MÁRCIO HENRIQUE FERREIRA
MÁRCIO ROBERTO DA SILVA
RODRIGO BARROCA DIAS FERRAZ

SISTEMA DEDICADO DE MEDIÇÃO DE FREQÜÊNCIA

Projeto de Formatura apresentado à disciplina
PCS 2502 – Laboratório deProjeto de Formatura II,
da Escola Politécnica da Universidade de São Paulo.

Orientadores: Prof. Edson Midorikawa
Prof. Ricardo Paulino Marques

V.1

São Paulo
2005

Agradecimentos

Gostaríamos de citar e agradecer o apoio de várias pessoas, que direta ou indiretamente contribuiram para o projeto. Algumas pessoas merecem uma menção especial.

O professor Edson Midorikawa, do PCS-USP, guiou o planejamento da solução baseada na FPGA, e permitiu que pudéssemos andar com nossas próprias pernas no projeto.

O professor Wang, do PSI-USP, contribuiu com muitas idéias relacionadas à otimização e implementação de partes do sistema, e informações valiosas sobre implementação e manufatura dos sistemas, placas, etc.

Gostaríamos de agradecer também o LSI-USP por ter cedido o kit de desenvolvimento com o qual foi realizado todo o projeto de formatura. A falta deste equipamento provavelmente inviabilizaria os trabalhos realizados.

Ao senhor Mário, da Unisal, pela ajuda no projeto dos circuitos eletrônicos auxiliares.

À BL Representações por fornecer amplos recursos de teste para a realização do projeto.

MÁRCIO HENRIQUE FERREIRA
MÁRCIO ROBERTO DA SILVA
RODRIGO BARROCA DIAS FERRAZ

Sistema Dedicado de Medição de Freqüências

Projeto de Formatura apresentado à disciplina
PCS 2502 – Laboratório de Projeto de Formatura II,
da Escola Politécnica da Universidade de São Paulo.

Aprovado em:

Banca Examinadora

Professor:_____

Instituição:_____ Assinatura:_____

Professor:_____

Instituição:_____ Assinatura:_____

Professor:_____

Instituição:_____ Assinatura:_____

Resumo

O projeto de formatura aqui desenvolvido consiste em projetar e construir o protótipo de um sistema dedicado de medição de freqüência, utilizando hardware dedicado baseado em FPGA, que realize a seguinte operação: receber dados de um medidor de gravidade (gravímetro) e de uma base de tempo estável (GPS ou cristal de precisão, ou uma simulação da mesma), processá-los e armazená-los da forma adequada em um computador.

O freqüencímetro servirá para realizar o processamento do sinal de um instrumento chamado gravímetro, que serve para levantar o valor da gravidade em um certo ponto. Este instrumento está sendo utilizado em um projeto feito para a Agência Nacional de Petróleo, de levantamento de dados geofísicos de uma certa área. O gravímetro é um instrumento de altíssima precisão e sensibilidade, que necessita de um sistema medidor de freqüência dedicado especial para ser compatível com esse tipo de medição. A medição de gravidade impõe várias restrições ao processamento do sinal.

O processamento do freqüencímetro deve responder com dados de alta precisão, com alta velocidade de funcionamento. Um circuito com estas características é alcançado através do projeto de um hardware dedicado. A maneira mais prática de fazer esse tipo de desenvolvimento é utilizando chips do tipo FPGA. A FPGA é um chip no qual se pode criar um circuito digital especial e utilizá-lo como se fosse um CI. Porém a FPGA é reconfigurável, e isto diminui o tempo de projeto, facilita a reconfiguração e mudanças de projeto, e propicia melhor desempenho em relação a um dispositivo baseado em microcontroladores.

Sumário

1 Introdução

1.1 Objetivo

Desenvolver um sistema dedicado medidor de freqüência que possa receber dados de um gravímetro (medidor de gravidade) e de uma base de tempo estável (GPS ou cristal de precisão), processá-los e armazená-los da forma adequada em um computador.

1.2 Motivação

O tema se relaciona à área de sistemas dedicados, processamento de sinais e instrumentação. O sistema proposto é parte do gravímetro do projeto da Agência Nacional De Petróleo (ANP), executado pelo Laboratório de Automação e Controle (LAC), de levantamento de dados geofísicos (entre outros tipos relevantes e dados) de algumas regiões do Brasil.

O gravímetro é um instrumento de alta precisão e que necessita de um sistema medidor de freqüência dedicado especial, com diversas características específicas. Para atender a estas necessidades foi iniciado o desenvolvimento deste projeto.

1.3 Organização

A equipe de desenvolvimento deste projeto é composta por três integrantes de diferentes ênfases de Engenharia Elétrica: Automação e Controle, Computação e Sistemas Eletrônicos. Por este motivo, os trabalhos foram orientados por um professor do Departamento de Automação e Controle da Escola Politécnica da USP (Prof. Ricardo Paulino Marques) e um professor da do Departamento de Computação e Sistemas Digitais (Prof. Edson Midorikawa). Isto foi possível devido a multidisciplinariedade evidente do projeto.

Por fazer parte de um projeto maior da ANP, desenvolvido pelo LAC, a concepção e o desenvolvimento foram acompanhados também pelo professor João Luiz de Paiva Martins que está ligado a equipe de desenvolvimento deste projeto do LAC.

2 Aspectos conceituais

O objetivo deste tópico é permitir ao leitor conhecer e ter um ponto de referência para acompanhar o desenvolvimento do projeto.

Nessa parte serão citadas as principais tecnologias aplicadas, algumas das ferramentas utilizadas no projeto, e criamos referências que podem ser consultadas sempre que algum ponto citado na implementação ou desenvolvimento não estiver suficientemente claro.

Apresentamos inicialmente as principais tecnologias referenciadas, e numa segunda parte serão cobertos conceitos freqüentemente utilizados e um pouco de teoria de circuitos digitais. Nem tudo que foi utilizado está descrito, mas os de maior influência podem ser facilmente reconhecidos.

2.1 Tecnologias

As tecnologias empregadas estão relacionadas à construção de circuitos digitais e transmissão de dados. Vamos explicar como é feito esse projeto e com que ferramentas.

2.1.1 Projeto VHDL

Este projeto consiste em desenvolver um hardware dedicado para realizar o processamento digital de sinais.

As linguagens de descrição de hardware são utilizadas justamente para descrever o funcionamento de circuitos digitais. Mais recentemente, com o advento de chips programáveis, elas têm se tornado muito difundidas para descrever e sintetizar projetos para FPGA's e circuitos ASIC's, que são basicamente projetos de circuitos digitais customizáveis, em projetos dos mais variados tipos, tais como aplicações customizadas, baixa quantidade, protótipos e sistemas no chip (SoC, System on Chip).

Existem várias linguagens de descrição de hardware, tais como o VHDL (nossa escolha), Verilog, SystemC, AHDL, etc. Muitas delas são específicas das empresas que as desenvolvem.

A linguagem VHDL se tornou um padrão do IEEE para descrição de circuitos digitais. O que se faz numa descrição VHDL é criar o circuito digital baseado em unidades lógicas básicas, tais como flip-flop, registradores, etc. É possível encapsular circuitos de modo a se fazer o reaproveitamento de código em outras partes do circuito, ou até mesmo outros projetos.

2.1.2 IDE e Simulador

Esta seção fala sobre o que sem dúvida é o objeto de trabalho principal durante o projeto. O termo IDE significa "Integrated Development Environment", ou seja, ambiente integrado de desenvolvimento. IDE's para o design de software são bem difundidos hoje em dia. Podemos citar como exemplos famosos o Visual Studio e o Eclipse.

Existem vários tipos de IDE's para diversos propósitos, e a sua função é prover ferramentas que facilitem a criação e gerenciamento de projetos de software ou outros tipos de design. No nosso caso, utilizamos o ISE 7.1i, que é um ambiente de desenvolvimento da Xilinx, que é a empresa que fabrica a FPGA de trabalho.

O processo de criar um sistema funcionando em uma FPGA não é trivial. Existem várias etapas para serem feitas e o uso do IDE realmente facilita o projeto. O ISE permite o acompanhamento do projeto desde as primeiras codificações até o envio do código de baixo nível para a placa ou EPROM/flash.

É possível durante o processo de desenvolvimento simular o código VHDL sintetizado. Isso é extremamente útil para depurar problemas na lógica de implementação e comportamentos imprevistos.

No simulador é possível acompanhar todos os sinais utilizados, internos ou externos, e seus valores a cada instante. Esse tipo de simulador é chamado simulador de waveform, pois podemos acompanhar as formas de onda dos sinais. Podemos simular diversas situações alterando as formas de onda de entrada, simular falhas, situações anormais, transitórios, etc. Os sinais podem ser agrupados de modo a fazer sentido. Por exemplo, podemos agrupar as saídas paralelas de um registrador para que possamos ver o número que eles representam, ao invés de uma cadeia de bits (por exemplo, 12 no lugar de 00001100).

2.1.3 Design em FPGA

Vamos fazer uma breve introdução sobre como é feito o projeto de um circuito digital em FPGA, para que o leitor possa acompanhar como foi desenvolvido o projeto do nosso processador digital de sinais.

O termo FPGA significa Field Programmable Gate Array, ou seja, um conjunto de dispositivos de lógica programável. Ou seja, a FPGA é um chip no qual é possível programar (e reprogramar) o circuito lógico digital que está operando no interior da mesma.

A FPGA é muitas vezes utilizada para circuitos dedicados, projeto customizados e em soluções que envolvem rápida prototipação ou baixo volume.

É possível gerar vários tipos de circuito para funcionarem na FPGA, ela é muito flexível. Existem vários tipos de FPGA, sendo que as mais potentes podem suportar rodar uma CPU virtual com a potência de uma CPU PowerPC inteiramente funcional. Esses circuitos são muitas vezes referenciados como "soft-cores", em uma referência à palavra software, para lembrar que não é um processador real, mas sim um circuito de CPU rodando em cima da FPGA. Alguns sistemas de FPGA possuem integrados uma CPU para aumentar o desempenho (pois o circuito da FPGA é mais lento que os chips comuns), dada a possibilidade de reconfigurabilidade, é normal se esperar um circuito de menor desempenho.

Para se criar um circuito em FPGA são necessários várias etapas. Não vamos descrevê-las aqui, pois seria precipitado. Na parte de desenvolvimento é possível acompanhar o processo com riqueza de detalhes e telas do programa ISE, para facilitar o acompanhamento.

2.1.4 Protocolo RS232

O protocolo RS232 para transmissão serial de dados é muito difundido. Apesar de ser um protocolo um pouco antigo, sua importância e relevância não são diminuidos.

Vários protocolos presentes atualmente se baseiam na transmissão serial, por exemplo, é possível no protocolo Bluettoth emular uma porta serial de comunicação. É possível utilizar conversores entre protocolos Serial e USB, o que permite utilizar componentes com comunicação serial utilizando cabo USB, ou fazer crer a um programa em um computador que ele possui uma porta serial (serial virtual), quando a ligação na verdade é USB ou Firewire.

Essa flexibilidade e a facilidade em implementar a comunicação serial nos fez optar por esse caminho na construção do protótipo. Esta solução pode ser facilmente adaptada ou modificada para outros casos, ou ser encapsulada em uma solução mais moderna, por exemplo.

O protocolo é simples e a implementação da controladora serial UART (Universal asynchronous receiver transmitter) não exige muitos componentes ou desempenho. Tanto é que no nosso projeto estamos implementando a controladora de serial internamente à FPGA, em VHDL. A UART é responsável por tratar os dados recebidos pelo cabo serial e preparar dados para serem enviados, além de conter mecanismos de detecção de erro de transmissão simples.

2.2 Conceitos utilizados

Este segundo bloco é um resumo das principais áreas estudadas e aplicadas ao trabalho de formatura.

2.2.1 Teoria de medição

O processo de medição por contagem é o método de medição de freqüência dominante. Neste método busca-se gerar eventos que ocorrem com a mesma peridiocidade que o fenômeno que se quer medir, e no caso da contagem digital, deseja-se que esse evento seja um sinal digital que seja passível de ser contado por um sistema digital em seguida.

Muito foi pesquisado pela equipe sobre instrumentação e medição de freqüência. Há relativa pouca informação a respeito desse campo, pois na maioria das aplicações uma contagem simples é suficiente, e com o advento de circuitos digitais cada vez mais potentes, os empecilhos

diminuíram bastante. Também não existem muitos fabricantes de medidores de freqüência de alta precisão e/ou calibradores de freqüência.

A medição de freqüência basicamente é feita de duas formas:

1) Medição por contagem: compara-se a repetição de um evento com outro conhecido ou de periodicidade conhecida

2) Medição por ressonância: método para alguns freqüencímetros (principalmente de microondas) que utilizam uma cavidade de ressonância para realizar a medida. Este segundo método, por ressonância, é pouco utilizado, pois funciona apenas para uma certa faixa de freqüências.

O processo de contagem é muito mais genérico, e baseia-se na simples comparação com uma referência conhecida. O uso de cristais fornecendo uma base de tempo estável e a crescente quantidade de circuitos digitais popularizou esse método. a geração de pulsos em eventos ou sinais digitalizados são fáceis de serem obtidos.

O método de contagem por comparação também tem alguns problemas intrínsecos. Em uma medição de um sinal senoidal, por exemplo, existem alguns problemas relacionados à detecção do ponto que o sinal é nulo, pois geralmente a medida é feita entre dois intervalos em que o sinal vale zero.

Pode-se reduzir esse problema digitalizando o sinal a ser medido (utilizando um comparador com AmpOp, para saturar o sinal em +Vdd e -Vdd, como no nosso caso), e obter uma onda quadrada com a mesma freqüência do sinal original. Uma vez obtida uma onda digital, é muito mais simples tratar da contagem, pois pode-se utilizar detectores de borda de subida ou descida.

Para um sinal periódico, o tratamento pode ser imediato. Pode-se introduzir algumas variações para o caso do sinal ter um duty cycle diferente de 50%, ou um buffer de entrada, se o

sinal for composto de pulsos irregularmente distribuídos no tempo (ou no caso de uma medição única, não contínua, chamada "single shot").

2.2.1.1 Métodos de contagem

Os métodos de medida da contagem são de 3 tipos:

1) Inversão da medição do período

Este método é utilizado para a medição de freqüências baixas e consiste em contar quantos ciclos de um sinal de clock conhecido (sinal de referência) ocorreram durante um ciclo executado pelo sinal desconhecido. Assim, obtém-se a duração do pulso como sendo n vezes o período do sinal de referência. Nota-se também que neste método há uma incerteza intrínseca de +/- 1 período do clock de referência. Outra coisa a se notar é que este caso corresponde à situação deste projeto, em que há um sinal de KHz a ser medido, e uma referência na ordem de MHz.

2) Número de pulsos ocorridos em um certo intervalo

Para sinais mais rápidos é mais interessante contar quantos ciclos ocorrem para um certo intervalo conhecido, ou seja, um ou mais períodos da base de referência.

3) Medição do tempo necessário para ocorrer um certo número de pulsos

É uma variação do método 2, para o caso em que a freqüência do sinal digital é variada, ou seja, onde se tem pulsos muito próximos ou muito distantes um do outro, sendo que o sinal seria uma combinação de um sinal de baixa e um de alta freqüência. Neste caso, deve-se tentar

absorver estas variações realizando um número de medições igual aproximadamente ao valor de saída que se espera. Por exemplo, se está sendo medido um sinal de MHz, com pulsos não uniformes, o ideal seria realizar um milhão de medições em um segundo, o que englobaria as variações de maneira suficiente. O problema é que é necessário ter em mãos uma estimativa da freqüência previamente.

Logo, utilizando o fato que devido ao circuito ser digital o valor medido sempre tem um erro intrínseco de mais ou menos um período do clock mais elevado, pode-se calcular de maneira simples o erro envolvido em cada caso, o que pode ser encontrado no artigo da National Instrumentos sobre precisão de contagem, na referência deste documento.

2.2.1.2 Análise do erro

Dados os três métodos básicos de medição simples, as fórmulas de erro de medição podem ser deduzidas. É importante dizer que como foi dito, são métodos simples e que servem de base para a realização de uma medida. Em uma aplicação real, utilizam-se esses métodos como artifícios que possibilitam aumentar a precisão do sistema.

Para fins de conhecimento, vamos enumerar os resultado abaixo. Sejam as variáveis de interesse:

Fk: freqüência conhecida, estável

Fx: freqüência a ser medida

n: resultado do contador

k: repetições de medida

Fm: freqüência medida

Erro método 1 (Fk>>Fx): Fm=Fk/n e err(Fm)=Fx/(Fk-Fx)

Erro método 2 (Fk>>Fx): Fm=Fk*n e err(Fm)=Fk/Fx

Erro método 3 (igual #1): err(Fm)=Fx/(k*Fk-Fx)

Para um efeito ilustrativo, vamos imaginar alguns cenários de medição (que por conveniência são próximos do nosso cenário).

Segundo o método 1, para um clock de 80MHz (pensando em um oscilador OCXO de 20MHz, e um sistema para quadruplicar esta freqüência), e tendo como alvo um valor de aproximadamente 27 bits (o valor deve ser próximo de 27 bits) e entrada na ordem de KHz, tem-se uma variação na medida devido somente a esse erro já na primeira casa decimal (ou pior). Ou seja, para um clock de 80 MHz, e sinal na casa de 1K, a máxima precisão limitada pelo método de divisão é de uma casa depois da vírgula.

2.2.1.3 Base estável de referência

Conforme foi citado, a contagem geralmente necessita de uma base estável de referência de tempo. As bases de tempo de referência são geralmente de dois tipos:

Cristal / Oscilador

Sinal de referência de GPS

O GPS fornece uma referência muito estável de 1pps (1Hz), ou seja, um sinal de 1 segundo de duração. Este sinal geralmente é usado para calibrar o instrumento (que pode ser um calibrador de instrumentos também), e geralmente é utilizado em equipamentos muito sensíveis.

Os instrumentos possuem também uma referência interna de um oscilador de cristal. Geralmente é utilizado um sistema chamado OCXO (Oven Compensated Crystal Oscillator), que

é composto por um oscilador, e um sistema de aquecimento com controle, de modo a manter o cristal em uma certa temperatura estável, pois a freqüência de oscilação de um cristal varia com a temperatura. É necessário manter a temperatura estável para manter a precisão da medida. Também é importante obedecer e levar em conta o tempo de "warm up" do circuito, que é o tempo que ele demora para estabilizar e funcionar com precisão máxima.

Em geral, pode-se obter OCXO's de algumas dezenas de MHz. A intenção neste projeto é usar um OCXO próximo de 180MHz para fornecer o sinal de referência, um valor elevado, de modo a reduzir o erro intrinseco na medida, além de poder utiliza-lo também como clock do circuito.

Nota-se também que os instrumentos mais caros possuem um cristal de Rubidio, aparentemente mais estável que os comuns. Há ainda a possibilidade de usar uma medida atômica, como um relógio de Césio, como existem em alguns equipamentos de instrumentação e calibração de laboratórios.

2.2.2 Organização de sistemas digitais

Sendo um projeto de hardware digital, foram utilizados muitos conceitos de Organização de Sistemas Digitais. Além disso, foram utilizados conceitos de projeto de sistemas dedicados (embedded systems).

2.2.2.1 Fluxo de Dados e Unidade de Controle

A estrutura básica de um circuito digital é composta por duas partes distintas que interagem para realizar o comportamento funcional do circuito: Fluxo de Dados e a Unidade de Controle.

O Fluxo de Dados é composto em geral por registradores, unidades funcionais e alguns elementos de interligação arranjados na topologia adequada a implementação do algoritmo executado pelo módulo digital.

A Unidade de Controle é um circuito seqüencial que determina as ações que devem ser executadas no Fluxo de Dados na ordem especificada pelo algoritmo do módulo e nos instantes corretos determinados pelas relações de tempo existentes entre o componente físico e o módulo. A Unidade de Controle é composta, em geral, por um registrador de estado, por uma lógica responsável pela determinação do próximo estado e por outra lógica responsável pela determinação dos valores da saída, além de, nos circuitos síncronos prover também a lógica de geração de todos os sinais de tempos necessários.

2.2.2.2 Síntese de processador dedicado

Como descrito, o funcionamento correto do circuito depende de ele funcionar com alto desempenho. É crítico que o circuito funcione com a máxima eficiência.

O processador dedicado é constituído de fluxo de dados com capacidade de guardar e manipular dados e um controlador capaz de manipular os dados através desse fluxo. Alguns sistemas requerem tarefas tão específicas que um processador dedicado deve ser projetado para cumprir tal tarefa. Este é o caso deste projeto. Esse tipo de circuito é chamado de "custom single-purpose processor".

Assim como os processadores comuns, estes também são baseados em lógica combinacional. Porém, convertemos o programa gerado em uma Máquina de Estados Finitos (FSM, finite state machine) representando a unidade de controle, que é criada através de exemplos construtivos adaptados para cada caso.

