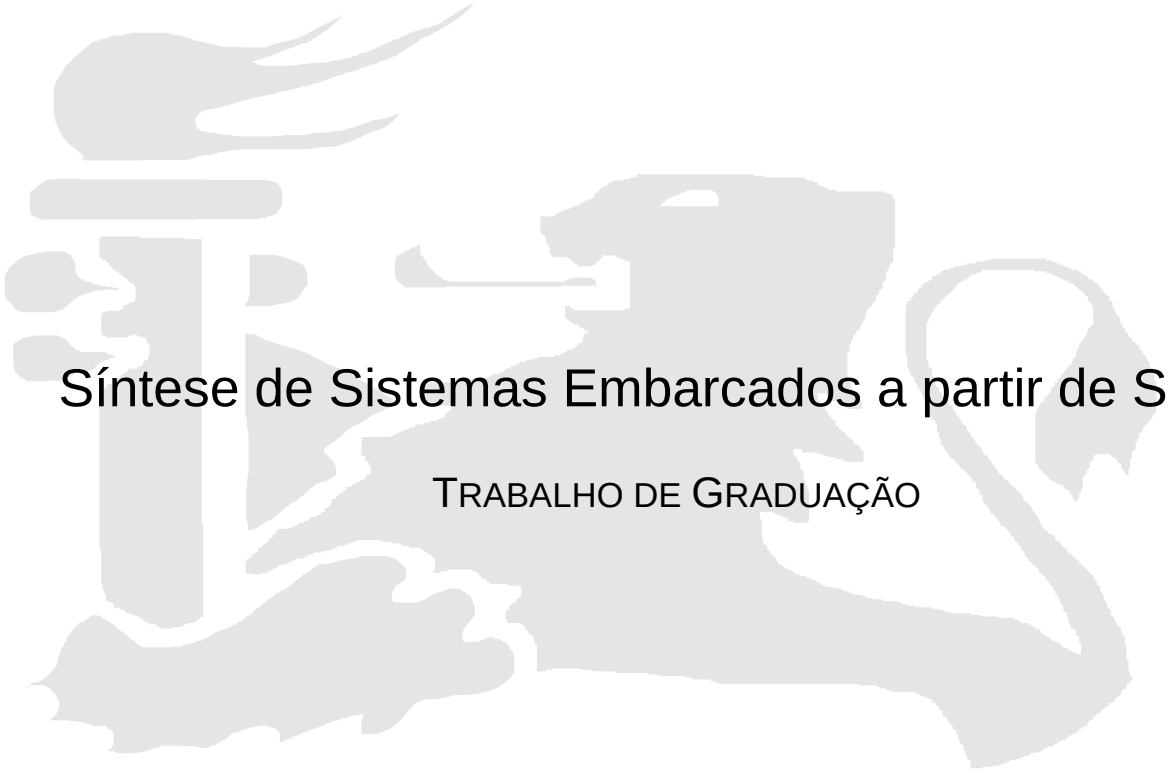
UNIVERSIDADE FEDERAL DE PERNAMBUCO
GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA



Síntese de Sistemas Embarcados a partir de Scripts

TRABALHO DE GRADUAÇÃO

Aluno: Theogenes Ferreira Duarte (tfd@cin.ufpe.br)
Orientador: Cristiano Coelho de Araújo (cca2@cin.ufpe.br)

Recife(PE), junho de 2010

UNIVERSIDADE FEDERAL DE PERNAMBUCO
GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA
2010.1

Síntese de Sistemas Embarcados a partir de Scripts

TRABALHO DE GRADUAÇÃO

Essa monografia apresentada no Centro de Informática da Universidade Federal de Pernambuco representa o trabalho de conclusão de curso em Engenharia da Computação

Aluno: Theogenes Ferreira Duarte (tfd@cin.ufpe.br)
Orientador: Cristiano Coelho de Araújo (cca2@cin.ufpe.br)

Recife(PE), junho de 2010

UNIVERSIDADE FEDERAL DE PERNAMBUCO
GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

Síntese de Sistemas Embarcados a partir de Scripts

TRABALHO DE GRADUAÇÃO

Essa monografia apresentada no Centro de Informática da Universidade Federal de Pernambuco representa o trabalho de conclusão de curso em Engenharia da Computação

Aprovada em ____ de _____ de _____.

