Universidade Federal da Paraíba

Centro de Energias Alternativas e Renováveis

Departamento de Engenharia Elétrica

GUSTAVO MAXIMO URQUIZA DE SÁ

MEDIÇÃO DE DISTÂNCIA COM RADAR: UM ESTUDO DE CASO DE SISTEMA DIGITAL UTILIZANDO A FERRAMENTA XILINX VIVADO HLS

João Pessoa, Paraíba
Junho de 2018

GUSTAVO MAXIMO URQUIZA DE SÁ

MEDIÇÃO DE DISTÂNCIA COM RADAR: UM ESTUDO DE CASO DE SISTEMA DIGITAL UTILIZANDO A FERRAMENTA XILINX VIVADO HLS

Trabalho de Conclusão de Curso submetido ao Departamento de Engenharia Elétrica da Universidade Federal da Paraíba como parte dos requisitos necessários para a obtenção do título de Engenheiro Eletricista.

Área de Concentração: Sistemas Digitais e Análise de Sinais Digitais

Orientador:
Prof. Lucas Vinícius Hartmann

João Pessoa, Paraíba
Junho de 2018

Catalogação na publicação
Seção de Catalogação e Classificação

Silva, Gustavo Maximo Urquiza de.
Medição de Distância com Radar: Um Estudo de Caso de Sistema Digital Utilizando a Ferramenta Xilinx Vivado HSL / Gustavo Maximo Urquiza de sa. - João Pessoa, 2018.
70 f.

Orientação: Lucas Hartmann.
Monografia (Graduação) - UFPB/CEAR.

1. Radar, FPGA, Verilog, Sistemas Digitais, PRF. I. Hartmann, Lucas. II. Título.

UFPB/BC

GUSTAVO MAXIMO URQUIZA DE SÁ

MEDIÇÃO DE DISTÂNCIA COM RADAR: UM ESTUDO DE CASO DE SISTEMA DIGITAL UTILIZANDO A FERRAMENTA XILINX VIVADO HLS

Trabalho de Conclusão de Curso submetido ao Departamento de Engenharia Elétrica da Universidade Federal da Paraíba como parte dos requisitos necessários para a obtenção do título de Engenheiro Eletricista.

Área de Concentração: Sistemas Digitais e Análise de Sinais Digitais.

Aprovado em / /

Professor Lucas Vinícius Hartmann
Universidade Federal da Paraíba
Avaliador

Professor Carlos Alberto de Souza Filho
Universidade Federal da Paraíba
Avaliador

Professor José Maurício Ramos de Souza Neto
Universidade Federal da Paraíba
Avaliador

Dedico este trabalho a Deus, meu Senhor e Salvador, à minha família e aos meus amigos, que sempre estiveram comigo me motivando e apoiando nos momentos mais difíceis.

AGRADECIMENTOS

Em primeiro lugar agradeço a Deus, meu Senhor, por todas as bênçãos em minha vida, sem Ele eu não estaria aqui.

Agradeço também à minha família por ter me apoiado incondicionalmente todos os tempos.

Aos meus amigos e colegas de curso que sempre estiveram presente nessa jornada, em especial ao grupo PPG.

RESUMO

Utilizar ferramentas de síntese para desenvolver sistemas digitais são uma atividade primordial para um Engenheiro Eletricista. Este Trabalho de Conclusão de Curso consiste em utilizar a ferramenta *Xilinx* Vivado HLS, para desenvolver um sistema digital medidor de distância. Este sistema irá realizar medições provenientes de sinais de radar do tipo PRF (*Pulse Repetition Frequency*) que serão emulados. Os dois sistemas, medidor e emulador, serão desenvolvidos para um dispositivo FPGA utilizando a linguagem de descrição de *hardware Verilog*.

A placa FPGA utilizada foi a Basys 3 da Digilent©.

Palavras-chave: Radar, FPGA, Verilog, Sistemas Digitais, PRF(*Pulse Repetition Frequency*).

ABSTRACT

Utilize synthesis tools to develop digital systems are an important activity to an Electrical Engineer. This work consists in use the Xilinx Vivado HLS to develop a digital system capable of ranging. This system will measure distance from radar PRF (Pulse Repetition Frequency) signals that will be in the form of emulation. Both systems will be set in a FPGA device with Verilog hardware description language.

The FPGA board used was the Basys 3 from Digilent©.

Keywords: Radar, FPGA, Verilog, Digital Systems, PRF (Pulse Repetition Frequency).

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Sistemas de radar é um campo muito explorado na engenharia por possuir um grande espectro de aplicações e também por apresentar fidelidade de informação. Um radar é capaz de detectar objetos a quilômetros de distância, informar sua distância, velocidade e direção de movimento, podendo ser empregado em diversas aplicações como atuar com controle de tráfego, auxiliar meteorologistas com análise do tempo, identificar resíduo em dutos nas aplicações industriais, entre outras aplicações (CURRY, 2004).

Para as pessoas em geral a palavra radar remete a utilização de radares em rodovias, porém o campo de radar é bem mais amplo. O desenvolvimento desta tecnologia é datado desde o século XIX, no fim da década de 1880 quando o cientista alemão Heirich Hertz realizava experimentos com radiação eletromagnética, como extensão ao trabalho teórico desenvolvido por James Clerk Maxwell (SKOLNIK, 2018).

Durante os anos 1930 iniciaram os estudos mais aprofundados dos radares, muito provocado por conta da guerra, pois quem possuísse um sistema de radar mais robusto estaria em maior vantagem durante os confrontos, pois saberiam com antecedência a presença dos inimigos.

Com advento da era digital, durante os anos 70, os sistemas de radar sofreram mais um período de expansão de suas tecnologias, a partir de agora contando então com sistemas de análise de *doppler* (RICHARD; SCHEER; HOLM, 2010). Desde então inúmeros os avanços neste âmbito de pesquisa e atualmente suas aplicações vão além das militares e de controle de tráfego.

Segundo Dickmann e Appenrondt (2014) esta ciência passou a ter implicações em outros campos como o industrial, conquistando espaço na área de meteorologia, controle de tráfego aéreo e nos carros autônomos. Ainda, notou-se também uma expansão de sua utilização no ramo da medicina, com exame do *status* de saúde, analisando batimentos cardíacos e trajetória do centro de gravidade do corpo humano, como apontado por Mizirin et al. (2010)

Os avanços das tecnologias de semicondutores com surgimento da miniaturização dos transistores transformaram as tecnologias digitais em um campo amplamente

utilizado, possibilitando construção de sistemas digitais com operações mais complexas, rápidas e ocupando a menor área possível (FERDJALLAH, 2011).

Assim, surgiram os processadores que são sistemas digitais capazes de efetuar operações matemáticas de forma rápida e precisa além de diversas outras funções (DAWOUD; PEPLOW, 2010).

Os sistemas digitais são desenvolvidos em diversos formatos, podendo ser programas em linguagem de programação aplicados a processadores ou microcontroladores, dispositivos digitais construídos para aplicações específicas, por exemplo, os SoC (*System on Chip*), e também há os dispositivos FPGA (*Field Programmable Gate-Array).*

Com o desenvolvimento dos dispositivos FGPA e das linguagens de descrição de *hardware* (HDL – *Hardware Description Language*) foi permitido que os desenvolvedores pudessem rapidamente desenvolver e simular circuitos digitais, permitindo a criação de protótipos para sistemas digitais, um exemplo são as desenvolvedoras de processadores que utilizam placas de FPGA para prototipação dos sistemas digitais que estão em desenvolvimento como apontado por Chu (2008).

Visto isso, a motivação para esse trabalho é que para o desenvolvimento de sistemas digitais é necessária a utilização de ferramentas que realizem síntese, análise e implementação. A escolha do tema deu-se devido à sua relação direta com elementos do curso de Engenharia Elétrica, principalmente conteúdos referentes à eletrônica, como analise de sinais e sistemas e utilização de ferramenta para síntese e implementação de sistema digital.

1.1 OBJETIVOS

O objetivo principal desse trabalho é apresentar dois sistemas digitais, desenvolvidos em Verilog, utilizando a ferramenta Vivado da Xilinx para sua síntese e implementação em que um dos sistemas irá emular sinais PRF e o outro sistema irá medir distância utilizando sinais PRF (*Pulse Repetition Frequency*). Os sistemas serão programados em um dispositivo FPGA (Basys 3).

1.1.1 OBJETIVOS ESPECÍFICOS

O trabalho tem como objetivos específicos o estudo de assuntos vistos na graduação como análise de sinais e sistemas, desenvolvimento de projetos e utilização de ferramentas para modelagem, os principais tópicos são:

- Estudo de Sinais de Sistema de Radar;
- Estudo de ferramenta de síntese de HDL;
- Estudo de sistema digital em FPGA.

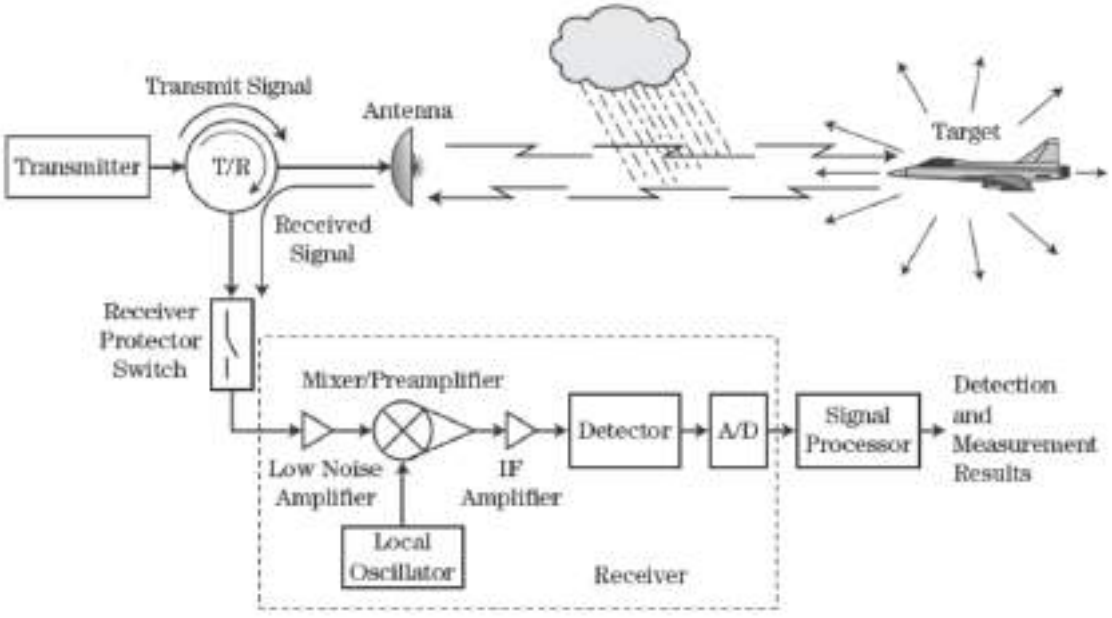
2 SISTEMAS DE RADAR

Os sistemas de radar são sistemas responsáveis por emitir ondas eletromagnéticas para uma área de interesse e através das ondas que são refletidas estimar informações acerca dos objetos presentes nessa área (RICHARDS; SCHEER; HOLM, 2010). Estas informações podem ser de qual material é composto o objeto, sua distância com relação à antena de transmissão e sua velocidade relativa ao sistema de radar, entre outras informações.

Os sinais de radar são ondas eletromagnéticas, transmitidas e recebidas por uma ou mais antenas transmissora e receptora, estes podem ser contínuos ou pulsados. Porém, para o processamento do sinal receptor é necessário primeiramente conhecer as características do sinal transmitido, como frequência, amplitude e fase.

Na figura 1, apresenta-se um sistema de radar hipotético no qual são mostrados os principais subsistemas que o compõem. Os componentes do sistema da figura 1 são o *transmitter* (transmissor), responsável por gerar o sinal a ser transmitido.

Figura 1 - Principais elementos de um sistema de radar.



Fonte: (RICHARDS; SCHEER; HOLM, 2010).

Em seguida apresenta-se a parte responsável por controlar o sinal na antena para transmissão ou recepção. Antes do componente receptor é utilizando uma chave de proteção, com o objetivo de proteger o circuito receptor do sinal de alta energia que é transmitido, uma vez que este é projetado para trabalhar com um sinal muito pequeno.

Ainda, este conjunto realiza o condicionamento do sinal, onde o primeiro passo é utilizar um LNA (*Low Noise Amplifier*) para reduzir a figura de ruído ao longo do circuito analógico do sinal e amplificar a potência do sinal. Em seguida, este é multiplicado pelo oscilador local, utilizado no transmissor, esta multiplicação permite que apareça um componente do sinal em uma frequência mais baixa, sendo esta a frequência intermediária.

Além disso, após esta etapa o sinal é amplificado e depois filtrado com o objetivo de eliminar os componentes de frequência indesejada, como também para propiciar que o conversor A/D possa digitalizar o sinal desejado. Por fim, o processador digital de sinal (*signal processor*) irá realizar as funções propriamente ditas do radar, sendo elas detectar e medir.

2.1 SINAIS DE RADAR

Os sinais de radar são ondas eletromagnéticas e são caracterizadas por frequência, fase e amplitude. A faixa de operação das ondas de um sistema de radar estão entre 3MHz e 110GHz, embora a maioria esteja entre 300MHz e 35GHz (RICHARDS; SCHEE; HOLM, 2010). A equação que rege a potência do sinal recebido pela antena receptora de um sistema de radar é descrita pela Equação (2.1) a seguir (BARTON, 2012).

$$P_R = \frac{P_T G_T G_R \lambda^2 \sigma}{(4\pi)^3 R^4} \tag{2.1}$$

Onde:

- P_R é a Potência Recebida em Watts;
- P_T é a Potência Transmitida em Watts;
- G_T é o ganho da Antena transmissora em Watts;
- G_R é o ganho da Antena Receptora em Watts;
- λ é o comprimento de onda em metros;

- σ é o RCS (*Radar Cross Section*), referente à área efetiva irradiada pelo objeto, é medido em m²;
- R é a distância entre o radar e o objeto em metros.

Dividindo a Equação (2.1) pela Potência do sinal do ruído, encontra-se a equação da relação sinal-ruído (SNR), mostrada na Equação (2.2) (BARTON, 2012).

$$\text{SNR} = \frac{P_T G_T G_R \lambda^2 \sigma}{(4\pi)^3 R^4 k T_s B L} \tag{2.2}$$

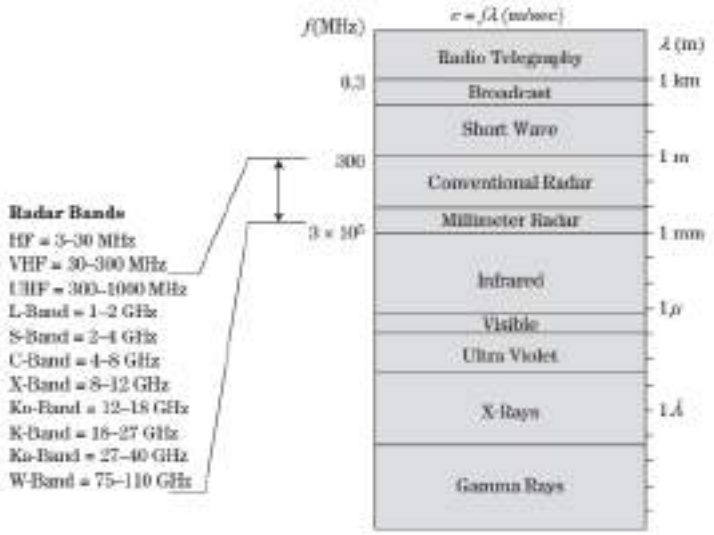
Onde:

- k é a constante de Boltzmann ($1{,}38x10^{-23}$ watt-segundos/K);
- T_s é a temperatura de ruído do sistema em Kelvin;
- L são as perdas adicionais no sistema;
- B é a banda de frequência do sistema receptor em Hz.

As Equações (2.1) e (2.2) são de grande importância para o desenvolvimento, posto que irão modelar o sistema de radar a ser construído, determinando as suas características. As equações expressam o maior valor da relação sinal-ruído através dos parâmetros do sistema, o mínimo valor da relação sinal-ruído que é necessário para a operação do radar e, portanto, o maior alcance para os parâmetros do radar (BARTON, 2012).

As Equações (2.1) e (2.2) são para um sistema de radar utilizando antenas direcionais, ou seja, antenas que enviam e recebem sinais em uma única direção. A figura 2 apresenta as frequências utilizadas pelos sinais de radar, assim como as suas bandas (RICHARDS; SCHEER; HOLM, 2010).

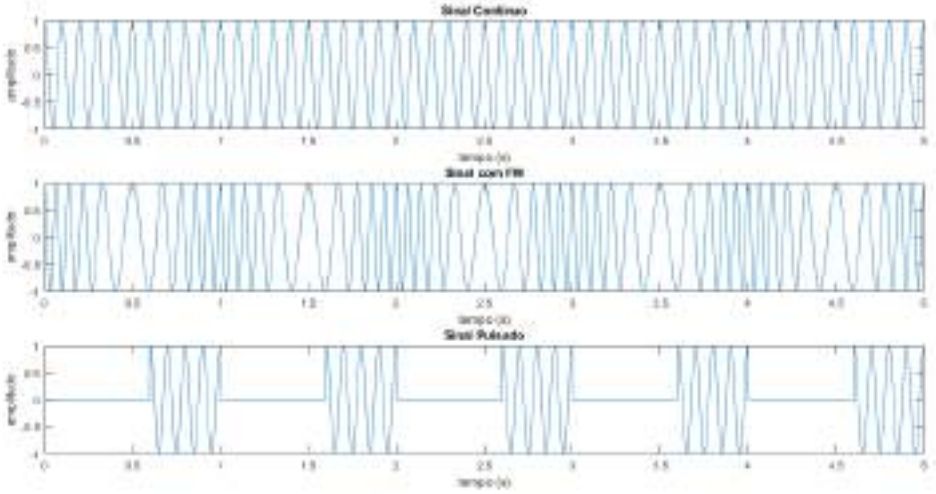
Figura 2 - Bandas de Frequência de sinais de Radar.



