “RTDX_ enableInput()” e “RTDX_enableOutput()”. Para enviar dados do Matlab para o DSP, a função “RTDX_read()” é utilizada tendo como argumentos o canal, o vetor do sinal de entrada e tamanho do vetor. A função “RTDX_write()” é utilizada para enviar dados de volta para o Matlab.

TI C6000 - Code Composer Studio

Executar o código gerado pelo Real-Time Workshop para um ambiente de desenvolvimento em particular requer que o aplicativo gere o código adequado para a plataforma específica. O código específico inclui dispositivos de entrada e saída e rotinas de interrupção. Uma vez que estas características são particulares para cada plataforma, deve-se certificar que as configurações estejam corretas.

Após descarregar o programa executável gerado no kit DSK, o mesmo será executado inteiramente na placa da plataforma DSP. No entanto, é possível acessar o processo em execução através do depurador do CCS ou através de uma conexão RTDX.

Uma vez configurados todos os elementos que compõem o sistema, não se faz necessário nenhum outro código adicional.

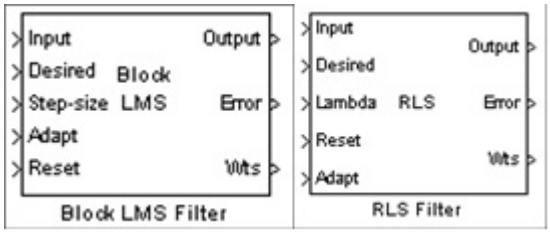
Módulo TI C6000 DSP

No âmbito do Simulink, a execução do módulo TI C6000 DSP permite o desenvolvimento de códigos específicos para os processadores da linha TMS320C6XXX. O Simulink é utilizado para modelar algoritmos de processamento digital de sinais a partir de blocos contidos na biblioteca Signal Processing Blockset e então utilizar o Real Time Workshop para gerar e construir códigos em linguagem C específicos para os kits de desenvolvimentos da Texas Instruments. O ambiente integrado de desenvolvimento CCS é utilizado na geração do arquivo executável e na depuração do programa. Após descarregar o código do programa para a placa, a aplicação é executada automaticamente.

Implementação de Filtros Adaptativos

O Simulink contém vários blocos destinados à filtragem adaptativa. Dentre eles, os blocos de algoritmo LMS, nLMS, RLS e KALMAN. A aparência dos blocos LMS e RLS pode ser observada na Figura 9. Como exemplo, o bloco LMS Filter implementa um filtro adaptativo do tipo LMS (least mean-square), onde a adaptação dos coeficientes dos filtros ocorre uma vez a cada novo bloco de amostras. O bloco estima os pesos dos coeficientes de maneira a minimizar o erro entre o sinal de saída y(n) e o sinal desejado d(n). O sinal que se deseja filtrar deverá ser conectado ao terminal Input. Este sinal de entrada pode ser um escalar baseado em amostragem ou um canal de dados. Conecte o sinal que deseja modelar no terminal Desired. O sinal desejado deve ter o mesmo tipo e dimensão do sinal de entrada. O terminal Output é onde é retirado o sinal filtrado. O terminal Error fornece o resultado da subtração do sinal de saída do sinal desejado. De forma análoga, o bloco RLS Filter do Simulink implementa um filtro RLS (recursive least squared), com a diferença que neste último, o parâmetro que define a velocidade de convergência é a entrada Lambda. Para maiores informações relativas aos blocos disponíveis no Simulink, consulte documentação online da Mathworks (MATHWORKS, 2011).

Figura 9. Blocos de filtragem adaptativa disponíveis no Simulink: a) LMS; b) RLS



Os principais parâmetros de projeto a serem considerados, especialmente o compromisso desempenho versus complexidade, quando se compara a eficiência dos algoritimos LMS, nLMS e RLS estão sintetizados na Tabela 1. Caso seja necessário manter o consumo de energia nos menores níveis possíveis e a aplicação não requer execução em tempo real, a melhor opção é a implementação de um filtro adaptativo LMS ou LMS normalizado (nLMS). Por outro lado, a melhor escolha para aplicações que requerem execução em tempo real e alta velocidade de convergência recai sobre o filtro adaptativo RLS.

