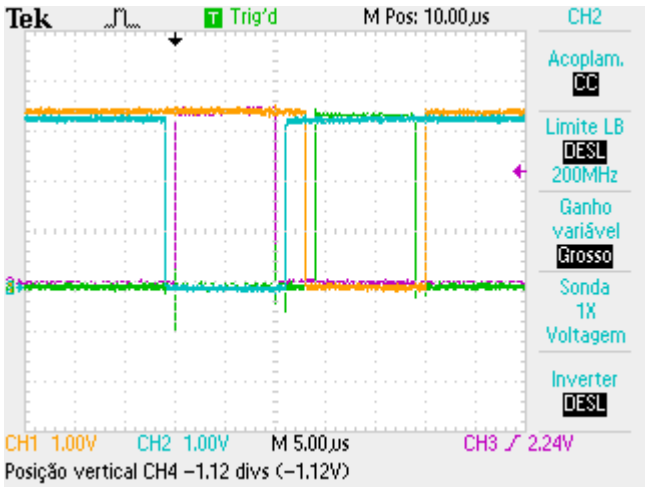
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA
Área Departamental de Engenharia de Sistemas de Potência e Automação



Controlo de Conversores de Potência Genéricos por FPGA

NUNO JOSÉ MACHADO MIQUELINA

(Licenciado)

Dissertação para a obtenção do grau de Mestre em
Engenharia Eletrotécnica – Ramo de Automação e Eletrónica Industrial

Orientadores:
Professor Doutor Fernando Manuel Fernandes Melicio (ISEL)
Professor Doutor Luís Manuel dos Santos Redondo (ISEL)

Júri:
Presidente: Professor Doutor José Manuel Prista do Valle Igreja (ISEL)

Vogais:
Professor Doutor Fernando Manuel Fernandes Melicio (ISEL)
Professora Doutora Maria da Graça Vieira Brito Almeida (ISEL)

Dezembro de 2015

Agradecimentos

Um agradecimento especial ao João Santos, um amigo e companheiro de muitas horas nesta aventura.

Ao Professor Doutor Fernando Melicio e ao Professor Doutor Luís Redondo pela oportunidade de efetuar este trabalho, pela orientação, paciência e motivação para que conseguisse concluir este trabalho.

A todos os elementos da EPS (EnergyPulseSystems) pelo acolhimento nas suas instalações e por garantirem as melhores condições de trabalho.

À Família – Cristina, Tomás e Sofia – por estarem sempre comigo e darem o apoio e carinho tão necessário durante esta jornada.

Sumário

Esta dissertação insere-se na área de investigação e desenvolvida no seio da secção de Automação e Eletrónica, com particular ênfase no âmbito dos circuitos eletrónicos embebidos. Assim, considerando que hoje em dia os conversores eletrónicos de potência são comandados e controlados por circuitos digitais programáveis, mais ou menos configuráveis, para geração dos sinais de comando dos semicondutores e diagnóstico do funcionamento do conversor, pretende-se nesta tese:

a) Apresentar os sinais necessários ao comando e diagnóstico do funcionamento dum conversor de potência genérico, bem como os tipos de circuitos digitais e as técnicas usadas para o efeito;
b) Projetar um circuito de controlo para um conversor de potência genérico, baseado numa lógica de *hardware* configurável do tipo FPGA (*Field Programmable Gate Array*), com interface gráfico com o utilizador, através dum *PLC (Programmable logic controller)*, para:
    a. Programação dos sinais de comando (frequência e fator de ciclo) de um determinado número de semicondutores;
    b. Recolha de sinais de entrada e saídas digitais para controlo de fontes de alimentação;
    c. Acionar proteções;
    d. Aquisição de sinais analógicos de diagnóstico de valores de tensão e corrente;
c) Construir o circuito e testar num conversor de potência do tipo conversor dc-ac (ou dc-impulsos bipolares)
d) Análise dos resultados obtidos com outras soluções existentes no mercado.

Summary

This dissertation is part of the research area and developed within the Automation and Electronics section, with particular emphasis in the context of embedded electronics. Thus, considering that nowadays electronic power converters are operated and controlled by digital programmable circuits, more or less configurable, for generating the control signals of semiconductor and diagnosis of the converter operation, it is intended in this thesis:

a) Provide the necessary command and diagnosis signals of operation of a generic power converter as well as the types of digital circuits and the techniques used for this purpose;
b) Projecting a control circuit for generic power converter based on a configurable hardware logic like FPGA (Field Programmable Gate Array) with graphical user interface through a PLC (Programmable logic controller) to:
    a. Programming of the control signals (frequency and cycle factor) of a certain number of semiconductor;
    b. Collection of input signals and outputs for controlling power supply;
    c. Activate the solution protections;
    d. Acquisition of analog signals diagnostic values of voltage and current;
c) Build and test the circuit in a power converter inverter type dc-ac (or dc-bipolar pulses);
d) Analysis of the obtained results with other existing solutions in the market

Símbolos e Notações

AC – Corrente Alternada (*Alternating Current*)

ADC – *Analog-Digital Converter*

AHDL – *A Hardware Description Language*

CPU – *Central Processing Unit*

DAC – *Digital-Analog Converter*

DC – Corrente Continua (*Direct Current*)

DSP - *Digital Signal Processor*

FPGA - *Field Programmable Gate Array*

GPIO – *General Purpose Input Output*

GTO - *Gate Turn-off Thyristors*

HPL - *Hardware Programming language*

I²C - *Inter-Integrated Circuit*

IGBT - *Insulated Gate Bipolar Transistors*

JTAG - *Joint Test Action Group*

LAB – *Logic Array Blocks*

LE – *Logic Element*

LED - *Light-emitting diode*

LUT – *Look-up Table*

MCU - *Microcontroller*

MOSFET - *Metal Oxide Semiconductor Field Effect Transistor*

MPP - *Maximum Power Point*

MUX - *Multiplexer*

PIT - *Programmable Interval Timer*

PLC - *Programmable logic controller*

PLL – *Phase Locked Loop*

PSU – *Power Supply Unit*

PWM - *Pulse Width Modulation*

RAM – *Random-Access Memory*

RTL - *Register-Transfer Level*

SPI - *Serial Peripheral Interface*

SPS – *Samples per Second*

UART - *Universal Asynchronous Receiver/Transmitter*

USB - *Universal Serial Bus*

VHDL – *VHSIC Hardware Description Language*

VHSIC – *Very High Speed Integrated Circuits*

Organização do Documento

Este documento está organizado em cinco capítulos, divididos da seguinte forma:

I. Introdução: São apresentados neste capítulo as motivações e pretensões deste trabalho, bem como as bases e conceitos utilizados durante esta dissertação. Não se pretende aprofundar cada um dos conceitos, mas sim explicar o seu objetivo e a sua utilidade no projeto;

II. *State-of-the-art*: visão generalista sobre o que atualmente já se faz em termos de controlo e geração de sinais de comando em conversores de potência;

III. Implementação do circuito de comando: Descrição das placas usadas, dos vários módulos e desenvolvimentos efetuados para garantir a geração dos sinais de comando. Serão também descritas as várias interfaces e protocolos codificados;

IV. Resultados: Descrição dos ensaios efetuados para alcançar os objetivos propostos nesta dissertação;

V. Conclusões: Elaboração das conclusões retiradas;

VI. Bibliografia e Referências: Indicação da origem dos diversos conteúdos utilizados neste trabalho de dissertação, dando os créditos a quem de direito pelo sua contribuição nestas áreas de investigação. Este capítulo é o resumo das várias referências indicadas no restante documento;

VII. Anexos: informação diversa que, não sendo específica a qualquer capítulo em particular, compõe este trabalho de dissertação com mais informação.

VIII. Notas: Apontamentos e orientações que possam ser uteis a quem decida continuar este trabalho, melhorando o que pode ser melhorado ou derivar para alternativas não exploradas.

Convenções

Neste documento foram utilizadas as unidades de medida convencionadas pelo Sistema Internacional (SI).

Este documento é redigido em Português e sempre que sejam utilizados estrangeirismos os mesmos serão redigidos em itálico (ex: *Field*).

O presente documento foi redigido segundo o acordo ortográfico em vigor.

Índice

Índice de imagens

Índice de Tabelas

I. Introdução

1 *O Tema*

O tema - Controlo de Conversores de Potência Genéricos por FPGA (*Field Programmable Gate Array)* – e o trabalho realizado, que será exposto neste documento, incide sobre a utilização de uma FPGA na geração de sinais de comando de um conversor de potência – um gerador de Marx bipolar.

A FPGA será programada para gerar sinais (nos seus terminais de GPIO - *General Purpose Input Output*) com dois níveis de tensão, 0V e 3.3V, que colocarão os IGBT's (*Insulated Gate Bipolar Transistors*) do gerador à condução ou ao corte. Esta sequência de comutações sincronizadas dos IGBT's permitirá executar aquele que é o normal funcionamento do gerador de Marx bipolar.

O dinamismo e flexibilidade das FPGA's permitiria que fossem gerados sinais de comando para um qualquer conversor de potência.

Na implementação efetuada foram criados mecanismos que permitem a definição parâmetros de operação, permitindo alterar a frequência dos sinais, os tempos mortos entre si e o período de carga dos condensadores do gerador de Marx, demonstrando assim que a FPGA é um dispositivo adaptável ao controlo de conversores de potência de um modo geral.

Ao longo deste documento serão apresentados os módulos que compõem a programação efetuada na FPGA e que permite não só a geração dos sinais de comando bem como a interface com um operador através de um PLC (*Programmable logic controller*) e um display de interface.

2 *A Motivação*

A escolha e uso de FPGA's neste trabalho prende-se em grande parte pela capacidade que a FPGA têm de controlar as suas entradas/saídas de GPIO, de forma independente ou em paralelo, bem como o número de GPIO's (entradas e saídas) disponíveis para gerar os sinais de comando necessários ao funcionamento dos conversores.

Existem muitos modelos disponíveis no mercado, a custos relativamente reduzidos – fator de escolha importante no ambiente empresarial – que diferem nas suas caraterísticas mais importantes: no número de portas de entrada e de saída, nas velocidades de relógio interno e no número de elementos lógicos disponíveis para desenhar o circuito interno.

No mesmo integrado (a FPGA) será possível desenhar os módulos (circuitos internos) que permitirão efetuar as interfaces com os sistemas externos (PLC's), geração dos sinais de comando e controlo de erros de forma independente, sem que a atividade de um módulo interrompa ou impossibilite a atividade de outro módulo.

As características dos sinais a gerar também são muito específicas, em que os sinais podem estar com tensão de 3.3 V durante 1 us e o restante tempo do período a 0 V. Para frequências de sinais de comando de algumas centenas de Hz, implica que seja gerada uma comutação de sinal na porta de entrada e saída muito rápida (0 V – 3.3.V – 0 V) e a FPGA tem capacidade para responder a estes requisitos.

II. State-of-the-art

3 *Controlo de Conversores de Potência*

3.1 Conversores de Potência

Na Engenharia eletrotécnica, os conversores de potência são os sistemas que permitem converter energia elétrica de uma forma para outra. Essa conversão poderá ser de *AC (Alternating Current)* para *DC (Direct Current)*, entre tensões, entre frequências ou uma combinação destas conversões [1].

Um conversor de potência é um dispositivo elétrico ou eletromecânico usado para converter energia elétrica em outra forma de energia elétrica. Pode ser um simples transformador para alterar a tensão *AC* mas também pode ser um sistema complexo.

Uma maneira de classificar os sistemas de conversão de potência é de acordo com o facto de as suas entradas ou saídas serem em corrente alternada (*AC*) ou em corrente contínua (*DC*).

A tabela seguinte exemplifica a classificação dos conversores segundo as suas entradas e saídas.

Tabela 1 - Classificação de Conversores de Potência

Entradas/Saídas	**AC**	**DC**
AC	• Conversor direto de frequência	• Retificador • *Mains power supply unit (PSU)* • *Switched-mode power supply*
DC	• Transformador/Auto-transformador • Conversor de tensão • Regulador de Tensão • *Cycloconverter* • Transformador de frequência variável	• Conversor DC-DC • Regulador de tensão • Regulador linear

De seguida poderemos observar a mesma informação mas de um modo gráfico.

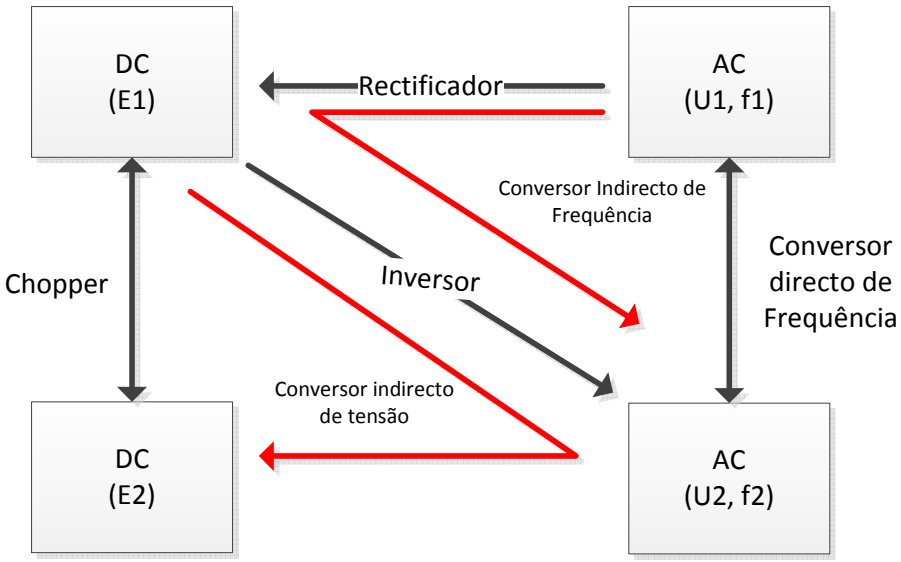


Figura 1 - Classificação de Conversores de Potência

A Figura 1 exemplifica outra forma gráfica de representar a classificação de conversores de potência, podendo existir conversões indiretas (passagem por um sistema intermédio).

No contexto do trabalho realizado e na utilização do gerador de Marx bipolar (que será descrito mais à frente neste trabalho), foi elaborado um sistema que permitisse gerar os sinais de comando necessários para que o gerador operasse. Estes sinais de comando são parte integrante dos conversores de potência, uma vez que sincronizam os diversos dispositivos do conversor dando indicação de quando devem de executar uma tarefa (por exemplo de estarem à condução ou ao corte).

3.2 Dispositivos de comando – Microcontroladores, DSP e FPGA

Neste capítulo serão descritos algumas tecnologias usadas no comando de conversores de potência, nomeadamente Microcontroladores, DSP's (*Digital Signal Processor*) e FPGA's.

3.2.1 Microcontroladores

3.2.1.1 Descrição

Um microcontrolador [2] (que por vezes é abreviado como µC, uC ou MCU - *Microcontroller*) é um pequeno computador embebido num único circuito integrado, contendo um núcleo de processamento, memória e periféricos de entrada/saída programáveis. Os microcontroladores são desenhados para aplicações embebidas, em contraste com os microprocessadores usados nos computadores pessoais e outras aplicações genéricas.

Os microcontroladores são usados em produtos e dispositivos controlados automaticamente, como os sistemas de controlo de automóveis, *appliances*, brinquedos e em outros sistemas embebidos. Por reduzir a dimensão e o custo comparado com uma implementação que use em separado um microprocessador, memoria e dispositivos de entrada/saída, os microcontroladores tornam economicamente vantajoso o controlo digital de ainda mais dispositivos e processos. É comum a utilização de microcontroladores de sinal misto, integrando componentes analógicas necessárias para o controlo de sistemas não digitais.

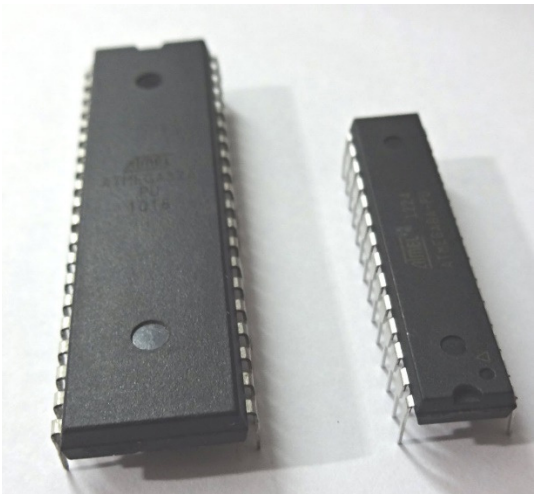


Figura 2 - Microcontroladores da ATMega [2]

Alguns microcontroladores podem usar palavras de quatro bits e operarem a taxas de frequência tão baixas como 4kHz, para um baixo consumo de energia (*miliwatts* ou *microwatts*). Os microcontroladores têm normalmente a capacidade de manter funcionalidade enquanto esperam por um evento como o premir de um botão ou outra interrupção; o consumo de energia enquanto está neste estado de adormecido (o relógio do CPU (*Central Processing Unit*) e muitos dos periféricos desligados) pode ser de apenas alguns *nanowatts*, tornando muitos dos microcontroladores uma peça fundamental para aplicações dependentes de baterias.

Os microcontroladores normalmente contêm de alguns a dúzias de pins de *input/output* genéricos (GPIO). Os pins de GPIO são configurados por *software* para um estado de input ou para um estado de output. Quando os pins GPIO são configurados para um estado de *input*, eles são normalmente usados para ler sensores ou sinais externos. Configurados para o estado de output, os pins GPIO podem controlar dispositivos externos como LED's (*Light-emitting diode*) ou motores que muitas vezes são controlados indiretamente através de eletrónica de potência externa.

Muitos sistemas embebidos necessitam de ler sensores que produzem sinais analógicos. Este é o propósito do conversor analógico-digital (ADC - *Light-emitting diode*). Uma vez

que os processadores são construídos para interpretar e processar dados digitais, isto é 1s e 0s, não são capazes de fazer nada com os sinais analógicos que lhe podem estar a ser enviados por um dispositivo. Assim o conversor analógico digital é usado para converter os dados que lhe estão a chegar para um formato que o processador possa reconhecer. Uma funcionalidade menos comum em alguns microcontroladores é um conversor digital-analógico (DAC - *Digital-Analog Converter*) que permite ao processador enviar sinais analógicos ou níveis de voltagem.

Em adição os conversores, muitos microcontroladores embebidos incluem também uma variedade de *timers*. Um dos mais comuns tipos de timers é o *Programmable Interval Timer* (PIT). Um PIT pode tanto decrementar de um valor para 0, ou incrementar até à capacidade do registo de contagem, que ultrapassado passa a 0. Uma vez que atinge o 0, ele envia uma interrupção para o processador indicando que terminou a contagem. Isto é útil para dispositivos como termostatos, que periodicamente testam a temperatura à sua volta para verificar se há a necessidade de ligar o ar condicionado ou outro dispositivo, por exemplo.

Um bloco dedicado de *Pulse Width Modulation* (PWM) torna possível ao CPU controlar conversores de potência, cargas resistivas, motores, etc., sem usar muitos recursos do CPU.

O bloco de *Universal Asynchronous Receiver/Transmitter* (UART) torna possível receber e transmitir dados sobre uma linha série com muita pouca carga no CPU. *Hardware* dedicado *on-chip* incluem muitas vezes capacidade de comunicação com outros dispositivos (circuitos integrados) em formatos digitais com por exemplo *Inter-Integrated Circuit* (I²C), *Serial Peripheral Interface* (SPI), *Universal Serial Bus* (USB) e *Ethernet*.

3.2.1.2 Utilização do microcontrolador

As aplicações dos microcontroladores no comando de conversores ou de outros sistemas de energia são variados, existindo um elevado número de estudos que revelam a adaptabilidade destes dispositivos em aplicações reais. Iremos aqui ilustrar apenas alguns exemplos que penso serem ilustrativos da utilização dos microcontroladores.

Controlo por Microcontrolador de um Conversor de Potência [3]

Neste trabalho realizado os autores têm como objetivo a substituição de sistemas de controlo analógicos (com módulos de potência composto por *thyristors*) por sistemas totalmente digitais, controlados por microcontrolador. Neste estudo foram usados IGBT's e MOSFET's (*Metal Oxide Semiconductor Field Effect Transistor*) para componente de *switching*. O microcontrolador é usado para gerar os sinais de comando para os IGBT's ou MOSFET's, colocando-os à condução ou ao corte.

Controlo de um conversor AC-DC [4]

Neste estudo é apresentado um modo de operação em paralelo de controlo de um módulo conversor de AC-DC via um *serial communication bus*. No sistema que é proposto, múltiplos conversores AC-DC são ligados em paralelo na saída para partilharem a corrente da carga. Cada módulo do conversor AC-DC é controlado individualmente por um microcontrolador.

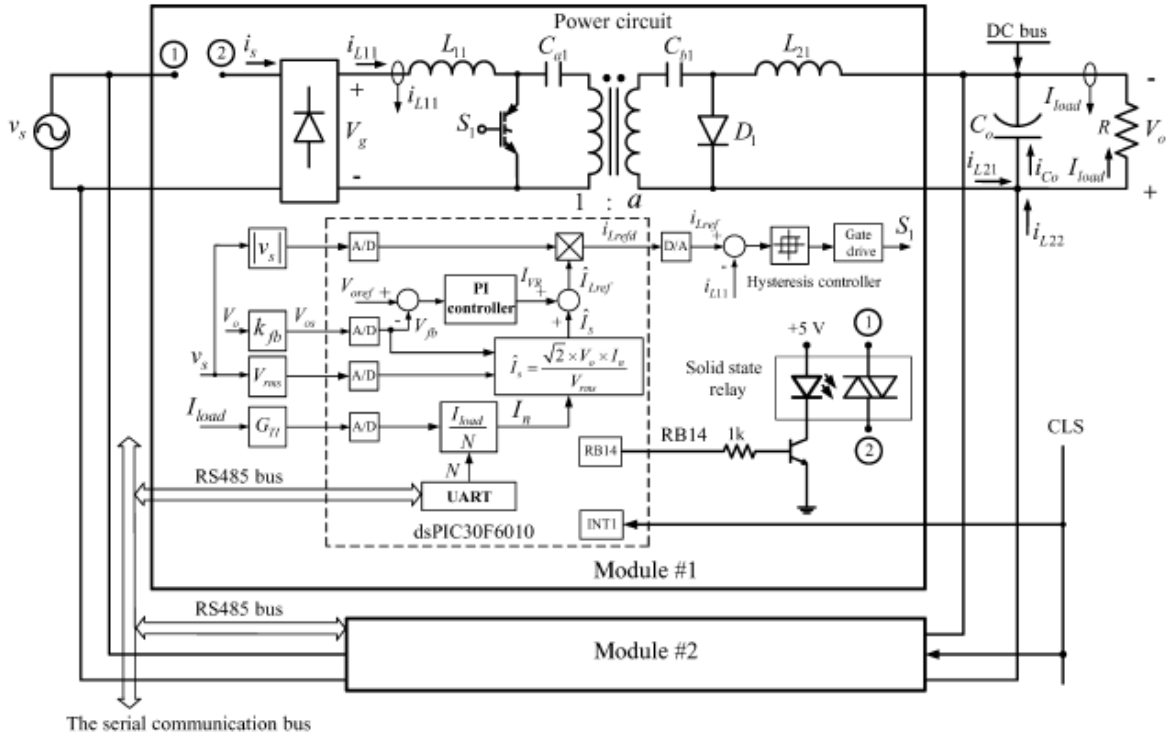


Figura 3 - Sistema proposto de um conversor AC-DC modular [4]

Podemos ver na Figura 3, partilhada pelos autores, o papel do microcontrolador (neste caso um dsPIC30F6010) num módulo do conversor. De notar a existência de um acoplamento ótico, necessário para que haja um isolamento do circuito de controlo e do circuito de potência.

