Implementação em FPGA de algoritmos de sincronismo temporal para OFDM

Diego Barragán Guerrero e Luís Geraldo P. Meloni

***Resumo*— A técnica OFDM tem sido empregada em muitos esquemas de modulação da atualidade. No entanto, esta técnica é bastante sensível a erros de sincronismo de tempo e frequência. Neste trabalho, apresenta-se a implementação, verificação e síntese em hardware digital de vários algoritmos de sincronismo temporal e de detecção do pacote usando símbolos do preâmbulo definidos no padrão IEEE 802.11a. É feita uma análise da implementação em termos de consumo de recursos, frequência de operação e variância de detecção. As arquiteturas propostas foram projetadas empregando-se VHDL na parte de implementação e sintetizadas em uma FPGA Xilinx Spartan 3E no ambiente de desenvolvimento ISE e ISim em modo de linha de comando. Os resultados obtidos constituem um instrumento importante na escolha de implementação de algoritmos de sincronismo em FPGA.**

***Palavras-Chave*— OFDM, detecção do pacote, sincronismo de tempo, VHDL, FPGA.**

***Abstract*— The OFDM technique has been currently employed in many modulation schemes. However, the OFDM scheme is very susceptible to synchronization errors in time and frequency. In this paper, we present the implementation, verification and synthesis in digital hardware of several algorithms for time synchronization and packet detection using the preamble symbols defined in IEEE 802.11a. An analysis of the implementation is carried out in terms of resource consumption, operating frequency and variance of detection. The proposed architectures were implemented using VHDL and synthesized in a FPGA Xilinx Spartan 3E using development environments ISE and ISim in command-line mode. The results are an important tool for choosing the synchronization algorithms for FPGA implementation.**

***Keywords*— OFDM, packet detection, timing synchronization, VHDL, FPGA.**

I. INTRODUÇÃO

A técnica de modulação em banda larga OFDM é caracterizada por dividir a banda do canal em várias subbandas ortogonais entre si. Esta técnica tem se mostrado muito robusta em canais com multipercurso. No entanto, os sistemas OFDM são bastante sensíveis a erros de sincronismo de tempo e frequência. O OFDM transmite, como parte do pacote, um conjunto de símbolos que são usados para sincronizar o sistema. O pacote transmitido é composto de um cabeçalho, seguido dos dados de carga útil. O cabeçalho contém informações acerca do pacote de que o receptor necessita (como duração do pacote e taxa de transmissão), além de conter um preâmbulo usado para sincronismo.

O preâmbulo é composto por dois conjuntos de símbolos: curtos (denominados STS, do inglês *Short Training Sequence*) e longos (denominados LTS, do inglês *Long Training Sequence*). Cada símbolo STS é composto de 16 amostras e um símbolo LTS de 64 amostras. Antes do primeiro símbolo LTS, é inserido um intervalo de guarda composto por 32 amostras do final do símbolo LTS [1].

Os símbolos repetidos presentes no preâmbulo têm o objetivo de facilitar ao receptor o reconhecimento da presença de símbolos semelhantes, independentemente dos efeitos do canal. O padrão IEEE 802.11a especifica que os primeiros sete símbolos curtos sejam usados para a detecção do sinal, o Controle Automático de Ganho (AGC) e a seleção de diversidade (para sistemas MIMO). Os últimos três símbolos curtos devem ser utilizados para estimativa grosseira de frequência e a sincronização de tempo. Os símbolos longos são destinados à estimação de canal e a estimativa fina de frequência. No entanto, também podem ser usados para refinar a estimativa de sincronismo de tempo [2].

Este trabalho descreve a implementação de vários algoritmos de sincronismo de tempo e de detecção do pacote para OFDM em FPGA usando os símbolos do preâmbulo definidos no padrão IEEE 802.11a. O desempenho dos algoritmos implementados é verificado utilizando modelos de canais tipo A e C do *European Telecommunications Standards Institute* (ETSI), em níveis de SNR de 5 dB, 10 dB, 15 dB e 20 dB com desvios de frequência de 0 kHz, 100 kHz e 200 kHz. O trabalho emprega métricas detalhadas para selecionar entre os possíveis algoritmos de sincronismo de tempo, com base na variância, recursos de hardware e frequência de operação, considerando diferentes condições de recepção.

