Técnicas para Multiplexação de Sinais de TV Digital no Padrão ISDB-TB

Juliano Silveira Ferreira, Luciano Leonel Mendes, Geraldo Gil Ramundo Gomes

***Resumo*— O objetivo deste artigo é apresentar as tecnologias utilizadas na implementação de um modelo proposto de um dos equipamentos mais importantes dentro do sistema de transmissão de TV Digital, que é o Multiplexador do sistema ISDB-TB. As normas de TV Digital estabelecem o que o Multiplexador deve realizar, mas não oferecem uma referência a respeito das tecnologias que podem ser empregadas na sua implementação. Este trabalho oferece uma sugestão de tecnologias que podem ser utilizadas para tal finalidade.**

***Palavras-Chave*— Multipexador, ISDB-TB, FPGA, sistemas embarcados.**

***Abstract*— This article aims to provide the technologies used in implementation of a proposed model a Multiplexer for ISDB-TB system, that is one of the most important equipment in a DTV system. Standards of Digital TV establish what the Multiplexer needs to perform, but don´t offer a reference about the technologies that can be used in Multiplexer´s implementation. This paper offers a suggestion of technologies that can be used for this purpose.**

***Keywords*— Multiplexer, ISDB-TB, implementation, FPGA, embedded systems.**

I. Introdução

Um processo de evolução deve ocorrer dentro das emissoras para que estas se adequem à nova era da televisão que se inicia com as transmissões de TV Digital em rede aberta no país. Investimentos em novos equipamentos, capacitação de profissionais, adequação de estúdios e melhorias em cenários devem estar envolvidas nesse processo de evolução. Em meio ao processo de implantação da TV Digital surge uma oportunidade para a indústria nacional também evoluir. A TV Digital deve ser encarada pela indústria brasileira como sendo uma oportunidade de desenvolvimento indústrial e tecnológico no país. Esta oportunidade é gerada por uma crescente demanda de equipamentos, treinamento e suporte. A indústria nacional pode favorecer-se à medida que ofereça equipamentos de qualidade e de menor preço que as empresas estrangeiras e a medida que ofereça prontamente aos clientes, o suporte e treinamento necessários. A possibilidade de outros países, principalmente na América Latina, adotarem o mesmo padrão de TV Digital que o Brasil aumenta ainda mais a oportunidade de crescimento da indústria nacional.

Devido a sua importância dentro do sistema de transmissão, o Multiplexador pode ser encarado como uma boa oportunidade de investimento da indústria, uma vez que é um equipamento necessário nas geradoras de TV e nas retransmissoras que desejem realizar inserção de programação local. Neste trabalho, tem-se a apresentação de um modelo proposto para implementação do Multiplexador e a apresentação das tecnologias que foram utilizadas. A intenção do trabalho é a de oferecer um ponto de partida para discussões a respeito das tecnologias que podem ser utilizadas na implementação do Multiplexador.

Este trabalho está estruturado da seguinte forma: a sessão II apresenta o Multiplexador e sua integração ao sistema de transmissão, enquanto a sessão III mostra o modelo de implementação utilizado; a sessão IV descreve as ferramentas e tecnologias utilizadas na implementação de um protótipo do Multiplexador e a sessão V fecha o trabalho com a conclusão.

Juliano Silveira Ferreira, Luciano Leonel Mendes, Geraldo Gil Ramundo Gomes, Inatel Competence Center, Inatel, Santa Rita do Sapucaí, MG, Brasil, E-mails: silveira@inatel.br, lucianol@inatel.br, ge@inatel.br.

II. Apresentação do Multiplexador

O Multiplexador funciona como uma interface entre o estúdio e o transmissor da emissora de TV, como pode ser visualizado na Figura 1. Ele recebe em sua entrada os vários fluxos ou TS's (*Transport Stream*) que a emissora deseja transmitir, os multiplexa com os parâmetros de configuração do transmissor (como modo de transmissão, intervalo de guarda, taxa de codificação, modulação e comprimento do entrelaçador temporal) e dados adicionais como guia de programação ou dados de interatividade. O fluxo entregue na saída do Multiplexador, chamado BTS (*Broadcasting Transport Stream*), é repassado ao transmissor [1].