Novamente o microcontrolador é utilizado na geração dos sinais de comando para *switch* (que na Figura 3 está identificado como S1 - MOSFET).

<u>Controlo de sistemas fotovoltaicos</u> [5]

Uma instalação isolada típica de um sistema fotovoltaico consiste em painéis fotovoltaicos, um regulador, baterias e um inversor. O regulador é o elemento ligado entre os painéis fotovoltaicos e as baterias, sendo a sua missão basicamente manter as baterias carregadas e evitar a sua sobrecarga.

No trabalho exposto pelos autores é proposto um novo regulador que pode operar no *Maximum Power Point* (MPP) de qualquer *array* fotovoltaico, independentemente de temperaturas ou condições meteorológicas e os seus efeitos na dispersão das características do *array* fotovoltaico. A ligação em paralelo destes reguladores também é proposta, que permite aumentar a instalação, mantendo os reguladores já instalados.

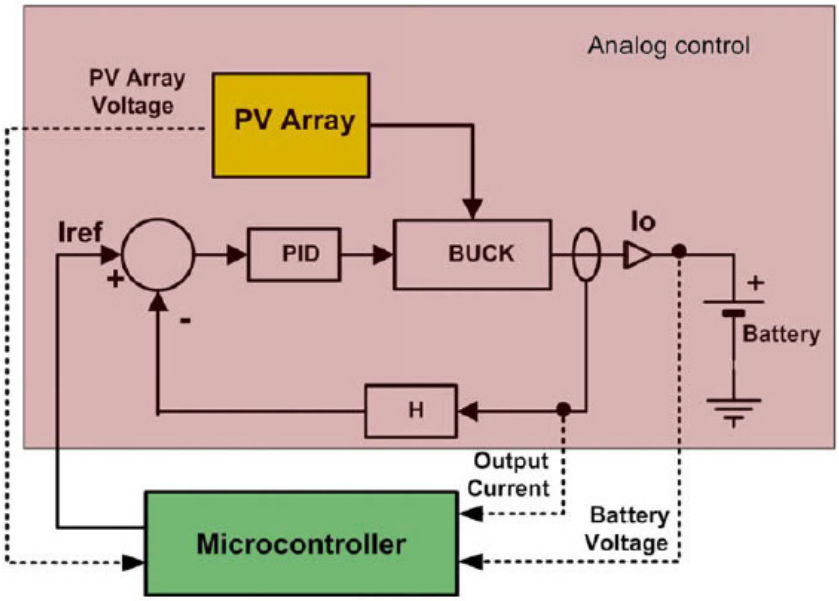


Figura 4 - Diagrama de blocos do sistema proposto [5]

Na Figura 4 podemos observar o papel do Microcontrolador no sistema proposto.

Neste trabalho o sinal de comando é uma corrente de referência que servirá para que o sistema reaja e minimize o erro entre a corrente de referência e o output.

3.2.2 DSP

3.2.2.1 Descrição

Um *Digital Signal Processor* (DSP) [6] é um microcontrolador especializado, em que a sua arquitetura é otimizada para as necessidades de operação no processamento digital de sinais.

O objetivo do DSP é normalmente a medição, filtragem e/ou compressão dos sinais analógicos do mundo real. Muitos dos microprocessadores de utilização geral também conseguem executar o processamento de sinais digitais com sucesso, mas DSP's dedicados têm normalmente mais eficiência energética. Esta eficiência leva a que os DSP sejam muito utilizados nos terminais móveis devido às suas limitações de consumo de energia. Os DSP's usam muitas vezes arquiteturas especiais de memória que são capazes de obter múltiplos dados e/ou instruções ao mesmo tempo.

Figura 5 - Sistema típico de processamento de sinal [6]

Normalmente os DSP's são circuitos integrados dedicados, no entanto as funcionalidades dos DSP's também podem ser produzidas utilizando uma FPGA.

3.2.2.2 Utilização do DSP

As aplicações dos DSP's no controlo de conversores ou de outros sistemas de energia são variados, existindo um elevado número de estudos que revelam a adaptabilidade destes dispositivos em aplicações reais. Iremos aqui ilustrar apenas alguns exemplos que penso serem ilustrativos da utilização dos DSP's.

Controlo de um *Intelligent Universal Transformer* via DSP [7]

Um *Intelligent Universal Transformer* é um equipamento de conversão de energia elétrica inteligente e com multifacetado, o qual é baseado num conversor eletrónico de potência. Este equipamento combina o transformador, inversor, sensores, monitores, *phase shifter* e o compensador de energia reativa num só, tendo assim vantagens que o transformador tradicional não tem.

Os autores deste artigo estudaram, para além do *Intelligent Universal Transformer*, o controlo do mesmo através de um sistema utilizando um DSP que é baseado no TMS320C28343.

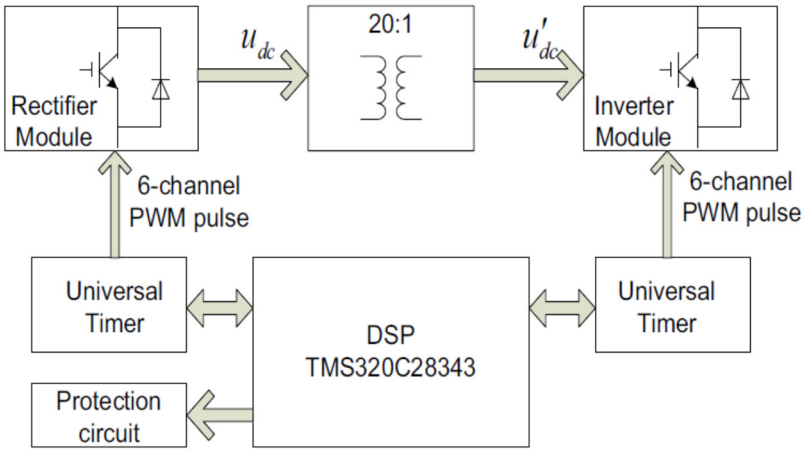


Figura 6 - Modelo do sistema projetado [7]

Podemos ver na Figura 6 a aplicação do DSP no sistema projetado.

A Figura 6 é possível observar que o DSP terá a responsabilidade de gerar um final PWM e comandará o sistema.

Controlo de um conversor *Half-Bridge Single-Stage Converter* via DSP [8]

Os autores deste estudo apresentam um novo método de controlo de um conversor de potência *half-bridge single-state*. Neste estudo usam um DSP (TMS320F2812 da *Texas Instruments*) para controlar a variação da frequência de comutação do conversor.

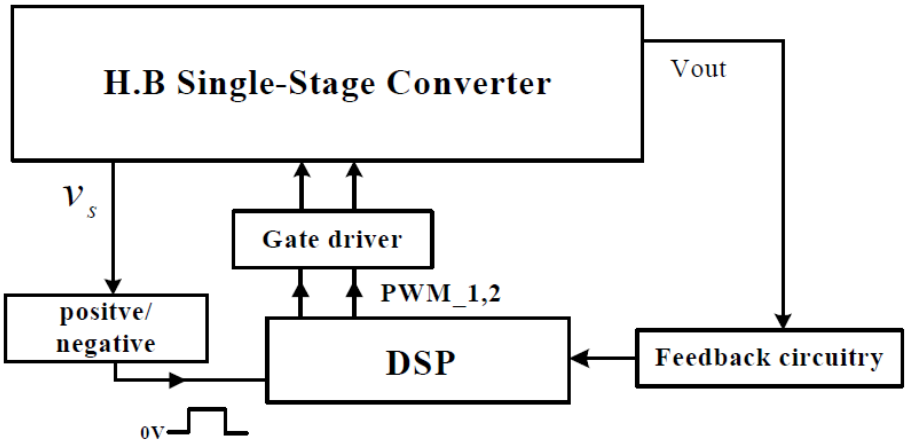


Figura 7 - Diagrama de blocos do conversor [8]

Podemos ver na Figura 7, no diagrama de blocos, a utilização do DSP para controlar a frequência de comutação, comandando assim o sistema.

Inversor com controlo por DSP [9]

Os autores deste estudo construíram um inversor para ser aplicado em sistemas fotovoltaicos, em que o controlo é efetuado por um DSP (TMS320F2812).

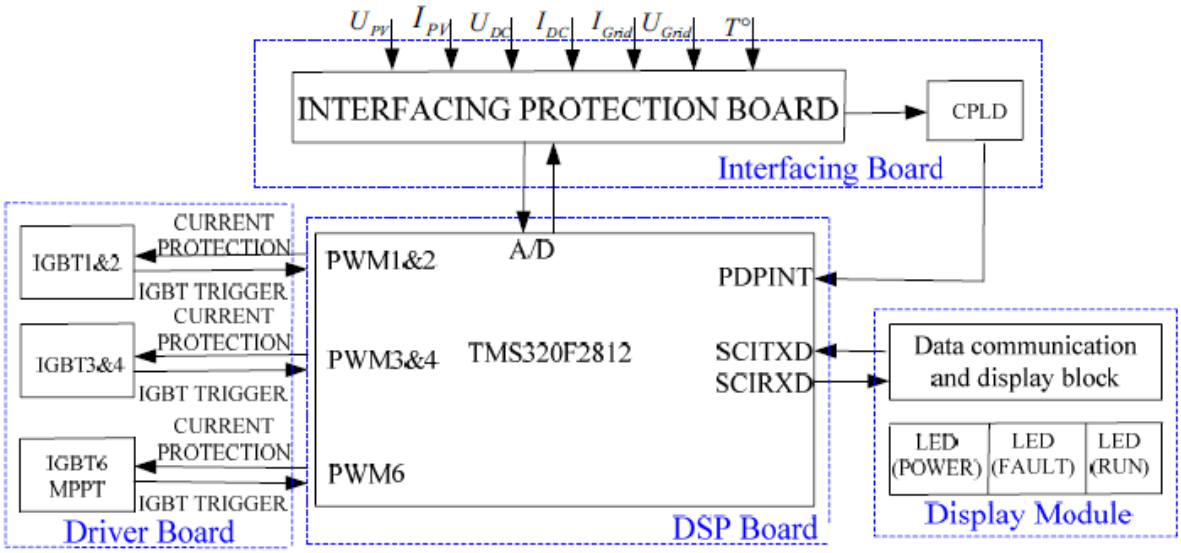


Figura 8 - Diagrama de blocos do inversor [9]

O DSP recebe sinais do sistema de energia, via a placa de interface, e conforme os sinais recebidos e gera os sinais de PWM que irão comandar o inversor. Os resultados foram os pretendidos conseguindo um sinal de saída com conteúdos mínimos de harmónicas.

Neste estudo o DSP também é responsável para comunicação série com um *display*.

Uso de DSP em carros elétricos e híbridos [10]

Os autores deste estudo sugerem o uso de DSP na monitorização e deteção de falhas do motor elétrico (ou hibrido) de um carro.

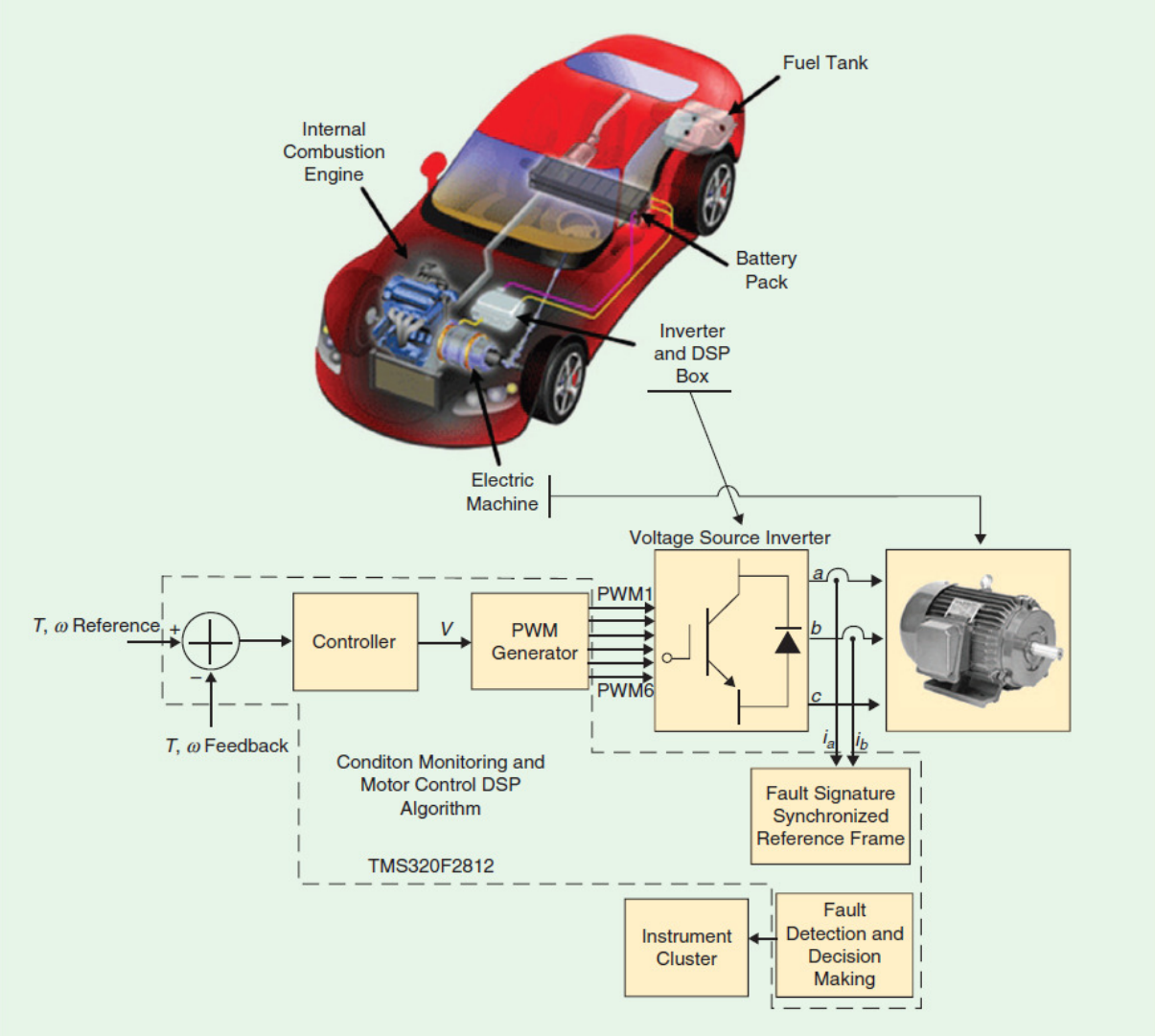


Figura 9 - Uso de DSP na indústria automóvel [10]

Os autores rematam no final que a monitorização da condição do motor e a deteção de falhas é vital para a segurança e para uma manutenção com menores custos. Esta monitorização é possível em *real-time* com este tipo de controlo via DSP.

3.2.3 FPGA

3.2.3.1 Descrição

A FPGA foi a tecnologia escolhida para este projeto, sendo descrita mais detalhadamente no ponto 2 do capítulo "VI Anexos". Serão também identificadas as qualidades deste tipo de tecnologia, que levaram à sua seleção neste trabalho.

3.2.3.2 Utilização da FPGA

As aplicações das FPGA's no controlo de conversores ou de outros sistemas de energia são variados, existindo um elevado número de estudos que revelam a adaptabilidade destes dispositivos em aplicações reais. Iremos aqui ilustrar apenas alguns exemplos que penso serem ilustrativos da utilização dos FPGA's.

O objetivo do trabalho realizado será demonstrar que é possível a utilização de uma FPGA para gerar os sinais de comando dos dispositivos de comutação de um conversor de potência, que no caso estudado é um gerador de Marx, permitido o funcionamento do mesmo conforme a sua implementação.

Controlo de um conversor de potência via FPGA [11]

Os autores deste estudo fazem um estudo de comparação entre várias estratégias de controlo de um conversor de potência através de uma FPGA. Em particular, a comparação foca-se tanto na performance como na complexidade da implementação com a FPGA. São analisadas e comparadas tanto estratégias de controlo não-linear como linear.

O objetivo do estudo foi de identificar a melhor aproximação para explorar a tecnologia da FPGA no controlo dos conversores de potência.

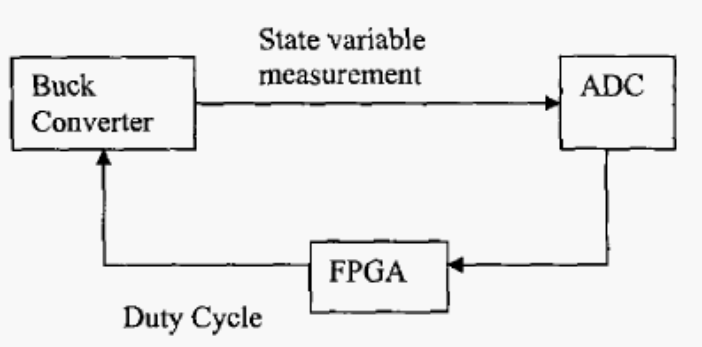


Figura 10 - Diagrama de blocos do conversor controlado por FPGA [11]

Na Figura 10 podemos ver, de uma forma simplificada, o uso da FPGA no sistema de conversão de potência.

FPGA e Unit Power Factor [12]

Neste estudo os autores desenham um sistema hibrido de controlo via FPGA e DSP, para um conversor trifásico. Os autores propõem este sistema para reduzir a poluição harmónica na rede de distribuição e atingir um fator de potência unitário.

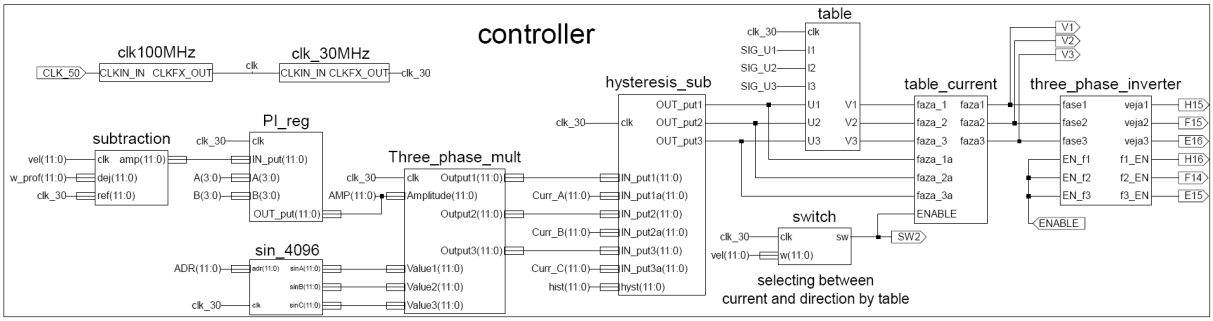


Figura 11 - Esquema VHDL do controlador por FPGA [12]

Esta é outra aproximação de controlo de conversores de potência, usando um sistema hibrido entre FPGA e DSP.

3.2.4 *FPGA* vs. Microcontrolador [24]

Existem muitas referências e comparações entre a FPGA e os Microcontroladores, em que são apresentadas as vantagens e desvantagens de cada um.

As tabelas seguintes podem ilustrar essas comparações:

Tabela 2 - FPGA vs Microsontrolador

	Microcontroller	*FPGA*
Advantages	• *Easy to program* • *Guaranteed Reliability* • *Power Saver mode* • *Easy to change design functionality* • *Short time-to-market*	• *Higher performance* • *Exact functionality that is needed is provided* • *Lower Cost*
Disadvantages	• *Paying for functionality that is not being used* • *More costly* • *Lower performance*	• *Defficult to design and debug* • *Constant power usage* • *Harder to change design functionality* • *Long time-to-market*

Para o trabalho em questão os pontos que mais pesaram na escolha da FPGA foram:

- "Higher performance": a flexibilidade de desenhar um circuito dentro da FPGA e poder ter componentes a operar totalmente em paralelo é uma grande vantagem;
- "Exact functionality that is needed is provided": A FGPA foi programada para fazer exatamente o que era necessário.

III. Implementação do circuito de comando

4 O Gerador de Marx bipolar

4.1 Descrição simplificadas de um gerador de Marx

O conversor utilizado neste estudo foi um Gerador de Marx bipolar, sendo o objetivo deste trabalho gerar os sinais de comando necessários, gerados a partir de uma FPGA. Este comando é feito através da geração de sinais de disparo dos IGBT's dos diversos estágios do gerador.

Um gerador de Marx é um conversor DC/impulsos (positivos e/ou negativos), que operando com relativamente baixas potências médias de entrada consegue gerar um impulso com potências de pico elevadas durante fracções de segundo (e.g. para potência média de entrada de alguns kW pode gerar transitoriamente à saída, durante dezenas de microssegundos tensões de dezenas de kV e centenas de amperes, com taxas de repetição de dezenas de hertz). Para conseguir isto, o circuito é normalmente alimentado por uma tensão relativamente reduzida, que carrega um determinado número de condensadores em paralelo, durante um tempo relativamente longo (i.e. milissegundos), e depois aplica a tensão dos condensadores ligados em série, durante um tempo bastante mais curto, a uma carga.

Esta passagem dos condensadores de paralelo para série é executada recorrendo a dispositivos de comutação, como por exemplo IGBT's, que sincronizados pelo sistema de controlo permitem que numa primeira fase os condensadores estejam a carregar em paralelo e numa segunda fase descarreguem a sua energia na carga.

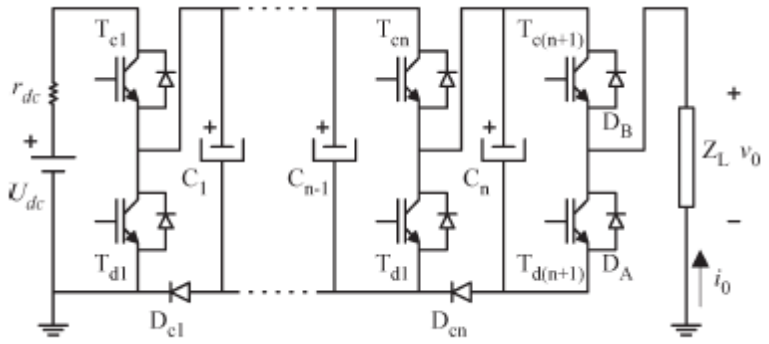


Figura 12 - Esquema simplificado de um gerador de Marx (*solid state*) [13]

Na figura 12 podem ver-se os condensadores Ci e os *IGBTs* Tci e Tdi, respectivamente, para colocarem os condensadores em paralelo com a fonte de tensão Udc e em série com a carga ZL, aplicado uma tensão negativa -nUdc.

Os sinais de controlo dos *IGBTS* (que se apresentam com exemplo no circuito da figura 12) serão os sinais alvo do estudo deste trabalho, sendo que o gerador de Marx utilizado terá vários módulos de disparo e é um gerador de Marx bipolar, pois gera à saída impulsos de tensão positivos e/ou negativos.

