DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

Desenvolvimento do firmware e testes de hardware da nova plataforma de cartões de controle

Relatório submetido à Universidade Federal de Santa Catarina

como requisito para a aprovação na disciplina

DAS 5511: Projeto de Fim de Curso

Vinicius de Morais Justo

Florianópolis, Julho de 2016

Desenvolvimento do firmware e testes de hardware da nova plataforma de cartões de controle

Vinicius de Morais Justo

Esta monografia foi julgada no contexto da disciplina
DAS5511: Projeto de Fim de Curso
e aprovada na sua forma final pelo
Curso de Engenharia de Controle e Automação

Prof. *Rômulo Silva de Oliveira*

_____
Assinatura do Orientador

Banca Examinadora:

Dirk Lindeke
Orientador na Empresa

Rômulo Silva de Oliveira
Orientador no Curso

Prof. Miguel Angelo Chincaro Bernuy
Avaliador

Gustavo Wolf de Carvalho
Daniel Valenti
Debatedores

Agradecimentos

Primeiramente gostaria de agradecer aos meus pais, minha irmã e a toda minha família pelo suporte incondicional para que eu pudesse realizar este projeto na cidade de Jaraguá do Sul. Agradeço também a todos aqueles que tive contato dentro da WEG, principalmente ao pessoal do setor de engenheirados, com quem aprendi muito e pude crescer pessoalmente e profissionalmente.

Agradeço a minha namorada, Jamille Luiz Tramontin, pelo suporte incondicional que tive durante todo o período em que estive em Jaraguá e também pelas orientações e correções sugeridas para esta monografia.

Agradeço a todos os amigos que fiz neste longo período de graduação, em particular a minha turma 10.1, que sempre esteve unida e com quem compartilhei grandes momentos ao longo desses anos.

Por fim, agradeço a todos os professores do DAS com quem tive contato e são, em grande parte, responsáveis pela formação do quem eu pude me tornar tanto no aspecto pessoal, quanto no aspecto técnico. Agradeço em especial ao prof. Rômulo Silva de Oliveira pela orientação recebida para este projeto e por ser este grande profissional na qual pude ter contato em todos estes anos de graduação.

Resumo

O projeto apresentado neste documento visa realizar testes de hardware da nova plataforma de cartões de controle desenvolvido no departamento de produtos engenheirados, bem como todo o firmware implementado para a realização destes testes. Os testes que serão apresentados têm como objetivo validar todos os conceitos de hardware, comunicação, entre outros conceitos, para que a plataforma possa vir a ser validada e, posteriormente, utilizada em produtos. A plataforma tem como objetivo aumentar a longevidade da linha de cartões de controle, bem como aumentar a produtividade no desenvolvimento de novos produtos através do uso de hardware/firmware comuns aos diversos setores do departamento.

Está plataforma quando concluída, irá substituir os cartões de controle de produtos renomados, como os inversores CFW09 e MVW01, bem como soft-starters, conversores CA/CC, inversores solar e eólico, entre outros. Para abranger tal gama de produtos, a plataforma conta com processamento em dois de seus cartões, como será apresentado ao longo desta monografia, diferentes tipos de memória e várias interfaces de comunicação.

Abstract

The project presented in this thesis has as main goals to test the hardware of the new platform of control boards developed by the engineered products department as well as implement all the firmware required to do so. This platform will increase the longevity of control boards as well as increase the productivity of the development process for new products through the use of hardware/firmware common to the various sectors of the department.

This platform when concluded, will replace the control boards of renowned products like CFW09 and MVW01. To cover such range of products, the platform has processing units on two of its boards, as will be shown throughout this thesis, different types of memories and many communication interfaces.

Sumário

Simbologia

CA – Corrente alternada

CC – Corrente contínua

CLP – Controlador Lógico Programável

IHM – Interface Homem-Máquina

FPGA – “Field Programmable Gate Array”, circuito integrado projetado para ser configurado por um consumidor ou projetista após fabricação

CS – “Chip Select”, sinal de comando utilizado pelo microprocessador para selecionar o periférico

CI – Circuito integrado

MPU – “Microprocessor Unit”, processador

IGBT – “Insulated Gate Bipolar Transistor”, transistor bipolar de porta isolada

BSC – “Bus Space Controller”, controlador de áreas do barramento

Capítulo 1: Introdução

Este PFC foi realizado no departamento de Pesquisa e Desenvolvimento da unidade Drives & Controls – Automação da empresa WEG S. A., mais especificamente, no setor de Engenheirados. Este setor é responsável por desenvolver produtos e novas tecnologias como inversores de média tensão, inversor solar, conversores CA/CC, Soft-Starters, inversores para tração elétrica, entre outros.

Com sede situada na cidade de Jaraguá do Sul – SC e unidades fabris e escritórios situados em países como Brasil, Argentina, Alemanha, China, Estados Unidos, entre outros, a WEG vem se consolidando entre as maiores fabricantes de motores e equipamentos elétricos do mundo, cujas aplicações variam desde motores para equipamentos domésticos a motores de grande porte utilizados pela indústria. Além de motores, a WEG também está atuando em mercados como Automação, Tintas, Transmissão e Distribuição de Energia, Turbinas e Aerogeradores. Atualmente a WEG é considerada uma das melhores empresas brasileiras, recebendo em julho de 2015 o título de “Empresa do Ano” do anuário Maiores & Melhores da Exame.

Na unidade WEG Drives & Controls são desenvolvidas ainda soluções completas de automação (incluindo produtos como CLPs e Softwares de Supervisão), equipamentos para fornecimento ininterrupto de energia (No-Breaks, retificadores, carregadores e bancos de bateria) e produtos para controle de fluxo e pressão em bombas.

O departamento de P&D, ou departamento de desenvolvimento de produtos é responsável pelo desenvolvimento de produtos para controle e acionamento de motores (como Inversores de Frequência [1] [2], Conversores CA/CC, Servoacionamentos, Soft-Starters).

O departamento de P&D é subdividido em dois subsetores: seriados, onde são desenvolvidos produtos de base mais genérica e que possuem uma grande demanda, e engenheirados, que são produtos com maior grau de customização e que atendem a demandas específicas, em geral produtos de maior volume e valor agregados.

Apesar da variedade de produtos desenvolvidos pelo setor de engenheirados, grande parte deles possuem um conjunto hardware/firmware bem similar, e é nesse contexto que este projeto está inserido. A plataforma que será apresentada neste documento, tem como objetivo aumentar a longevidade da linha de cartões de controle, através da redução da diversidade de cartões em estoque, e por consequência, aumentando a durabilidade destes cartões no mesmo. Além disso, outro objetivo da plataforma é aumentar a produtividade no desenvolvimento de novos produtos, pois estes não precisarão desenvolver um hardware próprio. Isto é possível, pois a plataforma conta com o uso de hardware/firmware comuns aos diversos setores do departamento.

Este projeto tem como objetivo realizar testes de hardware na plataforma de cartões com o objetivo de validar conceitos como comunicação paralela, comunicação serial diferencial, comunicação utilizando o protocolo ModBus, acesso a diferentes tipos de memórias e utilização de componentes conversores (A/D e D/A) e demais funcionalidades para o acionamento de um motor de indução com modulação três níveis. Para isso, todo firmware implementado para realização destes testes também será abordado. Paralelamente, uma engenharia de software será apresentada com o intuito de estruturar a plataforma para um futuro firmware que possa vir a ser implementado na mesma.

Capítulo 2: Conceitos e descrição da plataforma de cartões de controle

Neste capítulo serão abordados alguns conceitos relevantes que foram utilizados na execução deste projeto, bem como uma descrição do produto que é a plataforma de cartões de controle. Antes de iniciar a descrição da plataforma, um termo muito utilizado na indústria é o termo de cartões. Este termo se refere a qualquer placa eletrônica com uma finalidade específica, que neste caso, é uma placa com finalidade de processar o algoritmo de controle do produto.

2.1: Plataforma de catões de controle

A plataforma de cartões de controle consiste em um conjunto de “cartões” (placas eletrônicas) utilizados em diversos produtos como inversores, soft-starters, conversores CA/CC, para implementação do sistema de controle. A denominação sistema de controle é empregada quando “um determinado conjunto de componentes interconectados tem como função principal a realização de uma ou mais ações que são observadas ao longo do tempo e cuja modificação decorre da aplicação de sinais de entrada. Estas ações podem ser o controle (ou regulagem) de posição, velocidade ou força em um cilindro, ou de vazão ou pressão em um circuito” [3]. No âmbito deste projeto, o sistema de controle para qual esta plataforma deve dar suporte consiste a sistemas de controle para motores e os equipamentos que serão descritos no decorrer deste capítulo possuem aspectos construtivos para esta finalidade.

A plataforma de controle é composta por quatro cartões de finalidades distintas (Figura 1). O cartão principal de controle, nomeado CCE-03, é responsável pelo processamento do sistema de controle e a transmissão dos sinais de controle para um segundo cartão, nomeado iFOI (intelligent fiber optic interface). Para alimentar este conjunto eletrônico, dois cartões realizam a conversão da energia da rede para diversas tensões DC que são utilizadas neste conjunto.

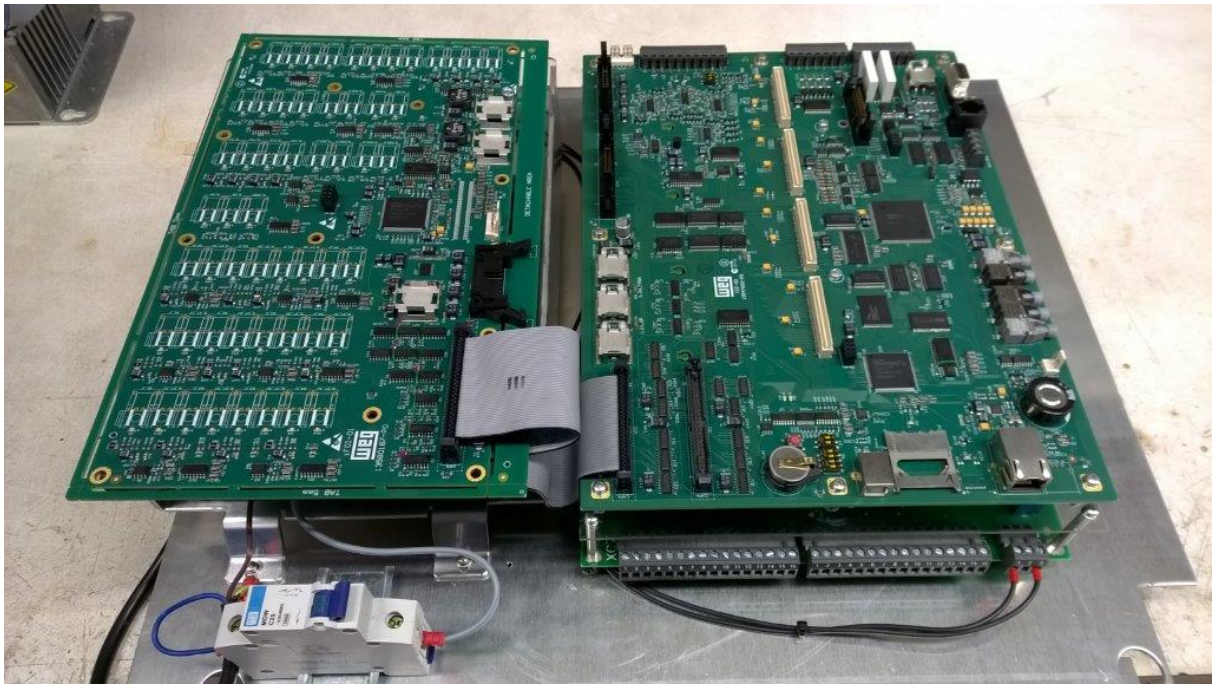

Figura 1 - Protótipo da nova plataforma de cartões de controle

2.1.1: Cartão CCE-03

O cartão CCE-03 (Figura 2) é o cartão principal da plataforma. Sua finalidade é processar o algoritmo de controle, enviar os sinais de controle para o cartão iFOI e monitorar eventuais falhas que podem ocorrer no sistema. Além destas funcionalidades, o cartão também conta com alguns dispositivos para oferecer suporte a interfaces com clientes, como interface IHM, USB, Ethernet.

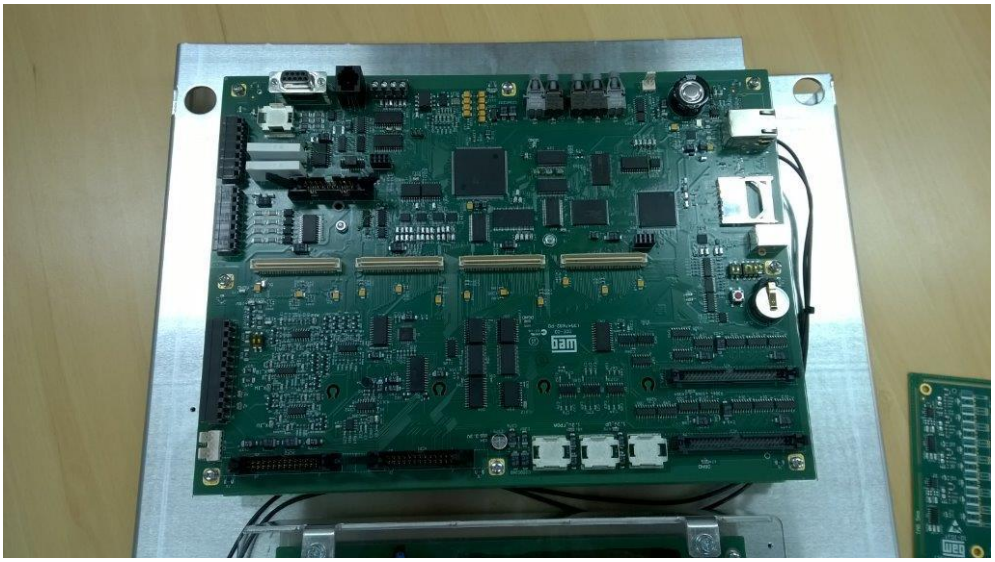

Figura 2 - Cartão de controle CCE-03

2.1.1.1: Unidades de processamento

Este cartão conta com duas unidades de processamento: um microprocessador, cuja finalidade principal é executar o algoritmo de controle e realizar o processamento de dados para interfacear com periféricos destinados a cliente, e uma FPGA, cuja finalidade principal é receber e processar (proteções IGBT´s) os dados de controle da MPU, realizar a transmissão do sinal de controle para a iFOI e fazer o monitoramento de eventuais falhas do sistema.

2.1.1.1.1: Microprocessador

O microprocessador é “um dispositivo lógico programável em um único chip de silício, concebido sob a tecnologia VLSI (circuito integrado em alta escala). Ele age sob o controle de um programa armazenado em memória, executando operações aritméticas, lógica booleana, tomada de decisão, além de entrada e saída, permitindo a comunicação com outros dispositivos periféricos” [5].

O microprocessador utilizado na plataforma é fabricado pela empresa Renesas Electronics e tem como finalidade o uso industrial. Este microprocessador conta com um core ARM Cortex-A9 e pode operar a uma frequência de até 400MHz.

Dentre as vantagens deste processador para plataforma estão:

- Ampla memória interna (10Mb de RAM);
- Conexão direta a SRAM, SDRAM e burst ROM;
- PWM destinado a controle de motor;
- 2 timers de alta capacidade (32 bits);
- 8 canais de transmissão serial;

Algumas desvantagens do uso deste processador incluem:

- Pouca experiência com o funcionamento do core ARM Cortex-A9;
- A/D de baixa velocidade e resolução (12 bits);
- Poucas áreas de chip select para uso de bus externo paralelo;

2.1.1.1.2: FPGA

Os PLDs (Programmable Logic Devices) são “circuitos integrados que podem ser configurados pelo próprio usuário. Não apresentam uma função lógica definida, até que sejam configurados. Possuem, como principal característica, a capacidade de programação das funções lógicas pelo usuário, eliminando-a do processo de fabricação do circuito integrado, o que facilita, assim, as prováveis mudanças de projeto” [4]. Um FPGA (Field Programmable Gate Array) é “um PLD que suporta a implementação de circuitos lógicos relativamente grandes. Consistem em um grande arranjo de células lógicas ou blocos lógicos configuráveis contidos em um único circuito integrado” [4].

O FPGA utilizado nesta plataforma é fabricado pela empresa Xilinx, Inc e conta com vários recursos destinados a uso Industrial.

Dentre as vantagens oferecidas por esta linha de FPGA estão:

- Aritmética e processamento de sinais de alta performance;
- Controlador de memória integrado (Taxa de transmissão de dados até 800Mb/s);
- Ferramentas de manuseio de clock para performances aprimoradas (DCMs e PLLs);

- Possibilidade de uso de um Microblaze (processador simulado).

Algumas desvantagens da FPGA especificada incluem:

- Número de pinos programáveis relativamente baixo para a aplicação;
- Ambiente de desenvolvimento carente de atualizações e com alguns bugs em relação a sistemas operacionais mais recentes;

2.1.1.2: Barramentos de comunicação

Barramento é um conjunto de linhas de comunicação que permitem a interligação entre dispositivos, como CPU, a memória e outros periféricos. Este cartão possuí três barramentos principais: o barramento paralelo, responsável pela comunicação do processador com os demais periféricos; o barramento serial diferencial via cabo *"flat"*, onde são realizadas as trocas de dados entre a FPGA do cartão CCE-03 e a FPGA do cartão iFOI; e o barramento serial via RS485, onde é feita a comunicação deste cartão com IHMs proprietárias.

2.1.1.2.1: Barramento Paralelo

O barramento paralelo pode conter vários fios, por onde passam vários sinais simultaneamente, um por cada fio. Este tipo de barramento possui 3 conjuntos de vias com funções distintas:

- Vias de comunicação de dados: onde trafegam os dados. Estas vias são do tipo bidirecional.
- Vias de comunicação de endereços: onde a unidade de processamento mestre indica o endereço a ser acessado. Estas vias são unidirecionais, sendo sempre controladas pelo dispositivo mestre.
- Vias de comunicação de controle: onde trafegam solicitações e confirmações. Estas vias também são do tipo bidirecional.

O cartão CCE-03 utiliza este tipo de barramento para comunicação do processador com seus principais periféricos, como: memória NOR, SDRAM, SRAM, periféricos de I/O, entre outros. A figura 3 mostra um esquemático deste barramento.