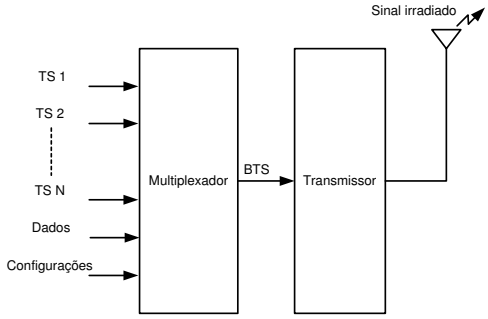

Fig. 1. O sistema de transmissão utilizando o Multiplexador.

III. Modelo para Implementação do Sistema de Multiplexação

A necessidade do Multiplexador realizar diversas tarefas simultâneas, que exigem processamento dedicado e precisão com relação as taxas e medidas de tempo, foram fatores decisivos na opção pelo FPGA (*Field Programmable Gate Array*) como plataforma de desenvolvimento. Por exemplo, a multiplexação de TS envolve a recepção e processamento

paralelo de diferentes TS além da execução de processo de correção de PCR (*Program Clock Reference*), que requer precisão de relógio e de medidas de tempo. O FPGA foi escolhido pelo fato desta tecnologia oferecer uma plataforma de processamento que é um *hardware* dedicado ao projeto desenvolvido, diferentemente, por exemplo, do desenvolvimento de uma solução em *software*, que teria de dividir a capacidade de processamento da CPU (*Central Processing Unit*) entre as requisições do sistema operacional, dos periféricos e do próprio *software* do projeto.

A principal linguagem de programação utilizada para implementação do projeto foi a linguagem VHDL (*Very High Speed Integrated Circuit Hardware Description Language*) [2] associada a códigos escritos em linguagem C. As ferramentas de desenvolvimento utilizadas foram o Quartus II [3] e SOPC Builder [4], ambos da empresa Altera [5].

Com relação a configuração do Multiplexador, partiu-se da idéia inicial de possibilitar que o mesmo pudesse ser configurado via rede. Buscando a praticidade de não ter de instalar nenhum *software* no PC para a comunicação com o equipamento, optou-se por criar uma interface embarcada. Neste caso, o usuário precisa somente digitar o endereço IP do equipamento em um navegador web a partir de um PC qualquer conectado na mesma rede do equipamento, para conseguir acessá-lo e configurá-lo. O fato de não se ter de instalar um *software* pode evitar problemas com relação a compatibilidade do software com o PC, como por exemplo conflito com outros programas instalados, incompatibilidade com diferentes *hardwares* e ou versões de sistema operacional.

O modelo proposto para implementação do Multiplexador pode ser visualizado na Figura 2 [1].

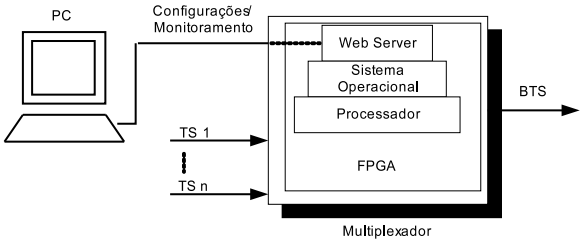

Fig. 2. Diagrama do modelo do Multiplexador implementado.

No modelo proposto, a interface com o usuário é construída utilizando linguagem HTML, que precisa ser suportada por um servidor Web. O servidor Web, por sua vez, funciona sobre um sistema operacional residente em uma CPU. No modelo implementado optou-se por utilizar uma CPU embarcada dentro do FPGA evitando-se, desta forma, a utilização de um processador externo que implicaria em aumento de custo do equipamento.

O processador utilizado é o Nios II. Este é um processador de propósito geral que foi desenvolvido pela Altera para ser embarcado dentro de seus FPGA´s [6].

