Implementação dos Principais Algoritmos de um Sistema OFDM em Plataforma de Desenvolvimento DSP-FPGA.

Marcos N. Prates e Moisés V. Ribeiro

***Resumo*— O presente artigo tem como objetivo a descrição da implementação dos principais algoritmos de um sistema OFDM em uma plataforma de desenvolvimento DSP-FPGA da Altera. Para tanto, é feita primeiramente uma discussão sobre o sistema OFDM, dando ênfase a seus princípios de funcionamento. Após, é discutida a placa de desenvolvimento da Altera utilizada nesta implementação, tendo como foco os principais recursos existentes na mesma. A seguir, é apresentado o sistema OFDM implementado, sendo discutido o funcionamento de cada bloco e sua implementação. Alguns resultados e discussões obtidos apontam o potencial das placas DSP-FPGA para a implementação de sistemas multiportadoras.**

***Palavras-Chave*— OFDM; sistemas multi-portadoras; comunicação digital de dados; transmissãode dados através da rede elétrica.**

***Abstract*— This paper aims at to discuss the implementation of the main algorithm of an OFDM system in an Altera's DSP-FPGA development platform. First, a brief review about OFDM system is provided, giving an special attention on its basic principles. A introduction on the development board used by the authors is made after that. After that, the implemented OFDM system is detailed. Some results and discussions highlight the potential of DSP-FPGA development platform for multicarrier system implementation.**

***Keywords*— OFDM; implementation on DSP-FPGA development boards; digital communication; Power Line Communication.**

I. Introdução

Atualmente, a tecnologia *Power Line Communication* (PLC) ou a transmissão de dados através das redes de energia elétrica passou a ser considerada como uma tecnologia promissora para aplicações banda larga e banda estreita e tem como principal vantagem o aproveitamento da infra-estrutura de distribuição e transmissão de energia elétrica já existente.

Neste contexto, o presente artigo descreve a implementação de alguns dos principais algoritmos da camada física de um sistema OFDM em placas de desenvolvimento *Digital Signal Processors - Field Programmable Gate Array* (DSP-FPGA) para aplicações de transmissão de dados através da rede elétrica [1], [2]. Esta implementação é baseada no conceito de orientação a objetos, o qual é disponibilizado pela Altera no software Simulink da Mathworks. Conforme é mostrado neste artigo, o uso de tal paradigma facilita consideravelmente o desenvolvimento e a implementação de sistemas OFDM.

O presente artigo é apresentado da seguinte maneira: na Seção II são apresentados os blocos do sistema OFDM implementados; na Seção III, uma introdução sobre o hardware e softwares utilizados na implementação é feita; na Seção IV, os algoritmos do sistema implementado na placa são discutidos e os resultados obtidos são apresentados e, por fim, na Seção V, as conclusões são discutidas.

Marcos Nogueira Prates e Moisés Vidal Ribeiro, *Laboratório de Processamento de Sinais e Telecomunicações* (LAPTEL), Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Juiz de Fora, Campus Universitário, Plataforma 5 - Bairro Martelos - Juiz de Fora, MG, Brasil, CEP: 36 036 330, E-mails: marcosnprates@gmail.com e mribeiro@engenharia.ufjf.br

II. O Sistema OFDM Implementado

Os termos *Discrete Multitone Transceiver* (DMT), multi-channel modulation e *multicarrier modulation* (MCM) são amplamente usados no campo de telecomunicações e às vezes se confundem com o termo OFDM. A principal diferença da modulação OFDM é que uma portadora é sempre ortogonal às outras, já na MCM, esta condição nem sempre é mantida. O termo DMT é geralmente utilizado quando o meio de transmissão é o cabo, enquanto o termo OFDM é amplamente usado quando o meio de transmissão é o ar.

Em sistemas de transmissão de dados seriais convencionais, os dados são transmitidos seqüencialmente, conseqüentemente o espectro de freqüência do símbolo ocupa a banda de freqüência disponível para a transmissão de dados. Já a técnica OFDM consiste na transmissão paralela de símbolos através de sub-canais contíguos e, conseqüentemente, fornecem taxas de transmissão por sub-portadora tão baixas quanto maior o número destas empregadas, assumindo-se uma banda de freqüência constante.

Neste contexto, o sistema OFDM implementado é ilustrado na Fig. 1, nela são mostrados os seguintes blocos: randomizador e sua inversa, modulador e demodulador 64-QAM, monta e desmonta símbolo OFDM e os blocos da Transformada Rápida de Fourier - *Fast Fourier Transform* (FFT) e Transformada Inversa de Fourier - *Inverse Fast Fourier Transform* (IFFT) [3].