II. ALGORITMOS DE SINCRONISMO

A. Detecção do pacote

A primeira etapa de sincronismo é a detecção do pacote. Em [3], aproveitando a repetição de símbolos no preâmbulo, a técnica usada é a autocorrelação junto com um detector de pico. A correlação é feita entre a sequência recebida com uma cópia atrasada no tempo da mesma sequência, com um atraso equivalente ao comprimento do símbolo. A operação de autocorrelação é efetuada por

$$R(d) = \sum_{m=0}^{L-1} \left(r_{d+m}^{*} \, r_{d+m+L} \right) \tag{1}$$

onde r_d representa o valor da d-ésima amostra recebida, r_d^* representa o conjugado de r_d, e no caso dos símbolos STS, L=16 amostras.

Diego Barragán Guerrero, Luís Geraldo P. Meloni, Departamento de Comunicações DECOM, RT-DSP, FEEC - UNICAMP, Campinas, SP, Brasil, E-mail: {diego,meloni}@decom.fee.unicamp.br

Uma grande vantagem da autocorrelação é que esse algoritmo pode ser implementado utilizando uma equação recursiva, o que reduz o processamento requerido. Uma fórmula recursiva que requer duas multiplicações, uma soma e uma subtração para cada novo valor de saída do detector é [4]

$$R\left(d+1\right) = R\left(d\right) + \left(r^*_{d+L}\, r_{d+2L}\right) - \left(r^*_{d}\, r_{d+L}\right) \qquad (2)$$

Em [5] é desenvolvido um método de detecção do pacote no qual a autocorrelação é normalizada pela soma móvel da potência recebida $P(d)$, tal como é mostrado por

$$P\left(d\right) = \sum_{m=0}^{L-1} \left|r_{d+m+L}\right|^2 \qquad (3)$$

$$M\left(d\right) = \frac{\left|R\left(d\right)\right|^2}{\left(P\left(d\right)\right)^2} \qquad (4)$$

O valor resultante é comparado a um limiar, *th*, considerando-se um pacote detectado se $M(d) > th$. O valor do limiar *th* deve ser escolhido com o fim de minimizar a ocorrência da detecção de falsos positivos e evitar falha na detecção de pacotes certos, que acontece quando o receptor é incapaz de detectar o preâmbulo.

B. Sincronismo temporal

A sincronização temporal consiste em determinar o tempo ótimo no qual a leitura dos símbolos deve ser feita, isto é, detectar o limite de cada símbolo OFDM e encontrar a posição correta para a janela da transformada rápida de Fourier. Um sincronismo pouco preciso no tempo provoca interferência inter-portadora (ICI), além de poder provocar também interferência intersimbólica (ISI) [4].

Em [6], o sincronismo de tempo é efetuado encontrando o índice $d_{dropoff}$ no qual $M(d)$ cai abaixo da metade do seu valor de pico. Esse ponto ocorrerá após o último símbolo curto, durante o intervalo de guarda que precede os símbolos longos.

Outra abordagem é introduzida em [7], cujo método depende do cálculo de $R(d)$ e do cálculo de outra sequência de autocorrelação, com uma separação de amostras de *2L*

$$R_2\left(d\right) = \sum_{m=0}^{L-1} \left(r^*_{d+m}\, r_{d+m+2L}\right) \qquad (5)$$

A diferença entre estas duas sequências é calculada

$$R_{dif}\left(d\right) = R\left(d\right) - R_2(d) \qquad (6)$$

O resultado da diferença tem tipicamente um pico triangular durante o intervalo de guarda LTS, e o índice, d_{MaxDif} deste pico pode ser utilizado para calcular o desvio de temporal.

