UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO DE CIÊNCIAS TECNOLÓGICAS
DEPARTAMENTO DE TELECOMUNICAÇÕES

TRABALHO DE CONCLUSÃO DO CURSO

RAFAEL MARTELLI

BLUMENAU
2006

RAFAEL MARTELLI

SISTEMA DE COMUNICAÇÃO COM GPS VIA DISPOSITIVO LÓGICO PROGRAMÁVEL

Trabalho de Conclusão do Curso submetido à Universidade Regional de Blumenau para a obtenção de créditos na disciplina de Trabalho de Conclusão de Curso do curso de Engenharia de Telecomunicações.

Prof. Orientador: Dr. Ricardo José de Oliveira Carvalho

BLUMENAU

2006

SISTEMA DE COMUNICAÇÃO COM GPS VIA DISPOSITIVO LÓGICO PROGRAMÁVEL

Por

RAFAEL MARTELLI

Este trabalho de conclusão de curso foi julgado adequado para a obtenção dos créditos na disciplina de trabalho de conclusão de curso obrigatória para a obtenção do titulo de Engenheiro de Telecomunicações, pela banca examinadora formada por:

Orientador _____
Prof. Ricardo José de Oliveira Carvalho – Orientador, FURB

Membro: _____
Prof. Fabio Luis Peres – Coordenador do TCC

Membro: _____
Prof.

Membro: _____
Prof.

Blumenau, dia Mês de 2006.

AGRADECIMENTOS

Aos professores que tornaram possível a elaboração deste trabalho, em especial ao Prof. Ricardo José de Oliveira Carvalho que gentilmente aceitou em ser meu orientador, colaborando arduamente na realização deste trabalho, como a orientação do tema, indicações sobre a bibliografia e conselhos inteligentes e oportunos nas horas mais difíceis.

Ao Departamento de Engenharia Elétrica e Telecomunicações, que disponibilizou o material necessário para a elaboração deste trabalho e também ao laboratório de qualidade de energia que demandou tal projeto.

Ao professor desta matéria que forneceu o material para consulta e modelos para a apresentação deste relatório, que segue os rígidos padrões da ABNT.

A toda a minha família que sempre colaborou com minha formação desde as primeiras matérias do curso, em especial à minha esposa que sempre esteve ao meu lado e soube dar-me palavras de conforto nas horas em que mais precisei.

E a Deus, por sempre estar ao meu lado, dando-me luz e esperança.

RESUMO

O uso de dispositivos lógicos programáveis, tornam-se cada vez mais necessários, especialmente em áreas de desenvolvimento de circuitos lógicos, pois são capazes de ser programados e reprogramados com facilidade e áreas de desenvolvimento de novos equipamentos, como os equipamentos registradores de eventos do Laboratório de Qualidade Energia da FURB, possibilitando formas de comunicações entre dispositivos que até então não eram possíveis.

O presente trabalho traz um sistema de comunicação entre dois dispositivos com protocolos diferentes de comunicação e o CPLD da ALTERA está encarregado de promover está interface entre um dispositivo GPS e um display LCD, utilizando da linguagem de programação de hardware VHDL, juntamente com o software de desenvolvimento QuartusII, também de propriedade da ALTERA Corporations.

Palavras-chave: CPLD, QuartusII, VHDL, GPS, Display LCD, Protocolo NMEA.

ABSTRACT

The use of programmable logical devices, they become more and more necessary, especially in areas of development of logical circuits, because they are capable to be programmed and reprogrammed with easiness and areas of development of new equipments, as the recording equipments of events of the Laboratório de Qualidade Energia of the FURB, making possible forms of communications among devices that until then were not possible.

The present work brings a communication system among two devices with protocols different from communication and CPLD of the ALTERA it is entrusted of promoting it is interface among a device GPS and a Display LCD, using of the hardware programming language VHDL, together with the development software QuartusII, also of property of the ALTERA Corporations.

Key-words: CPLD, QuartusII, VHDL, GPS, Display LCD, Protocolo NMEA.

SUMÁRIO

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ASCII	American Standard Code for Information Interchange
ASIC	Application Specific Integrated Circuit
CPLD	Complex Programmable Logic Device
DGPS	Differential GPS
DSP	Digital Signal Processing
EAB	Embedded Array Blocks
EEPROM	Erasable Programmable Read Only Memory
EPLD	Erasable PLD
FIFO	First-In First-Out
FPGA	Field Programmable Gate Array
GPS	Global Position System
IEEE	Institute of Electrical and Electronic Engineers
JTAG	Joint Test Action Group
LUT	Look-Up Table
NMEA	National Marine Electronics Association
PAL	Programmable array logic
PLA	Programmable Logic Array
PLCC	Plastic Leaded Chip Carrier
PLD	Programmable Logic Device
PROM	Programmable Read Only Memory
RAM	Random Access Memory
ROM	Read Only Memory
RQFP	R Quad Flat Pack
SPLD	Simple PLD
SPS	Standard Positioning Service
SRAM	Static Random Access Memory
TDI	Test Data In
TDO	Test Data Out
UP2	University Program Education Kit Version 2
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit

1 INTRODUÇÃO

Com o surgimento de novas tecnologias, em especial a tecnologia de posicionamento global via satélite, *Global Position System* (GPS) que desempenha especial importância em varias áreas do conhecimento, entre elas a localização de veículos, navegação, equipamentos registradores de eventos entre outras aplicações.

Este trabalho apresenta uma comunicação entre equipamentos com distintos protocolos de comunicação, como o protocolo NMEA[1] utilizado pelo modulo GPS. Ocupar-se-á de enviar os dados do GPS ao dispositivo lógico programável (PLD) e posteriormente enviando os dados ao display LCD[2].

O principal objetivo é disponibilizar uma base precisa de tempo, que pode ser usada para sincronizar os relógios internos utilizados nos equipamentos registradores de eventos da rede elétrica, proposto pelo Laboratório de Qualidade de Energia (LQE).

O trabalho será desenvolvido utilizando basicamente três módulos de hardware distintos e também dispositivos de software. Para uma melhor visualização do trabalho realizado, na figura 1 se apresenta um diagrama de blocos esquemático da comunicação entre os dispositivos.

Figura 1: Esquema geral do trabalho

1.1 JUSTIFICATIVA

Atualmente os equipamentos utilizados para registro de eventos do sistema elétrico, assim como os detectores de faltas em linhas de transmissão e distribuição, necessitam de uma referência de tempo baseada no sinal do GPS, neste caso se utilizará à hora UTC[3], comum nestes dispositivos.

---

[1] National Marine Electronics Association
[2] Liquid Cristal Display
[3] Universal Time Coordinated

O LQE da FURB, tem a necessidade de desenvolver uma comunicação entre um relé de proteção digital, previamente desenvolvido no LQE, e o receptor GPS. Este trabalho pretende aprofundar os conhecimentos na área de comunicação e atender à demanda do LQE.

Os dispositivos lógicos programáveis são capazes de sintetizar um sistema comunicação de uma forma otimizada, adaptando-se a qualquer protocolo de comunicação.

1.1.1 Definição do problema

O presente trabalho visa sintetizar um sistema de comunicação utilizando dispositivo lógico programável, utilizando o software QUARTUS II da ALTERA, para a programação e simulação do mesmo. Com isto se desenvolve uma interface de comunicação entre os módulos de hardware.

Esta comunicação possibilitará a aquisição dos sinais do sistema GPS, neste caso somente será relevante à hora UTC, uma referencia de tempo precisa e absoluta para vários equipamentos do sistema de aquisição dados remotos, que utilizam relógios internos, onde é crucial o sincronismo de tempo.

1.2 OBJETIVOS

1.2.1 Objetivo geral

Sintetizar um sistema de comunicação com o modulo receptor de GPS utilizando o dispositivo lógico programável, fazendo uso do software de desenvolvimento e possibilitando a visualização dos dados através de display LCD.

1.2.2 Objetivos específicos

1.2.2.1 Objetivos com relação ao modulo GPS

Viabilizar a aquisição dos sinais disponibilizados pelo sistema do modulo receptor de sinais GPS;

Comunicar o sistema de GPS com um relé de proteção digital através de uma interface apropriada para ambos os equipamentos;

Conhecer detalhadamente o protocolo utilizado pelo módulo GPS na comunicação com outros dispositivos digitais.

1.2.2.2 Objetivos com relação ao dispositivo lógico programável

Sintetizar o sistema de comunicação entre o modulo GPS e o PLD, para com isso possibilitar a comunicação entre os dois componentes;

Estudar e investigar o sistema de comunicação que melhor se adapta ao objetivo do projeto.

Desenvolver o programa que viabilizará a comunicação entre os equipamentos.

1.3 ESTRUTURA

O trabalho está estruturado em seis seções primarias. A primeira seção traz uma breve introdução do trabalho, assim como objetivos gerais e específicos uma descrição de como será feito o trabalho e seus componentes.

A fundamentação teórica está dividida em dispositivos de hardware e dispositivos de software, nas seções 2 e 3 respectivamente. As seções seguintes estão destinadas a explicar o desenvolvimento do projeto no software QuartusII e consecutivamente apresenta-se os resultados e as conclusões do trabalho.

A seção primaria 2 apresenta uma descrição completa de todos os dispositivos de hardware em que se foca o trabalho, estas informações são importantes para o total entendimento do trabalho.

Na seção 3 faz-se uma descrição detalhada dos dispositivos de software utilizados para a confecção do Trabalho de Conclusão do Curso (TCC), estes são de fundamental importância para o entendimento do trabalho realizado, pois o projeto está baseado nestes dispositivos.

A descrição do desenvolvimento do trabalho é feita na seção 4 juntamente com as complementações especificadas na seção de apêndices deste trabalho.

Os testes e simulações que foram necessários para a total realização do TCC estão descritas na seção 5 que acompanha os gráficos da tela do simulador utilizando o software de desenvolvimento QuartusII.

Nas seções finais deste trabalho estão as conclusões e apêndices do TCC. A seção de apêndice tem uma importância marcante, pois descreve os códigos fonte que foram criados utilizando a linguagem de descrição de hardware (VHDL) e como forma de complementar o entendimento destes códigos, os acompanha o fluxograma de cada um deles.

2 DISPOSITIVOS DE HARDWARE

Os seguintes dispositivos de hardware utilizados neste trabalho são: kit CPLD[4] ALTERA UP2; modulo GPS com antena; modulo LCD e um computador para as simulações.

2.1 INTRODUÇÃO AO CPLD

Previamente ao surgimento da tecnologia CPLD, havia o dispositivo lógico programável que era utilizado por projetistas para a realização tarefas elementares, este formava o grupo do *Simple PLD* (SPLD), e posteriormente com os avanços da tecnologia surgiu o CPLD.

2.1.1 Dispositivo lógico programável (PLD)

Os PLD's surgiram de uma necessidade eminente de muitos projetistas e engenheiros. Ao desenharem os diagramas lógicos, estes ao mesmo tempo, também necessitavam seus desenhos já prontos, ou implementados para a realização de testes e simulações, para encontrar futuras falhas em seus projetos.

A flexibilidade de alteração assegura a utilização destes componentes na prototipagem de projetos, pois podem ser reprogramados quantas vezes forem necessárias, garantindo assim, que os ciclos de projeto sejam muito curtos e com baixos custos. (RHOD, 2006)

2.1.1.1 A Evolução dos Dispositivos Lógicos Programáveis

"A memoria PROM[5] foi o primeiro tipo de chip programável pelo usuário, que podia implementar circuitos lógicos. As linhas de endereço eram utilizadas como entradas do circuito lógico, e as linhas de dados como saídas desses circuitos." (RHOD, 2006)

As funções lógicas, raramente requerem mais do que alguns termos de produto, e uma PROM resulta em uma arquitetura ineficiente para o projeto e realização de circuitos lógicos, e raramente são utilizadas para esse fim.

---

[4] Complex Programmable Logic Divice
[5] Programmable Read-Only Memory (memória programável de somente leitura.)

Cada saída do plano *Or* pode ser configurada para produzir uma soma lógica de quaisquer saídas do plano *AND*. A Figura 2 mostra o esquema simplificado de um PLA.

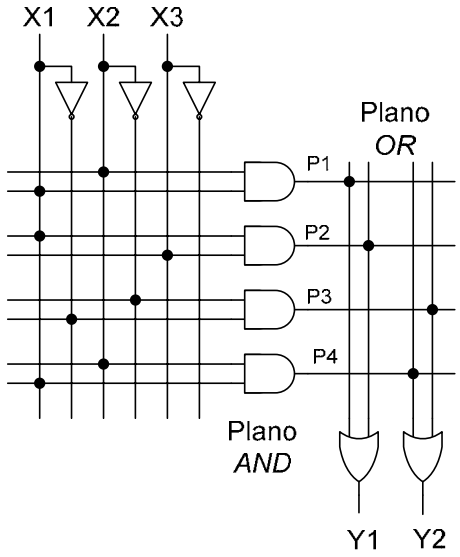


Figura 2: Esquema simplifcado de um PLA

Fonte: RHOD, 2006

> Os PLAs[6] foram os primeiros dispositivos criados especificamente para a implementação de circuitos lógicos. Introduzidos pela Philips no inicio dos anos 70. Estes dispositivos consistem de dois níveis de portas lógicas: um plano de portas wired - AND seguido por um plano de portas wired - OR, ambos programáveis. Um PLA é estruturado de tal forma que cada saida do plano AND pode corresponder a qualquer termo produto das entradas. (RHOD, 2006)

São adequados para as implementações de funções lógicas na forma de soma de produtos, se apresentam muito versáteis, tanto nos termos *AND* como nos termos *OR* podem ter muitas entradas.

> Porem, essa tecnologia também apresenta alguns problemas como alto custo de fabricação e baixo desempenho em termos de velocidade. Essas desvantagens existem devido aos dois níveis de lógica reconfiguravel. Os planos lógicos programáveis são difíceis de serem fabricados e introduzem atrasos significativos de propagação dos sinais elétricos. (RHOD, 2006)

A tecnologia PAL[7] foi então desenvolvida para superar essas deficiências. Os PALs possuem um único nível de programação, um custo mais baixo e um melhor desempenho. A Figura 3 apresenta um esquema simplificado de um PAL.

Todos estes PLDs (PALs, PLAs e similares) são agrupados em uma única categoria denominada SPLD, onde as principais características são: o baixo custo e o alto desempenho. A dificuldade em aumentar a capacidade da arquitetura PLD reside no fato de que a estrutura

---

[6] Programmables Logic Arrays
[7] Programmable array logic.

de planos lógicos programáveis aumenta muito rapidamente com o aumento do numero de entradas. Uma maneira viável de produzir dispositivos com maior capacidade seria então, integrar múltiplos SPLDs em um único chip, e prover interconexões programáveis para conectar os diversos PLDs. A Figura 4 apresenta esta estrutura. (RHOD, 2006)

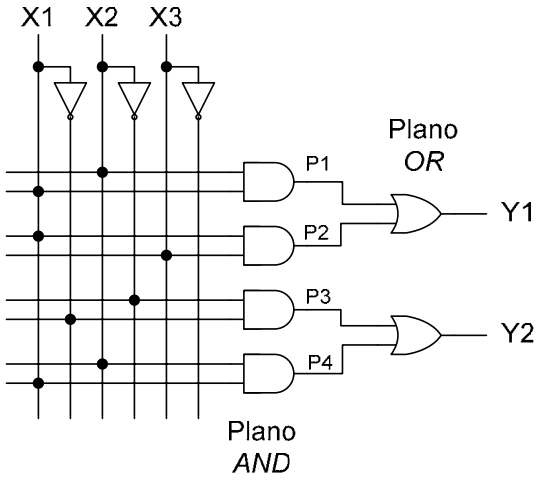


Figura 3: Esquema simplifcado de um PAL

Fonte: RHOD, 2006

2.1.2 Como Surgiram os CPLDs

Atualmente existem muitos produtos comerciais utilizando-se do esquema de integrar múltiplos SPLDs em um único chip, e prover interconexões programáveis para conectar os diversos PLDs. São os chamados CPLDs[8].