Prof. _____

Prof. _____

_____

(Orientador)

Recife(PE), junho de 2010

Agradecimentos

Gostaria, através deste, de agradecer primeiramente a Deus, por ter me dado a oportunidade de lutar pelo meu futuro. Em segundo lugar sou muito grato a minha família, por todo apoio financeiro, moral e psicológico, se não fosse pelos esforços de minha mãe e meu pai, eu nunca teria terminado esse curso. Da mesma forma, também não posso esquecer de Rafa, minha irmã mais velha, que se esforçou muito para que eu pudesse estudar, me motivando nesses 8 anos de Recife, bem como Mayara, minha outra irmã, que também soube me apoiar na dose certa. Realizar um agradecimento para tantas pessoas em uma página não é simples, no entanto, as pessoas que conviveram comigo na Casa do Estudante merecem um lugar especial, pois foram como meus irmãos nessa terra distante. Pessoas como Maurício, Francisco, Marcos, Diógenes, Admário, Jota, e muitos outros residentes. Eu nunca vou esquecer esses caras, foi com o apoio e o companheirismo deles que eu pude enfrentar os momentos difíceis de viver longe de casa. Queria homenagear também meu grande amigo Kelson, sempre sincero, simples, uma pessoa de muito valor que me incentivou bastante. Outra pessoa que tem lugar especial nesse pequeno texto é minha namorada Deide, companheira, amiga, confidente, está comigo desde 2006, me dando força, graças a ela, obtive motivação para superar os desafios nos momentos mais complicados do curso.

Um tópico a parte é o trecho que agradeço aos professores, em especial a Silvio Melo, que foi meu orientador da bolsa de manutenção da PROACAD, foi uma pessoa muito paciente e é um cara de um grande coração. Outro agradecimento especial é para Marcilia Campos, professora rigorosa e muito inteligente, foi depois de ter estudado com ela que entendi o verdadeiro significado de estudar para aprender, também não posso esquecer de José Dias, cara simples, dedicado e sincero, foi um grande professor para mim. Outra grande pessoa foi Cristiano, meu orientador do TG, cara simples também, me abriu as portas com esse projeto e me deu oportunidade de desenvolver uma atividade prazerosa, para terminar gostaria de agradecer também a todos os outros professores, da Área II, do CTG e do CIn, pois sem eles não teria chegado aqui.

Por fim, gostaria de homenagear também meus colegas de curso, como Diego, Emanuel (Papel), Gabriela, Dayse, Thiago Monteiro, Efrem, Ciro, Igino, Eduardo Fonseca, Pyetro, Paulo Fagner,

e todos os outros com quem virei muitas noites, fins de semana e feriados estudando e conversando nos últimos anos que passei aqui em Recife. Se eu fosse mencionar todas as pessoas das quais sou grato, teria que escrever metade do meu Trabalho de Graduação com os nomes deles, contudo queria que cada pessoa que me ajudou soubesse que sou muito agradecido por tudo que aprendi nessa cidade, a formação profissional e moral que tive aqui vou levar para o resto da minha vida. A cada dessas pessoas eu só posso falar uma coisa muito obrigado.