4.2 Gerador de Marx bipolar

O gerador de Marx bipolar permite que sejam aplicados à carga impulsos positivos e negativos. Esta particularidade eleva a complexidade do sistema, uma vez que é necessário um número superior de interruptores (i.e. IGBTs), como se mostra na figura 13 [14].

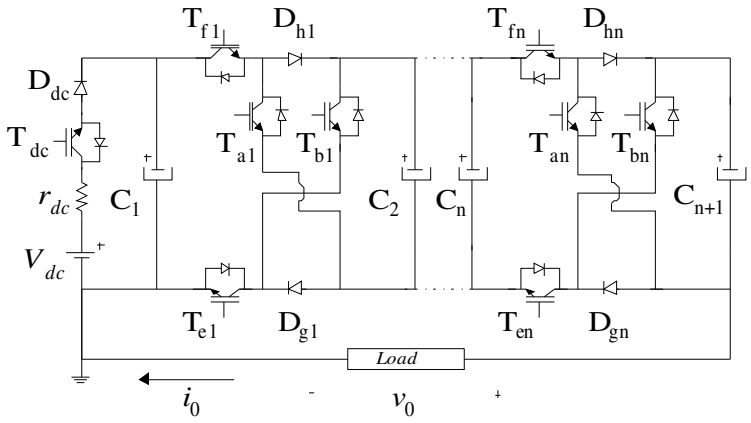


Figura 13 - Topologia simplificada do gerador de Marx bipolar - 4 IGBT's [14]

Na Figura 13 é apresentada uma tipologia simplificada do gerador de Marx bipolar, onde podemos encontrar apenas 4 IGBT's em cada de carregamento. O circuito pode apresentar pelo menos três modos de funcionamento: carga dos condensadores, impulsos positivo e impulsos negativo.

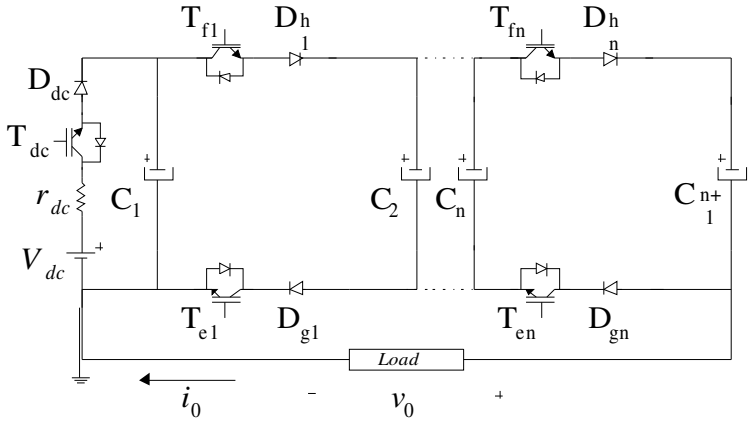


Figura 14 - Gerador de Marx em modo de carga [14]

Quando o gerador está em carga, os condensadores estão a ser carregados em paralelo. Neste caso os IGBT's T_{dc}, T_{f1} a T_{fn} e T_{e1} a T_{en} estão *ON* para permitir o carregamento dos condensadores.

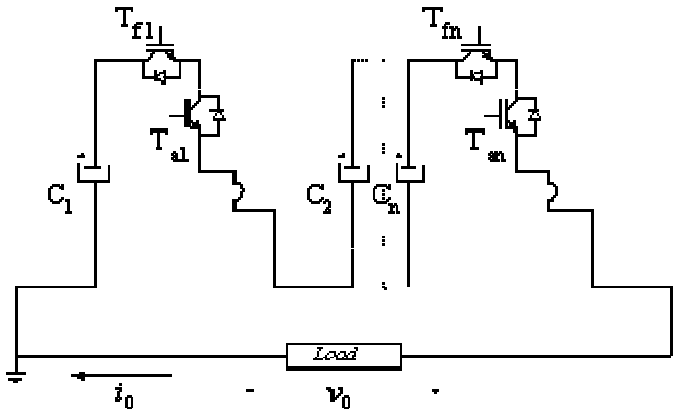


Figura 15 - Gerador de Marx - modo de impulso positivo [14]

Durante o impulso positivo (observado na Figura 15) a fonte é isolada, e os condensadores são colocados em série com a carga. Para que o impulso seja gerado, os IGBT's T_{f1} a T_{fn} e T_{a1} a T_{an} estão *ON*.

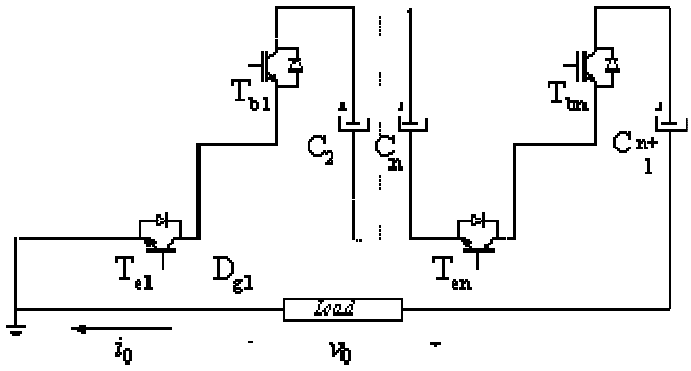


Figura 16 - Gerador de Marx - modo de impulso negativo [14]

Durante o impulso negativo (observado na Figura 16) é a vez dos IGBT's T_{e1} a T_{en} e T_{b1} a T_{bn} estarem *ON* permitir a passagem de corrente para a carga.

4.3 Sinais de disparo

Com a informação descrita na secção anterior, é fácil de observar a necessidade de gerar cinco sinais de controlo (por célula ou módulo) para permitir a operação do gerador de Marx bipolar.

A tabela seguinte irá resumir a utilização de cada um dos sinais de controlo dos IGBT's.

Tabela 3 - Sinais de controlo dos IGBT's

Sinais/Modos	*Repouso*	*Carga*	*Impulso positivo*	*Impulso Negativo*
T_{dc}	OFF	ON	OFF	OFF
T_{fi}	OFF	ON	ON	OFF
T_{ei}	OFF	ON	OFF	ON
T_{ai}	OFF	OFF	ON	OFF
T_{bi}	OFF	OFF	OFF	ON

Iremos observar neste trabalho que a comutação dos IGBT's não é em simultâneo entre os vários modos (repouso, carga, impulso positivo e impulso negativo). Serão introduzidos tempos mortos entre a passagem de modo, de forma a garantir que não existem curto-circuitos, bem como tempos de relaxamento entre os impulsos positivos e negativos.

5 Placa de Controlo dos sinais de disparo

5.1 Introdução

O controlo dos sinais de disparo foi desenhado recorrendo a uma FPGA, que implementa os interfaces necessários, a lógica e o controlo de disparo e mecanismos de segurança, que garantam a proteção da máquina e das pessoas que a operam.

Nos próximos capítulos será descrito o funcionamento da placa e as interfaces entre si e entre o PLC.

5.2 Inputs e Outputs

A placa onde está embebida a FPGA foi desenhada para receber as seguintes entradas e saídas, descritas na tabela seguinte:

Tabela 4 - Inputs e Outputs da Placa de controlo

Direcção[1]	Nome	Descrição
I	Power	Fornece energia à placa, que pode ser alimentada por uma tensão entre os 9V e os 18V (CC). Esta tensão de entrada alimenta um conversor CC/CC (REC6-1215SRW/R10/A) que fornece um output de 15V. Esta tensão passa depois por um regulador de tensão (UA7805 da Texas Instruments) que gera um output fixo de 5V. Estes 5V alimentam a placa da Altera (DE0-Nano) e o MAX232.

[1] (I) . Input; (O) – Output; (I/O) Input/Output

Direcção[1]	Nome	Descrição
O	RJ45	Esta interface permite a ligação ao PLC, via cabo standard RJ45, e internamente a um MAX232. As saídas e entradas do MAX232 serão fornecidas à FPGA por meio das ligações GPIO disponíveis. O protocolo usado para a comunicação entre o PCL e a FPGA
I/O	GPIO	Permite, via encaixe direto e via "Flat Cable" aceder aos GPIO disponibilizados pela placa da Altera (DE0-Nano).
I/O	RJ45	A placa disponibiliza interfaces 3xRJ45 que fazem a ligação às placas de disparo. Em cada uma destas ligações passam os seguintes sinais: • T1 – Corresponde ao sinal T_f • T2 – Corresponde ao sinal T_e • T3 – Corresponde ao sinal T_a • T4 – Corresponde ao sinal T_b
I	Fibra Óptica	Entradas de Fibra óptica que recebem informação de erro das placas de disparo.

Na Figura 17, que representa a modelo 3D da placa de controlo, foram colocadas as interfaces (principais) da seguinte forma:

- A : Interfaces RJ45 para as placas de disparo;
- B : Entradas de erro em fibra ótica;
- C: Interface RJ45 para o PLC;
- D: Localização da placa Altera De0-Nano (FPGA)
- E: Alimentação
- F: GPIO

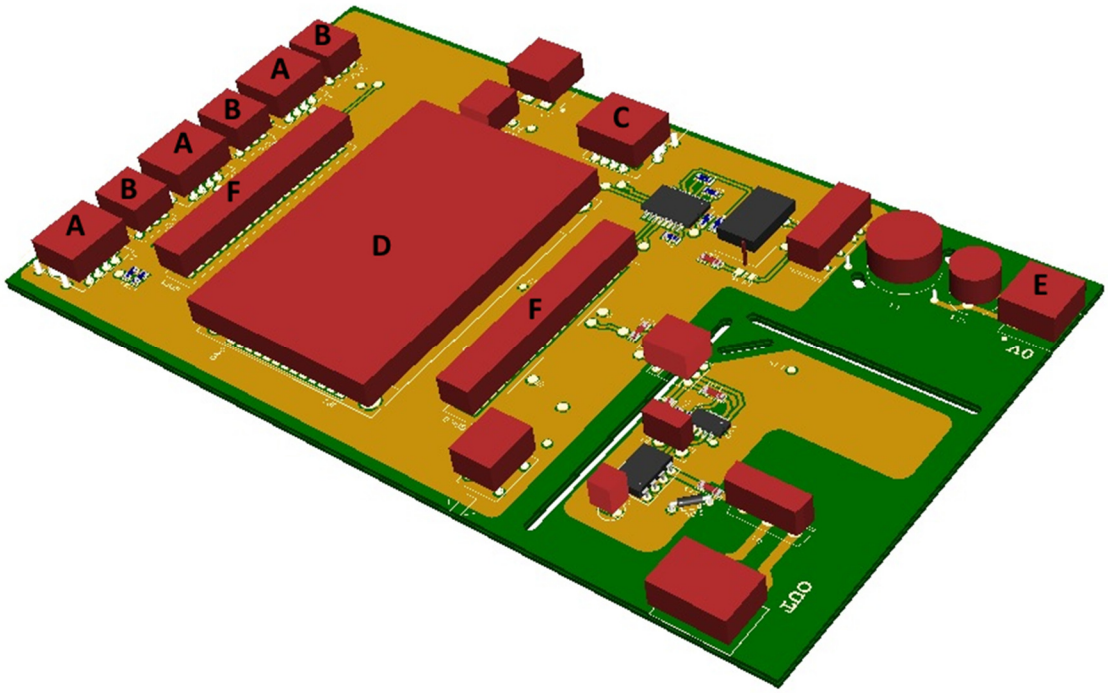


Figura 17- Modelo 3D da placa de comando

Podemos observar na Figura 17 a localização dos diversos componentes de I/O, de maior relevância, da placa desenvolvida no âmbito deste estudo.

6 Placa de disparo

6.1 Descrição

A placa de disparo foi desenvolvida para receber os sinais de disparo provenientes da placa de controlo e replica-lo para as diversas saídas de fibra ótica que dispõe. A réplica dos sinais é efetuada por *hardware*.

O I/O da placa é o seguinte:

Tabela 5 - I/O da placa de disparo

Direcção	Nome	Descrição
I	Power	Fornece energia à placa, alimentada por uma tensão de 24V (CC).
I	RJ45	A placa disponibiliza uma entrada RJ45 que faz a interface com a placa de controlo. Nesta ligação passam os seguintes sinais: • T1 – Corresponde ao sinal T_f • T2 – Corresponde ao sinal T_e • T3 – Corresponde ao sinal T_a • T4 – Corresponde ao sinal T_b
O	Fibra Ótica	Saída de fibra ótica que emite informação de erro da placa de disparo.
O	Fibra Ótica	Saída dos sinais de disparo para os IGBT's.

Na Figura 18, que representa a modelo 3D da placa de disparo, foram colocadas as interfaces (principais) da seguinte forma:

- A: Fibras óticas que emitem os sinais de disparo para os IGBT's;
- B: Alimentação;
- C: Interface RJ45 para a placa de controlo;

- D: Fibra ótica para emissão do sinal de erro.

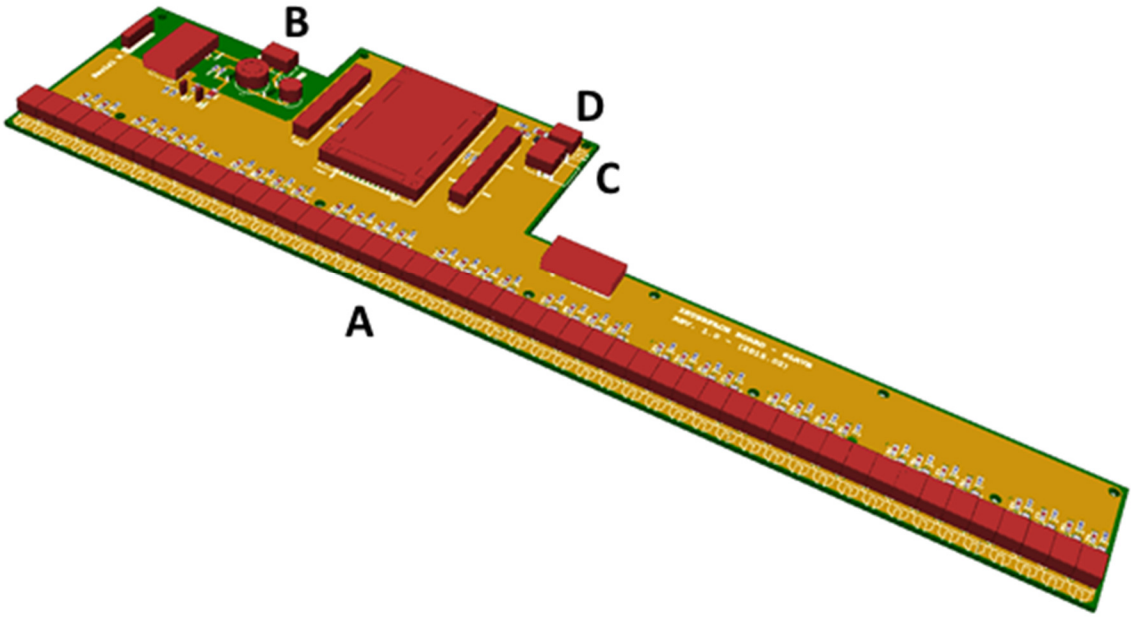


Figura 18 - Modelo 3D da placa de disparo

Podemos observar na Figura 18 a localização dos diversos componentes de I/O, mais relevantes, da placa desenvolvida no âmbito deste estudo.

7 Implementação de FPGA em VHDL

7.1 Introdução

A implementação efetuada na FPGA é modular, em que cada módulo tem funções específicas. Neste capítulo iremos descrever em detalhe os diversos módulos e as suas funcionalidades.

Os módulos foram todos desenvolvidos em VHDL (*VHSIC Hardware Description Language - Very High Speed Integrated Circuits*) no âmbito deste trabalho, com exceção dos módulos de PLL e RAM/Memory que são funções disponibilizadas pelo fabricante do circuito integrado da FPGA.

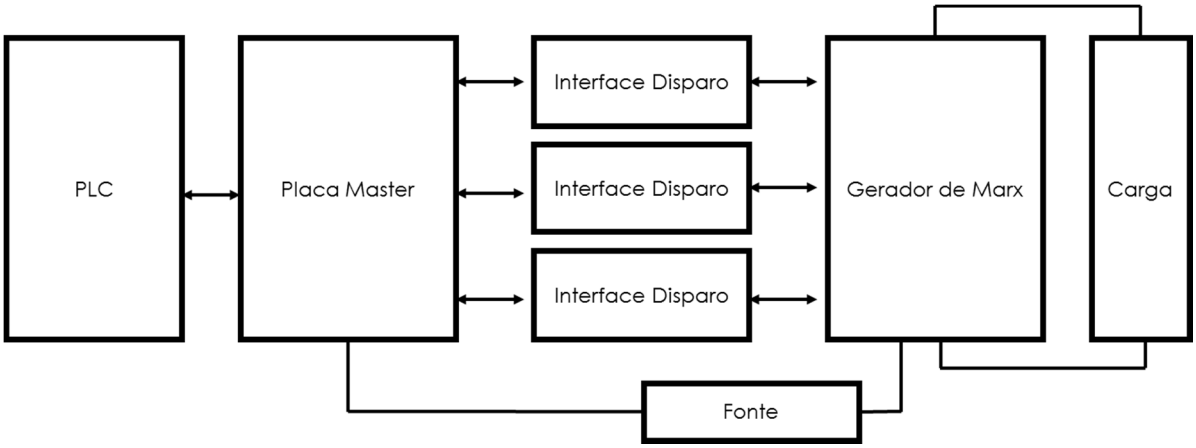


Figura 19 -- Esquema global da solução implementada

A solução implementada indide sobre o bloco “Placa Master” que podemos ver na imagem anterior.

Em termos globais, a implementação na FPGA tem os seguintes módulos, descritos na tabela seguinte.

Tabela 6 - Descrição dos módulo da FPGA

Módulo	Descrição
ModBus	Módulo responsável por fazer a interface com o PLC via ModBus sobre ligação física RJ45/RS232.
ADC	Módulo responsável pelas leituras analógicas.
PLL	(*Phase Locked Loop*) Módulo responsável por gerar sinais de relógio a diferentes frequências sincronizados entre si.
Memory	Memória interna da FPGA, usada para guardar valores das leituras e configurações dos disparos.
Signal Generator	Módulo responsável pela geração dos sinais de disparo para as 3 placas de disparo usadas neste projeto.

O fato de usarmos uma metodologia modelar no desenvolvimento do comando e geração de sinais de disparo, para além de uma boa prática, é também flexível para futuros melhoramentos da solução e reaproveitamento de código.

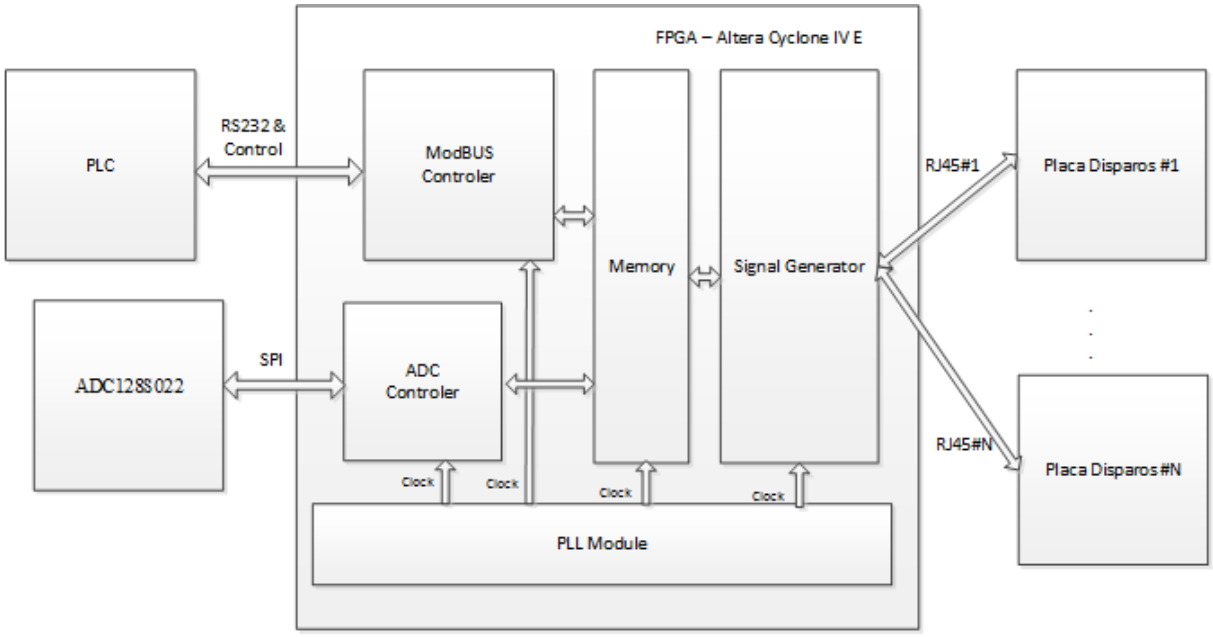


Figura 20 - Diagrama de blocos da implementação na FPGA

Podemos observar na Figura 20 um diagrama de blocos representativo da implementação efetuada na FPGA e que é responsável pela geração e controlo dos sinais de disparo.

7.2 VHDL

O VHDL [15] é uma linguagem descritiva de *Hardware*, usada para programar a estrutura, implementação e operação de circuitos eletrónicos (em que os mais comuns são os circuitos lógicos digitais).

As linguagens HDL (*Hardware Description Language*) [16] para circuitos digitais são muitas (proprietárias ou não) que passam por exemplo por:

- AHPL - *A Hardware Programming language*;
- Verilog
- ParC (Parallel C++)

- ESys.net (uma framework .net – Microsoft)

O VHDL é um *standard* (definido pelo *Standard IEEE 1076*) e é normalmente usada para escrever módulos de texto que descrevem um circuito lógico. O VHDL tem processos que gerem o paralelismo inerente à implementação de *hardware*.

Foi escolhido o VHDL na implementação deste projeto devido aos seguintes aspetos:

- É uma das mais reconhecidas linguagens de HDL;
- Forte comunidade *on-line* que desenvolve componentes com esta linguagem;
- Permite que o comportamento do sistema requerido possa ser descrito (modelado) e verificado (simulado) antes que seja traduzido (compilado) para o *hardware* real:
- Permite a descrição de um sistema concorrencial, ao contrário de outras linguagens de computação que são executadas sequencialmente;
- Permite a reutilização dos blocos desenvolvidos, funcionando como uma biblioteca de recursos;
- O VHDL é portátil, permitindo que a descrição efetuada possa ser utilizada noutra tecnologia.

Como exemplo de uma implementação de uma porta AND:

```
-- (this is a VHDL comment)

-- import std_logic from the IEEE library
library IEEE;
use IEEE.std_logic_1164.all;

-- this is the entity
entity ANDGATE is
  port (
    I1 : in std_logic;
    I2 : in std_logic;
    O  : out std_logic);
end entity ANDGATE;

-- this is the architecture
architecture RTL of ANDGATE is
begin
  O <= I1 and I2;
end architecture RTL;
```

Mesmo sem grandes conhecimentos da linguagem é possível observar que são definidos as portas de entrada (in) e as de saída (out) e a lógica a ser implementada, ou seja, a saída é o produto de uma operação "*and*" entre as duas entradas.