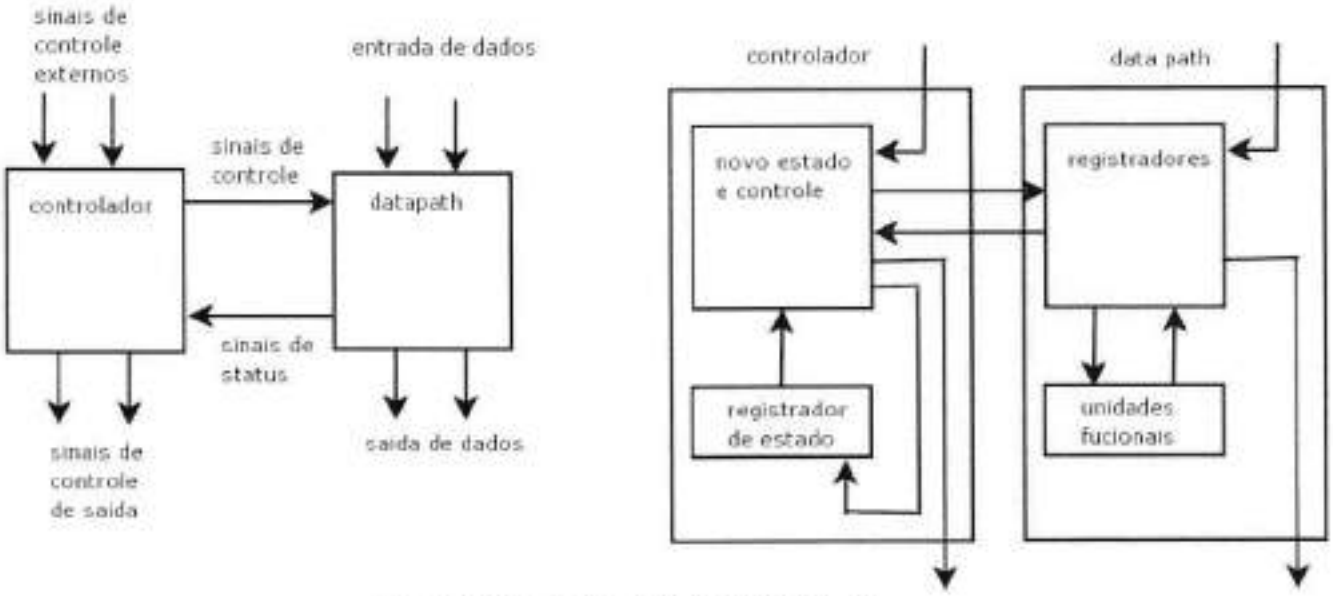


Figura 1: Estrutura geral de um módulo digital

O fluxo de dados (ou datapath) é criado observando a seguinte regra:

1) Criar registradores para as variáveis declaradas

2) Criar unidades funcionais para as operações de processamento que são executadas

3) Conectar as portas lógicas, registradores e unidades funcionais, com multiplexadores e selecionadores de sinais, se necessário

4) Criar identificadores únicos para cada sinal de controle e de status dos componentes do fluxo de dados

Após este procedimento básico é possível otimizar o sistema gerado, tanto o programa quanto a FSM. O uso de melhores algoritmos, simplificações podem tornar o programa mais eficiente. A máquina de estados pode ter estados repetidos, transições desnecessárias.

Pode-se usar técnicas de otimização e minimização para reduzir o tamanho dos circuitos e o número de estados. Esta tarefas são feitas durante a alocação de operações que serão realizadas em cada estado e na alocação de tipos de componentes que serão utilizados no fluxo de dados.

3 Especificação do Projeto

Esta seção consiste na descrição de toda a especificação do projeto, desde seus principais requisitos até a explicação de detalhes do seu funcionamento.

3.1 Especificação dos requisitos

Após conversa com os orientadores do projeto, foram definidas dentro do objetivo traçado algumas características importantes que o sistema deve possuir. Além disso, existem restrições de projeto que acabam por induzir uma certa abordagem ou tecnologia. Desta forma, os principais requisitos do projeto foram:

- **Possibilidade de construção em placa compacta**

Para poder ter operação tanto embarcada (aviões) como em terra e poder ser transportado sem grande dificuldade o projeto deve possibilitar uma futura implementação em placa de dimensões reduzidas.

- **Altíssima precisão na medida**

Para poder ser utilizado no projeto do gravímetro, o sistema desenvolvido necessita de altíssima precisão nas medidas, já que os valores de gravidade mensurados alcançarão 3 casas decimais.

- **Alta velocidade de operação**

Para se atingir o nível de precisão de medida desejado, é necessário que a aquisição e processamento dos sinais seja feita em uma velocidade elevada (localização da posição geográfica da medida).

- **Reconfigurabilidade**

Como o freqüencímetro poderá vir a ser utilizado também em um magnetômetro, ele deve possuir facilidade de alteração de suas configurações principais.

- **Facilidade de operação**

É desejado que o equipamento projetado possua uma certa facilidade de operação, para que os técnicos que o utilizarem futuramente não tenham que enfrentar maiores problemas.

- **Adequada disponibilização de dados**

Os dados de saída do sistema devem ser disponibilizados de forma adequada, para que possam ser utilizados facilmente pelo usuário final, podendo ser relacionados a dados recolhidos de outras fontes.

3.2 Detalhamento do projeto

A partir das limitações do projeto, começaram a ser pensadas soluções para o sistema medidor de freqüência. Estão listadas abaixo algumas das conclusões iniciais sobre o problema, para se atingir os requisitos.

- **O sistema será de operação dedicada**

Não é possível utilizar um sistema pronto de medição de freqüência. Isso é devido ao alto grau de especialização do sistema, não seria possível atingir os requisitos de projeto;

- **O sistema deve ser compacto, no formato de uma placa compacta**

Este formato é adequado ao de um sistema embarcado, não devendo ocupar muito espaço, e sendo robusta e de fácil operação;

- **O sistema será implementado em hardware**

Por exigir um grande desempenho (muitas contagens por segundo), um altíssimo nível de precisão e por ser um sistema dedicado, a implementação em hardware passou a ser altamente recomendável em detrimento a um possível desenvolvimento em software.

- **o tratamento de sinais será feito em FPGA**

A FPGA (field programmable gate array) é um circuito integrado especial que pode ser programado para conter diversos tipos de circuitos de lógica digital. Ele é denso (pode implementar grandes circuitos, até pequenas CPUs RISC), altamente reconfigurável e flexível, além de ser um hardware dedicado para a medição, o que dá desempenho e segurança à implementação. Ela traz grande flexibilidade, pois pode abrigar outras partes do projeto (como o controle do display, ou o driver de USB), desde que haja espaço (leia-se pinos suficientes) para i/o e área disponível para o circuito lógico na FPGA.

- **Os dados devem ser transmitidos para um computador**

O sistema deve fornecer os dados a um computador, através de uma saída comum, tais como cabos seriais (opção escolhida), USB ou FireWire, pois os dados serão tratados e incorporados ao conjunto de dados levantados pelos outros sistemas.

Foi utilizada para o desenvolvimento do trabalho de formatura uma placa de desenvolvimento em FPGA da Digilent Inc. A placa consiste em dois módulos: o módulo principal da FPGA e um módulo de I/O digital que utilizamos para verificar o funcionamento. A placa da Digilent contém uma FPGA model Spartan IIe, da Xilinx, que é a família (Spartan) de chips alvo do nosso desenvolvimento. Foi utilizado o ambiente de desenvolvimento da Xilinx chamado ISE 7.1i, que permite realizar o projeto do circuito digital e enviar para a placa de desenvolvimento através do protocolo JTAG. Na fase de desenvolvimento é possível utilizar um simulador de waveform chamado Modelsim MXE, que opera em conjunto com o ISE. A placa de desenvolvimento foi gentilmente cedida pelo LSI-USP, Laboratório de Sistemas Integráveis da USP.

3.3 *Características funcionais*

Neste tópico será tratado todo o funcionamento do sistema projetado e de seus subsistemas.

3.3.1 Como funciona o freqüencímetro

O objetivo desta seção é dar uma visão geral de como é feita uma medição utilizando o freqüencímetro. Será dada uma visão de alto nível do funcionamento, já que os detalhes de cada módulo estão explicados posteriormente, e dar uma introdução geral ao sistema.

Será explicado o sistema começando da saída do gravímetro até chegar no dado processado no computador, que são os limites extremos deste sistema, cuja tarefa é intermediar a "conversa" entre o gravímetro e o computador, e servir de ponte de ligação entre estas camadas.

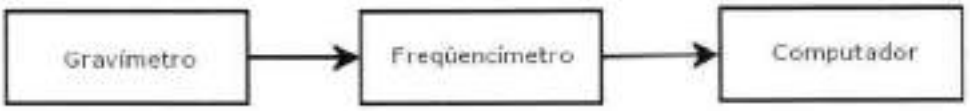


Figura 2: Sistema geral do gravímetro

O sinal do gravímetro é uma senóide. A freqüência dessa senóide representa uma indicação do valor de gravidade e é a informação que se quer coletar. Além disso, é importante localizar no tempo o momento da medição, pois como o instrumento pode se mover (embarcado no avião) esse dado também é importante para localizar a posição geográfica em que foi feita a medição.

O sinal é recebido e tratado, para evitar ruídos. Em seguida ele é digitalizado e entra na FPGA. A FPGA também recebe uma informação de base de tempo estável (geralmente de um cristal, ou um sinal de GPS, que é muito estável). O sinal é comparado com essa base de tempo para gerar a informação de tempo. O sinal depois é dividido por essa freqüência de base, para sabermos sua freqüência com alta precisão. Além disso, pelo número de ciclos entre cada medição podemos também estimar o tempo desde a última medição, que é o outro dado passado.

Os dados gerados após o processamento devem ser codificados segundo o protocolo estabelecido, para então ser transmitido pela controladora de comunicação serial (UART) para o computados. Este recebe através de uma porta serial (ou cabo USB, emulando uma porta serial virtual) os dados enviados pelo processador na FPGA. Um programa responsável por estabelecer e manter a comunicação serial recebe os dados da FPGA e decodifica as informações segundo protocolo de comunicação estabelecido em cima da comunicação serial, e pode mostrar os dados na tela, imprimir para um arquivo de dados (que são a freqüência e o tempo decorrido desde a última medição), ou gerar gráficos, entre várias outras opções.

3.3.2 Como funcionam os módulos

3.3.2.1 Aquisição do sinal

Neste tópico será descrito o funcionamento do sistema de aquisição de sinal. A visão geral desta etapa está ilustrada abaixo.

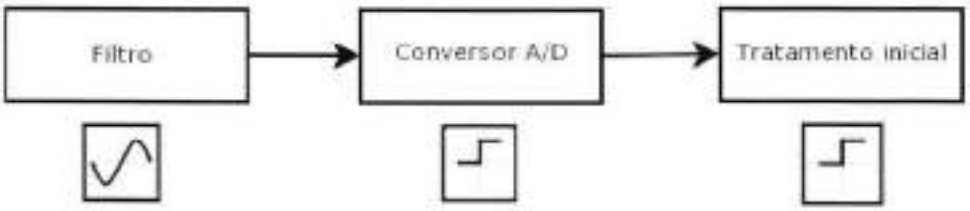


Figura 3: Sistema de aquisição de sinal

3.3.2.1.1 Conversão A/D

O sinal utilizado como entrada provém do gravímetro. Ele foi especificado pelos desenvolvedores do gravímetro como uma onda senoidal, com amplitude de 3 V, offset nulo, e freqüência variando de 250 a 3000 Hz.

Essa entrada deve ser digitalizada para realizar o processo de medição pelo método de contagem. O outro método que não necessita de digitalização é a medição de freqüência por ressonância. O método tradicional de medição de freqüência consistem em gerar eventos cíclicos que provoquem pulsos ou trens de pulsos e contar a quantidade de pulsos decorridas durante o evento repetitivo.

Para tanto, o sinal senoidal de entrada é convertido em uma onda quadrada que seja compatível como o circuito digital. Isso é feito através do CI comparador LM339. Esse CI consiste basicamente de amplificadores operacionais simples.

Foi utilizado um dos Amp-Op's do chip em aberto, e alimentado com uma tensão de Vcc=3,3V e a outra alimentação simétrica aterrada. Como o Amp-Op está em aberto e sem a ligação negativa, no semiciclo positivo da senóide, o sinal será saturado em Vcc, enquanto que no semiciclo negativo, o sinal ficará aterrado. Com isso é gerada uma onda quadrada que representa a mesma freqüência do sinal senoidal de entrada e compatível com o circuito de níveis de tensão LVTTL, e assim pode ser ligado diretamente a um pino de entrada da FPGA, para ser processado.

O processo de digitalização altera ligeiramente o duty cycle do sinal. Isso porque o sinal fica mais tempo em nível baixo do que em nível alto. Isso, porém não afeta a freqüência do sinal, pois para cada ciclo da senóide é gerada uma borda de subida, que é utilizada para disparar um processo de contagem.

3.3.2.1.2 Filtragem

O segundo estágio, último antes da digitalização, consiste na filtragem do sinal. Esse estágio não está implementado no protótipo montado a partir da placa de desenvolvimento, pois o sinal utilizado provém de um gerador de sinais, e não seria necessário filtrar tal sinal. Na implementação real, é necessário remover os ruídos presentes através de um adequado filtro analógico.

3.3.2.1.3 Pre-scaling

A operação de pre-scaling consiste em alterar um sinal de entrada para facilitar o funcionamento do circuito. Por exemplo, aquisita-se um sinal de microondas, e através de um divisor de freqüência reduz-se o sinal até que ele entre na faixa de freqüências de rádio. O processamento do sinal é feito através de um circuito que opera nesta região. Após o processamento, o sinal é restaurado para a freqüência de microondas.

Geralmente quanto mais rápido um sinal, mais tempo temos que amostrar ou maior o desempenho exigido do circuito. O efeito de pre-scaling do sinal (utilizar um sinal mais lento) equivale no nosso circuito a aumentar a quantidade de ciclos amostrados no sinal de entrada, e a idéia é semelhante: reduzir os erros de medição (diminuir a significância) pelo aumento do universo de dados existente.

3.3.2.2 *Processamento do sinal (FPGA)*

Este seção descreve o funcionamento do processamento do sinal pela FPGA. Seu esquema geral está ilustrado abaixo.

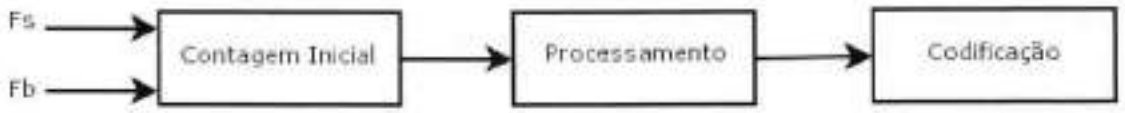


Figura 4: FPGA – alto nível

3.3.2.2.1 Contagem inicial

O primeiro estágio tem por tarefa armazenar as informações dos sinais de entrada. Os sinais de entrada são dois, sendo o primeiro deles o sinal medido do gravímetro, e o outro sinal consiste na base de tempo. No protótipo feito na placa de desenvolvimento Digilent o sinal da base de tempo é o próprio clock.

Os dados digitalizados são contados (bordas de subida) para determinar quantos ciclos foram executados durante um procedimento de medição.

Os contadores e registradores são controlados por uma U.C. (unidade de controle) que controla o momento em que a aquisição de dados se inicia e se encerra, e após a aquisição os dados são enviados para registradores de saída, para serem processados.

3.3.2.2.2 Processamento das entradas

O bloco de processamento das entradas é o coração do circuito lógico, e é também o maior bloco em VHDL do circuito. Ele é responsável por gerar os resultados pedidos de freqüência e tempo entre medidas para que eles sejam enviados para o micro, a partir das entradas do sistema.

Este bloco se subdivide em dois grandes blocos: o bloco divisor e o multiplicador. O bloco divisor recebe os sinais inicialmente e divide a freqüência de base pela contagem. O valor é multiplicado pelo número de períodos do sinal medido para obter o valor da freqüência esperado e então armazenado em um registrador de saída.

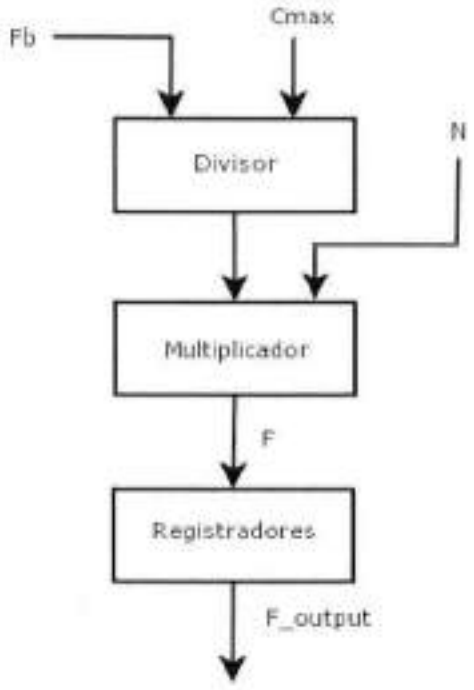


Figura 5: Processamento do sinal

Gera as saídas:

- (Fb/C)*N = freqüência
- Contagem de clocks (tempo entre medidas)

3.3.2.2.2.1 Divisor

O circuito divisor realiza a operação de divisão de números inteiros e é uma implementação baseada no circuito de divisão do processador Leonel, que é uma implementação de uma CPU compatível com a definição do processador Sparc V8, um processador RISC da Sun Microsystems, feita pela ESA (European Spacial Agency, Agência Espacial Européia).

O divisor implementa o algoritmo de divisão digital Radix-2. O algoritmo Radix é muito famoso e de uso comum comercial em processadores que implementam a operação de divisão em hardware, servindo de base para vários circuitos implementados em chips da Intel, Sun, etc.

3.3.2.2.2.2 Multiplicador

O segundo estágio muiltiplicador é muito mais simples que o primeiro, pois existem implementações simples de multiplicação em circuitos digitais, diferentemente do caso da divisão.

O segundo estágio é uma implementação em VHDL simples que realiza a multiplicação do valor do divisor pelo número de período contados para que se tenha a freqüência medida em Hz (e não em ciclos de clock ou contagem pura).

Existe um circuito de apoio para armazenar os valores e controlar o envio dos mesmos para estágios posteriores de processamento.

3.3.2.2.3 Codificação da comunicação

O terceiro bloco principal da FPGA é o bloco responsável pela comunicação entre a FPGA e o computador. As tarefas principais do bloco podem ser divididas em duas partes: a) Tratamento dos pacotes RS232 e b) Tratamento do protocolo em cima dos pacotes (encoding e decoding)

A primeira parte trata do envio e recebimento dos pacotes em padrão RS232. Essa parte compreende basicamente o bloco da UART. A parte do envio de está explicada no próximo tópico.

A outra parte necessária para se fazer a comunicação é a codificação das informações que será enviada pela UART.

Essa parte é composta basicamente por buffers e registradores que enviam as informações na ordem esperada pelo protocolo.

Os dados recebidos da UART para o sistema de decodificação são colocados em registradores conforme os pacotes de controle enviados. O protocolo completo está especificado em um tópico posterior.

Essa parte do circuito tem função e funcionamento análogos ao do programa em C que trata a "outra ponta" da comunicação, serial, do lado do computador.

Pode-se considerar esse bloco como uma codificação de nível superior à codificação de pacotes seriais da UART (ou seja, está em uma camada de rede superior), pois ela leva em conta os dados que estão sendo enviados.

3.3.2.3 *Comunicação de dados*

Esta seção trata do subsistema de comunicação do circuito da FPGA com o computador. A figura abaixo ilustra o esquema desta etapa do sistema.

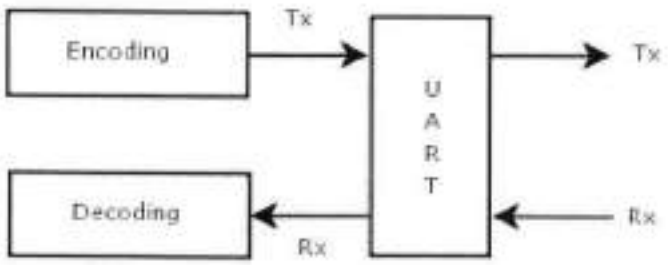


Figura 6: Sistema de comunicação

3.3.2.3.1 Implementação da UART

O termo UART significa Universal Serial Assyncronous Receiver-Transmitter, ou seja, é o circuito digital responsável por converter os bits para a transmissão de dados RS232,e vice-versa.

Através da UART o circuito "conversa" com o computador, podendo enviar os dados processados, e receber informações de reconfiguração.

O circuito para permitir a comunicação no padrão RS232 é de baixa complexidade, e não ocupa muita área. Muitos circuitos dedicados, tais como microcontroladores possuem chips de UART integrados no seu encapsulamento. Para evitar a inclusão de novos chips, fizemos o design da UART de forma interna à FPGA.

A UART foi implementada em VHDL e faz parte do circuito digital que funciona na FPGA. Abaixo podemos ver o símbolo funcional correspondente à UART no simulador ISE:

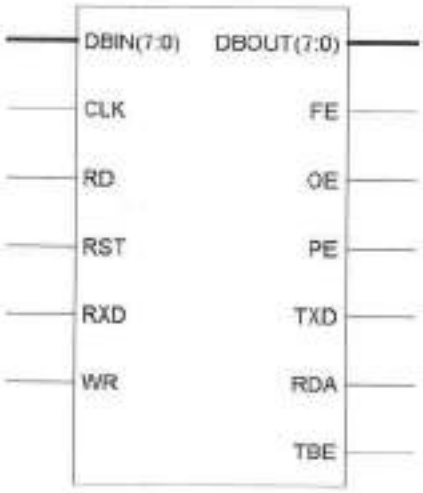


Figura 7: Símbolo funcional da UART no ISE.