Resumo

Sistemas embarcados estão se tornando cada vez mais comuns. Para responder a esse aumento no uso e conseqüentemente do consumo, os centros de pesquisa precisam produzir novos componentes num ritmo muito mais acelerado que há alguns anos atrás. Nesse sentido, o ciclo de vida de uso dos chips também tem sido reduzido gradualmente. Esse cenário requer aumento na produtividade para suprir essa demanda. Uma saída para aumentar essa produtividade no desenvolvimento de SoCs é o uso das ferramentas ESL, que são softwares que fornecem um nível de abstração mais alto para os projetistas. O objetivo desse trabalho é implementar uma ferramenta que possa ser usada como alternativa para a IDE Altera Quartus II, onde será possível orientar todas as etapas do processo de desenvolvimento de um sistema embarcado através de scripts. Essa ferramenta também tem o objetivo se integrar ao projeto de Síntese de Comunicação, recebendo um arquivo XML, com os arquivos HDL e retornando uma plataforma para configurar um chip programável, o FPGA. A idéia é mostrar a viabilidade dos sistemas ESL, e os testes e simulações serão feitos no FPGA Altera DE2-35, com o software sendo executado no sistema operacional Linux.

Palavras-chave: FPGA, ESL, XML, SoC, HDL

Sumário

Lista de figuras

Lista de Abreviaturas

ANSI - American National Standards Institute

API - Application Programming Interface

ASIC - Application-Specific Integrated Circuit

CAD - Computer Aided Design

CDF - Chain Description Files

CFI - Common Flash Interface

CMOS - Complementary Metal-Oxide-Semiconductor

DMIPS - Dhrystone Million Instructions Per Second

DOM - Document Object Models

EDA - Electronic Design Automation

EPCS - Erasable Programmable Configurable Serial

ESL - Electronic System Level

FPGA - Field Programmable Gate Array

FPS - Frames Per Second

HAL - Hardware Abstract Language

HDL - Hardware Description Language

I/O - Input/Output

IDE - Integrated Development Environment

IRQ - Interrupt Request Line

JTAG - Joint Test Action Group

LCD - Liquid Crystal Display

MMC - Multi-Media Card

MMU - Memory Management Unit

MPU - Memory Protection Unit

PCB - Printed Circuit Board

PIO - Parallel Input/Output

PLL - Phase-Locked Loop

QIP - Quartus II IP File

QSF - Quartus Settings File

RAM - Random Access Memory

RISC - Reduced Instruction Set Computer

ROM - Ready Only Memory

RTL - Register Transfer Level

SAX - Simple API for XML

SD - Secure Digital

SDRAM - Synchronous Dynamic RAM

SoC - System-on-a-chip

SOF - Static random access memory Object Files

SOPC - System on a Programmable Chip Builder

SRAM - Static RAM

TCL - Tool Command Language

TLM - Transaction Level Modeling

UART - Universal Asynchronous Receiver Transmitter

UML - Unified Modeling Language

VHDL - VHSIC Hardware Description Language

VHSIC - Very High Speed Integrated Circuits

W3C - World Wide Web Consortium

XML - eXtensible Markup Language

Lista de Tabelas

1. Introdução

Observando a essência dos mais variados equipamentos eletrônicos, desde fornos microondas a controladores de freios ABS, pode se notar que a maioria deles hoje possui dispositivos conhecidos como sistemas embarcados. O conjunto, onde estão contidos esses sistemas, possui elementos como processadores, microcontroladores, chips programáveis, entre outros, e seu emprego está sofrendo um aumento gradual ao longo dos anos.

O principal motivador desse crescimento é a competição entre os fabricantes que tem levado a uma diminuição no ciclo de vida de alguns produtos compostos por sistemas embarcados. Esse fato é constatado e impulsionado pela demanda cada vez maior de consumo das pessoas, que tem levado a uma necessidade por parte da indústria de ter que inovar nos lançamentos de forma mais ágil para se manter competitiva.

Se soma isso ao fato da chamada Convergência Digital [10] estar em franca expansão, ou seja, um celular que a pouco tempo atrás tinha a obrigação de apenas realizar e receber chamadas, agora possui câmera, adaptador Wi-Fi, receptor de TV, entre outros recursos. Todos esses “upgrades” têm aumentado bastante a complexidade dos sistemas embarcados.