7.3 ModBUS

O módulo de ModBUS é responsável pela comunicação entre o PLC e a FPGA, permitindo que o PLC aceda à memória da FPGA em modos de leitura e de escrita. Na Figura 20 podemos ver o diagrama de blocos da implementação do ModBus.

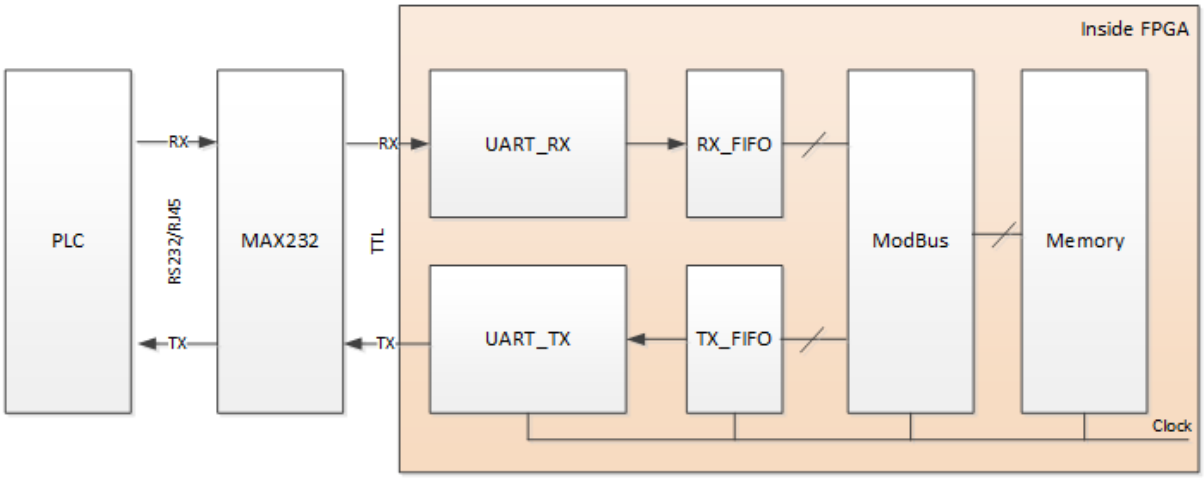


Figura 21 - Módulo ModBUS

Das funções existentes na definição do protocolo ModBus apenas foram implementadas duas, nomeadamente:

- *Read Holding Register*: função que permite ao ModBus ler da memória da FPGA com o código 0x03 (3);
- *Write Multiple Registers*: função que permite ao ModBus escrever na memória da FPGA com o código 0x10 (16).

A FPGA reage aos pedidos do PLC e nunca toma a iniciativa de comunicar com o PLC. O processo de comunicação é o seguinte:

1. O MAX232 (circuito integrado) converte os níveis de tensão do RS232 (-13V a 13V) em TTL (0V e 3.3V) e disponibiliza o sinal à entrada da FPGA (RX);
2. A FPGA deteta a variação do sinal, interpretando-os como bits de comunicação;
3. Quando o módulo "UART_RX" recebeu um byte completo (tendo em atenção aos "*start bits*", "*stop bits*" e paridades) transfere-o para um FIFO disponibilizado pelo módulo "RX_FIFO". A introdução de um FIFO de leitura (RX) melhora o controlo e baixa a complexidade de tratamento dos bytes que vão sendo disponibilizados pelo "UART_RX";
4. O módulo "Modbus" é informado pelo módulo "RX_FIFO" de que estão bytes disponíveis e inicia o seu processamento. O módulo "Modbus" irá consumir os bytes do FIFO até ter toda a informação disponível da função de ModBus requisitada (0x03 ou 0x10);
5. O módulo "Modbus" irá escrever na memória ou ler da memória, conforme a função evocada pelo PLC, verificando se a função é válida e se os endereços de memória também;
6. O módulo "Modbus" irá compor a resposta, dividindo-a em bytes e colocando-os no FIFO disponibilizado pelo "TX_FIXO";
7. O módulo "UART_TX" é alertado que tem bytes disponível no FIFO e irá processar cada byte para envio para o PLC;
8. O processo é reinicializado.

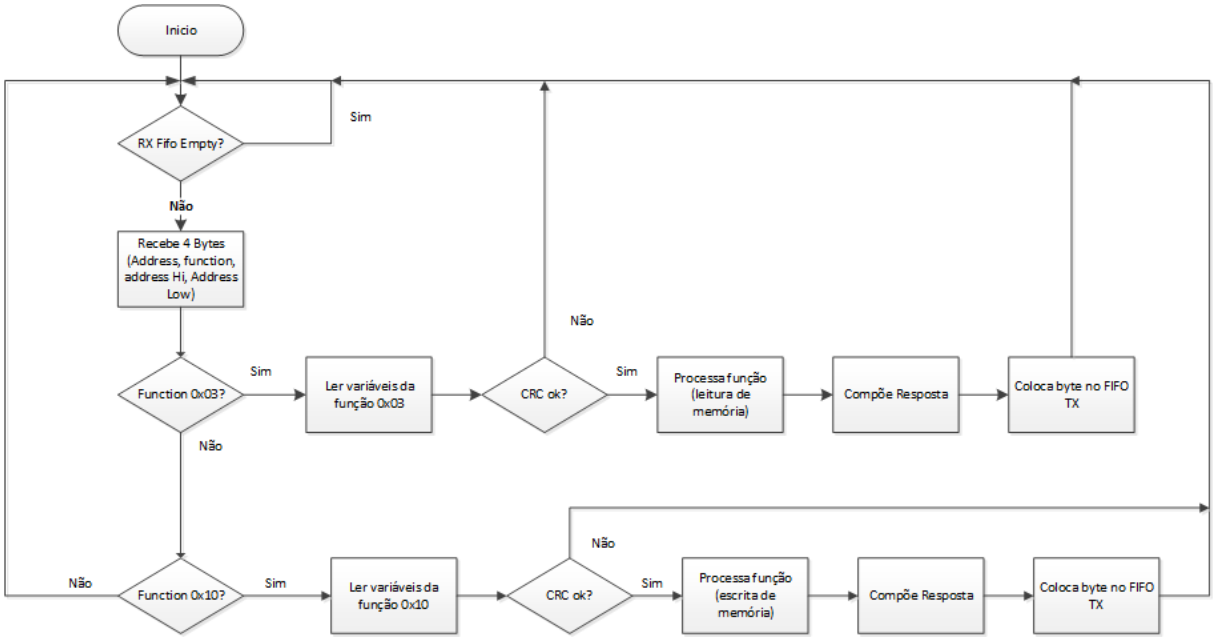


Figura 22 - Processo simplificado do módulo Modbus

Este processo é "infinito", pois enquanto a FPGA estiver operacional, estará sempre à espera de informação do PLC.

7.4 ADC

O ADC (*Analog Digital Converter*) utilizado – o ADC128S022 - é interno à FPGA usada, disponibilizando no GPIO da FPGA entradas utilizadas na leitura dos valores analógicos.

O módulo "ADC" é responsável por efetuar as leituras das portas analógicas.

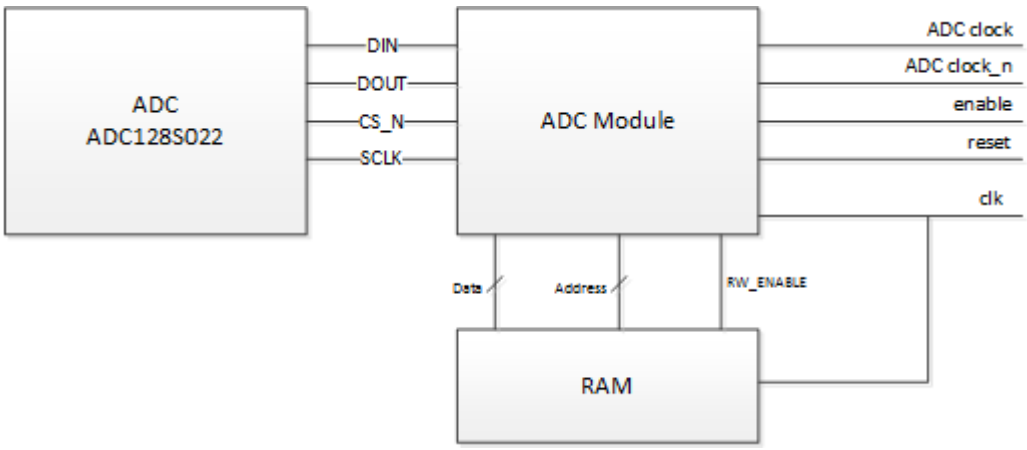


Figura 23 - Módulo ADC

Na Figura 23 podemos observar como o módulo ADC interage com as outras componentes.

Os valores são registados em zonas específicas de memória para que o PLC tenha acesso a esses mesmos valores.

7.4.1 ADC128S022 – Conversor Analógico-Digital [17]

7.4.1.1 Introdução

O conversor ADC128S022 é um semicondutor da "National Semiconductor" com as seguintes características principais:

- 8 Canais;
- Taxa de conversão de 5k SPS (*Samples per Second*) a 200k SPS
- 12 bits de resolução

Este circuito integrado está embebida na placa Deo-Nano e a FPGA obtém os valores de leitura através de protocolo SPI.

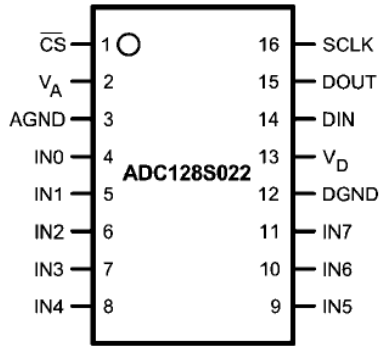


Figura 24 - Conversor analógico digital ADC128S022 [17]

Na Figura 24 podemos observar o diagrama de ligação do conversor ADC128S022.

7.4.1.2 Serial Interface - SPI

Foi necessário implementar um processo de leitura das diversas portas analógicas que respeitassem os requisitos do circuito, nomeadamente os seus diagramas temporais e especificações.

O protocolo SPI usa as seguintes sinais de entrada:

- CS_N: Seleção do circuito integrado em modo negado ou lógica inversa. O circuito integrado ADC128S022 [17] é ativado caso a entrada CS esteja em LOW;
- SCLK: Entrada de relógio digital. Segundo as especificações deste circuito integrado, a frequência ótima deste sinal de relógio é entre os 0.3MHz e os 3.2MHz. A velocidade de relógio usada tem implicação direta no controlo da conversão e das leituras.
- DIN: Entrada digital que identifica o endereço do canal a ser convertido;
- DOUT: Saída digital que conterá os valores de conversão.

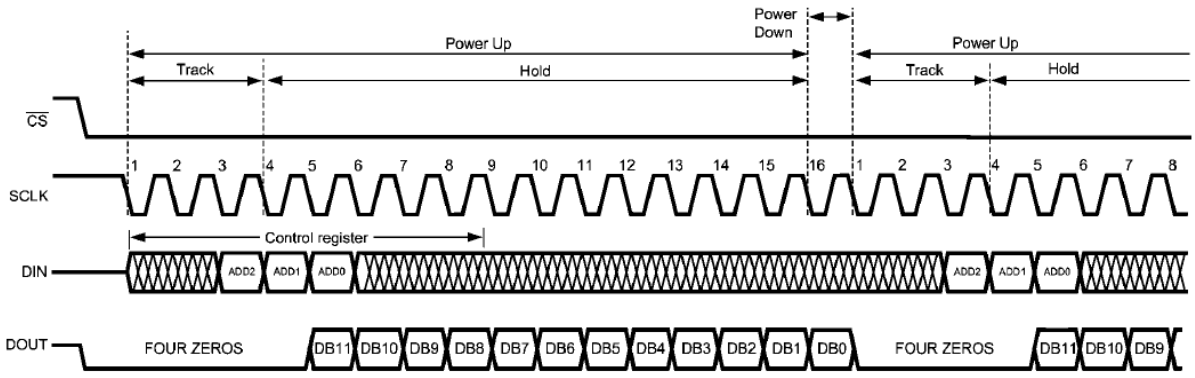


Figura 25 - Diagrama temporal operacional do conversor ADC128S022 [17]

Foi implementado um ciclo que executa leituras das portas analógicas, mediante o estado de disparo. Assim:

- Se T3 (T_a) ou T4 (T_b) estiverem em HI, são lidos os quatro primeiros canais do ADC (0 a 3);

- Se T3 e T4 estiverem em LOW, são lidos os últimos quatro canais do ADC (4 a 7)

Não há a necessidade de estar a registar os oito canais em simultâneo, visto que os valores a serem lidos, conforme o processo de disparo, terão valor espectável de 0. E desta forma, ao serem lidos 4 valores, acelera a obtenção desses mesmos valores.

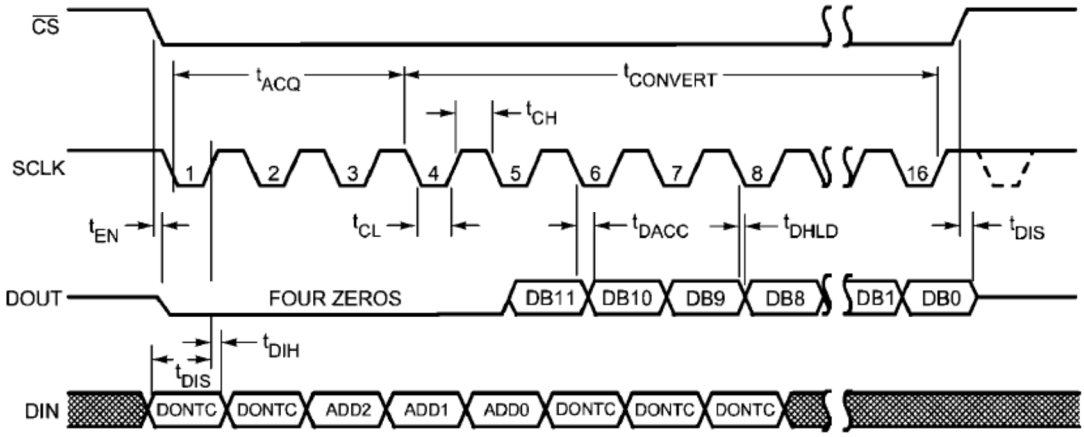


Figura 26 – Detalhe da leitura de um valor no conversor ADC128S022 [17]

Na Figura 26 podemos observar o detalhe do diagrama temporal do conversor, onde é explícito o modo de funcionamento do conversor:

- São passados 3 bits (DIN) para o endereço do canal a ler;
- São lidos 12 bits (DOUT) com o valor da conversão.

Foi necessário implementar uma lógica de leitura, porque o valor real da conversão apenas é disponibilizado no próximo ciclo de leitura.

7.5 PLL

O PLL é uma biblioteca de *software* da Altera (ALTPLL) que implementa um circuito de geração de sinais sincronizados com um sinal de *input*, com frequências diferentes.

Neste trabalho foi utilizado este módulo para produzir sinais de relógio com frequências mais reduzidas do que o relógio interno da FPGA. Foi também utilizado para gerar dois sinais de relógio com a mesma frequência mas desfasados em 180 graus.

7.5.1 ALTPLL [18]

Na implementação que foi executada neste trabalho foi necessário ter sinais de relógio a funcionarem em diferentes frequências mas era desejado que estivessem todos sincronizados ou em fase com o relógio principal da FPGA.

Assim, a FPGA tem um relógio interno de 50Mhz que é a entrada do módulo de PLL e são gerados 3 sinais de relógios:

- ModBus : sinal de relógio a 6 Mhz;
- ADC: sinal de relógio a 2 Mhz;
- ADC_N: sinal de relógio a 2Mhz desfasado em 180º do ADC;
- Fire: sinal de relógio de sincronismo dos disparos a 1Mhz;

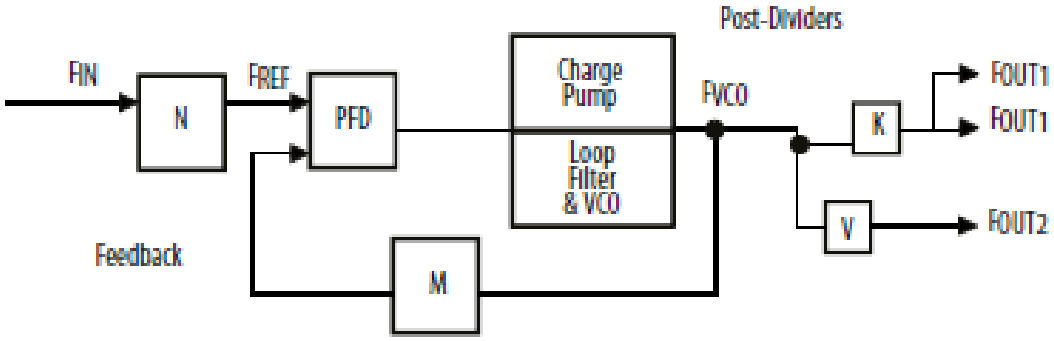


Figura 27 - Diagrama de blocos do PLL [18]

A Figura 27 podemos observar o diagrama de blocos do PLL, um módulo que é disponibilizado pela Altera e que é usado para criar sinais de relógio sincronizados.

O PLL consiste nos seguintes elementos:

- *Pre-divider counter (N counter)*;
- *Phase-frequency detector (PFD);*
- *Charge pump;*
- *Voltage-controlled oscillator* (VCO);
- *Feedback multiplier counter (M counter);*
- *Post-divider counters (K and V counters)*

O PFD deteta diferenças na fase e frequência entre o seu sinal de referência (f_{REF}) e o sinal de feedback, controla a *charge pump*, e controla o filtro do *loop* que converte a diferença de fase para uma tensão de controlo. Esta voltagem controla o VCO. Baseado na voltagem de controlo, o VCO oscila a uma frequência maior ou mais pequena, que afeta a fase e a frequência do sinal de feedback. Depois do sinal f_{REF} e o sinal de feedback terem a mesma fase e frequência, pode-se dizer que o PLL está em "*phase-locked*".

Inserindo o contador M no caminho do *feedback* causa que o VCO oscile a uma frequência que é M vezes a frequência do sinal de f_{REF}. O sinal f_{REF} é igual ao relógio de *input* (f_{IN}) dividido pelo contador de "*pre-scale*" (N).

A frequência de referência é descrita pela equação $f_{REF}= f_{IN}/N$. O frequência de saída do VCO é $f_{VCO}= (f_{IN} * M/N)$, e a frequência de saída do PLL é descrita pela equação $f_{OUT}= (f_{IN} * M)/(N*K)$ para os sinais.

7.6 Memória interna - RAM

Foi implementado um módulo de interno de memória RAM (*Random-Access Memory)*, utilizando uma função de gestão de memória da Altera com três portos (um de escrita e dois de leitura). Esta função da Altera tem o nome de "ALT3PRAM" e que será explica no capítulo seguinte.

Como apenas existe um porto de escrita na memória, foi necessário criar um componente que executasse um MUX (*multiplexer*) que selecionasse qual o módulo interno que iria escrever em memória: o módulo de registo de Erro, o módulo ADC ou o módulo ModBUS.

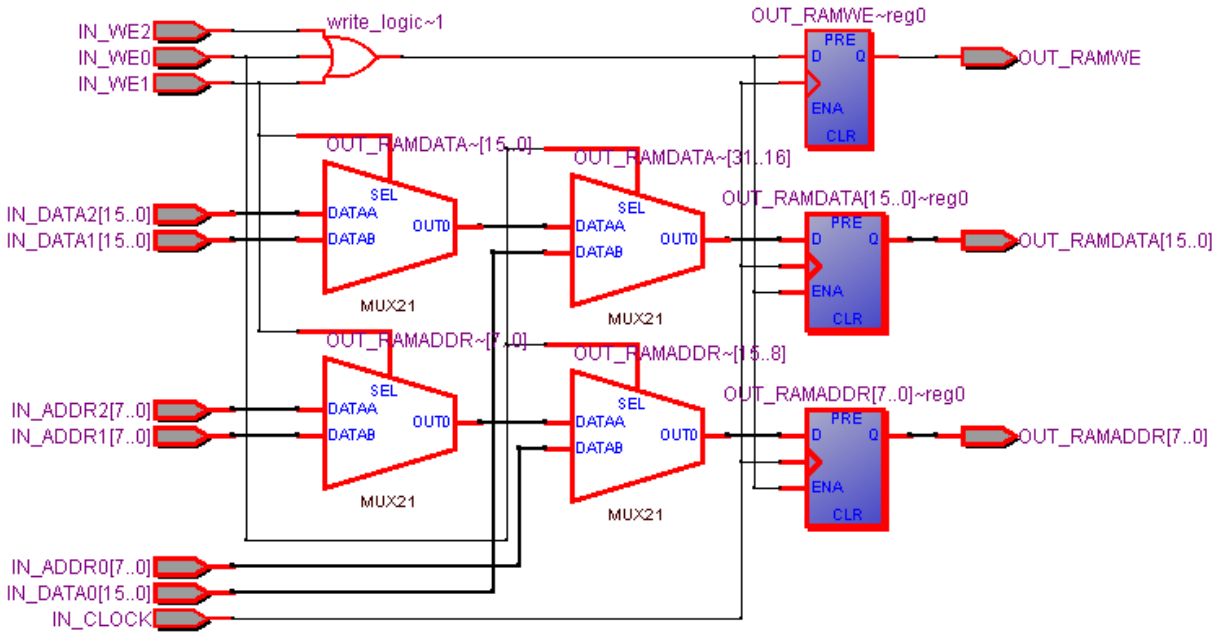


Figura 28 - Módulo MUX desenvolvido

Na Figura 28 podemos verificar o esquema desenvolvido em VHDL e convertido para um circuito pelo compilador da Altera.

Tabela 7 - Portos do MUX desenvolvido

Portos	Módulo	Prioridade
IN_ADDR0, IN_DATA0, IN_WE0	*Error Register*	*1*
IN_ADDR1, IN_DATA1, IN_WE1	*ADC*	*2*
IN_ADDR2, IN_DATA2, IN_WE2	*ModBUS*	*3*

A definição dos portos é a seguinte:

- IN_ADDR(N) : Porto de entrada com o endereço de memória a escrever;
- IN_DATA(N): Porto de entrada com os dados a escrever em memória;
- IN_WE(N) : Porto de entrada a indicar uma ação de escrita em memória;
- IN_CLOCK: Relógio de sincronismo;
- OUT_RAMWE: Porto de saída com indicação de uma ação de escrita em memória. Este porto está ligado ao módulo de RAM;
- OUT_RAMADDR: Porto de saída com o endereço da memória a escrever. Este porto está ligado ao módulo de RAM;
- OUT_RAMDATA: Porto de saída com os dados a escrever em memória. Este porto está ligado ao módulo de RAM;

7.6.1 ALT3PRAM

Como indicado no ponto anterior, o módulo "ALT3PRAM" [19] é um módulo disponibilizado pela Altera e que implementa a funcionalidade de memória RAM. Este módulo particular disponibiliza três portos – um porto de escrita, dois portos para leitura independentes.

