FUNDAÇÃO DE ENSINO “EURÍPIDES SOARES DA ROCHA”
CENTRO UNIVERSITÁRIO “EURÍPIDES DE MARÍLIA” – UNIVEM
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

EVANDRO SÉRGIO MARCONATO

AUTOMAÇÃO E CONTROLE DE PULVERIZAÇÃO EM MÁQUINAS AGRÍCOLAS

MARÍLIA
2007

EVANDRO SÉRGIO MARCONATO

AUTOMAÇÃO E CONTROLE DE PULVERIZAÇÃO EM MÁQUINAS AGRÍCOLAS

Trabalho de Conclusão de Curso apresentado ao Programa de Graduação do Centro Universitário Eurípides de Marília, mantido pela Fundação de Ensino Eurípides Soares da Rocha para obtenção do Título de Bacharel em Ciência da Computação. (Área de Concentração: Arquitetura de Sistemas e Computação Reconfigurável).

Orientador:
Prof. Dr. Ildeberto de Gênova Bugatti

MARÍLIA
2007

MARCONATO, Evandro Sérgio
Automação e controle de pulverização em máquinas agrícolas / Evandro Sérgio Marconato; orientador: Ildeberto de Gênova Bugatti. Marília, SP: [s.n.], 2007.

122 f.

Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Centro Universitário Eurípides de Marília - Fundação de Ensino Eurípides Soares da Rocha.

1. Automação 2. FPGA

CDD: 004.22

EVANDRO SÉRGIO MARCONATO

AUTOMAÇÃO E CONTROLE DE PULVERIZAÇÃO EM MÁQUINAS AGRÍCOLAS

Banca examinadora do Trabalho de Conclusão de Curso apresentado ao Programa de Graduação do UNIVEM/F.E.E.S.R., para obtenção do Título de Bacharel em Ciência da Computação. Área de Concentração: Arquitetura de Sistemas e Computação Reconfigurável.

Resultado:

ORIENTADOR: Prof. Dr. Ildeberto de Gênova Bugatti

1º EXAMINADOR: Prof. Rodolfo Barros Chiaramonte

2º EXAMINADOR: Prof. Dr. José Celso Rocha

Marília, 22 de novembro de 2007.

AGRADECIMENTOS

A Deus, por ter me proporcionado força para conciliar as atividades de minha profissão com os assuntos acadêmicos.

A minha noiva Renata, pela paciência e compreensão.

Aos colegas Denison e Diego, do Grupo de Iniciação Científica do UNIVEM, pela importante ajuda na elaboração e simulação dos circuitos lógicos.

Ao colega de empresa Rodrigo Marcon, pela ajuda na preparação do painel semestral de acompanhamento.

A empresa Máquinas Agrícolas Jacto S/A, que gentilmente cedeu material, equipamentos e instalações para os testes práticos do sistema desenvolvido.

Ao Professor Bugatti, pela orientação, paciência, ajuda e incentivo durante todo o desenvolvimento.

MARCONATO, Evandro Sérgio. **Automação e Controle de Pulverização em Máquinas Agrícolas**. 2007. 122f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação). Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2007.

RESUMO

O agronegócio possui participação fundamental no cenário econômico brasileiro, com reflexos importantes sobre o produto interno bruto, as exportações e a geração de empregos.
A viabilidade econômica do setor agropecuário a partir da redução dos custos de produção, do aumento de produtividade e da redução do impacto ambiental causado pelo excesso de insumos, depende da agricultura de precisão. O principal objetivo da agricultura de precisão é maximizar a eficiência na utilização de insumos a partir da utilização em máquinas agrícolas de recursos eletrônicos (eletrônica embarcada) e de informática como, por exemplo, sensores, atuadores, computadores de bordo, controladores de pulverização, controladores de adubação, mapeamento e aplicação via satélite. O presente trabalho tem como objetivo integrar elementos de eletrônica embarcada a partir de um estudo de caso de um controlador de pulverização nacional que utiliza microcontroladores, onde é proposta, visando a melhoria de desempenho e a otimização de *hardware*, a integração das funções mais nobres de processamento e a integração de diversos elementos periféricos aos microcontroladores em um FPGA (*Field Programmable Gate Array*).

Palavras-chave: Agricultura de precisão. Eletrônica embarcada. Pulverizador. Automação. Controlador de pulverização. Microcontrolador. FPGA.

MARCONATO, Evandro Sérgio. **Automação e Controle de Pulverização em Máquinas Agrícolas**. 2007. 122f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação). Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2007.

ABSTRACT

The agribusiness possesses fundamental participation in the Brazilian economical scenery, with important reflexes about the gross domestic product, the exports and the generation of employments. The economical viability of the agricultural section starting from the reduction of the production costs, of the productivity increase and of the reduction of the environmental impact caused by the excess of inputs, it depends on the precision agriculture. The principal objective of the precision agriculture is to maximize the efficiency in the use of inputs starting from the use in agricultural machines of electronic resources (embedded electronics) and of computer science as for instance, sensors, actuators, board computers, spraying controllers, fertilization controllers, mapping and application via satellite. The present work has as objective integrates embedded electronics elements starting from a case study of a national sprayer controller that uses microcontrollers, where it is proposed, seeking the acting improvement and the hardware optimization, the integration of the most important processing functions and the integration of several outlying elements to the microcontrollers in a FPGA (Field Programmable Gate Array).

Keywords: Precision agriculture. Embedded electronics. Sprayer. Sprayer controller. Microcontroller. FPGA.

MARCONATO, Evandro Sérgio. **Automação e Controle de Pulverização em Máquinas Agrícolas**. 2007. 122f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação). Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2007.

RESUMEN

La agroindustria posee la participación fundamental en el paisaje económico brasileño, con reflejos importantes sobre el producto doméstico grueso, las exportaciones y la generación de empleos. La viabilidad económica de la sección agrícola que empieza de la reducción de costos de producción, del aumento de productividad y de la reducción del impacto medioambiental causada por el exceso de entradas, depende de la agricultura de precisión. El objetivo principal de la agricultura de precisión es aumentar al máximo la eficacia en el uso de entradas que empiezan del uso en las máquinas agrícolas de recursos electrónicos (la electrónica integrada) y de informática en cuanto al caso, sensores, actuadores, las computadoras de panel, los controladores de pulverización, los controladores de fertilización, cartografía y aplicación vía el satélite. El trabajo presente tiene como el objetivo integrar elementos de la electrónica integrada que empiezan de un estudio del caso de un controlador de pulverización nacional que usa microcontroladores dónde se propone, mientras buscando la mejora del desempeño y la optimización del *hardware*, la integración de los procesos más importantes y la integración de varios elementos periféricos a los microcontroladores en un FPGA (*Field Programmable Gate Array*).

Palabras-Clave: Agricultura de precisión. Electrónica integrada. Pulverizador. Controlador de pulverización. Microcontrolador. FPGA.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

1.1. Objetivos

Segundo Cruvinel (2000), o panorama mundial aponta claramente para um futuro em que a agricultura dependerá inevitavelmente da automação. A automação poderá auxiliar profundamente na sustentabilidade do processo produtivo como do desenvolvimento econômico e social. A aplicação da automação é ampla e existe potencial de contribuição em várias áreas. Esse projeto irá propor e implementar sistemas de automação e controle de pulverização em máquinas agrícolas, equipamento utilizado de forma generalizada e universal nas mais diversas culturas agrícolas.

O uso da automação e controle de pulverização ocorre a partir da utilização da eletrônica embarcada e viabiliza a aplicação localizada de insumos em quantidades variáveis e em tempos específicos, otimizando custos de produção e insumos, colaborando para que níveis de produtividade pré-estabelecidos para uma determinada cultura sejam obtidos.

Para atingir os objetivos gerais e específicos do projeto foi definido um estudo de caso de um sistema existente comercialmente, onde as atividades do projeto contribuíram para gerar sistemas de automação padrões, propondo o incremento de funções mais nobres e especializadas e integrando-as em uma FPGA (*Field Programmable Gate Array*). A utilização de FPGA pode contribuir também para integrar em uma única pastilha elementos periféricos e de interface, tais como: *latches*, multiplexadores, PWM (*Pulse Wave Modulation)*, *watchdog*, entre outros.

O projeto foi desenvolvido em parceria com alunos do Grupo de Iniciação Científica, que auxiliaram no projeto e simulação dos circuitos lógicos individuais.

1.2. Organização do Trabalho

Este documento está organizado da seguinte forma: no Capítulo 2 é apresentado um panorama das principais culturas agrícolas brasileiras. No Capítulo 3, são apresentados os principais conceitos relativos à agricultura de precisão, os tipos de sensores e atuadores utilizados na instrumentação agrícola, além de considerações a respeito de pulverização, tipos de pulverizadores e controladores de pulverização. No Capítulo 4, são discutidos os principais aspectos relacionados com a tecnologia FPGA, apresentando conceitos como estrutura interna, roteamento e reconfiguração. No Capítulo 5 é apresentado o estudo de caso para o qual o sistema desenvolvido se aplica. No Capítulo 6, são discutidos os detalhes da implementação em FPGA das funções propostas no estudo de caso, além de ser apresentado o teste prático para validação do sistema. Por fim, no Capítulo 7 é apresentada uma discussão sobre os resultados, bem como as considerações finais e sugestões para trabalhos futuros.

2. PANORAMA DAS PRINCIPAIS CULTURAS AGRÍCOLAS BRASILEIRAS

> Moderno, eficiente e competitivo, o agronegócio brasileiro é uma atividade próspera, segura e rentável. Com um clima diversificado, chuvas regulares, energia solar abundante e quase 13% de toda a água doce disponível no planeta, o Brasil tem 388 milhões de hectares de terras agricultáveis férteis e de alta produtividade, dos quais 90 milhões ainda não foram explorados. (MINISTÉRIO DA AGRICULTURA, 2004).

Segundo dados do Ministério da Agricultura (2004), o agronegócio é responsável por 33% do Produto Interno Bruto (PIB), 42% das exportações totais e 37% dos empregos brasileiros. Nos últimos anos, poucos países tiveram um crescimento tão expressivo no comércio internacional do agronegócio quanto o Brasil. Os resultados levaram a Conferência das Nações Unidas para o Comércio e Desenvolvimento (UNCTAD) a prever que o país será o maior produtor mundial de alimentos na próxima década.

A seguir é apresentado um breve panorama das principais culturas agrícolas brasileiras.

2.1. Algodão

Segundo o Instituto FNP (2006), os preços mundiais do algodão consolidaram uma trajetória de recuperação em 2006 e essa tendência deve persistir durante o ano de 2007.

As perspectivas para os maiores produtores mundiais são:

- China – Há restrições de várias categorias, com destaque para a redução de áreas agricultáveis, a limitada disponibilidade de água e o forte êxodo rural.
- EUA – Recentemente o país sofreu derrotas na Organização Mundial do Comércio (OMC), relativas a financiamentos subsidiados para os produtores. A indústria têxtil desacelera, oprimida pela pesada concorrência dos tecidos chineses.
- Paquistão – Apresenta queda no nível tecnológico de toda a cadeia cotonícola e desinteresse de investidores internacionais em virtude da complicada fase política que o país atravessa.

- Índia – não tem restrições ao crescimento da cotonicultura, exceto a barreira tecnológica, terra e clima favorável. Apresentou produção recorde na safra 2004/05 e poderá ser o único produtor a apresentar ciclo de crescimento sustentável a médio e longo prazo.
- Austrália – Apresenta queda acentuada em sua produção devido a uma seca que assola o país causando falta de água e prejudicando as lavouras, que são 95% irrigadas.

No Brasil, a maior produção ocorre nos estados de Mato Grosso e Bahia, onde há expansão da área de plantio. Os produtores do Paraná e Mato Grosso do Sul tendem a manter ou até reduzir a área plantada.

Quando começa o plantio da safra brasileira de algodão em novembro, os países do Hemisfério Norte estão concluindo as colheitas e durante a colheita brasileira, que ocorre de maio a julho, há entressafra no mercado internacional, o que geralmente reforça o preço do produto e acaba favorecendo o produtor brasileiro.

2.2. Café

Segundo o Instituto FNP (2006), a recuperação dos preços do café a partir da safra 2004/05 deve continuar e a estimativa é que até 2008 os preços da saca fiquem acima de US$ 100, uma vez que está prevista uma forte redução dos estoques mundiais.

O Brasil é o maior produtor mundial de café (33% de participação na produção mundial) e o segundo maior consumidor mundial (atrás apenas dos Estados Unidos), apresentando custos de produção inferiores aos dos principais concorrentes, como Colômbia e Vietnã. As boas condições climáticas e a bi anualidade da produção deverão levar a safra brasileira de 2006/07 a atingir o patamar de 43 milhões de sacas, que é praticamente quatro vezes maior que a produção dos principais concorrentes.

O principal estado produtor brasileiro é Minas Gerais, seguido de Espírito Santo e São Paulo. A produção mineira é praticamente o triplo da produção capixaba e quatro vezes maior que a produção paulista.

2.3. Cana-de-açúcar

Segundo o Instituto FNP (2006), a sustentação dos preços do petróleo em patamares recordistas está contribuindo fortemente para aumentar a rentabilidade do setor sucroalcooleiro. Além disso, as vendas de carros *flex fuel* (tecnologia que permite uso de álcool e/ou gasolina como combustível) têm superado as expectativas mais otimistas, o que deve gerar demanda crescente por álcool nos próximos anos.

O crescimento da área colhida e as condições climáticas geram o aumento da produção tanto de açúcar como de álcool e mantém o Brasil como líder mundial no setor. Para o álcool, o crescimento da demanda deve-se ao forte aumento da demanda doméstica com a popularização dos carros *flex* e à utilização de maiores quantidades de álcool misturado à gasolina. Para o açúcar, houve redução nos estoques mundiais e quebra da produção indiana (segunda maior do mundo), o que gerou crescimento da demanda e alta nos preços em virtude da redução dos estoques.

A tendência mundial é que uma proporção crescente da produção de cana deverá ser destinada para a produção de álcool, sendo estimada para a safra 2014/15 a quantidade de 90 bilhões de litros de álcool em todo o mundo. O álcool desempenha papel de substituto do petróleo, já que é crescente o consenso a respeito da redução das reservas mundiais do combustível fóssil, com preços permanecendo em patamares elevados, e das mudanças climáticas causadas pelo Efeito Estufa.

A produção brasileira está concentrada no Centro-Sul, sendo o estado de São Paulo o principal produtor. Existe forte tendência de aumento de áreas plantadas nas regiões oeste e noroeste do estado. Outras regiões de expansão da indústria canavieira continuarão sendo o Triângulo Mineiro, Campos (RJ), sul de Goiás e a bacia do Rio Paraná, em Mato Grosso do Sul.

2.4. Citros

Segundo o Instituto FNP (2006), a Região Sudeste, sobretudo o Estado de São Paulo, é uma referência em citricultura, com uma produção em torno de 348 milhões de caixa de laranja na safra 2004/05, por exemplo. As exportações nacionais de suco trouxeram para o País aproximadamente US$ 1 bilhão em 2004. No entanto, os produtores de tangerina vêm amargando prejuízos. De um lado, os preços da fruta se mantém baixos há anos. De outro, os custos de produção aumentam.

O Triângulo Mineiro também é outra importante região produtora nacional, sendo o estado norte-americano da Flórida o principal concorrente da citricultura brasileira.

Os estoques mundiais de suco de laranja estão baixos o suficiente para deixar o mercado internacional sensível á redução de oferta de matéria-prima, tanto nacional como norte-americana.

2.5. Milho

Segundo o Instituto FNP (2006), a forte estiagem que recentemente assolou a região Sul, o estado de Mato Grosso, parte de São Paulo e Minas Gerais, causou uma queda significativa na área plantada e nos índices de produtividade para as lavouras de milho. A necessidade de importação, principalmente de milho argentino, é quase inevitável em situações assim.

O estado do Paraná é o maior produtor de milho do País, sendo os Estados Unidos o grande produtor mundial.

2.6. Soja

Segundo o Instituto FNP (2006), a soja foi a grande vedete da explosão do agronegócio brasileiro nas safras 2002/03 e 2003/04. A cadeia produtiva da soja concentrava um terço de todo o agronegócio brasileiro, gerando divisas de mais de US$ 10 bilhões/ano. Embora vivendo um período de crise, em virtude da baixa cotação do dólar e de superprodução nos EUA (maior produtor mundial), a expectativa é de melhoria a partir do

segundo semestre de 2007, com a queda dos estoques norte-americanos e alta nos preços. A China (maior consumidor mundial) deve aumentar ainda mais a demanda mundial de soja, importando o produto do Brasil e Argentina, já que no Hemisfério Sul a colheita ocorre na entressafra norte-americana.

Maior produtor brasileiro é o estado de Mato Grosso, seguido de Paraná e Rio Grande do Sul. Existem previsões de que, num futuro próximo, o Brasil se torne o maior produtor mundial de soja.

3. PANORAMA DO USO DA ELETRÔNICA EMBARCADA EM PULVERIZADORES AGRÍCOLAS

3.1. A Definição de Eletrônica Embarcada

De acordo com Saraiva (2006), a eletrônica embarcada na agricultura é representada pelo uso em máquinas agrícolas de sensores, atuadores, computadores de bordo, *softwares* e sistemas de informações geográficas via satélite (GPS). Seus objetivos são monitorar a operação das máquinas, realizar o controle automático e registrar dados para análise posterior. As vantagens do uso da eletrônica embarcada são melhoria da qualidade da produção, redução das perdas e desgastes, ajuda no planejamento do negócio e proteção ao meio ambiente.

3.2. Breve Histórico da Eletrônica Embarcada na Agricultura Brasileira

Segundo Balastreire (2000), a introdução da eletrônica embarcada na agricultura brasileira ocorreu de forma gradativa. No período entre 1970 e 1980, a agricultura não contava com os benefícios eletrônicos, mesmo porque a eletrônica na época era pouco difundida no Brasil. Praticamente toda forma de manuseio com a terra e suas culturas eram feitas mecanicamente e de forma semi-artesanal, com máquinas que ofereciam poucos recursos e alto custo para os usuários.

A partir de 1980, surgiram as primeiras máquinas com alguma tecnologia e recursos, mas que ainda apresentavam um custo elevado. Havia ainda o paradigma de que uma máquina eletrônica não iria suportar as condições do campo, além da confiabilidade e precisão do sistema eletrônico serem duvidosas. Mas no decorrer da década o paradigma foi sendo quebrado com melhorias elétricas e mecânicas em relação aos primeiros protótipos de máquinas com módulos eletrônicos.

A partir de 1990, houve a explosão do uso da eletrônica na agricultura, com a implantação de sistemas suportando tecnologias cada vez mais confiáveis, com recursos prontos e de fácil utilização, além de apresentar resistência às intempéries e alta precisão. As máquinas sofreram uma considerável queda nos custos, tornando-as acessíveis a pequenos e médios produtores.

3.3. Conceitos sobre Pulverização

Pulverizar é reduzir um corpo em pequenos fragmentos, borrifar em gotas [JACTO01].

Pulverizador é todo equipamento capaz de produzir gotas, em função de uma determinada pressão exercida sobre a calda (produto químico mais água) [JACTO01].

O sucesso de uma boa pulverização depende de um bom pulverizador, bom produto químico, operador treinado, boa qualidade de água, pH ideal e condições de tempo favoráveis.

É da maior importância que o administrador agrícola providencie o treinamento e a capacitação de seus técnicos e operadores de máquinas, pois isso propicia práticas que melhoram a uniformidade, a exatidão e a segurança das pulverizações.

Aplicações mais precisas e mais uniformes podem reduzir a quantidade de ingredientes ativos requerida para um dado controle.

Aplicações bem executadas reduzem os custos dos tratamentos e minimizam os efeitos poluentes.

Os parâmetros ambientais (vento, umidade relativa do ar, temperatura, etc.), a máquina utilizada (tipo, regulagem, deslocamento) e a superfície a ser tratada (folha, caule, sementes, solo) são os principais elementos que determinam, em cada caso, um comportamento ideal do defensivo em sua trajetória até o alvo.

De maneira geral, um critério que conduz a resultados satisfatórios é o de se começar por determinar na planta, onde a praga e a doença se localizam. A partir desta informação realiza-se uma regressão, chegando ao órgão de aplicação do defensivo (bicos de pulverização) e finalmente a máquina (pulverizador) e suas regulagens.

Em resumo, uma boa pulverização requer uma tecnologia de aplicação de defensivos agrícolas e a colocação de um produto biologicamente ativo no alvo, em quantidade adequada, de forma econômica e com riscos mínimos de contaminação ambiental.

3.4. Tipos de Pulverizadores Utilizados nas Principais Culturas Agrícolas Brasileiras

De acordo com Santos e Santos Filho (2001), os pulverizadores utilizados nas principais culturas agrícolas brasileiras podem ser divididos em três categorias: pulverizadores autopropelidos, pulverizadores tratorizados e pulverizadores turbo atomizadores.

Os principais fabricantes nacionais de pulverizadores são Máquinas Agrícolas Jacto, com sede em Pompéia (SP) e Pulverizadores Montana, com sede em São José dos Pinhais (PR).

Os principais fabricantes internacionais de pulverizadores são Case (EUA), John Deere (EUA) e Hardi (Dinamarca).

Segue abaixo uma breve descrição de tipos de pulverizadores em cada uma das três categorias.

3.4.1. Pulverizadores Autopropelidos

Lobo Júnior (2004) caracterizou os pulverizadores autopropelidos como máquinas muito rápidas, de alto desempenho, conseguindo desenvolver velocidades operacionais entre 15 e 30 km/h durante a aplicação de agroquímicos. Em situações extremamente favoráveis, é possível com esses equipamentos conseguir alcançar velocidades operacionais próximas dos 40 km/h.

A cabine deve ser hermeticamente fechada, impedindo qualquer possibilidade de contaminação do operador por agroquímicos. Visibilidade, espaço, conforto e facilidade no controle dos sistemas eletrônicos são as palavras-chave para esses equipamentos.

As barras de pulverização podem ser instaladas na parte traseira ou na parte frontal dos pulverizadores autopropelidos. As barras de pulverização possuem total acionamento hidráulico e medem entre 15 até 48 metros de comprimento.

No Brasil são utilizados em grandes áreas com cultivo de grãos.

As Figuras 1 e 2 mostram modelos de pulverizadores autopropelidos.

Figura 1 – Pulverizador autopropelido CASE

Figura 2 – Pulverizador autopropelido JACTO

3.4.2. Pulverizadores Tratorizados

Segundo Santos e Santos Filho (2001), pulverizadores tratorizados são montados nos três pontos ou na barra de tração (neste caso também são conhecidos como pulverizadores tipo carreta) e são acionados pela tomada de potência do trator. São utilizados em áreas com cultivo de grãos e cana-de-açúcar e desenvolvem velocidade menor que os autopropelidos. Têm como componentes básicos depósitos com agitadores, bomba, filtros, reguladores de pressão e bicos.

As Figuras 3 e 4 apresentam modelos de pulverizadores tratorizados.

Figura 3 – Pulverizador tipo carreta JACTO

Figura 4 – Pulverizador três pontos MONTANA

3.4.3. Pulverizadores Turbo Atomizadores

De acordo com Lobo Júnior (2005), pulverizadores turbo atomizadores apresentam o sistema de assistência de ar, que é um ventilador de grande vazão, do tipo axial, que com o auxílio de defletores expele o ar na forma de um leque perpendicular ao movimento do pulverizador, que geralmente é tracionada (transportada) por um trator.

São utilizados em áreas de fruticultura.

As Figuras 5 e 6 mostram modelos de pulverizadores turbo atomizadores.

Figura 5 – Pulverizador turbo atomizador JACTO

Figura 6 – Pulverizador turbo atomizador HARDI

3.5. Conceitos sobre Agricultura de Precisão

Uma das únicas formas de o setor agropecuário se manter economicamente viável é reduzir o custo de produção, aumentando ao mesmo tempo a produtividade das culturas (BALASTREIRE, 2000).

A excelência em tecnologias de aplicação está fortemente relacionada com a Agricultura de Precisão.

Os recentes avanços tecnológicos que marcaram as últimas décadas permitiram que a agricultura desenvolvesse um verdadeiro arsenal de conhecimento e de técnicas de produção e de gerenciamento: é o caso, por exemplo, da biotecnologia, da engenharia genética e da eletrônica embarcada.

E foi exatamente tal avanço tecnológico que criou condições para a expansão da Agricultura de Precisão, a qual passou a contar com a disponibilidade de equipamentos capazes de realizar automaticamente o monitoramento, o mapeamento e a aplicação localizada de insumos, a taxas variáveis.

Segundo Balastreire (2000), a Agricultura de Precisão possui, como um dos seus principais objetivos, maximizar a eficiência na utilização dos insumos agrícolas, aplicando-os diferencialmente ao longo da lavoura, de acordo com as reais necessidades de cada ponto de área. Com isto, busca reduzir o custo da produção e o impacto ambiental causado pelo excesso de insumos utilizados.

Perosa (2000) considerou três etapas fundamentais para o uso dos conceitos da Agricultura de Precisão, que são: o mapeamento dos atributos dos solos e das plantas; o mapeamento da produtividade e a aplicação localizada de insumos.

No mapeamento dos atributos dos solos e das plantas, são feitos levantamentos sistemáticos das características que se pretende estudar, de forma georeferenciada, ou seja, cada amostra é retirada de um ponto do qual se conhecem a longitude, latitude e altitude. Por esse motivo a implantação do sistema de posicionamento global (GPS) tem sido considerada um marco no desenvolvimento e na utilização dos conceitos da Agricultura de Precisão, uma vez que, o tratamento georeferenciado de pontos amostrados deixou de ser uma tarefa difícil e complicada.

No mapeamento da produtividade das culturas, há necessidade de utilização de colhedoras instrumentadas com algum tipo de sensor, capaz de medir a quantidade de material que está entrando na máquina a cada instante, e com um sistema de posicionamento que indique com uma acurácia ao redor de um metro, a localização da colhedora no campo. Com a quantidade colhida e a posição da máquina a cada instante, é possível traçar-se o mapa de produtividade da cultura estudada.

Na aplicação localizada de insumos, determina-se a quantidade de insumos que se deverá aplicar em cada talhão estudado, em função da análise dos dados obtidos na fase de mapeamento de atributos dos solos e das plantas, do conhecimento agronômico da cultura a implantar, e da experiência prévia obtida naquele campo em anos anteriores. As quantidades recomendadas são colocadas em um mapa de prescrição, armazenado no microcomputador que normalmente faz parte do sistema de controle para a aplicação localizada de insumos.

3.6. Tipos de Sensores Utilizados na Instrumentação Agrícola

Um sensor é um dispositivo que detecta um sinal, condição física ou produto químico. Geralmente é composto de um transdutor (converte um tipo de energia em outra) e uma parte que converte a energia resultante em um sinal elétrico.

Os tipos clássicos de sensores utilizados na instrumentação agrícola são os ultrasônicos, indutivos, capacitivos e fotoelétricos. Segue abaixo uma descrição desses tipos de sensores.

3.6.1. Sensores de Distância Ultrasônicos

São utilizados para detectar ou medir a posição de determinado objeto através da emissão de pulsos de ultra-som.

Os sensores trabalham com um transdutor ultrasônico usado para transmissão e recepção de sinais acústicos. Em cada ciclo, serão transmitidos entre 300 e 5000 sinais ultrasônicos especialmente codificados por segundo, dependendo da distância entre sensor e alvo. Os pulsos são então refletidos de volta pelo alvo, recebidos e decodificados pelo sensor. Através da medida, compensada pela temperatura, do tempo gasto pelo sinal acústico, a distância ao alvo é determinada com alto grau de precisão. A medida resultante pode ser amostrada na forma de um sinal de saída analógico ou digital.

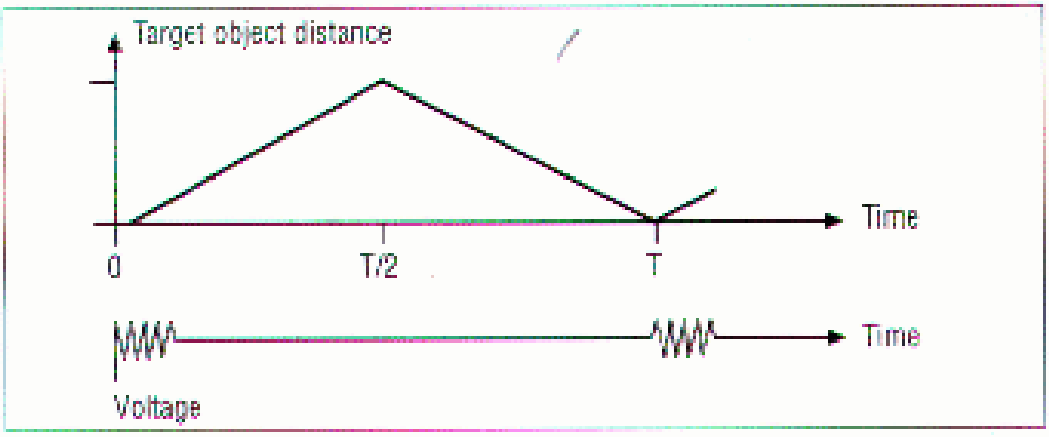

Figura 7 – Detecção do alvo no sensor ultrasônico

A Figura 7 mostra o tempo gasto pelo sinal acústico. O diagrama mostra como o pulso viaja do transdutor para o alvo, é refletido ao tempo T/2 e alcança o transdutor ao tempo T. Abaixo está um diagrama da voltagem no transdutor ultrasônico. O tempo decorrido T é diretamente proporcional à distância ao objeto a (a = CT/2, onde C é a velocidade do som).

Na Figura 8 são mostrados diferentes modelos de sensores ultrasônicos.

Figura 8 – Modelos de sensores ultrasônicos

3.6.2. Sensores de Proximidade Indutivos

Os sensores de proximidade são equipamentos eletrônicos capazes de detectar a aproximação de peças, componentes, elementos de máquinas, entre outras funções.

A detecção pode ocorrer sem que haja o contato físico entre o acionador e o sensor, aumentando a vida útil do sensor por não possuir peças móveis sujeitas a desgastes mecânicos.

O princípio de funcionamento (Figura 9) baseia-se na geração de um campo eletromagnético de alta freqüência, que é desenvolvido por uma bobina ressonante instalada na face sensora.

A bobina faz parte de um circuito oscilador, que em condição normal, gera um sinal senoidal. Quando um metal aproxima-se do campo, este por correntes de superfície, absorve a energia do campo, diminuindo a amplitude do sinal senoidal gerado no oscilador.

A variação de amplitude deste sinal é convertida em uma variação contínua, que comparada com um valor padrão, passa a atuar no estágio de saída.

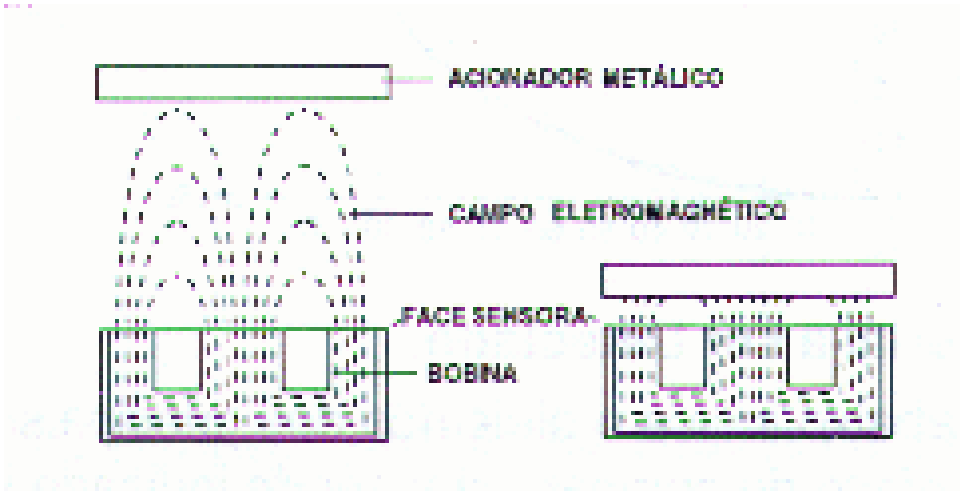

Figura 9 – Princípio de operação do sensor indutivo

A Figura 10 mostra alguns modelos de sensores indutivos.

Figura 10 – Modelos de sensores indutivos

3.6.3. Sensores de Proximidade Capacitivos

Os sensores de proximidade capacitivos são equipamentos eletrônicos capazes de detectar a presença ou aproximação de materiais orgânicos, plásticos, líquidos, madeiras, papéis, metais, etc.

O princípio de funcionamento (Figura 11) baseia-se na geração de um campo elétrico, gerado por um oscilador controlado por capacitor.

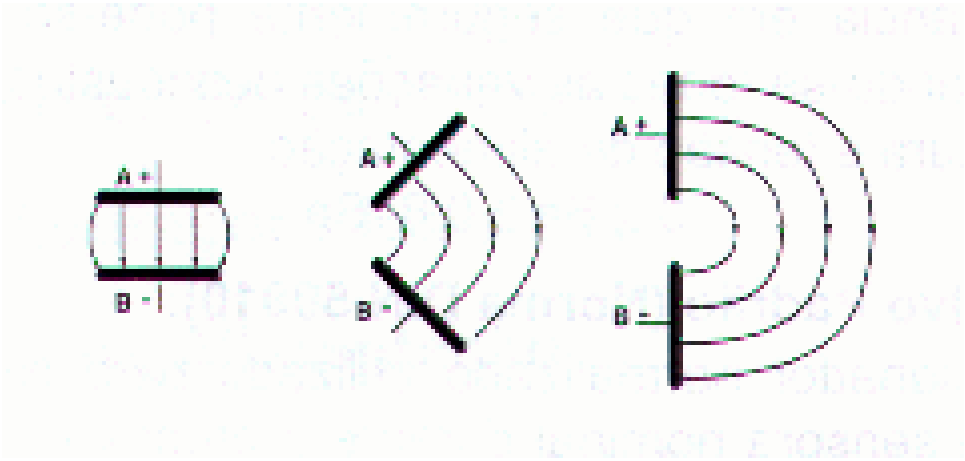

Figura 11 – Princípio de operação do sensor capacitivo

O sensor é formado por duas placas metálicas, carregadas com cargas elétricas opostas, de forma a projetar o campo elétrico para fora do sensor, formando assim um capacitor que possui como dielétrico o ar.

