UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS PATO BRANCO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

ALEXSANDRO BROCARDO LOPES

DESENVOLVIMENTO DE ESTRUTURAS PARALELAS EM FPGA E IMPLEMENTAÇÃO DE CONTROLADORES DIGITAIS PARA APLICAÇÃO EM FILTROS ATIVOS DE POTÊNCIA

DISSERTAÇÃO

PATO BRANCO
2012

ALEXSANDRO BROCARDO LOPES

DESENVOLVIMENTO DE ESTRUTURAS PARALELAS EM FPGA E IMPLEMENTAÇÃO DE CONTROLADORES DIGITAIS PARA APLICAÇÃO EM FILTROS ATIVOS DE POTÊNCIA

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de "Mestre em Engenharia Elétrica". Área de Concentração: Sistemas de Processamento de Energia.

Orientador: Prof. Dr. Emerson Giovani Carati

PATO BRANCO

2012

Catalogada na Fonte por Elda Lopes Lira CRB9/1295

L864d Lopes, Alexsandro
Desenvolvimento de estruturas paralelas em FPGA e implementação de controladores digitais para aplicação em filtros ativos de potência / Alexsandro Lopes – 2012.
92 f. : il.; 30 cm.

Orientador: Emerson Giovani Carati
Dissertação (Mestrado) - Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Engenharia Elétrica. Pato Branco / PR, 2012.
Bibliografia: f. 78 - 81

1.Controle Paralelo. 2.Filtros Ativos. 3.FPGA. 4.Controle Ressonante. 5. Com pensador Seletivo. I.Carati, Emerson Giovani, orient. II.Universidade Tecnoló gica Federal do Paraná. Programa de Pós-Graduação em Engenharia Elétrica III. Título.

CDD(22. ed.) 621.3

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Pato Branco
Diretoria de Pesquisa e Pós-Graduação
Programa de Pós-Graduação em Engenharia Elétrica

TERMO DE APROVAÇÃO

Título da Dissertação nº 010

Desenvolvimento de Estruturas Paralelas em FPGA e Implantação de Controladores Digitais para Aplicação em Filtros Ativos de Potência.

por

Alexsandro Brocardo Lopes

Dissertação apresentada às oito horas e trinta minutos do dia trinta de março de dois mil e doze, como requisito parcial para obtenção do título de MESTRE EM ENGENHARIA ELÉTRICA, Linha de Pesquisa – Análise de Sistemas Dinâmicos, Programa de Pós-Graduação em Engenharia Elétrica (Área de Concentração: Sistemas e Processamento de Energia), Universidade Tecnológica Federal do Paraná, *Campus* Pato Branco. O candidato foi argüido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho. . . . APROVADO . . . . . .

Banca examinadora:

Dr. Robinson Figueiredo de Camargo
UFSM

Dr. Fábio Favarim
UTFPR

Dr. Rafael Cardoso
UTFPR

Dr. Emerson Giovani Carati
UTFPR (Orientador)

Visto da Coordenação:

Prof. Dr. Rafael Cardoso
Coordenador do PPGEE

Aos meus pais, Adair e Gloriamar
e minha esposa e filha, Adriana e Bárbara

AGRADECIMENTOS

Ao professor Dr. Emerson Giovani Carati, orientador desta dissertação, por todo empenho, compreensão e sabedoria.

Aos professores Dr. Rafael Cardoso e Dr. Fábio Favarim, pelas correções, sugestões e tempo dispensado em prol deste trabalho.

Aos colegas da primeira e segunda turma do Programa de Pós Graduação em Engenharia Elétrica da UTFPR campus Pato Branco pelo apoio e a amizade.

Em especial a minha esposa e filha pela compreensão nos muitos momentos de ausência.

RESUMO

LOPES, Alexsandro Brocardo. **Desenvolvimento de estruturas paralelas em FPGA e implementação de controladores digitais para aplicação em filtros ativos de potência**. 2012. 89. Dissertação (Mestrado em Engenharia Elétrica) - Universidade Tecnológica Federal do Paraná. Pato Branco, 2012.

Este trabalho apresenta uma abordagem paralela em FPGA para implementação de controladores digitais, com a finalidade de reduzir o tempo computacional de execução das leis de controle. A técnica de controle utilizada na implementação proposta pode ser aplicada para a compensação seletiva de harmônicos em filtros ativos de potência (FAP). Para compensar mesmo um número reduzido de harmônicas são necessárias múltiplas instruções de cálculo, envolvendo multiplicações e adições. Desta forma, para melhorar o desempenho computacional do sistema, é proposta uma estrutura paralela para implementação do controlador. Para compensação de um número maior de harmônicas é realizada a decomposição das tensões em suas componentes harmônicas em eixos síncronos utilizando a Transformada Discreta de Fourier. Esta estratégia permite um tempo de cálculo fixo independente do número de harmônicas, até o limite imposto pela frequência de amostragem. Os resultados experimentais são apresentados para comparar o tempo de execução da abordagem paralela proposta com o tempo de execução sequencial convencionalmente utilizado na literatura.

Palavras-chave: Controle Paralelo. Filtros Ativos. FPGA. Controle Ressonante. Compensador Seletivo.

ABSTRACT

LOPES, Alexsandro Brocardo. **Development of parallel structures in FPGA and implementation of digital controllers for an active power filters**. 2012. 89. Dissertação (Mestrado em Engenharia Elétrica) - Federal Technology University - Parana. Pato Branco, 2012.

This paper presents a parallel approach to FPGA implementation of digital controllers, in order to reduce the computational time for implementing the control laws. The control technique used in the proposed implementation can be applied for selective harmonic compensation in active power filters (APF). To compensate for even a small number of harmonics requires multiple calculation instructions involving multiplications and additions. Thus, to improve the performance of the computer system is proposed to implement a parallel structure of the controller. To compensate for a larger number of harmonics is performed in the decomposition voltages of the harmonic components into their frames synchronous using the Discrete Fourier Transform. This strategy allows a calculation time fixed regardless of the number of harmonics up to the limit imposed by the sampling frequency. The experimental results are presented to compare the runtime of the proposed parallel approach with the sequential execution time conventionally used in the literature.

Keywords: Parallel Control. Active Filters. FPGA. Resonant Control. Selective Compensator.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

A/D	Analógico para Digital
ANEEL	Agência Nacional de Energia Elétrica
CA	Corrente Alternada
CC	Corrente Contínua
CISC	Computador com um Conjunto Complexo de Instruções (*Complex Instruction Set Computer*)
CLB	Blocos Lógicos Configuráveis (*Configurable Logic Blocks*)
D/A	Digital para Analógico
DFT	Transformada Discreta de Fourier (*Discrete Fourier Transform*)
DSP	Processador Digital de Sinais (*Digital Signal Processor*)
FAP	Filtros Ativos de Potência
FFT	Transformada Rápida de Fourier (*Fast Fourier Transform*)
FIR	Resposta ao Impulso Finita (*Finite Impulse Response*)
FPGA	Arranjo de Portas Programável em Campo (*Field Programmable Gate Array*)
HC	Compensador Harmônico (*Harmonic Compensator*)
HDL	Linguagem de Descrição de *Hardware* (*Hardware Description Language*)
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos (*Institute of Electrical and Electronics Engineers*)
IDE	Ambiente Integrado para Desenvolvimento (*Integrated Development Environment*)
KCPSM	K'constant Coded Programmable State Machine
LUT	Tabela *Look-Up* (*Look-Up Table*)
MAC	Multiplicador e Acumulador (*Multiply and Accumulate*)
MIPS	Milhões de Intruções Por Segundo (*Millions of Instructions Per Second*)
OSAP	One Sample Ahead Preview
PI	Controlador Proportional Integral (*Proportional-Integral Controller*)
PID	Controlador Proporciona, Integral e Derivativo (*Proportional–Integral–Derivative*

	Controller)
PR	Controlador Proporcional Ressonante (*Proportional-Resonant Controller*)
PWM	Modulação por Largura de Pulso (*Pulse Width Modulation*)
RAM	Memória de Acesso Aleatório (*Random Access Memory*)
RISC	Computador com um Conjunto Reduzido de Instruções (*Reduced Instruction Set Computer*)
SPI	Interface Serial para Periférico (*Serial Peripheral Interface*)
STD	Padrão (*Standard*)
THD	Distorção Harmônica Total (*Total Harmonic Distortion*)
ULA	Unidade Lógica Aritmética
VHDL	Linguagem de Descrição de *Hardware* VHSIC (*Very High Speed Integrated Circuit Hardware Description Language*)
VHSIC	Circuito integrado de altíssima velocidade (*Very High Speed Integrated Circuits*)

SUMÁRIO

1. INTRODUÇÃO

A energia elétrica tem sido uma das fontes de energia essenciais para o desenvolvimento tecnológico ocorrido no último século. Além disso, a qualidade dessa energia influencia diretamente nesse desenvolvimento. Neste sentido, até recentemente, a qualidade de energia era vinculada quase que exclusivamente a continuidade de fornecimento e nos limites aceitáveis de variações de tensão e frequência. Ainda, as cargas eram consideradas quase que exclusivamente lineares, o que implicaria em correntes senoidais de consumo que não distorcem as tensões fornecidas, as quais também são senoidais. Entretanto, nas últimas décadas, normas nacionais e internacionais (IEEE STD 519-1992, 1992; IEEE STD 929-2000, 2000; ANEEL - PRODIST, 2010) tem sido desenvolvidas e estabelecem um conjunto mais amplo de índices para medição e verificação da qualidade de energia, incluindo a distorção harmônica total (THD – *Total Harmonic Distortion*). A norma IEEE 519 especifica os limites de distorção de uma instalação elétrica, medindo no ponto de conexão à rede de distribuição. Este limite, definido pela THD, é dada em porcentagem da máxima corrente de demanda da instalação.

Com a evolução da eletrônica de potência os dispositivos eletroeletrônicos tornaram-se amplamente utilizados nas mais diversas aplicações e equipamentos. A desvantagem destes dispositivos é que os mesmos atuam como cargas não-lineares e, na maioria dos casos, drenam correntes não senoidais da rede de fornecimento, gerando assim correntes distorcidas. A soma de uma grande quantidade destes dispositivos pode provocar distorções consideráveis à tensão de alimentação fornecida pelas empresas concessionárias de energia. Para corrigir este problema uma das possíveis soluções é a aplicação de filtros ativos de potência (FAP). Os FAPs paralelos são comumente utilizados para compensação de harmônicas de corrente, para controle de potência reativa e para compensação de instabilidade na presença de cargas não-lineares.

Diversos métodos convencionais de controle utilizam como referência a teoria de potência instantânea (AKAGI, KANAZAWA e NABAE, 1984) para detecção de correntes harmônicas e são baseados na corrente de carga. Estas técnicas geralmente resultam em atraso que causa compensação incorreta e harmônicos indesejados restantes nas correntes de linha (MATTAVELLI, 2001). A técnica mais comum, proporcional integral (PI – *Proportional*

Integral) possui dois problemas conhecidos: incapacidade de rastrear uma referência senoidal sem erro de estado estacionário e baixa capacidade de rejeição de distúrbios. Outro controlador de ampla utilização, o controlador *deadbeat,* tem o objetivo de adequar o sinal de saída à referência em um tempo mínimo e com erro nulo (KUO, 1992), mas possui problemas relacionados à sensibilidade a variações paramétricas, o que frequentemente ocorre em filtros ativos (MALESANI, MATTAVELLI e BUSO, 1998). Para resolver estes problemas um outro tipo de regulador de estado estacionário, chamado Proporcional-Ressonante (PR – *Proportional-Resonant*) (ZMOOD e HOLMES, 2003) tem sido utilizado. O controlador PR permite um grande número de instruções em paralelo, sendo que cada harmônica precisa de um controlador e eles podem ser paralelizados. O problema desta implementação é o número de conversões DQ necessárias para as transformações para eixos estacionários. Um controlador que não precisa destas conversões foi proposto por Mattavelli e Fasolo (2000), onde é utilizado um controlador síncrono para frequências predefinidas, não requerendo conversões de eixos e utilizando o mesmo número de cálculos indiferente do número de harmônicas selecionadas.

Quanto a implementação, atualmente as principais ferramentas computacionais para síntese de sistemas de controle digital são os microcontroladores, microprocessadores, DSPs (*Digital Signal Processor)* e, recentemente, os FPGAs (*Field Programable Gate Array*). O uso de microcontroladores e DSPs para conversores Proporcional-ressonante, sem introduzir um atraso significativo, geralmente permite apenas a compensação de poucos harmônicos, devido ao processamento sequencial para cada harmônico. Os FPGAs são compostos de matrizes de blocos lógicos configuráveis (CLB - *Configurable Logic Blocks*) que podem ser configurados paralelamente sem competir por recursos em comum. Utilizando algumas abordagens (HWANG, 1979) de implementação de funções de multiplicação e adição pode-se implementar, por exemplo, filtros digitais em uma arquitetura FPGA. Neste sentido, alguns princípios de redução de tempo de processamento baseado em processamento paralelo estão sendo discutidos e implementados no meio científico (SUGAHARA, OIDA e YOKOYAMA, 2009; JENSEN et al., 2008). A questão do tempo de processamento é fundamental em aplicações que necessitem de uma grande quantidade de cálculos em tempo reduzido. Um conversor CC-CC, por exemplo, utilizando técnicas de controle mais complexas, com frequência de comutação elevadas, na ordem de dezenas ou centenas de kilohertz, exige um número de cálculos que os DSP podem não ser capazes de realizar em um ciclo de controle (WAN e FERDOWSI, 2008; JAKOBSEN e ANDERSEN, 2007). A grande quantidade de recursos de *hardware* disponíveis nos FGPAs atuais (Xilinx®; Altera®) permite a

implementação de vários processos simultaneamente e pode vir a contribuir para redução dos tempos de processamento em algoritmos de controle.

O tempo de processamento é o fator fundamental para escolha do algoritmo de controle de muitas aplicações, como discutido em (MATTAVELLI e FASOLO, 2000; MONTAGNER, CARATI e GRÜNDLING, 2000). Algoritmos com grande número de operações podem levar a um tempo de processamento muito elevado e, assim, se tornar inviáveis em sistemas que utilizam frequência de atuação relativamente elevada. Outro problema em controle digital de processos é realizar a atuação no mesmo ponto de amostragem, e não apenas na amostra seguinte, uma vez que a segunda abordagem leva a atrasos de transporte significativos. Apesar da disponibilidade de dispositivos de controle com elevado desempenho de processamento, a execução sequencial das operações de algoritmos de controle mais refinados podem necessitar de um grande número de ciclos e, desta forma, levar a tempos de processamento relativamente elevados. A execução paralela dos referidos algoritmos através de diversos dispositivos de processamento permite a realização de diversas operações paralelamente e, portanto, permite uma redução do tempo de processamento e a atuação mais próxima do instante ideal.

1.1. OBJETIVOS

Considerando as questões supracitadas, o objetivo deste trabalho é o desenvolvimento de estruturas paralelas, baseadas em FPGA, para processamento de alta velocidade na implementação de controladores digitais. Utilizando tal abordagem de implementação é possível aplicar algoritmos digitais de controle para compensação de um número elevado de harmônicos em filtros ativos de potência. Para isso, neste trabalho é utilizada a abordagem apresentada por Mattavelli e Fasolo (2000), no qual foi proposto um controlador síncrono para frequências predefinidas, projetado a partir de um compensador ressonante em cada frequência da harmônica a ser compensada. No referido trabalho utilizou-se implementação sequencial dos cálculos do algoritmo de controle em um DSP de ponto fixo. O sistema proposto neste trabalho emprega FPGAs de ponto fixo, nos quais é realizada a síntese de técnicas de processamento paralelo através de diferentes arquiteturas de hardware, com o objetivo de reduzir o tempo de execução dos cálculos do algoritmo de controle. Para verificação da abordagem proposta, resultados experimentais relativos aos tempos de processamento são comparados com os resultados apresentados por Mattavelli e Fasolo (2000). Desta forma, os objetivos específicos deste trabalho são:

- Realizar revisão da literatura a respeito de algoritmos de controle digital com estruturas apropriadas ao paralelismo, com aplicações em filtros ativos de potência;

- Analisar estruturas de implementação de hardware configurável e técnicas de processamento paralelo;

- Analisar, projetar e implementar filtros e controladores digitais utilizando processamento paralelo através de estruturas de hardware programável;

- Verificar experimentalmente o comportamento das estruturas de hardware desenvolvidas com relação a tempo de processamento;

- Comparar os resultados experimentais com implementações da literatura baseadas em execução sequencial.

1.2. ESTRUTURA DO TRABALHO

Este trabalho está organizado como segue: no Capítulo 2 são apresentados conceitos e a modelagem de um filtro ativo de potência. Ainda, é realizada uma discussão sobre alguns sistemas de controle mais utilizados para tais filtros. Os conceitos sobre FPGAs, arquiteturas paralelas, suas aplicações para implementação de cálculos de controle e a implementação dos controladores digitais em FPGA é apresentado no Capítulo 3. O Capítulo 4 apresenta os resultados experimentais obtidos neste trabalho e sua comparação com a literatura considerada. No capítulo 5 são apresentadas conclusões e propostas para trabalhos futuros.

2. FILTROS ATIVOS DE POTÊNCIA E SISTEMA DE CONTROLE

2.1. INTRODUÇÃO

O objetivo de um filtro de potência é reduzir o efeito de harmônicos causados por dispositivos comutadores, cargas não lineares, entre outros. Filtros passivos só filtram as frequências para as quais foram previamente definidos e, assim, seria necessário um circuito relativamente grande para a filtragem de várias frequências. Para substituir este método de filtragem tem sido utilizados filtros ativos de potência (FAP). Tais filtros desempenham a mesma função básica como filtros passivos, mas são baseados em dispositivos eletrônicos de potência comandados por controladores analógicos ou digitais, utilizando geralmente modulação em largura de pulso (PWM – *Pulse Width Modulation*). Neste capítulo é inicialmente abordado uma apresentação de uma estrutura de filtros ativos de potência e na sequência alguns controladores que tem sido aplicados em tais estruturas.

Inicialmente, neste capítulo, são apresentados o princípio de funcionamento dos filtros ativos de potência e a modelagem para um circuito equivalente monofásico de um FAP. Na sequência, são apresentados alguns controladores utilizados em FAP, como *deadbeat*, PID, proporcional-ressonante e, por fim, o compensador seletivo de harmônicas que foi implementado neste trabalho.

2.2. FILTROS ATIVOS DE POTÊNCIA

A principal aplicação de filtros que empregam elementos ativos é na redução de distorções harmônicas de corrente e tensão. Neste sentido, normas internacionais (IEEE STD 519-1992, 1992; IEC 61000-3, 1996) e nacionais (ANEEL - PRODIST, 2010) estabelecem valores máximos de geração de harmônicas de corrente e qualidade de energia para consumidores individuais. A combinação de filtros ativos de potência e compensadores é uma ferramenta importante para compensação de potência reativa e desequilíbrios de cargas não-lineares.

O FAP paralelo é conectado em paralelo com a rede e com a carga, a fim de drenar ou injetar corrente para tornar a corrente de saída da rede elétrica o mais próximo possível de uma função senoidal. Este filtro geralmente é empregado para compensar cargas não lineares tornando o comportamento da carga não linear, somada ao filtro, em uma carga linear para a fonte.

A Figura 1 exibe o circuito equivalente de um filtro ativo paralelo conectado a uma rede elétrica trifásica.

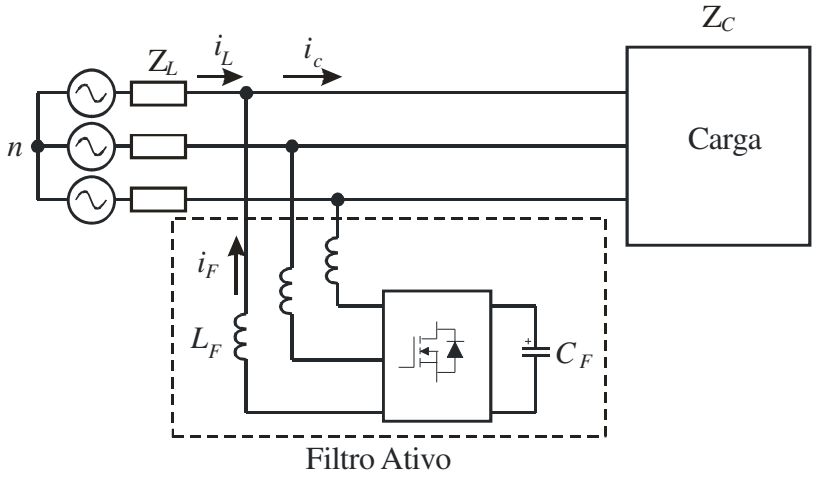


Figura 1 – Filtro ativo paralelo

Para simplificar o circuito trifásico pode-se aplicar a transformação linear de amplitude invariante nas grandezas de fase para obter o circuito equivalente monofásico conforme apresentado na Figura 2.

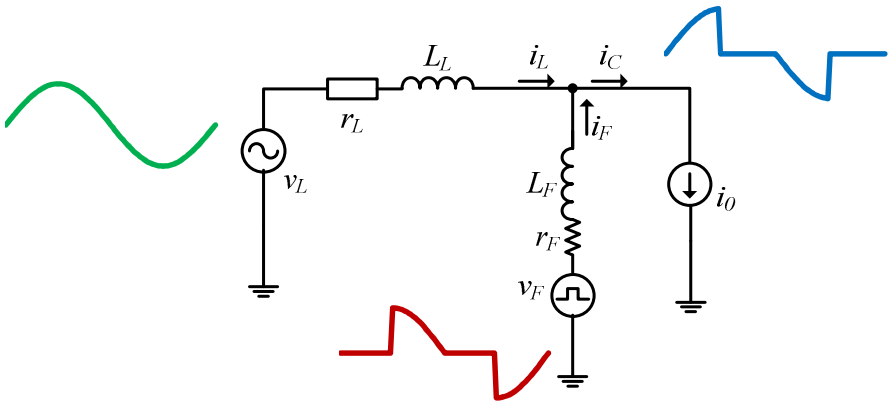


Figura 2 – Circuito equivalente monofásico do FAP conectado a uma rede trifásica

Assumido a linha como predominantemente indutiva, têm-se:

$$Z_L = r_L + sL_L \tag{2.1}$$

Desprezando as indutâncias de alta frequência Z_f e Z_C, a corrente de compensação poder ser escrita como a seguir:

$$\frac{di_F}{dt} = ai_F + bv_F + b_1 v_L + b_2 i_0 \tag{2.2}$$

sendo $a = -(r_F + r_L)/(L_F + L_L)$, $b = 1/(L_F + L_L)$, $b_1 = r_L/(L_F + L_L)$ e $b_2 = -1/(L_F + L_L)$.

O modelo discretizado para a corrente i_F pode ser representado pela seguinte equação a diferenças:

$$i_F(k+1) = g i_F(k) + h v_F(k) + h_1 v_L(k) + h_2 i_0 \tag{2.3}$$

sendo $g = e^{aT_s}$, $h = b\int_0^{T_s} e^{a\lambda} d\lambda$, $h_1 = b_1\int_0^{T_s} e^{a\lambda} d\lambda$, $h_2 = b_2\int_0^{T_s} e^{a\lambda} d\lambda$.

Conforme (STEFANELLO, 2010), para inserir o efeito de atraso de transporte associado à implementação digital pode ser definida uma variável $\psi(k+1)$ para $v_F(k)$ em (2.3). Neste caso, o modelo em espaço de estados é dado por:

$$\begin{bmatrix} i_F(k+1) \\ \psi(k+1) \end{bmatrix} = \begin{bmatrix} g & h \\ 0 & 0 \end{bmatrix} \begin{bmatrix} i_F(k) \\ \psi(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} v_F(k) + \begin{bmatrix} h_1 \\ 0 \end{bmatrix} v_L(k) + \begin{bmatrix} h_2 \\ 0 \end{bmatrix} i_0(k)$$
$$y_F(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} i_F(k) \\ \psi(k) \end{bmatrix} \tag{2.4}$$

2.3. SISTEMAS DE CONTROLE PARA FAPs

A finalidade do sistema de controle é ajustar o valor de v_F a fim de gerar o valor de i_F que compense as harmônicas na corrente de carca i_C. Algumas abordagens de controle são apresentadas nesta seção e formam uma base de comparação, utilizando controladores de execução sequencial, com os resultados obtidos nos ensaios de laboratório apresentados no Capítulo 4.

Nesta seção são discutidas algumas estratégias de controle, também como a possibilidade de ser implementada em paralelo no projeto do sistema de controle para FAPs. São abordadas técnicas clássicas como o controlador *deadbeat* e o controlador proporcional-integral-derivativo (PID – P*roportional–Integral–Derivative*). Na sequência é discutida a

estratégia de controladores proporcionais-ressonantes (PR – *Proportional-Resonant*) e a sua aplicação em FAPs, que permite a implementação paralela de cada microestrutura de controle.

O FAP com controle de realimentação mede a corrente da carga distorcida (i_C) e a corrente de saída do filtro (i_F). Um controlador com realimentação linear comanda a tensão de saída proveniente do conversor e leva o filtro a igualar os componentes harmônicos da corrente de carga (MOSSOBA e LEHN, 2003). Filtros com controle de realimentação de corrente podem evitar a ressonância, e os tamanhos dos seus componentes passivos podem ser substancialmente reduzidos em relação aos filtros passivos. Os filtros ativos também são mais adequados para a compensação do tempo de cargas variáveis, uma vez que podem eliminar uma ampla gama de harmônicos de baixa frequência.

Para cargas resistivas o filtro não necessita injetar ou drenar corrente, pois a carga já apresenta uma corrente drenada na forma senoidal e em fase com a tensão.

2.3.1. Controladores *Deadbeat*

O objetivo de um controlador *deadbeat* é adequar o sinal de saída à referência em um tempo mínimo e com erro nulo. Em geral, o sinal de saída do controlador *deadbeat* deve possuir as seguintes características:

- Apresentar erro nulo em regime estacionário.
- Mínimo tempo de adequação ao sinal de referência e mínimo tempo de subida.

Para um sistema discreto apresentar uma resposta *deadbeat*, a função de transferência em malha fechada pode ser definida conforme expressão abaixo:

$$\frac{C(z)}{R(z)} = \frac{1}{z^N} \tag{2.5}$$

na qual $C(z)$ é a saída da planta, $R(z)$ é a referência do sistema de controle e N é o atraso total no processo, em períodos de amostragem.

Em um sistema com entrada degrau unitário pode-se ter a resposta *deadbeat* como:

$$C(z) = z^{-N} + z^{-N-1} + z^{-N-2} + \ldots + z^{-N-n} \tag{2.6}$$

sendo n a ordem da planta.

Em aplicações como FAPs e UPSs, a técnica *deadbeat* pode apresentar bom desempenho para uma carga linear, mas geralmente não consegue manter este desempenho na presença de cargas variáveis ou cargas não lineares.

O controlador *One Sample Ahead Preview* (OSAP) consiste em uma modificação no controlador *deadbeat*, apresentada em 1988 por Kawamura, Haneyoshi e Hoft (1988). Dentre as vantagens do esquema proposto destacam-se:

- Baixa Distorção Harmônica Total (THD);
- Resposta rápida a transitórios;
- Utilização de apenas um sensor de tensão;
- Estabilidade para várias condições de cargas;
- Aplicável para sistemas trifásicos.

Os controles *deadbeat* e OSAP apesar de sua simplicidade e velocidade de resposta apresentam bons resultados para cargas conhecidas, mas são instáveis a variações de parâmetros ou a existência de dinâmicas não modeladas.