Port Name	Required	Description	Comments
data[]	Yes	Data input to the memory.	Input port WIDTH wide.
rdaddress_a[]	Yes	Read address input to the memory.	Input port WIDTHAD wide.
rdaddress_b[]	Yes	Read address input to the memory.	Input port WIDTHAD wide.
wraddress[]	Yes	Write address input to the memory.	Input port WIDTHAD wide.
wren	Yes	Write enable input.	
inclock	No	Positive-edge-triggered clock.	Used for registered write ports, for example, data, wraddress[], and wren. Can also be used for registered read ports, for example, rdaddress_a[], rdaddress_b[], rden_a, and rden_b.
inclocken	No	Clock enable for inclock.	
rden_a	Yes	Read enable input. Disables reading when low (0).	
rden_b	Yes	Read enable input. Disables reading when low (0).	
outclock	No	Positive-edge-triggered clock.	Used for the registered q_a[] or q_b[] port. Can also be used for registered read ports, for example, rdaddress_a[], rdaddress_b[], rden_a, and rden_b.
outclocken	No	Clock enable for outclock.	
aclr	Yes	Asynchronous clear input.	Affects registered inputs and outputs.

Figura 29 - Portas de entrada do módulo de RAM (*datasheet*) [19]

Na Figura 29 podemos observar os diversos portos de entrada do módulo ALT3PRAM.

Port Name	Required	Description	Comments
qa[]	Yes	Data output from the memory.	Output port WIDTH wide.
qb[]	Yes	Data output from the memory.	Output port WIDTH wide.

Figura 30 - Portas de saída do módulo de RAM (*datasheet*) [19]

Na Figura 30 podemos observar os diversos portos de saída do módulo ALT3PRAM.

7.7 Signal Generator

7.7.1 Módulo Fireup

Este módulo, o *Signal Generator*, é o responsável por gerar os sinais de disparo para o gerador de Marx. Ao longo deste capítulo será explicado o modo de operação deste módulo, as variáveis de configuração e o seu ciclo de disparo.

Este módulo, sendo o coração do sistema, está inserido dentro de um outro módulo – Fireup – que prepara o ambiente e definições do *Signal Generator*.

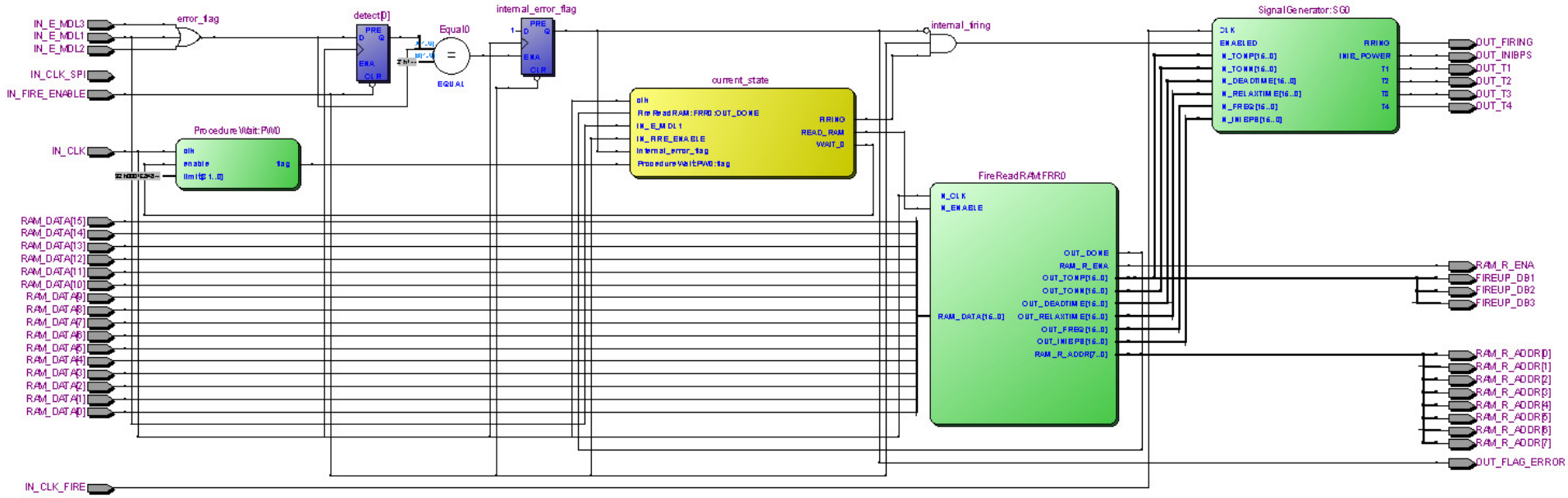


Figura 31 - Netlist do módulo Fireup

Na Figura 31 poderemos observar os módulos que compõem o módulo *Fireup*:

- Wait: Módulo de aguarda a contagem de ciclos de relógio e aquando atinge a contagem alvo ativa uma porta. Este módulo é necessário para criar pausas;
- FireReadRAM: Módulo que é responsável por ler da memória as variáveis necessárias ao *SignalGenerator*. Os valores lidos são guardados em registos internos;
- SignalGenerator: Módulo principal de geração dos sinais de disparo do gerador de Marx.

É também possível observar que foi implementado uma máquina de estados que controla a fase do processo de disparos: *current_state*:

7.7.1.1 Portos de entrada e saída - Fireup

Na tabela seguinte serão identificadas os portos de entrada e saída do módulo *Fireup*.

Tabela 8 - Entradas e saídas do Módulo Fireup

Porto	Direção	Tipo VHDL	Descrição
IN_CLK	*IN*	std_logic	Sinal de relógio interno da FPGA (50 MHz)
IN_CLK_FIRE	*IN*	std_logic	Sinal de relógio de disparo (1 MHz)
IN_FIRE_ENABLE	*IN*	std_logic	Sinal de indicação que existe indicação para que estejam a ser gerados sinais de disparo. Este sinal mantém-se *enable* (1) durante toda a duração dos disparos.
IN_E_MDL1	*IN*	std_logic	Indicação de erro no módulo de disparo 1.
IN_E_MDL2	*IN*	std_logic	Indicação de erro no módulo de disparo 2.
IN_E_MDL3	*IN*	std_logic	Indicação de erro no módulo de disparo 3.
RAM_R_ADDR	*OUT*	std_logic_vector	Endereço de memória a aceder (ler).
RAM_R_ENA	*OUT*	std_logic	Indicação de leitura de memória.
RAM_DATA	*IN*	std_logic_vector	Dados lidos da memória. Esta entrada é atualizada quando o sinal RAM_R_ENA é colocado a 1.
OUT_T1	*OUT*	std_logic	Sinal de disparo T1.
OUT_T2	*OUT*	std_logic	Sinal de disparo T2.

Porto	Direção	Tipo VHDL	Descrição
OUT_T3	*OUT*	std_logic	Sinal de disparo T3.
OUT_T4	*OUT*	std_logic	Sinal de disparo T4.
OUT_INIBPS	*OUT*	std_logic	Indicação para a fonte do Sistema que não deverá fornecer energia. A fonte apenas deverá fornecida durante o estado de carga.
OUT_FIRING	*OUT*	std_logic	Indicação que o módulo está em fase de disparo.
OUT_FLAG_ERROR	*OUT*	std_logic	Indicação de erro.

Alguns portos não foram colocados na tabela anterior, pois apenas são usados para situações de *debug*.

7.7.1.2 Ciclo de Estados - Fireup

O módulo Fireup opera segundo um ciclo ou processo criado para garantir o início dos disparos.

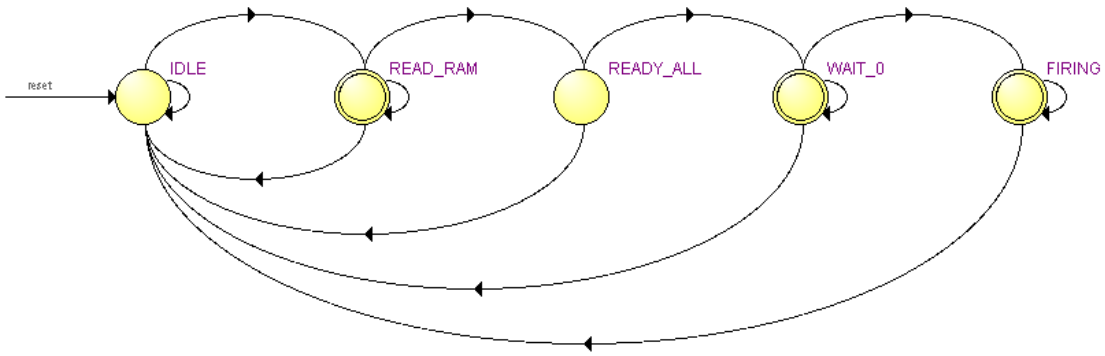


Figura 32 - Estados do módulo Fireup

Os estados são os seguintes, sendo a sequência dos estados ilustrada na Figura 32:

Tabela 9 - Estados do módulo Fireup

Estado	Descrição
IDLE	Estado inicial e onde o processo se mantém enquanto não tem indicação que deverá ser iniciado o processo de disparos (entrada IN_FIRE_ENABLE).
READ_RAM	Depois de indicação que os disparos devem começar a ser executados, o próximo passo é ler os valores de configuração guardados em memória. Este estado ativa o módulo FireReadRAM.
READY_ALL	Estado em que o processo aguarda que o módulo FireReadRAM indique que a memória foi lida.
WAIT_0	Estado que aguarda alguns ciclos de relógio para garantir que o processo está sincronizado.
FIRING	Estado que ativa o módulo SignalGenerator. O processo mantém-se neste estado enquanto existir indicação que os disparos devem de ser efetuados.

7.7.2 Variáveis de configuração

No módulo FireReadRAM são lidas as seguintes variáveis da memória RAM:

Tabela 10 - Variáveis lidas pelo módulo FireReadRAM

Variável	Descrição
OUT_TONP	Indicação em milissegundos (ms) do tempo de impulso do ciclo positivo do gerador de Marx.
OUT_TONN	Indicação em milissegundos (ms) do tempo de impulso do ciclo negativo do gerador de Marx.
OUT_DEADTIME	Indicação em milissegundos (ms) do tempo morto entre os sinais de disparo. Este tempo evita que haja curtos circuitos nas comutações dos IGBT's. (não existem assim duas comutações em simultâneo)
OUT_RELAXTIME	Indicação em milissegundos (ms) do tempo existente entre o impulso positivo e o negativo do gerador de Marx.
OUT_FREQ	Indicação da frequência de disparo
OUT_INIBPS	Indicação em milissegundos (ms) do tempo dado à fonte do sistema para carregar os condensadores do gerador de Marx.

Estas variáveis são lidas uma a uma, sendo disponibilizadas depois nos registos internos do módulo para que sejam acessíveis ao módulo *SignalGenerator*.

7.7.3 Arranque dos disparos

Já tivemos oportunidade de observar o processo de preparação das configurações do *SignalGenerator*, pelo que neste ponto vamos observar como é efetuado o processo de disparo.

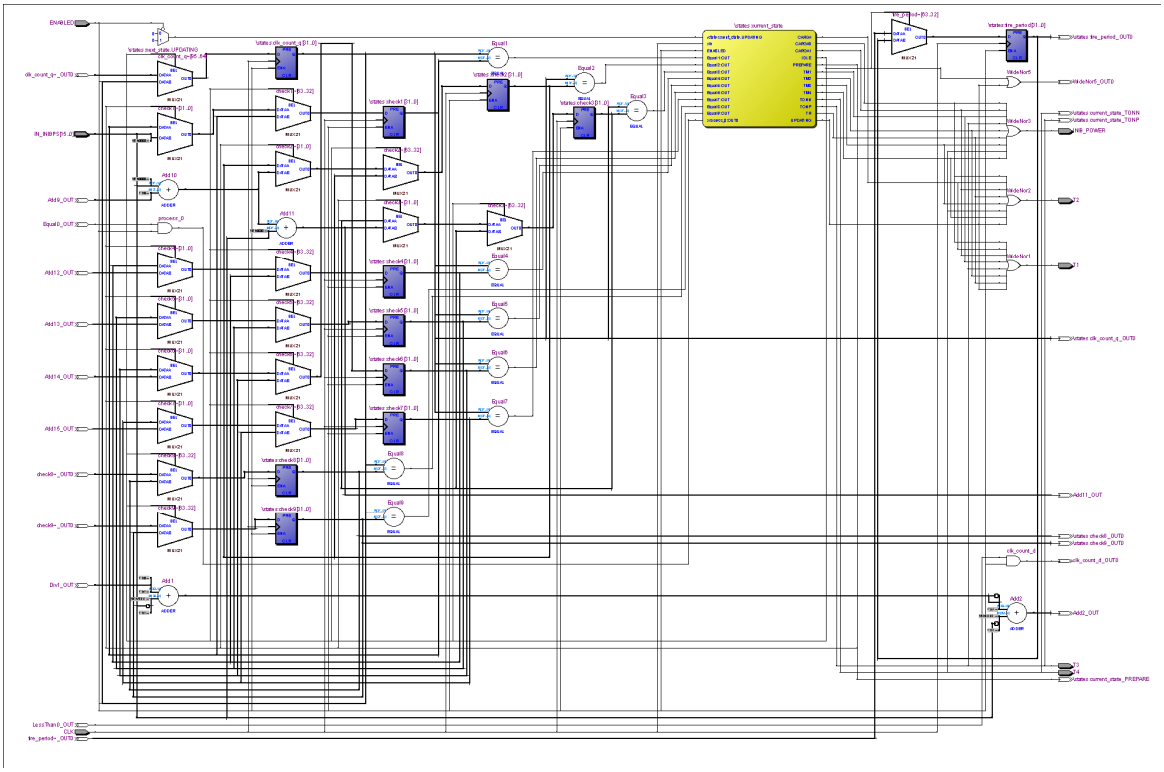


Figura 33 - Aspeto do circuito RTL do SignalGenerator

Na Figura 33 podemos ver o módulo RTL (*Register-Transfer Level*) gerado pelo compilador da Altera, onde se destaca a amarelo os estados do processo implementado no interior do módulo *SignalGenerator*.

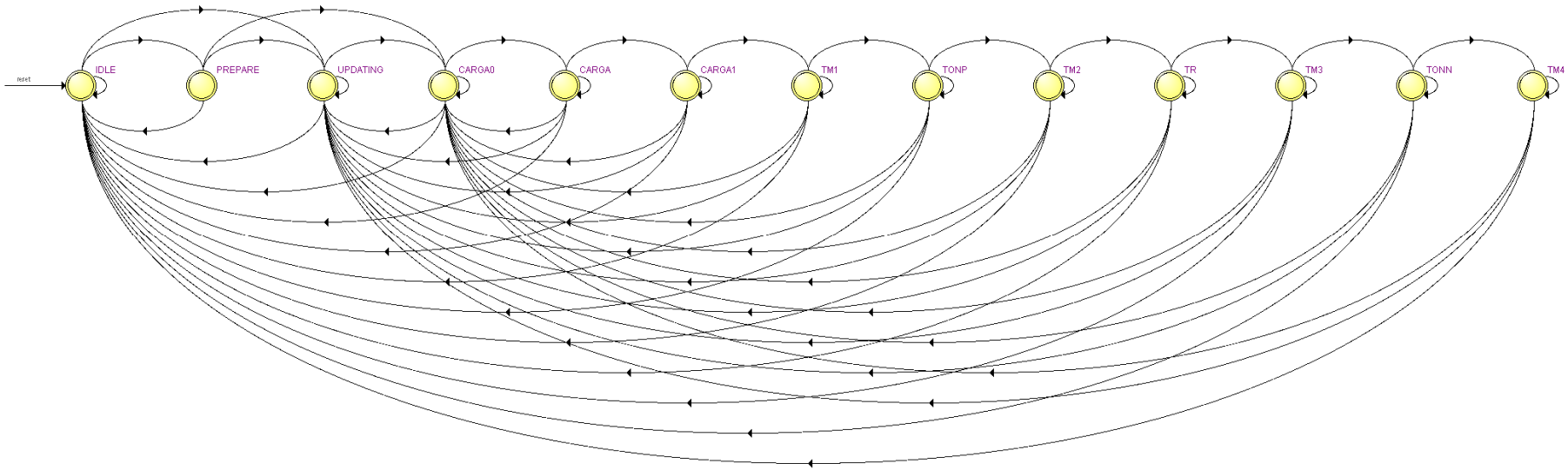


Figura 34 - Estados do processo de disparo

Os estados são os seguintes, sendo a sequência dos estados ilustrada na Figura 34:

Tabela 11 - Estados de disparo do módulo SignalGenerator

Estado	Descrição	Sinais de disparo e controlo
IDLE	Estado inicial e onde o processo se mantém enquanto não tem indicação que deverá ser iniciado o processo de disparos.	T1 = T_f = 0 T2 = T_e = 0 T3 = T_a = 0 T4 = T_b = 0 INIBPS = 0 FIRING = 0
PREPARE	Estado onde são preparados os contadores internos. No próximo ponto será melhor detalhado este estado.	T1 = T_f = 0 T2 = T_e = 0 T3 = T_a = 0 T4 = T_b = 0 INIBPS = 0 FIRING = 0
UPDATING	Estado previsto mas não utilizado.	
CARGA0	Estado que, fazendo parte do tempo de carga, inibe a fonte do sistema de fornecer energia.	T1 = T_f = 1 T2 = T_e = 1 T3 = T_a = 0 T4 = T_b = 0 INIBPS = 1 FIRING = 0
CARGA	Estado que permite que a fonte do sistema forneça energia ao sistema.	T1 = T_f = 1 T2 = T_e = 1 T3 = T_a = 0 T4 = T_b = 0 INIBPS = 0 FIRING = 0
CARGA1	Estado que, fazendo parte do tempo de carga, inibe a fonte do sistema de fornecer energia.	T1 = T_f = 1 T2 = T_e = 1 T3 = T_a = 0 T4 = T_b = 0 INIBPS = 1 FIRING = 0
TM1	Estado de tempo morto.	T1 = T_f = 1 T2 = T_e = 0 T3 = T_a = 0 T4 = T_b = 0 INIBPS = 1 FIRING = 0

Estado	Descrição	Sinais de disparo e controlo
TONP	Estado que indica a geração dos sinais de controlo que permitam ao gerador de Marx efetuar a descarga positiva.	$T1 = T_f = 1$ $T2 = T_e = 0$ $T3 = T_a = 1$ $T4 = T_b = 0$ INIBPS = 1 FIRING = 1
TM2	Estado de tempo morto.	$T1 = T_f = 1$ $T2 = T_e = 0$ $T3 = T_a = 0$ $T4 = T_b = 0$ INIBPS = 1 FIRING = 0
TR	Tempo de repouso entre os ciclos positivos e negativos do gerador de Marx.	$T1 = T_f = 1$ $T2 = T_e = 1$ $T3 = T_a = 0$ $T4 = T_b = 0$ INIBPS = 1 FIRING = 0
TM3	Estado de tempo morto.	$T1 = T_f = 0$ $T2 = T_e = 1$ $T3 = T_a = 0$ $T4 = T_b = 0$ INIBPS = 1 FIRING = 0
TONN	Estado que indica a geração dos sinais de controlo que permitam ao gerador de Marx efetuar a descarga negativa.	$T1 = T_f = 0$ $T2 = T_e = 1$ $T3 = T_a = 0$ $T4 = T_b = 1$ INIBPS = 1 FIRING = 1
TM4	Estado de tempo morto.	$T1 = T_f = 0$ $T2 = T_e = 1$ $T3 = T_a = 0$ $T4 = T_b = 0$ INIBPS = 1 FIRING = 0

Podemos ver graficamente a evolução dos sinais nas imagens seguintes.

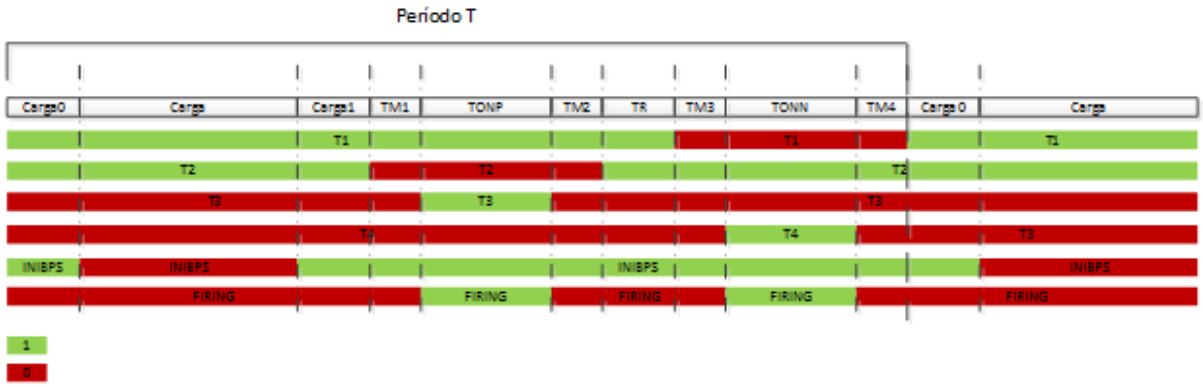


Figura 35 - Evolução dos sinais de disparo de controlo

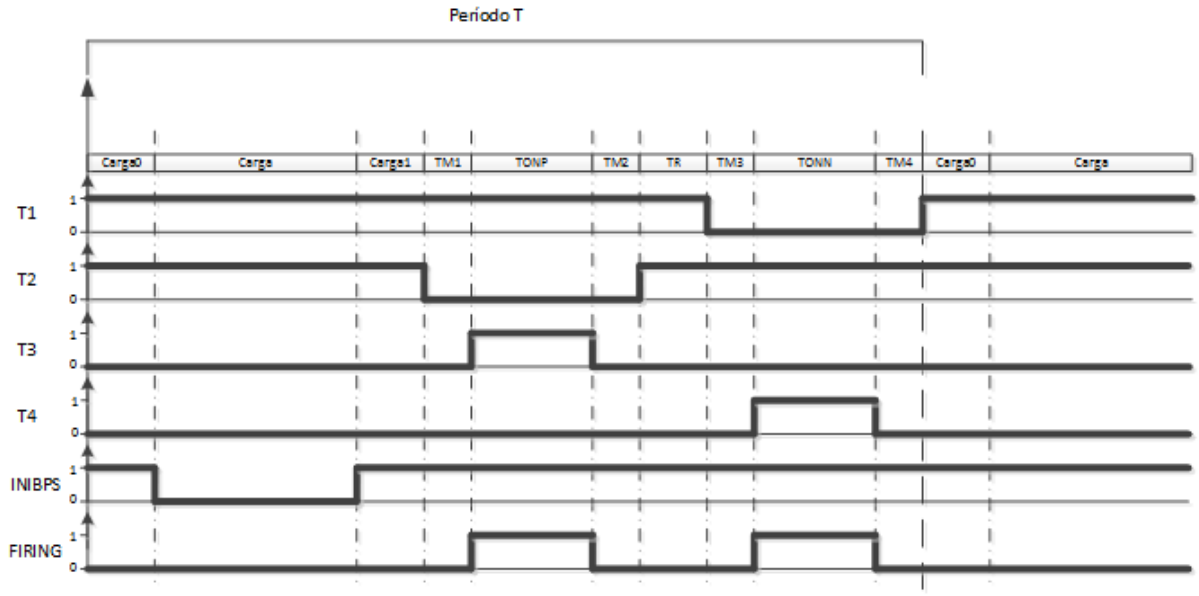


Figura 36 - Diagrama temporal dos sinais gerados

As imagens anteriores mostram as variações dos sinais de disparo T1, T2, T3 e T4, e os sinais de controlo INIBPS e FIRING.

É também observável que não há duas variações simultâneas de qualquer sinal T1, T2, T3 ou T4. Tal como foi referenciado são utilizados os tempos mortos (TM1, TM2, TM3 e TM4) para efetuar as variações.