O primeiro bloco implementado foi o randomizador e este algoritmo é utilizado para prevenir longas seqüências de 0s ou 1s que podem causar problemas na sincronização do sistema. O esquema de randomização empregado é o mesmo aplicado no padrão IEEE 802.16a [7] e ele é composto por um registrador de deslocamento de 15 bits e duas portas XOR.

Após os dados serem randomizados, estes são modulados utilizando-se para isto um esquema de modulação 64-QAM (*Quadrature Amplitude Modulation*) com constelação quadrada e mapeamento Gray [7], [8].

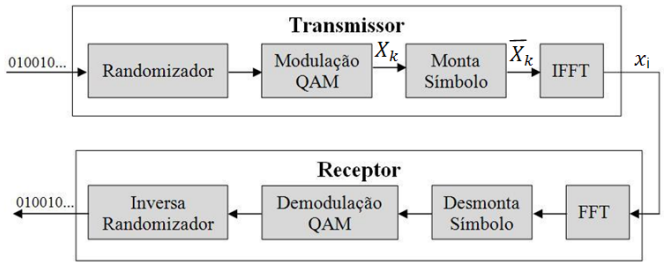

Fig. 1. Diagrama de Blocos do Sistema Implementado

Após a modulação QAM, se o sistema é DMT, então o algoritmo de montagem do símbolo, que também foi implementado, transforma a seqüência $\{X_k\}$ obtida na saída do modulador, onde $k = 0,\ 1,\ \ldots,\ N/2-1$, na seqüência $\{\overline{X}_k\}$ para $k = 0,\ 1,\ \ldots,\ N-1$ a partir da expressão [4]

$$\overline{X}_k = \begin{cases} X_k & ,k = 1,\ \ldots,\ N/2-1 \\ \Re(X_{N/2-1}) & ,k = 0 \\ \Im(X_{N/2-1}) & ,k = N/2 \\ \left(X_{N-(k+1)}\right)^* & ,k = N/2+1,\ \ldots,\ N-1 \end{cases} . \quad (1)$$

Nesta nova seqüência é aplicada a IFFT de tamanho N (onde $N = 64$ para esta implementação) na seqüência $\overline{X}_k$, obtendo a seqüência de números reais $\{x_i\}$, $0 \leq i \leq N-1$, dada por

$$x_i = \sum_{k=0}^{N-1} \overline{X}_k e^{j\frac{2\pi}{N}ik}, \quad 0 \leq i \leq N-1. \quad (2)$$

Usando este método, tanto o transmissor quanto o receptor podem ser implementados usando a versão rápida da *Discrete Fourier Transform* (DFT), a qual reduz o número de operações de N^2 na DFT para aproximadamente $N\log_2(N)$ [4], onde N é o comprimento do vetor de entrada da FFT.

Esta seqüência de números reais é transmitida pelo canal de comunicação o as operações inversão são implmentadas assumindo-se que a resposta ao impulso do canal é um impulso, a recuperação da portadora e o sincronismo são perfeitos. assim sendo, podemos explictar que a seqüência recebida é, primeiramente submetida à FFT, o demodulador 64-QAM (que não foi mostrado na Figura 1), implementa um decisão abrupta (*hard decision*) para a obtenção da seqüência de bits associada ao símbolo da constelação 64-QAM. Finalmente, a seqüência de bits é submetida à inversa do randomizador, para a obtenção da da seqüência de bits transmitida.

A implementação desses algoritmos na placa DSP-FPGA é apresentada na Seção IV.

III. Tecnologia DSP-FPGA Altera

Atualmente, está em desenvolvimento uma infinidade de circuitos digitais, e a integração entre hardware e software vem aumentando gradativamente. Neste contexto, novas tecnologias têm sido criadas no sentido de facilitar a implementação destes circuitos em placas de teste para posterior produção do produto final em larga escala, utilizando para isto de linguagens como *Very High Hardware Description Language* (VHDL) e Verilog [5], que têm como principais alvos dispositivos *Field Programmable Gate Array* (FPGA), que serão objeto de estudo e análise neste capítulo.

A. Descrição sobre as tecnologias DSP-FPGA

O FPGA foi criado pela Xilinx Inc., e teve o seu lançamento no ano de 1985 como um dispositivo que poderia ser programado de acordo com as aplicações do usuário (programador). Ele é um dispositivo semicondutor, utilizado principalmente no processamento de informações digitais e que pode ser dividido em três componentes principais: blocos de entrada e saída (IOB - *in/out Blocks*), blocos lógicos configuráveis (CLB - *Configurate Logicals Blocks*) e chaves de interconexão (*Switch Matrix*).