Quando um material aproxima-se da face sensora, ou seja, do campo elétrico, o dielétrico do meio se altera, alterando também o dielétrico do capacitor frontal do sensor. Como o oscilador do sensor é controlado pelo capacitor frontal, quando aproximamos um material a capacitância também se altera, provocando uma mudança no circuito oscilador. Esta variação é convertida em um sinal contínuo, que comparado comum valor padrão, passa a atuar no estágio de saída.

A Figura 12 apresenta modelos de sensores capacitivos.

Figura 12 – Modelos de sensores capacitivos

3.6.4. Sensores Fotoelétricos

Os sensores fotoelétricos, também conhecidos por sensores ópticos, manipulam a luz de forma a detectar a presença do acionador, que na maioria das aplicações é o próprio produto.

Seu princípio de funcionamento baseia-se na transmissão e recepção de luz infravermelha (invisível ao ser humano), que pode ser refletida ou interrompida por um objeto a ser detectado.

Os sensores fotoelétricos são compostos por dois circuitos básicos: um responsável pela emissão do feixe de luz, denominado transmissor, e outro responsável pela recepção do feixe de luz, denominado receptor.

O transmissor envia o feixe de luz através de um fotodiodo, que emite flashes, com alta potência e curta duração, para evitar que o receptor confunda a luz emitida pelo transmissor com a iluminação ambiente.

O receptor é composto por um fototransistor sensível a luz, que em conjunto com um filtro sintonizado na mesma freqüência de pulsação dos flashes do transmissor, faz com que o receptor compreenda somente a luz vinda do transmissor. A Figura 13 mostra alguns modelos de sensores fotoelétricos.

Figura 13 – Modelos de sensores fotoelétricos

3.7. Tipos de Atuadores Utilizados na Instrumentação Agrícola

Atuador é um elemento que produz ações, atendendo a comandos que podem ser manuais ou automáticos. Existe uma infinidade de elementos atuadores. Dentre os mais utilizados na automação de instrumentos agrícolas estão: atuadores de movimento induzido por cilindros pneumáticos ou cilindros hidráulicos; motores hidráulicos e elétricos e dispositivos rotativos com acionamento de diversas naturezas. As Figuras 14 e 15 mostram alguns desses atuadores. Na automação da atividade de pulverização, os atuadores clássicos são os reguladores de pressão de pulverização, geralmente baseados na utilização de motores elétricos.

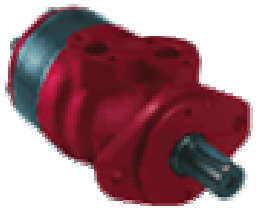

Figura 14 – Exemplo de motor hidráulico

Figura 15 – Exemplo de regulador de pressão elétrico

3.8. Sistema de Posicionamento Global (GPS)

Segundo Hurn (1999), o Sistema de Posicionamento Global, conhecido pela sigla GPS (*Global Positioning System*), é fruto de um empreendimento de 12 bilhões de dólares do Departamento de Defesa dos Estados Unidos e é obtido através de um conjunto de 28 satélites trabalhando simultaneamente. Nessa constelação de satélites 24 estão efetivamente ativados. Quatro são utilizados para assegurar confiabilidade de funcionamento contínuo e por longo tempo do sistema GPS. Esses quatro satélites são utilizados como elementos redundantes no sistema, assegurando o funcionamento do sistema mesmo com ocorrência de falhas em um ou mais satélites. Essa redundância é necessária não só para manter a credibilidade e eficiência do sistema. Mas também para prever e evitar falhas ocasionais sem a necessidade de realização de manutenções corretivas, pois essa manutenção de satélites pode ser inviável tanto financeiramente quanto tecnicamente. Os satélites orbitam a Terra e geram sinais que permitem apontar posições em qualquer lugar do mundo, de forma ininterrupta, 24 horas por dia.

O sistema foi considerado totalmente operacional em 1995 e é controlado pelo Departamento de Defesa dos Estados Unidos, que fornece dois tipos de serviço: *Standard* e *Precision*. Os serviços estão disponibilizados em qualquer parte do mundo.

O sistema está dividido em três partes: espacial, de controle e utilizador. O segmento espacial é composto pela constelação de satélites. O segmento de controle é formado pelas

estações terrestres dispersas pelo mundo ao longo da Zona Equatorial. O segmento do utilizador consiste num receptor que capta os sinais emitidos pelos satélites.

O GPS é baseado no raio de ação dos satélites. Isto significa que qualquer posição na Terra pode ser descoberta medindo-se a distância a partir de um grupo de pelo menos três satélites no espaço. Os satélites atuam como pontos de referência para triangular a posição em algum lugar da Terra.

O GPS é considerado o mais exato sistema de navegação global dos dias atuais. Sua precisão pode ser aumentada utilizando uma técnica conhecida como GPS diferencial. Com ela, o GPS pode alcançar precisões nas medidas menores do que um metro. E isto possibilita a utilização do GPS em diferentes aplicações.

O uso do GPS na topografia já está bastante difundido e permite o cálculo de posições com precisão de centímetros.

Segundo Perosa (2000), na agricultura o GPS diferencial permite medição precisa da área dos talhões e também o controle de pulverização, uma vez que com o auxílio de computadores e programas adequados, são realizadas duas atividades: o rastreamento da área e o mapeamento do grau de infestação de pragas para que posteriormente o controlador de pulverização controle a quantidade de produtos químicos necessários para a aplicação de defensivos. Em tratores e pulverizadores, o GPS também é utilizado em dois sistemas: a barra de luz (Figuras 16 e 19), que é um equipamento utilizado para a orientação de um veículo em faixas adjacentes, diminuindo a sobreposição da operação agrícola entre passadas consecutivas e o *auto-pilot*, que é um equipamento que guia o veículo sem a necessidade de intervenção do operador.

Na adubação, devido aos diferentes níveis de fertilidade do solo, pode-se realizar o mapeamento da área através do GPS. Podem então ser utilizados controladores de adubação que dosam a quantidade de adubo em função do mapeamento realizado.

As Figuras 17 e 18 mostram exemplos de receptores GPS.

Figura 16 – Barra de luz com display gráfico

Figura 17 – Receptores GPS

Figura 18 – Receptor GPS miniatura

Figura 19 – Barra de luz com display de caracteres

3.9. Tipos de Controladores Eletrônicos de Pulverização

Os controladores eletrônicos de pulverização utilizados na agricultura geralmente atuam sobre o comando de pulverização (também conhecido como comando de defensivo) e/ou bomba de pulverização. O comando de pulverização (Figura 21) permite a regulagem da pressão do circuito de pulverização bem como a distribuição do líquido nos segmentos da barra de pulverização. A bomba de pulverização gera o fluxo de líquido para o comando a partir do tanque. O tanque tem a função de armazenar, proteger e transportar o líquido a ser pulverizado.

A Figura 20 mostra os elementos básicos de um circuito de pulverização.

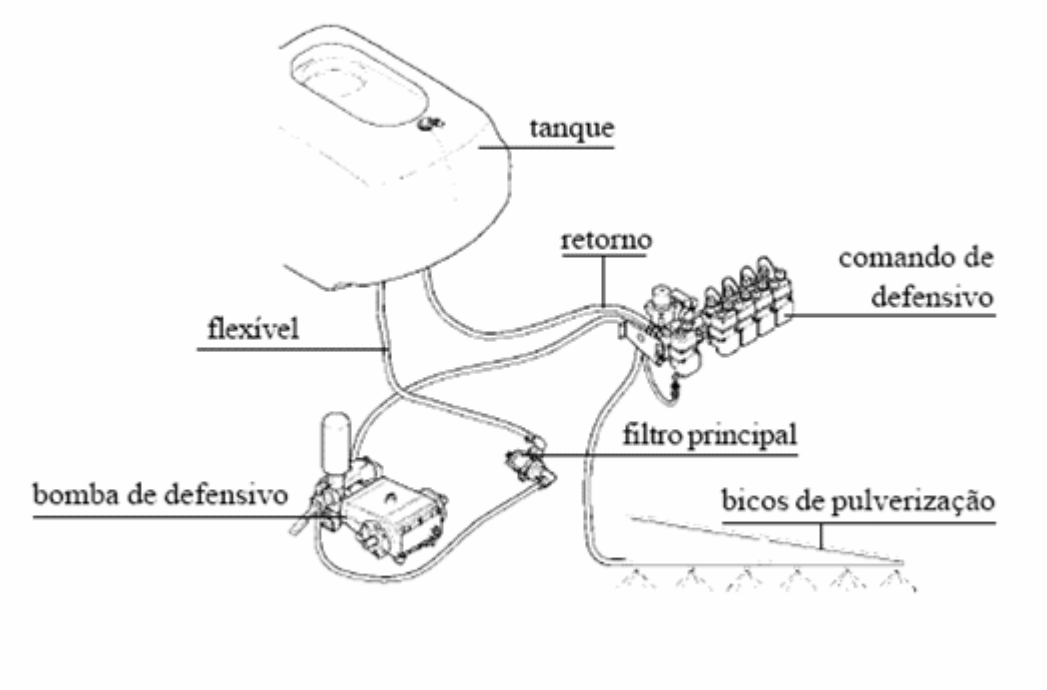

Figura 20 – Exemplo de circuito de pulverização

Figura 21 – Comando de pulverização JACTO

A função básica dos controladores eletrônicos de pulverização é manter constante a quantidade de líquido por unidade de área independentemente das variações da velocidade de trabalho do pulverizador.

A medição da vazão de líquido e da velocidade de trabalho do pulverizador é realizada a partir de sinais gerados por sensores indutivos.

O operador necessita informar ao controlador, via teclado, a dose desejada de defensivo por hectare e o controlador irá fazer a regulagem automática da pressão dependendo da velocidade de trabalho para garantir a dose ajustada pelo operador [JACTO02].

Os controladores de pulverização fornecem aos agricultores dados importantes para administrar a operação de aplicação de defensivos. Os valores de velocidade de trabalho, média da pulverização, tempo de operação, distância percorrida pulverizando, litros aplicados por minuto, áreas parcial e total tratadas, volume parcial aplicado e volume total aplicado são apresentados via display.

Além das funcionalidades básicas citadas acima, os controladores de pulverização também possuem interfaces para mapas de aplicação (GPS), cartões de memória, comunicação serial RS232, RS485, CAN *bus* (*Controller Area Network)* e *wireless*, por exemplo. O sistema operacional pode ser *Windows*, *Linux* ou proprietário. As funcionalidades variam de acordo com o modelo e o fabricante.

Alguns exemplos de fabricantes de controladores de pulverização são Máquinas Agrícolas Jacto (Brasil), Arag (Itália), Dickey John (EUA), TeeJet (EUA) e Case (EUA).

As Figuras 22 a 27 mostram modelos de controladores de pulverização.

Figura 22 – Controlador ARAG BRAVO300

Figura 23 – Controlador ARAG SKIPPER

Figura 24 – Controlador DICKEY JOHN LAND MANAGER II

Figura 25 – Controlador CASE SCS4400

Figura 26 – Controlador TEEJET 844AB

Figura 27 – Controlador JACTO JMC1000/4

4. TECNOLOGIA FPGA

Seguem abaixo alguns tópicos sobre a tecnologia FPGA, que foi empregada no estudo de caso proposto. Um breve histórico é apresentado como tópico introdutório, sendo os demais tópicos relacionados com a estrutura interna de um FPGA, técnicas e principais conceitos relativos ao assunto.

4.1. Histórico

A criação do transistor em 1947 permitiu o desenvolvimento da indústria eletrônica, pois possibilitou o desenvolvimento de circuitos mais velozes, mais confiáveis, ocupando menor espaço físico, gerando menos calor dissipado. Tudo isso aliado a custos reduzidos.

Com o surgimento dos circuitos integrados em 1961, passou a ser possível a junção de vários transistores em uma única pastilha, permitindo então integrar circuitos relativamente grandes em um único componente. Os chips microprocessadores e microcontroladores são frutos da evolução da tecnologia de circuitos integrados. Os microcontroladores apresentam maior auto-suficiência em relação aos microprocessadores, pois possuem memória, interfaces de entrada/saída e conversores de sinais embutidos. Os microcontroladores são amplamente utilizados em eletrodomésticos, telefones celulares, instrumentação médica, eletrônica embarcada, etc.

Em muitos sistemas digitais como, ambientes de tempo real, os microcontroladores não apresentam desempenho satisfatório. Ordonez (2003) afirma que a utilização de processadores digitais de sinais (DSP) e processadores de aplicação específica, melhora a performance desses sistemas.

Uma técnica relativamente nova surgida em 1985 é a utilização da tecnologia FPGA (*Field Programmable Gate Array*) para implementar circuitos de aplicação específica. A rápida evolução da velocidade e capacidade dessa tecnologia, permite a implementação de circuitos e arquiteturas cada vez mais complexas integradas em uma única pastilha.

De acordo com Silva (2006), a utilização da tecnologia FPGA apresenta como principal vantagem a possibilidade de modificação da estrutura de *hardware* do sistema através de um processo denominado reconfiguração, o qual permite o desenvolvimento incremental, correção de erros de projeto, além da adição de novas funções de *hardware.*

Martins (2003) afirma que dentre os principais fabricantes de FPGA destacam-se a Altera, Xilinx, Actel e Atmel, sendo que as famílias de dispositivos e suas respectivas aplicações variam de acordo com o fabricante.

Segundo Gericota (2003), alguns exemplos de aplicações da tecnologia FPGA são eletrônica de consumo, telefonia celular, sistemas de telecomunicações, processadores embarcados e sistemas de satélite para fins civis e militares.

4.2. Estrutura Interna de um FPGA

Segundo Ordonez (2003), os FPGAs são circuitos programáveis formados por conjuntos de células lógicas em arranjo matricial. Cada célula contém capacidade computacional para implementar funções lógicas e realizar roteamento para comunicação entre elas. O roteamento e as funções lógicas das células são configuráveis via *software*.

De acordo com Martins (2003), a estrutura básica de um FPGA (Figura 28) é composta pelos seguintes elementos:

1) CLB (*Configurable Logic Block*): unidade lógica configurável.

2) IOB (*In/Out Block*): unidade de entrada e saída.

3) SB (*Switch Box*): unidade de conexão entre os diversos CLBs.

4) Canais de roteamento: interligam as unidades de conexão para formar a rede de interconexão programável.

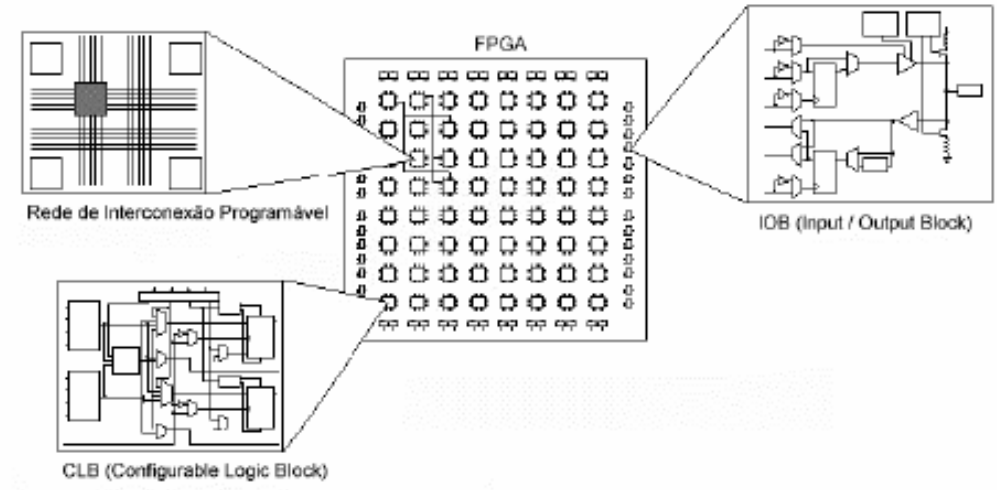

Figura 28 – Elementos básicos de um FPGA

Ordonez (2003) cita as seguintes configurações para a arquitetura interna de um FPGA:

1) Matriz simétrica: possui canais de roteamento verticais e horizontais com grande flexibilidade.

2) PLD hierarquico: matriz de blocos lógicos interligados e que podem ser agrupados entre si.

3) *Row-based*: blocos lógicos em disposição horizontal com área dedicada para o roteamento.

4) *Sea of gates*: blocos logicos de complexidade pequena dispostos em grande numero por unidade de area. Nao há area dedicada para o roteamento.

4.3. Conceitos sobre Roteamento

Segundo Ordonez (2003), a interconexão entre os blocos lógicos através de uma rede de camadas de metal é chamada de roteamento. Fisicamente, transistores controlados por bits de memória (PIP) ou chaves de interconexão (*switch matrix*) são responsáveis pelas conexões físicas entre os blocos lógicos.

Podem ser definidos ainda alguns elementos básicos presentes na malha de roteamento da familia XC4000 (Xilinx), os quais seguem abaixo descritos.

1) Conexões globais: rede de interconexão entre linhas e colunas ligadas por meio de chaves de interconexão.

2) Matriz de conexão: chaves de interconexão para o roteamento entre blocos lógicos atraves de conexões globais.

3) Conexões diretas: interligação entre CLBs vizinhos que permite menores tempos de atraso, uma vez que não faz uso de recursos globais de roteamento.

4) Linhas longas: conexões que interligam sinais longos e percorrem todo o circuito sem passar pelas matrizes de conexão.

4.4. Conceitos sobre Reconfiguração

A melhoria na performance dos computadores, principalmente nas aplicações em tempo real, exige cada vez mais alto poder de processamento. Ordonez (2003) afirma que ao acoplar-se um dispositivo programável FPGA a um processador de propósito geral (GPP), torna-se possível a exploração do potencial da computação reconfigurável.

Uma definição para computação reconfigurável seria a seguinte:

Computação reconfigurável representa uma nova idéia em filosofia de computação, na qual algum agente de *hardware* de propósito geral é configurado para realizar uma tarefa específica, mas pode ser reconfigurado sob demanda para realizar outras tarefas específicas [RECONF].

A reconfiguração permite ao projetista criar novas funções, com a execução de operações com número de ciclos e de unidades funcionais complexas consideravelmente menores do que é necessário em GPPs.

Ordonez (2003) classificou os métodos de reconfiguração em:

1) Reconfiguração total: alteração total do dispositivo reconfigurável

2) Reconfiguração parcial: somente parte do dispositivo pode ser reconfigurada. Pode ser de dois tipos: não-disruptiva, onde a parte do sistema que não sofre a reconfiguração parcial permanece completamente funcional durante o processo de reconfiguração; disruptiva, onde a reconfiguração parcial afeta outras partes do sistema que não sofrem a reconfiguração, o que necessariamente deve gerar uma parada no funcionamento desse sistema.

3) Reconfiguração dinâmica: reconfiguração em tempo de execução. Não há necessidade de reiniciar o circuito ou remover elementos reconfiguráveis para programação.

4) Reconfiguração extrínseca: reconfigura parcialmente o sistema, considerando cada FPGA como uma unidade atômica de reconfiguração.

5) Reconfiguração intrínseca: reconfigura parcialmente cada unidade FPGA que compõe o sistema.

4.5. Desenvolvimento de Aplicações em Sistemas Computacionais Reconfiguráveis

Para o desenvolvimento de aplicações em sistemas computacionais reconfiguráveis, é interessante compreender as diferenças entre os sistemas computacionais tradicionais e os sistemas computacionais reconfiguráveis.

Segundo Martins (2003), temos:

1) Sistemas computacionais tradicionais: as aplicações para sistemas computacionais tradicionais são divididas em dois tipos de implementação: *hardware* e *software*. A escolha do tipo de implementação é feita com base no tipo de problema, disponibilidade de tecnologia, custos e no tempo de execução exigido pela aplicação. A solução em *software* é baseada na utilização de algoritmos codificados em alguma linguagem de programação. O programa fonte resultante é executado em nível de sistema operacional ou de processador/microcontrolador. A flexibilidade da solução em *software* é grande, uma vez que podem ser implementadas várias funcionalidades que variam em função do *software* que está sendo executado. A solução em *hardware* envolve a utilização de técnicas de engenharia eletrônica ou engenharia de *hardware* e geralmente ao invés de um algoritmo, é implementado somente um circuito eletrônico com funcionalidade fixa, o que acarreta perda de flexibilidade em relação à solução em *software*. Porém, em termos de desempenho, a solução em *hardware* é melhor, uma vez que não são necessárias a leitura e decodificação de instruções.

2) Sistemas computacionais reconfiguráveis: as aplicações para sistemas computacionais reconfiguráveis apresentam implementação intermediária entre *hardware* e *software*, que procura mesclar o desempenho de aplicações em *hardware* com a flexibilidade de aplicações em *software*. Inicialmente, o desenvolvimento de aplicações em *hardware* reconfigurável era semelhante ao das aplicações tradicionais em *hardware* fixo, mas, com a evolução da tecnologia FPGA e dos ambientes de auxilio ao projeto, acabou ocorrendo a aproximação entre o desenvolvimento de soluções em sistemas reconfiguráveis e o desenvolvimento de *software*, sendo desejável como resultado final um programa e não um circuito.

Martins (2003) cita que as abordagens para o desenvolvimento de soluções em sistemas reconfiguráveis são:

1) Diagramas esquemáticos: a solução implementada em *hardware* reconfigurável é desenvolvida utilizando ferramentas que a partir da captura do esquemático geram os bits de configuração de um determinado dispositivo reconfigurável. O esquemático é uma representação visual de portas e componentes lógicos combinacionais e seqüenciais do *hardware* que faz parte da solução implementada no sistema reconfigurável. Geralmente nesse tipo de ferramenta é possível usar componentes mais complexos já prontos, como somadores, ULAs (Unidade Lógica Aritmética) e memórias, por exemplo. Essa abordagem geralmente não é aplicada a projetos grandes por causa da dificuldade que existe em se fazer uma representação gráfica de muitos componentes.

2) Linguagens de descrição de *hardware*: utiliza linguagens de descrição de *hardware* como VHDL - *VHSIC (Very High Speed Integrated Circuits) Hardware Description Language* ou Verilog. Programas escritos nessas linguagens são compilados usando-se ferramentas que podem gerar os bits de configuração para um dispositivo específico. VHDL é uma linguagem estruturada que oferece a possibilidade de descrever o *hardware* e este ser simulado antes de sua síntese, facilitando a validação ou verificação, tanto em termos de funcionamento quanto em termos de tempos de atraso dos componentes e desempenho, sem a necessidade da prototipação do sistema. Um programa em VHDL pode ser escrito basicamente usando dois tipos de descrição: estrutural e comportamental. Na descrição estrutural, a organização física e topológica do sistema é descrita. Isso quer dizer que são especificadas as entradas e/ou saídas, os componentes lógicos, a interligação deles e os sinais que compõem o sistema. Existem bibliotecas em VHDL que contêm entidades que podem ser usadas nos projetos, como somadores, contadores, multiplicadores. Na descrição comportamental, é necessário descrever somente o comportamento. Nessa abordagem, são descritas as funções do sistema. Um programa que utiliza esse tipo de descrição possui o mesmo formato de um programa fonte escrito em uma linguagem de alto nível. Essa abordagem aumenta a facilidade de desenvolvimento do sistema. No entanto, os sistemas gerados a partir desse tipo de descrição podem não ser tão otimizados em questões de desempenho e área de dispositivo ocupada quanto os descritos em VHDL estrutural.

5. ESTUDO DE CASO

O estudo de caso utiliza o controlador eletrônico de pulverização para pulverizadores autopropelidos JMC1000/4 (Figura 29), fabricado pela empresa Máquinas Agrícolas Jacto S/A.

As funções de processamento do controlador JMC1000/4 são implementadas em microcontroladores.

Figura 29 – Controlador JACTO JMC1000/4

5.1. Objetivos

O objetivo do estudo de caso é propor a integração das funções de processamento mais nobres do controlador em um FPGA. Além disso, é proposta também a integração de elementos periféricos aos microcontroladores em um FPGA.

A integração leva em conta dois aspectos: a melhoria de desempenho e a otimização do hardware do controlador. Além disso, o FPGA com as funções embutidas resultante poderia ser utilizado como módulo universal para o projeto de novos modelos de controladores de pulverização.

5.2. Função do Controlador JMC1000/4

O controlador de pulverização JMC1000/4 tem como função básica manter a dose de defensivo em litros por hectare (L/ha) desejada pelo usuário, independente das variações de velocidade do pulverizador.

A equação básica da pulverização é:

*Dose(L/ha)= (Vazão(L/min)*600)/(Velocidade(Km/h)*Comprimento da barra(m))*

O controlador faz a leitura de pulsos de dois sensores indutivos acoplados nas rodas do pulverizador para calcular a velocidade de deslocamento em quilômetros por hora (Km/h).

A vazão em litros por minuto (L/min) é calculada a partir da leitura de pulsos de um sensor indutivo localizado no medidor de vazão, o qual está localizado no comando de pulverização.

O comprimento da barra de pulverização em metros (m) é digitado pelo usuário via teclado.

A dose desejada em L/ha é digitada pelo usuário via teclado.

O controlador calcula então a dose real em L/ha a partir da equação mostrada e compara com o valor desejado. Se diferente, o controlador atua no regulador de pressão do comando de pulverização aumentando ou diminuindo a pressão e, consequentemente, a vazão nas barras de pulverização, para manter sempre a dose em L/ha desejada pelo usuário. Isso garante economia de produto, cobertura eficiente da lavoura e riscos mínimos de contaminação ambiental.

O controlador também faz a leitura de um sensor de nível de líquido no tanque de pulverização e informa ao usuário quando um nível mínimo é atingido.

5.3. Detalhes de *Hardware* do Controlador JMC1000/4

Segue abaixo uma descrição do hardware do controlador JMC1000/4.

1 Microcontrolador MSP430F149 (escravo) = tratamento do *display* gráfico 240x128

1 Microcontrolador MSP430F149 (mestre) = varredura do teclado, leitura dos sinais dos sensores para cálculo da vazão em L/min, velocidade em Km/h e dose em L/ha para atuar no regulador de pressão do comando de pulverização, comunicação serial RS485 com módulos auxiliares e salvamento em EEPROM *(Electrically-Erasable Programmable Read Only Memory*).

1 Timer LMC555 = *watchdog* do microcontrolador mestre

2 Multiplexadores HC4051= varredura do teclado

2 Memórias EEPROM 93LC66 = salvamento de constantes de calibração

1 Driver RS485 ADM483 = comunicação serial

3 Drivers +3.3V / 5V HCT245 = interface para barramento do *display*

2 Drivers +12V / 3.3V TLC3704 = interface para entrada dos sinais dos sensores

1 Array de transistores ULN2003 = interface

2 Drivers 3V3 / 12V IR2010S = interface para saída de sinais

5.4. Definição e Projeto do Sistema Proposto

Conforme afirmado no item 5.2, a função do controlador JMC1000/4 é manter a dose aplicada de defensivo independente das variações de velocidade do pulverizador. Para realizar essa tarefa, o controlador realiza a leitura dos pulsos dos sensores de roda e vazão para calcular respectivamente, a velocidade de deslocamento do pulverizador e a vazão de líquido nas barras de pulverização. O controlador faz também a leitura de um sensor de nível mínimo de líquido no tanque de pulverização para alertar o usuário. Essas funções, por estarem relacionadas com o desempenho do sistema de pulverização, são consideradas funções nobres

e por isso foram escolhidas para serem sintetizadas, projetadas e implementadas em um FPGA.

Para permitir a navegação entre os menus mostrados no display e a digitação de valores, o controlador realiza a leitura das chaves que formam o teclado momentâneo. Para saber o estado das chaves que ligam e desligam os segmentos da barra de pulverização, o controlador realiza a leitura do teclado de segmentos. Além disso, existe uma função de gerenciamento do controlador (*watchdog*), que permite a recuperação do mesmo em caso de falha do microcontrolador mestre, evitando sua entrada em estado de *deadlock* para não prejudicar a operação de pulverização. Essas funções são chamadas periféricas, pois não estão diretamente relacionadas com o desempenho do sistema de pulverização, mas são consideradas importantes do ponto de vista operacional e, por esse motivo, também foram escolhidas para serem sintetizadas, projetadas e implementadas em um FPGA.

5.4.1. Descrição das Funções Periféricas

As funções periféricas de *watchdog* e leitura de teclado foram integradas e sintetizadas no FPGA. Segue abaixo a descrição das funções periféricas.

1) Sintetizar e projetar a função de *watchdog* ("cão de guarda") do microcontrolador mestre, realizada pelo timer 555. O microcontrolador mestre gera um trem de pulsos com freqüência de aproximadamente 1 kHz durante a execução do código que inibe, via circuito a transistor, o oscilador astável configurado com o timer 555. Caso o microcontrolador deixe de executar o código por ruídos ou interferências e entre em estado de *deadlock*, o trem de pulsos deixa de ser gerado e o oscilador gera pulsos no pino de *reset* do microcontrolador a cada 1 segundo. A idéia é que o FPGA receba o trem de pulsos gerado pelo microcontrolador e, detectando sua ausência, atue no pino de *reset* do mestre. O *reset* do microcontrolador escravo é realizado pelo mestre.

A configuração atual do circuito de *watchdog* segue na Figura 30.

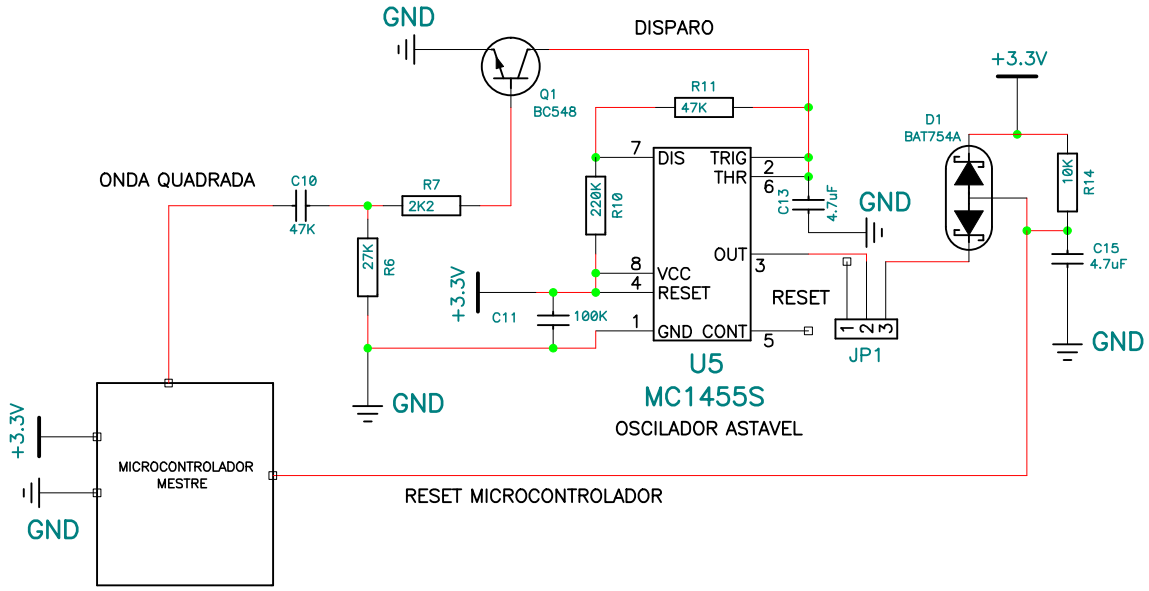

Figura 30 – Circuito de *watchdog* do Controlador JMC1000/4

A configuração proposta utilizando FPGA segue na Figura 31.

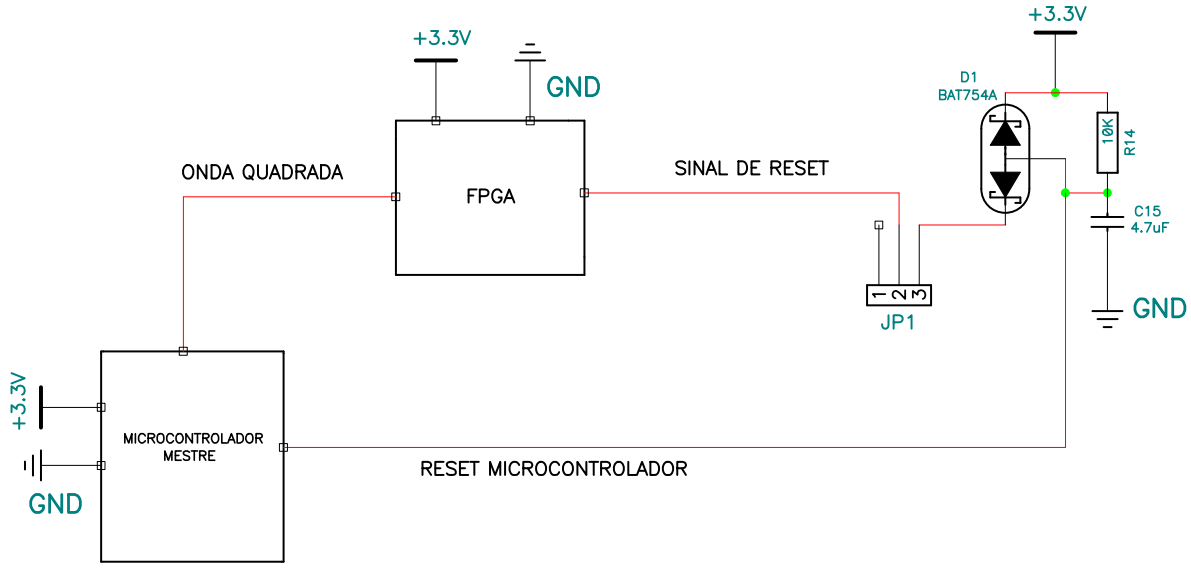

Figura 31 – Circuito de *watchdog* com FPGA

2) Sintetizar e projetar a função de leitura de teclado via FPGA. No *hardware* atual do controlador, são utilizados dois multiplexadores de 8 canais HC4051 que recebem os sinais do teclado e do sensor de nível do tanque de pulverização. Os multiplexadores estão divididos da seguinte forma:

Chaves momentâneas e sensor de nível:

Canal 0 = chave mais
Canal 1 = chave menos
Canal 2 = chave automático/manual
Canal 3 = sensor de nível
Canal 4 = chave função p/ baixo
Canal 5 = chave zerar
Canal 6 = chave função p/ cima
Canal 7 = chave memória

Chaves segmentos da barra de pulverização:

Canal 0 = chave segmento 5
Canal 1 = chave segmento 6
Canal 2 = nc
Canal 3 = chave segmento 4
Canal 4 = chave segmento 1
Canal 5 = chave alívio rápido
Canal 6 = chave segmento 2
Canal 7 = chave segmento 3

Para cada multiplex, o software associa uma variável de 8 bits em RAM (*Random Access Memory*) onde, para cada bit, o valor 0 significa chave desacionada e o valor 1 significa chave acionada.

