UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE CIÊNCIAS E TECNOLOGIA
COORDENAÇÃO DO CURSO DE ENGENHARIA ELÉTRICA

RELATÓRIO DE ESTÁGIO

CHRISTIAN FARIAS DA SILVA

Relatório apresentado à Coordenação de Estágios em Engenharia Elétrica da UFPB como parte integrante dos requisitos necessários à obtenção do título de Engenheiro Eletricista.

Campina Grande – PB, 10 de Março de 2000

Biblioteca Setorial do CDSA. Fevereiro de 2021.

Sumé - PB

ÍNDICE

APRESENTAÇÃO

Este relatório tem como objetivo apresentar as atividades realizadas pelo aluno CHRISTIAN FARIAS DA SILVA durante estágio supervisionado no Laboratório de Instrumentação Eletrônica e Controle (LIEC) no período de 03 de dezembro de 1999 à 28 de fevereiro de 2000, sob a orientação do professor RAIMUNDO CARLOS SILVÉRIO FREIRE.

O estágio foi realizado na área de concepção e prototipagem de ASICs (*Application Specific Integrated Circuit)* em dispositivos lógicos programáveis, mais especificamente em FPGAs (*Field Programmable Gate Array).*

Foi realizada uma revisão bibliográfica tanto em eletrônica digital quanto em arquiteturas de sistemas digitais, o que porém não foi suficiente, sendo necessária a busca por novos conceitos em outras áreas (circuitos integrados reconfiguráveis e eletrônica de potência, como veremos mais adiante) para que fosse possível o trabalho com as ferramentas disponíveis para o desenvolvimento do projeto e o entendimento (pelo menos parcial) da teoria por trás do mesmo.

O projeto proposto para ser implementado em FPGA foi de um circuito digital capaz de gerar sinais modulados em largura de pulso para o acionamento de inversores trifásicos. Esse trabalho já havia sido iniciado porém ainda não havia atingido a fase de programação do dispositivo, a qual conseguimos atingir.

A divisão desse relatório foi feita do seguinte modo: na primeira parte é dada uma introdução sobre a principal forma de se projetar circuitos digitais atualmente, através de linguagens de descrição de hardware (HDL – *Hardware Description Language*); o projeto de circuitos integrados digitais e suas várias formas é abordado na segunda parte; na terceira parte é apresentado o sistema de prototipagem com o qual trabalhamos no LIEC, desde o software até a placa, passando pelo dispositivo em si (FPGA); a quarta parte é reservada a apresentação do projeto do gerador de sinais PWM (*Pulse Width Modulation*), desde os conceitos empregados até os resultados obtidos; na quinta parte são feitas algumas considerações sobre o estágio e suas implicações.

1. Introdução a Linguagens de Descrição de Hardware

Define-se como HDL, de maneira geral, toda linguagem utilizada para descrever um sistema digital [1]. Essas linguagens geralmente apresentam diversos níveis de descrição, desde o mais baixo, como a disposição de transistores num circuito integrado, até uma descrição puramente comportamental, sendo esse o mais alto nível. Nesses níveis, se assemelham muito a linguagens de alto nível já existentes. Só pra exemplificar, as duas maiores HDLs, Verilog e VHDL, em seus níveis mais altos se assemelham às linguagens C e ADA, respectivamente [1].

Há um nível de descrição intermediário chamado de RTL (*Register Transfer Level*), no qual descreve-se os registradores e as transferências de vetores de informação entre os mesmos. Esse é o nível mais usado para descrição de sistemas digitais por ser o mais alto nível de descrição que ainda mantém uma estrutura de síntese lógica simples e por ainda apresentar independência quanto a tecnologia de fabricação do circuito integrado. Se a descrição é feita nos níveis mais básicos, a dependência com a tecnologia é grande e a flexibilidade de fabricação é perdida.

Sistemas digitais em geral não são simples. Em detalhes, podem consistir em milhões de elementos, isto é, transistores ou portas lógicas. Deste modo, para sistemas digitais de grande porte, o projeto no nível de portas lógicas torna-se inviável. Por muitas décadas, esquemáticos das portas lógicas serviram como a linguagem comum dos projetos digitais. Hoje, a complexidade do hardware cresceu de tal maneira que um esquemático com portas lógicas torna-se quase inútil, uma vez que apenas mostra uma teia de conexões entre os componentes, não evidenciando nada com relação à funcionalidade do projeto [1].

A mais significativa mudança ocorrida nas últimas décadas no modo de se projetar sistemas digitais foi sem dúvida a transição do uso de esquemáticos lógicos para o uso de ferramentas de modelagem comportamental e síntese lógica. Durante muito tempo as habilidades essenciais de um projetista de circuito integrado digital estavam na capacidade do manuseio eficiente do *layout* do circuito no nível dos transistores e na capacidade do uso de ferramentas de captura capazes de, através desses

layouts, gerar um esquemático no nível das portas lógicas. Essas habilidades, no entanto, foram sendo substituídas, gradativamente, por outras relacionadas ao domínio das HDLs. Segundo *Devadas, et al.*, "A importância de cada uma dessas capacidades está agora se tornando secundária com relação a capacidade de escrita de um modelo eficiente em HDL de um circuito integrado"[2].

O trabalho com linguagens de descrição de hardware surgiu no final dos anos 70 e cresceu paralelo ao desenvolvimento de várias das tecnologias empregadas atualmente no desenvolvimento de circuitos integrados digitais. Como citamos anteriormente, as duas mais utilizadas HDLs, inclusive nas indústrias, são Verilog e VHDL (*VHSIC Hardware Description Language*). Verilog é a principal HDL, utilizada por mais de 10000 projetistas e fabricantes de hardware como a Apple Computer, Motorola e Sun Microsystems [1].

A linguagem utilizada no laboratório foi Verilog, sendo essa escolha realizada em função de já existirem trabalhos envolvendo a mesma no LIEC. Além disso, o sistema de prototipagem de ASICs disponível no LIEC, suporta Verilog, como será mostrado na parte III deste relatório. A seguir é dada uma introdução informativa sobre Verilog.

I.1. VERILOG HDL

Embora seja a mais utilizada, é difícil dizer se Verilog é também a melhor das HDLs. Muitos afirmam que Verilog é mais utilizada por ser mais fácil de aprender que VHDL e também por ser muito parecida com a linguagem C, a mais aceita e difundida das linguagens de alto nível. Ambas já são padrões IEEE (*Institute of Electrical and Electronic Engineers*), o que atesta a importância das mesmas.

A linguagem Verilog foi desenvolvida em 1985 pela Gateway Design System Corporation, agora parte da Cadence Design Systems. Até maio de 1990, com a formação da *Open Verilog International* (OVI), organização que comanda o desenvolvimento da linguagem, Verilog era propriedade da Cadence. A Cadence abriu a linguagem a domínio público com a expectativa de que o mercado para *softwares* relacionados à mesma crescesse rapidamente [1], o que de fato aconteceu.

A linguagem Verilog em seu nível mais elevado de abstração, o nível comportamental, apresenta-se de forma muito semelhante a um código fonte em C ou

pascal. O projetista deve se preocupar apenas com a concepção funcional do circuito, sem se preocupar com a estrutura física (circuito digital) necessária para a realização da função. Por exemplo, uma multiplicação pode ser perfeitamente inserida no código da forma como é inserida num código em linguagem C ou MATLAB© .

À medida que o projetista avança rumo à implementação do circuito digital, isto é, desce na hierarquia de abstração do circuito, funções como essa precisam serem substituídas por códigos baseados em funções sintetizáveis logicamente. Uma soma pode ser considerada atualmente uma função plenamente sintetizável. Assim, a multiplicação que anteriormente era descrita simplesmente por A x B, precisa agora ser substituída por alguma estrutura de multiplicação baseada em deslocamentos e somas [3]. Essas dificuldades surgem geralmente no nível RTL.

Descendo mais ainda na hierarquia de abstração, o projetista ainda pode fazer uso da linguagem Verilog para ajustar a implementação física do circuito pós-síntese e/ou pós-geração das máscaras. Isso ocorre porque um circuito VLSI (*Very Large Scale Integration)* obtido através da síntese de um projeto descrito em Verilog, pode ser muito lento e ocupar muita área na pastilha de silício, sendo necessária a interferência do projetista nesses níveis. Essa prática requer um grau elevado de especialização em circuitos integrados digitais e tem se tornado rara a medida que as ferramentas de síntese tornam-se mais eficientes e os circuitos mais complexos.

II. Projeto de Circuitos Integrados Digitais

A rápida evolução dos circuitos digitais nas últimas décadas alterou de forma radical o processo de projeto dos mesmos. Com o aumento do número de componentes dos circuitos, isto é, o número de portas lógicas, foi atingido o chamado nível de VLSI (*Very Large Scale Integration)*. O surgimento de softwares empregados no desenvolvimento desses circuitos, bem como o aparecimento das HDLs, aceleraram os seus ciclos de projeto. Somando-se a isso temos o ciclo de vida dos produtos modernos se tornando menor que o próprio ciclo de projeto [4]. Assim, surge a necessidade de uma prototipagem cada vez mais rápida dos circuitos, de modo que não ocorram atrasos devido a erros em etapas avançadas do projeto, como na própria fabricação.