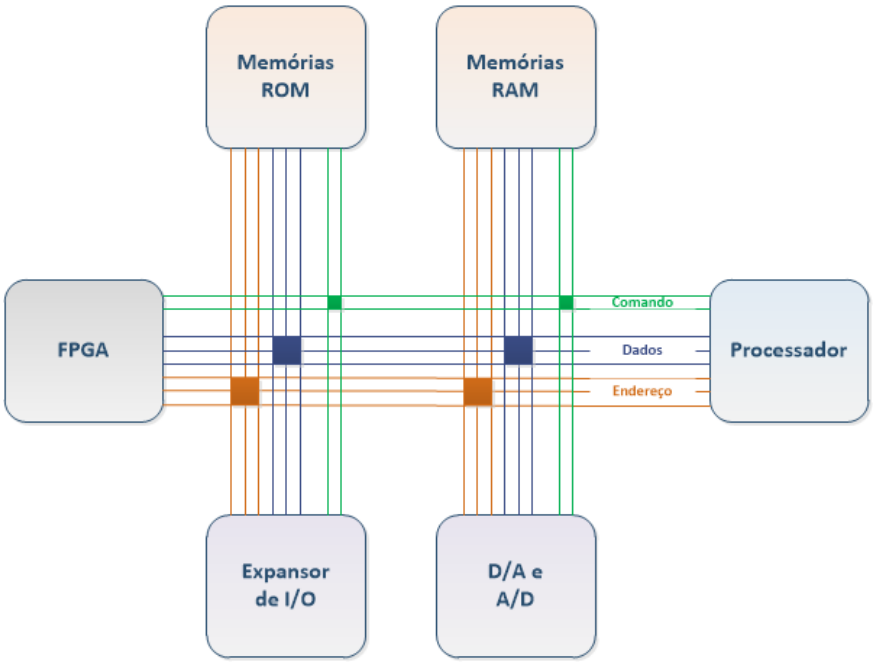

Figura 3 - Esquemático do barramento paralelo do cartão CCE-03

2.1.1.2.2: Barramento serial diferencial via cabo "flat"

O barramento serial diferencial é composto por 4 vias de transmissão (2 canais diferenciais para envio "*TX*" e 2 canais diferenciais para recepção *"RX"*). Os dados são transmitidos serialmente e utilizam um cabeamento do tipo *"flat"* como meio físico.

O cartão CCE-03 utiliza este barramento para realizar a comunicação entre sua FPGA e a FPGA do cartão iFOI (Figura 4). Conforme será mostrado em capítulos subsequentes, a comunicação entre estas duas FPGAs pode chegar a uma velocidade de transmissão de até 40MHz.

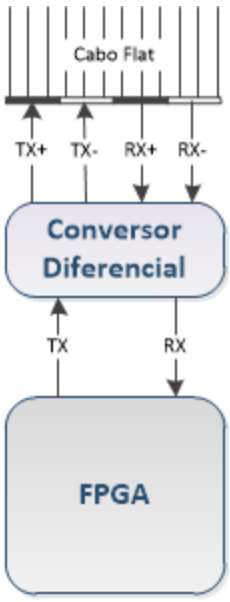

Figura 4 - Esquemático da transmissão serial diferencial

2.1.1.2.3: Barramento serial RS485

O padrão RS-485 é “baseado na transmissão diferencial de dados, através de um par de fios, que é ideal para transmissões em altas velocidades, longas distâncias e em ambientes propícios a interferência eletromagnética” [6].

O cartão CCE-03 faz uso deste padrão devido a compatibilidade com diversas IHMs presentes na casa.

2.1.1.3: Memórias

Memórias são dispositivos responsáveis pelo armazenamento de dados e instruções em forma de sinais digitais. Existem dois tipos principais de memórias: *ROM (Read-Only Memory)* que são memórias não voláteis, ou seja, os dados continuam armazenados após desenergização; e *RAM (Random Access Memory)* que são memórias voláteis de acesso rápido.

Este cartão possui memórias do tipo EEPROM, Flash QSPI, Flash NOR, SDRAM e SRAM.

2.1.1.3.1: EEPROM

As memórias EEPROM (Electrically erasable programmable read-only memory) são memórias não voláteis que permite a regravação de dados de maneira elétrica. Ao contrário de outras memórias não voláteis, bytes individuais de uma EEPROM podem ser lidos, apagados e reescritos de forma independente. O cartão de controle CCE-03 utiliza uma memória EEPROM com interface serial para dar suporte a plataformas mais antigas que já utilizavam este tipo de memória para armazenamento do firmware.

2.1.1.3.2: Flash QSPI (Quad-SPI) e Flash NOR

A memória Flash, embora seja uma EEPROM, é “um tipo de memória com características muito semelhantes à memória RAM, apenas com o diferencial crucial de ser não volátil. ”[7].

As memórias Flash possuem as seguintes características principais:

- A memória flash é silenciosa (em relação a outros meios de armazenamento em massa como discos rígidos);
- Possuí um acesso rápido;
- Pequena em tamanho;
- Baixo custo;
- Armazena uma quantidade grande de dados.

A memória flash do tipo NOR foi a primeira memória flash a se popularizar no mercado por volta de 1988. Os chips de memória NOR possuem uma interface de endereços similar à de uma memória RAM, o que possibilita que softwares armazenados no chip de memória flash sejam executados diretamente, sem precisarem ser copiados para a memória RAM. Por causa desta característica, o cartão CCE-03 tem uma memória NOR com a finalidade de possibilitar uma alternativa para armazenar o firmware que será executado na hora do boot do sistema. As memórias NOR tem a desvantagem de serem lentas para gravação, tornando-as inadequadas para usos como o de um drive de estado sólido *SSD*.

A memória flash QSPI é uma memória flash de interface serial que pode ser acessada via protocolo SPI tradicional, SPI de 2 bits (Duo) ou SPI com 4 (Quad) canais entrada/saída. Esta nova tecnologia de memória utiliza clocks de alta frequência para ter uma performance similar, ou em muitos casos, superior a tradicionais memórias paralelas, como as memórias flash NOR. Esta memória é mais uma alternativa para armazenamento do firmware do cartão CCE-03.

2.1.1.3.3: SRAM e SDRAM

Os circuitos de memória RAM (Random Access Memory) são classificados em dois grupos principais: memórias dinâmicas (DRAM) ou memórias estáticas (SRAM).

As memórias dinâmicas usam minúsculos capacitores para armazenarem os dados em seu interior. Como os capacitores perdem carga ao longo do tempo, é necessário fazer de tempos em tempos uma varredura na memória recarregando estes capacitores, este processo é chamado de *"refresh".*

As memórias estáticas utilizam circuitos chamados flip-flops para armazenarem dados em seu interior. Estes circuitos são bem maiores se comparados aos capacitores das memórias dinâmicas, entretanto estas memórias são mais rápidas, já que não precisam de *refresh.* [8]

Este cartão contêm uma memória estática (SRAM) e uma memória dinâmica (SDRAM) para que o usuário da plataforma possa trabalhar durante operação. A memória SRAM possuí ainda um supercapacitor, cuja duração da carga pode chegar até 1 semana, para que o usuário da plataforma possa armazenar alguns dados de forma rápida caso haja uma falha de energização do sistema.

2.1.1.4: Conversores analógico-digital

Diversas grandezas físicas com as quais lidamos são grandezas analógicas por natureza. No âmbito deste projeto devemos lidar com grandezas como nível tensão e nível de corrente elétrica que são grandezas representadas por valores contínuos e que para serem processadas pela plataforma precisam ser convertidas para uma cadeira de bits. De maneira similar, para que a plataforma possa se comunicar através destas

grandezas, é necessário que haja uma decodificação de uma cadeia de bits para um valor analógico (contínuo) [9]. A figura 5 detalha este processo de conversão.

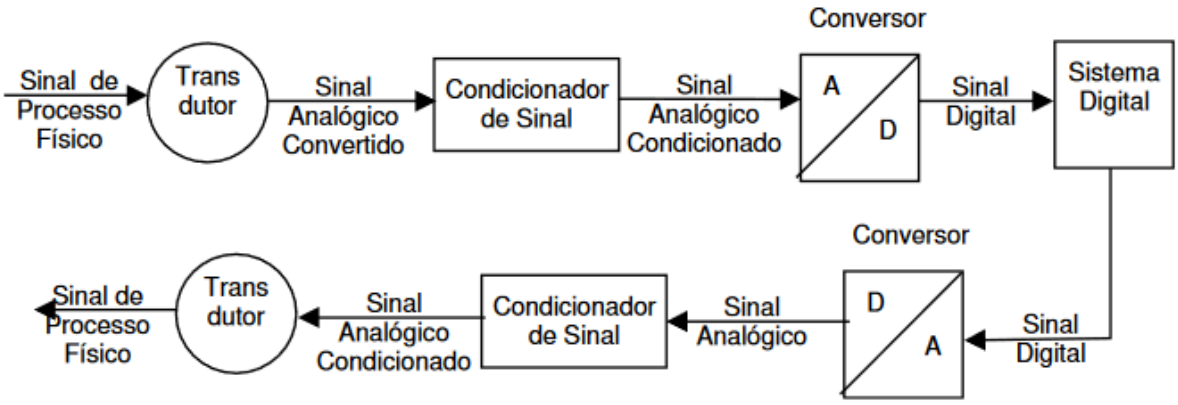

Figura 5 - Esquemático de conversão de grandezas contínuas em sistemas digitais e vice-versa

2.1.1.4.1: Conversores A/D

A conversão de grandezas analógicas para números digitais é realizada por conversores Analógico/Digital (A/D). Diferente dos conversores D/A, que serão vistos na seção seguinte, existem diversos circuitos para realizar a conversão A/D. Destes circuitos conversores apenas um deles realiza a conversão diretamente, que é conhecido como “Conversor flash” e é muito rápido, os demais conversores utilizam circuitos realimentados onde o valor digital é obtido pela comparação do valor analógico com o valor digitalmente estimado para ele. Estes circuitos conversores por aproximação são muito baratos, quando comparados a conversores flash, mas possuem a desvantagem de terem um tempo de conversão muito maior [10].

O cartão de controle CCE-03 possui um conversor A/D do tipo aproximação com 6 canais de entrada e com resolução de 12 bits. O tempo de conversão deste CI é de aproximadamente 33us para a conversão simultânea de dois canais. Este componente foi escolhido por ser um componente consolidado por já vir sendo utilizado em cartões de outras plataformas.

2.1.1.4.2: Conversores D/A

Em sua grande maioria os conversores Digital/Analógico (D/A) são circuitos assíncronos e que se utilizam de circuitos analógicos para realizar a conversão. A conversão de números binários para valores digitais é realizada por circuitos lineares capazes de somar tensões ou correntes com pesos proporcionais aos pesos dos bits que produziram individualmente cada corrente ou tensão [10].

O conversor D/A presente neste cartão possui 4 canais de saída e 12 bits de resolução de entrada. Assim como o conversor A/D, este CI foi escolhido por ser um componente relativamente barato e por já vir sendo utilizado em cartões de outras plataformas.

2.1.1.5: Conectores para slot de expansão

Esta nova linha de cartões CCE-03 conta com slots de expansão para que o usuário da plataforma possa utilizar cartões de acessório. As funcionalidades dos diversos acessórios variam desde expansão e processamento auxiliar de controle até acessórios com funcionalidades de interface com o cliente, como Ethernet, fieldbus, CAN, entre outros.

2.1.2: Cartão iFOI (Interlligent Fiber Optic Interface)

O cartão iFOI (Figura 6) tem como sua principal finalidade a transmissão dos sinais de controle através de fibras óticas. Este cartão possui também uma unidade de processamento para realizar um pré-tratamento de sinais de feedback (que de outra forma teriam que ir até o cartão de controle) e também realizar a comunicação com o mesmo.

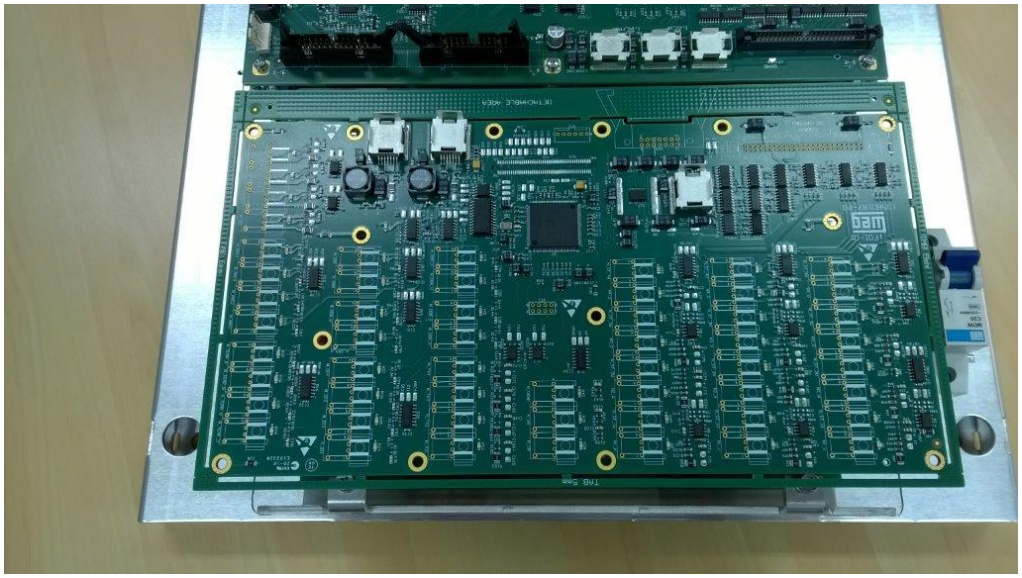

Figura 6 - Cartão de interface fibra ótica (iFOI)

2.1.2.1: Processamento

Este cartão conta com um FPGA para realizar o pré-processamento de sinais vindos das fibras óticas e implementar o protocolo de comunicação com o cartão de controle. O FPGA escolhido para este cartão é o mesmo que foi utilizado no cartão de controle CCE-03 devido a boa compatibilidade deste com a aplicação e a redução de custos de fabricação.

2.1.2.2: Barramentos de comunicação

Este cartão possui a extensão do barramento diferencial via cabo “flat” utilizado pelo cartão de controle. É por ele que ocorrem as trocas de informações (envios de comandos e recebimento de *feedbacks*) entre os dois cartões.

2.1.2.3: Canais de fibras óticas

Este cartão possuí duas versões de fabricação onde a principal diferença entre eles está no número de fibras óticas. A versão mais básica possui em torno de 55 canais de transmissão/recepção, enquanto a versão mais completa deste cartão possui cerca de 168 canais.

2.1.3: Cartão PIC (Power Interface Card) e fonte de alimentação

Este conjunto de cartões foi herdado de outros produtos WEG e é responsável pela alimentação da plataforma. Em resumo o conjunto é responsável por converter a energia alternada que vem da rede para tensões de 5V, 15V, -15V e 24V DC isoladas. Este cartão também faz o tratamento de sinais elétricos usados no controle dos conversores, como TC´s (transformador de corrente), entradas e saídas digitais.

2.2: Conclusão do capítulo

Neste capítulo foi realizada uma descrição da plataforma de cartões de controle, bem como uma descrição mais detalhada de seus principais cartões e componentes. A descrição de cada cartão e seus principais componentes servirá de embasamento para a concepção dos capítulos seguintes.

Capítulo 3: Engenharia de software

Este capítulo irá abordar a engenharia de software realizada para estruturação do firmware da nova plataforma. Também serão abordadas algumas discussões a respeito de problemas de estruturação enfrentados por outras plataformas da empresa e algumas propostas de solução.

O conteúdo apresentado neste capítulo é resultado da análise de estruturações de outras plataformas da empresa, da colaboração de colegas pesquisadores e de soluções encontradas através de pesquisas.

3.1: Estruturação da plataforma

A estruturação do firmware plataforma foi realizada com o objetivo de modularizar o acesso a todos os componentes presentes no cartão, deixando assim uma plataforma transparente para o usuário e evitando acessos indevidos.

Para realizar esta estruturação do firmware, foi necessário identificar todos os *"drivers"* presentes na plataforma e isolar o acesso de hardware destes drives de suas funções lógicas de acesso. Para que isso fosse possível, os seguintes passos foram executados:

1. Apenas os componentes de hardware que possuem um acesso ativo (que possuem uma lógica de ativação) foram adicionados ao firmware da plataforma;
2. Para cada *device driver* adicionado, um módulo lógico deve ser adicionado para seu controle;
3. Por último é necessário verificar a necessidade de incluir novos módulos lógicos de funções essenciais para a plataforma.

A identificação dos módulos da plataforma resultou em 20 módulos de *device drivers* e 37 módulos lógicos. Para cada módulo identificado, foi criado um arquivo de

cabeçalho para finalizar a estruturação do firmware da plataforma. A figura 7 mostra a estruturação final da plataforma de maneira resumida.

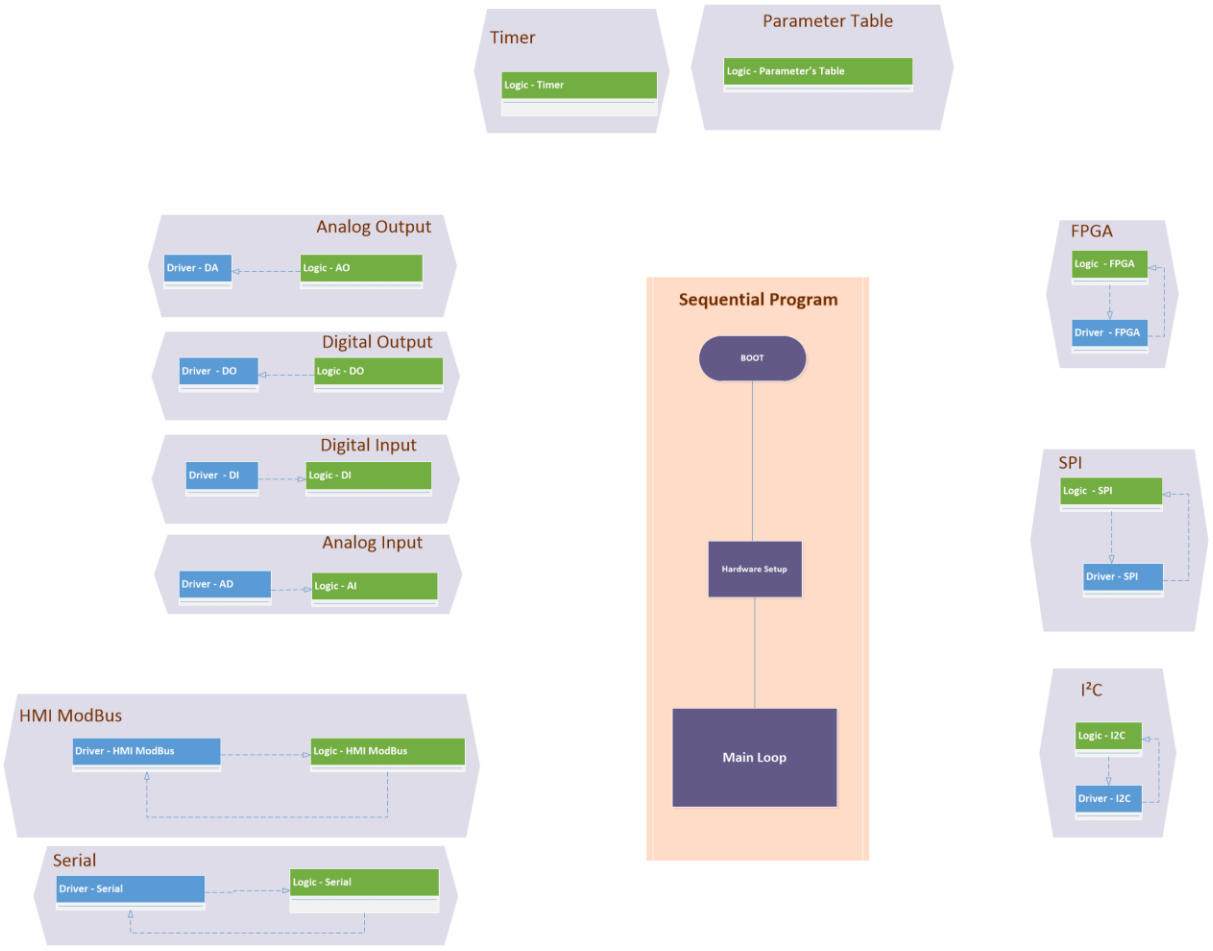

Figura 7 - Estruturação resultante da plataforma de maneira resumida

3.2: Padronização na escrita de códigos

Para que fosse elaborado os cabeçalhos da estruturação mostrada na seção 3.1, foi necessário criar um modelo padrão de arquivos de cabeçalho e arquivos de conteúdo de forma a padronizar todo código escrito para a plataforma.