2.3.2. Controladores PID

O controlador PID tem sido utilizado em uma grande variedade de processos, incluindo em sistemas e controle de geração elétrica, fontes ininterruptas de energia, filtros ativos, entre outros. O mesmo é formado por um conjunto de três ações de controle: proporcional (P), integral (I) e derivativa (D). Este controlador é usualmente representado em tempo contínuo por:

$$U(t)=K_p e(t)+K_i \int e(t)dt + K_d \frac{d}{dt} e(t) \tag{2.7}$$

sendo K_p o ganho proporcional, K_i o ganho integral e K_d o ganho derivativo e expressos em função do tempo de *reset* integral e do tempo derivativo, conforme segue:

$$\begin{cases} K_p = K \\ K_i = \dfrac{K}{T_i} \\ K_d = KT_d \end{cases} \tag{2.8}$$

Em tempo discreto a expressão acima pode ser colocada na seguinte forma:

$$u_k = K_p e_k + K_i \sum_{i=0}^{k} \frac{e_i + e_{i\text{-}1}}{2} + K_d (e_k - e_{k\text{-}1}) \tag{2.9}$$

O controlador PID, isoladamente, assim como *deadbeat* e OSAP, é de simples implementação. Por outro lado, para que esta técnica seja projetada com uma robustez maior, geralmente se tem uma perda de desempenho. De forma geral, é inevitável a presença de transitórios em distúrbios, não compensando significativamente harmônicas decorrentes destes distúrbios. Além disso, o uso de controladores PID para se fazer as transformações para

eixos estacionários exige um número elevado de transformações DQ, visto que para cada harmônica é preciso um controlador. Estas transformações exigem um grande número de cálculos complexos, que podem tornar impraticável a aplicação do referido controlador.

Os controladores proporcionais-ressonantes atuam de maneira síncrona ao sistema e são uma solução interessante para evitar as transformações DQ. Esta estratégia de controle pode ser utilizada para reduzir a corrente harmônica do lado da fonte do sistema (JUNG e SUL, 2009) e permite a execução de cálculos em uma estrutura paralela. Como esta proposta é embasada em paralelismo de ações de controle, o uso de uma técnica que permita a execução de cálculos em paralelo, como controladores PR, é uma escolha interessante.

2.3.3. Controladores Proporcionais-Ressonantes (PR)

A ideia básica do controlador PR é a transformação de um controlador PI em um controlador com compensação para sinais alternados (ZMOOD e HOLMES, 2003). Este controlador combina um laço de controle proporcional e um laço de controle ressonante.

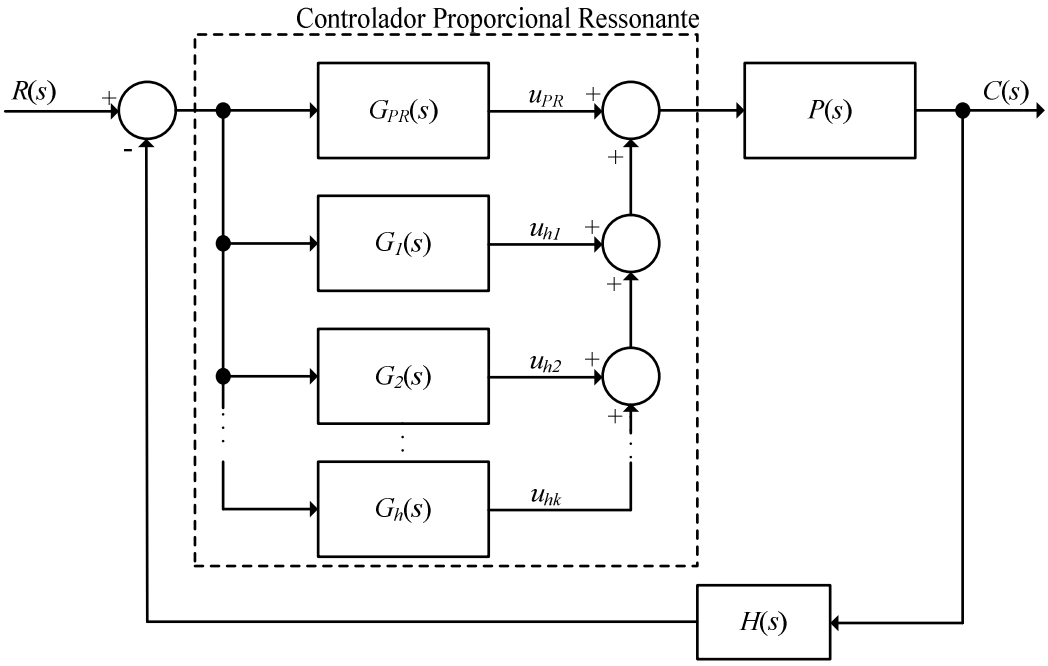


Figura 3 – Controlador Proporcional Ressonante

Um controlador PR em tempo contínuo introduz um ganho infinito, em malha aberta, em uma frequência de ressonância selecionada para eliminar o erro estacionário desta frequência. A Figura 3 exibe um diagrama do controlador PR, no qual é possível visualizar a

estrutura paralela formada pelos compensadores harmônicos (HC - *Harmonic Compensator*) representados por $G_h(s)$ e pelo controlador proporcional na frequência fundamental $G_{PR}(s)$.

O controlador PR no domínio do plano *s* é exibido a seguir:

$$G_{PR}(s) = K_p + \frac{K_r s}{s^2 + \omega_0^2} \tag{2.10}$$

sendo K_p o ganho proporcional, K_r é o ganho ressonante e ω_0 é a frequência de ressonância.

O compensador harmônico HC $G_h(s)$ pode ser definido como em Zmood e Holmes (2003):

$$G_h(s) = K_{Ih} \frac{s}{s^2 + (\omega_0 h)^2} \tag{2.11}$$

sendo que K_{Ih} é o ganho integral do compensador de cada harmônica (h = 3,5,7…).

Em Sera et al (2005) um termo adicional deve ser adicionado para obter ganho ajustável na ressonância (2.12), chamada frequência de corte, ω_c.

$$G_{PR}(s) = K_p + K_r \frac{s + \omega_c}{s^2 + 2\omega_c s + \omega_c^2 + \omega_0^2} \tag{2.12}$$

Para os compensadores harmônicos (2.12) pode ser reescrita como:

$$G_{PRh}(s) = K_{rh} \frac{s + \omega_{ch}}{s^2 + 2\omega_{ch} s + \omega_{ch}^2 + (h\omega_0)^2} \tag{2.13}$$

A forma discretizada do controlador PR pode ser vista em (2.14).

$$H_{PR}(z) = K_P + \frac{K_R}{\omega_0^2 + \omega_c^2} \left[\frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}} \right] \tag{2.14}$$

Estes filtros ressonantes oferecem maior domínio sobre o comportamento do controlador em função da frequência. Entretanto, deve-se perceber que utilizando o operador "*z*" pode-se gerar erro de ressonância, uma vez que é necessária uma precisão elevada de seus coeficientes, especialmente em frequências de amostragem mais elevadas (NEWMAN e HOLMES, 2003).

Para uma simulação de um filtro de ressonância e validação da técnica analisada foram escolhidos os mesmos valores de ganhos utilizados por (SERA et al., 2005). Neste caso, K_p = 800 e K_r = 5, para uma frequência de ressonância, ω_0 = 60 Hz e a menor frequência de interrupção da função de transferência, ω_c = 1 Hz. Este sistema tem uma resposta em frequência mostrada na Figura 4. A fase de saída em estado estacionário e o erro de magnitude alcançada por este compensador serão aproximadamente zero, desde que se obtenha um ganho relativamente alto na frequência de referência.

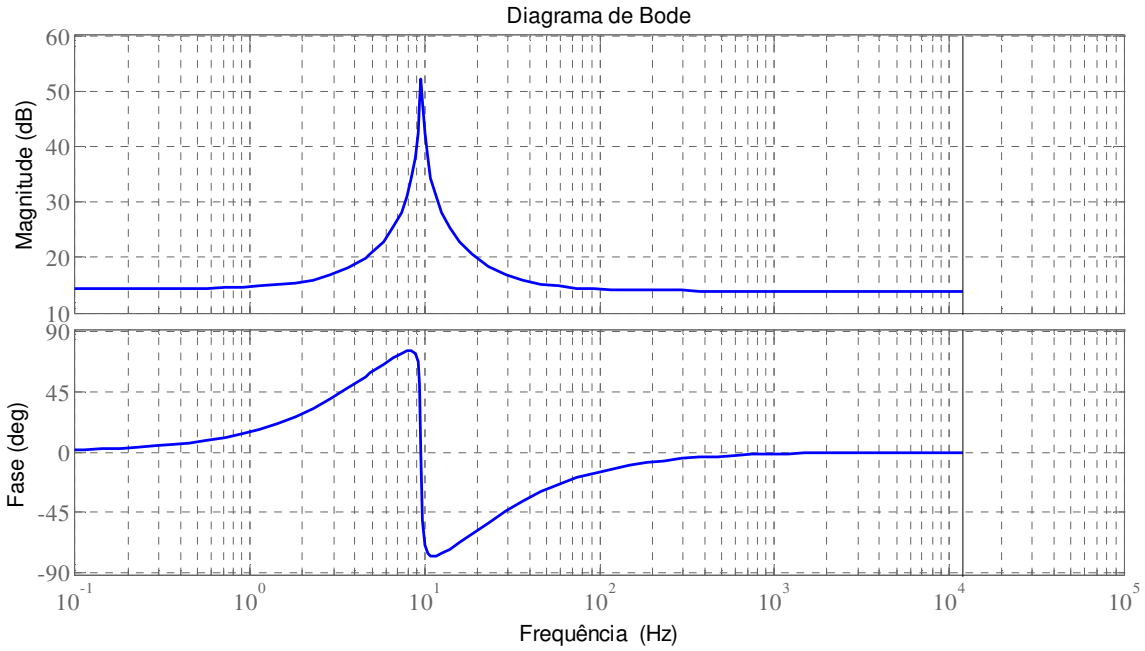


Figura 4 – Diagrama de Bode sem compensação harmônica de (2.12)

A compensação harmônica em várias frequências selecionadas (3ª, 5ª, 7ª e 9ª) é apresentada na Figura 5. Neste caso, modificações de ganho ocorrem praticamente somente nas frequências escolhidas.

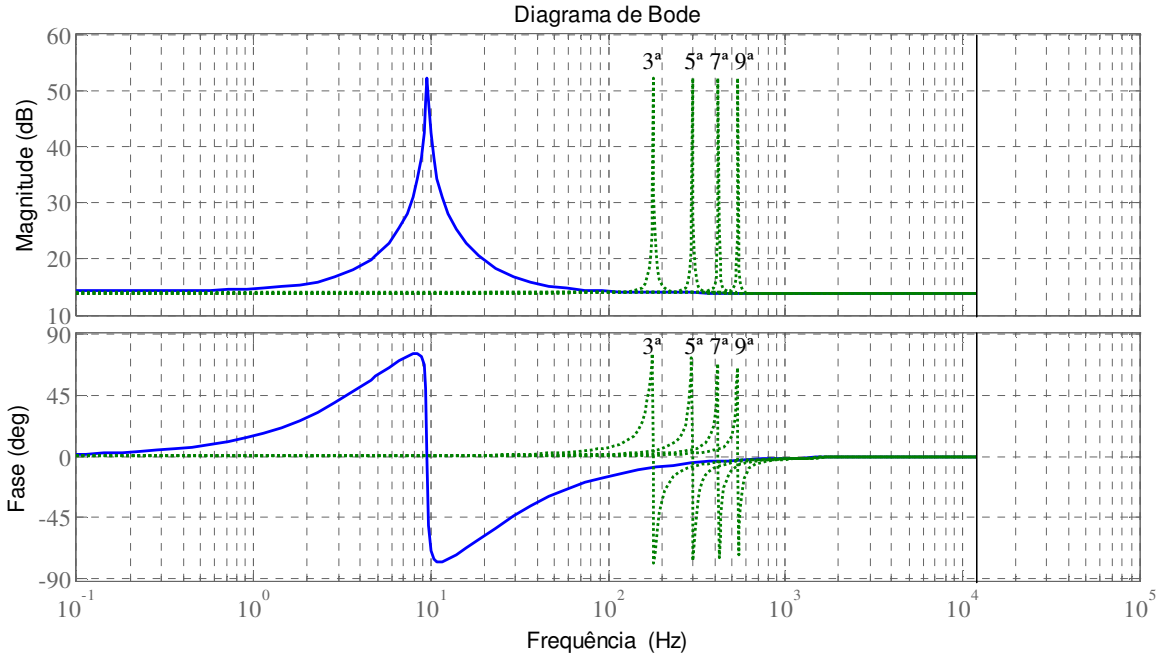


Figura 5 – Diagrama de Bode do controlador PR com compensação harmônica (2.13)

A linha pontilhada na Figura 5 indica altos ganhos para as harmônicas que estão sendo selecionadas pelo controlador.

Em (SERA et al., 2005) é apresentada uma discussão sobre a precisão do operador Delta em relação à baseada em operador de deslocamento na implementação de controladores ressonantes. O erro utilizando operador de deslocamento é desprezível em baixas frequências de amostragem, mas torna-se acentuado com o aumento desta frequência. O ganho do controlador é deslocado da frequência de ressonância podendo torna-lo reduzido na frequência que realmente se tem interesse. Estes problemas podem ser minimizados utilizando o operador Delta, que pode ser visto no APÊNDICE A – Operador Delta.

2.3.4. Compensador Seletivo de Harmônicas

Com o objetivo de superar problemas relacionados a limitação de frequência de comutação, que em muitos casos pode não ser suficientemente elevada para compensar as harmônicas necessárias, vários esquemas foram propostos (MATTAVELLI e FASOLO, 2000), *deadbeat*, realimentação de estado ideal, entre outros (DOTE e HOFT, 1998).

Uma solução diferente baseada em controladores repetitivos, destinados a eliminar distúrbios periódicos é apresentada por Von Jouanne (1996), que utiliza a Transformação Discreta de Fourier (DFT – *Discrete Fourier Transform*) para fazer a análise das harmônicas e a compensação por controladores integrais. Porém esta solução não permite a compensação de desequilíbrio e sua resposta transitória é fortemente limitada pela filtragem DFT.

Seguindo este raciocínio, Mattavelli e Fasolo (2000) implementam um controle de eixos síncronos para algumas frequências selecionadas, onde utiliza rotação de eixos para componentes de sequencia negativa e positiva das frequências selecionadas. Também sugere um controle convencional para controle de variações de carga e faz várias modificações para aplicações em ponto fixo, reduzindo erros de quantização. A implementação da solução descrita por Mattavelli e Fasolo (2000) utiliza DSP e processamento sequencial, e será descrita a seguir.

A Figura 6 mostra o diagrama de um sistema de controle síncrono aplicado a um filtro ativo de potência, com a implementação do compensador seletivo de harmônicas, que será abordado na Seção 2.3.4, e um controlador convencional. Neste sistema o FAP auxilia o

sistema de potência com a corrente i_F para realizar a correção da corrente da carga não linear i_L. Desta forma, a parcela harmônica é fornecida pelo FAP e a corrente i_S pode ser senoidal.

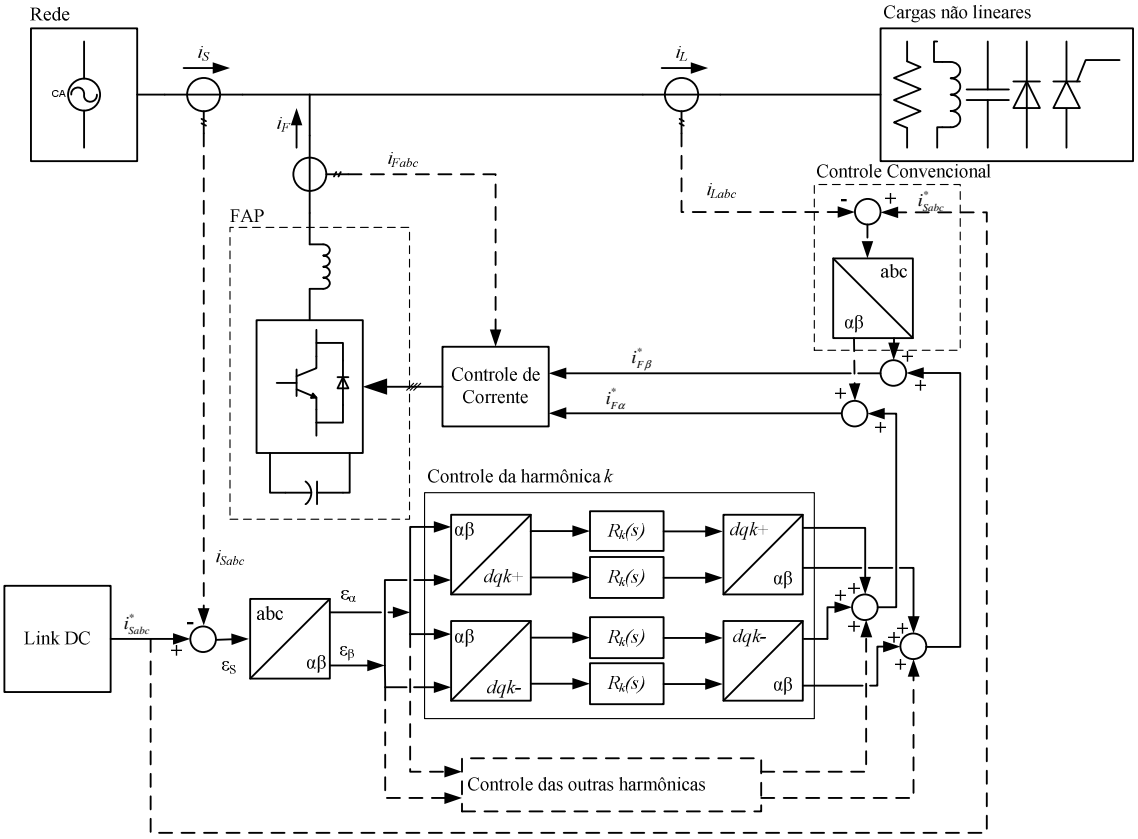


Figura 6 – Diagrama em blocos do sistema de potência e do sistema de controle síncrono.

O diagrama (Figura 6) mostra o controle da harmônica *k* utilizando a diferença entre a corrente de referência i^*_{Sabc} e a corrente da rede i_{Sabc}. Cada harmônica que se deseja atenuar precisa de um controlador que pode ser implementado em paralelo com o controlador da harmônica *k*. Um controlador convencional é adicionado para ajuste da corrente de carga, i_L. A corrente do filtro, i_F, é controlada pelo filtro ativo de potência, utilizando as correntes de referência do filtro $i^*_{F\alpha}$ e $i^*_{F\beta}$, geradas pelo compensador, e a própria corrente medida na saída do filtro, i_{Fabc}.

A Figura 7 mostra o diagrama em blocos de um controlador de eixo síncrono. Neste controlador as entradas ε_α e ε_β são erros de corrente na saída do sistema, $I_{F\alpha*}$ e $I_{F\beta*}$ são as correntes de referência. A compensação é realizada pelos reguladores R_k, que garantem erro nulo de estado estacionário para as sequencias positivas e negativas de cada componente harmônico. As saídas dos reguladores são convertidas novamente para eixos de referência

estacionária com a adição, se necessário de um ângulo de deslocamento, ϕ_k, para compensar atrasos do restante do processo.

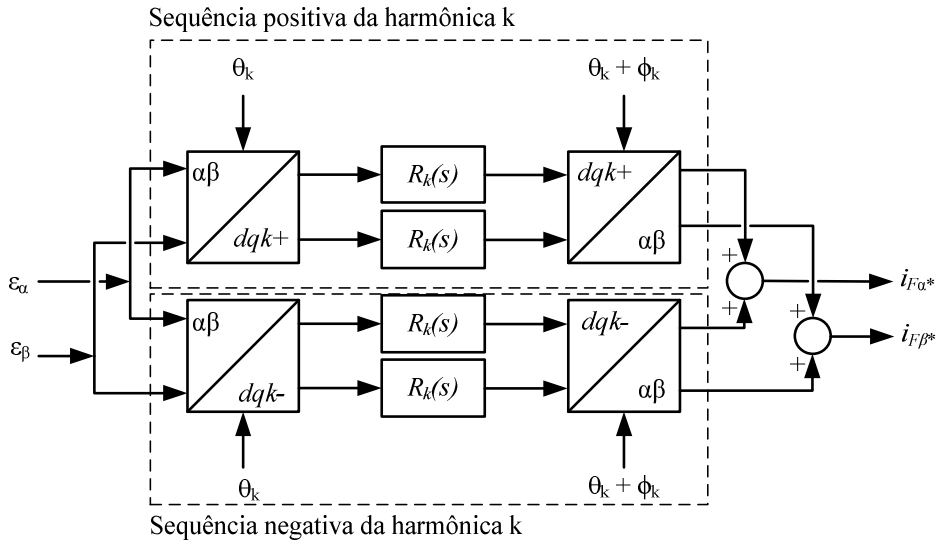


Figura 7 – Diagrama em blocos do controle de eixo síncrono.

A compensação das sequencias positivas e negativas da harmônica *k* é equivalente a demodulação síncrona dos componentes *αβ* apresentados na Figura 8.

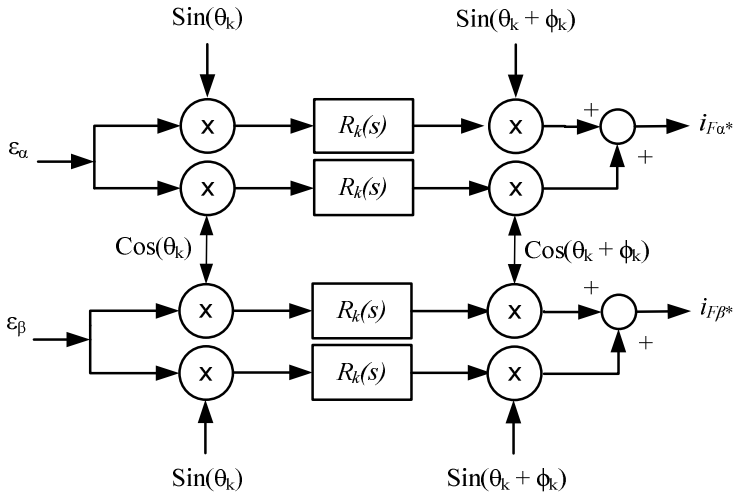


Figura 8 – Demodulação síncrona dos componentes αβ

Simplificando a Figura 8 pode-se redesenhar o compensador como o esquema apresentado na Figura 9.

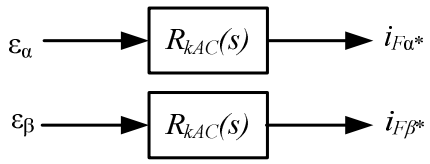


Figura 9 – Regulador de eixo estacionário.

O regulador $R_{kAC}(s)$ por ser matematicamente representado por:

$$\begin{aligned} R_{kAC}(s) = \cos\phi_k \left[R_k(s - jk\omega_0) + R_k(s + jk\omega_0) \right] + \ldots \\ \ldots + j\sin\phi_k \left[R_k(s - jk\omega_0) - R_k(s + jk\omega_0) \right] \end{aligned} \tag{2.15}$$

A técnica descrita anteriormente permite eliminar erros de estado estacionário para as frequências selecionadas, mas não possui um desempenho dinâmico necessário a aplicações de potência. Motivado por este raciocínio, Mattavelli e Fasolo (2000) propôs um esquema de controle de 3 camadas, sendo um termo proporcional K_p para resposta de alta velocidade e a divisão dos controladores rotacionais para os componentes fundamentais foram separados da regulação dos componentes harmônicos.

Para reduzir os erros de regime estacionário foi inserido um integrador tanto nas sequencias positivas quanto nas negativas:

$$R_k(s) = \frac{K_{Ik}}{s} \tag{2.16}$$

Utilizando a transformação (2.15) e o regulador (2.16) a equivalência pode ser escrita como:

$$R_{kAC}(s) = 2K_{Ik}\left(\frac{s\cos(\phi_k) - k\omega_s \sin(\phi_k)}{s^2 + (k\omega_s)^2} \right) \tag{2.17}$$

A fim de implementar uma solução para ponto fixo, Mattavelli e Fasolo (2000) propõe uma série de modificações e aproximações, que são descritas abaixo, partindo da função de transferência desejada para o controle de harmônicas.

Considerando $\phi_k = 0$ tem-se:

$$R_{hAC}(s) = \sum_{h\in N_h} \frac{2K_{Ih}s}{s^2 + (h\omega_0)^2} = K_F \sum_{h\in N_h} \frac{F_h}{1 - F_h} \approx K_F \frac{\sum_{h\in N_h} F_h}{1 - \sum_{h\in N_h} F_h} \tag{2.18}$$

sendo que N é o número de pontos por ciclo e h é o índice da harmônica.

Em (2.18) K_F é definido como:

$$K_F = \frac{K_{Ih}}{\xi_h h\omega_0} \tag{2.19}$$

sendo que o fator de amortecimento do filtro passa banda é $\xi_h = 1/2h\omega_0$. Substituindo ξ_h em (2.19) tem-se:

$$K_F = 2K_{Ih} \tag{2.20}$$

O filtro F_h pode ser expresso como:

$$\frac{F_h}{1-F_h} = \frac{s}{s^2 + (h\omega_0)^2} \tag{2.21}$$

sendo que F_h é um filtro passa banda com ganho unitário e fase zero na harmônica h, e com boa seletividade pode ser definido como:

$$F_h = \frac{s}{s^2 + 2\xi_h h\omega_0 s + \omega_0^2} \tag{2.22}$$

Substituindo (2.22) em (2.21) tem-se

$$F_h = \frac{s}{s^2 + (2\xi_h h\omega_0 - 1)s + (h\omega_0)^2} \tag{2.23}$$

Fazendo $2\xi_h h\omega_0 = 1$ elimina-se o segundo termo do denominador, resultando em:

$$F_h = \frac{s}{s^2 + (h\omega_0)^2} \rightarrow \cos(h\omega_0 t) \tag{2.24}$$

Substituindo (2.24) em (2.18), e com objetivo de se obter a forma discreta de implementação, pode-se determinar os coeficientes da DFT para cada harmônica:

$$\begin{aligned}
F_1(z) &= \frac{2}{N}\sum_{i=0}^{N-1}\cos\left(\frac{2\pi}{N}1i\right)z^{-i} = \alpha_0 + \alpha_1 z^{-1} + a_2 z^{-2} + ... + \alpha_{N-1} z^{-N+1} \\
F_2(z) &= \frac{2}{N}\sum_{i=0}^{N-1}\cos\left(\frac{2\pi}{N}2i\right)z^{-i} = \beta_0 + \beta_1 z^{-1} + \beta_2 z^{-2} + ... + \beta_{N-1} z^{-N+1} \\
&\vdots \\
F_N(z) &= \frac{2}{N}\sum_{i=0}^{N-1}\cos\left(\frac{2\pi}{N}(N-1)i\right)z^{-i} = \zeta_0 + \zeta_1 z^{-1} + \zeta_2 z^{-2} + ... + \zeta_{N-1} z^{-N+1}
\end{aligned} \tag{2.25}$$

A soma dos coeficientes de mesmo índice resulta em um único vetor de coeficientes Ω, que possui o mesmo tamanho *N*, sendo que *N* é o número de pontos de amostragem por ciclo. Ainda, é de grande importância notar que o tamanho do vetor não depende do número de harmônicas selecionadas.

$$\Omega_i = \alpha_i + \beta_i + ... + \zeta_i \quad , \quad i = [1, N] \tag{2.26}$$

No domínio de tempo discreto tem-se:

$$y(k) = \Omega_0 x(k) + \Omega_1 x(k-1) + \Omega_2 x(k-2) + ... + \Omega_{N-1} x(k-N+1) \tag{2.27}$$

Para concluir, o ganho K_F (2.19) depende de K_{Ih}, o qual pode ser expresso, segundo Mattavelli e Fasolo (2000), por:

$$K_{Ih} = \frac{2,2}{n_{pk} T_S} \tag{2.28}$$

sendo n_{pk} o número de ciclos da fundamental em que se deseja obter a resposta dinâmica e T_S é o período de amostragem.

Para verificar a capacidade de filtragem do filtro digital FIR apresentado em (2.27) são determinados os coeficientes para as 3ª, 5ª e 9ª harmônicas, utilizando (2.25) e N=200. A análise em frequência de (2.27) é apresentada na Figura 10, que mostra o diagrama de bode de magnitude. Conforme se percebe o filtro apresenta ganho unitário (0dB) nas frequências selecionadas e ganho reduzido nas demais frequências harmônicas.