3.3.2.3.2 Protocolo de transmissão

Para a demonstração do conceito de transmissão e reconfiguração por transmissão serial do nosso sistema criamos um protocolo básico para a comunicação.

Cada pacote de transmissão serial compreende 11 bits. Cada pacote possui 8 bits de dados que podem ser transmitidos (payload do pacote). Os 3 bits adicionais (que não fazem parte da informação) são:

- Bit de início de transmissão

- Bit de fim de transmissão

- Bit de paridade

Foi definido pela equipe o conteúdo que cada comunicação serial deveria ter. Segue abaixo o conteúdo dos pacotes recebidos pelo micro e aqueles que são enviados.

- **Protocolo FPGA para micro**

Os pacotes recebidos da FPGA compreendem a maior parte dos dados em funcionamento normal. Uma comunicação completa da FPGA com o PC deve conter todas as informações de uma medição realizada pelo freqüencímetro. Foram adicionados também alguns bits para detectar erros e separar variáveis. Separou-se sempre em grupos de 8 bits (ou seja, 1 byte) para que eles sejam enviados juntos em um pacote serial RS232. O esquema de envio segue abaixo:

- 1 byte de identificação de transmissão: seqüência característica que indique que um conjunto de dados de medição será enviado

- 1 byte de informações: essas informações compreendem 2bits com o ID do freqüencímetro/gravímetro/sinal (para o caso de existir mais de um). 2 bits do modo de operação, 4 bits reservados para uso futuro.

- 1 byte de início da medida

- 4 bytes da medida de freqüência

- 1 byte de fim da medida

- 4 bytes da medida de tempo

- 1 byte de fim de transmissão

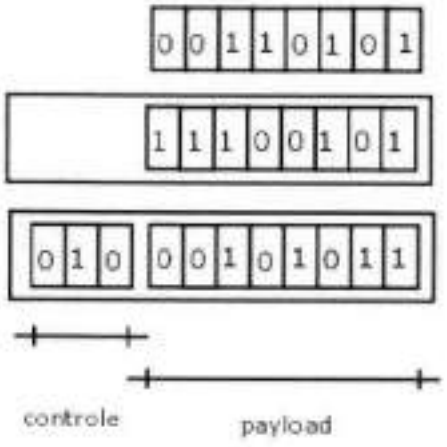


Figura 8: Protocolo / encapsulamento.

- **Protocolo micro para FPGA**

Os pacotes enviados do micro para a FPGA visam a reconfiguração do circuito durante a medição. O protocolo abaixo corresponde a um exemplo básico em que é possível alterar a freqüência de base e o número máximo de contagem por medição (Fb e Cmax, que são valores de 32bits) para um certo sistema de aquisição na FPGA. Abaixo segue o esquema:

- 1 byte de identificação de transmissão

- 1 byte de informações: 2 bits de freq-id, 2 bits modo de operação, 4 bits reservados para uso futuro

- 1 byte de início de dados

- 4 bytes com a Fb

- 1 byte separador

- 4 bytes do Cmax

- 1 byte de fim de transmissão

3.3.2.4 Conversão de sinais e alimentação

Nesta seção será descrito o funcionamento de todos os blocos de conversão de sinais e níveis necessários ao sistema. Sua estrutura geral está ilustrada no esquema abaixo.

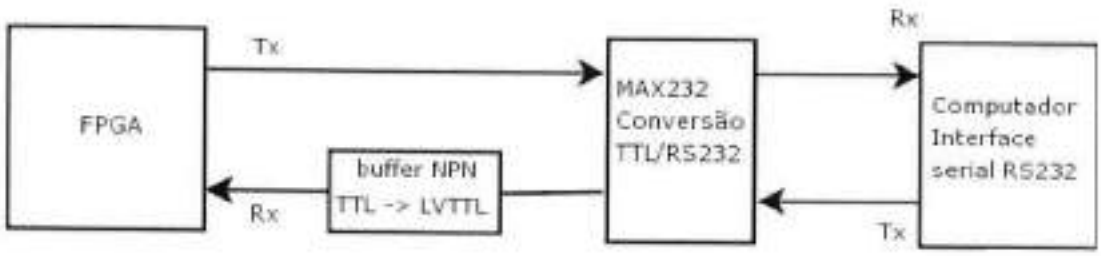


Figura 9: Sistema de conversão de sinais.

3.3.2.4.1 Conversão de níveis TTL / LVTTL / RS232

O sistema do freqüencímetro utiliza componentes auxiliares à placa da FPGA principalmente para a comunicação de dados. Esses circuitos e sistemas auxiliares necessitam de algumas conversões de sinal para o funcionamento.

Nem todos os circuitos auxiliares operam com o mesmo nível de tensão. Os componentes TTL são facilmente encontrados, enquanto que os LVTTL não são tão comuns assim.

Além disso, precisa-se fazer com que a FPGA possa se comunicar com PC em padrão RS232. E para isso é necessário mais um sistema de conversão de tensão.

Na seção sobre o desenvolvimento do projeto, maiores detalhes sobre como obter as transições de tensão e quando elas são necessárias serão tratados. Às vezes a conversão não é explicitamente necessária, por exemplo, no caso de LVTTL para TTL, pois o nível 0 lógico coincide, e o nível 1 do LVTTL, apesar de menor, já é suficiente para o circuito TTL perceber como 1 lógico.

3.3.2.5 Processamento dos dados no computador

3.3.2.5.1 Programa receptor de dados

O programa receptor de dados é um programa de testes muito simples para testar o recebimento e transmissão dos dados. Para testar o funcionamento utilizamos priori um programa de terminal de texto comum, tal como o HyperTerminal no Windows ou o minicom no Linux.

A parte de recebimento dos dados deve basicamente emular o comportamento de um cliente de terminal. O motivo de não ser utilizado simplesmente um cliente de terminal é que um

cliente de terminal sempre converte as seqüências de 8 bits transmitidas em caracteres. Porém nem toda seqüência de bits gera um caractere válido no mapa de caracteres, e é possível que alguns resultados não sejam visualizados.

A parte de envio é codificada individualmente através de variáveis numéricas, representando cada conjunto de 8 bits. Após a codificação, é feita uma transformação de casting (mudança do tipo de variável) para codificar o dado a ser enviado em um tipo de 8 bits. Assim, é possível codificar qualquer seqüência de 8 bits sem o risco de ser necessário utilizar um caractere inexistente ou não imprimível.

O programa deve emular a característica do terminal serial, ou seja, manter um baud rate de 9600, paridade par e controle de fluxo por software. O programa pode ser facilmente implementado em C, acessando o dispositivo de porta serial disponível (ttyS0 no Linux, ou COM1 no Windows). Uma vez que seja possível manter a comunicação serial, programa deve conhecer o protocolo de transmissão dos dados (codificação) para permitir a exibição correta dos dados na tela e o encoding de dados a serem enviados para a FPGA.

4 Metodologia

Esta seção se dedica a explicar um pouco sobre nossa filosofia de projeto, e como foi pensado o desenvolvimento do trabalho de formatura em toda a sua extensão.

4.1 Implementação

"Rule #1: Modularity: Write simple parts connected by clean interfaces.", from The Basics of Unix Philosophy

O projeto, desde a sua concepção foi pensado sempre em termos de módulos, unidades funcionais e interligações.

As palavras chave deste processo são:

- Modularidade
- A união faz a força
- Faça pouca coisa, mas faça bem feito.
- KISS (keep it simple, son)

Muitos dos tópicos relembram a filosofia Unix de desenvolvimento. Hove sempre a busca em manter o código limpo. Manter a depurabilidade do código (habilidade de debugar o sistema).

Os blocos que adicionam funcionalidades foram mantidos com o mínimo necessário. Interligar blocos simples para prover funcionalidades complexas. Manter as interfaces simples ou sistemas complexos encapsulados (como o caso do sistema de divisão).

4.2 Metodologia de Teste

A metodologia de teste se baseia em uma linha que decorre diretamente da filosofia da concepção.

Testamos inicialmente pequenos blocos. Tentamos manter o teste em simulação o máximo possível. Uma vez que vimos que o sistema funciona, adicionamos novos sistemas, anteriores ou posteriores e testamos funcionalidade do dois corretamente. Mesmo que ambos aparentem terem funcionamento correto individualmente, isso não implica de maneira alguma que os blocos vão funcionar bem quando interligados.

Esse crescimento e teste coordenado a partir de uma "semente de blocos" permitem manter o sistema debugável e num ponto funcional a cada instante. Se alguma coisa não funciona corretamente, pode-se começar a investigar o problema na interface entre o último bloco adicionado e o sistema principal estabelecido.

Foi iniciado o desenvolvimento dos sistemas principais da FPGA, e foram sendo adicionados blocos adjacentes, até concluir toda a FPGA. Uma vez que a FPGA esteja funcionando corretamente, foram adicionados os módulos de interface que se comunicam com o sistema principal da FPGA, notoriamente a parte de aquisição e tratamento da entrada, e a parte da comunicação serial do dado processado de saída.

No desenvolvimento dos circuitos "físicos" (ou seja, os circuitos que não são baseados em VHDL), tentou-se manter o mesmo conceito, de ir verificando até que ponto o circuito está funcionando corretamente, e ir adicionando partes até o término do sistema.

4.3 Organização

Este subtópico relacionado à Metodologia de Trabalho foi reservado para falar um pouco das ferramentas utilizadas pelo grupo para facilitar a dispersão da informação e organizar os dados de forma compreensível. O uso destas ferramentas permitiu um considerável aumento de produtividade, além de facilitar, em geral, a organização do grupo.

4.3.1 Ferramentas de apoio

Geralmente a principal ferramenta utilizada para organizar a comunicação e informação do grupo é o contato por email. Além do email, buscou-se organizar as informações disponíveis em um lugar unificado, de fácil acesso para todos e que estivesse constantemente atualizado.

Além disso, houve uma certa dificuldade na organização das várias informações que necessitam ser compiladas em um relatório. Por isso, foi testado no fim do projeto, o uso de uma ferramenta para organizar idéias, brainstormings.

Como são ferramentas interessantes, serão citadas aqui algumas idéias que ficam também como sugestão para quem precisar gerenciar alguma situação semelhante.

4.3.1.1 Wiki

O wiki é uma espécie de site http, em que qualquer visitante possui a capacidade de alterar o conteúdo. Isso permite agilidade na troca de informações, uma facilidade para que em algum momento que se acesse o wiki, tenhamos os dados mais atualizados.

Por exemplo, neste projeto foram mantidos no wiki dados como cronograma, contatos realizados para o TF, todas as documentações e links utilizados, uma área de armazenamento com datasheets, manuais e referências úteis, além de vários outros dados.

Quando era necessário obter alguma informação sobre o wiki, por exemplo, o que foi colocado na última apresentação de trabalho de formatura, ou qual o link para um exemplo de utilização de freqüencímetros, todos do grupo sabiam que aquela informação estaria no wiki, não sendo necessário perder tempo com uma busca novamente.

É necessária uma certa organização no uso da ferramenta, e também um comprometimento do grupo em atualizar o wiki com os dados, o que pode ser até uma tarefa cansativa, mas que traz bons retornos.

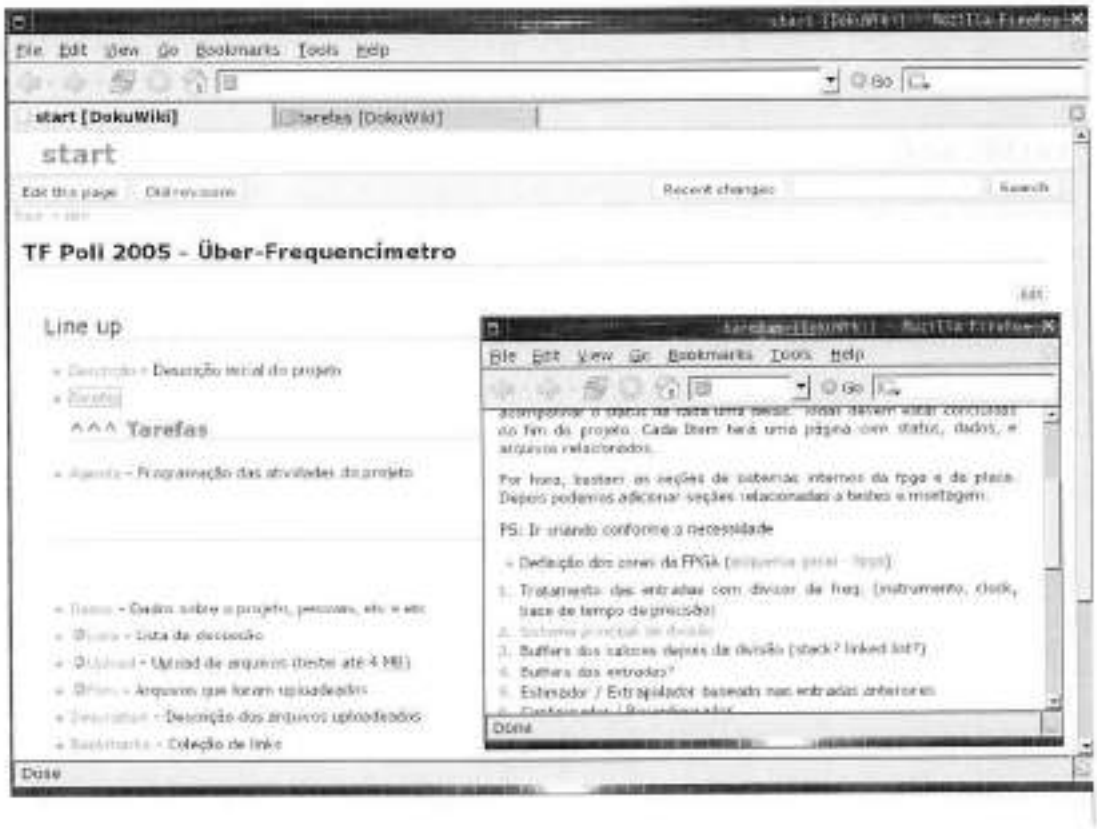


Figura 10: Tela principal do nosso Wiki.

4.3.1.2 *Kdissert/view-your-mind*

O Kdissert é baseado numa técnica chamada "View your mind", mas adaptada para a produção de documentos, tais como dissertações (daí o nome kdissert).

O funcionamento do programa baseia-se em criar caixas com idéias relacionadas ao documento e interligá-las hierarquicamente até gerar uma estrutura complexa que represente o que se quer expor no documento. É possível adicionar notas e textos em cada tópico, anexar documentos links, figuras, etc. Cada link possui também uma indicação de estados (concluído, com problemas, idéia, planejado, etc).

Assim a partir de idéias iniciais pode-se moldar o documento de acordo com as necessidades, e modificar seções inteiras através do religamento de alguns vínculos, de uma forma espacial, o que não seria possível (ou seria muito mais difícil) com uma estrutura de dados seqüencial, como num processador de texto comum.

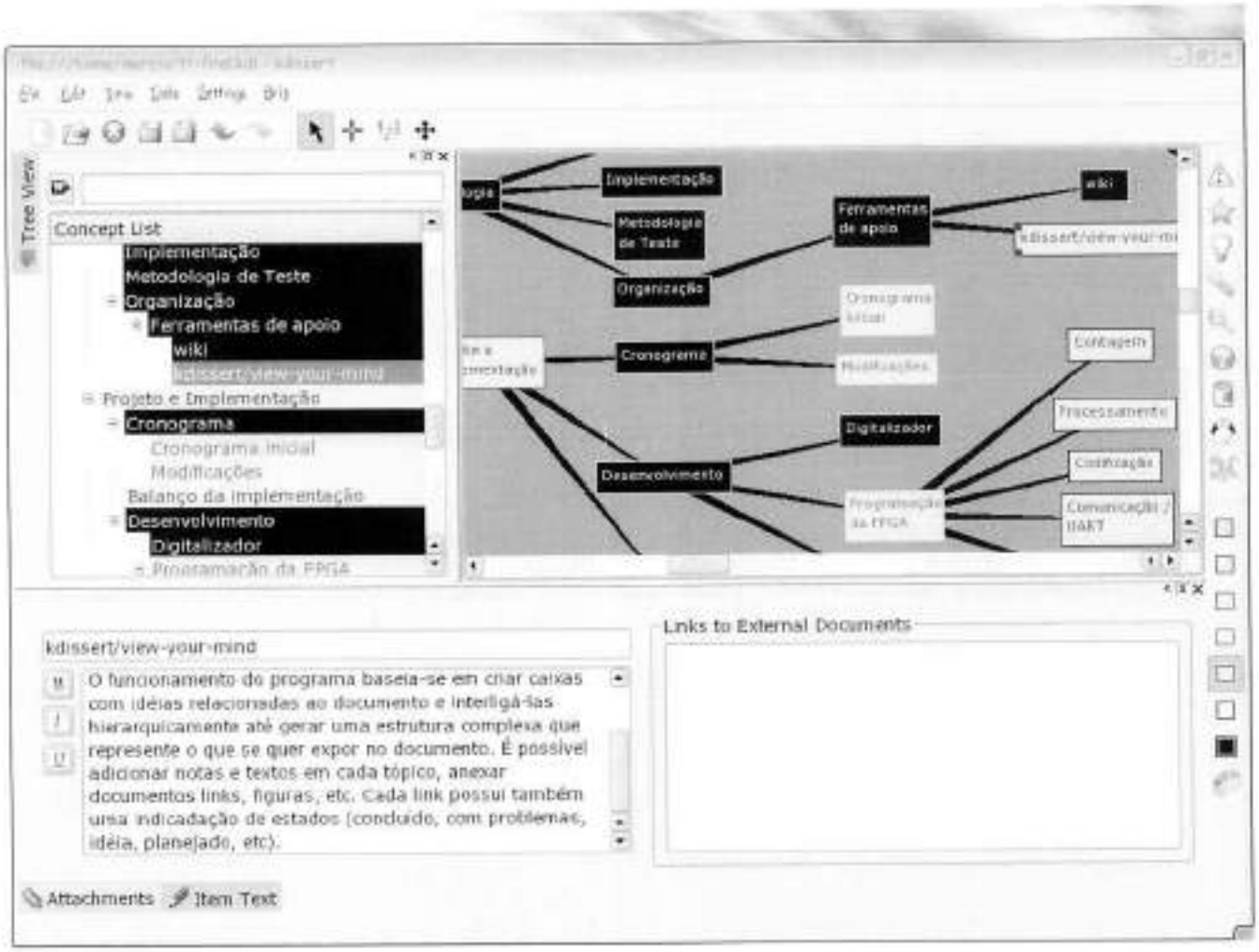


Figura 11: Tela principal da ferramenta Kdissert/view-your-mind

5 Projeto e Implementação

Nesta seção serão detalhadas todas as ações realizadas para a implementação propriamente dita do projeto. Desde as decisões quanto a sua concepção quanto todas as fases da implementação de todos os seus subsistemas. Serão explicados desde testes iniciais de ferramentas importantes utilizadas até os últimos passos do desenvolvimento do sistema final, culminando com uma análise completa de todo este processo.

5.1 Pesquisa e concepção

Neste item serão descritas e explicadas várias etapas de pesquisa, experiência e estudo realizadas para a escolha e familiarização com as ferramentas, componentes e demais tecnologias utilizadas no desenvolvimento de todo o sistema. Além disto, serão exploradas também as decisões fundamentais da concepção do sistema conseqüentes destes trabalhos.

Quando foram definidos os requisitos de projeto junto com os orientadores, concluiu-se que o problema deveria ter uma abordagem diferente dos processadores de sinais convencionais. A problema da medição em alta velocidade necessitaria de um hardware especial para que o problema fosse resolvido. Uma solução comercial pronta do tipo COTS (commercial off the shelf system) certamente não seria adequada.

A aquisição de dados de alta velocidade geralmente é feita utilizando hardware dedicado sob medida. A flexibilidade de usar a FPGA para esse design torna essa opção muito atraente para esse tipo de projeto, além do que geralmente não se fazem largas quantidades para justificar

um design de um chip dedicado. É possível corrigir falhas e fazer upgrades com a simples troca de uma memória ROM.

A escolha do design em FPGA pareceu bem lógico para o desempenho esperado do circuito. Apesar do circuito não ter um throughput de medições muito alto ou exigente, manter o padrão da medida exige um circuito muito rápido, com clock elevado.

Estas foram as principais decisões quanto à concepção do projeto. As demais serão tratadas ao longo dos próximos tópicos.

5.1.1 Familiarização com o ISE 7.1 / ModelSim MXE

Foi utilizado um simulador de design HDL chamado ISE Simulator 7.1i. Ele é produzido pela Xilinx Inc., mesma empresa que produz a FPGA, e basicamente funciona como uma IDE de programação para a FPGA em HDL (VHDL, verilog, etc). Além disso, existe um programa auxiliar chamado ModelSim MXE, que realiza simulações de forma de onda (waveform sim) para o circuito.

Utilizando o ISE Project, o MXE e uma placa de desenvolvimento da (placa de I/O Digital + FPGA), temos um ambiente completo para desenvolver, testar e simular aplicações em FPGA descritas em HDL.