A correspondência dos tempos (configurações) aos estados é o seguinte:

Tabela 12 - Correspondência dos estados às configurações

Estado	Variável
CARGA0	OUT_INIBPS
CARGA	--
CARGA1	OUT_INIBPS
TM1	OUT_DEADTIME
TONP	OUT_TONP
TM2	OUT_DEADTIME
TR	OUT_RELAXTIME
TM3	OUT_DEADTIME
TONN	OUT_TONN
TM4	OUT_DEADTIME

Na tabela anterior observa-se que o estado CARGA não tem um tempo (ou variável) associado, isto porque o tempo de CARGA é calculado para que a frequência desejada (OUT_FREQ) possa ser atingida.

$$T\ (s) = 1 / OUT\_FREQ\ (f)$$

$$CARGA = T - (CARGA0 + CARGA1 + TM1 + TONP + TM2 + TR + TM3 + TONN + TM4)$$

$$CARGA = T - (2 * OUT\_INIBPS + 4* OUT\_DEADTIME + OUT\_TONP + OUT\_RELAXTIME + OUT\_TONN)$$

Como exemplo, se pretendemos uma frequência de disparo de 50Hz e os seguintes valores:

- OUT_INIBPS = 250 * 10^{-6} (s) = 250 (us)
- OUT_DEADTIME = 5 * 10^{-6} (s) = 5 (us)
- OUT_RELAXTIME = 20 * 10^{-6} (s) = 20 (us)
- OUT_TONP = 20 * 10^{-6} (s) = 20 (us)
- OUT_TONN = 20 * 10^{-6} (s) = 20 (us)
- OUT_FREQ = 50 Hz

Teremos:

$$T = 1 / (1/50) = 0{,}02\ (s) = 20000\ (us)$$

$$CARGA = T - (2 * OUT\_INIBPS + 4* OUT\_DEADTIME + OUT\_TONP + OUT\_RELAXTIME + OUT\_TONN)$$

$$CARGA = 20000 - (2 * 250 + 4* 5 + 20 + 20 + 20)$$

$$CARGA = 20000 - 580$$

$$CARGA = 19420\ (us)$$

7.7.4 Preparação dos disparos

O *SignalGenerator* recebe um sinal de relógio com uma frequência fixa de 1Mhz = ($1*10^{-6}$ (s) = 1 (us)). Todas as configurações de disparo são múltiplos deste tempo mínimo de resolução.

Durante a fase (estado) de PREPARE são calculados os *checkpoints* dos vários estados de disparo. Também, e como vimos no ponto anterior, é calculado o tempo do estado de CARGA.

Em termos de processo de disparo, existe um contador que é inicializado a 0, por cada impulso de relógio (1Mhz) é incrementado com o valor 1. Conforme vai atingindo os *checkpoints*, vai forçando o processo a alterar de estado. Quando atinge o ultimo *checkpoint*, que reflete o estado TM4, o contador passa para 0 e o ciclo é reiniciado.

7.8 Interface PLC

A interface do PLC não foi desenvolvida no âmbito desta tese, no entanto será feita uma breve descrição da mesma.

O PLC tem como funcionalidade o controlo do disparo, ativando uma porta digital para dar início aos disparos. Recebe também via porta analógica informação de Erro. As restantes variáveis que são escritas ou lidas chegam ao PLC via protocolo ModBus implementado para o efeito. Em termos de ModBus o PLC atua como *master*.

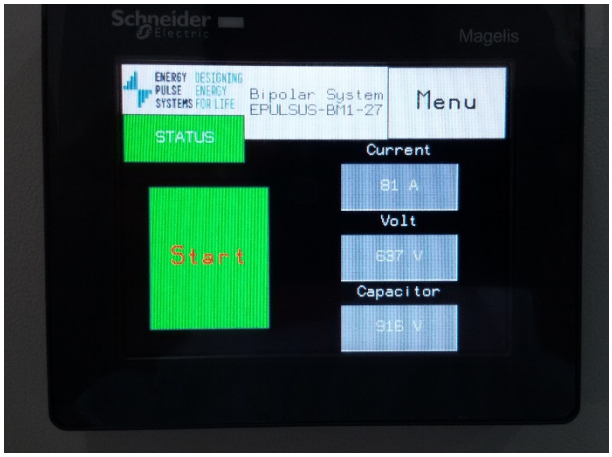


Figura 37 - Interface gráfica PLC - Menu Inicial

Na Figura 37 podemos observar a interface gráfica implementada para o sistema. O PLC obtém periodicamente os valos de tensões e correntes da placa FPGA e mostra-os na interface gráfica.

É aqui também que se dá a ordem de arranque dos disparos, premindo o botão “Start”.

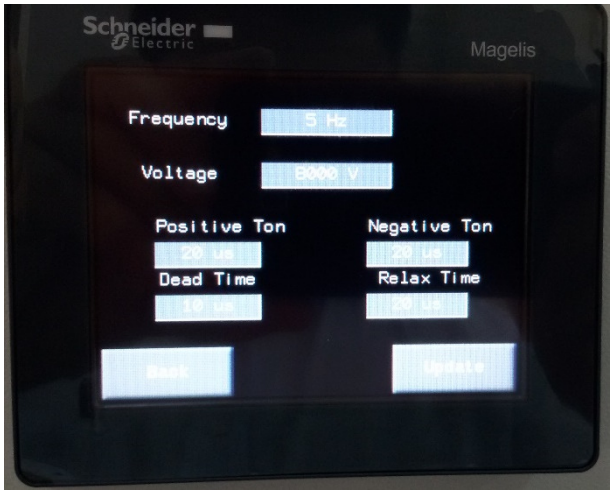


Figura 38 - Parâmetros do sistema

Na Figura 38 podemos observar um ecrã com a informação dos parâmetros do sistema. São estes valores que o PLC passa para a FPGA e que definirão a sequência de disparos, ou seja, o sinal de saída do gerador de Marx.

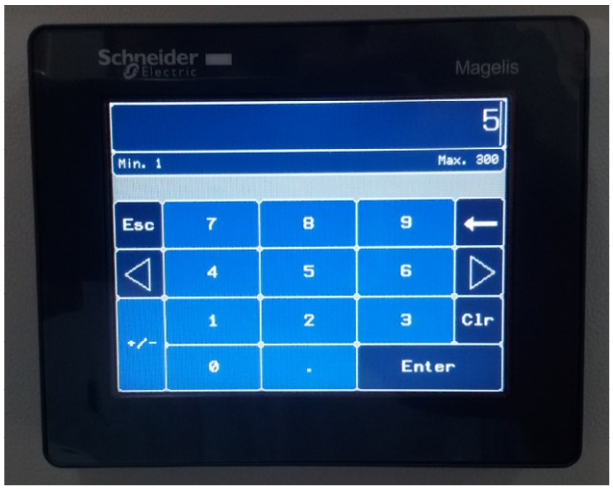


Figura 39 - Introdução dos valores dos parâmetros

Na Figura 39 podemos verificar o modo como podem ser inseridos os valores das configurações.

Figura 40 - Informação de Erro

Na Figura 40 podemos verificar a informação de erro e contém duas informações importantes: Qual ou quais o módulos em estado de erro e qual o primeiro módulo que deu erro (que pode ser importante em despistes de problemas)

Figura 41 - Painel na porta do *rack*

Na Figura 41 podemos ver o aspeto do painel completo, tendo outras botões e informação luminosa da operação do sistema e que são controladas pelo PLC.

IV. Resultados

8 Processo de desenvolvimento

O código gerado em VHDL para este trabalho foi estruturado de uma forma modular, em que cada módulo tem a sua função específica na operação da placa desenvolvida. Este conceito modular foi o que mais garantidas dava de evolução das funcionalidades propostas, pois será possível no futuro substituir um módulo por outro com outras funcionalidades mas que interage da mesma forma com os módulos circundantes.

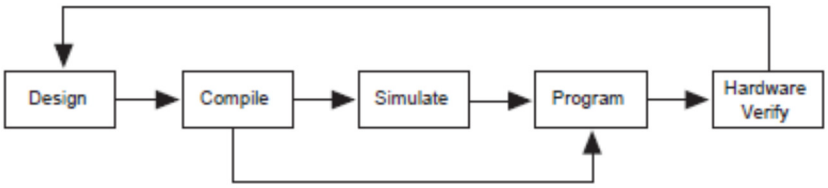


Figura 42 - Processo de desenvolvimento

Na Figura 42 podemos observar o processo que foi seguido, nomeadamente:

- *Design*: fase de implementação do módulo ou da solução de forma a verificar os diversos componentes necessários;
- *Compile*: fase de codificação dos elementos necessários;
- *Simulate*: simulação da codificação efetuada e verificação dos resultados;
- *Program*: programação da FPGA;
- *Hardware Verify*: verificação do código efetuado e aplicado à realidade. Foram por diversas vezes encontrados problemas que nas simulações não era possível encontrar, isto devido ao meio físico como por exemplo o ruido eletromagnético produzido pelo gerador de Marx.

9 Simulações

9.1 Introdução

As simulações são umas das capacidades fundamentais que a Altera oferece no seu ambiente de desenvolvimento. Permite a quem desenvolve para a FPGA simular o comportamento do seu circuito (ou programação) sem que tenha de usar uma FPGA fisicamente. Logo neste passo podemos ver se a lógica corresponde ao pretendido e alterar a codificação se necessário.

É necessário também ter algum cuidado, pois em simulação é-nos permitido inserir pausas nos processos, o que não é possível na placa física.

Durante o trabalho foram desenvolvidos mini programas de VHDL que serviram para testar cada um dos módulos mais importantes individualmente e verificar o seu correto funcionamento antes de programar a FPGA.

9.2 Sinais de relógio internos

A FPGA contém um sinal de relógio interno de 50Mhz, mas alguns módulos necessitam de sinais de relógio mais reduzidos e a frequências especificas. Para criar sinais baseados neste de 50Mhz foi utilizado o módulo de PLL já explicado anteriormente, e o efeito foi o seguinte:

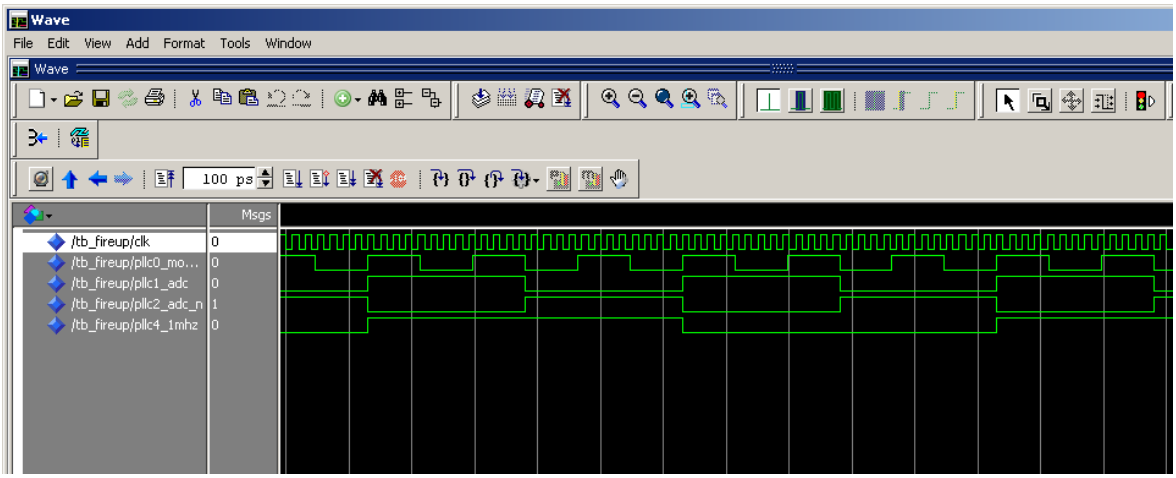


Figura 43 - Sinais de relógio internos

Podemos observar na Figura 43 o sinal de relógio da FPGA (50Mhz – clk) e os sinais de relógios gerados:

- pllc0_modbus : frequência de 6 Mhz e o sinal é utilizado no módulo de Modbus;
- pllc1_adc: frequência de 2 Mhz e o sinal é utilizado no módulo de ADC;
- pllc2_adc_n: frequência de 2 Mhz, em fase com o sinal de relógio pllc1_adc, e o sinal é utilizado no módulo de ADC;
- pllc4_1Mhz: frequência de 1 Mhz e o sinal é utilizado no módulo de "*Signal Generator*";

9.3 Arranque dos disparos

Na simulação é possível ver ao detalhe o arranque do processo de disparo e o estado do processo.

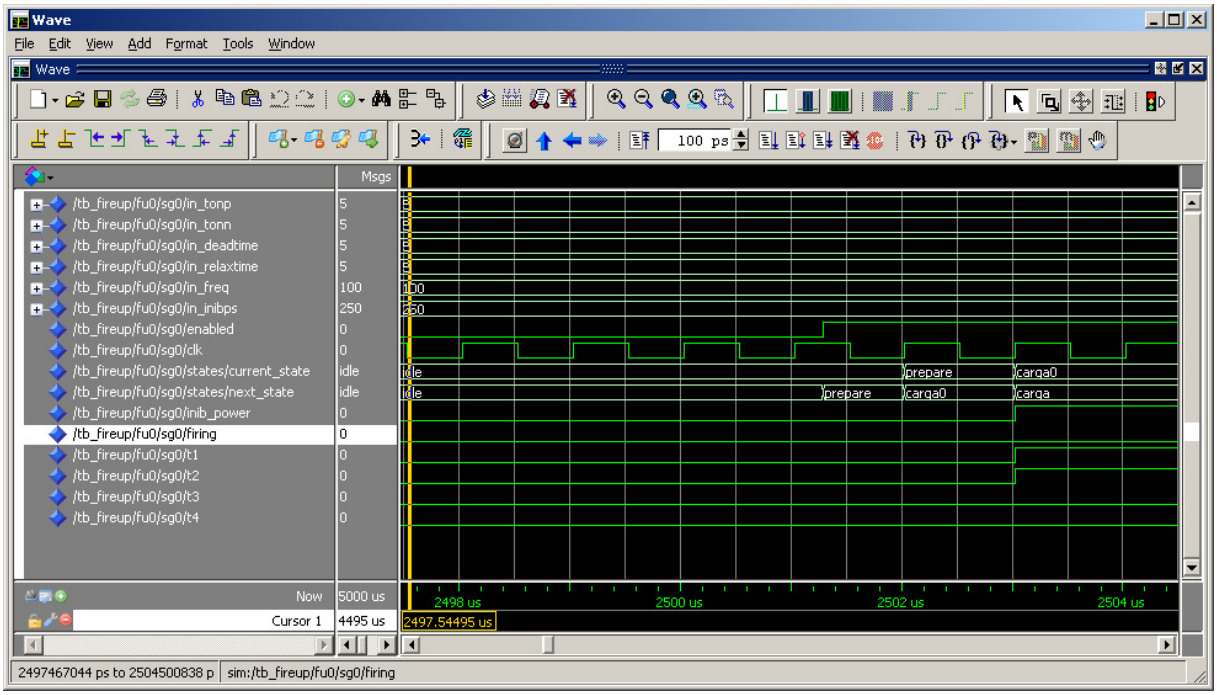


Figura 44 – Simulação do início do processo de disparo

Na Figura 44 podemos ver no simulador os valores que foram configurados para a operação (in_tonp=5us, in_tonn=5us, in_deadtime=5us, in_relaxtime=5us, in_freq=100Hz, in_inibps=250us) e o sinal de arranque do disparo (enabled). Depois do sinal de "*enabled*" passar de 0 para 1, podemos ver que o estado do processo (current_state) passa de "*idle*", para "*prepare*" e depois para "carga0". O estado de "*prepare*" é um estado transitório que apenas acontece no início do processo e serve para calcular variáveis internas de controlo do processo.

9.4 Inibição da fonte

Durante a fase de disparo não é necessário que a fonte do sistema esteja a fornecer energia. Assim, foi implementado no ciclo de disparo a inibição da fonte de alimentação.

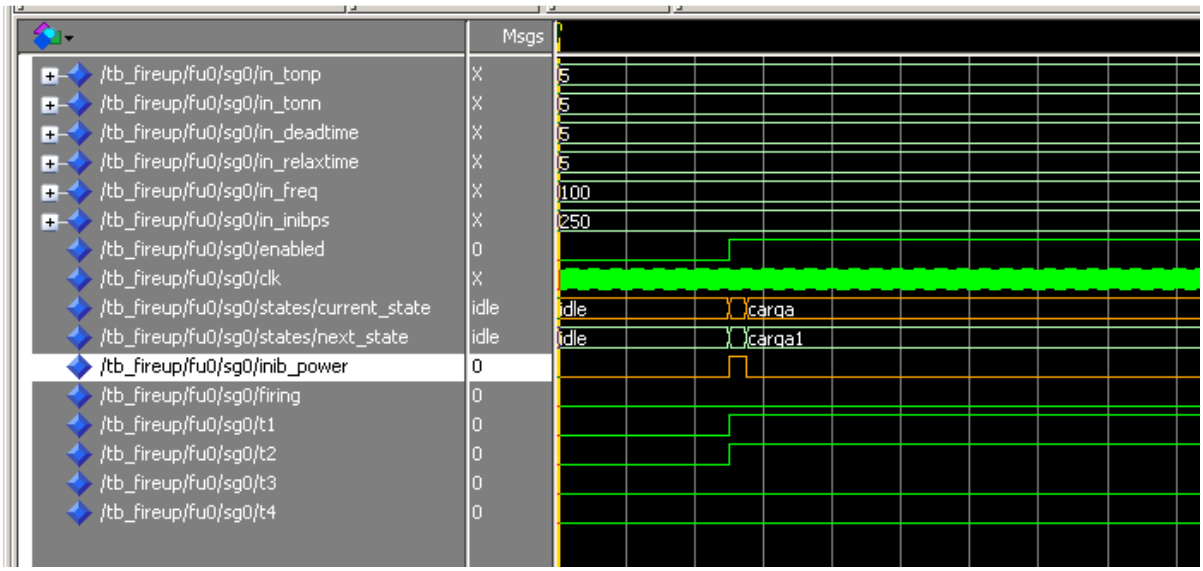


Figura 45 - Inibição da fonte

Na Figura 45 podemos observar o sinal de inibição da fonte (inib_power) durante a fase do processo “carga”.

9.5 Sinais de comando do Gerador

É o resultado principal do trabalho, a geração dos sinais de comando. Durante as simulações foi possível observar que o resultado era o esperado. Nas próximas imagens serão apresentados os sinais e as variáveis usadas para os controlar.

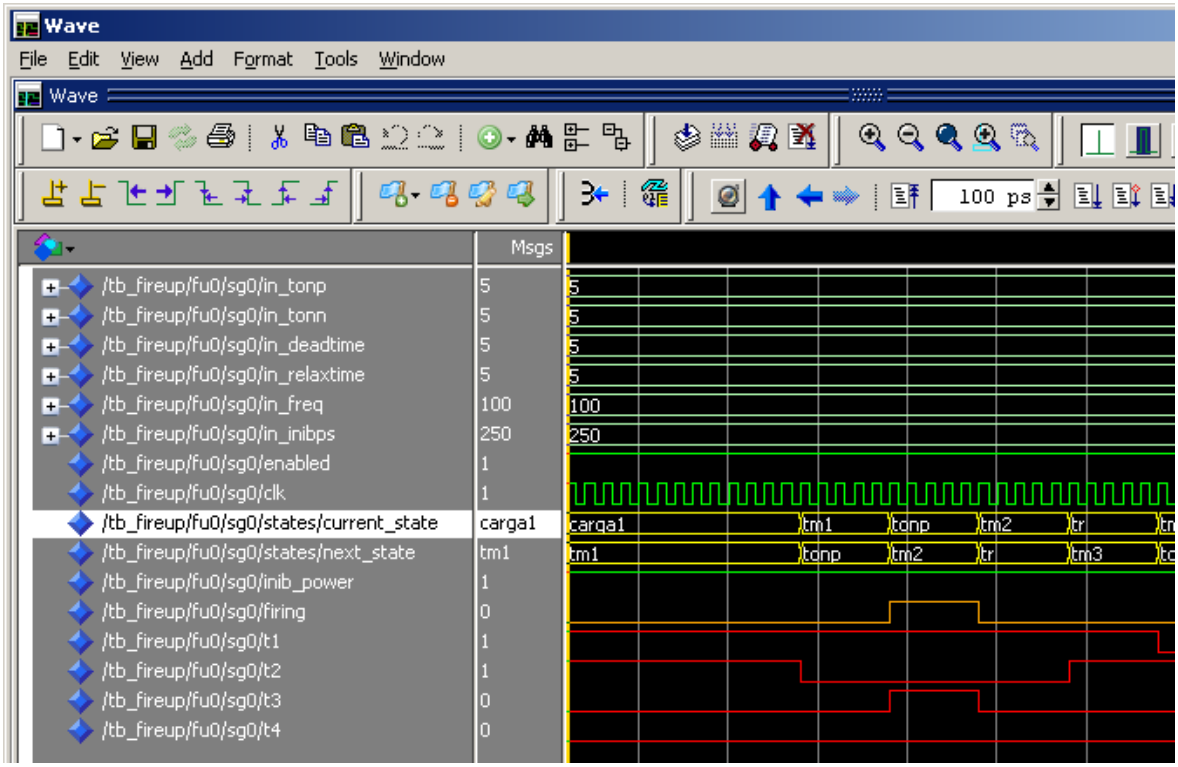


Figura 46 - Simulação dos sinais de disparo

Podemos observar na Figura 46 os 4 sinais de comando do gerador de Marx (t1, t2, t3 e t4) que variam os seus valores conforme o estado do processo (tm1, tonp, tm2, tr, tm3, tonn, tm4, carga0 e carga1).

Há adicionalmente um sinal que é gerado (*firing*) quando está a acontecer o disparo (no ciclo positivo ou negativo) e serve para que o ADC faça uma amostragem da tensão e corrente na carga, pois só nesta altura existe tensão e corrente na carga.

10 Resultados experimentais

10.1 Introdução

As simulações são muito importantes mas é na realidade que se tem que observar os resultados. Assim, o código desenvolvido foi aplicado a uma FPGA e esta foi ligada a um gerador de Marx bipolar.

Foram observados comportamentos que dificilmente seriam detetados nas simulações como a existência de ruído eletromagnético (que provoca alterações aos sinais gerados) ou um condensador mal dimensionado que fazia com que o GPIO "flutuasse" no valor de aquisição da FPGA introduzindo um comportamento aleatório do processo. Estes comportamentos foram observados em conjunto com a equipa da empresa EnergyPulse Systems Lda., mas a resolução dos mesmos está fora do âmbito deste trabalho.

Ao longo deste capítulo serão mostradas captações dos sinais de comando e também da tensão na carga.

10.2 Sinais de comando do gerador de Marx

Os sinais gerados correspondem, como esperado, às simulações efetuadas durante o desenvolvimento. O sistema físico pode estar em execução durante várias horas sem que haja qualquer dessincronização dos sinais de disparo.