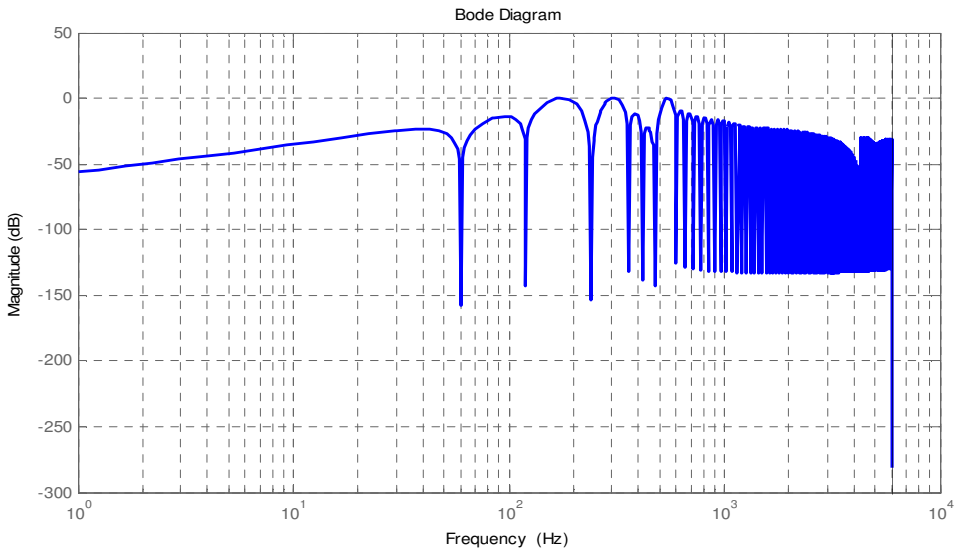


Figura 10 – Diagrama de Bode do compensador seletivo.

Para verificar a atuação do compensador somente nos pontos de interesse, foi adicionada também no sinal de entrada a 7ª harmônica, além das demais descritas. O resultado da simulação do compensador seletivo, apresentado na Figura 11, permite verificar que o compensador apenas atenuou as harmônicas selecionadas, de forma que no sinal de saída, após a compensação, a 7ª harmônica permanece com valor semelhante ao do sinal de entrada, da mesma forma que a parcela na frequência fundamental.

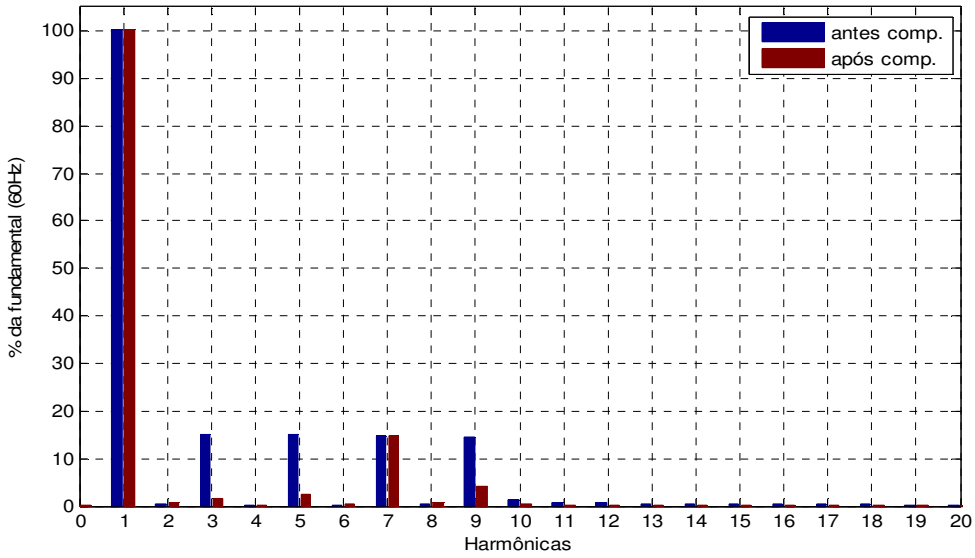


Figura 11 -- Análise harmônica da simulação computacional do compensador seletivo de harmônicas com 200 pontos: antes e após a compensação.

2.4. SUMÁRIO

A implementação dos controladores ressonantes utilizando as técnicas descritas em (SERA et al., 2005) resulta em precisão suficiente em sistemas com implementação em ponto fixo. Desta forma, esta estrutura de controle pode ser utilizada para aplicação em filtros ativos de potência e processada através de dispositivos de ponto fixo em paralelo. Diversos trabalhos (SERA et al., 2005; MIDDLETON e GOODWIN, 1990) tem relatado bons resultados nestas aplicações. Entretanto, os mesmos geralmente tem aplicado a técnica apenas para compensação de um conjunto reduzido de harmônicas, sendo que este número de harmônicas selecionadas é proporcional à exigência computacional necessária.

O compensador seletivo de harmônicas proposto por Mattavelli e Fasolo (2000) também permite boa precisão em implementações de ponto fixo e reduzida sensibilidade a erros de quantização. Por outro lado, a exigência computacional deste compensador independe do número de harmônicas que se deseja eliminar. Neste trabalho, se propõe o uso

de sistemas de *hardware* programável para implementação paralela do compensador seletivo de harmônicas visando alta velocidade de cálculo e a compensação de um conjunto mais amplo de componentes harmônicas. No capítulo seguinte são apresentadas algumas técnicas de processamento paralelo e definição da técnica mais apropriada a este trabalho.

3. IMPLEMENTAÇÃO DE ESTRUTURAS PARALELAS EM FPGA

3.1. INTRODUÇÃO

Os FPGA são dispositivos de processamento de dados reprogramáveis que permitem a simulação de combinações lógicas das mais simples a circuitos de extrema complexidade. Neste capítulo é apresentado inicialmente uma discussão a respeito de FPGAs e ferramentas para programação destes dispositivos. Visando a aplicação dos mesmos para implementação de controladores paralelos, na sequência, é discutida a implementação de operações matemáticas em FPGA, processamento paralelo, e questões sobre paralelismo em dispositivos FPGA.

3.2. FPGA e VHDL

Os FPGA são compostos de matrizes de blocos lógicos configuráveis (CLB) que podem ser configurados para assumir funções específicas no circuito. A combinação dos CLB, bem como as suas funções são descritas por uma linguagem de descrição de *hardware* (HDL) como *Verilog* HDL ou VHDL (*Very High Speed Integrated Circuit Hardware Description Language*).

A linguagem VHDL é padronizada pela IEEE-1076 (IEEE STD 1076-2008, 2008) e foi desenvolvida por dois motivos (BROWN e VRANESIC, 2004): 1 – ser utilizada como linguagem de documentação de circuitos digitais complexos, padronizada pala IEEE. 2 – a linguagem VHDL fornece recursos para definir o comportamento dos circuitos digitais, permitindo assim a sua aplicação em *softwares* de desenvolvimento e simulação.

A Figura 12 mostra um circuito representativo de duas entradas, uma porta AND, uma porta OR e duas saídas, que em linguagem VHDL é descrito conforme código da Figura 13.

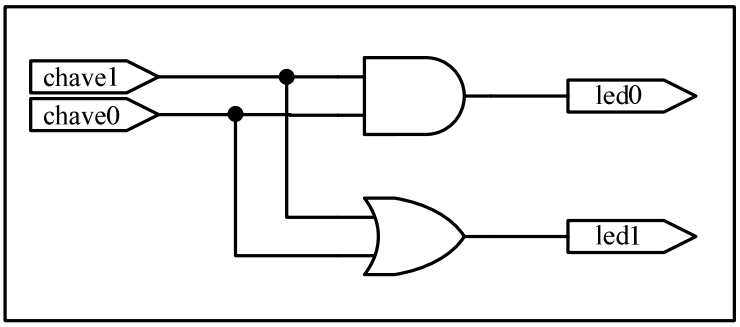


Figura 12 – Circuito de exemplo de representação de portas lógicas em VHDL

```
entity principal is
    Port ( led0 : out  STD_LOGIC;
           led1 : out  STD_LOGIC;
           chave0 : in  STD_LOGIC;
           chave1 : in  STD_LOGIC);
end principal;

architecture Behavioral of principal is
begin
   led0 <= chave0 and chave1;
   led1 <= chave0 or chave1;
end Behavioral;
```

Figura 13 – Código de exemplo escrito em VHDL

3.3. OPERAÇÕES ARITMÉTICAS EM UM FPGA

O núcleo do processamento digital em FPGA é o multiplicador e acumulador (MAC - *Multiply and Accumulate*), uma implementação eficiente do algoritmo de processamento é a base para o desempenho das operações aritméticas (SHANTHALA e KULKARNI, 2009).

A Figura 14 mostra a estrutura básica de um MAC, pode ser visto a entrada de duas palavras de n bits no multiplicador, resultado em uma palavra de $2n$ bits e na sequência a passagem pelo acumulador que deve ser de $2n+1$ bits com o bit de *carry*.

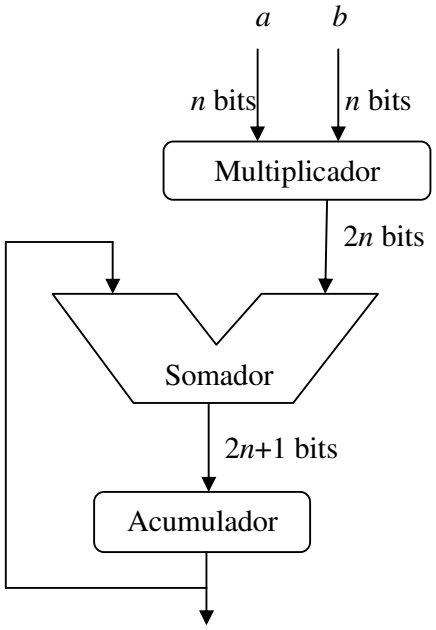


Figura 14 – Estrutura básica de um MAC

O Multiplicador utiliza um CLB e uma porta *AND* de duas entradas para cada *bit* do produto. O número total de CLB é n^2. No exemplo de uma palavra de 8 bits são necessários 64 CLB para compor o multiplicador. A Figura 15 ilustra uma célula de um multiplicador com circuito adicional necessário para a combinação da matriz celular que resulta no multiplicador completo de *n bits* apresentado na Figura 16.

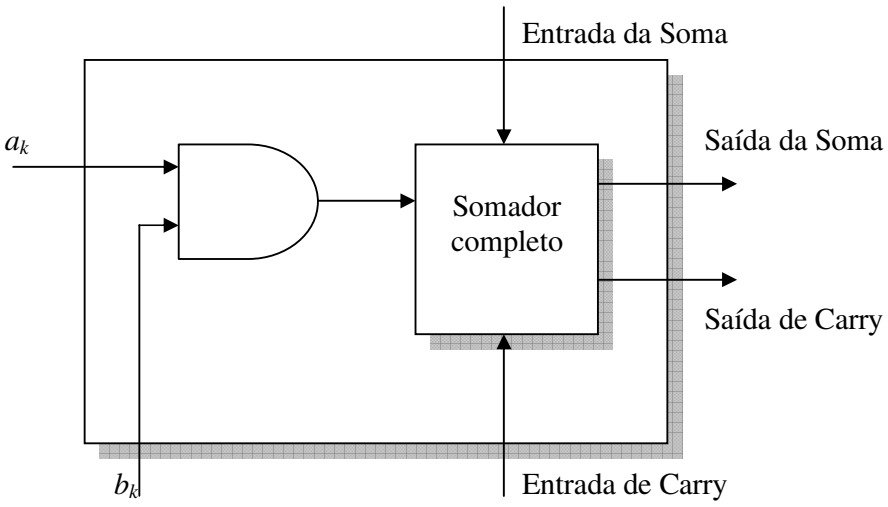


Figura 15 – Célula de um multiplicador combinacional

A Figura 16 demonstra uma secção do diagrama em blocos de multiplicador de 8 bits, onde *a* e *b* são as palavras de 8 bits a serem multiplicadas e *S* o resultado.

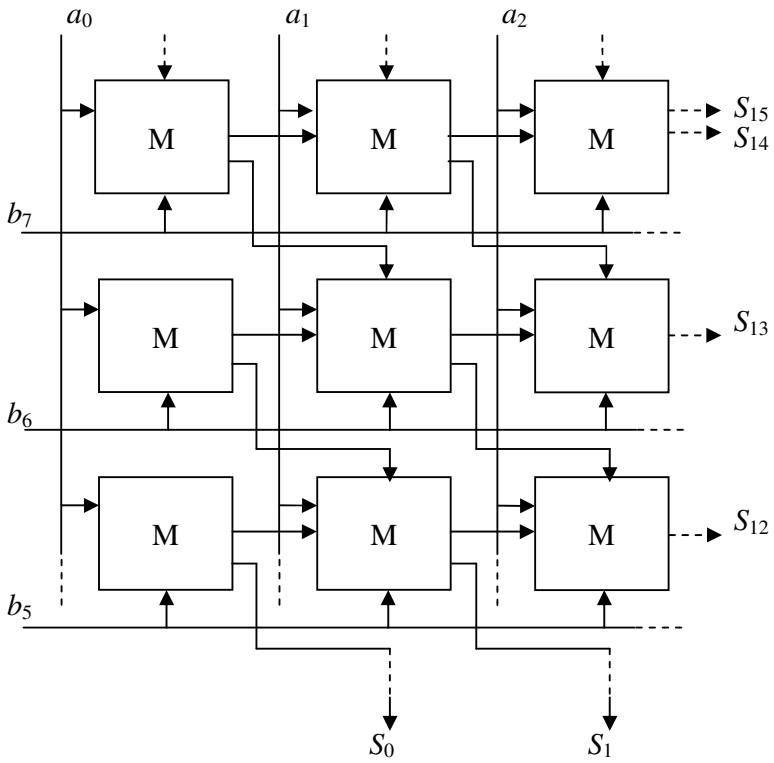


Figura 16 – Parte do Diagrama em blocos de um multiplicador combinacional de 8 bits

O MAC é composto do multiplicador e de um acumulador. O acumulador pode ser desenvolvido utilizando o *bit* de *carry* do CLB, ativo por configuração. Dois geradores de função de 4 entradas podem ser configurados como um somador de 2 bits e combinados para qualquer tamanho necessário.

3.4. PROCESSAMENTO PARALELO

O processamento paralelo surgiu da necessidade de aumento da capacidade de processamento dos computadores. Este aumento poderia ser pelo desenvolvimento de processadores mais velozes, o que esbarrava em custo e tecnologia ainda indisponível, ou no emprego de um conjunto de processadores trabalhando em paralelo (STALLINGS, 2002).

Nos primeiros trabalhos de Von Neumann (NEUMANN, 1945), as arquiteturas paralelas já eram discutidas como forma de melhorar a capacidade de processamento. Em 1920, Vanevar Bush, apresentava um computador analógico capaz de resolver equações diferenciais em paralelo (DE ROSE e NAVAUX, 2008). A utilização da técnica de *pipeline*, uma das primeiras técnicas de processamento seguinte à utilização de acesso simultâneo à memorias divididas em blocos, surge em 1959 com as máquinas STRETCH (BLOCH, 1959) e LARC (ECKERT et al., 1959).

A técnica de *pipeline* apresenta problemas de dependências de instruções, onde uma instrução depende de outra para dar sequência a um cálculo. Este problema foi minimizado com o surgimento das memórias pequenas e de acesso rápido, chamadas memórias *cache*. Com a memória *cache* também desenvolveu-se uma evolução da técnica *pipeline*, com previsão e escalonamento de instruções para reduzir a dependência entre elas, técnica esta chamada de arquitetura superescalar. Ambas as técnicas e suas variações são apresentadas a seguir.

3.4.1. Pipeline

Uma das primeiras formas de aumentar a velocidade de processamento pela execução de instruções em paralelo foi a técnica *pipeline*. Inicialmente uma forma de *pipeline* a nível de memória foi utilizada no IBM 7094, mas efetivamente a nível de instrução foi utilizada nas máquinas de STRECH e *LARC* em 1959.

O princípio de funcionamento do *pipeline* é a subdivisão de uma tarefa em subtarefas menores, que ao final da execução equivalem a tarefa inicial. Para funcionamento adequado as subtarefas não podem depender de resultados de tarefas concorrentes.

As subtarefas devem possuir tamanhos semelhantes e o tempo de processamento final será igual ao tempo da subtarefa mais lenta.

A eficiência do pipeline pode ser medida pelo *speedup*, que é a razão entre o tempo de processamento sequencial e o tempo de processamento com aplicação do *pipeline*, conforme equação (3.1).

$$S = \frac{T_s}{T_p} = \frac{n.k}{k + (n-1)} \tag{3.1}$$

sendo que S é o *speedup*, T_s o número de períodos de *clock* necessários para o processamento sequencial, T_p o número de períodos de *clock* necessários para o processamento do pipeline, n

é o número de estágios do pipeline e *k* é o número de ciclos necessários para terminar a primeira tarefa.

Quanto maior o valor de *k* em relação a *n*, mais o valor do *speedup* tenderá a *k*, seu valor ideal. Utilizando o exemplo da Figura 17, são necessárias 3 etapas de 5 ciclos de *clock* cada para o processamento sequencial (T_s). Utilizando processamento paralelo com *pipeline* são necessários 7 ciclos e o *speedup* é igual a 2,14.

A técnica *pipeline* pode ser classificada em três tipos (DE ROSE e NAVAUX, 2008), *Pipeline* de Instrução, *Pipeline* Aritmético e *Pipeline* de Processador, suas características são descritas a seguir.

3.4.1.1. *Pipeline* de Instrução

É o tipo mais comum de emprego da técnica *pipeline* e consiste em buscar instrução na memória, decodificá-la, buscar operando na memória e executar. Está divisão pode ser maior ou menor em função do tamanho do processador. Hoje em dia é utilizada em quase todos os microprocessadores e é chamada de *instruction lookahead.*

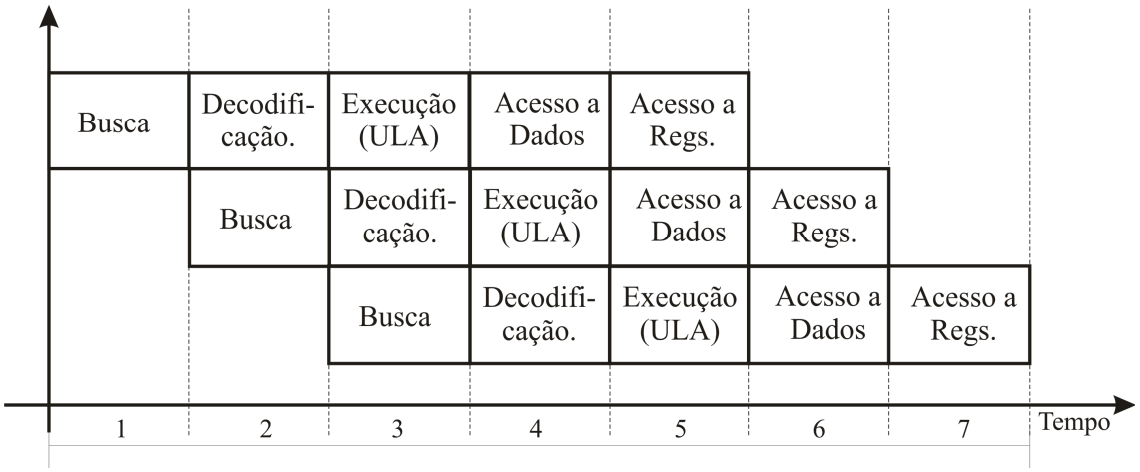


Figura 17 – Organização de operações no *pipeline* de instrução

No *pipeline* de instrução o processamento de uma instrução é divido em três etapas, quando necessário ainda pode ser adicionada uma etapa de acesso à memória e/ou registradores. As três etapas iniciais consistem:

- Busca de Instrução: Leitura da próxima instrução a ser executada;
- Decodificação: Decodificação da instrução e leitura do estado dos registradores;

- Execução: Processamento dos dados utilizando a ULA.

Considerando uma instrução que necessite acesso a memória de dados e registradores, um *pipeline* de instrução pode ser representado pela Figura 17.

O *Pipeline* de instrução otimiza o *hardware* para atender a um maior número de operações por ciclo de *clock*. Na Figura 17, por exemplo, é possível ver a ULA sendo utilizada em sequência nos ciclos 3, 4 e 5, enquanto que em uma execução sequencial, com 5 ciclos por instrução, a ULA só retornaria ao trabalho após aguardar estes 5 ciclos. No FPGA é possível criar diversos recursos de *hardware*, como uma ULA, para serem executados ao mesmo tempo, não havendo obrigatoriamente a necessidade de utilizar a técnica de *pipeline* de instrução, uma vez que o processador pode ser personalizado para atender as necessidades dos processos.

3.4.1.2. *Pipeline* Aritmético

É a forma mais complexa e consiste na divisão da unidade lógica de processamento em vários blocos para executarem separadamente cada instrução. Pode realizar tarefas em um tempo menor que a pipeline de instrução e é empregada em supercomputadores e processadores mais modernos. A complexidade deste pipeline se dá pela necessidade de domínio dos cálculos a serem executados, de forma que as dependências sejam isoladas e não sejam executadas em ciclos de *clock* incorretos.

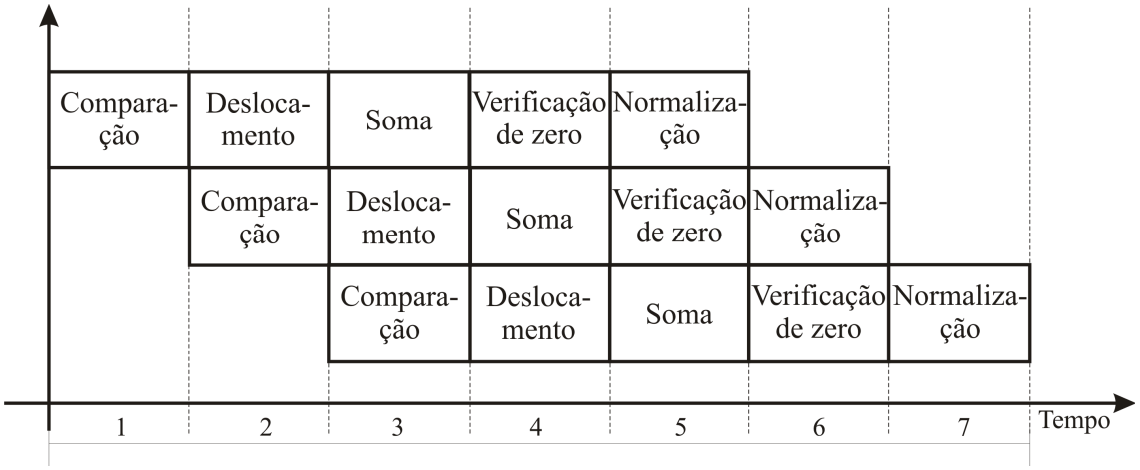


Figura 18 – Estágios do *pipeline* aritmético

A Figura 18 mostra três processos executados utilizando pipeline *aritmético*, cada processo contém os estágios de comparação, deslocamento, soma, verificação de zero e normalização. Entre os estágios de cálculo os dados resultantes são armazenados em registradores chamados de registradores de *pipeline.*

O princípio básico serve para todos os tipos de *pipeline*, ou seja, para paralelizar os estágios eles precisam ser divididos em várias subtarefas que podem ser executadas ao mesmo tempo, desde que deslocadas em um ciclo de máquina uma das outras, como demostrado na Figura 18. Enquanto a primeira subtarefa está no estágio de deslocamento, a próxima tarefa inicia o estágio de comparação, paralelamente. Na soma da primeira subtarefa, a segunda estará no deslocamento e uma terceira pode iniciar a comparação.

O controlador proposto neste trabalho necessita de vários cálculos e pode ser implementado utilizando a estrutura de *pipeline* aritmético.

3.4.1.3. *Pipeline* de Processador

Consiste na utilização de vários processadores, cada um executando uma subtarefa, cada processador ao encerrar a sua subtarefa guarda o resultado em uma memória e ao final de todas as subtarefas os dados são agrupados.

Com a utilização desta técnica associada com o pipeline aritmético, ou seja, além de vários processadores cada um implementa ainda um pipeline aritmético, o tempo de processamento pode ser reduzido significativamente.

A estrutura apresentada na Figura 19 ilustra a divisão de uma tarefa em subtarefas e estas são divindades entre diversos processadores.

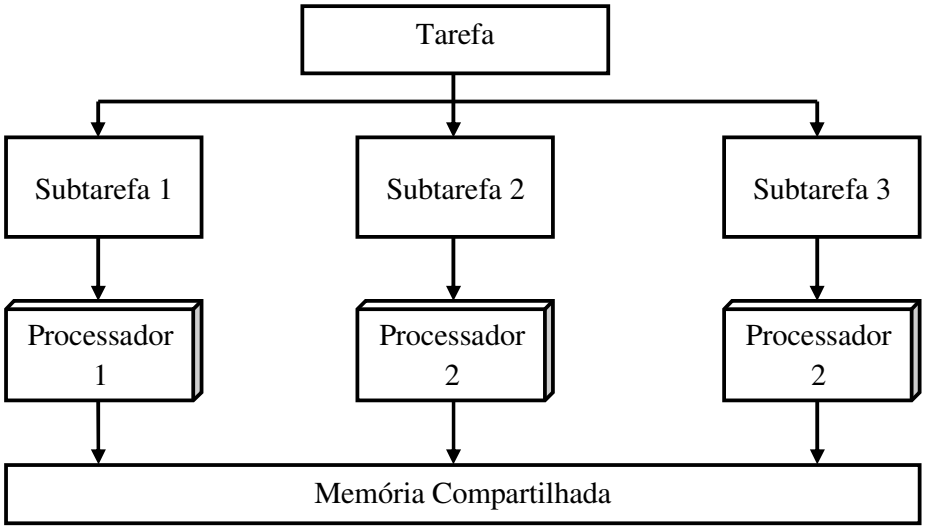


Figura 19 – Diagrama de *pipeline* de processador

A estrutura apresentada anteriormente será aplicada neste projeto para divisão das tarefas entre dois FPGAs.

3.4.2. Arquiteturas Superescalares

A técnica de paralelismo de arquiteturas superescalares é uma evolução da técnica *pipeline*, e procura solucionar a perda de tempo devido a ocorrência de dependências entre as instruções que não eram possíveis de ser solucionadas na técnica *pipeline*. O início desta evolução se deu com o desenvolvimento das memórias cache, que possibilitavam uma redução significativa no tempo de busca de instrução, decodificação e busca de operador, antes realizadas por memórias externas e mais lentas.

Na década de 80 as arquiteturas RISC (*Reduced Instruction Set Computer*) permitiram que os pipelines reduzissem a perda por estágios de execução com tempos de vários ciclos, como ocorria com as arquiteturas CISC (*Complex Instruction Set Computer*). Por fim, o surgimento dos processadores Superescalares, com o IBM801 e o RISC de Berkeley, permitiram a execução de mais de uma instrução de cada vez, buscando e decodificando várias instruções por ciclo. Estas instruções são analisadas e executadas de forma a paralelizar duas instruções que possuam dependência entre elas. Apesar de todas as divisões nas execuções a ordem final deve ser a mesma designada pelo programador, mantendo assim a semântica original.

A arquitetura de um sistema baseado em FPGA permite a criação de um processador superescalar, como já desenvolvido em trabalhos científicos (CASTILHO CARDIM, 2006; CARRILLO e CHOW, 2001; CASILLO et al., 2005), mas requer mais recursos de *hardware* para implementar buscas de instruções independentes e sua execução em paralelo. O compensador seletivo de harmônicas, proposto neste trabalho, apresenta poucas instruções aritméticas dependentes entre si, facilitando a paralelização aritmética.

3.4.3. Processamento Vetorial

O processamento vetorial é utilizado em supercomputadores capazes de resolver problemas matemáticos relativos a processos reais, tipicamente caracterizados pela necessidade de alta precisão numérica com operações de ponto flutuante em grandes vetores de números (STALLINGS, 2002). Os supercomputadores tem preços elevados, existem

poucas unidades no mercado, geralmente em centros de pesquisa e agências governamentais de desenvolvimento científico.

No processamento vetorial as instruções aplicadas aos elementos do vetor são executadas em grupos sequenciais, onde parte das instruções podem ser paralelizadas, o que já reduz o tempo total de processamento. Em um somatório de multiplicações, como o desenvolvido neste trabalho, o número de multiplicações vetoriais ainda é menor que o número de multiplicações escalares (STALLINGS, 2002).