Os programas são complexos e gerenciam todas as partes de um projeto em FPGA. Basicamente, podemos dividir o processo de gerar um programa em HDL com a IDE nas seguintes partes:

- **Fazer a descrição do hardware em HDL ou Schematics**

- **Parsing e compilação do código**
- **Gerar bloco funcional**
- **Verificar restrições de tempo de propagação e freqüência máxima de operação**
- **Síntese de baixo nível baseado na biblioteca de blocos funcionais disponíveis**
- **Verificação da pinagem do circuito**
- **Placing e routing dos sinais (alocação física e interligação de blocos)**
- **Gerar fluxo de dados de programação para PROM ou para download na FPGA**

Começamos nos familiarizando com o projeto e simulação com este sistema, além de nos acostumarmos com a programação em VHDL e verilog, e desenvolvendo alguns testes nos mesmos. Um ponto importante é que não é feita a melhor otimização do circuito, pois esta é uma versão gratuita do programa.

Foram realizados vários procedimentos para familiarização com estas ferramentas. Em um deles foi utilizado um pequeno código padrão em VHDL de um divisor padrão IEEE, disponível gratuitamente pela comunidade OpenCores (http://opencores.org), para testar o projeto na IDE (Integrated Development Environment) ISE 7.1i.

Podemos ver abaixo um diagrama genérico de blocos de um sistema divisor padrão utilizado:

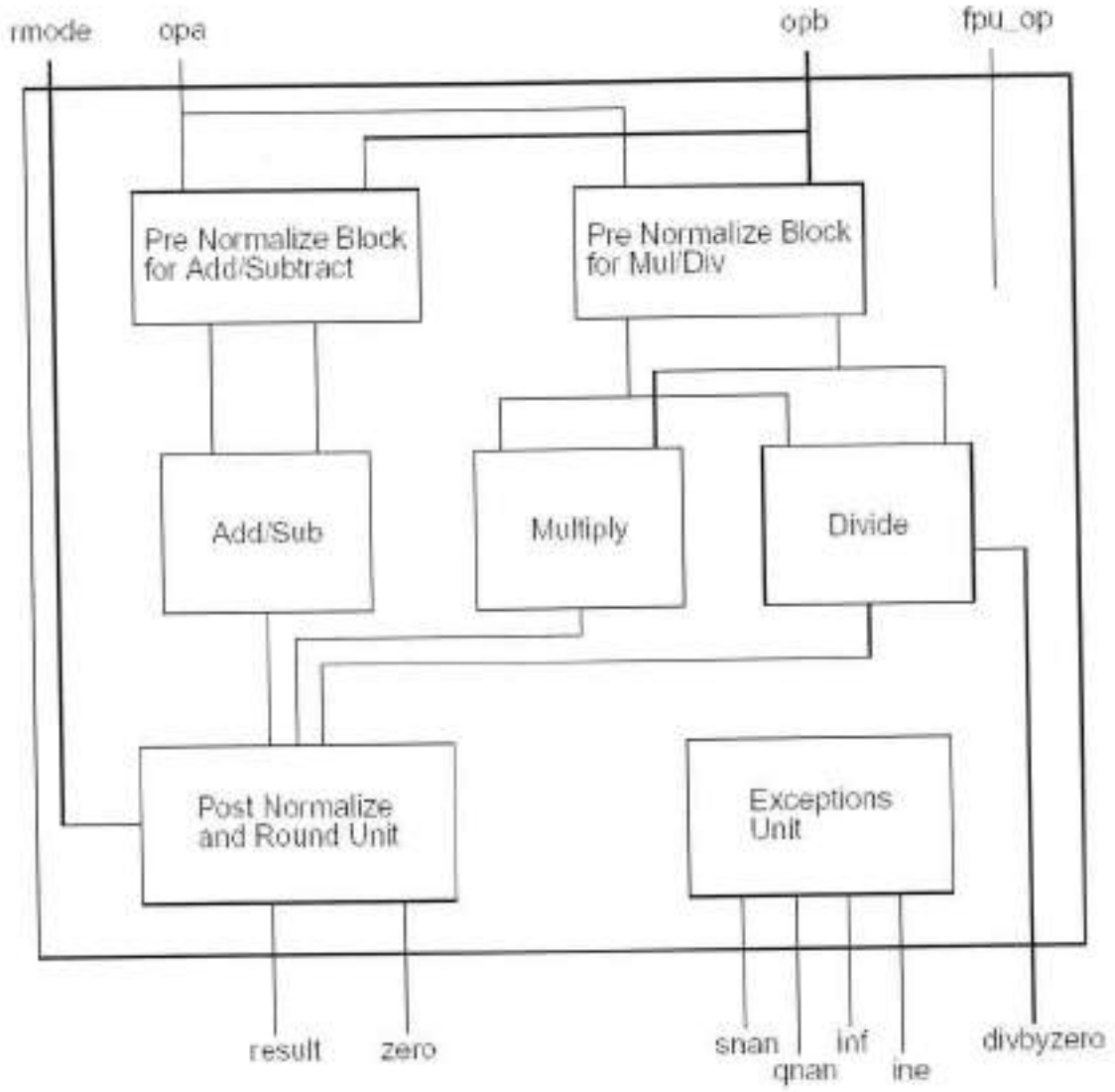


Figura 12: Sistema de divisão usado para teste.

Pôde-se simular o circuito no simulador de waveform ModelSim MXE, testar o funcionamento do circuito e também se familiarizar com o uso do simulador de waveform.

Pôde-se ainda verificar no ISE restrições de projeto, ou criar restrições de tempo de alguns sinais, máxima freqüência de operação, etc, e verificar na simulação de waveform que os requisitos estão sendo atingidos, ou se necessitam de adaptações. O processo completo de teste e geração do código, desde o VHDL até a FPGA pode ser verificado no anexo 1.

5.2 Familiarização com a placa Digilent D2SB/DIO4

Foi utilizada uma placa de desenvolvimento para a FPGA Spartan IIe, emprestada do LSI – USP. A placa contém uma FPGA e seu circuito auxiliar e uma placa de I/O Digital.

Utilizamos o ISE para baixar alguns programas para testar a placa e o funcionamento dos LEDs, display, switches push-buttons. Verificamos que felizmente a placa estava funcionando, e o sistema que faz a gravação na FPGA também funciona corretamente. Este sistema é de alta relevância, pois algo semelhante deve ser feito posteriormente para carregar o programa de uma PROM, ao invés do computador. O programa utilizado também é o ISE 7.1i, e depois de um pouco de problemas com a documentação complicada, conseguimos realizar todos os testes com sucesso.

Outro ponto é que existe uma versão Unix (Linux e Solaris) deste programa, mas ela simplesmente não funciona, apresenta várias falhas, e como possuímos licença gratuita para estudantes, não podemos contar com o suporte. Problemas semelhantes aos nossos estão abertos nos fóruns da Xilinx sem resposta há anos. Notamos uma certa falta de interesse e de usuários dessas versões, dado o número de problemas encontrados, que impossibilitaram o uso da versão

para Linux, infelizmente. Outros problemas com as licenças foram encontradas no decorrer do uso dos programas ISE e MXE, acabando por vários dias com a produtividade dos programadores deste projeto. Apesar disso, pode-se dizer que acabamos nos familiarizando com a maioria dos problemas encontrados no uso dos programas, e conseguimos evoluir com os problemas de codificação em VHDL e alguns códigos em verilog.

Rotina de testes realizada:

- **Instalação dos programas**
- **Teste do ISE 7.1i**
- **Teste do ModelSim MXE**
- **Teste de síntese e compilação de VHDL e Schematic**
- **Teste da placa principal**
- **Teste da placa de I/O Digital**
- **Teste de interação switches/push-buttons**
- **Testes no simulador de wave-form**
- **Teste de um core de FPU**

A placa digital é composta por uma parte com a FPGA e as entradas e uma placa de I/O digital. O desenho da placa é dado abaixo, junto com os I/O's digitais providos pela placa auxiliar. Os conjuntos de conectores C1 e C2 da D2SB encaixam nos conjuntos P1 e P2 da DIO.

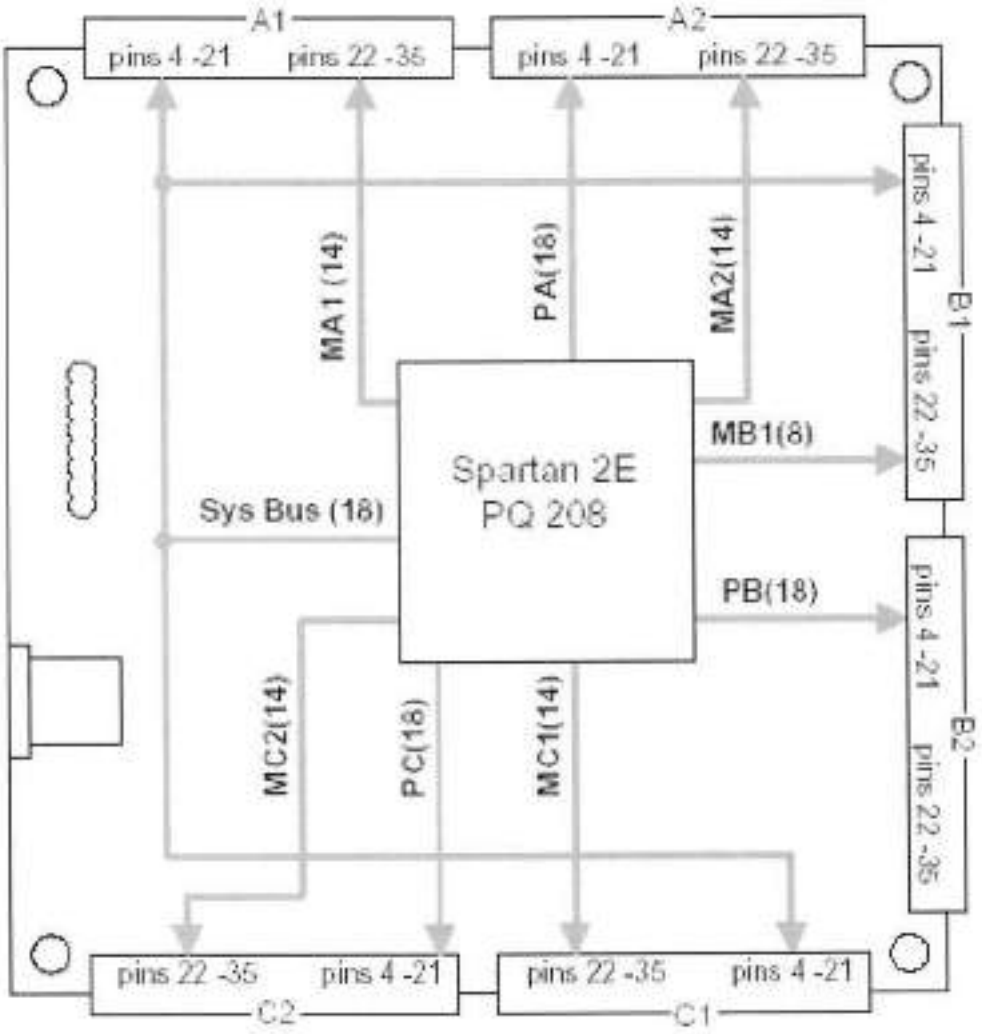
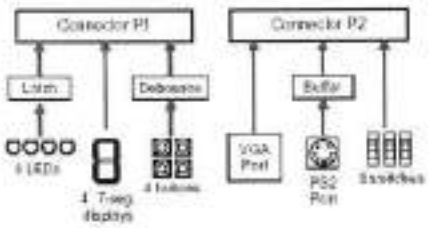


Figure 4. Expansion Connector Signal Routing



DIO4 circuit board block diagram

Figura 13: Esquema das ligações da FPGA no kit de desenvolvimento.

5.2.1 Estudo dos algoritmos de divisão e modelos

Antes de implementar os circuitos, principalmente à parte do circuito divisor, estudamos alguns algoritmos que poderiam ser utilizados:

- **Método Simples (subtrações sucessivas);**

O método simples não tem nada de muito especial, como o próprio nome sugere. Este método consiste e se realizar subtrações sucessivas do valor do dividendo no divisor, convergindo para o resultado até esgotar o valor do divisor, e a diferença é o resto e o número de operações executadas é o quociente.

- **Divisão bit a bit;**

Este método consiste em utilizar operações bit a bit entre dividendo e divisor para calcular o resultado. A idéia é semelhante ao anterior, mas as operações são realizadas entre binários, e com isso ele é mais eficiente.

- **Método Radix 8;**

O método Radix é uma evolução do Radix 4, que é o algoritmo utilizado nas unidades aritméticas dos processadores Pentium, da Intel. É um algoritmo interativo que consiste em dividir alguns casos em que é possível extrair mais informações do que numa divisão bit a bit sem nenhuma otimização. Através das informações dos últimos restos é possível determinar casos de execução mais otimizados.

- **Divisão de ponto flutuante padrão IEEE.**

Gostaríamos de simular um divisor padrão de uma biblioteca de VHDL do IEEE, para efeitos comparativos em relação aos outros métodos, e verificar eficiência de cada um para os tipos de entrada que vamos ter no freqüencímetro. Este era um estágio para quando tivéssemos os algoritmos em simulação.

5.2.2 Estudo de alternativas para otimização de desempenho

Outro ponto que é importante notar é a filosofia de implementação no hardware. Existem otimizações de projeto e construção que podem influenciar muito no desempenho do circuito.

Duas que estudamos foram o efeito dos algoritmos single clock versus multi clock (detalhados posteriormente) e da implementação dos estágios do divisor através da implementação de um Pipeline.

A implementação em Pipeline é um conceito simples, mas que otimiza o funcionamento do circuito. Basicamente a idéia do pipeline é manter o circuito ocupado o máximo possível, e para isso a informação te que se manter fluindo entre os vários estágios do circuito. Um exemplo disso seria no circuito do freqüencímetro a implementação de pipeline entre o bloco divisor e o bloco de saída usb. Um dado é calculado no divisor e imediatamente enviado para um buffer ou registrador de armazenamento. No ciclo de operações seguinte, enquanto o divisor gera outro resultado, o bloco de saída já envia o dado anterior e prepara-se

para o próximo, ou seja, não ocorre de um dado ser computado, e depois ser enviado e só então começar um novo ciclo de funcionamento. Existiam várias idéias sobre como implementar uma arquitetura pipeline no projeto, ou criar circuitos extras que ajudem no paralelismo (já que são múltiplos instrumentos enviando dados ao mesmo tempo).

Esta outra otimização também envolve diretamente a área disponível para o circuito na FPGA. Um circuito projetado de forma correta em uma simulação pode se comportar diferente quando sintetizado em uma FPGA, sujeita a restrições de área, time constraints, caminho crítico, número de entradas/saídas, etc.

5.2.3 Melhorias no processo de medição

Retomando os métodos de medição, não podemos utilizar simplesmente uma medida simples pelo método I, pois isso nos daria uma medida muito ruim. Precisamos levantar os parâmetros do circuito que aceitam modificações, para que possamos melhorar o desempenho.

O método I parece não poder fornecer o melhor resultado possível. Por exemplo, obviamente alguma pessoa observando a situação por 1 minuto notaria que é melhor fazer que a medição dure mais que um período.

Porém, o caso apresentado tem o seu valor, pois serve como base para discutirmos onde podemos modificar o método. Além disso, em um caso mais comum, aquele método poderia ser suficiente, e também possui um valor máximo de *throughput* de medições, e por isso devemos balancear o quanto deve ser modificado e onde para obtermos um bom resultado no final.

Como melhorar o processo de medida:

- **Usar pre-scaling**

Pre-scaling é uma técnica utilizada em circuitos de alta freqüência que de forma parecida pode ser útil. Não porque medimos em alta freqüência, mas sim porque o intervalo entre medições deve ser pequeno.

O pre-scaling do sinal de entrada poderia diminuir o erro relativo de +/- 1 período. É neste caso equivalente a deixar a medição por mais períodos.

- **Detecção da borda de descida**

Assumimos o erro de +/- 1 período no circuito digital, mas podemos melhorar a estimativa. Podemos também detectar a borda de descida do circuito. Com isso, a incerteza estaria confinada a +/-0.5 período. Podemos pensar que aumentamos a granularidade do circuito para metade do período (pois pela última borda podemos detectar se a medição terminou em alta ou em baixa, isso é trivial em VHDL).

- **Aumentar o clock do sistema**

Aumentar o clock garante que o período de incerteza seja cada vez menor. Apesar de altamente eficaz com 100% de certeza, é o mais delicado parâmetro a ser mexido, pois como sempre, os recursos computacionais são finitos.

Estávamos pensando em utilizar um clock de até 80MHz para o circuito. Nossa FPGA (Xilinx Spartan IIe) suporta circuitos complexos de até 300MHz. Podemos ficar muito mais próximos da margem de tolerância, e passar o circuito para 180–200MHz. Essa é uma boa freqüência porque existem OCXO's com essa freqüência que poderiam fornecer o clock do sistema. Podemos operar com um clock elevado, mesmo com um circuito complexo, desde que não se aumente muito o número de I/O's na FPGA. O desempenho se perde mais rapidamente com o número de I/O's do que com a complexidade do circuito.

Outra coisa a ser considerada é que essa FPGA é relativamente antiga. No site da Xilinx, por exemplo, só é encontrada uma versão nova dessa FPGA (a Spartan III), que deve ter

melhores recursos, que se necessário for, poderia ser utilizada. Além disso, a série Spartan é uma família "low-end" da Xilinx. Se realmente fosse muito necessário, podemos ainda passar para a família Virtex, que é a FPGA de alto desempenho da Xilinx. O ideal seria manter a Spartan IIe, pois possuímos um exemplar, para testes.

- **Uso de cristal de maior freqüência**

Como foi dito anteriormente, podemos aumentar a freqüência do clock do sistema utilizando um oscilador de maior freqüência. Estávamos planejando um oscilador impreciso comum de 20MHz (com um multiplicador de 4x) para o clock, e um sinal de OCXO preciso para medição. Podemos substituir ambos por um OCXO de valor mais alto (algo próximo de 200MHz), e existem OCXO's com essa freqüência. Isso permitiria a melhora na medição, também.

5.2.4 Implementação single clock X multi clock

Os circuitos geralmente são implementados de uma de duas formas: a single-clock e a multi-clock. O circuito single clock realiza toda a sua operação em apenas um ciclo de clock, enquanto que o multi-clock necessita de vários ciclos de operação para que a operação que ele realiza seja completada.

Cada uma das abordagens possui vantagens de desvantagens. Um circuito single clock é simples de se implementar, mas geralmente é oneroso na quantidade de portas lógicas, além de limitar o circuito a valor baixo de clock, obviamente, já que teremos mais operações sendo realizadas em um menor período de tempo.

O circuito multi-clock tem um planejamento mais complexo, e como vimos, demora vários ciclos de clock para poder dar um resultado. Isso significa esperar um tempo maior pelo

resultado, porém, podemos usar clocks mais elevados, já que esperamos operações simples a serem realizadas em cada ciclo de clock. Além disso, um sistema multi clock pode utilizar pipeline, ou seja, se for possível dividir o processo em sub-operações seqüenciais, podemos antes mesmo que um dado seja inteiramente processado, começar o processamento de outros dados, o que aumenta o throughput do sistema. Tratar de tantas variáveis tem um custo, que é a complexidade do circuito.

No nosso caso, como escolher entre a simplicidade de um circuito single clock ou projetar um multiclock? E qual funcionaria melhor? Essas questões são muito complexas. No nosso caso específico o throughput de saídas de medição não é muito elevado, porém o processamento de sinal necessita do maior clock possível, para diminuir o erro intrínseco de medição.

O cenário parece perfeito para a utilização de um circuito multi clock, porém a complexidade do circuito poderia comprometer o desempenho ou roubar muito tempo (precioso) de outras atividades. O ideal seria manter o projeto direto em single clock, e conciliar este projeto com problema de desempenho do circuito.

Felizmente isso é possível e outra idéia do prof. Wang é implementar o circuito com clocks variáveis. Assim poderíamos ter a aquisição de dados e o controle crítico de dados na maior velocidade possível e outras partes do circuito de baixo desempenho podem ter o projeto simplificado através de implementações single clock.

Isso é possível no nosso caso devido justamente à especificidade do tipo de medição. Para reduzir a significância do erro da medição necessitamos manter a contagem de medição por vários períodos de clock, e somente após todo o processo estar concluído é que processamos o resultado da aquisição. Assim, enquanto é feita uma medição temos muitos ciclos de clock ociosos que podem ser aproveitados.

Através de um divisor de freqüência podemos gerar clocks com maior período para partes específicas do circuito que não poderiam utilizar um clock de alta freqüência. Poderíamos, por exemplo, manter um divisor single clock lento, mas simples, que opera em baixa freqüência processando a última leitura enquanto se faz a aquisição da próxima.

5.2.5 Trade-off de desempenho

Para exemplificar o problema de desempenho do circuito que foi exposto acima, vamos mostrar uma equação que ilustra o desempenho de sistemas de arquitetura de computador CISC x RISC, que é análogo ao nosso problema de clocks e desempenho de circuito.

O problema de desempenho típico em hardware digital consiste em medir o desempenho, do circuito. Talvez uma das mais claras figuras de mérito para decidir se o circuito é bom é sua capacidade de entregar respostas rapidamente, ou seja, o chamado "throughput". O throughput é uma combinação de diversos fatores, e representa basicamente quantos programas eu consigo executar em um certo tempo.

Considerando que o tamanho do programa, ou seja, o que o circuito terá que fazer é basicamente constante, independem do método de implementação, temos que atacar a freqüência de operação do clock e o número de instruções. Como vimos, o desempenho do circuito digital está ligado à quantidade de ciclos que se pode executar, a quantidade de operações que se pode realizar em um ciclo e, claro, a quantidade de operações necessárias para que a minha tarefa (programa) seja concluída.