Tabela 1. Principais parâmetros de desempenho dos algoritmos para filtragem adaptativa analisados

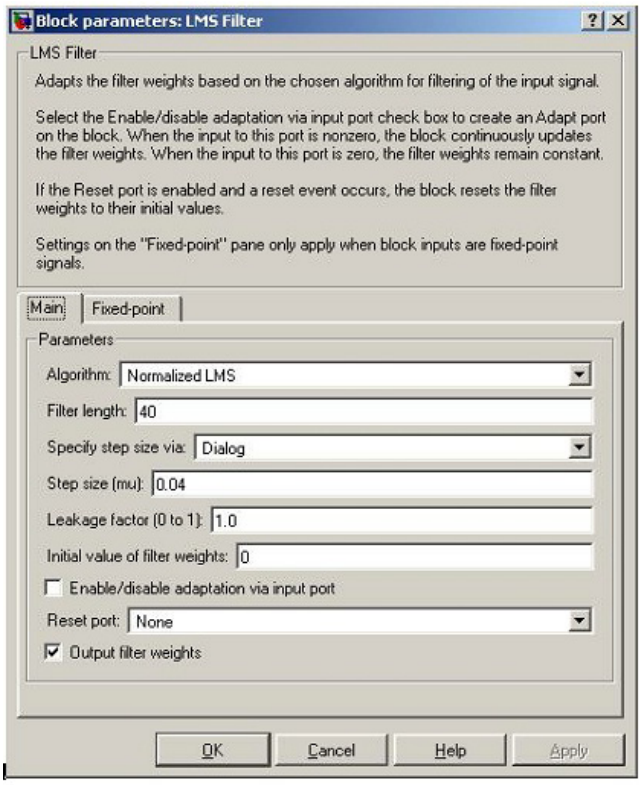
Parâmetro	LMS	nLMS	RLS
Convergência	Muito Lenta	Lenta	Rápida
Estabilidade	Muito Estável	Estável	Instável
Complexidade	Muito Baixa	Baixa	Alta
Consumo (MIPS)	Muito Baixo	Baixo	Alto
Implementação	Muito Simples	Simples	Média/Complexa

Exemplo de Projeto

Com a utilização dos blocos descritos na seção anterior e de outros eventualmente necessários, podem ser desenvolvidos algoritmos adaptativos para processos de identificação de sistemas, equalização de canal, supressão de interferências, entre outras aplicações.

Para efeito de aplicação da metodologia, considera-se nesta seção o projeto de um cancelador de ruído adaptativo utilizando o algoritmo LMS normalizado (nLMS), que se mostra bastante adequado para aplicações de tempo real, já que apresenta um ótimo compromisso entre desempenho e complexidade, conforme pode ser analisado na Tabela 1. Os parâmetros de configuração utilizados podem ser vistos na Figura 10. O objetivo deste exemplo de projeto é implementar em DSP um filtro adaptativo que elimine um sinal interferente de fundo de um outro sinal captado através de um microfone. Assim, serão enviados (e recebidos) sinais para o C6713 DSK. Foram utilizados os blocos C6713 DSK ADC e C6711 DSK DAC, para os sinais de entrada e saída respectivamente. Essencialmente, os blocos C6713 DSK ADC e C6713 DSK DAC adicionam um driver de software que controla o comportamento do CODEC.

Figura 10. Configuração do bloco nLMS a partir da tela de configuração de parâmetros do Simulink



Procedimentos para a Construção do Modelo

Primeiramente cria-se o modelo no Simulink conforme mostrado na Figura 11. A inserção dos blocos DSK6713 no modelo é obtida a partir do acesso aos blocos da biblioteca TI C6000 DSP. Esta biblioteca é composta por cinco blocos que são utilizados para controlar os diversos periféricos embutidos na placa DSK:

· Bloco de entrada (C6713 DSK ADC)

· Bloco de Saída (C6713 DSK DAC)

·Bloco de LEDs (C6713 DSK LED)

·Bloco de reset de software (Reset C6713 DSK)

·Bloco de chaves DIP (C6713 DSK DIP Switch)

Digitando "c6713dsklib" no prompt de comando do Matlab uma janela é aberta mostrando os blocos que compõem a biblioteca, conforme observado na Figura 12.