O Multiplexador deve inserir no seu fluxo de saída novas tabelas compatíveis com o TS MPEG-2 e com sistema ISDB-TB. Os principais multiplexadores encontrados no mercado desenvolvidos em *hardware* necessitam de um *software* externo para geração destas tabelas. Neste *software*, o usuário normalmente deve cadastrar manualmente todas as informações necessárias para o correto funcionamento do sistema.

Na implementação em questão, a geração das principais tabelas do sistema, como a PAT (*Program Association Table*), PMT (*Program Map Table*), NIT (*Network Information Table*) e TOT (*Time Offset Table*) é realizada de maneira automática pelo MUX. Ele se encarrega de analisar os fluxos de entrada, extrair as informações de configuração do sistema de transmissão e gerar as tabelas necessárias. Uma grande vantagem desta abordagem é a facilidade de operação do equipamento e o fato de se ter uma grande versatilidade nas alterações do sistema, como reconfigurar algum codificador de vídeo ou alterar alguma informação do sistema de transmissão. Assim que detectada qualquer alteração em alguns destes parâmetros, novas tabelas são geradas levando em consideração sempre as últimas informações.

A geração destas tabelas é uma tarefa complexa de ser implementada utilizando a linguagem VHDL devido a necessidade de se trabalhar com manipulação de vários vetores de informação em paralelo para gerar um vetor final de saída. Optou-se então pelo desenvolvimento da geração das tabelas utilizando a linguagem C. Estes códigos são executados pela CPU embarcada no FPGA.

IV. Técnicas e Ferramentas para Implementação do Multiplexador

A seguir apresenta-se uma breve descrição das tecnologias, linguagens e *softwares* utilizados no processo de desenvolvimento do protótipo do Multiplexador.

A. Hardware utilizado

A Figura 3 apresenta uma foto do *hardware* utilizado para a implementação do protótipo do Multiplexador. Foi utilizado como base um *kit* de desenvolvimento da Altera (Nios II Development Kit - EP2S60) associado a uma placa de 4 entradas e 2 saídas ASI desenvolvida pela empresa Linear Equipamentos Eletrônicos S/A. A escolha pelos FPGAs da Altera se deve ao fato deste dispositivo já ser empregado por esta empresa em outros equipamentos. Como o custo do FPGA diminui de acordo com a quantidade de peças compradas, optou-se pela utilização de FPGAs semelhantes aos já utilizados dentro da empresa, visando menor custo final do equipamento. A implementação a partir de um *kit* de desenvolvimento foi utilizada para que, pelo menos num primeiro momento, não houvesse preocupação com o desenvolvimento de placas de circuito impresso e que problemas relacionados ao *hardware* utilizado fossem minimizados.

O FPGA empregado neste *kit* é o modelo EP2S60F672 pertencente a família Stratix II da Altera; possui 60.440 elementos lógicos e 2.544.192 bits de memória RAM [7].

B. Tecnologia FPGA

O primeiro FPGA comercializado foi desenvolvido pela empresa Xilinx Inc. no ano de 1983; consiste de um dispositivo

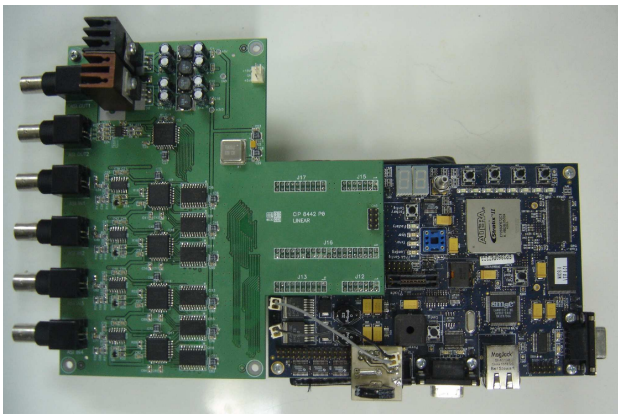

Fig. 3. Foto do *hardware* utilizado na implementação.