Outra alternativa é o método da correlação cruzada entre a sequência recebida e os valores originais do preâmbulo, cujo cálculo é efetuado por

$$\Lambda\left(d\right) = \sum_{m=0}^{L-1} c^*_m\, r_{d+m} \qquad (7)$$

onde o termo c^*_m é o complexo conjugado dos valores das amostras do preâmbulo, *L* é o comprimento do símbolo (*L=16* para STS e *L=64* para LTS) e r_d é a sequência recebida.

Neste algoritmo, para cada valor de saída, é necessária a realização de L multiplicações complexas. Assim, para economizar recursos computacionais, pode ser aplicada uma quantização aos dados de entrada sem afetar o desempenho do algoritmo [8].

O algoritmo de correlação cruzada usa os símbolos LTS e um detector de pico que procura o valor máximo de $\Lambda\left(d\right)$.

$$d_{xc\,\max} = \arg\max_{d}\left(\left|\Lambda\left(d\right)\right|\right) \qquad (8)$$

Os símbolos STS também podem ser usados no algoritmo de correlação cruzada. Se os símbolos originais STS forem correlacionados com a sequência recebida, haverá um conjunto de picos em $\Lambda\left(d\right)$ a cada 16 amostras. O deslocamento de tempo é calculado no ponto em que aqueles picos de correlação deixam de ocorrer.

Os métodos de sincronismo são comparados na seção IV.

C. Modelo de canal

O modelo de canal implementado é o denominado *tapped-delay*, que inclui os efeitos de desvanecimento por multipercurso e ruído gaussiano aditivo branco (AWGN). Um desvio Doppler de 52 Hz é assumido para cada *tap* [9]. A resposta ao impulso é descrita por

$$h(\tau) = \sum_{r} h_r\, \delta\left(\tau - \tau_r\right) \qquad (9)$$

Em (9) o atraso e o ganho do *r*-ésimo percurso são, respectivamente, τ_r e h_r. Realizando-se a convolução do sinal transmitido $x(t)$ com a resposta ao impulso do canal e adicionando-se o ruído do canal, $v(t)$, o sinal em banda base recebido é dado por

$$z\left(t\right) = \sum_{r} h_r\, x\left(t - \tau_r\right) + v\left(t\right) \qquad (10)$$

Da mesma forma, é introduzido um desvio de frequência na sequência de entrada, detalhado em (11). O valor de cada desvio simula o caso com condições ótimas ($\Delta f = 0$ kHz), moderadas ($\Delta f = 100$ kHz) e severas ($\Delta f = 200$ kHz). Esse desvio de frequência, na entrada do receptor com a *n*-ésima amostra no tempo do *i*-ésimo símbolo é amostrada como segue por

$$r_{i,n} = z\left(t\right) e^{j2\pi\Delta f t}\Big|_{t=i(N+N_g)\,T_s\, +\, N_g\,T_s\, +n\,T_s} \qquad (11)$$

onde N é o número de amostras por símbolo, N_g é o número de amostras do prefixo cíclico e T_s é o tempo duração de uma amostra.

III. IMPLEMENTAÇÃO

A. Detecção do pacote

A detecção do pacote é baseada em uma operação de autocorrelação normalizada pela potência recebida, conforme descrito na seção anterior. Quando o valor de $M(d)$ é maior que um limiar th, considera-se um pacote detectado. Com o intuito de economizar a operação de divisão, realiza-se uma comparação entre $|R(d)|$ com o valor de $0.5 \times |P(d)|$. O valor de 0.5 proporciona um bom desempenho e permite implementar a multiplicação por $P(d)$ com uma operação de deslocamento [2]. Os circuitos necessários para a detecção do pacote são um registro de atraso, multiplicador complexo, acumulador, comparador e um registro de deslocamento de bit.

Considerando-se o problema de picos momentâneos nos valores de (4), um circuito de média foi introduzido. Este circuito terá na saída um valor não nulo apenas se o sinal de detecção de pacote for não nulo para um determinado número M de ciclos de relógio. O valor usado na implementação foi $M = 32$, equivalente ao comprimento de dois símbolos curtos [10].