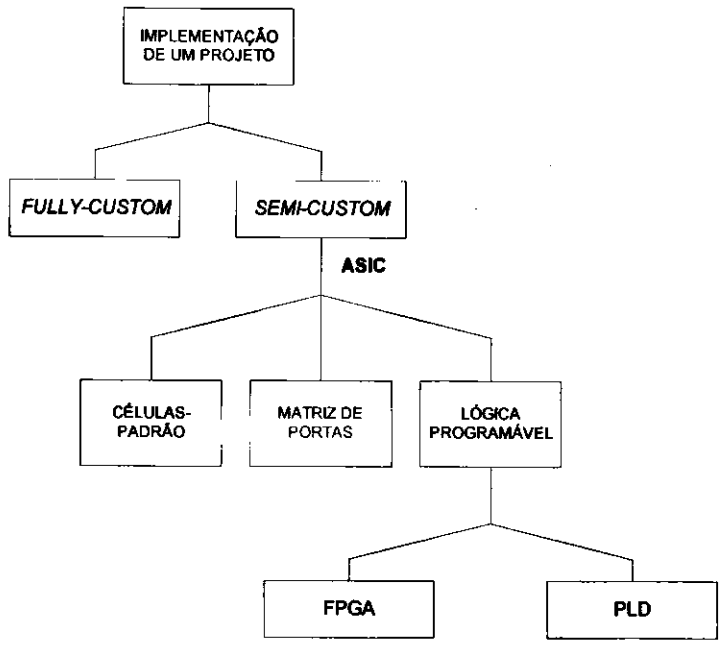
Esses e outros fatores impulsionaram o surgimento de dispositivos lógicos programáveis a fim de que fosse facilitada a tarefa de prototipagem do circuito digital, através da redução do tempo de implementação física do circuito. Esses dispositivos tornaram-se assim uma alternativa muito utilizada hoje em dia não só para prototipagem, mas também para sistemas reconfiguráveis [5].

Com a adição desses dispositivos, a implementação de circuitos integrados digitais ganhou mais uma forma de realização, e atualmente pode ser dividida em duas grandes categorias principais, *fully custom* e *semi-custom*, como mostra a figura II.1.

A escolha de uma das implementações é feita com base em diversos fatores, sendo os principais a disponibilidade tecnológica e o custo do projeto.

Na clássica categoria de projeto *full-custom*, cada função lógica primitiva ou transistor é manualmente projetado e otimizado. Isso resulta numa pastilha mais compacta, de alta velocidade e de baixa dissipação de potência. No entanto, o investimento inicial ou custo NRE (*Non-Recurring Engineering)* é mais alto comparado aos outros estilos de projeto. O projetista precisa manipular as formas geométricas que representam cada transistor na pastilha. Um projeto relativamente simples de 3000 portas pode requerer a manipulação de 300000 retângulos por pastilha [2]. Embora este estilo de projeto tenha sido o único usado nos primeiros circuitos integrados, os engenheiros raramente o utilizam na implementação de ASICs atualmente devido aos

altos custos de engenharia e baixa produtividade dos projetistas. A produtividade para um projeto digital *full custom* é de 6 a 17 transistores por dia [2].



II.1. Implementação de Circuitos Integrados Digitais

Já na categoria *semi-custom*, a atuação do projetista é mais limitada na fase de implementação do projeto, sendo essa a categoria mais aplicada à implementação de ASICs. Existem diversas maneiras de se implementar um circuito integrado nessa categoria, como mostra a figura II.1. De modo geral, elas dividem-se em células padrão (*standard cells*), matrizes de portas e lógica programável.

A implementação em células padrão é a mais utilizada quando o projetista chega até a geração das máscaras do seu ASIC. Essa geração de máscaras, realizada através de um software de maneira automatizada, é feita com base numa biblioteca de células-padrão fornecida pela própria fábrica que irá produzir o circuito integrado. Essa é a única das implementações de circuitos na categoria *semi-custom* que necessita de uma etapa de fabricação.

A implementação através de matrizes de portas já não é muito utilizada por ser uma forma pouco flexível de implementar circuitos digitais, estando já quase que totalmente substituída pela lógica programável.

Dispositivos de lógica programável são dispositivos pré-fabricados, nos quais a lógica é implementada programando-se eletricamente as interconexões entre as células

internas do dispositivo, em geral em um laboratório ao invés de uma fábrica [4]. Esses dispositivos apresentam diversas arquiteturas e se dividem em duas grandes categorias que ás vezes se confundem, a dos PLDs (*Programmable Logic Device*) e a dos FPGAs (*Field Programmable Gate Array*). Essa confusão no que diz respeito ao que é PLD e o que é FPGA é produto da evolução dos dispositivos. Isso se deve também às nomenclaturas utilizadas pelos fabricantes desses dispositivos. A ALTERA® , por exemplo, denomina seus dispositivos de PLDs, mas todos conhecem seus dispositivos como sendo FPGAs. Isso porque FPGA é a denominação utilizada por outro fabricante de dispositivos de lógica programável, a XILINX®, que difundiu esse tipo de dispositivo. De qualquer forma, mesmo sendo um dispositivo produto da evolução do outro, vamos aqui fazer uma distinção entre os mesmos de maneira genérica e superficial.

PLDs são, de maneira simplificada, matrizes programáveis de estruturas de soma de produtos (AND-OR). São utilizados em circuitos onde não é exigida muita flexibilidade nem muita lógica sequencial. Já os FPGAs são dispositivos mais complexos, que consistem em vários blocos lógicos, nos quais o projeto é realizado. Essa divisão em blocos e a composição dos mesmos varia de forma radical de fabricante para fabricante, não sendo aqui possível a explicação genérica dessa divisão para todos os dispositivos. Mais adiante será apresentado o dispositivo utilizado no nosso trabalho e serão dadas maiores explicações sobre essas estruturas.

As conexões internas são realizadas através de diferentes formas, desde por células de RAM (*Random Access Memory*) estática, passando por EEPROMs (*Electrical Erasable Programmable Read-Only Memory*) até *anti-fuses*.

As principais vantagens desse tipo de dispositivo são: a disponibilidade no mercado, já que são produzidos em larga escala; a possibilidade de substituírem circuitos integrados digitais VLSI; a facilidade com que erros podem ser corrigidos; o baixo risco com relação ao funcionamento do dispositivo e a capacidade de reprogramabilidade, que existe atualmente em quase todos os dispositivos.

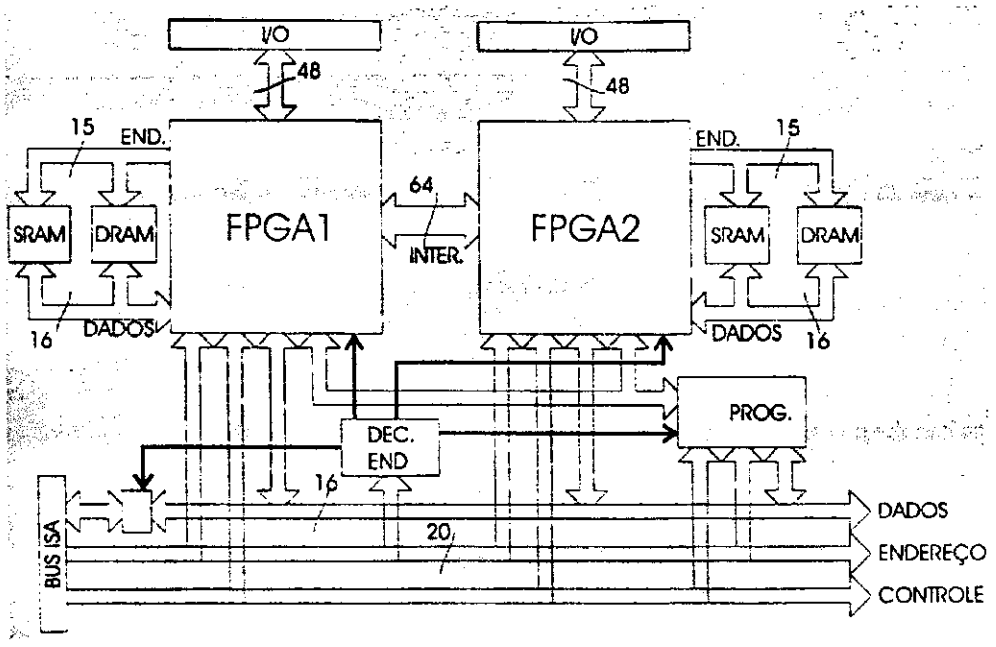
Embora os FPGAs representem, apenas, uma pequena porcentagem do total de vendas de mercado de ASICs, estatísticas indicam que, aproximadamente, metade de todos os projetos de pastilhas desenvolvidas atualmente iniciaram utilizando-se FPGAs [2].

Os dispositivos utilizados no nosso estágio fazem parte da família FLEX10K® da ALTERA . Antes de falarmos sobre esses dispositivos, daremos uma rápida introdução ao sistema de prototipagem disponível no LIEC, iniciando pela placa de programação utilizada, projetada no próprio DEE (Departamento de Engenharia Elétrica).

III. O Sistema De Prototipagem em FPGA

III.1. A placa SPRIT

O sistema utilizado para a programação dos FPGAs foi desenvolvido dentro do trabalho de dissertação de Mestrado do aluno Marcos R. A. Morais [6]. Denominado SPRIT, esse sistema foi concebido para a prototipagem rápida em FPGA de algoritmos de processamento de imagens em tempo real. No entanto, isso não impede que o sistema seja utilizado (como veremos mais adiante) para outras aplicações, como o próprio projetista frisou [6]. A arquitetura da placa é mostrada na figura seguinte.



III.1. Arquitetura da Placa SPRIT

Os dois FPGAs disponíveis para programação são conectados de forma que possam processar em cadeia, com diversos pinos de interconexão, possuindo barramentos que permitem a comunicação com o hospedeiro (PC) numa taxa de transferência relativamente alta. Cada FPGA possui muitos recursos de memória local e de E/S (Entrada/Saída), sendo programáveis pela tecnologia SRAM.

Como o objetivo deste relatório não é a descrição detalhada da placa SPRIT, daremos a seguir algumas das informações que consideramos relevantes para o entendimento da mesma [6].