Os modelos criados baseiam-se no uso da ferramenta Doxygen, que é uma ferramenta de auxílio a documentação de códigos de programas, e seguiram as seguintes práticas de boa programação [11]:

- Cada módulo (com seus respectivos .h e .c) deve corresponder a uma funcionalidade bem clara;

- Arquivos de cabeçalho devem possuir uma prevenção de inclusão múltipla;
- Toda declaração necessária para uso do módulo deve estar em seu arquivo de cabeçalho;
- Declarações locais ao módulo não devem aparecer no arquivo de cabeçalho.

Os arquivos modelo podem ser visualizados nos apêndices A e B.

3.2.1: Doxygen

Doxygen é um software livre de geração de documentação através de anotações presentes em códigos fonte. Este programa suporta linguagens como C, C++, C#, PHP, Java, Python, IDL, Fortran, VHDL e Tcl.

Este aplicativo permite que a documentação seja gerada na forma de documentação online para ser visualizada no browser do computador (Figura 8) ou um manual de referência (em Latex).

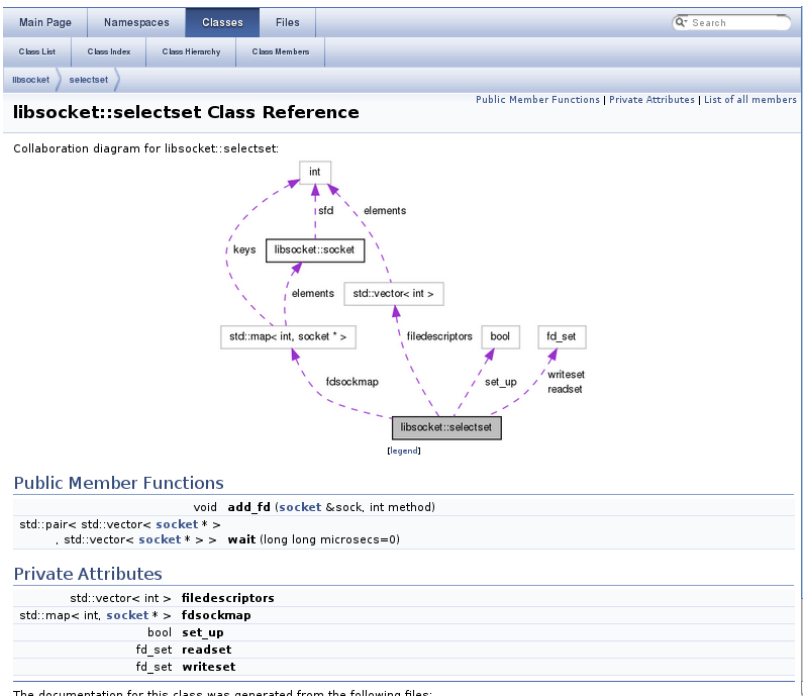

Figura 8 - Imagem ilustrativa da documentação gerada pela ferramenta Doxygen para visualização em browser

3.3: Estruturação de variáveis

Devido a limitação de recursos das plataformas anteriores (tamanho da pilha, frequência de trabalho do processador, tamanho da memória), uma solução rápida utilizada para elaboração do firmware era o uso de variáveis globais. Esta prática de programação deve ser evitada, pois pode ocasionar acessos e alterações não esperadas em variáveis críticas do sistema.

Para evitar a prática mencionada acima, foi solicitado pela empresa que fosse realizado uma nova estruturação para as variáveis desta plataforma. Após a realização de pesquisas e utilização de conhecimentos adquiridos no curso de controle e automação, foi proposto que as variáveis do firmware desta nova linha de cartões deverão seguir a seguinte ordem de implementação (Figura 9):

1. As variáveis deverão ser locais a cada módulo, não podendo ser acessadas externamente;
2. Quando há necessidade de uma variável ser acessada por outros módulos e não ocorre acesso simultâneo a ela, a variável deve permanecer local a seu módulo de origem e funções de acesso (funções *set* e *get*) devem ser implementadas;
3. Caso haja a necessidade de uma variável ser acessada por outros módulos e seu acesso pode ser feito de forma simultânea, três soluções foram propostas: Estrutura simplificada de exclusão mútua baseada em semáforo; Bufferização de variáveis; e Controle de variáveis por solicitação.

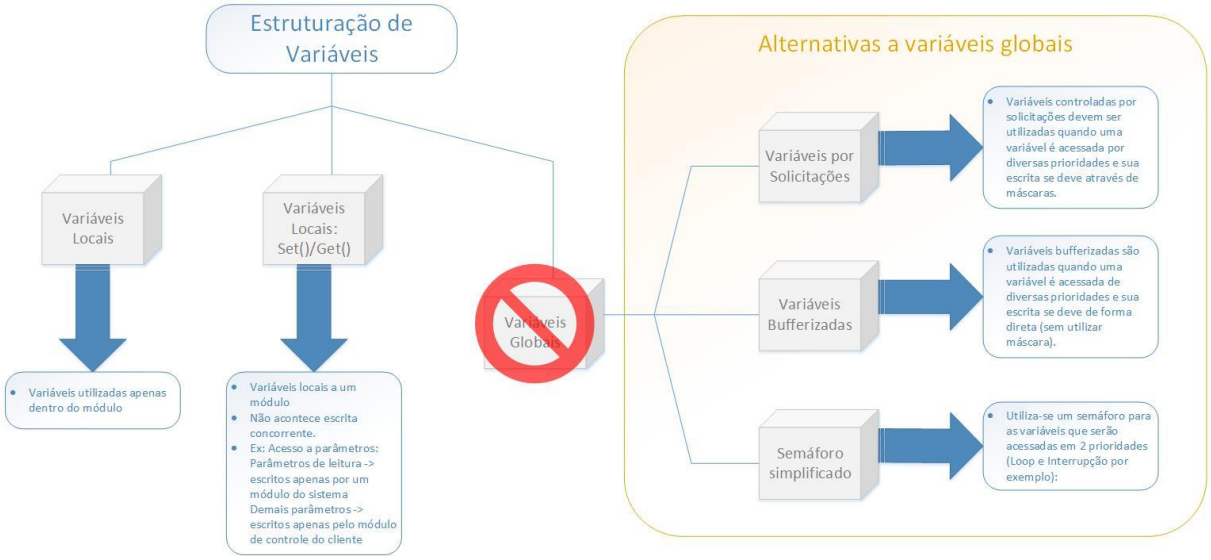

Figura 9 - Proposta de estruturação das variáveis do firmware da plataforma de cartões de controle

3.3.1: Estrutura simplificada de exclusão mútua baseada em semáforo

A estrutura de semáforo é muito utilizada por sistemas operacionais para controle de variáveis, porém esta plataforma não fará uso de um sistema operacional por requisito de engenheiros de controle do departamento. Utilizando os conceitos de semáforo, uma estrutura mais simplificada foi proposta para controle de variáveis que possuem acesso simultâneo por apenas 2 prioridades (Figura 10).

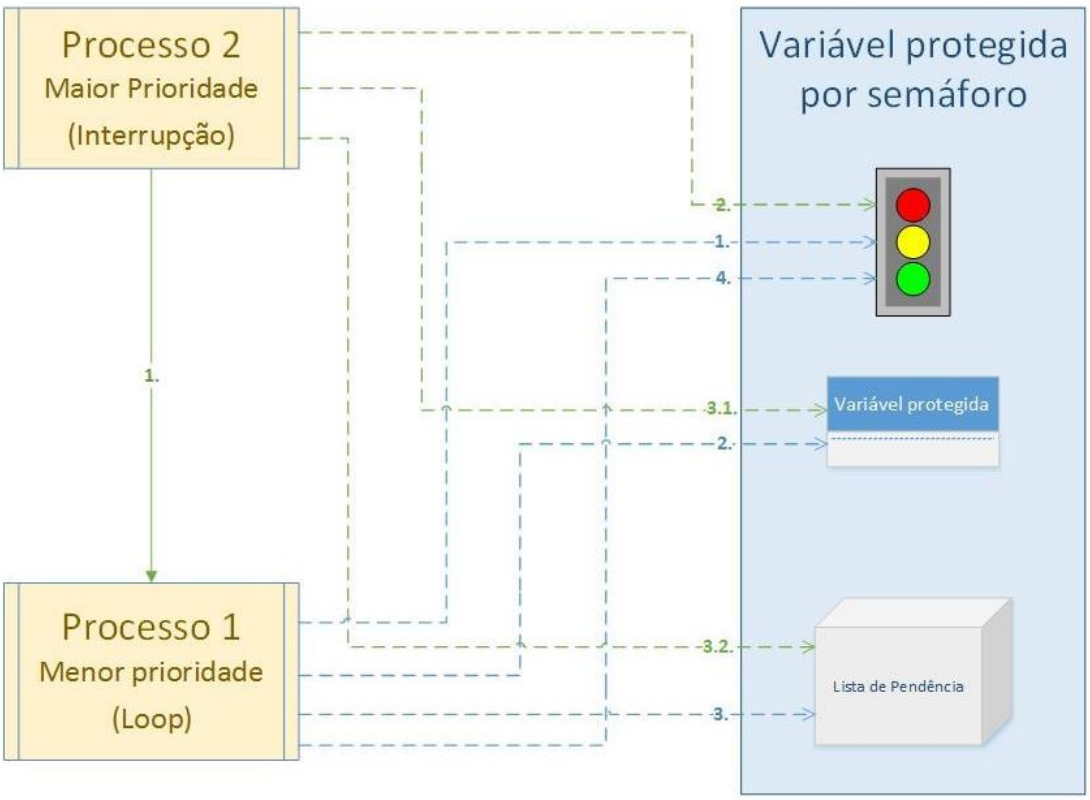

Figura 10 - Estrutura simplificada baseada em semáforos

Nesta estrutura temos uma variável que é acessada por apenas duas prioridades. Esta variável, independente de qual módulo a gerou, deve ser exportada para a estrutura de semáforo onde será feito o seu controle de acesso.

Nesta situação, o módulo de maior prioridade nunca vai ser interrompido pelo segundo módulo e se o acesso à variável estiver liberado, ele pode utilizá-la sem nenhum problema. Caso o acesso à variável não estiver liberado, ele deve deixar uma pendência de utilização para ser executada quando a variável for liberada.

Olhando pelo ponto de vista do módulo de menor prioridade ele pode ser interrompido pelo segundo módulo a qualquer instante, então quando desejar utilizar a variável ele deve travá-la antes de realizar seu acesso e liberá-la ao termino do mesmo. Antes de liberar o acesso a variável, a lista de pendencia deve ser executada para tratar eventuais acessos que tenham ocorrido enquanto a variável estava bloqueada. A figura 11 mostra o fluxograma de acesso de dois módulos de prioridades diferentes (neste caso um loop principal e uma interrupção).

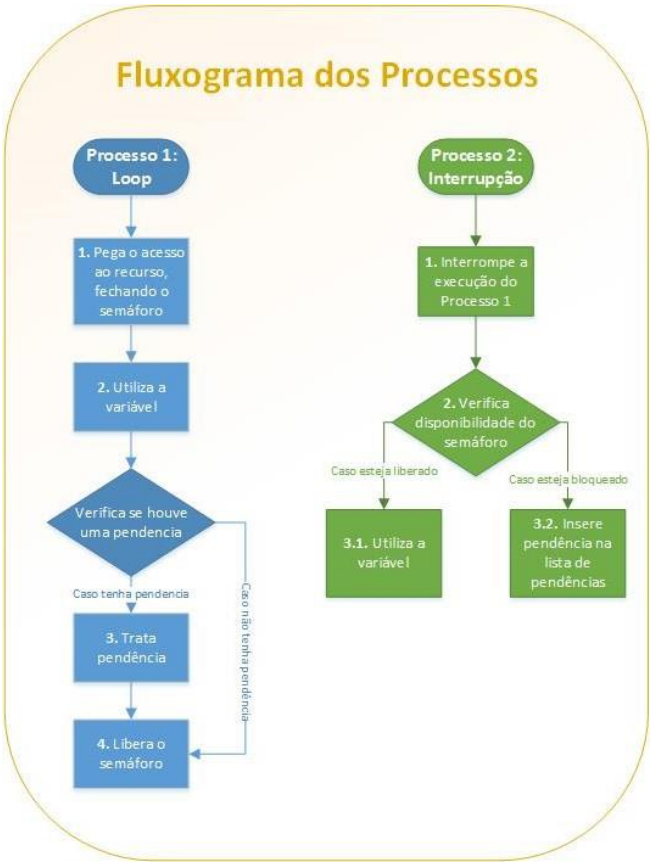

Figura 11 - Fluxograma de acesso de dois módulos a uma variável controlada por "semáforo"

3.3.2: Bufferização de variáveis

A segunda estrutura proposta para controle de acesso a variáveis é a bufferização de variáveis. Nesta estrutura o conjunto de variáveis controladas terão um buffer de entrada cujas alterações podem ocorrer sem que haja uma alteração no valor da variável.

Esta estrutura será utilizada quando vários processos querem alterar o valor de uma variável e esta variável é escrita sem o uso de máscaras, ou seja, estamos preocupados em ter apenas o valor mais recente salvo na variável. Um exemplo de uso é o caso do valor de um parâmetro, que pode ser escrito de diversas fontes (IHM, CLP, via serial, entre outras).

A figura 12 mostra a estrutura proposta. Nela, a variável controlada é escrita de diversos processos, mas a atualização de seu valor (contido no buffer) é realizado

apenas em um lugar. A leitura da variável controlada pode ser realizada de maneira direta, pois ela irá conter o valor real.

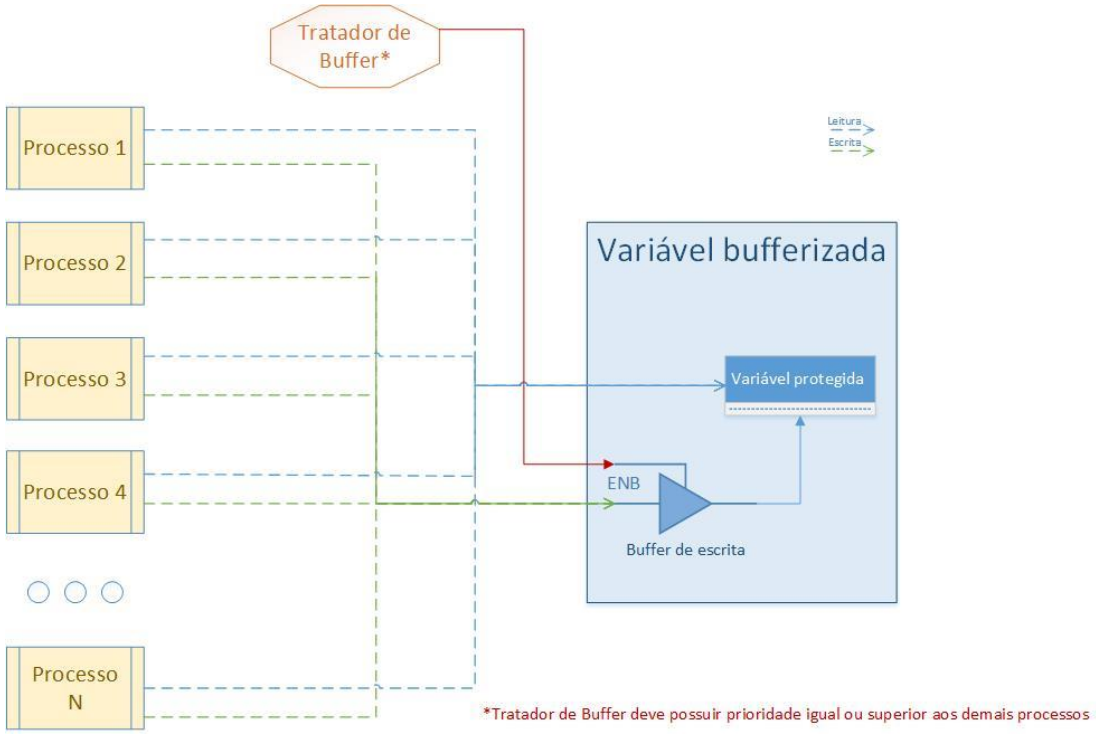

Figura 12 - Estrutura de variáveis bufferizadas

3.3.3: Controle de variáveis por solicitações

A última proposta para lidar com variáveis compartilhadas é a estrutura de controle por solicitações (Figura 13). Nessa estrutura, os processos que desejarem alterar uma variável devem realizar uma solicitação para o controlador, indicando qual variável querem alterar, qual a máscara deve ser aplicada e qual o valor a ser escrito.

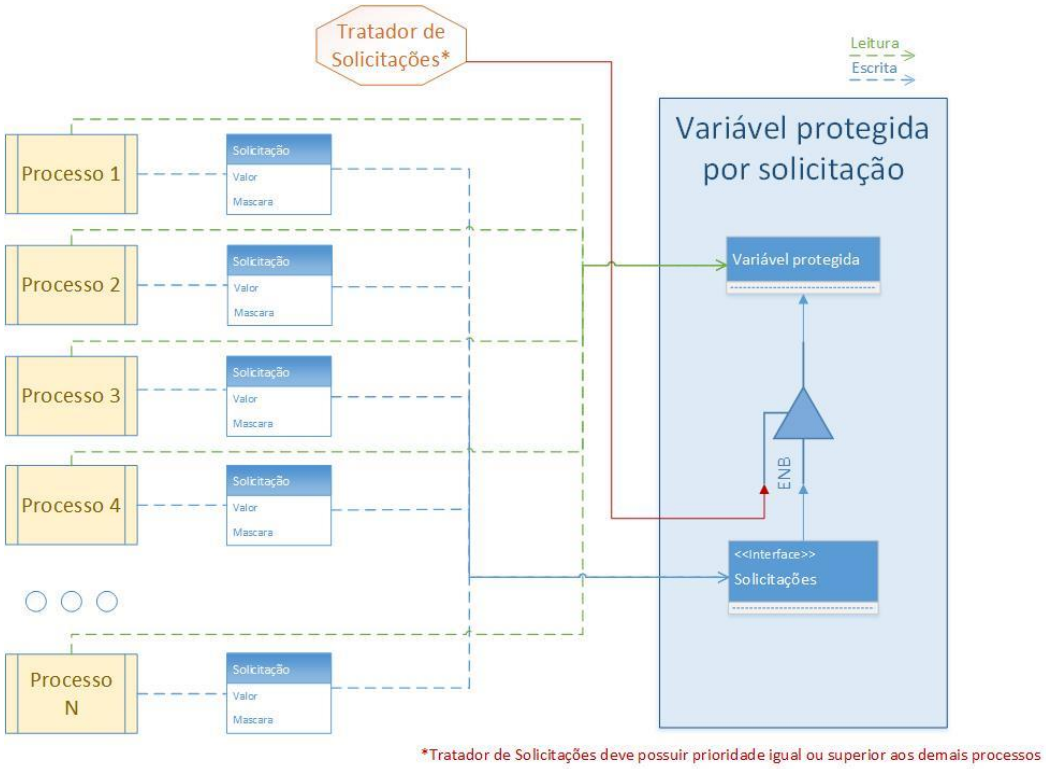

Figura 13 - Estrutura de variáveis controladas por solicitações

Essa estrutura deve ser utilizada quando uma variável compartilhada é acessada de diversas fontes para alterar parcelas de seus valores, ou seja, quando o valor já escrito nela é levado em conta e quando deve ser feito o uso de máscaras para realizar a alteração. Um exemplo bem cotidiano dessas variáveis é o acesso a portas de um microprocessador, onde cada bit de uma porta tem uma função e devem ser escritos de maneira independente.

