Projeto de Graduação

DEE DEPARTAMENTO DE ENGENHARIA ELÉTRICA

03 de dezembro de 2014.

DESENVOLVIMENTO DE INTERFACE GRÁFICA PARA ANÁLISE DO ESTADO DE POLARIZAÇÃO DA LUZ ATRAVÉS DE PLATAFORMA FPGA

Felipe Calliari

PUC RIO

www.ele.puc-rio.br

Projeto de Graduação

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

DESENVOLVIMENTO DE INTERFACE GRÁFICA PARA ANÁLISE DO ESTADO DE POLARIZAÇÃO DA LUZ ATRAVÉS DE PLATAFORMA FPGA

Aluno(s): Felipe Calliari

Orientador(es): Jean Pierre von der Weid

Trabalho apresentado com requisito parcial à conclusão do curso de Engenharia Elétrica na Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil.

Agradecimentos

À minha esposa Beatriz Vianna, aos meus avós, pais, irmãos, e a toda minha família que, com muito carinho e apoio, não mediram esforços para que eu chegasse até esta etapa de minha vida.

Agradeço também a todos os professores do curso, que foram tão importantes na minha vida acadêmica e no desenvolvimento desta monografia, e aos amigos e colegas, pelo incentivo e pelo apoio constantes.

Resumo

A polarização é uma propriedade de ondas eletromagnéticas. As fontes luminosas, geralmente, emitem ondas eletromagnéticas na forma de luz que não são polarizadas, isto é, os campos elétrico e magnético vibram em várias direções. Existem substâncias que deixam passar apenas parte da onda eletromagnética quando são atravessadas, desta forma é possível obter luz polarizada, ou seja, a luz se propaga apenas em um plano. Um exemplo de material polarizador é o Polaroid® que é um tipo de plástico que oferece resistência à passagem da luz em uma direção e menos resistência na outra direção.

A importância da polarização das ondas eletromagnéticas em fotônica é devido à dependência do estado de polarização que alguns equipamentos apresentam e que a polarização da luz pode variar dentro da fibra óptica, com isso se o receptor não estiver alinhado com a polarização que chegou apenas parte da potência poderá ser aproveitada. Existem outras aplicações que também mostram a importância da polarização da luz, alguns exemplos são medidas de propriedades de substâncias químicas, em antenas é possível enviar até dois sinais numa mesma frequência desde que as polarizações sejam ortogonais, entre outros.

O objetivo deste projeto é desenvolver uma estrutura de hardware digital baseada em FPGA que seja capaz de processar os dados provenientes de conversores analógico-digitais conectados às saídas de um polarímetro analógico convencional, e calcular os Parâmetros de Stokes para que seja representar o estado de polarização da luz graficamente em um monitor LCD através da Esfera de Poincaré.

Palavras-chave: FPGA; Estado de Polarização; Parâmetros de Stokes; Esfera de Poincaré

DEVELOPMENT OF A GRAPHICS INTERFACE FOR ANALYSIS OF THE STATE OF POLARIZATION OF LIGHT THROUGH FPGA PLATFORM

Abstract

Polarization is a property of electromagnetic waves. Generally the light sources emit electromagnetic waves in the form of light not polarized, that is, the electric and magnetic fields vibrates in all directions randomly. There are substances that let through only part of the electromagnetic waves when they are traversed, this way it is possible to obtain polarized light, i.e., light propagates in only one plane. An example of polarizing material is Polaroid® which is a type of plastic that offers resistance to the passage of light in one direction and less resistance in the other direction.

The importance of polarization of electromagnetic waves in photonics is due to the dependence of the state of polarization that some equipment present and that the polarization of light can vary within the optical fiber, thus if the receiver is not aligned with the polarization that came only part of the power can be harnessed. There are other applications which also show the importance of polarization of light, some examples are that the chemical properties can be measured by analysing of the polarization that came through, another example is that when we use antennas it is possible to send up to two signal on the same frequency since the polarizations are orthogonal, among others.

The objective of this project is to develop an FPGA-based digital hardware structure that is able to process data from analog to digital converters connected to the outputs of a conventional analog polarimeter, and calculate the Stokes parameters to later represent the state of polarization of light graphically on an LCD monitor in the Poincaré Sphere.

Keywords: FPGA; SOP; Stokes parameters; Poincaré sphere

Sumário

Lista de figuras

1. Introdução

Uma FPGA é um dispositivo programável composto de um número muito grande de Blocos Lógicos Programáveis, pequenas células que contém as estruturas básicas da eletrônica digital. Através da programação da malha de interconexões que conecta esses blocos, é possível criar qualquer tipo de circuito digital. A linguagem VHDL é uma linguagem específica para descrição de hardware que auxilia na criação, tanto na etapa de síntese quanto na de simulação, dos circuitos digitais desejados.

A polarização é uma propriedade de ondas eletromagnéticas, entre elas a luz, que pode ser utilizada para as mais diversas aplicações, como os televisores LCD, óculos com lentes Polaroid, etc. Além disso, é um fator de grande impacto nas transmissões utilizando fibras ópticas. Com a ajuda de um controle de polarização eficaz é possível reduzir os impactos da dispersão dos modos de polarização o que possibilita taxas de transmissão de dados maiores em fibras ópticas. Um dispositivo capaz de identificar o estado de polarização da luz (polarímetro) é uma ferramenta, portanto, de grande demanda em projetos envolvendo comunicações ópticas.

Aliando um polarímetro analógico convencional, conversores analógico-digitais, um hardware de processamento específico desenvolvido em FPGA e um monitor LCD, é possível criar uma estrutura autônoma (sem a conexão com um computador) de visualização de estados de polarização. Além de apresentar dimensões reduzidas, essa estrutura também elimina os tempos de transmissão entre o computador e a coleta de dados, o que a torna mais veloz em grande parte das aplicações. Dentre os circuitos possíveis de serem estruturados em uma FPGA, pode-se destacar, do ponto de vista desse projeto: a interface entre a FPGA e o monitor; a interface entre a FPGA e os conversores analógico-digitais; e uma unidade de processamento digital de sinal que conecte as duas interfaces.

2. Desenvolvimento

Inicialmente, iremos apresentar os tópicos que necessários para a implementação do projeto em uma FPGA, começando com uma apresentação do que é uma FPGA, da linguagem de descrição de hardware VHDL, do polarímetro analógico, dos conversores analógico-digitais, entre outros, e da motivação para o seu desenvolvimento e as vantagens que serão alcançadas com a utilização deste sistema.

a. FPGA

Uma FPGA (Field Programmable Gate Arrays) é um dispositivo programável composto de um grande número de Blocos Lógicos Programáveis, pequenas células que contém as estruturas básicas da eletrônica digital. Através da programação da malha de interconexões que conecta esses blocos, é possível criar qualquer tipo de circuito digital.

Para implementar alguma estrutura lógica em uma FPGA utilizamos algum tipo de Software que é capaz de gerar um arquivo de configurações que contém as conexões entre os blocos lógicos. Posteriormente, devemos enviar esse arquivo para a FPGA.

As FPGA's possuem dois modos de operação USB e ROM, no modo USB utilizamos a conexão USB para enviar o arquivo gerado pelo software e assim que desligamos energia da FPGA, esta volta ao seu estado original, isto é, todas as conexões entre os Blocos Lógicos Programáveis dentro da malha de interconexões não estão definidas. No modo ROM podemos enviar o arquivo gerado para a FPGA e esta salva as configurações em um chip de memória Flash ROM, e toda vez que ligamos a FPGA ela irá iniciar este arquivo da memória.

As primeiras FPGA's surgiram em meados da década de 80, a primeira FPGA comercialmente viável foi criada Xilinx Inc em 1985, o XC2064 tinha apenas 64 Blocos Lógicos Programáveis. Na década de 90 as FPGA's começaram a evoluir e ficar mais sofisticadas e logo começaram a ser utilizadas em projetos de telecomunicações e redes, já no final da década de 90, as FPGA's eram utilizadas, também, pela indústria automotiva e em aplicações industriais.

Na virada do século, as entidades educacionais começaram a integrar as FPGA's em seus cursos e os vendedores começaram a desenvolver placas de desenvolvimento. Essas placas de desenvolvimento vinham com diversos componentes integrados, leds, botões, portas USB, serial, etc.

Antigamente criar projetos com designs complexos requeriam muito tempo, e somente após a etapa de produção (que pode demorar até quatro meses) é que estes chips poderiam ser debugados. Com isto, eram gastos milhões para desenvolver chips com alguma complexidade. Algumas das vantagens da utilização de FPGA's são que as fases de design, prototipação e debug são muito rápidas e baratas, portanto quando enviamos o projeto para alguma empresa produzir nosso chip este já virá funcionando como esperado, pois já foi testado na FPGA. A placa utilizada neste projeto é uma XEM3005 da Opal Kelly, baseada em uma FPGA Xilinx Spartan-3E.

Figura 1 – XEM3005 da Opal Kelly.

b. VHDL – Linguagem de Descrição de Hardware

O desenvolvimento de módulos em FPGA's é feito através de linguagens de descrição de hardware. Uma HDL (Hardware Description Language) é uma linguagem usada para programar a estrutura e a operação de circuitos eletrônicos e circuitos digitais.

A linguagem de descrição de hardware nos permite criar uma descrição precisa e formal do circuito eletrônico a ser analisado. Além disso, com a utilização de uma HDL é possível realizar análises automatizadas para verificar erros de projeto, bem como é possível realizar simulações de circuitos eletrônicos.

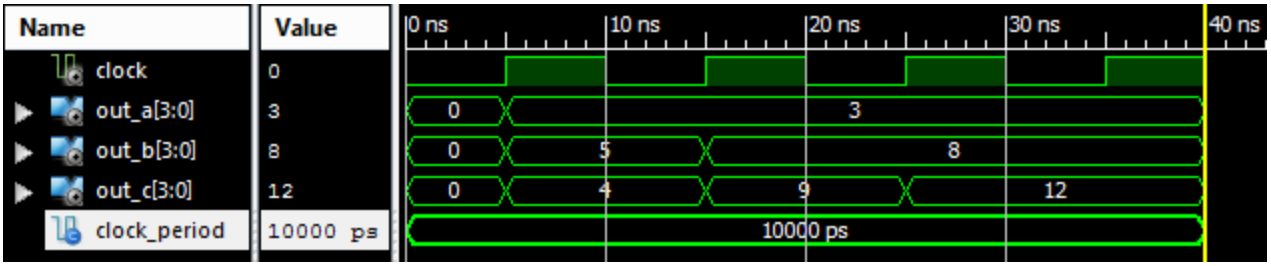

Figura 2 - Simulação do código VHDL citado acima.

As primeiras HDLs (Hardware Description Language) surgiram na década de 60. Uma das primeiras HDLs amplamente utilizadas foi o *ISP language*. No final de 1970, os primeiros PLDs (Dispositivos Lógicos Programáveis) se tornaram populares, embora fossem limitados em relação ao tamanho dos circuitos que poderiam ser implementados.