→ Foram empregados dispositivos da família FLEX10K da ALTERA ;

→ Os dispositivos escolhidos apresentam encapsulamento SMD de 240 pinos (RC240);

→ Podem ser utilizados dispositivos de 20 a 70 mil portas equivalentes;

→ Comunicação com o hospedeiro (PC) é feita através do barramento ISA;

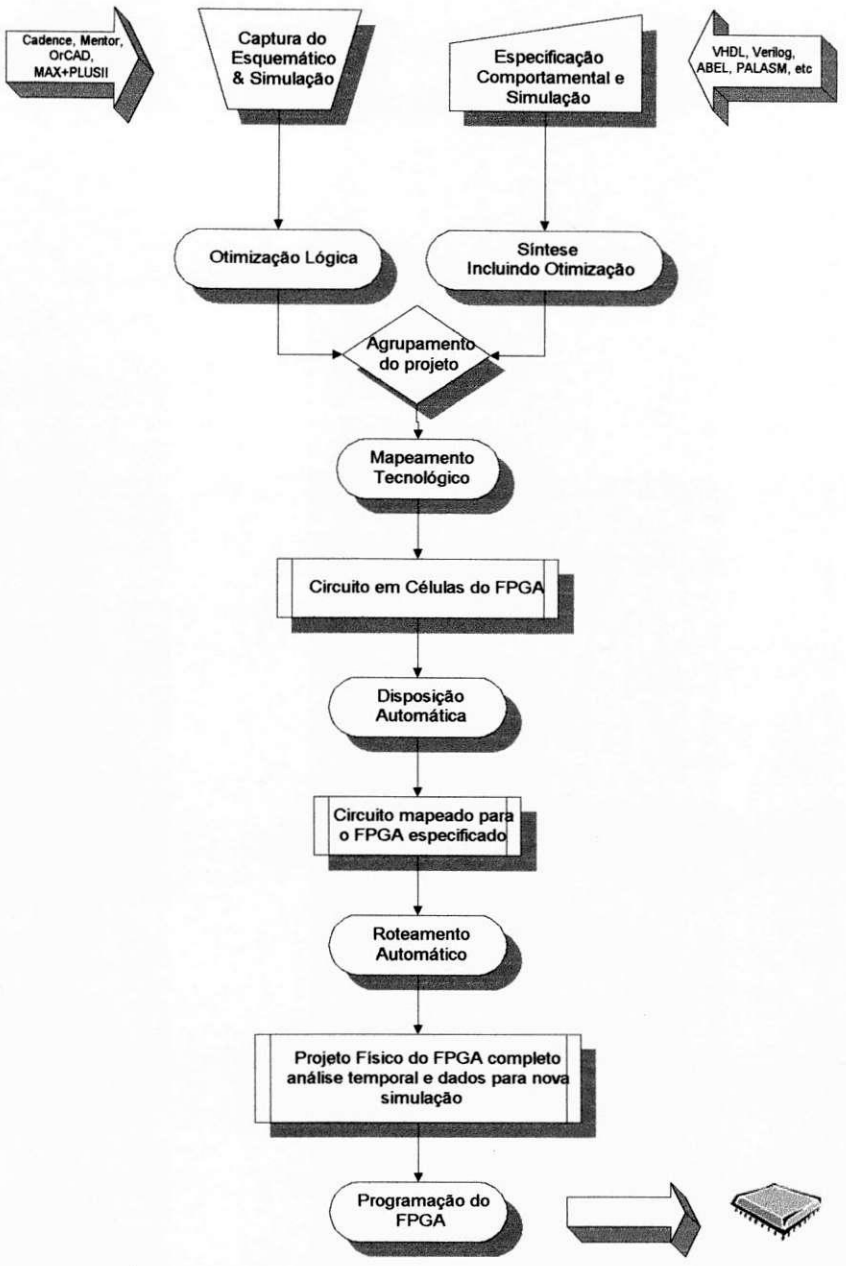
→ Um pequeno programa é utilizado para a programação do FPGA através do arquivo *bitstream* gerado pelo software da ALTERA .

A placa não se comunica diretamente com o software utilizado para o desenvolvimento do projeto em FPGA, no nosso caso, o MAX+PLUS© II e por isso existe a necessidade de uma etapa de carregamento do arquivo gerado pelo mesmo para o FPGA. Apresentaremos agora de maneira suscinta o software MAX+PLUS II e algumas de suas mais importantes características.

III.2. O MAX+PLUSII

O software MAX+PLUS II (*Multiple Array MatriX Programmable Logic User System II*) disponibiliza um completo ambiente de projeto com várias ferramentas de desenvolvimento que abrangem praticamente todas as formas possíveis de se desenvolver o projeto de um circuito integrado digital [7].

Ferramentas de CAD (*Computer Aided Design*) para FPGAs geralmente seguem um fluxo de projeto tal como é mostrado na figura III.2. . No MAX+PLUS II todas essas etapas são realizadas, sendo possível a interferência do projetista em praticamente todas elas .

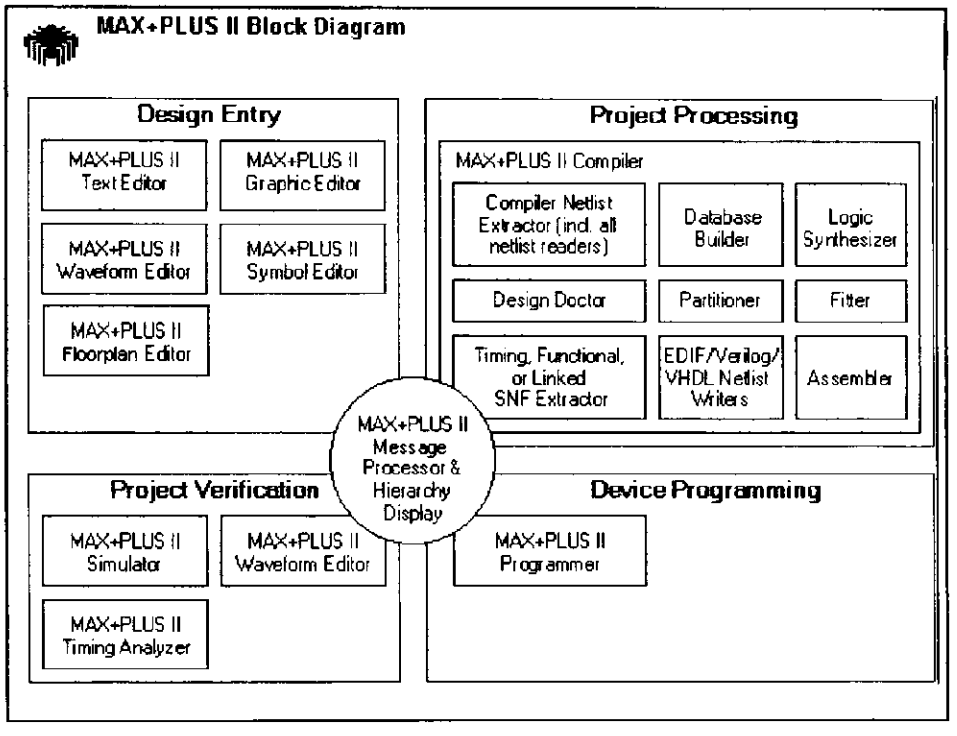


III.2. Fluxograma típico de projetos utilizando sistemas CAD para FPGAs

O MAX+PLUS II suporta diversas famílias de dispositivos programáveis da ALTERA (Classic©, MAX©5000, MAX7000, MAX9000, FLEX©6000, FLEX 8000, e FLEX 10K). Entre suas características principais estão: a possibilidade de três métodos de entrada de projeto; a possibilidade do projetista editar a disposição do circuito no dispositivo; síntese lógica de alto desempenho; o particionamento do projeto, quando necessário; a realização de simulação funcional e simulação junto com a placa; a

programação do dispositivo e a verificação diretamente do ambiente de desenvolvimento (se o hardware de programação da ALTERA estiver presente).

O MAX+PLUS II também lê arquivos *netlist* EDIF padronizados, arquivos *netlist* VHDL, Verilog, arquivos esquemáticos OrCAD e *netlists* da XILINX. Também exporta grande parte dos seus dados para outros formatos padrões na indústria. A figura seguinte nos mostra através de um diagrama de blocos toda a integração das capacidades do MAX+PLUS II [7] .



III.3. Diagrama de Blocos do MAX+PLUS II

III.3. DISPOSITIVOS FLEX10K

A família de dispositivos FLEX10K da ALTERA apresenta as seguintes características fundamentais :

- Conexões realizadas por SRAM (*Static Random Acess Memory*);
- De 10000 à 250000 portas para a realização de um sistema digital;
- Operação com alimentação de 3,3 V ou 5,0 V;
- Opção de estado de alta impedância em todos os pinos de entrada e saída;

- Blocos de RAM embutida (*Embedded RAM*).

A família FLEX10K é composta de vários dispositivos, que apresentam a mesma estrutura e só diferem nas suas capacidades de implementarem circuitos digitais. A seguir encontra-se uma tabela com alguns desses dispositivos e suas características.

	10K10	10K20	10K30	10K40	10K50	10K70	10K100
PORTAS	10000	20000	30000	40000	50000	70000	100000
ELEMENTOS LÓGICOS	576	1152	1728	2304	2880	3744	4992
BITS DE RAM	6144	12288	12288	16384	20480	18432	24576
REGISTRADORES	720	1344	1968	2576	3184	4096	5392
PINOS DE E/S PARA USO	150	198	248	278	310	358	406

Tabela III.1. : Dispositivos FLEX10K.

Cada dispositivo da família FLEX10K contém uma matriz embutida para implementar funções de memória e funções lógicas mais complexas, e uma matriz lógica para implementar funções lógicas mais simples [8].