Não é nenhuma surpresa que nenhum desses fatores seja independente um do outro. Balancear as alterações em cada um é a chave para obter o melhor desempenho no projeto. Uma

alteração brusca em um dos quesitos pode parecer uma boa idéia, mas pode afetar bastante um outro quesito e não dar nenhum ganho de desempenho.

5.2.6 Montagem da placa

Pesquisamos sobre como fazer a montagem das placas. Quando se fabrica em escala, é possível realizar testes em FPGAs e então quando o projeto está maduro, gerar as máscaras e o chip ASIC para produção em larga escala. Porém, em projetos de poucas unidades isso seria custoso demais. Ao invés disso, a placa final utiliza a própria FPGA, o que se enquadra no nosso caso. Com o desempenho cada vez melhor das FPGAs isso não chega a ser um problema na maioria dos casos hoje em dia.

Apesar disso, um problema razoavelmente sério consistiria em como conectar a FPGA na placa, sem comprometer a FPGA em caso de um erro ou falha na placa. Isso é muito pertinente, pois uma FPGA como a nossa atual (Xilinx Spartan IIe) possui mais de 200 pinos de várias funções. Isso impossibilita a solda manual, e soquetes DIP em geral. Não seria indicado também soldar diretamente a FPGA em uma placa, pois se ela sofresse algum dano, a FPGA (que é de longe o componente mais caro na placa) ficaria inutilizada.

Uma solução encontrada foi através da empresa Aries Electronics. Esta empresa faz soquetes, adaptadores e todo tipo de encaixes, soquetes e conexões de chips de vários tipos, inclusive com projetos personalizados. Alguns dos adaptadores ajudam a resolver falhas de projeto, como a falta de espaço para um dissipador em um chip, ou um chip colocado de maneira errônea pode ter suas saídas deslocadas com um shift determinado (por exemplo, de forma que o pino 1 de saída corresponda ao pino 8 de um chip).

Este fabricante faz inclusive soquetes especiais para FPGAs, o que é extremamente interessante para o caso. Abaixo temos alguns exemplos dos produtos:

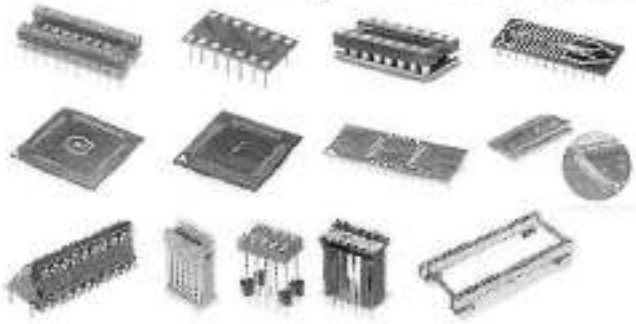


Figura 14: Exemplos de adaptadores fabricados pela Aries Eletronics.

Como será explicado em outro tópico, acabamos por decidir não fazer a manufatura da placa agora, mas em um estágio futuro do projeto. As complicações na aquisição e importação de componentes e alguns outros fatores mostraram que era preferível focar no desenvolvimento da solução agora e depois então se preocupar com a montagem da placa.

5.3 Desenvolvimento

Na seção de desenvolvimento vamos retomar muitos dos tópicos já relacionados na seção de características funcionais e especificação. Além de retomar as idéias especificadas, acrescentaremos também como foi implementado cada módulo.

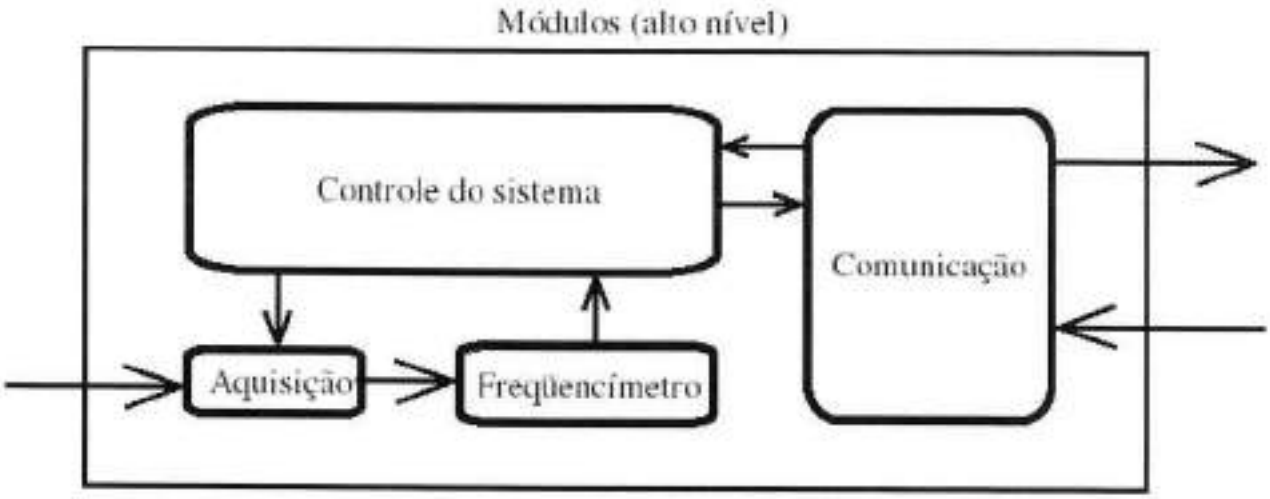


Figura 15: Principais módulos desenvolvidos.

5.3.1 Digitalizador

O sistema digitalizador, que processa o sinal de entrada da FPGA (que é também a saída do gravímetro) é implementado utilizando um CI LM 339, que é um CI que empacota comparadores baseados em Amp-Op's.

Basicamente utilizamos um dos Amp-Op's do LM339 em aberto, com uma das entradas diferenciais aterradas, e a outra é utilizada para entrada do sinal.

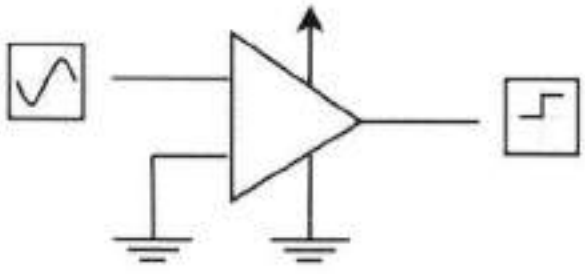


Figura 16: Esquema genérico da "digitalização" do sinal.

Na configuração de comparador é comum, conforme sugestão do datasheet, foi colocado um resistor próximo de 3KOHMs na saída.

Outra peculiaridade do circuito é que esses Amp-Op's não necessitam de alimentação simétrica, coisa que no nosso circuito também não é necessária. Além disso, podemos utilizar como tensão de alimentação valores que vão de 2 a 36 volts. Assim, é possível compatibilizar a saída do CI com sistemas TTL, MOS, CMOS, etc. Utilizamos uma alimentação de 3.3V. Deste modo, a saída já será compatível com o circuito LVTTL da FPGA.

Este circuito converte uma onda senoidal de amplitude 3V e offset nulo em uma onda quadrada com níveis 0V e 3.3V, ou seja, eliminando o semiciclo negativo e saturando o semiciclo positivo no valor da tensão de alimentação.

5.3.2 Programação da FPGA

A FPGA é o cérebro do circuito, e onde foi colocado a maior parte do tempo de projeto. Vamos explicar mais um pouco sobre o trabalho na FPGA, e cobrir os tópicos ainda não completamente cobertos na fase de projeto citada anteriormente.

Inicialmente há uma parte introdutória sobre como foi feito o projeto e alguns instrumentos de teste. Depois entramos em alguns detalhes sobre os módulos.

5.3.2.1 A arquitetura UC/FD

O Fluxo de Dados compreende os componentes para o processamento da informação, como, por exemplo, divisores, multiplicadores, somadores, contadores, comparadores etc, bem como de blocos de memória (registradores) e roteamento (multiplexadores), que serão descritos

em primitivas básicas do chip em questão, através de bibliotecas específicas que o fabricante mantém no ambiente de programação para sua linha de produtos. A síntese e simulação pode ser feita com elementos genéricos, mas a pós síntese as próximas fases devem ser implementadas com as bibliotecas do componente, para que sejam computados parâmetros de desempenho, restrições de tempo de execução, quantidade de sinais, pinagem, etc.

Optamos por implementar em VHDL cada componente separadamente e interconectá-los na descrição estrutural (poderiam ser utilizadas outras descrições), que compõem o Fluxo de Dados. Podemos facilmente detectar erros de design ou programação e temos uma descrição "limpa" do funcionamento. Podemos posteriormente realizar otimizações em diversos estágios da programação para deixar o circuito mais eficiente.

Já a Unidade de Controle, que implementa a máquina de estados do algoritmo, é responsável por gerenciar os dados que estão sendo processados na FD através da troca de sinais de controle e de saída com FC. A UC deve enviar sinais para seleção de entrada, reiniciar processo, enviar saídas, carregar dados, etc. A FD envia sinais de status para a UC, tais como a saída do processamento, que indica que a UC já pode ordenar o processamento de novos dados.

5.3.2.2 *Programação e testes*

O próximo passo depois da descrição do circuito é prepará-lo para ser enviado para a FPGA. Este foi o segundo ponto em que trabalhamos. Gostaríamos de testar a placa de desenvolvimento processando sinais reais e enviando saídas em níveis de tensão reais. Para tanto, possuímos uma aparelhagem no estágio de um dos elementos do grupo com uma instrumentação suficiente para o teste do circuito.

Para o teste utilizamos:

- Osciloscópio;

- Gerador de sinal;

- Placas de aquisição de baixa latência;

O sistema de aquisição e o software já estavam disponíveis e integrados, e a operação já era familiar para um dos elementos do grupo. Pudemos nos preocupar o mínimo possível com a instrumentação ao redor da placa de desenvolvimento e focar nos problemas da placa de testes da FPGA.

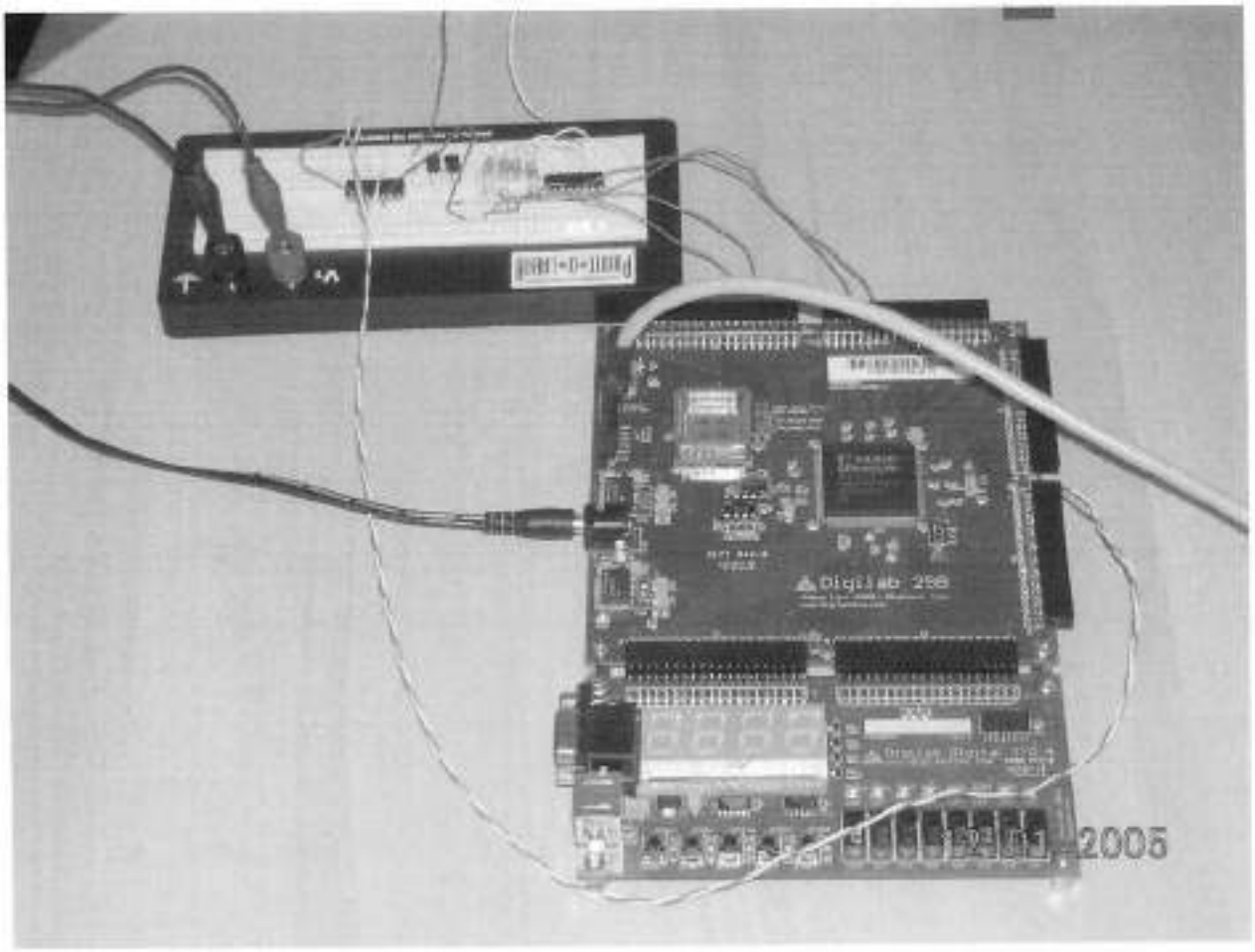


Figura 17: Foto da placa de desenvolvimento utilizada, suas conexões e o circuito auxiliar desenvolvido.

5.3.2.3 Sistemas desenvolvidos na FPGA

Abaixo estão os esquemas de alguns dos sistemas desenvolvidos na FPGA.

5.3.2.3.1 Sistema completo

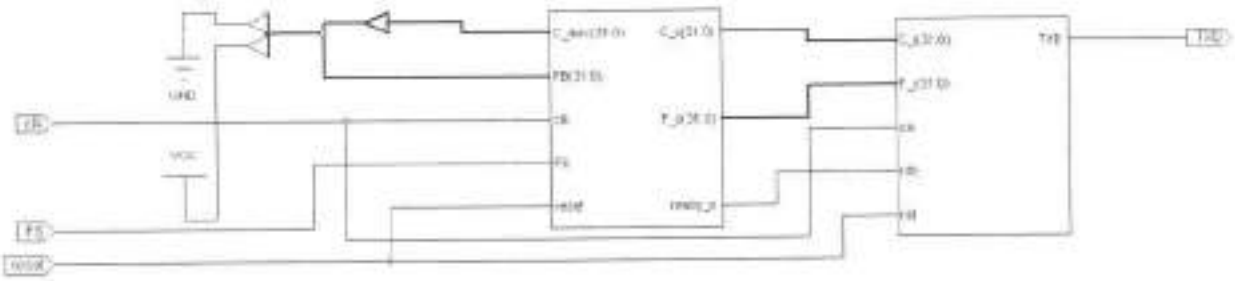


Figura 18: Esquema do sistema completo

5.3.2.3.2 Freqüencimetro

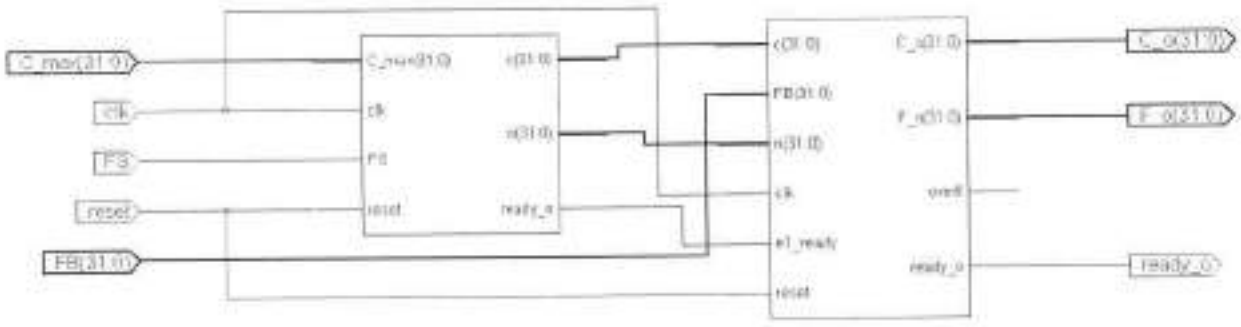


Figura 19: Esquema do freqüencimetro desenvolvido.

5.3.2.3.3 Contagem

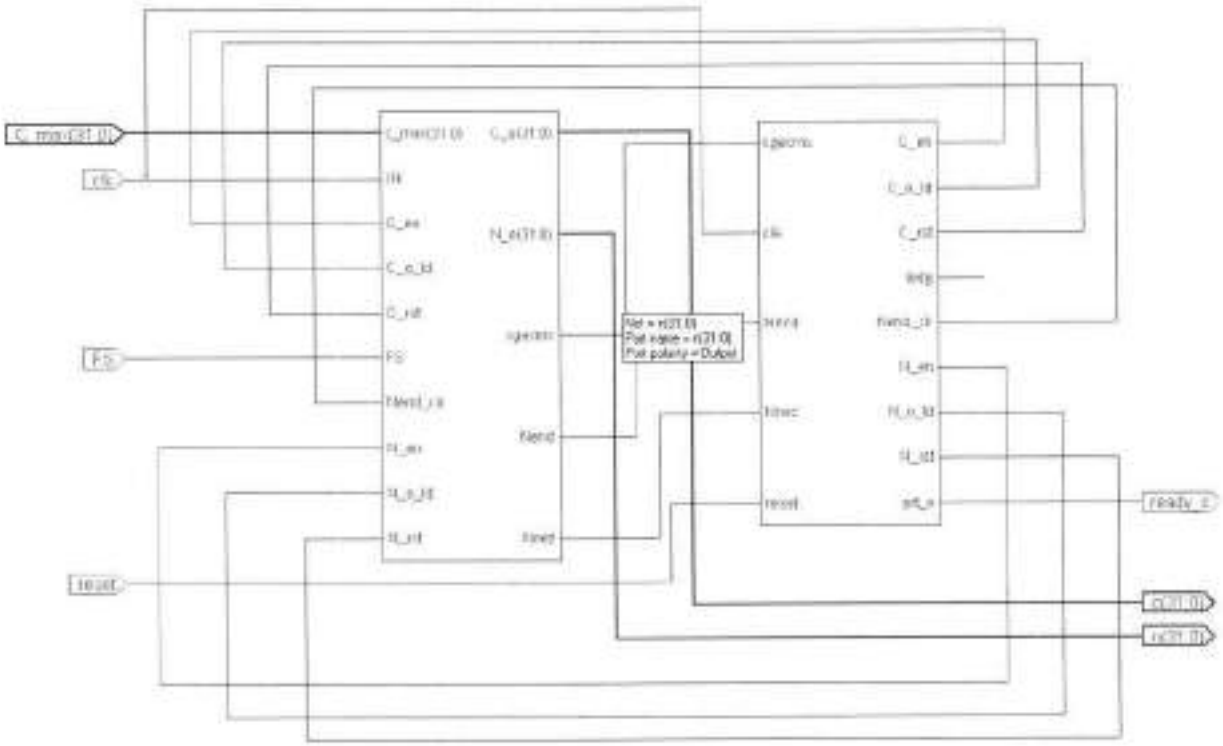


Figura 20: Esquema do sistema de contagem desenvolvido.

5.3.2.3.4 Processamento

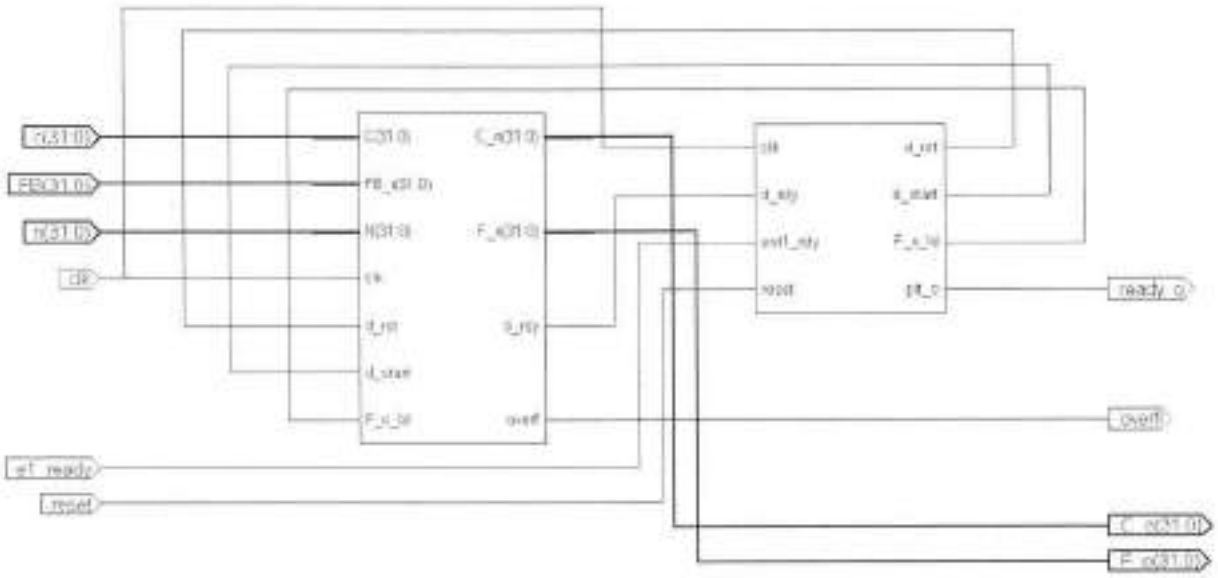


Figura 21: Esquema do sistema de processamento desenvolvido.