Posteriormente, em meados da década de 80, conforme novas tecnologias foram surgindo e a técnica de VLSI[1] foi se desenvolvendo, surgiu a necessidade de uma linguagem que fosse capaz de lidar com circuitos tão grandes, surgiu então o KARL. A linguagem KARL foi desenvolvida pela University of Kaiserslautern, em 1979. A primeira HDL moderna foi o Verilog, criada pela companhia Gateway Design Automation, em 1985.

Em 1987 a linguagem VHDL foi desenvolvida a pedido do Departamento de Defesa dos EUA (DARPA), com o objetivo de documentar e simular circuitos já implementados. O VHDL foi criado baseado na linguagem ADA e na experiência adquirida com o desenvolvimento do ISPS (sucessor da linguagem ISP). Atualmente, a linguagem VHDL é padronizada pelo IEEE.

Nesta época o processo de criação de circuitos integrados (VSLI) já havia sido desenvolvido e engenheiros criavam circuitos integrados através de esquemáticos e, posteriormente, realizavam simulações com auxílio das linguagens Verilog e VHDL.

As linguagens HDLs ganharam ainda mais popularidade após a introdução da etapa de síntese lógica (logic synthesis) para HDL. Isto é, transformar o código HDL para RTL (Register Transfer Level), com o objetivo de refinar o código RTL no nível de netlist e de executar etapas de verificação de erros lógicos, para em seguida transformar esse código RTL em uma implementação de design em termos de portas lógicas.

Ou seja, a linguagem HDL poderia ser traduzida para uma linguagem de fácil entendimento para os projetistas por meio de uma representação esquemática. Algumas das portas lógicas mais comuns e mais utilizadas são as portas lógicas AND, OR e NOT. As portas lógicas são constituídas basicamente de transistores, daí o nome TTL (Transistor-Transistor Logic). Os circuitos das portas lógicas citadas acima estão representados abaixo.

[1] VSLI – Very-large-scale integration é um processo de criação de circuitos integrados através da combinação de milhares de transistores em um único chip. O VLSI surgiu na década de 70, antes do surgimento desta tecnologia, os circuitos integrados tinham funções limitadas e podiam ser implementados com poucos transistores, com isso era necessário utilizar vários CIs, com o VLSI foi possível integrar vários CIs em um único chip.

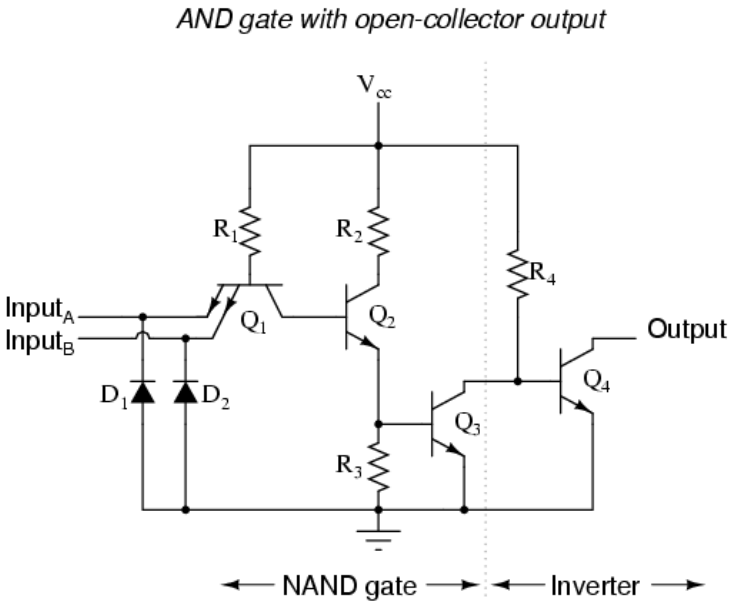

Figura 3 - Porta lógica AND.
Fonte: site AllAboutCircuits

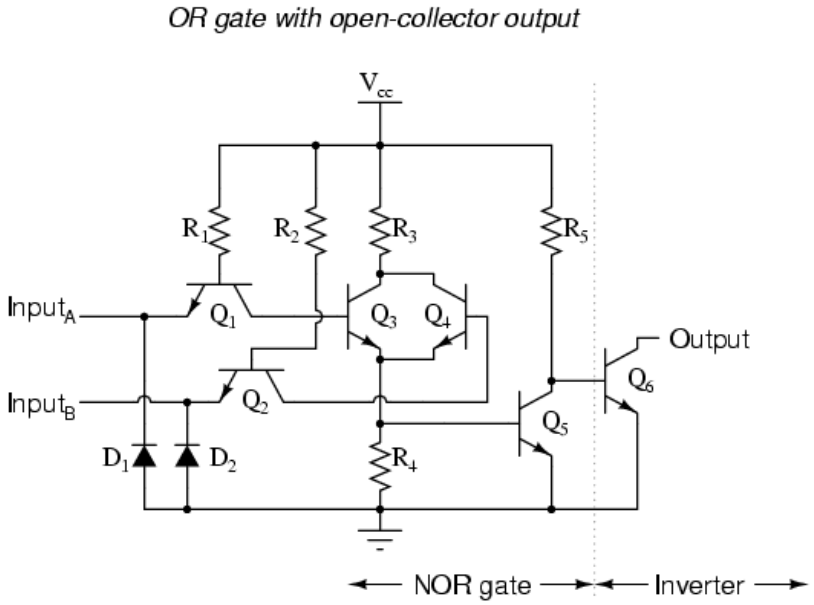

Figura 4 - Porta lógica OR.
Fonte: site AllAboutCircuits

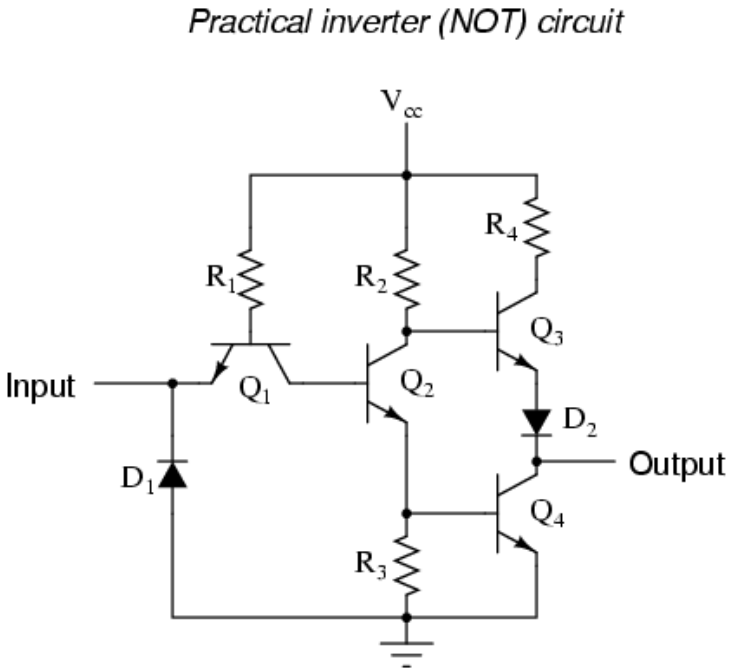

Figura 5 - Porta lógica NOT.
Fonte: site AllAboutCircuits

Um dos problemas encontrados na linguagem VHDL é que escrever códigos RTL sintetizáveis requerem prática e disciplina por parte do designer. Outro fato importante de se citar é que se fossemos comparar a implementação de um circuito grande por um engenheiro qualificado utilizando métodos convencionais (layout esquemático de circuitos) e outro utilizando linguagem VHDL, provavelmente o circuito implementado pelo método tradicional seria mais rápido.

Porém devemos levar em consideração que já existem várias otimizações embutidas nos sintetizadores de VHDL e que, utilizando-se "boas práticas" de programação, é possível criar circuitos em VHDL com performance comparável àquela dos circuitos implementados por engenheiros utilizando métodos convencionais.

Com o passar dos anos, o VHDL e o Verilog se tornaram as HDLs dominantes na indústria de eletrônicos, porém ambas linguagens possuem as mesmas limitações que são: nenhuma das duas linguagens é capaz de descrever circuitos analógicos ou mistos (com sinais analógicos e digitais). Para resolver estas limitações existem HDLs especializadas em circuitos analógicos, por exemplo a HDL Confluence que foi descontinuada há alguns anos.

> While there are already a plethora of design languages available, Hawkins feels he has something unique. Confluence, he says, provides a simple and clean way to describe extremely complex systems in just a few lines of code, with much more flexibility than today's HDLs. Confluence is then compiled into VHDL, Verilog, Phython, or C. (GOERING, Richard. Engineer creates HDL generation language)[2].

A versão inicial do padrão VHDL foi criada em 1987 e é definida pelo IEEE pela numeração 1076. A versão do padrão VHDL utilizado neste projeto é definida pelo documento IEEE 1076-1993. Nesta versão algumas das modificações incluem: uma sintaxe mais consistente, permitiu maior flexibilidade na atribuição de nomes, permitiu a utilização dos caracteres imprimíveis do padrão ISO-8859-1, a adição do padrão IEEE 1164, entre outros.

[2] Disponível em: <http://www.eetimes.com/document.asp?doc_id=1217201>. Acesso em: 2 de novembro de 2014

O VHDL é usado para descrever circuitos lógicos em forma de textos e. após passar pelas etapas de síntese e simulação, onde são verificados e corrigidos erros de prototipação, é possível utilizar ferramentas de síntese para traduzir o design em hardware real. Uma vez que um bloco é criado, este pode ser reutilizado em outros projetos sem a necessidade de modificações.

O código abaixo implementa a simulação mostrada na Figura 2.

```
-- isto é um comentário em VHDL
-- é necessário importar as bibliotecas que iremos utilizar
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
-- esta é a entidade que iremos descrever, e é fácil perceber
-- que ela possui uma porta entrada e três portas de saída
entity TEST is
   port (
      CLOCK : IN STD_LOGIC;
      OUT_A : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
      OUT_B : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
      OUT_C : OUT STD_LOGIC_VECTOR(3 DOWNTO 0)
   );
end TEST;
-- aqui serão declaradas variáveis e sub-blocos
architecture Behavioral of TEST is

   signal A : STD_LOGIC_VECTOR(3 DOWNTO 0) := (others => '0');
   signal B : STD_LOGIC_VECTOR(3 DOWNTO 0) := (others => '0');
   signal C : STD_LOGIC_VECTOR(3 DOWNTO 0) := (others => '0');
-- aqui descrevemos o comportamento
begin

   process(CLOCK)
   begin
      if (CLOCK'event and CLOCK = '1') then
         A <= "0011";
         B <= A + "0101";
         C <= B + "0100";
      end if;
   end process;

   OUT_A <= A;
   OUT_B <= B;
   OUT_C <= C;

end Behavioral;
```

c. Polarização da Luz

A polarização é uma das propriedades de ondas eletromagnéticas, assim como a cor, que varia com o comprimento de onda (λ), e a intensidade luminosa (variação de potência luminosa). A polarização pode ser utilizada para as mais diversas aplicações, como os televisores com tecnologia 3D, óculos com lentes Polaroid, entre outros. Um exemplo são as fotos apresentadas abaixo.