A saída do circuito de detecção do pacote inclui um sinal de controle que indica que o pacote foi detectado, como também os valores da autocorrelação e da potência calculados. O diagrama de blocos para o circuito detector de pacotes é mostrado na Fig. 1.

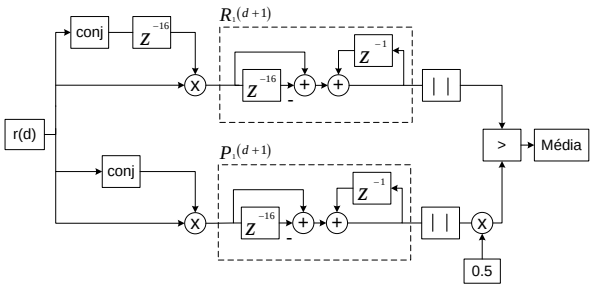

Fig. 1. Diagrama de blocos do detector do pacote.

A Fig. 2 mostra o resultado do circuito de autocorrelação e potência, quando o preâmbulo está presente na entrada. Note-se que, após 160 amostras (10 símbolos STS), o sinal da autocorrelação decai; no entanto, a potência do sinal é mantida, devido ao fato de que os símbolos STS e LTS possuem a mesma potência [11].

Em [12] é apresentada uma alternativa ao detector do pacote, que consiste em mudar o valor do deslocamento do acumulador da Fig. 1 de z^{-16} para z^{-144}. O resultado é um sinal em forma triangular, no qual, usando a mesma normalização pela metade da frequência, é determinada a presença do pacote. O resultado da autocorrelação pode ser utilizado para o sincronismo de tempo e para a estimação grosseira de frequência [11]. Na Fig. 3 é apresentado o resultado dessa nova autocorrelação, com um sinal sem ruído nem desvio de frequência.

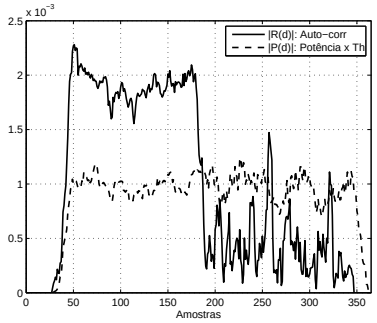

Fig. 2. Autocorrelação e potência.

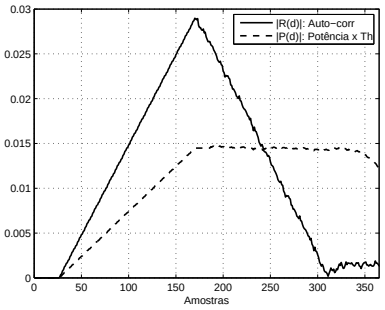

Fig. 3. Autocorrelação e potência com atrasador de 144 amostras.

B. Sincronismo de tempo

1) Método de autocorrelação básica: Neste método, as entradas para o sincronizador são o valor $R(d)$ de (1) e o valor de $P(d)$ de (4). Este algoritmo inicia um contador quando o pacote é detectado e termina quando o nível de $R(d)$ está por baixo do limiar de $0.5 \times |P(d)|$. Nesse instante, é determinado o limite do pacote.

Usando o resultado da autocorrelação da Fig. 3, a posição do valor máximo é determinada com um circuito detector de pico. Esse circuito é composto por um contador, dois registros e um comparador. O detector de pico devolve um índice que indica a posição da amostra em que ocorre o valor máximo, indicando o limite do pacote.

2) Método de diferença de autocorrelação: Neste método, uma outra expressão de autocorrelação $R_2(d)$ é introduzida, sendo realizada com um atraso de 32 amostras e empregando uma arquitetura semelhante àquela exibida na Fig. 1. Reutilizando-se o resultado de $R_1(d)$ e subtraindo-se do valor de $R_2(d)$, esse sinal resultante serve de entrada a um bloco de detecção de pico, conforme mostrado na Fig. 4. Esse valor é usado para determinar o limite do pacote.