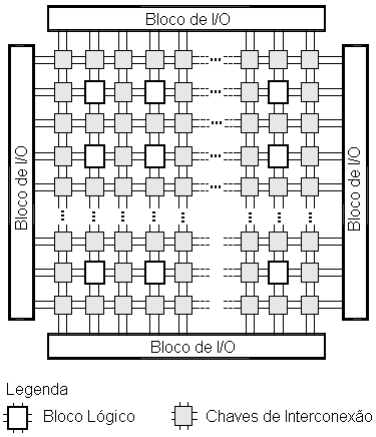

Fig. 4. Estrutura básica de formação de um FPGA.

lógico programável (*Programmable Logic Device* - PLD) que suporta a implementação de circuitos lógicos relativamente grandes [8].

O FPGA se destaca na implementação de circuitos digitais, oferecendo uma boa relação custo/benefício. Ele oferece maior flexibilidade que microprocessadores de uso geral e menor custo que os circuitos integrados de aplicação específica (*Application Specific Integrated Circuit* - ASIC) [9]. Normalmente as aplicações implementadas em FPGAS são mais lentas e consomem mais energia que as aplicações implementadas em um ASIC. Porém, o custo envolvido na fabricação de um ASIC só é justificável em caso de produção em larga escala. O tempo envolvido no desenvolvimento de um ASIC também é maior que o tempo gasto em uma aplicação desenvolvida para um FPGA [10].

O FPGA é composto por uma matriz de blocos ou elementos lógicos; cada um desses elementos lógicos é capaz de realizar pequenas funções e operações lógicas que podem ser programados ou configurados com relação à que função lógica irão empenhar. Os elementos lógicos são interconectados entre si de maneira a se obter o resultado lógico final desejado em cada projeto implementado.

Pode-se visualizar na Figura 4 a estrutura básica de formação de um FPGA, que é composto por três tipos principais de recursos: os blocos lógicos, os blocos de entrada e saída (I/O), e as chaves programáveis de interconexão [8],[9],[10].

Os blocos lógicos são organizados em linhas e colunas e as chaves de interconexão programáveis são dispostas de forma a possibilitar a conexão dos blocos lógicos da maneira que for conveniente, além de conectar estes aos blocos de I/O. Quando se implementa um circuito em FPGA, os blocos lógicos são configurados de forma a realizar as operações lógicas necessárias e as chaves de interconexão estabelecem as conexões entre os blocos de maneira a se obter o resultado lógico final requerido pelo projeto.

C. VHDL

As duas principais linguagens de descrição de *hardware* mais utilizadas são o VHDL e o Verilog. A seguir estão algumas das diferenças entre as duas linguagens que contribuíram para a escolha do VHDL como sendo a linguagem utilizada na implementação do Multiplexador. A linguagem VHDL necessita de mais linhas de código para descrever uma funcionalidade do que um código Verilog equivalente. Esta é uma característica intencional da linguagem VHDL, que foi criada com intuito de servir também como parte da documentação de um projeto de *hardware*. E por fim, a linguagem possui uma semântica para simulação menos ambígua. A ambiguidade do Verilog pode fazer com que resultados diferentes sejam encontrados ao se utilizar diferentes ferramentas de simulação, o que não deve acontecer em uma simulação de um código VHDL [11].

A Linguagem VHDL foi desenvolvida pelo Departamento de Defesa dos Estados Unidos da América (DARPA). Eles buscavam desenvolver uma linguagem de descrição de *hardware* que forçasse o desenvolvedor a escrever um código mais estruturado e compreensível para que, dessa forma, o próprio código servisse como um documento de especificação do projeto [12],[13]. A linguagem VHDL foi padronizada pelo IEEE (*Institute of Electrical and Electronic Engineer*) em 1987 através da criação do padrão IEEE 1076-1987; em 1993 foi publicada uma versão revisada deste padrão, que deu origem ao padrão IEEE 1076-1993, que é o mais utilizado atualmente [12].