Com a tecnologia FPGA, é possível implementar circuitos lógicos relativamente grandes através de um arranjo de células lógicas, também chamadas de blocos lógicos configuráveis, que estão reunidos em um circuito integrado. Estes blocos implementam funções lógicas e podem se comunicar entre si.

Porém, para muitas aplicações, é necessário o processamento em tempo-real dos sinais, principalmente na área de telecomunicações. Para que dispositivos FPGA também pudessem ser usados nessas áreas, em que o processamento em tempo real de dados se faz necessário, foram implementados em seus circuitos integrados Processador Digital de Sinal - *Digital Signal Processor* (DSP). Estes processadores possuem arquiteturas específicas para manipular estruturas típicas de processamento digital de sinais, tais como filtros digitais, transformada rápida de Fourier, manipulação vetorial, etc [10].

Assim, associando o FPGA ao processador DSP, pode-se executar sistemas com capacidades de processamento e ordens de grandeza superiores se comparadas com a implementação destes sistemas utilizando apenas Processadores Digitais de Sinal - *Digital Signal Processors* (DSPs) ou FPGAs convencionais.

B. Software e Hardware Utilizados

Nesta implementação, foi utilizada uma placa FPGA da Altera da família Stratix II, modelo EP2S60 que, segundo [6], [10], é implementada em tecnologia de 90 nm, com interconexão em cobre, freqüência interna de operação máxima de 500 MHz para a lógica e de 1 GHz para interfaces seriais rápidas. Ela possui uma memória total de aproximadamente 318 Mbytes, 12 *Phase Locked Loop* (PLLs) e uma capacidade de simular 60.440 elementos lógicos.

Para o processo de desenvolvimento dos algoritmos do sistema OFDM, foi utilizado os softwares Simulink do Matlab [9] e o DSP Builder da Altera. O uso integrado destas duas ferramentas permite o desenvolvimento de algoritmos e sistemas de forma muito eficiente, já que é possível utilizar blocos, chamados de megacores, que têm implementados internamente certos algoritmos, como por exemplo a megacore FFT, que foi muito utilizada nessa implementação.

O DSP Builder ainda possui um bloco chama de HIL (*Hardware-in-the-Loop*) ou co-simulação [10], [9]. Com este

bloco é possível simular o sistema diretamente do simulink na placa, podendo-se até utilizar blocos próprio do Simulink ou vetores do Matlab.

IV. Implementação dos Algoritmos do Sistema OFDM na placa DSP-FPGA

Nesta Seção, será apresentado os blocos implementados do sistema OFDM no Simulink/DSP Builder. Além disso, uma breve explanação do funcionamento de cada algoritmo implementado e os problemas enfrentados durante o processo de implementação dos mesmos é discutida. Por fim, os resultados obtidos com a implementação são apresentados.

A. Randomizador e sua inversa

A implementação deste algoritmo na placa DSP-FPGA é ilustrada na Fig. 2. Na parte (a) desta, vemos a entrada de 6 bits sendo separada através do bloco bus splitter e cada bit atrasado de M ciclos de clock, onde M é igual a ordem do bit, sendo o MSB (*Most Significant Bit*) neste caso igual a 5. Estes passam no bloco LFSR1 e na sua saída, são atrasados novamente, agora na ordem inversa, para que formem novamente um número de 6 bits, porém agora com os bits ”randomizados”. Nesta implementação, foi adotado por questões de simplificações que o dados de entrada seriam de 6 bits, valores inteiros, positivos ou negativos.

Já na Fig. 2 (b) vemos o bloco denominado LFSR1, onde neste se encontram os registradores de deslocamento de 15 bits (LFSR - *Left Shift Registers*) e as portas XORs. São utilizados uma porta por bit e um registrador.

A inversa do Randomizador é idêntica ao randomizador e seu funcionamento o mesmo. Porém, como os dados chegarão nesta somente após passarem por todo o sistema implementado, devemos atrasar os valores do registrador de deslocamento que chegam nas portas XORs, com o auxilio do bloco delay, de 451, porque este é o atrasado provocado pelo sistema até que o primeiro valor chegue a inversa do randomizador. Se não for aplicado este atraso, o sistema irá perder a sincronização.