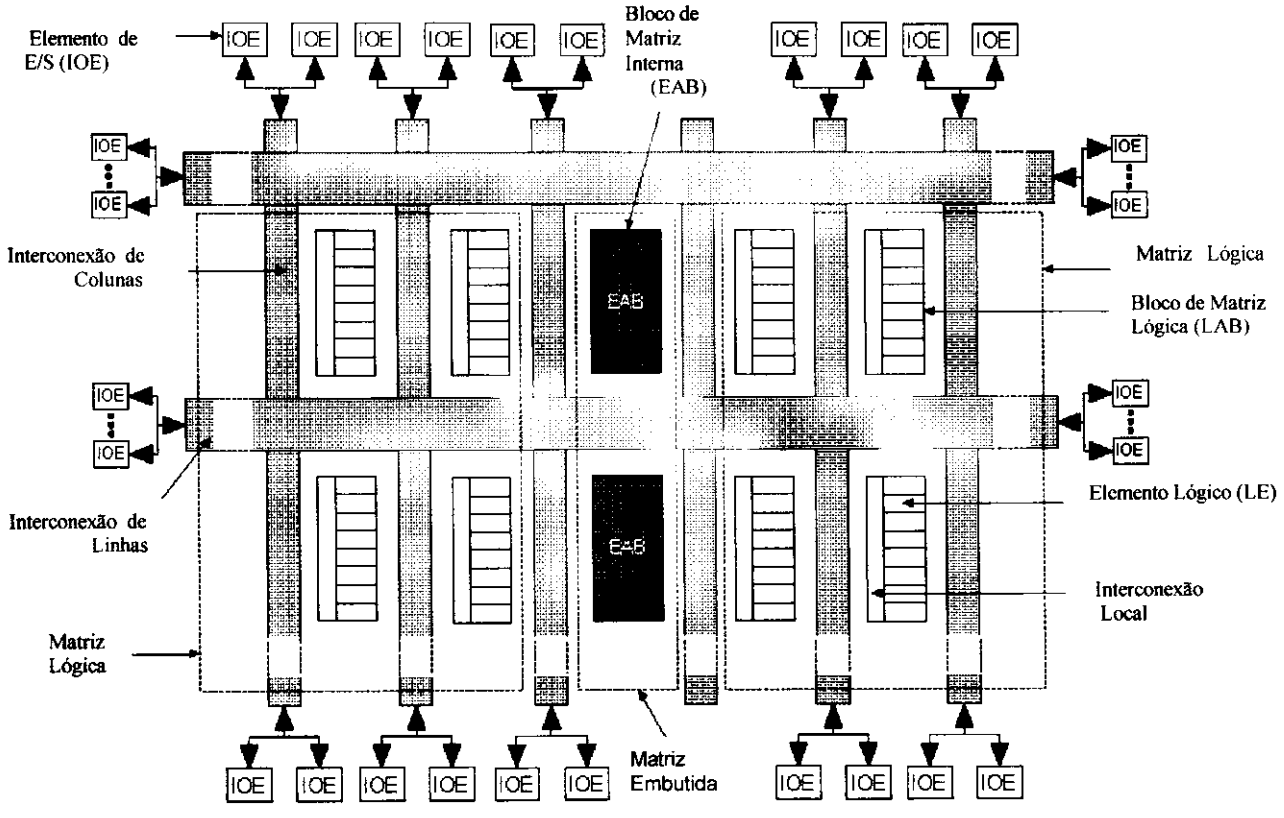
A matriz embutida consiste numa série de EABs (*Embedded Array Blocks*). Esses blocos quando implementando funções de memória, disponibilizam um total de 2048 bits, cada um podendo ser usado para criar RAM, ROM, ou funções de FIFO (*First In-First Out*). Quando implementando lógica, cada EAB pode contribuir com 100 à 600 portas para funções lógicas complexas, como multiplicadores, microcontroladores, máquinas de estado, e funções de DSP (*Digital Signal Processing*). Esses blocos podem ainda ser utilizados de maneira independente ou conjunta, para implementar maiores funções [8].

A matriz lógica consiste numa série de LABs (*Logic Array Blocks*). Cada LAB contém oito LEs (*Logic Elements*) e uma interconexão local. Um LE consiste em 4 entradas de uma tabela *look-up* , um flip-flop programável, e vias de sinal dedicado para funções de cascata e *carry*. Os oito LEs podem ser usados para criar blocos de lógica de médio porte, como contadores de 8 bits, decodificadores de endereço, ou

máquinas de estado. Os LABs podem ser usados em conjunto para criar blocos lógicos maiores. Cada LAB representa cerca de 96 portas lógicas utilizáveis pelo projetista [8].

Cada pino de entrada/saída é alimentado por um elemento de I/O (*In/Out*), chamado de IOE (*I/O Element*), localizado no fim de cada linha e coluna do *FastTrack Interconnect*® , que consiste numa série de vias que percorrem todo o comprimento e a largura do dispositivo. Cada dispositivo IOE contém um *buffer* bidirecional e um flip-flop que pode ser usado tanto como um registrador de saída quanto como de entrada para alimentar sinais de entrada, saída ou mesmo bidirecionais [8].

A figura III.4 mostra um diagrama de blocos da arquitetura dos dispositivos da família FLEX10K. Cada grupo de LEs combina-se em um LAB. Os LABs por sua vez são arranjados em linhas e colunas. Cada linha também contem um único EAB. Os LABs e os EABs são interconectados pelo *FastTrack Interconnect*. Os IOEs são localizados no fim de cada linha e coluna do *FastTrack Interconnect* [8].



III.4. Arquitetura dos dispositivos FLEX10K

IV. Projeto do ASIC Para Acionamento De Inversores Trifásicos: Concepção e Resultados Obtidos

O sistema montado no LIEC para a prototipagem de ASICs em FPGA é simples e consiste em:

→ Um computador PC K6II de 300 MHz, funcionando como hospedeiro da placa SPRIT, com o software MAX+PLUS II versão 8.3 instalado;

→ Uma placa SPRIT equipada com um FPGA da família FLEX10K de 20 mil portas (equivalentes);

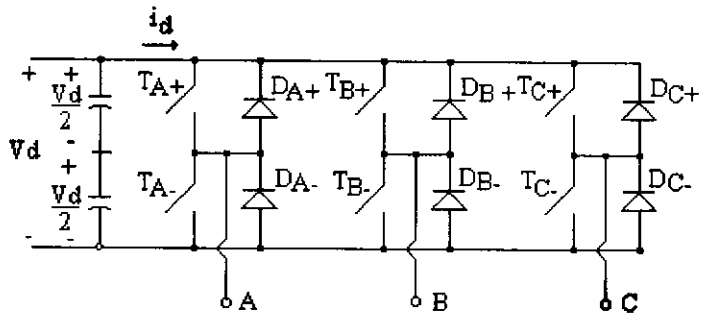
→ Uma chave de Hardware utilizada para o uso do software MAX+PLUS II.

Além da placa SPRIT com o dispositivo de 20 mil portas, existem outras com outros dispositivos de maior capacidade (40 mil e 70 mil portas). Existe também no LIEC um sistema da XILINX, porém o mesmo ainda não foi montado.

Com o sistema descrito anteriormente foi possível a prototipagem de um ASIC para a geração de sinais PWM (*Pulse Width Modulation*) para o acionamento de inversores trifásicos. Esse trabalho foi proposto em [9], passando por diversas etapas ([3], [10]) , até a sua prototipagem efetiva em FPGA. A seguir é dada uma rápida explanação sobre os conceitos que envolvem esse projeto e logo após os resultados experimentais (obtidos durante o nosso estágio no LIEC) são mostrados.

IV.1.1. INVERSORES E SINAIS DE CONTROLE PWM

Conhecidos como inversores, os conversores de potência dc → ac são compostos geralmente de dispositivos semicondutores de potência controlados por sinais eletrônicos, bem como de elementos armazenadores de energia, como indutores e capacitores. Junto com os retificadores, são os mais comuns conversores de potência existentes. A figura IV.1 mostra um inversor utilizado para uma carga trifásica.



VI.1. Inversor trifásico

Os pulsos de controle ou, como são mais conhecidos, as funções de chaveamento de um inversor trifásico, são basicamente gerados através de dois métodos: um, baseado em portadora, e outro, baseado na teoria dos vetores espaciais, conhecido como modulação vetorial [9].

Os controladores baseados em portadora podem ser monofásicos ou trifásicos e a implementação dos mesmos pode ser resumida, no caso monofásico, a um comparador de tensão cujas entradas são: um sinal de referência, chamado de modulante, e uma onda triangular, chamada de portadora. Esse processo é denominado de triangulação e, no caso trifásico, é realizado pela comparação de uma única portadora com três sinais modulantes defasados de 120° [9].

O outro método, típico para a aplicação em inversores trifásicos, não poderia ser chamado de modulação, já que no mesmo não há uma interação portadora-modulante. No entanto, o termo modulação vetorial é usado mesmo assim para denominá-lo.

Esse método se baseia na representação das tensões de saída do inversor através de vetores espaciais, ditos vetores de tensão, os quais ao serem projetados no plano complexo (α,β) permitem a sintetização de um vetor denominado de referência. Esse vetor possui localização e magnitude determinadas no plano (α,β), pelas referências trifásicas. Nesse método, o princípio da relação tensão-tempo, presente em qualquer método de geração de funções de chaveamento, está contido na idéia de intervalo de aplicação de um vetor tensão [9].

A geração de pulsos de controle para inversores trifásicos tem sido realizada de diversas formas, empregando diversos dispositivos para tal propósito, entre eles, circuitos eletrônicos analógicos com componentes discretos, microcontroladores, processadores digitais de sinais (DSPs- *Digital Signal Processors*), placas de controladores programáveis [9] e FPGAs [11] .

O emprego de FPGAs para a realização de estratégias de controle PWM apresenta vantagens com relação às demais, devido à rápida prototipagem, a simplicidade do hardware e dos softwares envolvidos no projeto, entre outras. A razão pela qual se optou pela implementação do ASIC proposto em FPGA foi exatamente a rápida prototipagem do circuito integrado concebido em [9].