Uma linguagem de descrição de *hardware* é uma forma de se descrever ou especificar, através de um programa, o comportamento de um circuito. Ela deve descrever o que o circuito faz e de que maneira ele o faz. O *hardware* pode se referir tanto a sistemas complexos, como um microprocessador, quanto a sistemas simples, como uma porta lógica. A linguagem VHDL abrange diversas as escalas de aplicações podendo ser utilizada para descrever o funcionamento de *hardware* em geral [13].

D. Ambiente de desenvolvimento

O *software* utilizado para implementar e simular as etapas desenvolvidas em VHDL foi o Quartus II, desenvolvido pela Altera. Ele foi escolhido por ser uma ferramenta que integra todas as etapas de desenvolvimento de projetos em FPGA

(descrição, síntese e posicionamento e interligação) [12], além de oferecer ambiente de simulação e de já ter integrado ferramentas para criação de processadores embarcados. Outro fator positivo é que o Quartus foi desenvolvido pela própria Altera, que é o fabricante do FPGA utilizado. Pode-se, através do ambiente de simulações, estabelecer sinais à entrada do projeto e analisar os resultados obtidos na saída do mesmo. O Quartus gera um arquivo final de gravação do projeto no FPGA e oferece a ferramenta para a gravação do mesmo.

Durante o ciclo de desenvolvimento, a descrição do *hardware* deve se tornar precisa o bastante para que seja possível a transformação do código em *hardware*. A transformação, automática, da descrição menos detalhada (código desenvolvido) numa informação mais mais precisa e elaborada é a chamada de síntese. Algumas ferramentas de síntese são capazes de mapear a linguagem de descrição de *hardware* diretamente em determinados circuitos integrados [13], como é o caso da ferramenta Quartus II, que possibilita a síntese do projeto direcionada à tecnologia FPGA.

Para a criação do processador embarcado foi utilizado o *software* SPOC Builder que é uma ferramenta do Quartus II; a finalidade desta ferramenta é a de possibilitar a geração de sistemas integrados em um único dispositivo ou SOPC (*system-on-a-programmable-chip*) [4]. Utilizando o SOPC Builder pode-se especificar o processador NIOS II, bem como seus periféricos a serem implementados dentro do FPGA.

Para o desenvolvimento de aplicações a serem executadas diretamentes pelo processador, foi utilizada outra ferramenta do Quartus II, o NIOS II IDE (*Integrated Development Environment*). O NIOS II IDE é o sistema integrado para o desenvolvimento de *software* para o NIOS, que inclui o IDE Eclipse e compilador C/C++;

E. Processador Nios II

O processador NIOS II é um processador RISC (*Reduced Instruction Set Computer*) implementado usando linguagem de descrição de *hardware*, para ser sintetizado dentro do FPGA. Ele é classificado como sendo um *soft-core processor* já que o processador não está fixo no silício, ou seja, não foi implementado durante a confecção da pastilha de silício do FPGA. Ele é configurável, uma vez que pode-se selecionar as características do processador bem como seus periféricos. Através da ferramenta SOPC Builder pode-se adicionar diversos periféricos padrões, que já estão prontos, como interfaces de comunicação serial, interfaces com memórias, relógios (*timers*) e outros periféricos de propósito geral. Além disso, há a possibilidade de se criar periféricos personalizados e instruções também personalizadas (é possível fazer uso de até 256 instruções personalizadas) [6]. Estas facilidades oferecidas permitem uma grande versatilidade e flexibilidade durante o desenvolvimento de projetos [6]. Um fator que merece destaque é que o processador NIOS possui um conjunto de instruções de 32 bits e barramento de dados e espaço de endereçamento também de 32 bits; possui 32 registradores de propósito geral, 5 registradores de controle, além de possuir 32 fontes de interrupção externa e possui também Unidade de Gerenciamento de Memória (MMU - *Memory Management Unit*) opcional, para suportar sistemas operacionais com MMU. O desempenho do processador pode chegar a até 250 DMIPS (*Dhrystone million instructions per second*) [6].