B. Modulador e Demodulador QAM

Essa modulação pode ser implementada através de uma tabela, onde cada seqüência de bits representa um símbolo. Supondo-se que a distância, d, entre os símbolos da constelação QAM seja igual a 2, então o número zero, que tem representação binária igual a 000000b, representa o sinal modulado igual a -7 +7i. Já o número 1, que tem representação binária igual a 000001b, quando modulado se transforma no símbolo -7 +5i. Assim, já podemos perceber que os 3 primeiros bits mais significativos representam a parte real, e os outros três menos significativos a parte imaginária. Ainda podemos notar por analogia, que os bits 000 representam o número -7 (parte real) e +7 (parte imaginária) quando modulado.

Fazendo essa comparação para todos os números, obtemos as Tabs. I e II, as quais relacionam as seqüências binárias de três bits e os valores correspondentes dos símbolos da constelação 64-QAM às partes reais e imaginárias de cada símbolo da constelação 64-QAM.

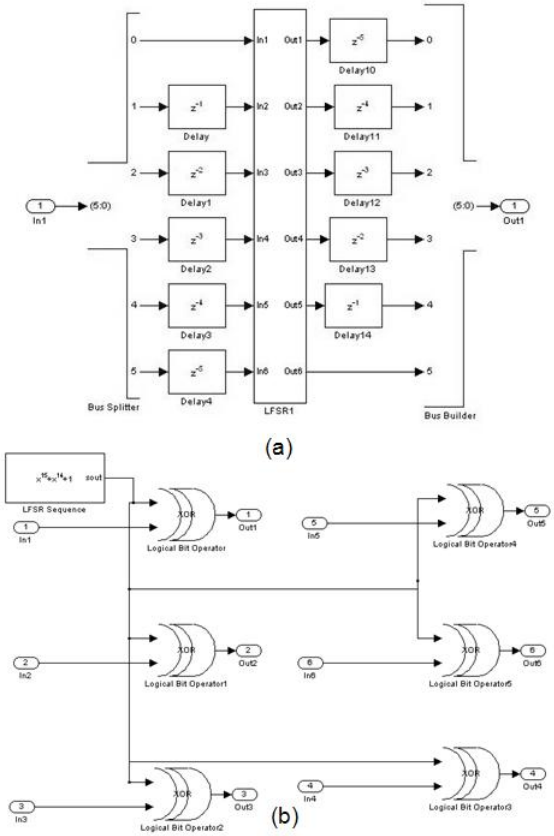

Fig. 2. (a) Bloco Randomizador (b) Bloco Left Shift Register e XORs.

TABELA I
ANALOGIA ENTRE SÍMBOLOS QAM, PARTE REAL.

$-7 \longleftrightarrow 000b$	$-1 \longleftrightarrow 010b$
$+7 \longleftrightarrow 100b$	$+1 \longleftrightarrow 110b$
$-5 \longleftrightarrow 001b$	$-3 \longleftrightarrow 011b$
$+5 \longleftrightarrow 101b$	$+3 \longleftrightarrow 111b$

Baseando-se na tabela I e II, o bloco de modulação QAM pode ser implementado através de tabelas chamadas de *Lookup Table* (LUT). Essas tabelas, conforme mostrado na Fig. 3, são blocos do DSP Builder cuja saída é o valor que está armazenado na linha apontada pela entrada, assim sendo, se quisermos modular os bits 011, armazenamos na linha 3 (011b) o valor -3 para a parte real e o valor 3 para a parte imaginária.

Na demodulação, o que muda é que agora a entrada é o símbolo, e a saída tem de ser os bits, então, coloca-se um bus splitter na saída da LUT para que um vetor de 6 bits seja formado na saída, conforme mostra a Fig. 4. Note que nesta implementação, desconsideramos, por questão de sim-

TABELA II
ANALOGIA ENTRE SÍMBOLOS QAM, PARTE IMAGINÁRIA.

$+7 \longleftrightarrow 000b$	$+1 \longleftrightarrow 010b$
$-7 \longleftrightarrow 100b$	$-1 \longleftrightarrow 110b$
$+5 \longleftrightarrow 001b$	$+3 \longleftrightarrow 011b$
$-5 \longleftrightarrow 101b$	$-3 \longleftrightarrow 111b$

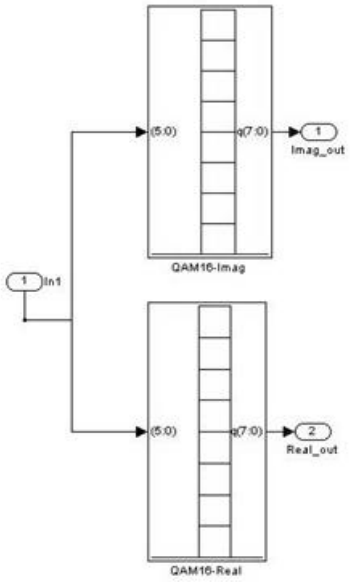

Fig. 3. Modulação 64-QAM.

plicidade, a implementação da detecção baseada nos critérios MAP (*maximum a posteriori*) e ML (*maximum likelihood*) [8].