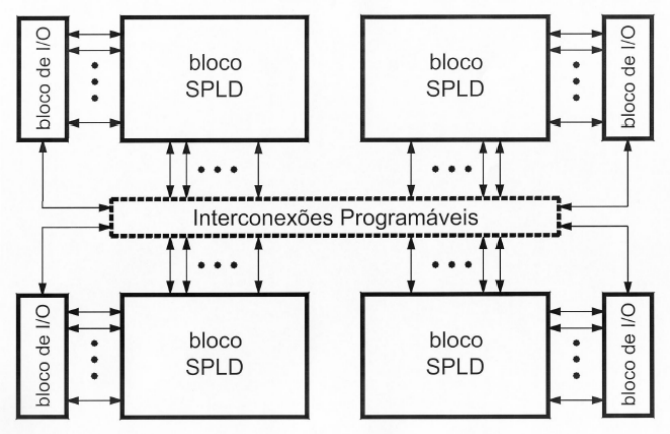


Figura 4: Estrutura simplificada de um CPLD

Fonte: RHOD, 2006

Os CPLDs foram introduzidos pela Altera Corp., inicialmente com sua família de componentes chamada Classic EPLDs (Erasable PLDs) e em seguida com três series

[8] Complex Programmable Logic Devices

adicionais, chamadas MAX 5000, MAX 7000 e MAX 9000. Devido ao rápido crescimento do mercado para grandes PLDs, outros fabricantes desenvolveram dispositivos na categoria CPLD e há atualmente muitas opções disponíveis.

"Cada tipo de PLD apresenta vantagens que os tornam mais adequados para algumas aplicações do que outros. Cabe ao projetista entender qual a sua melhor utilização, escolher um fabricante especifico, aprender a utilizar as ferramentas, para só então começar a projetar o hardware." (RHOD, 2006)

2.1.3 Procedimento na programação de CPLDs

Todo o processo de programação se visualiza melhor em um diagrama de fluxo, como na figura 5, onde de forma esquemática se observa o processo da programação.

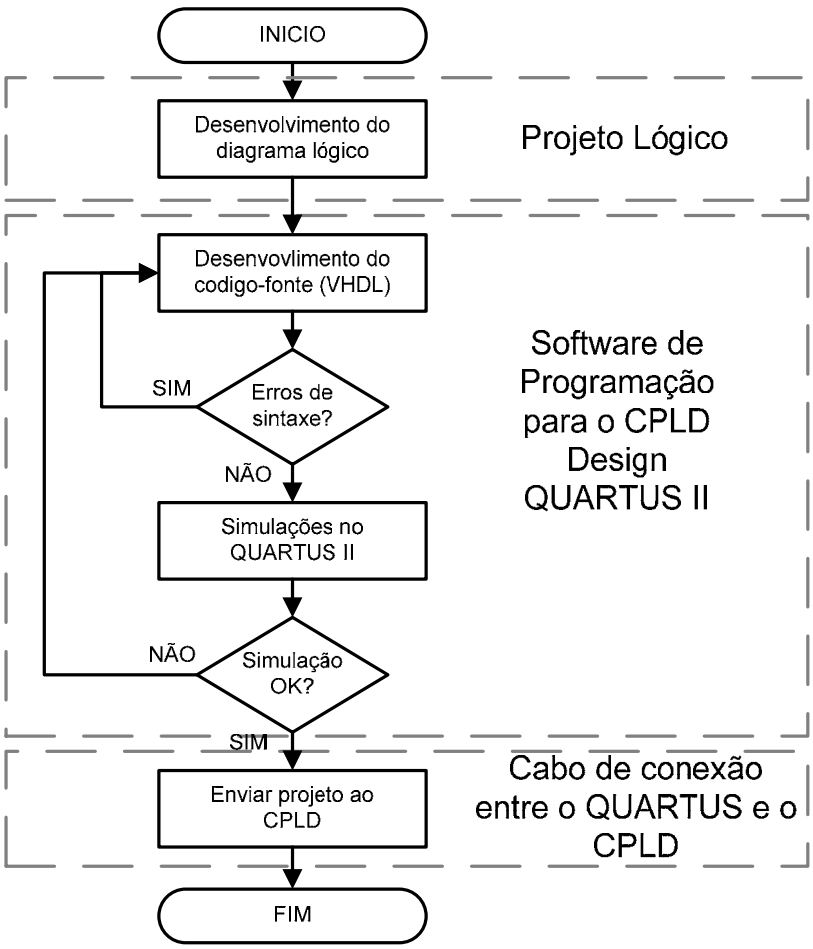


Figura 5: Etapas de programação de um CPLD

Inicia-se pelo projeto lógico, que rege o sistema a ser implementado. Após o projeto lógico, parte-se para a programação do PLD utilizando um software de programação, por exemplo, o QUARTUS II e a linguagem de programação de hardware (VHDL)[9]. Por fim termina-se a programação do PLD com a gravação do mesmo, utilizando o mesmo software e conectando o cabo de comunicações que acompanha o CPLD.

2.2 KIT DE DESENVOLVIMENTO DA ALTERA

O kit de desenvolvimento da ALTERA encontra-se disponível no laboratório de qualidade de energia, desta universidade. Com ele é possível implementar os projetos que primeiramente foram simulados e depurados no software QUARTUS II.

> O Equipamento de Educação em Programação Universitária UP2 "University Program UP2 Education Kit" foi projetado para satisfazer as necessidades das universidades que ensinam o design da lógica digital com ferramentas de desenvolvimento de dispositivos de lógica programáveis (PLDs). O kit provê todas as ferramentas necessárias para criar e implementar projetos em lógica digital (ALTERA, 2006)

Entre as suas características se destacam as seguintes:

- Quartus® II Web – edição de desenvolvimento de software;
- *Protoboard* [10] de Educação UP2;
- Um dispositivo EPF10K70 com 240 pinos e encapsulado RQFP;
- Um dispositivo EPM7128S com 84 pinos e encapsulado PLCC;
- Cabo ByteBlasterTM II para download de dados utilizando a porta paralela;

2.2.1 UP2 Education Board

O University Program Education Kit (UP2), é uma placa ou também chamada *board*, para experiências e já vem pronta para o uso, está baseada em um dispositivo FLEX® 10K e num dispositivo MAX® 7000. Quando usado com o software QuartusII, este *board* provê uma plataforma superior para aprender o desenho da lógica digital e usa ferramentas de desenvolvimento de padrão industrial e PLDs. A figura 6 mostra o kit UP2 com seus componentes já implementados.

---

[9] VHSIC Hardware Description Language

[10] Refere-se a que o kit já vem implementado com displays, *pushbottons*, LEDs e outros componentes, em uma placa, como se observa na figura 6.

O kit é projetado para satisfazer as necessidades de instrutores e estudantes em um ambiente de laboratório. O dispositivo EPF10K70 pode ser configurado em sistema[11] ou com o cabo de *download ByteBlaster II* ou com a configuração dispositivo EPC1. O dispositivo de EPM7128S pode ser programado em sistema com o cabo *ByteBlaster II*.

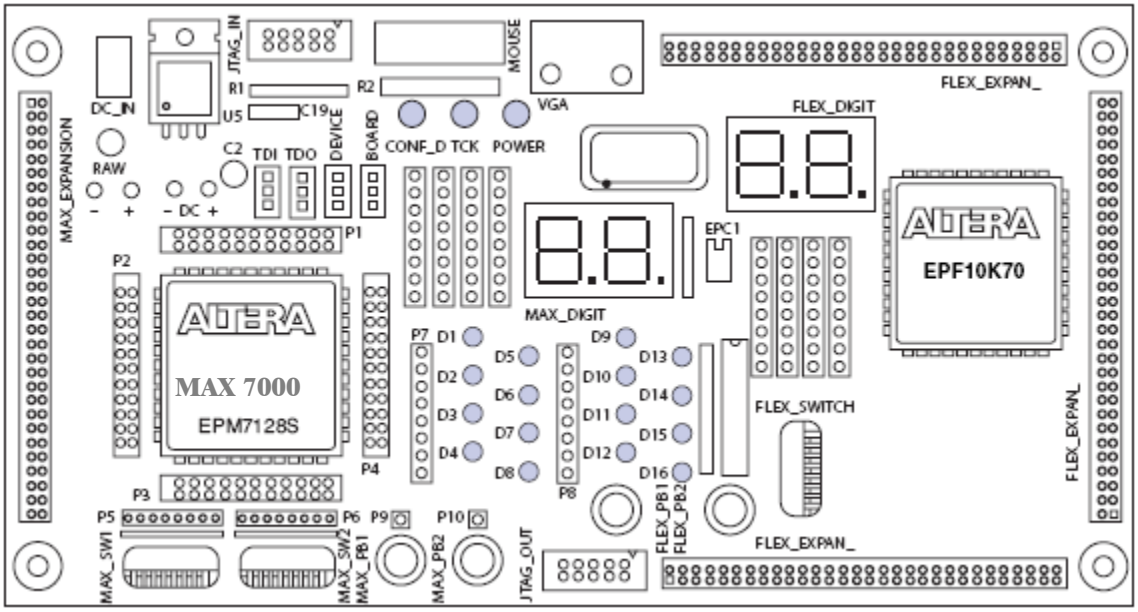


Figura 6: Diagrama do UP2 Education Board
Fonte: ALTERA, 2006

2.2.2 Dispositivo EPF10K70

O dispositivo de EPF10K70 está baseado na tecnologia SRAM[12]. Está disponível em um encapsulado RQFP de 240 pinos e têm 3,744 elementos lógicos (LEs) e 9 *embedded array blocks* (EABs). Cada LE consiste em quatro entradas LUT, um flipflop programável, e caminhos dedicados para funções de *carry-and-cascade*. Cada EAB provê 2,048 localidades de memória que pode ser usada para criar RAM, ROM, ou funções *first-in first-out* (FIFO). EABs também pode implementar funções lógicas, como multiplicadores, micro controladores, máquina-de-estados, e funções de *Digital Signal Processing* (DSP). Com 70,000 gates típicos, o dispositivo de EPF10K70 é ideal para intermediários, para cursos de desenhos digitais avançados, inclusive arquitetura de computador, comunicações, e aplicações de DSP.

---

[11] Referindo-se ao processo de programar diretamente com o dispositivo conectado a um PC com o Quartus II.
[12] Static Random Access Memory

2.2.3 Dispositivo EPM7128S

O dispositivo de EPM7128S, um membro da alta densidade, do alto-desempenho da família MAX 7000S, está baseado nos elementos (EEPROM)[13].

O dispositivo EPM7128S se caracteriza por um encapsulado plástico de 84 pinos (PLCC)[14] e tem 128 macro células. Cada macro células tem um (AND/fixed – OR) programável, bem como, um registro configurável com relógio de programação independentemente, *clock enable*, *clear*, e funções *preset*. Com uma capacidade de 2.500 *gates* e uma arquitetura simples, o dispositivo EPM7128S é ideal para desenhos introdutórios como também maiores combinações e funções de lógica seqüencial.

2.2.4 Oscilador

O UP2 Education Board contém um cristal oscilador 25,175MHz. A saída do oscilador dirige uma distribuição de relógio global no dispositivo EPM7128S, pino 83 e uma contribuição de relógio global no dispositivo FLEX10K, pino 91.

2.2.5 Jumpers

O *UP2 Education Board* tem quatro jumpers (TDI, TDO, DEVICE e BOARD) para fixar a configuração do conector JTAG[15], que pode ser fixada para uma variedade de configurações (programar só o dispositivo EPM7128S, configurar só o dispositivo FLEX10K, configurar e programar ambos os dispositivos, ou conectar múltiplos *UP2 Education Board*). Figura 7 mostra as posições dos três conectores (C1, C2, e C3) em cada dos quatro jumpers.

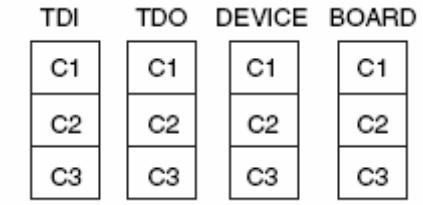


Figura 7: Diagrama dos Jumpers do dispositivo

Fonte: ALTERA, 2006

A tabela 1 apresenta a configuração para os pinos do jumper JTAG apresentados na figura 7, e os mesmos servem para as varias configurações apresentadas anteriormente.

---

[13] Erasable Programmable Read Only Memory
[14] Plastic J-Lead Chip Carrier
[15] Joint Test Action Group

Tabela 1 – Descrição dos pinos do conector JTAG

AÇÃO DESEJADA	*TDI*	*TDO*	*DEVICE*	*BOARD*
Programar somente o dispositivo EPM7128S	C1 & C2	C1 & C2	C1 & C2	C1 & C2
Configurar apenas o dispositivo FLEX10K	C2 & C3	C2 & C3	C1 & C2	C1 & C2
Programa e configura ambos dispositivos (1)	C2 & C3	C1 & C2	C2 & C3	C1 & C2
Conecta múltiplas UP2 juntas (2)	C2 & C3	ABERTO	C2 & C3	C2 & C3

Notas: (1) O primeiro dispositivo conectado no JTAG é o FLEX10K e o Segundo é o EPM7128S;

(2) Idem (1). O ultimo *board* no conector deve estar fixado para configuração simples (configure somente um dispositivo do *board*) e o ultimo *board* não deve ser conectado junto a outros *boards*.

Fonte: ALTERA, 2006.

2.2.6 Dispositivo FLEX 10K

O UP2 dispõe dos seguintes recursos para o dispositivo FLEX10K. Os pinos para este dispositivos já estão pré-assinados para os switches e LED's da placa *board*.

- Conector JTAG para o cabo ByteBlaster II.
- Socket para a configuração de um dispositivo EPC1.
- Dois switches *push button*[16] momentâneos.
- Um switch octal com encapsulado DIP
- Dois displays de sete segmentos.
- Oscilador *on-board* de 25.175 MHz.
- Porto VGA.
- Porto Mouse.
- Três portas de expansão, cada uma com 42 pinos I/O e 7 pinos globais.

2.2.7 FLEX_PB1 & FLEX_PB2

O FLEX_PB1 e o FLEX_PB2 são dois *push buttons* que provê um sinal de ativação em baixo, para dois pinos I/O de propósito geral sobre o dispositivo FLEX10K. O FLEX_PB1 se conecta ao pino 28, e o FLEX_PB2 ao pino 29. Cada *push button* está conectado internamente a um resistor de 10 KΩ.

[16] Este botão quando pressionado gera um sinal baixo (bit zero), em seu estado normal está em alto.

2.3 SISTEMA DE POSICIONAMENTO GLOBAL – GPS

O sistema de posicionamento global – *Global Positioning System* (GPS) – consiste em 24 satélites em órbita terrestre destinados a calcular a posição de qualquer lugar no mundo durante as 24 horas do dia e durante o ano inteiro.

Estes satélites são controlados pelo Departamento de Defesa dos Estados Unidos (DoD) e eles continuam administrando e controlando os satélites, para sistemas de navegação militares, o uso comercial deste sinal é limitado ao serviço SPS[17]. (SANJOSE, 2005)

2.3.1 Precisão

A maioria dos receptores comerciais de GPS, que são planejados para o uso geral terá uma precisão de 25 metros ou mais. O DoD introduziu a Disponibilidade Seletiva (SA) para manter uma vantagem militar. O SA é um erro randômico designado ao receptor de SPS. O tamanho destas mudanças de erros, raramente excede a 100 metros. A precisão do receptor GPS para usuários civis, afetado por SA, é suficiente para navegação geral. (SANJOSE, 2005)

2.3.2 Differential GPS (DGPS)

O *Differential* GPS (DGPS) computa o tamanho de um erro e aplica esta informação ao posicionar. Isto é normalmente feito por uma baliza de radiodifusão diferencial em terra. O receptor diferencial recebe e demodula o sinal, enviando então ao receptor do usuário, que aplica a correção à informação de posicionamento, corrigido posição e os dados de navegação. (SANJOSE, 2005)

2.3.3 Antena

Uma antena de GPS deve cobrir um ângulo de espaço amplio para receber o maior número de sinais. A exigência comum é receber sinais de todos os satélites a aproximadamente 5 graus sobre o horizonte. (BAO, TSUI, 2005)

A antena deve rejeitar ou minimizar efeito de *multipath* ou caminho múltiplo é o efeito sobre as reflexões no sinal do GPS, de alguns objetos que alcançam a antena indiretamente, podendo causar erro no cálculo de posição de usuário. (BAO, TSUI, 2005)

---

[17] Standard Positioning Service

2.3.4 Tempo de GPS e UTC

O tempo de GPS é usado como uma referência de tempo primária para toda a operação de GPS, é a referência de tempo coordenado universal - *universal coordinated time* (UTC).