Figura 6 - Exemplo de polarização da luz em um monitor LED.

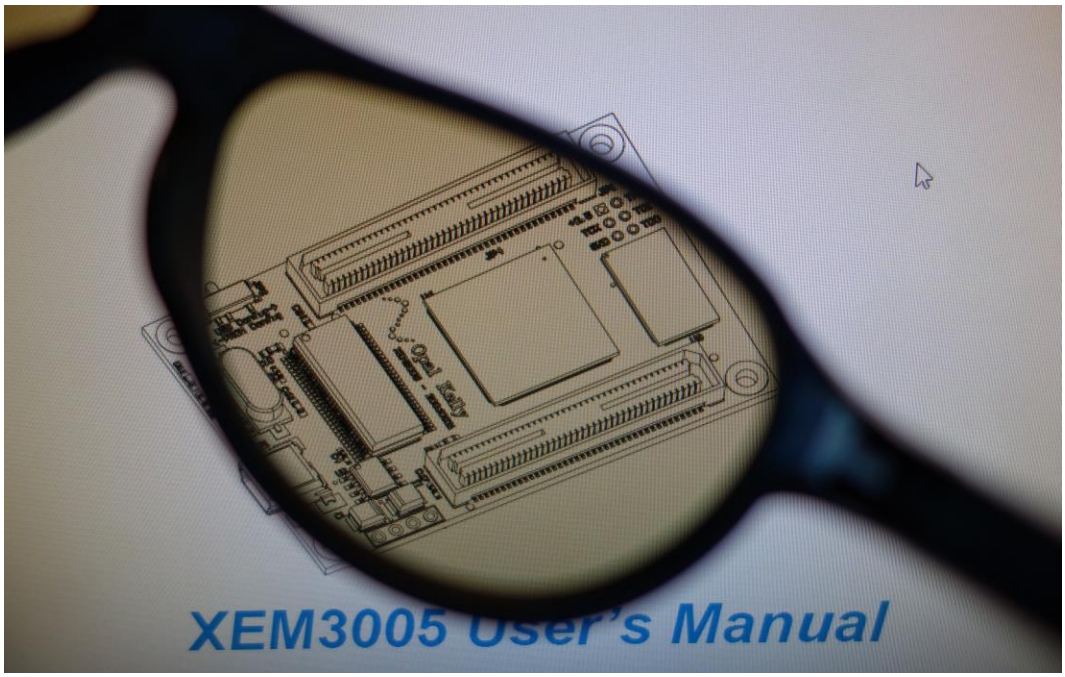

Figura 7 - Exemplo de polarização da luz em um monitor LED.

Um fato importante de ser notado é que se fosse utilizado um monitor de CRT (Catodic Ray Tube), no exemplo acima, não seria possível ver diferença alguma ao girar a lente. Isto ocorre devido ao modo como a luz é gerada neste tipo de monitores. O princípio de funcionamento de um monitor CRT é que utilizando-se um canhão de elétrons, na parte de trás do tubo, e uma tela fosforescente, na parte da frente, a tela emita luz quando atingia pelos elétrons (efeito conhecido como luminescência). Como o monitor de CRT emite luz despolarizada, apenas a luz com a mesma polarização da lente será capaz se atravessá-la. Contudo, em média 50% da luz atravessa a lente polarizada.

A polarização é um parâmetro de grande importância em sistemas de transmissão de dados, como é o caso de fibras ópticas, pois grande parte dos equipamentos utilizados possuem um eixo de polarização bem definido. Ou seja, variando o estado de polarização é possível maximizar ou minimizar a potência entregue a determinados equipamentos.

Nosso interesse é maximizar a potência entregue. Portanto, é muito importante conhecermos o estado de polarização da luz. Com a ajuda de um controle de polarização eficaz, é possível reduzir os impactos da dispersão dos modos de polarização, o que possibilita taxas de transmissão de dados maiores em fibras ópticas.

A polarização de uma onda eletromagnética é, por definição, o desenho traçado pelo campo elétrico no plano perpendicular à direção de propagação. Toda onda eletromagnética é composta por dois campos ortogonais, elétrico e magnético, e se desloca no sentido definido pelo vetor de Poynting, $\vec{S} = \vec{E} \times \vec{H}$. A Figura 8 mostra o deslocamento de uma onda eletromagnética em função do tempo.

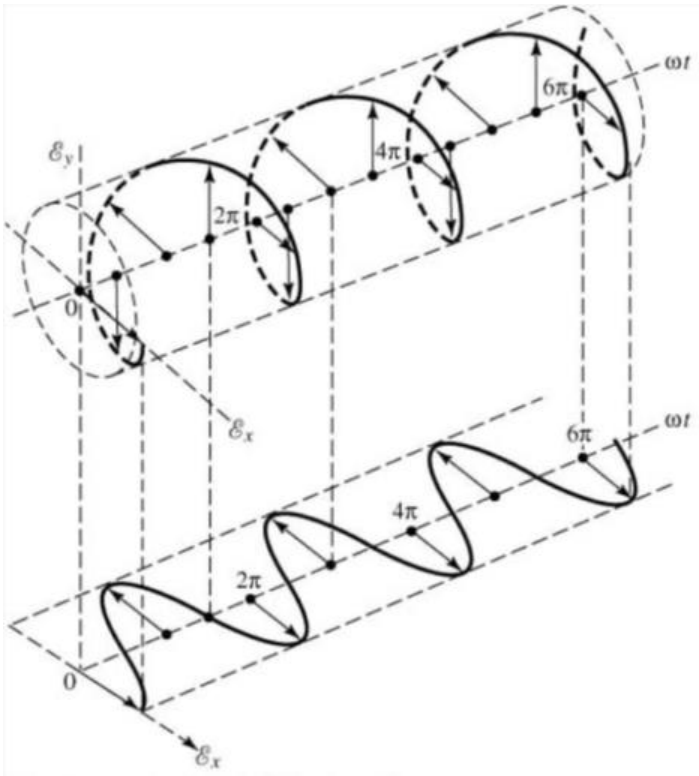

Figura 8 - Representação de uma onda eletromagnética se deslocando no tempo.
Fonte: Teoria de Antenas, Balanis[3].

A expressão do Vetor Campo Elétrico pode ser definida como:

$$\vec{E} = E_{xo} \cdot \cos(kz + wt + \phi_x) + E_{yo} \cdot \cos(wt + kz + \phi_y) \quad (1)$$

[3] Esta figura é protegida por direitos autorais (Copyright © 2005 por Constantine A. Balanis), e não pode ser modificada, copiada ou reproduzida de qualquer forma sem consentimento por escrito do autor, Constantine A. Balanis.

A polarização pode ser: linear, circular ou elíptica.

A. Polarização Linear – Ocorre quando a diferença de fase entre as duas componentes for um múltiplo de pi, como mostrado na equação 2:

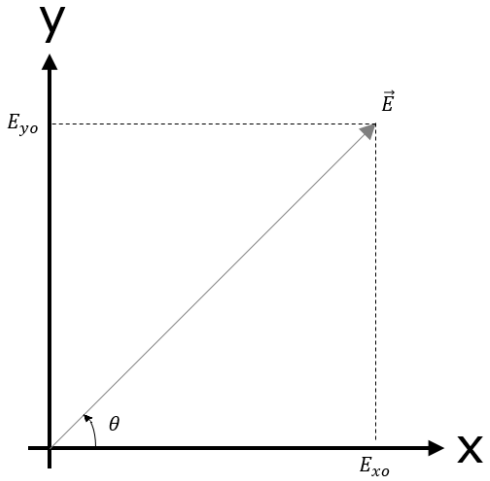

Figura 9 - Representação da Polarização Linear.

$$\Delta\phi = \phi_y - \phi_x = n\pi, \quad n \in \mathbb{Z} \tag{2}$$

Na polarização linear E_{xo} e E_{yo} podem assumir quaisquer valores desde que obedeçam à equação 2. Na Figura 9, podemos perceber que as componentes E_{xo} e E_{yo} possuem mesmo módulo, portanto:

$$\theta = \tan^{-1}\left(\frac{E_{yo}}{E_{xo}}\right) = 45° \tag{3}$$

B. Polarização Circular – Ocorre quando as magnitudes das duas componentes forem iguais e a diferença de fase entre elas for múltiplo de $\pi/2$.

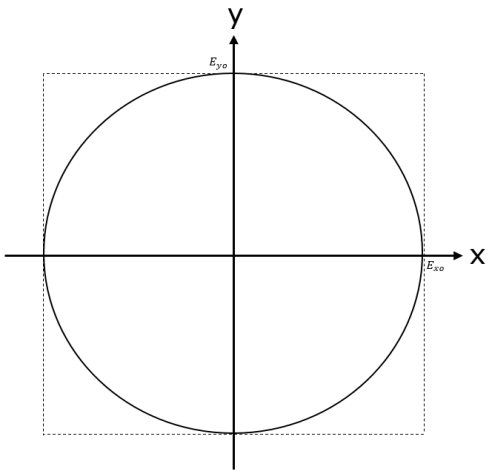

Figura 10 - Representação da Polarização Circular.

$$|E_{xo}| = |E_{yo}| \tag{4}$$

$$\Delta\phi = \phi_y - \phi_x = \begin{cases} +\left(\frac{1}{2} + 2n\right)\pi, & n \in \mathbb{Z} \quad \rightarrow Circular\ \text{à}\ direita\ (\circlearrowright). \\ -\left(\frac{1}{2} + 2n\right)\pi, & n \in \mathbb{Z} \quad \rightarrow Circular\ \text{à}\ esquerda\ (\circlearrowleft). \end{cases} \quad 5$$

C. Polarização Elíptica – Ocorre quando a diferença de fase for um múltiplo ímpar de $\pi/2$ e as magnitudes forem diferentes, **ou** quando a diferença de fase não for um múltiplo de $\pi/2$ e as magnitudes podem assumir quaisquer valores, podem até mesmo ser iguais.