A atualização da variável para o novo valor ocorre em um único lugar, assim como na estrutura bufferizada.

3.4: Distribuição de recursos

Outro problema encontrado em plataformas anteriores está na distribuição dos recursos do cartão. Estes recursos podem ser uma memória flash, uma saída serial, o acesso ao cartão SD, acesso as saídas digitais, entre outros recursos disponíveis no cartão.

Muitos destes recursos disponíveis no cartão são compartilhados por diversas tarefas e seu uso pode ser solicitado de forma concorrente. Visando estas situações, foi solicitado pela empresa que um mecanismo de compartilhamento de recursos fosse criado para vir a tratar da melhor maneira possível este uso concorrente. Após a realização de pesquisas e utilização de conhecimentos adquiridos no curso de controle e automação, foi proposto um escalonador não preemptivo a base de prioridade.

3.4.1: Escalonador não preemptivo por prioridade

O escalonador de recursos proposto é um escalonador não preemptivo por prioridade. Um algoritmo de escalonamento é considerado não preemptivo quando uma vez concedido o recurso a um processo, este só é liberado de forma voluntária pelo mesmo. Como já mencionado na seção 3.3.2, este processador não fará uso de um sistema operacional, o que restringe o uso de escalonadores para apenas escalonadores não preemptivos.

O escalonador por prioridade tem o seu funcionamento baseado nas prioridades instantâneas dos processos. Nele, o processo que possuir o maior recurso utilizará o recurso. Para que não haja problema de inanição (um processo nunca receber o recurso), as prioridades são alteradas a cada alocação de recurso da seguinte forma (Figura 14):

- O processo que recebeu a alocação do recurso tem a sua prioridade reduzida para a prioridade inicial;
- Os processos que não foram atendidos têm suas prioridades incrementadas em uma quantidade programada pelo escalonador.

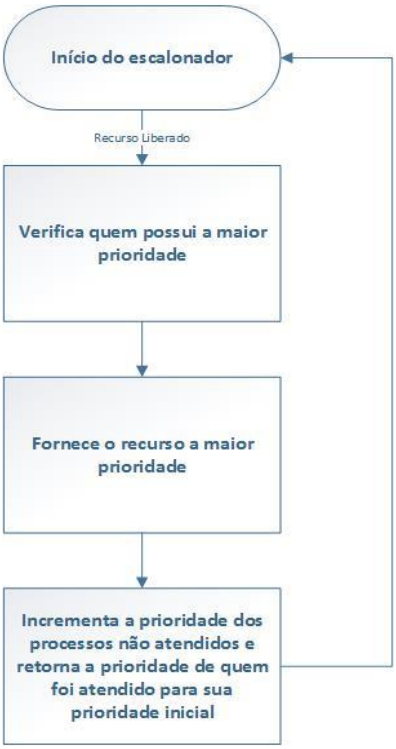

Figura 14 - Fluxograma do escalonador não preemptivo por prioridade

Um dos usos deste escalonador para a plataforma pode ser observado no controle do recurso de memória flash. Esta memória pode ser acessada por diversos processos como log de eventos, IHM, CLP, entre outros. A tabela 1 mostra um exemplo de configuração do escalonador para acesso a este recurso, onde o incremento da prioridade é realizado de 2 em 2.

Tabela 1 - Exemplo de uso do escalonador não preemptivo por prioridade

Sequência de atendimento	Prioridade IHM pré-atendimento	Prioridade CLP pré-atendimento	Prioridade Log pré-atendimento	Processo atendido	Prioridade IHM pós-atendimento	Prioridade CLP pós-atendimento	Prioridade Log pós-atendimento
1	15	12	10	**IHM**	15	14	12
2	15	14	12	**IHM**	15	16	14
3	15	16	14	**CLP**	17	12	16
4	17	12	16	**IHM**	15	14	18
5	15	14	18	**Log**	17	16	10

3.5: Conclusão do capítulo

Neste capítulo foi abordado a engenharia de software utilizada para estruturação do firmware da nova plataforma de cartões de controle. Os capítulos seguintes abordam os testes realizados para validação da plataforma, as implementações dos firmwares utilizados em cada teste, bem como os resultados obtidos.

Capítulo 4: Roteiro de testes de hardware

Este capítulo irá abordar os testes que serão realizados na plataforma de cartões de controle, os conceitos envolvidos e o objetivo de realizar cada teste. Os cronogramas de execução de cada teste, bem como seus objetivos, foram propostos em conjunto com os responsáveis pela plataforma de cartões de controle. Os testes serão apresentados na ordem de execução, tendo em vista que pode haver dependência de um módulo para outro. A implementação do firmware utilizado em cada teste se dará no capítulo seguinte.

4.1: Teste do barramento paralelo

O barramento paralelo, apresentado na seção 2.1.1.2 desta monografia, é o canal de comunicação mais utilizado pelo microprocessador para comunicar com seus periféricos. Um canal paralelo bem consolidado é essencial para aumentar a produtividade do projeto e aumentar a robustez do produto.

4.1.1: Cronograma de teste

O cronograma de teste deste canal consiste em seguir os seguintes passos:

- Verificar se não há curto e nem trilha rompida;
- Verificar se os sinais estão com níveis de tensão esperados;
- Implementar um módulo de testes para configurar o Bus State Controller (BSC) do processador e realizar testes de comportamento.

4.1.2: Objetivos do teste

O objetivo de realizar este teste é verificar a integridade dos sinais em todos os pontos do cartão através de leituras realizadas com osciloscópio e multímetro, e também verificar o comportamento dos sinais de controle (RD, WR, CS), dados e endereços para verificar compatibilidade realizar adequações de periféricos.

4.1.3: Conceitos envolvidos

O primeiro conceito envolvido nesse teste é o dimensionamento adequado das trilhas. Temos que a resistência elétrica de uma trilha é inversamente proporcional a sua largura, ou seja, quanto mais larga uma trilha, menor será sua resistência elétrica. Dependendo da corrente que passará nessa trilha, a queda de tensão ocasionada pode ser grande o suficiente para resultar em um mal funcionamento do barramento.

O segundo conceito envolvido é o de sinais de controle (CS, RD, WR). Quase todos os periféricos que se comunicam através de um barramento paralelo utilizam estes sinais de controle para verificar se a informação do barramento é destinada a ele e qual seria o comando: Escrita (WR) ou leitura (RD). Como cada processador atua de uma maneira, este teste é necessário para verificar o comportamento quando uma operação de leitura ou escrita é efetuada pelo microprocessador.

4.2: Teste de memórias

As memórias, apresentadas na seção 2.1.1.3, são responsáveis por armazenar o firmware do cartão e ser utilizadas para armazenamento rápido de informações durante execução. Para isso todas as memórias devem ser testadas para verificar sua integridade, tempo de acesso, comportamento, entre outros.

4.2.1: Cronograma de teste

O cronograma de teste das memórias consiste em realizar o seguinte:

- Análise do datasheet de cada memória para saber a configuração de barramento necessária para uso de cada periférico;
- Análise de hardware para verificar se a memória está em uma área de endereçamento (CS) adequada;
- Implementar um módulo que configure o barramento e realize leituras e escritas em cada memória.

4.2.2: Objetivos do teste

Este teste tem como objetivo verificar a compatibilidade da memória com o microprocessador, verificar se a memória localiza-se na área de endereçamento adequada e também verificar a integridade dos CIs.

4.2.3: Conceitos envolvidos

O primeiro conceito utilizado neste teste é o de tempo de acesso as memórias. Cada memória, dependendo do seu tipo, fabricante e modelo, possui um tempo de acesso específico para escrever, apagar ou ler um dado. Estes tempos devem ser levados em conta na hora de configurar o barramento.

Assim como os tempos de acesso, memórias do tipo SDRAM possuem outro comportamento que deve ser levado em conta: elas necessitam que seja feito refresh de tempos em tempos para que os capacitores não percam a carga. Para isso o processador possui uma ou mais áreas de endereçamentos (CS) específicos para uso da SDRAM, onde o periférico interno ao processador se responsabiliza de efetuar esses refresh.

4.3: Teste de I/Os

O cartão CCE-03 conta com saídas digitais, relés, saídas de fibra ótica, entre outros periféricos de I/O disponíveis para serem utilizadas pelo usuário da plataforma. Um rápido teste de funcionamento destes periféricos é necessário para validação do hardware.

4.3.1: Cronograma de teste

O cronograma de teste dos periféricos de I/O consiste em:

- Análise do datasheet de cada componente para verificar se há algum problema de projeto de hardware;
- Implementar módulos teste para realizar um simples teste de funcionamento de cada periférico.

4.3.2: Objetivos do teste

Este teste tem como objetivo verificar se todos os periféricos de I/O foram projetados de forma adequada e se todos os CIs estão funcionando como o esperado.

4.3.3: Conceitos envolvidos

Os módulos de teste de I/Os utilizam os conceitos apresentados na seção 4.1.3, pois muitos destes periféricos possuem acesso através de uma escrita ou leitura de um endereço. Outro conceito utilizado foi o de multiplexar áreas de CS através de expansores de I/O.

4.4: Teste da comunicação entre FPGA e MPU

A FPGA do ponto de vista do microprocessador funciona como uma memória. A comunicação do microprocessador com a FPGA acontece através de leituras e escritas realizadas pela MPU na área de CS destinada à FPGA. Para que isso se torne possível, um conjunto de registradores deve ser implementado dentro do firmware da FPGA.

4.4.1: Cronograma de teste

Para realizar o teste de comunicação entre MPU e FPGA, os seguintes passos foram seguidos:

- Implementação de um banco de registradores no firmware da FPGA;
- Configuração do barramento da MPU para adequar-se aos tempos impostos pelos registradores por parte da FPGA;
- Análise de banda do canal MPU-FPGA.

4.4.2: Objetivos do teste

Este teste tem como objetivos verificar a possibilidade de estabelecer uma comunicação entre FPGA e MPU através de leituras e escritas em registradores

implementados na FPGA e obter uma análise de banda total deste canal de comunicação através de testes de performance.

4.4.3: Conceitos envolvidos

O conceito principal envolvido nesse teste está na versatilidade de se programar uma FPGA para que o microprocessador possa enviar e receber dados dela de maneira a utilizar os recursos já disponíveis, que neste caso é o barramento paralelo.

A implementação do firmware da FPGA envolve conceitos de *clock* e DCM (Digital Clock Manager) para configuração do *clock* interno, conhecimentos em sistemas digitais como *flip-flops*, *mux*, e outras estruturas para criar um banco de registradores que funcionam como uma espécie de memória RAM, conceitos de propagação de sinais que podem influenciar quando se trabalha em altas frequências, entre outros conceitos que são aplicados no desenvolvimento de um hardware.

4.4.3.1: Clock e DCM

Uma das vantagens de se utilizar uma FPGA é a possibilidade de alterar o clock interno para um clock completamente programável. Um sinal de clock externo entra na FPGA através de um de seus pinos dedicados a esta função e pode ser modificado pelo DCM para gerar um clock programável pelo usuário.

O DCM (Digital Clock Manager) é um componente disponível em muitas FPGAs com a finalidade de manipular sinais de clock e evitar a geração de clock skew (variação da fase do clock). Os DCMs em geral possuem as seguintes funcionalidades:

- Multiplicam ou dividem um clock de entrada;
- Garantem que o clock gerado possua um *duty cycle* estável;
- Possibilidade de adicionar uma fase ao clock gerado;
- Eliminar clock skew resultante de uma implementação de um projeto.

4.4.3.2: Flip-Flops

Um flip-flop consiste de um circuito digital pulsado capaz de servir como uma memória de um bit. Estes circuitos possuem 4 topologias principais: flip-flop T, flip-flop S-R, flip-floop J-K e flop-flop D.

Os flip-flops do tipo T (“Toggle”) possuem um sinal de entrada, um sinal de clock e um sinal de saída. A saída deste flip-flop mudará seu estado toda vez que o sinal de entrada possuir um nível lógico alto e ocorrer um evento de borda no sinal de clock.

Os flip-flops do tipo S-R (“Set/Reset”), por sua vez, possuem 2 sinais de entrada, um sinal de clock e um sinal de saída. Para estes flip-flops, a saída assumirá um valor lógico alto se o sinal de entrada “Set” possuir nível lógico alto e o sinal de entrada “Reset” possuir nível lógico baixo e assumirá um valor lógico baixo se o sinal de entrada “Reset” possuir nível lógico alto e o sinal de entrada “Set” possuir nível lógico baixo. Caso ambos os sinais de entrada possuírem nível lógico alto, o comportamento deste flip-flop é imprevisível. Todas as trocas de estado acontecem apenas quando ocorre um evento de borda no sinal de clock.

Os flip-flops do tipo J-K aprimoram o funcionamento dos flip-flops do tipo S-R interpretando a condição onde as duas entradas possuem nível lógico alto. Neste caso, quando ocorrer de os dois sinais de entrada possuírem nível lógico alto junto de um evento de borda do clock, a saída deste flip-flop mudará seu estado.

Os flip-flops do tipo D (“Data”) possuem apenas uma entrada que é ligada diretamente a saída quando ocorre um evento de borda do clock, ou seja, sua saída assumirá o valor lógico da entrada.

4.4.3.3: MUX

O MUX ou multiplexador é um componente que possui diversas entradas e apenas uma saída. Este componente seleciona uma das entradas de acordo com seus canais de seleção e a transfere para saída. De forma resumida, os multiplexadores funcionam como uma chave de múltiplas posições controlada digitalmente.

Algumas das aplicações deste componente são:

- Roteamento de dados;
- Conversor paralelo-série;
- Sequenciamento de operações;

4.4.3.4: Propagação de sinais

"Timing" é um termo utilizado em circuitos digitais que referencia o tempo que um sinal leva para se propagar de um flip-flop, passando através de um circuito combinacional, e depois por outro flip-flop (Figura 15).

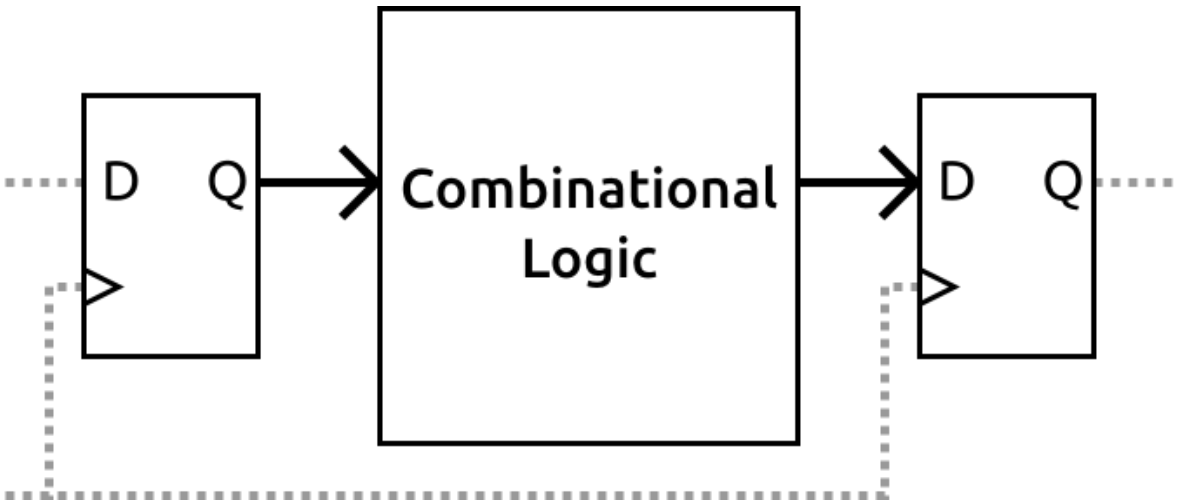

Figura 15 - Propagação de sinais em um circuito digital

O circuito combinacional apresentado na figura acima não é instantâneo. O sinal necessita de um tempo para que possa se propagar através dos MOSFETs que implementam a lógica combinacional. Além da lógica combinacional, os flip-flops também possuem um tempo de resposta (Figura 16).

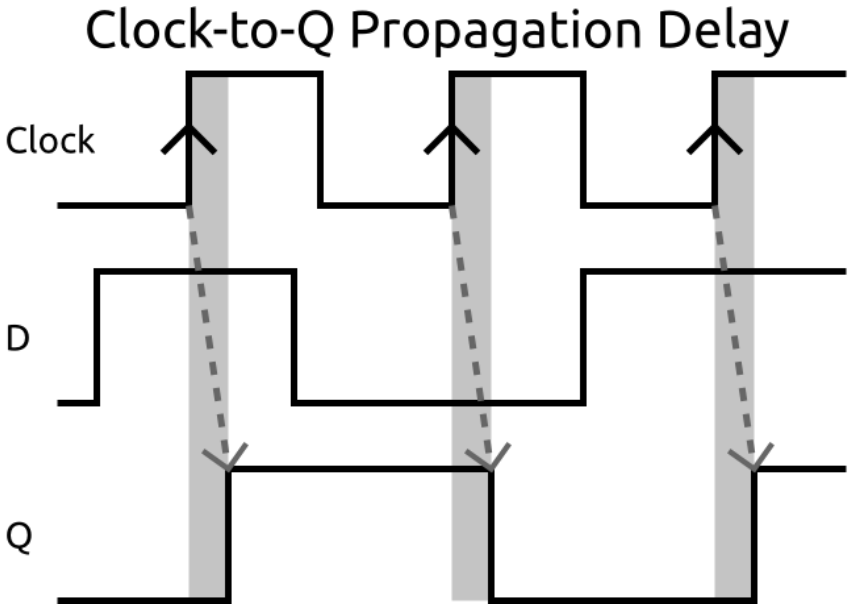

Figura 16 - Tempo de propagação do sinal em um flip-flop D

Para que estes flip-flops tenham um comportamento adequado, um tempo de setup deve ser respeitado e sua entrada deve permanecer constante durante um período pré e pós borda de clock.

A somatória do tempo de propagação do circuito combinacional, com o tempo de propagação do flip-flop e o tempo de setup vão limitar o período mínimo de clock na qual o sistema deva funcionar adequadamente.

4.5: Teste do canal serial diferencial

A comunicação entre cartão CCE-03 e o cartão iFOI é realizada através de canais seriais diferenciais na qual são transmitidos telegramas em alta frequência da FPGA presente no CCE-03 para a FPGA da iFOI. Este teste verificará a viabilidade de se utilizar este sistema de comunicação, bem como seus limitantes.

4.5.1: Cronograma de teste

Este teste requer os seguintes passos:

- Implementação de canais UART no firmware da FPGA do cartão CCE-03 que recebam dados vindos da MPU e transmitam para o cartão iFOI;

- Implementação de canais UART no firmware da FPGA do cartão iFOI para receber os dados recebidos e retransmiti-los para o cartão CCE-03 com objetivo de testar o canal nas duas vias;
- Análise de banda deste canal de comunicação.

4.5.2: Objetivos do teste

Este teste tem como objetivo verificar a viabilidade de uso deste canal de comunicação para os requisitos da plataforma através da análise de confiabilidade na transmissão dos dados e da análise de banda máxima que este meio é capaz de fornecer.