Fonte: (RICHARDS; SCHEER; HOLM, 2010).

O sistema de transmissão de radar pode utilizar vários tipos de sinal, podendo estes serem contínuos com uma frequência fixa no tempo, contínuos com frequência variando (Sinal FM) ou pulsados modulados a uma onda contínua. Na figura 3 tem-se três tipos de sinais utilizados em transmissores de radar.

Figura 3 - Exemplo de sinais transmitidos em sistemas de radar.



Fonte: (Do Autor, 2018).

Atualmente, os sinais contínuos não são mais utilizados pois os benefícios da utilização dos outros sinais apresentam mais vantagens em comparação com o custo de projeto, conforme apontado por Levanon e Mozeson (2004).

Para a realização das funções do radar o processamento de um sinal contínuo sofre mais com efeitos de ruído em relação aos outros sinais, acarretando em maiores dificuldades para as medições de distância e velocidade. A distância é calculada pela Equação (2.1), onde se observa que os valores da potência do sinal recebido são pequenos.

Por exemplo, para um transmissor com potência na ordem de 1000 W, com ganhos de 10 W nas antenas de transmissão e recepção, λ de 0,15 m (considerando um sinal de frequência de 2 GHz), um *Radar Cross* Section (RCS) para um veículo com $\sigma = 100$ m² e considerando um alvo a 100 m de distância, temos uma potência recebida de 7,5589 μW, que é um sinal com potência considerada pequena. Desta forma, acrescentando os efeitos de ruído, o sistema de radar com transmissão de sinal contínuo precisaria de um transmissor de potência muito elevada para aumentar a sua relação sinal-ruído.

Alguns sistemas de radar utilizam o sinal transmitido como um sinal com frequência variante no tempo, um exemplo dessa aplicação é o *CanRadar*, desenvolvido por Guimarães (2014). Nesta aplicação, a partir do apresentado por Guimarães (2014) para realizar a medição de distância, o sinal transmitido era modulado em frequência por um sinal modulante triangular. Por sua vez, quando recebido ele era multiplicado pelo sinal transmitido e por meio do *software* MATLAB© era processado e então apresentados os valores da distância medida.

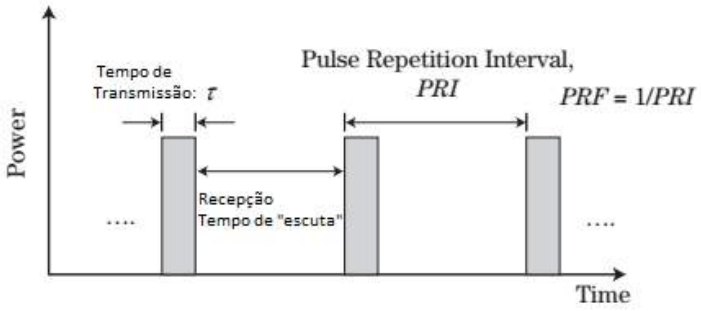
Os sinais pulsados são sinais a partir dos quais as ondas eletromagnéticas são transmitidas por um intervalo curto de tempo, durante o qual o receptor é isolado. Após a transmissão existe o tempo de "escuta", este intervalo serve para a antena receptora captar o sinal de retorno e é apresentado na figura 4 onde apresenta-se o PRF (*Pulse Repetition Frequency*), a frequência entre pulsos, e o PRI (*Pulse Repetition Interval*), o intervalo de tempo entre pulsos, e na figura 5 se mostra uma representação do sinal de retorno.

Observando as figuras 4 e 5, é possível então perceber que para que o sistema reconheça o sinal de retorno, este deve retornar durante o tempo estabelecido de "escuta". Isto é para evitar a ambiguidade na medição de distância, que é quando um alvo emite o eco com um tempo de retorno é maior que o tempo de "escuta".

A ambiguidade pode provocar cálculos errôneos no sistema, pois na prática esse eco que surge no receptor, é referente ao primeiro pulso. A figura 6 exemplifica esse efeito, onde o sinal de cor azul é referente ao transmissor, o de cor vermelha é a um alvo cujo eco está dentro do tempo de "escuta" e em verde representando um alvo cujo eco

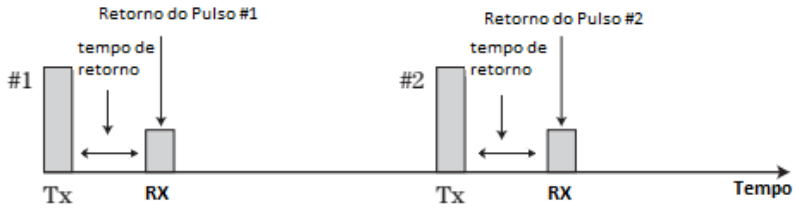
provoca ambiguidade na medição de distância, onde sua recepção se dá após a transmissão do segundo pulso.

Figura 4 - Representação do sinal PRF.



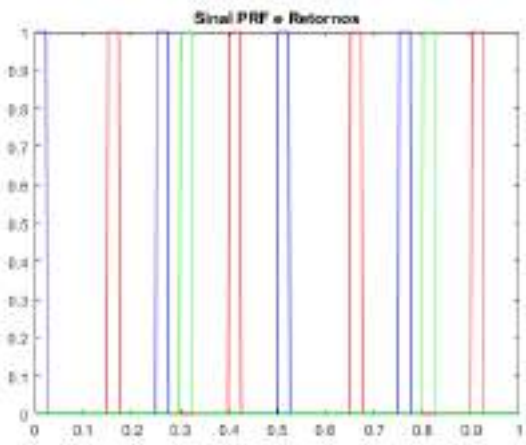
Adaptado de (RICHARDS; SCHEER; HOLM, 2010).

Figura 5 - Representação do sinal de retorno.



Adaptado de (RICHARDS; SCHEER; HOLM, 2010).

Figura 6 - Sinal PRF e retornos com e sem ambiguidade.



Fonte: (Do Autor, 2018).

Para calcular o alcance não ambíguo do sistema é utilizada a Equação (2.3) (RICHARD; SCHEER; HOLM, 2010).

$$R_{UA} = \frac{c}{2 * PRF} \tag{2.3}$$

Onde:

- R_{UA} é o alcance não ambíguo do sistema, em metros;
- PRF é a frequência dos pulsos do sinal, em Hertz;
- c é a velocidade da luz em m/s;

Esse efeito da ambiguidade com o alcance do sinal é de importância na hora do cálculo de efeito doppler, de modo que o controle é feito por PRF.

A maior vantagem da utilização do sinal pulsado é que ele apresenta maiores valores para a relação sinal ruído, de modo que quanto mais pulsos forem integralizados na transmissão do sinal, maior será o valor de SNR.

A razão para esse aumento na relação sinal-ruído é que por utilizar o sinal pulsado, o sistema obtém mais dados referentes ao alvo, assim quanto mais pulsos forem transmitidos mais dados do alvo irão ser obtidos pelo sistema. Sendo assim, com a transmissão de pulsos maior será o sinal desejado com relação ao ruído. Para a um sistema que utilize a transmissão de sinal pulsado, a nova relação de sinal-ruído é apresentada conforme a equação (2.4) (RICHARDS; SHCEER; HOLM, 2010).

$$\mathrm{SNR}_c(n_p) = n_p \cdot SNR(1) \tag{2.3}$$

Onde:

- n_p é o número de pulsos integrados no sinal;
- $\mathrm{SNR}_c(n_p)$ é a relação sinal ruído para o sinal pulsado;
- $\mathrm{SNR}(1)$ é a relação sinal ruído para o sinal de pulso único;

A potência de transmissão do sinal é um fator muito importante para as características do Radar, como pode-se observar pelas equações (2.1) e (2.2). A utilização de um sinal pulsado permite ao transmissor emitir um sinal com pico mais elevado,

possuindo assim um sinal durante a transmissão do pulso com mais energia, em comparação ao sinal contínuo.

Retornando à figura 3, que ilustra os exemplos de sinais contínuos e pulsado, pode-se afirmar que para manter uma potência média no sinal o sinal pulsado permite que se aumente a potência de pico do sinal, propiciando a emissão de um sinal com mais potência (energia) durante um intervalo de tempo. Pode-se então calcular o valor da potência média do sinal pulsado através da equação (2.4), e por meio da equação (2.5, tem-se o cálculo do *duty-cycle* de um sinal pulsado representado pela figura 4 (RICHARDS; SCHEER; HOLM, 2010).

$$P_{médio} = P_{Pico} * \ d_t = P_{Pico} * \tau * PRF \tag{2.4}$$

$$d_t = \tau * PRF \tag{2.5}$$

Onde:

- $P_{médio}$ é a potência média do sinal;
- P_{pico} é a potência de pico do sinal;
- d_τ é o *duty-cycle* do sinal;
- τ é a largura do pulso no tempo;
- PRF é a frequência do sinal pulsado;

3 PLATAFORMA DE DESENVOLVIMENTO

Um sistema digital é um sistema que possui sinais digitais na sua entrada que são processados e produz sinais digitais na sua saída. Os dispositivos são em sua maioria eletrônicos, embora existam sistemas digitais mecânicos, magnéticos e até pneumáticos (INGLE; PROAKIS, 2010).

Os sistemas digitais apresentam características mais atrativas, por serem mais simples de se projetar, permitirem o armazenamento da informação, e propiciarem então maior precisão no processamento, posto que, como apresentado por Ingle e Proakis (2010), a informação não varia durante o processo, é programável, sofre menores efeitos de ruído, e é possível a miniaturização obtendo mais blocos lógicos em dispositivos com menor área

Os avanços da integralização em enorme escala (VLSI – *Very Large Scale Integration)* tornaram possível o desenvolvimento de chips que podem ser programáveis por um usuário para implementar diferentes circuitos lógicos (Ferdjallah, 2011).

Os chips programáveis são conhecidos por PLD (*Programmable Logic Device*), em que sua configuração possui diversas chaves programáveis e para a programá-lo o usuário deve apenas configurar essas chaves para performar a função digital desejada, a configuração é feita através de linguagem de descrição de *hardware* (HDL) (Ferdjallah, 2011).

Desse modo, os dispositivos PLD são uma excelente ferramenta para prototipação de sistemas digitais e chips padrão. Ferdjallah (2010) comenta que a principal desvantagem deles é que a performance do sistema neles implementado não é a melhor em comparação com um chip funcional equivalente. Isto se dá devido ao circuito implementado no PLD não ser o circuito com os componentes lógicos desejados, mas uma realização com os blocos lógicos internos do próprio PLD. Os principais dispositivos PLD são:

- SPLD – *Simple Programmable Logic Device*;
- PAL – *Programmable Array Logic*;
- PLA – *Programmable Logic Array*;
- GAL – *Generic Array Logic*;

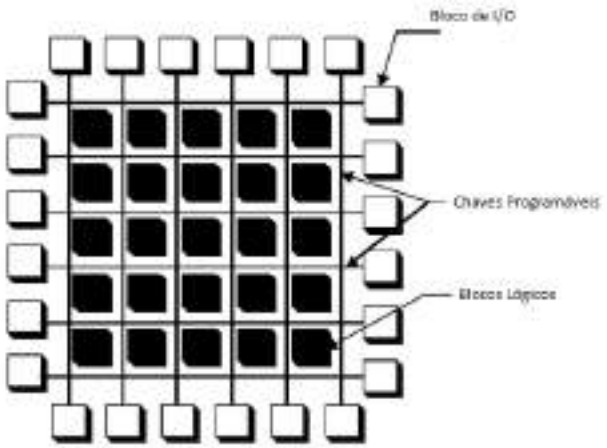
- CPLDs – *Comples Programmable Logic Devices*;
- FPIC – *Field Programmable Inter-connect*;
- FPGA – *Field Programmable Gate Arraye*

Os diferentes tipos de PLDs variam na arquitetura interna para implementação dos blocos lógicos e interconexão das chaves programáveis. Dentre os PLDs, os FPGAs possuem o maior número de portas, o que permite acomodar sistemas maiores que os SPLDs e CPLDs poderiam implementar (FERDJALLAH, 2011).

3.1 FPGA

O FPGA possui internamente um *array* bidimensional de células genéricas e chaves programáveis, contornado por um sistema de blocos para entrada e saída do sinal. Esta configuração é observada na figura 7.

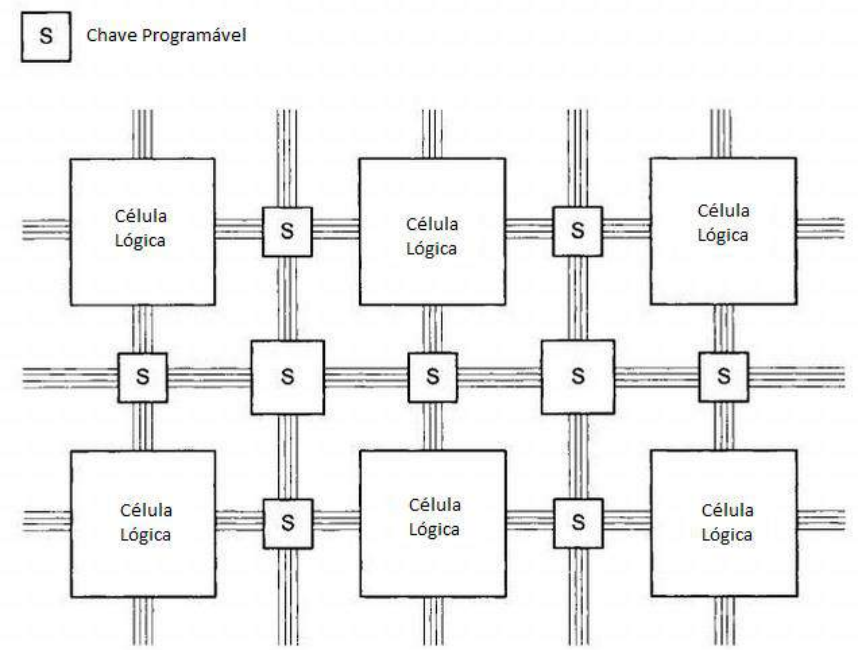
Figura 7 - Estrutura interna de um FPGA.



Adaptado de (FERDJALLAH, 2011).

Por sua vez, a figura 8 ilustra a estrutura conceitual do dispositivo, apresentando as ligações entre os blocos lógicos e o sistema de chaves programáveis.

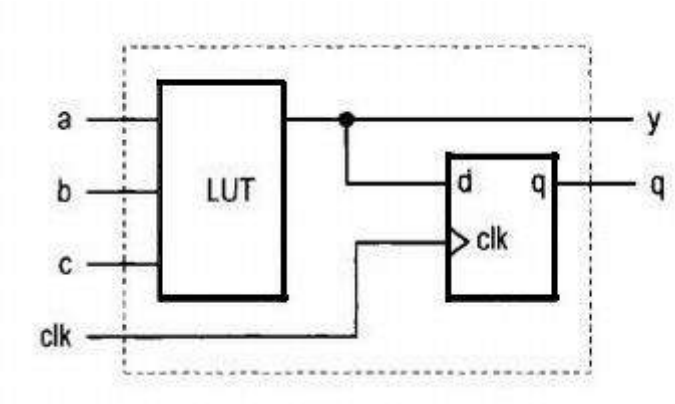
Figura 8 - Estrutura conceitual de um FPGA.



Adaptado de (CHU, 2008).

As células lógicas contêm um pequeno circuito combinacional configurável em conjunto com um registrador flip-flop tipo D (DFF). O método mais comum para implementar o circuito combinacional configurável é utilizar LUTs (*Look-Up Table*), que são blocos capazes de reproduzir uma função booleana. A figura 9 mostra o diagrama conceitual de uma célula lógica.

Figura 9 - Diagrama conceitual de uma célula lógica.



Fonte: (CHU, 2008).

A programação de um FPGA é feita com uso de uma linguagem de descrição de *hardware* (HDL). Entretanto, um FPGA não é programado através de código, e sim de uma descrição de circuito digital, essa descrição irá configurar as LUTs e programar as conexões das chaves.

Uma das linguagens HDL mais utilizadas é a *Verilog* padronizada pela norma IEEE 1364, ela é utilizada para criação de sistemas eletrônicos e foi desenvolvida para ser simples, intuitiva e efetiva em muitos níveis de abstração com um formato textual para softwares de design permitindo verificação, análise no tempo e em teste e síntese. Por essas razões a *Verilog* foi adotada por vários projetistas de circuitos integrados (IEEE Standards Association, 2005). Há, contudo, outras linguagens de descrição de *hardware*, como *SystemVerilog* e VHDL e estas também são padronizadas pelas normas da IEEE.

Existem diversas fabricantes de chips FPGA, dentre as quais destacam-se a *Altera Corporation, Xilinx Inc., Lattice Semiconductor, Cypress Semiconductor, Atmel, Actel, Lucent Technologies e QuickLogic*. Cada uma destas desenvolvedoras disponibilizam diferentes chips FPGA e cada um com características diferentes, permitindo a melhor aplicação de algumas funções lógicas.

3.2 BASYS 3

A placa Basys 3 é uma plataforma para desenvolvimento de circuito digital utilizando o chip *Artix*®-7 FPGA da *Xilinx*®, a Basys 3 é fabricado pela *Digilent*®. Ela permite que seja desenvolvido de circuitos combinacionais a circuitos sequenciais complexos como processadores embutidos e controladores (DIGILENT, 2017).