Outra abordagem é a utilização de múltiplos processadores que executam tarefas independentes e podem processar as multiplicações vetoriais de forma paralela, aqui denominada *processamento paralelo*. Uma organização que pode ser utilizada para os processadores é a distribuição paralela das unidades lógicas aritméticas (ULA), Figura 20. No caso de implementação de um filtro digital, que geralmente requer cálculos vetoriais, esta abordagem pode reduzir o tempo de processamento destes cálculos.

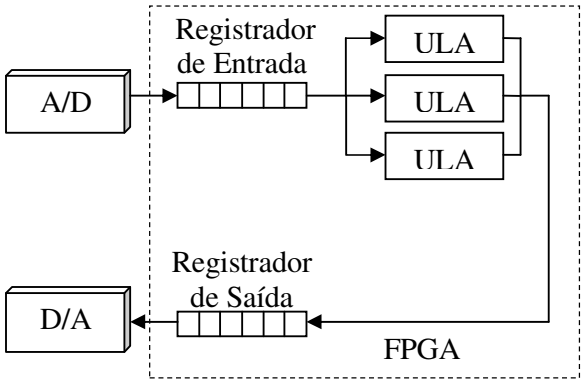


Figura 20 – ULA paralelas

A distribuição paralela das ULAs pode ser feita no mesmo processador ou em múltiplos processadores paralelos, dividindo as tarefas entre os processadores. No projeto do compensador seletivo apresentado na Seção 4.6.2 foram implementadas várias ULAs para atender as necessidades do compensador, estas ULAs foram dispostas em paralelo em cada FPGA, e ainda dois FPGAs também foram paralelizados, praticamente dobrando o número de operações por ciclo de *clock*.

3.5. PARALELISMO EM DISPOSITIVOS FPGA

A execução das operações aritméticas no FPGA pode utilizar o paralelismo que só depende do número de CLB e de entradas livres. Enquanto o processamento sequencial

reutiliza o *hardware* e faz cada operação em novos ciclos de *clock*, o paralelismo exige um número maior de componentes de *hardware*, mas pode executar várias operações simultâneas gastando apenas um ciclo de processamento. A Figura 21 representa a execução de um MAC na forma paralela e sequencial da expressão abaixo (3.2).

$$S = AB + CD + EF + GH \tag{3.2}$$

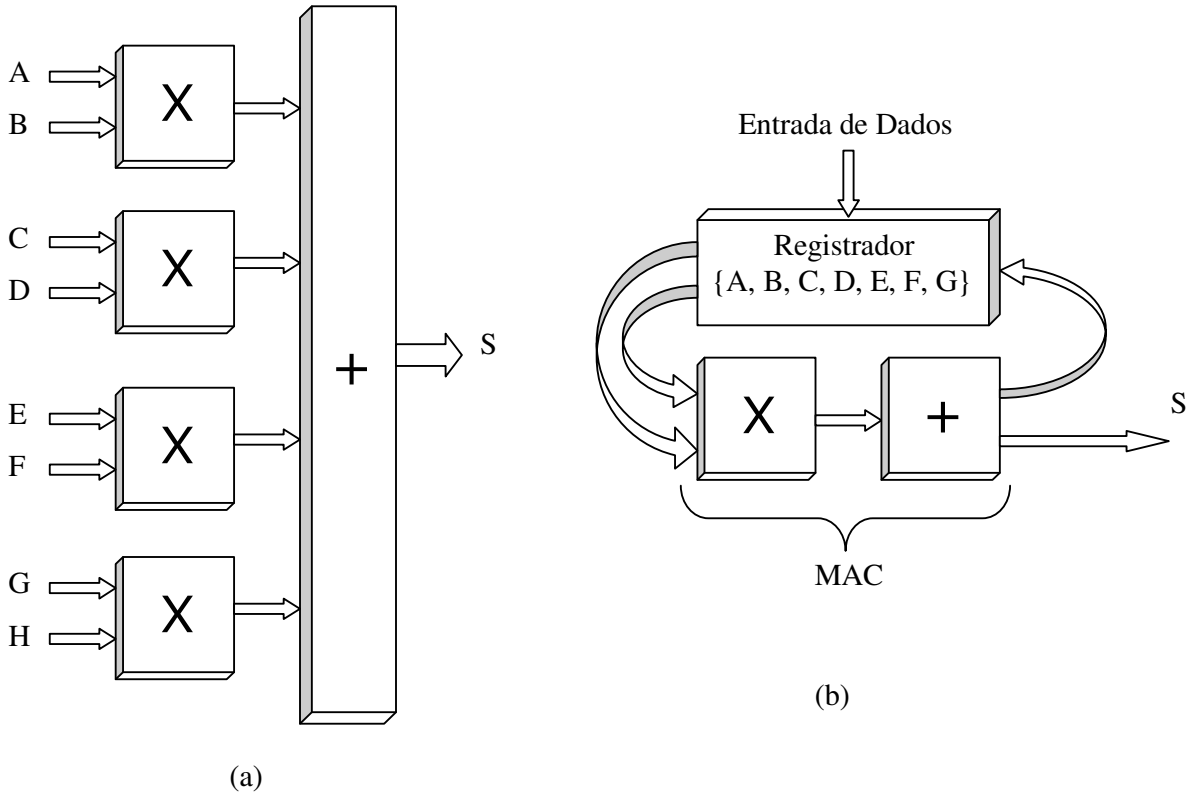


Figura 21 – Operação de multiplicação e adição com palavras de 8 bits: (a) Paralela (b) Sequencial

A execução em paralelo permite a multiplicação simultânea das entradas, e uma única operação de adição. A operação de adição também pode ser executada em etapas, o que requer um somador de menor número de *bits*, mas necessita mais ciclos de máquina. A execução sequencial necessita de um tempo maior de processamento, em relação a execução paralela, isto é, a execução sequencial, processando uma mesma expressão que a realizada na execução paralela, necessita de um número maior de ciclos de máquina, consequentemente demanda de mais tempo. Este tempo varia em função de vários elementos, entre eles o número de variáveis e suas dependências, operações aritméticas e capacidades de *hardware*.

A construção de algoritmos complexos em linguagem de reconfiguração de *hardware*, Verilog ou VHDL, pode ser muito trabalhosa. Quando o algoritmo requer a

implementação de técnicas de paralelismo a dificuldade de implementação torna-se maior ainda ou em alguns casos impraticável. Para facilitar o desenvolvimento do código, alguns trabalhos científicos apresentam possíveis soluções, como métodos de exploração de técnicas *pipeline* em FPGA (SUN, WIRTHLIN e NEUENDORFFER, 2007), programação assíncrona com matrizes *pipeline* (TEIFEL e MANOHAR, 2003) e ferramentas de integração entre a ferramenta MabLab® e a geração de código HDL para FGPA (BANERJEE, 2003; HALDAR et al., 2001).

Para utilização de técnicas de programação paralela que dependem de vários processadores é possível criar um processador utilizando células, multiplicadores e portas de entrada e saída do FPGA. Para isto uma solução já é oferecida pela Xilinx® gratuitamente, mediante apenas a um registro, um microcontrolador RISC de 8 bits, o PicoBlaze.

O microcontrolador PicoBlaze possui os recursos básicos de um microcontrolador (Figura 22), como unidade lógica aritmética (ULA), registradores, memória de instruções programável, memória RAM, interrupções, portas de entrada e saída e velocidade de até 200 MHz, está última dependente do modelo do FPGA.

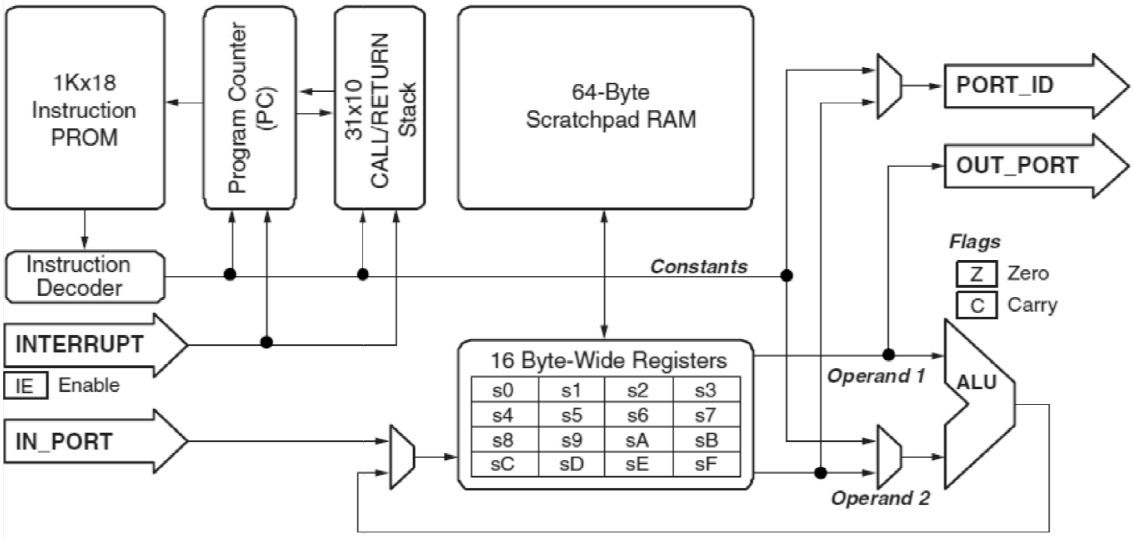


Figura 22 – Diagrama em blocos do microcontrolador PicoBlaze (XILINX PICOBLAZE USER GUIDE, 2004)

Uma importante característica do PicoBlaze é sua otimização, cada microcontrolador utiliza 96 *slices,* fatias de circuito combinacional, em um FPGA XC3S500E isto representa apenas 0,3% do tamanho do FPGA (XILINX PICOBLAZE USER GUIDE, 2004).

O microcontrolador PicoBlaze possui suporte para as famílias de FPGA Xilinx Spartan 3 e 6 e Virtex II, 4, 5 e 6. Na família Spartan 3 o código do microcontrolador que possui o nome genérico de KCPSM (*K'constant Coded Programmable State Machine*) é

acrescido do algarismo “3”, sendo chamado de KCPSM3. A Figura 23 mostra a interface de conexões do microcontrolador PicoBlaze.

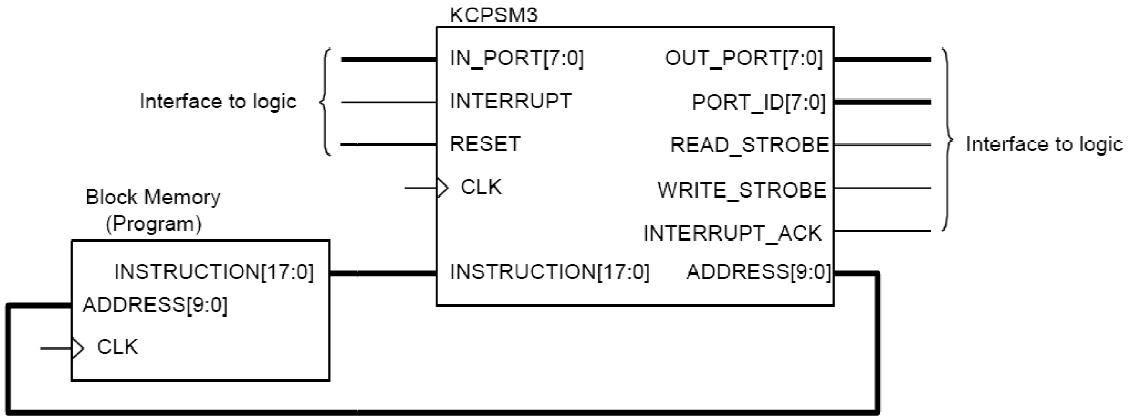


Figura 23 – Conexões do microcontrolador PicoBlaze (XILINX PICOBLAZE USER GUIDE, 2004)

Com o tamanho reduzido é possível implementar vários microcontroladores em apenas um FPGA e configurá-los para tarefas independentes, sem perder a disponibilidade de configurar os recursos não utilizados do FPGA para outras tarefas que requeiram maior velocidade. A Figura 24 ilustra a possível utilização de dois microcontroladores em paralelo acessando uma memória RAM compartilhada.

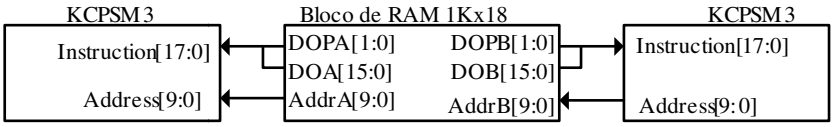


Figura 24 – Acesso compartilhado de dois microcontroladores a mesma memórias RAM

Com a construção de vários processadores é possível utilizar técnicas de *pipeline* de acesso a memória compartilhada, em nível de instrução ou mesmo a *pipeline* de processador, variando a técnica em função das dependências existentes nas rotinas necessárias para os cálculos de controle.

3.5.1. Ferramentas para desenvolvimento de aplicações utilizando paralelismo

O desenvolvimento de código de descrição de *hardware*, por sua característica diferenciada, pode ser muito penoso, principalmente na aplicação de rotinas mais complexas ou na implementação de técnicas de processamento paralelo.

Duas formas de criação de código HDL utilizando *softwares* baseados em linguagem C e que fazem conversão de C para VHDL podem ser vistos em (INOGUCHI, 2004). Existem

compiladores comerciais (eXCite) e projetos gratuitos como o *Omni OpenMP compiler* (Omni). *Softwares* como o *OpemMP* (Omni) possuem diretivas com suporte ao paralelismo e podem facilitar e agilizar a descrição de algoritmos complexos. A Figura 25 exibe uma sequencia de três operações aritméticas escritas em série, Figura 25(a), e em paralelo, Figura 25(b), onde é possível ver que a implementação paralela apenas utiliza a diretiva *par* aplicada a três processos independentes entre si. Havendo dependência entre os processos os mesmos não podem ser executados de forma totalmente paralela (Figura 25(a)).

```
for (i=0; i<N; i++)
{
   a[i] = b[i] + 1;
   x[i] = y[i] * c;
   z[i] = x[i] / a[i];
}
```

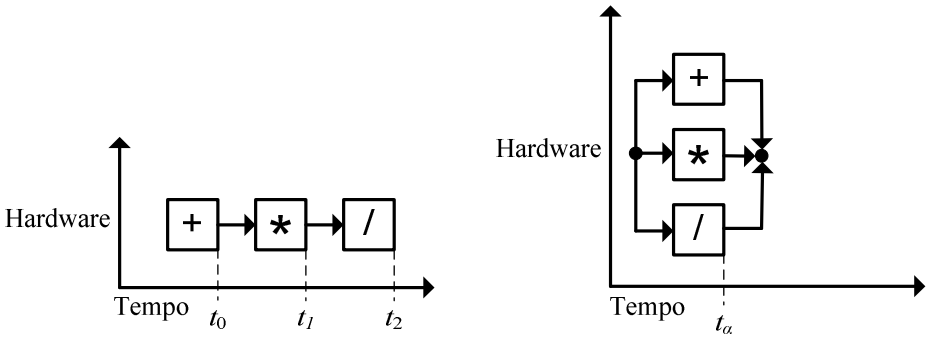
(a)

```
for (i=0; i<N; i++)
{
  par{
   a[i] = b[i] + 1;
   x[i] = y[i] * c;
   z[i] = y[i] / p;
  }
}
```

(b)

Figura 25 – Exemplo de código C: (a) código serial (b) código paralelo

A relação tempo por recursos de *hardware* pode ser exemplificada na Figura 26 que ilustra a necessidade de mais componentes de *hardware* para a execução de uma mesma tarefa quando executada em paralelo (Figura 26(b)). Enquanto a execução serial (Figura 26(a)) requer uma quantidade menor de componentes de *hardware*, pois podem ser reutilizados, mas em contrapartida necessita de um tempo de processamento maior.



(a) (b)

Figura 26 – Relação Tempo x *Hardware*: (a) Processamento Sequencial (b) Processamento Paralelo

A Figura 26 mostra que $t_\alpha < t_2$, onde t_α representa o término do conjunto de instruções paralelas e o tempo t_2 o término do conjunto de instruções seriais.

3.6. SUMÁRIO

A implementação de técnicas de processamento paralelo em FPGA aparece naturalmente pela linguagem de descrição de *hardware* VHDL, pois a reconfiguração de *hardware* mantém todos os componentes em um mesmo nível de tempo de processamento, a menos que sejam reconfigurados para instantes diferentes de tempo. A complicação no desenvolvimento é a separação dos conjuntos de instruções independentes em instantes de tempo únicos. Instruções dependentes precisam de resultados prévios, o que implica em uma estrutura serial de disposição das instruções.

As ferramentas de desenvolvimento de código HDL mostraram produtividade, mas não permitiram um total domínio do *hardware*, sendo descartadas neste trabalho.

O microcontrolador PicoBlaze permitie um paralelismo de ULA no mesmo processador e uma maior facilidade de desenvolvimento, mas consume recursos de *hardware* acima do disponível no dispositivo XC3S500E, utilizado neste trabalho. Além disso, o controle de registradores limitado a 8 bits requer processos extras para alcançar a precisão necessária.

Várias técnicas e ferramentas foram discutidas neste capítulo, mostrando os caminhos possíveis para implementação da solução proposta neste trabalho. No capítulo seguinte é apresentado, além de resultados experimentais, a aplicação de dois filtros sendo executados em paralelo utilizando técnicas de *pipeline* aritmético e a implementação do compensador seletivo de harmônicas utilizando processamento vetorial com paralelismo de ULA e processadores.

4. RESULTADOS EXPERIMENTAIS

4.1. INTRODUÇÃO

O objetivo da análise experimental que segue é verificar a abordagem proposta em condições reais, os recursos necessários, tanto de *hardware* como de *software*, para implementação em FPGA de métodos de controle para FAPs, discutidos neste trabalho. Desta forma, diversos testes foram realizados e resultados são apresentados, desde a implementação de multiplicadores a partir de circuitos combinacionais, filtros digitais de ordem reduzida, até um filtro seletivo de harmônicas de alta ordem (N=100).

No primeiro estágio de avaliação é apresentada a análise dos tempos de comunicação nos barramentos de dados e os tempos de processamento de operações de multiplicação em série e em paralelo, operações básicas para a implementação de filtros digitais. Os testes foram executados em dispositivos FPGA com interface com um DSP, o qual é utilizado para geração de dados de entrada e captura de dados de saída. Na sequência, é apresentada a análise de dois filtros digitais passa-baixas de segunda ordem, comparando os tempos de resposta de processamento da execução de apenas um filtro e posteriormente a execução dos dois filtros em paralelo.

A validação experimental da implementação de estruturas de controle em hardware paralelo é realizada considerando a abordagem de controle proposta por Mattavelli e Fasolo (2000) e discutida no Capítulo 2. Foram implementados dois compensadores seletivos de harmônicas com 50 e com 100 pontos, sendo o de 50 pontos executado em um único FPGA e o de 100 pontos em dois FPGA paralelos. A verificação dos resultados numéricos é realizada comparando a implementação em FPGA com resultados de simulação a partir da mesma abordagem em software Matlab®. Os resultados relativos a tempo de execução do algoritmo são comparados aos apresentados por Mattavelli e Fasolo (2000) em uma implementação sequencial com DSP.

4.2. DESCRIÇÃO DA PLATAFORMA

Considerando a proposta de implementação de filtros e controladores digitais em FPGAs, foi utilizado um dispositivo XC3S500E com *clock* de 50 MHz, do fabricante Xilinx®, para implementação das estruturas analisadas. Este dispositivo possui 10.476 células

lógicas, 20 multiplicadores dedicados embarcados e 232 pinos de entrada e saída. O dispositivo em questão é montado sobre uma plataforma de desenvolvimento Spartan-3E Starter Kit (Figura 27), que possui conversor A/D (Analógico Digital), com precisão de 14 bits e 1,7 μs de tempo de conversão e comunicação com o FPGA através de interface serial para periféricos (SPI - *Serial Peripheral Interface*). Além disso, possui conversor D/A (Digital Analógico) com 12 bits e tempo de conversão de aproximadamente 640 ns, que recebe dados do FPGA através de SPI. Para comunicação SPI são utilizados módulos de circuitos lógicos roteados no próprio FPGA-XC3S500E.

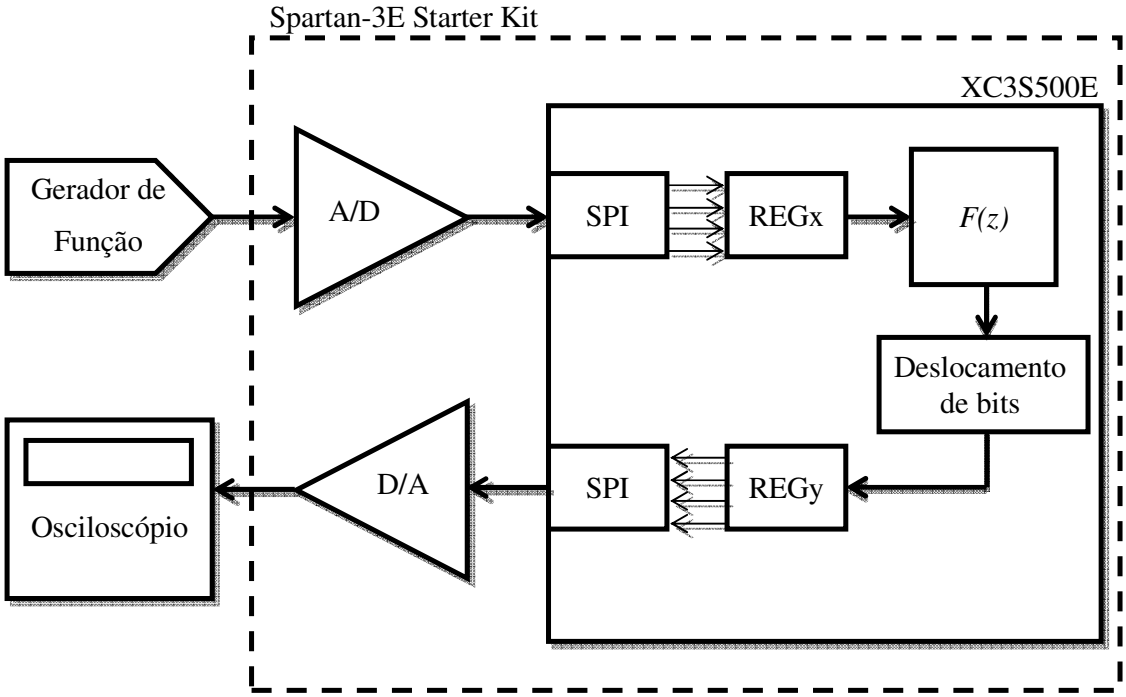


Figura 27 – Diagrama de blocos do *setup*

O sistema de teste consiste na leitura de um sinal com harmônicas, gerados através de um gerador de funções, utilizando o conversor A/D do Kit. Este faz a conversão analógico-digital e envia os dados serialmente por protocolo SPI para o FGPA. Um módulo de *hardware* decodifica e armazena os dados em registradores de entrada. A função de controle, simbolizada como *F(z)*, utiliza os dados do registrador de entrada para realizar as operações aritméticas do filtro. Após os cálculos o resultado é deslocado, permanecendo válidos apenas os 14 bits mais significativos e estes são armazenados nos registradores de saída. Essa saída é enviada para o conversor D/A também por protocolo SPI. A leitura do sinal de saída do conversor D/A é realizada através de um osciloscópio DPO4034 (350 MHz, 2,5 GS/s, com uma ponteira P6139A (500 MHz)).

4.3. DETERMINAÇÃO DA VELOCIDADE DO BARRAMENTO

Para analisar o desempenho do FPGA no Spartan-3E Starter Kit e da aplicação de filtros digitais em VHDL, diversos experimentos foram realizados: como primeiro, um teste de velocidade do barramento de dados; posteriormente, um somador de 4 bits sem multiplicador embarcado comparado com a execução de 3 multiplicadores embarcados de 18 bits, em série e em paralelo; na sequência, desenvolveu-se um filtro digital de segunda ordem.

A velocidade de comunicação do barramento de dados do FPGA, XC3S500E, foi considerada como o tempo necessário para interpretar um sinal lógico alto em um pino de entrada, o acionamento de uma saída e a captura deste sinal no conector externo. Este tempo foi medido em osciloscópio digital DPO4034, ANEXO B – PLATAFORMA DE ENSAIOS, sendo de 8,6 ns, conforme apresentado na Figura 28. O XC3S500E interpreta um nível lógico alto a partir de 0,4 V. Na Figura 28 é possível ver os dois cursores (‘a’ e ‘b’), sendo que o tempo encontrado é a diferença entre os pontos marcados pelos dois cursores. Este procedimento é utilizado para indicações de tempo nas demais medições deste trabalho.

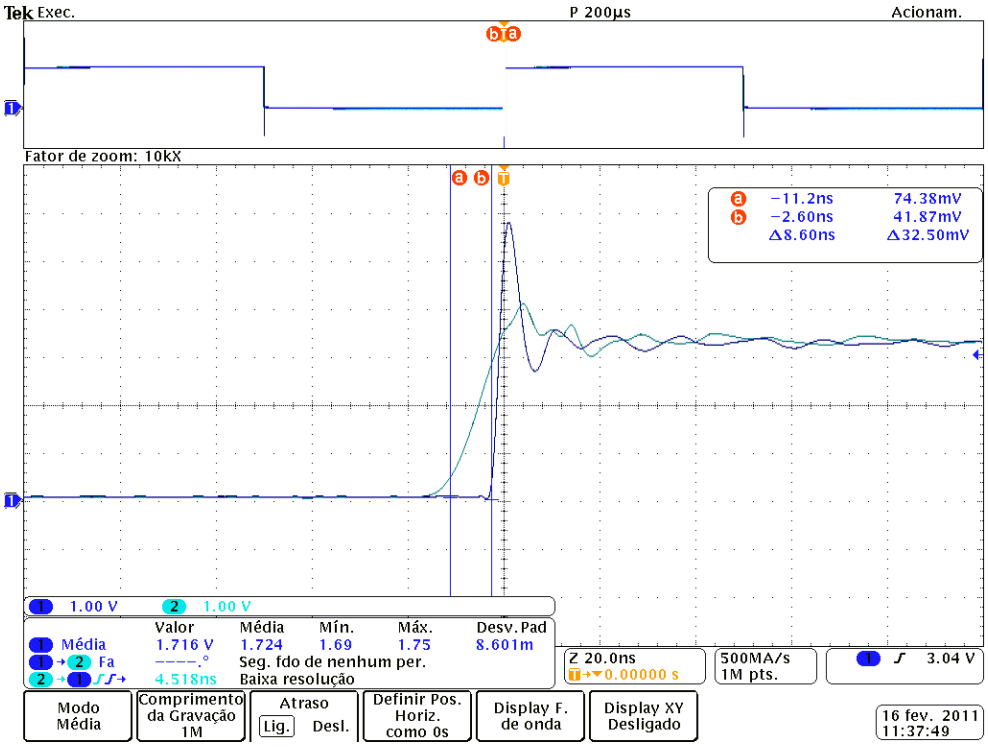


Figura 28 – Velocidade de comunicação do barramento de entrada e saída do XC3S500E.

4.4. DETERMINAÇÃO DA VELOCIDADE DAS ESTRUTURAS LÓGICAS E ARITMÉTICAS

Uma vez que a operação de multiplicação é uma das mais essenciais e críticas para rotinas de controle, em termos de tempo de processamento, foi desenvolvido um multiplicador binário de 4 bits, a partir de lógica combinacional, com o objetivo de analisar os tempos de execução de circuitos deste tipo na implementação de algoritmos de controle. Foi realizada uma medição da velocidade de processamento de uma multiplicação binária com entradas de 4 bits. O circuito apresentou tempo de resposta de 54,2 ns, conforme resultado das formas de onda da Figura 29. Este é o tempo de processamento matemático ignorando o tempo do barramento.