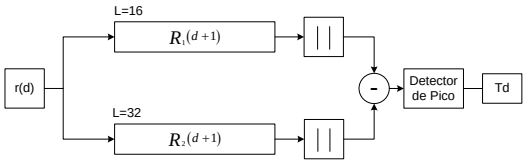

Fig. 4. Algoritmo de diferença de autocorrelações.

A Fig. 5 mostra o resultado da diferença das autocorrelações. No sinal resultante é buscado o pico máximo, cuja posição determina o limite do pacote.

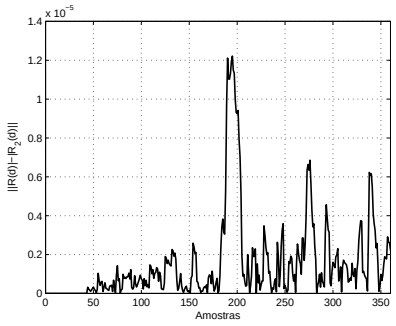

Fig. 5. Diferença de autocorrelação.

3) Correlação cruzada: A correlação cruzada multiplica o sinal que chega pelo canal pelo símbolo original transmitido [4]. Os valores do símbolo original são armazenados em constantes dentro do programa.

Os elementos necessários para esta implementação são multiplicadores, registros de atraso e somadores. O resultado é um circuito com 64 tipos semelhantes de operações, conforme apresentado na Fig. 6. Esse tipo de implementação é denominada filtro casado. Na figura, r_d representa a amostra de entrada, c_n^* é o complexo conjugado da amostra do símbolo armazenado em memória e $\Lambda\,(d)$ é o resultado da equação (7).

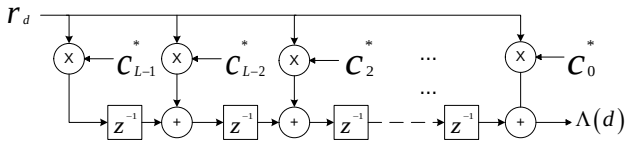

Fig. 6. Correlação cruzada implementada com um filtro casado.

Com o intuito de economizar recursos multiplicadores, uma quantização é realizada com as amostras de entrada da seguinte forma: toma-se os bits mais significativos de cada amostra (bit de sinal) de entrada, logo esses bits são arranjados em um *buffer* de comprimento igual ao tamanho do LTS. Consegue-se assim transformar a multiplicação complexa do símbolo de entrada com o símbolo sem ruído em uma operação de soma e diferença binária [8].

A saída do correlator é apresentada na Fig. 7. Pode-se ver que o sinal resultante expõe claramente três picos: um ao final do intervalo de guarda de 32 amostras, outro ao final do primeiro LTS e um último pico ao final do segundo LTS.

No entanto, conforme é mostrado em [13] e [12], é possível diminuir os recursos da FPGA e aumentar a frequência de operação usando apenas 32 amostras do LTS. Na Fig. 8, é exibido o resultado dessa nova correlação cruzada, pelo qual é fácil distinguir a presença de dois picos, cujas posições são usadas para determinar o limite do pacote. Em condições ideais, a posição do primeiro pico fica na metade do primeiro símbolo LTS.

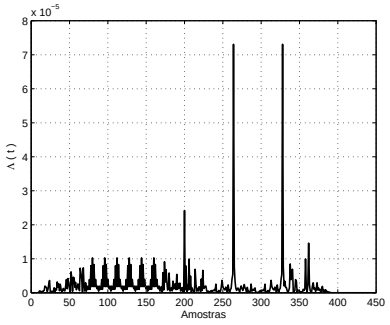

Fig. 7. Correlação cruzada com LTS.