IV.2. Técnica Empregada Na Geração Dos Pulsos

A técnica proposta em [9] e aplicada na concepção do ASIC, resumidamente, foi a seguinte:

→ Através de uma tabela de valores correspondentes a uma onda senoidal, são geradas três formas de onda defasadas de 120°;

→ Os três valores tomados de cada vez são comparados e com base nos mesmos são gerados vetores que indicam qual das fases é a menor, a maior e a de valor intermediário. São gerados também sinais que indicam quando há cruzamento com zero (c) ou cruzamento entre fases (p) ;

→ Os vetores e os sinais anteriores são responsáveis pela geração de sinais senoidais distorcidos ($U_1^d{}_{ref}$, $U_2^d{}_{ref}$ e $U_3^d{}_{ref}$) como podemos ver pelas seguintes equações:

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} U_{X\,ref}(t) \\ U_{Y\,ref}(t) \\ U_{Z\,ref}(t) \end{bmatrix} \quad \text{(IV.1)}$$

$U_{X\,ref}$, $U_{Y\,ref}$ e $U_{Z\,ref}$ são respectivamente a maior fase, a fase intermediária e de menor valor.

$$\begin{bmatrix} U_{1\,ref}^d(t) \\ U_{2\,ref}^d(t) \\ U_{3\,ref}^d(t) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & M_X - \overline{M}_X & M_Z - \overline{M}_Z \\ 1 & N_X - \overline{N}_X & N_Z - \overline{N}_Z \\ 1 & P_X - \overline{P}_X & P_Z - \overline{P}_Z \end{bmatrix} \begin{bmatrix} (1-\tau_1-\tau_2)(1-2\mu) \\ \tau_1 \\ \tau_2 \end{bmatrix} \quad \text{(IV.2)}$$

Os valores M_x, M_z, N_x, N_z, P_x, P_z só podem assumir os valores 0 ou 1 e são indicadores da ordem das tensões de fase. M se refere à primeira fase, N a segunda e P a terceira. Os índices indicam a ordem de grandeza, sendo X utilizado para indicar a maior fase e Z para indicar a menor. Por exemplo, se M_x é igual a 1, isso indica que a primeira fase é a de maior valor;

→ Geradas as referências distorcidas, as mesmas são comparadas com a portadora triangular para a geração dos sinais modulados em largura de pulso. Essa comparação é simples e tem como resultado 0 (se a referência é maior que a portadora) ou 1 (caso contrário) .

Maiores detalhes podem ser encontrados na própria tese [9].

IV.3. IMPLEMENTAÇÃO EM FPGA

O ASIC foi desenvolvido em vários módulos, nos quais são realizadas diversas funções. Essas funções foram divididas de forma a se cumprir algumas exigências do projeto de um ASIC. Entre essas exigências estão: a hierarquização, obtida com a divisão do circuito em sub-módulos, e a regularidade, obtida através da utilização de estruturas bem definidas do ponto de vista digital (memórias, por exemplo) [6].

A estrutura proposta para o ASIC foi baseada principalmente na idéia de ter-se o máximo de utilização das chamadas MegaFunctions® do ambiente MAX+PLUSII . MegaFunctions são códigos desenvolvidos pela ALTERA para a implementação de várias funções, como memória, multiplicação, etc. Essas MegaFunctions são geralmente realizadas nas EABs e por serem desenvolvidas pelo próprio fabricante, são implementadas facilmente nos dispositivos. O emprego de uma Megafunction é feito da mesma forma que o acionamento de uma estrutura de hierarquia mais baixa dentro de um projeto.

Para chegarmos à implementação da expressão dada em IV.2, precisamos de início construir, a partir de uma tabela de dados, três referências senoidais. Logo, precisamos de um elemento armazenador de dados, uma memória, para a partir dela

requisitar os dados de maneira ordenada, isto é, seguindo os defasamentos de 120° entre elas.

Em seguida é necessária a geração de sinais que indiquem a ordem de amplitude das tensões de fase, para que seja possível a determinação da equação IV.1 e dos coeficientes da equação IV.2. Essa geração é feita com base em comparações entre os valores obtidos da memória para as três referências senoidais.

Feitas essas comparações e realizadas as operações da equação IV.2, é necessária agora a comparação entre essas referências distorcidas e uma portadora de forma triangular. Essa portadora é gerada e comparada com as referências distorcidas, para que sejam gerados os pulsos de acionamento. Esses pulsos passam ainda por um tratamento de forma que sinais que comandam um mesmo braço do inversor não mudem de estado no mesmo instante, possibilitando assim um curto. Desse modo, um atraso entre as mudanças de estado dos sinais de saída e seus simétricos é inserido.

Os módulos utilizados na descrição do circuito são apresentados a seguir de maneira separada.

IV.3.1. MÓDULO ROM

O módulo ROM consiste numa memória ROM na qual são armazenadas 512 palavras de oito bits, equivalentes aos dois primeiros quadrantes de uma onda senoidal, e de um circuito combinacional responsável por uma decodificação de endereço, necessário para que seja gerada uma forma de onda senoidal completa partindo-se da metade dos pontos. Essa decodificação é muito simples e é baseada apenas no valor da palavra de endereço. Se ela for maior que o número de pontos presente na tabela, simplesmente o bit mais significativo da palavra de endereço (10 bits) é descartado e o dado correspondente é tomado em complemento de 2 (valor negativo).

Para implementação dessa estrutura de memória, foram utilizados recursos de memória do próprio dispositivo, pois o uso de uma ROM externa era desnecessário para a pequena quantidade de bits que era preciso armazenar-se. Foi utilizada uma MegaFunction, o que é, inclusive, recomendado pela própria ALTERA.

IV.3.2. O MÓDULO PORTADORA

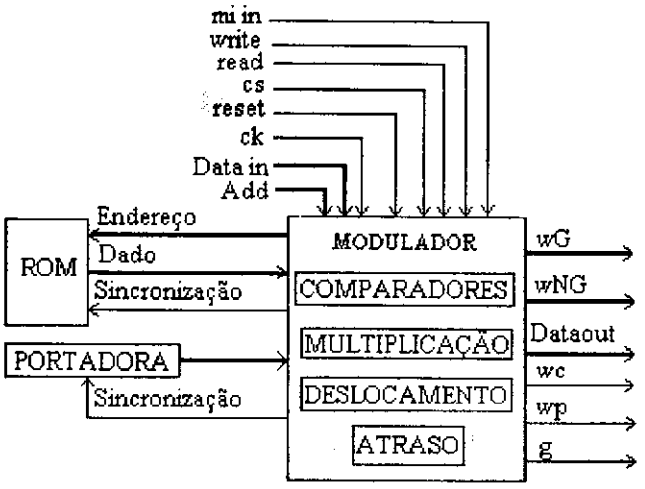
Este módulo gera os valores de uma forma de onda triangular que é utilizada para a comparação com os sinais modulantes, isto é, as referências senoidais distorcidas, sendo essa comparação a operação responsável pela geração dos sinais de saída PWM.

A sua estrutura consiste em contadores que são incrementados com um valor definido pelo usuário, chamado de razão de frequência, a cada pulso de clock.

IV.3.3. O MÓDULO MODULADOR

O módulo modulador é responsável pela geração dos pulsos de controle do inversor trifásico. Ele gerencia o acionamento de todos os módulos anteriores (rom e portadora) e é o de maior hierarquia. Nele são realizadas as equações IV.1 e IV.2, bem como as comparações referências distorcidas → portadora. O diagrama de blocos da figura IV.2 dá uma idéia de como o sistema está estruturado.

Dentro da estrutura implementada e testada, ainda não existem os sinais de controle read, write, cs, datain e add apenas por motivos de dificuldade na implementação do hardware necessário para o acionamento dos mesmos. Eles são responsáveis pela habilitação do circuito, e pela programação das variáveis necessárias a escolha da estratégia de modulação utilizada. A escolha da estratégia é feita escolhendo-se o valor da variável menu (que é carregada através dos pinos do barramento de entrada Data in) entre oito possíveis valores, de 0 até 7. Nas simulações e nos testes realizados optamos por definir previamente esse valor diretamente no código de descrição do circuito, de modo a testarmos o seu funcionamento. Não foram inseridos os atrasos nem a multiplicação das referências senoidais pelo índice de modulação.

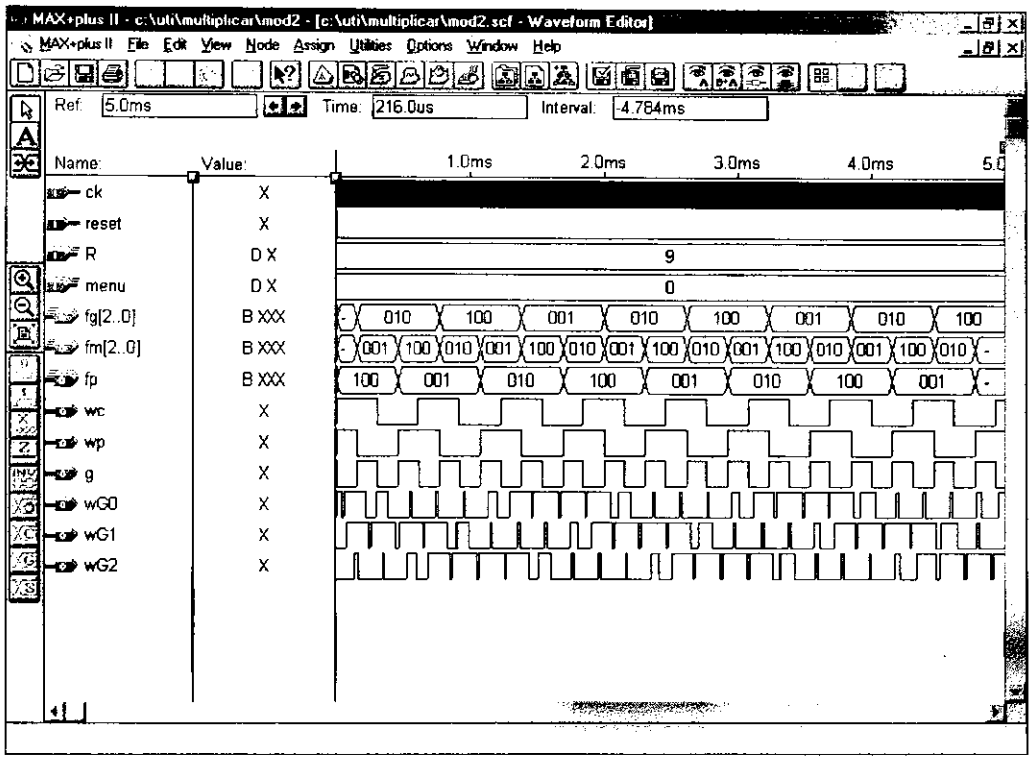


IV.2. Diagrama simplificado do CI

IV.4. Simulações e Resultados Experimentais

As simulações dos códigos de descrição do ASIC foram realizadas no MAX+PLUS II, comparadas com simulações realizadas no MATLAB (descrição em linguagem de alto nível) e em seguida com os resultados obtidos nas saídas da placa de programação do FPGA.