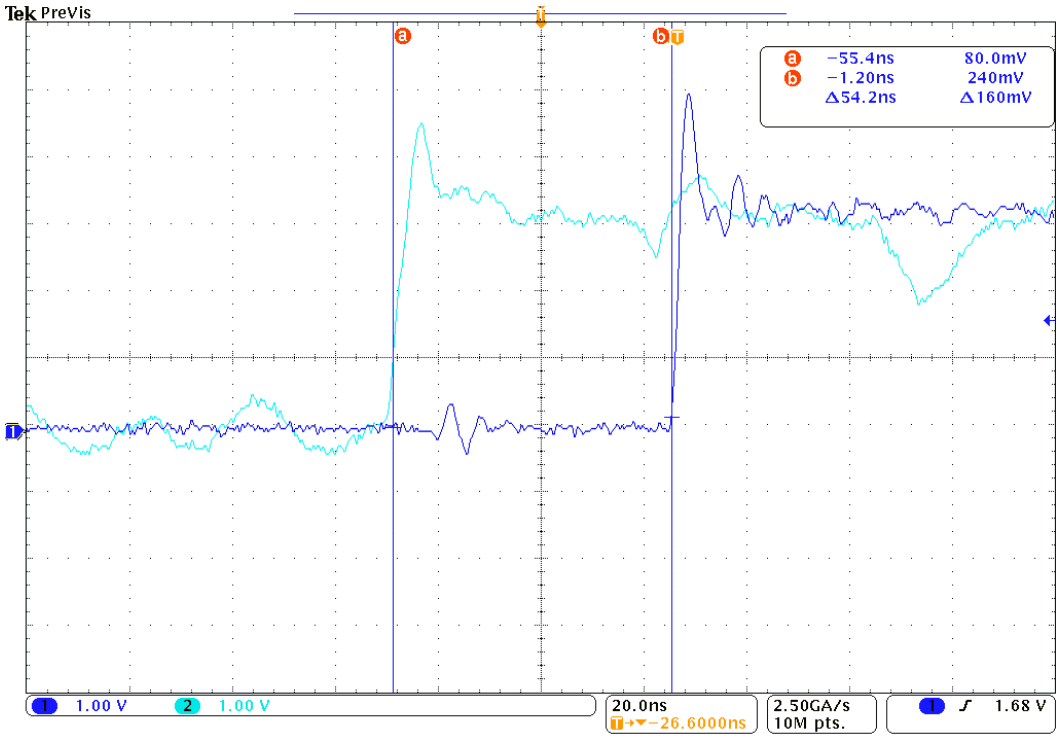


Figura 29 – Tempo de processamento de um multiplicador combinacional de 4 bits no XC3S500E.

Para comparação do circuito multiplicador combinacional desenvolvido com multiplicadores dedicados de 18 bits embarcados no XC3S500E, foram implementadas execuções paralelas e seriais de 3 multiplicadores. As multiplicações em paralelo com os multiplicadores embarcados necessitaram de um tempo de aproximadamente 7,8 ns para serem concluídas, vide Figura 30, enquanto que a execução serial necessitou de 29,2 ns, conforme Figura 31. Na verificação de tais tempos os coeficientes da multiplicação são gerados por um DSP externo, processados pelo FPGA e a saída é apresentada em uma porta

de IO. Um bit de saída é levado a nível 1 quando o coeficiente é enviado pelo DSP e outro é levado a nível 1 quando o resultado se altera no barramento de saída do FPGA. A diferença entre os dois, apresentada pela diferença entre os cursos ‘a’ e ‘b’ (Figura 30), é o tempo necessário para comunicação e processamento do multiplicador.

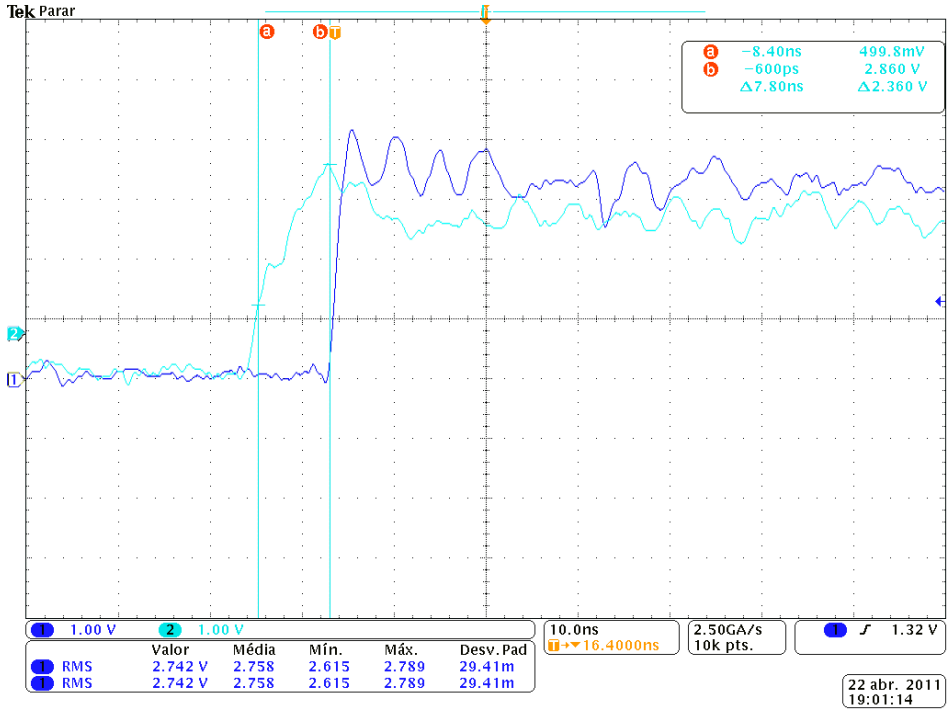


Figura 30 – Tempo de execução de 3 multiplicadores dedicados executados em paralelo no FPGA XC3S500E.

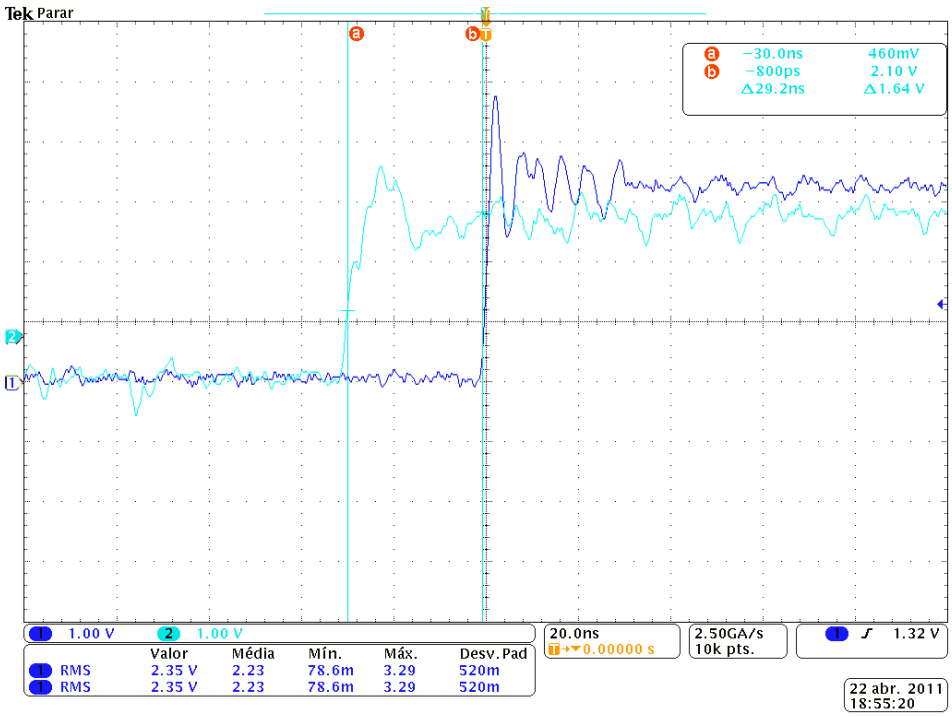


Figura 31 – Tempo de execução de 3 multiplicadores dedicados executados em série no FPGA XC3S500E.

Comparando-se os tempos de processamento do multiplicador combinacional (4 bits) com o dos multiplicadores embarcados (18bits) verificou-se que o tempo de processamento do primeiro é relativamente elevado. Desta forma, é conveniente desenvolver rotinas de controle que utilizem os multiplicadores embarcados, evitando-se o uso de lógica combinacional para realização de tais operações.

4.5. IMPLEMENTAÇÃO DE FILTROS DIGITAIS EM FPGA

Essencialmente, as rotinas de controle executam operações relacionadas a lógica, ganho e filtragem de sinais. Desta forma, filtros digitais estão entre operações mais típicas a serem executadas por um controlador digital. Para comprovar a possibilidade de implementação em FPGA e analisar a velocidade de resposta de filtros digitais com operações paralelas foram projetados e implementados dois filtros passa-baixas de segunda ordem, um com frequência de corte (f_c) em 600 Hz e outro em 1,2 kHz.

4.5.1. Projeto do Filtro

A função de transferência de um filtro passa-baixas de segunda ordem pode ser dada por

$$F(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \tag{4.1}$$

sendo ω_n a frequência de corte e ξ o fator de amortecimento.

Os parâmetros utilizados na simulação e implementação dos filtros estão na Tabela 1.

Tabela 1 – Parâmetros dos filtros

Parâmetros	**f_c = 600 Hz**	**f_c = 1,2 kHz**
Frequência de Corte em rad/s (ω_n)	3769,911	7539,822
Coeficiente de amortecimento (ξ)	1,0	1,0
Frequência de Amostragem (f_s)	24 kHz	24 kHz

A forma discreta, obtida utilizando-se a transformada Z, pode ser dada por

$$\frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}} \tag{4.2}$$

De (4.2) pode-se obter a forma recursiva de tempo discreto para a saída do filtro:

$$y[k] = b_1 x[k-1] + b_2 x[k-2] - a_1 y[k-1] - a_2 y[k-2] \tag{4.3}$$

Os coeficientes calculados para frequência de corte de 600 e 1200 Hz podem ser vistos nas Tabela 2 e Tabela 3, respectivamente. A precisão dos coeficientes foi limitada a quatro casas decimais para adequar os cálculos aos multiplicadores por *hardware* de 18 bits embarcados no FPGA XC500S3E. Em uma aplicação que requeira mais precisão os coeficientes podem utilizar multiplicadores em cascata, mas como objetivo deste teste é medir os tempos de resposta dos filtros em paralelo, a estrutura mais simples apresenta com mais clareza os tempos de execução.

Tabela 2 – Coeficientes do filtro (4.2) com f_c = 600 Hz.

Coeficientes	
a	***b***
$a_0 = 1$	$b_0 = 0$
$a_1 = -1,7093$	$b_1 = 0,0111$
$a_2 = 0,7304$	$b_2 = 0,0100$

Tabela 3 – Coeficientes do filtro (4.2) com f_c = 1,2 kHz.

Coeficientes	
a	***b***
$a_0 = 1$	$b_0 = 0$
$a_1 = -1,4608$	$b_1 = 0,0401$
$a_2 = 0,5335$	$b_2 = 0,0325$

A partir de (4.3) pode-se verificar que as multiplicações das amostras pelos coeficientes podem ser executadas paralelamente, todas em um único ciclo de *clock*, utilizando quatro multiplicadores dedicados para cada filtro, a não ser no transitório de partida. Para os dois filtros são necessárias oito multiplicações, que são armazenadas em variáveis isoladas e somadas em um segundo ciclo de *clock.* Em um terceiro e quarto ciclos são atualizadas as variáveis de atrasos do sinal de entrada e saída. Com a finalidade de verificar o resultado com sinais reais, o processo é complementado pelas conversões de bases entre valor de entrada lido através do conversor A/D para a entrada do filtro e a saída do filtro para o conversor D/A.

Como o FPGA XC3S500E possui 20 multiplicadores dedicados, é possível implementar até 5 filtros iguais ao descrito anteriormente, sem acrescer tempo de processamento, visto que o hardware permite a execução dos 5 ao mesmo tempo. Por não requerer nenhum compartilhamento de *hardware* o tempo de execução dos cinco filtros é igual ao tempo de execução de um filtro isoladamente. A partir do quinto filtro é necessário o

compartilhamento de multiplicadores em instantes de tempo diferentes ou utilização de multiplicadores combinacionais como visto no Capítulo 3, Figura 16. Todos os cálculos foram realizados com variáveis inteiras utilizando deslocamento de bits para representação real dos valores.

Para testar o *hardware* desenvolvido com sinais reais foi considerado um sinal de entrada do filtro com frequência de 60 Hz acrescida de uma harmônica de 4,2 kHz. O primeiro teste consistiu em verificar o tempo de execução de um filtro excluindo os tempos de comunicação SPI necessários para interação com o conversor A/D e o conversor D/A do Spartan-3E Starter kit. O segundo teste consistiu em verificar a execução de dois filtros em paralelo. Os resultados dos dois testes apresentaram o mesmo valor, 260 ns, equivalente a 13 ciclos de *clock* em 50 MHz, velocidade do oscilador interno do Kit. As operações realizadas nestes ciclos de clock foram as multiplicações, as somas, leituras e escritas nos registradores e deslocamentos de bits. Os procedimentos são semelhantes aos implementados no compensador seletivo de harmônicas que será detalhado na Seção 4.6.2. A Figura 32 mostra o tempo execução de um filtro e a Figura 33 o tempo dos dois filtros executados em paralelo. O tempo é medido pelo intervalo entre o final da conversão A/D e o início da conversão D/A.

Os cálculos necessários para aplicação de cada filtro são executados utilizando a técnica de *pipeline* aritmético, onde cada bloco de *hardware* executa separadamente as operações aritméticas. Isoladamente, o segundo filtro utiliza outra estrutura de processamento para executar suas tarefas, formando estruturas separadas de processadores, técnica chamada de *pipeline* de processador. Neste filtro não foram utilizadas as técnicas de processamento vetorial, estas serão implementadas nos próximos ensaios, a fim de reduzir ainda mais o tempo de processamento.

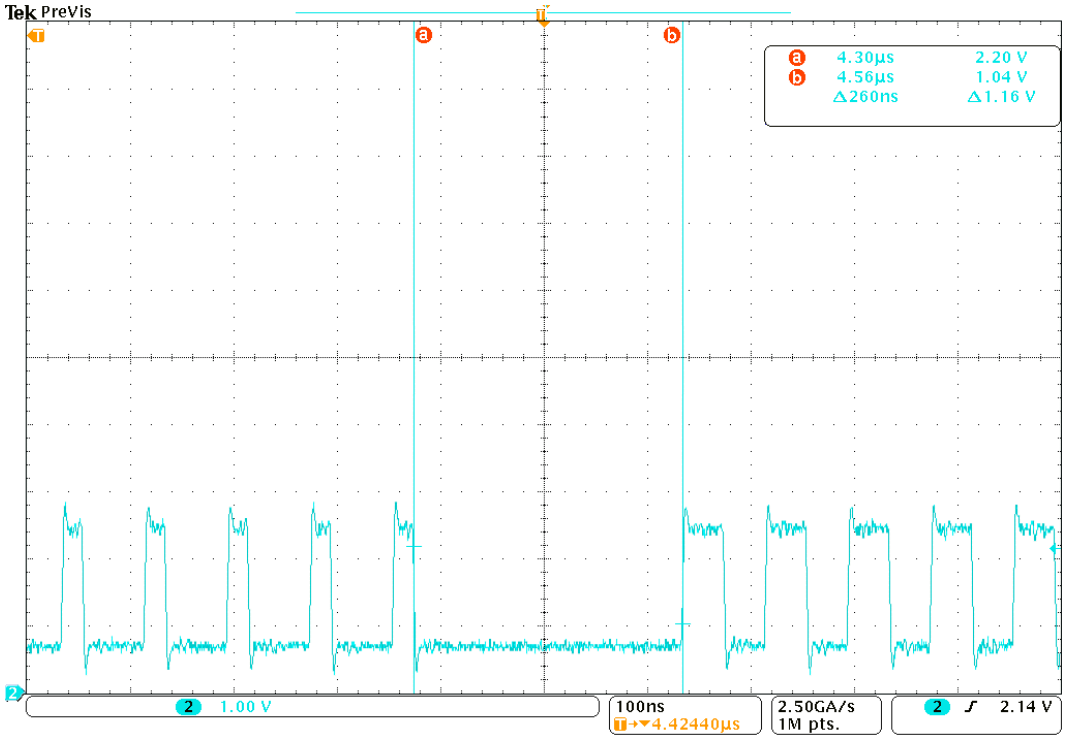


Figura 32 – Tempo de execução de um filtro digital passa-baixas de segunda ordem no XC3S500E.

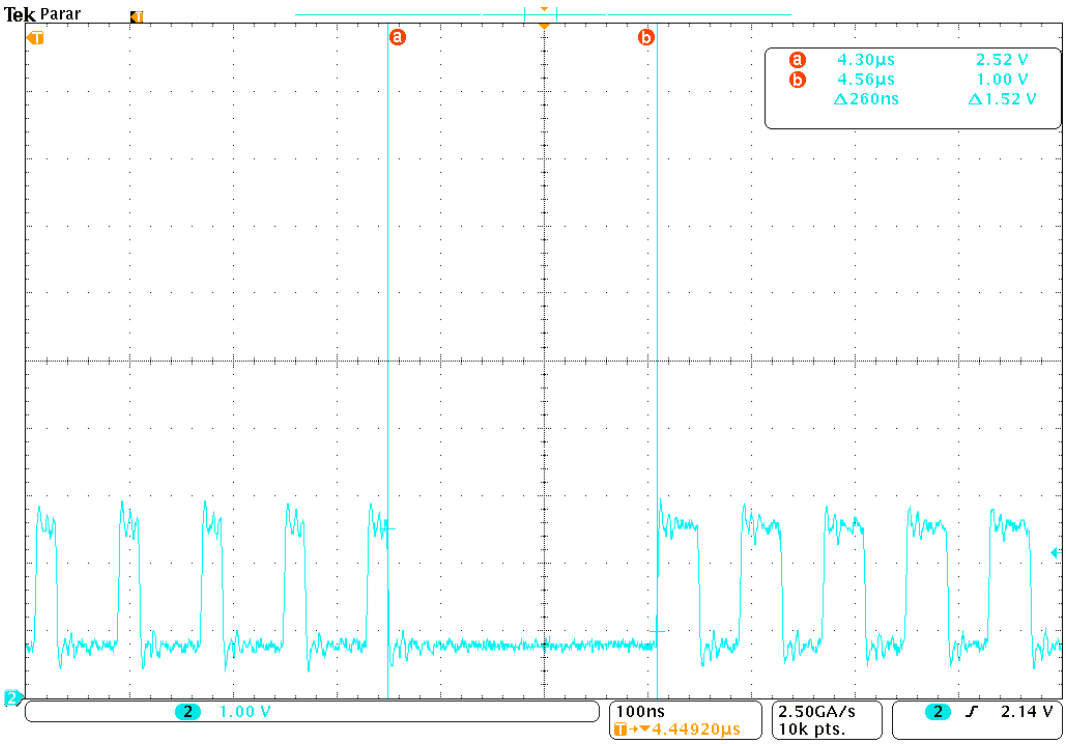


Figura 33 – Tempo de execução de dois filtros digitais passa baixas de segunda ordem, em paralelo, no XC3S500E.

As formas de onda dos sinais aplicados nos filtros de 2ª ordem descritos anteriormente, também como os sinais obtidos na saída do conversor D/A, podem ser visualizados nas figuras que se seguem.

A Figura 34 exibe as formas de onda do sinal de entrada (canal 1), sintetizado pelo gerador de funções AFG-3022 (25 MHz), e dos sinais de saídas dos dois filtros (canais 3-4). A Figura 35 mostra a transformada rápida de Fourier (FFT – *Fast Fourier Transform*) da entrada, onde a frequência de 4,2 kHz apresenta nível -20 dB. A saída filtrada com frequência de corte de 600 Hz apresenta uma atenuação de -32,7 dB. A saída do filtro de frequência de corte de 1.2 kHz apresenta uma atenuação menor, -30,7 dB. Verificou-se portanto que o filtro com frequência de corte de 600 Hz apresentou 2 dBs a mais de atenuação que o filtro de 1,2 kHz de frequência de corte. Este resultado é o esperado, considerando-se a diferença entre as frequências de corte dos filtros e levando em conta o número limitado de casas decimais utilizado nos coeficientes, o que limita a sua precisão.

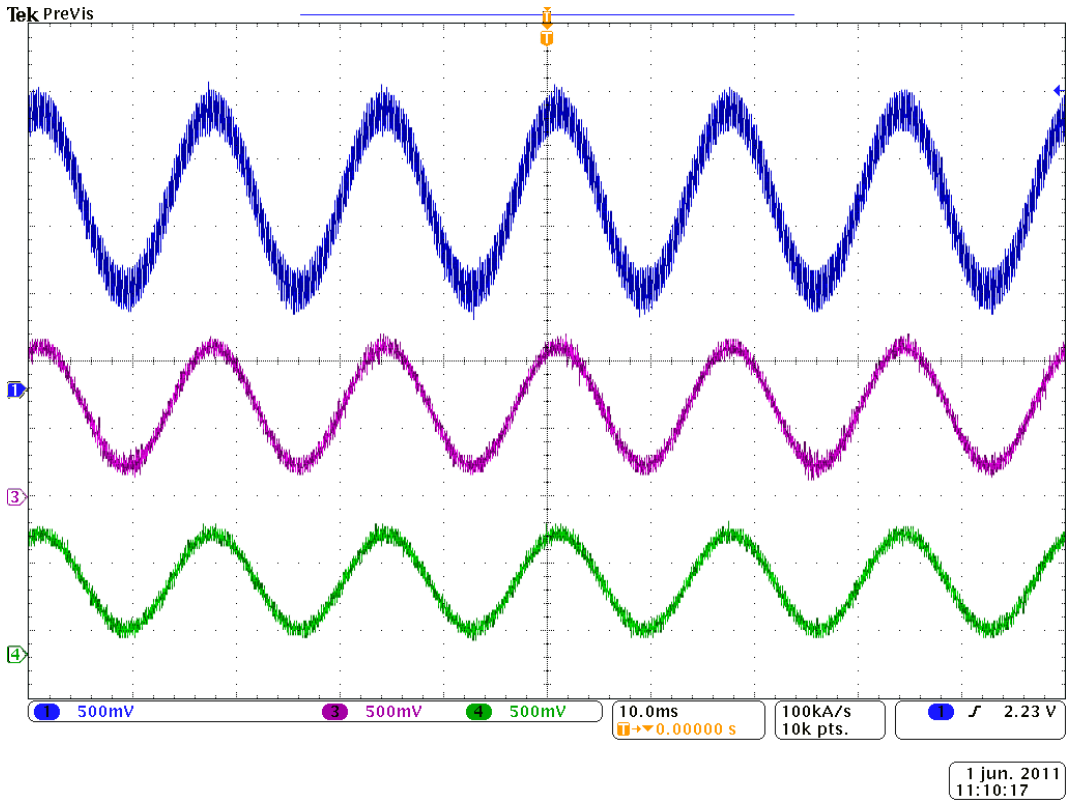


Figura 34 – Sinal de entrada (1), saída do filtro com *fc* = 600 Hz (3) e saída do filtro com *fc* = 1200 Hz (4)

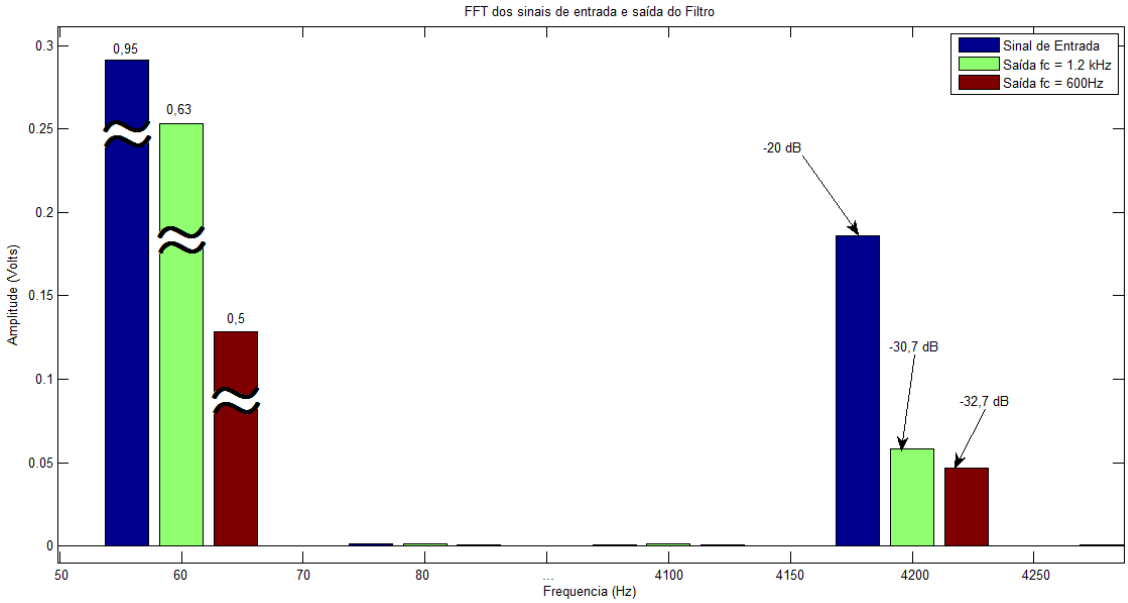


Figura 35 – FFT dos sinais de entrada e saída dos Filtros.

Os filtros foram testados com uma frequência harmônica mais afastada da frequência de corte. Utilizando-se harmônica com frequência de 9 kHz, a resposta, que pode ser vista na Figura 36, apresentou atenuação significativa da referida harmônica, de aproximadamente -50dB.

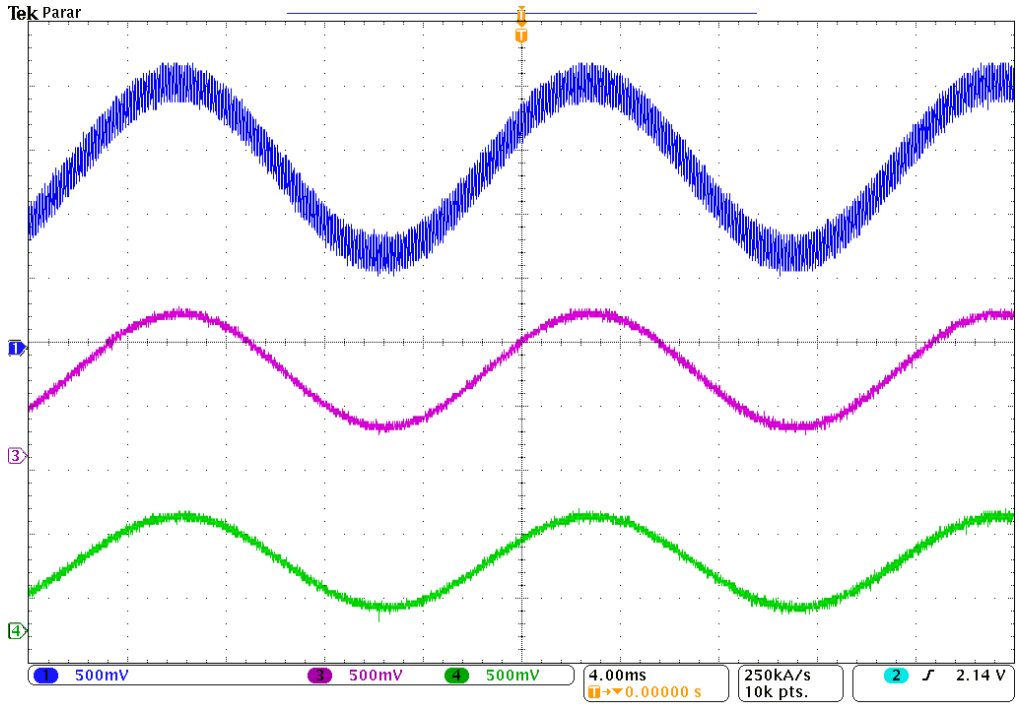


Figura 36 – Resposta dos filtros a uma frequência harmônica de 9 kHz.