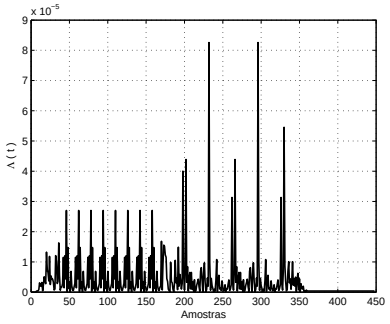

Fig. 8. Correlação cruzada com 32 amostras do LTS.

Igualmente, é possível casar as amostras dos símbolos de entrada com as amostras dos símbolos STS sem ruído. O resultado é um sinal com vários picos localizados ao final de cada símbolo curto, conforme mostrado na Fig. 9.

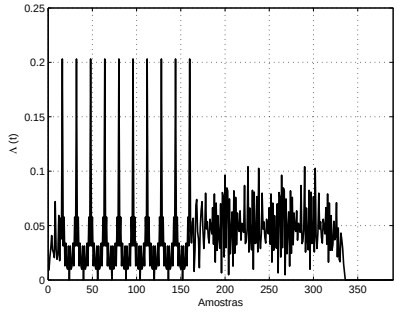

Fig. 9. Correlação cruzada com STS.

IV. Resultados

Conforme descrito no início deste trabalho, os indicadores de comparação são a variância, o número de recursos da FPGA e a frequência de operação. Os sinais de teste foram gerados no Matlab, armazenados em formato binário em um arquivo *.txt* e lidos pelo *test bench*, cujo resultado foi também armazenado em um arquivo de texto para ser plotados e contabilizados em Matlab. Com o intuíto de diminuir o tempo de teste, o Xilinx ISE foi usado em modo de linha de comandos [14]. Nas Figuras 10 e 11 estão plotados os resultados das implementações com os algoritmos em questão, indicando a

variância de seus estimadores temporais em diferentes níveis de SNR, para 1200 sinais de teste quantizados com 12 bits de precisão.

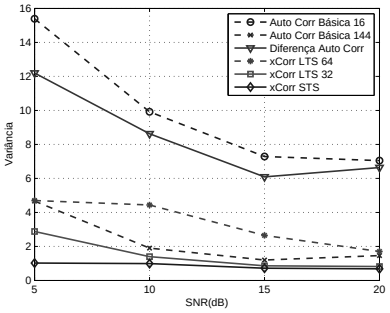

Fig. 10. Variância da estimativa temporal para SNR variável: canal ETSI A.

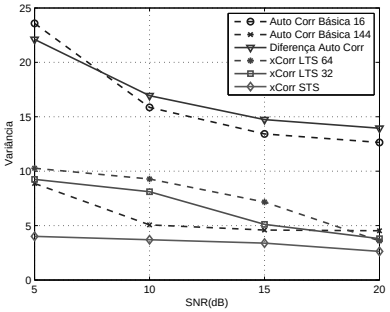

Fig. 11. Variância da estimativa temporal para SNR variável: canal ETSI C.

A Tabela I apresenta os recursos de hardware e frequência de operação para cada um dos algoritmos implementados.

TABELA I
SLICES E FREQUÊNCIA DE OPERAÇÃO.

Algoritmo	Freq. (MHz)	Slice
Autocorr. bás. 16	138,49	825
Autocorr. bás. 144	138,49	1299
Dif. de autocorr.	132,43	1160
xCorr LTS (64)	13,06	2045
xCorr LTS (32)	22,43	1109
xCorr STS	37,40	727

Os algoritmos de autocorrelação básica e de diferença de autocorrelação apresentam a maior variância e uma frequência de operação semelhante. No entanto, o algoritmo de autocorrelação básica possui um menor consumo de *slices*. O algoritmo de autocorrelação básica com atrasador de 144 amostras possui uma variância menor mantendo a frequência de operação, porém usa mais recursos de *slices*. Já entre os algoritmos baseados em correlação cruzada com LTS, o melhor resultado em cada um dos parâmetros foi aquele que usava 32 amostras do símbolo. A correlação cruzada com os símbolos STS possui a menor variância, a maior frequência de operação e um menor consumo de *slices* se comparada com os algoritmos que usam a correlação cruzada com LTS.