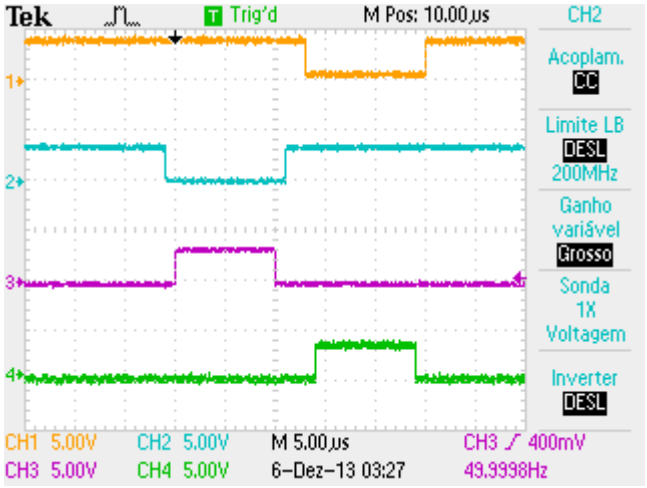


Figura 47 - Detalhe dos sinais de comando

Na Figura 47 podemos observar os quatro sinais de comando:

- Laranja: T1;
- Azul: T2;
- Roxo: T3;
- Verde: T4;

Podemos observar que a frequência dos disparos é de aproximadamente 50Hz, conforme a configuração do sistema.

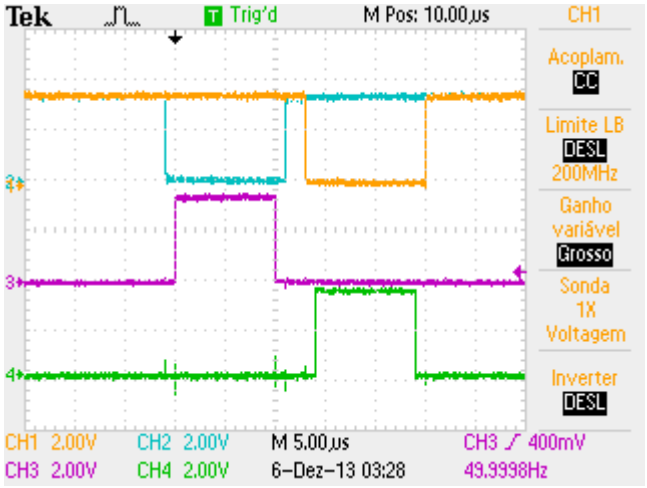


Figura 48 - Detalhe adicional dos sinais de comando

Na Figura 48 temos uma perspetiva diferente dos sinais, com foque no T3 e T4 que darão origem aos impulsos positivos e negativos, respetivamente, do gerador de Marx.

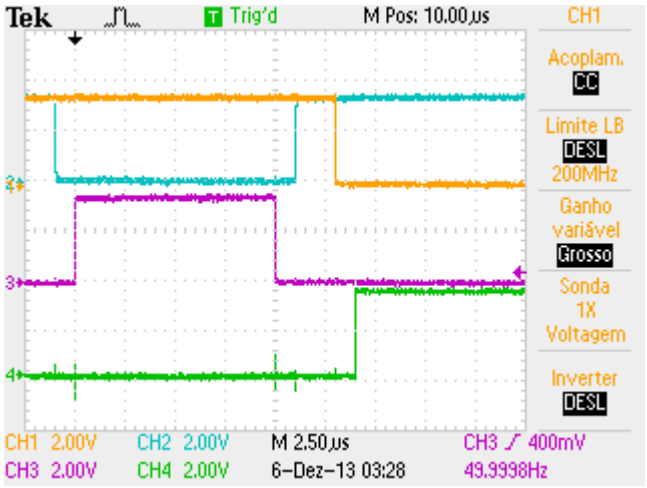


Figura 49 - Detalhe dos tempos mortos e de relaxe

Na Figura 49 podemos observar um mais detalhe os tempos mortos e o tempo de relaxe (tempo entre o impulso positivo e o impulso negativo do gerador de Marx).

Tempos mortos: O sistema não permite que haja variação simultânea de dois ou mais sinais, assim podemos observar entre cada variação de sinais um pequeno tempo de espera (tm1, tm2, tm3 e tm4 do processo de disparo). Na Figura 49 podemos observar pela escala que o tempo morto foi definido em 1us.

Tempo de relaxe: Foi introduzida uma pausa entre os impulsos positivos e negativos do gerador de Marx. Podemos observar essa pausa entre a variação dos sinais T1 (Laranja) e T2 (Azul), sendo essa pausa de 2us. O tempo total de pausa (relaxe) será os 2us adicionando os dois tempos mortos de 1us, pelo que dará 4us de pausa.

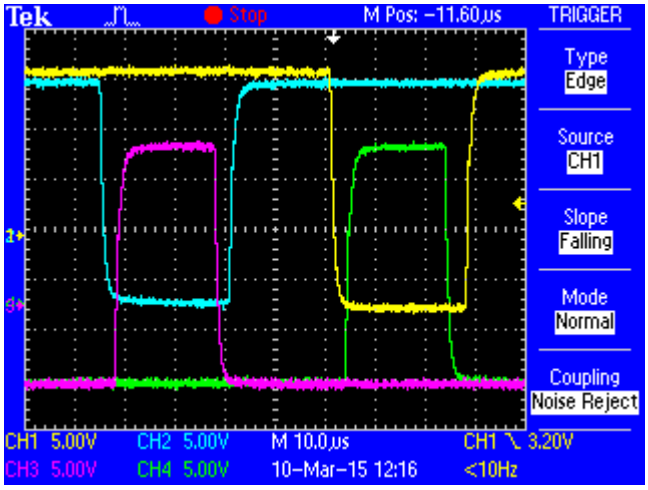


Figura 50 - Detalhe adicional dos tempos mortos

A Figura 50 apresenta também o detalhe dos sinais de comando e os tempos entre existentes entre as variações dos sinais.

10.3 Tensões e corrente na carga

Depois de verificados os sinais de comando era necessário observar na carga as correntes e tensões para confirmar o correto funcionamento do sistema. Assim foram colocadas pinças amperimétricas na carga para medir as tensões e correntes aplicadas.

O comportamento do sistema foi assim validado e considerado o desejado.

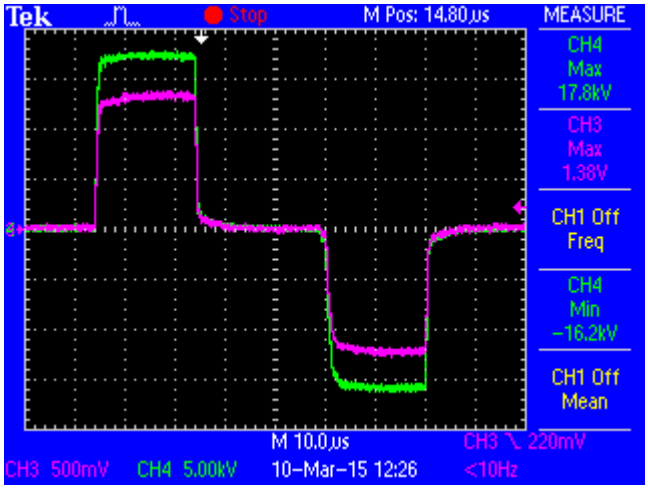


Figura 51 - Tensão (verde) e corrente (roxo) na carga: impulsos bipolar com 5 kV/div e 50 A/div numa carga resistiva, 10 µs/div.

Na Figura 51 podemos observar impulsos com uma largura de 20 µs, corrente (roxo) e a tensão (verde) aplicada na carga. De verificar que foi atingido um máximo de 17,8 kV de tensão e 135 A na carga.

Também e como esperado sendo o sistema aplicado a uma carga resistiva, a curva da corrente é uma réplica da curva de tensão na carga.

V. Conclusões

Avaliando o processo efetuado e avaliando o uso da FPGA na máquina de geração de sinais final pode-se afirmar que o objetivo foi cumprido na sua totalidade. Ou seja, foi conseguido através de uma placa FPGA comandar um conversor de potência. O conversor alvo foi um gerador de Marx bipolar mas poderia ter sido qualquer outro tipo de gerador devido ao dinamismo e capacidade de gerar sinais da FPGA.

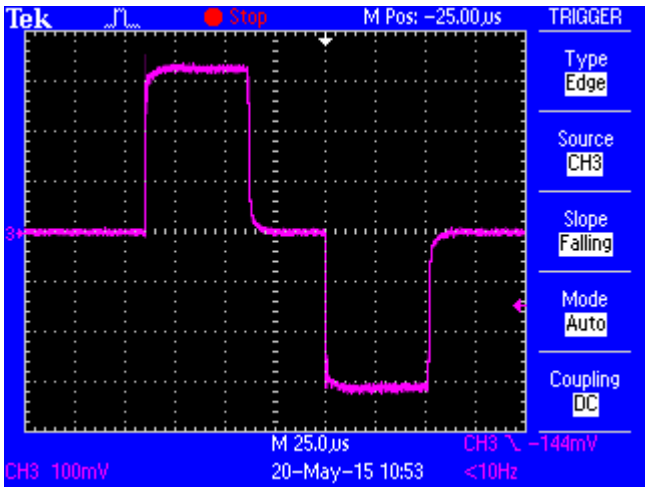


Figura 52 - Curva de tensão na Carga

Na Figura 52 podemos ver a curva de tensão na carga, em que podemos ver os dois impulsos (positivo e negativo) com duração de 50us.

Com a implementação feita foi possível controlar não só estes impulsos positivos e negativos, mas também o tempo morto entre estes dois impulsos, a frequência a que são gerados e controlar a fonte de alimentação para apenas alimentar os condensadores do gerador de Marx quando necessário.

Adicionalmente a FPGA dispõe de entradas GPIO que são sinais de erro provenientes do gerador de Marx e em caso de alguma falha, o sistema inibe automaticamente a geração de sinais de comando, bem como identifica qual o módulo do gerador em erro.

O sistema tem ainda muito por onde possa evoluir (funcionalidades), perfeitamente ao alcance das capacidades da FPGA.

VI. Referências

Bibliografia

[1] F. L. Luo, H. Ye e M. H. Rashid, Power Digital Power Electronics and Applications, Elsevier, 2005.

[2] Wikipedia, "Microcontrolador," [Online]. Available: https://pt.wikipedia.org/wiki/Microcontrolador.

[3] O. V. Ryazancev, M. V. Kylik e A. S. Manukyan, Efficient Application of Power Modules in Energy Converter with Microprocessor Control, Dneprodzerzhinsk, Ukraine: Dneprodzerzhinsk State Technical University, 2010.

[4] Y. Kanthaphayao, U. Kamnarn e V. Chunkag, The Parallel Operation Control of a Modular AC to DC Converter via Serial Communication Bus, Wiley Online Library, 2013.

[5] A. M. Pernía, J. Arias, M. J. Prieto e J. Á. Martínez, A modular strategy for isolated photovoltaic systems based on microcontroller, University of Oviedo, Department of Electrical Engineering, Campus de Viesques s/n, 33204 Gijo´n, Asturias, Spain: Elsevier, 2009.

[6] Wikipedia, "Digital signal processor," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Digital_signal_processor.

[7] Y. LIJIAO e S. SHIPING, Study of Intelligent Universal Transformer Operation Model and DSP control, The International Conference on Advanced Power System Automation and Protection, 2011.

[8] S.-Y. Ou, C.-H. Tien e C.-W. Tsai, Implementation of a Half-Bridge Single-Stage Converter Control Using DSP, Taipei, Taiwan: 2009 International Conference on

Power Electronics and Drive Systems, 2009.

[9] H. Zhou, C. Tong, M. Mao e C. Gao, Development of Single-phase Photovoltaic Grid-connected Inverter Based on DSP Control, IEEE International Symposium on Power Electronics for Distributed Generation Systems, 2010.

[10] B. Akin, S. Choi e H. A. Toliyat, DSP Applications in Electric and Hybrid Electric Vehicles, IEEE SIGNAL PROCESSING MAGAZINE, 2012.

[11] J. L. Bastos, L. M. Farronato, H. P. Figueroa, D. Franzoni, S. Lentijo, A. Monti, A. Smith e X. Wu, FPGA-Based Control of Power Converter: Comparing - FPG Alternative Solutions, IEEE, 2005.

[12] K. Jezernik e R. Horvat, FPGA Hybrid Controller for Unity Power Factor, IEEE, 2010.

[13] L. M. Redondo e J. F. Silva, "Repetitive High-Voltage Solid-State Marx Modulator Design for Various Load Conditions," *IEEE Transactions on Plasma Science (DOI: 10.1109/TPS.2009.2023221),* vol. Vol. 37, nº No. 8, pp. 1632-1637, August 2009.

[14] H. Canacsinh, L. M. Redondo e J. F. Silva, Optimizing Repetitive Bipolar Solid-State Marx Generators, Chicago: 18th IEEE International Pulsed Power Conference, 2011.

[15] Wiki, "VHDL," [Online]. Available: https://en.wikipedia.org/wiki/VHDL.

[16] Wiki, "Hardware description language," [Online]. Available: https://en.wikipedia.org/wiki/Hardware_description_language.

[17] N. Semiconductor, "ADC128S022," *Datasheet National Semiconductor ,* 2010.

[18] Altera, "ALTPLL (Phase-Locked Loop) IP Core User Guide," *Altera,* 2014.

[19] Altera, RAM Megafunction - User Guida, San Jose: Altera, 2004.

[20] R. Wain, I. Bush, M. Guest, M. Deegan, I. Kozin e C. Kitchen, An overview of FPGAs and FPGA programming, Daresbury: Computational Science and Engineering Department, CCLRC Daresbury Laboratory, 2006.

[21] Altera, "Altera," [Online]. Available: http://www.altera.com/devices/fpga/cyclone-iv/overview/architecture/cyiv-architecture.html.

[22] Altera , Altera Cyclone IV Device Handbook, Altera , 2014.

[23] D. J. Smith, "VHDL & Verilog Compared & Contrasted," VeriBest Incorporated , [Online]. Available: http://www.angelfire.com/in/rajesh52/verilogvhdl.html.

[24] [Online]. Available: http://www.ece.msstate.edu/courses/design/ece4532/2002_fall/av_switchbox/rd/Microcontroller%20VS%20FPGA.pdf.

[25] K. WAN, DVANCED CURRENT-MODE CONTROL TECHNIQUES FOR DC-DC POWER, MISSOURI UNIVERSITY OF SCIENCE & TECHNOLOGY, 2009.

[26] S. C. G. Sanders e H. D., Solid State Marx Generator, Applied Pulsed Power Inc., Freeville, NY 13068 USA.

VII. Anexos

11 Field-Programmable Gate Array (FPGA)

11.1 Introdução

Um circuito integrado *Field-Programmble Gate Array* (ou *FPGA*) [20] é um circuito integrado de silicone que contem um *array* de blocos lógicos configuráveis (*configurable logic blocks – CLB*). Ao contrário de um *Application Specific Integrated Circuit* (*ASIC*), que pode executar uma única e específica função durante o seu ciclo de vida, uma *FPGA* pode ser reprogramada para executar uma função diferente numa questão de microssegundos. Antes de ser programada, a *FPGA*, não conhece nada sobre como comunicar com os dispositivos circundantes. Este facto tanto é uma bênção como uma maldição, uma vez que permite uma grande flexibilidade em usar a *FPGA* mas incrementa bastante a complexidade da sua programação. A possibilidade de reprogramar as FPGAs têm-nas conduzido para uma grande variedade de utilização entre os *designers* de hardware na elaboração dos seus protótipos.

Outra grande vantagem das FPGA's (se não a maior) é o grau paralelismo que se consegue atingir na implementação dos circuitos lógicos. Podemos ter uma FPGA com várias funcionalidades independentes sem necessitar de sincronismo entre essas mesmas funcionalidades.

Nos pontos seguintes deste capítulo iremos descrever a FPGA utilizada, a Altera Cyclone IV E, e por associação a arquitetura genérica das FPGAs. As diversas FPGA's existentes no mercado diferem no seu modelo e capacidade física (número de I/O's, capacidade de memória, *clock* interno e outros) mas não nos conceitos básicos de utilização e de programação.

11.2 FPGA Altera Cyclone IV E

Neste capítulo será descrita a FPGA usada, a Altera Cyclone IV E, durante a componente experimental deste trabalho. Esta FPGA está embebida numa placa onde as interfaces com o circuito integrado da FPGA foram facilitadas, quer na programação quer nas interligações de I/O.

Será também dada uma introdução sobre a arquitetura e componentes das FPGA da Altera.

11.2.1 Arquitectura

As FPGAs Cyclone® IV [21] continuam a tradição da série Cyclone de oferecer uma sem precedente combinação de energia reduzida, funcionalidades elevadas e custo reduzido.

A arquitetura da FPGA Cyclone IV E consiste em até 115K (115.000) LE's verticalmente alinhados, 4 Mbits de memória embebida definida como blocos de 9-Kbit (M9K) e 266 (18x18) multiplicadores embebidos.

O núcleo lógico e de roteamento é rodeado por "I/O Elements" (IOEs) e por "phase-locked loops" (PLLs), como identificado na Figura 53. A FPGA utilizada contém quatro PLLs para uso genérico localizados em cada canto da matriz Este modelos tem também I/Os nos quatro lados da matriz, ao contrário de outros modelos da Altera.

A arquitetura desta FPGA inclui uma interconexão altamente eficiente e uma rede do sinal de relógio com "low-skew" (fenómeno no qual o sinal de relógio – "clock" – chega a diferentes componentes em diferentes tempos), fornecendo conectividade entre as estruturas lógicas para o "clock" e para os sinais de dados.

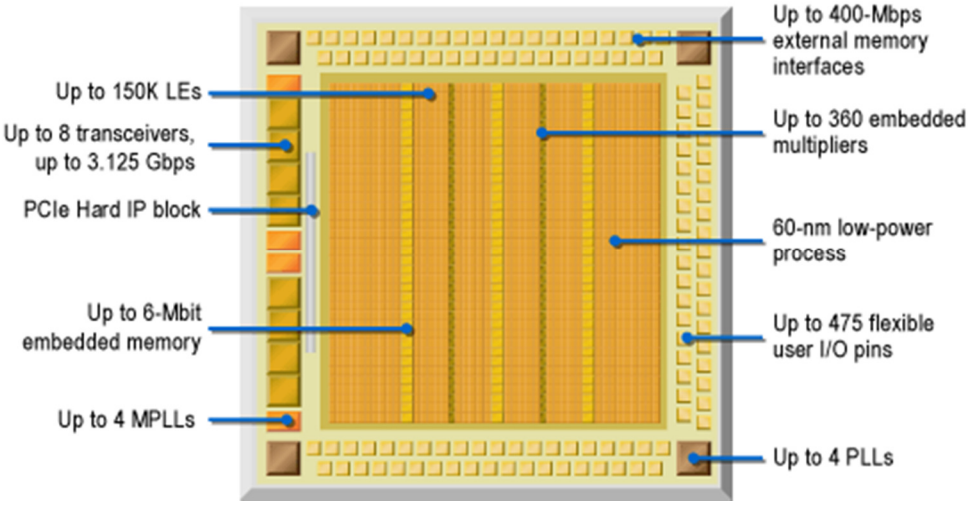


Figura 53 - Arquitectura da FPGA Cyclone IV E

A Figura 53 ilustra a arquitetura da FPGA Cyclone IV E.

11.2.2 *Logic Elements – LE*

Os "Logic Elements" (LEs) [22] são as unidades de lógica mais pequenas na arquitetura do dispositivo Cyclone IV. Os LEs são compactos e fornecem funcionalidades avançadas com uma utilização de lógica eficiente. Cada LE tem as seguintes funcionalidades:

- Uma "look-up table" (LUT) de quatro entradas, que pode implementar qualquer função de quatro variáveis;
- Um registo programável;
- Uma ligação de "carry" (que permite à FPGA ser eficiente nas operações aritméticas e na comunicação com LEs que a circulam)
- Uma ligação de "register" (que permite a comunicação com LEs que a circulam)
- A capacidade de conduzir as seguintes interconnecções:
    - Locais
    - Linha ("row")

- Coluna
- "Register Chain"
- Link directo

- "Register Packing" (uma funcionalidade que permite o registo ser combinado com outros elementos na mesma célula lógica, reduzindo o número de elementos lógicos necessários numa implementação)
- Suporte de "Register feedback"

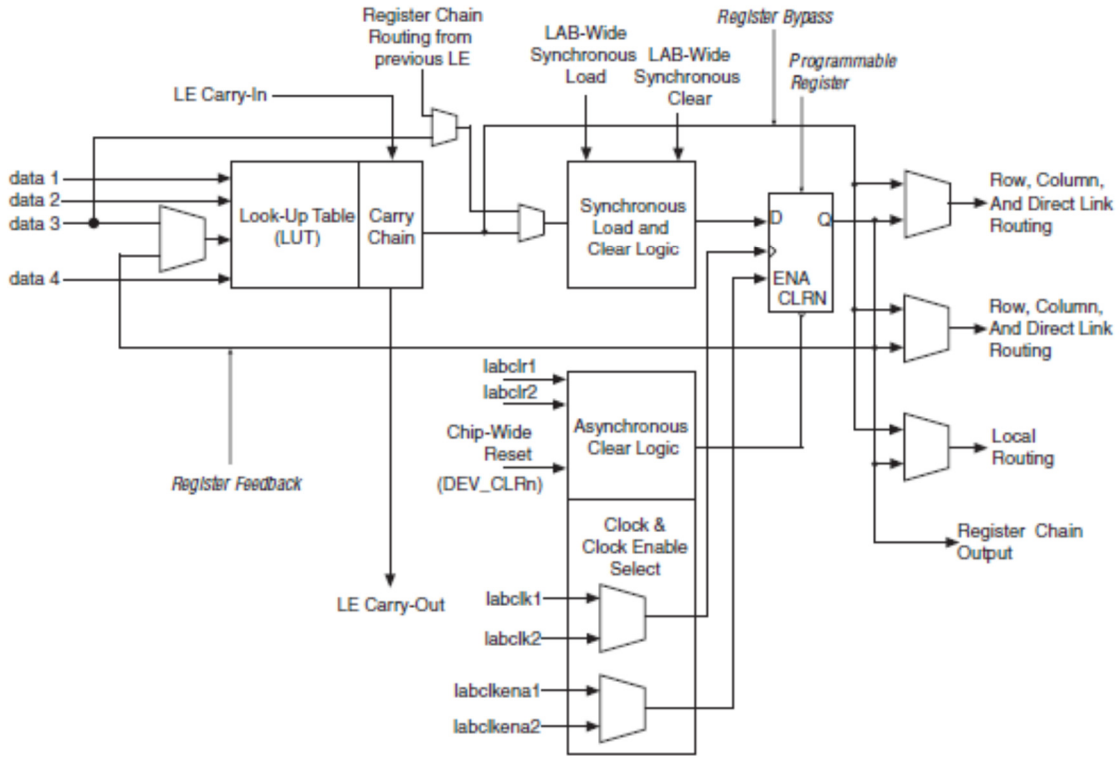


Figura 54 - Arquitectura dos LEs

A Figura 54 ilustra a arquitetura de um LE.

11.2.3 *Logic Array Blocks – LAB*

Os "Logic Array Blocks" (LABs) [22] contêm grupos de LEs.