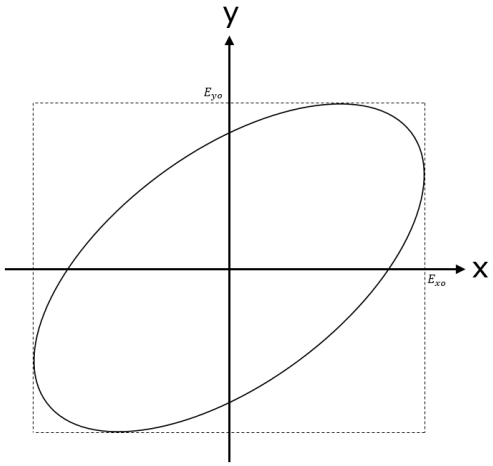

Figura 11 - Representação da Polarização Elíptica

Condições:

a) $$|E_{xo}| \neq |E_{yo}| \quad 6$$

$$\Delta\phi = \phi_y - \phi_x = \begin{cases} +\left(\frac{1}{2} + 2n\right)\pi, & n \in \mathbb{Z} \quad \rightarrow Elíptica\ \text{à}\ direita\ (\circlearrowright). \\ -\left(\frac{1}{2} + 2n\right)\pi, & n \in \mathbb{Z} \quad \rightarrow Elíptica\ \text{à}\ esquerda\ (\circlearrowleft). \end{cases} \quad 7$$

Podemos perceber que são as mesmas condições da polarização circular, com exceção da equação 6.

b) $$\Delta\phi = \phi_y - \phi_x \neq \begin{cases} +\frac{n}{2}\pi, & n \in \mathbb{Z}^+ \quad \rightarrow Elíptica\ \text{à}\ direita\ (\circlearrowright). \\ -\frac{n}{2}\pi, & n \in \mathbb{Z}^+ \quad \rightarrow Elíptica\ \text{à}\ esquerda\ (\circlearrowleft). \end{cases} \quad 8$$

Podemos perceber que as magnitudes podem assumir quaisquer valores, desde que a condição da equação 8 seja atendida.

Caso essa condição não seja atendida e $\Delta\phi$ for múltiplo de π, voltamos às condições de polarização linear. Ou seja, a polarização elíptica é o caso geral, e as polarizações linear e circular são formas derivadas da polarização elíptica.

d. Parâmetros de Stokes e Esfera de Poincaré

Uma forma alternativa de se representar o vetor campo elétrico, equação 1, é através do vetor de Jones:

$$\vec{E} = \begin{bmatrix} E_{xo}\, e^{j\phi_x} \\ E_{yo}\, e^{j\phi_y} \end{bmatrix} e^{j(kz-wt)} \qquad 9$$

Uma forma de se obter a intensidade de campo quando utilizamos a notação de Jones é calculando o quadrado do seu módulo, isto é, utilizando-se a equação 10.

$$I = {E_{xo}}^2 + {E_{yo}}^2 \qquad 10$$

Outra forma de representação do vetor de Jones é através de sua forma normalizada, pois não é necessário saber a intensidade do campo na determinação do estado de polarização. As equações se resumem a:

$$\vec{E} = \begin{bmatrix} \cos(\chi) \\ \text{sen}(\chi)\, e^{j\delta} \end{bmatrix} \qquad 11$$

Onde:

$$\delta = \phi_y - \phi_x, \qquad 0 \le \delta \le 2\pi \qquad 12$$

$$\chi = \tan^{-1}\left(\frac{E_{yo}}{E_{xo}}\right), \qquad 0 \le \chi \le \frac{\pi}{2} \qquad 13$$

Podemos definir o vetor de Jones para as polarizações: linear e circular da seguinte forma:

a) Polarização linear

Horizontal

$$\begin{cases} E_x = E_{xo}\cos(wt + \phi_x) \\ E_y = 0 \end{cases} \therefore \quad \chi = 0° \rightarrow |H> = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad 14$$

Vertical

$$\begin{cases} E_x = 0 \\ E_y = E_{yo}\cos(wt + \phi_y) \end{cases} \therefore \quad \chi = 90° \rightarrow |V> = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad 15$$

Diagonal ($\theta = 45°$)

$$\begin{cases} E_x = E_0\cos(wt + \phi) \\ E_y = E_0\cos(wt + \phi) \end{cases} \therefore \begin{cases} \chi = 45° \\ \delta = 0° \end{cases} \rightarrow |D> = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad 16$$

Anti-diagonal ($\theta = -45°$)

$$\begin{cases} E_x = E_0\cos(wt) \\ E_y = E_0\cos(wt + \pi) \end{cases} \therefore \begin{cases} \chi = 45° \\ \delta = 180° \end{cases} \rightarrow |A> = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix} \qquad 17$$

b) Polarização circular

Direita

$$\begin{cases} E_x = E_0\cos(wt + \phi_x) \\ E_y = E_0\cos(wt + \phi_y) \end{cases} \therefore \begin{cases} \chi = 45° \\ \delta = 90° \end{cases} \rightarrow |A> = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ i \end{bmatrix} \qquad 18$$

Esquerda

$$\begin{cases} E_x = E_0 \cos(wt + \phi_x) \\ E_y = E_0 \cos(wt + \phi_y) \end{cases} \therefore \begin{cases} \chi = 45° \\ \delta = -90° \end{cases} \rightarrow |A> = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -i \end{bmatrix} \quad 19$$

Um dos problemas da forma de vetorial de Jones é que ela só é aplicável à luz totalmente polarizada, e frequentemente nos deparamos com casos em que a luz está parcialmente polarizada, daí surgiu a necessidade uma forma de representação matemática em que fosse possível caracterizar tanto luz polarizada quanto não polarizada.

Os parâmetros de Stokes são um conjunto de valores que descrevem o estado de polarização de uma onda eletromagnética, e foram definidos por George Gabriel Stokes em 1852, como uma forma conveniente de descrever a polarização de uma onda eletromagnética incoerente ou parcialmente polarizada em termos de intensidade, grau de polarização e diferença de fase. Além disso, o vetor de Jones era representado por números complexos, já os parâmetros de Stokes são representados por números reais.

A notação utilizada para representar os parâmetros de Stokes também é sob forma vetorial, sendo um vetor coluna com 4 elementos:

$$S = \begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{bmatrix} = \begin{bmatrix} I_x + I_y \\ I_x - I_y \\ I_{45°} - I_{-45°} \\ I_{CE} - I_{CD} \end{bmatrix} \quad 20$$

Sendo I_x, I_y, $I_{45°}$ e $I_{-45°}$ as intensidades de campo linearmente polarizado nos eixos x, y, $45°$ e $-45°$, respectivamente. E I_{CE} e I_{CD}, as intensidades de campo circularmente polarizado à esquerda e à direta, respectivamente.

Portanto o estado de polarização da luz pode ser descrito pelos parâmetros de Stokes. Sendo que o parâmetro S_0 representa a intensidade total de luz, ou seja, luz polarizada e não polarizada. Normalizando os parâmetros de Stokes por S_0, podemos chegar à seguinte fórmula:

$$SOP = \frac{S_1}{S_0} + \frac{S_2}{S_0} + \frac{S_3}{S_0} = s_1 + s_2 + s_3 \quad 21$$

Onde SOP representa o estado de polarização da luz. Um espaço geométrico descrito pelos parâmetros de Stokes normalizados gera a casca esférica conhecida como Esfera de Poincaré. A Figura 12 representa os estados de polarização na Esfera de Poincaré. Nesta representação podemos perceber que os estados de polarização linear correspondem ao equador da esfera, e que os estados de polarização circular correspondem aos pólos da esfera.

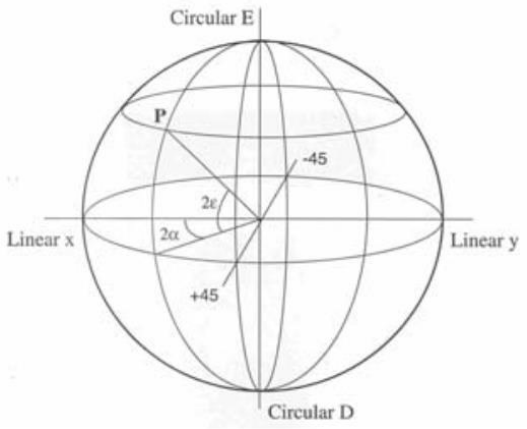

Figura 12 - Representação da Esfera de Poincaré e dos principais estados de polarização.

e. Polarímetro Analógico

Um polarímetro é um instrumento capaz de medir as intensidades de campo linearmente polarizado nos eixos x, y, $45°$ e $-45°$, e as intensidades de campo circularmente polarizado à esquerda e à direta, e, portanto, trata-se de um instrumento capaz de identificar tanto o estado de polarização da luz (SOP) quanto o grau de polarização da luz (DOP).

"Os polarímetros disponíveis comercialmente nos dias de hoje são baseados na divisão do sinal luminoso em amostras no tempo ou no espaço. Cada uma dessas amostras passa através de polarizadores e de lâminas de quarto de onda de forma a determinar os parâmetros de Stokes do sinal." (TEMPORÃO, Guilherme P., 2003, p. 47).

Para realizar as medições das intensidades de campo, os polarímetros utilizam polarizadores com eixos de transmissão x, y, $45°$, $-45°$ e lâminas de quarto de onda, e, posteriormente, a luz passa por fotodetectores.

O polarímetro utilizado nesta tese de monografia é um polarímetro analógico da General Photonics, PolaDetect (POD-001). O PolaDetect (POD-001) é um polarímetro projetado para ser de baixo custo com capacidade de caracterização da polarização em alta velocidade sem que seja necessário interromper o tráfego de dados. Juntamente com ele vieram matrizes de calibração que levam em consideração o comprimento de onda do laser utilizado. Cada matriz de calibração deve ser usada para uma banda de comprimento de onda +/- 5 nm. Nos foram dadas cinco matrizes de calibração para os comprimentos de onda de 1520 nm, 1530 nm, 1540 nm, 1550 nm e 1560 nm.

Os parâmetros de Stokes podem ser obtidos através da seguinte fórmula:

$$\begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix} \quad 22$$

f. Conversor Analógico-Digital

Os conversores analógico-digitais são dispositivos eletrônicos capazes de, a partir de uma grandeza analógica, gerar uma representação digital. Os conversores analógico-digitais são frequentemente abreviados por conversores AD ou ADC.

Uma grandeza analógica pode ser entendida como algo contínuo, ou seja, são sinais que variam continuamente com o tempo, alguns exemplos são tensão, disco de vinil, fita magnética, fita cassete, fotografia em película, etc.

Grandezas digitais, por outro lado, são conjuntos de valores descontínuos que são amostrados com uma certa frequência de amostragem, ou seja, um sinal digital só é definido para certos instantes de tempo e o conjunto de valores que ele pode assumir é finito. A Figura 13 deixa clara a diferença entre a representação analógica (em vermelho) e a digital (em cinza) de um mesmo sinal real.