A figura seguinte mostra o ambiente de simulação do MAX+PLUS II, no qual se define as entradas (primeiros pinos) de maneira simples, utilizando-se a barra de ferramentas do mesmo. Pode-se entrar com sinais de clock, contadores ou mesmo valores fixos. Para o sinal de clock escolhemos uma freqüência de 5 MHz (maior freqüência de simulação possível), para o R escolhemos o valor 9 e o valor de entrada menu foi escolhido segundo a estratégia de modulação desejada em cada simulação. A figura também mostra os sinais sinais fg, fm, fp, p (wp), c (wc) e d (g) anteriormente descritos, bem como os sinais wG0, wG1 e wG2, que são os pulsos para o acionamento do inversor trifásico.



IV.3. Simulação geral no Max+PlusII para menu=0

A apresentação das simulações na forma anterior não é necessária uma vez que, exceto pelo valor de entrada menu e pelos sinais de saída modulados em largura (WGs), todos os outros se repetem. As formas de onda obtidas nos três níveis discutidos anteriormente (MAX+PLUSII, MATLAB e experimentais) para as sete primeiras estratégias de modulação são apresentadas no apêndice A. A oitava estratégia não pode ser programada no FPGA por necessitar de um sinal de entrada (miin) e ainda não haver uma interface segura para a aplicação de sinais na placa de programação do FPGA.

IV. Considerações Finais

O objetivo inicial de trabalhar com sistemas de prototipagem de ASICs em FPGAs foi alcançado, como se pode observar nesse relatório. Além disso, o contato com ferramentas avançadas de simulação e desenvolvimento de CIs digitais permitiu o amadurecimento de conceitos vistos durante a graduação nas disciplinas de eletrônica digital e sistemas digitais.

O trabalho desenvolvido no LIEC deverá ser base para o desenvolvimento de novos ASICs em FPGAs, a fim de que conceitos propostos em trabalhos acadêmicos sejam testados e validados, tal como ocorreu em [9].

A importância do domínio da técnica de prototipagem de ASICs em FPGAs está na dificuldade que se tem de enviar um CI para a fabricação, pelo menos a partir da nossa Universidade. Além da dificuldade econômica, existe o atraso que isso acarreta ao projeto. Enquanto que a programação do FPGA pode ser feita em segundos, a fabricação de um ASIC pode durar semanas ou meses, incluso o tempo gasto no transporte.

Comercialmente, existem diversos fatores que ainda restringem o uso geral de FPGAs, essas informações podem ser encontradas em [12], porém não são aplicáveis a nossa realidade, que, até o presente momento, é puramente acadêmica.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Hyde, Daniel C. . *"CSCI 320 Computer Architecture Handbook on Verilog HDL"*. Agosto de 1997.

[2] Landis, Dave . *"Programmable Logic and Application Specific Integrated Circuits"*. Pennsylvania State University, 1997. publicação na Internet.

[3] Catunda, Sebastian Y. C. "Implementação do Circuito Integrado para Modulação Vetorial em ASIC". Projeto e Pesquisa 97.3, COPELE- DEE-UFPB, 1997.

[4] P. K. Chand and S. Mourad, *"Digital Design Using Field Programmable Gate Array"*. Englewood Cliffs, NJ: Prentice-Hall, 1994.

[5] Moreira, C. S., *"Sistema de desenvolvimento reconfigurável aplicado ao processamento de sinais analógicos usando FPGA e Microcontrolador"*, proposta de dissertação de mestrado submetida a COPELE, setembro 1999.

[6] Morais, M. R. A., *Sistema de Prototipagem Rápida Dirigido ao Processamento de Imagens,* Dissertação de Mestrado, UFPB, 1998.

[7] MAX+PLUS II Help On-line, versão 9.01.

[8] FLEX 10K Data Sheet Version 3.13, Outubro de 1998.

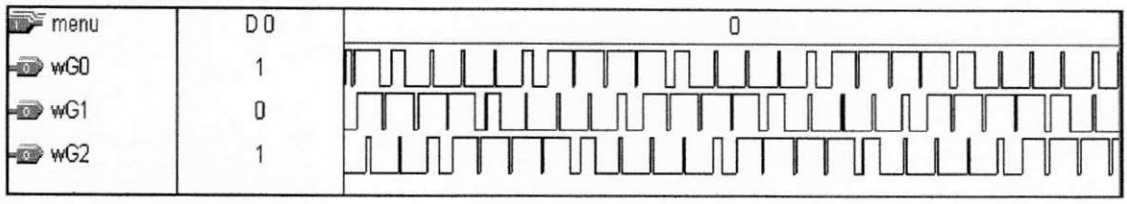
[9] R. N. C. Alves, *Análise e Implementação de Técnicas de Modulação em Largura de Pulsos para o Uso em Inversores Trifásicos,* **tese de doutorado em Engenharia Elétrica,** UFPB, defendida e aprovada em março de 1998.

[10] Silva, C. F. , *Desenvolvimento em FPGA de um Circuito Integrado para Modulação Vetorial* , Relatório Final de Iniciação Científica, Agosto de 1999.

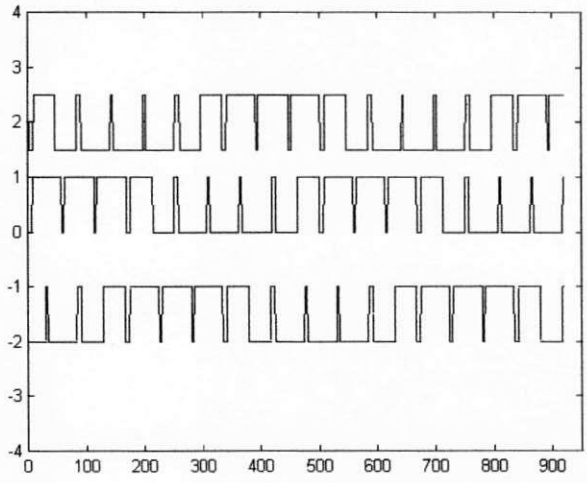
[11] Tzou, Y. and H. Hsu, *"FPGA Realization of Space-Vector PWM Control IC for Three-Phase PWM Inverters"*. IEEE Transactions on power electronics, Vol. 12, Nº 06, November 1997.

[12] Hopkin, Vince ."*Cost or Performance – ASICs Still Beat FPGAs*", Artigo publicado na Integrated System Design Magazine, 1998.