4.5.3: Conceitos envolvidos

O principal conceito envolvido nesse teste é o de comunicação UART. Além disso os conceitos apresentados na seção 4.4.3 sobre implementação do firmware da FPGA também foram utilizados, acrescentando-se a eles o conceito de máquinas de estado.

4.5.3.1: UART

As UARTs (Universal Asynchronous Receiver/Transmitter ou Receptor/Transmissor universal assíncrono) são sistemas que possuem como finalidade a transmissão e recepção de dados originalmente disponíveis na forma paralela [12] através de uma comunicação serial.

A transmissão de dados UART ocorrem em quadros (“frames”), onde os bits de dados são acrescidos de um bit de início de transmissão e um ou dois bits de fim de transmissão. Este quadro pode conter também um bit de paridade com intuito de validar os dados entre transmissor e receptor.

Além da estrutura de quadro, outro parâmetro importante é a taxa de transmissão (“baudrate”), que fixa o número de símbolos transmitidos por segundo. Como a transmissão serial é assíncrona, ambos receptor e transmissor devem ser

configurados de forma a possuírem taxas de transmissão bem próximas para evitar erros na recepção dos dados.

4.5.3.2: Máquinas de estado

O conceito de máquinas de estado, por mais que pareçam fugir do escopo dos conceitos utilizados na realização deste teste, são muito utilizados na hora da implementação do firmware presente na FPGA.

A teoria de autômatos é a base por traz do tradicional modelo computacional e é usado em vários propósitos que variam de projeto de circuitos, construção de compiladores, especificação e classificação de linguagens de programação, entre outros. Como autômatos são modelos matemáticos que produzem valores dependentes de um estado interno e alguns valores de entrada, eles são chamados de máquinas de estado [13].

Uma máquina de estado pode ter um número finito ou infinito de estados e possuir um comportamento determinístico ou não determinístico. Neste projeto foram utilizadas apenas máquinas de estado finitas com comportamento determinístico, que incluem a máquina de Mealy e a máquina de Moore.

A máquina de Mealy é uma máquina de estado finito que produz um resultado baseando-se no estado em que se encontra e na entrada de dados. A máquina de Moore por sua vez é uma máquina de estado finito que produz seu resultado baseando-se apenas no estado atual da máquina. O esquemático dos circuitos que implementam ambas as máquinas de estado podem ser visualizados nas figuras 17 e 18.

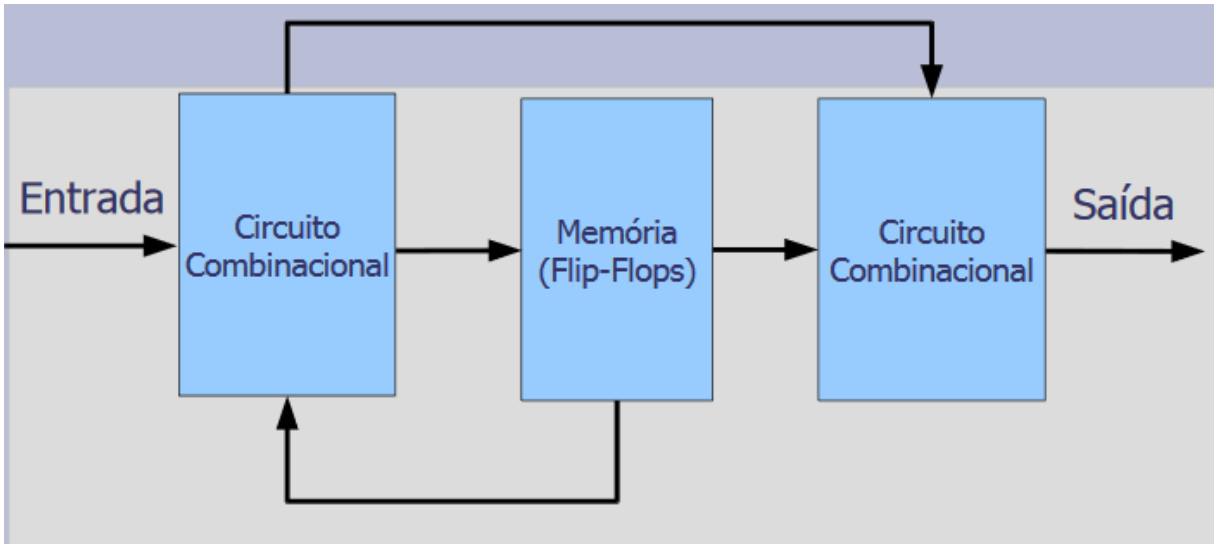

Figura 17 - Esquemático do circuito que implementa a máquina de Mealy

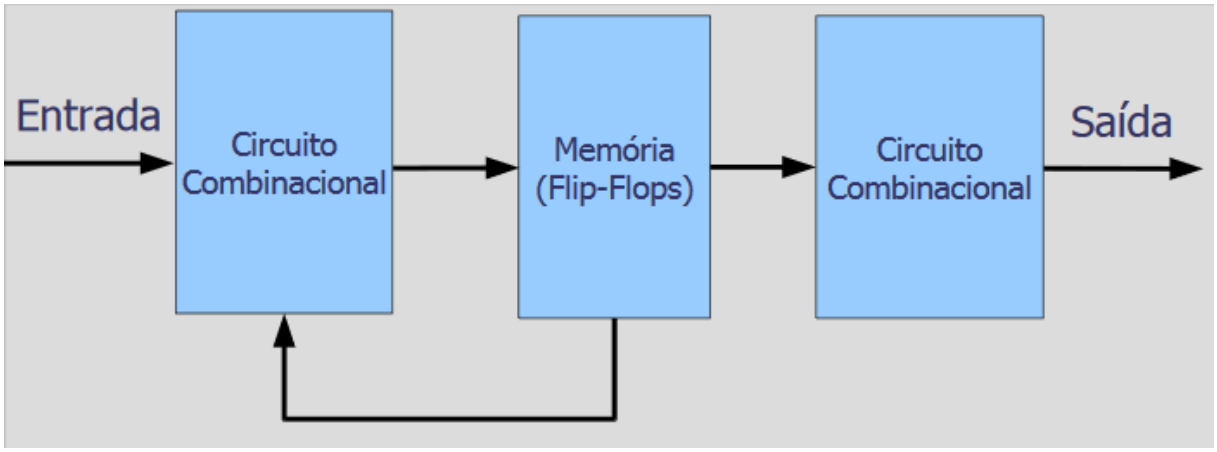

Figura 18 - Esquemático do circuito que implementa a máquina de Moore

4.6: Teste do canal serial – Comunicação com IHM

A comunicação entre o cartão CCE-03 e IHM (interface Homem-Máquina) é realizada através de um canal de comunicação serial padrão RS-485 utilizando o protocolo ModBus RTU para transmissão dos dados, que é bem consolidado na indústria e permite compatibilidade do cartão com diversas IHMs produzidas pela WEG. Este teste visa validação deste canal.

4.6.1: Cronograma de teste

Para teste desse canal é necessário realizar os seguintes passos:

- Verificar se o hardware não possui problemas construtivos (curto-circuito, níveis de tensões adequados);
- Configurar o canal serial da MPU;
- Implementar o protocolo ModBus RTU (escravo);
- Implementar a tabela de parâmetros que são lidos/escritos pela IHM.

4.6.2: Objetivos do teste

Este teste tem como objetivo implementar a comunicação serial feita através do protocolo ModBus RTU e validar a implementação utilizando IHMs da casa.

4.6.3: Conceitos envolvidos

O canal de comunicação serial RS-485 através do protocolo MobBus utiliza os conceitos apresentados na seção 4.5.3.1 sobre UART acrescidos do conceito de protocolo de comunicação, em específico o protocolo ModBus RTU.

Um protocolo de comunicação é um conjunto de normas que estão obrigadas a cumprir todas as máquinas e programas que interveem em uma comunicação de dados entre computadores sem os quais a comunicação seria caótica e, portanto, impossível.

4.6.3.1: Protocolo ModBus RTU

A rede ModBus RTU utiliza o sistema mestre-escravo para a troca de mensagens, permitindo até 247 escravos com apenas 1 mestre. Toda comunicação inicia com o mestre fazendo uma solicitação para um de seus escravos, e este responde ao mestre o que foi solicitado. Em ambos os telegramas (solicitação e resposta) a estrutura utilizada é a mesma [14]:

- Endereço: se refere ao endereço do escravo para qual se destina a mensagem

- Código da Função: especificação do tipo de serviço ou função solicitada ao escravo (leitura, escrita, etc.)
- Dados: Campo com tamanho variado onde o formato e conteúdo deste campo dependem da função utilizada e dos valores transmitidos.
- CRC: Checagem de erros de transmissão

No protocolo ModBus RTU não existe um caractere específico que indique o início ou fim de um telegrama. Essa indicação é feita pela ausência de transmissão de dados na rede, por um tempo mínimo de 3,5 vezes o tempo de transmissão de um byte de dados (11 bits – Start/Dados/2 Stop). Se um telegrama tenha sido iniciado após este período mínimo, os elementos da rede irão assumir que o primeiro caractere recebido representa o início de um novo telegrama (Figura 19).

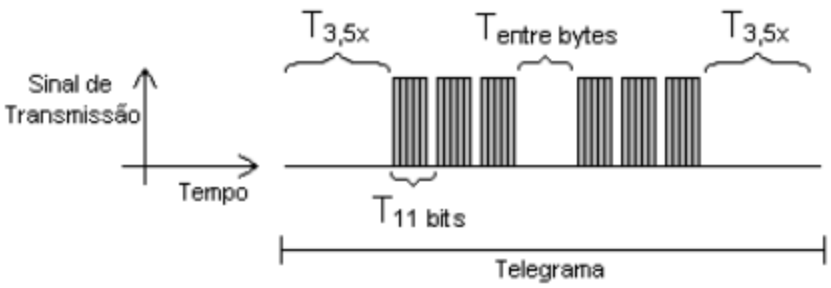

Figura 19 - Telegrama do protocolo ModBus RTU

4.7: Teste do A/D

Um componente muito utilizado no cartão CCE-03 para leitura de variáveis analógicas é o conversor A/D. Este teste visa verificar a integridade e comportamento do conversor A/D.

4.7.1: Cronograma de teste

Para realização desse teste apenas 2 passos são necessários:

- Análise do datasheet de cada componente para verificar se há algum problema de projeto de hardware;

- Implementação do módulo de teste do conversor A/D.

4.7.2: Objetivos do teste

Este teste tem como objetivo verificar se o sistema de conversão A/D foi projetado adequadamente e seu funcionamento ocorre como o esperado.

4.7.3: Conceitos envolvidos

O principal conceito envolvido nesse teste está no funcionamento de um conversor A/D por aproximação.

O conversor A/D transforma um sinal analógico, contínuo no tempo, em um sinal amostrado, discreto no tempo, quantizado dentro de um número finito de valores inteiros. O primeiro elemento na entrada de um conversor A/D realiza uma amostragem periódica do sinal analógico e o mantém estável até que o conversor propriamente dito possa transformar este valor em um número digital [15]. Este circuito amostrador é chamado de *Sample & Hold* (Figura 20).

Após a realização da amostragem é feito um processo chamado de quantização das amostras, onde o sinal é transformado em uma representação digital. Este processo depende de cada conversor A/D e os sinais de saída gerados podem estar na forma de números inteiros, complemento 2, entre outros.

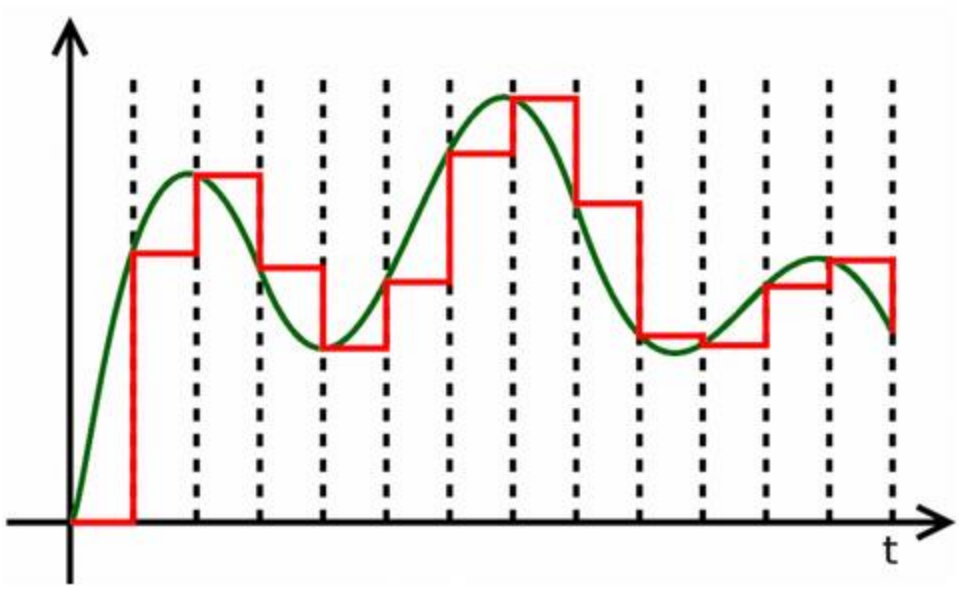

Figura 20 - Saída do circuito Sample & Hold

4.8: Teste do D/A

Assim como o conversor A/D, outro componente muito utilizado no cartão é o conversor D/A. O teste a ser realizado visa verificar a integridade e o comportamento do componente conversor.

4.8.1: Cronograma de teste

Apenas 2 passos são necessários para realizar este teste:

- Análise do datasheet de cada componente para verificar se há algum problema de projeto de hardware;
- Implementação do módulo de teste do conversor D/A.

4.8.2: Objetivos do teste

Assim como o teste realizado para o conversor A/D, este teste tem como objetivo verificar se o sistema de conversão D/A foi projetado adequadamente e seu funcionamento ocorre como o esperado.

4.8.3: Conceitos envolvidos

O principal conceito utilizado neste teste está no funcionamento do conversor D/A, que foi apresentado na seção 2.1.1.4.2. Deve se atentar para o tempo de conversão ao utilizar este componente.

4.9: Conclusão do capítulo

Neste capítulo foram apresentados os testes de hardware realizados durante este projeto. O capítulo seguinte apresentará como foi realizado a implementação dos módulos para cada teste que foi apresentado.

Capítulo 5: Implementação dos módulos

Neste capítulo serão abordadas as implementações realizadas para cada teste descrito no capítulo 4. Estas implementações foram desenvolvidas exclusivamente pelo autor desta monografia, tendo o auxílio de fornecedores e outros colaboradores da empresa apenas em situações atípicas.

Os módulos de teste feitos para o processador foram escritos na linguagem C e os módulos destinados as FPGAs foram escritos na linguagem VHDL.

5.1: Bus State Controller (BSC)

Este módulo foi implementado para realizar o teste do barramento e será utilizado em praticamente todo módulo subsequente. O módulo é bem simples e consiste na configuração dos registradores do BSC para cada área de CS e na execução de escritas e leituras em uma área desejada para teste de comportamento.

O BSC possui um registrador de uso geral comum a todas as áreas de CS, dois registradores para cada área (um registrador para definir seus aspectos básicos e outro registrador cuja função é variável e depende da configuração do primeiro) e registradores de acesso a SDRAM.

O registrador de uso geral é comum a todas as áreas de CS e é utilizado para configurar alguns aspectos dos pinos de comando como nível lógico do pino de fim de transmissão, nível lógico do pino de resposta, DMA e impedância dos pinos em diversos modos de operação.

O registrador que define os aspectos básicos de cada área é chamado de *CSn Space Bus Control Register* e tem a função de definir qual o tamanho do barramento de dados, qual o tempo de acesso entre leituras/escritas (em ciclos do barramento) e principalmente qual é o tipo de memória conectada a sua área de CS, que pode ser uma área normal (área de I/O por exemplo), memória ROM com burst, MPX-I/O, SRAM com seleção de byte ou SDRAM. Neste projeto o tamanho do barramento de dados é

o mesmo para todas as áreas de CS (16-bits) e as divisões de área dos CS ficaram da seguinte forma:

- CS0: Acesso a FPGA
- CS1: Memória NOR
- CS2: Acesso a periféricos I/O
- CS3: Memória SDRAM
- CS4: Memória RAM

O segundo registrador destinado para cada área é o registrador de tempos de espera. Nele são definidos os tempos que os sinais de comando, principalmente os comandos de escrita (WR), leitura (RD) e seleção da área (CS), ficarão ativos e os tempos entre cada sinal de comando. Este registrador varia de acordo com o tipo de memória selecionada anteriormente.

Os registradores destinados a acesso da SDRAM configuram todos os aspectos da SDRAM utilizada neste projeto, que vão de número de bits para endereçamento de linha e coluna à configuração do refresh.

Uma vez configurado os registradores do BSC, foi realizado uma sequência de leituras e escritas em diversas áreas de CS para verificar o comportamento do barramento.

5.2: Módulo de teste de memórias e I/Os

Os módulos para realizar testes nas memórias e nos periféricos de I/O são muito similares ao teste de barramento, pois tratam de escritas e leituras de uma certa área de memória que já foi configurada no módulo BSC.

5.2.1: Memórias

Para verificar a integridade das memórias, foi implementado um módulo de testes que realiza as seguintes operações para cada memória da plataforma:

1. Escrita de um bloco da memória (50 endereços) com o valor zero;

2. Leitura do mesmo bloco para validação;
3. Escrita no bloco de valores crescentes (valor 1 para o primeiro endereço, 2 para o segundo e assim sucessivamente);
4. Leitura do bloco para validação;

5.2.2: Periféricos de I/O

Este módulo foi implementado para realizar os testes nos periféricos de I/O, que no cartão CCE-03 consiste em saídas digitais, acionamento de relés e acionamento de LEDs. Para que fosse verificado a integridade de todos os periféricos de I/O, podendo realizar sua verificação de forma visual, foi implementado uma função de alternância dos valores das saídas dentro de um temporizador lento (0,5s).

5.3: Implementação do teste de comunicação entre MPU e FPGA

Este teste foi separado em dois módulos distintos: o módulo da MPU e o módulo de registradores da FPGA. O módulo da MPU consiste em executar escritas e leituras no barramento na área destinada ao FPGA (CS0), já o módulo da FPGA consiste em implementar um banco de registradores e um controlador de barramento dentro da FPGA, para que esta possa funcionar como uma memória do ponto de vista da MPU.

5.3.1: Módulo implementado na MPU

O módulo implementado pela MPU consiste em utilizar o módulo para teste de memória, seção 5.2.1, realizando as escritas e leituras na área de CS0.

5.3.2: Módulo implementado na FPGA

Como apresentado na introdução dessa seção, o módulo implementado na FPGA consiste em um banco de registradores e um controlador de barramento para que a MPU possa interpretar a FPGA como um acesso a uma memória. A figura 21 mostra um esquemático do módulo.