O tempo zero de GPS está definido como a meia-noite da noite de 5 de janeiro ou a madrugada de 6 de janeiro de 1980. A unidade maior usada declarando tempo de GPS é uma semana, definido como 604,800 segundos (7 × 24 × 3600). (BAO, TSUI, 2005)

O tempo de GPS pode diferir do tempo UTC porque o tempo de GPS é uma escala de tempo contínua, enquanto que o UTC é corrigido periodicamente com um número inteiro de segundos. A escala de tempo GPS é mantida para estar dentro de 1μs de UTC (modulo de um segundo). Isto significa que às duas horas podem ser diferentes por um número inteiro de segundos. (BAO, TSUI, 2005)

2.3.5 Modulo GPS FV-12

O modulo GPS FV-12 é um equipamento de 12 canais de alto desempenho que vem com um pino de tempo real PPS, que serve de cronômetro ou atualização para relógio interno, entrada DGPS com seleção da taxa de transferência (em bauds), pinos de saída no padrão NMEA 0183 e memória não volátil no circuito integrado *onboard*. A figura 8 apresenta o modulo GPS FV-12 com a tecnologia SiRF®. (SANJOSE, 2005)

Figura 8: Modulo receptor de GPS

Fonte: SANJOSE, 2005

2.3.5.1 Pinagem do dispositivo

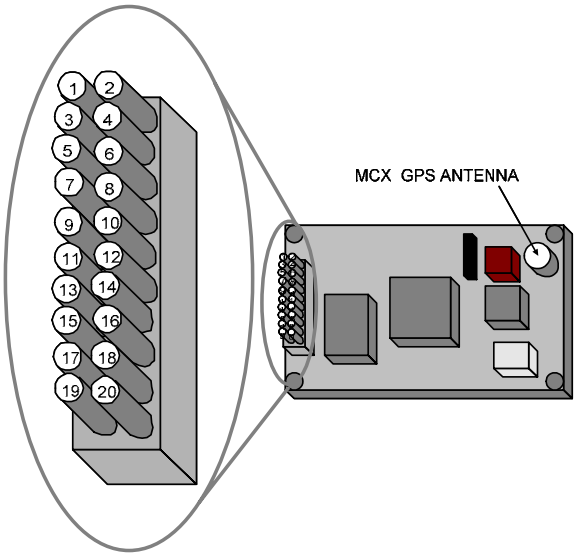


Figura 9: Pinos do modulo GPS em detalhe.

Fonte: SANJOSE, 2005

Tabela 2 – Pinagem do modulo GPS

Nº. PINO	***NOME***	***DESCRIÇÃO***	***TIPO***
1	VANT	Tensão DC para a antena	entrada
2	VDC	Entrada DC (3,8 – 6,5V)	entrada
3	VBAT	Bateria Backup	entrada
4	VDC	Em curto com pino 2	entrada
5	PBRES	Reiniciar a entrada	entrada
6	RESERVADO	(Não é utilizado)	
7	SELECT	Gravar dados da interface RS232 para ROM	
8	RESERVADO	(Não é utilizado)	
9	RESERVADO	(Não é utilizado)	
10	GND	Terra	
11	TXA	Dados seriais (GPS)	saída
12	RXA	Dados seriais (comandos)	entrada
13	GND	Terra	
14	TXB	Dados seriais (sem uso)	saída
15	RXB	Dados seriais (DGPD)	entrada
16	GND	Terra	
17	RESERVADO	(Não é utilizado)	
18	GND	Terra	
19	TIMEMARK	Pulso PPS	saída
20	RESERVADO	(Não é utilizado)	

Fonte: SANJOSE, 2005.

A tabela 2 representa a descrição dos pinos da figura 9. A figura 9 mostra o detalhe dos pinos de comunicação do modulo GPS.

O circuito integrado com tecnologia de SiRF®, FV-12 é particularmente proeminente para a aquisição do sinal *SnapLock*, dupla rejeição do sinal *Multi-Path* e sensibilidade do sinal *FoliageLock*. (SANJOSE, 2005)

2.3.5.2 Aquisição *SnapLock*

Renova a posição do satélite dentro de um décimo de um segundo depois de emergir de uma região bloqueada. Isto é crítico para restabelecer a precisão da posição rapidamente, como um automóvel que passa por interseções dentro um "canyon" urbano. (SANJOSE, 2005)

2.3.5.3 Posicionamento *SingleSat*

Permite um sistema de GPS (SiRF) prover informação de posicionamento durante intervalos em que só um satélite for "visível." (SANJOSE, 2005)

2.3.5.4 Rejeição dupla *Multi-path*

Elimina efetivamente os dados derivados de sinais que levaram um caminho indireto, por exemplo, sinais refletidos de objetos pertos e distantes. (SANJOSE, 2005)

2.3.5.5 Sensibilidade de sinal *FoliageLock*

Possibilita ao dispositivo operar em áreas de folhagem densa. (SANJOSE, 2005)

2.4 DISPLAY LCD

Os módulos LCD's são interfaces de saída muito úteis em sistemas microprocessados. Estes módulos podem ser gráficos ou do tipo caracteres. Os módulos LCD gráficos são encontrados com resoluções de 122x32, 128x64, 240x64 e 240x128 *dots pixel* (pontos de pixel), e geralmente estão disponíveis com 20 pinos para conexão. Os LCD mais comuns são de caracteres e estão especificados em número de linhas por colunas.

Os módulos podem ser encontrados com LED *backlight* (iluminação de fundo) para facilitar as leituras durante a noite. Neste caso, a alimentação deste LED faz-se normalmente pelos pinos 15 e 16 para os módulos comuns e 19 e 20 para os módulos gráficos, sendo os pinos 15 e 19 para ligação ao anodo e os pinos 16 e 20 para o catodo.

A corrente de alimentação deste LED varia de 100 a 200mA, dependendo do modelo. A figura 10 ilustra o display que será utilizado no presente trabalho.

Figura 10: Display LCD
Fonte: WINSTAR, 2006.

2.4.1 Ajuste de intensidade

O módulo LCD permite um ajuste na intensidade da luz emitida ou ajuste de contraste, isto é possível variando a tensão no pino 3. A Figura 11 mostra o circuito recomendado pela maioria dos fabricantes para efetuar este ajuste. Alguns fabricantes recomendam o uso de um resistor de 4K7 em série com o potenciômetro de 10K. (IME, 2006)

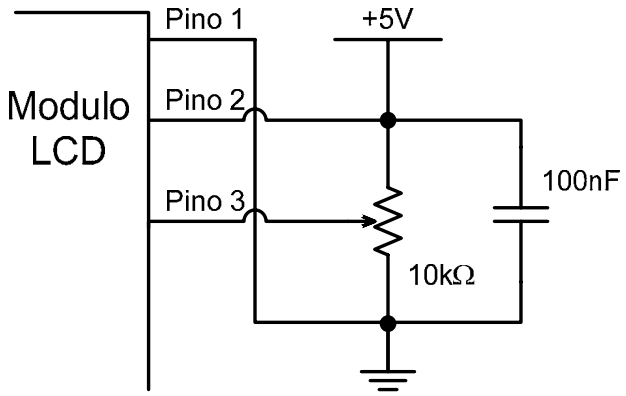


Figura 11: Circuito recomendado para o ajuste de intensidade
Fonte: IME, 2006.

2.4.2 Pinagem

Além de alimentar e conectar os pinos do módulo com a placa do usuário deve haver um protocolo de comunicação entre as partes, que envolve o envio de bytes de instruções e bytes de dados pelo sistema do usuário. A Tabela 3 descreve cada pino do display para conexão. (IME, 2006)

Tabela 3 – Pinagem do modulo LCD

PINO	***FUNÇÃO***	***DESCRIÇÃO***
1	Alimentação	Terra ou GND
2	Alimentação	VCC ou +5V
3	V0	Tensão para ajuste de contraste
4	RS	Seleção: 1 - Dado, 0 – Instrução.
5	R/W	Seleção: 1 - Leitura, 0 – Escrita.
6	E	Chip select1 ou (1 0) - Habilita, 0 – Desabilitado
7	B0 (LSB)	Barramento de Dados
8	B1	
9	B2	
10	B3	
11	B4	
12	B5	
13	B6	
14	B7 (MSB)	
15	Ânodo	Anodo p/ LED backlight
16	Kátodo	Catodo p/ LED backlight

Fonte: IME, 2006.

2.4.3 Controlador KS0066U

O KS0066U é um controlador de matriz de pontos para a operação do modulo LCD, instalado internamente no modulo. É um circuito integrado de amplia escala de integração (LSI) fabricado com tecnologia CMOS para um baixo consumo de energia. Pode exibir 1 ou 2 linhas com o formato de 5X8 pontos ou 1 linha com 5X11 pontos. (SAMSUNG, 2006)

2.4.4 Comandos

Os comandos ou instruções do modulo, são indicadas para estabelecer o modo de comunicações e fixar o modo de operação. A tabela 4 descreve estas instruções para operação do controlador do LCD, fornecidas pelo fabricante. (SAMSUNG, 2006)

Tabela 4 – Descrição dos comandos do modulo LCD

INSTRUÇÃO	RS	R/W	B7	B6	B5	B4	B3	B2	B1	B0	DESCRIÇÃO	t
Limpa Display	0	0	0	0	0	0	0	0	0	1	Limpa todo o display e retorna o cursor para a primeira posição da primeira linha	1.6 ms
Home p/Cursor	0	0	0	0	0	0	0	0	1	*	-Retorna o cursor para 1ª coluna 1ª linha, Retorna a mensagem previamente deslocada a sua posição original.	1.6 ms
Fixa o modo de funcionamento	0	0	0	0	0	0	0	1	ID	SH	-Estabelece o sentido de deslocamento do cursor (ID=0 p/ esquerda, ID=1 p/ direita)- Estabelece se a mensagem deve ou não ser deslocada com a entrada de um novo caractere (SH=1 SIM, ID=1 p/ direita) - Esta instrução tem efeito somente durante a leitura e escrita de dados.	40 us
Controle do Display	0	0	0	0	0	0	1	D	C	B	Liga (D=1) ou desliga display (D=0) - Liga (C=1) ou desliga cursor (C=0) - Cursor Piscante (B=1) se C=1	40 us
Desloca cursor ou mensagem	0	0	0	0	0	1	C	R	*	*	-Desloca o cursor (C=0) ou a mensagem (C=1) para a Direita se (R=1) ou esquerda se (R=0) - Desloca sem alterar o conteúdo da DDRAM	40 us
Fixa o modo de utilização do módulo LCD	0	0	0	0	1	Y	N	F	*	*	-Comunicação do módulo com 8 bits (Y=1) ou 4 bits (Y=0) - Número de linhas: 1 (N=0) e 2 ou mais (N=1) - Matriz do caractere: 5x7 (F=0) ou 5x10 (F=1) - Esta instrução deve ser ativada durante a inicialização	40 us
Posiciona no endereço da CGRAM	0	0	0	1	Endereço da CGRAM						-Fixa o endereço na CGRAM para posteriormente enviar ou ler o dado (byte)	40 us
Posiciona no endereço da DDRAM	0	0	1	Endereço da DDRAM							-Fixa o endereço na DDRAM para posteriormente enviar ou ler o dado (byte)	40 us
Leitura do Flag Busy	0	1	BF	AC							-Lê o conteúdo do contador de endereços (AC) e o BF. O BF (bit 7) indica se a última operação foi concluída (BF = 0 concluída) ou está em execução (BF=1).	0
Escreve dado na CG/DDRAM	1	0	Dado a ser gravado no LCD								- Grava o byte presente nos pinos de dados no local apontado na *posição do cursor*.	40 us
Lê dado CG/DDRAM	1	1	Dado lido do módulo								- Lê o byte no local apontado pelo contador de endereços	40 us

Legenda: * - Indiferente; t – tempo de execução da instrução.

Fonte: SAMSUNG, 2006.

2.4.5 Ciclo ENABLE

Os módulos LCD são projetados para conectar-se com a maioria das CPU's disponíveis no mercado, bastando para isso que esta CPU atenda as temporizações de leitura e escrita de instruções e dados, fornecidas pelo fabricante do módulo. (IME, 2006)

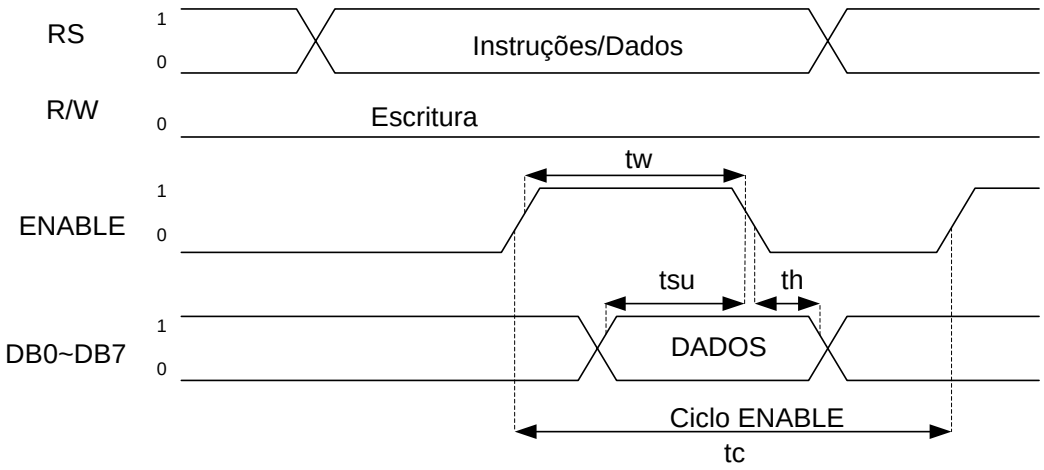


Figura 12: Diagrama de tempos para o pulso *ENABLE*

Fonte: SAMSUNG, 2006.

A tabela 5 descreve as temporizações mínimas para o ciclo Enable, fornecida pelo fabricante do controlador do modulo KS0066U.

Tabela 5 – Descrição dos tempos para a figura 12

SIGLA	***VALOR MÍNIMO***	***DESCRIÇÃO***
Tc	500 ns	Período do pulso ENABLE
Tsu	80 ns	Tempo de ativação dos dados
Th	10 ns	Tempo de fixação dos dados
Tw	230 ns	Largura do pulso Enable (1 → 0)

Fonte: SAMSUNG, 2006.

Estes módulos utilizam um controlador próprio, permitindo sua interligação com outras placas através de seus pinos, onde deve ser alimentado o módulo e interligado ao barramento de dados e controle do módulo com a placa do usuário.

2.4.6 Inicialização do display

Toda vez que se alimenta o módulo LCD deve ser executado o procedimento de inicialização, que consiste no envio de uma seqüência de instruções para configurar o modo de operação. Em muitos displays este procedimento ocorre automaticamente, dentro das condições específicas que envolvem as temporizações mínimas, referente a transição do ciclo *Enable*, que ativa a instrução ou o dado que será escrito.