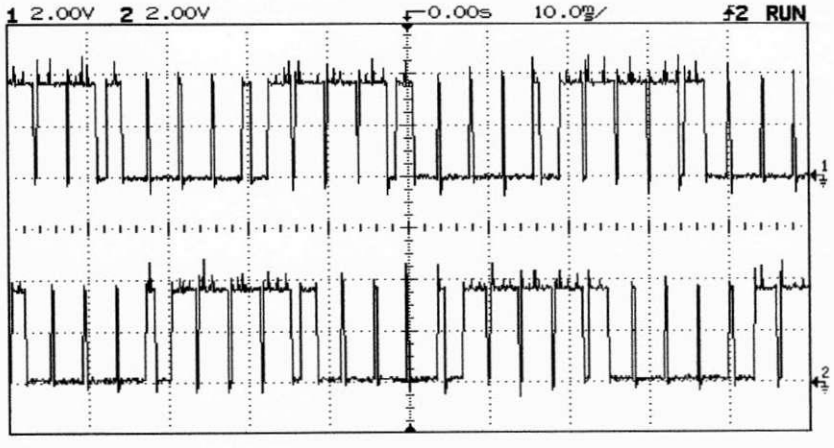
APÊNDICE A



(a) MAX+PLUSII

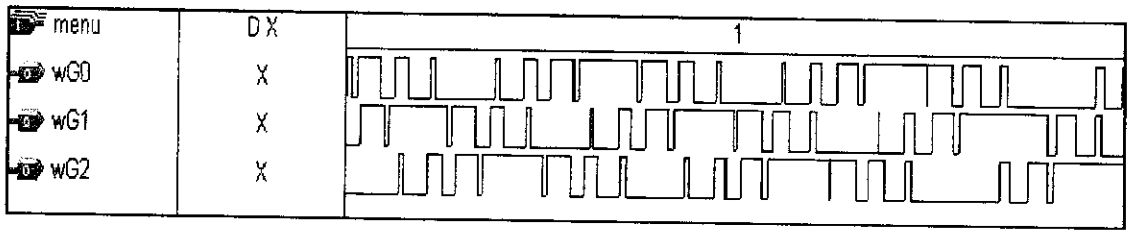


(b) MATLAB

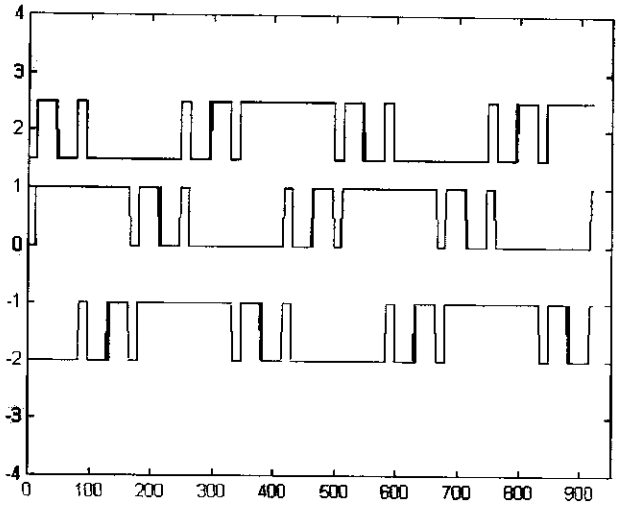


(c) Osciloscópio

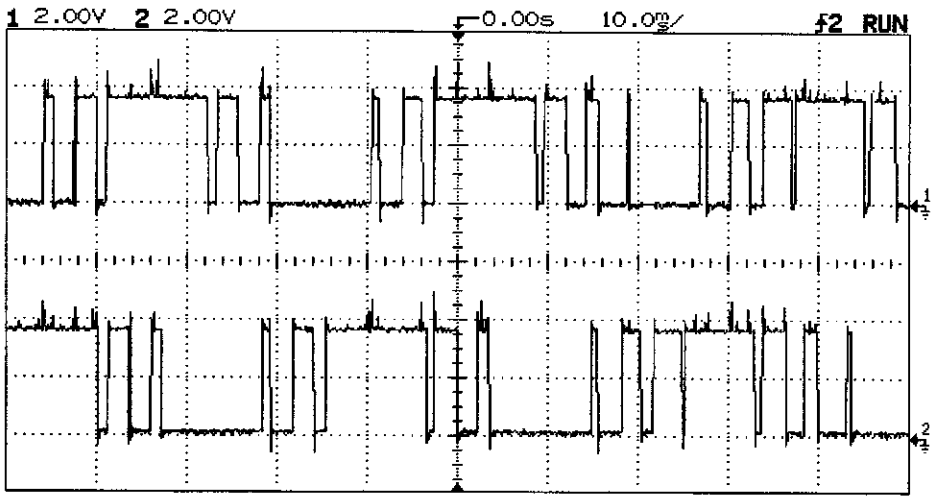
A.1. menu = 0, μ = 0,5



(a) MAX+PLUSII

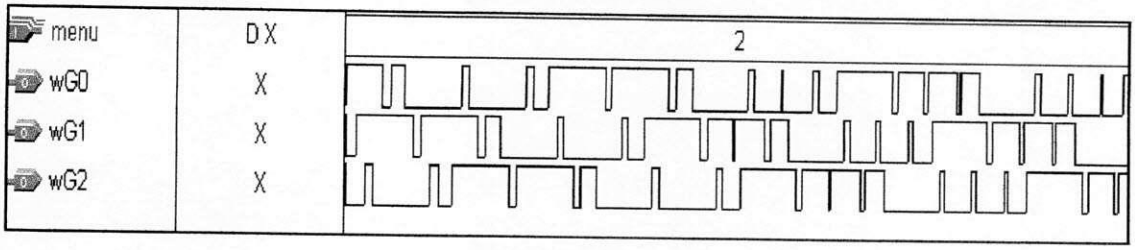


(b) MATLAB

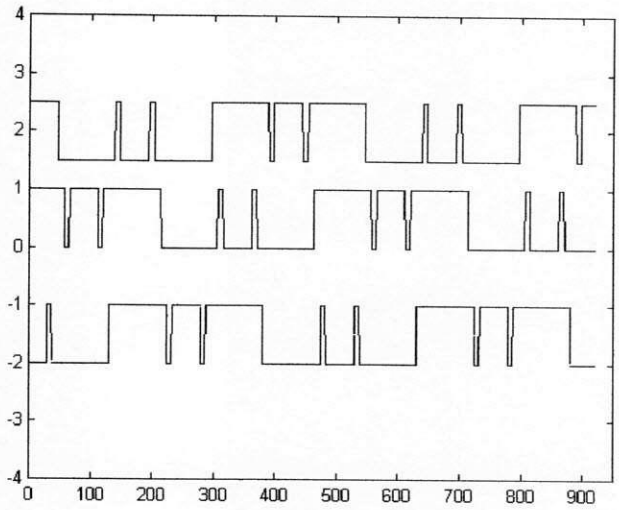


(c) Osciloscópio

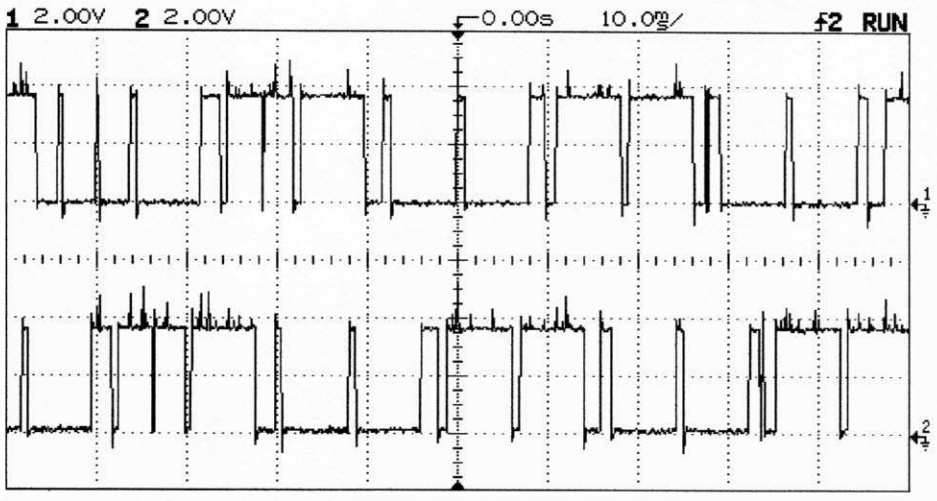
A.2. menu = 1, μ = c



(a) MAX+PLUSII

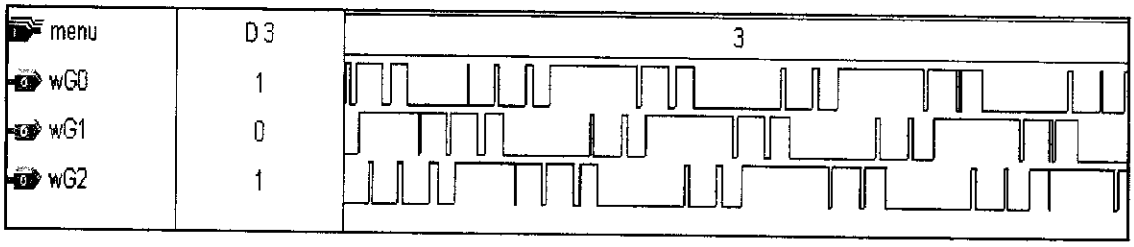


(b) MATLAB

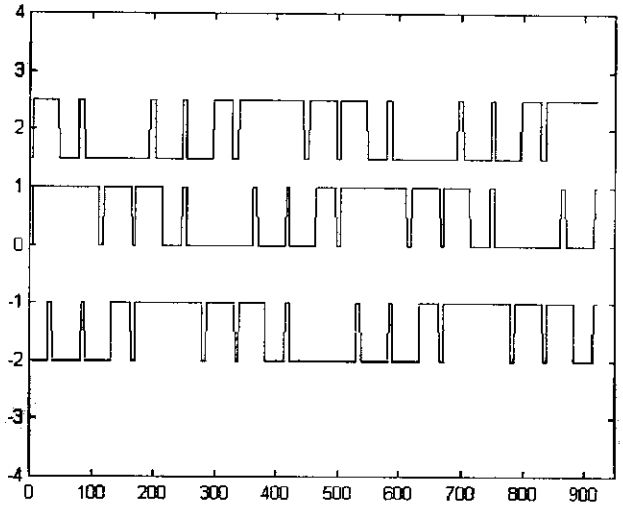


(c) Osciloscópio

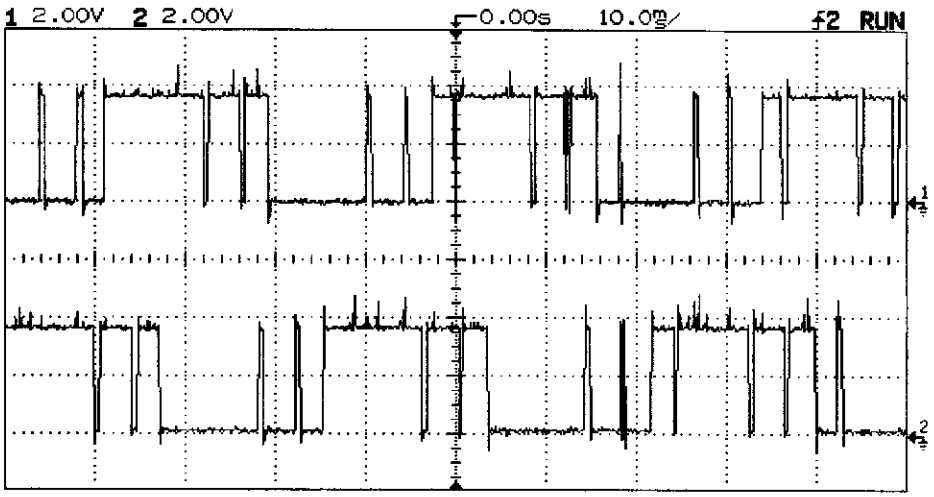
A.3. menu = 2, μ = ~ c