Figura 11. Modelo em Simulink para o cancelamento de ruído.

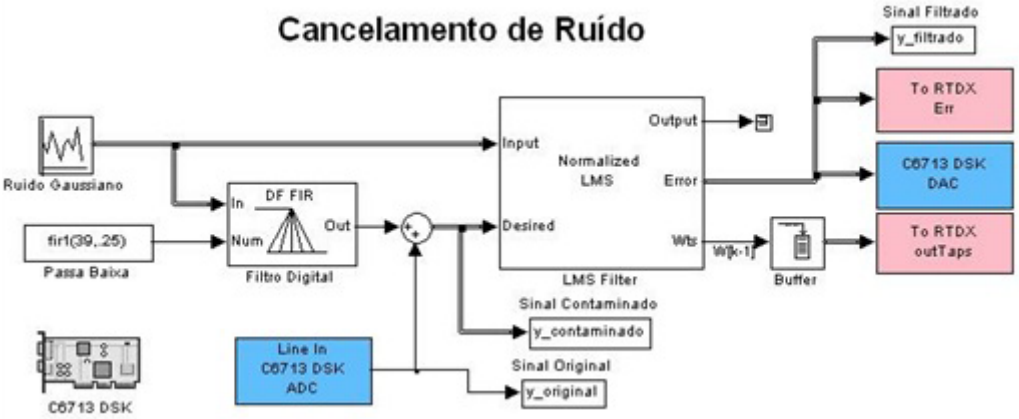
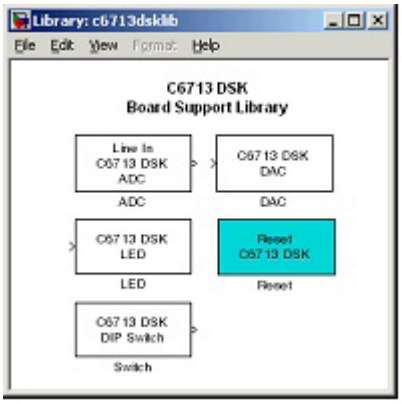


Figura 12. Biblioteca de Blocos para o kit DSK6713



Para descarregar e executar este modelo no kit DSK, basta seguir os seguintes passos:

1. Seleção dos blocos constituintes da biblioteca Signal Processing Blockset blocks e outros necessários à construção do modelo dentro do ambiente Simulink.

2. Adição de blocos da biblioteca TI C6000 DSP que permitem que sua fonte de sinal e saídas se comuniquem com o kit DSK 6713, que são C6713 DSK ADC e C6713 DSK, os quais podem ser encontrados na biblioteca de blocos TI C6000 DSP c6000lib. Os blocos disponíveis podem ser verificados clicando-se no ícone C6713 DSK Board Support.

3. Incorporação dos blocos C6713 DSK ADC e C6713 DSK DAC ao modelo, através da técnica de programação visual "Arrastar e Soltar".

4. Configuração dos parâmetros dos blocos especificamente para o kit de desenvolvimento (DSK) utilizado.

5. Configuração dos parâmetros para o modelo, incluindo os parâmetros do Solver, tais como tempo de início e parada e também as opções do Real-Time Workshop, tais como tipo de destino e opções de compilação.

6. Finalmente o modelo é testado no kit de desenvolvimento. O sinal a ser processado é injetado na entrada e observado através de um osciloscópio na saída da placa DSK.

Os blocos do CODEC podem ser configurados a partir das seguintes etapas:

1. Configuração das opções do bloco C6713 ADC:

a. Clique sobre o ícone C6713 DSK ADC para selecioná-lo.

b. Selecione Block Parameters a partir do barra de menus do Simulink.

c. Configure os parâmetros conforme mostrado na Figura 13.

d. Inclua um caminho do sinal diretamente da entrada para a saída de maneira que seja possível observar no osciloscópio ambos os sinais simultaneamente para comparação.

e. Clique OK para encerrar a caixa de diálogo.