O *software* SOPC Builder possibilita escolher 3 possíveis configurações padrão para a CPU do NIOS: econômica (*economic*), padrão (*standard*) e rápida (*fast*). A CPU Nios II/e é a que utiliza a menor quantidade de recursos (*economic*) do FPGA, mas em contrapartida é a que possui menor desempenho. Já a CPU Nios II/f apresenta a melhor desempenho (*fast*) em detrimento do uso de maior número de recurso do FPGA. A CPU Nios II/s (*standard*) oferece um meio termo em relação o desempenho e uso de recurso.

A opção de se utilizar a CPU embarcada dentro do FPGA faz com que não seja necessário o investimento na compra de um processador dedicado, e diminui, consequentemente, o número de componentes na placa de circuito impresso, diminuindo também a sua complexidade e preço. Mas a utilização da CPU embarcada no FPGA, em contrapartida, oferece o custo de utilização de lógica do FPGA.

F. Sistema operacional uClinux

A utilização de um sistema operacional compacto e livre de custos foram características que levaram a escolha pelo sistema operacional uCLinux.

O sistema operacional uClinux é uma variante do Linux que foi desenvolvida para ser utilizada em sistemas embarcados (*embedded systems*). Mais especificamente, foi desenvolvido visando o uso em microcontroladores de baixo custo; um fator que contribui para o baixo custo dos microcontroladores é o fato de, comumente, não se oferecer suporte à MMU. O uClinux é, por este motivo, um sistema operacional que não faz uso de MMU [14]-[16]. Enquanto o Linux, fazendo uso da MMU, provê um endereçamento virtual de memória para cada processo a ser executado, o uClinux oferece um único espaço de endereço que é compartilhado entre todos os processos. Isto faz com que o gerenciamento de memória no uClinux fique mais complexo, mas possibilita que ele seja mais compacto [16].

O projeto do uClinux teve início em 1997 e em 1998 teve sua primeira versão disponibilizada publicamente, sendo proposta como um sistema operacional alternativo para o Palm Pilot. Atualmente o uClinux vem ganhando cada vez mais espaço dentre as aplicações embarcadas, sendo visto como um dos mais populares sistemas operacionais para tal aplicação. Uma característica que contribui para este sucesso é o fato dele ser um sistema gratuito e de código aberto (*open-source*) além do fato de ser suportado em diversas plataformas como ARM, ColdFire, NIOS entre outras. O sistema operacional oferece suporte a vários protocolos de redes, incluindo a pilha TCP/IP (*Transmission Control Protocol / Internet Protocol*) e sistemas de arquivos.

V. CONCLUSÕES

Através de testes utilizando transmissores e receptores comerciais pode-se verificar a compatibilidade do Multiplexador

implementado. Foram realizados testes do protótipo em conjunto com transmissores das empresas Eiden e Linear e com receptores das seguintes empresas: *set-top-box* Zinwell e Aiko, receptor para automóveis da Gradiente e receptor USB da Sanwa.

Através do *software* de análise de fluxos, o *StreamXpert* desenvolvido pela empresa Dectek [17], pode-se validar a integridade do fluxo de saída do Multiplexador, o BTS, como pode-se visualizar na Figura 5. Através de testes por mais de 72 horas ininterruptas, pode-se, através também do *StreamXpert*, verificar a estabilidade de funcionamento do Multiplexador.