5.3.3 Conversores de tensão

Enquanto construíamos os circuitos auxiliares da placa de desenvolvimento, acabamos nos deparando com a necessidade de implementação de dois pequenos circuitos de conversão de nível. As abordagens utilizadas estão descritas a seguir, nos tópicos.

5.3.3.1 TTL para LVTTL

O circuito da FPGA utiliza níveis de tensão LVTTL, ou seja, o nível High desse circuito é aproximadamente 3.3V, enquanto que o nível Low é por volta de 0V.

Porém, os circuitos adicionais são de nível TTL comum, ou seja, com nível baixo de 0V e alto de 5V. Isso se deve ao fato que os componentes TTL são de fácil aquisição no Brasil, e em baixa quantidade. Assim, é mais fácil utilizar um conversor de nível TTL para RS232 (serial) do que tentar obter um CI semelhante para o nível LVTTL.

A conversão de nível de LVTTL para TTL não é obrigatória, pois um sinal em High de LVTTL também é considerado High no circuito TTL (o nível baixo, de Zero lógico, é coincidente). Porém a volta não é verdadeira. Um sinal em alto da parte TTL pode queimar o circuito LVTTL da placa de desenvolvimento, já que a mesma possui um limite máximo de tensão de 4,0V.

Então, o que fizemos foi colocar um circuito intermediário que quando recebe um nível lógico alto em TTL converte o nível para tensões de LVTTL. Isso é feito através de dois transistores NPN que funcionam como buffer, e isolam as regiões de tensão diferente.

A solução foi utilizar dois transistores BC458 npn, que funcionam como inversores. Como eles são alimentados com 3,3V quando a entrada está no máximo, a saída vai estar no máximo com esse valor, que é o do Vcc, como foi dito.

Abaixo segue o diagrama de funcionamento desse circuito:

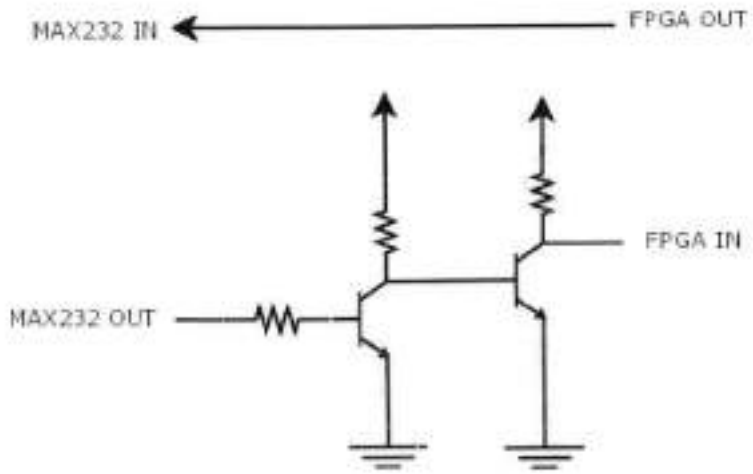


Figura 22: Circuito elétrico da conversão TTL/LVTTL

5.3.3.2 Conversão TTL <-> RS232

O CI MAX232 é um chip que faz a conversão de sinais TTL para RS232 e vice-versa. Ele permite a saída dos sinais para o cabo serial que utilizamos para alcançar o micro.

Alimentamos o CI com 5,0V (nível TTL), e utilizamos alguns capacitores (de 1uF) dobradores de tensão, necessários para o funcionamento correto do CI. Existe também uma versão já com os capacitores inclusos no encapsulamento.

A tarefa desse CI é converter sinais da porta serial do micro para os níveis TTL e os sinais TTL para as tensões do protocolo RS232. Os níveis de tensão convertidos são:

RS232 TTL

- 8.5V (1) ---------- 5,0V (1)

+ 8.5V (1) ---------- 0V (1)

A seqüencialização de sinais codificados para o padrão RS232 é feita pela UART programada em VHDL no interior da FPGA. Então para completar a transmissão serial basta a conversão dos níveis de tensão.

5.4 Cronograma

Nesta seção comentaremos sobre como se desenvolveu o projeto em relação ao cronograma finalizado, o que foi modificado, o que causou tal modificação, e como nosso cronograma se alterou durante o projeto.

Tivemos grandes alterações no cronograma. O principal fator responsável pela mudança no cronograma foi a desistência da produção de uma placa de protótipo (devido à complexidade de produção, dificuldade de importação de peças e por motivos financeiros), e manter o foco na implementação do VHDL e realizar os testes na placa de desenvolvimento.

As metas iniciais foram um pouco ambiciosas. A implementação em VHDL foi muito mais complexa do que se tinha imaginado. As ações que impactaram no cronograma estão descritas a seguir (bem como na parte Desenvolvimento, com maior riqueza de detalhes), assim como as alterações e o rumo atual.

5.4.1 Cronograma inicial

A definição do cronograma no início do projeto foi feita buscando atingir o objetivo de se construir uma placa de protótipo que fosse semelhante à placa final.

Como iremos ver a seguir, o cronograma para o projeto está dividido em três partes:

- Estudo e definição dos sistemas e do projeto
- Implementação do sistema
- Teste do sistema

O cronograma correu bem, só se desviando um pouco foi durante a implementação dos sistemas, como comentaremos na próxima seção. Essas alterações e reprogramações foram

naturais, devido à alteração no trabalho que foi feito, buscando melhorar o projeto frente aos fatos ocorridos.

Buscamos manter sempre em mente a resolução do problema principal, que é processar o sinal do gravímetro de forma robusta. Acreditamos que o projeto atingiu seu objetivo.

Segue abaixo os gráficos do cronograma inicial que nos propomos a seguir.

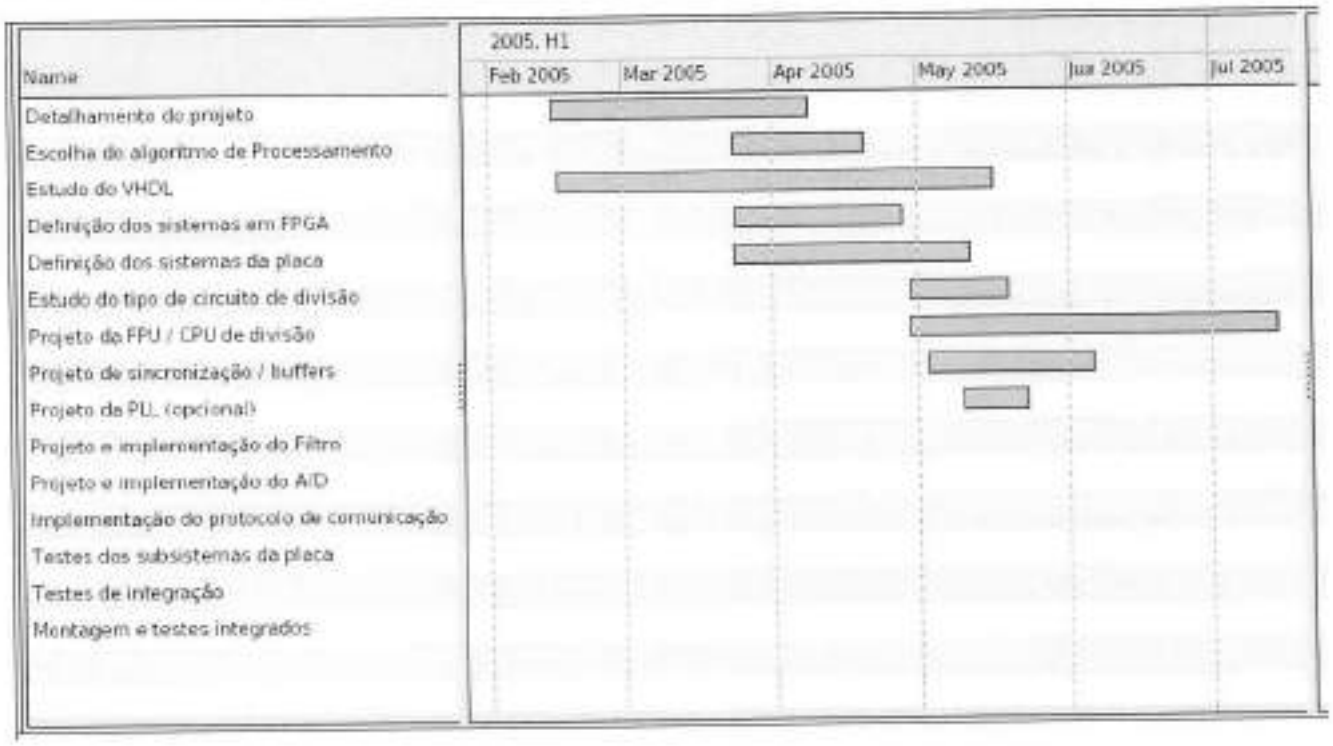


Figura 23: Cronograma inicial do primeiro semestre.

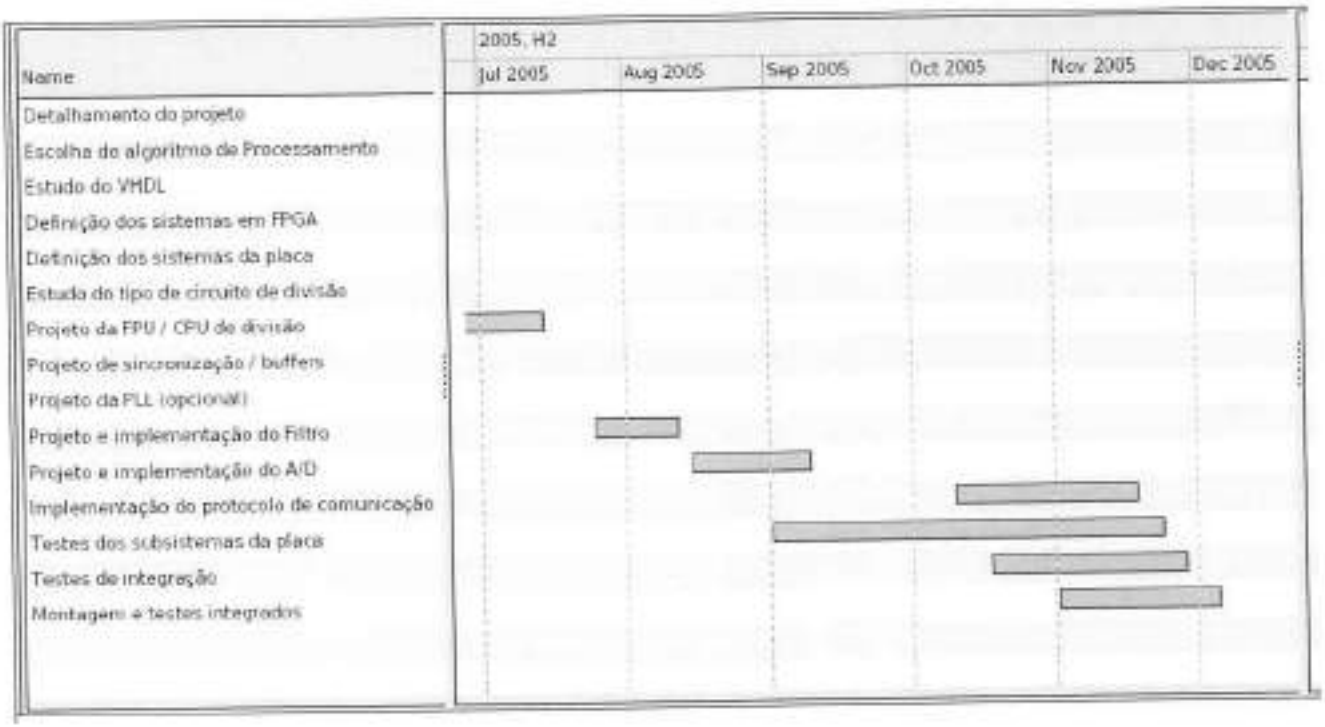


Figura 24: Cronograma inicial do segundo semestre.

5.4.2 Modificações

Tivemos uma grande mudança de projeto quando ficamos impossibilitados de comprar as peças para a placa, que nos levou a desenvolver o protótipo baseado na placa de desenvolvimento Digilent, pois era o material que já estava em nossas mãos.

Como citamos anteriormente, optamos por trabalhar os conceitos do sistema até eles ficarem consolidados e funcionais, deixando que os problemas de manufatura da placa sejam feitos num estágio posterior do projeto. De certa forma privilegiamos a parte da criação de um conceito que será bem conhecido, agregando conhecimento ao projeto como um todo, e deixamos as partes de montagem, dimensionamento, etc para uma fase futura.

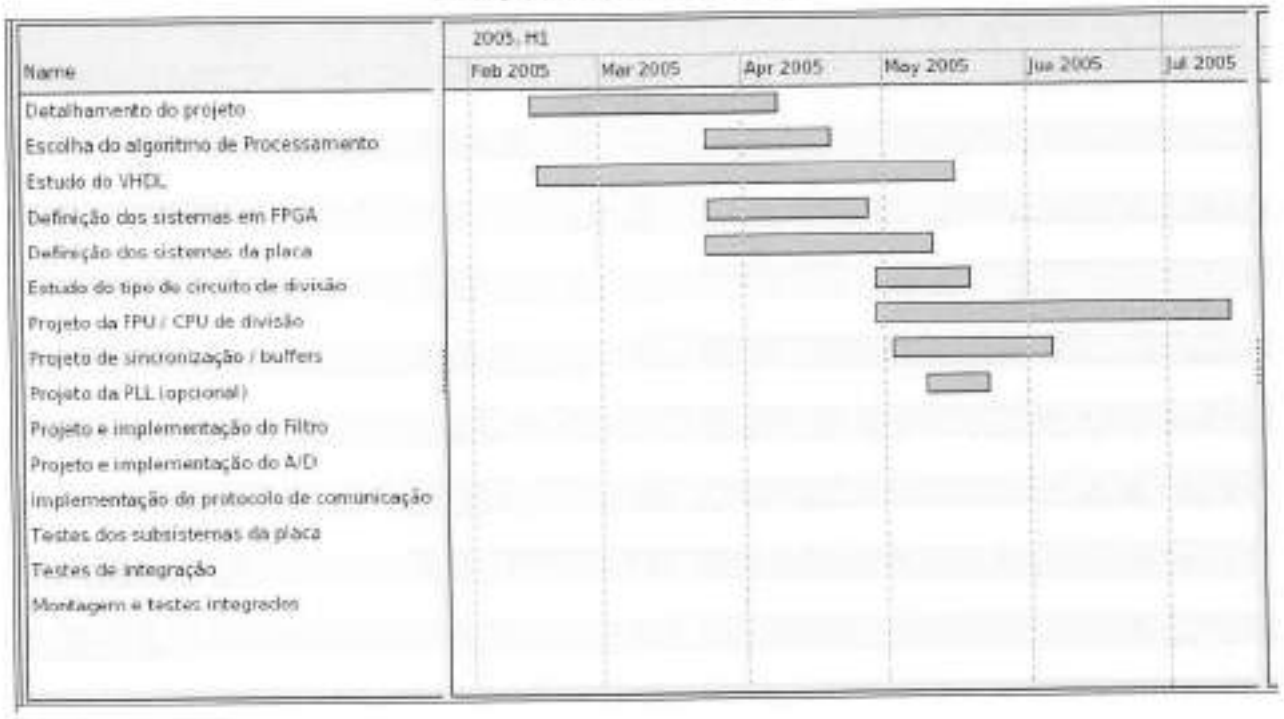
Essa opção foi colocada também porque no projeto em que o gravímetro está envolvido, existem outros instrumentos e sistemas que podem ser utilizados em mais de um instrumento. Por

exemplo, um filtro de entradas de sinal já está em desenvolvimento e teste por um outro grupo, e seria redundante trabalhar com um filtro agora, sendo que existe um outro em construção, em fase com certeza mais adiantada que o que poderíamos fazer.

Apesar dessas mudanças, o cronograma não se alterou muito. As diferenças substanciais estão na parte intermediária, que contemplam a construção dos sub-módulos, pois retiramos alguns módulos, alteramos outros. Por exemplo, a parte de configuração do sistema foi alterada para ser interna à FPGA e configurada pela comunicação serial, ao invés de existirem comandos para configurar o sistema na própria placa.

As principais alterações estão listadas sob o tópico Desenvolvimento. Abaixo vamos listar de forma simplificada como foram afetadas as datas (principalmente na parte intermediária) e como ficamos com o cronograma atual. As alterações, como foi dito, não foram muitas, e referem-se basicamente a alterações e supressões ocorridas em alguns módulos do sistema.

Cronograma - Primeiro Semestre



Cronograma - Segundo semestre

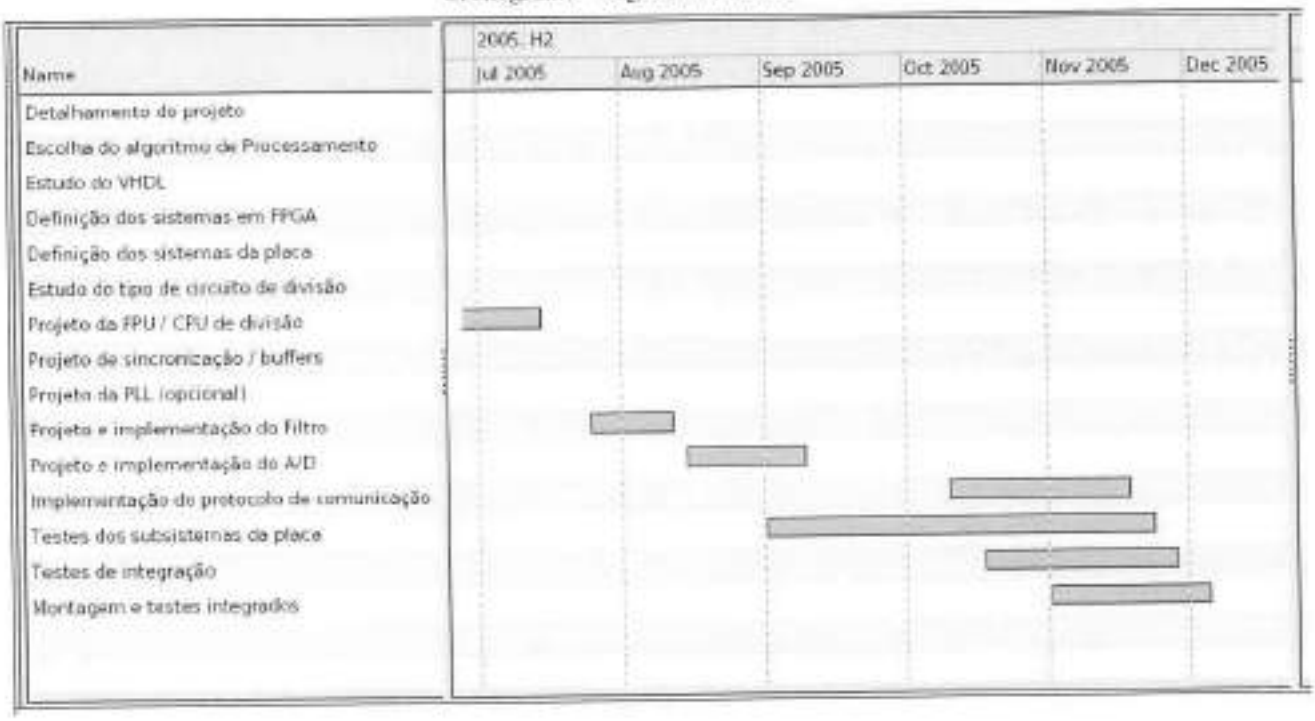


Figura 25: Cronograma final.

5.5 Problemas encontrados

Esta parte do relatório foi idealizada para dar uma visão geral dos problemas que existiram durante o projeto. Resumimos os principais problemas, e os descrevemos em ordem cronológica, desde o planejamento até os problemas de implementação. Apesar disso, a idéia é montar um panorama que represente todo o trabalho, e não entrar em detalhes, pois isso implicaria em muitas páginas desgastantes de texto. Assim, alguma pessoa que necessite de informações sobre o projeto para continuar ou refazer algumas partes do mesmo, pode ter uma visão dos pontos críticos e ações tomadas, ajudando num planejamento futuro.

Quando decidimos desenvolver o freqüencímetro baseado em FPGA, havia muito pouco conhecimento do grupo a respeito do tema, que não é muito comum. Apesar de todos conhecerem vhdl, instrumentação e sistemas digitais (em níveis diferentes de conhecimento), a peculiaridade dos requisitos de projeto chamou a atenção.

Existiam algumas diretrizes sobre como atacar o problema, baseado em outras soluções conhecidas anteriormente, e alguns sistemas parecidos, mas o conhecimento real sobre como realizar todo o projeto não estava consolidado entre todos os envolvidos.

Isso levou ao primeiro problema de vulto: especificar o sistema e seus componentes sem conhecer completamente o sistema. Contamos com a ajuda do prof João, que é co-orientador do projeto, que ajudou a descrever o sistema.