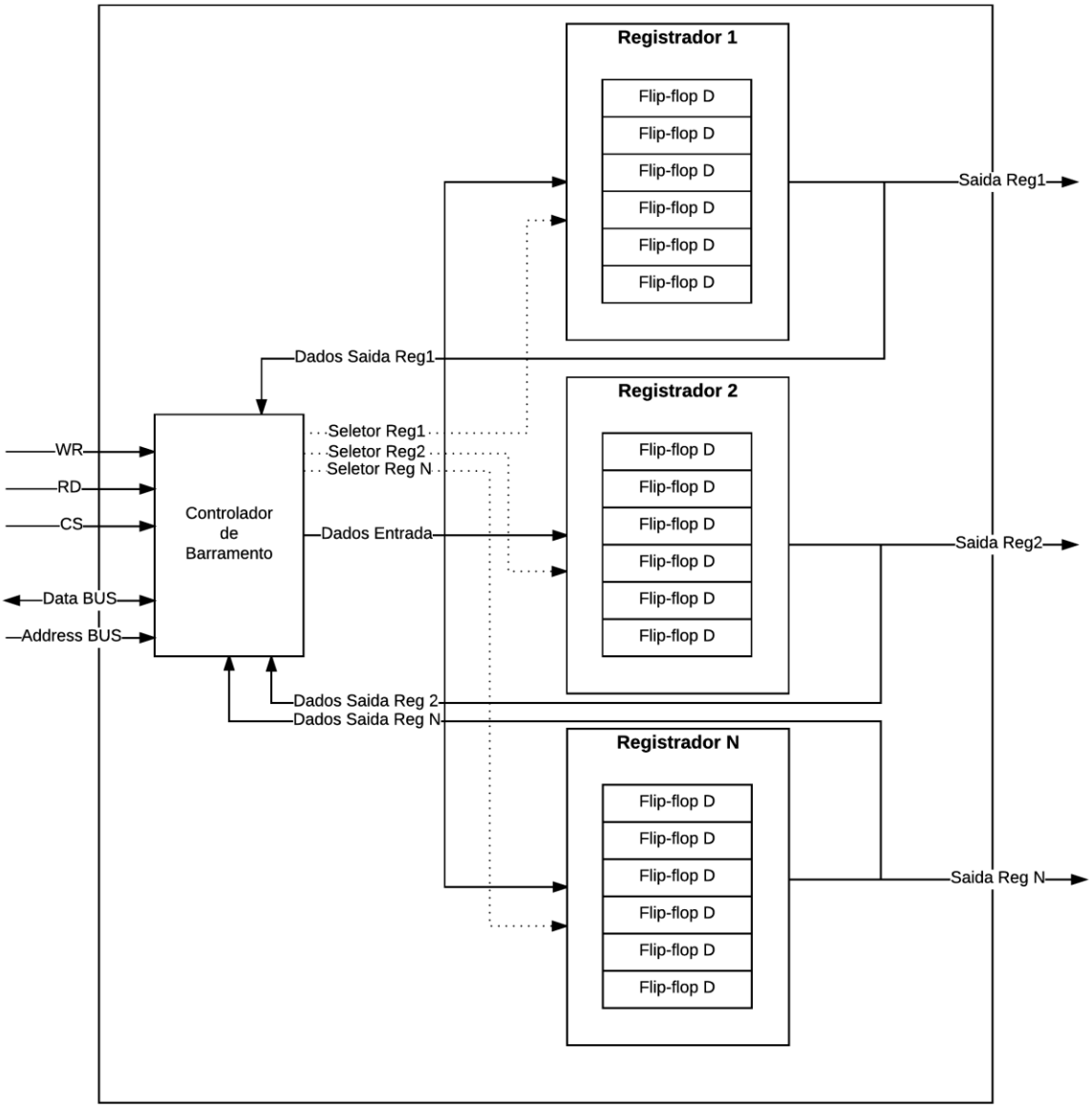

Figura 21 - Esquemático do módulo de comunicação entre FPGA e MPU

Os registradores deste módulo são compostos por um agrupamento, do tamanho do barramento de dados, de flip-flops do tipo D. A ideia por traz desses registradores é que o dado de entrada seja armazenado apenas quando o controlador de barramento, através de um comando seletor, selecione o registrador.

O controlador de barramento, por sua vez, recebe todos os sinais de comando (*CS, RD, WR*), dados e endereços vindos do barramento paralelo, realiza o tratamento desses sinais e executa a operação solicitada pela MPU (leitura/escrita). Quando a solicitação é uma operação de leitura, o controlador analisa qual endereço foi passado

pela MPU e retorna ao barramento de dados a informação armazenada no registrador referente aquele endereço. Na operação de escrita, o controlador verifica qual é o registrador que a MPU está tentando escrever através do barramento de endereços e seleciona este registrador para que os dados possam ser armazenados.

Um detalhe muito importante a respeito da implementação do controlador de barramento é que o barramento de dados deve funcionar em modo tri-state (alto, baixo ou de alta impedância) para que não ocorra interferência deste módulo no barramento paralelo.

5.4: Implementação do teste do canal serial diferencial

A implementação deste módulo de teste consiste no projeto de três módulos: um módulo serial (UART) que será utilizado em ambas as FPGAs; um módulo de controle para a FPGA do cartão CCE-03 com a função de extrair os dados de registradores específicos e enviar para a UART; e um módulo de controle para a FPGA do cartão iFOI com a função de receber os dados da UART e retransmiti-los para o cartão CCE-03. Este módulo de teste utilizará o módulo da seção 5.3 que implementa os registradores na FPGA.

5.4.1: Módulo UART

O módulo de comunicação serial UART é responsável por implementar os canais de comunicação TX (transmissor) e RX (Receptor). Para isso ele implementa uma serie de controladores que realizam a emissão/recepção byte a byte de um telegrama. Este módulo conta com dois submódulos: um módulo receptor e um módulo transmissor.

5.4.1.1: Módulo principal e seus controladores

O módulo principal é composto por dois buffers de dados, um controlador para recepção e um controlador para transmissão dos dados. A figura 22 mostra um esquemático deste módulo.

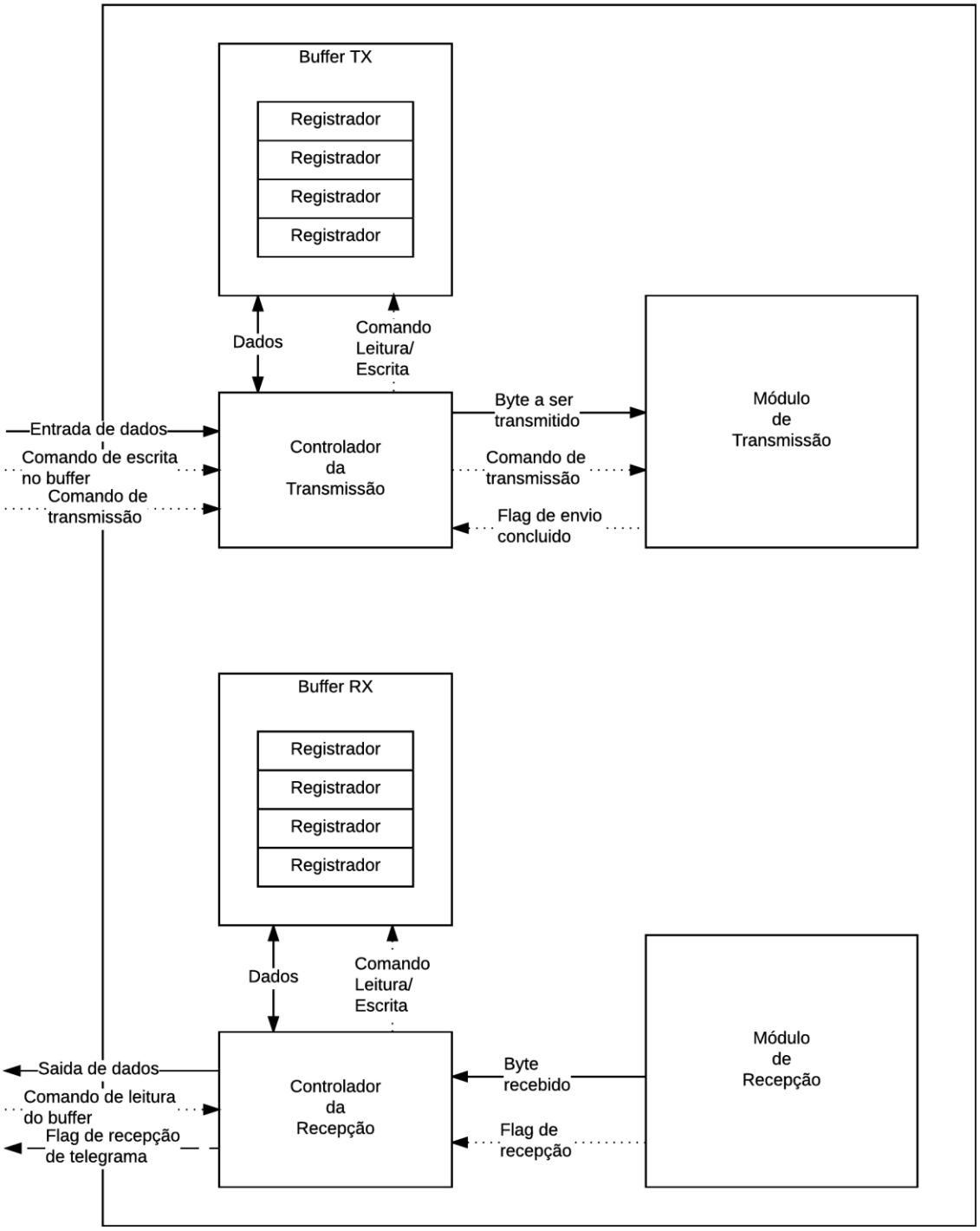

Figura 22 - Esquemático do módulo de comunicação UART

Os buffers são compostos por registradores e sua capacidade é programável em tempo de projeto para o número de bytes que serão enviados no telegrama. Para consumir menos recursos da FPGA, este conjunto de registradores foi programado de

tal forma que a FPGA alocasse um bloco RAM para sua construção, ao invés de flip-flops D, como mostrado anteriormente.

O controlador da recepção de dados da UART tem como função armazenar o byte recebido pelo módulo de recepção assim que receber um sinalizador do mesmo. Quando o telegrama for recebido de forma integral, este controlador sinalizará para seu exterior que o telegrama já se encontra no buffer e pode ser lido. A leitura acontece byte a byte e o dado é disponibilizado em sua saída após um comando de leitura.

O controlador de transmissão de dados da UART tem como função receber do exterior o telegrama byte a byte, armazenando-o em seu buffer, e enviá-lo ao receber um comando de transmissão. A escrita de dados no buffer é feita disponibilizando um byte em sua entrada de dados e enviando um comando de escrita. Quando todos os bytes do telegrama foram colocados no buffer, um sinal de transmissão é necessário para que o controlador inicie o processo de transmissão. A transmissão é realizada enviando um byte para o módulo de transmissão seguido de um comando para transmitir. O envio do byte subsequente apenas é realizado quando um sinalizador de fim de transmissão é emitido pelo submódulo.

5.4.1.2: Módulo de transmissão

O módulo de transmissão é responsável por realizar a transmissão de um byte pelo canal. Ele é composto por um controlador responsável por enviar os bits na sequência correta e um gerador de baud rate que irá gerar os pulsos na taxa de transmissão selecionada pelo usuário (Figura 23).

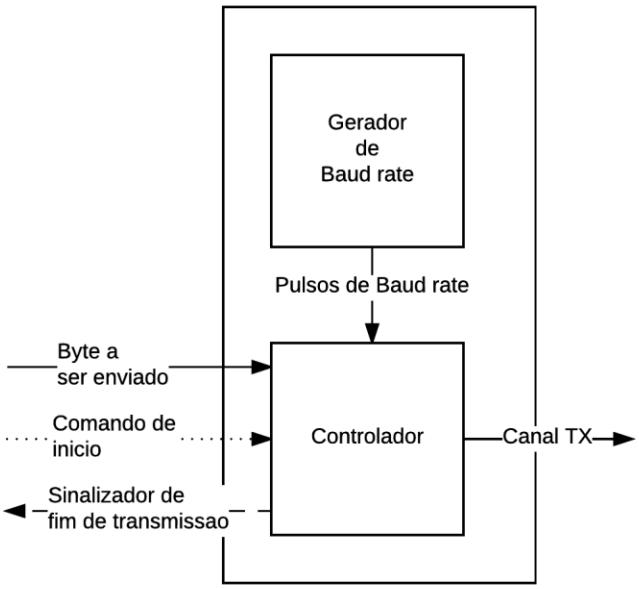

Figura 23 - Esquemático do módulo de transmissão

O gerador de baud rate presente neste módulo utiliza o clock interno do sistema para gerar pulsos na frequência de transmissão selecionada pelo usuário. Estes pulsos têm como função orientar o controlador de quando enviar o próximo bit.

O controlador deste módulo pode ser visto como um *shift register.* Ele é responsável por adicionar os bits de início e fim de transmissão junto ao byte recebido pelo módulo UART. Uma vez adicionado, ele envia o primeiro bit (bit de início) para o canal de saída (TX) quando recebe um pulso do gerador. Ao enviar o primeiro bit, ele rotacional o buffer interno para que o segundo bit fique disponível para ser enviado no próximo pulso do gerador. Este processo é repetido até todos os bits serem enviados. Um sinal de fim de transmissão é enviado para o módulo UART assim que o envio do dado é concluído.

5.4.1.3: Módulo de recepção

O módulo de recepção é responsável por captar um byte que foi transmitido pelo canal. Assim como o módulo de transmissão, ele é composto por um gerador de pulsos com frequência maior que o baud rate, para realizar o *oversampling* do sinal, e

um controlador responsável por armazenar os bits recebidos do canal em um buffer intermediário (Figura 24).

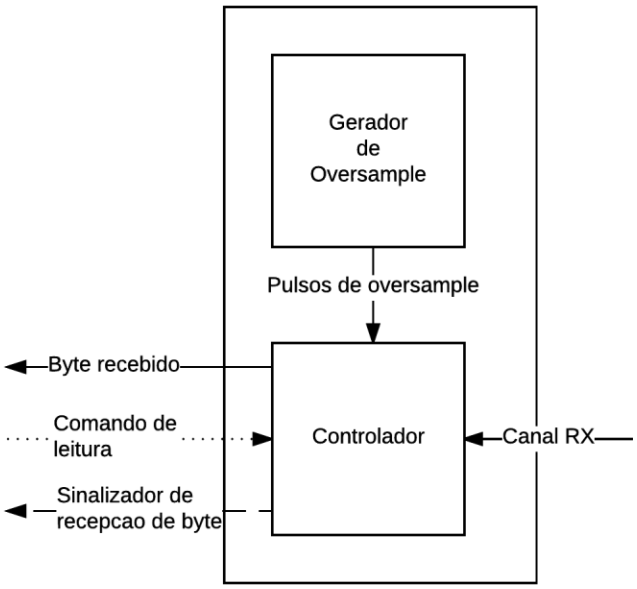

Figura 24 - Esquemático do módulo de Recepção

O gerador de *oversample* que compõe este módulo utiliza o clock interno do sistema para gerar pulsos com frequência superior ao baud rate da transmissão. Esta frequência geralmente é da ordem de 8 a 16 vezes o baud rate. Os pulsos gerados serão utilizados pelo controlador para realizar a leitura dos bits recebidos pelo canal.

O controlador presente no receptor, assim como no módulo de transmissão, pode ser visto como um *shift register*. Ele utiliza os pulsos recebidos do gerador para realizar a amostragem do bit bem no meio da janela de transmissão e colocá-lo no buffer intermediário. A recepção se inicia quando o controlador detecta que o canal foi para nível lógico baixo, indicando o bit de início de uma transmissão. A partir deste momento uma contagem é realizada para amostrar o próximo bit bem no meio da janela de transmissão. Este bit é amostrado, armazenado no buffer intermediário e o buffer é rotacionado à espera do próximo bit. Este processo se repete até todos os bits serem armazenados. Quando é concluído a recepção, o controlador sinaliza para o módulo UART que um byte foi recebido com sucesso e pode ser lido.

5.4.2: Módulo de controle presente no cartão CCE-03

O controlador presente no cartão CCE-03 para a comunicação serial tem como função verificar se a MPU escreveu um telegrama novo nos registradores para serem enviados, realizar a leitura deste telegrama nos registradores e enviá-los para o módulo UART e realizar a escrita nos registradores de telegramas recebidos pelo módulo UART. O modulo de registradores é o mesmo que foi implementado para o teste entre MPU e FPGA.

A verificação realizada para identificar se um telegrama novo foi escrito pela MPU é feita através do módulo de registradores. Esta funcionalidade foi acrescentada ao módulo 5.3.2, implementando um registrador específico que irá disparar um sinalizador caso a MPU escreva um valor de comando para iniciar uma transmissão.

A transmissão é iniciada pelo controlador ao receber um sinalizador vindo do módulo de registradores para esta finalidade. Ao receber este sinalizador, o controlador realiza a leitura do telegrama nos registradores e escreve no buffer do módulo UART byte por byte do telegrama. Ao terminar a escrita do telegrama no buffer do módulo UART, o controlador envia um comando de início de transmissão para que o módulo UART possa iniciar a transmissão do telegrama pelo canal.

A recepção de um telegrama é iniciada pelo controlador ao receber um sinalizador vindo do módulo UART, indicando que um telegrama está pronto em seu buffer. Ao receber este sinalizador, o controlador realiza a leitura do telegrama e realiza a escrita do mesmo nos registradores para que possam ser acessados pela MPU.

5.4.3: Módulo de controle presente no cartão iFOI

O módulo projetado para o cartão iFOI é bem similar ao módulo apresentado anteriormente para o cartão CCE-03. A funcionalidade do controlador é a mesma, realizar a escrita do telegrama, presente nos registradores, no módulo UART quando receber um sinalizador de início de transmissão e realizar o armazenamento do telegrama nos registradores quando receber um sinalizador vindo do módulo UART.

A diferença entre os dois módulos está no fato de que o sinalizador emitido para a transmissão de um telegrama vem do próprio controlador ao armazenar um

telegrama recebido. Isto ocorre, pois, para realizar este teste de comunicação, a função do cartão iFOI é receber um telegrama vindo do cartão CCE-03 e retransmiti-lo de volta. De uma forma geral, o controlador recebe um telegrama vindo da UART, armazena-o em um conjunto de registradores, emite um sinalizador para início de uma transmissão, realiza a leitura dos mesmos registradores e os envia para o módulo UART.

5.5: Implementação do teste de comunicação com as IHMs

A implementação deste teste consiste em projetar três módulos na MPU: uma tabela de parâmetros utilizada pela IHM para leitura e escrita; o protocolo ModBus; e um módulo serial responsável por transmitir/receber os dados via RS-485.

5.5.1: Tabela de parâmetros

A tabela de parâmetros consiste em um módulo que implementa uma tabela de parâmetros na memória do CCE-03 que espelha a tabela presente nas IHMs WEG, em particular uma IHM gráfica utilizada neste teste.

5.5.2: Módulo serial

Este módulo serial, diferente do implementado na FPGA, consiste apenas em configurar e utilizar o periférico existente na MPU. As configurações utilizadas pelas IHMs da casa utilizam as seguintes configurações:

- Baud rate de 38000bps;
- 2 Stop bits;
- Não possuem paridade.