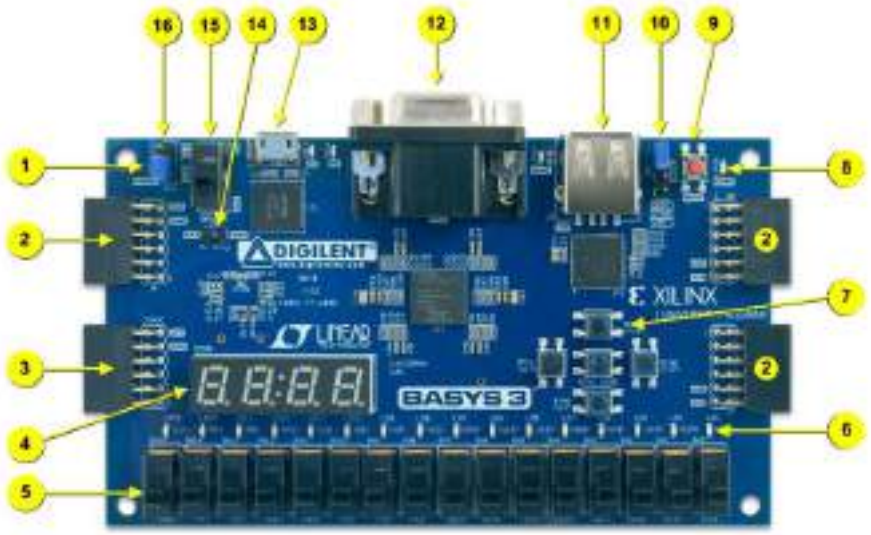
Na figura 10 tem-se a imagem da vista superior da placa onde pode-se observar seus componentes numerados conforme indicação na tabela 1. As descrições técnicas são:

- 33.280 células lógicas em 5200 *slices*, cada *slice* contém quatro LUT de 6 entradas e 8 flip-flop;
- 1.800Kbits de bloco RAM;
- 5 tipos de manuseio de *clock*, cada um com um PLL (*Phase-Locked Loop*);
- 90 slices de DSP (*Digital Signal Processing*);
- *Clock* interno de 100 MHz;

- Conversor Analógico-Digital interno (XADC).

Ainda sobre a placa, pode-se dizer que esta possui 16 chaves manuais, 4 displays de 7-segmentos, uma saída VGA de 12 bits, 16 LEDs, três portas Pmod, 5 botões, uma Pmod para o XADC, uma porta USB-JTAG para programação com FPGA e comunicação, entre outros.

Figura 10 - Vista Frontal da Placa Basys 3.



Fonte: (DIGILENT, 2017).

Tabela 1 – Especificação dos Componentes da Placa Basys 3.

Número	Descrição do Componente	Número	Descrição do Componente
1	LED de Ligado ou Desligado	9	Botão Reset
2	Portas Pmod	10	Jumper para o modo de programação
3	Porta Pmod para sinal analógico (XADC)	11	Conector USB
4	Quatro Displays de 7-segmentos	12	Conector VGA
5	Chaves (16)	13	Porta USB para JTAG e UART
6	LEDs (16)	14	Conector de alimentação externa
7	Botões (5)	15	Chave de On/Off
8	LED de informação de programação FPGA	16	Jumper para selecionar a entrada de alimentação

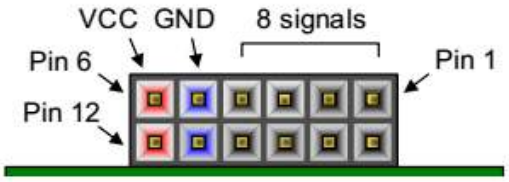
Fonte: (DIGILENT, 2017).

A Basys 3 pode ser alimentada por uma fonte externa que deve ser entre 4,5 V e 5,5V com ao menos 1A de corrente que pode ser conectado na indicação 14 da figura 10, sendo devidamente ajustado pelo jumper apontando em 16. Mais comumente utilizado, a alimentação também pode ser feita pela entrada USB, indicado por 13.

O dispositivo possui um oscilador interno de 100MHz conectado no pino W5, internamente a ele. As portas Pmod, apresentadas na indicação 2, presente na figura 10, possuem 2 pinos para alimentação Vcc (3.3V) e 2 pinos para definição do GND, juntos com mais 8 portas de entrada ou saída a serem utilizadas.

Na figura 11, ilustra-se os pinos de uma porta Pmod. O Pmod XADC apresenta as mesmas características das outras portas, porém, possuindo um chip conversor analógico-digital interno que pode ser configurado de acordo com a necessidade do desenvolvedor.

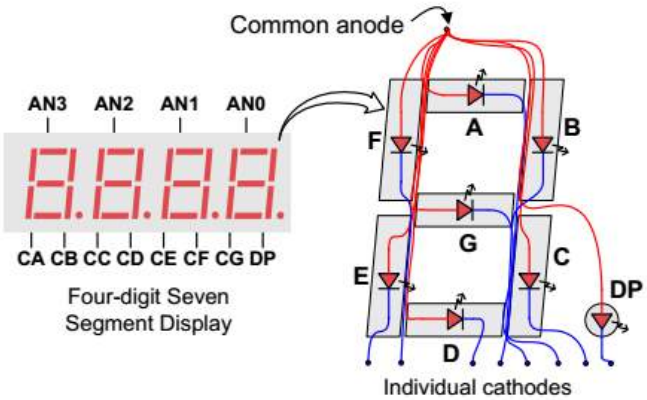
Figura 11 - Apresentação das portas Pmod.



Fonte: (DIGILENT, 2017).

Em se tratando dos displays de 7-segmentos, apontado por 4 na figura 10, possuem o ânodo comum a todos os displays e os cátodos dos segmentos similares são comuns. Na figura 12 mostra-se as ligações internas de um display de 7-segmentos.

Figura 12 - Representação da ligação dos LEDs dos displays de 7-segmentos.



Fonte: (DIGILENT, 2017).

Para programar a Basys 3 é utilizado o software Vivado® que possui uma versão fornecida gratuitamente pela *Xilinx*®.

3.3 VIVADO

O Vivado é um software de HLS (*High Level Synthesis*) produzido pela *Xilinx*® para síntese e análise de projetos em linguagem de descrição de *hardware* (HDL). O Vivado permite a análise do circuito desenvolvido através de simulações comportamentais do código escrito, funcionais ou por *timing*. Estas permitem ao usuário a análise dos sinais do sistema, verificando funcionamento pós-síntese e pós-implementação, calculando variáveis de tempo inferidas nos circuitos integrados, como tempo de atraso nos blocos lógicos e tempo de propagação do sinal, auxiliando o projetista a saber o comportamento real do circuito desenvolvido (XILINX, 2017a).

A *Xilinx*® dispõe de diversos modelos do Vivado, sendo cada qual especializada para uma função. Atualmente, os modelos do Vivado disponíveis são:

- Vivado HL *Design Edition*: inclui o *Partial Reconfiguration* – que é uma função que permite modificar blocos lógicos dinamicamente – e o Vivado *High Synthesis Level*;
- Vivado HL *System Edition*: Todas as funções do modelo *Design Edition* com acréscimo o gerador de sistemas para DSP (*Digital Signal Processing*);
- Vivado HL *WebPACK*™ *Edition*: Versão gratuita e com limitação de dispositivos da Vivado HL *Design Edition*;
- Vivado *Lab Edition*: Versão para ambientes de laboratórios com maior facilidade na instalação.

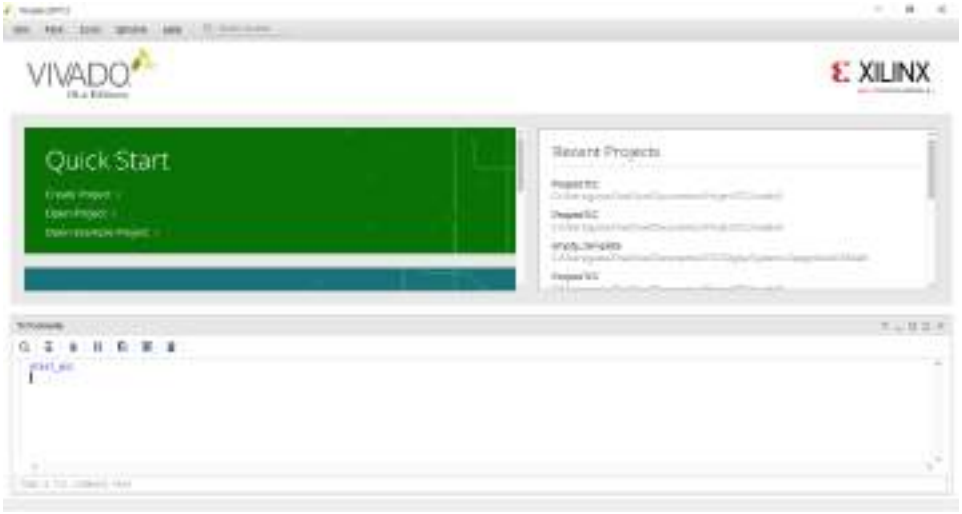
Neste trabalho foi utilizado o Vivado HL *WebPACK*™ *Edition*, por ser gratuito e de o dispositivo utilizado ser suportado.

3.3.1 CRIAÇÃO DE PROJETO COM O VIVADO

A figura 13 mostra a página inicial do Vivado, onde se pode criar um novo projeto, abrir um projeto existente ou abrir um projeto de exemplo. Ainda, tem-se também o *TCL Console* que é uma área onde se pode realizar os comandos e funções do Vivado, esta é semelhante ao existente no software AutoCAD®.

O Vivado opera os circuitos desenvolvidos no nível de RTL (*Register Tranfer Level*) que significa que o circuito desenvolvido nesta plataforma estará no nível de registradores e transferência de dados pelos blocos lógicos, ou seja, o Vivado não analisa nem implementa um circuito no nível de transistor.

Figura 13 - Página Inicial do Vivado.



Fonte: (Do Autor, 2018).

Para criar um novo projeto é preciso definir o nome deste e o local onde serão salvos os módulos e todos os arquivos referentes a ele. Em seguida deve-se escolher o tipo do projeto, se será um RTL (*Register Transfer Level*), em que se adiciona fontes ou módulos no projeto, configura blocos de sistemas pré-prontos, realiza análise, síntese e implementação do RTL e planejamento do design e análise.

Outra opção é o projeto de pós-síntese, onde se pode adicionar fontes ao projeto, observar os recursos do dispositivo e realizar análise, planejamento e implementação. A terceira opção é o *I/O Planning*, que permite ao usuário interatividade ao explorar, visualizar, atribuir e validar portas de entrada e saída (I/O) e clock lógico. Por último, o usuário pode escolher entre importar um projeto ou de utilizar um exemplo disponível (XILINX, 2017b). A figura 14 apresenta a janela das opções para o tipo de projeto.

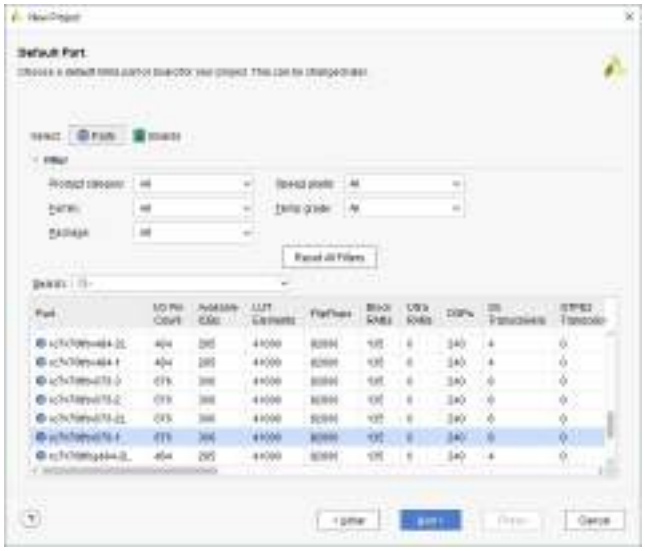
Figura 14 - Etapa da escolha do tipo de projeto.

Fonte: (Do Autor, 2018).

Para o desenvolvimento e a implementação do projeto, deve-se escolher a opção de *RTL Project*, podendo optar entre adicionar os arquivos fonte neste momento (módulos do projeto em *Verilog*) ou mais adiante. Seguindo, deve-se selecionar qual o dispositivo FPGA que será utilizado na implementação, essa escolha pode ser feita utilizando um chip FPGA padrão, pois esse dispositivo pode ser modificado mais adiante.

Na figura 15 é mostrada a janela em que se escolhe o dispositivo, a placa Basys 3 possui um chip FPGA Artix-7 desenvolvido pela Xilinx, o valor das suas especificações são: XC7A35T-1CPG236C (DIGILENT, 2017).

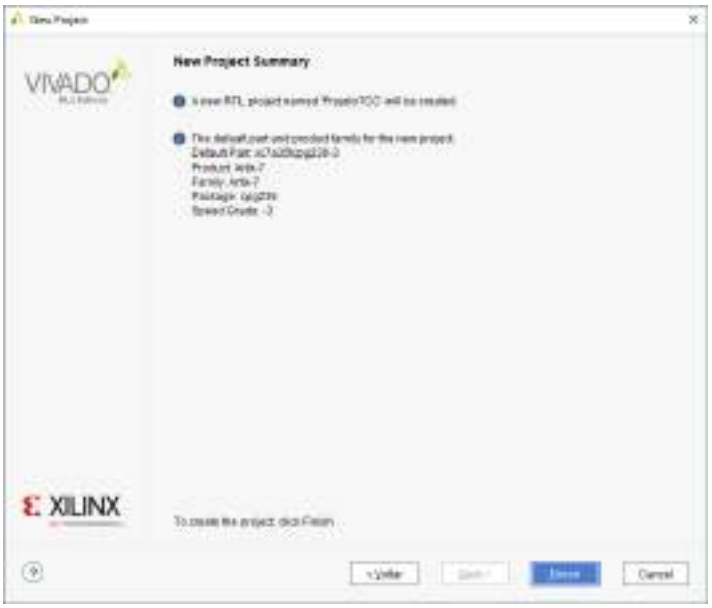
Figura 15 - Escolha do dispositivo FPGA utilizado no projeto.



Fonte: (Do Autor, 2018).

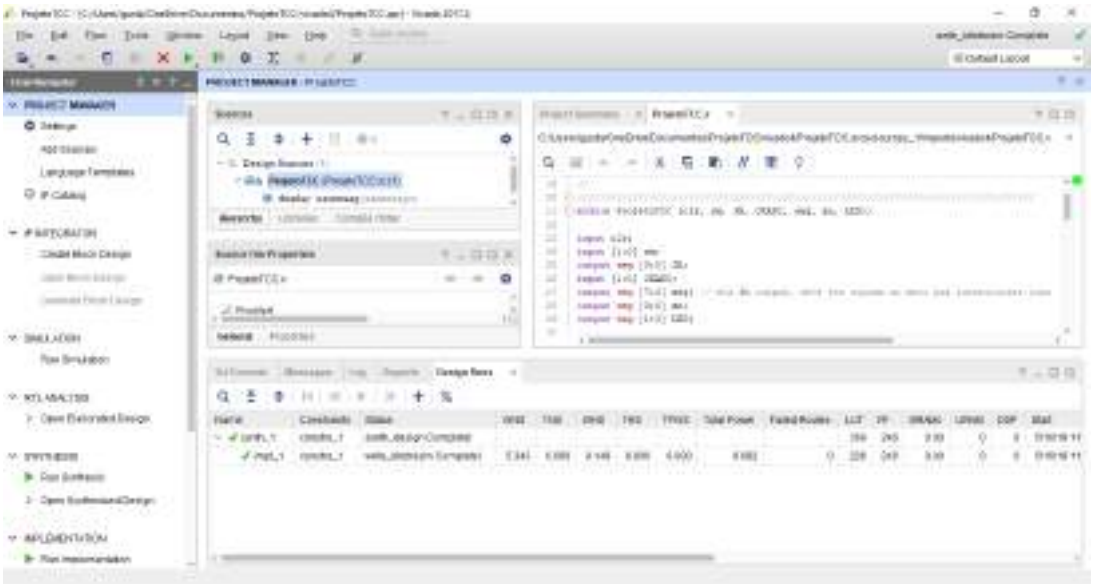
De acordo com o *datasheet* do chip *Artix-7*, escolhe-se no Vivado a opção de xc7a35tcpg236-3, pois ele representa o chip com sua melhor performance (XILINX, 2018). Finalizado as especificações do projeto, é apresentado um sumário com as escolhas registradas, este é mostrado na figura 16. Segue-se então para a página do gerenciamento de projeto, esta apresentada na figura 17.

Figura 16 - Sumário do novo Projeto.



Fonte: (Do Autor, 2018).

Figura 17 - Página de Gerenciamento do Projeto.

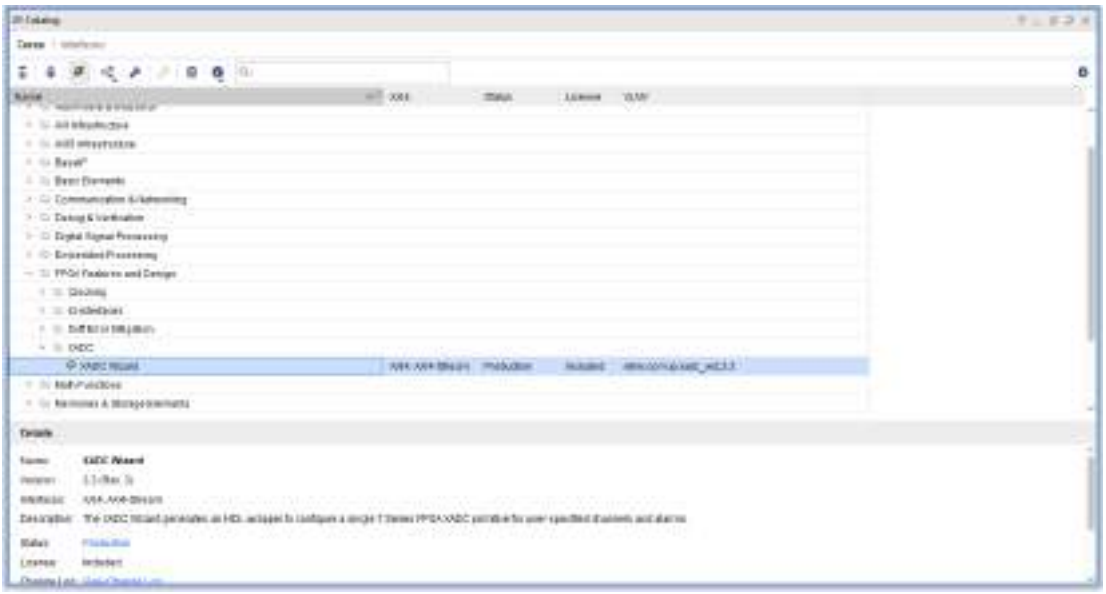


Fonte: (Do Autor, 2018).