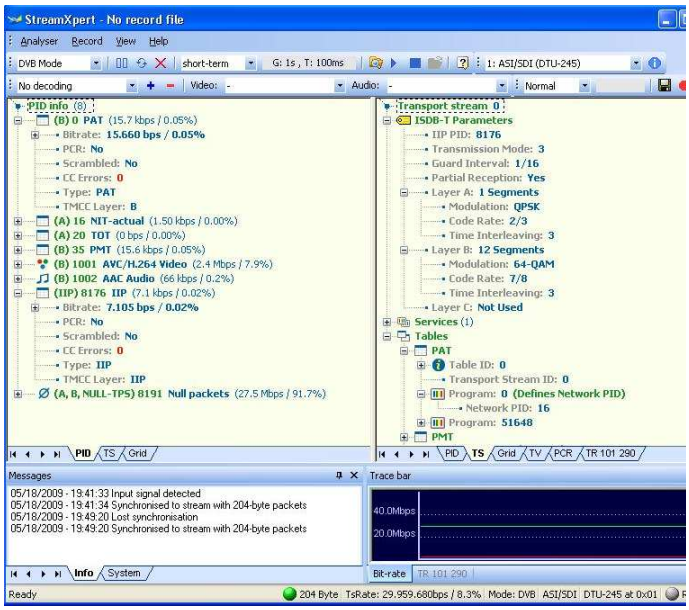

Fig. 5. Análise do BTS de saída do Multiplexador no *software StreamXpert.*

Pode-se concluir que o modelo de implementação adotado e as tecnologias utilizadas se mostraram compatíveis com as necessidades do Multiplexador.

AGRADECIMENTOS

Agradecimentos ao INATEL - Instituto Nacional de Telecomunicações - e à Linear Equipamentos Eletrônicos S/A por oferecerem a estrutura e o suporte necessário para o desenvolvimento deste trabalho. Agradecimento especial ao acadêmico Bruno Borsato de Souza Lima e aos engenheiros Cleomárcio Almeida, Dário Daniel Ribeiro de Morais, Farley Igor Martins Balbino e Rafael Adami Pivoto, que contribuíram de maneira expressiva para esta implementação.

REFERÊNCIAS

[1] Juliano Silveira Ferreira, ”Solução de implementação do multiplexador do sistema ISDB-TB”, Simpósio Brasileiro de Telecomunicações - SBRT, 2008.
[2] Sudhakar Yalamanchili, ”*VHDL Starter’s Guide*”, Prentice Hall, 1998.
[3] Quartus II Handbook Version 9.0. Disponível em: <http://www.altera.com/literature/lit-qts.jsp> Acessado em Abril de 2009.
[4] Quartus II Handbook Version 9.0 Volume 4: SOPC Builder. Disponível em: <http://www.altera.com/literature/lit-sop.jsp> Acessado em Abril de 2009.
[5] Altera Corporation. Disponível em: <http://www.altera.com> Acessado em Abril de 2009.
[6] Nios II Processor Reference Handbook. Disponível em: <http://www.altera.com/literature/lit-nio2.jsp> Acessado em Abril de 2009.
[7] Stratix II Device Family Data Sheet. Disponível em: <http://www.altera.com/literature/hb/stx2/stx2_sii5v1_01.pdf> Acessado em Abril de 2009.
[8] César da Costa, ”Projetando controladores digitais com FPGA”, Editora Novatec, primeira edição, Maio de 2006.
[9] Remy Eskinazi,”FPGAS dinamicamente reconfiguráveis: Fluxo de projeto e vantagens na concepção de circuitos integrados”, Unibratec / UPE-Escola Politécnica.
[10] Jan mendonça Corrêa, ”Arquiteturas em FPGA para comparação de sequências biológicas em espaço linear”, Universidade de Brasília, 2008.
[11] Stephen Bailey, ”Comparison of VHDL, Verilog and SystemVerilog”, Digital Simulation White Paper, Model Technology.
[12] Roberto d’Amore, ”VHDL Descrição e Síntese de Circuitos Digitais”, LTC Editora, 2005.
[13] VHDL Tutorial, Disponível em: <www.vhdl-online.de/tutorial/≫”Acessado em Abril de 2009.
[14] Kimmo Nikkanen, ”uClinux as an embedded solution”, Turku Polytechnic, 2003.
[15] ucLinux - Embedded Linux Microcontroller Project. Disponível em: <http://www.uclinux.org> Acessado em Junho de 2008.
[16] Hyok-Sung Choi, Hee-Chul Yun, ”Context Switching and IPC Performance Comparison between uClinux and Linux on the ARM9 based Processor”, SAMSUNG Tech. Conference.
[17] DekTec. Disponível em: <http://www.dectek.com> Acessado em Abril de 2009.