1.1. Contexto

Esse aumento da complexidade já começou a ser percebido desde o início da década de 1970 com o surgimento dos primeiros microprocessadores integrados. Contudo nessa época o investimento em ferramentas do tipo EDA ainda era pequeno devido à baixa eficiência frente aos projetos que podiam ser feitos com o auxilio de tabelas verdades e lógicas de mais baixo nível. As ferramentas EDA por sua vez são uma categoria de softwares usados para projetar circuitos integrados e placas com elementos semicondutores, e nesse contexto, as ESL [23] podem se incluir.

Para ser mais preciso, mesmo atualmente, o emprego de ferramentas para abstrair a complexidade no desenvolvimento de sistemas embarcados ainda não foi completamente difundido.

Esse fato pode ser constatado com a afirmação do Analista de indústria Gary Smith, da Gary Smith EDA, que observou, que embora ESL tenha tido um impulso maior a partir de 2004, o aumento de sua adoção está acontecendo em um ritmo lento [1].

Por outro lado, mesmo que o uso dos ESL ainda não tenha conquistado uma utilização consolidada e mais ampla, é importante mencionar que sua adoção vem acontecendo gradualmente em um ritmo mais acelerado do que foi a inserção da tecnologia RTL, que é a abstração usada para realizar os projetos HDL, que inclui linguagens como VHDL e Verilog, essa última foi utilizada nesse Trabalho de Graduação (TG). Esse aumento da participação dos ESL no desenvolvimento de sistemas embarcados é tão expressivo que estima-se por volta de 2012 eles já estejam presentes em boa parte dos projetos, e fazendo uma comparação com a tecnologia RTL que levou cerca de 8 anos para se consolidar, a adesão dos ESL se mostra bem promissora [12].

Dessa forma, os ESL são uma proposta que representa um novo uso de uma tendência antiga no meio tecnológico, ou seja, o paradigma de aumentar a abstração para diminuir o tempo de produção e conseqüentemente aumentar a produtividade, já vem sendo usado há bastante tempo no desenvolvimento de software, basta mencionar que programação há alguns anos era feita em linguagem de montagem como Assembly [2] e hoje se programa com C++ [3] ou Java [4]. Nesse sentido uma proposta também bastante conhecida é o SystemC, que é um padrão para o desenvolvimento de hardware e software baseado na linguagem C++, com biblioteca própria e orientada a hardware [14].

Evidentemente que é maior o desempenho de um sistema, concebido em um paradigma de mais baixo nível, e essa é ainda a grande desvantagem dos ESL, entretanto é justamente pretendendo melhorar a relação ***produtividade x desempenho*** que pesquisas, como esse TG, vêm sendo feitas.

Outro fator que motiva a criação desse tipo de ferramenta (ESL) é a possibilidade de ampliar o número de pessoas desenvolvendo sistemas embarcados, pois a partir do momento que o projetista não precisar se preocupar com tempo, área do chip, ciclos de máquina para escrita, endereçamento de componentes, entre outras tarefas, a produção vai ser mais difundida, abrindo espaço para soluções melhores no desenvolvimento de sistemas e componentes.

Sendo assim, segundo Gary Smith [1], o uso de plataformas ESL em combinação com dispositivos off-the-shelf (componentes prontos) pode diminuir a distância entre o desenvolvimento das tecnologias de silício e a complexidade dos dispositivos produzidos. Ou seja, com o aperfeiçoamento dos ESL vai ser possível, o projetista se preocupar mais com a funcionalidade do sistema e não tanto com as restrições físicas e eletrônicas do dispositivo embarcado. Nesse sentido, pode-se mencionar algumas soluções existentes como a Catapult C Synthesis [5] da Mentor Graphics e o CoWare Platform Architect [6] da CoWare. E a difusão do uso dos ESL não acontece por acaso ou por vontade que os projetistas tenham de trabalhar menos, por outro lado, é quase uma exigência do mercado que possui produtos com ciclos de vida muito mais curtos que no passado e precisa de inovação em software e hardware com eficiência e dinamismo, sabendo que essa é uma das coisas principais para se manter competitivo e sobreviver no mercado de tecnologia [26].