3.3.2 GERENCIAMENTO DE PROJETO

No gerenciamento de projeto é possível adicionar módulos em *Verilog* do circuito a ser analisado e instanciar no projeto blocos a partir do *IP Catalog*, que é um catálogo com funções prontas para implementação e que permitem configuração. Estes blocos, por sua vez, permitem ao usuário utilizar funções no circuito a partir de componentes e blocos lógicos já existentes na placa, como o do conversor A/D, o de multiplicação para DSP, o de cálculo de transformadas, implementar memórias, módulos de comunicação dentre outros. Na figura 18, tem-se o *IP Catalog* com detalhes do módulo do XADC Wizard (Conversor A/D da placa).

Figura 18 - Apresentação do *IP Catalog.*



Fonte: (Do Autor, 2018).

Para o projeto em HDL, o Vivado realiza prontamente uma verificação de sintaxe do código *Verilog*, reconhecendo quais são os módulos principais e quais apresentam funções secundárias, dessa forma o usuário pode observar automaticamente erros de sintaxe no código.

A etapa de síntese é responsável por analisar o circuito e verificar erros, ao término desta é apresentado o relatório onde se pode observar os erros, *warnings* e *critical warnings*, do circuito desenvolvido, identificando os elementos e as linhas no código referente.

Ainda, nas verificações da síntese é possível identificar se todos os elementos estão conectados à saída, se há *loop* nos blocos combinacionais – onde a entrada de um

bloco esteja conectada na sua própria saída – se há elementos com duas entradas em conflito, entre outros. Para realizar a síntese é preciso que o código não possua erros de sintaxe, a sua inicialização é simples, dá-se apenas clicando no elemento "*run synthesis*", conforme visto na parte esquerda da figura 16.

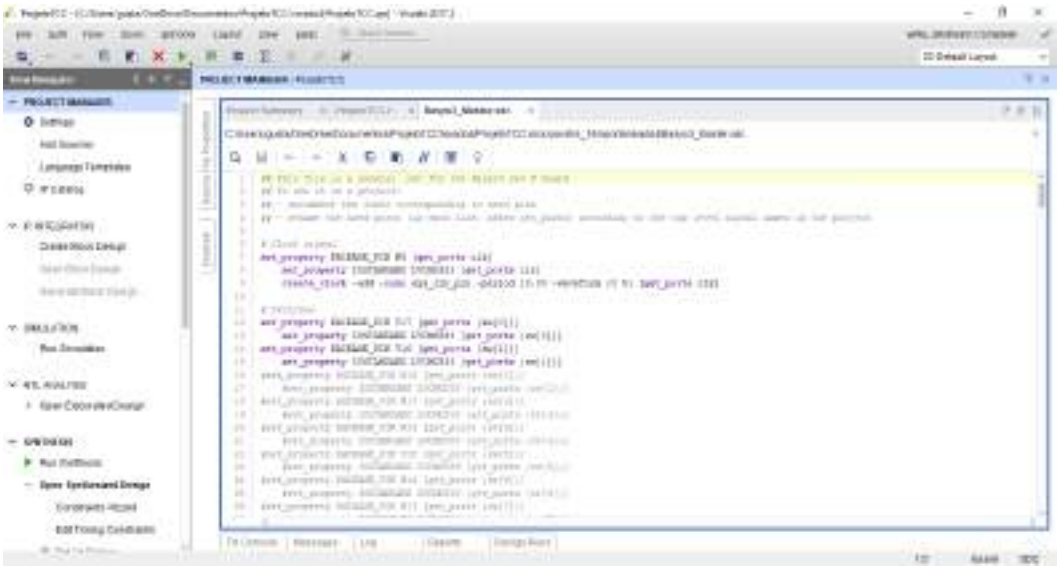
Após a etapa de síntese é realizada a implementação. Esta fase é responsável por gerar a programação das chaves e das células lógicas dentro do FPGA, sendo responsável por conectar as entradas e saídas do circuito descrito com HDL com os respectivos pinos na placa. Para iniciar a implementação é preciso clicar em "*run implementation*", mostrado no lado esquerdo da figura 16.

Para realizar as conexões é necessário a utilização de um arquivo de *constraints* que possui as informações acerca das entradas e saídas do circuito lógico e indicam em quais pinos internos da placa essas ligações serão feitas. Por exemplo, para a placa Basys 3, o pino do clock é o W5. A figura 19 mostra o início do arquivo de *constraints* da placa, este arquivo é fornecido pela *Digilent*, sendo necessário alterar o nome das ligações para os nomes das entradas e saídas do circuito desenvolvido.

No arquivo de *constraints* os pinos que não serão implementados são marcados como comentário, para isso é utilizado a sintaxe de "#", como pode-se observar na figura 19.

Ao término da implementação o usuário pode programar a placa com o sistema implementado, para essa programação o Vivado gera o arquivo de *bitstream* que converte a implementação em dados para a programação das chaves e dos LUTs do chip FPGA.

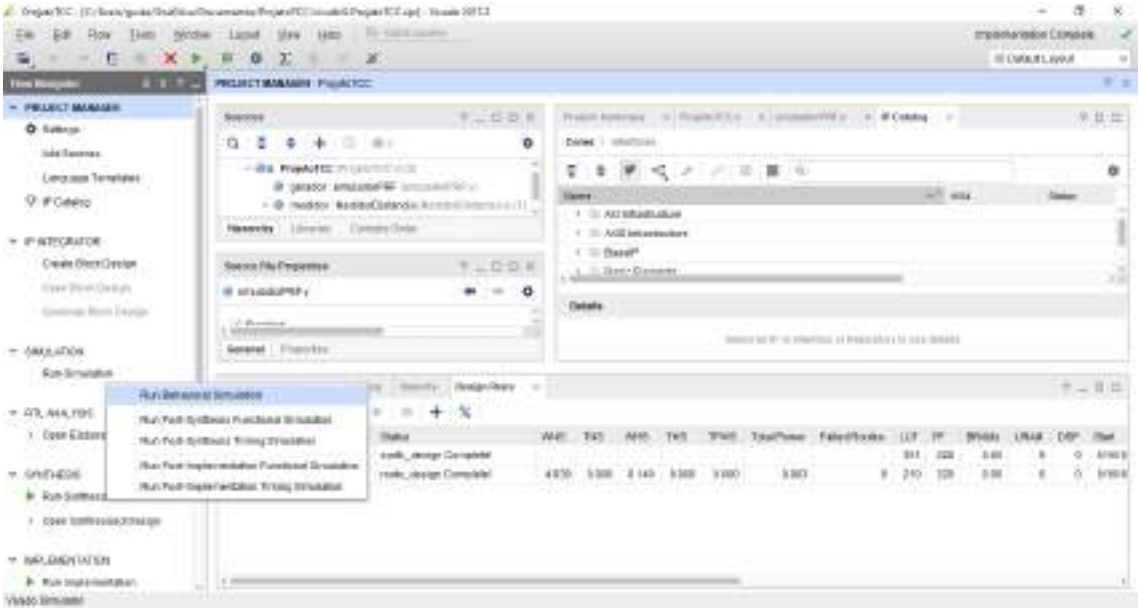
Figura 19 - Arquivo de *Constraints* da Basys 3.



Fonte: (Do Autor, 2018).

Com o sistema sintetizado e implementado, podem ser realizadas as simulações do sistema, estas podem ser apenas comportamental – simula o projeto não sintetizado – ou analisar o funcionamento do circuito pós-síntese ou pós-implementação, ou simulação levando em consideração os atrasos dentro dos blocos lógicos. A Figura 19 mostra onde se escolhe o tipo de simulação a ser realizada.

Figura 20 - Tipos de Simulação no Vivado.



Fonte: (Do Autor, 2018).

3.3.3 *TESTBENCH*

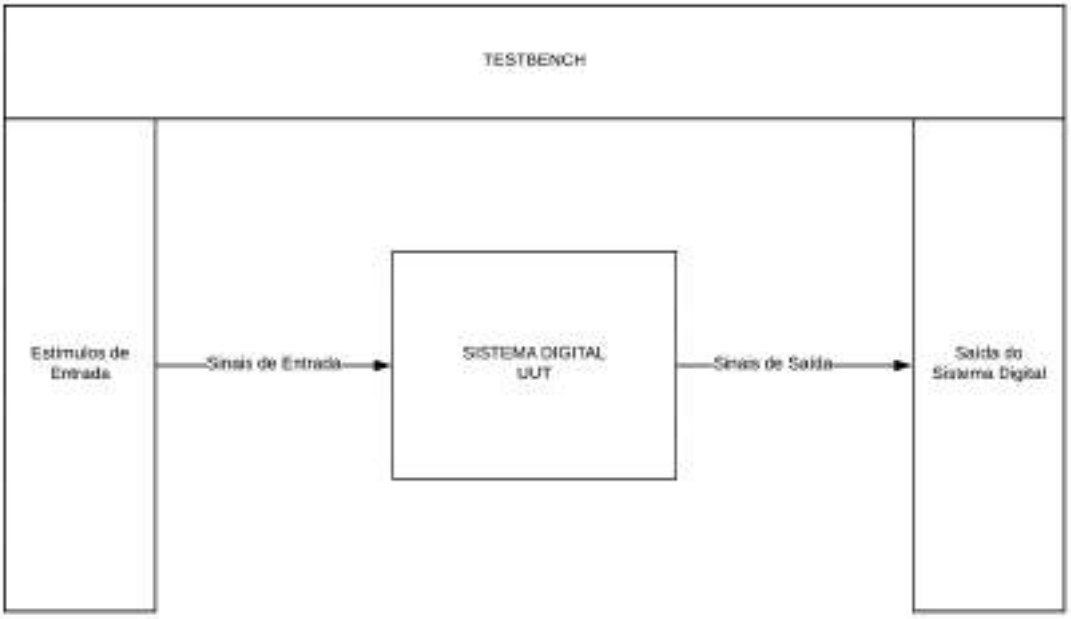
Para a simulação é preciso utilizar um arquivo de *testbench*, este pode ser um circuito descrito em HDL ou um programa em linguagem de programação. A função deste é estimular sinais de entrada no projeto desenvolvido e analisar as suas saídas.

No *testbench* podem ser escolhidas várias formas de análise dos sinais de saída do sistema, esta análise pode ser feita por comparação com programas ou outros *softwares* de simulação ou emitir as formas de onda e o usuário observar a saída para verificar o funcionamento do circuito.

Sistema digitais complexos, que envolvam muitos módulos, utilizam métodos para a simulação com comparação de resultados obtidos em outros *softwares*, muitas vezes implementando no dispositivo FPGA o *testbench* para testar a prototipação, embora isso possa acarretar em desempenho do circuito a ser testado.

Devido a sua importância, as desenvolvedoras de *hardware* possuem equipes inteiras destinadas a apenas confecção do *testbench*, pois esse arquivo irá validar o funcionamento do sistema, como também localizar falhas. A figura 21 ilustra o esquemático de um *Testbench*, onde UUT é a sigla para *Unity Under Test* – Unidade sob Teste.

Figura 21 - Esquemático de um *Testbench*.



Fonte: (Do Autor, 2018).

4 IMPLEMENTAÇÃO

Nesta seção será detalhada a etapa de implementação que foi realizada utilizando a ferramenta do Vivado. Foi desenvolvido um sistema digital para medir distância a partir do eco de sinais PRF de Radar. Para os sinais PRF foi descrito um circuito digital capaz de gerar o sinal do transmissor e três sinais emulados do receptor.

Os dois sistemas desenvolvidos foram simulados utilizando a simulação do tipo *timing* pós-implementação, que utiliza os atrasos nos registradores e calcula o tempo de propagação de sinal no sistema.

Foi projetado para possuir dois sistemas digitais distintos, o primeiro circuito é o emulador de sinal PRF e o segundo é o circuito digital capaz de medir distância.

4.1 EMULADOR DO SINAL PRF

Conforme descrito no item 2.1, sabe-se que o sinal mais utilizado para radares é o sinal PRF. Em um sistema de radar esse sinal é modulado com um sinal contínuo senoidal gerando uma onda senoidal pulsada, como visto na figura 3.

Em um sistema de radar real, existiriam os módulos de transmissão do sinal estes responsáveis por modular o sinal pulsado de saída do FPGA com um sinal senoidal contínuo gerado por um oscilador. O sinal seria então amplificado e transmitido por uma antena transmissora, percorreria o espaço, incidiria com um objeto e então uma parte do sinal seria irradiada.

Esse pequeno retorno seria captado pela antena receptora, passaria por um condicionamento de sinal similar ao apresentado anteriormente na figura 1, e então seria introduzido na entrada da placa FPGA, onde se iria realizar o processamento da informação captada.

Conforme apresentado anteriormente nas figuras 4 e 5, existe um tempo após a transmissão do pulso que é reservado para o recebimento do sinal, conforme a definição da velocidade na equação (4.1), temos que com a equação (4.2) podemos calcular a distância medindo o intervalo de tempo entre os pulsos.

$$v = \frac{\Delta S}{\Delta T} \tag{4.1}$$

Onde:

- v é a velocidade em m/s;
- ΔS é a distância em m;
- ΔT é o intervalo de tempo em segundos;

$$D = \frac{c * \Delta T}{2} \tag{4.2}$$

Observa-se que $c = 3x10^8 m/s$, velocidade da luz e D é a distância calculada. O valor é dividido por 2, por que a onda percorre duas vezes a distância até o objeto, uma na "ida" e outra na "volta".

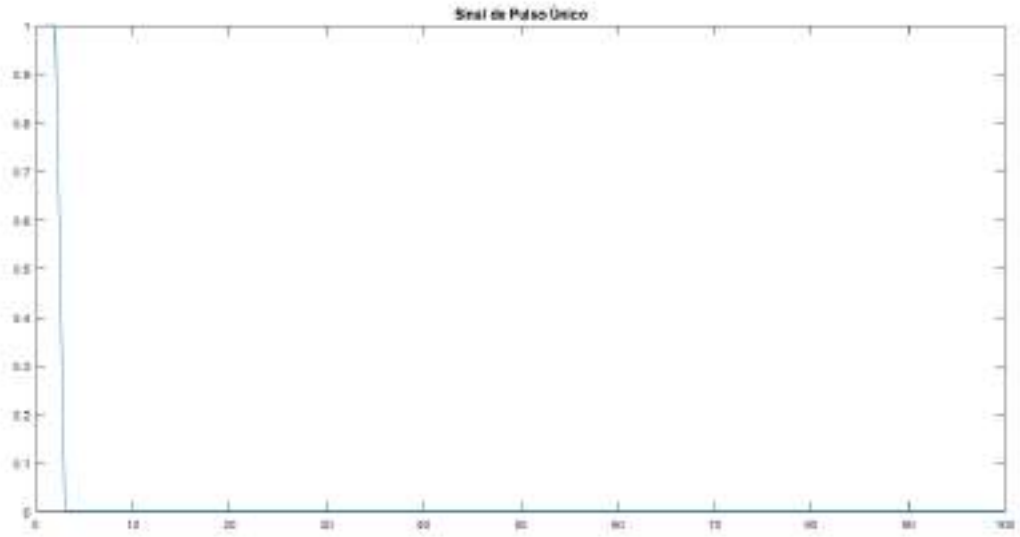
Foi decidido utilizar um sistema emulador para gerar os sinais de transmissão e recepção ao invés de construir um sistema de radar físico e então aplicar o sistema medidor de distância. Isto se deu devido ao circuito analógico de transmissão e o circuito condicionador de sinais utilizarem componentes específicos RF, estes não disponíveis nos laboratórios da universidade.

4.1.1 FUNCIONAMENTO DO CIRCUITO

Como o sistema utilizado é digital, o sinal gerado também será digital, sendo assim para gerar o sinal foram digitalizados cem valores de um sinal de pulso único, com *duty cycle* de 2% e armazenado dentro do programa em *Verilog*, "pwm" foi o nome dado ao sinal digitalizado e gravado no programa. Logo, para apresentar na saída um sinal pulsado foi desenvolvido um contador de 0 a 99 que percorre o sinal digitalizado e em cada instante do *clock* de 100MHz leva para a saída um ponto do sinal digitalizado para a porta Pmod JA, da placa. O sinal digitalizado pwm é apresentado na figura 22.

O contador do sinal transmissor inicia em zero, portanto a sinal de saída apresenta o valor gravado em pwm[0], no próximo instante será transmitido o valor pwm[1], seguindo até 99, quando o contador é zerado e retorna ao valor 0.

Figura 22 - Sinal digitalizado de pulso único.



Fonte: (Do Autor, 2018).

Para emular o sinal recebido pela antena receptora é necessário apenas criar um atraso no sinal e aplicá-lo à saída do FPGA, através de um dos pinos da porta JA. Para esse processo é iniciado o valor do seu contador em um ponto diferente de 0, deste modo, esse contador irá ser iniciado com um avanço em relação ao contador do sinal do transmissor, produzindo na saída do sistema um sinal deslocado no tempo.

Conforme dito na seção 3.2, a placa FPGA possui um clock interno de 100 MHz, que é o clock utilizado neste projeto. Os valores de início dos contadores do emulador de sinais receptores são de 4, 32 e 90, estes valores indicam que os contadores irão iniciar a emissão do sinal para a saída nos pontos pwm[4], pwm[32] e pwm[90], respectivamente.

Como o pulso está presente no início do sinal, conforme apresentado na figura 22, o sinal emulado iniciando no ponto pwm[4] irá apresentar o maior valor de atraso, posto que neste ponto o valor do sinal é 0 e assim seguirá até o contador chegar 99 e ser zerado. Quando for transmitido o pulso, o sinal emulado iniciando no ponto pwm[90] irá apresentar o menor valor de atraso, pois quando transmitido para a saída da placa ele irá iniciar a transmissão do pulso primeiro, em comparação aos outros dois sinais.