Cada LAB consiste nas seguintes funcionalidades:

- 16 LEs
- Sinais de controlo da LAB
- Malha de sinais "carry" dos LEs
- "Register Chains"
- "Local interconnect"

A "Local interconnect" transfere sinais entre LEs do mesmo LAB. As ligações "Register Chains" transferem o output de um registo de uma LE para um registo adjacente de outra LE dentro da mesma LAB. O compilador "Quartus II" coloca a logica associada num LAB ou em LABs adjacentes, permitindo o uso das ligações locais e "Register chains" para uma melhor performance e maior eficiência da área.

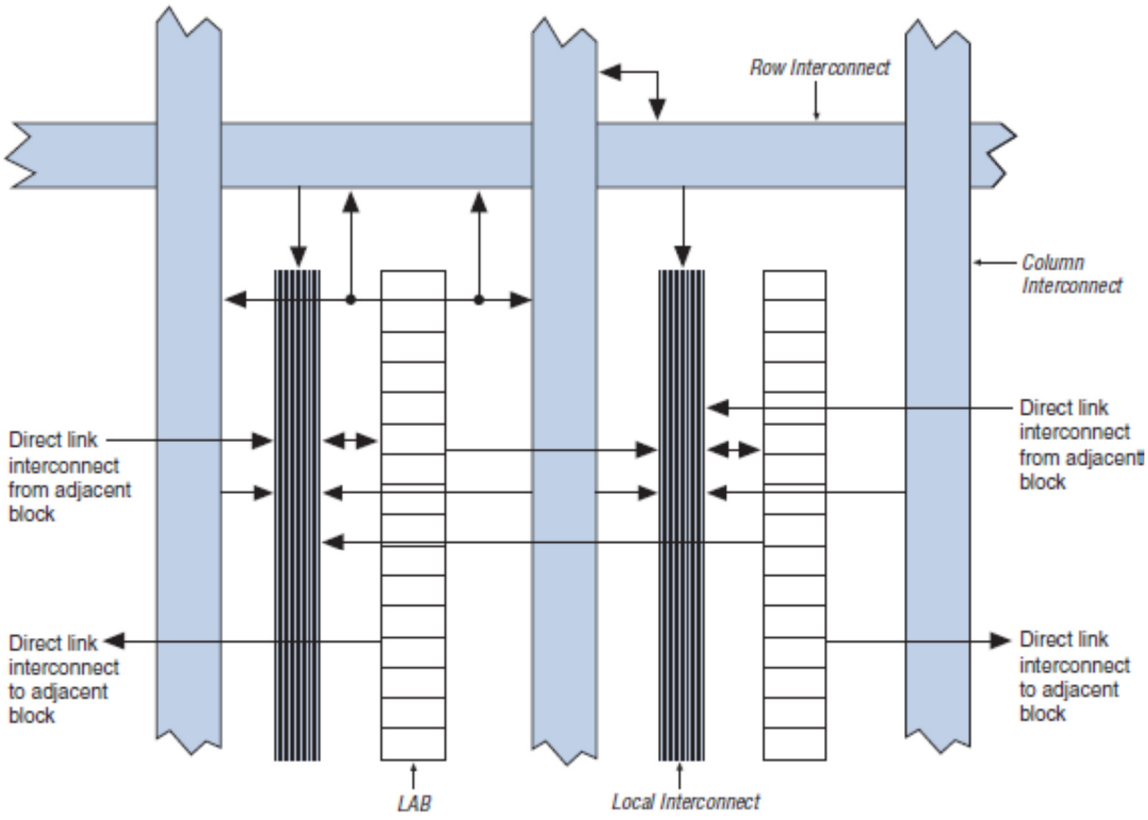


Figura 55 - Arquitetura dos LABs

A figura anterior ilustra a arquitetura de um LAB.

11.3 Programação da FPGA

O modelo standard de projeto e programação da FPGA começa pela utilização de esquemas ou por utilizando o "*Hardware Description Language*" (HDL), com sendo o Verilog HDL ou VHDL ("*Very high speed integrated circuit Hardware Description Language*"). Neste passo de projeto é criado o circuito que será implementado dentro da FPGA. O fluxo prosseguirá através da compilação, simulação, programação e verificação no *hardware* da FPGA.

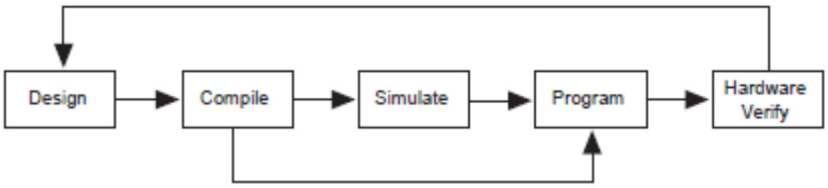


Figura 56 – Processo de programação da FPGA [23]

A Figura 56 ilustra o processo de programação da FPGA.

No projeto de implementação deste trabalho foi utilizado o VHDL. As duas linguagens de implementação apresentam capacidades iguais, no entanto foi escolhido o VHDL por permitir uma maior capacidade de modulação e de abstração [23].

Quaisquer das linguagens são *standards* de mercado.

Foi utilizado neste trabalho as ferramentas da Altera para programação da sua FPGA, nomeadamente o IDE "Quartus II 10.1 Web Edition (64-Bit)".

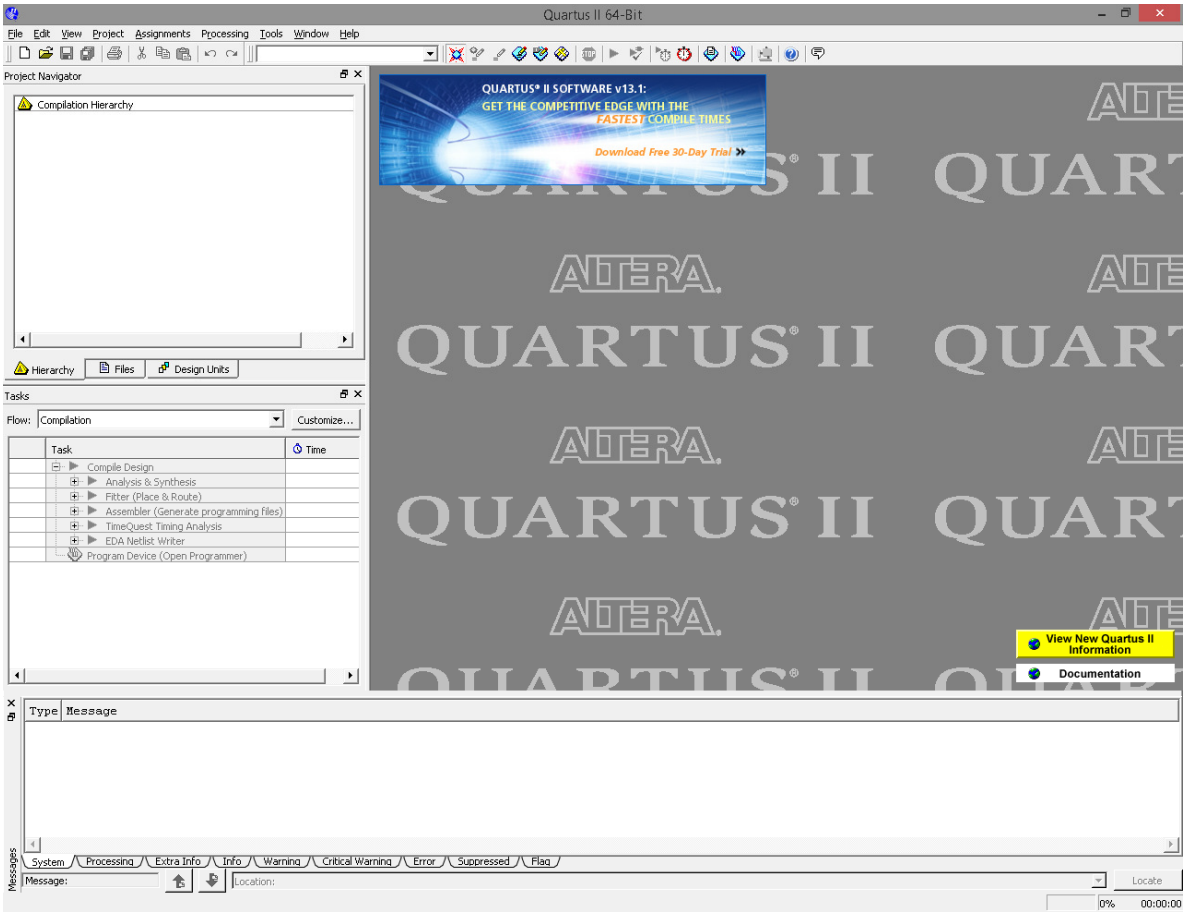


Figura 57 - Altera Quartus II

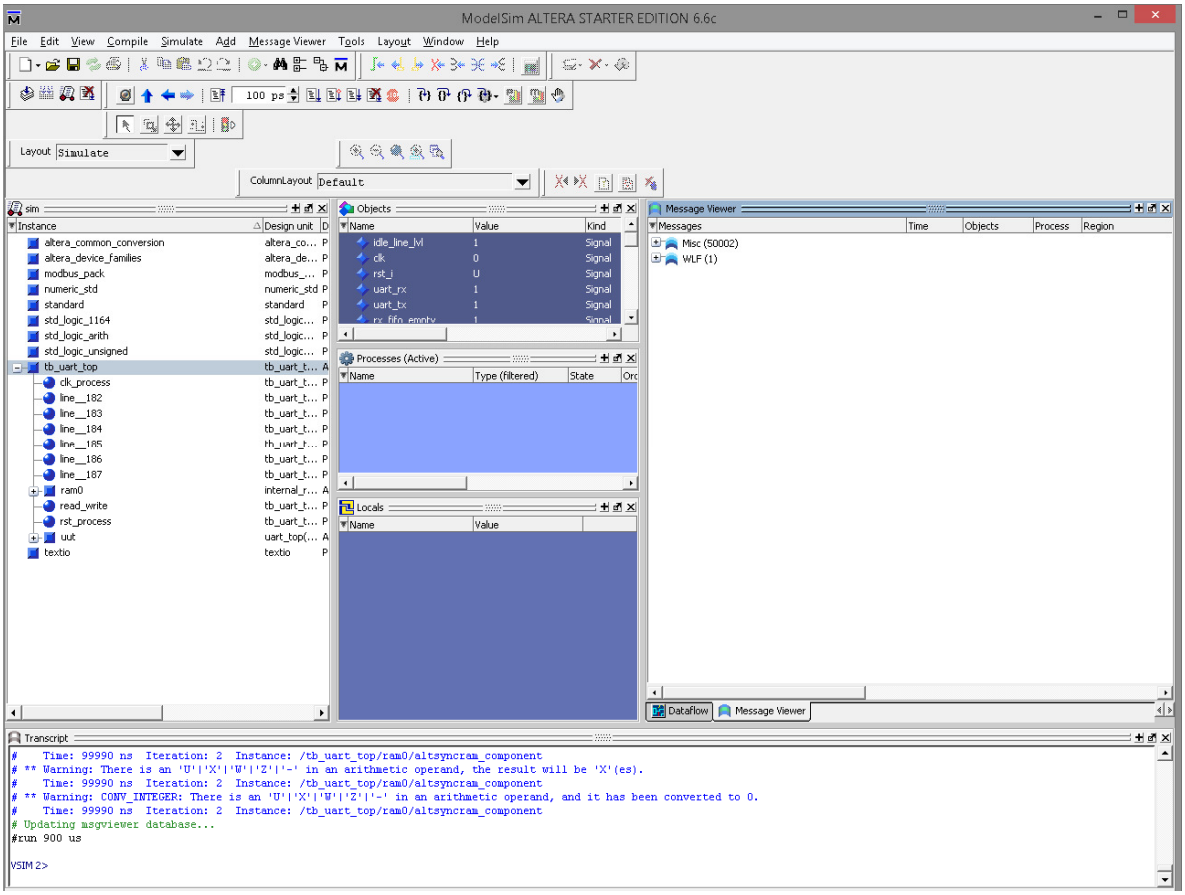


Figura 58 - Simulador da Altera - ModelSim

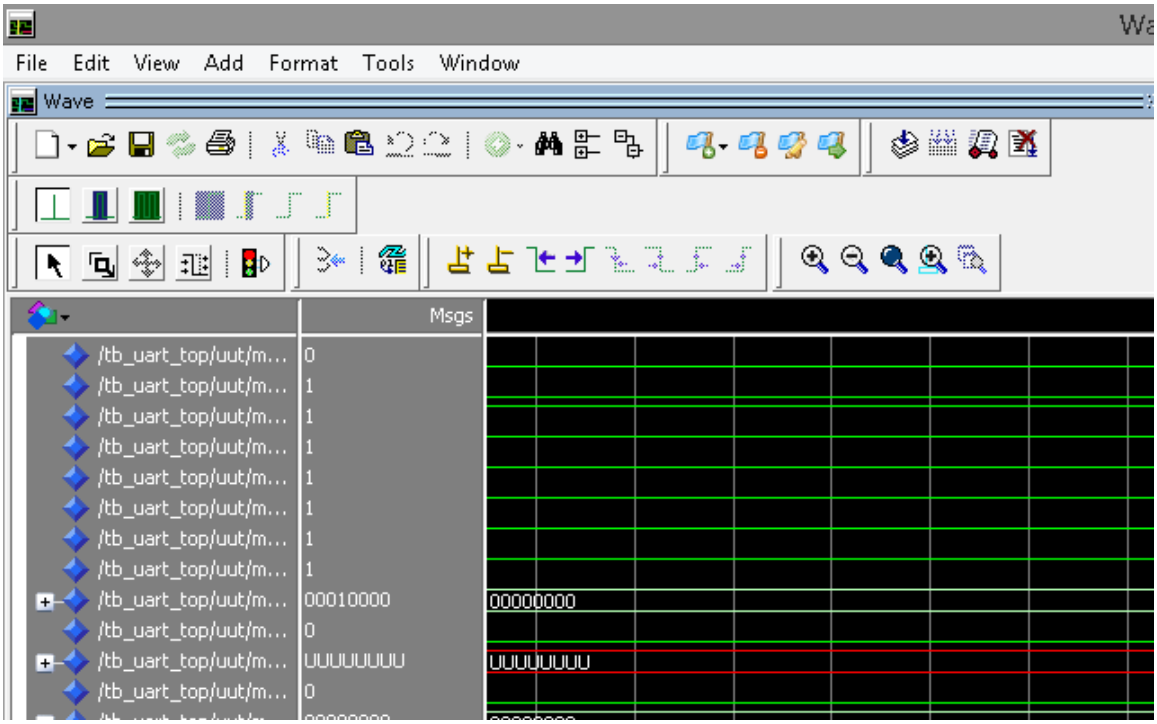


Figura 59 - Visualizador dos sinais em simulação

Nas três figuras anteriores são apresentados "*print-screens*" das ferramentas utilizadas da Altera:

- O IDE e compilador
- O Simulador
- O Visualizador dos sinais simulados

12 FPGA – Deo-Nano

12.1 Introdução

O trabalho efetuado assentou no uso de placas Deo-Nano, da Terasic, que entre outros componentes, contem uma FPGA Altera Cyclone® IV EP4CE22F17C6N com a capacidade máxima de 153 pins de I/O.

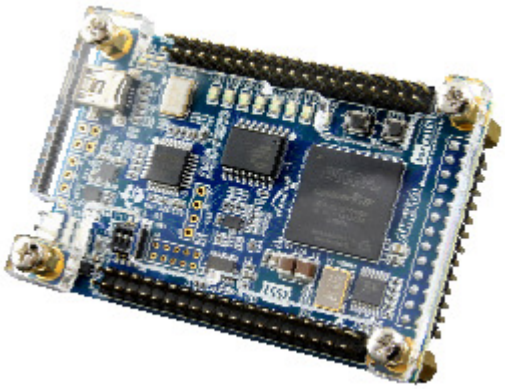


Figura 60 - Placa Deo-Nano

A Figura 60 mostra o aspeto físico da placa utilizada. É uma placa de dimensões reduzidas, bastantes pins de I/O disponíveis e de preço reduzido o que confere uma boa solução. Uma das grandes vantagens desta placa é que contém componentes como por exemplo as interfaces de USB para programação da FPFA ou a EEPROM para guardar a programação necessária no arranque da FPGA.

Mais detalhes serão descritos da Deo-Nano serão descritos no ponto seguinte.

12.2 Características da Deo-Nano

As características e funcionalidades principais da placa utilizada são as seguintes:

- FPGA embebida
  - Altera Cyclone® IV EP4CE22F17C6N
  - Máximo de 153 Pins de I/O
- Estado de configuração e elementos de *Setup*
  - Circuito USB-Blaster embebido na placa para a programação
  - Dispositivo Spansion EPCS64
- *Headers* de expansão
  - 2 Headers de 40 Pins (GPIOs) que fornecem 72 Pins de I/O, Pins de 5V, 2 Pins de 3.3V e 4 pins de massa
- Dispositivos de memória
  - SDRAM de 32MB
  - I2C EEPROM de 2Kb
- Inputs/Outputs de uso geral
  - 8 LEDs verdes
  - 2 botões
  - 4 *switchs* DIP
- *G-Sensor*
  - Acelerómetro de 3 eixos com alta resolução (13 bits), modelo ADI ADXL345
- Conversor A/D
  - Conversor A/D de 12 bits, 8 canais, modelo NS ADC128S022
  - Velocidades de conversão de 50 Ksps a 200 Ksps
- Sistema de Clock
  - Oscilador de 50Mhz embebido
- Alimentação

- Entrada USB tipo mini-AB de 5V
- Pin de 5V DC para cada *Header* de GPIO
- *Header* de energia externa de 2 Pins (3.6-5.7V)

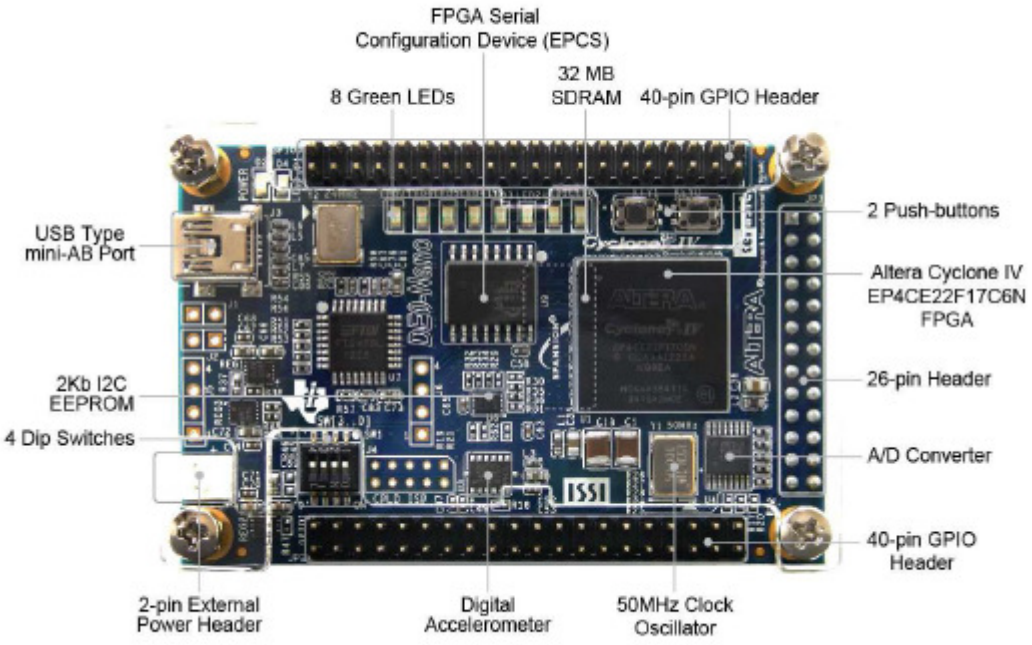


Figura 61 - Aspecto do Layout da Deo-Nano

Na Figura 61 podemos observar a disposição dos componentes da placa Deo-Nano e em especial da localização da FPGA.

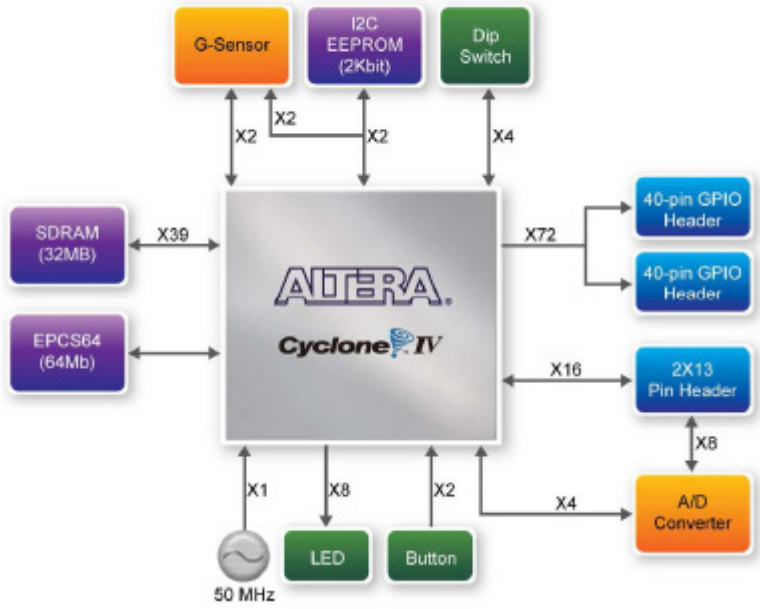


Figura 62 - Diagrama de Blocos da Deo-Nano

Na Figura 62 ilustra as ligações internas na Deo-Nano à FPGA, onde podemos observar os Pins disponíveis (40+40+8) para as interfaces que este trabalho irá utilizar no controlo e no disparo do gerador de Marx.

13 Referências sobre FPGA vs Microcontroladores

Neste capítulo apresentamos a listagem de algumas referências a estudos sobre as diferenças entre FPGA's e Microcontroladores.

[1]

http://www.edaboard.com/thread26691.html - FPVA vs Microcontrollers

[2]

http://www.fpga4fun.com/FPGAinfo1.html - FPGA

[3]

http://rtcmagazine.com/articles/view/102015 - FPGA vs Microcontrollers

[4]

http://www.differencebetween.net/technology/difference-between-fpga-and-microcontroller/

[5]

http://www.ece.msstate.edu/courses/design/ece4532/2002_fall/av_switchbox/rd/Microcontroller%20VS%20FPGA.pdf

[6]

http://en.wikibooks.org/wiki/Embedded_Control_Systems_Design/Processors

VIII. Notas

Ao longo deste trabalho foram muitos os desafios encontrados mas sempre ultrapassados com a ajuda das ferramentas disponíveis.

É importante salientar a importância vital das simulações e das capacidades de depuração (*debug*) da própria FPGA. Com a correta escolha do sistema de FPGA podemos ter acesso a portos JTAG (*Joint Test Action Group*) onde pode ser observado em "real-time" os valores internos dos registos. Só assim podem ser compreendidas as situações que acontecem quando o sistema é utilizado na realidade (fora do contexto da simulação).

Em termos de evolução do trabalho desenvolvido penso que passará por:

- Enriquecer a interface com mais informação;
- Controlar os módulos do gerador de Marx sem separado, podendo obter outras configurações de aplicação de tensão à carga:
    - Em sequência com tempos entre cada atuação dos módulos;
    - Ativação ou desativação de algum módulo em específico;
- Controlo da tensão da fonte;
- Pré-configurações no PLC baseado nas melhores práticas (frequência de disparo, largura de impulsos, tempos mortos, etc).