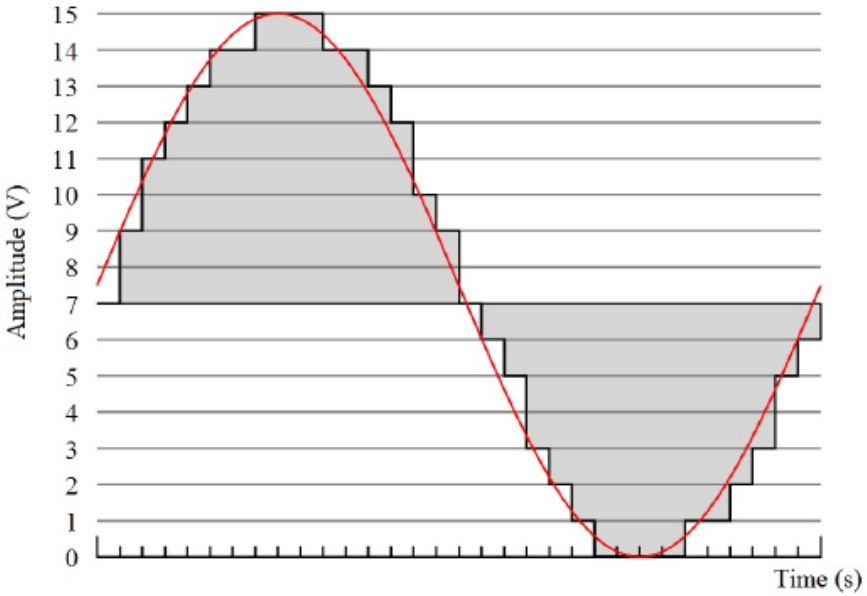

Figura 13 – Representação de um sinal analógico e de um sinal digital.

Os conversores AD são muito importantes e, por isso, merecem uma seção especial. Diariamente utilizamos diversos conversores AD sem nos darmos conta, eles estão presentes em controles remotos, computadores, celulares, entre outros dispositivos. Sendo que estes dois últimos, computadores e celulares, são formados por diversos conversores AD, por exemplo, para gravar a voz, para reproduzir sons, para gerar imagens na tela, sensores de luminosidade, etc.

Neste projeto empregamos um conversor AD bastante utilizado em nosso laboratório de pesquisas por possuir uma alta taxa de amostragem e por ser bastante confiável, além de simples. Os conversores AD utilizados foram montados utilizando as recomendações descritas no datasheet do ADS805e.

A Figura 14 mostra como devemos realizar as ligações no ADS805e juntamente com os valores dos capacitores, de forma a adequá-los à nossa aplicação (sinais entre 0 e 5 Volts e 10 MHz de banda).

Um esquema completo do circuito dos conversores AD é mostrado na Figura 15. Neste esquema estão envolvidas algumas melhorias como, por exemplo, o uso de um regulador de tensão que garante que a tensão de alimentação não seja superior à suportada pelo circuito integrado ADS805e.

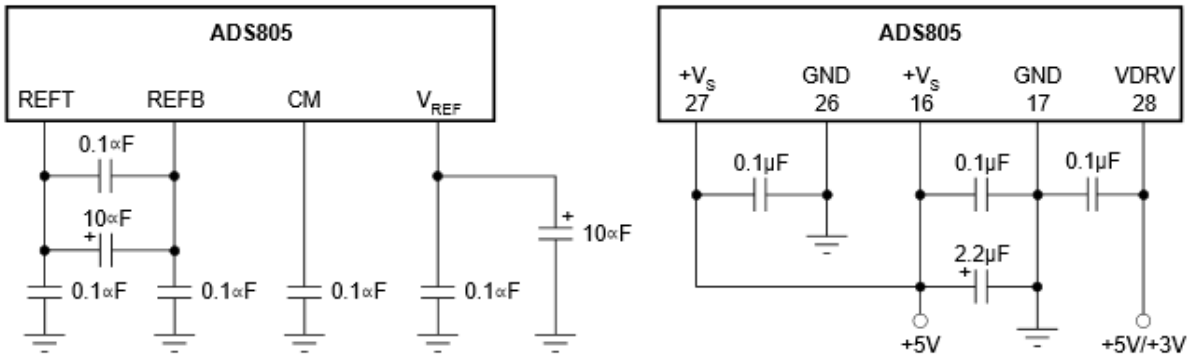

Figura 14 - Configuração do ADS805e.

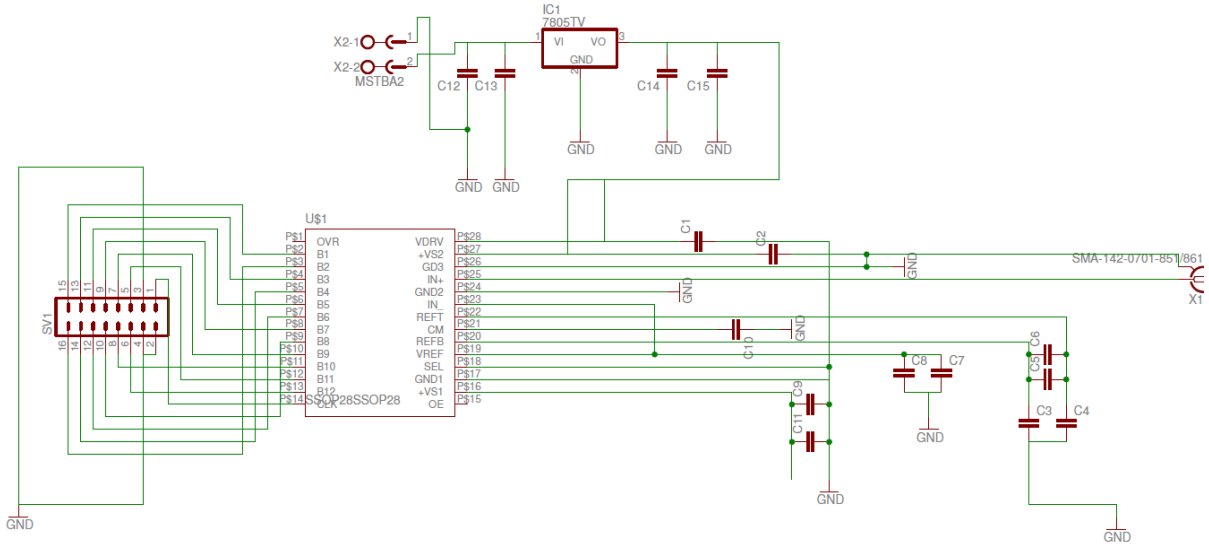

Figura 15 – Esquema do conversor AD utilizado no projeto.

O próximo passo foi a montagem deste circuito em uma placa de circuito impresso, a Figura 16 mostra o esquemático do conversor AD, porém, antes de montar a placa foi preciso organizar os componentes e fazer um roteamento das ligações. Uma dificuldade comum é que algumas trilhas não podem cruzar outras trilhas, portanto, a organização correta dos componentes na placa de circuito impresso pode facilitar este trabalho. Um software muito utilizado neste tipo de aplicação é o CadSoft Eagle.

Depois de gerado o arquivo com o roteamento das linhas, uma fresa automática do laboratório é capaz de interpretá-lo e criar uma placa de circuito impresso. O processo utilizado pela fresa é o de remover a cobertura de cobre de uma placa "virgem" gerando as conexões entre elementos. O próximo passo foi soldar os elementos à placa, o que foi realizado em uma das estações de solda do laboratório como mostrado na Figura 17.

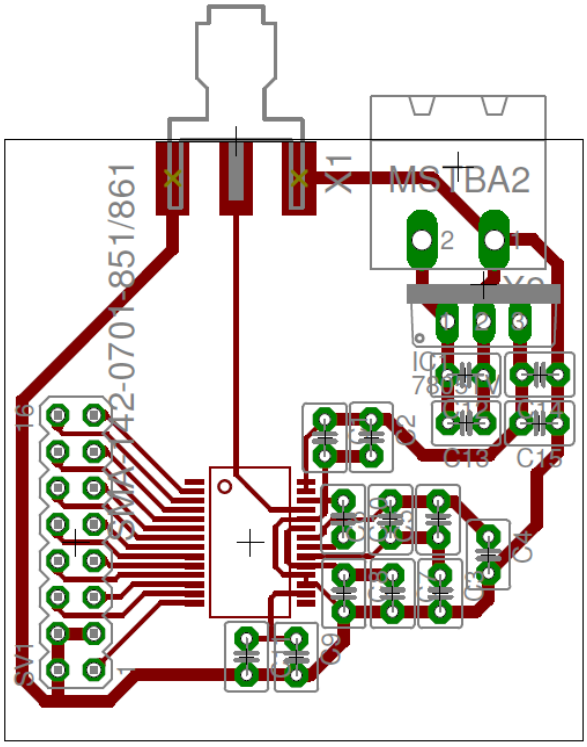

Figura 16 – Circuito do conversor AD.

Figura 17 - Estação de solda laboratorial.

O resultado final do processo de criação da placa do ADC pode ser observado na Figura 18, que traz tanto a frente quanto o verso da placa.

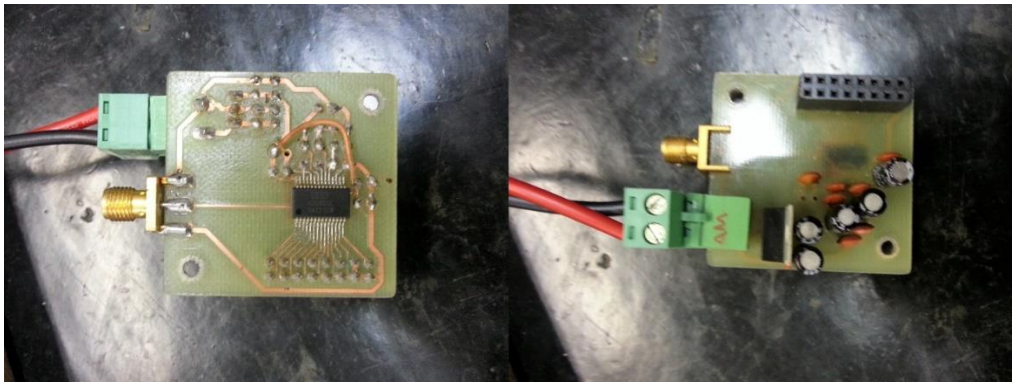

Figura 18 - Frente e verso da placa de circuito impresso criada para o ADC.

g. Protocolo VGA e Perspectiva Isométrica

Um dos primeiros passos do projeto, foi a o estudo sobre a comunicação entre uma placa de vídeo e um monitor LCD usando o protocolo VGA Dessa forma, o circuito digital necessário para transmitir os dados da FPGA para um monitor genérico poderia ser posto em teste. Além de representar um desenvolvimento visual do projeto, nessa etapa se deu o primeiro contato direto com a linguagem VHDL e o paradigma da síntese de circuitos digitais.

A história por trás da padronização do protocolo VGA começa em 1980. Até esta data ainda não havia um padrão amplamente aceito para comunicações com um monitor. Isto foi resolvido com o surgimento do VESA (Video Electronics Standards Association), uma organização internacional fundada em 1980 por nove fabricantes de adaptadores de vídeo cujo objetivo inicial era produzir um padrão para a resolução SVGA de 800 x 600 pixels.

O termo VGA (Video Graphics Array) é utilizado para denominar um protocolo padrão de transmissão de vídeo criado para tornar possível e prática a transmissão de imagens de um computador para um monitor genérico.