2. Configuração das opções do bloco C6713 DSK DAC:

a. Clique sobre o ícone C6713 DSK DAC para selecioná-lo.

b. Selecione Block Parameters a partir do barra de menus do Simulink.

c. Configure os parâmetros conforme mostrado na Figura 14.

d. Clique OK para encerrar a caixa de diálogo.

Figura 13. Parâmetros do Bloco ADC.

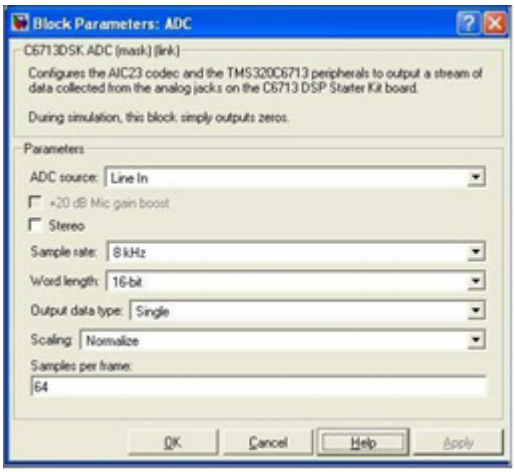
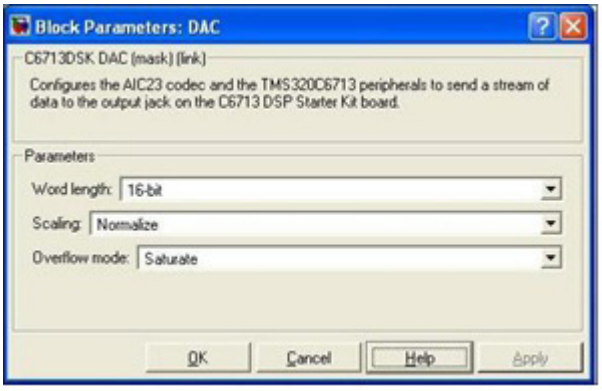


Figura 14. Configuração do bloco DAC



Resultados para o Exemplo de Projeto

O resultado da geração de código pode ser observado na Figura 15. Após compilação, o código executável é transferido para o kit DSK 6713 e então, inicia-se a etapa de testes elétricos propriamente dito. Para essa aplicação, foi conectado um microfone à entrada Mic In do kit DSK 6713, ao qual foi inserido um som de bateria acústica previamente gravado. A resposta foi observada simultaneamente no alto-falante conectado à saída Line Out do DSK e no ambiente de desenvolvimento. Ressalte-se que é possível também observar o sinal com a utilização do osciloscópio.

Resultados típicos para esse exemplo de projeto podem ser vistos nas Figuras 16 e 17. Verifique-se que após adaptação, Figura 16.c e 17, o sinal filtrado resultou muito próximo ao sinal original, demonstrado a eficácia do metodologia de projeto do filtro adaptativo em DSP. Com os parâmetros adotados neste exemplo de projeto, o tempo de convergência resultou aproximadamente 3 segundos e o erro quadrático médio entre o sinal original e o filtrado após convergência permaneceu no patamar de aproximadamente 23 dB.