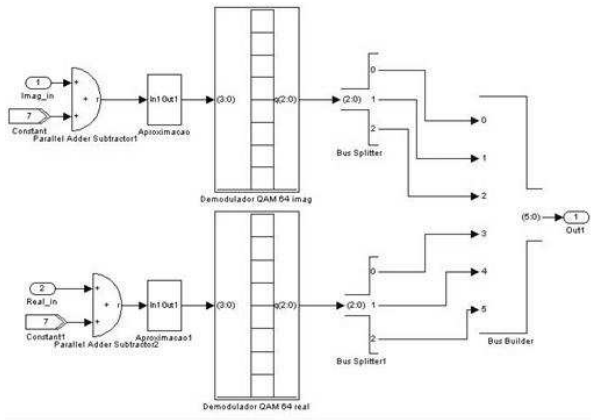

Fig. 4. Demodulador 64-QAM.

Outro bloco importante no demodulador é o que faz a decisão pela seqüência de bits associada ao símbolo da constelação 64-QAM, como dito anteriormente. Ele é necessário, pois os símbolos, que são obtidos na saída do bloco FFT, são valores discretos cujas amplitudes variam de acordo com a potência do símbolo OFDM transmitido e a influência do canal de comunicação, devido às distorções em amplitude e fase, além da presença de ruídos. Apesar da detecção usando os critérios MAP (*maximum a posteriori*) e ML (*maximum likelihood*) serem as soluções que minimizam a probabilidade de erro [8], na implementação descrita nesta contribuição optou-se pela implementação de uma lógica if - else. Porém, como neste trabalho a proposta é implementar todo o sistema utilizando-se o DSPBuilder, esta tarefa se tornará ainda mais complexa, já que terá de ser feita com os blocos disponíveis, no caso, comparadores, multiplicadores, constantes e um somador.

C. Monta e Desmonta Símbolo

Novamente, como o sistema é implementado com o auxilio da ferramenta computacional DSP Builder, a tarefa se torna um pouco mais complexa já que trabalha-se com multiplicadores, divisores, geradores de pulso, atrasos, Dual-Port RAMs e constantes para se implementar a equação 1.

Para que essa montagem de símbolos fosse possível, aproveitaram-se as saídas do modulador 64-QAM que já estavam divididas em partes real e imaginário. A Fig. 5 mostra o bloco monta símbolo imaginário, enquanto que a Fig. 6 mostra o bloco para o símbolo real. O novo bloco que aparece nesta implementação é o Dual-Port RAM, este bloco permite gravar ou ler arquivos gravados na memória RAM da placa, com as posições de memória definidas pelo programador. Neste caso, o endereço da memória é controlado por contadores (crescentes para escritas e decrescentes para leituras). Desse modo, é possível montar o símbolo conforme a equação 1 sem dificuldades.

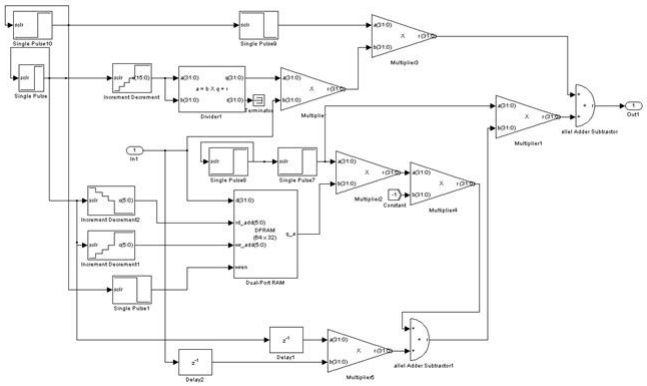

Fig. 5. Montagem do símbolo imaginário.

Da Fig. 5, pode-se ver que os dados são lidos e guardados na memória RAM da placa FPGA e ao mesmo tempo são passados para a saída através de uma multiplicação por 1 pelo multiplicador que está em cima (multiplier3). Na amostra $N/2+1$, onde N é o comprimento do vetor da saída do modulador QAM, o mesmo multiplicador (multiplier3) não permite que os valores cheguem à saída, zerando-o e então a memória RAM passa a ser lida do final para o começo, seus dados são multiplicados por -1 (esta multiplicação se faz necessária para se obter o sinal complexo conjugado) e disponibilizados na saída, obtendo-se assim a parte imaginária do símbolo OFDM.