Uma vez que tínhamos definido o projeto e aproximadamente o que queríamos fazer para atacar o problema, foi a hora de passar para a ação. Fomos ao Laboratório de Sistemas Integráveis, pois sabíamos que o LSI possuía kits de desenvolvimento em FPGA para alguns projetos relacionados. Eles gentilmente emprestaram um dos kits, e a recomendação era ir ao site

da fabricante da placa e da FPGA buscar documentação sobre como operar o sistema. Além disso, o professor Edson Midorikawa (orientador do PCS) recomendou o uso do ambiente de desenvolvimento ISE para programar e simular o sistema da FPGA.

Com as ferramentas em mãos, tanto o programa ISE, quanto a placa de desenvolvimento com a FPGA, passamos a testar e nos familiarizar com o sistema. A documentação não era abundante, e não tínhamos muito a quem recorrer. O nosso conhecimento de VHDL teve que ser muito melhorado nesse período. A falta de entrosamento com o programa e a placa trouxeram (somados com a falta de prática de programação em VHDL, e a documentação parca) todo tipo de problemas que se possa imaginar. Com o tempo dominamos melhor o simulador e a linguagem e o trabalho fluiu muito melhor. Os pequenos tropeços com a FPGA e a placa Digilent nos acompanharam sempre. Alguns pequenos detalhes causaram grandes travamentos no projeto. Por exemplo, a necessidade de ativar alguns componentes da placa (sinais de enable), ou a existência de diferentes tipos de entrada para sinais diferentes, entre outros, várias vezes induziram-nos a pensar que a placa estava travada, em deadlock ou com algum bug, ou que o simulador estava com problemas. Algumas vezes eram bugs (resolvidos com um update de versão), outras foram causadas por não termos lido corretamente a documentação, e uma outra parte ainda decorre da não existência de documentação e outros problemas aleatórios.

Instalamos o simulador primeiramente no Linux, pois como a maior parte do grupo utiliza Linux essa possibilidade foi bastante atraente. Porém a tentativa de usar o programa no Linux não teve sucesso. A versão estava compilada para uma versão antiga de Linux, travava muito, os problemas encontrados e listados no site da Xilinx não tinham solução. Parecia um software abandonado. A dor de cabeça causada nos fez desistir da versão Linux, e instalar três máquinas Windows para utilizar o programa. A versão Windows funcionou bem, mas tivemos alguns problemas de licença (o programa é gratuito, mas é necessário baixar uma licença com trava para

o hardware em que o programa está instalado), e reinstalamos duas vezes do zero o ISE e o Windows devido aos problemas com licenças, que travavam o funcionamento do simulador algumas vezes. Tivemos que fazer alguns updates do programa, devido a alguns bugs encontrados e problemas. Isso resolveu praticamente todos os problemas com o simulador. Apesar do programa em si ser muito bom e os erros e falhas serem bem documentados (geralmente é possível olhar um link no site da Xilinx com os erros e possíveis soluções), o suporte ao programa no que se refere a baixar updates, licenças, fazer instalações, se registrar, fazer downloads, instalações de licenças são muito complicados. Colocar o programa para funcionar acabou sendo muito trabalhoso.

Se por um lado os problemas com o equipamento estavam sendo sanados, por outro ainda tínhamos problemas em entender a implementação necessária, principalmente no algoritmo de medição.

Essa demora na parte de equipamento e algoritmos nos levou a modificar alguns sistemas e principalmente a ter um certo atraso. Isso gerou uma grande mudança no projeto. Devido ao prazo muito justo, acabamos notando que não seria possível comprar as partes para a montagem de um protótipo de teste fora da placa de desenvolvimento da Digilent.

As raízes desse problema são duas: a compra de peças importadas e a dificuldade em encontrar alguém disposto a fazer a placa do protótipo. A FPGA em si é um equipamento caro e importado. O trâmite de importação deveria demorar meses. Como não tínhamos o projeto fechado e nem muita certeza sobre a precisão da medição, não sabíamos qual o modelo mínimo para atender os requisitos (apesar de ser possível simular a síntese e parâmetros de tempo para outras FPGAs que não aquela que possuíamos na placa). Conversamos com o sr. Edson Horta, da BP&M, representante xilinx no Brasil, e ele recomendou a compra pela internet ou o pedido de amostras para a Xilinx. Ambas as alternativas demorariam muito tempo para serem liberadas.

Mas a FPGA não era o único problema. A manufatura da placa é muito complicada. Isso se deve principalmente à necessidade de se fazer uma placa especial em baixa quantidade (a rigor, uma ou duas placas) e o outro ponto era a definição do que deveria ser colocado na placa, pois era possível que algumas partes da placa contivessem componentes manufaturados por outras partes do projeto do gravímetro (como por exemplo, a PLL). A placa é especial devido à necessidade de soldar a FPGA a placa. A FPGA é um chip muito denso em quantidade de pinos. Um chip de baixa densidade pode facilmente ter facilmente 25 pinos por centímetro, e um total que varia de 200 a 600 pinos no total. As pessoas que entramos em contato para fazer a placa tinham restrições quanto à soldagem, sendo que nos casos mais densos a soldagem não poderia ser feita manualmente. A solução pensada seria em utilizar um soquete especial para FPGAs, mas o soquete era fabricado na Inglaterra (pela Aries Electronics), e ainda teríamos todos os problemas de importação, enviar a FPGA, descobrir como fazer a solda, etc. Concordamos que seria melhor focar no funcionamento do projeto do chip e testes e quando este estivesse bem consolidado e completamente definido, então num estágio futuro começaria o ataque à construção da placa. Isso foi bom para focalizar os esforços, diminuir a carga de trabalho sobre o grupo e criou a possibilidade de fazer uma placa melhor posteriormente, incorporando partes de outros projetos, e com a parte da FPGA dissecada.

Paralelamente à coleta de informações sobre a manufatura, começamos a testar a precisão do nosso sistema de aquisição e divisão, para parametrizar dados como tempo de medição, máxima precisão, clock necessário. Após alguns testes descobrimos que nosso método não alcançava a precisão requerida. A partir daí começamos a buscar modificações no circuito para aumentar a precisão, e tivemos até que uma boa melhora modificando o circuito e com alterando o método de medição. Depois em uma reunião com o prof. João, descobrimos que estávamos

considerando uma precisão maior que a necessária, e que o método de medição tinha algumas modificações. Definido isto a implementação voltou a ficar coerente com os requisitos de projeto.

Quando resolvemos todos esses problemas, o projeto já estava numa fase avançada. Começamos a fazer alguns circuitos eletrônicos auxiliares de interligação da placa da FPGA tanto na parte de entrada do sinal quanto na parte da comunicação serial com o computador. Montamos em uma protoboard um circuito digitalizador baseado em um CI comparador com Amp-Op e um circuito que convertia níveis de tensão RS232 em TTL. Os circuitos funcionaram bem, com os problemas tradicionais de mau contato em protoboards. Adicionamos um conversor de tensão com transistores, pois a placa não é TTL, mas sim LVTTL (em que o nível alto é por volta de 3.3V), para proteger a placa e a FPGA (que possuem máxima tensão de operação de 4,0V).

Os últimos problemas que tivemos foram com o controle do circuito digital da FPGA e sincronização com os sinais reais. Apesar de termos o circuito simulado, tivemos alguns problemas na hora de baixar o circuito sintetizado para a placa que não se manifestavam na simulação. Outro problema que não aparecia nas simulações era a síntese do controle dos estados dos circuitos, ou seja, das máquinas de estado. As máquinas de estado que formam as unidades de controle dos circuitos devem seguir alguns padrões para que todos os estados sejam detectados e otimizados corretamente. A dificuldade em gerar algumas máquinas corretamente causava a não detecção de estados e travamentos na placa da FPGA. A modificação de alguns parâmetros de temporização (como aumentar o tempo entre eventos, esperar alguns ciclos de clock em idle) e uma explicitação no código das máquinas de estado finito resolveram a maior parte dos problemas.

6 Balanço da implementação

A implementação foi feita praticamente em sua totalidade. Sanamos quase todos os problemas encontrados durante a síntese e download do código para a FPGA. O sistema foi completamente simulado e testado, como é possível ver no tópico Testes (que é o próximo).

O sistema deve gerar duas informações para cada medida: a informação de freqüência medida e o intervalo de tempo decorrido desde a última medida. Isso foi feito, e pode ser observado no computador com o auxílio de um programa de terminal serial. O intervalo de tempo desde a última medida é dado por um número puro que indica quantos ciclos de clock decorreram desde a última medida. Assim, multiplicando pelo período do clock podemos identificar o tempo decorrido desde a última medida.

Projetamos e testamos todo o sistema que permite receber um dado e uma base de tempo, fazer a aquisição e comparação de dados, processá-los e enviar este dado via serial para um computador. Pudemos acompanhar os dados de saída na tela de um programa terminal de serial e aferir que eles batem com a entrada.

Algumas partes do sistema não foram implementadas, mas foram previstas, por falta de tempo hábil. As configurações do circuito estão hard-coded, e não podem ser alteradas, e não fizemos um programa que envia dados seriais do micro para a placa. Esta parte é simples, ainda mais porque sabemos que o envio no outro sentido (fpga para computador) funciona. Isso porque para o teste de recebimento utilizamos um terminal modificado chamado Realterminal, que é melhor que os terminais comuns, pois ele mostra caracteres não imprimíveis, ou seja, podemos ver qualquer seqüência de 8 bits nele, diferentemente do HyperTerminal, do Windows. Como não

precisamos codificar um programar semelhante de terminal para o teste de entrada, não codificamos um programa para a saída.

O programa de saída (que gere os pacotes de reconfiguração do sistema) não é um terminal genérico, ele deve ter mais funcionalidades. Acabamos por priorizar outros testes e deixamos de lado essa parte. Ela não é muito complicada, e quando definido o que se quer exatamente pode-se codificar rapidamente tanto para Windows quanto para Linux.

Através de entradas no simulador pudemos ver que o erro estava sempre abaixo de 3 casas decimais, como foi nossa proposta. Porém, não conseguimos testar com um sinal real e um ambiente controlado, tais como um calibrador de freqüências ou placas de alta velocidade, também por falta de tempo hábil. Apesar de termos tido a oportunidade de utilizar alguns desses recursos, por uma série de fatores não pudemos testar com eles nos últimos períodos do projeto de formatura.

Fizemos o circuito principal do projeto funcionar. Implementamos uma solução em hardware dedicado para realizar o processamento digital do sinal do gravímetro. Montamos um protótipo em placa de desenvolvimento do circuito, que é funcional e atende os requisitos de projeto.

7 Testes e depuração

Durante a realização do projeto, sempre estivemos preocupados em realizar testes e simulações que indicassem o funcionamento do circuito. Nesta seção mostraremos alguns testes de simulações do sistema no MXE ModelSim e observado resultados no terminal.

Muitas vezes foram necessárias horas de depuração de circuito tanto simulado quanto na placa. O simulador mostra vários problemas, mas existem falhas em outras "camadas" de desenvolvimento que só aparecem no teste real, tais como problemas de roteamento de sinais e pinos, de síntese das máquinas de estado, tipos de entradas, etc.

Testamos várias vezes cada bloco do sistema, observando na placa alguns sinais internos para observar se o sistema funcionava. Outro artifício era associar leds da placa de desenvolvimento a estados do sistema e das máquinas de estado finito para detectar uma eventual entrada do sistema em deadlock ou travamento.

Os circuitos de comunicação e alguns outros são muito sensíveis ao clock, devido à necessidade de sincronia de sinais. Tínhamos que diminuir o clock para observar o efeito de alguns sinais na sincronização. Por exemplo, um pulso de enable muito rápido pode não ser suficiente para mudar o estado do circuito, ou pode disparar uma mudança no circuito enquanto outra parte ainda não está preparada para isso. É importante deixar folgar em sinais e às vezes alguns ciclos em idle (sem execução) para ter certeza que esses efeitos não irão ocorrer.

7.1 Resultados obtidos

O circuito em VHDL funcionou como um todo. Pudemos observar o circuito processando entradas corretas tanto no simulador quanto no terminal de recepção serial, quando a placa enviava os dados processados para o micro via cabo RS232.

Os circuitos eletrônicos de entrada e de conversão de níveis também estão funcionando, pudemos atestar o funcionamento com fontes de tensão, e também com osciloscópio, quando excitados por um gerador de sinal. Abaixo temos algumas telas da simulação do circuito:

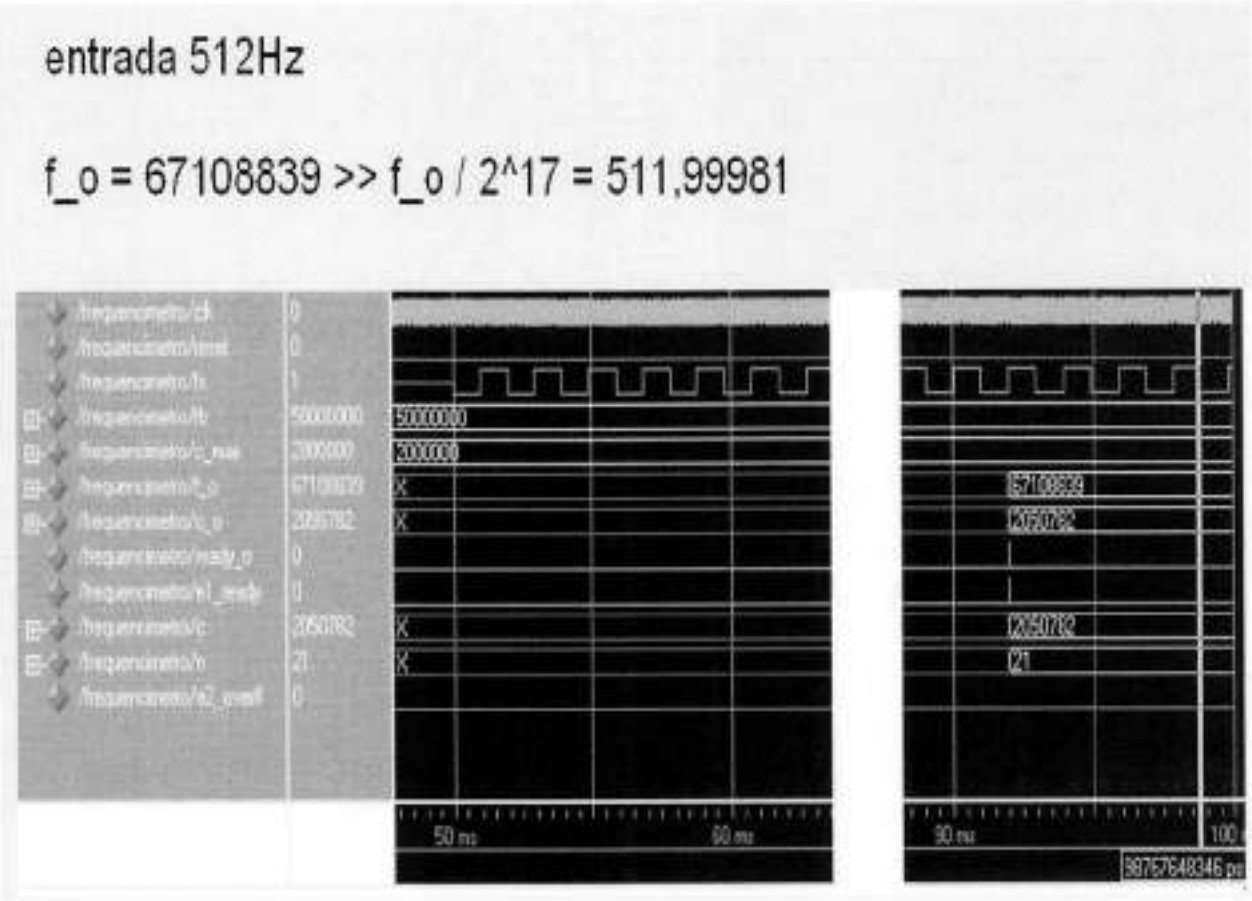


Figura 26: Depuração da contagem: precisão na terceira casa.

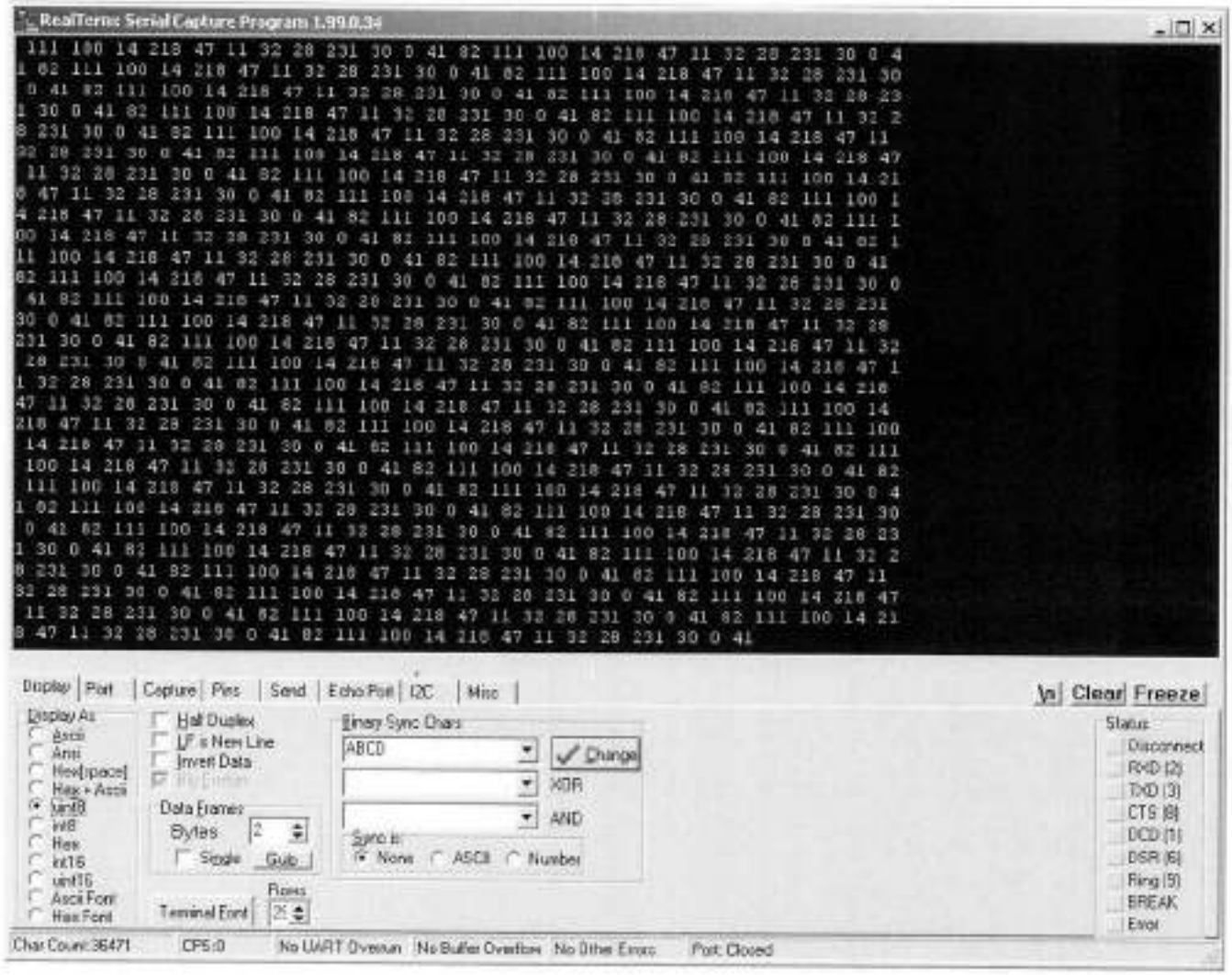


Figura 27: Tela do terminal recebendo os dados

7.2 *Análise do desempenho do circuito*

Durante a implementação do circuito estivemos sempre preocupados com o desempenho do circuito, pois o funcionamento do circuito depende diretamente da sua capacidade de operar com alto clock. Isso implica em escolher implementações otimizadas, com menor caminho crítico do sinal, e maior clock máximo.

Outro ponto de otimização seria na área ocupada do circuito ou quantidade de I/O's (entradas e saídas) utilizadas. Porém, nesse projeto estes parâmetros não são críticos, e sua importância é menor, comparada com os anteriormente citados.

O circuito deve manter um alto clock para permitir a aquisição suficiente de dados para dar a resposta do processamento com alta precisão. Ou seja, o throughput do circuito não é tão elevado, mas é necessária uma alta velocidade de operação para que se tenha dados suficientes para garantir a precisão da medida.

Outro fato é que o sistema não utiliza buffers. A velocidade de operação é fixa (uma vez fixada o design e implementação de baixo nível), e praticamente independente da entrada. Isso impossibilita o uso de buffers para amortecer uma eventual lentidão entre os circuitos. Logo, os circuitos de processamento posteriores devem suportar a taxa de dados e funcionamento do estágio anterior. Se eventualmente isso não funcionasse, então começaríamos a ter perda de dados no circuito (o que ocorreria se tivéssemos buffers, pois eles seriam lotados).

Por isso, estivemos preocupados em analisar e simular cada bloco a partir da entrada para verificar se o circuito atendia os requisitos. Algumas partes do circuito não podem ter um clock elevado (por exemplo, o circuito divisor não suportaria clocks muito mais altos que 50MHz), mas é possível utilizar técnicas para vencer estas limitações, tais como dividir o circuito em áreas de clock diferente. Isso é excelente, pois às vezes se possui circuitos single-clock (realiza todo o processamento em um único ciclo de clock) que não podem operar com um clock elevado (devido à sua complexidade), e nesse caso podemos utilizar também circuitos single-clock, em uma região de baixa velocidade.