O termo VGA também é utilizado para denominar o conector analógico VGA, criado pela IBM em 1987, que na época era capaz de reproduzir até 256 cores, com resolução máxima de 640 x 480 pixels. A Figura 19 apresenta o desenho de um conector do tipo VGA com seus 15 pinos além da função de cada um.

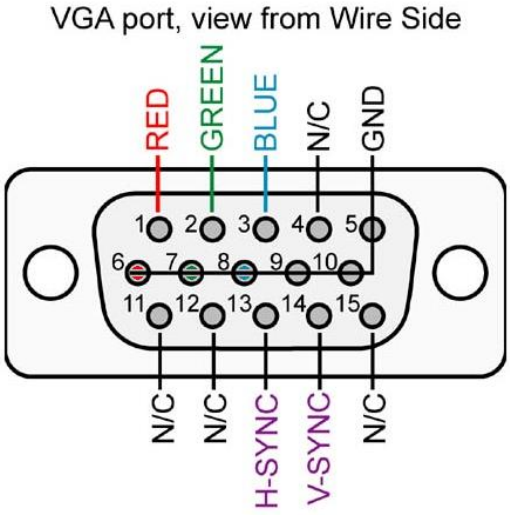

Figura 19 - Conector VGA

Um sinal VGA é composto por 5 sinais (vide Figura 19) sendo eles: H-SYNC (sincronização horizontal), V-SYNC (sincronização vertical), RED, GREEN e BLUE (intensidade de coloração RGB). O sinal H-SYNC indica que uma nova linha será gerada, enquanto que o sinal V-SYNC indica que um novo quadro será gerado.

A frequência de sincronismo gira em torno de 60 Hz, podendo chegar a 75 Hz, e tanto V-SYNC quanto H-SYNC são formados por trens de pulso de tensão entre 3.3 e 5 V. Os sinais RED, GREEN e BLUE são sinais analógicos cuja tensão varia entre 0 V (brilho mínimo) e 0.7 V (brilho máximo) e correspondem à intensidade de cada uma das componentes de cor de cada pixel da tela do monitor.

O processo de atualização da tela começa no canto superior esquerdo e pinta um pixel de cada vez, da esquerda para a direita. Ao final da primeira linha, o sinal H-SYNC indica que uma nova linha será pintada. Uma vez que toda a tela foi pintada, o sinal V-SYNC indica o início de um novo quadro, e o processo de atualização da tela começa novamente.

Durante o tempo em que não estão sendo exibidos pixels e o feixe está retornando para a coluna esquerda para iniciar uma nova linha, os sinais RGB devem ser definidos na cor preta (tudo zero do ponto de vista do sinal digital).

Um dos objetivos da estrutura de visualização FPGA-VGA era ser capaz de desenhar uma esfera no monitor. Isso está intimamente ligado à representação de estados de polarização através da Esfera de Poincaré. Uma esfera, contudo, para ser bem visualizada em um monitor (uma plataforma bidimensional) deve contar com uma representação em perspectiva, ou seja, um modo de veicular a idéia de uma forma tridimensional sobre uma plataforma bidimensional.

Existem alguns métodos para realizar o processamento de uma figura em hardware. Um deles é a geometria analítica discretizada, ou seja, criar estruturas que, dadas as coordenadas horizontal e vertical de um pixel, calculem se o pixel faz parte da forma geométrica ou não dentro de uma margem de erro que está associada com a resolução do monitor.

Apesar de ser um método necessário quando as formas são dinâmicas, no caso de uma forma estática, como é a do caso estudado, há a opção de criar uma estrutura de memória fixa (uma ROM – Read-Only Memory) que guarda as posições dos pixels da esfera e um comparador que testa se as coordenadas estão dentro da estrutura geométrica. O modo como essa memória foi criada dentro da FPGA será comentado mais adiante.

A Figura 20 traz algumas perspectivas utilizadas na representação de estruturas tridimensionais. Alguns testes foram feitos para a identificação de qual dessas perspectivas seria a mais indicada para a visualização do estado de polarização, que será representado como um ponto na superfície da esfera de Poincaré. Alguns exemplos práticos já implementados (como os programas AutoCAD, Inventor Series, SolidWorks, SketchUp e outros), além dos resultados dos testes com os pesquisadores do laboratório identificaram que a perspectiva ideal seria a Isométrica.

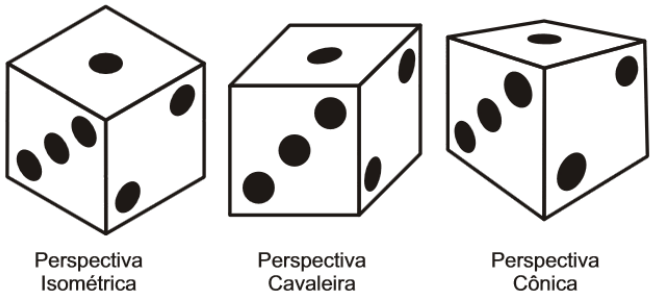

Figura 20 – Exemplos de perspectivas.

Depois da escolha da perspectiva de representação da esfera, um ponto importante é a determinação da matriz de rotação necessária para mudar o eixo de coordenadas retangulares canônicas (x,y,z) para o eixo de coordenadas isométricas (u,v,w). Isso é fundamental, uma vez que as coordenadas processadas pela placa estarão no eixo de coordenadas (x,y,z) mas a sua visualização será no eixo de coordenadas (u,v,w). Quando tratamos cada uma dessas componentes como as coordenadas de um vetor tridimensional, a transformação necessária passa a ser uma matriz.

Nesta parte utilizamos o Maple[4], que é um sistema algébrico computacional, ou seja, é um software comercial que facilita o cálculo matemático no computador através de bibliotecas que podem ser de: álgebra linear, diferenciais e integrais, gráficas, simbólicas, etc. Para objetos tridimensionais, podemos realizar uma rotação utilizando uma das seguintes matrizes de rotação[5]:

$$M_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & -sen(\alpha) \\ 0 & sen(\alpha) & cos(\alpha) \end{bmatrix} \quad \begin{matrix} Rotação \\ em\ torno \\ do\ eixo\ x \end{matrix} \qquad 23$$

$$M_y = \begin{bmatrix} cos(\beta) & 0 & sen(\beta) \\ 0 & 1 & 0 \\ -sen(\beta) & 0 & cos(\beta) \end{bmatrix} \quad \begin{matrix} Rotação \\ em\ torno \\ do\ eixo\ y \end{matrix} \qquad 24$$

[4] A PUC-Rio concede uma licença do software Maple para todos os alunos do Centro Técnico Científico (CTC).

[5] As matrizes de rotação estão orientadas no sentido horário.

$$M_z = \begin{bmatrix} cos(\gamma) & -sen(\gamma) & 0 \\ sen(\gamma) & cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{matrix} Rotação \\ em\ torno \\ do\ eixo\ z \end{matrix} \quad (25)$$

A Figura 21 mostra um exemplo de rotação de um vetor (ponto vermelho) no Maple, cujo eixo de coordenadas retangulares canônicas (x,y,z) foi transportado, através de uma transformação linear, para o eixo de coordenadas isométricas (u,v,w).

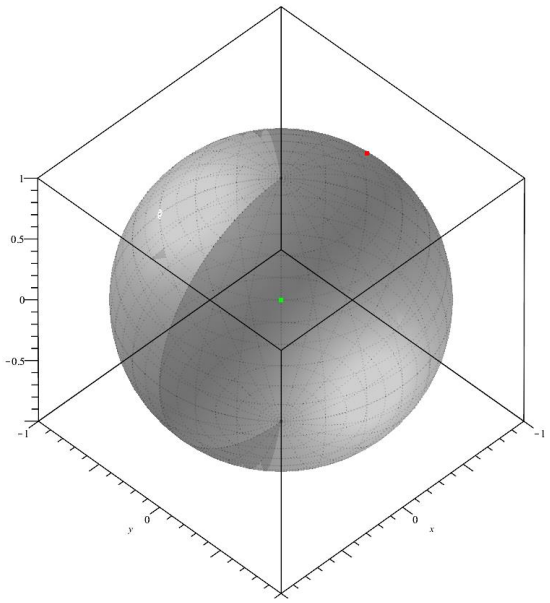

Figura 21 – Perspectiva isométrica no Maple.

Quando realizamos 3 rotações sucessivas devemos multiplicar as matrizes em ordem inversa. Por exemplo, quando realizamos primeiro uma rotação em torno do eixo x, depois, eixo z e eixo y, a matriz resultante será:

$$M_R = M_y \cdot M_z \cdot M_x \quad (26)$$

Chegamos, portanto, à seguinte matriz de rotação:

$$Matriz = \begin{bmatrix} -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{6}}{6} & \frac{\sqrt{6}}{6} & -\frac{\sqrt{6}}{3} \\ -\frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3} \end{bmatrix} \quad (27)$$

h. Implementação da Esfera Isométrica em VHDL

A implementação do desenho de uma esfera estática em VHDL foi uma tarefa que envolveu muitos processos, inclusive a criação de um algoritmo em MATLAB, um software interativo de alta performance voltado para o cálculo numérico a que se tem acesso graças à licença para alunos da PUC-Rio. Em um primeiro momento, os cálculos das dimensões da esfera foram realizados no papel levando em conta o tamanho da tela, o tamanho desejado da esfera e a posição que ela deveria ficar.

Decidimos centrar a esfera no monitor, como a resolução utilizada é SVGA, ou seja, 800 x 600, tivemos que criar uma esfera cujo centro ficasse na posição (400,300). Outro aspecto que abordamos foi sobre o raio da esfera que deveria ser menor que 300, pois a menor dimensão da tela é 600 pixels, mas, ao mesmo tempo, queríamos utilizar ao máximo o monitor, portanto, decidimos que o raio seria de 270 pixels.

Inicialmente esse passo foi realizado através da utilização de software para desenho. A imagem salva em formato "bitmap" foi importada para o MATLAB que, através do algoritmo desenvolvido, criava uma lista de palavras de 16 bits contendo as informações de coordenadas que seriam ocupadas por cada um dos bits da esfera. O algoritmo foi desenvolvido de forma que o resultado já estivesse formatado para ser escrito em linguagem VHDL, acelerando bastante o tempo entre testes.

O próximo passo foi criar um algoritmo no MATLAB que fosse capaz de criar uma imagem em perspectiva isométrica. Portanto, outro tema de estudo foi sobre o formato "bitmap" e como o MATLAB lida com ele. Após isso, foi criado um algoritmo que além de desenhar a esfera em perspectiva isométrica fosse, também, capaz de desenhar os eixos principais e desenhar de cores diferentes a parte da frente e a parte de trás da esfera, pois, desta forma, facilitaria o trabalho de reconhecer se um ponto está na parte da frente ou detrás. Outra idéia incorporada foi que, além da esfera, o ponto que representa o estado de polarização também tivesse 2 cores, uma se ele estivesse na parte da frente e outra cor se ele estivesse na parte de trás da esfera. A Figura 22 mostra o resultado desse algoritmo.