D. FFT e IFFT

Após a montagem do símbolo OFDM no domínio da freqüência, aplica-se a inversa da DFT. Para este processo usou-se a megacore da Altera, versão 7.1, que implementa as versões rápidas da DFT e sua inversa, denominadas de FFT (*Fast Fourier Transform*) e IFFT (*Inverse Fast Fourier Transform*). O uso de tal megacore facilitou muito a implementação, porém, o único inconveniente desta é sua inicialização, que por

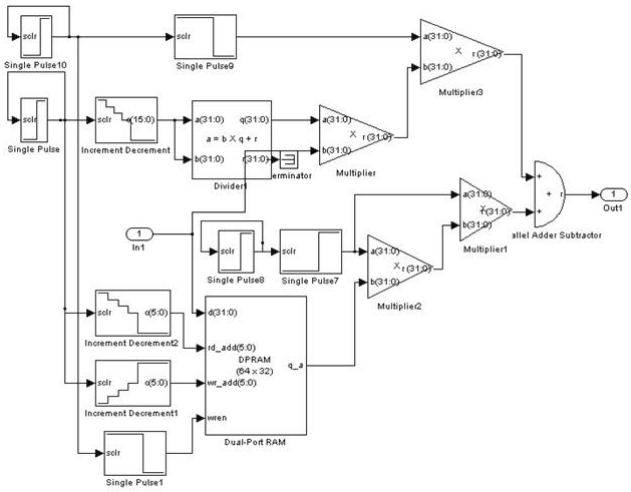

Fig. 6. Montagem do símbolo real.

não apresentar um manual claro, pode se tornar uma tarefa extremamente complexa e tediosa.

Para que o uso da megacore FFT/IFFT fosse realizado com sucesso, foi necessário amplificar o sinal por um fator de 1000 e multiplicar a saída por $\sqrt{4}$, o que também não constava no manual da mesma. O bloco implementado da IFFT é de tamanho 64 e pode ser visto na Fig. 7. Nesta Figura pode-se notar geradores de pulso que são utilizados na inicialização da *megacore*, conforme mencionado acima.

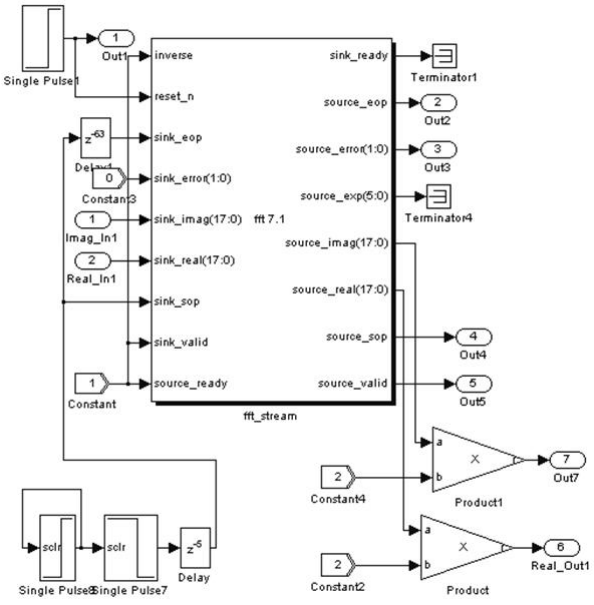

Fig. 7. Inversa da Transformada Rápida de Fourier.

O bloco da transformada rápida de Fourier, mostrado na Fig. 8, se diferencia apenas na inicialização do pino *inverse* da megacore, que para a configuração FFT tem de ser inicializado com 0, enquanto para a IFFT, por 1. O restante das inicializações deste bloco foram feitas através das saídas da IFFT, para que ocorra o sincronismo entre os blocos. Novamente as saídas têm de ser multiplicadas por $\sqrt{4}$ e divididas pelo fator igual a 1000.