4.6. IMPLEMENTAÇÃO DE COMPENSADORES SELETIVOS DE HARMÔNICAS EM FPGA

Para verificar a proposta de implementação de controladores digitais em tempo real utilizando execução paralela em FPGA, foram implementados dois compensadores seletivos de harmônicas, utilizando a técnica proposta por Matavelli e Fasolo (2000). Os compensadores foram projetados com N=50 e N=100 pontos por ciclo, respectivamente e considerando um sinal com frequência fundamental de 60 Hz e com 3ª e 5ª harmônicas. Para os testes em questão foi utilizado um gerador de funções gerando um sinal de 60Hz e adicionadas 3ª e 5ª harmônicas, ambas com 15% da fundamental.

4.6.1. Compensador Seletivo de Harmônicas de Ordem 50

Os 50 pontos por ciclo (N=50) considerando frequência fundamental de 60 Hz equivalem a uma taxa de amostragem de 3000 Hz, ou a um tempo de amostragem de 333,33 *u*s, conforme expressão abaixo:

$$T_s = \frac{1}{f\,N} \tag{4.4}$$

sendo T_s o tempo de amostragem, f é a frequência fundamental e N o número de pontos.

A equação do compensador no domínio do tempo discreto é apresentada em (4.5).

$$y(k) = \Omega_0 x(k) + \Omega_1 x(k-1) + \Omega_2 x(k-2) + \ldots + \Omega_{N-1} x(k-N+1) \tag{4.5}$$

Pela expressão (4.5) percebe-se que serão necessárias N multiplicações. Como o dispositivo FPGA possui apenas 20 multiplicadores dedicados, na implementação do compensador foram considerados três estágios de multiplicação, sendo dois com 20 multiplicações e um estágio com 10 multiplicações. Os coeficientes do compensador são calculados *a priori* pela equação (4.6), utilizando o *software* matemático MATLAB®, e importados pela interface de desenvolvimento (Xilinx IDE - *Integrated Development Environment*) utilizada para programar o FPGA. Os valores determinados podem ser visualizados na tabela do ANEXO A – COEFICIENTES DOS COMPENSADORES SELETIVOS DE HARMÔNICAS.

$$
\begin{aligned}
F_1(z) &= \frac{2}{N}\sum_{i=0}^{N-1}\cos\left(\frac{2\pi}{N}1i\right)z^{-i} = \alpha_0 + \alpha_1 z^{-1} + a_2 z^{-2} + \ldots + \alpha_{N-1} z^{-N+1} \\
F_2(z) &= \frac{2}{N}\sum_{i=0}^{N-1}\cos\left(\frac{2\pi}{N}2i\right)z^{-i} = \beta_0 + \beta_1 z^{-1} + \beta_2 z^{-2} + \ldots + \beta_{N-1} z^{-N+1} \\
&\vdots \\
F_N(z) &= \frac{2}{N}\sum_{i=0}^{N-1}\cos\left(\frac{2\pi}{N}(N-1)i\right)z^{-i} = \zeta_0 + \zeta_1 z^{-1} + \zeta_2 z^{-2} + \ldots + \zeta_{N-1} z^{-N+1}
\end{aligned}
\tag{4.6}
$$

Para determinação do tempo de cálculo, a Figura 37 mostra a medição realizada com o osciloscópio DPO4034 em um pino de saída do FPGA, que é registrado no início do processo e permanece em nível lógico baixo enquanto os cálculos vetoriais do filtro são processados. Este teste é realizado para comparação com o trabalho de Mattavelli e Fasolo (2000) que apresentou um tempo de 3,2 μs por fase (α e β) para o bloco de processamento do controle de harmônicas utilizando um DSP de ponto fixo da Analog Devices (ADMC401), na mesma ordem de velocidade. Neste caso, o tempo do mesmo procedimento executado em paralelo no FPGA XC3S500E da Xilinx foi de 226 ns, conforme Figura 37. O DSP da Analog Devices possui *core* de 26 MIPS custa U$ 46,25, enquanto o FPGA da Xilinx, encapsulamento XC3S500E-5FGG320C, custa U$ 40,23, ambos com cotação do distribuidor DigiKey Corporation.

Em simulações computacionais, utilizando MATLAB®, o filtro seletivo para eliminação das harmônicas apresentou bons resultados. Entretanto, apesar da eliminação ser apenas parcial, o que é explicável considerando-se um reduzido número de pontos de amostragem, é apropriado para comparações de tempo de processamento. A Figura 38 mostra o resultado de uma simulação computacional do compensador seletivo de harmônicas utilizando 50 pontos de amostragem por ciclo da frequência fundamental de 60 Hz. A implementação em FPGA desta configuração apresentou resultados muito semelhantes.

As formas de onda de entrada e saída e as FFT dos sinais podem ser vistos na Figura 39 e na Figura 41, para entrada, e Figura 40 e Figura 42, para a saída.

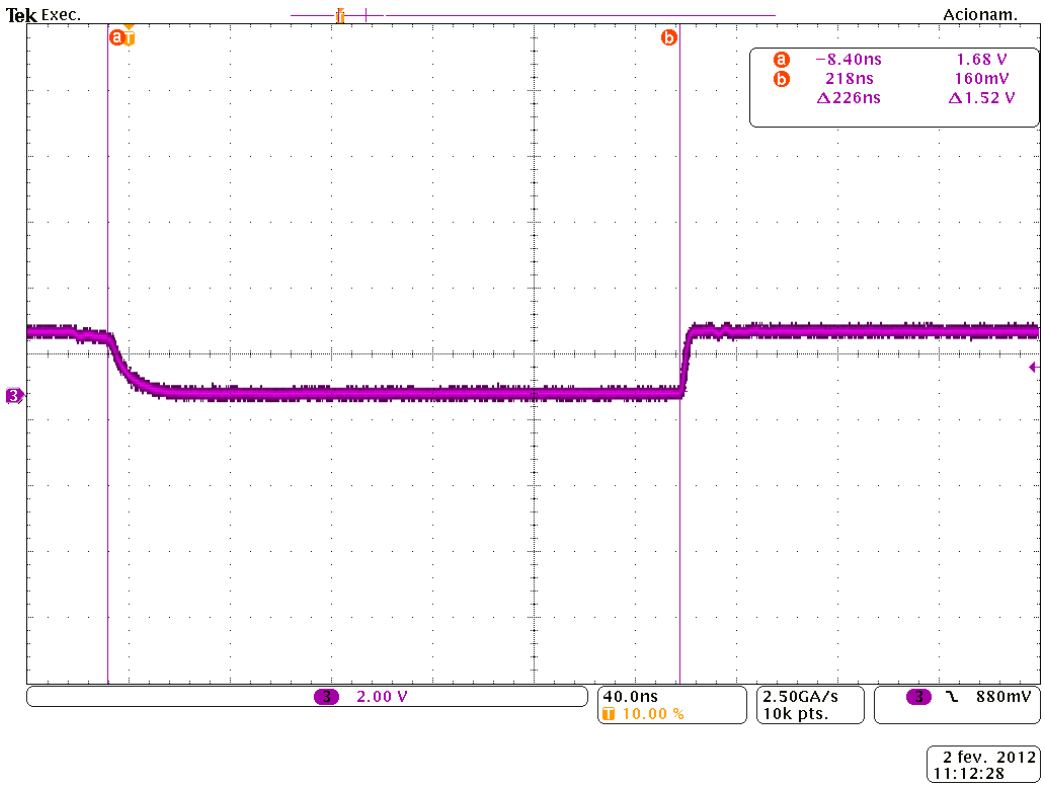


Figura 37 –Tempo de processamento do compensador seletivo de harmônicas com 50 pontos de amostragem

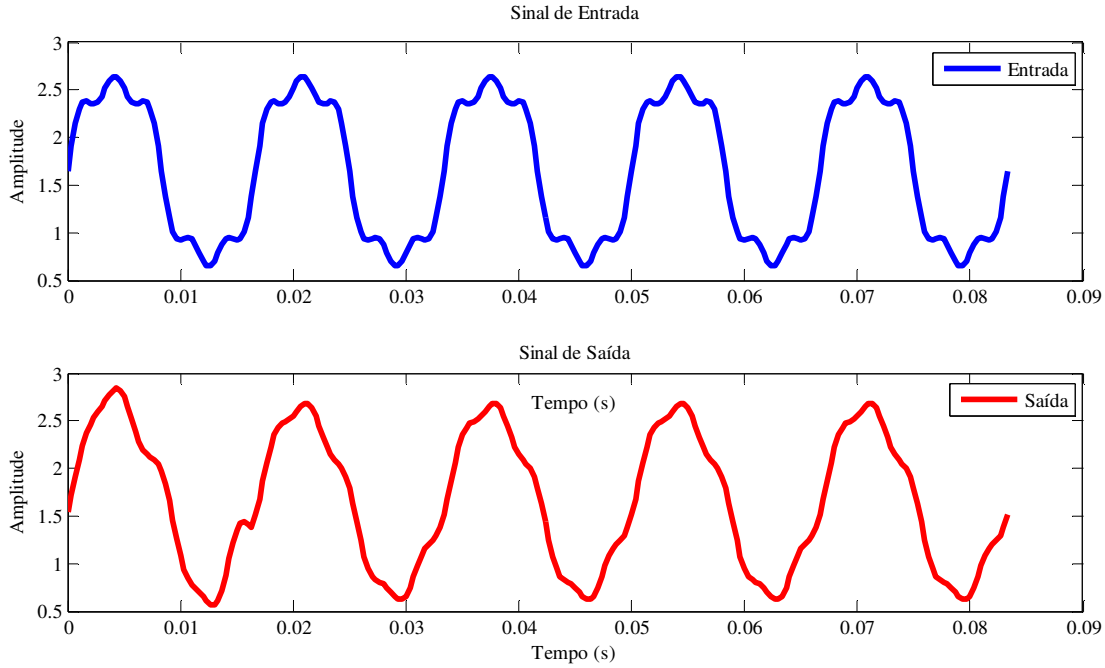


Figura 38 – Simulação computacional do compensador seletivo de harmônicas utilizando 50 pontos de amostragem.

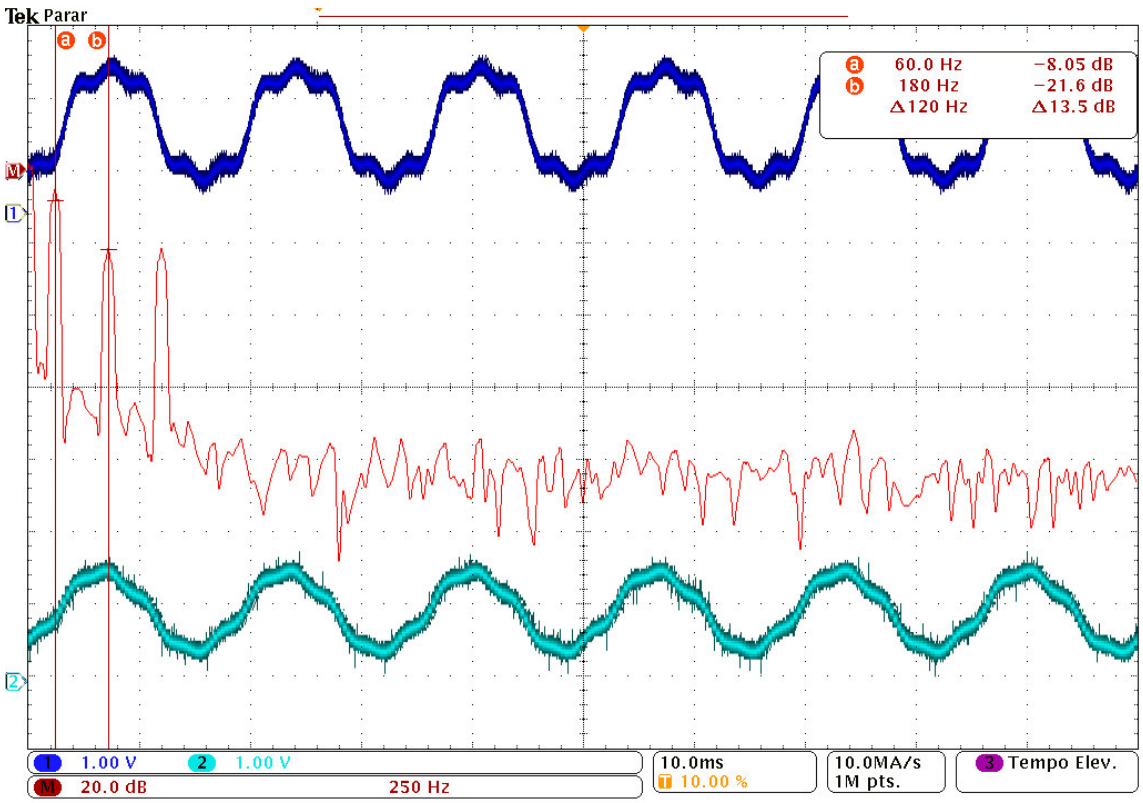


Figura 39 – Formas de onda de entrada (1), saída (2) e FFT (M) do sinal de entrada, 60 e 180 Hz

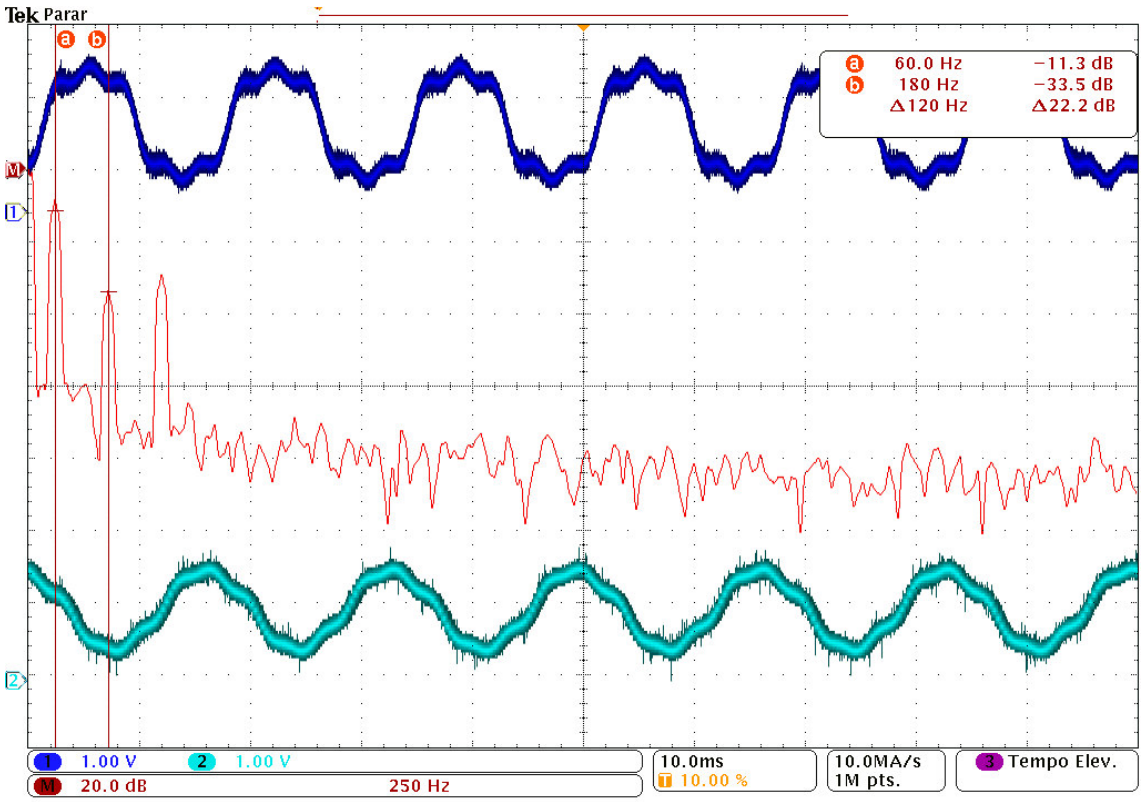


Figura 40 – Formas de onda de entrada (1), saída (2) e FFT (M) do sinal de saída, 60 e 180 Hz

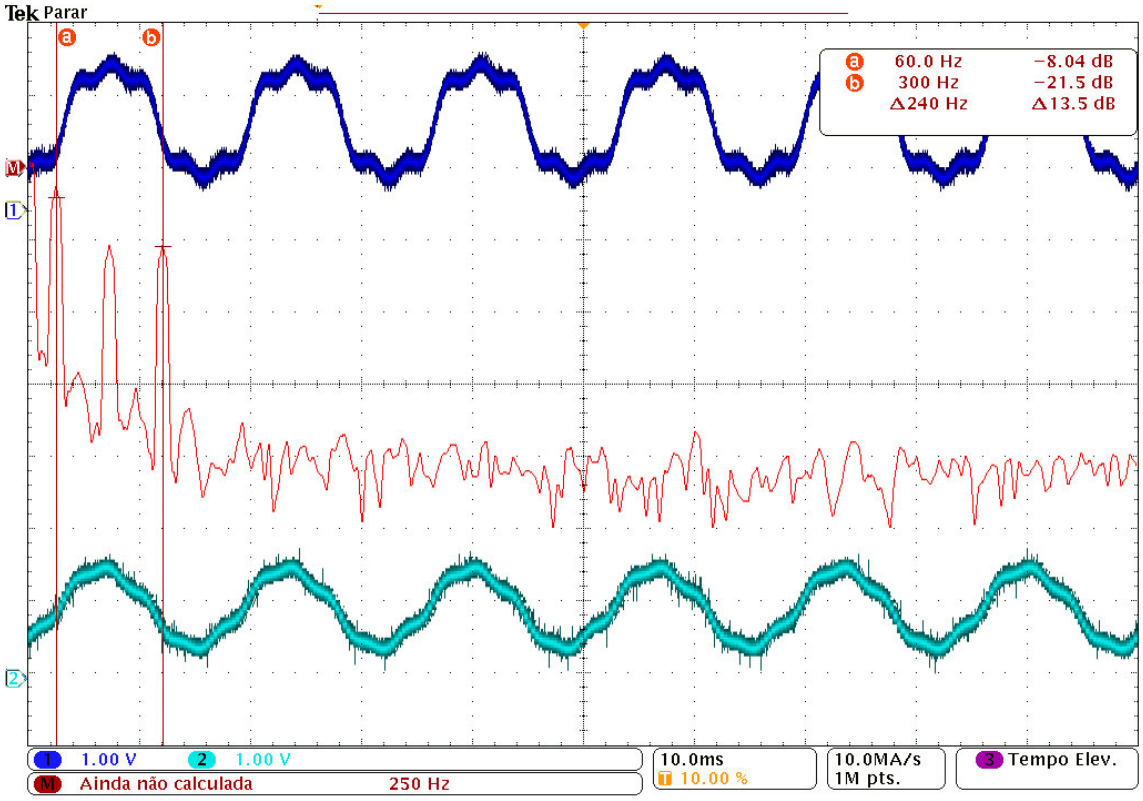


Figura 41 – Formas de onda de entrada (1), saída (2) e FFT (M) do sinal de entrada, 60 e 300 Hz

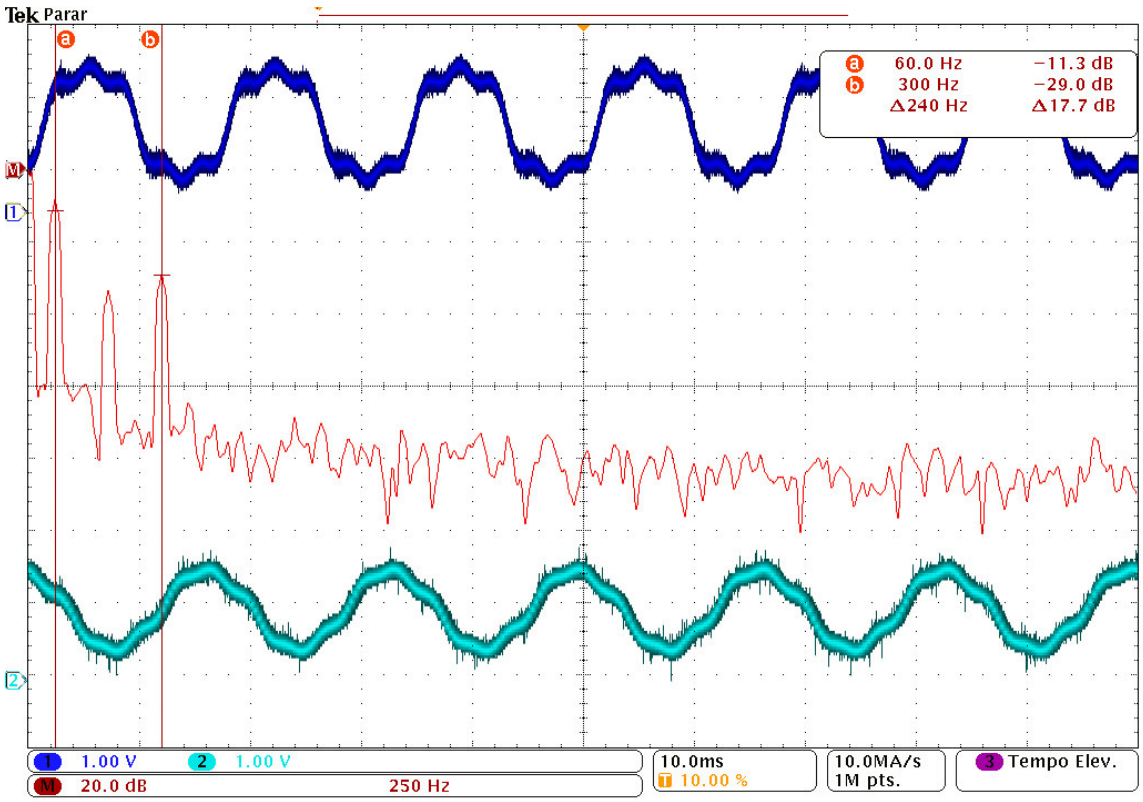


Figura 42 – Formas de onda de entrada (1), saída (2) e FFT (M) do sinal de saída, 60 e 300 Hz

4.6.2. Compensador Seletivo de Harmônicas de Ordem 100 Utilizando Paralelismo de FPGAs

Para melhorar a qualidade do sinal de saída, Mattavelli e Fasolo (2000) utilizam 200 pontos de amostragem por ciclo, consumindo um tempo de aproximadamente 18,4 μs para o processamento do bloco de compensação harmônica das fases α e β. A Tabela 4 mostra o consumo de *hardware* para a execução do filtro de 50 pontos, mostrando que o dispositivo estava no seu limite de utilização (*Number OF MULT 1818SIOs* = 100% e *Number of occupied Slices* = 98%). Os recursos excedentes são os *Flip Flops* (9%) e portas de entrada e saída (IOBs = 13%), que não são necessários em grande quantidade nos cálculos, no caso dos *Flip Flops,* ou não alteram sua quantidade em função do número coeficientes, como as portas de entrada e saída.

Os recursos com utilização exponencialmente crescente em função do número de cálculos são os CLBs (*Slices* = 98%) utilizados principalmente nas somas e deslocamentos e as tabelas Look-Up (LUTs = 76%), utilizadas como registradores.

Tabela 4 – Relatório de utilização do Dispositivo XC3S500E na aplicação do compensador seletivo de harmônicas utilizando 50 pontos de amostragem

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	906	9,312	9%	
Number of 4 input LUTs	7,144	9,312	76%	
Number of occupied Slices	4,591	4,656	98%	
Number of Slices containing only related logic	4,591	4,591	100%	
Number of Slices containing unrelated logic	0	4,591	0%	
Total Number of 4 input LUTs	7,293	9,312	78%	
Number used as logic	7,143			
Number used as a route-thru	149			
Number used as Shift registers	1			
Number of bonded IOBs	32	232	13%	
Number of BUFGMUXs	1	24	4%	
Number of MULT18X18SIOs	20	20	100%	
Average Fanout of Non-Clock Nets	4.06			

Para realizar a comparação e utilizar a técnica de paralelismo de ULA e de processadores foram utilizadas duas FPGAs, que permitiram um filtro seletivo de 100 pontos, ou seja, metade do utilizado do trabalho citado anteriormente. Considerando os recursos de *hardware* do FPGA XC3S500E seriam necessários quatro dispositivos para realizar o mesmo cálculo utilizando a estrutura que aproveita o máximo de paralelismo possível. A Figura 44 mostra um diagrama relacionando o tempo de processamento (ciclos), com os processos

realizados em ambas as FPGA, aqui nomeada de FPGA Mestre e FPGA Escravo. O FPGA Mestre é responsável pela leitura do conversor analógico/digital (A/D) e o envio deste dado para o registrador de entrada $x(s)$ e ao mesmo tempo para o FPGA Escravo. É importante ressaltar que o cálculo dos coeficientes é feito *a priori* e baseado nos parâmetros de ganhos e números de harmônicas a serem atenuadas, conforme visto no Capítulo 2, Seção 2.3.4.

A Figura 43 exibe o gráfico dos valores dos coeficientes utilizados para compensação da 3ª e 5ª harmônicas, com N=100. Os coeficientes variam em função das harmônicas que se deseja compensar.

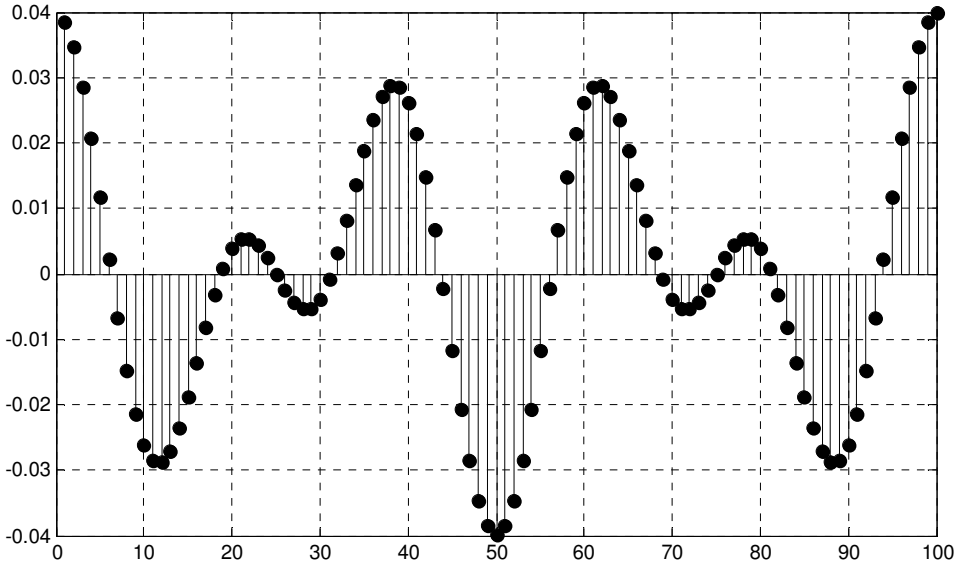


Figura 43 – Gráfico com os valores dos coeficientes do compensador seletivo de harmônicas com 100 pontos de amostragem

Os coeficientes para os filtros de 50 e 100 pontos podem ser vistos nos anexos A.1 e A.2 deste trabalho. Estes coeficientes foram calculados no *software* MATLAB® e salvos em registradores dos dois FPGA, conforme ilustrado na primeira etapa da Figura 44. Esta figura ilustra todas as etapas de processamento do compensador seletivo. Na etapa seguinte, ciclo 2, são realizadas 20 multiplicações por FPGA, limite este dado pelo número de multiplicadores dedicados do dispositivo utilizado nos testes. Com um dispositivo que possua mais multiplicadores dedicados o número de multiplicações pode ser maior, visto que neste processo os elementos dos vetores não são dependentes um dos outros. Após cada ciclo de multiplicação é realizado um somatório dos mesmos, fazendo desta forma até concluir as 50 multiplicações em cada FPGA. Na etapa seguinte, 8/9 e 10, são necessários três ciclos para acumular as somas de todas as multiplicações, estas precisaram ser divididas em mais de um ciclo para reduzir o consumo de hardware que ficou no limite para executar as operações. No

ciclo 11 os resultados são deslocados reduzindo a precisão para 14 bits, resolução esta utilizada pelo conversor digital/analógico (D/A). Nos ciclos 12 e 13 os resultados das multiplicações são subtraídos do sinal de entrada, neste ponto o sinal deve ser levando em conta para acrescer ou subtrair o erro do sinal de entrada $x[n]$. Finalmente o dado é enviado para o conversor D/A.