Pode-se calcular o tempo de atraso que estes sinais possuem através da equação (4.3). Os valores desses atrasos foram escolhidos de forma arbitrária, é necessário atentar que estes sinais deslocados no tempo representam objetos colocados a uma certa distância do radar, porém, esses objetos não existem simultaneamente, ou seja, o programa do

sistema de radar não medirá a distância dos três objetos ao mesmo tempo, apenas a distância de um alvo por vez.

$$T = \frac{100 - \ Início\ do\ contador}{frequência\ do\ clock} \tag{4.3}$$

A frequência do clock é de 100 MHz, sendo assim, temos na tabela 2 os valores do atraso em nano segundos para os sinais simulados desenvolvidos.

Tabela 2 – Atraso no Tempo entre o sinal transmitido e o sinal recebido.

Início do Contador	Atraso em ns
4	960
32	680
90	100

Fonte: (Do Autor, 2018).

O desenvolvimento do projeto foi realizado de forma a se possuir duas chaves de controle, SW[0] e SW[1], que podem ser vistas item 5, retornando à figura 10. A chave SW[0] é utilizada para a função de reset, responsável por apagar os dados de todos os registradores, propiciando que o circuito reinicie com os valores definidos como padrão. Por outro lado, a chave SW[1] é utilizada para iniciar os contadores e gerar na saída os sinais emulados produzidos no circuito desenvolvido.

4.1.2 SIMULAÇÃO DO EMULADOR

Para a simulação, foi desenvolvido o código do *testbench* utilizando a linguagem de descrição de *hardware Verilog*, o que permitiu criar um circuito que circunda o sistema digital desenvolvido neste trabalho. O código do *testbench* consistiu de instanciar o circuito emulador de sinais, gerar o estimulo de clock e controlar as chaves de controle simulando os usos das chaves SW[0] e SW[1]. Para verificar as saídas do circuito gerador de sinais foi preferido observar as formas de onda dos sinais do circuito.

Nas figuras 23 e 24 são apresentados os resultados obtidos através da simulação por *timing* pós-implementação no Vivado. Nelas são observados os sinais descritos na tabela 3. Essa simulação leva em conta os atrasos do sinal nos registradores e blocos

lógicos do dispositivo. Pode ser visto na figura 23 que os sinais iniciam com um tempo de atraso de 0,1 ns.

Ainda na figura 23 tem-se a apresentação dos contadores nas posições que foram descritas anteriormente, sendo o contador “i” para o sinal transmissor (iniciando em 0), o contador “j” para o primeiro sinal emulado (iniciando em 4), o contador “k” para o segundo sinal emulado (iniciando em 32) e o contador “l” referente ao terceiro sinal emulado (iniciando em 90)

Tabela 3 – Sinais Apresentados na simulação.

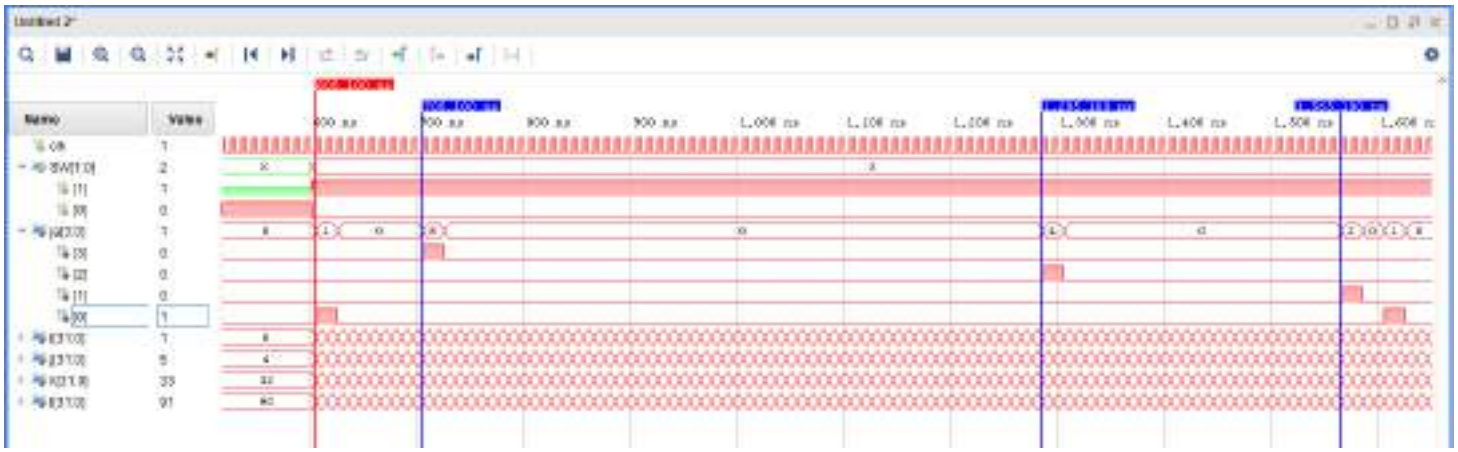
Sinal		Utilidade
clk		clock do sistema - 100MHz
SW	SW[0]	Chave de Controle Reset
	SW[1]	Chave de Comando da Transmissão dos sinais
JA	JA[0]	Porta de saída do sinal simulado do transmissor
	JA[1]	Porta de saída do sinal emulado do receptor com atraso de 960ns
	JA[2]	Porta de saída do sinal emulado do receptor com atraso de 680ns
	JA[3]	Porta de saída do sinal emulado do receptor com atraso de 100ns
i		Contador do sinal simulado do transmissor
j		Contador do sinal simulado do receptor com atraso de 960ns
k		Contador do sinal simulado do receptor com atraso de 960ns
l		Contador do sinal simulado do receptor com atraso de 100ns

Fonte: (Do Autor, 2018).

Figura 23 - Detalhe do início da geração, quando a chave SW[0] é zerada e a transmissão é iniciada SW[1].

Fonte: (Do Autor, 2018).

Figura 24 - Detalhe dos sinais simulados com marcações de tempo.



Fonte: (Do Autor, 2018).

A partir da figura 24, é construída a tabela 4, que apresenta as marcações de tempo do início do pulso do sinal transmitido e dos sinais emulados inicial.

Tabela 4 – Análise dos atrasos dos sinais na Figura 13.

Sinal	Tempo (ns)	Intervalo (ns)
JA[0]	605.1	-
JA[1]	1565.1	960
JA[2]	1285.1	680
JA[3]	705.1	100

Fonte: (Do Autor, 2018).

A escolha da simulação por *timing* pós-implementação é de que ela se assemelha ao funcionamento real do dispositivo, de forma que se o circuito apresentar a resposta esperada, o seu funcionamento real também será o desejado.

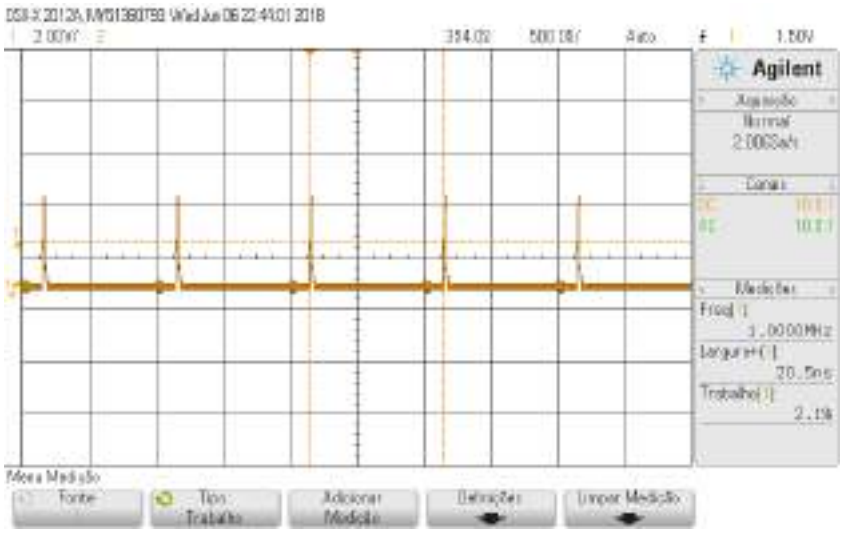
4.1.3 EXPERIMENTAÇÃO EM LABORATÓRIO

Posto que o resultado esperado foi atingido, passou-se a etapa de verificação em laboratório, com o auxílio de um osciloscópio, foi analisado se os sinais programados iriam de fato responder como planejado.

Na figura 25 observa-se o sinal simulado do transmissor visto no osciloscópio, na cor amarela. Analisando as medições dos experimentos laboratoriais, nota-se que o sinal apresenta frequência de 1 MHz, a explicação é de que devido à digitalização, do sinal de pulso, único ter sido de 100 amostras e como o clock do programa é de 100 MHz, a frequência do sinal realmente esperada é de 1 MHz. Por sua vez, o valor da largura do

pulso medido é de 20,5 ns, essa diferença está relacionada aos ruídos existentes, e sendo ela de 0,5 ns, o que representa uma diferença de 2,5% referente ao valor desejado, não irá afetar o sistema, posto que este calcula o intervalo de tempo entre pulsos.

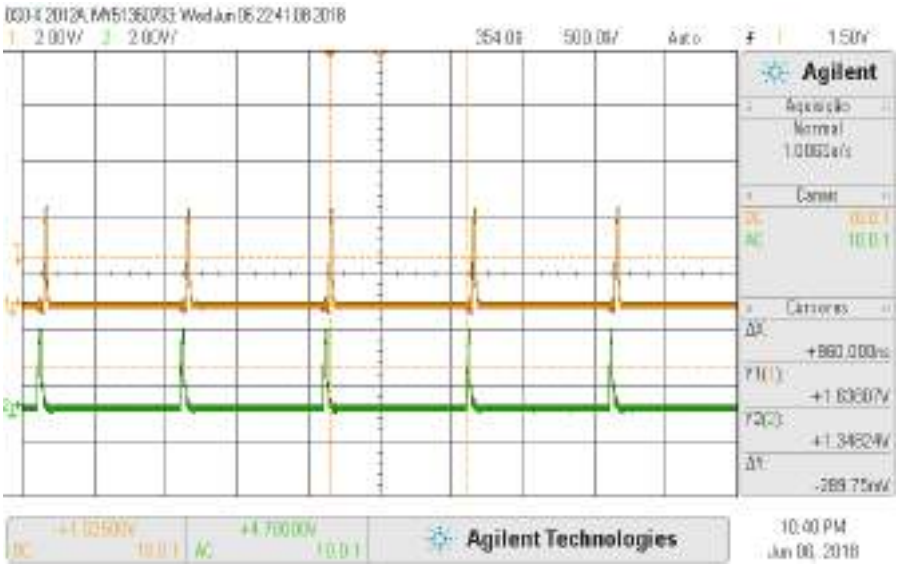
Figura 25 - Forma de onda do sinal de Transmissão visto pelo osciloscópio.



Fonte: (Do Autor, 2018).

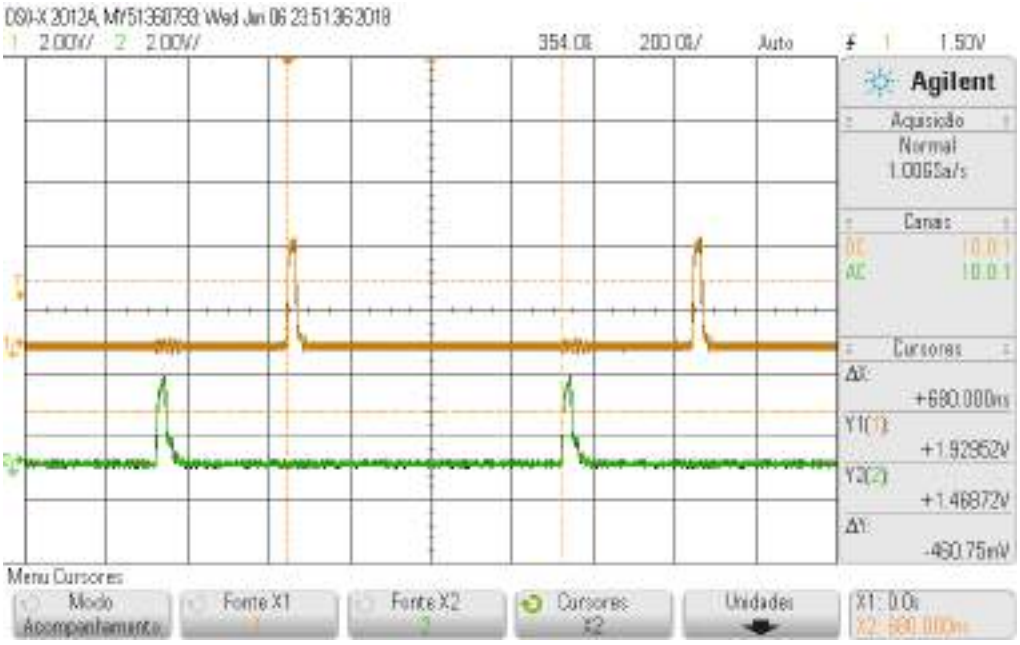
Utilizando os cursores do osciloscópio é medido o intervalo de tempo entre os pulsos, sendo o sinal de amarelo, o sinal do transmissor e o sinal verde, o sinal emulado do receptor. Nas figuras 26, 27 e 28, estão apresentados os intervalos de tempo medidos no osciloscópio entre o sinal do transmissor e os sinais emulados, com atrasos de 960 ns, 680 ns e 100 ns, respectivamente

Figura 26 - Deslocamento no tempo do sinal com atraso de 960ns.



Fonte: (Do Autor, 2018).

Figura 27 - Deslocamento no tempo do sinal com atraso de 680ns.



Fonte: (Do Autor, 2018).

Figura 28 - Descolamento no tempo do sinal com atraso de 100ns.

Fonte: (Do Autor, 2018).

Por meio das figuras, pode-se notar que os valores do deslocamento no tempo do sinal receptor são idênticos aos valores escolhidos no programa, conforme apresentando anteriormente na tabela 2.

Analisando o programa em *Verilog* feito para placa FPGA, é observado que a menor diferença de tempo possível é obtida quando o sinal receptor aparece logo após a

borda negativa do sinal transmissor, resultando em uma diferença de 20 ns e a maior diferença é quando o sinal receptor surge um instante antes da próxima borda positiva do sinal transmissor. Assim tem-se que o maior intervalo de tempo é de 990 ns, levando em consideração que é desejado evitar ambiguidade no processamento do sinal. Pode-se então calcular o alcance máximo por meio da equação (4.4)

$$R_{UA} = \frac{c}{2 * PRF} = \frac{3x10^8 m/s}{2x1MHz} = 150\ metros \tag{4.4}$$

4.2 MEDIÇÃO DE DISTÂNCIA

Conforme já observado nos itens anteriores, para calcular a distância entre o alvo e o sistema de radar, é necessário medir o intervalo de tempo entre os pulsos, esse intervalo de tempo está diretamente ligado a distância através da equação (4.2). Para medir essa velocidade é necessário desenvolver um programa em *Verilog* capaz então de contar o este intervalo.

4.2.1 FUNCIONAMENTO DO CIRCUITO

O código em *Verilog* foi desenvolvido de forma a possuir duas entradas de sinais, uma sendo o sinal transmitido (TX) e a outra o sinal recebido (RX). Para este desenvolvimento pensou-se em diversas maneiras, como por exemplo, iniciar o contador no *negedge* do TX (borda negativa do TX) e parar o contador no *negedge* do RX (borda negativa do RX).

Porém esse método se tornou problemático quando passou pela síntese do programa, posto que foi verificado um *critical warning* de *multi-driven*, indicando que um sinal podia gerar conflito na entrada do bloco lógico, o que poderia ocorrer, por exemplo, no caso de existir *negedge* do transmissor e receptor ao mesmo tempo, provocando dois valores distintos em uma única entrada.

A solução encontrada foi de criar no *posedge* do clock (borda positiva do clock) uma condição *if else*, de modo que o contador é iniciado quando o transmissor for igual a 1 e o sinal do receptor for igual a 0, e o contador é interrompido quando o sinal do

transmissor for igual a 0 e o sinal do receptor for igual a 1. Desta forma, previne-se de receber um sinal enquanto a antena ainda está emitindo o sinal transmissor.

O sistema medidor de distância foi desenvolvido de modo que com o contador sendo controlado como descrito acima, após o término da medição de tempo entre pulsos, o valor do contador é inserido na equação (4.5) e o resultado da distância calculada é apresentada utilizando o display de 7-segmentos.

Para o cálculo da medição da distância a equação (4.2) é simplificada, resultando na equação (4.5).

$$D = \frac{c * Cont_{delay} * \tau}{2} = \frac{3x10^{8} * Cont_{delay} * \frac{1}{100x10^{6}}}{2}$$
$$D = \frac{3Cont_{delay}}{2} \quad (4.5)$$

Sendo $Cont_{delay}$ é o valor do contador entre o sinal do transmissor e o sinal receptor contado pelo programa, em pulsos de clock.

O circuito de acionamento do display de 7-segmentes utilizado é um código fornecido pelo Prof. Robert Reese durante o curso de *Digital Systems Design* durante o período do *Spring 2016*, no *Mississippi State University*, o qual o aluno cursou durante o intercâmbio do Ciência Sem Fronteiras durante os meses de junho 2015 a agosto 2016.

Conforme mostrado na equação (4.2), pode-se realizar os cálculos, para os três exemplos de sinais simulados e desenvolvidos neste trabalho. Os resultados calculados são vistos na tabela 5.

Tabela 5 – Distância calculada para os sinais simulados.

Início do Contador	Tempo de Atraso em ns	Distância em metros
4	960	144
32	680	102
90	100	15

Fonte: (Do Autor, 2018).

Os sistemas digitais são regidos por bits que podem ser representados por um sistema hexadecimal, de forma que a cada 4 bits se representa 1 dígito hexadecimal. O display de 7-segmentos funciona de maneira a apresentar em cada display um valor em hexadecimal (0 a F). Logo, para que seja exibido um valor em decimal é necessário

realizar uma conversão. Esta é realizada de binário para BCD (*Binary-Coded Decimal*) que é um número binário, porém a cada 4 bits é representado um valor numérico (0 a 9).

O algoritmo para esta conversão é simples e pode ser encontrada na internet programas em *Verilog* que realização a conversão binário-BCD. Trata-se de converter um número binário de 8 bits em outro número binário só que com 12 bits. Esta conversão é feita de modo que o número binário de 12 bits possua um valor numérico que vá de 0 até 9 a cada 4 bits, representado em binário 0000 até 1001.