Figura 15. Projeto e Código Gerado no Code Composer Studio

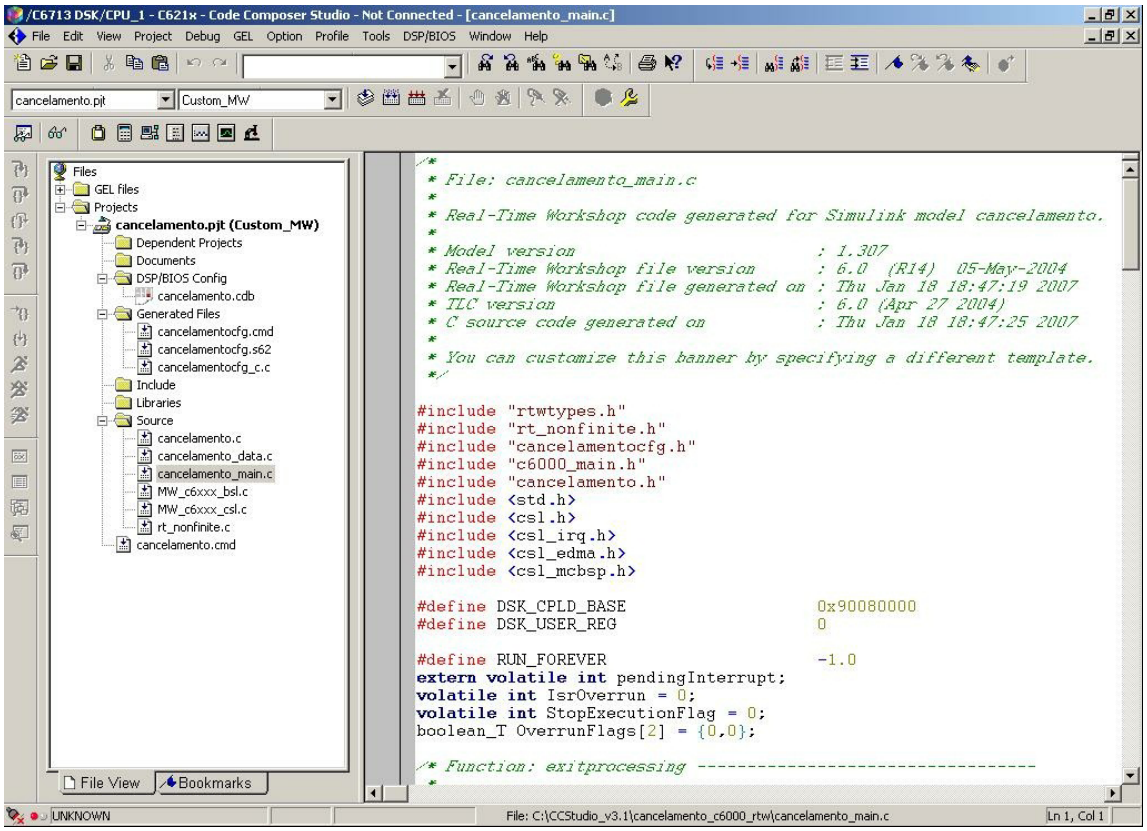


Figura 16. Comparação dos sinais: a) sinal original; b) sinal + ruído; c) sinal após o filtro adaptativo