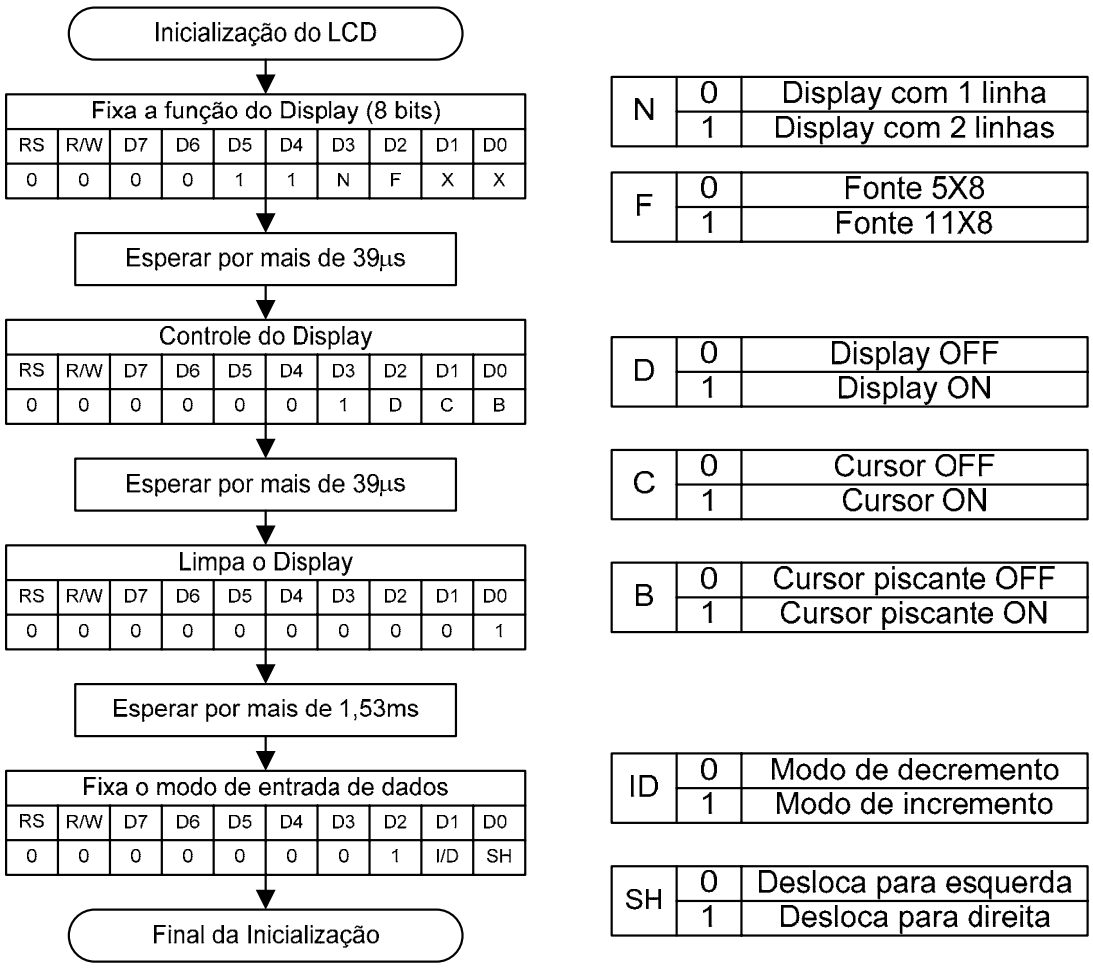


Figura 13: Diagrama de fluxo para a inicialização do Display

Fonte: SAMSUNG, 2006.

3 DISPOSITIVOS DE SOFTWARE

Os dispositivos de software têm uma importância especial no trabalho, pois são a peça fundamental para a realização deste, onde será feita as implementações e também as simulações para a verificação da funcionalidade.

Dar-se-á uma descrição completa sobre as funcionalidades da linguagem de programação de hardware (VHDL), amplamente utilizada neste trabalho, e também o que demandou maior tempo de aprendizagem e simulações para obter-se o grau de entendimento necessário para proceder com a implantação do trabalho final.

3.1 QUARTUS II

O software Universitário QuartusII contém muitas das características da versão comercial do software, inclusive um fluxo de design completamente integrado e uma interface gráfica de usuário.

Este software suporta a captura esquemática e entrada de design em linguagem de descrição de hardware (HDL), baseada em texto, como Verilog HDL, VHDL, e a linguagem *Altera Hardware Description* (AHDL).

Também provê programação, compilação e apoio de verificação para todos os dispositivos suportados pelo QuartusII (edição web), inclusive o EPM7128S e o EPF10K70. Este software é distribuído livremente aos estudantes, na internet.

Provê um ambiente de design completo de multi-plataforma que facilmente se adapta as necessidades específicas do usuario. Também permite usar a interface gráfica de usuário, ferramentas de interface EDA ou interface de comando-linha para cada fase do design. (QUARTUS II, 2005)

3.1.1 Projeto de design

Todo trabalho no QuartusII, constitui um projeto e segue uma estrutura básica de desenvolvimento. Primeiramente é necessário organizar os objetivos do projeto, para prosseguir com o design, entre eles é importante saber como deverá ser o funcionamento do projeto, definir as entradas e as saídas e finalmente saber o que se espera para os resultados. (DUECK, 2005)

Ao criar um projeto dever-se-á seguir os seguintes passos:

- Utilizar o menu ***FILE*** e selecionar a opção ***New Project Wizard***;
- Indicar a pasta para a alocação do projeto e também o nome do arquivo principal do projeto, que pode ser esquemático ou textual como VHDL.
- Indicar o dispositivo que será utilizado. Se o cabo *ByteBlaster* estiver conectado ao kit, o software automaticamente detecta o dispositivo.

A figura 14 mostra a tela visualizada pelo usuário no momento em que cria um projeto no QuartusII, ilustrando os passos descritos anteriormente.

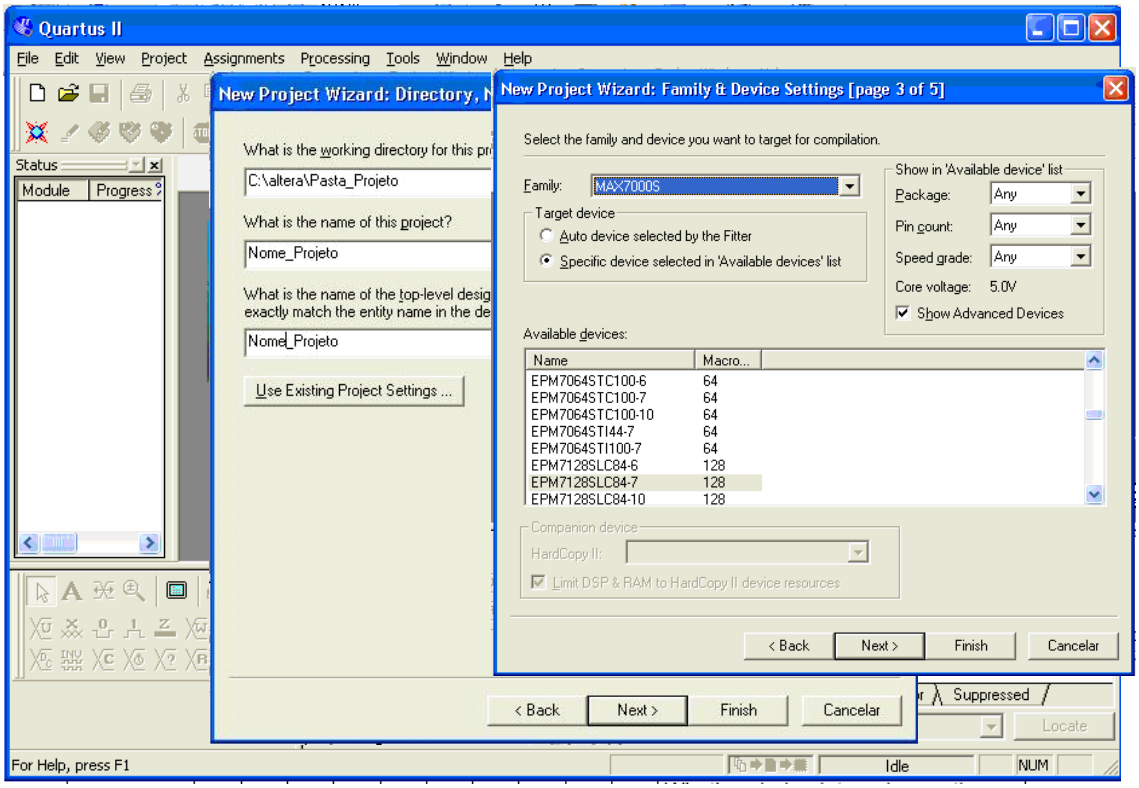
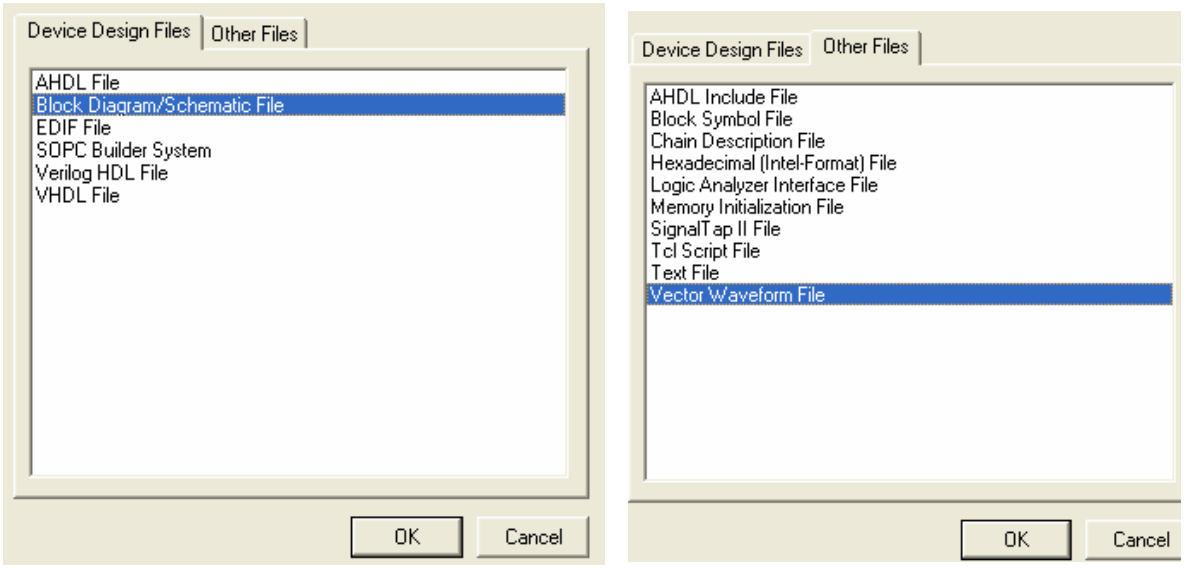


Figura 14: Novo projeto no QuartusII

Uma vez criado o projeto procede-se a elaboração do arquivo de entrada, neste trabalho somente foi utilizado a forma gráfica (*Block Diagram File*) e forma textual utilizando a linguagem de hardware (VHDL), que será detalhada na seção 3.2 deste trabalho.

A figura 15 a., mostra um exemplo de como ingressar um novo arquivo ao projeto, procede-se da seguinte forma:

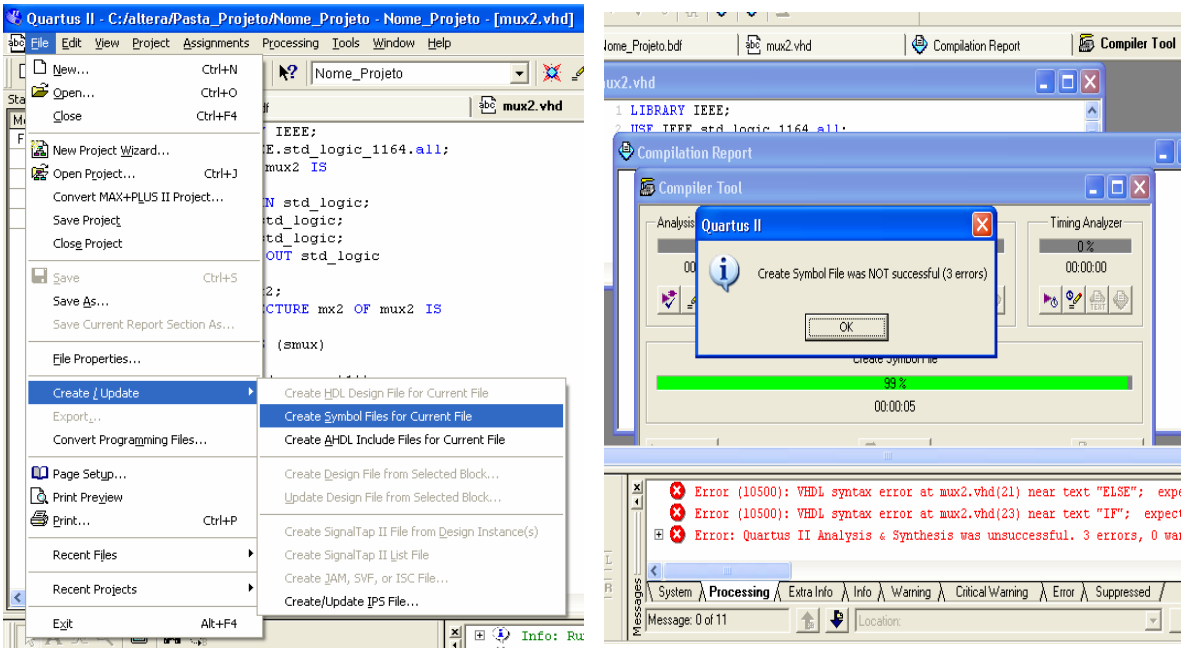
- Utilizando o menu ***FILE*** ou ícone de novo documento, seleciona-se o arquivo que será utilizado, neste exemplo procede-se a criar um arquivo de diagrama em blocos ou arquivo gráfico.



a) **b)**

Figura 15: a) arquivos de design; b) outros arquivos.

Após a criação de um novo arquivo, seja ele gráfico ou textual, é possível gerar um bloco de símbolos com a funcionalidade nele descrita, do arquivo que foi criado. A figura 16 a., ilustra o processo de criação deste símbolo gráfico, já a figura 16 b. mostra um erro na compilação.



a) **b)**

Figura 16: a) criar um simbolo gráfico apartir de arquivo; b) Erro na compilação do objeto.

Todo o projeto no QuartusII, deve ser compilado antes de proceder a simulação, se houver erros no processo, estes devem ser solucionados, pois não será possível fazer as simulações pretendidas pelo projeto.

Na seção de desenvolvimento deste trabalho, será novamente abordados estes temas, de uma forma mas detalhada e explicando cada arquivo que foi criado seja ele gráfico ou textual, pois estes são a peça fundamental deste trabalho de conclusão do curso.

3.2 VHDL

A *VHSIC*[18] *Hardware Description Language* (VHDL) – linguagem de descrição de hardware – é uma linguagem usada para facilitar o design (projeto e concepção) dos circuitos digitais em FPGAs[19] e ASICs[20].

> A VHDL foi originalmente desenvolvida sob o comando do Departamento de Defesa (DoD) dos Estados Unidos, em meados da década de 80, para documentar o comportamento de ASICs que compunham os equipamentos vendidos às Forças Armadas americanas. Isto quer dizer que a linguagem VHDL foi desenvolvida para substituir os complexos manuais que descreviam o funcionamento dos ASICs. Até aquele momento, a única metodologia largamente utilizada no projeto de circuitos era a criação através de diagramas esquemáticos. (WIKIPÉDIA, 2006)

O desenvolvimento da VHDL serviu inicialmente aos propósitos de documentação do projeto VHSIC. Entretanto, nesta época buscava-se uma linguagem que facilitasse o projeto de um circuito; ou seja, a partir de uma descrição textual, um algoritmo, desenvolver o circuito, sem necessidade de especificar explicitamente as ligações entre componentes. O VHDL presta-se adequadamente a tais propósitos, podendo ser utilizado para as tarefas de documentação, descrição, síntese, simulação, teste, verificação formal e ainda compilação de software, em alguns casos. (PERRY, 2002)

Após o sucesso inicial do uso da VHDL, a sua definição foi posta em domínio público, o que levou a ser padronizada pelo *Institute of Electrical and Electronic Engineers* (IEEE) em 1987. O fato de ser padronizada e de domínio público ampliou ainda mais a sua utilização, novas alterações foram propostas, como é natural num processo de aprimoramento e a linguagem sofreu uma revisão e um novo padrão mais atualizado foi lançado em 1993. (PERRY, 2002)

---

[18] Very High Speed Integrated Circuit
[19] Field Programmable Gate Array
[20] Application Specific Integrated Circuit

O problema com esta metodologia é o fato de que desenho tem menor portabilidade, são mais complexos para compreensão e são extremamente dependentes da ferramenta utilizada para produzi-los. (RHOD, 2006)

O seguinte apartado foi extraído em partes do livro *"Lenguajes de Alto Nivel para Diseño de Circuitos Integrados Digitales"* (ECHANOVE, 2001), onde os exemplos empregados neste livro foram substituídos pelo código fonte que foi utilizado no presente trabalho.

3.2.1 Estrutura de um desenho em VHDL

Cada unidade na hierarquia tem uma entidade, como um circuito com entradas, saídas e funções. A unidade hierárquica é designada pela palavra reservada ***ENTITY*** (entidade). A **entidade** tem associado um nome de identificação usualmente relativo à função que realiza. Cada vez que se faz uso deste circuito utiliza-se o nome associado a esta, definindo sinais de enlace com o exterior e uma **arquitetura** funcional.