No circuito implementado o FPGA realiza a varredura do teclado (chaves momentâneas e chaves dos segmentos) e entrega para o microcontrolador mestre, via barramento de 8 bits, o status das mesmas.

As configurações propostas utilizando FPGA seguem nas Figuras 32 e 33.

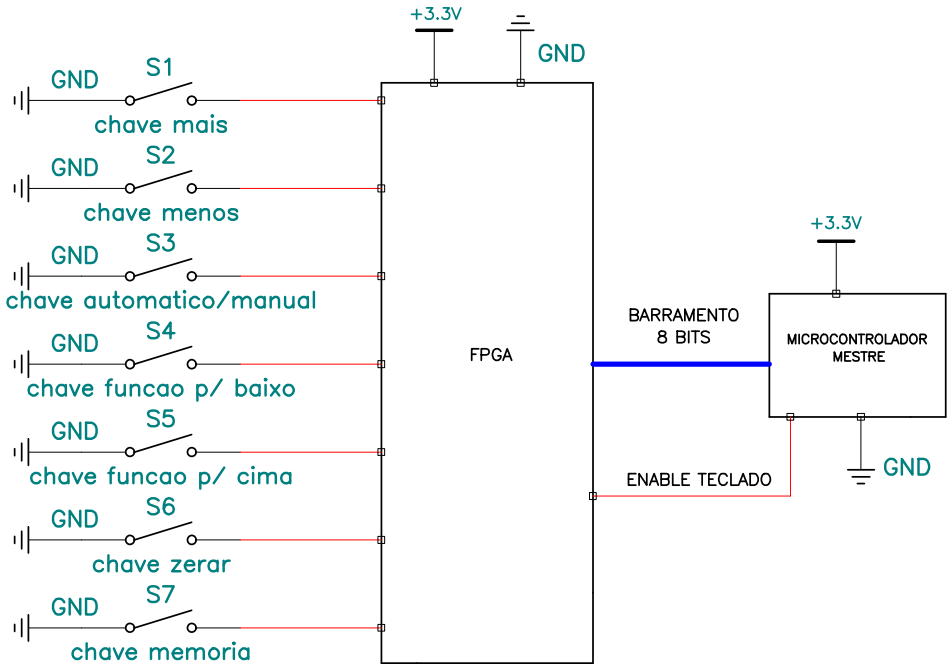

Figura 32 – Circuito multiplexador do teclado momentâneo com FPGA

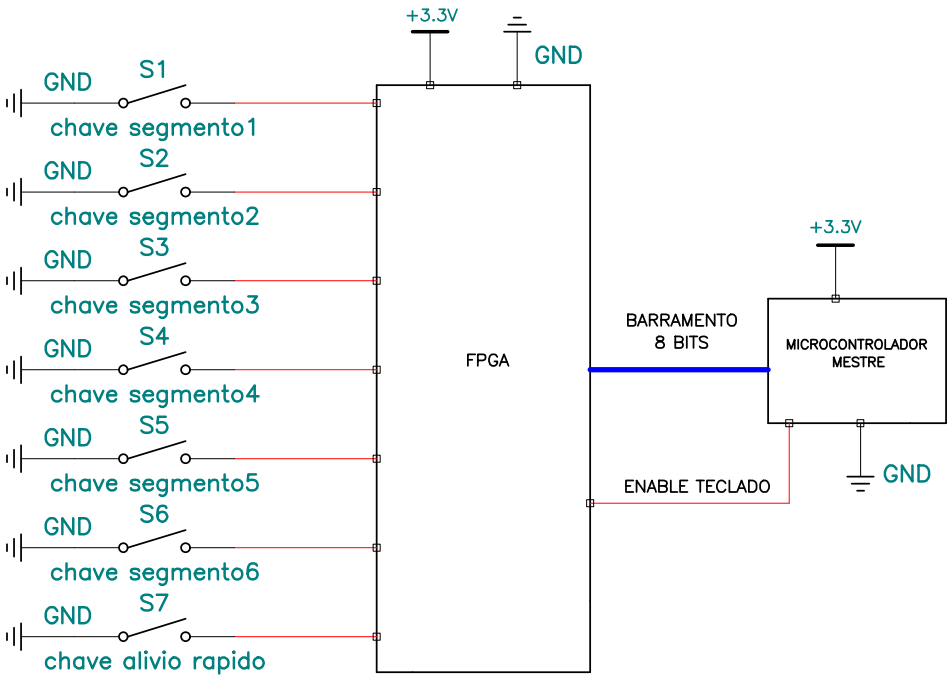

Figura 33 – Circuito multiplexador do teclado dos segmentos com FPGA

5.4.2. Descrição das Funções Nobres

As funções de captura de pulsos dos sensores de roda, vazão e nível mínimo do tanque de pulverização também foram sintetizadas e integradas no FPGA. A captura dos sinais de roda e vazão via FPGA é uma função nobre do controlador, uma vez que os valores são utilizados para o cálculo da dose real em L/ha. A captura atual, feita via microcontrolador, embora utilize interrupções, sofre com problemas de temporização como latência no atendimento das interrupções em virtude do atendimento de outras interrupções da comunicação serial, por exemplo. Isso acaba deixando a captura "quase" em tempo real. Como o FPGA tem poder de processamento superior ao de um microcontrolador, o tratamento dos pulsos seria praticamente em tempo real e os valores do período seriam entregues ao microcontrolador quando solicitado ao FPGA. A captura em tempo real melhora o desempenho do sistema de pulverização.

A captura envolve a leitura do período do sinal de vazão pelo FPGA, com número de pulsos coletados em função de um sinal enviado pelo microcontrolador mestre (mudança de escala para aumentar a precisão de leitura), enviando o valor do período via barramento paralelo. Atualmente, os valores dos períodos dos sinais de roda e vazão são lidos em variáveis de 16 e 24 bits. A resolução de leitura para a roda é de 1 milisegundo (ms) e para a vazão, 1 microsegundo (μs). Por problemas de pinagem limitada do microcontrolador, seria interessante utilizar o mesmo barramento de leitura dos dados dos multiplexadores (8 bits), ou seja, o microcontrolador escolhe quando o FPGA entrega os valores de teclado momentâneo, de teclado dos segmentos, de período de roda 1, de período de roda 2 ou período de vazão, sendo os três últimos entregues em pacotes de 8 bits, sincronizados por pinos dedicados.

A faixa de freqüência para os sinais dos sensores de roda é aproximadamente 1 a 50 Hz e para o sinal do sensor de vazão é 5 a 1,5 kHz. Atualmente, o software faz a leitura do período para 4 pulsos de roda (sempre) e 32 ou 64 pulsos de vazão (dependendo da vazão em L/min).

Para o sensor de nível mínimo do tanque de pulverização, não há trem de pulsos e o FPGA somente deve detectar o nível lógico 0 ou 1, sinalizando para o microcontrolador via pino dedicado.

A configuração proposta utilizando FPGA segue na Figura 34.

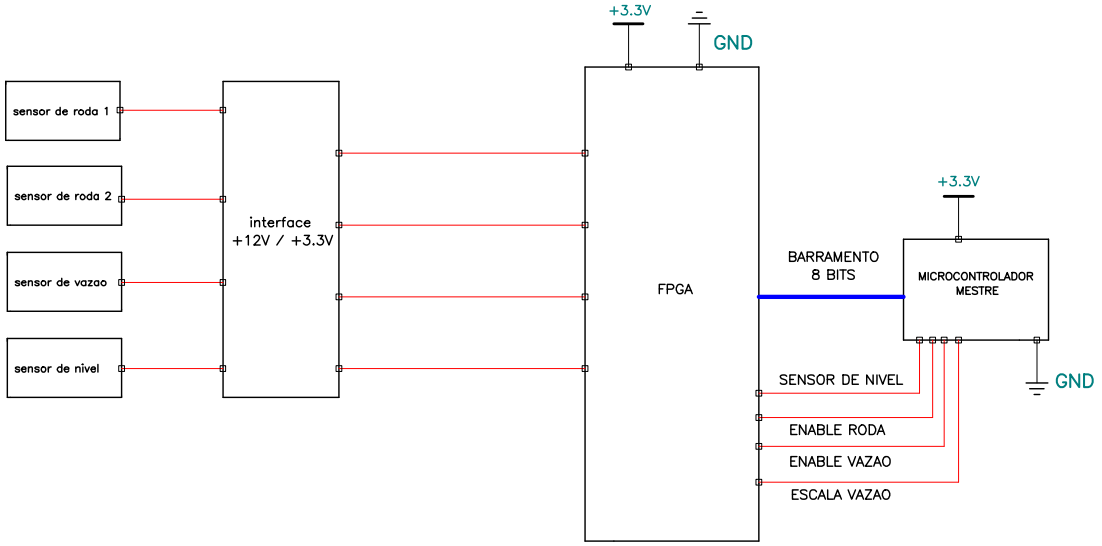

Figura 34 – Circuito para captura dos sinais dos sensores com FPGA

A configuração final englobando todas as funcionalidades segue na Figura 35.

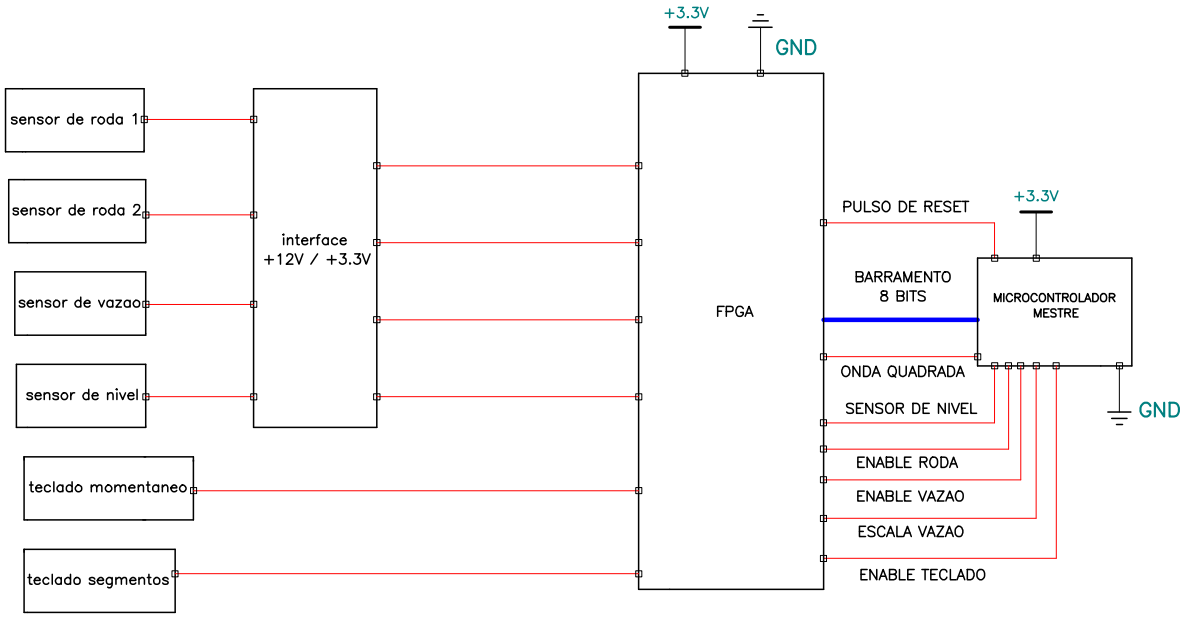

Figura 35 – Circuito completo para o estudo de caso

6. IMPLEMENTAÇÃO DAS FUNÇÕES EM FPGA

Para a implementação das funções em FPGA, foi desenvolvida a seqüência de trabalho com os itens que seguem abaixo relacionados.

1) Definição do FPGA utilizado.

2) Escolha das ferramentas de desenvolvimento para elaboração dos circuitos lógicos.

3) Projeto dos circuitos lógicos relacionados com cada função, o que engloba as etapas de síntese, projeto lógico, implementação propriamente dita, simulação e validação.

3) Integração de todas a funções em um único FPGA, gerando o circuito geral.

4) Testes de bancada do circuito geral.

Conforme definido no estudo de caso e ilustrado nas figuras 35 e 40, as funções que foram implementadas no FPGA seguem abaixo relacionadas.

1) Funções periféricas:

a) *Watchdog* do microcontrolador: função que tem como objetivo monitorar o funcionamento do microcontrolador, atuando em seu pino de *reset* caso ocorra, por motivo de ruído ou interferência, uma entrada em estado de *deadlock.*.

b) Leitura do teclado momentâneo e do teclado de segmentos da barra de pulverização: função que tem como objetivo realizar a leitura dos sinais das chaves que formam o teclado momentâneo e o de segmentos, entregando os valores para o microcontrolador. As chaves e suas funções vêm a seguir.

b1) Chaves momentâneas:

- Chave automático/manual = escolher modo de operação do sistema de pulverização
- Chave mais = ajuste de valores
- Chave menos = ajuste de valores
- Chave zerar = zerar valores
- Chave memória = gravar valores
- Chave função (acima) = acessar menus de usuário

- Chave função (abaixo) = acessar menus de usuário

b2) Chaves de segmento:

- Chave segmento 1 = ligar/desligar pulverização no segmento 1 da barra de pulverização
- Chave segmento 2 = ligar/desligar pulverização no segmento 2 da barra de pulverização
- Chave segmento 3 = ligar/desligar pulverização no segmento 3 da barra de pulverização
- Chave segmento 4 = ligar/desligar pulverização no segmento 4 da barra de pulverização
- Chave alívio rápido = ligar/desligar pulverização em todos os segmentos da barra de pulverização

2) Funções nobres:

a) Captura dos pulsos dos sensores de roda: função que tem como objetivo fazer a leitura dos períodos dos pulsos dos sensores de roda, entregando os valores ao microcontrolador para cálculo da velocidade de deslocamento da máquina. São utilizados dois sensores indutivos para geração dos pulsos.

b) Captura dos pulsos do sensor de vazão: função que tem como objetivo fazer a leitura do período dos pulsos do sensor de vazão, entregando o valor ao microcontrolador para cálculo da vazão de líquido na barra de pulverização. É utilizado um sensor indutivo para a geração dos pulsos.

c) Captura do pulso do sensor de nível mínimo do tanque de pulverização: função que tem como objetivo fazer a leitura do pulso do sensor de nível mínimo, entregando o valor ao microcontrolador para geração de sinal de alerta ao operador da máquina. É utilizado um sensor indutivo para a geração do pulso.

A seguir são apresentados os detalhes da seqüência de trabalho apresentada.

6.1. Definição do FPGA utilizado

Levando-se em conta a disponibilidade de material oferecido pelo LAS (Laboratório de Arquitetura e Sistemas), foi escolhido o FPGA XC4010XLPC84 integrante da família XC4000XL do fabricante Xilinx, para a implementação do circuito completo. Trata-se de um FPGA com encapsulamento PLCC (*Plastic Lead Chip Carrier*) de 84 pinos, sendo 61 deles disponíveis para entrada e saída.

Para auxiliar o teste funcional de validação, foi utilizada a placa de desenvolvimento XS40 versão 1.2 do fabricante Xess Corporation (Figura 36). A placa contém basicamente uma FPGA XC4010XLPC84 funcionando a 3.3 V, um microcontrolador 80C31, uma memória RAM estática de 32 Kbytes, um oscilador de 12 MHz, um display de 7 segmentos e um conector VGA para leitura de sinais de vídeo. A programação do FPGA é feita via PC através da porta paralela.

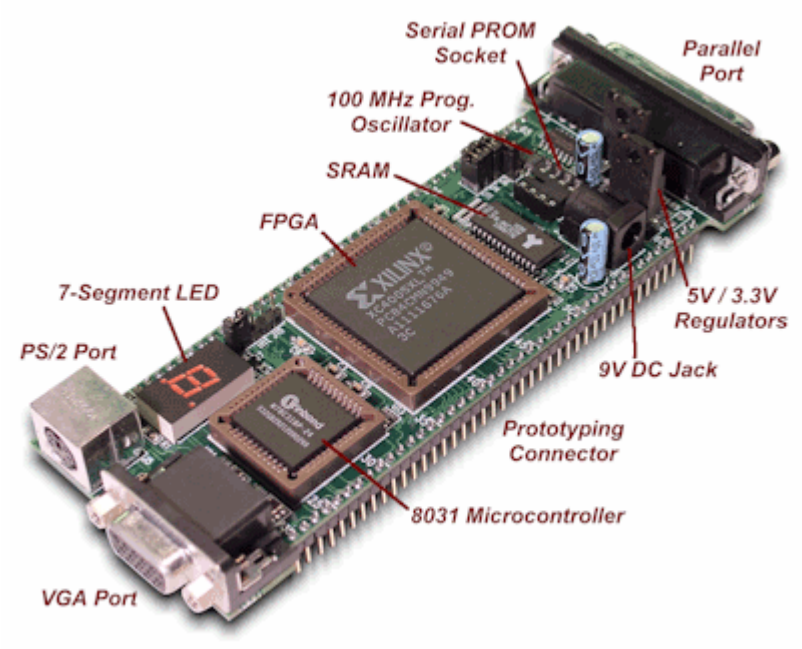

Figura 36 – Placa de desenvolvimento XS40

6.2. Escolha de Ferramentas de Desenvolvimento para Elaboração dos Circuitos Lógicos

Nas atividades de descrição, simulação e implementação dos diagramas esquemáticos que formam o circuito geral, foi utilizado a ferramenta de desenvolvimento *Project Manager* (Figura 37) do ambiente *Xilinx Foundation F3.1i*. A partir dos diagramas esquemáticos implementados, a ferramenta faz a verificação da sintaxe do projeto e fornece um simulador bastante completo para testar e validar funcionalmente o subsistema eletrônico proposto.

O *Project Manager* também é capaz de gerar os bits de configuração para diversos dispositivos reconfiguráveis da Xilinx. Para isso, a ferramenta faz o mapeamento da solução no dispositivo, e gera os bits de configuração (*bitstream*) que podem ser enviados para o dispositivo reconfigurável.

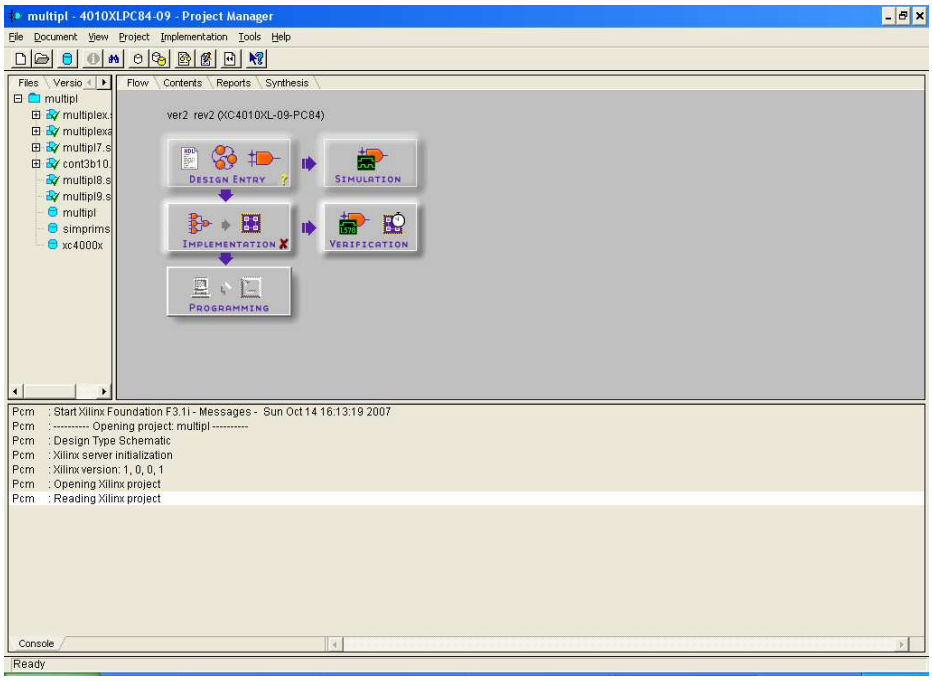

Figura 37 – Ambiente *Project Manager* da Xilinx

Com relação aos passos de desenvolvimento, a partir da escolha do modelo de FPGA adequado para o projeto, o circuito lógico foi construído utilizando-se o editor gráfico *Schematic Editor*, onde é possível selecionar diversos componentes a partir das bibliotecas, fazer as interligações entre eles, bem como agrupá-los em blocos criando macros para

reutilização de funções. Após a construção do circuito lógico, foi criado e exportado o *netlist* utilizado para compilar o circuito para o FPGA escolhido. A geração do *netlist* fornece informações importantes para o desenvolvimento, como avisos e mensagens de erro. Após a geração e exportação do *netlist*, foram simuladas as diversas funcionalidades do circuito utilizando-se a ferramenta *Logic Simulator*, onde podem ser observadas as diversas formas de onda para os sinais de entrada e saída escolhidos para verificação.

Após a etapa de simulação, o *netlist* exportado foi utilizado como base para as fases do processo de implementação que é realizado pela ferramenta *Flow Engine*, que faz a compilação e gera o *bitstream* para o FPGA. O *bitstream* gerado fica armazenado num arquivo com extensão *.bit*. As fases do processo de implementação são tradução, mapeamento, roteamento, temporização e configuração, sendo que para cada uma é gerado um relatório com dados estatísticos, avisos e mensagens de erro, caso ocorram.

O *download* do arquivo *.bit* no FPGA da placa de desenvolvimento foi realizado utilizando-se a ferramenta *GXSLOAD* (Figura 38), também disponível no LAS (Laboratório de Arquitetura e Sistemas). O arquivo carregado na ferramenta é transferido para o FPGA via porta paralela.

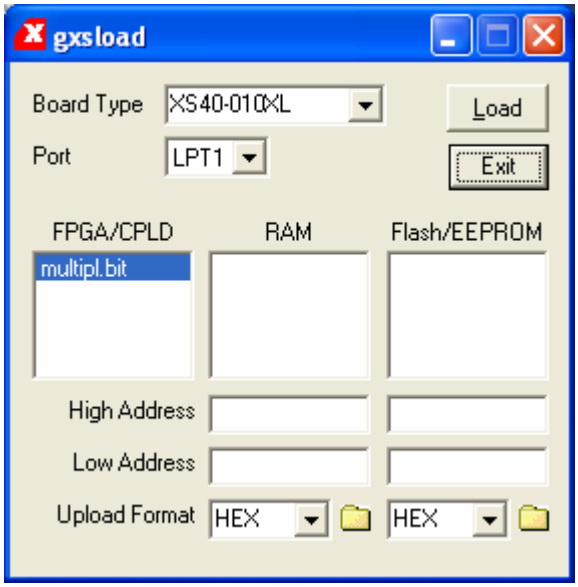

Figura 38 – Ferramenta *GXSLOAD*

O ANEXO G apresenta um guia rápido descrevendo como utilizar as ferramentas descritas.

A Figura 39 ilustra os passos de desenvolvimento descritos.

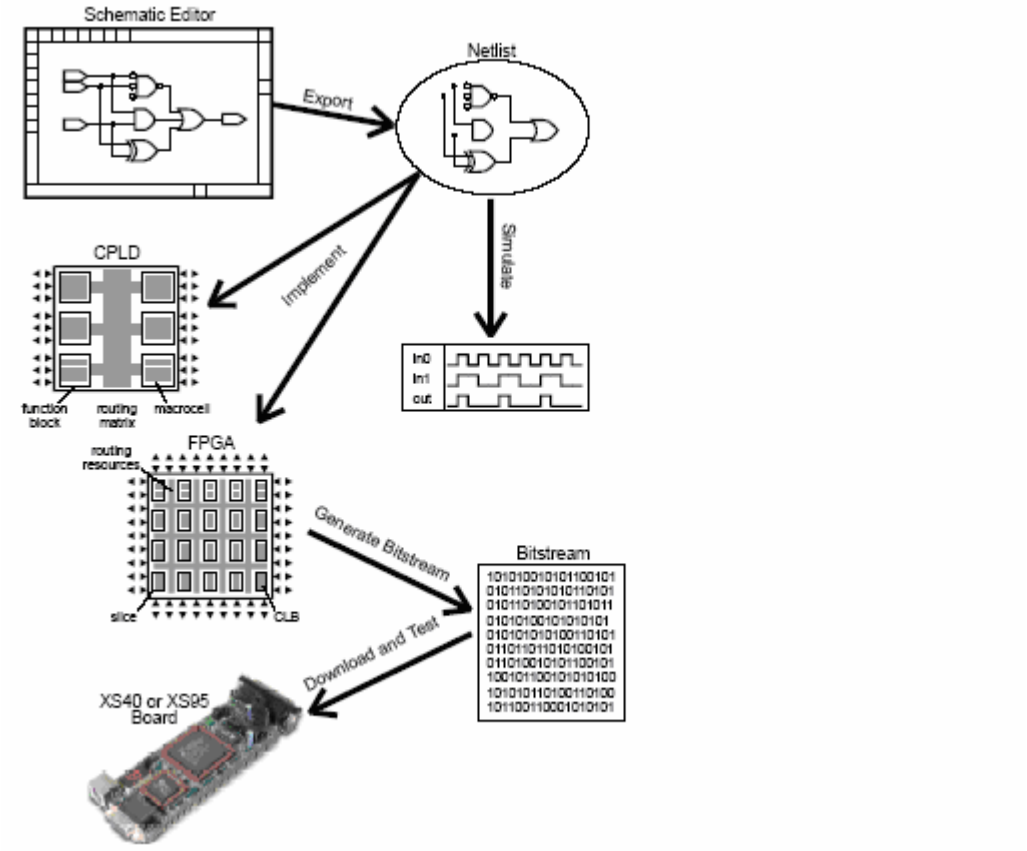

Figura 39 – Passos de desenvolvimento para projetos baseados em FPGA

6.3. Projeto dos Circuitos Lógicos

A seguir são apresentados dados relativos ao projeto e descrição do funcionamento dos circuitos lógicos implementados em FPGA.

6.3.1. Composição do Circuito Geral

De maneira global, a composição do circuito geral para o estudo de caso é formada pelos seguintes blocos:

1) Blocos de captura do período dos sinais dos sensores de roda 1 e 2.

2) Bloco de captura do período do sinal do sensor de vazão.

3) Bloco de leitura dos sinais das chaves momentâneas e chaves dos segmentos da barra de pulverização.

4) Bloco de leitura do sinal do sensor de nível mínimo do tanque de pulverização.

5) Bloco de tratamento do *watchdog* do microcontrolador.

6) Bloco de seleção da informação a ser lida pelo microcontrolador.

O diagrama de blocos do sistema implementado segue na Figura 40.

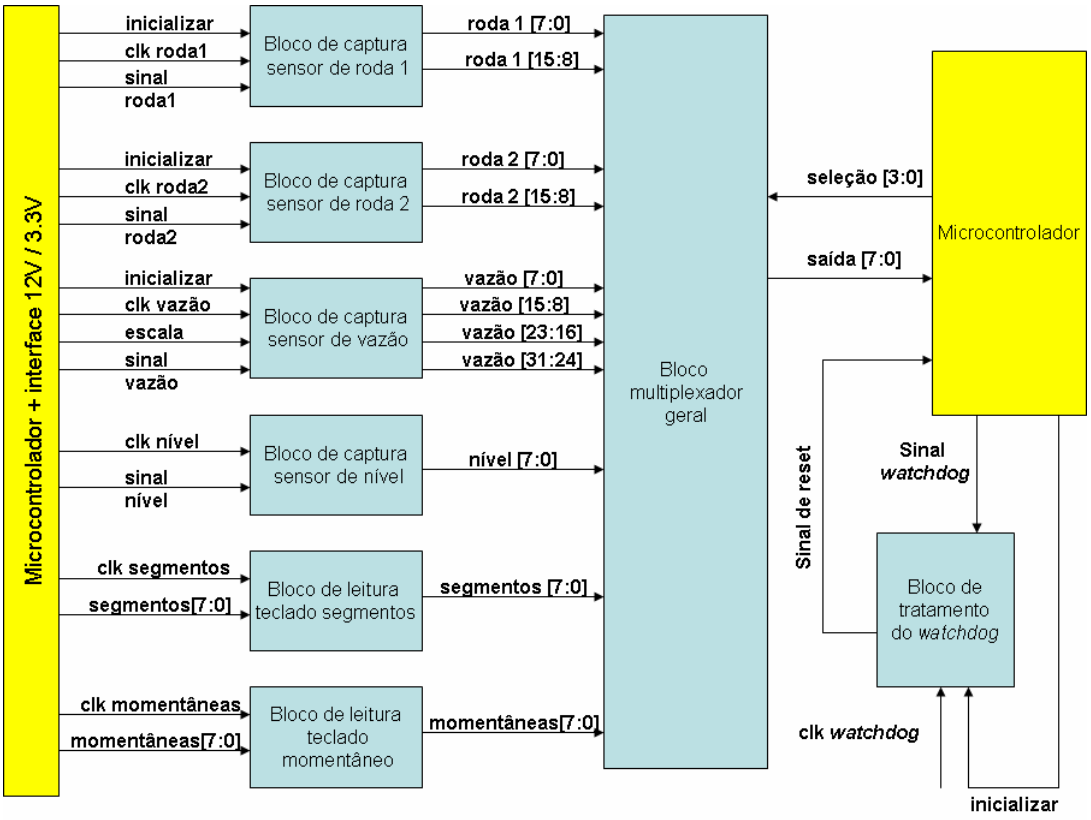

Figura 40 – Diagrama de blocos do sistema

A relação dos sinais de entrada e saída dos blocos segue abaixo descrita.

1) INICIALIZAR: sinal ativo em nível lógico "1" para inicialização dos contadores e registradores internos dos blocos. O sinal é gerado pelo microcontrolador.

2) CLK RODA1: refere-se ao sinal de *clock* que gera a resolução de leitura do período dos pulsos dos sensores de roda. Este sinal possui freqüência de 1 kHz, o que gera resolução de leitura de 1ms. O sinal é gerado pelo microcontrolador.

3) SINAL RODA1: refere-se ao sinal enviado pelo sensor de roda 1 com nível de tensão já ajustado pela interface +12V/+3V3. Consiste numa onda quadrada com 3,3V de amplitude e freqüência variando de 1 a 50 Hz aproximadamente, de acordo com a velocidade de deslocamento da máquina.

4) CLK RODA2: refere-se ao sinal de *clock* que gera a resolução de leitura do período dos pulsos dos sensores de roda. Este sinal possui freqüência de 1 kHz, o que gera resolução de leitura de 1ms. O sinal é gerado pelo microcontrolador.

5) SINAL RODA2: refere-se ao sinal enviado pelo sensor de roda 2 com nível de tensão já ajustado pela interface +12V/+3V3. Consiste numa onda quadrada com 3,3V de amplitude e freqüência variando de 1 a 50 Hz aproximadamente, de acordo com a velocidade de deslocamento da máquina.

6) CLK VAZAO: refere-se ao sinal de *clock* que gera a resolução de leitura do período dos pulsos do sensor de vazão. Este sinal possui freqüência de 1 MHz, o que gera resolução de leitura de 1us. O sinal é obtido internamente no FPGA a partir da divisão de freqüência de um sinal de *clock* externo de 8 MHz gerado pelo microcontrolador.

7) ESCALA: refere-se ao sinal que indica o número de pulsos do sinal do sensor de vazão a serem coletados pelo circuito de captura. A Tabela 1 é a tabela verdade para o sinal.

Nível lógico do sinal	Número de pulsos coletados
0	32
1	64

Tabela 1 – Tabela verdade para sinal de escala de pulsos do sensor de vazão

8) SINAL VAZAO: refere-se ao sinal enviado pelo sensor de vazão com nível de tensão já ajustado pela interface +12V/+3V3. Consiste numa onda quadrada com 3,3V de amplitude e freqüência variando de 5 Hz a 1,5 kHz aproximadamente, de acordo com a vazão de líquido nas barras de pulverização.

9) CLK_NIVEL: refere-se ao sinal de *clock* para sincronização do circuito de leitura do sinal do sensor de nível. O sinal é obtido internamente no FPGA a partir da divisão de freqüência de um sinal de *clock* externo de 8 MHz gerado pelo microcontrolador.

10) SINAL NIVEL: refere-se ao sinal do sensor de nível mínimo do tanque de pulverização. O nível lógico “1” indica o nível mínimo.

11) CLK SEGMENTOS: refere-se ao sinal de *clock* para sincronização do circuito de leitura do teclado de segmentos. O sinal é obtido internamente no FPGA a partir da divisão de freqüência de um sinal de *clock* externo de 8 MHz gerado pelo microcontrolador.

12) SEGMENTOS[7..0]: refere-se aos sinais do teclado de segmentos. O nível lógico “0” na entrada indica chave do teclado acionada.

13) CLK MOMENTANEAS: refere-se ao sinal de *clock* para sincronização do circuito de leitura do teclado momentâneo. O sinal é obtido internamente no FPGA a partir da divisão de freqüência de um sinal de *clock* externo de 8 MHz gerado pelo microcontrolador.

14) MOMENTANEAS[7..0]: refere-se aos sinais do teclado momentâneo. O nível lógico “0” na entrada indica chave do teclado acionada.

15) RODA1[7..0]: refere-se ao byte menos significativo gerado pelo circuito de captura do sensor de roda 1.

16) RODA1[15..8]: refere-se ao byte mais significativo gerado pelo circuito de captura do sensor de roda 1.

17) RODA2[7..0]: refere-se ao byte menos significativo gerado pelo circuito de captura do sensor de roda 2.

18) RODA2[15..8]: refere-se ao byte mais significativo gerado pelo circuito de captura do sensor de roda 2.

19) VAZAO[7..0]: refere-se ao byte menos significativo gerado pelo circuito de captura do sensor de vazão.