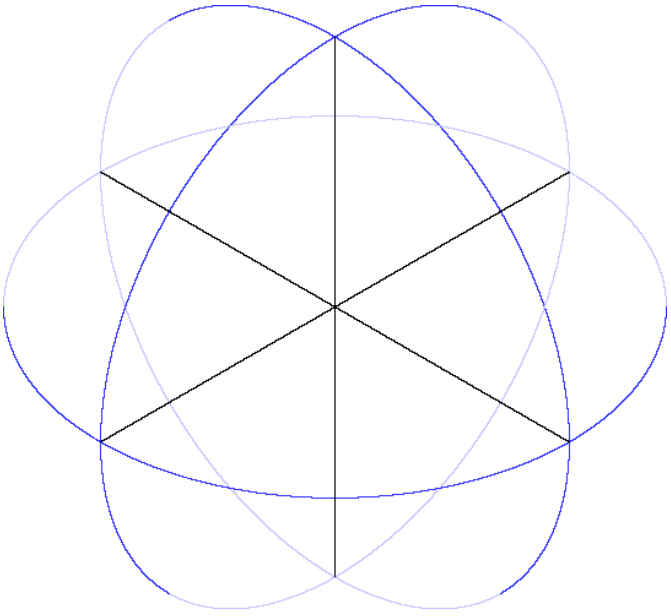

Figura 22 – Esfera em perspectiva isométrica criada pelo algoritmo para MATLAB.

O algoritmo foi aprimorado de forma que, em sua forma final, já era possível incluir um ponto em coordenadas (x,y,z) e ter a visualização a priori do resultado esperado quando mostrado no monitor pela FPGA em coordenadas (u,v,w).

O próximo passo foi o desenvolvimento de uma estrutura de memória para lidar com a esfera, este passo começou de maneira bastante primitiva. Inicialmente, foi idealizada uma memória que contivesse a informação sobre cada pixel na tela, ou seja, qual era a cor que cada pixel deveria assumir. Além de tomar uma área muito grande de memória da placa (tornando o projeto inviável), o método de comparação para cada pixel seria muito lento (pois muitas posições deveriam ser arguidas) e não estaria de acordo com o protocolo de atualização de imagem.

A seguir foi pensado em um tipo de memória que contivesse apenas os pixels da esfera, porém, mesmo utilizando menos memória, seria um trabalho dispendioso para à FPGA pois, desta forma, ela ainda deveria verificar pixel por pixel se deveria ou não desenhar um ponto na tela.

A solução foi criar um algoritmo em MATLAB que criasse pequenos "blocos de pixels", caso algum pixel dentro deste bloco fizesse parte do desenho da esfera, e armazenasse suas posições em uma memória estática. Além disso, foi pensado um jeito de resolver o problema do método de comparação que anteriormente testava cada pixel. A solução foi a criação de uma memória ROM que contivesse os "blocos da esfera" e de um módulo sequencial que teria informações de onde desenhar a esfera, a Figura 23 mostra como esse processo funciona. Este método pode ser comparado à utilização de ponteiros em programação, embora de uma forma muito primitiva, o que é bastante interessante do ponto de vista do entendimento desse tipo de estrutura.

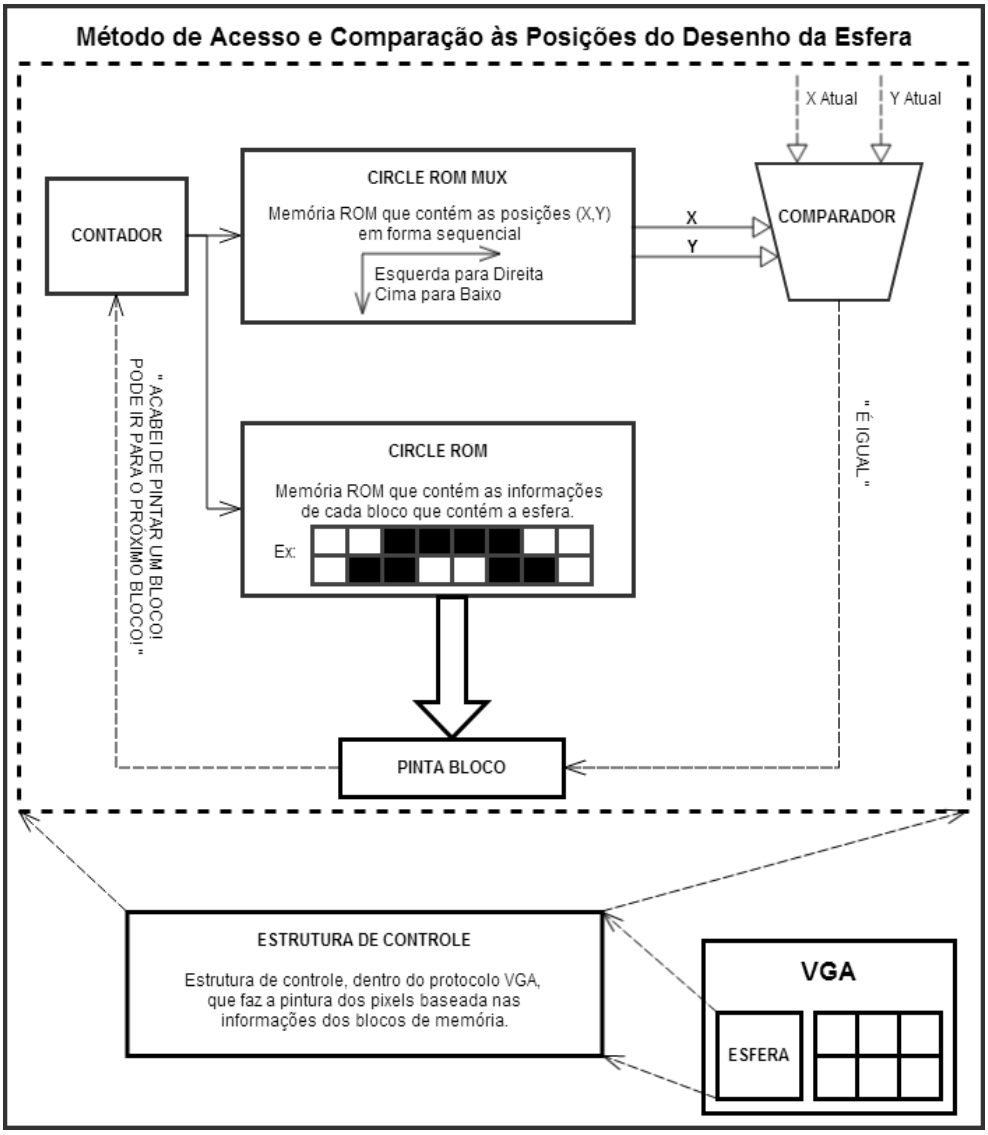

Figura 23 – Algoritmo criado para ser executado pela FPGA.

A Figura 24 mostra a estrutura interna à FPGA, sendo ela composta por 4 blocos principais. São eles: VGA_CONTROL (protocolo VGA), XYZ to ISOMETRIC (algoritmo que recebe os parâmetros de Stokes e que os passa para coordenadas isométricas), Python (interface entre o computador e a FPGA), ADs Management (controle dos conversores AD).

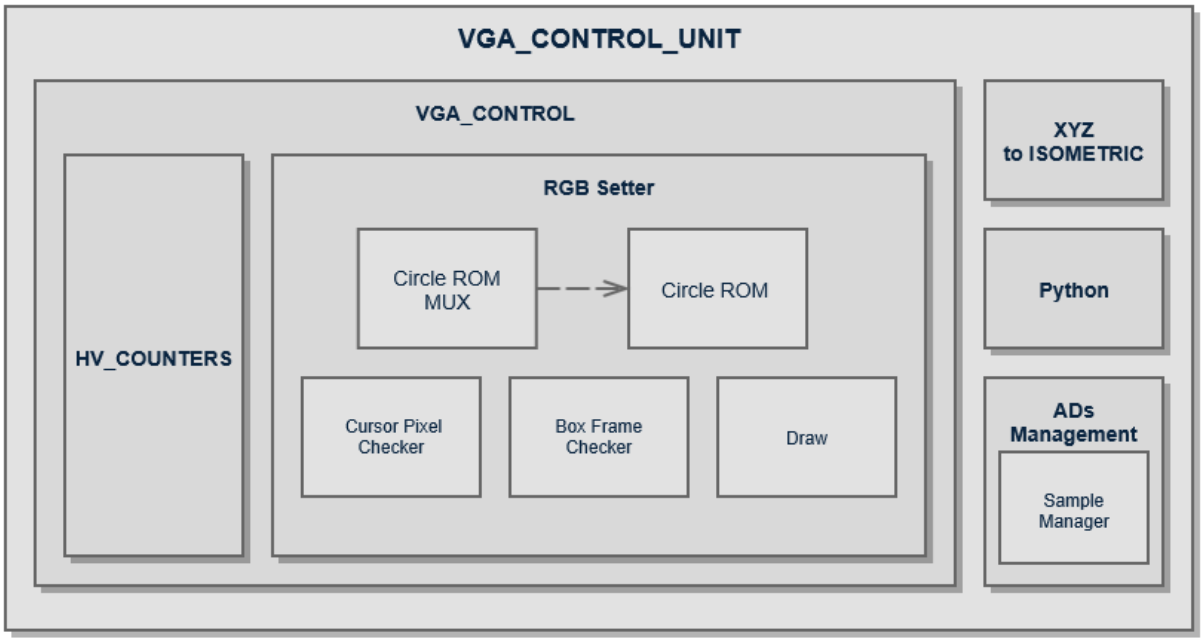

Figura 24 – Diagrama de blocos da estrutura interna à FPGA.

O resultado final dessa etapa do projeto, ou seja, a informação do desenho da esfera enviada pela FPGA para o monitor de LCD, pode ser visualizada na Figura 25.