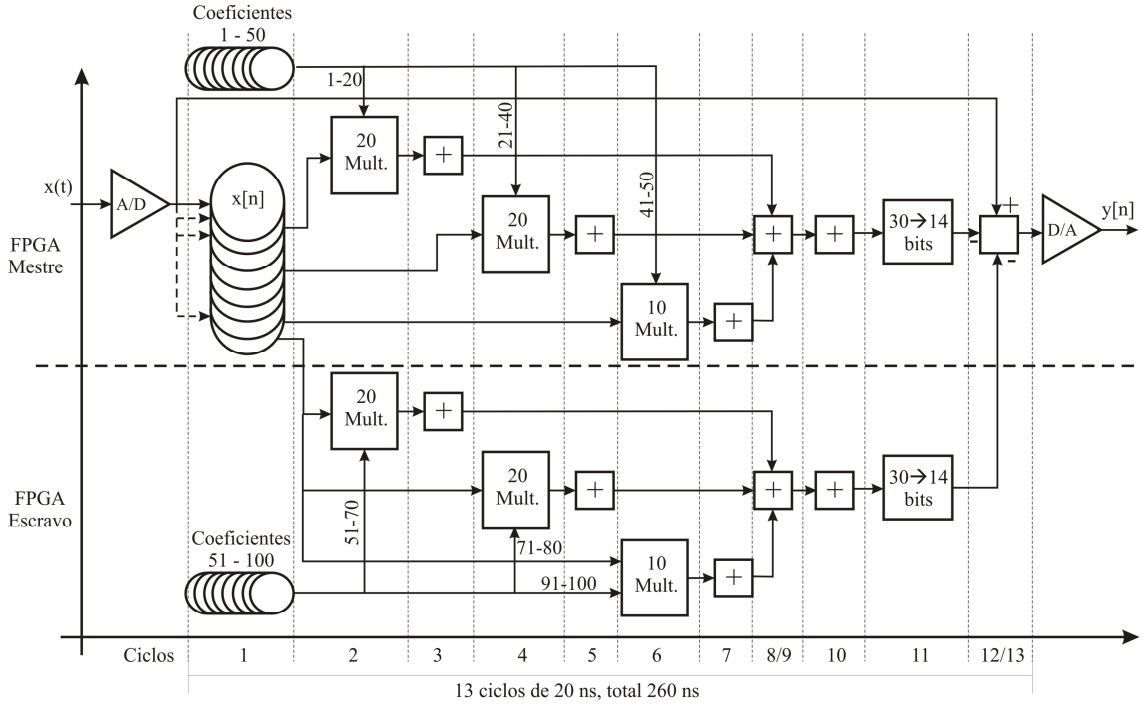


Figura 44 – Diagrama de processos paralelos entre os dispositivos FPGA

Utilizando uma técnica de processamento vetorial de acesso direto a registradores e não a memória principal é possível melhorar a operação da *pipeline*. Os elementos de entrada, coeficientes e os resultados são colocados nos vetores de registradores, assim as operações requerem acesso apenas aos registradores, fazendo acesso à memória principal apenas no início e término do processo. Esta estrutura de processamento chama-se *pipeline dentro de uma operação* (STALLINGS, 2002) porque tem-se uma única operação aritmética a vetores, com múltiplos elementos sendo processados ao mesmo tempo.

Em relação ao filtro com 50 pontos o tempo de processamento teve incremento de aproximadamente 17%, apesar de duplicarem o número de cálculos. No filtro com 100 pontos de amostragem o número total de ciclos para os cálculos de compensação harmônica foram de 13 ciclos, equivalente a 260 ns, a medição de tempo levando em conta os tempos de comunicação foram de 265 ns, conforme Figura 45. O ensaio realizado por Mattavelli e Fasolo (2000) com 200 pontos precisou de aproximadamente 9,2 μs por fase (α e β), proporcionalmente um filtro nas mesmas condições utilizando 100 pontos ficaria em torno de

5,2 µs. Ainda assim, este resultado é muito acima dos 265 ns obtidos com a plataforma FPGA e as técnicas de paralelismo utilizadas.

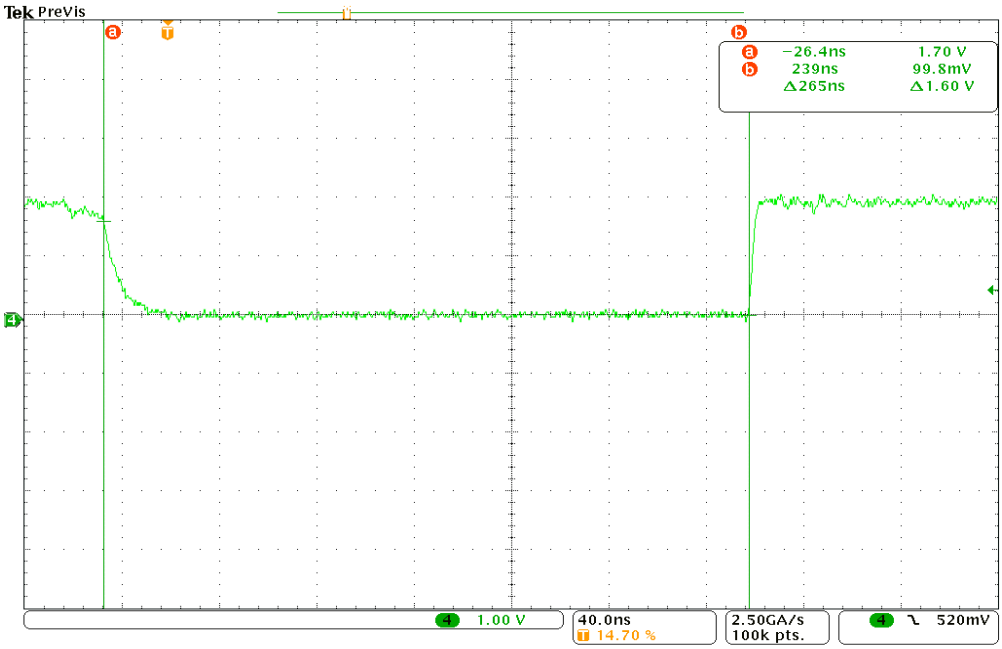


Figura 45 – Tempo de processamento do controle de harmônicas para o compensador seletivo de harmônicas utilizando 100 pontos de amostragem

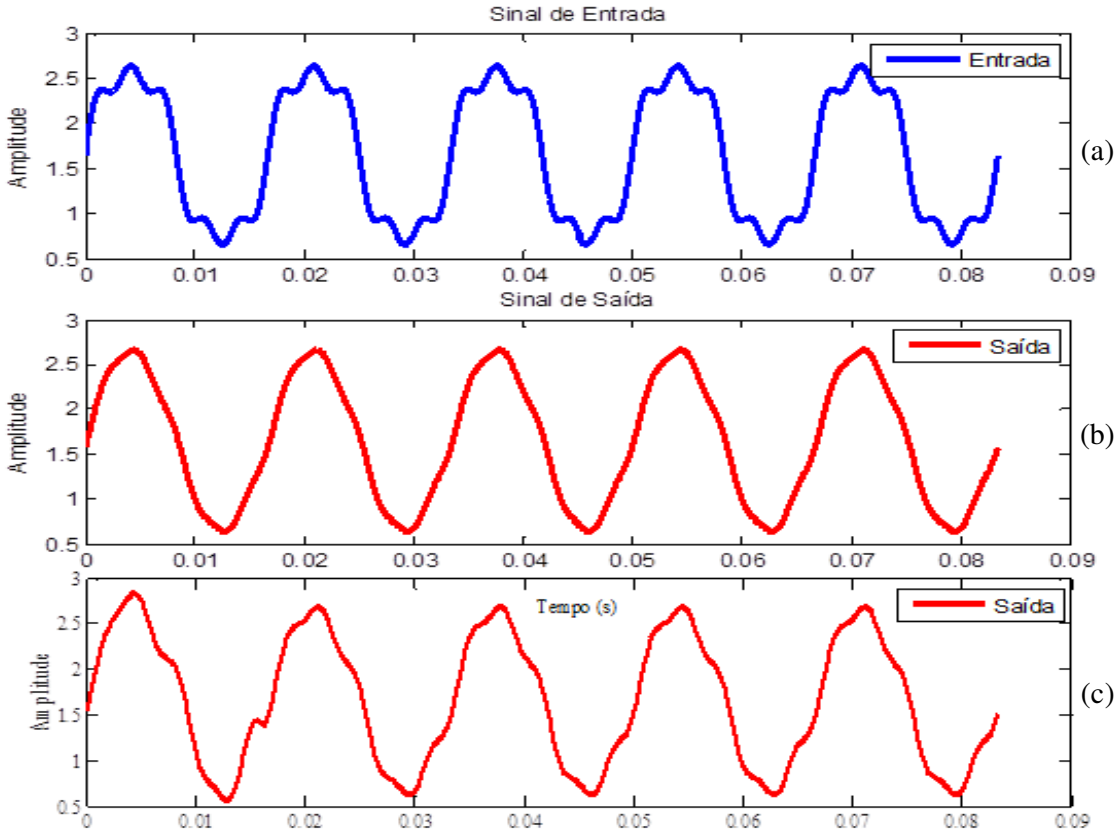


Figura 46 – Simulação computacional do compensador seletivo de harmônicas: (a) Sinal de Entrada, (b) Sinal de saída com 100 pontos e (c) Sinal de saída com 50 pontos.

Para avaliação dos sinais envolvidos nesta condição, foram realizadas simulações computacionais no *software* MabLab® e as formas de onda podem ser vistas na Figura 46. Percebe-se neste caso que com 100 pontos de amostragem (Figura 46b) o sinal de saída é significativamente melhorado em relação compensador com 50 pontos (Figura 46c).

A Figura 47 apresenta a análise harmônica do resultado da simulação computacional do compensador seletivo de harmônicas com 100 pontos. Verifica-se que antes da compensação o sinal de entrada possui aproximadamente 15% da fundamental como harmônicas de 3ª e 5ª ordem. Após a compensação, as referidas harmônicas são significativamente atenuadas sem alterar o valor da fundamental de forma acentuada.

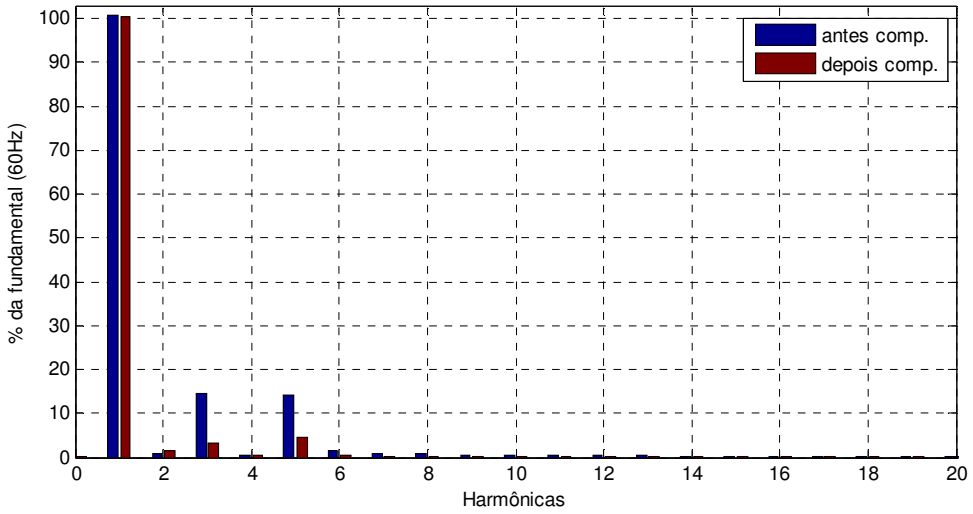


Figura 47 – Análise harmônica da simulação computacional do compensador seletivo de harmônicas com 100 pontos: antes e após a compensação.

As figuras que seguem apresentam os níveis do sinal de entrada das componentes harmônicas de 180 Hz, 57,7 mV (Figura 48) e 300 Hz, 56,9 mV (Figura 50). Apresentam também as formas de onda de entrada (2) e saída (1). Na sequência na Figura 49, é possível observar a redução da componente harmônica de 180 Hz para 11,1 mV e na Figura 51 a componente harmônica de 300 Hz é reduzida para 17,8 mV. Estes valores são bem melhores que o filtro de 50 pontos, mas ainda inferiores aos resultados que poderiam ser encontrados com 200 pontos, valores que para a implementação iriam requerer mais recursos de *hardware* do que o disponível para este projeto.

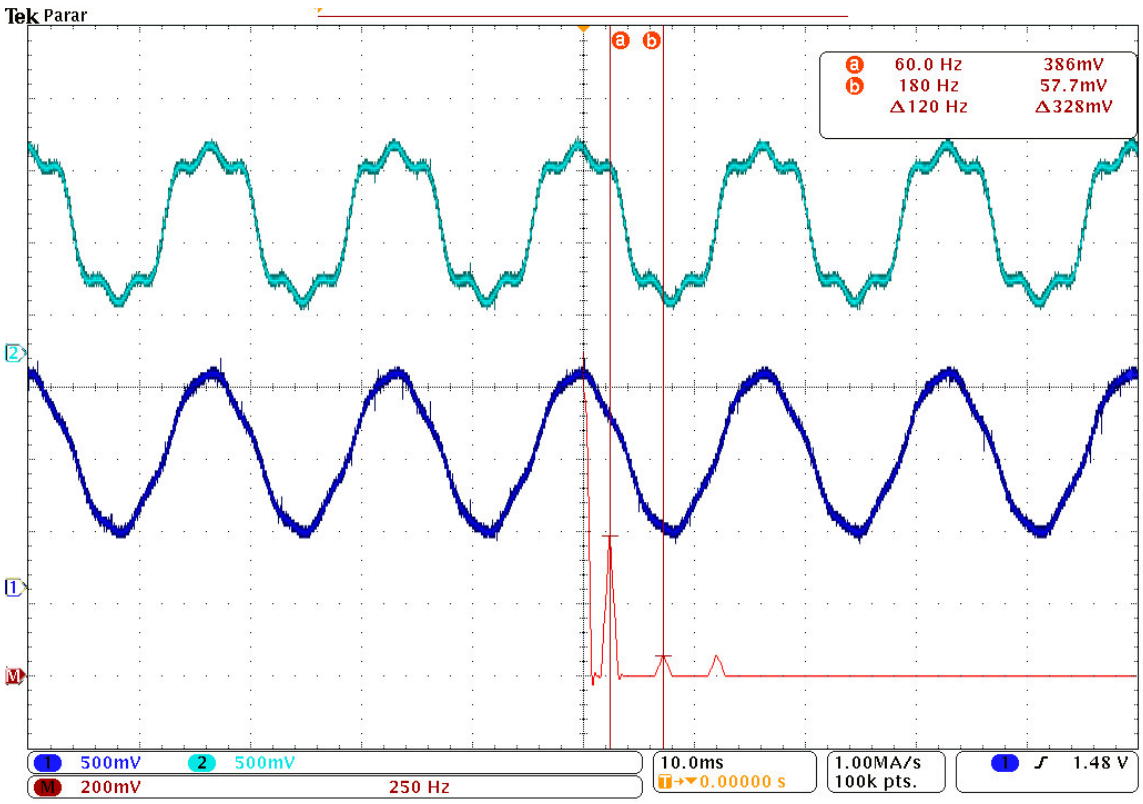


Figura 48 – Formas de onda de entrada (2), saída (1) e FFT (M) do sinal de entrada, 60 e 180 Hz.

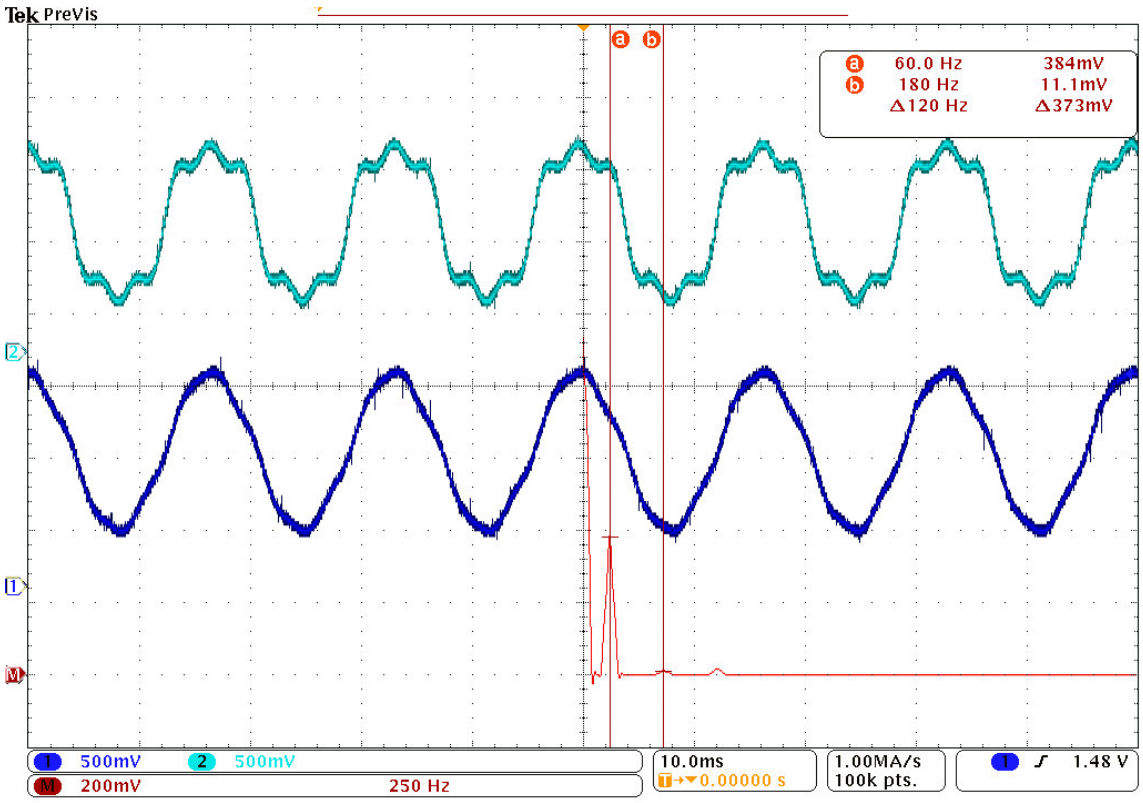


Figura 49 – Formas de onda de entrada (2), saída (1) e FFT (M) do sinal de saída, 60 e 180 Hz.

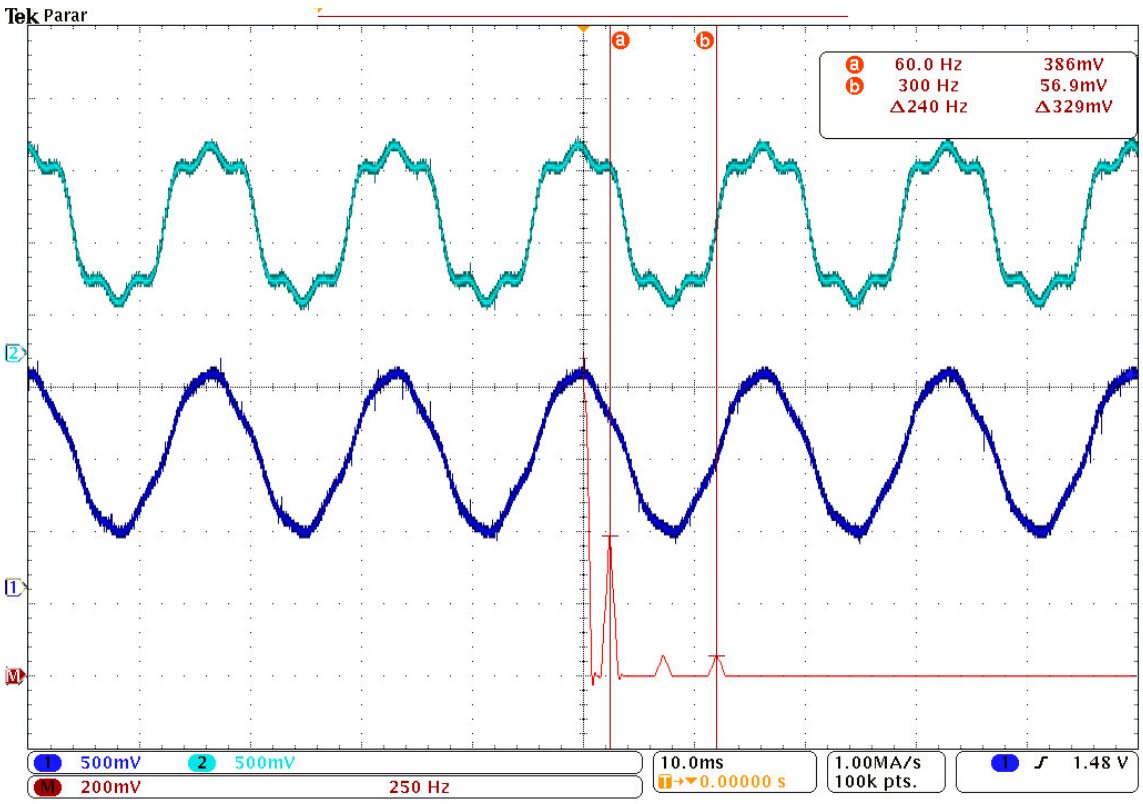


Figura 50 – Formas de onda de entrada (2), saída (1) e FFT (M) do sinal de entrada, 60 e 300 Hz.

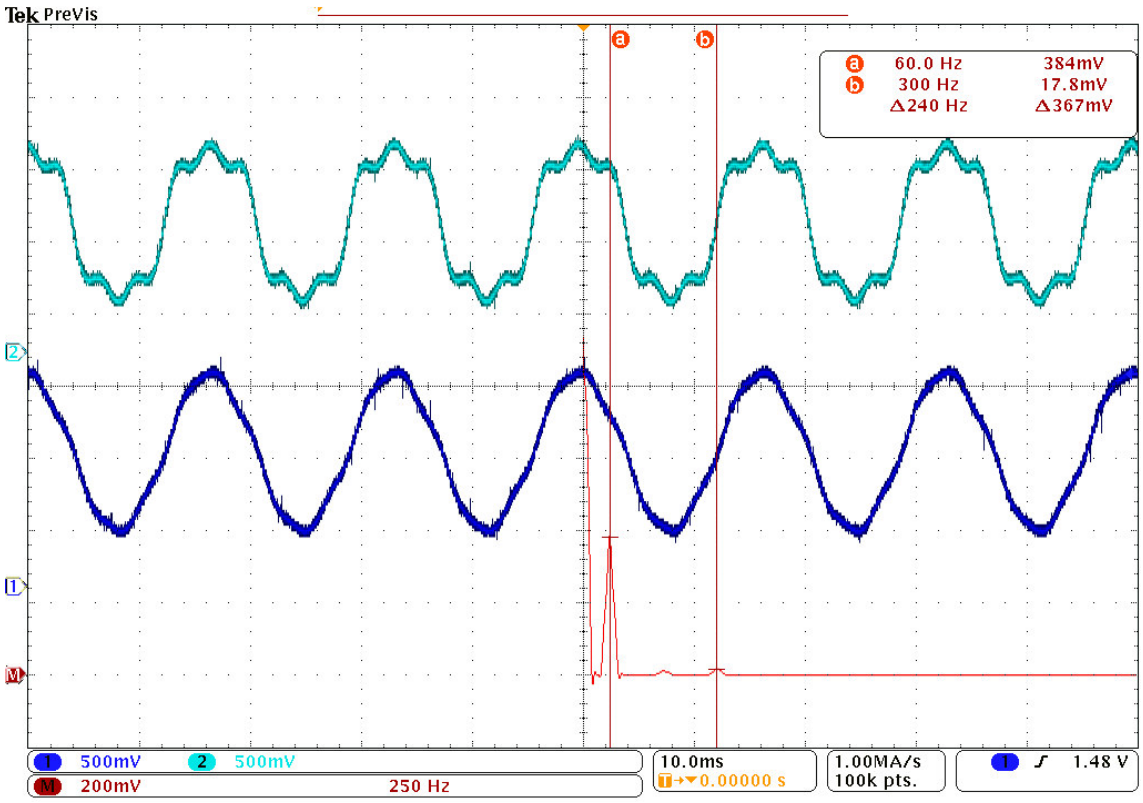


Figura 51 – Formas de onda de entrada (2), saída (1) e FFT (M) do sinal de saída, 60 e 300 Hz.

Na Tabela 5 são apresentadas as comparações entre os valores de THD das simulações e dos resultados práticos.

Tabela 5 – THD das simulações e dos resultados práticos

	Simulação Computacional	Resultados Práticos
Sinal de Entrada	21,21%	21,26%
Sinal de saída com compensador de 50 pontos	10,81%	12,92%
Sinal de saída com compensador de 100 pontos	5,46%	6,99%
Sinal de saída com compensador de 200 pontos	2,74%	-

Os resultados da THD das simulações ficaram próximos aos resultados práticos, mostrando que a aplicação dos cálculos está correta. Entre as causas das pequenas diferenças entre ambos, podem ser citados os erros de quantização utilizados no FPGA e limitações do A/D e D/A. Para avaliar o comportamento do sistema proposto com maior número de pontos foi realizada uma simulação do mesmo com 200 pontos em cada período da fundamental. O resultado obtido para o sinal de saída apresentou THD = 2,74%, sendo que não foram determinados resultados experimentais para este caso devido as limitações do *hardware* descritas neste Capítulo.

A Tabela 6 apresenta a comparação entre os tempos de processamento obtidos por Mattavelli e Fasolo (2000) e os resultados obtidos neste trabalho.

Tabela 6 – Comparação de tempos de processamento

N	Mattavelli e Fasolo utilizando DSP com estrutura sequencial	Este trabalho utilizando FPGA com estrutura paralela	Diferença
50 pontos	3,2 µs	226 ns	~14x
100 pontos	5,2 µs	265 ns	~19x

A comparação de tempos mostra uma considerável redução no tempo de processamento.

4.7. SUMÁRIO

Neste capítulo foram apresentados resultados experimentais obtidos com a implementação em FPGA de estruturas paralelas para filtros e controladores digitais. Comparados às simulações computacionais, as implementações apresentaram uma resposta muito aproximada, como pode ser verificado pelos resultados apresentados. A validação das respostas de filtros digitais permitem comprovar a viabilidade de implementação de estruturas aritméticas em tempo real em dispositivos FPGA.

O tempo de processamento necessário para os cálculos dos compensadores seletivos de harmônicos, apresentados no Capítulo 2 e implementados neste capítulo, foi na ordem de 265 ns, o que possibilita a utilização dos mesmos em altas frequências de atuação. É importante salientar que o controle escolhido tem como características um grande número de cálculos, mas sem grandes dependências entre eles, o que possibilita uma melhoria de desempenho quando executados em paralelo. O capítulo seguinte trata de forma mais detalhada estas considerações, fazendo um apanhado geral do trabalho e propondo possíveis desenvolvimentos em trabalhos futuros.

5. CONCLUSÃO GERAL E TRABALHOS FUTUROS

Este capítulo apresenta as considerações finais sobre a implementação de filtros e técnicas de controle digital utilizando processamento paralelo em dispositivos FPGA, e que podem ser aplicadas em filtros ativos de potência. As conclusões e análise dos dados obtidos são apresentadas na Seção 5.1. Propostas para trabalhos futuros dentro da área de estudo deste trabalho são apresentadas na Seção 5.2

5.1. CONCLUSÕES

Neste trabalho foi apresentado um estudo sobre algumas técnicas de controle utilizadas em filtros ativos de potência. Dentre estas, buscou-se as técnicas que se apresentassem mais apropriadas para programação paralela, e com características interessantes como redução do conteúdo harmônico e tempos de execução compatíveis com implementações em tempo real. Neste sentido considerou-se principalmente aquelas possíveis de serem implementadas através de dispositivos FPGA, e utilizando ponto fixo, já que em ponto flutuante os DSP que possuem um núcleo próprio para este fim são mais rápido que a utilização de bibliotecas de ponto flutuante disponíveis para FPGA.