20) VAZAO[15..8]: refere-se ao primeiro byte intermediário gerado pelo circuito de captura do sensor de vazão.

21) VAZAO[23..16]: refere-se ao segundo byte intermediário gerado pelo circuito de captura do sensor de vazão.

22) VAZAO[31..24]: refere-se ao byte mais significativo gerado pelo circuito de captura do sensor de vazão.

23) NIVEL[7..0]: refere-se ao byte gerado pelo circuito de captura do sensor de nível, sendo de interesse somente o bit menos significativo.

24) SELECAO[3..0]: refere-se aos sinais de seleção controlados pelo microcontrolador para escolha do tipo de dado a ser lido no duto de dados de saída. A Tabela 2 é a tabela verdade para os sinais de seleção.

Saída	S3	S2	S1	S0
RODA1[7..0]	0	0	0	0
RODA2[7..0]	0	0	0	1
RODA1[15..8]	0	0	1	0
RODA2[15..8]	0	0	1	1
VAZÃO[7..0]	0	1	0	0
VAZÃO[15..8]	0	1	0	1
VAZÃO[23..16]	0	1	1	0
VAZÃO[31..24]	0	1	1	1
MOMENTÂNEAS[7..0]	1	0	0	0
SEGMENTOS[7..0]	1	0	0	1
NÍVEL[7..0]	1	0	0	0

Tabela 2 – Tabela verdade para os sinais de seleção

25) SAIDA[7..0]: refere-se ao duto de dados de saída lido pelo microcontrolador.

26) CLK WATCHDOG: refere-se ao sinal de *clock* para sincronização do circuito de *watchdog* do microcontrolador. O sinal é obtido internamente no FPGA a partir da utilização de um oscilador interno.

27) SINAL WATCHDOG: refere-se ao sinal gerado pelo microcontrolador para sinalizar seu funcionamento ao circuito de *watchdog*.

28) SINAL DE RESET: refere-se ao sinal gerado pelo FPGA para gerar pulsos no pino de *reset* do microcontrolador a cada 1 segundo aproximadamente.

6.3.2. Blocos de Captura dos Períodos dos Sinais dos Sensores de Roda

Os blocos de captura dos períodos dos sinais dos dois sensores de roda têm como função fazer a leitura dos períodos dos pulsos dos sensores de roda, entregando os valores ao microcontrolador para cálculo da velocidade de deslocamento da máquina.

Conforme afirmado no item 5.4.2, a faixa de freqüência para os pulsos do sensor de roda é de 1 a 50 Hz, variando de acordo com a velocidade de deslocamento da máquina. Considerando a relação de 3,5 pulsos por metro e o número de pulsos coletados igual a quatro, temos:

1) Para 1 Hz => período = 1 s x 4 pulsos = 4 s

De acordo com o item 5.4.2, considerando resolução de varredura = 1 ms = 4000 contagens = 0FA0H = 2 bytes

2) Para 50 Hz => período = 20 ms x 4 pulsos = 80 ms

De acordo com o item 5.4.2, considerando resolução de varredura = 1 ms = 80 contagens = 50H = 1 byte

Como no pior caso o resultado da leitura do período utiliza 2 bytes para armazenamento, pode-se afirmar então que um contador de 16 bits atende a faixa de freqüência para a velocidade de deslocamento.

De acordo com a Figura 40, cada bloco de captura recebe em sua entrada os sinais de inicialização, *clock* de varredura e pulsos do sensor de roda correspondente e disponibiliza na saída o resultado da leitura do período. O diagrama lógico do bloco segue na Figura 41.

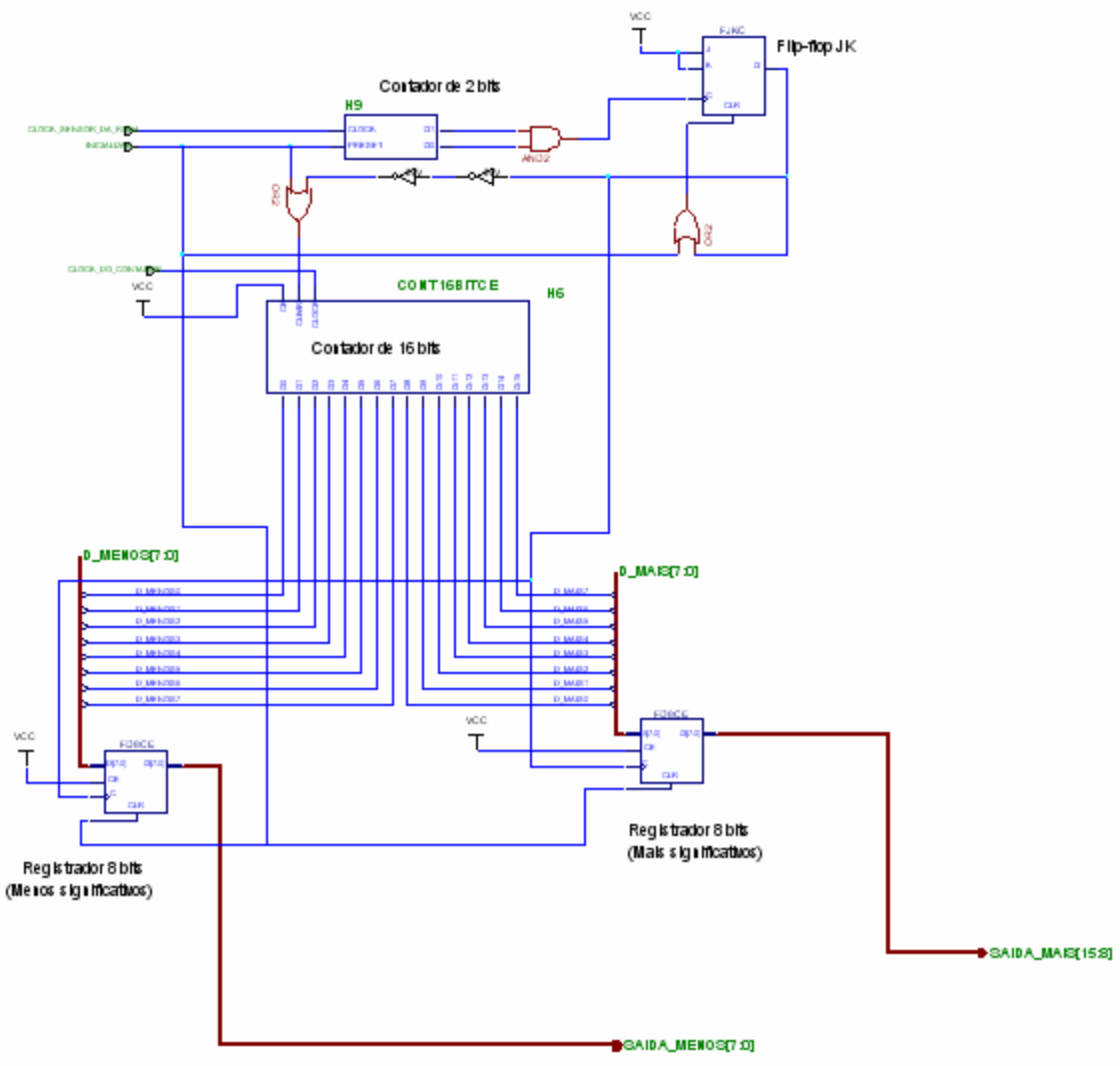

Figura 41 – Diagrama lógico do bloco de captura do sensor de roda

Os blocos de captura do período dos sinais dos sensores de roda possuem funcionamento idêntico. Em cada bloco existe um contador de 2 bits que recebe como *clock* os pulsos do sensor de roda correspondente. Um contador de 16 bits com *clock* de 1 kHz gerado pelo microcontrolador, faz a contagem do período até que sejam contados 4 pulsos do sensor pelo contador de 2 bits. A contagem dos 4 pulsos dispara um flip-flop JK cuja saída sincroniza o carregamento dos 16 bits resultantes da contagem do período em dois registradores de 8 bits bem como faz o *reset* dos contadores para o próximo ciclo de contagem. As arquiteturas internas dos registradores de 8 bits e dos contadores de 2 e 16 bits seguem no ANEXO F. Os resultados da simulação do bloco de captura do sensor de roda seguem no ANEXO A.

6.3.3. Bloco de Captura do Período do Sinal do Sensor de Vazão

O bloco de captura do período do sinal do sensor de vazão tem como função fazer a leitura do período dos pulsos do sensor de vazão, entregando o valor ao microcontrolador para cálculo da vazão de líquido na barra de pulverização.

Conforme descrição contida no item 5.4.2, a faixa de freqüência para os pulsos do sensor de vazão é de 50 Hz a 1,5 kHz, variando de acordo com a vazão desejada na barra de pulverização. Considerando a relação de 600 pulsos por litro e que acima de 73,6 L/min coleta 64 pulsos e abaixo coleta 32 pulsos, temos:

1) Para 50 Hz = 5.0 L/min => período = 20 ms x 32 pulsos = 640 ms

De acordo com o item 5.4.2, considerando resolução de varredura = 1 microseg = 640000 inc = 9C400H = 3 bytes

2) Para 1,5 KHz = 150,0 L/min => período = 666,67 microseg x 64 pulsos = 42,67 mseg

De acordo com o item 5.4.2, considerando resolução de varredura = 1 microseg = 42670 inc = A6AEH = 2 bytes

Como no pior caso o resultado da leitura do período utiliza 3 bytes para armazenamento, pode-se afirmar então que um contador de 24 bits atende a faixa de freqüência para a vazão.

De acordo com a Figura 40, o bloco de captura recebe em sua entrada os sinais de inicialização, *clock* de varredura, pulsos do sensor de vazão e escala, disponibilizando na

saída o resultado da leitura do período. O diagrama lógico do bloco vem a seguir na Figura 42.

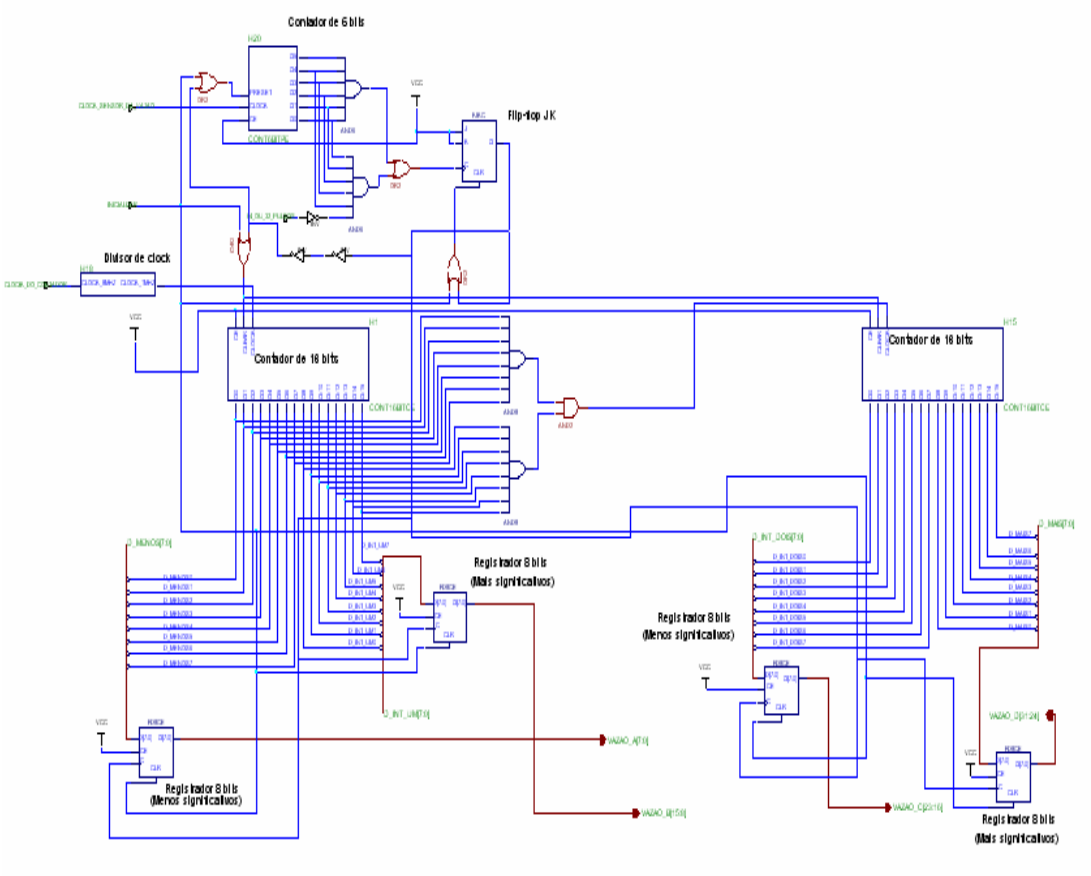

Figura 42 – Diagrama lógico do bloco de captura do sensor de vazão

No bloco de captura de vazão existe um contador de 6 bits que recebe como *clock* os pulsos do sensor de vazão. Um contador de 16 bits com *clock* de 1 MHz obtido pelo divisor de *clock* a partir do *clock* de 8 MHz gerado pelo microcontrolador, faz a contagem do período até que sejam contados, dependendo do sinal de escala, 32 ou 64 pulsos do sensor pelo contador de 6 bits. Dependendo da freqüência do sinal do sensor de vazão, o *overflow* do contador de 16 bits dispara a contagem de outro contador de 16 bits, o que permite capturar valores de período de até 32 bits, embora 24 bits sejam suficientes, conforme afirmado anteriormente. A contagem dos pulsos dispara um flip-flop JK cuja saída sincroniza o carregamento dos 32 bits resultantes da contagem do período em quatro registradores de 8 bits bem como faz o *reset* dos contadores para o próximo ciclo de contagem. As arquiteturas internas do divisor de *clock*, dos contadores de 6 e 16 bits e dos registradores de 8 bits seguem no ANEXO F. Os resultados da simulação do bloco de captura do sensor de vazão seguem no ANEXO B.

6.3.4. Blocos de Leitura das Chaves Momentâneas e Chaves dos Segmentos da Barra de Pulverização

Os blocos de leitura dos sinais das chaves têm como função realizar a leitura dos sinais das chaves que formam o teclado momentâneo e o de segmentos da barra de pulverização, entregando os valores para o microcontrolador.

De acordo com a Figura 40, cada bloco recebe em sua entrada os sinais de teclado e *clock*, disponibilizando na saída o resultado da leitura das chaves. O diagrama lógico dos blocos segue na Figura 43.

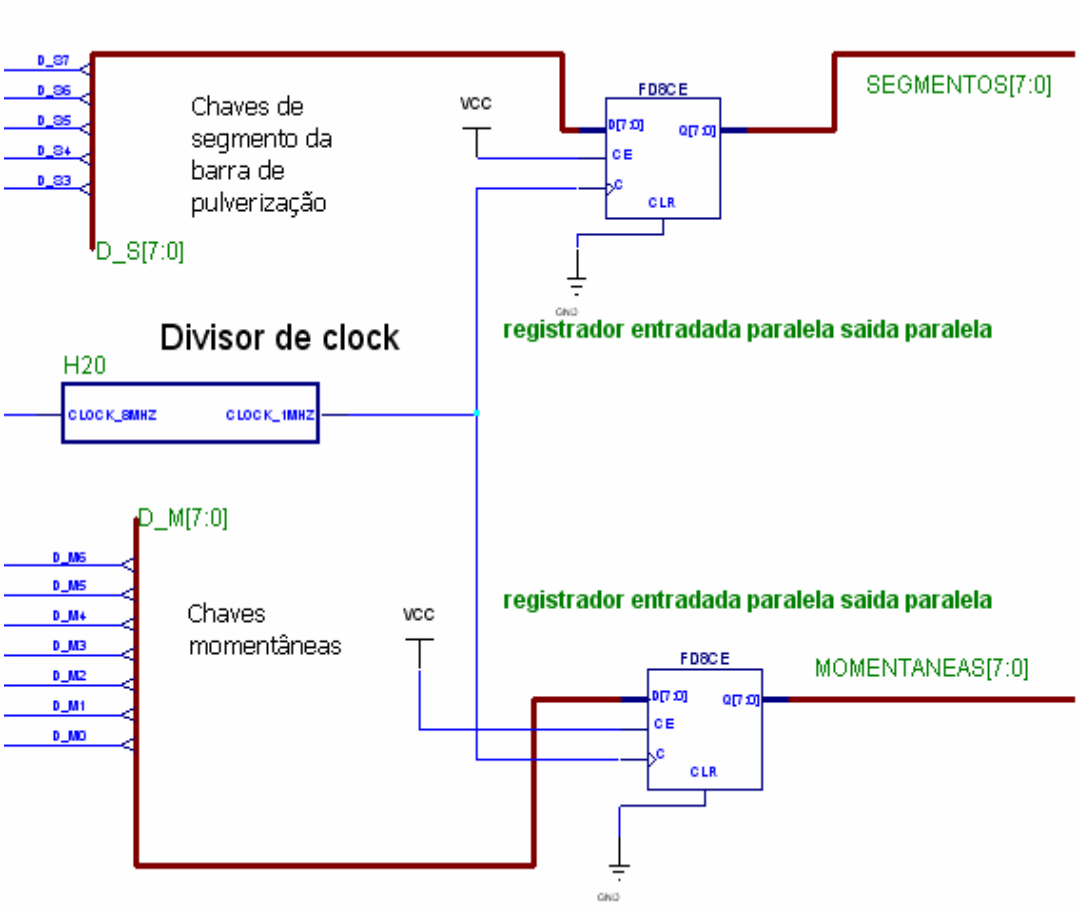

Figura 43 – Diagrama lógico dos blocos de leitura de teclado

Cada bloco de leitura dos sinais das chaves é constituído por um registrador de 8 bits com entrada paralela e saída paralela com *clock* de 1 MHz obtido pelo divisor de *clock* a partir do *clock* de 8 MHz gerado pelo microcontrolador.

O circuito transfere os sinais provenientes dos teclados, momentâneo e de segmentos, para as saídas dos registradores. As arquiteturas internas do divisor de *clock* e dos registradores de 8 bits seguem no ANEXO F. Os resultados das simulações dos blocos de leitura do teclado seguem no ANEXO C.

6.3.5. Bloco de Captura do Sensor de Nível Mínimo do Tanque de Pulverização

O bloco de captura do pulso do sensor de nível mínimo do tanque de pulverização tem como função fazer a leitura do pulso do sensor de nível mínimo, entregando o valor ao microcontrolador para geração de sinal de alerta ao operador da máquina.

De acordo com a Figura 40, o bloco de captura recebe em sua entrada o pulso do sensor de nível e *clock*, disponibilizando na saída o resultado da leitura. O diagrama lógico do bloco segue na Figura 44.

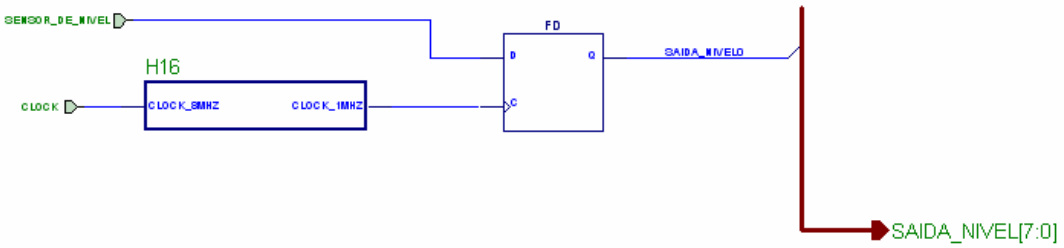

Figura 44 – Diagrama lógico do bloco de leitura do sensor de nível

O bloco de leitura do sinal do sensor de nível mínimo do tanque de pulverização consiste basicamente num flip-flop tipo D com *clock* obtido internamente no FPGA a partir da divisão de freqüência de um sinal de *clock* externo de 8 MHz gerado pelo microcontrolador. O flip-flop transfere o sinal do sensor de nível para a saída. A arquitetura interna do divisor de *clock* segue no ANEXO F. Os resultados da simulação do bloco de leitura do sensor de nível seguem no ANEXO D.

6.3.6. Bloco de Tratamento do *Watchdog* do Microcontrolador

O bloco de tratamento do *watchdog* do microcontrolador tem como função monitorar o funcionamento do microcontrolador, atuando em seu pino de *reset* caso ocorra, por motivo de ruído ou interferência, uma entrada em estado de *deadlock*.

De acordo com a Figura 40, o bloco de *watchdog* recebe em sua entrada os sinais de inicialização, onda *watchdog* e *clock*, disponibilizando na saída o sinal para atuar no pino de *reset* do microcontrolador. O diagrama lógico do bloco vem a seguir na Figura 45.

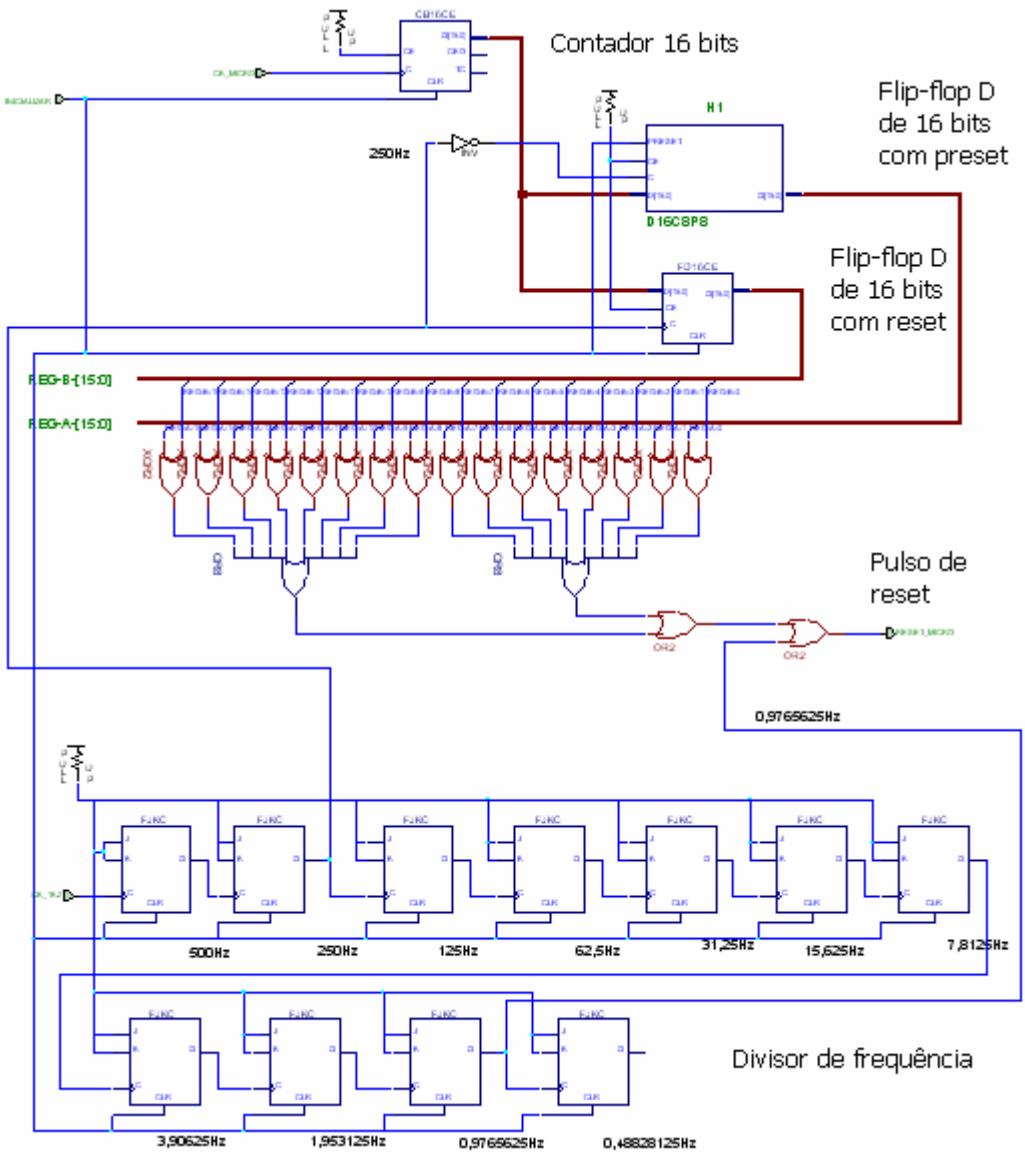

Figura 45 – Diagrama lógico do bloco de tratamento do *watchdog*

Basicamente, um contador de 16 bits tendo como sinal de *clock* a onda quadrada de aproximadamente 1 kHz gerada pelo microcontrolador para sinalizar seu funcionamento (onda *watchdog*), transfere o valor da contagem para dois flip-flops tipo D com *clock* de aproximadamente 250 Hz. Como os flip-flops trabalham com freqüência 25% mais baixa, os valores de contagem serão bem diferentes por intervalo. Além disso, os flip-flops possuem sincronização oposta, um na borda de descida do *clock* e o outro na borda de subida, o que garante a transferência de dois valores diferentes para o circuito comparador baseado em portas XOR. Enquanto existir a onda *watchdog*, o circuito comparador sempre detecta valores diferentes em suas entradas, o que inibe o pulso de reset a cada 1 segundo gerado pelo divisor de freqüência que recebe como *clock* o sinal do oscilador interno do FPGA, justamente para garantir independência em relação ao *clock* do microcontrolador. Quando o microcontrolador deixar de gerar a onda *watchdog*, o contador de 16 bits cessa a contagem e valores iguais serão detectados pelo circuito comparador, que libera então o pulso de *reset*. Para garantir ainda que o próprio bloco de tratamento de *watchdog* não gere o *deadlock* do microcontrolador durante a inicialização do sistema, os flip-flops são inicializados com valores diferentes. A arquitetura interna do flip-flop tipo D com *preset* segue no ANEXO F. Os resultados da simulação do bloco de tratamento do watchdog seguem no ANEXO E.

6.3.7. Bloco Multiplexador Geral

O bloco multiplexador geral tem como função receber os dados fornecidos pelos blocos de captura dos sensores e de leitura do teclado, disponibilizando via barramento de 8 bits, o dado escolhido pelo microcontrolador através dos sinais de seleção da informação.

De acordo com a Figura 40, o bloco multiplexador geral recebe em suas entradas os 16 bits resultantes do bloco de captura dos pulsos do sensor de roda 1, os 16 bits resultantes do bloco de captura dos pulsos do sensor de roda 2, os 32 bits resultantes do bloco de captura dos pulsos do sensor de vazão, os 16 bits resultantes dos blocos de leitura de teclado e os 8 bits resultantes do bloco de captura do sensor de nível. Além dos dados citados, outras entradas são os sinais de seleção gerados pelo microcontrolador para leitura de informação específica (vide Tabela 2). A saída do bloco multiplexador geral é um barramento de 8 bits que fornece a informação desejada ao microcontrolador. O diagrama lógico do bloco vem a seguir na Figura 46.

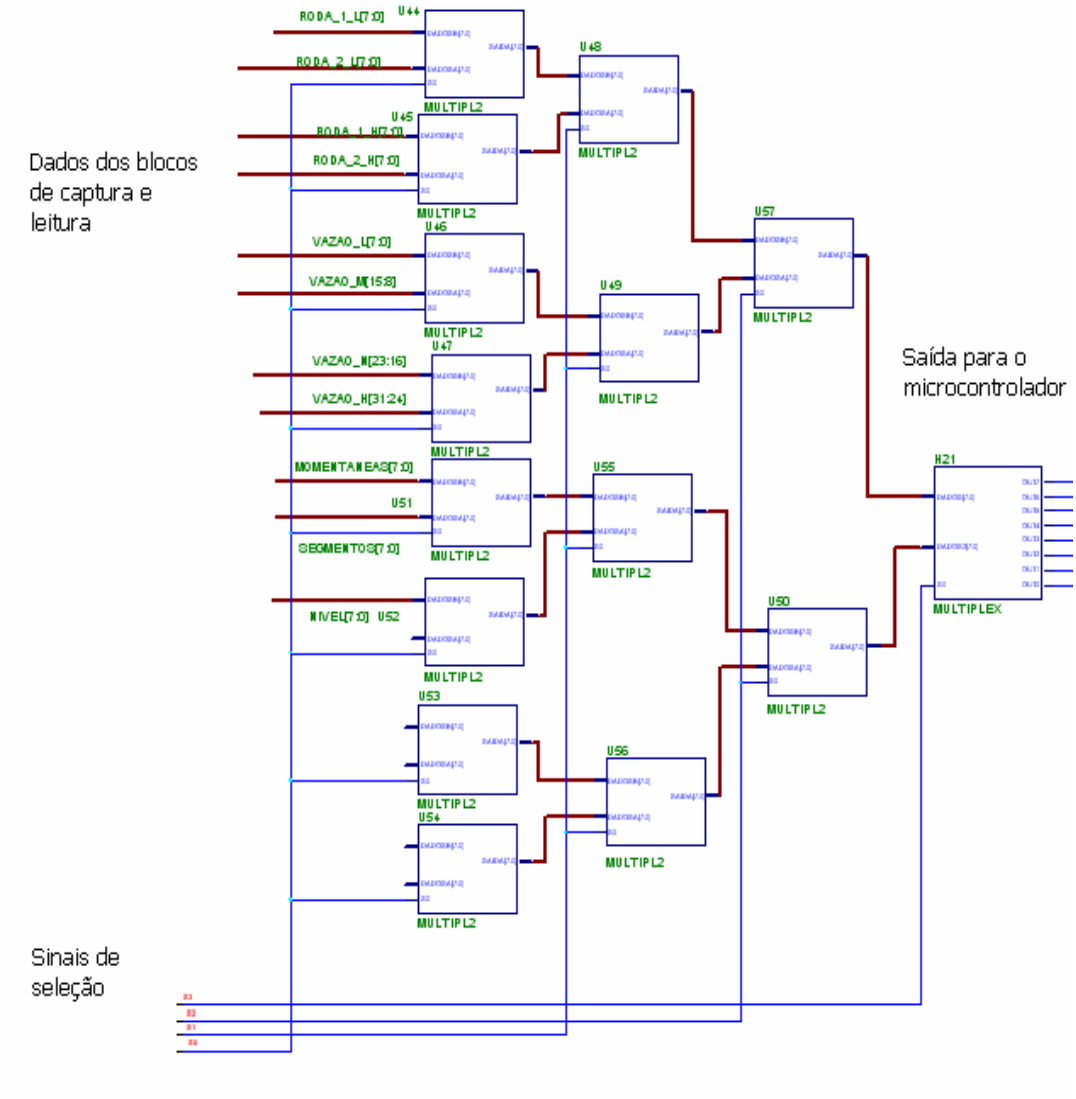

Figura 46 – Diagrama lógico do bloco multiplexador geral

A unidade básica do bloco multiplexador geral é um multiplexador de dois canais de 8 bits (1 bit de seleção), mostrado na Figura 47.

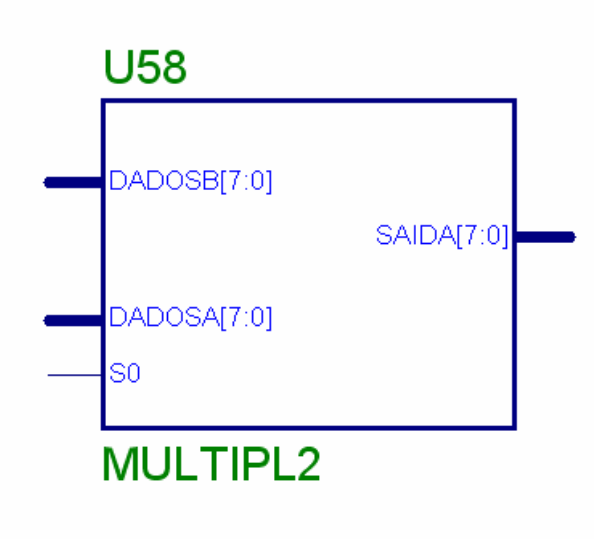

Figura 47 – Multiplexador de dois canais de 8 bits

O bit de seleção S0 com nível lógico “0” coloca na saída os dados do canal superior. O nível lógico “1” coloca na saída os dados do canal inferior. A arquitetura interna do multiplexador segue no ANEXO F.

O bloco multiplexador geral é constituído por quatro estágios de multiplexadores estruturados da seguinte forma:

1) Primeiro estágio (8 multiplexadores): cada multiplexador recebe em seus canais os 16 bits resultantes da leitura dos sinais referentes aos sensores de roda, sensor de vazão, sensor de nível, teclado momentâneo e teclado de segmentos. O bit de seleção dos multiplexadores recebe o sinal menos significativo entre os sinais de seleção gerados pelo microcontrolador. Dois multiplexadores não recebem sinal em seus canais e serão utilizados para expansão.

2) Segundo estágio (4 multiplexadores): cada multiplexador recebe em seus canais os 16 bits resultantes do estágio de seleção anterior, sendo um deles responsável pelo chaveamento entre os dados de roda, outro pelos dados de vazão e outro pelos dados de nível e teclado. O bit de seleção dos multiplexadores recebe o primeiro sinal intermediário entre os sinais de seleção gerados pelo microcontrolador. O multiplexador restante será utilizado para expansão.

3) Terceiro estágio (2 multiplexadores): cada multiplexador recebe em seus canais os 16 bits resultantes do estágio de seleção anterior, sendo um deles responsável pelo chaveamento entre os dados de roda e vazão e outro pelos dados de nível e teclado. O bit de

seleção dos multiplexadores recebe o segundo sinal intermediário entre os sinais de seleção gerados pelo microcontrolador.

4) Quarto estágio (1 multiplexador): o multiplexador recebe em seus canais os 16 bits resultantes do estágio de seleção anterior. O bit de seleção do multiplexador recebe o sinal mais significativo entre os sinais de seleção gerados pelo microcontrolador. De acordo com a Tabela 2, a saída do terceiro estágio contém os 8 bits desejados para a leitura do microcontrolador. O resultado da simulação do bloco multiplexador geral pode ser observado nos anexos contando as simulações dos blocos dos circuitos individuais descritos anteriormente.