5.5.3: Protocolo ModBus RTU

Este módulo tem como função implementar o protocolo ModBus RTU, apresentado na seção 4.6.3.1, trabalhando na função de escravo. Para isso, foi utilizado a seguinte máquina de estados (Figura 25):

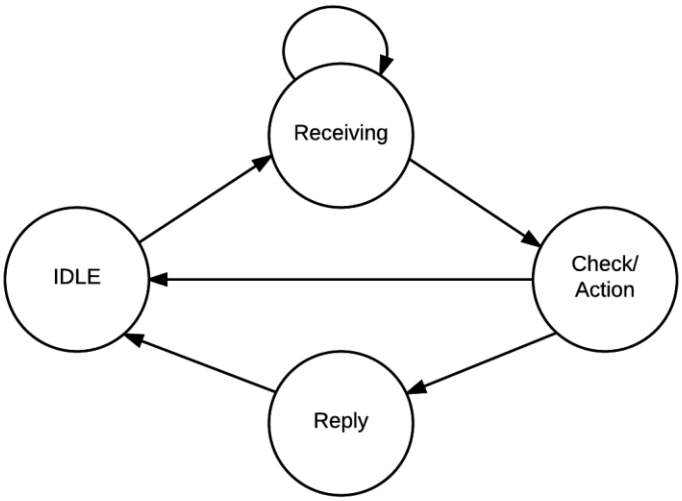

Figura 25 – Máquina de estados que implementa o protocolo ModBus RTU

- IDLE: É o estado inicial da máquina. Neste estado aguarda-se o recebimento do telegrama. Quando o primeiro byte do telegrama é recebido, a máquina de estados muda para o estado *Receiving.*
- Receiving: Neste estado realiza-se a recepção dos bytes do telegrama, armazenando-os na memória. Quando o tempo entre bytes ultrapassa o tempo especificado pelo protocolo, assume-se que o telegrama se encerrou e a máquina muda para o estado *Check/Action*.
- Check/Action: A primeira etapa realizada neste estado é a verificação do CRC, que analisa se ocorreram falhas na recepção do telegrama. Após concluída a verificação, a ação de escrita/leitura solicitada pelo telegrama é realizada (apenas se o telegrama for destinado a este cartão e não ocorreram falhas na recepção). Em seguida, a máquina transita para o estado *IDLE* se o telegrama não era destinado a este dispositivo ou para o estado *Reply* caso contrário.
- Reply: Neste estado é formulado o telegrama de resposta. Caso tenha ocorrido um erro de recepção de telegrama, a resposta indicará o erro na transmissão. Caso contrário, a resposta conterá o que foi solicitado

pelo telegrama inicial. Esta resposta é então transmitida para o mestre e a máquina de estados volta a seu estado inicial.

Para que essa máquina de estados possa ser executada de forma correta, uma função de temporizador foi programada para ser a base de execução da máquina de estados, em outras palavras, está máquina ira ser executada periodicamente em uma frequência adequada para controlar os timeouts envolvidos no protocolo.

5.6: Implementação do teste de A/D e D/A

Os componentes A/D e D/A, do ponto de vista da MPU, consistem apenas em realizar escritas e leituras de endereços específicos e utilizará os mesmos conceitos de implementação mostrados nas seções 5.1 e 5.2.

Para realizar este teste, uma função “dente de serra”, gerada utilizando um temporizador, foi escrito no D/A para ser analisada externamente. A saída do D/A, foi então conectada a entrada do A/D, e a leitura do A/D foi utilizada para comparação dos sinais, validando ambos os componentes.

5.7: Conclusão do capítulo

Este capítulo mostrou as implementações feitas para realizar os testes apresentados no capítulo 4. O próximo capítulo apresentará os resultados obtidos dos testes e algumas discussões sobre a respeito de cada resultado.

Capítulo 6: Resultados e discussões

Neste capítulo serão apresentados os resultados obtidos com cada teste apresentado e implementado nos capítulos 4 e 5. Os resultados foram analisados junto aos responsáveis pela plataforma de cartões de controle para que a validação da plataforma pudesse ser concluída. Também será apresentado neste capítulo uma breve discussão a respeito de cada resultado.

Em grande parte dos testes realizados, a medição para captura dos resultados foi realizada utilizando um osciloscópio da marca Tektronix de 4 canais ou um multímetro simples.

6.1: Bus State Controller

O teste do barramento paralelo teve como principal objetivo verificar o comportamento dos sinais de comando em operações de escrita e leitura. Os primeiros passos de verificação de hardware, que consistem em verificar curtos e níveis de tensões, não constataram nenhuma falha de projeto. O segundo passo foi analisar o datasheet do processador para implementar o módulo de teste, seção 5.1, e assim verificar o comportamento do barramento. Nesta etapa foi constatado que a área de CS alocada no projeto do cartão não é compatível com o uso da SDRAM por motivos construtivos do microprocessador. A SDRAM, diferente do apresentado na seção 5.1, estava localizada no CS2 que era uma área não permitida para este tipo de memória. A solução encontrada foi colocar a SDRAM no CS3, cuja área é preparada para utilização deste tipo de memória, e os periféricos de I/O, que estavam no CS3 e poderiam ir para qualquer área, passaram para o CS2.

Após implementado o programa de teste, foi feito a captura dos sinais de comando CS, RD e WR para identificar o comportamento deste barramento. A operação de escrita (Figura 26) aconteceu como esperado, tendo o CS sendo ativado antes do comando de escrita (WR). A operação de leitura (Figura 27), por sua vez, teve

um comportamento diferente do que era esperado para a configuração feita para este teste.

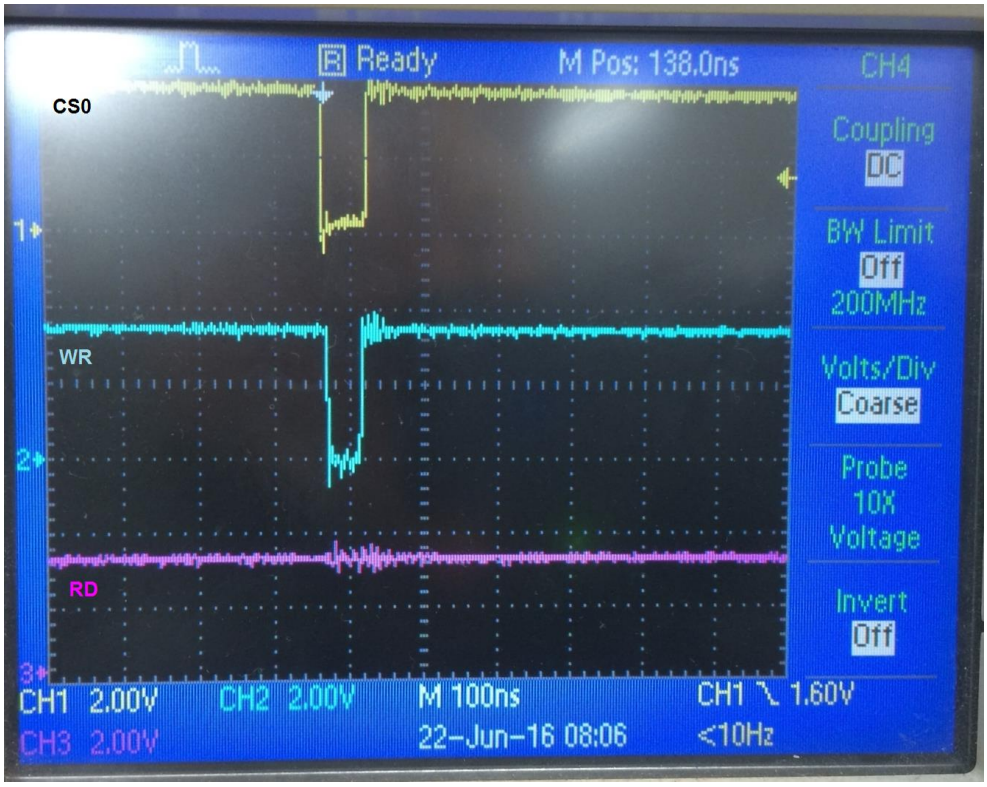

Figura 26 - Operação de escrita do barramento paralelo

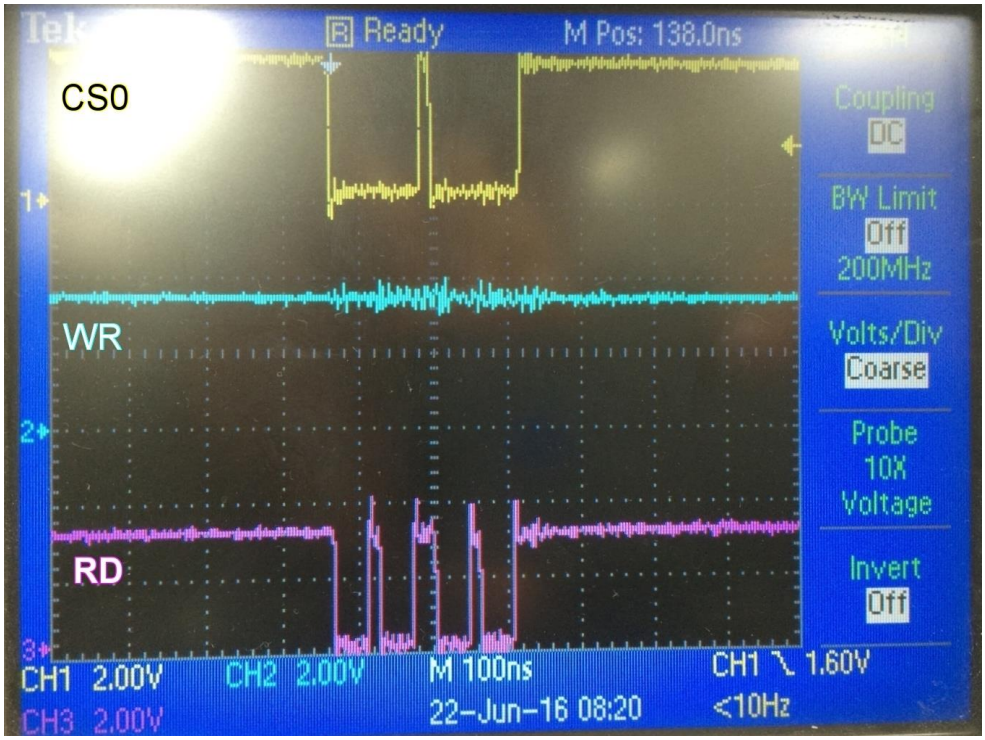

Figura 27 - Operação de leitura do barramento paralelo

Ao realizar leituras para variáveis de 16, 32 ou 64 bits, o barramento sempre opera como se a leitura fosse realizada para uma variável de 64-bits, ou seja, tendo um barramento de dados de 16-bits, o processador irá realizar 4 operações de leitura de endereços consecutivos. A partir do material de apoio do fabricante não foi possível determinar uma resposta conclusiva a respeito deste comportamento. Entende-se que este comportamento ocorre devido a configuração do barramento interno do processador (AMBA) que controla o barramento externo. Tal configuração é realizada pelo fabricante em tempo de projeto e não pode ser alterada em tempo de execução. A configuração de registradores que levam a este comportamento ainda está pendente de verificação pelo fabricante da MPU.

6.2: Teste das memórias

Este teste teve como principal objetivo verificar a integridade das memórias presente no cartão CCE-03. O teste levou em consideração os problemas relatados na seção anterior em relação as operações de leitura na implementação do teste mostrado em 5.2. Os resultados foram positivos e todas as memórias foram testadas com êxito.

6.3: Comunicação entre MPU e FPGA

A primeira etapa deste teste foi a implementação do banco de registradores por parte da FPGA. Esta etapa acabou sendo um pouco trabalhosa devido a frequência do clock interno da FPGA que estávamos trabalhando. Inicialmente, com a utilização de um DCM, foi possível colocar o clock interno em 320MHz para que o acesso feito pela MPU precisasse de apenas um ciclo de espera para ser concluído.

Um ciclo de espera significa que, na operação de escrita, o barramento mantém o dado por um ciclo a mais depois de o dado ser disponibilizado. Já na operação de leitura, o barramento insere um ciclo entre disponibilizar o endereço a ser lido e fazer a leitura do dado do barramento.

Esta configuração por parte da FPGA resultou em problemas de “timing”, explicado na seção 4.4.3.4, quando vários registradores foram criados no módulo. Estes problemas estavam na distância da trilha interna que a FPGA sintetizou para alguns registradores (problema que é apontado pelo sintetizador ao realizar o roteamento das trilhas internas), o que ocasiona atraso em alguns sinais e torna o comportamento do componente sintetizado imprevisível. A solução encontrada para este problema foi diminuir o clock interno do sistema do limite superior para uma banda especificada de 160MHz para que estes atrasos passassem a não interferir no comportamento dos registradores. Isto levou a aumentar o tempo de espera do barramento de um ciclo para dois ciclos.

Após a implementação dos registradores, foi realizado uma análise da banda total do canal entre MPU e FPGA. Com a configuração acima de dois ciclos de espera, além de 1.5 ciclos entre o CS e os comandos de leitura e escrita, o tempo total de acesso em uma operação de escrita realizada pela MPU aos registradores da FPGA ficou em 75ns (Figura 28). Considerando uma especificação de projeto, o tempo disponível para acesso da MPU à FPGA é de 10% do tempo total do processador. Com isso temos que a banda total do canal em termos de operações de escrita é de 1,33Mwords/s, ou seja, 1,33 milhões de palavras de 16-bits por segundo.

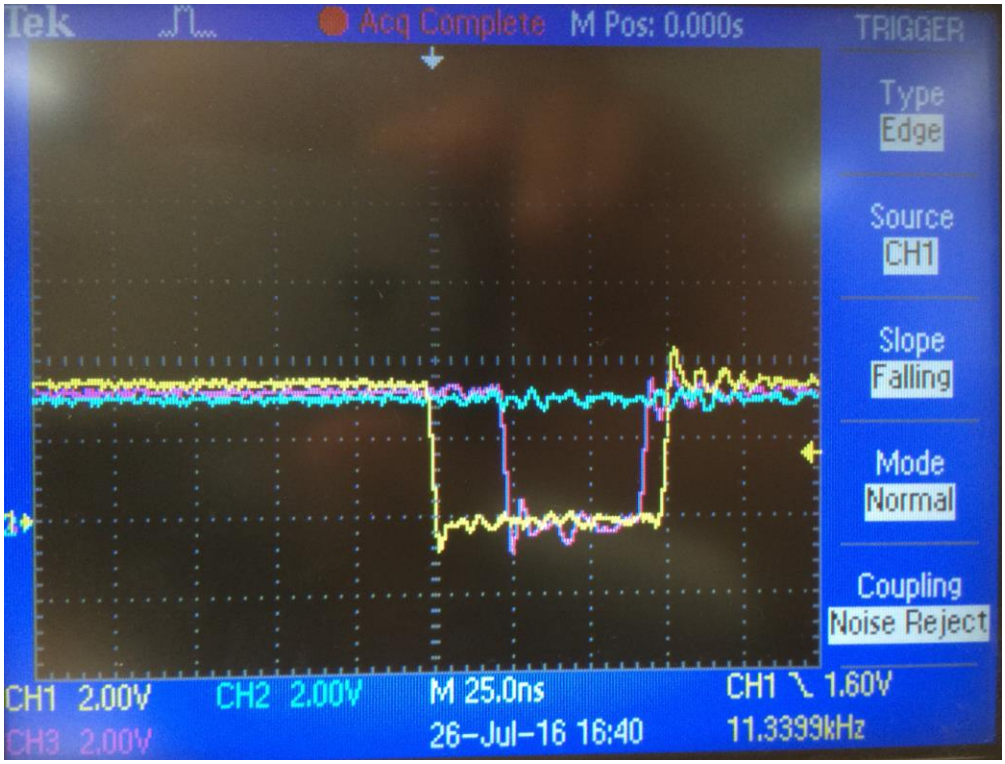

Figura 28 - Operação de escrita de um registrador da FPGA

Para cálculo desta banda foi utilizado apenas a operação de escrita devido ao comportamento do barramento em operações de leitura, porém o tempo de acesso de uma leitura (64-bits) pode ser visualizado na figura 29, que resultaria em uma banda de 1.25Mwords/s.

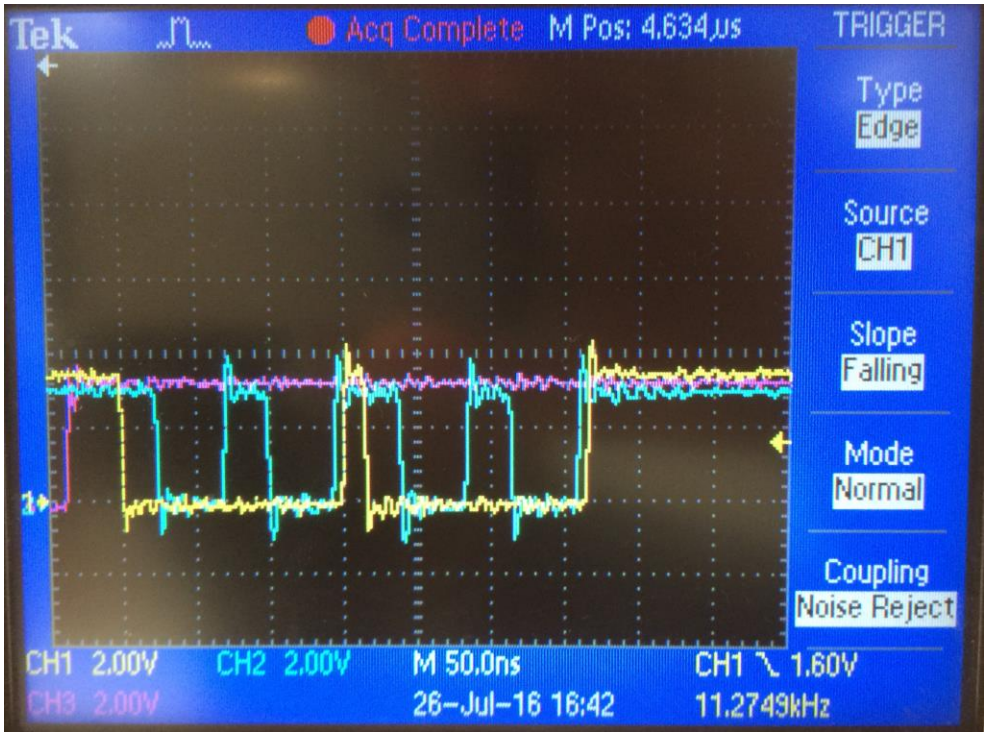

Figura 29 - Operação de leitura de 4 registradores consecutivos da FPGA (1 operação de leitura)

6.4: Canal serial diferencial entre CCE-03 e iFOI

Este teste vem a validar um conceito novo desta plataforma que é a utilização da comunicação serial diferencial para transmitir dados entre os cartões em altas frequências. A primeira implementação realizada foi a do módulo UART que é um módulo padrão e possui várias fontes de auxílio disponíveis na internet. O único problema encontrado ao realizar esta implementação estava na captura do canal RX, possivelmente por oscilações do canal. A solução encontrada foi “bufferizar” este canal de entrada, através de um flip-flop D, antes de colocá-lo no sistema. Isto introduziu um atraso de um ciclo de clock que foi compensado posteriormente.

A implementação do módulo completo de ambos os cartões não resultou em problemas aparentes e os módulos foram criados de maneira a serem configurados nos seguintes quesitos:

- Número de bytes do telegrama;
- Taxa de transmissão;

- Taxa de oversample.

Ambas taxas de transmissão e oversample são baseadas no clock disponibilizado para este módulo. A figura 30 mostra uma simulação, utilizando o simulador interno do ambiente de desenvolvimento Xilinx, de envio e recepção realizada por parte do cartão iFOI e a figura 31 a captura real do sinal.