Esta conversão é útil, uma vez que o display de 7-segmentos utiliza uma lógica que determina que o conteúdo de cada um deles é representado por um número de 4 bits. Neste trabalho foi utilizado um código desenvolvido e disponibilizado online por Daniel (2013).

Em resumo, o código desenvolvido em *Verilog* para medir a distância de um objeto utilizando sinais de radar foi desenvolvido seguindo o algoritmo mostrado na Figura 29.

Após a construção do circuito digital, seguindo os passos mostrados na figura 29, é necessário realizar simulações com o Vivado, a fim de verificar o funcionamento do circuito. O circuito digital medidor de distâncias possui apenas uma chave de controle que é SW[0], funcionando como reset do sistema, zerando todos os registradores ou retornando eles para suas configurações padrão pré-definidas no código.

Figura 29 - Processo de funcionamento do programa medidor de distância.

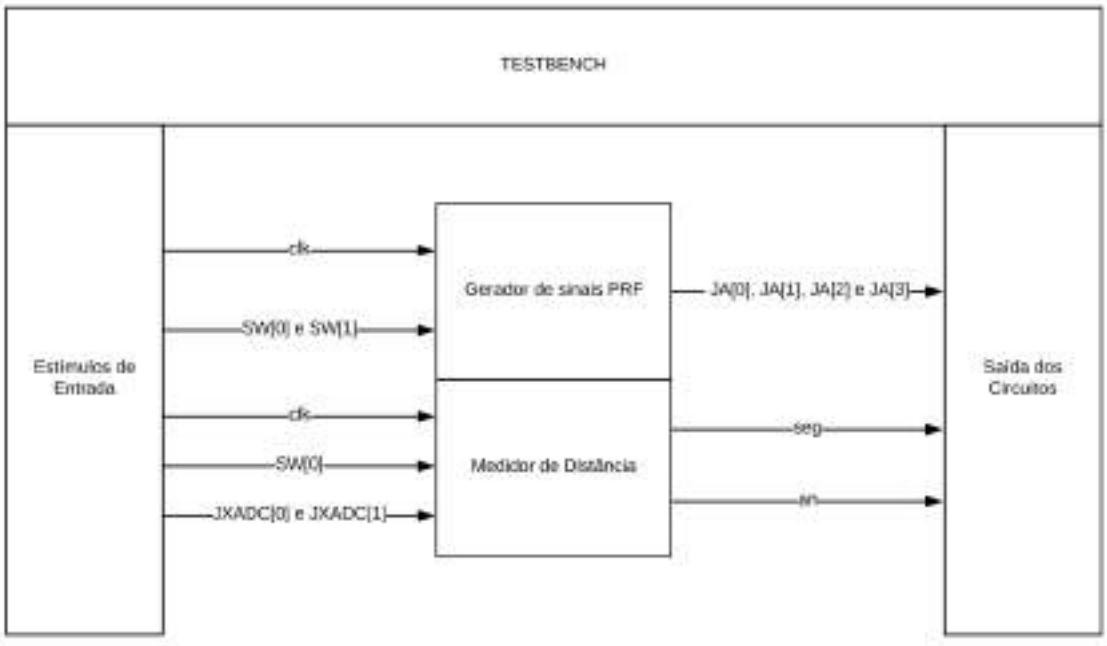
Fonte: (Do Autor, 2018).

4.2.2 SIMULAÇÃO DO MEDIDOR DE DISTÂNCIA

Para realizar a simulação foi desenvolvido um arquivo *testbench* em *Verilog*, semelhantemente ao realizado na seção 4.1.2. Para a execução é necessário estimular sinais PRF na entrada do sistema, portanto, foi acoplado ao circuito medidor de distâncias o circuito gerador de sinais PRF descrito em 4.1. Na figura 30 é apresentado um esquemático das conexões de *input* e *output* do programa simulador de sinais PRF e o do programa medidor de distância no *testbench*.

Percebe-se que dentro do *testbench* é realizada a ligação das portas JA[0], JA[1], JA[2] e JA[3] com as portas JXADC[0] e JXADC[1]. Para poder analisar todos os sinais desenvolvidos pelo gerador, o *testbench* foi feito para que o sinal JA[0] estivesse sempre conectado à porta JXADC[0] e os sinais JA[1], JA[2] e JA[3] fossem conectados na porta JXADC[1] por intervalos definidos de tempo, permitindo a análise do programa utilizando diferentes sinais RX. A simulação do programa medidor de distância apresenta os sinais mostrados na tabela 6. O código do *testbench* pode ser observado no Apêndice B.

Figura 30 - Esquemático do *testbench* para simulação do programa medidor de distância.



Fonte: (Do Autor, 2018).

Tabela 6 – Sinais visualizados na simulação do programa.

Sinal		Função
clk		clock do sistema - 100MHz
SW[0]		Chave de Controle Reset
JXADC	JXADC[0]	Porta de entrada do sinal TX
	JXADC[1]	Porta de entrada do sinal RX
start_cont		Inicia o Contador
saida_delay		Valor contado no contador (em pulsos de clock)
display_din		Valor convertido da distância em metros

Fonte: (Do Autor, 2018).

A simulação realizada foi do tipo *timing* pós-implementação. As figuras de 31 a 34 apresentam as formas de onda do sinal do circuito medidor através da simulação.

Figura 31 - Simulação do programa medidor de distância.

Fonte: (Do Autor, 2018).

Figura 32 - Detalhe da simulação do sinal RX com objeto a 144 metros.

Fonte: (Do Autor, 2018).

Figura 33 - Detalhe da simulação do sinal RX com objeto a 102 metros.

Fonte: (Do Autor, 2018).

Figura 34 - Detalhe da simulação do sinal RX com objeto a 15 metros.

Fonte: (Do autor, 2018).

Os resultados da simulação são vistos nas figuras 31, 32, 33 e 34. Observa-se que na implementação os primeiros cálculos do atraso pelo contador não foram transferidos para o registrador “saída_delay”, as razões para isso são de possíveis defeitos na ferramenta do Vivado, uma vez logo em seguida o sistema conseguiu medir a distância de forma automática.

Outro fato interessante verificados pós-implementação é de que os registradores tiveram seus tamanhos reduzidos. Durante o processo da síntese e da implementação o código é otimizado, visando consumir o menor número de componentes lógicos possíveis, dessa forma vê-se que foram aplicadas alterações em alguns registradores, que acarreta em uma economia de energia consumida pelo sistema.

A escolha da simulação por *timing* pós-implementação é de que ela se assemelha ao funcionamento real do dispositivo, de forma que se o circuito apresentar a resposta esperada, o seu funcionamento real será também o desejado.

4.2.3 TESTES NA PLACA

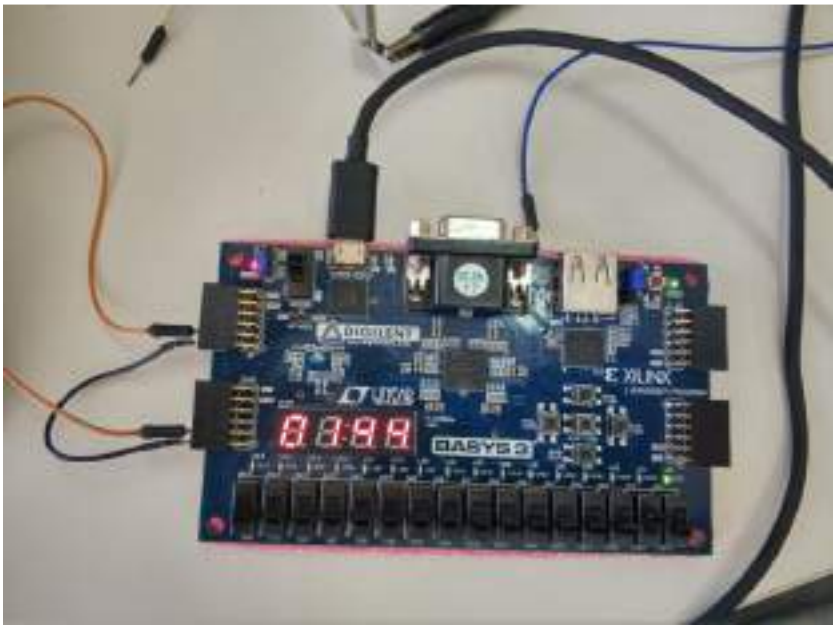
Após a análise da simulação pós-implementação, levando em consideração os atrasos existentes nos blocos lógicos do dispositivo FPGA, é então realizada a última etapa de teste de funcionamento, testar na placa o funcionamento desse sistema.

Para esta realização foi incluído no projeto o circuito medidor de distância e o circuito emulador de sinais PRF, dessa forma obtendo na saída da porta Pmod JA os sinais emulados de radar, conforme mostrado no item 4.3.

Utilizando cabos do tipo *jumpers* é realizada uma ligação externa entre os pinos da placa, conectando a saída do sinal na porta JA com a entrada do sinal na porta JXADC.

Realizando a programação na placa através do cabo USB com o computador com o Vivado e feita a ligação entre os pinos JA[0] e JA[1] com os pinos JXADC[0] e JXADC[1], respectivamente, pode-se observar na figura 35 o display da placa informando o valor medido da distância em metros.

Figura 35 - Medição de distância para o sinal recebido a uma distância de 144 metros.

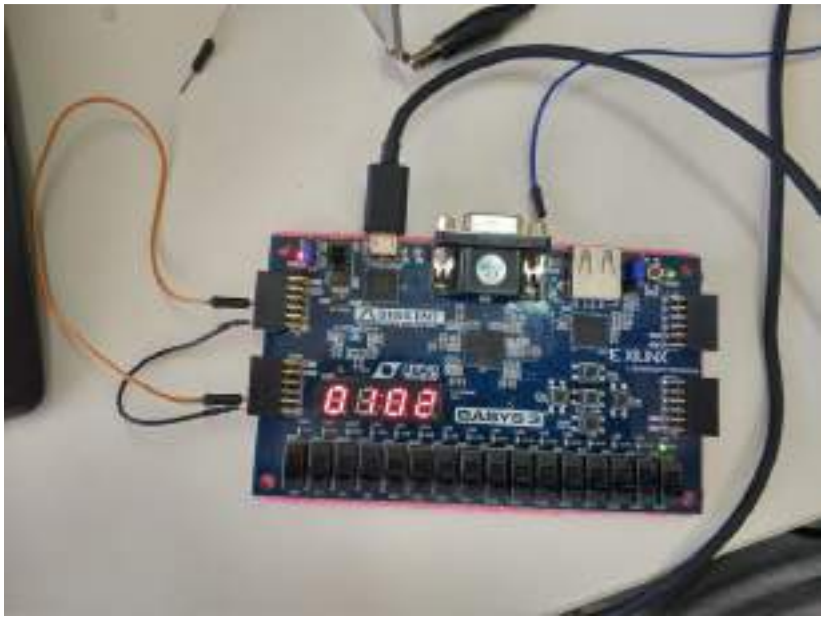


Fonte: (Do Autor, 2018).

Na figura 36 obtém-se no display o número 102 indicando que foi medido uma distância de 102 metros considerando os sinais TX, relacionado à porta JA[0] conectada a JXADC[0], e RX, conectando porta JA[2] com a porta JXADC[1]. De maneira semelhante, observa-se o mesmo efeito na figura 37, quando a porta JA[3] é ligada na

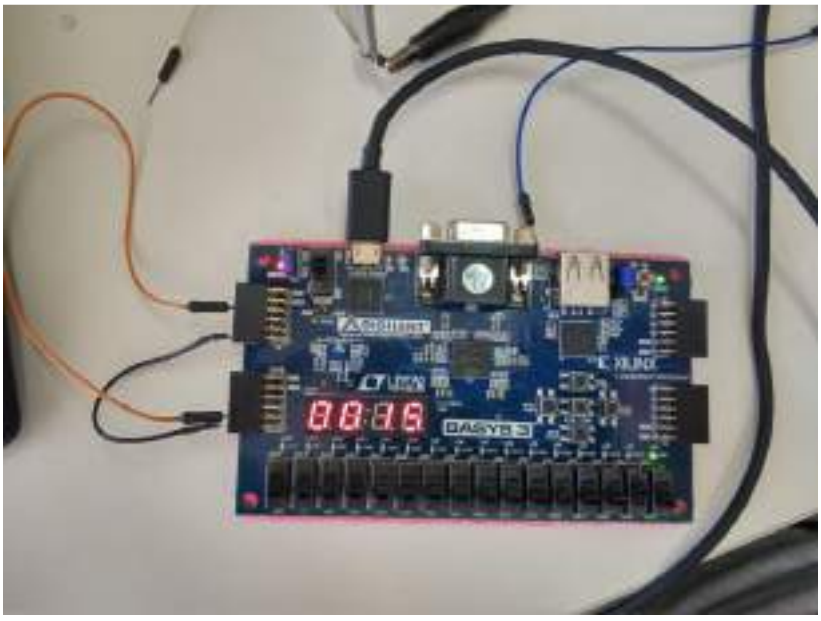
porta de RX, JXADC[1], mostrando no display o número 15 que significa 15 metros de distância.

Figura 36 - Medição de distância para o sinal recebido a uma distância de102 metros.



Fonte: (Do Autor, 2018).

Figura 37 - Medição de distância para o sinal recebido a uma distância de 15 metros.



Fonte: (Do Autor, 2018).

Pode-se verificar, no entanto, alguma dificuldade para a identificação dos resultados do display nas figuras 35, 36 e 37, devido à qualidade de impressão e a luz no ambiente na hora da foto.

4.3 CARACTERÍSTICAS DO SISTEMA

O objetivo sistema implementado foi de instanciar os circuitos do gerador de sinais (emuladorPRF) e do medidor de distância (MedidorDistancia) em um único módulo, sintetizar, implementar, gerar o arquivo de *bitstream* e programar na placa. No Apêndice A estão disponíveis os códigos em *Verilog* do sistema implementado e programado na Basys 3, e no Apêndice B está disponível o código do *testbench*, dos circuitos simulados nas seções 4.1 e 4.2.

O sistema implementado possui um clock de 100 MHz e o gerador de pulsos produz um sinal com PRF de 1 MHz, portanto, o circuito medidor de distância realiza 1 milhão de medições por segundo.

O menor intervalo de tempo possível para ser medido é de 20 ns, pois essa é a largura de pulso do sinal PRF, já o maior intervalo de tempo medido é de 990 ns, correspondente, ao intervalo de tempo antes de ser transmitido o próximo pulso. Sendo assim, pode-se determinar a distância mínima e máxima que o sistema pode medir seguindo a equação (4.5), assim teremos os valores disponíveis na tabela 7.

Tabela 7 – Região de medição do sistema desenvolvido.

	Tempo (ns)	Alcance
Mínimo	20	3
Máximo	990	148,5

Fonte: (Do Autor, 2018).

A resolução do sistema depende do clock adotado, sendo utilizada a equação (4.6) para calcular a resolução do sistema desenvolvido.

$$R = \frac{c}{2 * clock} \tag{4.6}$$

Pode-se observar que a resolução do sistema é de 1,5 m. Para melhorar a resolução, seria necessário introduzir um clock mais rápido no sistema. Esta alteração é apresentada na tabela 8, onde se observa valores de resolução para diferentes valores de clock.

Tabela 8 – Valores de Resolução para valores de clock.

clock	Resolução
100MHz	1,5
200MHz	0,75
500MHz	0,3
1GHz	0,15
2GHz	0,075

Fonte: (Do Autor, 2018).

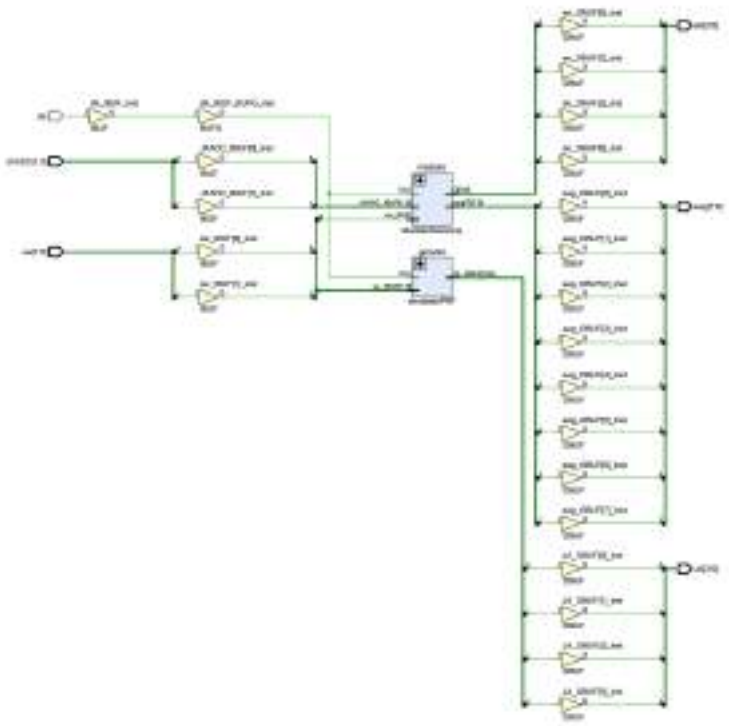
Sendo assim, para aumentar o alcance máximo, é necessário diminuir a PRF do sistema, conforme a equação (4.4). Para diminuir o alcance mínimo, por sua vez, pode-se diminuir a largura de pulso do sinal, diminuindo o *duty-cycle*, assim podendo receber a resposta de sinais que estão mais próximos. Por outro lado, para conseguir resolução menor é preciso aumentar o clock do sistema, como indicam os resultados apresentados na tabela 8. As características do sistema de medição de distância dependem fundamentalmente dessas três variáveis, clock do sistema digital, largura de pulso do sinal PRF e frequência do sinal PRF.

O Vivado disponibiliza o relatório de utilização do circuito digital na placa, este fornece os valores da quantidade dos componentes lógicos da placa que foram utilizados, estas informações estão contidas no Apêndice C.