6.3.8. Integração dos Circuitos

A integração dos circuitos foi realizada a partir da criação de macros que permitiram o encapsulamento dos circuitos lógicos dos blocos individuais. Foi acrescentado ainda um bloco chamado "sinalizador de *clock*" para auxiliar durante os testes de bancada. O bloco simplesmente recebe o *clock* de 8 MHz gerado pelo microcontrolador e, a partir da divisão de freqüência, atua nos segmentos inferior e superior do display de 7 segmentos da placa de desenvolvimento XS40 (vide Figura 31), que piscam com temporizações diferentes para sinalizar visualmente que o FPGA está recebendo o sinal de *clock* do microcontrolador.

Foram escolhidos também os pinos do FPGA mais adequados para interfaceamento com os sinais dos sensores, do teclado e para comunicação com o microcontrolador.

A Figura 48 mostra, na forma de diagramas lógicos, a estrutura final do circuito geral gerado e contido na Figura 40 em diagrama de blocos.

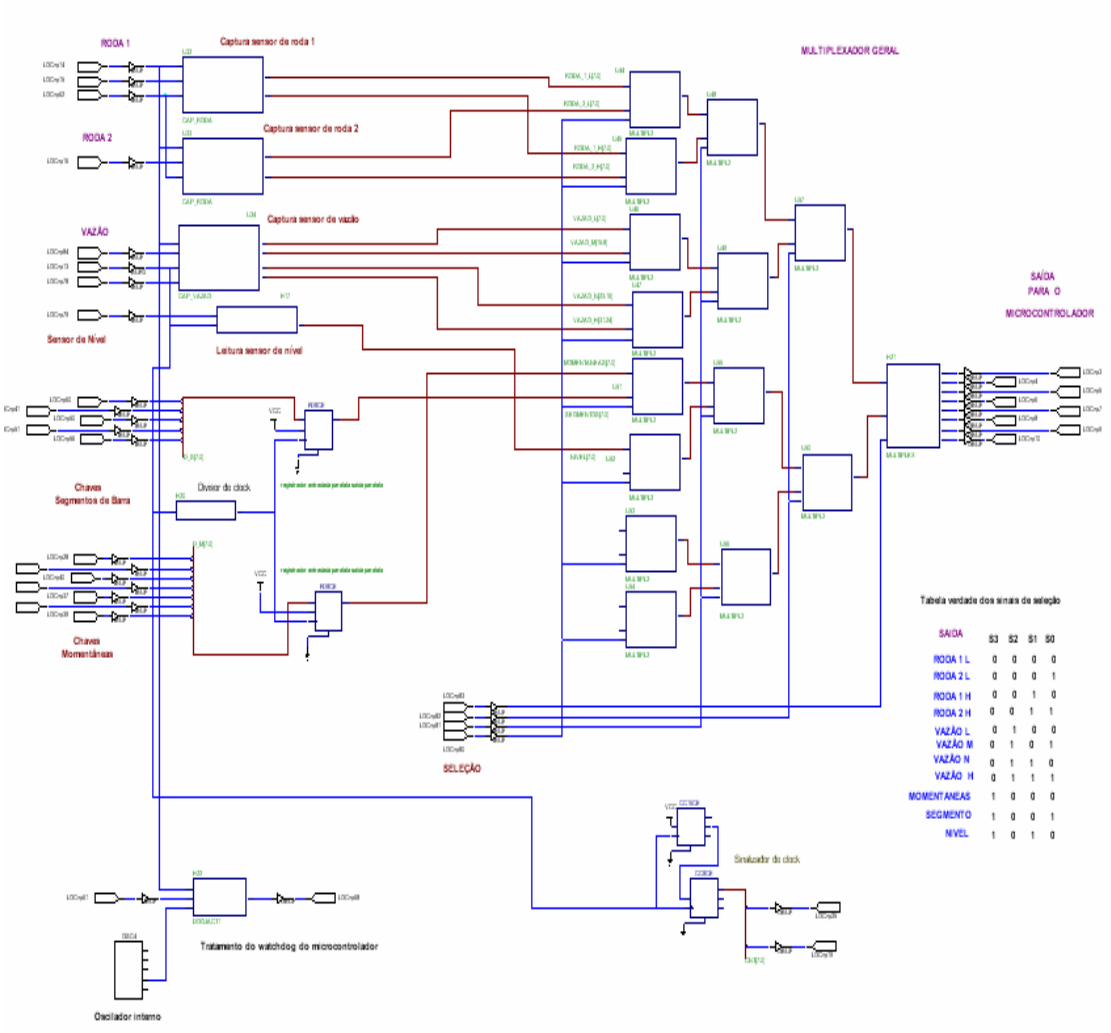

Figura 48 – Diagrama lógico do circuito geral

Considerando os sinais apresentados nas Figuras 32, 33, 40 e 48, a Tabela 3 apresenta a pinagem escolhida para o FPGA utilizado no teste de bancada.

Pino FPGA	Nome do Sinal	Tipo de Sinal
14	INCIALIZAR	ENTRADA
15	SINAL RODA 1	ENTRADA
62	CLK RODA (1 kHz)	ENTRADA
16	SINAL RODA 2	ENTRADA
84	SINAL VAZAO	ENTRADA
13	CLK GERAL (8 MHz)	ENTRADA
78	ESCALA PULSOS	ENTRADA
79	SINAL NIVEL	ENTRADA
60	CHAVE ALÍVIO RÁPIDO	ENTRADA
41	CHAVE SEGMENTO 1	ENTRADA
50	CHAVE SEGMENTO 2	ENTRADA
51	CHAVE SEGMENTO 3	ENTRADA
56	CHAVE SEGMENTO 4	ENTRADA
28	CHAVE AUTO/MANUAL	ENTRADA
29	CHAVE MAIS	ENTRADA
40	CHAVE MEMÓRIA	ENTRADA
35	CHAVE MENOS	ENTRADA
37	CHAVE FUNÇÃO (ACIMA)	ENTRADA
38	CHAVE FUNÇÃO (ABAIXO)	ENTRADA
39	CHAVE ZERAR	ENTRADA
61	SINAL WATCHDOG	ENTRADA
68	SINAL DE RESET	SAÍDA
25	S0 (DISPLAY 7 SEG)	SAÍDA
19	S6 (DISPLAY 7 SEG)	SAÍDA
80	SELEÇÃO S0	ENTRADA
81	SELEÇÃO S1	ENTRADA
82	SELEÇÃO S2	ENTRADA
83	SELEÇÃO S3	ENTRADA
3	DADO 7	SAÍDA
4	DADO 6	SAÍDA
5	DADO 5	SAÍDA
6	DADO 4	SAÍDA
7	DADO 3	SAÍDA
8	DADO 2	SAÍDA
9	DADO 1	SAÍDA
10	DADO 0	SAÍDA

Tabela 3 – Pinagem determinada para o FPGA

6.4. Teste de Bancada

Para realizar o teste de bancada, foi necessário realizar a interligação da placa de desenvolvimento XS40 (Figura 36) com a placa de circuito impresso do controlador JMC1000/4 (Figura 29). A interligação foi realizada utilizando cabos flexíveis com 0,3 mm² de bitola, que foram soldados em pontos específicos da placa do controlador para obter os sinais com nível de tensão ajustado para 3,3 V e nas vias das barras de pinos laterais disponibilizadas pela placa XS40 para acesso aos pinos do FPGA. Os cabos foram etiquetados um a um para auxiliar a identificação dos sinais para medição e observação das formas de onda. A Figura 49 apresenta a interligação entre as placas de circuito impresso.

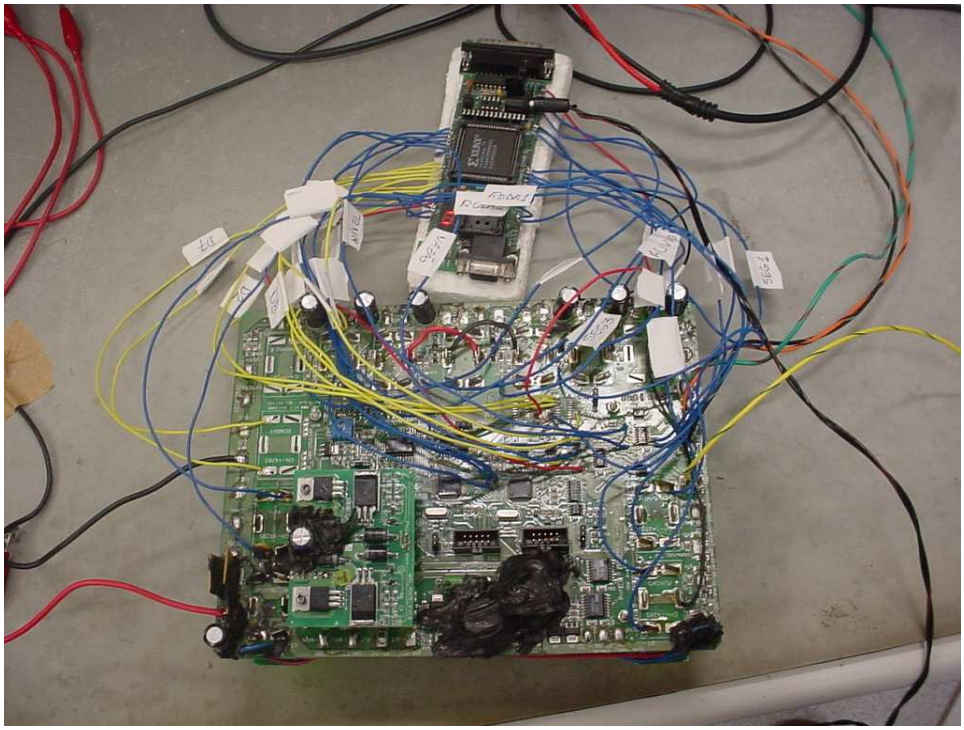

Figura 49 – Interligação entre as placas de circuito impresso

Para alimentar a placa do controlador foi utilizada uma fonte de tensão contínua ajustada para 13,8 V (Figura 50). Uma fonte de tensão contínua de 6 V foi utilizada para alimentar a placa de desenvolvimento XS40.

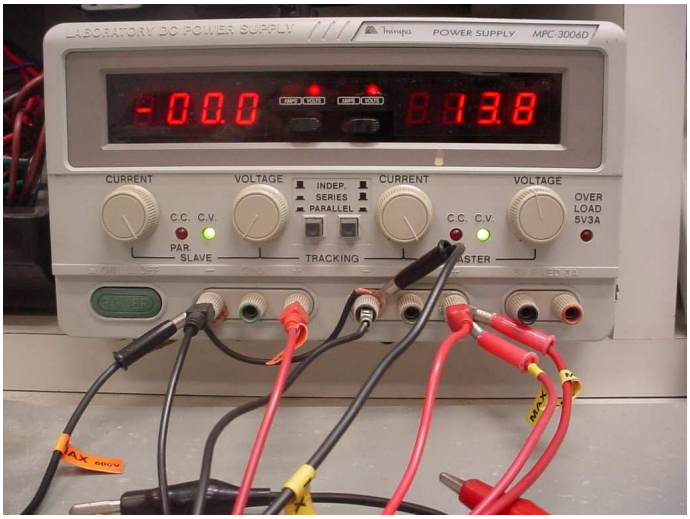

Figura 50 – Fonte de alimentação para a placa do controlador

Para simular os pulsos dos sensores de roda e vazão, foram utilizados geradores de sinais de onda quadrada (Figura 51). A simulação do pulso do sensor de nível foi realizada utilizando-se um sinal de 0V, que simula um nível lógico zero, para indicar nível mínimo do tanque de defensivos.

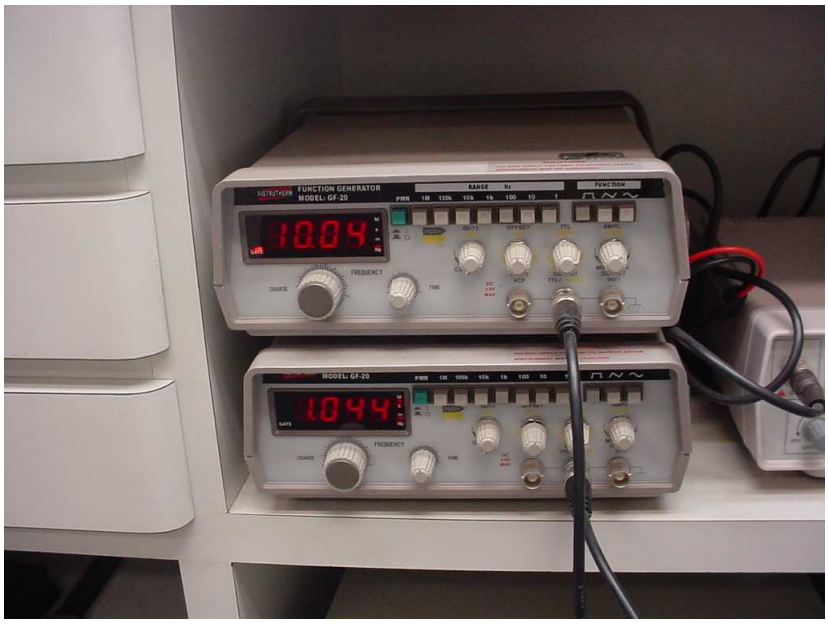

Figura 51 – Geradores de sinais para simulação dos sensores de roda e vazão

A visualização das formas de onda dos sinais, bem como a medição de nível de tensão e freqüência, foram realizadas utilizando-se um osciloscópio (Figura 52). O

osciloscópio utilizado basicamente apresenta as formas de onda de sinais capturados pelas pontas de prova em dois canais, sendo um canal para cada ponta. Para cada canal é possível ao usuário ajustar as escalas de tensão e tempo para realização das medições.

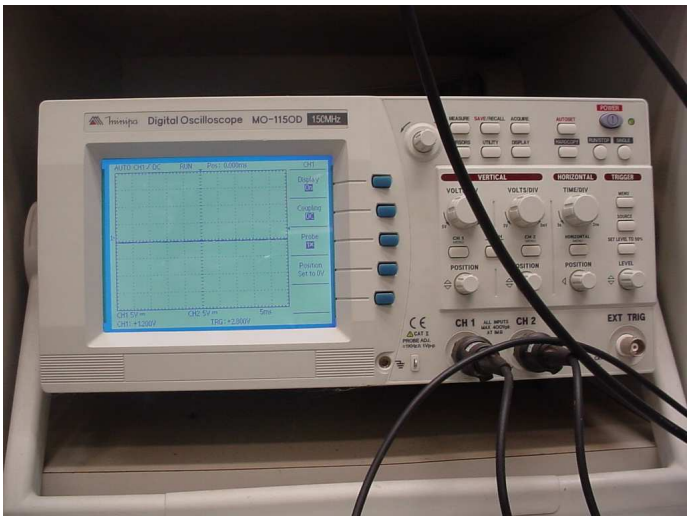

Figura 52 – Osciloscópio utilizado no teste de bancada

O *software* original do controlador foi alterado para desabilitar as rotinas residentes para captura de pulsos dos sensores e leitura de teclado. Foram acrescentadas rotinas para realizar a comunicação com o FPGA e obter os dados desejados (período dos sinais dos sensores de roda e vazão, status do sensor de nível e estado das chaves do teclado momentâneo e de segmentos). As demais funcionalidades de interfaceamento com o usuário foram mantidas para auxiliar os testes e a visualização de valores no *display* do controlador (Figura 53). O gerenciador de *watchdog* por *hardware* e os circuitos multiplexadores dos sinais de teclado descritos no item 5.4.1 foram desligados.

Após descarregar o arquivo *.bit* no FPGA da placa de desenvolvimento, a captura dos sensores de roda e vazão foi verificada observando-se os valores calculados para velocidade e vazão a partir das freqüências dos sinais de entrada gerados. A captura do sensor de nível foi verificada a partir da geração do alerta de nível mínimo apresentado no *display* do controlador, injetando na entrada o sinal de 0 V. A leitura do teclado momentâneo foi verificada a partir do acionamento das chaves e dos eventos gerados por cada uma como, por exemplo, navegação entre as diversas funções do menu de operação do controlador e digitação de valores. A leitura do teclado de segmentos foi verificada a partir do desenho dos segmentos da barra de pulverização que o *display* do controlador apresenta em função do

acionamento das chaves. O tratamento do watchdog foi verificado inibindo-se a onda quadrada gerada pelo microcontrolador e observando os pulsos gerados pelo FPGA no pino de *reset* do microcontrolador.

A Figura 54 apresenta uma visão geral da bancada de teste.

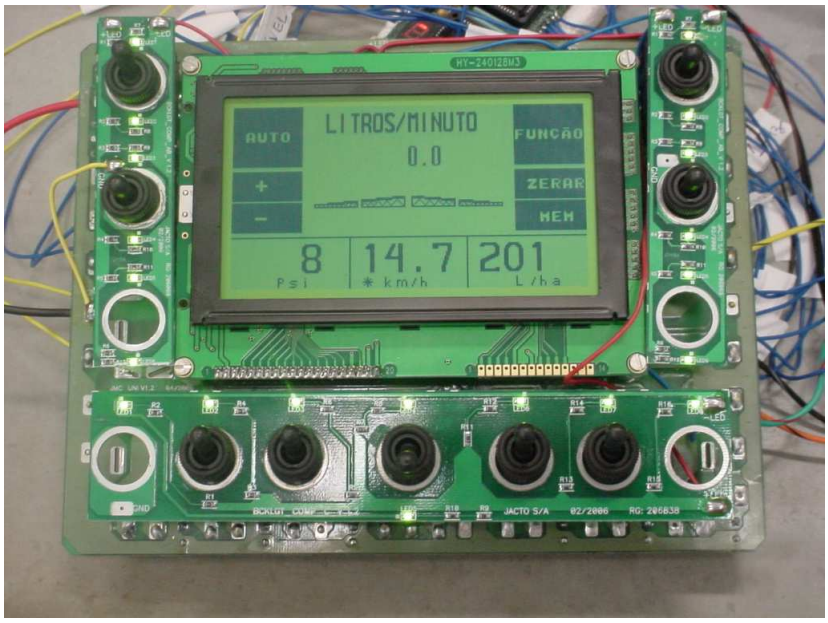

Figura 53 – Display do controlador JMC1000/4

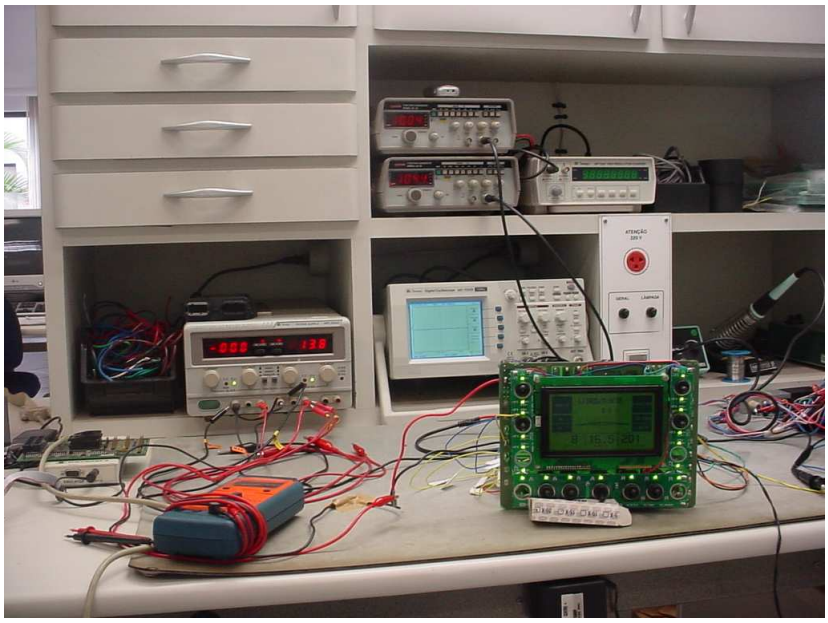

Figura 54 – Visão geral da bancada de teste

7. CONCLUSÃO

As etapas de desenvolvimento do trabalho foram inicialmente delineadas da seguinte forma:

1) Levantamento bibliográfico.

2) Estudo de ferramentas de projeto, simulação e implementação de subsistemas eletrônicos digitais utilizando componentes programáveis (FPGAs).

3) Estudo e classificação de equipamentos de automação e controle de pulverização em máquinas agrícolas comercializados.

4) Definição de Estudo de caso

5) Levantamento de requisitos do estudo de caso

6) Definição das atividades e/ou funções viáveis de serem automatizadas em um projeto piloto.

7) Projeto, implementação dessas funções em um FPGA.

8) Simulação e testes de validação do sistema implementado.

9) Elaboração de documentação técnica e de usuário do sistema implementado.

10) Resultados e conclusões obtidas.

11) Documentação e elaboração da monografia

12) Sugestões de continuidade.

A Figura 55 apresenta o cronograma previsto para as etapas.

Cronograma de desenvolvimento:										
Etapas	**Tempo previsto**									
	fev	**mar**	**abr**	**mai**	**jun**	**jul**	**ago**	**set**	**out**	**nov**
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										
11										
12										

Figura 55 – Cronograma de desenvolvimento

As etapas de 1 a 6 foram cumpridas no primeiro semestre. As demais etapas foram realizadas ao longo do segundo semestre e somente a etapa 9 ainda não foi concluída, pois foram priorizadas outras etapas consideradas mais importantes para a finalização dos trabalhos.

Os alunos do Grupo de Iniciação Científica, Diego e Denison participaram efetivamente do projeto. Nas etapas 1, 2, 5, 7 e 8 a participação dos mesmos foi realizada de forma relevante e contribuiu decisivamente para a obtenção dos resultados obtidos.

Durante os testes de bancada surgiu uma dificuldade interessante relativa ao funcionamento do circuito. Embora a geração do *netlist* várias vezes não indicou a presença

de erros, após descarregar o arquivo *.bit* na placa de desenvolvimento, foi observado comportamento inadequado do sistema, com o microcontrolador recebendo valores nulos dos circuitos de captura de sensores e leitura de teclado. O problema somente foi solucionado após estudo minucioso dos relatórios gerados pelas ferramentas de implementação do *Project Manager*, conforme citado no item 6.2. Estava sendo utilizado nos circuitos citados, a macro de um registrador de 8 bits com entrada e saída paralela desenvolvido durante os trabalhos e o *Project Manager* simplesmente estava desligando todas as ligações e blocos envolvidos com o registrador, sem gerar mensagens de erro na geração do *netlist*. Essas informações somente apareceram nos relatórios de implementação, embora somente como aviso, uma vez que o arquivo *.bit* era gerado sem erros. O problema foi solucionado trocando-se a macro do registrador desenvolvido por um registrador do mesmo tipo existente na biblioteca da Xilinx.

Durante os testes práticos observou-se também instabilidade de leitura dos valores disponibilizados pelos circuitos de roda e vazão ao microcontrolador. Como as simulações provam que os circuitos projetados estão conceitualmente corretos, a instabilidade pode ser atribuída ao fato de que a contagem de pulsos ocorre continuamente no tempo e a consulta do microcontrolador ao barramento de informação é assíncrona. O microcontrolador pode estar solicitando valores em instantes próximos do término das contagens, o que gera leituras incorretas em virtude da reinicialização dos contadores. Propagação de atrasos nas portas lógicas dos circuitos de reinicialização dos contadores também pode ser uma causa da instabilidade na leitura. Outro fato observado foi que após cessarem os pulsos dos sensores, fica o sempre o último valor de contagem disponibilizado ao microcontrolador, que nunca recebe a informação de ausência de sinal.

O trabalho desenvolvido mostra ser possível a integração de elementos de eletrônica embarcada em FPGA, além de proporcionar, dentro do estudo de caso proposto, a universalização de funções, uma vez que pode ser criado um chip universal para ser utilizado em outros projetos de controladores de pulverização, minimizando o número de componentes de hardware. Atualmente o controlador JMC1000/4 utiliza dois microcontroladores e componentes periféricos para atender as especificações de projeto, conforme citado no item 5.3. A melhoria no desempenho também ocorre, uma vez que a captura de pulsos dos sensores realizada pelo FPGA ocorre praticamente em tempo real com atraso mínimo e o controlador JMC1000/4 faz uma captura “quase em tempo real” via interrupções, conforme citado no item 5.4.2.

Algumas sugestões de continuidade vem a seguir.

1) Sincronização da captura de pulsos do FPGA com o microcontrolador como tentativa de resolução do problema da instabilidade de leitura, ou seja, o microcontrolador solicita, via sinais de seleção (existem combinações de bits disponíveis e não utilizadas na Tabela 2) o status de “ocupado” aos circuitos de varredura, ou seja, enquanto o FPGA está coletando os pulsos, ele informa a partir de um código pelo barramento de dados que a coleta não está pronta. O micro então não solicita o valor do período dos pulsos. Quando o micro solicitar novamente o status de "ocupado" e a coleta estiver pronta, a FPGA disponibiliza os dados via barramento e inibe uma nova contagem de período enquanto o micro não fizer a leitura. Após a leitura, o micro envia via barramento de seleção outro código para liberar a leitura de um novo período. Outra solução mais simples seria o microcontrolador confirmar as leituras de cada byte disponibilizado pelo FPGA e somente aceitando valores iguais por ciclo de leitura. Incluir circuitos de atraso nos contadores é mais uma possível solução.

2) Criar circuito de *timeout* no FPGA para detectar ausência de pulsos dos sensores como tentativa para resolver o problema do microcontrolador não receber a informação da ausência de pulsos.

3) Estudar nova família de FPGA para criação do chip universal, uma vez que a família XC4000 está obsoleta.

4) Projetar nova placa de circuito impresso do controlador de pulverização utilizando o *chip* universal para teste funcional em um pulverizador.

5) Integrar o *core* completo do microcontrolador dentro do FPGA, o que levaria a um projeto de microcontrolador totalmente nacional.

REFERÊNCIAS BIBLIOGRÁFICAS

ATUADOR. In: **Wikipédia**: A Enciclopédia Livre. Disponível em: <http://pt.wikipedia.org/wiki/Atuador>. Acesso em: 20 mai. 2007.

BALASTREIRE, LUIZ ANTONIO. **O estado da arte da Agricultura de Precisão no Brasil**. ed. Piracicaba, SP : L.A. Balastreire, 2000. 224 p.

CRUVINEL, Paulo E. **Instrumentação agropecuária no agronegócio brasileiro do século XXI**: Parte 1. Disponível em: <http://www.embrapa.br/noticias/artigos/2000/artigo.2004-12-07.2432500046/mostra_artigo>. Acesso em: 09 jun. 2007.

GERICOTA, Manuel Gradim de Oliveira. **Metodologias de Teste para FPGAs (Field Programmable Gate Arrays) Integradas em Sistemas Reconfiguráveis**. 2003. Dissertação (Doutorado em Engenharia Eletrotécnica e de Computadores) – Faculdade de Engenharia da Universidade do Porto, Porto, 2003.

HONEYWELL. **Ultrasonic Distance Sensors**. Catalogue E90. Issue, December 1999. 51p. Catálogo.

HURN, JEFF. **GPS: Um Guia para a Próxima Utilidade**. ed. E.U.A. : Trimble, 1999. 76 p.

INSTITUTO FNP. **Agrianual 2006**: Anuário da Agricultura Brasileira. São Paulo, SP, p.149-434, out. 2005.

LOBO JÚNIOR, Manoel Ibrain. **Aplicação Terrestre com Pulverizadores Autopropelidos**. Disponível em: < http://www.pulverizador.com.br/aplic-terrestre-01.htm>. Acesso em: 15 mai. 2007.

LOBO JÚNIOR, Manoel Ibrain. **Pulverizador Tratorizado com Barras e Bicos Hidráulicos**. Disponível em: < http://www.pulverizador.com.br/aplic-terrestre-02.htm>. Acesso em: 15 mai. 2007.

LOBO JÚNIOR, Manoel Ibrain. **Pulverizador Turbo Atomizador**. Disponível em: <http://www.pulverizador.com.br/aplic-terrestre-03.htm>. Acesso em: 15 mai. 2007.

[JACTO 01] MÁQUINAS AGRÍCOLAS JACTO S/A. **Manual Técnico sobre Orientação de Pulverização**. Pompéia : s.n., 2000. 32 p.

[JACTO02] MÁQUINAS AGRÍCOLAS JACTO S/A. **Manual de Instruções**: JSC 4000 controlador eletrônico de pulverização. Pompéia : s.n., 1998. 27 p.

MARTINS, C. A. P. S., et al. **Computação Reconfigurável**: conceitos, tendências e aplicações. In: Ciência, Tecnologia e Inovação: Atalhos para o Futuro: anais. Campinas, 2003, v. 2, p. 339-388.

MICROCONTROLADOR. In: **Wikipédia**: A Enciclopédia Livre. Disponível em: <http://pt.wikipedia.org/wiki/Microcontrolador>. Acesso em: 01 jun. 2007.

ORDONEZ, Edward David Moreno et al**. Projeto, Desempenho e Aplicação de Sistemas Digitais em Circuitos Programáveis (FPGAs)**. 1. ed. atual. Pompéia, SP: Bless, 2003. 300p.

PEROSA, Gleyson Cortez. **A Eletrônica na Agricultura**. 2000. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica) – Universidade de Marília, Marília, SP, 2000.

[RECONF] **Reconfigurable Computing Definition**. Disponível em: <http://www.acm.uiuc.edu/sigarch/projects/reconf/report_1.html>. Acesso em 02 jun. 2007.

SANTOS, João Eduardo Granetti Garcia dos; SANTOS FILHO, Abílio Garcia dos. **Apostila de Máquinas Agrícolas**. Universidade Estadual Paulista, Campus Universitário de Bauru, Faculdade de Engenharia, Bauru, SP, 2001, p. 81-82.

SARAIVA, Antônio Mauro. **Eletrônica Embarcada e ISOBUS**. In: Congresso Brasileiro de Agricultura de Precisão, 2006, São Pedro, SP, p.4-5.

SENSE. **Sensores e Instrumentos**: Catálogo Geral'99. s.l., 1999. 230 p. Catálogo.

SENSOR. In: **Wikipédia**: A Enciclopédia Livre. Disponível em: <http://pt.wikipedia.org/wiki/Sensor>. Acesso em: 20 mai. 2007.

SILVA, Daniella Dias Cavalcante da. **Desenvolvimento de um IP Core de Pré-Processamento Digital de Sinais de Voz para Aplicação em Sistemas Embutidos**. 2006. Dissertação (Mestrado em Informática) - Universidade Federal de Campina Grande, Campina Grande, PE, 2006.

ANEXO A - Simulação dos Blocos de Captura dos Sensores de Roda

Seguem abaixo resultados da simulação dos blocos de captura dos sensores de roda, gerados pela ferramenta *Logic Simulator* do *Project Manager*.

1) Bloco de captura do sensor de roda 1

A Figura 56 apresenta o resultado da simulação considerando que o microcontrolador solicitou o byte menos significativo gerado pelo circuito de captura.

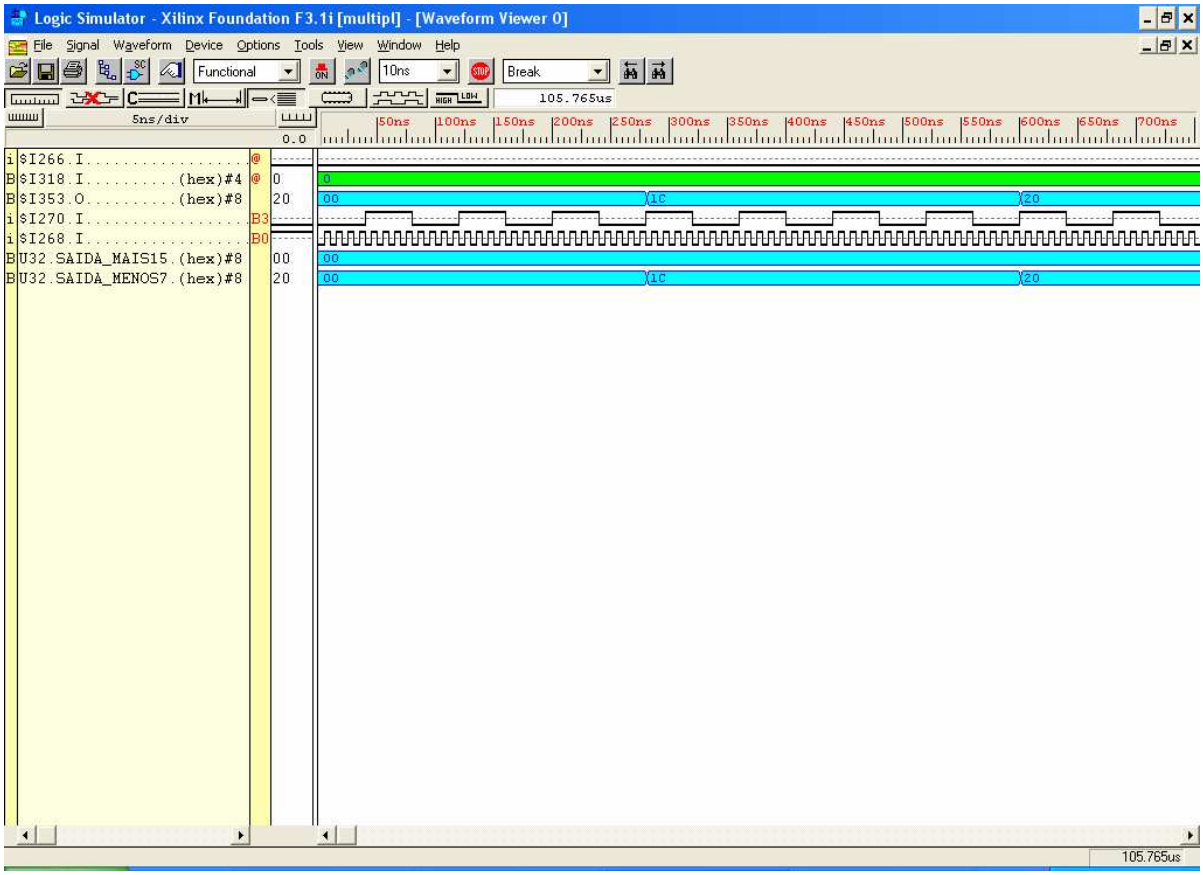

Figura 56 – Simulação da leitura do byte menos significativo (roda 1)

Observações, sendo as linhas de sinal orientadas de cima para baixo:

a) Linha 1 = sinal de inicialização

b) Linha 2 = sinais de seleção (de acordo com a Tabela 2, 0H = byte menos significativo para o sensor de roda 1)

c) Linha 3 = barramento de saída lido pelo microcontrolador (valor 20H)

d) Linha 4 = sinal hipotético que representa os pulsos do sensor de roda 1

e) Linha 5 = sinal hipotético que representa o *clock* de varredura de 1kHz gerado pelo microcontrolador

f) Linha 6 = byte mais significativo gerado pelo circuito de captura

g) Linha 7 = byte menos significativo gerado pelo circuito de captura

As linhas 6 e 7 contém o número de contagens (20H) do *clock* de varredura para 4 pulsos do sensor de roda 1.