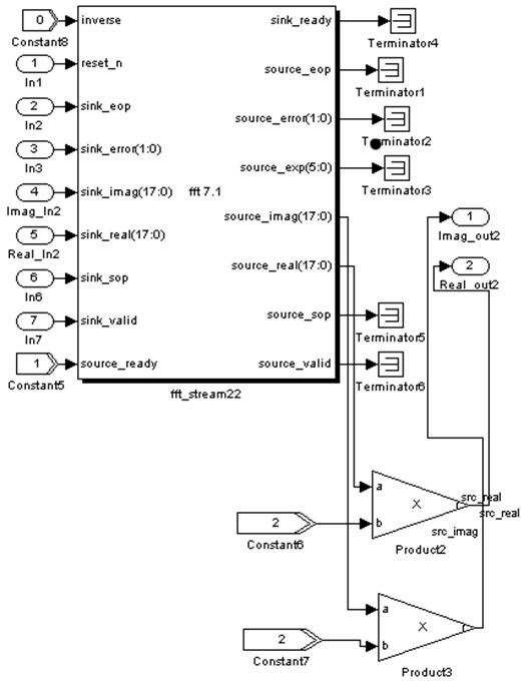

Fig. 8. Transformada Rápida de Fourier.

Juntando estes blocos aos mostrados anteriormente, chega-se ao fim da implementação do sistema OFDM. Na próxima Seção será mostrado o bloco HIL, no qual o sistema completo é compilado e simulado diretamente na placa da Altera.

E. Hardware-in-the-Loop - HIL

Esta é a etapa onde finalmente o sistema é compilado e simulado na placa, como dito anteriormente. A compilação é feita através do bloco Signal Compiler, onde se escolhe a placa utilizada e as especificações de pinagens corretas da placa. Assim, será gerado um arquivo .qpf (*Quartus Project File*), sendo assim possível simular o sistema na placa através do HIL.

Na Fig. 9 é apresentado o bloco HIL. Nele, é aberto o arquivo .qpf gerado pelo Signal Compiler, são feitas as especificações de clock do sistema e uma nova compilação é feita e, posteriormente, gravada na placa DSP-FPGA. Estes dois processos de compilação, para o sistema implementado neste trabalho, levaram cerca de 1h e 45min.

Os outros blocos mostrados na Fig. 9 são responsáveis pela inicialização da IFFT, os quais foram feitos por blocos do software Simulink do Matlab.

Na Fig. 9 também são mostrados os blocos *From Workspace* e *To Workspace*. Eles são responsáveis pela entrada e saída de dados no sistema, respectivamente. O vetor de dados foi criado no Matlab através da seguinte linha de código:

- $\mathbf{t} = [0:3001].';$
- $\mathbf{a} = ceil([rand(3002,1) * 10]);$

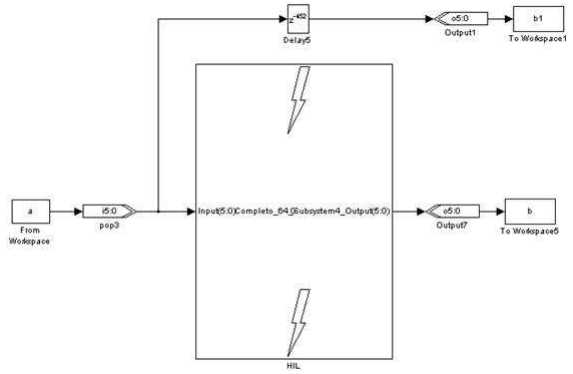

Fig. 9. Simulação do sistema na placa através do Hardware in Loop.

onde o operador $ceil$ implementa a função $\lceil x \rceil = \min\{n \in Z | n \geq x\}$ e o operador $rand(y)$ gera uma variável aleatória cuja distribuição é uniforme no intervalo $[0, 1]$.

Com o sistema gravado na placa, simulou-se o mesmo através do Simulink e os dados foram gravados em dois vetores no matlab, o vetor **b** com os dados de entrada atrasados de 452 ciclos de clock e o vetor **b1**, com os dados de saída do sistema. Este atraso no vetor **b** é feito para a comparação dos dados, já que as inicializações dos blocos IFFT e FFT atrasam a saída de dados, atrasos estes também provocados por blocos que precisam de mais de um ciclo de clock para executarem suas funções, além de atrasos colocados propositalmente no sistema, assim, totalizando um total de 452 ciclos de clock.

Na Fig. 10 são mostrados 32 dados aleatórios dos vetores b e b1 em um gráfico. Como podemos notar, o símbolo OFDM transmitido foi corretamente recebido, validando assim a implementação.