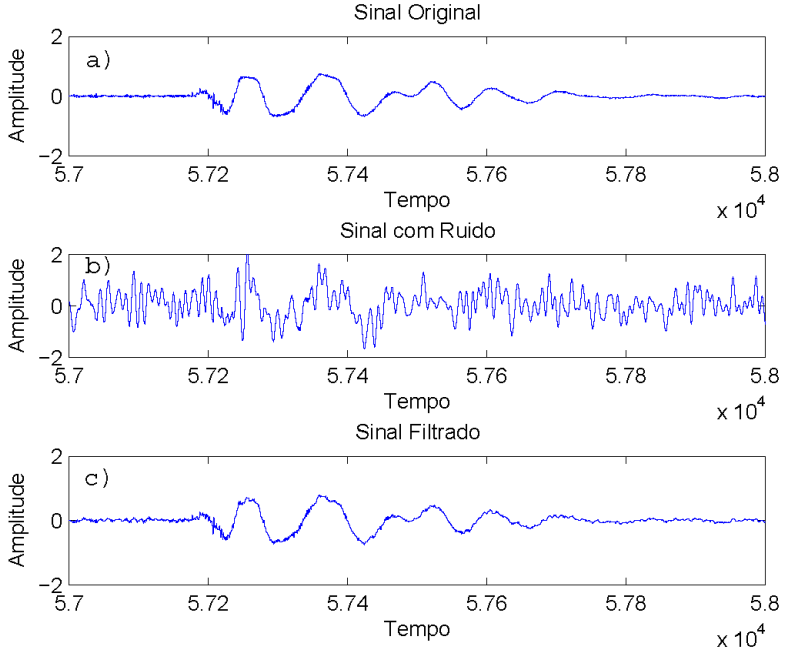
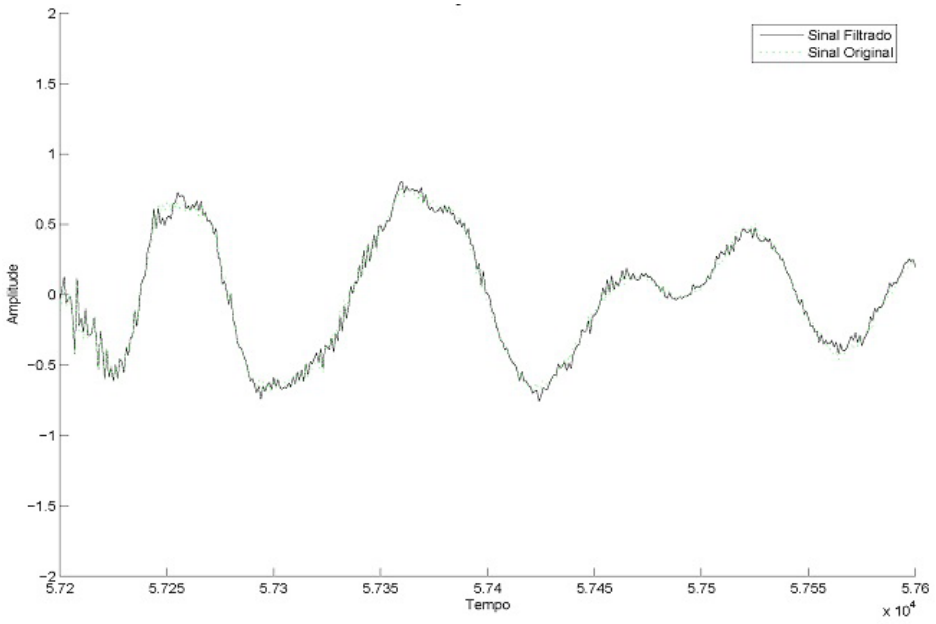


Figura 17. Sobreposição dos sinais original (livre de ruído) e de saída após a adaptação do nLMS



Conclusões

Neste trabalho foi apresentada uma metodologia para o desenvolvimento acelerado de aplicações de tempo real em DSPs, elaborada a partir da descrição do modelo em Matlab/Simulink e posteriormente transcrito para outros dois módulos, o CCS e o DSK. Especificamente, foram identificados e descritos os pontos mais importantes e críticos da metodologia. A principal vantagem dessa metodologia consiste em fornecer ao projetista de filtros digitais adaptativos uma procedimento prático eficiente para o desenvolvimento, testes e validação de algoritmos adaptativos. Esta metodologia pode ser facilmente adaptada e aplicada a outros tripos de projetos de subsistemas eletro-eletrônico que demandem processamento digital de sinais.

A metodologia de projeto apresentada é relevante para o mercado atual, onde a velocidade com que os produtos são lançados não são mais medidos em anos, mas em meses, a partir da concepção do projeto até a sua efetiva colocação no mercado de consumo final. Assim, ferramentas que acelerem o processo de análise, projeto, simulação, implementação, testes e validação são essenciais para o sucesso de qualquer projeto nesta escala de tempo. Outro aspecto importante considerado nesta metodologia de projeto é a eliminação da etapa de programação em baixo nível, que geralmente é tediosa e cujo processo de depuração é oneroso e de longa duração.