Analisando o relatório de utilização o circuito implementado, utilizou-se 222 *Slices LUT* sendo todas as LUTs do tipo lógico e 236 *Slice Registers*, com 232 registradores do tipo *flip-flop* e 4 do tipo *latch*.

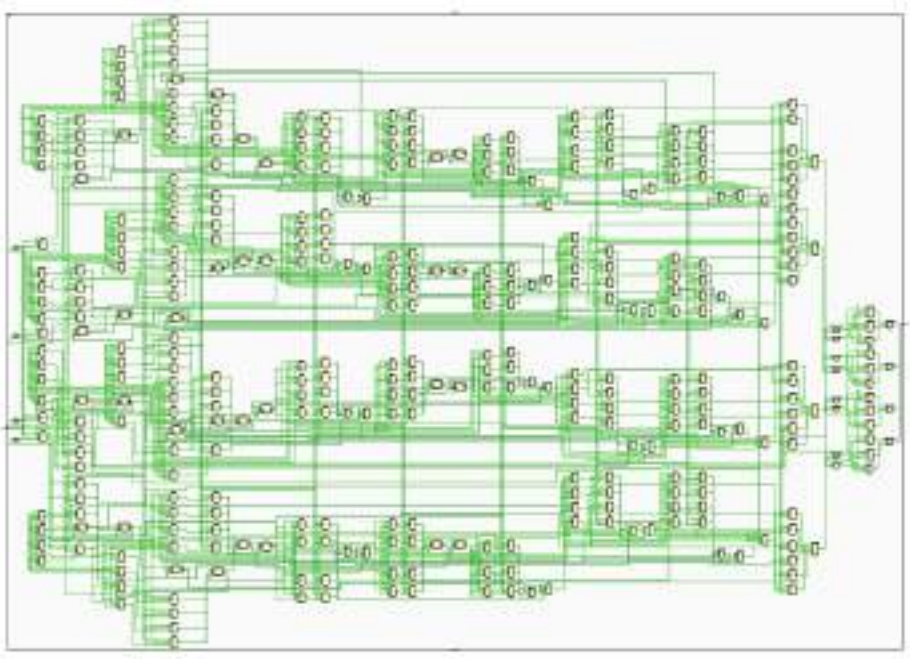
Ainda, também se tem acesso ao esquemático do circuito digital implementado na placa, apresentando as ligações entre os blocos lógicos da placa, conforme visto nas figuras 38, 39 e 40, nas quais se observa o esquemático do módulo principal e dos módulos emuladorPRF e MedidorDistancia, respectivamente.

Figura 38 - Esquemático das ligações implementadas do bloco principal.



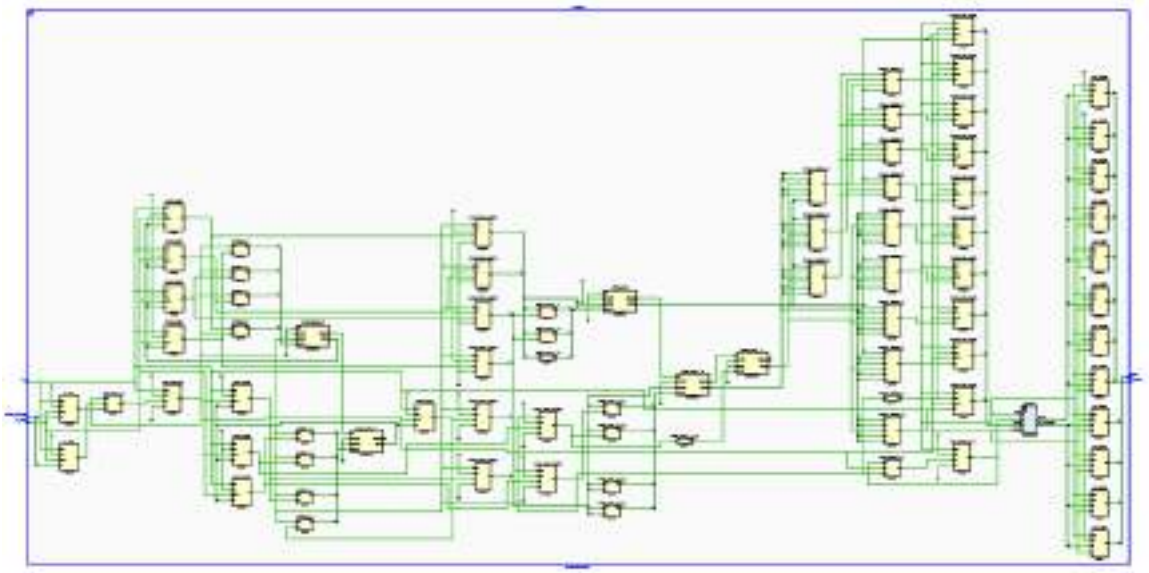
Fonte: (Do Autor, 2018).

Figura 39 - Esquemático do módulo emuladorPRF.



Fonte: (Do Autor, 2018).

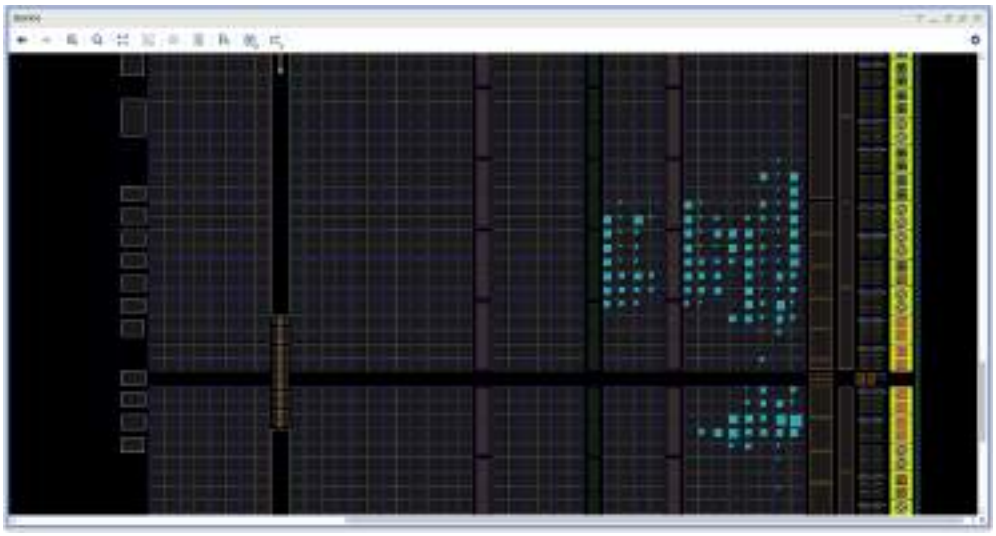
Figura 40 - Esquemático do módulo MedidorDistancia.



Fonte: (Do Autor, 2018).

A partir dos esquemáticos apresentados nas figuras 38, 39 e 40, também se obtém a apresentação dos *slices* utilizados dentro do chip FPGA da Basys 3, essa representação permite tanto observar onde o circuito digital foi implementado dentro do chip, como quais unidades lógicas foram utilizadas. As figuras 41 e 42 apresentam tal informação sendo na figura 43 visto em detalhe o esquema implementado mostrando quais blocos lógicos foram utilizados em seus *slices*.

Figura 41 - Visualização do circuito digital implementado no chip da placa FPGA.



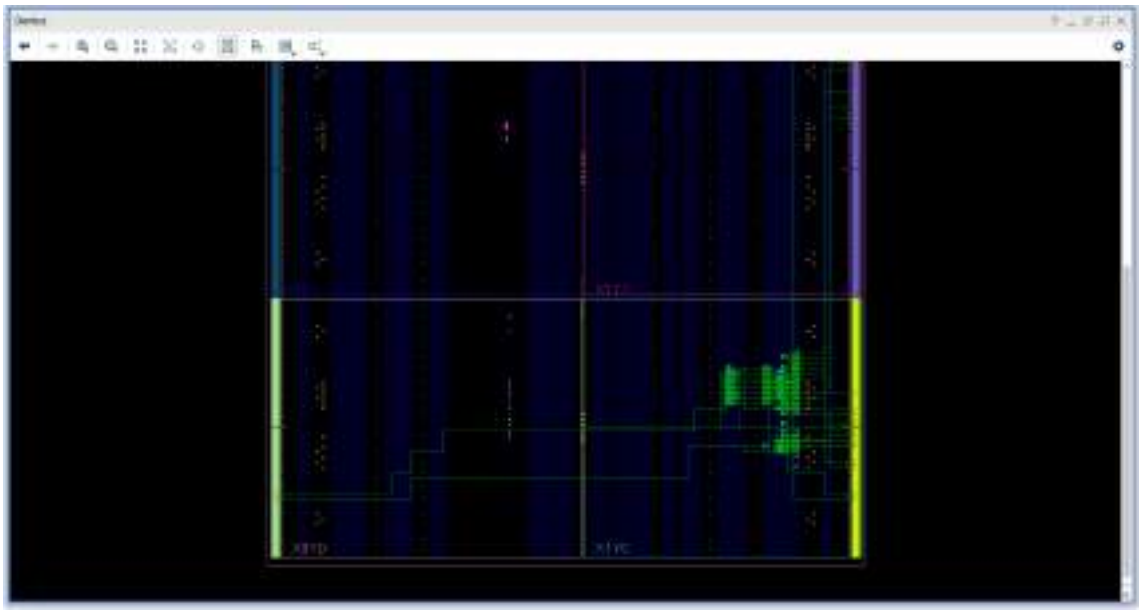
Fonte: (Do Autor, 2018).

Figura 42 - Componentes lógicos utilizados da placa vista com detalhe.

Fonte: (Do Autor, 2018).

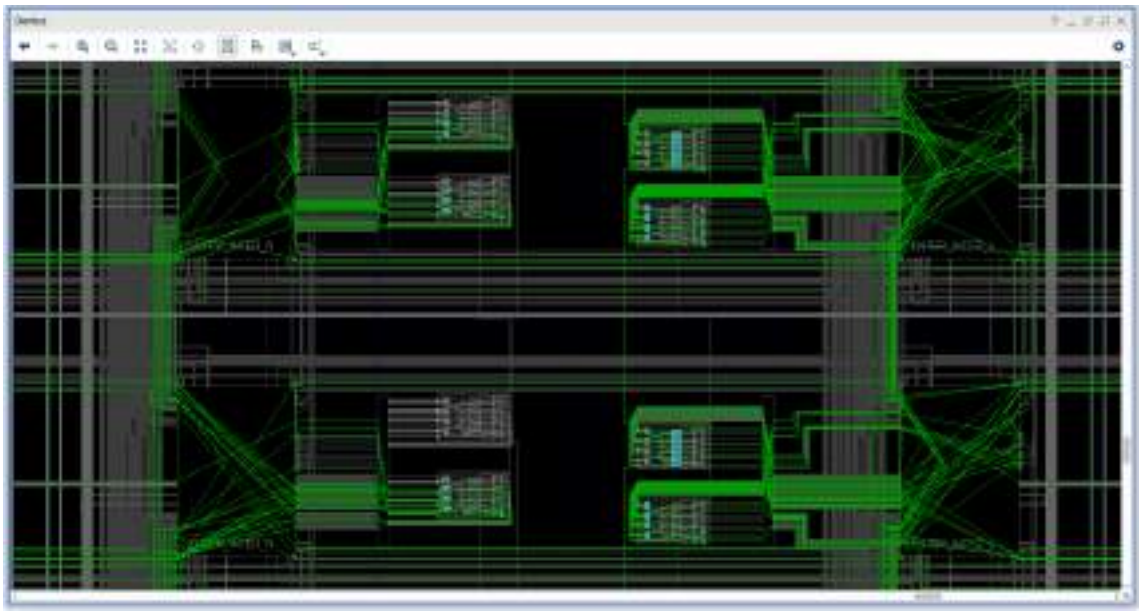
A ferramenta permite também a observação do circuito implementado apresentando os componentes lógicos mostrados na figura 42 junto as conexões das chaves que existem no interior do chip FPGA da Basys 3, as figuras 43 e 44 apresentam as figuras 41 e 42 com detalhe das ligações realizadas.

Figura 43 - Implementação apresentando as conexões realizadas na placa.



Fonte: (Do Autor, 2018).

Figura 44 - Implementação apresentando as ligações com detalhe.



Fonte: (Do Autor, 2018).

5 CONCLUSÃO

A utilização de ferramentas para desenvolvimento de sistemas digitais é de grande importância para a eletrônica pois elas permitem a realização do projeto eletrônico digital, simulação, verificação e testes de protótipos. Existem no mercado diversas ferramentas de síntese de projeto digital, como o Quartus II®, desenvolvido pela Altera©, Stratus®, da Cadence® dentre outros.

Os sistemas digitais são primordiais para os sistemas de radar, possibilitando o processamento de funções complexas que realizam as funções do radar de forma ágil e confiável, economizando energia e área de processamento. Os radares são uma tecnologia em sempre expansão, conquistando novas fronteiras e seu desenvolvimento necessita de ferramentas capazes de realizar suas funções em alto desempenho.

A pretensão desse trabalho é de utilizando a ferramenta Vivado HLS desenvolver um sistema digital capaz de medir distância utilizando um emulador de sinal PRF de radar como entrada. Desta forma, analisando os resultados obtidos o trabalho conseguiu cumprir com os objetivos propostos de utilizar o Vivado para síntese, análise e implementação de um sistema digital desenvolvimento um emulador de sinal e um sistema de medição de distância a partir do eco de sinais PRF de radar.

Assim, para o sistema medidor de distância, a utilização do dispositivo FPGA foi por questão de conveniência de o aluno já possuir uma placa FPGA, porém, para essa implementação, poderiam ser utilizados outros dispositivos, como microcontroladores ou até circuitos integrados de portas lógicas. Por outro lado, ao utilizar a Basys 3, foi possível implementar o sistema com um clock de 100 MHz, o qual a maioria dos microcontroladores não dispõem, o que permitiu uma resolução de 1,5 m, em que caso contrário seria maior.

Para continuação do trabalho, pode-se utilizar o sistema digital desenvolvido para incrementar mais funções de sistema de radar, como também montar o circuito analógico referente à transmissão e recepção, com o condicionamento do sinal, para aplicar o sistema desenvolvido em um ambiente real.

Essa atividade conseguiu unir vários elementos da eletrônica digital, como análise de sinais e sistemas digitais e projeto digital em *hardware*, trabalhando com a união desses tópicos para desenvolver um programa útil que pode ser aplicado, por exemplo,

em sistemas de radar para carros autônomos ou carros modernos, não autônomos, fornecendo ao motorista informações acerca de veículos ao seu redor.

BIBLIOGRAFIA

BARTON, David K. **Radar Equations for Modern Radar**. Norwood, MA: Artech House, 2012.

CHU, Pong P. **FPGA Prototyping by Verilog Exemples**: Xilinx Spartan – 3 Version. Hoboken NJ: John Wiley & Sons, 2008.

CURRY, Richard. **Radar System Performance Modeling**. 2. Ed. Norwood, MA: Artech House, 2004.

Daniel**. Binary to Binary-Coded Decimal (BCD) Converter**. Disponível em: <http://www.deathbylogic.com/2013/12/binary-to-binary-coded-decimal-bcd-converter/#more-441>. Acesso em: 25 abr. 2018.

DICKMANN, Jürgen; APPENRONDT, Nils; BRENK, Carsten. **Making Bertha See**. IEEE Spectrum, Agosto 2014, p. 44-49.

Digilent. 1. Ed. *Basys 3 FPGA Reference Manual*. Pulman, WA, 2017

FERDJALLAH, Mohammed. **Introduction to Digital Systems**: Modeling, Synthesis and Simulation Using VHDL. Hoboken, NJ: John Wiley & Sons, 2011.

GUIMARÃES, Gabriel T. N. **Construção de um sistema de radar capaz de medir distância, efeito doppler e imageamento por abertura sintética**. 60 p. Trabalho de Conclusão de Curso (Curso de Engenharia Elétrica) - Universidade Federal da Paraíba, João Pessoa, 2014.

IEEE Standards Association. (2005). 1364-2005. *IEEE Standard for Verilog Hardware Description Language*. Fonte: https://standards.ieee.org/findstds/standard/1364-2005.html.

INGLE, Vinay K.; PROAKIS, John G. **Digital Signal Processing Using Matlab**. Stamford, CT: Cengage Learning, 2010

LEVANON, Nadav; MOZESON, Eli. **Radar Signals**. Hoboken, NJ: John Wiley & Sons, 2004.

Mizirin, A.V., Timasheva, T.G., Fedorov, V.A. et al. (Julho 2010). **The PULSAR Radar Instrument and Its Use for Comprehensive Examination of Health Status**. Biomedical Engineering. Vol.44. Pp 50-56. Fonte: https://doi-org.ez15.periodicos.capes.gov.br/10.1007/s10527-010-9154-0.

PEPLOW, R., DAWOUD, Shenouda. **Digital System Design –** Use of Microcontroller. Alborg, Dinamarca: River Publishers, 2010.

RICHARDS, Mark A.; SCHEER, James A.; HOLM, William A. **Principles of modern radar:** vol. 1 Basic Principles. Edinson, NJ: Scitech Publishing, 2010.

SKOLNIK, Merrill I. **Radar**: History of Radar: Encyclopædia Britannica, Inc., 2018. Disponível em: <https://www.britannica.com/technology/radar/History-of-radar>. Acesso em: 30 maio 2018.

Xilinx. (2017a). UG973. *Vivado Design Suite User Guide – Release Notes, Installation and Licensing*. Fonte: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_2/ug973-vivado-release-notes-install-license.pdf.

Xilinx. (2017b). UG949. *UltraFast Design Methodology Guide for the Vivado Design Suite.* Fonte: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_1/ug949-vivado-design-methodology.pdf.

Xilinx. (2018). DS181. *Artix-7 FPGAs Data-sheet: DC and AC Switching Characteristics*. Fonte: https://www.xilinx.com/support/documentation/data_sheets/ds181_Artix_7_Data_Sheet.pdf.