A Figura 57 apresenta o resultado da simulação considerando que o microcontrolador solicitou o byte mais significativo gerado pelo circuito de captura.

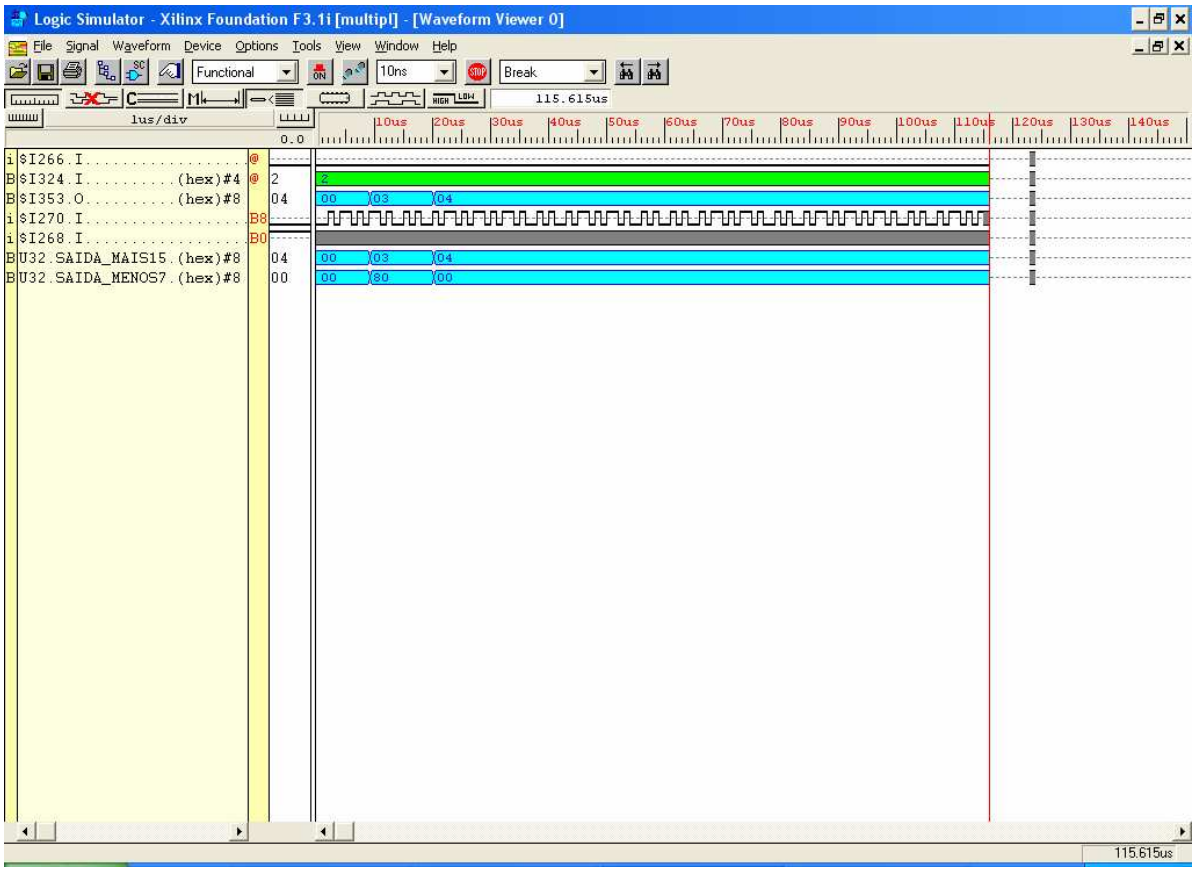

Figura 57 – Simulação da leitura do byte mais significativo (roda 1)

Observações, sendo as linhas de sinal orientadas de cima para baixo:

a) Linha 1 = sinal de inicialização

b) Linha 2 = sinais de seleção (de acordo com a Tabela 2, 02H = byte mais significativo para o sensor de roda 1)

c) Linha 3 = barramento de saída lido pelo microcontrolador (valor 04H)

d) Linha 4 = sinal hipotético que representa os pulsos do sensor de roda 1

e) Linha 5 = sinal hipotético que representa o *clock* de varredura de 1kHz gerado pelo microcontrolador

f) Linha 6 = byte mais significativo gerado pelo circuito de captura

g) Linha 7 = byte menos significativo gerado pelo circuito de captura

As linhas 6 e 7 contém o número de contagens (400H) do *clock* de varredura para 4 pulsos do sensor de roda 1.

2) Bloco de captura do sensor de roda 2

A Figura 58 apresenta o resultado da simulação considerando que o microcontrolador solicitou o byte menos significativo gerado pelo circuito de captura.

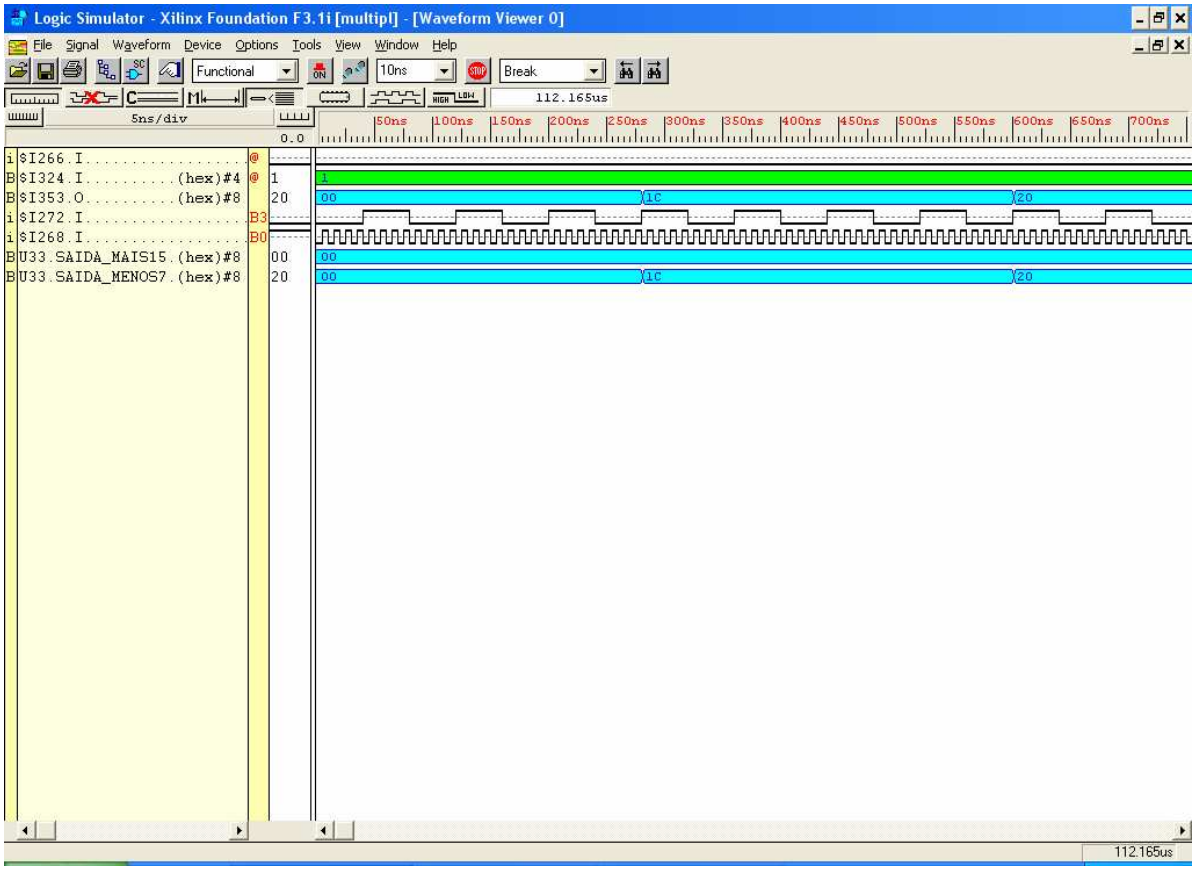

Figura 58 – Simulação da leitura do byte menos significativo (roda 2)

Observações, sendo as linhas de sinal orientadas de cima para baixo:

a) Linha 1 = sinal de inicialização

b) Linha 2 = sinais de seleção (de acordo com a Tabela 2, 01H = byte menos significativo para o sensor de roda 2)

c) Linha 3 = barramento de saída lido pelo microcontrolador (valor 20H)

d) Linha 4 = sinal hipotético que representa os pulsos do sensor de roda 2

e) Linha 5 = sinal hipotético que representa o *clock* de varredura de 1kHz gerado pelo microcontrolador

f) Linha 6 = byte mais significativo gerado pelo circuito de captura

g) Linha 7 = byte menos significativo gerado pelo circuito de captura

As linhas 6 e 7 contém o número de contagens (20H) do *clock* de varredura para 4 pulsos do sensor de roda 2.

A Figura 59 apresenta o resultado da simulação considerando que o microcontrolador solicitou o byte mais significativo gerado pelo circuito de captura.

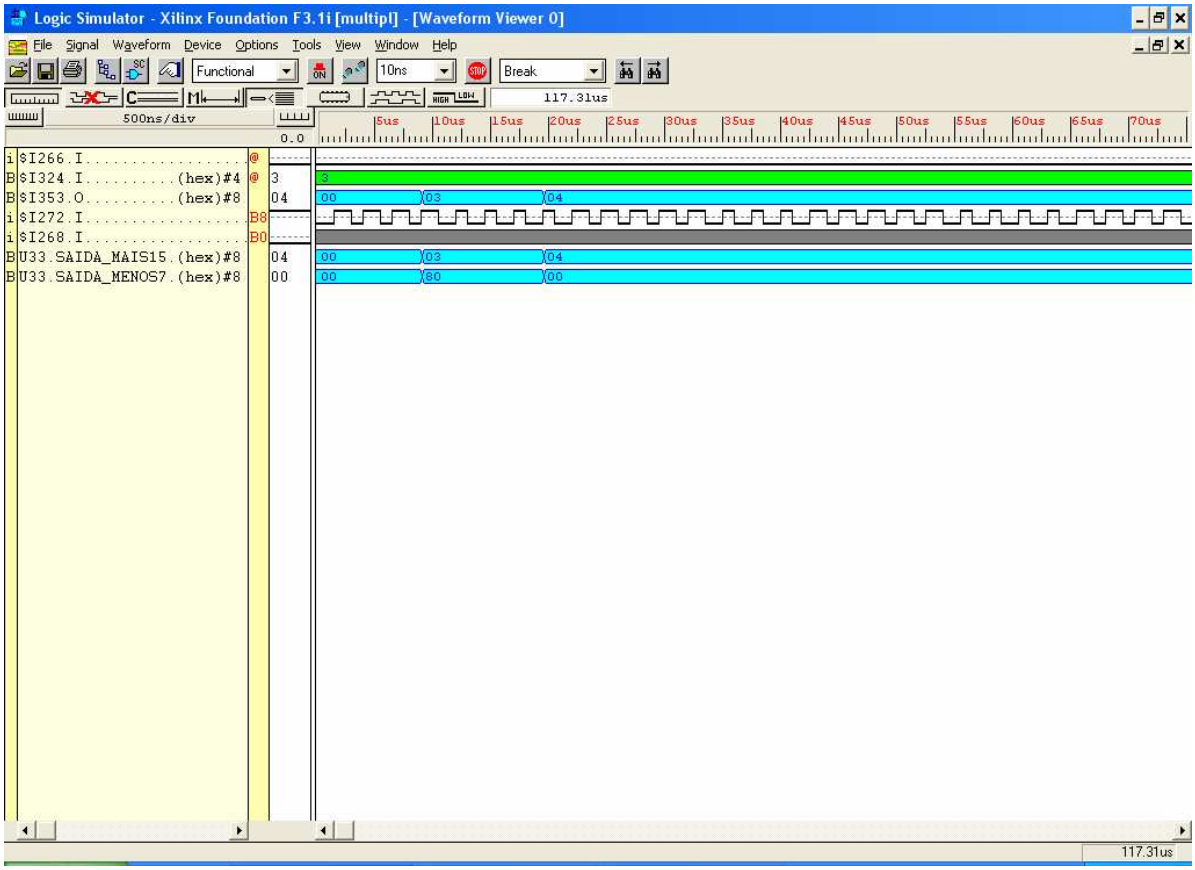

Figura 59 – Simulação da leitura do byte mais significativo (roda 2)

Observações, sendo as linhas de sinal orientadas de cima para baixo:

a) Linha 1 = sinal de inicialização

b) Linha 2 = sinais de seleção (de acordo com a Tabela 2, 03H = byte mais significativo para o sensor de roda 2)

c) Linha 3 = barramento de saída lido pelo microcontrolador (valor 04H)

d) Linha 4 = sinal hipotético que representa os pulsos do sensor de roda 2

e) Linha 5 = sinal hipotético que representa o *clock* de varredura de 1kHz gerado pelo microcontrolador

f) Linha 6 = byte mais significativo gerado pelo circuito de captura

g) Linha 7 = byte menos significativo gerado pelo circuito de captura

As linhas 6 e 7 contém o número de contagens (400H) do *clock* de varredura para 4 pulsos do sensor de roda 2.

ANEXO B - Simulação do Bloco de Captura do Sensor de Vazão

Seguem abaixo resultados da simulação do bloco de captura do sensor de vazão, gerados pela ferramenta *Logic Simulator* do *Project Manager*.

A Figura 60 apresenta o resultado da simulação considerando que o microcontrolador solicitou o byte menos significativo gerado pelo circuito de captura e que são coletados 32 pulsos por captura.

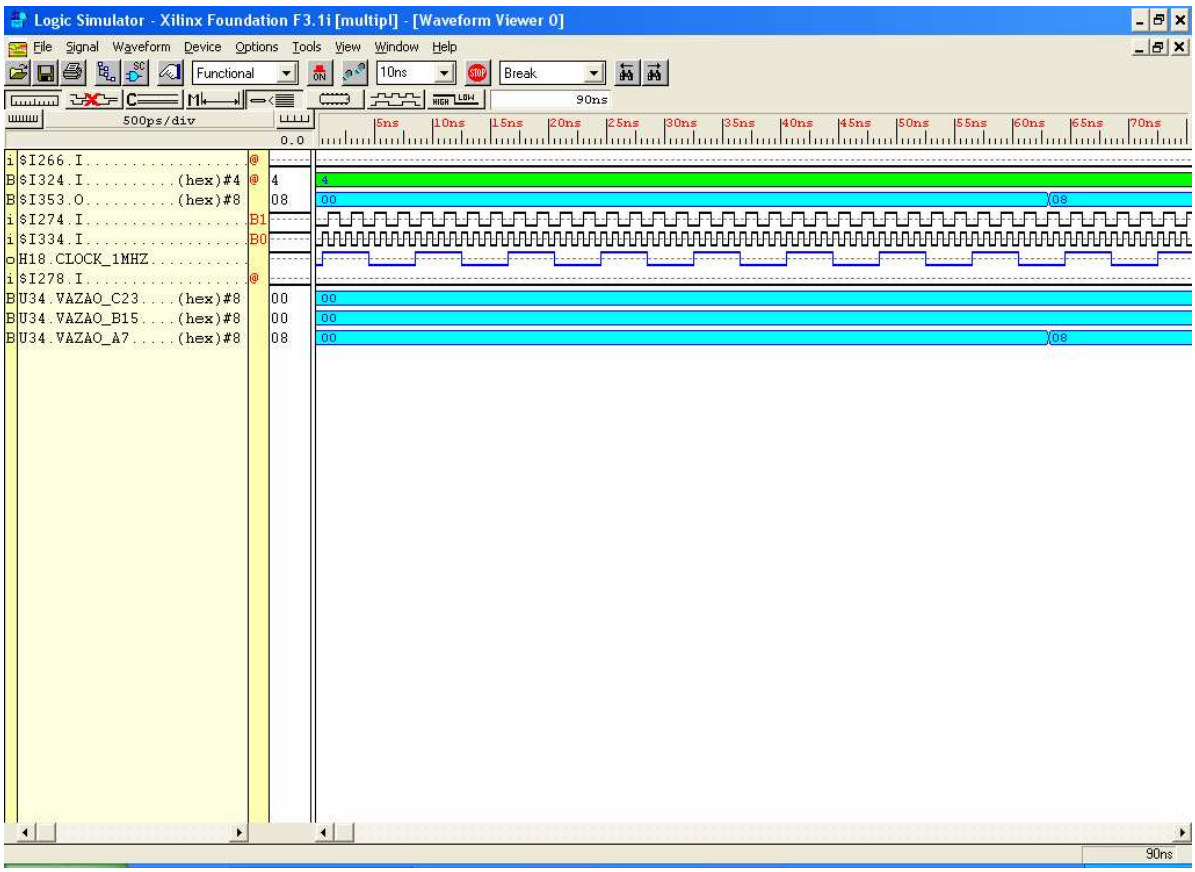

Figura 60 – Simulação da leitura do byte menos significativo (vazão) para 32 pulsos

Observações, sendo as linhas de sinal orientadas de cima para baixo:

a) Linha 1 = sinal de inicialização

b) Linha 2 = sinais de seleção (de acordo com a Tabela 2, 04H = byte menos significativo para o sensor de vazão)

c) Linha 3 = barramento de saída lido pelo microcontrolador (valor 08H)

d) Linha 4 = sinal hipotético que representa os pulsos do sensor de vazão

e) Linha 5 = sinal hipotético que representa o *clock* de 8 MHz gerado pelo microcontrolador

f) Linha 6 = *clock* de varredura (obtido a partir da divisão por 8 do *clock* gerado pelo microcontrolador)

g) Linha 7 = sinal de escala de pulsos (de acordo com a Tabela 1, o nível lógico "0" sinaliza coleta de 32 pulsos)

h) Linha 8 = byte mais significativo gerado pelo circuito de captura

i) Linha 9 = byte intermediário gerado pelo circuito de captura

j) Linha 10 = byte menos significativo gerado pelo circuito de captura

As linhas 8, 9 e 10 contém o número de contagens (08H) do *clock* de varredura para 32 pulsos do sensor de vazão.

A Figura 61 apresenta o resultado da simulação considerando que o microcontrolador solicitou o byte intermediário gerado pelo circuito de captura e que são coletados 32 pulsos por captura.

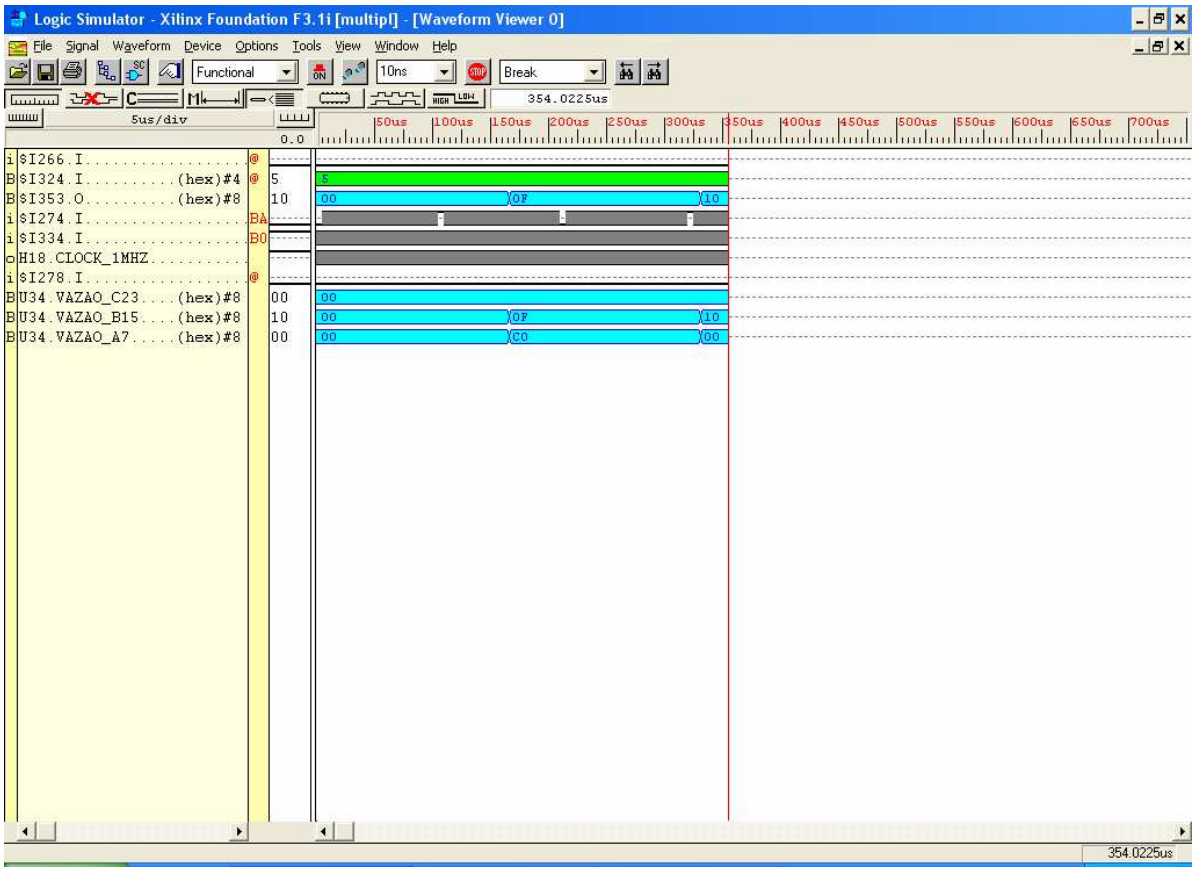

Figura 61 – Simulação da leitura do byte intermediário (vazão) para 32 pulsos

Observações, sendo as linhas de sinal orientadas de cima para baixo:

a) Linha 1 = sinal de inicialização

b) Linha 2 = sinais de seleção (de acordo com a Tabela 2, 05H = byte menos intermediário para o sensor de vazão)

c) Linha 3 = barramento de saída lido pelo microcontrolador (valor 10H)

d) Linha 4 = sinal hipotético que representa os pulsos do sensor de vazão

e) Linha 5 = sinal hipotético que representa o *clock* de 8 MHz gerado pelo microcontrolador

f) Linha 6 = *clock* de varredura (obtido a partir da divisão por 8 do *clock* gerado pelo microcontrolador)

g) Linha 7 = sinal de escala de pulsos (de acordo com a Tabela 1, o nível lógico “0" sinaliza coleta de 32 pulsos)

h) Linha 8 = byte mais significativo gerado pelo circuito de captura

i) Linha 9 = byte intermediário gerado pelo circuito de captura

j) Linha 10 = byte menos significativo gerado pelo circuito de captura

As linhas 8, 9 e 10 contém o número de contagens (1000H) do *clock* de varredura para 32 pulsos do sensor de vazão.

A Figura 62 apresenta o resultado da simulação considerando que o microcontrolador solicitou o byte mais significativo gerado pelo circuito de captura e que são coletados 32 pulsos por captura.

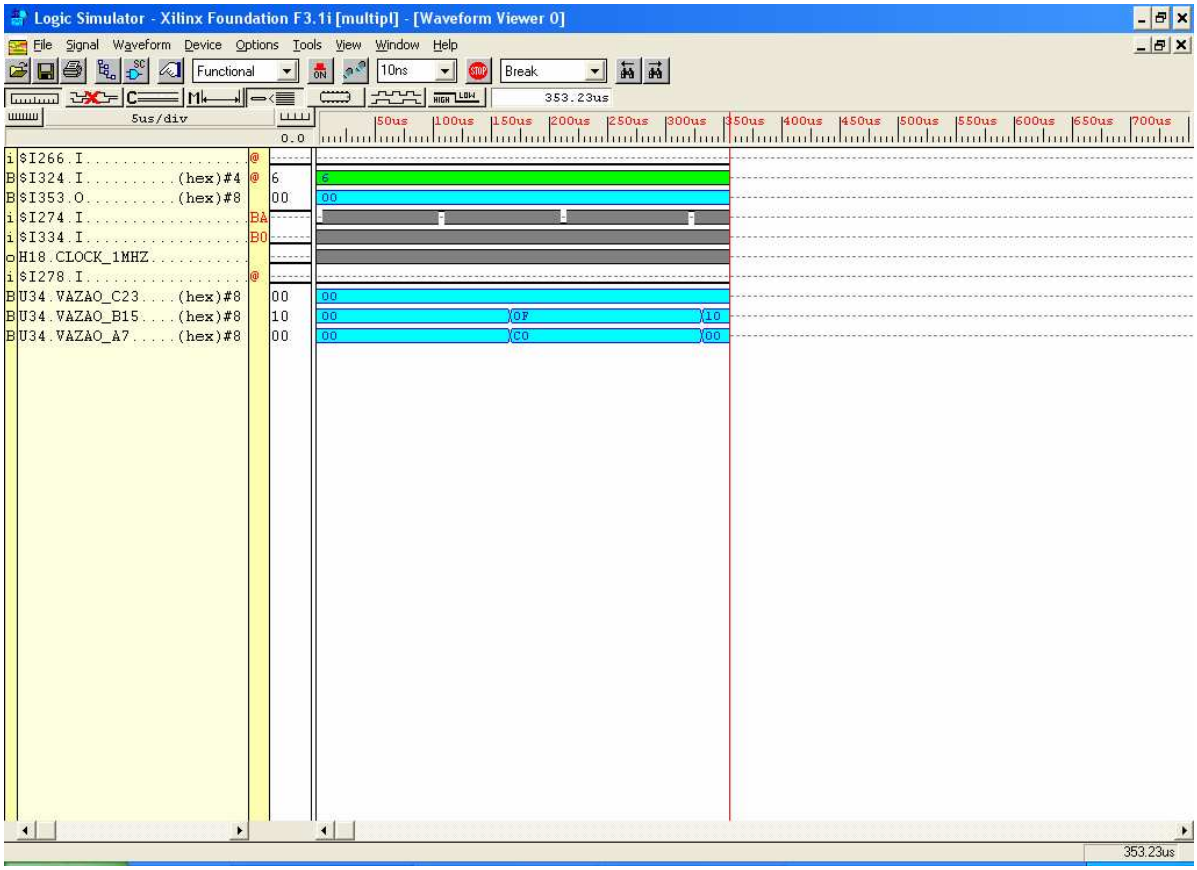

Figura 62 – Simulação da leitura do byte mais significativo (vazão) para 32 pulsos

Observações, sendo as linhas de sinal orientadas de cima para baixo:

a) Linha 1 = sinal de inicialização

b) Linha 2 = sinais de seleção (de acordo com a Tabela 2, 06H = byte mais significativo para o sensor de vazão)

c) Linha 3 = barramento de saída lido pelo microcontrolador (valor 0H)

d) Linha 4 = sinal hipotético que representa os pulsos do sensor de vazão

e) Linha 5 = sinal hipotético que representa o *clock* de 8 MHz gerado pelo microcontrolador

f) Linha 6 = *clock* de varredura (obtido a partir da divisão por 8 do *clock* gerado pelo microcontrolador)

g) Linha 7 = sinal de escala de pulsos (de acordo com a Tabela 1, o nível lógico “0" sinaliza coleta de 32 pulsos)

h) Linha 8 = byte mais significativo gerado pelo circuito de captura

i) Linha 9 = byte intermediário gerado pelo circuito de captura

j) Linha 10 = byte menos significativo gerado pelo circuito de captura

As linhas 8, 9 e 10 contém o número de contagens (1000H) do *clock* de varredura para 32 pulsos do sensor de vazão.

A Figura 63 apresenta o resultado da simulação considerando que o microcontrolador solicitou o byte menos significativo gerado pelo circuito de captura e que são coletados 64 pulsos por captura.

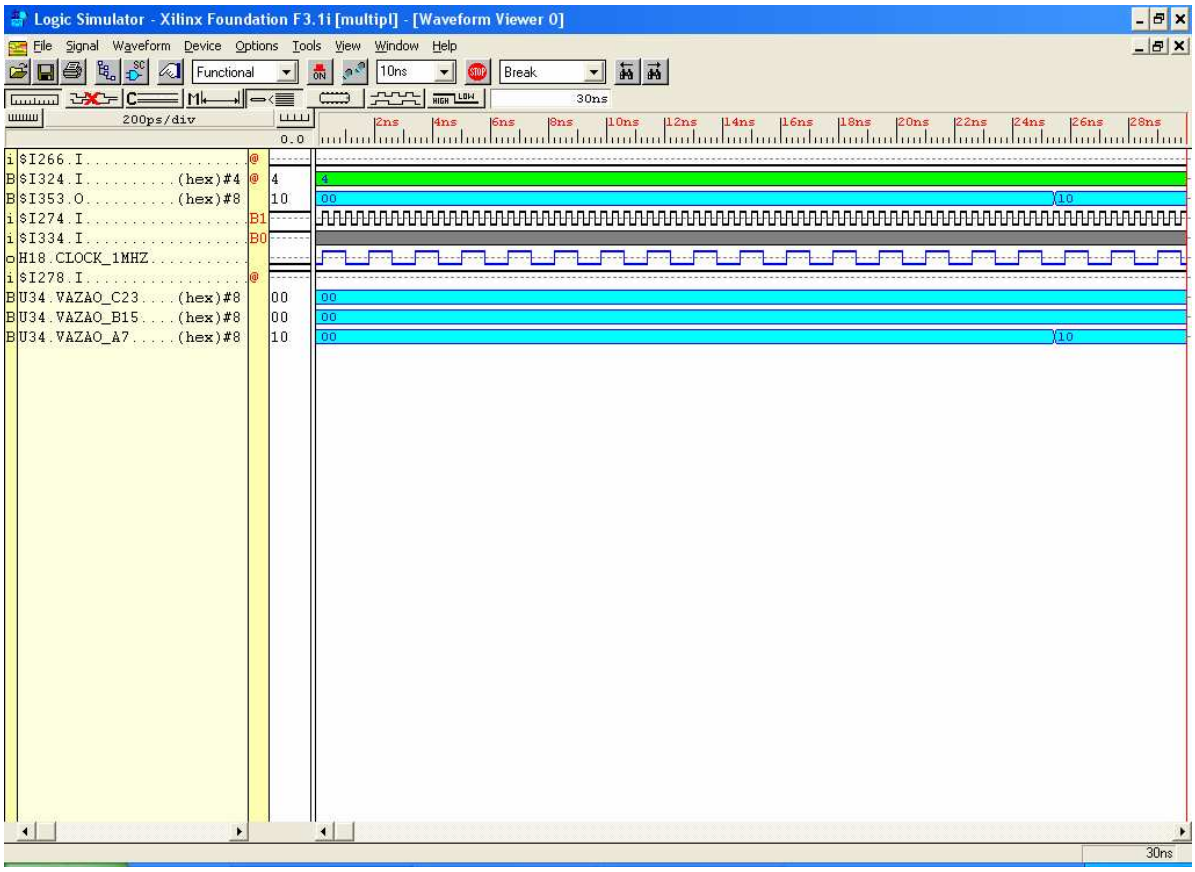

Figura 63 – Simulação da leitura do byte menos significativo (vazão) para 64 pulsos

Observações, sendo as linhas de sinal orientadas de cima para baixo:

a) Linha 1 = sinal de inicialização

b) Linha 2 = sinais de seleção (de acordo com a Tabela 2, 04H = byte menos significativo para o sensor de vazão)

c) Linha 3 = barramento de saída lido pelo microcontrolador (valor 10H)

d) Linha 4 = sinal hipotético que representa os pulsos do sensor de vazão

e) Linha 5 = sinal hipotético que representa o *clock* de 8 MHz gerado pelo microcontrolador

f) Linha 6 = *clock* de varredura (obtido a partir da divisão por 8 do *clock* gerado pelo microcontrolador)

g) Linha 7 = sinal de escala de pulsos (de acordo com a Tabela 1, o nível lógico “1" sinaliza coleta de 64 pulsos)

h) Linha 8 = byte mais significativo gerado pelo circuito de captura

i) Linha 9 = byte intermediário gerado pelo circuito de captura

j) Linha 10 = byte menos significativo gerado pelo circuito de captura

As linhas 8, 9 e 10 contém o número de contagens (10H) do *clock* de varredura para 64 pulsos do sensor de vazão.

Por analogia com o caso 32 pulsos, não serão apresentados os resultados da simulação para os demais bytes, uma vez que funcionou a troca de escala.

ANEXO C - Simulação dos Blocos de Leitura do Teclado

Seguem abaixo resultados da simulação dos blocos de leitura do teclado, gerados pela ferramenta *Logic Simulator* do *Project Manager*.

1) Bloco de leitura do teclado de segmentos

A Figura 64 apresenta o resultado da simulação da leitura do teclado de segmentos.