Deve-se salientar que a existência de tal ferramenta não dispensa que o projetista deva conhecer os aspectos teóricos que fundamentam a análise e modelagem do sistema. Tal ferramenta de projeto deve ser vista como um complemento para aqueles que, conhecendo a teoria por trás dos filtros e outros subsistemas fundamentais ao processamento digital de sinais, desejam implementar no mundo real modelos de subsistemas desenvolvidos previamente, teóricamente e/ou por simulação computacional. Finalmente, mas não menos importante, pode-se afirmar que a técnica discutida neste trabalho é factível e eficaz no projeto e implementação de qualquer bloco que empregue processamento digital de sinais.

Referências

KEHTARNAVAZ, N.; KIM, N.; PANAHI, I. Digital signal processing system design: using labview and

TMS320C6000. In: 11th IEEE DIGITAL SIGNAL PROCESSING WORKSHOP,AND THE 3rd IEEE SIGNAL PROCESSING EDUCATION WORKSHOP, New York, p. 10-14, 2004.

GAN, W. S. Teaching and learning the hows and whys of real-time digital signal processing. IEEE *Transactions on Education*, New York, v. 45, n. 4, p. 336-343, Nov. 2002.

SPANIAS, A.; BERISHA, V.; KWON, H. M.; HUANG, C-W; NATARAJAN, A.; FERZLI, R. *U*sing the Java-DSP Real-Time hardware interface in undergraduate classes. In: 36th ASEE/IEEE ANNUAL FRONTIERS IN EDUCATION CONFERENCE, Oct. 2006, San Diego. p. 12-17.

GALANIS, M. D.; PAPAZACHARIAS, A.; ZIGOURIS, E. A DSP Course for Real-Time systems design and implementation based on the TMS320C6211 DSK. In: IEEE INTERNATIONAL CONF. ON DIGITAL SIGNAL PROCESSING, 14, 2002, Santorini, 2002. v. 2, p. 853-856.

OROFINO, D. Rapid prototyping of a surveillance video compression system. *Matlab Digest*, *The MathWorks,* Inc. Massachusetts, v. 11, n. 5, p. 1-5, Sept. 2003.

GAN, W.; CHONG, Y.; GONG, W.; TAN, W. Rapid prototyping for teaching real-time digital signal processing. *IEEE Transactions on Education*, New York, v. 43, n. 1, p. 19-24, Feb. 2000.

TEXAS INSTRUMENTS. T*MS320C6000 optimizing C compiler: Tutorial*. Dallas, Aug. 2002.

TEXAS INSTRUMENTS. T*MS320C6000 optimizing compiler v. 6.1: user's guide*. Dallas, Oct. 2002.

WRIGHT, C. H. G.; WELCH, T. B.; GOMES, W. J. *Teaching DSP Concepts Using Matlab and the TMS320C31 DSK*. In: Proc. of the IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING, vol.6, Washington: IEEE, 1999. p. 3573-3576.

FERZLI, R.; KARAM, L. J. *An online web-based real time digital signal processing course*. In: ASEE/IEEE FRONTIERS IN EDUCATION CONFERENCE, 36., 2006, San Diego. p. 6-11.

CHACON, M. I.; VALENZUELA, I. *Fast image processing application development scheme for the DSK C6711 using Matlab and Simulink.* In: 11th IEEE DIGITAL SIGNAL PROCESSING WORKSHOP, and THE 3rd IEEE SIGNAL PROCESSING EDUCATION WORKSHOP, New York, v.1, n.1 p. 79-83, 2004.

GONZALEZ, R. C.; WOODS, R. E. D*igital image processing*. 2. ed. Denver: Prentice Hall, 2002.

MUSSI, A. M.; RIBEIRO, R. O.; ABRÃO, T. Metodologia de projeto e implementação em DSP de detectores multiusuário heurísticos DS/CDMA. *Semina: Ciências Exatas e Tecnológicas,* Londrina, v. 31, n. 2, p. 165-180, jul./dez. 2010.

MATHWORKS Inc - Accelerating the Pace of Engineering and Science. *R2011a Documentation: simulink.* Disponível em: < http://www.mathworks.com/help/>. Acesso em: 31/05/2011.

Recebido em 27 Janeiro 2011- Received on January 27, 2011.

Aceito em 27 Abril, 2011 - Accepted on April 27, 2011.