O seguinte exemplo descreve a estrutura geral de um desenho em VHDL.

```
LIBRARY nome_livraria;                    -- Zona de declaração das livrarias
USE pacote_funções_livraria.all;          -- Cabeçalho da entidade.
ENTITY nome_entity IS
        PORT(.....);                      -- Corpo da entidade.
END nome_entity;
ARCHITECTURE nome_architecture OF nome_entity IS
BEGIN                                     -- Descrição da funcionalidade
        PROCESS (lista de sensibilidades)
        VARIABLE a: std_logic;            -- Declaração das variáveis do processo.
        BEGIN
                                          -- Corpo do processo.
        END PROCESS;                      -- Fim do processo.
END nome_architecture;                    -- Fim da arquitetura.
```

Note-se que em VHDL os comentários são escritos apartir da colocação dos dois sinais de menos (--), e o QuartusII está predeterminado com a cor verde para designar os comentários, então ao digitar automaticamente o texto muda para a cor verde.

Em VHDL está previsto a possibilidade de modelar diferentes arquiteturas para uma mesma entidade, existindo assim a necessidade de designar nomes tanto para a entidade como a arquitetura.

3.2.1.1 Seção ENTITY

Identifica a unidade hierárquica de forma unívoca. Não existindo a possibilidade de duas entidades de mesmo nome definidas de forma diferente. Junto com a palavra ENTITY está associado à cláusula PORT.

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
ENTITY corac IS                    -- nome da entidade.
    PORT(                          -- São definidas as portas de entrada e saída da entidade.
    clk, entra, s: IN std_logic;
    a, b, c, d, e, f, g, h : OUT std_logic);
END corac;                         -- Fim da entidade.
```

A cláusula PORT indica os sinais que relacionam a entidade com o resto do desenho. Possui a seguinte sintaxe:

Nome_sinal :* DIREÇÃO *tipo_do_sinal;

O nome do sinal identifica univocamente o sinal em questão, não podendo haver outro sinal com o mesmo nome e diferente direção dentro da arquitetura. A direção indica o sentido do fluxo dos dados deste sinal da entidade podendo tomar os seguintes valores:

- IN: indica que é uma entrada.
- OUT: para indicar que o sinal é uma saída.
- INOUT: indicar que o sinal pode ser uma entrada ou uma saída.
- BUFFER: tipo estendido de saída.

3.2.1.2 Seção ARCHITECTURE

A arquitetura da entidade é a descrição da funcionalidade da mesma. Consta de duas partes, a parte declarativa onde se especificam os elementos do tipo estrutural que irão compor a arquitetura, como segue:

Sinais internos: são enlaces entre os diferentes elementos que definem a arquitetura.

Entidades de ordem inferior na hierarquia, chamados de componentes (**component**). No presente trabalho não foi utilizado esse recurso.

```
ARCHITECTURE cmc OF corac IS
SIGNAL sai, output, letra: std_logic_vector (8 DOWNTO 1); -- Sinais internos.
BEGIN
PROCESS ( clk,s)                        -- Lista de sensibilidades.
VARIABLE r, conta: INTEGER:=0;          -- Variáveis do processo.
VARIABLE setres: std_logic:='0';
BEGIN                                   -- Começo do processo.
      IF (clk'EVENT AND clk='1')
      THEN
                                        -- Clausulas da sentença IF
      END IF;
      CASE conta IS
                                        -- Clausulas da sentença CASE
      END CASE;
END PROCESS;
END cmc;
```

Na parte descritiva da arquitetura se define a funcionalidade da entidade, onde é possível incluir os comandos e funções necessárias para o funcionamento correto do desenho digital. Especificam-se os sinais mediante a palavra ***"signal"*** (antes mesmo do começo do corpo da arquitetura), seguido de um ou vários nomes. Todos os sinais especificados pertencem ao mesmo tipo, como se pode apreciar a sintaxe a seguir:

SIGNAL nome1, nome2,... : *Tipo_do_sinal*:= *valor inicial*;

Este é o mesmo esquema utilizado na declaração de sinais na entidade, com a diferença de omitir a direcionalidade do sinal, sendo que o valor inicial é opcional. Também é possível declarar valores constantes ao desenho, segundo a sintaxe:

CONSTANT nome: *Tipo_ do_ sinal*:= *valor inicial*;

3.2.1.3 Seção LIBRARY

Mediante as bibliotecas define-se os componentes, funções e procedimentos, tais como unidades de linguagem que podem ser úteis em muitas aplicações, sendo um elemento de reutilização da linguagem.

3.2.1.3.1 Livraria std_logic_1164

A linguagem VHDL possui um limitado numero de tipos e funções que resultam insuficientes na hora de descrever completamente o comportamento de um circuito digital. Por tanto é necessário que a extensão da linguagem contemple um numero amplio de situações, sendo que os tipos de dados se organizam da seguinte forma:

- **BIT** , {0,1}
- **BOOLEAN**, {false, true}
- **CHARACTER**, {a tabela ASCII desde 0 a 127}
- **INTEGER**, {-2147483647, 2147483647}
- **NATURAL**, {0, 2147483647}
- **POSITIVE**, {1, 2147483647}
- **STRING**, array {POSITIVE range <>} de CHARACTER
- **BIT_VECTOR**, array {NATURAL range <>}de BIT
- Tipos físicos, como **TIME**
- **REAL**, {-1E38, 1E28}
- **POINTER**, para acessos indiretos de dados.
- **FILE**, para acessos aos fichários e pilhas de dados do disco rígido.

Trata-se do pacote *std_logic_1164* da livraria IEEE, que normalmente dá um sentido novo aos tipos de dados, o *std_logic*, é um tipo de enumerado de dados. O cabeçalho *standard* da livraria é tipicamente assim:

LIBRARY IEEE;
USE IEEE.std_logic_1164.**all**;

3.2.2 Conceito de concorrência.

Um circuito eletrônico é de natureza paralela, onde ao aplicar um sinal em uma determinada entrada se obtém um resultado específico ao qual está projetado.

Quando se introduz sentenças ou blocos (também chamados de processos), todos são concorrentes entre si, não existe precedência entre eles, não importando a ordem em que se apresentam no desenho, todos interagem ao mesmo tempo, quando há uma mudança nos sinais que os afetam.

Existem em VHDL cinco tipos de elementos concorrentes:

- Blocos (**BLOCK**). Grupos de sentenças concorrentes.
- Sentenças (**GENERATE**). Geração de copias de blocos de hardware.
- Chamadas a procedimentos e funções (**PROCEDURE** e **FUNCTION**). Os procedimentos são algoritmos combinacionais que computam e designa valores aos sinais.
- Declaração e definição de processos (**PROCESS**). Definem algoritmos seqüenciais que lêem valores de sinais e designam valores a outros sinais. Designações de sinais computados a través de expressões booleanas ou aritméticas.

No presente trabalho somente foi utilizado o código seqüencial (**PROCESS**), que é o mais comum entre os tipos. Um processo é um bloco que contém código seqüencial, mas que externamente se contempla como código concorrente.

3.2.3 Estrutura PROCESS

Para compreender melhor o funcionamento do processo é conveniente introduzir uma perspectiva de simulação e de síntese. Sobre o ponto de vista de simulação o processo contém uma seqüência de transformações sobre os sinais que são produzidos uma vez que foi ativado o mesmo.

Desde o ponto de vista de síntese, o processo representa uma porção de um circuito é uma imagem de hardware, o conjunto de sentenças que o compõem.

3.2.3.1 Elementos de um processo.

O processo se introduz na zona da arquitetura e não existe restrição alguma quanto ao seu numero e extensão. Os elementos de um processo seguem a seguinte sintaxe:

```
PROCESS (lista de sensibilidade)
VARIABLE ...
BEGIN
        --seqüência de ordens
END PROCESS;
```

3.2.3.2 Lista de sensibilidade.

A lista de sensibilidade e um artifício para realizar computacionalmente o paralelismo, principalmente em simulação. E um conjunto de sinais que ativam o processo, mediante uma mudança em qualquer um dos sinais da lista, o processo é avaliado completamente.

A seguir apresenta-se o processo utilizado no código fonte deste trabalho:

```
PROCESS ( clk , s)                          -- Lista de sensibilidades.
VARIABLE r, conta: INTEGER:=0;                  -- Variaveis do processo.
VARIABLE setres: std_logic:='0';
BEGIN
        IF (clk'EVENT AND clk='1')          -- Começo do processo.
        THEN
                                            -- Clausulas da santença IF
        END IF;
        CASE conta IS
                                            -- Clausulas da santença CASE
        END CASE;
END PROCESS;
```

3.3 PROTOCOLO NMEA

A *National Marine Electronics Association* (NMEA) é uma associação sem fins lucrativos de fabricantes, distribuidores, negociantes, instituições educacionais, e outros que se interessam por ocupações periféricas de eletrônica marinha. O padrão NMEA 0183 define uma interface elétrica e um protocolo de dados para comunicações entre instrumentos marinhos. (BETKE, 2001)

NMEA 0183 é um padrão voluntário da indústria, o primeiro foi lançado em março de 1983. Foi atualizado de vez em quando; a mais recente atualização foi em agosto de 2001 com a versão 3.0 e está disponível no escritório do NMEA.

3.3.1 Sintaxe do protocolo NMEA

A sintaxe da mensagem do protocolo NMEA, que usa vírgulas para a separação da informação relevante:

$GPGGA,161229.487,3723.2475,N,12158.3416,W,1,07,1.0,9.0,M, , , ,.0000*18

Tabela 6 – Descrição do protocolo NMEA

NOME	***EXAMPLE***	***UNITS***	***DESCRIPTION***
Cabeçalho	$GPGGA		GGA cabeçalho do protocolo
Hora UTC	161229.487		hhmmss.sss
Latitude	3723.2475		ddmm.mmmm
Indicador N/S	N		N=norte ou S=sul
Longitude	12158.3416		dddmm.mmmm
Indicador E/W	W		E=leste ou W=oeste
Fixa o indicador de Posição	1 (ver legenda)		GPS SPS mode,fix valid
Satélites usados	07		Range 0 to 12
HDOP	1.0		Horizontal Dilution of Precision
Altitude MSL	9.0	meters	
Unidades	M	meters	
Separação Geoidal		meters	
Unidades	M	meters	
Ages of Diff. Corr.		second	Nulo quando DGPS não é usado.
Diff. Ref. Station ID	0000		
Checksum	*18		
<CR> <LF>			Final da mensagem do protocolo

Legenda: 0=invalid; 1=Fix valid GPS; 2=DGPS Fix; 3=GPS PPS Fix.

Fonte: SANJOSE, 2005.

3.3.2 Interface elétrica

- Taxa de bits: 4800 bits por segundo
- Número de bits de dados: 8 (bit 7 é 0)
- Bit de STOP: 1 (ou mais)
- Paridade: nenhum

4 DESENVOLVIMENTO

O principal desenvolvimento deste trabalho é a elaboração do desenho de hardware com o software QUARTUS II, utilizando a linguagem VHDL. O desenho de hardware demandou maior parte do tempo destinado a este trabalho.

A principal dificuldade encontrada durante este processo, radicou em compreender este "novo método de programação"[21], pois a programação em paralelo difere muito dos outros métodos convencionais como, C++, Pascal, Visual Basic, etc.

Quando se atinge o grau de conhecimento necessário para a elaboração do primeiro desenho de hardware, um novo horizonte se abre, surgem novas idéias, e novos projetos são possíveis de realizar, utilizando esta poderosa ferramenta.

A figura 5 mostra um diagrama esquemático do projeto para visualizar melhor às partes principais deste projeto que posteriormente será internamente descrita e detalhada.

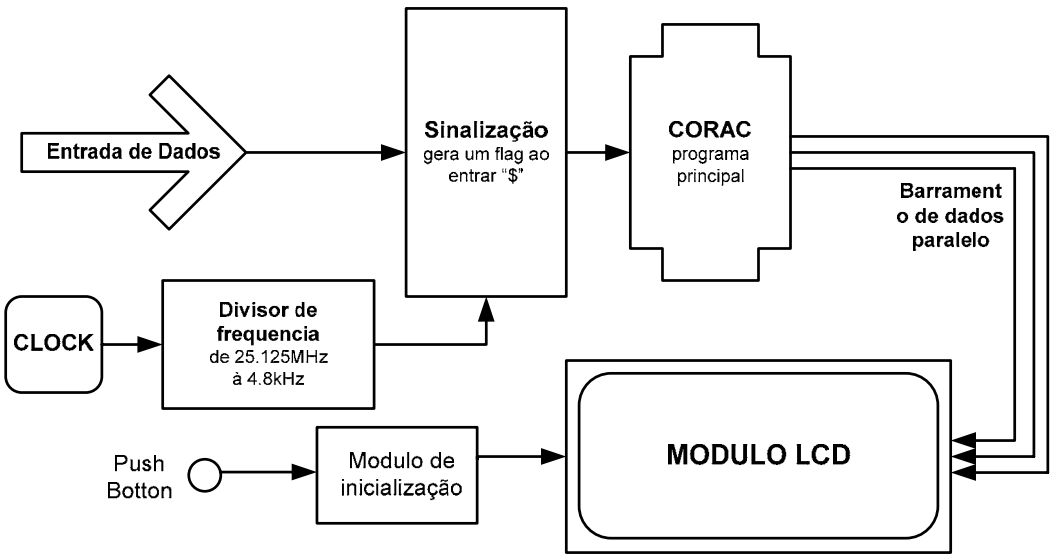


Figura 17: Descrição esquematica do projeto realizado.

O diagrama completo do projeto é mostrado na figura 18, em seguida será feita à descrição de todas as partes do diagrama, de cada bloco especifico. Os fluxogramas dos blocos e seus códigos fontes estarão detalhados na seção de apêndices.

---

[21] Neste trabalho foi o primeiro contato que o autor teve com essa linguagem.

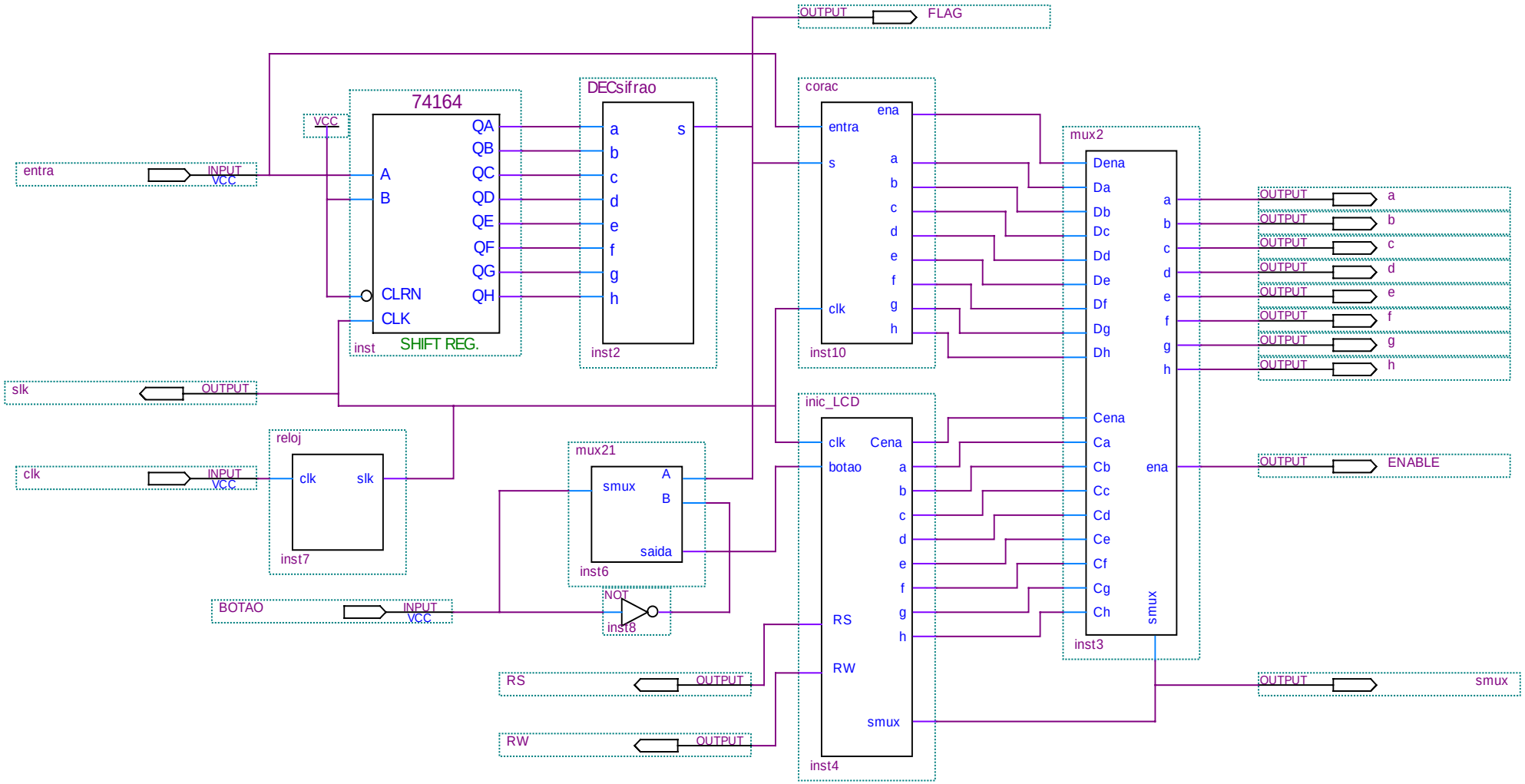


Figura 18: Descrição total do projeto realizado no QUARTUS II.

4.1 BLOCO 74164

É um registro de 8 bits com entrada de dados serial e uma saída para cada um dos oito bits da palavra da entrada. Os dados entram em forma serial por uma das duas entradas (A ou B), qualquer uma destas entradas pode ser usada como um habilitador ativo em alto para a outra entrada de dados. Cada transição de baixo a alto na variação do relógio, desloca o bit da entrada para a primeira saida (QA). Um baixo nível na entrada *clear*, anula todas as outras entradas e limpa o registro assíncrono, forçando todas as saídas Q para o estado baixo.

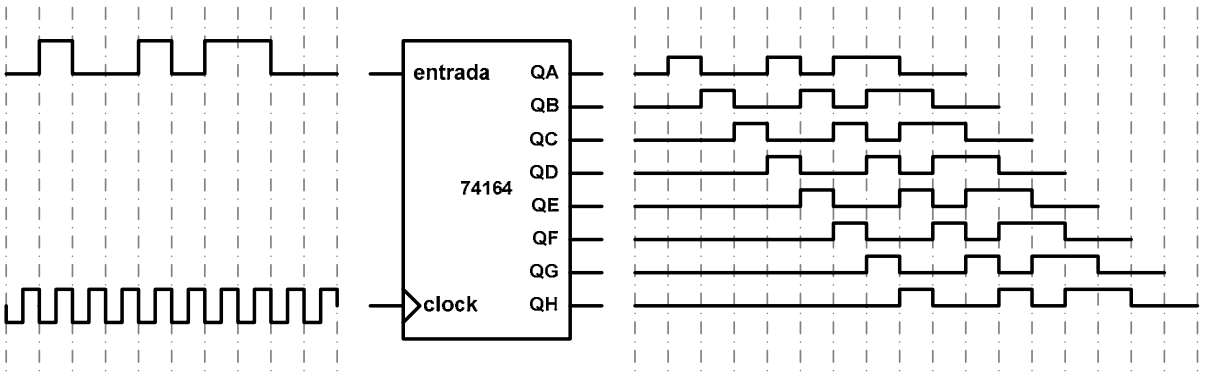


Figura 19: Descrição da funcionalidade do bloco 74164.

4.2 BLOCO DECSIFRAO

Este bloco permite detectar o "$" do cabeçalho do protocolo NMEA. O caractere "$" em código ASCII, está representado pelo numero binario "00100100" e ao detectar este caractere na entrada, o bloco gera um bit de *flag* que será utilizado pelo bloco "corac" para o começo de seu processo interno.

No código ASCII não existem dois caracteres iguais, ou com a designação dos números binários iguais e tampouco existe na mensagem do protocolo NMEA, dois símbolos "$" repetidos na mensagem, então sempre e quando este bloco detecta a presença deste símbolo, estar-se-á começando uma nova mensagem NMEA.

Para sua construção foram utilizadas comportas lógicas do tipo XOR e BAND8, como está descrito na figura a seguir:

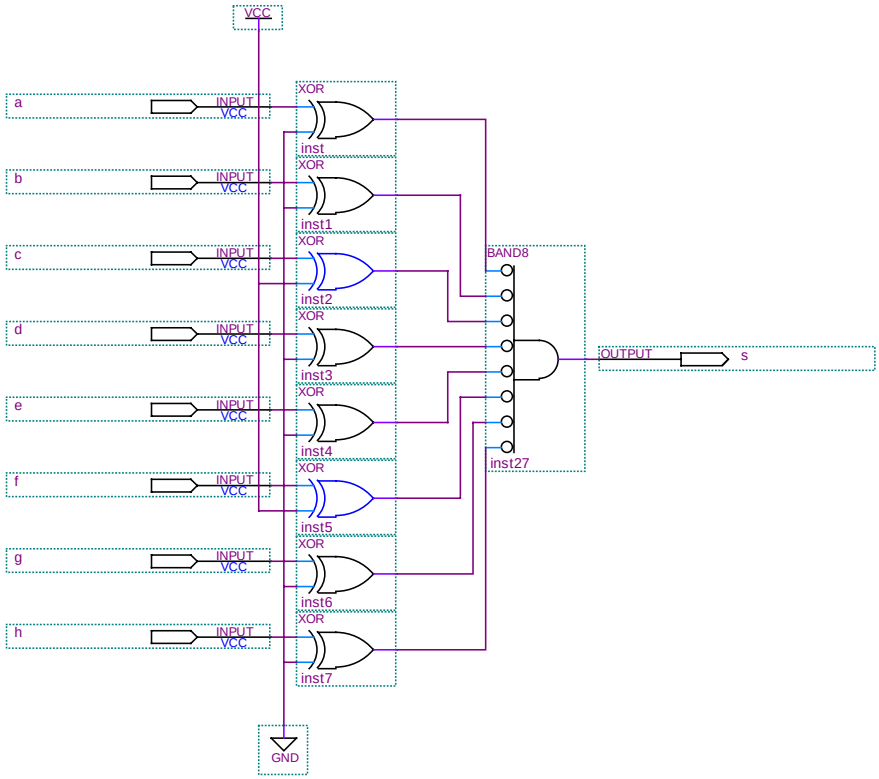


Figura 20: Comportas utilizadas para o modulo DECsifrao.

4.3 BLOCO RELOJ

A função deste bloco é reduzir freqüência do *clock on board* da placa UP2, para adaptar ao *clock* de envio de dados do modulo GPS. Os dados enviados pelo receptor de sinais GPS são transmitidos a uma taxa de bits de 4800 bits por segundos e o relógio do kit UP2 tem uma freqüência já estabelecida de 25,175MHz.

O programa parte da seguinte lógica:

$\frac{25.175.000}{4.800} \cong 5.244{,}8$ pulsos de *clock* da ALTERA, constituem um pulso necessário para o *clock* do GPS, ao considerar um ciclo de trabalho de 50%, serão necessários aproximadamente 2622,4 pulsos para fazer a mudança de estado, que caracteriza o *clock* do bloco reloj. No apêndice G e H estão descritos respectivamente, o diagrama de fluxo e o código fonte deste bloco.

Para diminuir o erro pela aproximação ao valor real e pelo fato de que não existe resíduo de bits, somente pode-se trabalhar com números inteiros, para isso foi implementado dois sistemas de contagem, para evitar este erro.

A cada 5 transições de estado do *clock* de 4800Hz, aumenta-se 2 bits do *clock* de 25,175MHz e a cada 96 vezes que procede-se a incrementar os 2 bits, ignora-se o incremento, tal como mostra a seguinte figura 21:

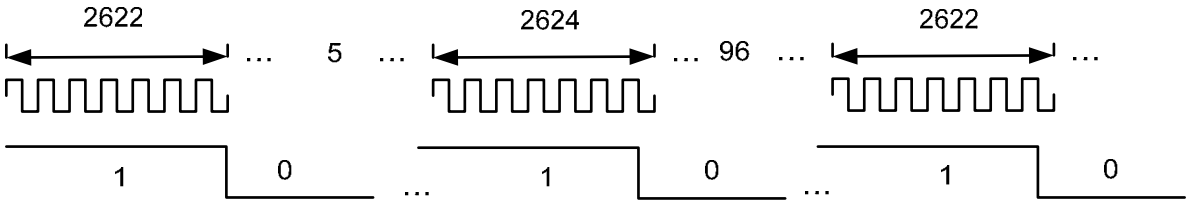


Figura 21: Procedimento para gerar o clock.

4.4 BLOCO CORAC

A designação CORAC refere-se à palavra coração, devido que esta é a parte principal do projeto, capaz de realizar a separação dos dados importantes do protocolo NMEA, valores de importância ao projeto.

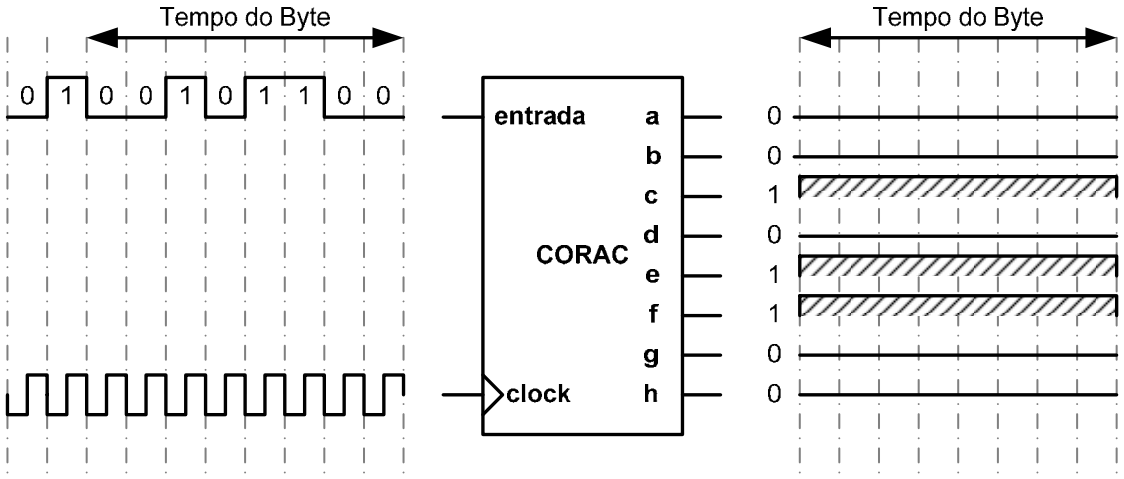


Figura 22: Descrição da funcionalidade do bloco CORAC.

Este bloco realiza o processo de encher uma localidade de memória que para fins da nomenclatura do VHDL é chamado de vetor, com isso cada bit na entrada do bloco, que são os dados seriais enviados pelo modulo GPS, ocupa uma posição dentro do vetor.

Depois de ingressar os bits de cada caractere da mensagem do protocolo NMEA, verifica-se se este caractere faz parte da hora UTC, uma vez confirmado, este byte é enviado completamente ao barramento de dados do modulo LCD.

Após ser concluído o envio de todos os caracteres da hora UTC, num total de 10 caracteres incluindo a vírgula de separação decimal, o processo cessa e aguarda o envio de uma nova mensagem do GPS, esta que é enviada com um intervalo de 100ms.

O fluxograma deste bloco, bem como o seu código fonte em linguagem VHDL é apresentado no apêndice A e B, respectivamente.

4.5 BLOCO MUX2

A função deste bloco é permitir a separação entre o bloco “corac” e bloco de inicialização do modulo LCD chamado “ini_LCD”, já que estes utilizam o mesmo barramento de dados do modulo LCD.

Este bloco evita conflitos entre estes os dois blocos, pois se o bloco “corac” envia dados ao barramento do LCD, enquanto o bloco “ini_LCD” está ocupando o barramento, os comandos não serão compreendidos pelo controlador do modulo LCD.

O fluxograma e o código fonte estão descritos nos apêndices E e F, respectivamente.

4.6 BLOCO INI_LCD

Este bloco é utilizado para enviar comandos de instruções para o modulo LCD, tal como foi descrito na seção 2.4.6 deste trabalho. A figura 13 apresenta um fluxograma com os tempos mínimos para cada instrução, e também as instruções mais importantes para inicialização do modulo LCD. O fluxograma e o código fonte estão descritos nos apêndices C e D, respectivamente.

Associado a este bloco está um MUX de 2 entradas com uma saída, que bloqueia a entrada do pulso de *flag* enviado pelo bloco “DECsifrao”, quando pressionado o botão do kit UP2, pois tanto o botão (*push botton*), como o bit de flag, acionam o envio de instruções ao modulo LCD. Uma vez acionado este bloco de envio de instruções, este envia o bit seletor “smux” ao bloco “mux2” para cancelar a entrada de dados ao barramento do modulo LCD.

O bloco “mux21”, não está descrito na cessão de apêndices por ser elemental e seu código fonte é semelhante ao bloco “mux2”.

5 SIMULAÇÕES

As simulações deste trabalho foram feitas utilizando o software QuartusII em sua interface de simulação, com o arquivo ***waveforms***, após ser feita a compilação do diagrama final do projeto, procede-se a criação do arquivo com as possíveis formas de onda para as entradas e indicando as saídas que posteriormente serão simuladas.

A figura a seguir mostra a tela do simulador com os valores já calculados para as saídas do projeto final, elaborado neste trabalho.

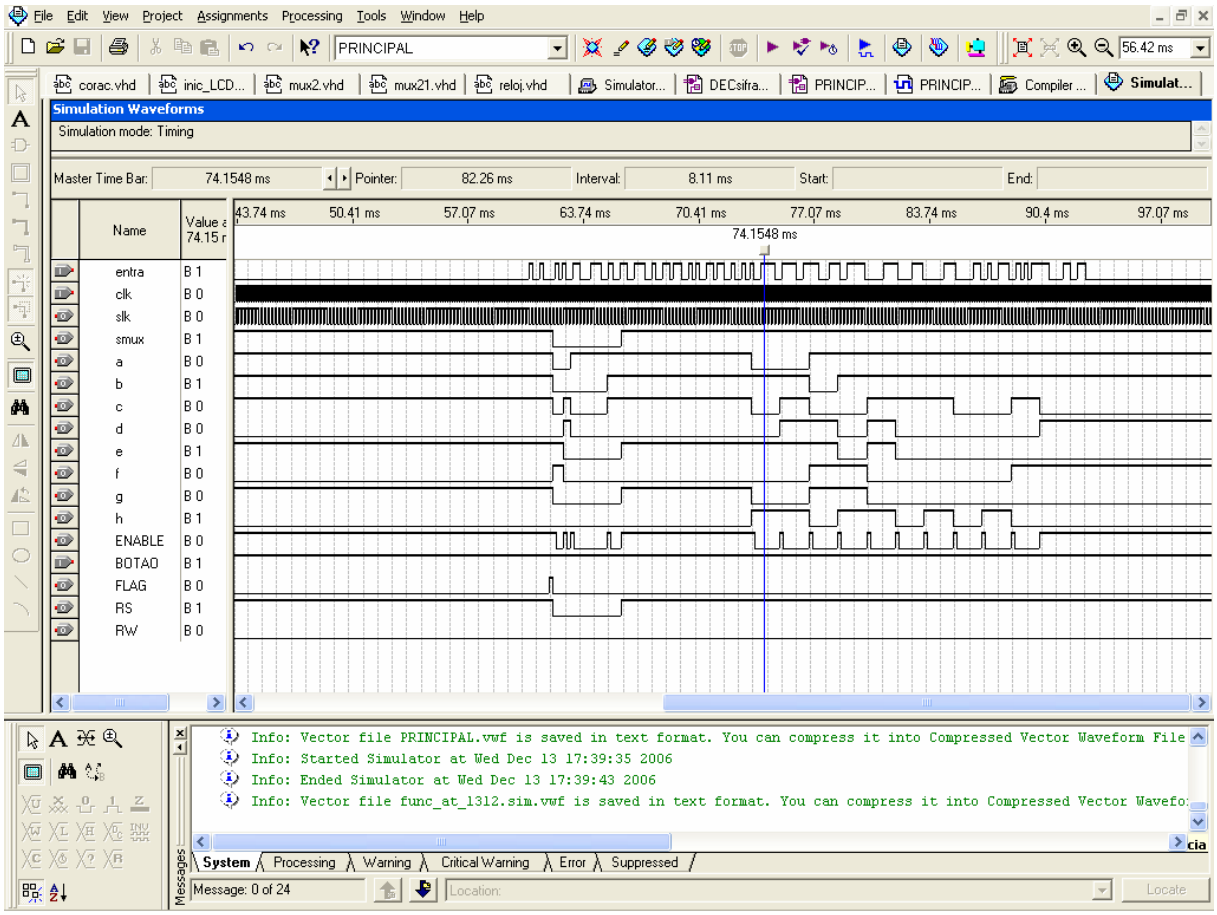


Figura 23: Tela do simulador do QuartusII.

5.1 DESCRIÇÃO DOS PONTOS RELEVANTES

Os pontos de maior importância que serão detalhados são: a inicialização do display utilizando o *push botton* e a incialização do módulo LCD a partir do bit de *flag*.

5.1.1 Inicialização pelo usuário

Quando o usuário pressiona o botão do kit UP2 da Altera, o valor deste muda de estado de normalmente em alto, para o estado baixo (0 lógico). A figura mostra as formas de onda para o botão e as saídas do projeto, já simuladas.

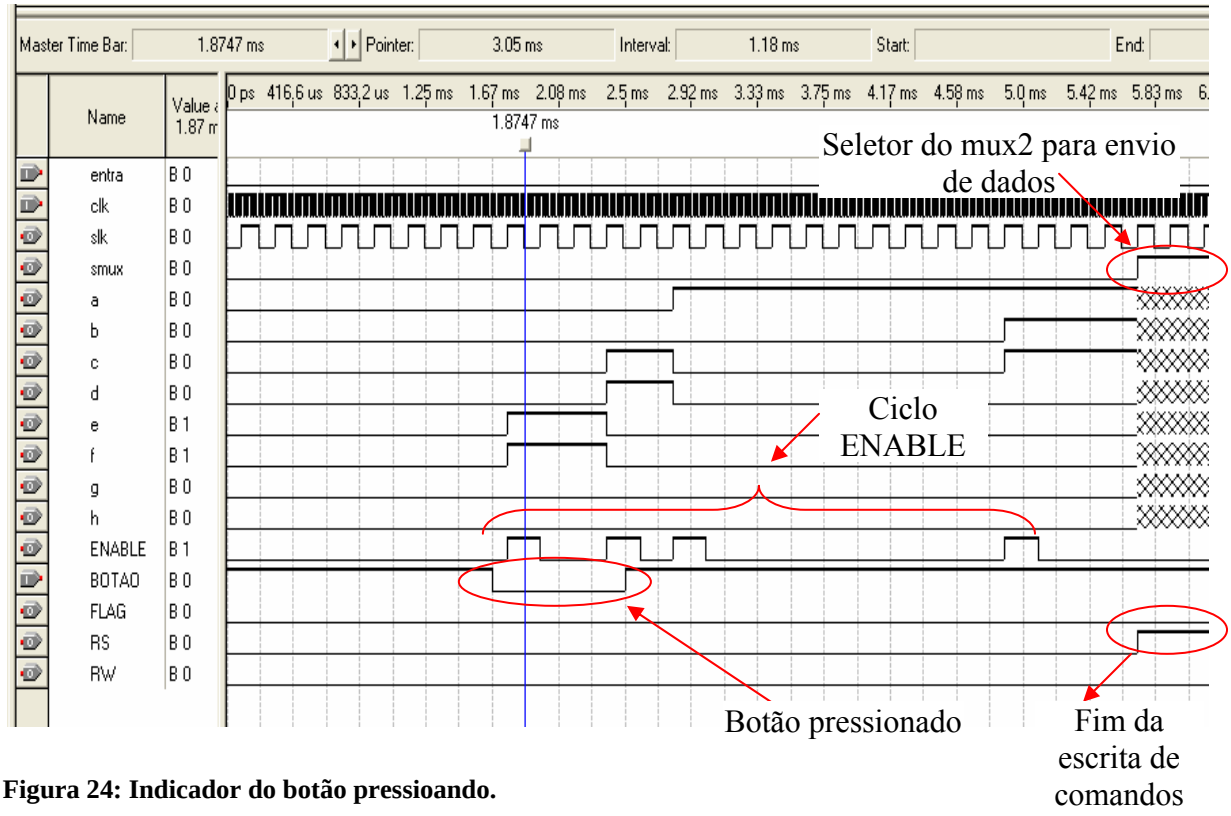


Figura 24: Indicador do botão pressioando.

5.1.2 Inicialização pelo bit de *flag*

A inicialização do modulo LCD utilizando o bit de *flag*, é utilizado para a limpeza da tela de cristal liquido, por uns instantes a tela do LCD será literalmente desligada, ou melhor, será limpa, mas como o processo de limpeza ocorre em aproximadamente 10ms, é praticamente impossível o olho humano perceber o LCD piscando.

A visão humana somente consegue perceber um objeto piscando, se este estiver oscilando a uma freqüência menor que 24 Hz, um exemplo disso é que os televisores convencionais utilizam uma freqüência de 24 a 30Hz. O modulo LCD deste trabalho tem uma oscilação de aproximadamente de 100Hz e neste caso o usuário somente perceberá a mudança dos décimos de segundo no LCD.

A figura 25 mostra os resultados do simulador para o bit de *flag*.

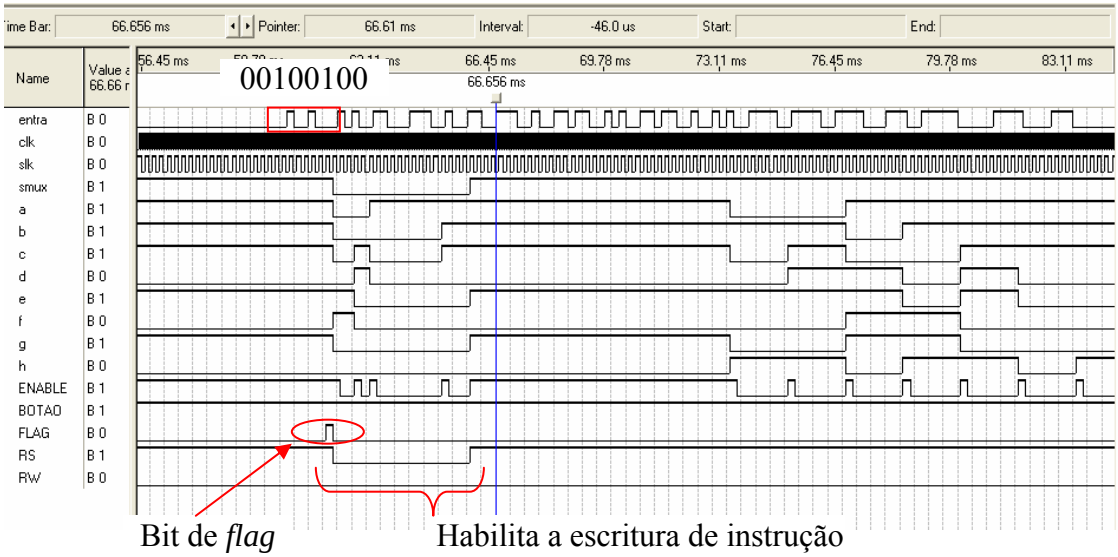


Figura 25: Indicador de flag.

5.2 ESCRITURA DOS DADOS

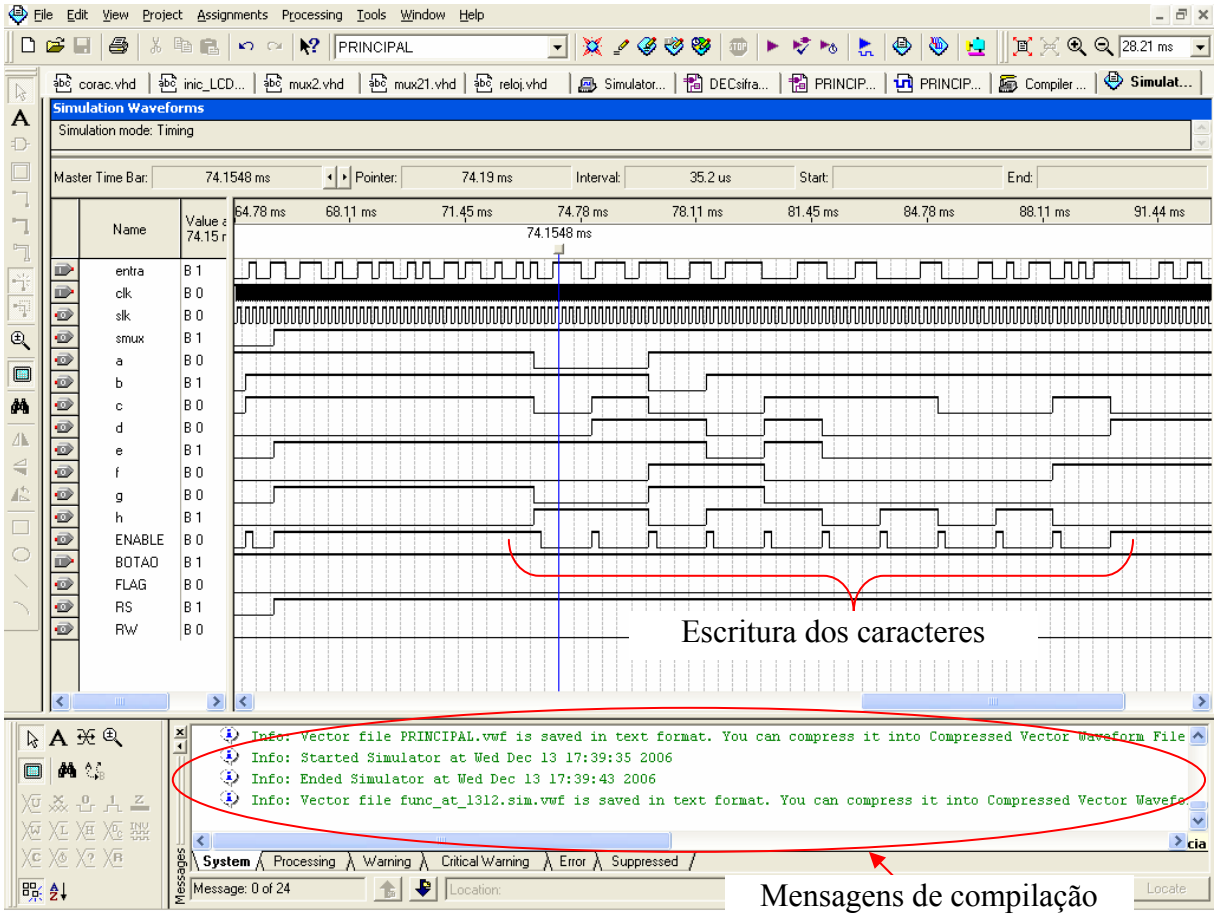


Figura 26: Simulação geral do projeto.

6 CONCLUSÕES

O presente trabalho teve como objetivo o desenvolvimento de uma interface de comunicação entre distintos dispositivos, que se comunicam de forma diferente, e com este trabalho foi possível realizar o entendimento entre as partes envolvidas.

Ao utilizar dispositivos lógicos programáveis logra-se o resultado pretendido e ao mesmo tempo é desenvolvido um circuito elétrico, que não é possível de observar visualmente, pois está internamente no dispositivo lógico, mas reduz-se enormemente o tamanho físico do circuito, reduzindo também consumo de potencia e custos com componentes funcionais ao projeto.

Com isso o maior legado que deixa este TCC, é a possibilidade que os CPLD's têm de adaptar-se a qualquer função que é requerida pelo usuário ou projetista de circuitos digitais, também demonstra que na área das telecomunicações tem suas utilidades em projetos, adaptando-se aos distintos protocolos de comunicação.

As maiores dificuldades encontradas na realização deste trabalho, radica no entendimento da linguagem VHDL e como seus elementos concorrentes funcionam todos ao mesmo tempo e não de forma ordenada e seqüencial, como em outras linguagens de software de programação existentes no mercado.

6.1 TRABALHOS FUTUROS

As modificações que possivelmente poderão ser feitas estão relacionadas com a estética de visualização, pois é possível agregar textos adicionais, na visualização da hora UTC, bem como indicações aos usuários, mensagens de localização ou mensagens de disponibilização satelital, entre outras.

Na proposta do LQE, para utilizar a hora UTC do satélite como sincronismo interno dos equipamentos registradores de eventos e de detecção de faltas em sistemas elétricos, possivelmente não será o utilizado o modulo de visualização para fins de mostrar a hora do GPS, com isso as modificações estéticas não serão necessárias.

Outras modificações podem ser feitas para agregar-lhe mais funcionalidades ao projeto, que previamente devem ser definidas pelo LQE, como sistemas de alarmes ou programações pré-estabelecidas para um determinado horário.

REFERÊNCIAS

ALTERA. **University Program UP2 Education Kit – User Guide**. San Jose, CA: Altera Corporation, 2004.

BAO, J.; TSUI, Y. **Fundamentals of Global Positioning System Receivers.** New Jersey: John Wiley & Sons Ltd, 2005.

BETKE, K. **The NMEA 0183 Protocol.** New Bern, USA: NMEA, Org., 2001.

DUECK, R. K. **Digital design with CPLD Applications and VHDL 2e.** New York: Thomson Delmar Learning, 2005.

ECHANOVE, M. A. A. **Lenguajes de Alto Nivel para Diseño de Circuitos Integrados Digitales.** Sevilla: DIEESI Universidad de Sevilla, 2001.

IME, **Display LCD.** Disponível em: <http://aquarius.ime.eb.br/~pinho/micro/apostila/lcdport.pdf>. Acesso em: 13 out. 2006.

PERRY, D. L. **VHDL: Programming by Example Fourth Edition.** United States of America: McGraw-Hill Companies, Inc., 2002.

QUARTUS II. **Quartus II Version 5.1 Handbook**. San Jose, CA: Altera Corporation, 2005.

RHOD, E. L. **Desenvolvimento de um controlador de display LCD utilizando componentes lógicos programáveis.** Disponível em: <http://www.inf.ufrgs.br/~elrhod/main.pdf>. Acesso em: 17 nov. 2006.

SAMSUNG, E. **KS0066U (16COM / 40SEG Driver & Controller For Dot Matrix LCD).** Disponível em: <http://micro8051.com.sapo.pt/diversos/ks0066u.pdf>. Acesso em: 4 out. 2006.

SANJOSE, N. **FV-12 GPS Receiver Module – User's Guide**. Taiwan: Taipei Hsien, 2005.

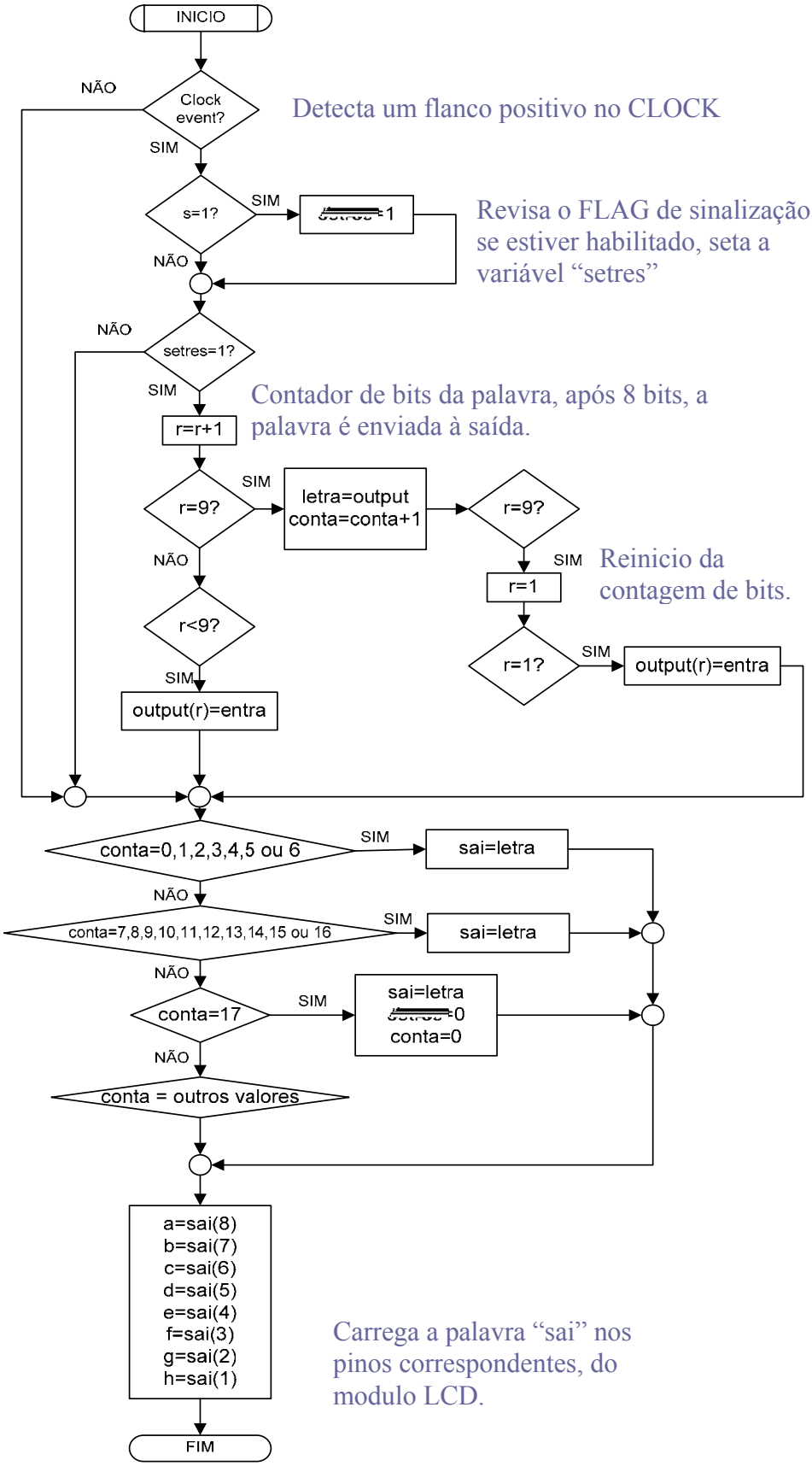
SiRF TECHNOLOGY, Inc. **GSP2e/LP Family – Datasheet**. Taiwan: Keelung Road Taipei, 2001.

WIKIPÉDIA, **VHDL.** Disponível em: <http://pt.wikipedia.org/wiki/VHDL>. Acesso em: 5 nov. 2006.

WINSTAR, **WH2004A/B.** Disponível em: <http://www.winstar.com.tw/detail_view.php?pd_num=WH&pd_nu2=2004A/B>. Acesso em: 4 out. 2006.

APÊNDICES

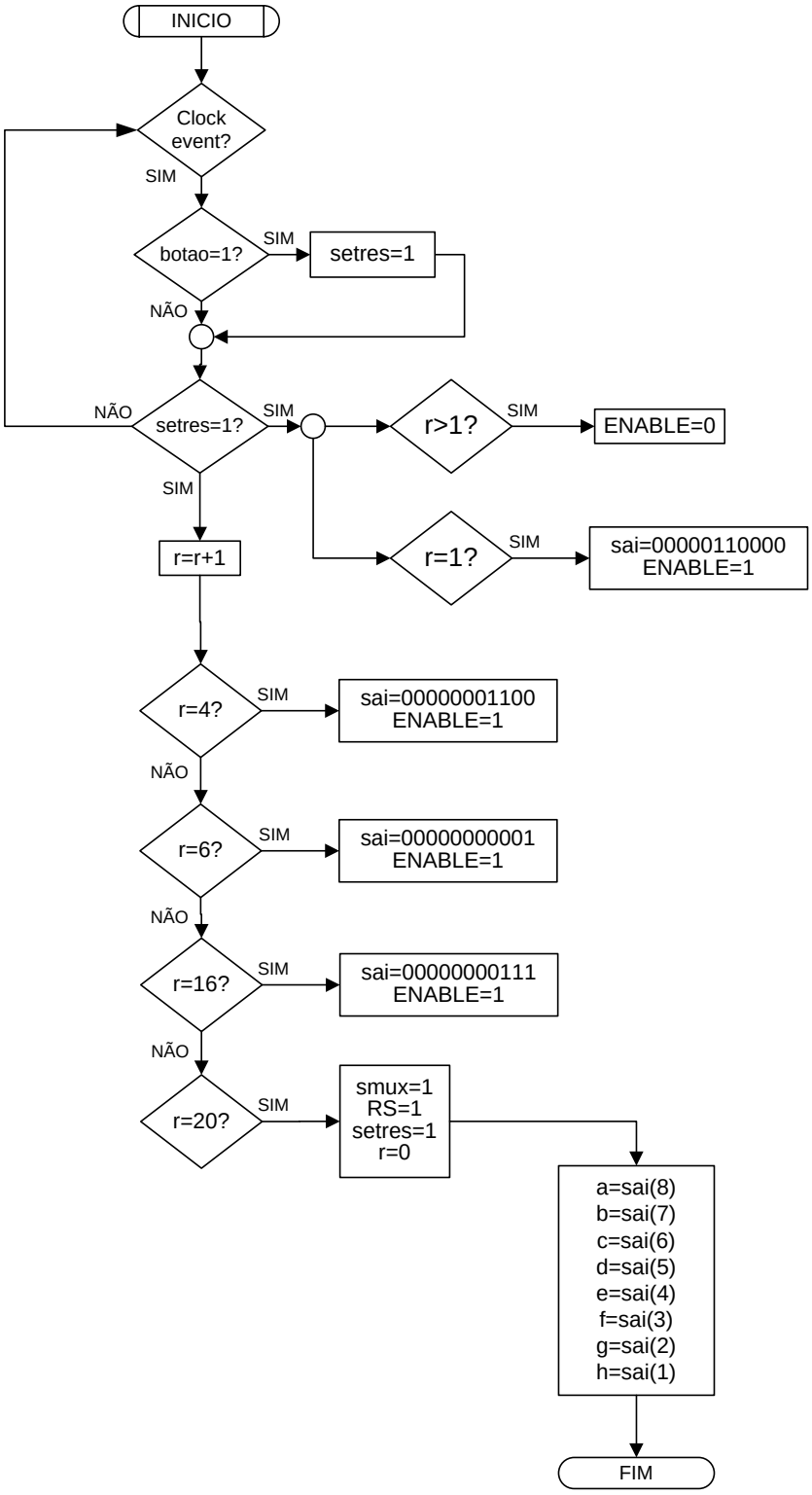
APÊNDICE A – FLUXOGRAMA DO MÓDULO CORAC



APÊNDICE B – CÓDIGO FONTE DO MODULO CORAC

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
ENTITY corac IS
PORT( clk: IN std_logic; entra: IN std_logic; s: IN std_logic; a, b, c, d, e, f, g, h,ena : OUT std_logic );
END corac;
ARCHITECTURE cmc OF corac IS
SIGNAL sai: std_logic_vector (8 DOWNTO 1);
SIGNAL output: std_logic_vector (8 DOWNTO 1);
SIGNAL letra: std_logic_vector (8 DOWNTO 1);
SIGNAL enbl: std_logic;
BEGIN
PROCESS ( clk,s)
VARIABLE r: INTEGER:=0;
VARIABLE conta: INTEGER:=0;
VARIABLE setres: std_logic:='0';
VARIABLE ven: std_logic:='0';
BEGIN
        IF (clk'EVENT AND clk='1')
        THEN
                IF (s='1') THEN setres:='1'; END IF;
                IF (setres='1')
                THEN
                r:=r+1;
                        IF (r=9) THEN
                        letra <= output;
                        conta:=conta+1;
                                IF conta > 6 THEN ven:='1'; END IF;
                                IF (r=9) THEN r:=1;
                                        IF (r=1) THEN output(r) <= entra; END IF;
                                END IF;
                        ELSIF (r<9)
                        THEN output(r)<= entra;
                        IF conta > 6 THEN ven:='0'; END IF;
                        END IF;
                END IF;
        END IF;
        CASE conta IS
                WHEN
                0 | 1 | 2 | 3 | 4 | 5 | 6 =>
                sai <= UNAFFECTED;
                WHEN
                7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 => sai <= letra;
                WHEN 17 => sai <= letra; setres:='0'; conta:=0;
                WHEN OTHERS =>  sai <= UNAFFECTED;
        END CASE;
        enbl<=ven;
END PROCESS;
a <= sai(8);
b <= sai(7);
c <= sai(6);
d <= sai(5);
e <= sai(4);
f <= sai(3);
g <= sai(2);
h <= sai(1);
ena<=enbl;
END cmc;
```