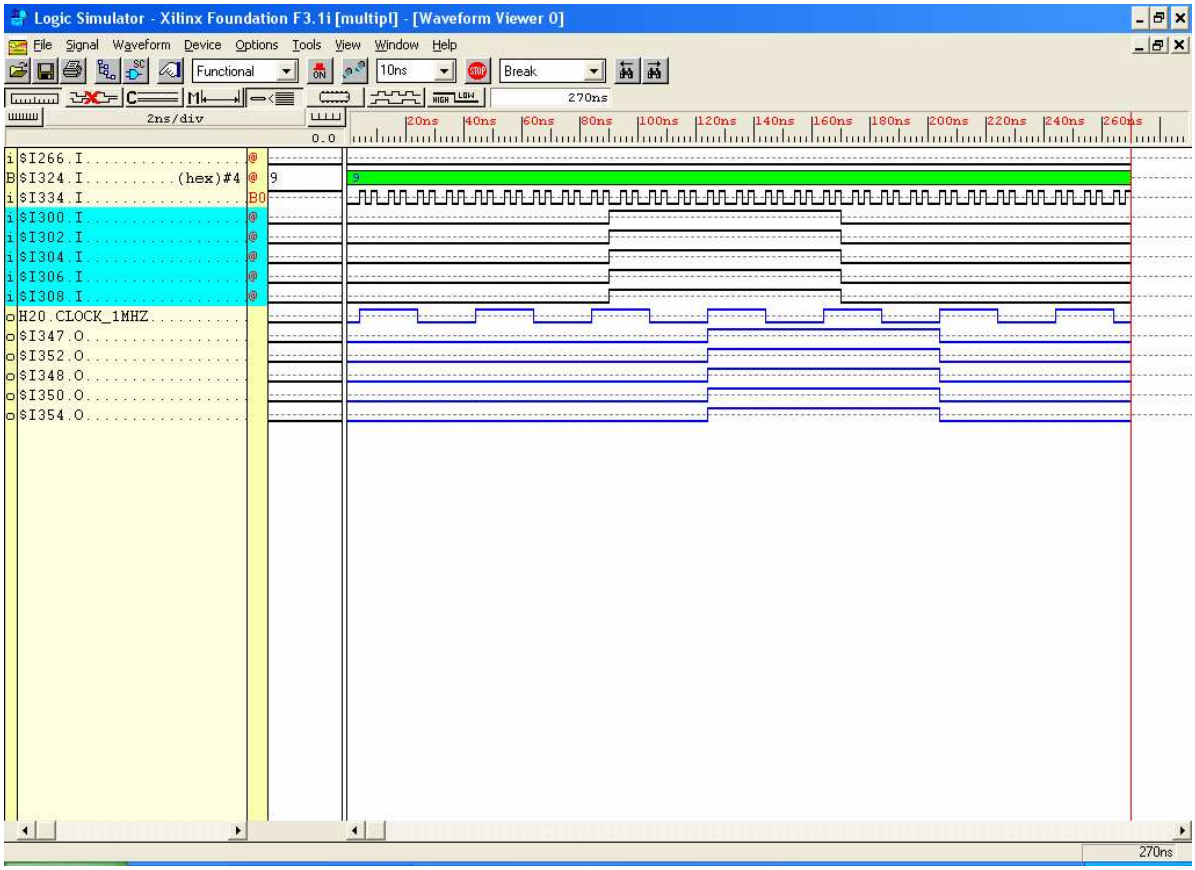

Figura 64 – Simulação da leitura do teclado de segmentos

Observações, sendo as linhas de sinal orientadas de cima para baixo:

a) Linha 1 = sinal de inicialização

b) Linha 2 = sinais de seleção (de acordo com a Tabela 2, 09H = byte relativo ao teclado de segmentos)

c) Linha 3 = sinal hipotético que representa o *clock* de 8 MHz gerado pelo microcontrolador

d) Linhas 4 a 8 = sinais das chaves de segmento

e) Linha 9 = *clock* de varredura (obtido a partir da divisão por 8 do *clock* gerado pelo microcontrolador)

f) Linhas 10 a 14 = barramento de saída lido pelo microcontrolador

O barramento de saída lido pelo microcontrolador é fiel ao estado das chaves de segmento. De acordo com a Tabela 3, são utilizados 5 sinais para as chaves de segmento.

2) Bloco de leitura do teclado momentâneo

A Figura 65 apresenta o resultado da simulação da leitura do teclado momentâneo.

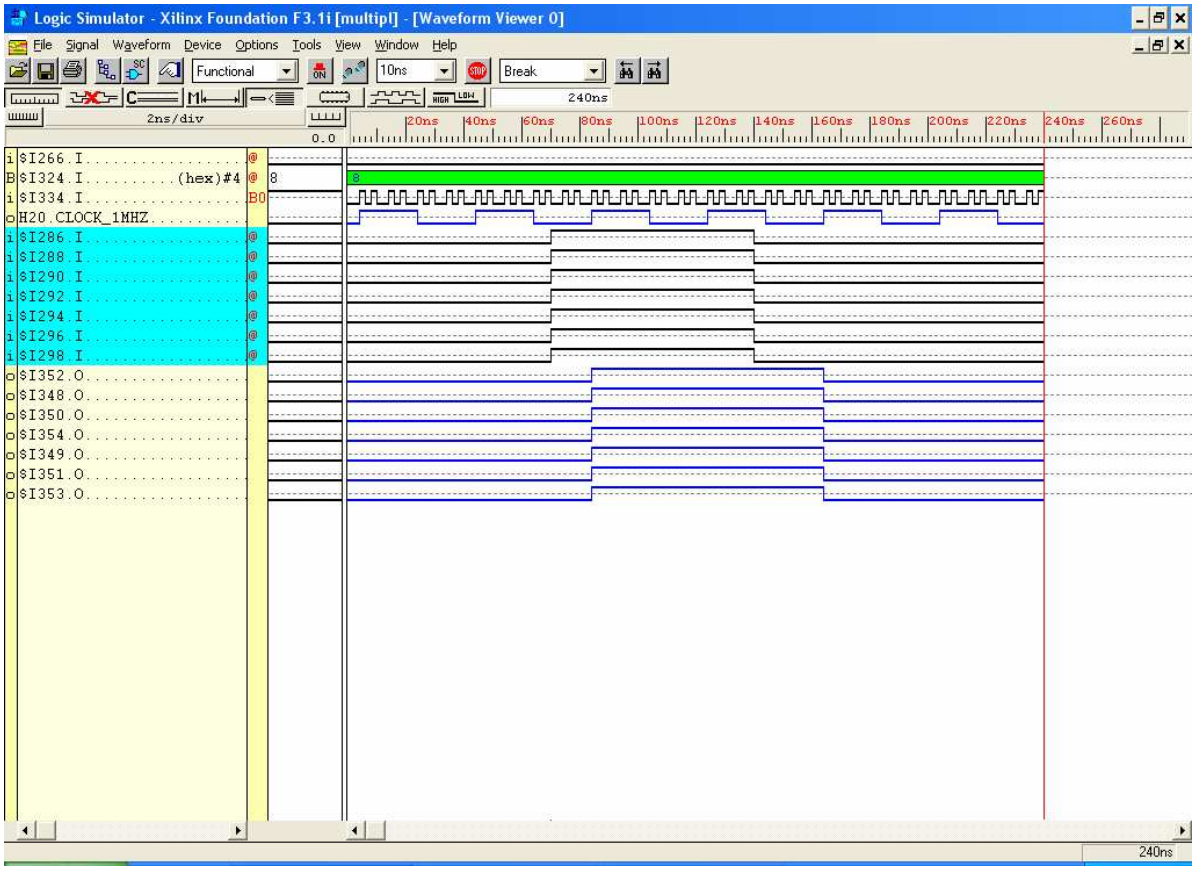

Figura 65 – Simulação da leitura do teclado momentâneo

Observações, sendo as linhas de sinal orientadas de cima para baixo:

a) Linha 1 = sinal de inicialização

b) Linha 2 = sinais de seleção (de acordo com a Tabela 2, 08H = byte relativo ao teclado momentâneo)

c) Linha 3 = sinal hipotético que representa o *clock* de 8 MHz gerado pelo microcontrolador

d) Linha 4 = *clock* de varredura (obtido a partir da divisão por 8 do *clock* gerado pelo microcontrolador)

e) Linhas 5 a 11 = sinais das chaves momentâneas

f) Linhas 12 a 18 = barramento de saída lido pelo microcontrolador

O barramento de saída lido pelo microcontrolador é fiel ao estado das chaves momentâneas. De acordo com a Tabela 3, são utilizados 7 sinais para as chaves momentâneas.

ANEXO D - Simulação do Bloco de Leitura do Sensor de Nível

Seguem abaixo resultados da simulação do bloco de leitura do sensor de nível, gerados pela ferramenta *Logic Simulator* do *Project Manager*.

A Figura 66 apresenta o resultado da simulação da leitura do sensor de nível.

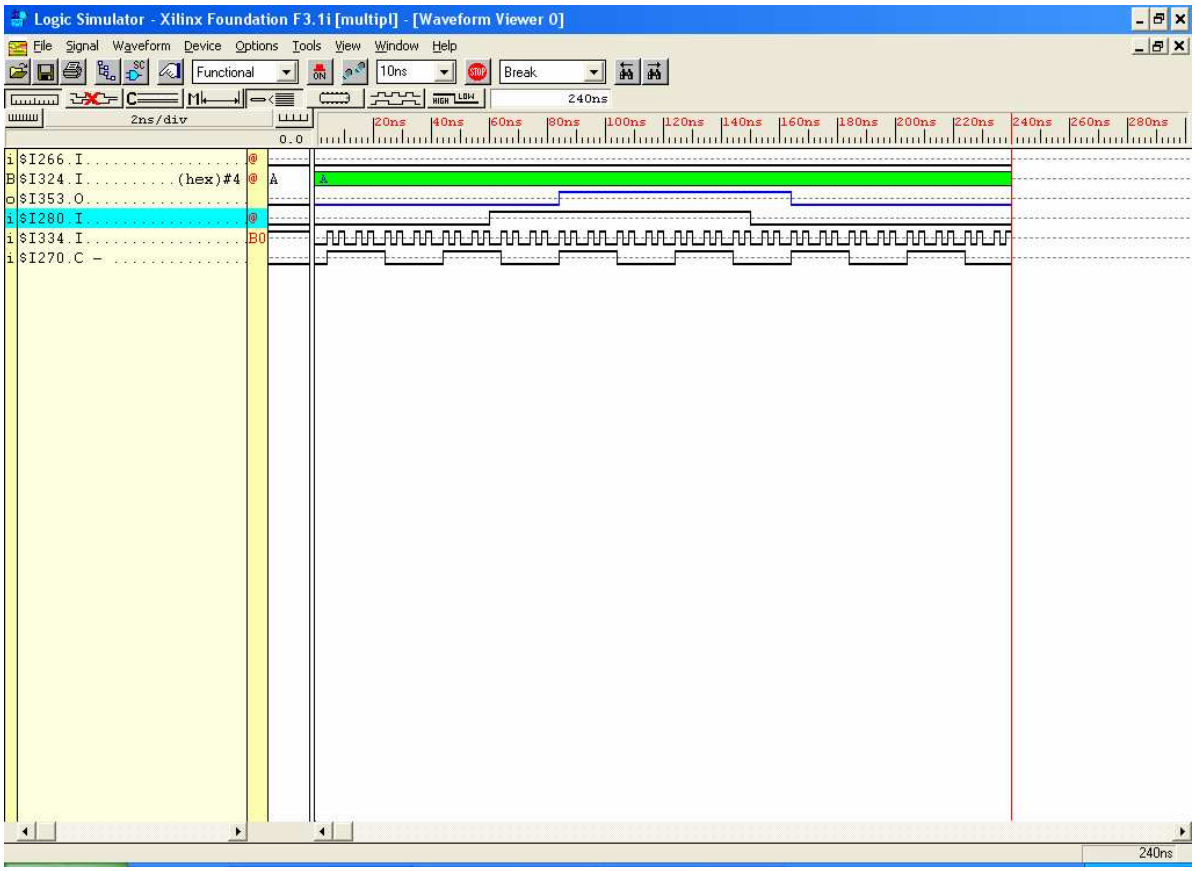

Figura 66 – Simulação da leitura do sensor de nível

Observações, sendo as linhas de sinal orientadas de cima para baixo:

a) Linha 1 = sinal de inicialização

b) Linha 2 = sinais de seleção (de acordo com a Tabela 2, 0AH = byte relativo ao sensor de nível, embora seja lido somente o bit menos significativo pelo microcontrolador)

c) Linha 3 = sinal de saída lido pelo microcontrolador

d) Linha 4 = sinal do sensor de nível

e) Linha 5 = sinal hipotético que representa o *clock* de 8 MHz gerado pelo microcontrolador

f) Linha 6 = *clock* de varredura (obtido a partir da divisão por 8 do *clock* gerado pelo microcontrolador)

O sinal de saída lido pelo microcontrolador é fiel ao sinal do sensor de nível.

ANEXO E - Simulação do Bloco de Tratamento do *Watchdog*

Seguem abaixo resultados da simulação do bloco de tratamento do *watchdog*, gerados pela ferramenta *Logic Simulator* do *Project Manager*.

A Figura 67 apresenta o resultado do bloco de tratamento do *watchdog*, considerando operação normal do microcontrolador.

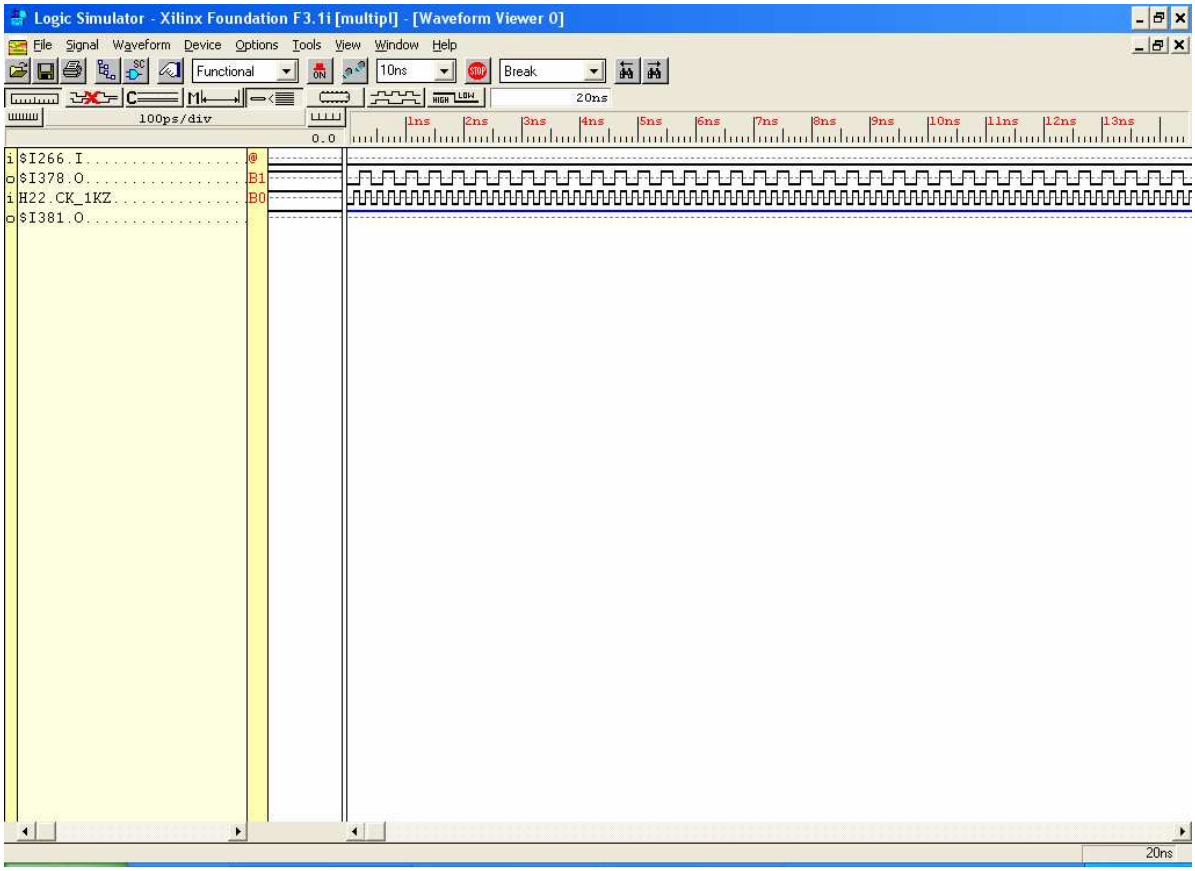

Figura 67 – Simulação do *watchdog* em operação normal do microcontrolador

Observações, sendo as linhas de sinal orientadas de cima para baixo:

a) Linha 1 = sinal de inicialização

b) Linha 2 = sinal *watchdog* gerado pelo microcontrolador para sinalizar seu funcionamento

c) Linha 3 = sinal hipotético que representa o *clock* gerado internamente no FPGA

d) Linha 4 = sinal de *reset* do microcontrolador

O sinal de reset fica sempre em nível lógico alto, uma vez que o microcontrolador está em funcionamento normal.

A Figura 68 apresenta o resultado do bloco de tratamento do *watchdog*, considerando falha no microcontrolador.

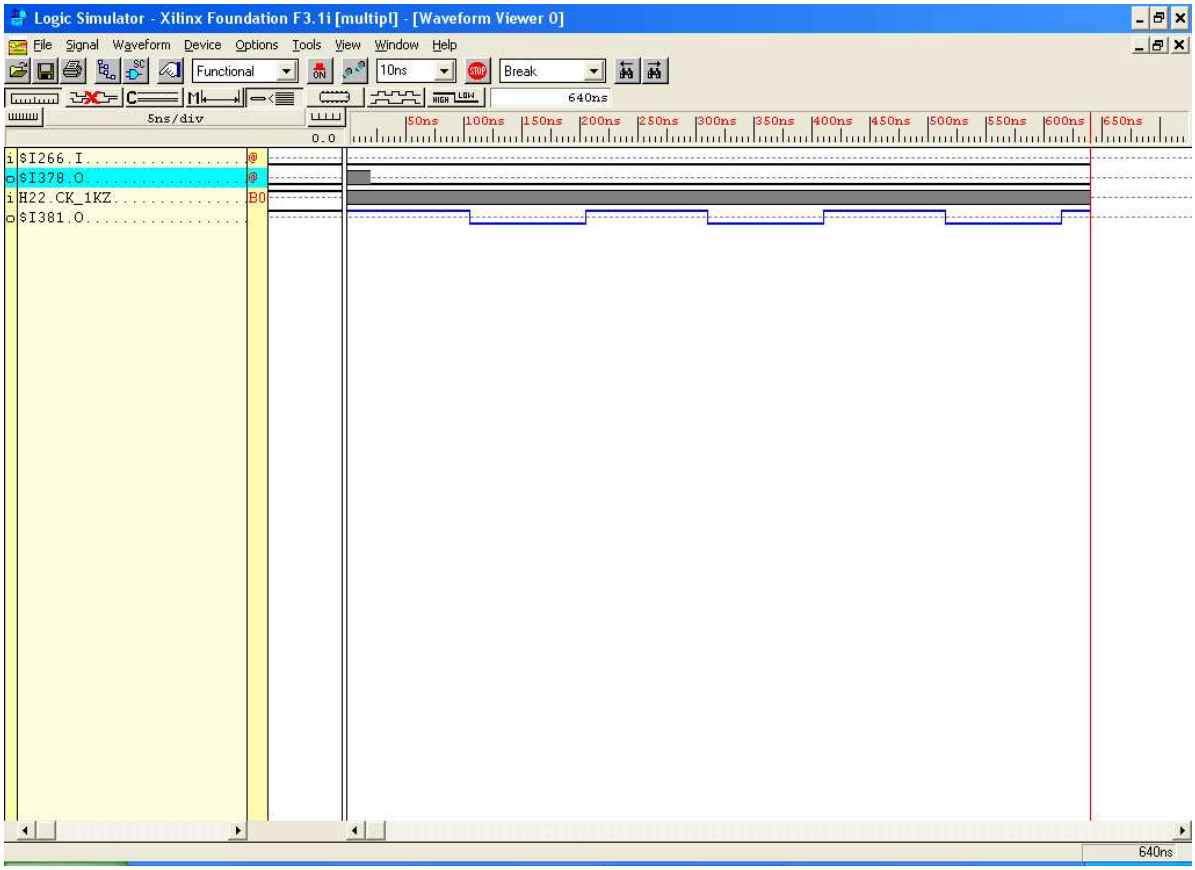

Figura 68 – Simulação do *watchdog* com falha no microcontrolador

Observações, sendo as linhas de sinal orientadas de cima para baixo:

a) Linha 1 = sinal de inicialização

b) Linha 2 = sinal *watchdog* gerado pelo microcontrolador para sinalizar seu funcionamento

c) Linha 3 = sinal hipotético que representa o *clock* gerado internamente no FPGA

d) Linha 4 = sinal de *reset* do microcontrolador

Depois de detectada a parada do microcontrolador, são gerados pulsos de *reset* para restabelecimento da normalidade.

ANEXO F – Outros Circuitos Importantes

Seguem os diagramas lógicos de outros circuitos considerados importantes e que fazem parte dos blocos apresentados.

1) Contador de 16 bits com *clear* e *clock enable*

Onde é utilizado: circuitos de captura de roda e vazão.

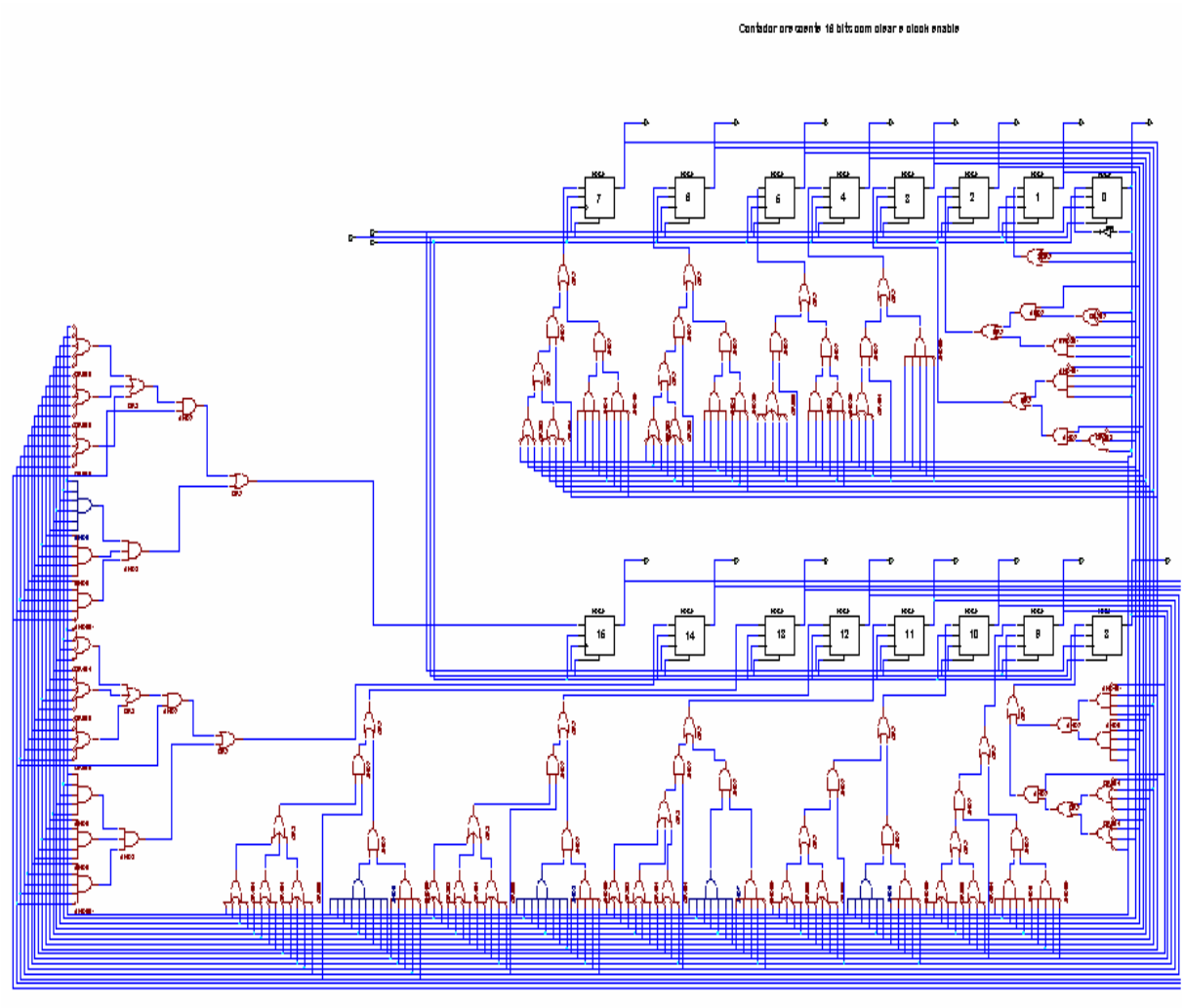

Figura 69 – Contador de 16 bits com *clear* e *clock enable*

2) Contador de 2 bits com *preset*

Onde é utilizado: circuitos de captura de roda.

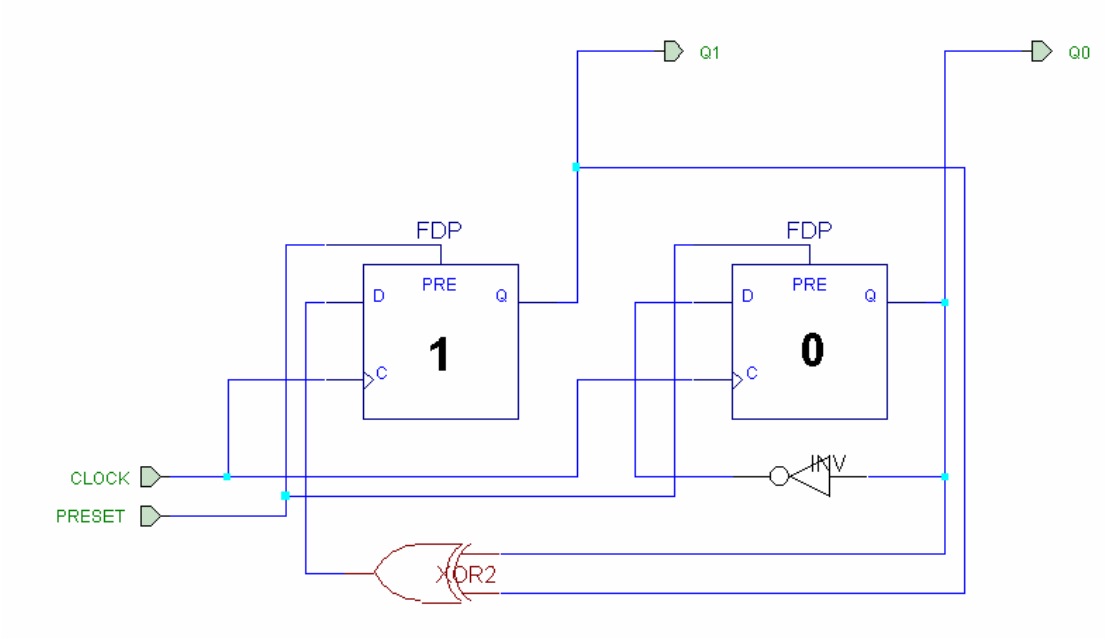

Figura 70 – Contador de 2 bits com *preset*

3) Registrador de 8 bits com *clear* e *clock enable*

Onde é utilizado: circuitos de captura (roda e vazão) e leitura de teclado.

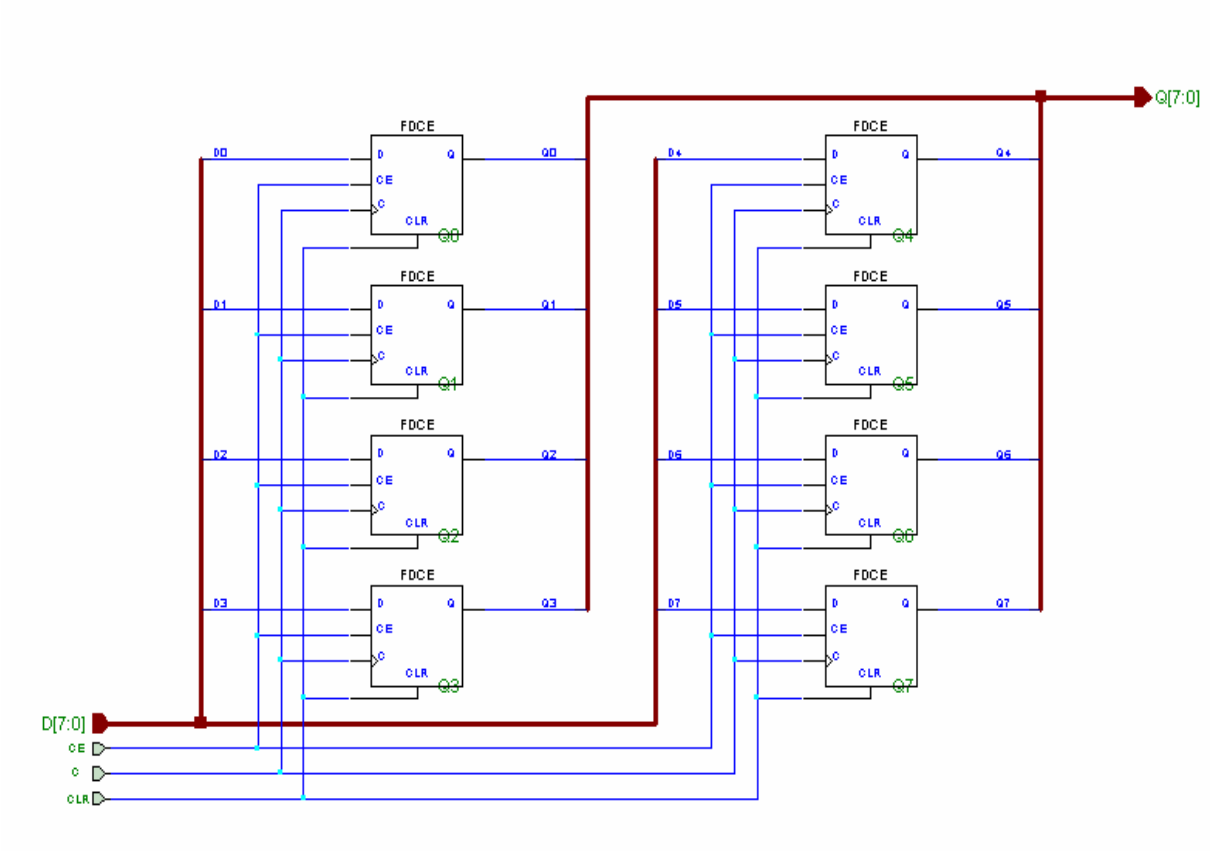

Figura 71 – Registrador de 8 bits com *clear* e *clock enable*

4) Contador de 6 bits com *preset* e *clock enable*

Onde é utilizado: circuito de captura de vazão.

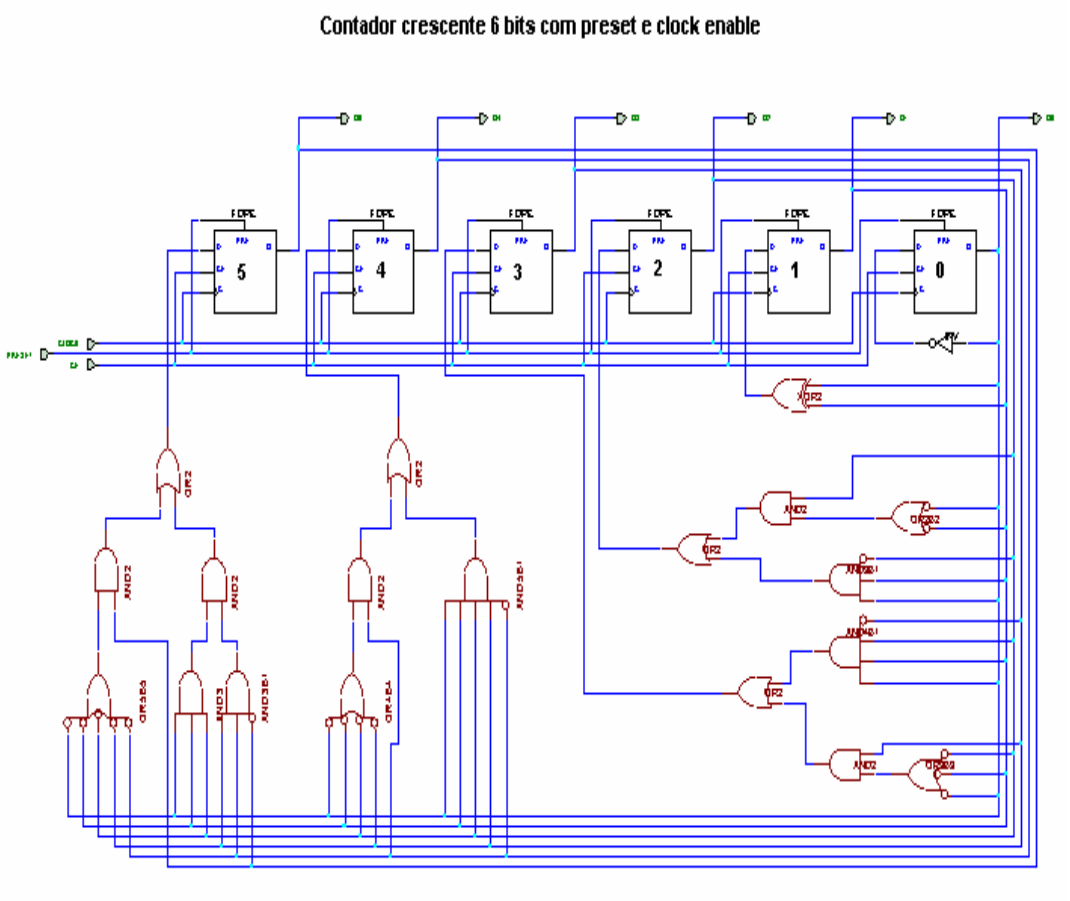

Figura 72 – Contador de 6 bits com *preset* e *clock enable*

5) Divisor de *clock*

Onde é utilizado: circuitos de captura de vazão, nível e leitura de teclado.

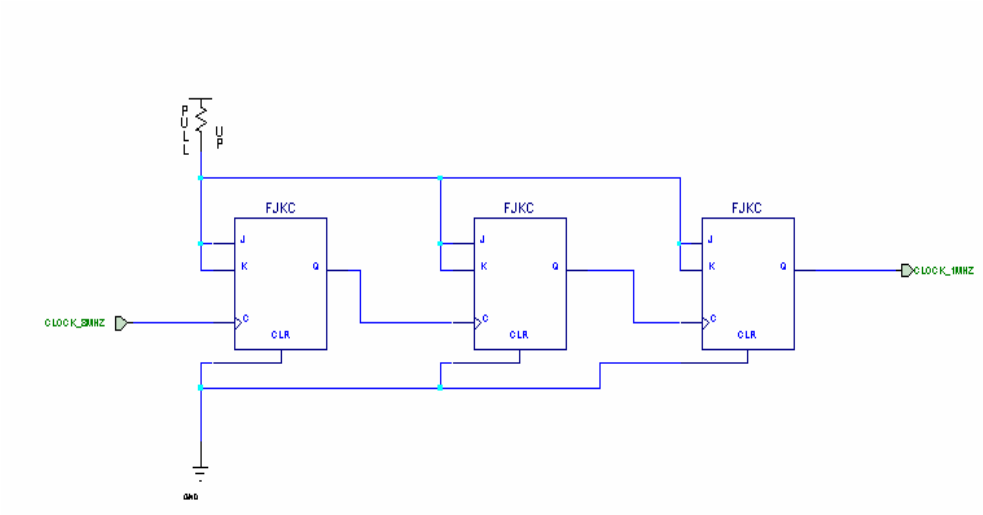

Figura 73 – Divisor de *clock*

6) Multiplex

Onde é utilizado: bloco multiplexador geral.