V. CONCLUSÕES

Vários algoritmos de detecção de pacote e sincronismo de tempo foram implementados em FPGA, considerando diversas condições de recepção. A caracterização de cada algoritmo é baseada na variância do resultado, número de *slices* e frequência de operação. Em termos de variância, os algoritmos baseados em correlação cruzada apresentam um resultado melhor. Em termos de frequência de operação, os algoritmos baseados em autocorrelação permitem maior frequência. Finalmente, o algoritmo de correlação cruzada com 64 amostras possui o maior consumo de *slices*.

AGRADECIMENTOS

Este trabalho foi apoiado pela Rede Nacional de Ensino e Pesquisa (RNP), contrato no. 001334, “Construindo Cidades Inteligentes: da Instrumentação dos Ambientes ao Desenvolvimento de Aplicações”.

REFERÊNCIAS

[1] Ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pages 1–2793, 2012.

[2] Joseph Pierri. Design and implementation of an ofdm wlan synchronizer. Master thesis, Electrical and Computer Engineering, University of Waterloo, 2007.

[3] T. Keller and L. Hanzo. Orthogonal frequency division multiplex synchronisation techniques for wireless local area networks. In *Personal, Indoor and Mobile Radio Communications, 1996. PIMRC'96., Seventh IEEE International Symposium on*, volume 3, pages 963–967 vol.3, 1996.

[4] T.D. Chiueh, P.Y. Tsai, and I.W. Lai. *Baseband Receiver Design for Wireless MIMO-OFDM Communications*. Wiley, 2012.

[5] T.M. Schmidl and D.C. Cox. Robust frequency and timing synchronization for ofdm. *Communications, IEEE Transactions on*, 45(12):1613–1621, 1997.

[6] Jianhua Liu and Jian Li. Parameter estimation and error reduction for ofdm-based wlans. *Mobile Computing, IEEE Transactions on*, 3(2):152–163, 2004.

[7] K. Wang, J. Singh, and M. Faulkner. Fpga implementation of an ofdm-wlan synchronizer. In *Field-Programmable Technology, 2004. Proceedings. 2004 IEEE International Conference on*, pages 89–94, 2004.

[8] C. Dick and F. Harris. Fpga implementation of an ofdm phy. In *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on*, volume 1, pages 905–909 Vol.1, 2003.

[9] Anna Berno and Nicola Laurenti. Time and frequency synchronization for hiperlan/2. In *Proceedings of the Second International IFIP-TC6 Networking Conference on Networking Technologies, Services, and Protocols*, pages 491–502, London, UK, UK, 2002. Springer-Verlag.

[10] Jinpeng Xie, Yingqiang Ding, Shouyi Yang, and Lin Qi. Fpga implementation of frame synchronization and symbol timing synchronization based on ofdm system for ieee 802.11 a. In *Intelligent Signal Processing and Communication Systems (ISPACS), 2010 International Symposium on*, pages 1–4. IEEE, 2010.

[11] André Michielin Câmara. Sincronização de redes de pacotes ofdm. Tese de graduação, Universidade Federal do Rio Grande do Sul. Escola de Engenharia. Curso de Engenharia Elétrica, 2007.

[12] María José Canet, Javier Valls, Vicenç Almenar, and José Marín-Roig. Fpga implementation of an ofdm-based wlan receiver. *Microprocessors and Microsystems*, 36(3):232 – 244, 2012.

[13] M.J. Canet, I. Wassel, V. Almenar, and J. Valls. Performance evaluation of fine time synchronizer for wlans. In *Proceedings of 13th European Conference on Signal Processing (EUSIPCO 2005)*, volume 2, pages II–341–4 vol.2, 2005.

[14] Evgeni Stavinov. Using xilinx tools in command-line mode. Online at http://outputlogic.com/xcell_using_xilinx_tools/74_xperts_04.pdf, Julho 2011.