Entre as estruturas analisadas, a que reuniu características interessantes, como pouca dependência entre os cálculos necessários, foi o compensador seletivo de harmônicas. Instruções dependentes entre si dificultam o processamento paralelo, pois para realizar um cálculo, que depende de outro valor ainda a ser processado, este precisa aguardar o término do primeiro para depois iniciar o seu processamento. O compensador seletivo de harmônicas baseou-se em algumas referências clássicas, como Mattavelli e Fasolo (2000) que foi desenvolvida em um dispositivo DSP utilizando processamento sequencial. Este trabalho serviu de referência para metodologia de projeto e para comparação de tempos de processamento com os resultados obtidos nesta dissertação. Na execução sequencial, mostrada do artigo citado anteriormente, um controlador de 50ª ordem, que executa a multiplicação de 50 coeficientes por 50 amostras em um filtro FIR (*Finite Impulse Response*), foram necessários aproximadamente 3,2 μs por fase (α e β) para executar os cálculos. O mesmo controlador apresentado neste trabalho, implementado em um FPGA e utilizando processamento paralelo, realizou esta tarefa em apenas 226 ns. O tempo de processamento sequencial foi de aproximadamente 14 vezes (3,2 μs/226 ns) maior que o paralelo, apesar do

FPGA ter menos do dobro velocidade do DSP utilizado no processamento sequencial, 50 MHz do FPGA XC3S500E da Xilinx e 26MIPS ADMC401 da Analog Devices.

Além de ser mais rápido, o XC3S500E da Xilinx é, hoje, aproximadamente 13% mais barato que o ADMC401 da Analog Devices, preços consultados na DigiKey Corporation, uma das principais distribuidoras mundiais de ambos os fabricantes.

O controlador implementado, com 50 amostras por ciclo, foi utilizado apenas para comparações de velocidade, uma vez que para melhores resultados o ideal é o uso de um número maior de pontos de amostragem por ciclo. Desta forma, um segundo controlador foi implementado com ordem 100, utilizando dois FPGA e paralelismo de processadores. Apesar do dobro do número de cálculos, a diferença do tempo de processamento foi apenas de 17,25% maior do que controlador de ordem 50, totalizando de 265 ns. Na execução sequencial um controlador de ordem 200 com um dispositivo DSP utilizou 9,2 µs por fase. Na proporção de acréscimo obtida de 50 para 200 pontos, um controlador de 100 pontos em execução sequencial levaria em torno de 5,2 µs para concluir os cálculos. Neste caso a diferença de velocidade aumenta para aproximadamente 19 vezes (5,2 µs/265 ns).

Uma vez que o custo de dois FPGA do modelo utilizado não chega a duas vezes o preço do DSP, o desempenho apresentado para 100 pontos supera em muito a diferença de preços, e, além disso, permite um aumento significativo na frequência de controle em malha fechada. Por outro lado, uma estrutura para 50 pontos, com todo processamento necessário, absorve praticamente todo o *hardware* de um FPGA e desta forma, para implementação de um controlador de ordem 200 iria requerer quatro dispositivos FPGA, ou uma família de FPGA diferente, como um Spartan-6, onde os recursos de um XC6SLX75 de U$ 94,56 já seriam suficientes para implementar o compensador de ordem 200 em um tempo ainda menor.

Do ponto de vista de amostragem de sinais, o FPGA não possui embarcado todos os recursos adicionais que existem em DSPs, necessitando se adicionar conversores A/D e D/A, contadores PWM, entre outros periféricos que geralmente são necessários para implementação da solução completa. Assim estes custos precisam também ser considerados. Desta forma, a solução mais viável é o uso do FPGA como um coprocessador, deixando-o trabalhar nos processos que necessitam de velocidade e que podem fazer uso de paralelismo, enquanto o processador principal, um DSP, controla o restante das operações.

De um modo geral os resultados obtidos foram muito satisfatórios, o controlador apresentou um tempo de processamento significativamente menor e sem implicações significativas de custo. Ainda, os resultados de desempenho dinâmico em diversas frequências

ficaram muito próximos aos resultados de simulações computacionais realizadas em MATLAB®.

5.2. TRABALHOS FUTUROS

Os resultados provenientes deste trabalho despertam possíveis trabalhos futuros, entre eles:

- Implementação experimental de um FAP utilizando as técnicas de processamento paralelo propostas neste trabalho;
- Implementação de outros controladores utilizando as mesmas técnicas de processamento paralelo para comparação de resultados;
- Implementação do mesmo controlador com um número menor de multiplicações por ciclo para otimização de hardware e possível aumento de tempo, mas ainda dentro do aceitável.
- Desenvolvimento do mesmo controlador proposto neste trabalho para um FPGA de maior capacidade, caso em que é possível implementar um controlador de maior ordem, ou seja, mais amostragens por ciclo e uma compensação de harmônicas de ordens mais elevadas;
- Desenvolvimento de estruturas de controle e/ou processamento de sinais utilizando outros *hardwares* adequados para processamento paralelo, como GPUs de última geração e processadores multinúcleos, como processadores Cell e Core i7;
- Desenvolvimento de estruturas de controle utilizando ponto flutuante para comparação com resultados de ponto fixo.

6. REFERÊNCIAS

AKAGI, Hirofumi; KANAZAWA, Yoshihira; NABAE, Akira. Instantaneous Reactive Power Compensators Comprising Switching Devices without Energy Storage Components. **IEEE Trans. Ind. Appl.**, 20, 1984. 625-630.

ANEEL - PRODIST. **Procedimentos de Distribuição de Energia Elétrica no Sistema Elétrico Nacional – PRODIST**. Módulo 8 – Qualidade da Energia Elétrica, Revisão 2, vigente a partir de 01/01/2011. [S.l.]: [s.n.]. 2010.

BANERJEE, P. An Overview of a Compiler for Mapping MATLAB Programs onto FPGAs. **Proc. ASPDAC**, 2003. 477 - 482.

BLOCH, Erich. The engineering design of the stretch computer. **Proc. IRE/AIEE/ACM Eastern Joint Computer Conference**, New York, 1959.

BROWN, Stephen; VRANESIC, Zvonko. **Fundamentals of Digital Logic with VHDL Design**. 2ª. ed. New York: McGraw-Hill, 2004.

CARRILLO, Jorge E.; CHOW, Paul. The Effect of Reconfigurable Units in Superscalar Processors. **Proc. Ninth International Symposium on Field Programmable Gate Arrays, FPGA 2001**, New York, 2001. 141-150.

CASILLO, Leonardo A. et al. Projeto e Implementação em um FPGA de um processador com conjunto de instrução reconfigurável utilizando VHDL. **In proceedings of XI Workshop Iberchip**, Salvador, 2005.

CASTILHO CARDIM, Marcio H. **RTRASSOC51: Módulo de Pipeline para um Processador com Arquitetura Harvard Superescalar Embarcado (PAHSE)**. Dissertação de Mestrado, Centro Universitário "Eurípedes de Marília" – UNIVEM. Marília. 2006.

DE ROSE, César A. F.; NAVAUX, Philippe O. A. **Arquiteturas Paralelas**. Porto Alegre: Bookman, 2008.

DOTE, Yasuhiko; HOFT, Richard G. **Intelligent Control - Power Electronic Systems**. [S.l.]: Oxford University Press Inc, 1998.

ECKERT, J. P. et al. Design of Univac®-LARC system: I. **Proc. IRE/AIEE/ACM Eastern Joint Computer Conference**, New York, 1959.

EXCITE. **eXCite**. Disponivel em: <http://www.yxi.com/products.php>. Acesso em: 23 Outubro 2010.

HALDAR, Malay et al. A SYSTEM FOR SYNTHESIZING OPTIMIZED FPGA HARDWARE FROM MATLAB. **IEEEACM International Conference on Computer Aided Design ICCAD 2001**, 2001. 314-319.

HWANG, Frank K. **Computer Arithmetic:** Principles, Architecture and Design. [S.l.]: John Wiley & Sons Inc, 1979.

IEC 61000-3. **Electromagnetic compatibility**. Tech. Rep., International Electrotechnical Commission. [S.l.]: [s.n.]. 1996.

IEEE STD 1076-2008. **IEEE Standard VHDL Language Reference Manual**. IEEE Std 1076-2008. [S.l.]: [s.n.]. 2008.

IEEE STD 519-1992. **IEEE Recommended practices and requirements for harmonic control in electrical power systems**. IEEE Std 519-1992. [S.l.]: [s.n.]. 1992.

IEEE STD 929-2000. **IEEE Recommended Practice for Utility Interface of Photovoltaic (PV) Systems**. IEEE Std 929-2000. [S.l.]: [s.n.]. 2000.

INOGUCHI, Yasushi. Outline of the Ultra Fine Grained Parallel Processing by FPGA. **Seventh International Conference on High Performance Computing and Grid in Asia Pacific Region**, 2004.

JAKOBSEN, Lars T.; ANDERSEN, Michael A. E. Digitally Controlled Point of Load Converter with Very Fast Transient Response. **Power Electronics and Applications**, set. 2007.

JENSEN, Lars B. W. et al. **A hybrid FPGA/coarse parallel processing architecture for multi-modal visual feature descriptors**. International Conference on Reconfigurable Computing and FPGAs. Cancun: [s.n.]. 2008. p. 241-246.

JUNG, Eunsoo; SUL, SSeung-Ki. **Implementation of grid-connected single-phase inverter based on FPGA**. TwentyFourth Annual IEEE Applied Power Electronics Conference and Exposition. Washington: [s.n.]. 2009. p. 889.

KAWAMURA, Atsuo; HANEYOSHI, Toshimasa; HOFT, Richard G. Deadbeat controlled PWM inverter with parameter estimation using only voltage sensor. **IEEE TRANSACTIONS ON POWER ELECTRONICS**, Abril 1988.

KUO, Benjamin C. **Digital Control Systems**. 2ª. ed. New York: Oxford University Press, 1992.

MALESANI, Luigi; MATTAVELLI, Paolo; BUSO, Simone. **Dead-Beat Current Control for Active Filters**. Annual Conference of the IEEE Industrial Electronics Society. Aachen: [s.n.]. 1998. p. 1859.

MATTAVELLI, Paolo. A Closed-Loop Selective Harmonic Compensation for Active Filters. **IEEE Transactions on Industry Applications**, 37, 2001. 81-89.

MATTAVELLI, Paolo; FASOLO, Sandro. Implementation of synchronous frame harmonic control for high-performance AC power supplies. **Industry Applications Conference**, 2000. pp. 1988-1995.

MATTAVELLI, Paolo; FASOLO, Sandro. Implementation of synchronous frame harmonic control for high-performance AC power supplies. **Industry Applications Conference**, 2000. 1988-1995.

MIDDLETON, Richard H.; GOODWIN, Graham C. **Digital Control And Estimation:** a unified approach. [S.l.]: Prentice, 1990.

MONTAGNER, Vinícius F.; CARATI, Emerson G.; GRÜNDLING, Hilton A. An Adaptive Linear Quadratic Regulator with Repetitive Controller Applied to Uninterruptible Power Supplies. **Industry Applications Conference**, 2000. pp. 2231 - 2236.

MOSSOBA, Joseph; LEHN, Peter W. A controller architecture for high bandwidth active power filters. **IEEE Transactions on Power Electronics**, 18, 2003. 317 - 325.

NEUMANN, Von J. First Draft of a Report on the EDVAC, 1945.

NEWMAN, Michael J.; HOLMES, Donald G. Delta operator digital filters for high performance inverter applications. **IEEE Transactions on Power Electronics**, v. 18, p. 447 - 454, 2003.

OMNI. **Omni:** OpenMP compiler project. Disponivel em: <http://phase.hpcc.jp/Omni/>. Acesso em: 25 Outubro 2010.

SERA, Dezso et al. **Low-Cost digital implementation of proportional-resonant current controllers for PV inverter applications using delta operator**. Annual Conference of IEEE Industrial Electronics Society. Raleigh: [s.n.]. 2005. p. 6.

SHANTHALA, S; KULKARNI, S Y. VLSI Design and Implementation of Low Power MAC Unit with Block Enabling Technique. **European Journal of Scientific Research**, 2009. pp.620-630.

STALLINGS, William. **Arquitetura e Organização de Computadores**. 5ª Ed.\. ed. São Paulo: Prentice Hall, 2002.

STEFANELLO, Márcio. **Controle Adaptativo Robusto de Estrutura Variável por Modelo de Referência Aplicado a Filtros ativos de Potência**. Universidade Federal de Santa Maria. Santa Maria. 2010.

SUGAHARA, Kazuki; OIDA, Shinsuke; YOKOYAMA, Tomoki. High Performance FPGA Controller for Digital Control of Power Electronics Applications. **Power Electronics and Motion Control Conference**, New York, Maio 2009.

SUN, Welson; WIRTHLIN, Michael J.; NEUENDORFFER, Stephen. FPGA Pipeline Synthesis Design Exploration Using Module Selection and Resource Sharing. **IEEE Trans Comput Aided Design Integr Circuits Syst 26**, 2007. 86 - 95.

TEIFEL, John; MANOHAR, Rajit. Programmable Asynchronous Pipeline Arrays. **Proceedings of the 13th International Conference on Field Programmable Logic and Applications**, Lisboa, Setembro 2003. 345 - 354.

WAN, Kai; FERDOWSI, Mehdi. Reducing Computational Time Delay in Digital Current-Mode Controllers for Dc-Dc Converters. **Telecommunications Energy Conference**, Setembro 2008.

XILINX PICOBLAZE USER GUIDE. **PicoBlaze 8-bit Embedded Microcontroller User Guide**. [S.l.]: [s.n.]. 2004.

ZMOOD, Daniel N.; HOLMES, Donald G. Stationary frame current regulation of PWM inverters with zero steady-state error. **IEEE Trans. on Power**, 18, 2003. 814-822.

ANEXO A – COEFICIENTES DOS COMPENSADORES SELETIVOS DE HARMÔNICAS

A.1 COEFICIENTES DO COMPENSADOR SELETIVO DE HARMÔNICAS COM 50 PONTOS DE AMOSTRAGEM POR CICLO

A Tabela 7 apresenta os valores calculados pelas equações apresentadas no Capítulo 2, Seção 2.3.4.

Tabela 7 – Coeficientes do filtro com 50 pontos de amostragem por ciclo

Coeficientes		
Coeficiente	**Valor Decimal**	**Valor Hexadecimal**
1	9068	236C
2	5413	1525
3	609	0261
4	-3892	0F34
5	-6827	1AAB
6	-7543	1D77
7	-6182	1826
8	-3562	0DEA
9	-832	0340
10	996	03E4
11	1425	0591
12	634	027A
13	-634	027A
14	-1425	0591
15	-996	03E4
16	832	0340
17	3562	0DEA
18	6182	1826
19	7543	1D77
20	6827	1AAB
21	3892	0F34
22	-609	0261
23	-5413	1525
24	-9068	236C
25	-10430	28BE
26	-9068	236C
27	-5413	1525
28	-609	0261

29	3892	0F34
30	6827	1AAB
31	7543	1D77
32	6182	1826
33	3562	0DEA
34	832	0340
35	-996	03E4
36	-1425	0591
37	-634	027A
38	634	027A
39	1425	0591
40	996	03E4
41	-832	0340
42	-3562	0DEA
43	-6182	1826
44	-7543	1D77
45	-6827	1AAB
46	-3892	0F34
47	609	0261
48	5413	1525
49	9068	236C
50	10430	28BE

A.2 COEFICIENTES DO COMPENSADOR SELETIVO DE HARMÔNICAS COM 100 PONTOS DE AMOSTRAGEM POR CICLO

A Tabela 8 apresenta os valores calculados pelas equações apresentadas no Capítulo 2, Seção 2.3.4.

Tabela 8 – Coeficientes do filtro com 100 pontos de amostragem por ciclo

Coeficientes		
Coeficiente	**Valor Decimal**	**Valor Hexadecimal**
1	5041	13B1
2	4534	11B6
3	3734	0E96
4	2707	0A93
5	1533	05FD
6	304	0130
7	-884	0374
8	-1946	079A

9	-2807	0AF7
10	-3413	0D55
11	-3736	0E98
12	-3772	0EBC
13	-3542	0DD6
14	-3091	0C13
15	-2480	09B0
16	-1781	06F5
17	-1070	042E
18	-416	01A0
19	121	0079
20	498	01F2
21	695	02B7
22	712	02C8
23	573	023D
24	317	013D
25	0	0000
26	-317	013D
27	-573	023D
28	-712	02C8
29	-695	02B7
30	-498	01F2
31	-121	0079
32	416	01A0
33	1070	042E
34	1781	06F5
35	2480	09B0
36	3091	0C13
37	3542	0DD6
38	3772	0EBC
39	3736	0E98
40	3413	0D55
41	2807	0AF7
42	1946	079A
43	884	0374
44	-304	0130
45	-1533	05FD
46	-2707	0A93
47	-3734	0E96
48	-4534	11B6
49	-5041	13B1
50	-5215	145F
51	-5041	13B1
52	-4534	11B6
53	-3734	0E96

54	-2707	0A93
55	-1533	05FD
56	-304	0130
57	884	0374
58	1946	079A
59	2807	0AF7
60	3413	0D55
61	3736	0E98
62	3772	0EBC
63	3542	0DD6
64	3091	0C13
65	2480	09B0
66	1781	06F5
67	1070	042E
68	416	01A0
69	-121	0079
70	-498	01F2
71	-695	02B7
72	-712	02C8
73	-573	023D
74	-317	013D
75	0	0000
76	317	013D
77	573	023D
78	712	02C8
79	695	02B7
80	498	01F2
81	121	0079
82	-416	01A0
83	-1070	042E
84	-1781	06F5
85	-2480	09B0
86	-3091	0C13
87	-3542	0DD6
88	-3772	0EBC
89	-3736	0E98
90	-3413	0D55
91	-2807	0AF7
92	-1946	079A
93	-884	0374
94	304	0130
95	1533	05FD
96	2707	0A93
97	3734	0E96
98	4534	11B6

99	5041	13B1
100	5215	145F

ANEXO B – PLATAFORMA DE ENSAIOS

B.1 KIT DE DESENVOLVIMENTO SPARTAN-3E

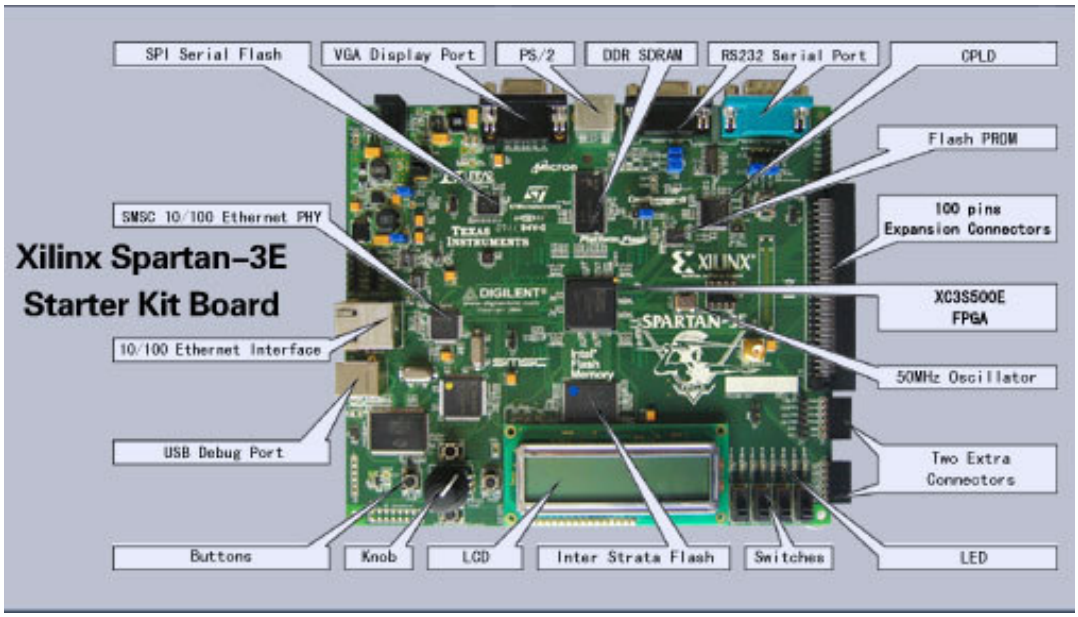


Figura 52 – Kit de Desenvolvimento Spartan-3E

B.2 BANCADA DE TESTES

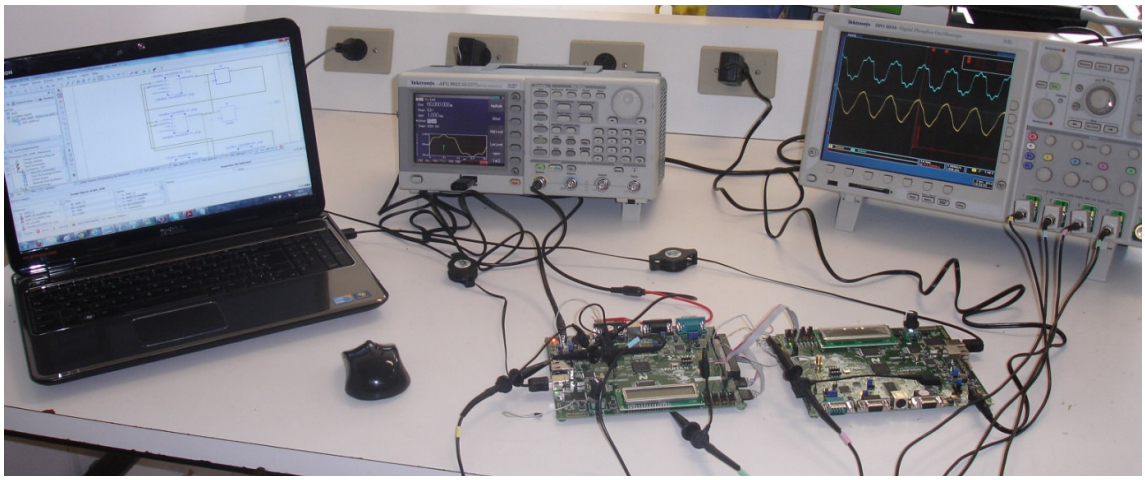


Figura 53 – Bancada de testes

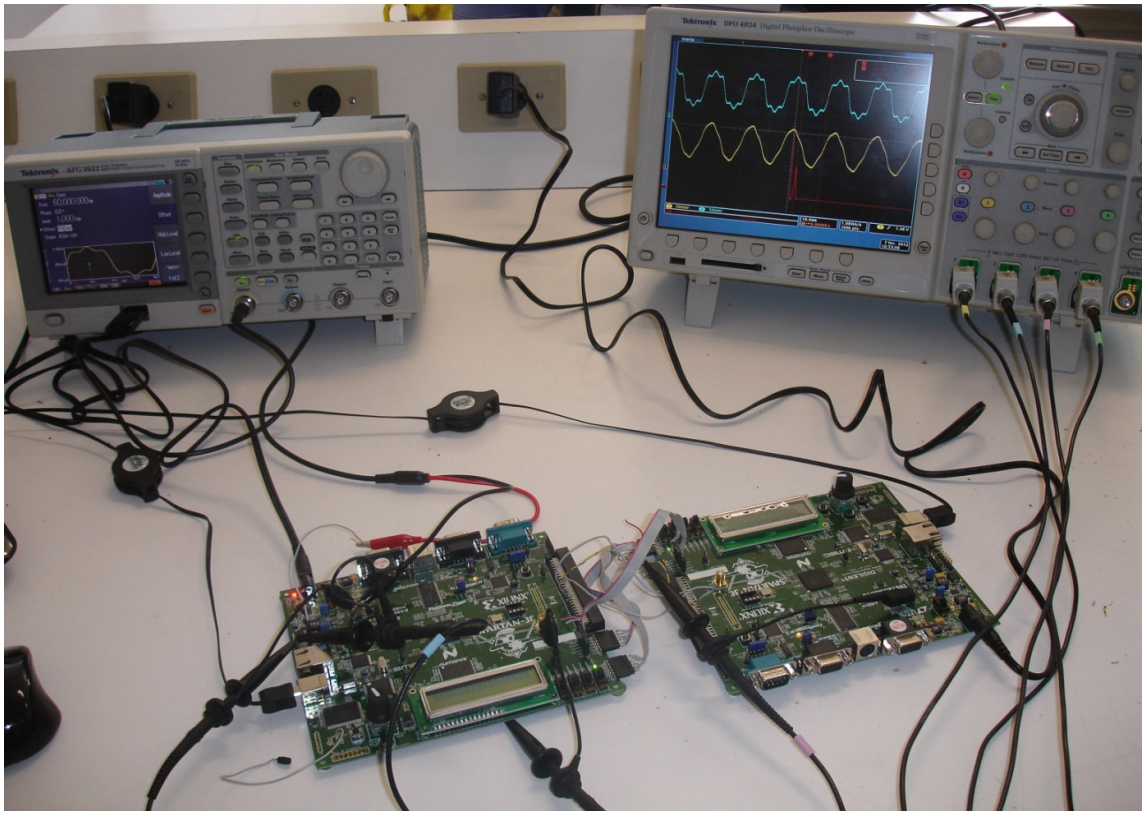


Figura 54 – Instrumentos na bancada de testes

APÊNDICE A – Operador Delta

Para definir o operador Delta, primeiramente é preciso definir o operador de deslocamento (SERA et al., 2005), o operador de deslocamento q muda a entrada de tempo discreto, com uma amostra onde x_k é um elemento da entrada:

$$qx_k = x_{k+1} \tag{A.1}$$

Substituindo q por z e invertendo o operador de deslocamento porque a equação (A.1) não é causal, tem-se:

$$z^{-1}x_k = x_{k-1} \tag{A.2}$$

Para encontrar o operador delta substitui-se o operador de deslocamento na equação (A.2) pelo operador de diferenças:

$$\delta = \frac{q-1}{\Delta} \tag{A.3}$$

onde Δ o período de amostragem.

Por analogia da relação entre q e z, γ pode ser definido como:

$$\gamma = \frac{z-1}{\Delta} \tag{A.4}$$

Substituindo (A.3) em (A.1) temos:

$$\delta x_k = \frac{x_{k+1} - x_k}{\Delta} \tag{A.5}$$

Para uso em aplicações práticas a melhor forma de γ:

$$\gamma^{-1} = \frac{z^{-1}\Delta}{1-z^{-1}} \tag{A.6}$$

Em (MIDDLETON e GOODWIN, 1990) é apresentada a definição da transformada z de uma sequência $Z\{x_k\}$:

$$X(z) = Z\{x_k\} = \sum_{k=0}^{\infty} x_k z^{-k} \tag{A.7}$$

Substituindo z por $1+\Delta\gamma$em (A.7) a transformação delta tem a seguinte definição:

$$X(\gamma) = D\{x_k\} = \sum_{k=0}^{\infty} x_k (1+\Delta\gamma)^{-k} \tag{A.8}$$

A forma canônica do filtro de segunda ordem utilizando operador de deslocamento pode ser visto abaixo:

$$H_{PR}(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}} \tag{A.9}$$

Da mesma forma pode ser definida a forma canônica do filtro de segunda ordem com o operador γ:

$$H_{PR}(\gamma) = \frac{\beta_0 + \beta_1 \gamma^{-1} + \beta_2 \gamma^{-2}}{\alpha_0 + \alpha_1 \gamma^{-1} + \alpha_2 \gamma^{-2}} \tag{A.10}$$

A relação entre os coeficientes de $H_{PR}(z)$ e $H_{PR}(\gamma)$ pode ser vista na Tabela 9.

Tabela 9 – Relação entre os coeficientes de $H_{PR}(z)$ e $H_{PR}(\gamma)$

Coeficientes	
α	**β**
$\alpha_0 = 1$	$\beta_0 = b_0$
$\alpha_1 = \frac{2 + a_1}{\Delta}$	$\beta_1 = \frac{2b_0 + b_1}{\Delta}$
$\alpha_2 = \frac{1 + a_1 + a_2}{\Delta^2}$	$\beta_2 = \frac{b_0 + b_1 + b_2}{\Delta^2}$

A partir dos coeficientes da Tabela 1 pode-se implementar um controlador ressonante na forma recursiva:

$$u_{Hi}(k) = \alpha_1' u_{Hi}(k-1) + \alpha_2' u_{Hi}(k-2) + \beta_0' e(k) + \beta_1' e(k-1) + \beta_2' e(k-2) \tag{A.11}$$

onde os coeficientes estão normalizados em relação a α_0.