É possível ter essa flexibilidade nesse projeto, pois uma medição dura vários períodos (milhões) de clock, mas outras partes do circuito necessitam apenas de algumas dezenas de ciclos. Durante os ciclos restantes, o circuito pode processar algum outro dado, inclusive em baixa velocidade, utilizando, por exemplo, um clock 10 ou 15 vezes menor que o original.

Outro ponto em que isso é interessante é quando se utiliza uma FPGA mais rápida. Por exemplo, simulamos o circuito para uma FPGA da família Virtex 4, de alto desempenho, e obtivemos que é possível operar o circuito a 100 MHz. Para fins de obter uma melhor precisão gostaríamos de utilizar o clock mais elevado possível. Nesse projeto, a unidade de divisão RISC não suporta mais esse clock (o limite é por volta e 60MHz). Então rodaríamos o estágio de aquisição de dados em alta velocidade e o circuito do divisor em uma região separada com menor clock.

8 Conclusão / Considerações finais

Dividimos a conclusão em duas partes. A primeira relaciona nossas considerações a respeito do projeto depois dos meses que se passaram no projeto e implementação. Depois um pouco sobre o futuro do trabalho de formatura, sobre uma possivel continuidade.

8.1 Balanço do projeto

Apesar de todas as dificuldades, podemos dizer sem medo de incorrer em erros que acreditamos termos alcançado sucesso no projeto. Com todas as dificuldades do tema, a falta de experiência, conseguimos criar um protótipo baseado no hardware disponível que processa um sinal de entrada com alta precisão e envia o resultado através de comunicação serial para o computador.

Muitas partes do sistema ficaram faltando, por diversos motivos, inclusive falhas do grupo. Porém, não nos esquecemos do objetivo proposto, e procuramos privilegiar as partes que iriam proporcionar um estado mais próximo da solução.

Dominamos o design básico do sistema. Uma vez comprado o hardware e anexados os sistemas auxiliares, um protótipo de pré-produção no formato de uma placa dedicada do instrumento estará concluido, e pronto para testes em operação real, com um sinal vindo do gravímetro, base de tempo de precisão real e poderá ser operada embarcada em um avião.

O trabalho trouxe grande satisfação e orgulho para o grupo. As metas e a falta de experiência chegaram a causar certo desânimo, mas quando conseguíamos vencer os desafios

impostos, tínhamos certeza que alcançaríamos o objetivo de maneira satisfatória. Ou pelo menos tentamos ao máximo fazer com que isso fosse alcançado.

Certamente, é um projeto memorável para nós do grupo, seja pelo desafio, pelo resultado final, por podermos colocar em prática várias áreas de conhecimento que nós gostamos de trabalhar (que foi um dos motivos da escolha deste tema de trabalho de formatura).

8.2 *Perspectiva de Continuidade*

Aprendemos bastante sobre o funcionamento do processador de sinais e os diversos problemas relacionados à nossa medição de freqüências. Esperamos que o projeto seja continuado em breve, já que ele faz parte de um projeto real. Apesar de não termos feito todo o sistema, fizemos todo o projeto com a preocupação de não criar nenhuma limitação à expansibilidade ou a implementação completa do sistema de medição de freqüências.

Alguns sistemas foram projetados e previstos, mas não foram implementados. Incluímos essas partes na documentação, para que no futuro esses pontos sejam relembrados. O design modularizado foi sempre focado.

Em outros pontos incluímos dados propositalmente pensando num projeto futuro, tais como termos reservado espaço para informações sobre o modo de operação, identificação do freqüencímetro e origem do sinal no protocolo de envio de dados, separando áreas de clock conforme o desempenho (apesar do clock ser o mesmo em todo o circuito agora), para que se pudesse facilmente alterar alguns parâmetros de desempenho quando fosse utilizado um chip de maior desempenho que nossa FPGA atual.

A idéia foi preparar o terreno o máximo possível, e permitir que os construtores e projetistas que possam vir a trabalhar no projeto tenham dados suficientes para completar e

melhorar o design sem ficar perdendo tempo tentando entender o que foi feito ou corrigindo falhas de design no protótipo.

Anexo I: Processo de teste e geração de código

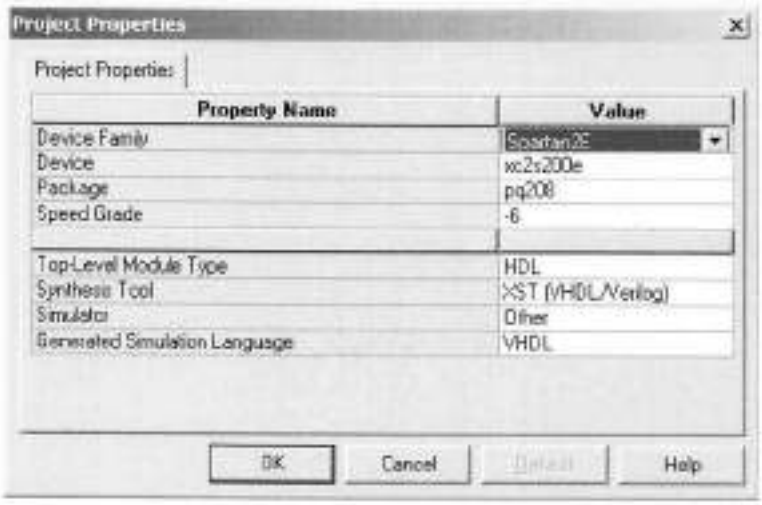
Abaixo temos um "tour" pelo processo de gerar e testar o código pelo qual passamos quando utilizamos o ambiente ISE 7.1i, da Xilinx para auxiliar o design. Importante mencionar que após a síntese, se tivermos os parâmetros dos dispositivos setados (ou seja, se utilizarmos bibliotecas do dispositivo) é possível chamar o programa MXE ModelSim para realizar simulações de waveform, como foi mencionado em relatórios anteriores.

Passo a passo - do vhdl à FPGA:

0) Selecionar dispositivo, empacotamento, pinos, e grau de desempenho do dispositivo

A primeira coisa a se fazer é selecionar o tipo de FPGA que se quer usar. Existem várias descrições a serem feitas, e é possível obter as informações do datasheet da FPGA, além de elas serem gravadas em cada chip também. Algumas informações são um pouco confusas, mas em geral é possível inferir alguma informação estranha identificando todas as outras e procurando no datasheet o que falta, como por exemplo, o 6C no final da nossa FPGA da placa de desenvolvimento que o dispositivo é comercial (existem os industriais, mais resistentes), com speed grade -6 (versão normal de velocidade).

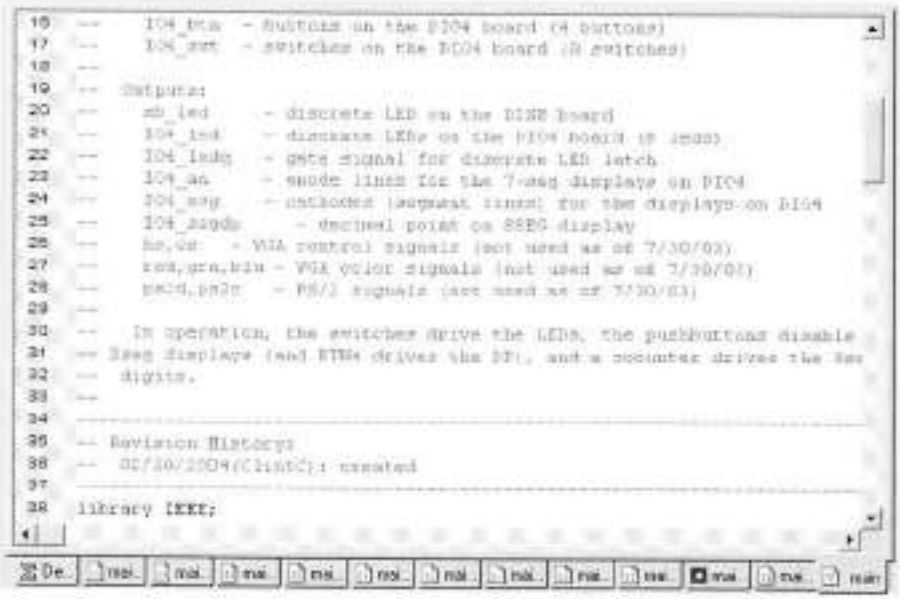
Descrição da FPGA da placa de desenvolvimento:



1) Gerar descrição do circuito digital e dos sinais

Podemos descrever o circuito através de linguagem de descrição de hardware (Verilog, VHDL) ou através de "schematics" de bibliotecas do programa (semelhante a diagramas do PSpice, ou do MaxPLUS/Quartus).

Se utilizarmos VHDL, é possível gerar posteriormente durante a síntese um bloco funcional que representa o circuito codificado. Assim, podemos reaproveitar o design e outros circuitos, se necessário. Abaixo, um editor com arquivos .vhd abertos:



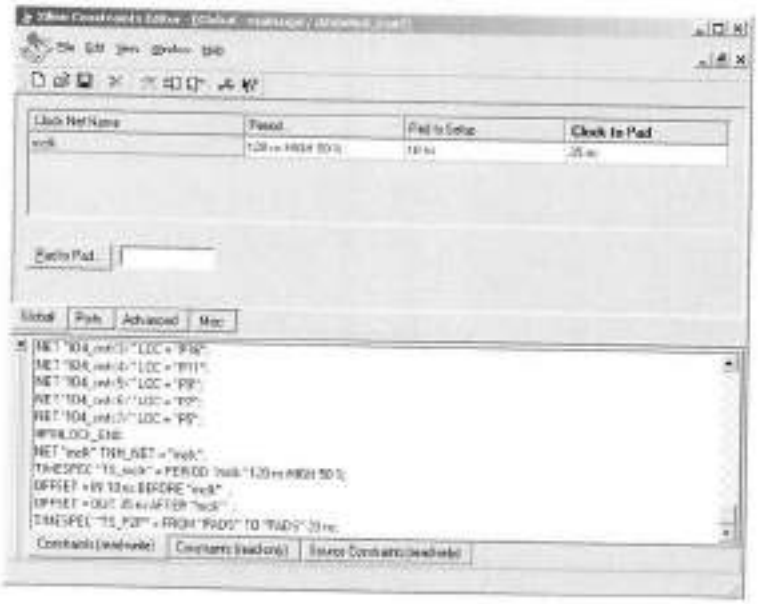
2) Restrições de projeto (opcional, para simulação não precisa)

É possível também adicionar restrições de projeto nesta parte. Basicamente as restrições são de dois tipos: de temporização (tem pode setup, delay, largura do clock) ou de área ocupada (quantas portas lógicas estão sendo utilizadas). Além disso, assinalamos os pinos utilizados também.

Além disso, nesta fase é possível mudar os seguintes requisitos:

- temporização (restrições de sinais);
- restrições de área ocupada;
- assinalar pinos.

Abaixo, podemos ver a tela da edição de constraints, com uma linha para o sinal de clock:



3) Síntese

O processo inicial para teste do código é chamado de síntese. Nesta parte ocorre a decomposição do circuito em blocos lógicos básicos, conforme a biblioteca de componentes que se está utilizando. Normalmente é gerado um arquivo "schematic" contendo um bloco que representa o circuito e os sinais de entrada e saída (ou é possível começar do schematic também para descrever o circuito).

A descrição VHDL é checada e compilada. É possível testar o código no simulador depois da síntese. As tarefas desenvolvidas nessa etapa são:

- compilação do VHDL;

- análise / checagem do VHDL;

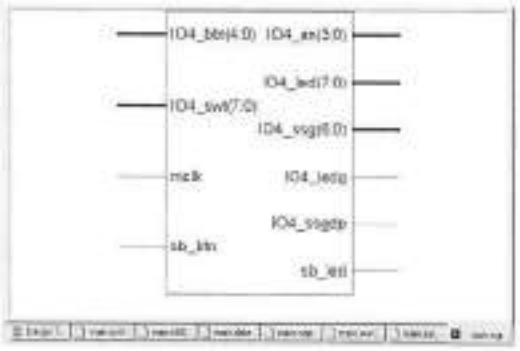
- síntese VHDL;

- geração do modelo para simulação;

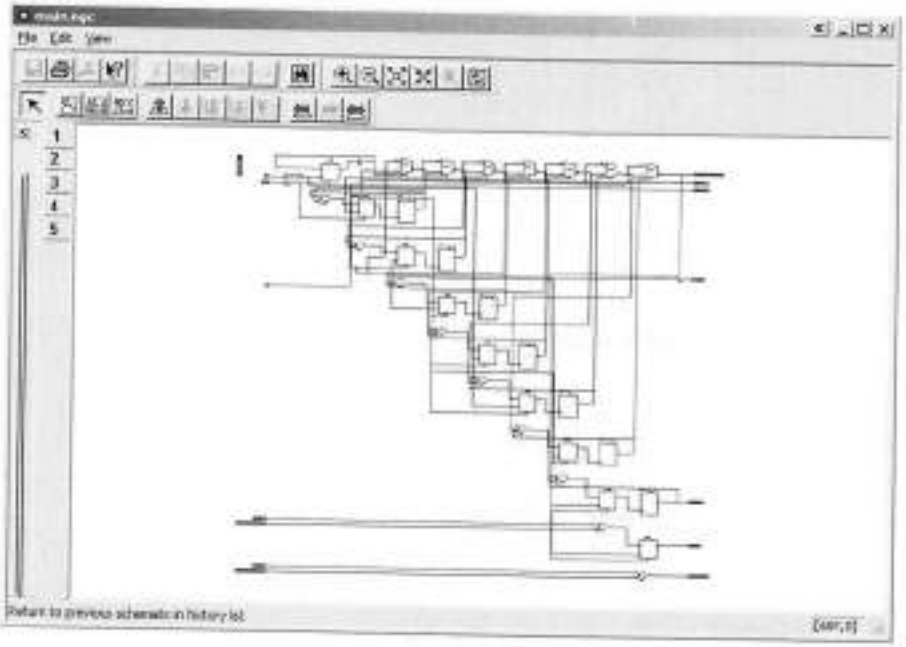
- levantamento dos parâmetros de desempenho, ocupação, etc.

Após a síntese é possível fazer a simulação no MXE ModelSim, observar os blocos básicos do circuito, e uma coisa importante: é possível visualizar parâmetros interessantes como taxa de ocupação de componentes da FPGA, atraso máximo para alguns sinais, freqüência máxima de alguns sinais, etc.

Abaixo vemos o schematic top-level gerado



A seguir, o mesmo circuito, expandido em blocos básicos (primeira página, de cinco no total)



Nesta fase é possível escolher o tipo de otimização, se otimizar para minimizar área, ou tempo de propagação ou aumentar a freqüência de operação. O circuito pode ser recombinado com outros componentes ou priorizando outros sinais conforme o interesse.

O processo de síntese é um dos mais sensíveis, e mudanças aqui podem definir o desempenho de um circuito para o bem ou para o mal. É possível realizar a síntese em outros programas, que não o da Xilinx. Um software famoso é o Leonardo Spectrum, da Mentor Graphics, que faz otimizações durante a síntese.

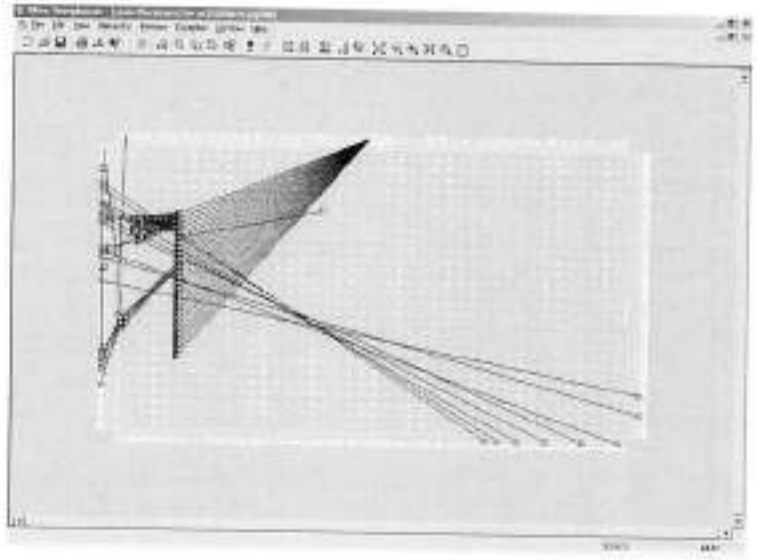
Pretendemos testar fazer a síntese no Leonardo, que está disponível em uma sala de micros do PSI, e ver se obtemos algum resultado compensador. O problema é que este software não é gratuito, e pode implicar em problemas de licença, se usado com finalidade que não seja acadêmica. Para fins de TF, não há problemas.

4) Implementação do design

Na fase de implementação do design começamos a tratar da interação do nosso código com o dispositivo que pretendemos usar. É possível escolher o tipo de otimização, especificar os componentes básicos da FPGA que serão utilizados, e selecionar opções específicas ao chip que se está utilizando.

O tipo de otimização para encodar a máquina de estados FSM (finite state machine) é muito importante, pois permite definir sinais de controle mais rápidos. Como orientação do prof. Wang passamos a utilizar a opção chamada "one hot", em que se gasta mais portas lógicas para codificar a FSM (um por estado), mas diminuímos o tempo de resposta, o que torna a resposta do circuito mais rápida.

Tela do Floorplan/PACE que editam os pinos e utilização de área:



Esse é um estágio longo, com várias operações, como se pode ver abaixo:

- Tradução: checagem de especificações, alocação de recursos da FPGA;

- Temporização: análise da temporização dos sinais;

- Floorplan: alocação física no chip das portas lógicas;

- Análise do consumo/potência/térmica da FPGA;

- Análise dos níveis de tensão, impedância, tempo de subida, descida, etc em cada pino (simulação SPICE);

- Place and route: verificação se o design funciona para aquele dispositivo (é possível modificar o grau de otimização utilizado, para tentar um design menor ou mais rápido, mas o custo é o aumento do tempo de implementação);

- Simulação IBIS (I/O Buffer Information Specification): contém informações de características elétricas do circuito, tais como níveis de tensão, corrente, em cada pino.

5) Gerar arquivo para download

Após todos os testes, resta baixar o design para o dispositivo. Isso pode ser feito diretamente para a FPGA, ou então se pode gerar um arquivo que pode ser gravado em uma memória PROM posteriormente.

Podemos selecionar opções relacionadas ao download e geração do código de máquina (bitstream), tipo de clock que será utilizado. Normalmente, no desenvolvimento utilizamos um cabo paralelo ou USB para baixar o programa via interface JTAG para a FPGA da placa de desenvolvimento.

Referências

Livros

- Cavanagh. Digital Computer Arithmetic - Ed. McGraw Hill
- Null, Linda e Lobur, Julia. Essentials of Computer Organization and Architecture - Ed. Jones & Bartlett
- Oliver, B. M., Cage, J. M. Electronic Measurements and Instrumentation - Ed. McGraw-Hill
- Patterson, David A. Computer Architecture - A quantitative approach - Ed. Morgan Kaufmann

Papers

- A new Algorithm for Division in Hardware: paper in: International Conference on Computer Design, IEEE
- Design of PLL-based clock generation circuits: paper in: Solid-State Circuits, IEEE Journal of
- Kantabutra, V. A new theory for High-Radix Division in Hardware. Department of Mathematics – Idaho State University
- Making accurate frequency measurements. Article in "National Instruments Developer Zone" http://zone.ni.com

- Ruggiero, W. Metodologia de Projeto de Sistemas Digitais. Apostila. EPUSP, São Paulo, 1998.
- Statistical Analysis of Floating Point Flaw: Intel White Paper -Intel White Paper: at: http://support.intel.com/support/processors/pentium/sb/CS-013008.htm

Sites URLs

- Digilent Inc D2SB e DIO4 Boards – References, brochures e Schematics: http://www.digilentinc.com/products/Documentation.cfm
- Fluke frequency counters standards
- IEEE Standard 754 Floating-Point Standard: http://www.math.byu.edu/~schow/work/IEEEFloatingPoint.htm
- The ISE 7.1i Reference Manual: http://www.xilinx.com (disponível para download de usuários registrados)
- The Open Cores Community – referência em design de cores e HDL: http://www.opencores.org
- The VHDL CookBook:http://tech-www.informatik.uni-hamburg.de/vhdl/doc/cookbook/VHDL-Cookbook.pdf

Apêndice I – Apresentação do CD do projeto dirigido

O CD do projeto dirigido reúne a documentação, o código fonte e outros materiais utilizados no desenvolvimento do projeto de Sistema Dedicado de Medição de Freqüência. O conteúdo de cada diretório está descrito a seguir:

\DOC\ Documentação utilizada no projeto

- PCS2502-frequencimetro-DF-2005.doc: Documentação final do projeto
- PCS2502-frequencimetro-DF-2005.pdf: Documentação final do projeto

\SRC\ Descrição em VHDL de cada módulo do sistema

\ETC\ Outros materiais

- PCS2502-frequencimetro-A-2005.ppt : Apresentação final
- PCS2502-frequencimetro-A-2005.pdf: Apresentação final
- PCS2502-frequencimetro-P-2005.ppt: Pôster do projeto
- PCS2502-frequencimetro-P-2005.pdf: Pôster do projeto