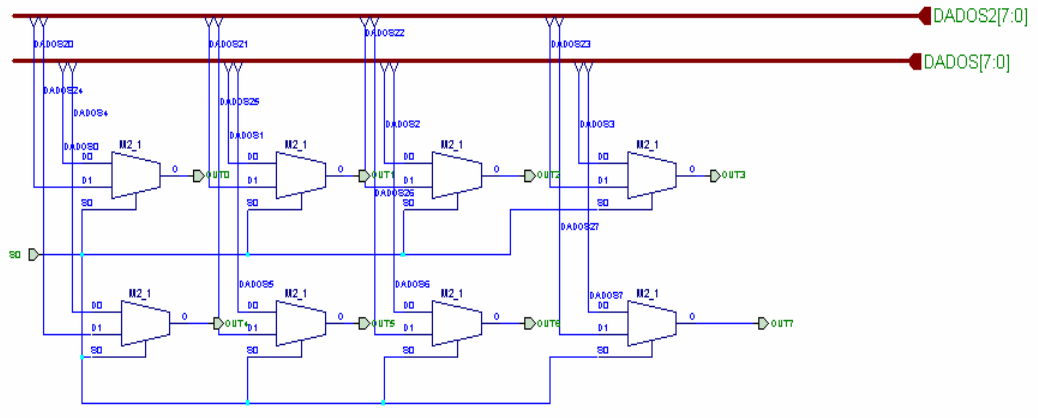

Figura 74 – Multiplex

7) Registrador de 16 bits com *clear*, *preset* e *clock enable*

Onde é utilizado: circuito de tratamento do *watchdog*.

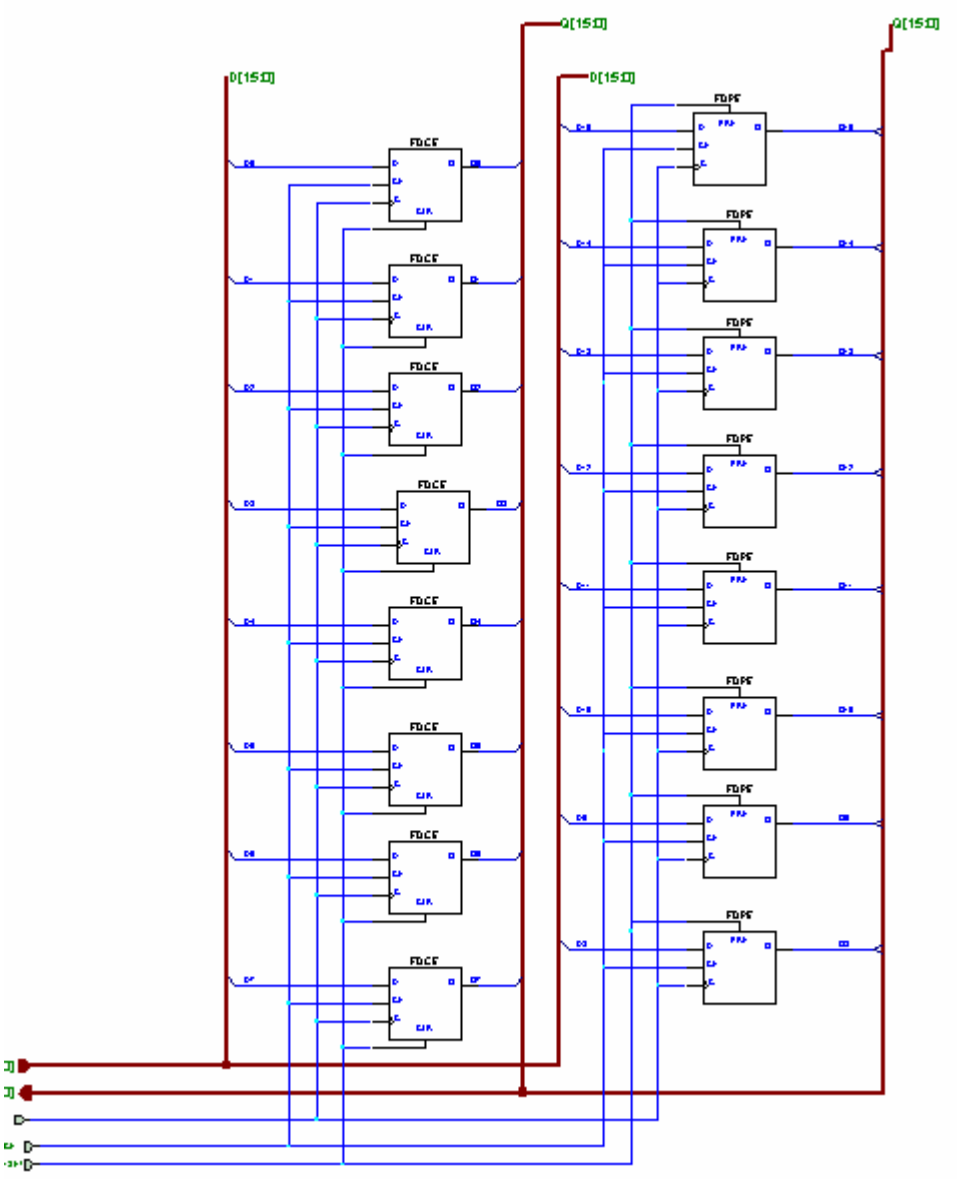

Figura 75 – Registrador de 16 bits com *clear*, *preset* e *clock enable*

ANEXO G – Utilizando as Ferramentas *Project Manager* e *GXSLOAD*

Segue um guia rápido para utilização das ferramentas *Project Manager* e *GSXLOAD* para um projeto exemplo simples.

1) *Project Manager*

a) Criando um novo projeto de diagrama esquemático (*File / New Project*):

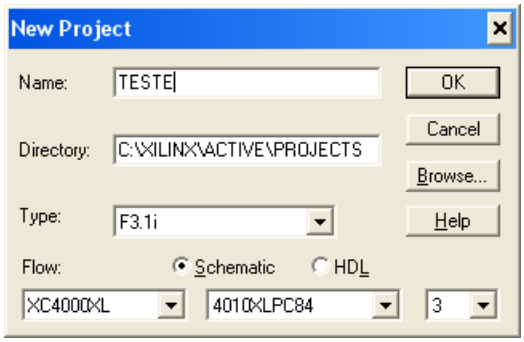

Figura 76 – Criando um novo projeto

b) Entrando no editor de diagrama esquemático (*Schematic Editor*):

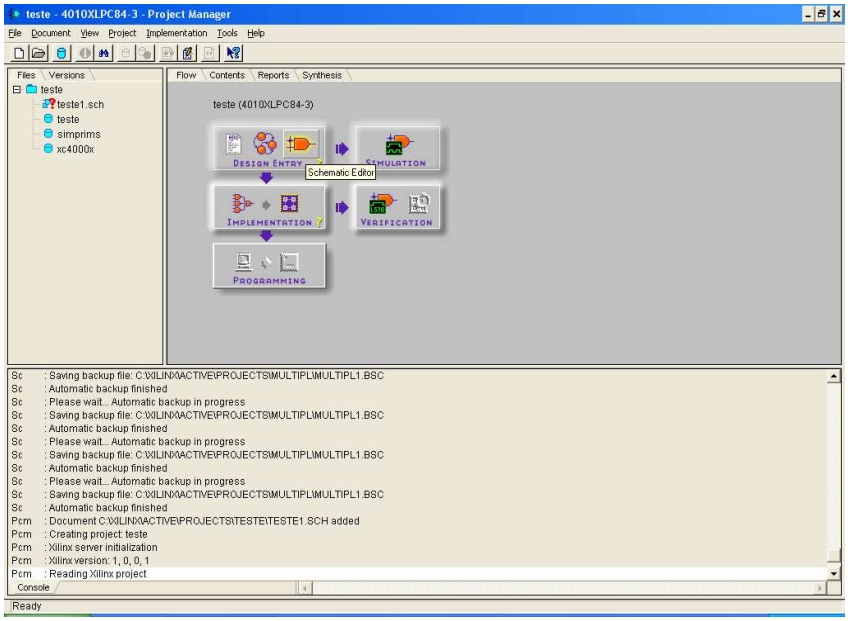

Figura 77 – Entrando no *Schematic Editor*

c) Adicionando componentes no diagrama (*Symbols toolbox*):

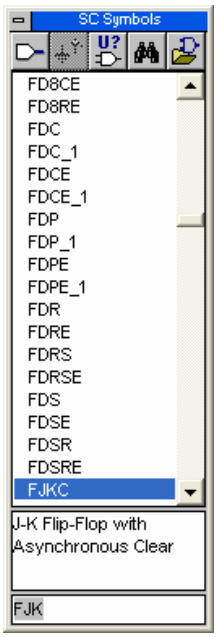

Figura 78 – Caixa de componentes (*Symbols toolbox*)

d) Ligando componentes no diagrama (*Draw wires* + *Draw buses*):

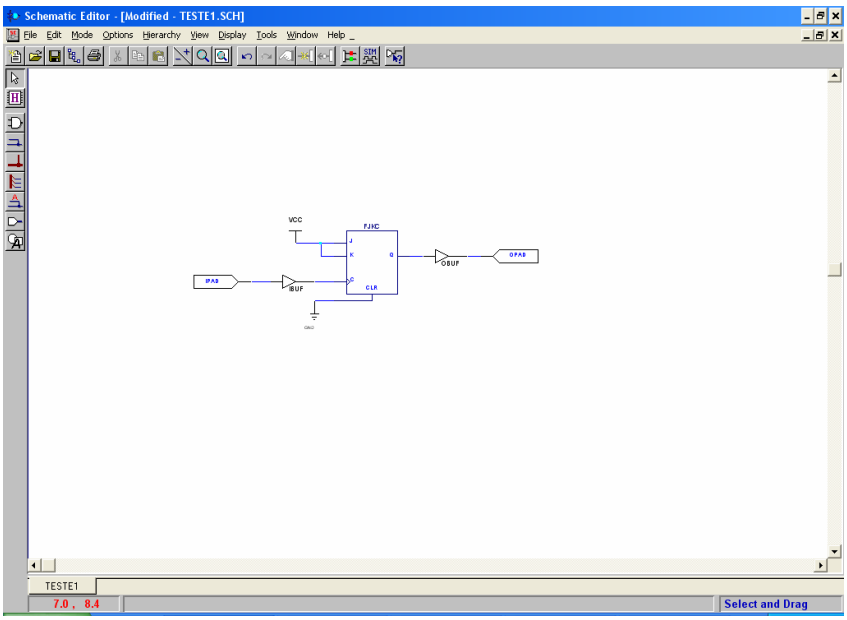

Figura 79 – Diagrama esquemático simples

e) Gerando o *netlist* (*Options / Create netlist*):

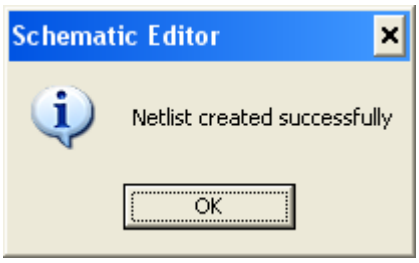

Figura 80 – Criação do *netlist*

f) Exportando o *netlist* (*Options / Export netlist*):

O Project Manager automaticamente salva o netlist no caminho *Xilinx\active\projects\teste* dentro de seu diretório de instalação. Deve ser escolhida a extensão *.EDN* para salvamento.

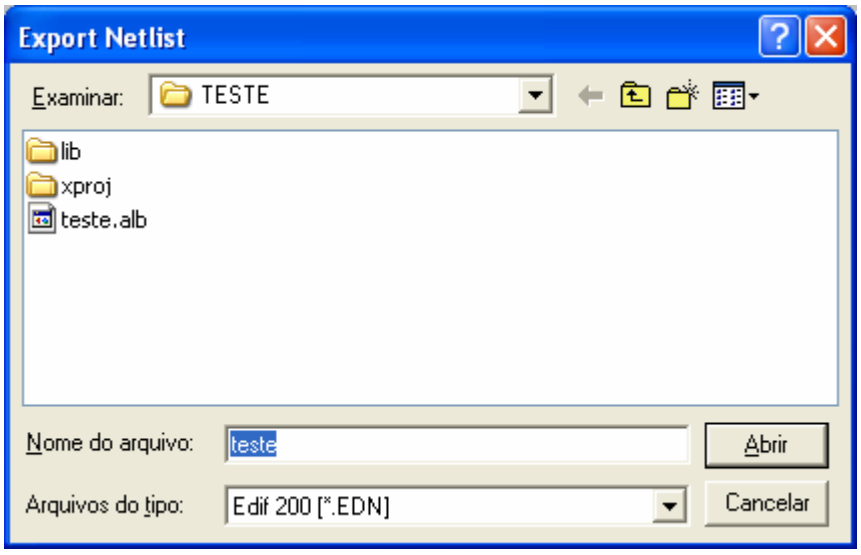

Figura 81 – Exportação do *netlist*

g) Adicionando pontas de prova para simulação (*Simulation toolbox / Probe tool*):

A caixa de ferramentas *SC Probes* permite adicionar pontas de prova (*Probe Tool*) para simulação e observação do comportamento de sinais específicos.

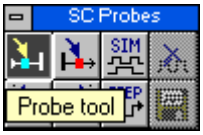

Figura 82 – Pontas de prova para simulação (*Simulation toolbox*)

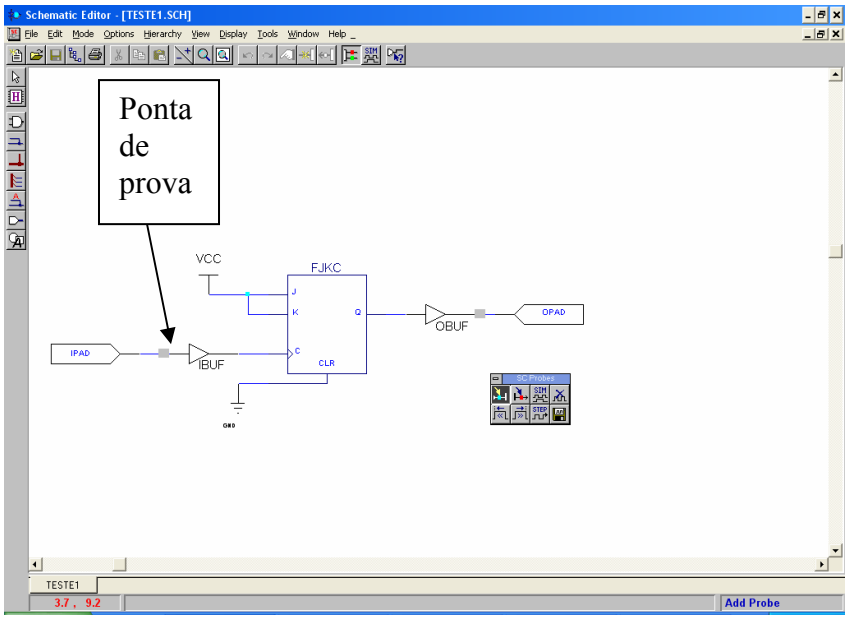

Figura 83 – Adicionando as pontas de prova

h) Abrindo o simulador (*Simulator*):

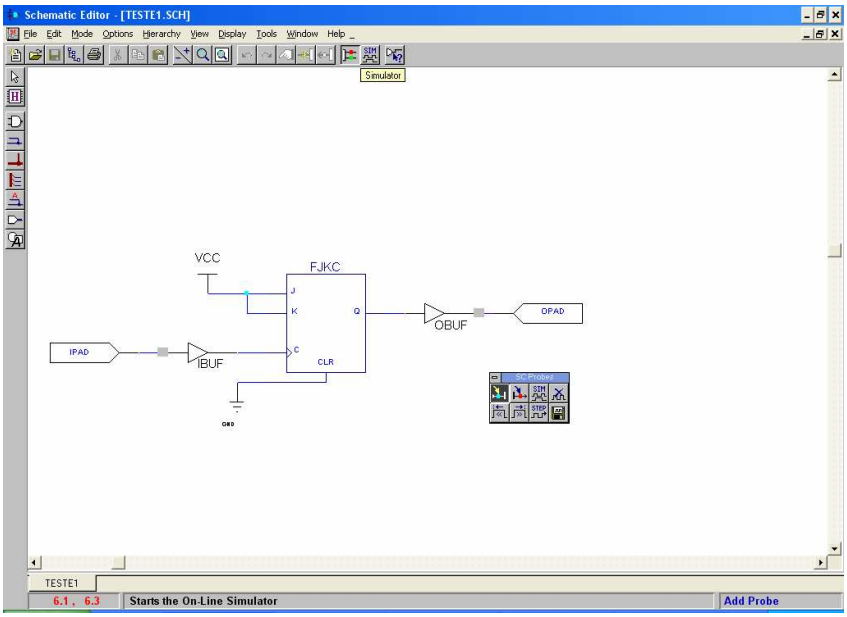

Figura 84 – Abrindo o simulador

i) Simulando o projeto (*Select stimulators + Logical States + Power On + Simulation Step*):

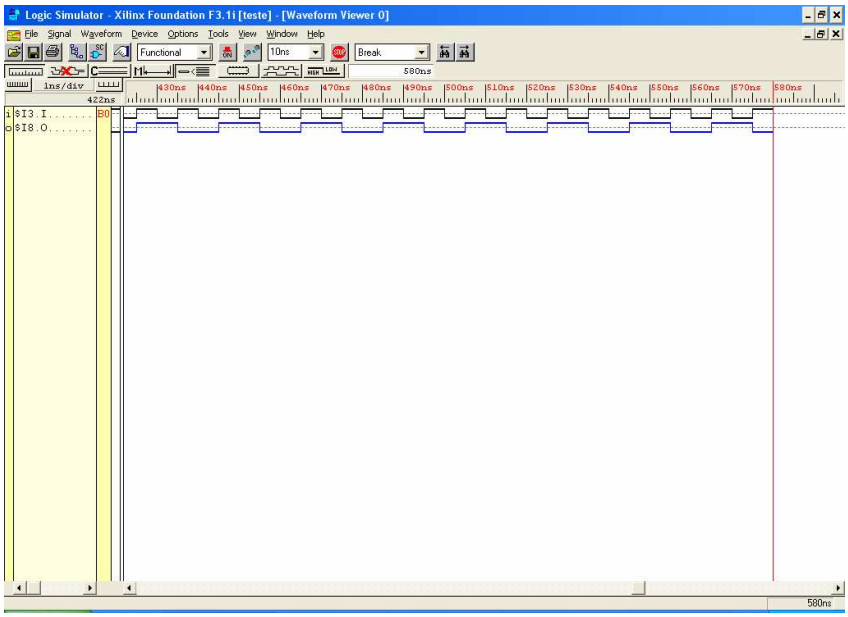

Figura 85 – Simulando o projeto

j) Implementando o projeto para geração do arquivo *.bit* (*Implementation*):

O botão *Implementation* permite a entrada na função de implementação, que terá como produto final o arquivo *.bit* para *download* no FPGA.

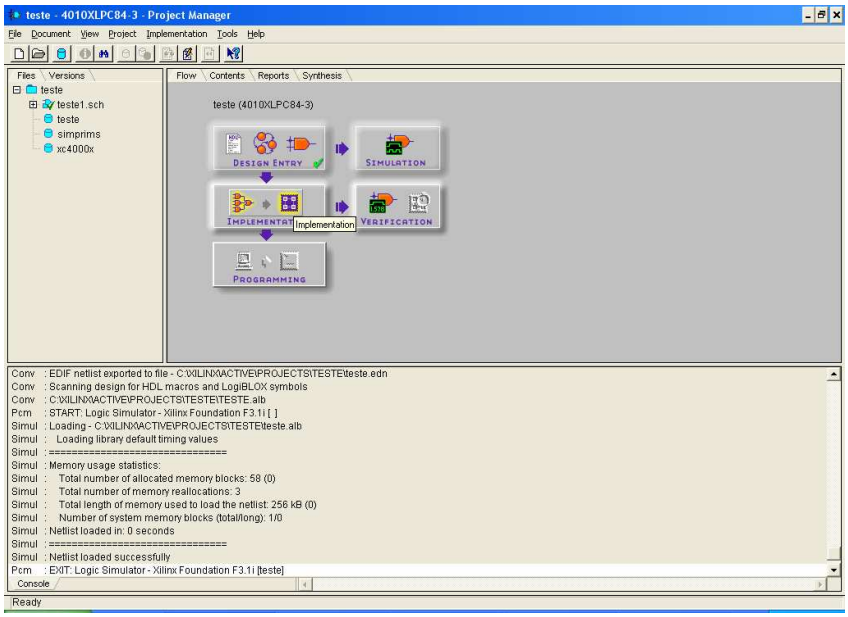

Figura 86 – Entrando na função de implementação

k) Ativando a implementação (*Implement design / Run*):

Após a entrada na função de implementação (*Implement Design*), a mesma é ativada clicando no botão *Run*. Observar que é mostrado o tipo de dispositivo FPGA escolhido inicialmente para o projeto, bem como os nomes da versão e revisão.

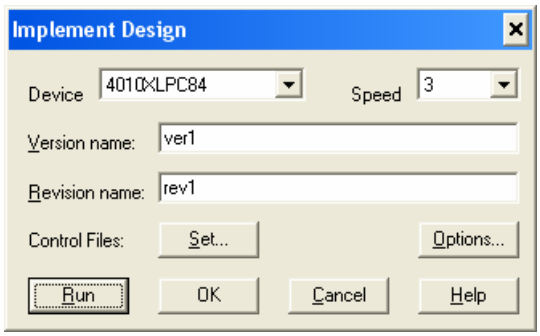

Figura 87 – Ativando a implementação

l) Passos do processo de implementação (tradução, mapeamento, roteamento, temporização e configuração):

Após a ativação, os passos do processo de implementação seguem na seqüência indicada pelas setas.

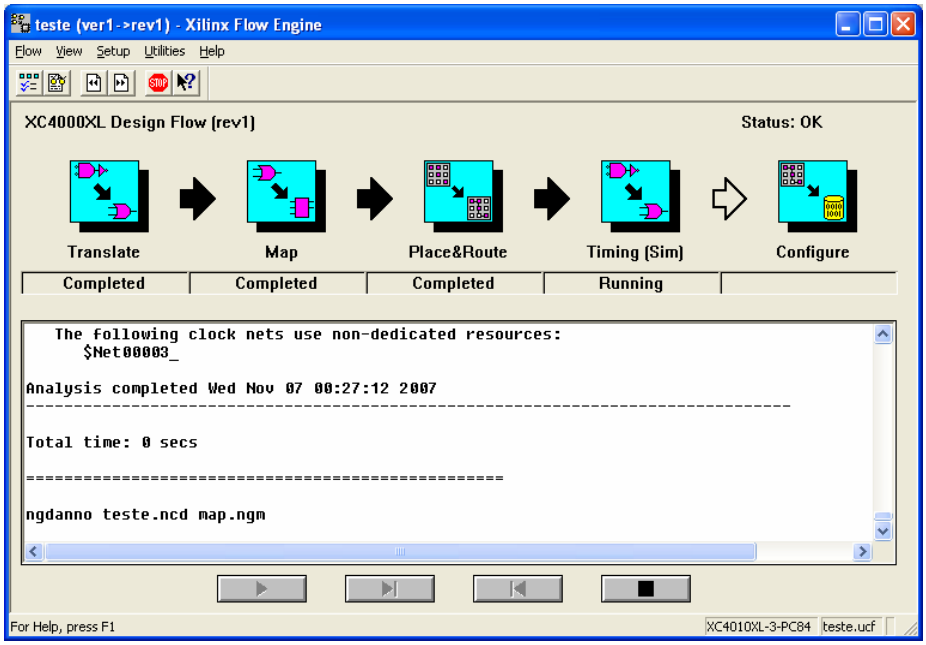

Figura 88 – Passos do processo de implementação

m) Conclusão do processo de implementação:

Ao término do processo de implementação, é exibida mensagem na tela. Caso ocorram erros, será exibida mensagem de alerta.

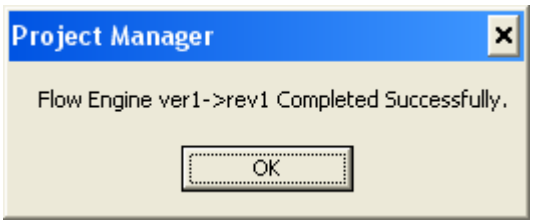

Figura 89 – Fim do processo de implementação

n) Verificando os relatórios de cada fase do processo de implementação (*Reports / Implementation Report Files*):

Os relatórios gerados por cada fase da implementação podem ser visualizados para verificar estatísticas de ocupação do FPGA, temporização e eventuais erros ocorridos.

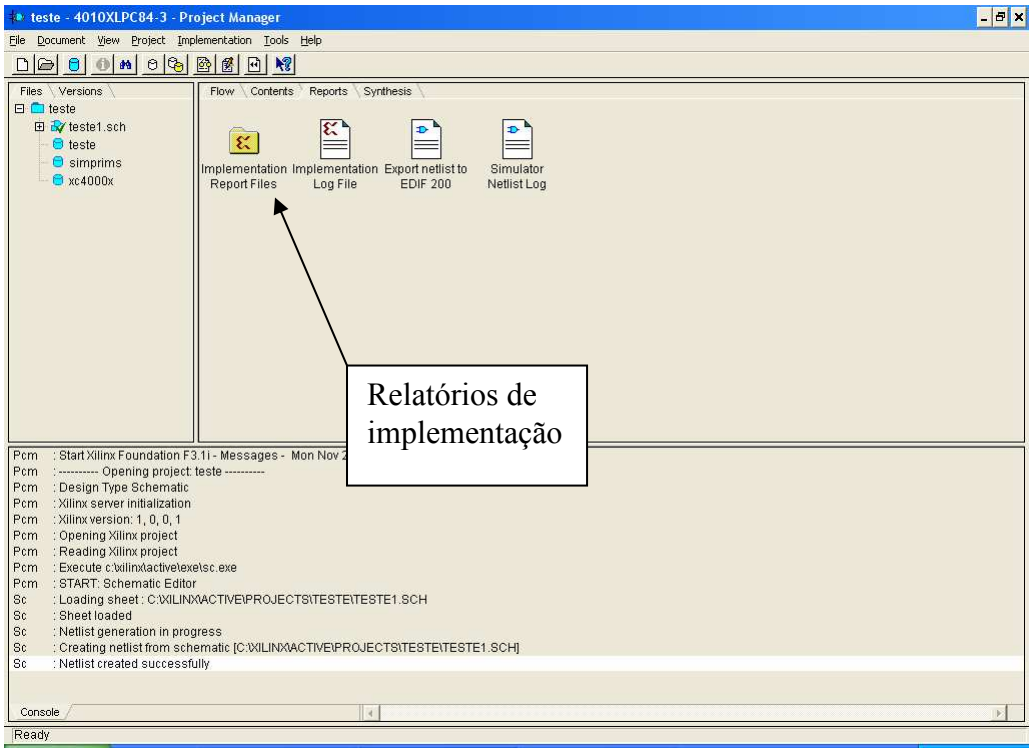

Figura 90 – Buscando os relatórios do processo de implementação

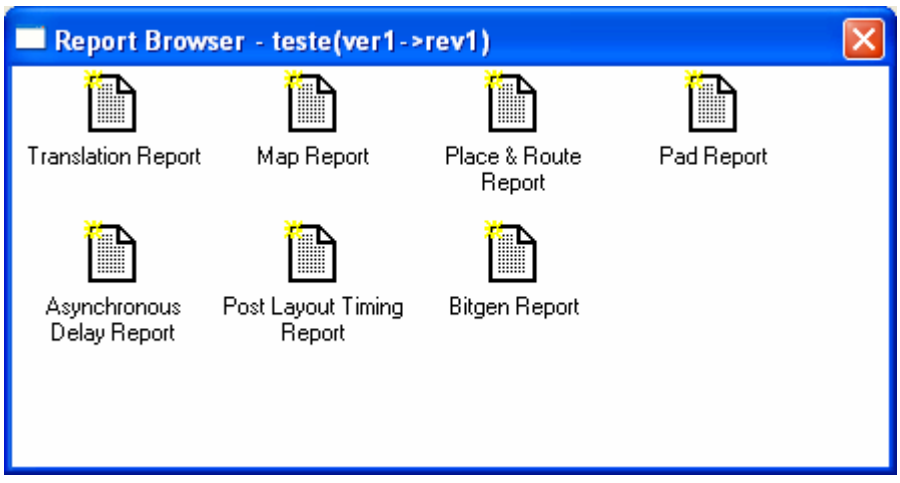

Figura 91 – Relatórios gerados pelo processo de implementação

2) *GXSLOAD*

Para descarregar o arquivo *.bit* no FPGA da placa de desenvolvimento XS40, basta buscar e arrastar para a área "FPGA/CPLD" de uma janela aberta da ferramenta *GXSLOAD*, o arquivo *.bit* que o Project Manager automaticamente salva no caminho *Xilinx\active\projects\teste\xproj\ver1\rev1* dentro de seu diretório de instalação.

Observar que "teste" é o nome do projeto exemplo.

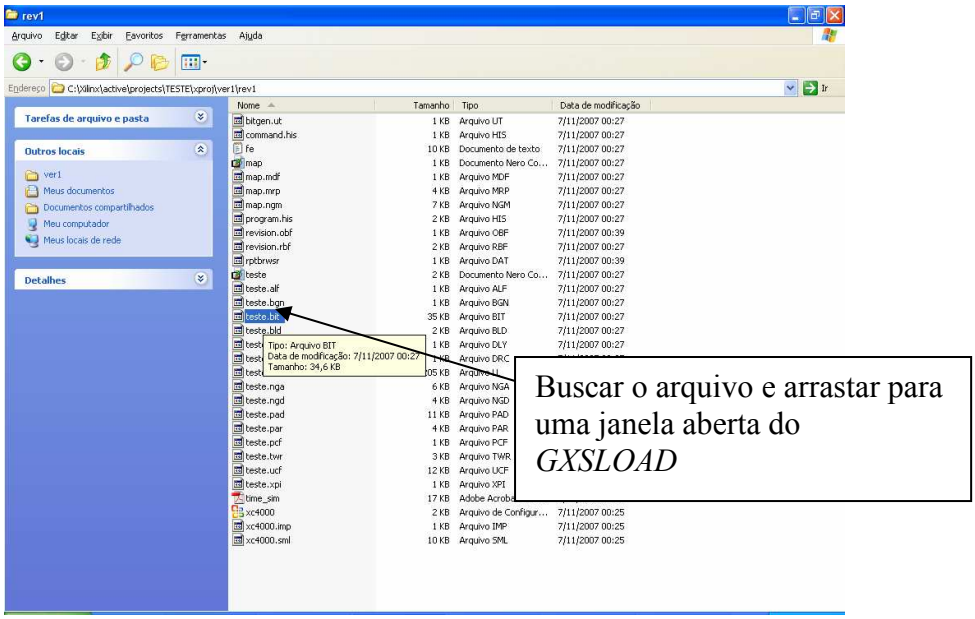

Figura 92 – Buscando o arquivo *.bit*

Na janela do *GSXLOAD* deve ser escolhido o tipo de placa de desenvolvimento e o nome da porta paralela a ser utilizada para descarregar o arquivo *.bit* no FPGA. Feito isso, basta clicar no botão *LOAD* para descarregar.

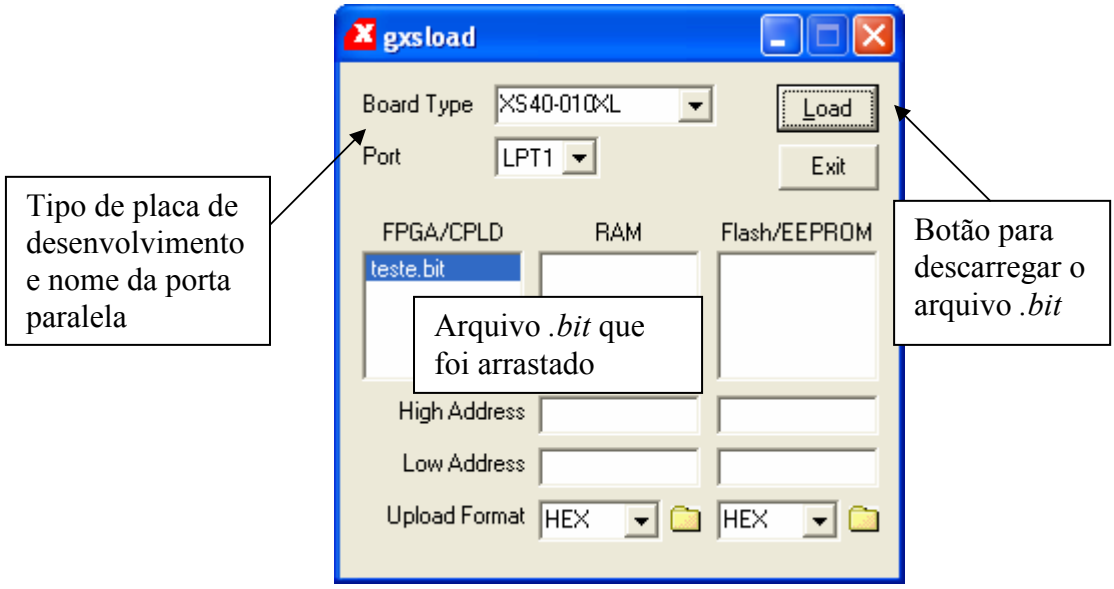

Figura 93 – Janela do *GXSLOAD* para descarregar o arquivo *.bit*

Após receber o arquivo *.bit*, a placa de desenvolvimento está pronta para os testes práticos.