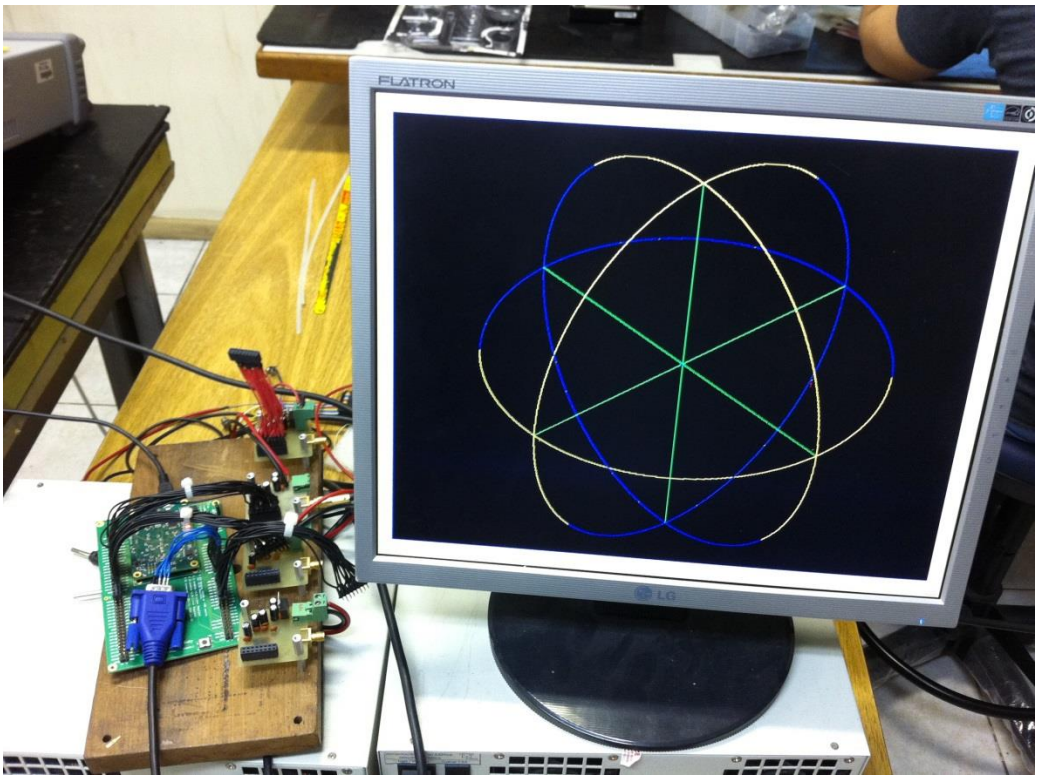

Figura 25 - Desenho da representação isométrica de uma esfera enviada da FPGA para o monitor LCD.

3. Conclusão

Através da síntese de circuitos digitais em FPGA, e do estudo de protocolos de comunicação digital, bem como o estudo das propriedades de ondas eletromagnéticas e da representação do estado de polarização da luz, foi possível criar uma estrutura física que, em função de sinais de entrada provenientes de um polarímetro analógico, faça a representação gráfica do estado de polarização da luz.

A estrutura criada até então representa um grande avanço no desenvolvimento de uma ferramenta essencial para projetos de comunicações ópticas. Como trabalho futuro, pode-se citar a calibração dos sinais de entrada após a conversão analógico-digital, a criação de uma única placa de circuito impresso com os quatro conversores analógico-digitais, assim como a otimização dos códigos VHDL.

4. Referências

[1] R. Pengelly, "All About Circuits," [Online]. Available: http://www.allaboutcircuits.com/. [Acesso em 15 de Novembro de 2014].

[2] P. J. Ashenden, "The VHDL Cookbook," 1990. [Online]. Available: http://www.ics.uci.edu/~alexv/154/VHDL-Cookbook.pdf. [Acesso em 20 de Agosto de 2014].

[3] A. Rushton, VHDL for Logic Synthesis, 3ª ed., Chichester, West Sussex: John Wiley & Sons, Ltd, 2011.

[4] C. A. Balanis, Teoria de Antenas, 3ª ed., vol. 1, Rio de Janeiro, RJ: LTC - Livros Técnicos e Científicos Editora S.A., 2009.

[5] D. Halliday, Resnick e J. Walker, Fundamentos de Física - Óptica e Física Moderna, 8ª ed., vol. 4, Rio de Janeiro, RJ: LTC - Livros Técnicos e Científicos Editora S.A., 2009.

[6] G. P. Temporão, *Um Polarímetro de Baixo Custo,* Rio de Janeiro, RJ: Dissertação de Mestrado – Departamento de Engenharia Elétrica, PUC-Rio, 2003.

[7] A. S. Sedra e K. C. Smith, Microeletrônica, 5ª ed., São Paulo, SP: Pearson Prentice Hall, 2007.

[8] G. C. d. Amaral, *FPGA Applications on Photon Detection Systems,* Rio de Janeiro: MsC Thesis, DEE, PUC-Rio, 2014.

[9] General Photonics Corporation, *PolaDetect (POD-001),* General Photonics Corporation, Datasheet..

[10] Texas Instruments, *ADS805e,* Texas Instruments, Datasheet.

[11] OpalKelly, *XEM3005 User Manual,* OpalKelly, Technical Report.

5. Anexos

a) Interface gráfica do código que gera a esfera de Poincaré.

Foto da interface gráfica do código que gera a esfera mostrada na Figura 22. Este código foi feito em Matlab, e após a utilização desta interface gráfica uma imagem será criada na pasta raiz, juntamente com o código VHDL da memória ROM.

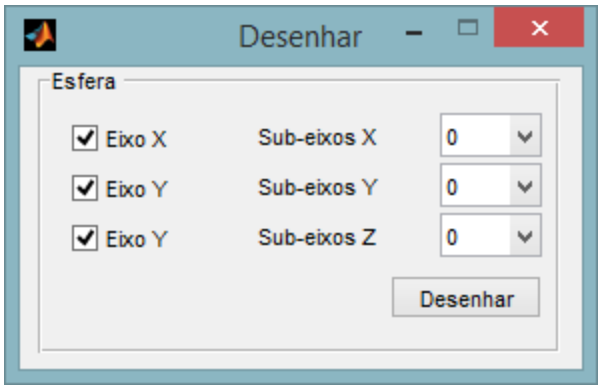

Figura 26 - Interface gráfica do programa feito em Matlab.

Utilizando os parâmetros acima, apenas os eixos serão desenhados, como mostrado na figura abaixo:

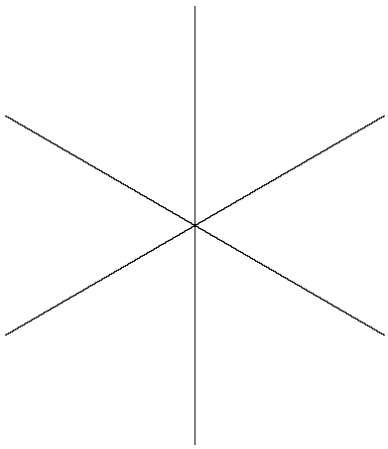

Figura 27 - Eixos em perspectiva isométrica.

Quando selecionamos a quantidade de sub-eixos, o programa pede em que posição queremos desenhar o eixo. Os valores possíveis para as posições variam entre $[-1;1]$. Um exemplo usando 1 sub-eixo x, 1, sub-eixo y e 7 sub-eixos z, com valores 0, 0, 0, -0.96, +0.96, -0.80, +0.80, -0.3 e +0.3, respectivamente, gerou a esfera mostrada na Figura 29.

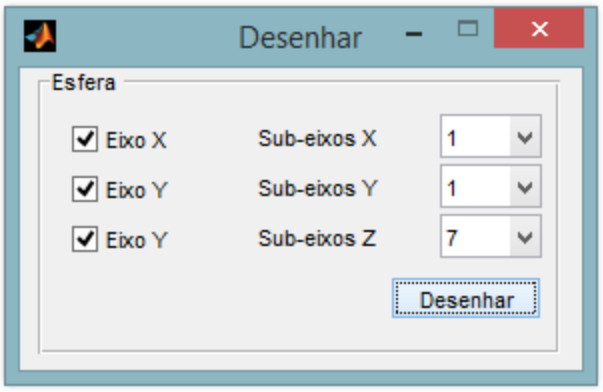

Figura 28 - Interface gráfica com sub-eixos selecionados.

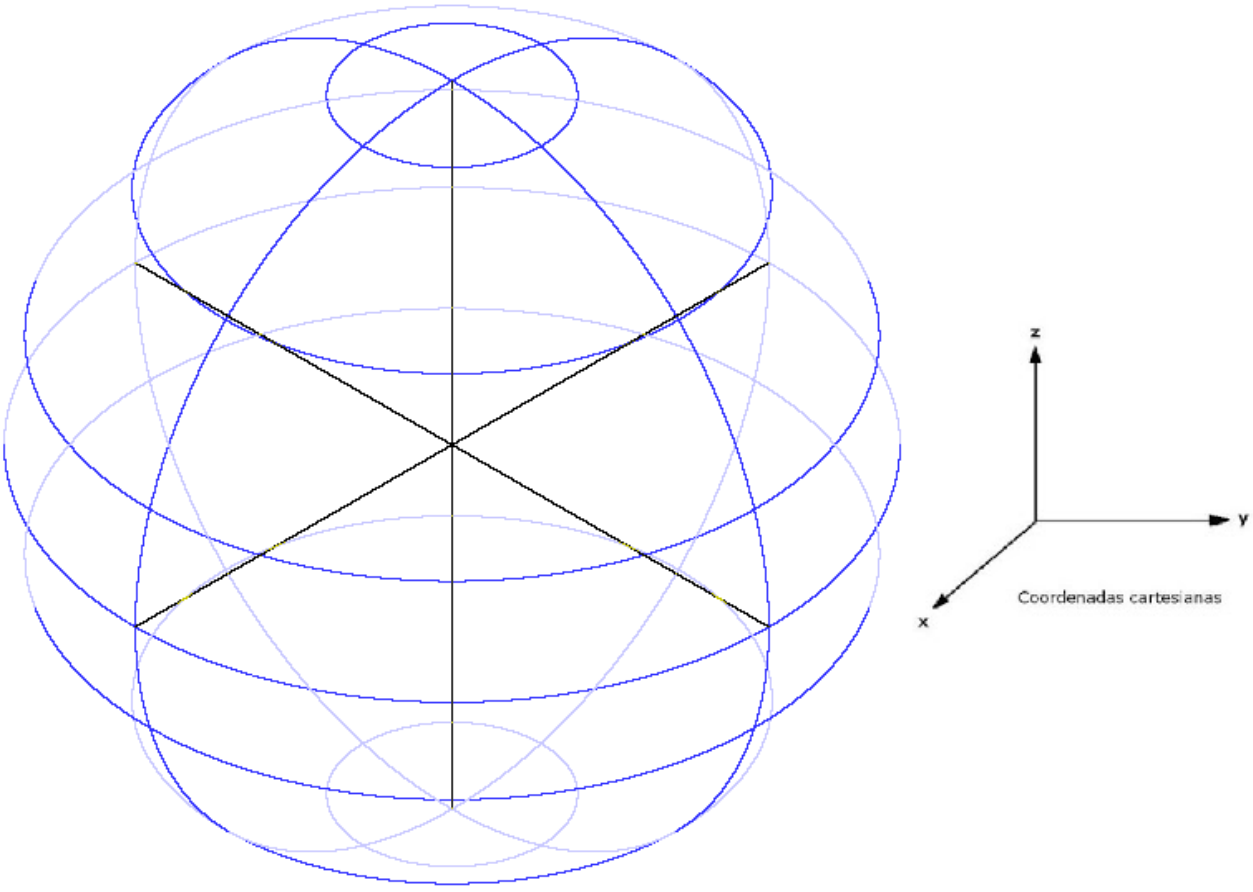

Figura 29 - Esfera utilizando as informações citadas no texto acima.