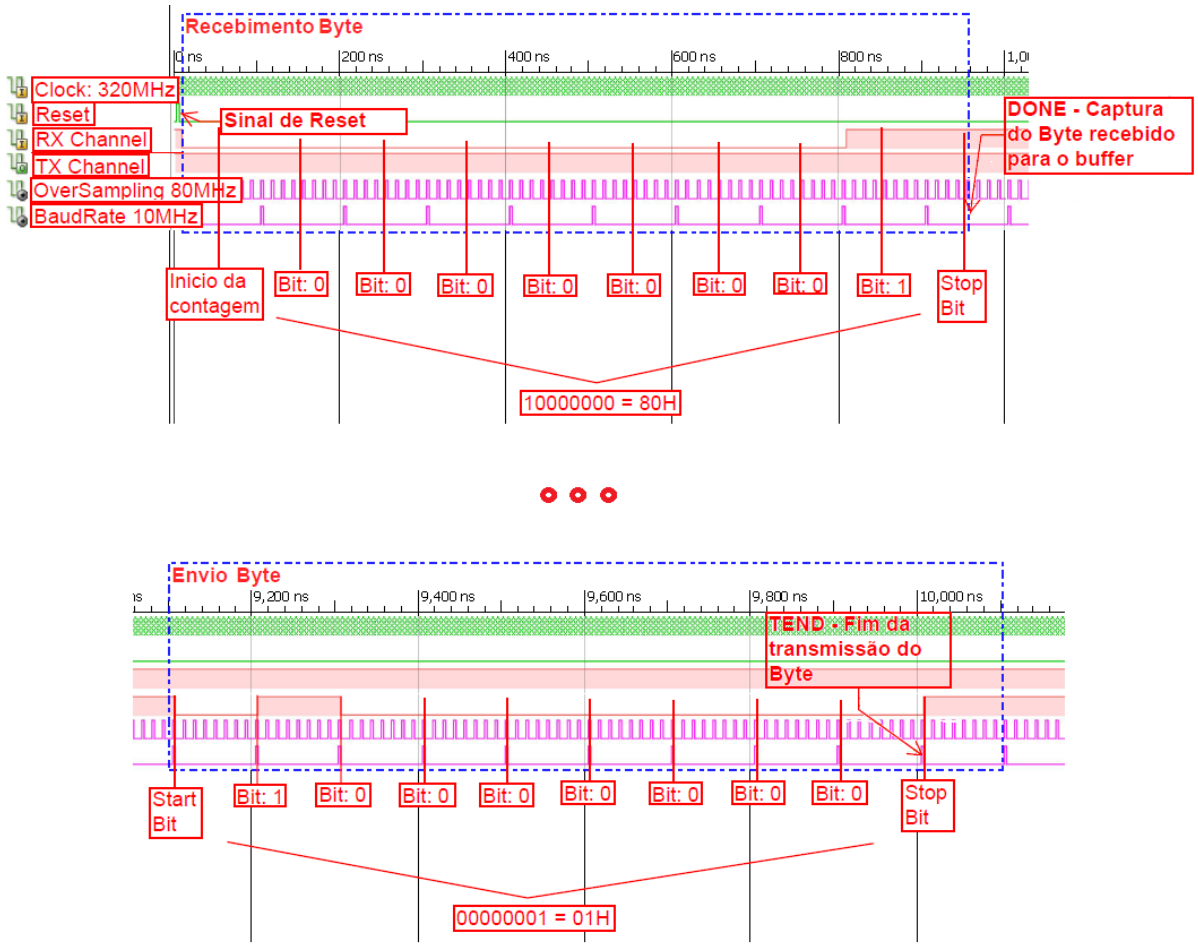

Figura 30 - Simulação de envio e recebimento de um byte utilizando o módulo UART

Após a implementação dos módulos por parte dos dois cartões, foi realizado uma análise da banda total do canal. Com taxas de transmissão podendo chegar até 40Mbaud/s em cada canal e somando todos os canais disponíveis para comunicação (múltiplos TX e RX), a banda total resultante é de 320 Mbits.

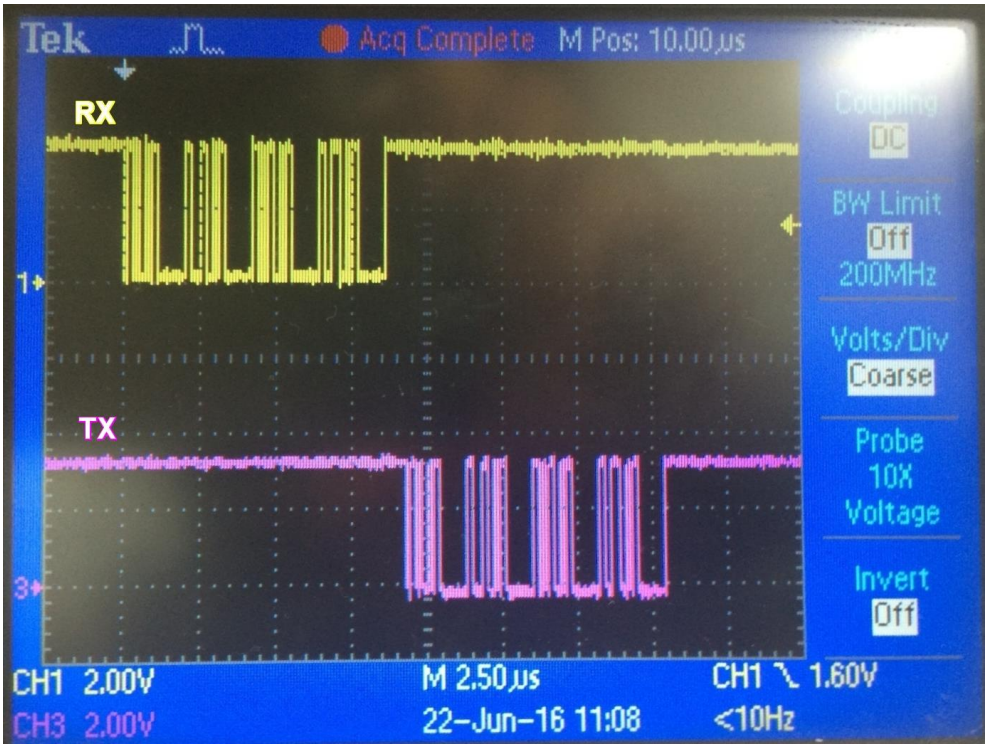

Figura 31 - Captura dos sinais RX e TX realizadas no cartão iFOI para uma transmissão com baud rate de 10MHz

Outro resultado analisado foi a confiabilidade deste canal em longa duração. Para esta verificação foi realizado um teste em que um telegrama padrão é enviado para o cartão iFOI e este retransmite o mesmo telegrama para o cartão CCE-03 onde os dados recebidos seriam armazenados para serem avaliados pela MPU. Este teste ficou em execução diversas vezes por períodos de até 72h ininterruptas e resultou em 0 (zero) perdas de telegrama.

6.5: Comunicação com IHMs WEG

Este teste vem a implementar a comunicação com IHMs produzidas pela WEG com a finalidade de utilizá-la como uma ferramenta de depuração para testes futuros. Ao realizar a implementação do módulo apresentado em 5.5, o primeiro teste de validação foi realizado utilizando um software monitor proprietário. Este programa realiza a parte mestre do protocolo ModBus RTU, onde pode-se solicitar escritas e leituras de apenas um registrador ou de múltiplos registradores em um só telegrama.

Também é possível acompanhar através deste programa a resposta enviada pelo cartão.

O resultado da utilização deste programa foi positivo, tendo o cartão comunicado como esperado, sem que houvesse falha de nenhum telegrama. A figura 32 mostra a janela do programa monitor.

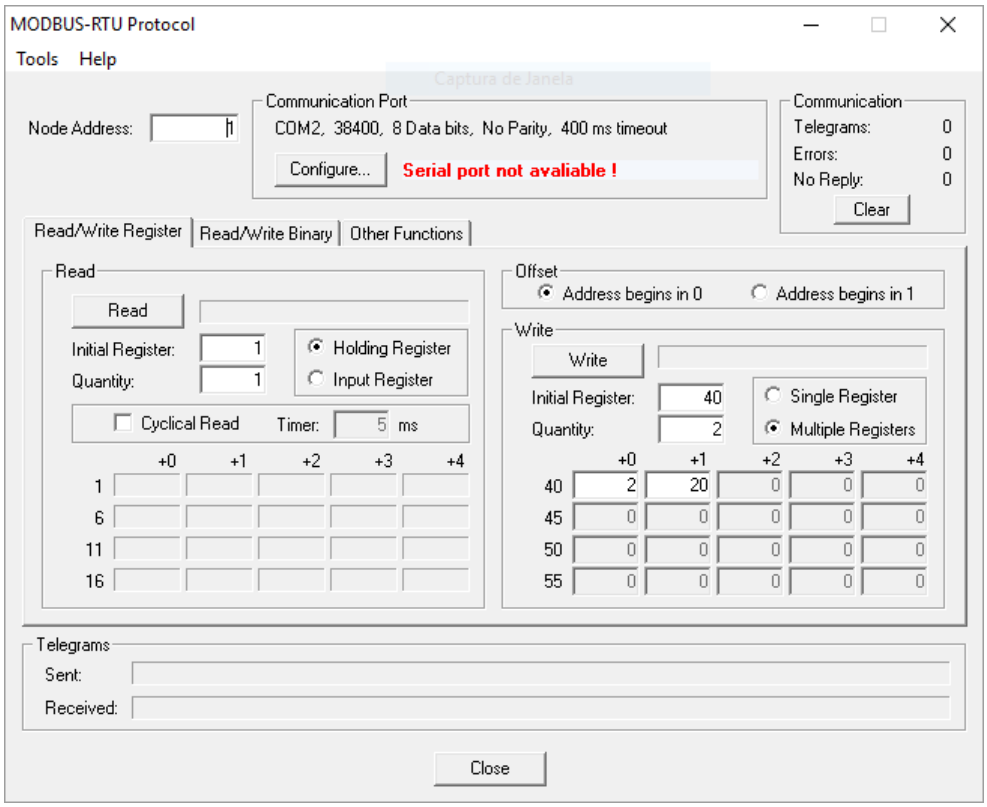

Figura 32 - Janela principal do programa MultiCom

A segunda validação deste módulo foi através da comunicação com uma IHM gráfica (Figura 33), onde pode-se ver os parâmetros da tabela presente no cartão CCE-03. Assim como o ocorrido com o programa monitor, este módulo se comunicou sem nenhum problema com a IHM e alterações realizadas na tabela presente no cartão puderam ser visualizadas pela mesma sem nenhuma perda de informação.

Figura 33 - Comunicação entre cartão e IHM realizado com sucesso

6.6: A/D e D/A

Assim como os testes realizados para as memórias, este teste teve como objetivo verificar a integridade dos circuitos conversores A/D e D/A. Ao realizar a implementação deste teste foi constatado um desvio no projeto de hardware em que o CS do componente D/A não estava conectado a nenhum CS da MPU. Este desvio foi corrigido e o teste destes componentes ocorreu como esperado. A figura 34 mostra a saída gerada pelo componente D/A através de uma função dente de serra (magenta), a entrada do componente A/D (ciano) e uma segunda saída do D/A referente aos dados captados pelo A/D (amarelo).

Figura 34 - Captura dos sinais referentes ao teste realizado nos componentes A/D e D/A

6.7: Conclusão do capítulo

Neste capítulo foram apresentados os resultados obtidos com cada um dos testes apresentados e implementados nos capítulos 4 e 5. Também foram realizadas discussões a respeito de alguns comportamentos encontrados ao longo dos testes. O próximo capítulo apresentará uma conclusão do que foi realizado neste projeto e algumas perspectivas para trabalhos futuros.

Capítulo 7: Conclusões e Perspectivas

Os objetivos apresentados para este projeto foram todos concluídos com resultados que atendem com certa margem os parâmetros da especificação. Os testes validaram o hardware para todos os conceitos novos apresentados pela plataforma, sendo que alguns ajustes pontuais resultantes de desvios no projeto de hardware foram apresentados.

Esta plataforma ainda não está com todo o cronograma de execução completo, mas as funcionalidades fundamentais necessárias para o controle de conversores assim como os conceitos básicos de comunicação entre cartões, acessos a memórias, comunicação com IHMs e conversores A/D - D/A já estão implementados, testados e validados. Com a especificação e estruturação da engenharia de software estes módulos já estão preparadados para integrá-los ao firmware da plataforma quando este for o foco do cronograma de desenvolvimento.

Não foi mencionado nesta monografia, mas um teste funcional foi realizado no fim deste projeto para validar os conceitos utilizados nesta plataforma de controle. Foi implementado um modulador 3 níveis por parte da FPGA e da MPU e, também, módulos de medição para tensões e correntes vindas do motor. As implementações destes módulos tiveram como base os códigos já existentes no banco de dados da empresa.

As projeções para este projeto são de concluir os testes voltados à produto, como implementação do controle, funcionalidades de cliente, interações do produto e demais proteções de hardware para que o cronograma de desenvolvimento da plataforma possa ser concluído e a mesma liberada para uso nos diversos produtos engenheirados WEG.

Com relação a realização deste PFC, muitas disciplinas estudadas no curso de controle e automação foram fundamentais para que ele pudesse ser realizado em tempo hábil e com êxito, entre elas: Metodologia para Desenvolvimento de Sistemas; Redes de Computadores para Automação Industrial; Informática Industrial; entre outras.

Bibliografia

[1] WEG Drives & Controls, "MVW01 - Inversor de Freqûencia de Média Tensão," WEG, Março 16. [Online]. Available: http://ecatalog.weg.net/files/wegnet/WEG-inversor-de-frequencia-de-media-tensao-mvw01-10413103-catalogo-portugues-br.pdf. [Accessed 27 Junho 16].

[2] WEG Drives & Controls, “CFW09 - Inversores de Frequência,” WEG, Abril 12. [Online]. Available: http://ecatalog.weg.net/files/wegnet/WEG-cfw-09-inversor-de-frequencia-10413064-catalogo-portugues-br.pdf. [Acesso em 16 Junho 27].

[3] V. J. D. Negri, "Introdução aos Sistemas para Automação e Controle Industrial," LASHIP, Florianópolis, 2004.

[4] C. A. d. Oliveira, J. A. d. Aguiar and M. G. S. Fontanini, "Dispositivos Lógicos Programável," UNESP - Campus Guaratinguetá, São Paulo.

[5] EESC USP, "Microprocessadores," [Online]. Available: http://iris.sel.eesc.usp.br/sel433a/Micros.pdf. [Accessed 30 Junho 2016].

[6] Sistemas Autônomos e Ubíquos Centro Politécnico - UCPEL, "PADRÃO RS-485," [Online]. Available: http://olaria.ucpel.tche.br/autubi/lib/exe/fetch.php?media=padrao_rs485.pdf. [Accessed 04 Julho 2016].

[7] L. M. R. Codá, "Memória Flash," Departamento de Engenharia Elétrica e de Computação - EESC USP, [Online]. Available: http://disciplinas.stoa.usp.br/pluginfile.php/343991/mod_resource/content/1/MEM%C3%93RIA%20FLASH_2013.pdf. [Accessed 04 Julho 2016].

[8] G. Torres, "Clube do Hardware," 04 Setembro 2005. [Online]. Available: http://www.clubedohardware.com.br/dicionario/termo/sram/272. [Accessed 05 Julho 2016].

[9] EPUSP, "CONVERSÃO ANALÓGICO-DIGITAL," [Online]. Available: http://www.netsoft.inf.br/aulas/1_SIN_Arquitetura_de_Computadores/conversao_analogico_digital.pdf. [Accessed 05 Julho 2016].

[10] UFRJ, "Instrumentação e Técnicas de Medidas - Conversores AD e DA," 2012. [Online]. Available: http://www.peb.ufrj.br/cursos/eel710/EEL710_Modulo13.pdf. [Accessed 2016 Julho 05].

[11] D. Kieras, "C Header File Guidelines," University od Michigan, 19 Dezembro 2012. [Online]. Available: http://umich.edu/~eecs381/handouts/CHeaderFileGuidelines.pdf. [Accessed 30 Abril 2016].

[12] Instituto Newton C. Braga, "Como funcionam as UARTs," [Online]. Available: http://newtoncbraga.com.br/index.php/telecom-artigos/1709-tel006.html. [Accessed 16 Julho 2016].

[13] A. D. P. A. G. Juliana Chiuchisan, "Finite State Machine Design and VHDL Coding Techniques," Internarional Conference on Development and Application Systems, p. 273, 27 May 2010.

[14] WEG S.A., "ModBus RTU," WEG, Abril 2013. [Online]. Available: http://ecatalog.weg.net/files/wegnet/WEG-plc300-comunicacao-modbus-rtu-10000850708-manual-portugues-br.pdf. [Accessed 17 Julho 2016].

[15] H. Puhlmann, "Trazendo o mundo real para dentro do processador - Conversor A/D," 18 Setembro 2015. [Online]. Available: http://www.embarcados.com.br/conversor-a-d/. [Accessed 17 Julho 2016].

Apêndice A

Modelo cabeçalho:

```
////////////////////////////////////////////////////////////////////////////
/// \file        module_name.h
/// \brief       Brief Description
///
/// \author      Author
/// \note        Copyright (c) WEG - All Rights reserved.
///
/// \ingroup     Group declared on the description file
///
/// \date        XX/XX/XXXX
/// \version     X.XX
/// \bug         Bug Description
///
/// \pre         Pre-condition Description
////////////////////////////////////////////////////////////////////////////
//------------------------------------------------------------------------//
//                     Multiple inclusion prevention macro                //
//------------------------------------------------------------------------//
#ifndef MODULE_NAME_H
#define MODULE_NAME_H

//------------------------------------------------------------------------//
//                           Includes for this header                     //
//------------------------------------------------------------------------//

//------------------------------------------------------------------------//
//                          Structure Type Declarations                   //
//------------------------------------------------------------------------//
//--------------------------Definitions and Macros------------------------//

//--------------------------Typedefs and Structures-----------------------//

//---------------------------------Constants------------------------------//

//------------------------------------------------------------------------//
//                              Extern Variables                          //
//------------------------------------------------------------------------//

//------------------------------------------------------------------------//
//                             Function Prototypes                        //
//------------------------------------------------------------------------//

//------------------------------------------------------------------------//
//                             End of header file                         //
//------------------------------------------------------------------------//
#endif
```

Apêndice B

Modelo código fonte:

```
////////////////////////////////////////////////////////////////////////////
/// \file        module_name.c
/// \brief       Brief Description
///
/// \author      Author
/// \note        Copyright (c) WEG - All Rights reserved.
///
/// \ingroup     Group declared on the description file
///
/// \date        XX/XX/XXXX
/// \version     X.XX
/// \bug         Bug Description
/// \pre         Pre-condition Description
////////////////////////////////////////////////////////////////////////////
//------------------------------------------------------------------------//
//                          Includes for this source                      //
//------------------------------------------------------------------------//
//-  Description of the include  -//

//------------------------------------------------------------------------//
//                         Structure Type Declarations                    //
//------------------------------------------------------------------------//

//---------------------------Definitions and Macros----------------------//

//---------------------------Typedefs and Structures---------------------//


//------------------------------------------------------------------------//
//                              Static Variables                          //
//------------------------------------------------------------------------//
/// Description of the variable

//------------------------------------------------------------------------//
//                             Function Prototypes                        //
//------------------------------------------------------------------------//


//------------------------------------------------------------------------//
//                              Module Functions                          //
//------------------------------------------------------------------------//
////////////////////////////////////////////////////////////////////////////
/// \fn          function name
/// \brief       Brief Description
/// \warning
/// \param       param paramDescription
/// \return      valueReturned
////////////////////////////////////////////////////////////////////////////
```