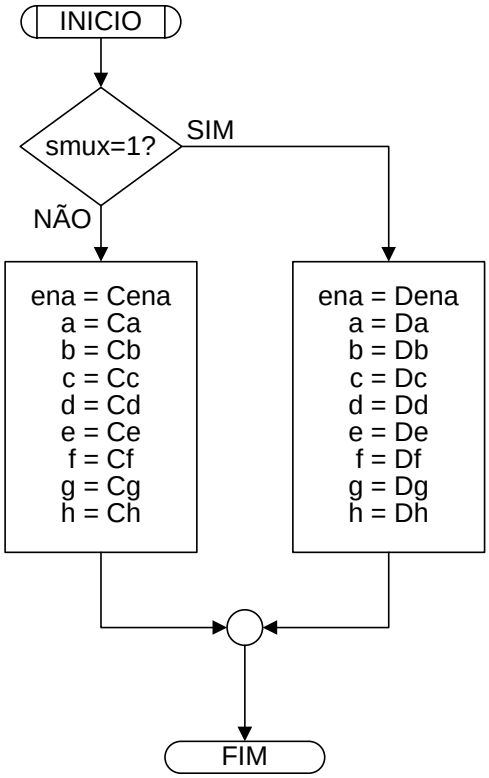
APÊNDICE C – FLUXOGRAMA DO MODULO INI_LCD