(a) MAX+PLUSII

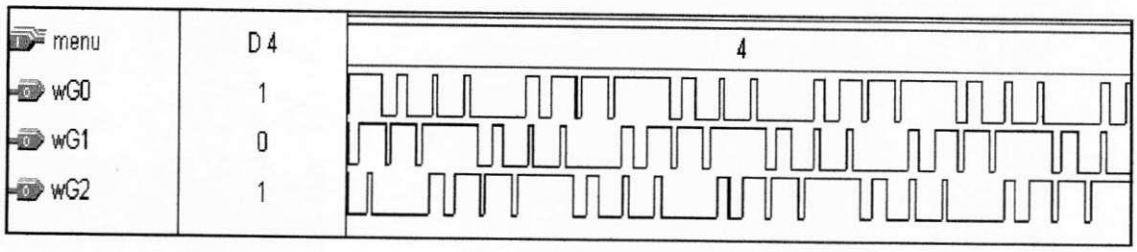


(b) MATLAB

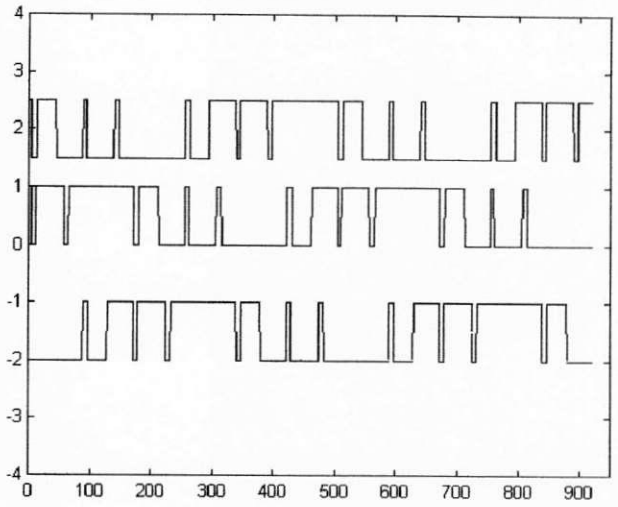


(c) Osciloscópio

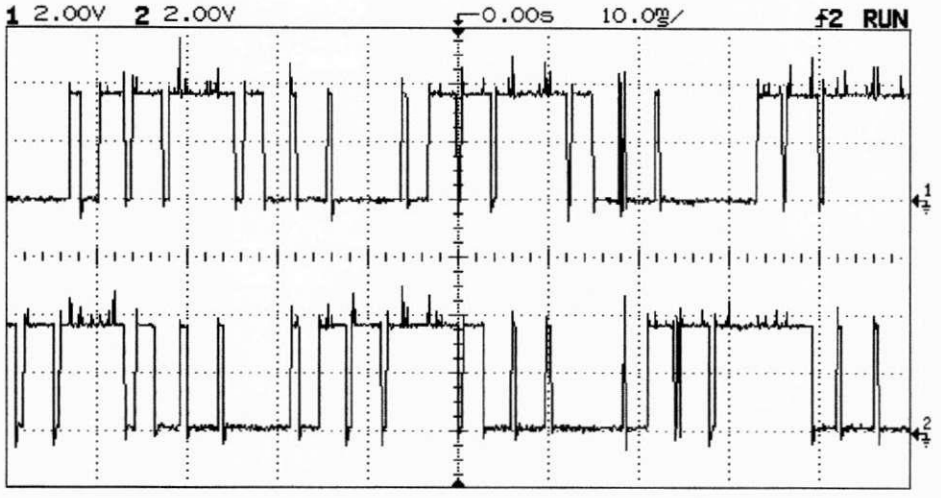
A.4. menu = 3, μ = p



(a) MAX+PLUSII

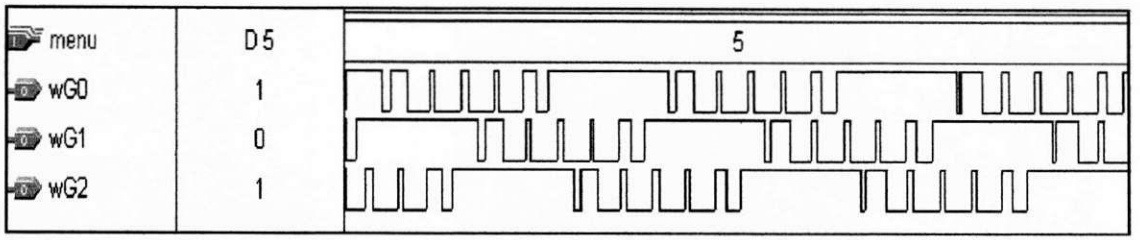


(b) MATLAB

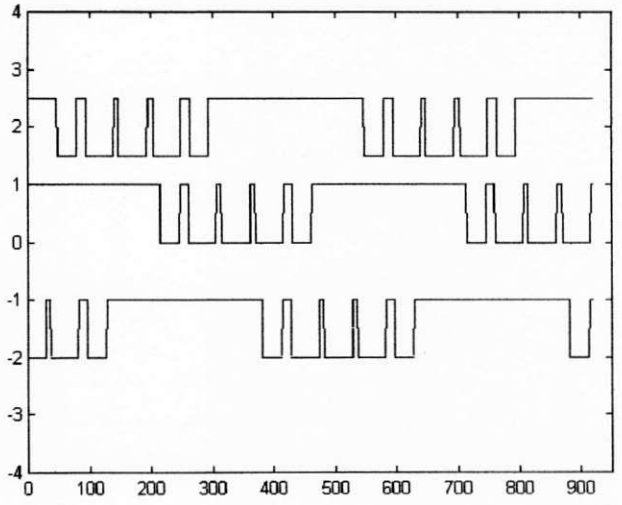


(c) Osciloscópio

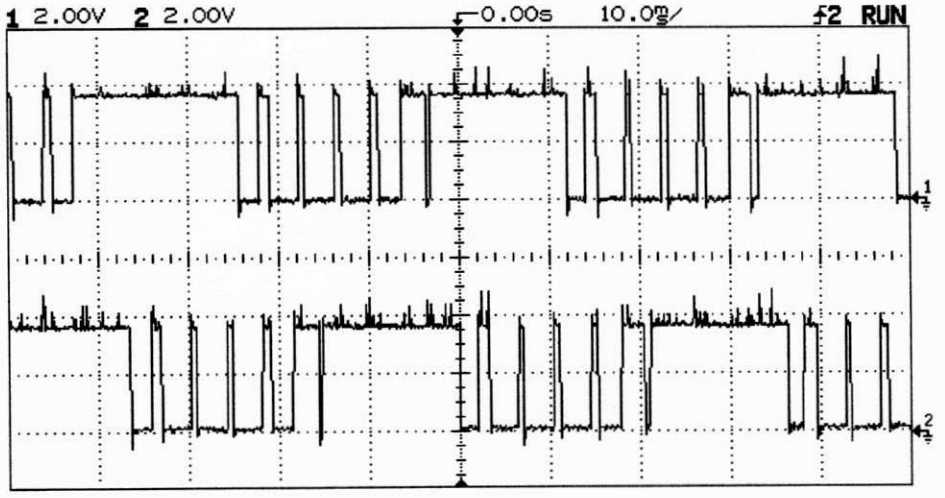
A.5. menu = 4 μ = ~ p



(a) MAX+PLUSII

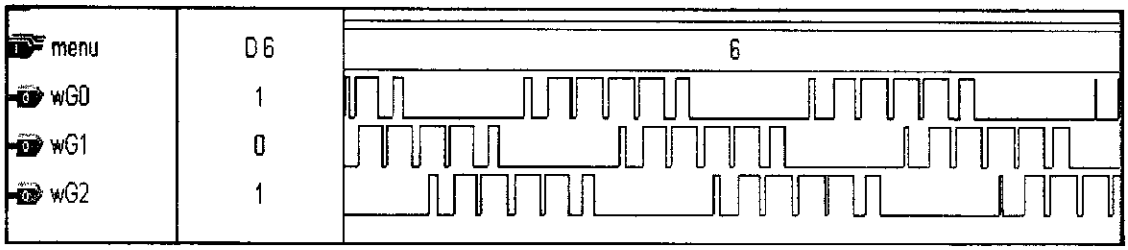


(b) MATLAB

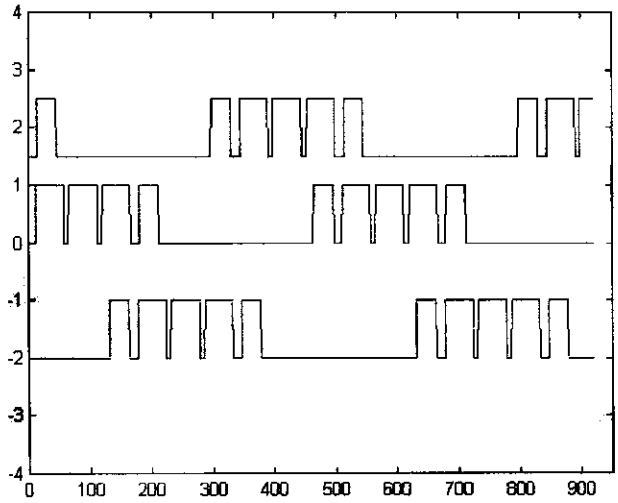


(c) Osciloscópio

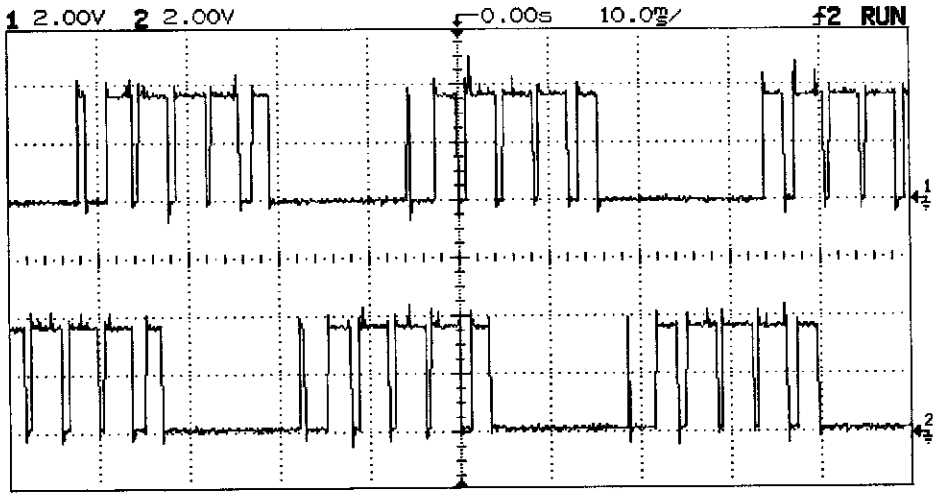
A.6. menu = 5, μ = 0



(a) MAX+PLUSII



(b) MATLAB



(c) Osciloscópio

A.7. menu = 6 , μ = 1