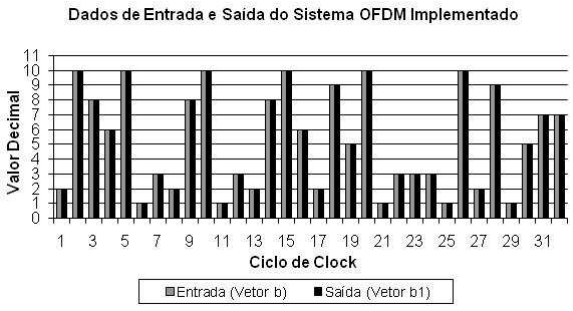

Fig. 10. Gráfico de comparação entre os vetores de entrada e saída.

Este sistema simulado na placa DSP-FPGA mostrou-se muito mais rápido do que no computador, já que a arquitetura de processamento de dados da mesma é projetada para simular sistemas deste tipo. Com um clock de 100MHz, como se demora 392 ciclos de clock para se inicializar o sistema, há um atraso de $3,92\mu s$ para que o sistema seja inicializado e para se transmitir um vetor de 32 simbolos de 6 bits, demoram-se 64 ciclos de clock, o equivalente a $0,64\mu s$ ou $640ns$. Assim sendo, este sistema é capaz de processar 100 Msps (*Mega-samples per second*).

Assumindo-se que o comprimento de um símbolo OFDM é igual ao dobro do número de subportadoras do sistema OFDM e supondo-se que a modulação 64-QAM seja aplicada nas subportadoras, podemos concluir que a taxa de transmissão de bits é igual a 600 Mbps. Assumindo-se que 20% desta taxa seja consumida pelo uso do prefixo cíclico, então o sistema implementado é, dadas as suposições acima, capaz de transmitir cerca de 420 Mbps.

Como o sistema ainda depende de dados vindos do Simulink, já que estamos trabalhando com o HIL, e os mesmos serão gravados nos vetores do Matlab, o processo não é tão rápido. Porém, nota-se nitidamente que este tempo de simulação é bem mais rápido com o sistema gravado na placa, utilizando-se o HIL, do que com o sistema sendo simulado somente no software Simulink do Matlab (não foi possível o calculo exato deste valor).

V. Conclusões

O presente artigo apresentou a descrição da implementação dos principais algoritmos de um sistema de transmissão de dados baseado na técnica OFDM numa placa DSP-FPGA.

Este foi implementado utilizando-se de ferramentas computacionais diferentes das convencionais, o que se mostrou de extrema eficiência, pois o mesmo foi concebido em apenas 6 meses e possíveis alterações que possam surgir serão feitas com extrema facilidade.

Um resultado importante é que a implementação deste sistema permite atingir taxas de transmissão de dados na camada física superiores a 420 Mbps se pelo menos a modulação 64-QAM é aplicada em todas as portadoras.

Agradecimentos

A Universidade Federal de Juiz de Fora, pela estrutura física, apoio acadêmico e financeiro que sem estes, a realização deste trabalho não seria possível e aos colegas do Labsel. Ao CNPq, FAPEMIG, CAPES e FINEP pelo auxilio financeiro.

Referências

[1] M. N. Prates, *Implementação de um Sistema OFDM em Plataforma de Desenvolvimento DSP-FPGA*. Monografia de Final de Curso. Universidade Federal de Juiz de Fora, Fev. 2008.

[2] F. P. V. Campos, *Análise de Desempenho do Sistema Clustered-COFDM para a Transmissão de Dados Via Rede Elétrica*. Dissertação de Mestrado. Juiz de Fora, MG: Universidade Federal de Juiz de Fora, Agosto 2007.

[3] W. Y. Zou e Y. Wu, *COFDM: An overview*. IEEE Transactions on Communications, v. 41, n. 1, pp. 1–8, Maio 1995.

[4] A. R. S. Bhahai e B. R. Saltzberg, *Multi-Carrier Digital Communications; Theory and Applications of OFDM*. New York: Kluwer Academic/Plenum Publishers, 1999.

[5] M. BARR, *Programmable Logic: What's it to Ya?*. Embedded Systems Programming, pp. 75–84, Junho 1999.

[6] N. M. DUARTE, *PI Componentes*. Disponível em: http://www.picomponentes.com.br/files_pi/DSPBuilder.pdf. Acesso em: 01 Dezembro 2007.

[7] IEEE. *IEEE Standard for Local and Metropolitan Area networks Part 16: Air Interface for Fixed Broadband Wireless Access Systems. IEEE Standard*, 801.16 , 2004.

[8] J. G. Proakis, *Digital Communications*, 4^a Ed., McGraw-Hill, 2000.

[9] Matlab, http://www.mathworks.com/.

[10] Quartus II, http://www.altera.com/.