APÊNDICE D – CÓDIGO FONTE DO MODULO INI_LCD

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
ENTITY inic_LCD IS
PORT(
clk: IN std_logic;
botao: IN std_logic;
Cena, RS, RW, a, b, c, d, e, f, g, h,smux : OUT std_logic );
END inic_LCD;
ARCHITECTURE inlcd OF inic_LCD IS
SIGNAL sai: std_logic_vector (11 DOWNTO 1);
SIGNAL enbl: std_logic;
BEGIN
PROCESS ( clk, botao)
VARIABLE r: INTEGER:=0;
VARIABLE setres: std_logic:='0';
BEGIN
	IF (clk'EVENT AND clk='1')
	THEN
		IF (botao = '1') THEN setres:='1'; END IF;
		IF (setres='1')
		THEN
		r:=r+1;
		IF r > 1 THEN enbl<='0'; END IF;
		IF r = 1 THEN enbl<='1'; sai<="00000110000"; END IF;
			IF (r=4) THEN
			 sai<="00000001100";
			enbl<='1';
			ELSIF (r=6) THEN
			sai<="00000000001";
			enbl<='1';
			ELSIF (r=16) THEN
			 sai<="00000000111";
			enbl<='1';
			ELSIF (r=20) THEN
			 sai(11)<='1';
			sai(9)<='1';
			setres:='0';
			r:=0;
			END IF;
		END IF;
	END IF;
END PROCESS;
Cena <= enbl;
smux <= sai(11);
RW <= sai(10);
RS <= sai(9);
a <= sai(1);
b <= sai(2);
c <= sai(3);
d <= sai(4);
e <= sai(5);
f <= sai(6);
g <= sai(7);
h <= sai(8);
END inlcd;
```

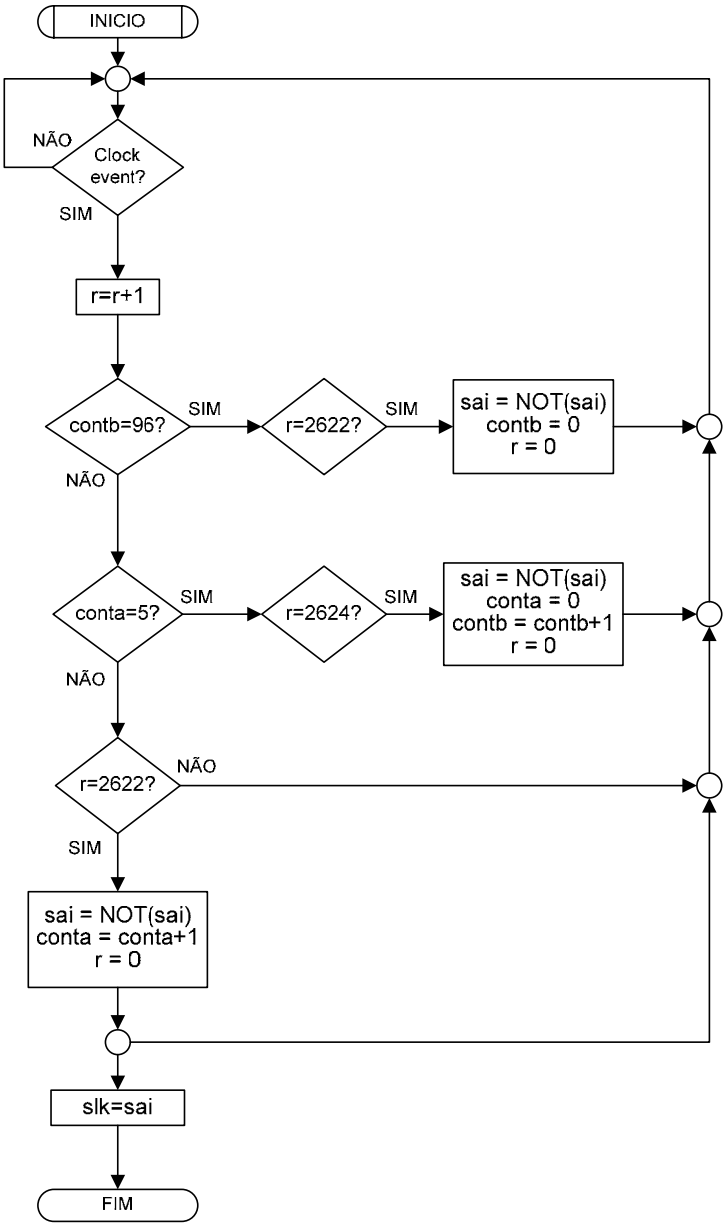
APÊNDICE E – FLUXOGRAMA DO MODULO MUX2



APÊNDICE F – CÓDIGO FONTE DO MODULO MUX2

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
ENTITY mux2 IS
PORT(
smux: IN std_logic;
Dena, Da, Db, Dc, Dd, De, Df, Dg, Dh: IN std_logic;
Cena, Ca, Cb, Cc, Cd, Ce, Cf, Cg, Ch: IN std_logic;
ena, a, b, c, d, e, f, g, h : OUT std_logic
);
END mux2;
ARCHITECTURE mx2 OF mux2 IS
BEGIN
PROCESS (smux)
BEGIN
        IF (smux = '1')
        THEN
                ena <= Dena;
                a <= Da;
                b <= Db;
                c <= Dc;
                d <= Dd;
                e <= De;
                f <= Df;
                g <= Dg;
                h <= Dh;
        ELSE
                ena <= Cena;
                a <= Ca;
                b <= Cb;
                c <= Cc;
                d <= Cd;
                e <= Ce;
                f <= Cf;
                g <= Cg;
                h <= Ch;
        END IF;
END PROCESS;
END mx2;
```

APÊNDICE G – FLUXOGRAMA DO MODULO RELOJ



APÊNDICE H – CÓDIGO FONTE DO MODULO RELOJ

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
ENTITY reloj IS
PORT(
clk: IN std_logic;
slk : OUT std_logic );
END reloj;
ARCHITECTURE rlj OF reloj IS
SIGNAL sai: std_logic:='0';
BEGIN
PROCESS ( clk)
VARIABLE r: INTEGER:=0;
VARIABLE conta: INTEGER:=0;
VARIABLE contb: INTEGER:=0;
BEGIN
        IF (clk'EVENT AND clk='1')
        THEN
        r:=r+1;
            IF contb = 96
            THEN
                IF r = 2622
                THEN
                sai<=NOT(sai);
                contb:=0;
                r:=0;
                END IF;
            ELSIF conta = 5
            THEN
                IF r = 2624
                THEN
                sai<=NOT(sai);
                conta:=0;
                contb:=contb+1;
                r:=0;
                END IF;
            ELSE
                IF r = 2622
                THEN
                sai<=NOT(sai);
                conta:=conta+1;
                r:=0;
                END IF;
            END IF;
        END IF;
END PROCESS;
slk <= sai;
END rlj;
```