APÊNDICE A – CÓDIGO DO TRABALHO DESENVOLVIDO

I – MÓDULO PRINCIPAL

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Universidade Federal Da Paraibba - Centro de Energias Alternativas Renováveis - Coordenação da Engenharia Elétrica
// Aluno:                   Gustavo Maximo Urquiza de Sá
//Matrícula:                11118276
//
// Data de Criação:         08/04/2018
// Nome do Trabalho:        Trabalho de Conclusão de Curso
// Nome do Módulo:          ProjetoTCC
// Nome do Projeto:         Medição de Distância a Partir de Sinal de Radar
// Dispositivo Utilizado:   Basys 3
// Versão da Ferramenta:    Vivado 2017.2
// Descrição:               Projeto em Verilog utilizado na Monografia
//
//
//////////////////////////////////////////////////////////////////////////////////
module ProjetoTCC (clk, sw, JA, JXADC, an, seg);

input clk;
input [1:0] sw;
input [1:0] JXADC;
output [3:0] JA;
output [3:0] an;
output [6:0] seg;


emuladorPRF gerador (.clk(clk),
                     .rst(sw[0]),
                     .controle(sw[1]),
                     .sinais(JA));
//

MedidorDistancia medidor (.clk(clk),
                          .rst(sw[0]),
                          .RX(JXADC[1]),
                          .TX(JXADC[0]),
                          .an(an),
                          .seg(seg));
//


endmodule
```

II – EMULADOR PRF

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////////////////////
3   // Universidade Federal Da Paraíba - Centro de Energias Alternativas Renováveis -
    Coordenação de Engenharia Elétrica
4   // Aluno:                   Gustavo Maximo Urquiza de Sá
5   //Matrícula:                11118276
6   //
7   // Data de Criação:         08/04/2018
8   // Nome do Trabalho:        Trabalho de Conclusão de Curso
9   // Nome do Módulo:          emuladorPRF
10  // Nome do Projeto:         Medição de Distância a Partir de Sinal de Rádar
11  // Dispositivo Utilizado:   Basys 3
12  // Versão da Ferramenta:    Vivado 2017.2
13  // Descrição:               Projeto em Verilog utilizado na Monografia
14  //
15  //
16  //////////////////////////////////////////////////////////////////////////////////
17  module emuladorPRF (clk, rst, controle sinais);
18
19  input clk;              //Clock do sistema - 100MHz.
20  input rst;          //Chaves de controle de Reset (rst) e controle da saída dos sinais
    simulados (ccontrole).
21  input controle;
22  output reg [3:0] sinais;    //Porta de Saída dos Sinais TX (sinais[0]) e dos três
    sinais RX (sinais[1], sinais[2] e sinais[3].
23
24  // CERAÇÃO DE SINAIS
25
26  reg [0:0] pwm [95:0];
27  integer i;
28  integer j;
29  integer k;
30  integer l;
31  initial begin
32  i = 0;
33  j = 0;
34  k = 0;
35  l = 0;
36  pwm[0] = 1;
37  pwm[1] = 1;
38  pwm[2] = 0;
39  pwm[3] = 0;
40  pwm[4] = 0;
41  pwm[5] = 0;
42  pwm[6] = 0;
43  pwm[7] = 0;
44  pwm[8] = 0;
45  pwm[9] = 0;
46  pwm[10] = 0;
47  pwm[11] = 0;
48  pwm[12] = 0;
49  pwm[13] = 0;
50  pwm[14] = 0;
51  pwm[15] = 0;
52  pwm[16] = 0;
53  pwm[17] = 0;
54  pwm[18] = 0;
55  pwm[19] = 0;
56  pwm[20] = 0;
57  pwm[21] = 0;
58  pwm[22] = 0;
59  pwm[23] = 0;
60  pwm[24] = 0;
61  pwm[25] = 0;
62  pwm[26] = 0;
63  pwm[27] = 0;
64  pwm[28] = 0;
65  pwm[29] = 0;
66  pwm[30] = 0;
```

```
67   pwm[31] = 0;
68   pwm[32] = 0;
69   pwm[33] = 0;
70   pwm[34] = 0;
71   pwm[35] = 0;
72   pwm[36] = 0;
73   pwm[37] = 0;
74   pwm[38] = 0;
75   pwm[39] = 0;
76   pwm[40] = 0;
77   pwm[41] = 0;
78   pwm[42] = 0;
79   pwm[43] = 0;
80   pwm[44] = 0;
81   pwm[45] = 0;
82   pwm[46] = 0;
83   pwm[47] = 0;
84   pwm[48] = 0;
85   pwm[49] = 0;
86   pwm[50] = 0;
87   pwm[51] = 0;
88   pwm[52] = 0;
89   pwm[53] = 0;
90   pwm[54] = 0;
91   pwm[55] = 0;
92   pwm[56] = 0;
93   pwm[57] = 0;
94   pwm[58] = 0;
95   pwm[59] = 0;
96   pwm[60] = 0;
97   pwm[61] = 0;
98   pwm[62] = 0;
99   pwm[63] = 0;
100  pwm[64] = 0;
101  pwm[65] = 0;
102  pwm[66] = 0;
103  pwm[67] = 0;
104  pwm[68] = 0;
105  pwm[69] = 0;
106  pwm[70] = 0;
107  pwm[71] = 0;
108  pwm[72] = 0;
109  pwm[73] = 0;
110  pwm[74] = 0;
111  pwm[75] = 0;
112  pwm[76] = 0;
113  pwm[77] = 0;
114  pwm[78] = 0;
115  pwm[79] = 0;
116  pwm[80] = 0;
117  pwm[81] = 0;
118  pwm[82] = 0;
119  pwm[83] = 0;
120  pwm[84] = 0;
121  pwm[85] = 0;
122  pwm[86] = 0;
123  pwm[87] = 0;
124  pwm[88] = 0;
125  pwm[89] = 0;
126  pwm[90] = 0;
127  pwm[91] = 0;
128  pwm[92] = 0;
129  pwm[93] = 0;
130  pwm[94] = 0;
131  pwm[95] = 0;
132  pwm[96] = 0;
133  pwm[97] = 0;
134  pwm[98] = 0;
135  pwm[99] = 0;
```

```
136     end
137
138
139     always @(posedge clk or posedge rst) begin
140         if(rst == 1) begin
141             sinais[0] <= pwm[i];
142             sinais[1] <= pwm[j];
143             sinais[2] <= pwm[k];
144             sinais[3] <= pwm[l];
145
146             i <= 0;
147             j <= 4;
148             k <= 32;
149             l <= 96;
150         end else begin
151             sinais[0] <= pwm[i];
152             sinais[1] <= pwm[j];
153             sinais[2] <= pwm[k];
154             sinais[3] <= pwm[l];
155             if(controle == 1) begin
156                 j <= j + 1;
157                 i <= i + 1;
158                 k <= k + 1;
159                 l <= l + 1;
160             end
161
162             if(i == 99) i <= 0;
163
164             if(j == 99) j <= 0;
165
166             if(k == 99) k <= 0;
167
168             if(l == 99) l <= 0;
169
170         end
171     end
172
173     endmodule
```

III – MEDIDOR DE DISTÂNCIA

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Universidade Federal Da Paraíba - Centro de Energias Alternativas Renováveis - Coordenação de Engenharia Elétrica
// Aluno:                   Gustavo Maximo Urquiza de SÃ;
//Matrícula:                11118276
//
// Data de Criação:         08/04/2018
// Nome do Trabalho:        Trabalho de ConclusÃ£o de Curso
// Nome do Módulo:          MedidorDistancia
// Nome do Projeto:         Medição de Distância a Partir de Sinal de Radar
// Dispositivo Utilizado:   Basys 3
// Versão da Ferramenta:    Vivado 2017.2
// Descrição:               Projeto em Verilog utilizado na Monografia
//
//
//////////////////////////////////////////////////////////////////////////////////
module MedidorDistancia (clk, rst, RX, TX, an, seg);

input clk;              //Clock do sistema - 100MHz.
input rst;              //Reset
input RX;               //
input TX;               //
output reg [7:0] seg;   //Saída dos Cátodos para o display de 7-segmentos.
output reg [3:0] an;    //Saída dos os Ânodos para o display de 7-segmentos.


reg start_cont;
reg [15:0] saida_delay;
reg [15:0] display_din;
reg load_display;
reg rx;
reg tx;

always @(posedge clk or posedge rst) begin
    if(rst == 1) begin
        rx <= 0;
        tx <= 0;
    end else begin
            tx <= TX;
            rx <= RX;
        end
end


always @(posedge clk) begin
    if((tx == 1) & (rx == 0)) begin
        start_cont <= 1;
    end else if ((tx ==0) & (rx ==1))start_cont <= 0;

end

//Cáculo do intervalo de tempo entre os pulsos

reg [15:0] cont;

always @(posedge clk or posedge rst) begin
    if(rst == 1) begin
        cont <= 0;
    end else begin
            if(start_cont == 1) begin
                cont <= cont + 1;
            end else cont <= 0;
        end
end

always @(negedge start_cont) begin
    saida_delay <= cont;
```

```
69  end
70
71  // calculo da distância e conversão de binário p/ bcd
72
73  reg [15:0] bcd;
74
75  reg [19:0] shift;
76
77  integer k;
78  always @(saida_delay) begin
79
80  shift [19:8] = 0;
81  shift [7:0] = (3 * saida_delay[7:0]) / 2;   // Como o Alcance é de 150 metros, 8 bits
    são suficiente para representar.
82                                              // Nota-se que BCD possui 16 bits por e
                                                cada 4 bits representando um dígito numérico
83  for(k=0; k<8; k=k+1) begin
84      if(shift[11:8] >= 5) shift [11:8] = shift[11:8] + 3;
85      if(shift[15:12] >= 5) shift[15:12] = shift[15:12] + 3;
86      if(shift[19:16] >= 5) shift[19:16] = shift[19:16] + 3;
87      shift = shift << 1;
88  end
89
90  bcd = {4'b0000, shift[19:8]};
91
92  end
93
94
95  always @(posedge clk or posedge rst) begin
96      if (rst == 1) begin
97          display_din <= 0;
98      end else begin
99              if(start_cont == 0) begin
100                 display_din <= bcd;
101                 load_display <= 1;
102             end else begin
103                 load_display <= 0;
104             end
105         end
106 end
107
108 wire [7:0] seg_display;
109 wire [3:0] an_display;
110
111
112 sevenseg display    (.clk(clk),
113                      .reset(rst),
114                      .ld(load_display),
115                      .din(display_din),
116                      .dp(4'b1111),
117                      .SSEG_CA(seg_display),
118                      .SSEG_AN(an_display));
119 //
120
121
122 always @(posedge clk) begin
123     if(rst == 1) begin
124         seg <= 8'hFF;
125         an <= 4'hF;
126     end else begin
127             seg <= seg_display;
128             an <= an_display;
129       end
130 end
131
132 endmodule
```

APÊNDICE B – CÓDIGO DO *TESTBENCH*

```
module tb_tcc;

    reg clk;
    reg [3:0]SW;
    reg [1:0] jxadc;

    wire [3:0]ja;
    wire [7:0]SSEG;
    wire [3:0]ANSEG;

    ProjetoTCC uut( .clk(clk),
                    .sw(SW),
                    .JA(ja),
                    .JXADC(jxadc),
                    .seg(SSEG),
                    .an(ANSEG));
    //

    initial begin
    clk = 0;

    #100

    forever #5 clk = ~clk;
    end

integer l;
integer m;
integer n;
    initial begin

        #1
        SW[0] = 1;
        clk = 0;
        l = 0;
        m = 0;
        n = 0;

        #600
        SW[0] = 0;
        SW[1] = 1;

        while(l <= 300) begin
            @(negedge clk);
            jxadc[0] = ja[0];
            jxadc[1] = ja[1];
            l = l+1;
        end

        while(m <= 300) begin
            @(negedge clk);
            jxadc[0] = ja[0];
            jxadc[1] = ja[1];
            m = m+1;
        end

        while(n <= 300) begin
            @(negedge clk);
            jxadc[0] = ja[0];
            jxadc[1] = ja[1];
            n = n+1;
        end

    end

endmodule
```

APÊNDICE C - UTILIZATION REPORT PÓS IMPLEMENTAÇÃO

```
Copyright 1986-2017 Xilinx, Inc. All Rights Reserved.
------------------------------------------------------------------------------------
------------------------
| Tool Version : Vivado v.2017.2 (win64) Build 1909853 Thu Jun 15 18:39:09 MDT 2017
| Date         : Sun Jun 10 17:33:48 2018
| Host         : LAPTOP-94EOKAFK running 64-bit major release  (build 9200)
| Command      : report_utilization -file ProjetoTCC_utilization_placed.rpt -pb
ProjetoTCC_utilization_placed.pb
| Design       : ProjetoTCC
| Device       : 7a35tcpg236-1
| Design State : Fully Placed
------------------------------------------------------------------------------------
------------------------

Utilization Design Information

Table of Contents
-----------------
1. Slice Logic
1.1 Summary of Registers by Type
2. Slice Logic Distribution
3. Memory
4. DSP
5. IO and GT Specific
6. Clocking
7. Specific Feature
8. Primitives
9. Black Boxes
10. Instantiated Netlists

1. Slice Logic
--------------

+-------------------------+------+-------+-----------+-------+
|        Site Type        | Used | Fixed | Available | Util% |
+-------------------------+------+-------+-----------+-------+
| Slice LUTs              |  222 |     0 |     20800 |  1.07 |
|   LUT as Logic          |  222 |     0 |     20800 |  1.07 |
|   LUT as Memory         |    0 |     0 |      9600 |  0.00 |
| Slice Registers         |  236 |     0 |     41600 |  0.57 |
|   Register as Flip Flop |  232 |     0 |     41600 |  0.56 |
|   Register as Latch     |    4 |     0 |     41600 | <0.01 |
| F7 Muxes                |    0 |     0 |     16300 |  0.00 |
| F8 Muxes                |    0 |     0 |      8150 |  0.00 |
+-------------------------+------+-------+-----------+-------+


1.1 Summary of Registers by Type
--------------------------------

+-------+--------------+-------------+--------------+
| Total | Clock Enable | Synchronous | Asynchronous |
+-------+--------------+-------------+--------------+
| 0     |            _ |           - |            - |
| 0     |            _ |           - |          Set |
| 0     |            _ |           - |        Reset |
| 0     |            _ |         Set |            - |
| 0     |            _ |       Reset |            - |
| 0     |          Yes |           - |            - |
| 16    |          Yes |           - |          Set |
| 181   |          Yes |           - |        Reset |
| 12    |          Yes |         Set |            - |
| 27    |          Yes |       Reset |            - |
+-------+--------------+-------------+--------------+




2. Slice Logic Distribution
---------------------------
```

Site Type	Used	Fixed	Available	Util%
Slice	119	0	8150	1.46
SLICEL	79	0		
SLICEM	40	0		
LUT as Logic	222	0	20800	1.07
using O5 output only	0			
using O6 output only	211			
using O5 and O6	11			
LUT as Memory	0	0	9600	0.00
LUT as Distributed RAM	0	0		
LUT as Shift Register	0	0		
LUT Flip Flop Pairs	133	0	20800	0.64
fully used LUT-FF pairs	7			
LUT-FF pairs with one unused LUT output	126			
LUT-FF pairs with one unused Flip Flop	118			
Unique Control Sets	22			

* Note: Review the Control Sets Report for more information regarding control sets.

3. Memory

Site Type	Used	Fixed	Available	Util%
Block RAM Tile	0	0	50	0.00
RAMB36/FIFO*	0	0	50	0.00
RAMB18	0	0	100	0.00

* Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodate only one FIFO36E1 or one FIFO18E1. However, if a FIFO18E1 occupies a Block RAM Tile, that tile can still accommodate a RAMB18E1

4. DSP

Site Type	Used	Fixed	Available	Util%
DSPs	0	0	90	0.00

5. IO and GT Specific

Site Type	Used	Fixed	Available	Util%
Bonded IOB	21	21	106	19.81
IOB Master Pads	9			
IOB Slave Pads	12			
Bonded IPADs	0	0	10	0.00
Bonded OPADs	0	0	4	0.00
PHY_CONTROL	0	0	5	0.00
PHASER_REF	0	0	5	0.00
OUT_FIFO	0	0	20	0.00
IN_FIFO	0	0	20	0.00
IDELAYCTRL	0	0	5	0.00
IBUFDS	0	0	104	0.00
GTPE2_CHANNEL	0	0	2	0.00
PHASER_OUT/PHASER_OUT_PHY	0	0	20	0.00
PHASER_IN/PHASER_IN_PHY	0	0	20	0.00
IDELAYE2/IDELAYE2_FINEDELAY	0	0	250	0.00
IBUFDS_GTE2	0	0	2	0.00
ILOGIC	0	0	106	0.00
OLOGIC	0	0	106	0.00

6. Clocking

Site Type	Used	Fixed	Available	Util%
BUFGCTRL	1	0	32	3.13
BUFIO	0	0	20	0.00
MMCME2_ADV	0	0	5	0.00
PLLE2_ADV	0	0	5	0.00
BUFMRCE	0	0	10	0.00
BUFHCE	0	0	72	0.00
BUFR	0	0	20	0.00

7. Specific Feature

Site Type	Used	Fixed	Available	Util%
BSCANE2	0	0	4	0.00
CAPTUREE2	0	0	1	0.00
DNA_PORT	0	0	1	0.00
EFUSE_USR	0	0	1	0.00
FRAME_ECCE2	0	0	1	0.00
ICAPE2	0	0	2	0.00
PCIE_2_1	0	0	1	0.00
STARTUPE2	0	0	1	0.00
XADC	0	0	1	0.00

8. Primitives

Ref Name	Used	Functional Category
FDCE	177	Flop & Latch
LUT5	149	LUT
CARRY4	42	CarryLogic
LUT4	30	LUT
FDRE	27	Flop & Latch
LUT2	26	LUT
OBUF	16	IO
FDPE	16	Flop & Latch
LUT6	13	LUT
FDSE	12	Flop & Latch
LUT1	8	LUT
LUT3	7	LUT
IBUF	5	IO
LDCE	4	Flop & Latch
BUFG	1	Clock

9. Black Boxes

Ref Name	Used

10. Instantiated Netlists

Ref Name	Used