1.2. Objetivo

O objetivo desse Trabalho de Graduação é o desenvolvimento de 3 módulos de tradução possibilitando a conversão de uma descrição XML para um modelo de plataforma que poderá ser baixada em um FPGA diretamente. Todo o processo será orientado por meio de scripts, fornecendo um nível mais alto de abstração para o usuário, que não vai precisar de grandes conhecimentos técnicos em circuitos integrados para poder sintetizar seus dispositivos. Esses componentes serão integrados ao projeto de Síntese de Comunicação [11], sendo do tipo ESL, possibilitando compilação de circuitos remotamente.

1.3. Estrutura do trabalho

A forma escolhida de apresentar esse trabalho vai seguir a mesma ordem do seu desenvolvimento, e será dividido em 8 seções. A primeira seção traz um resumo da ferramenta e do TG. Na segunda será focada descrição dos componentes da plataforma Altera DE2. Na terceira seção é feito um resumo das principais ferramentas usadas para implementar esse TG. Na quarta seção é feita uma abordagem das soluções existentes para fins de contextualização. Na quinta seção se encontra uma descrição do funcionamento e características da ferramenta Power ESL. Na sexta seção é explicado o estudo de caso e finalmente seguem as conclusões e referências na sétima e oitava respectivamente. No transcorrer dessas seções serão mostradas as estratégias e tecnologias usadas nesse TG.

2. Plataforma Altera DE2

Considerando um sistema computacional simples, 4 elementos são essenciais para o seu funcionamento: Barramento, Processador, Memória e **I/O**. Sem eles o modelo de computador de Von Neuman [14], mostrado na Figura 1, não é possível.

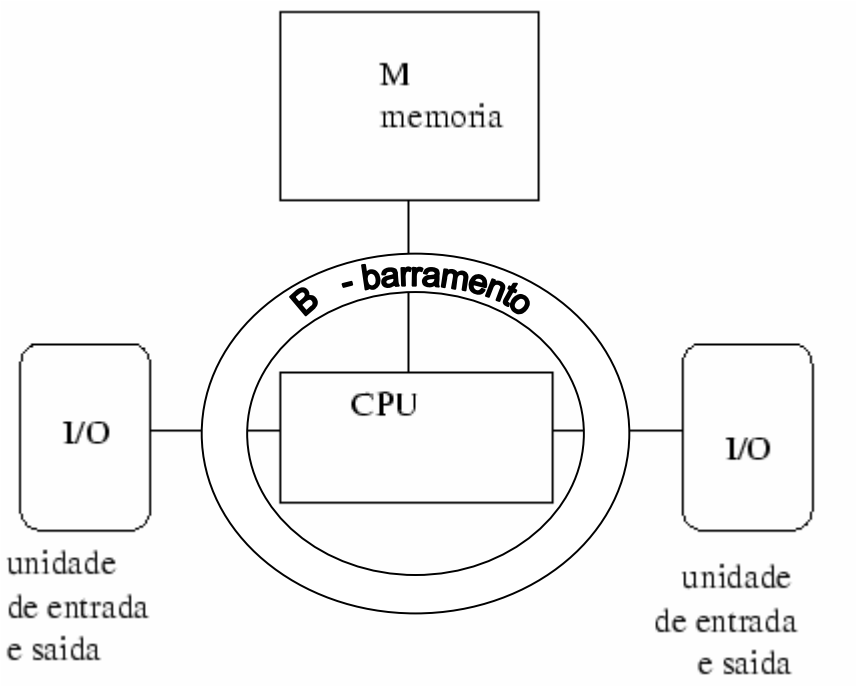


Figura 1. Arquitetura computacional de Von Neuman

Pensando nisso e sabendo que seria necessário uma plataforma segura e eficiente para receber módulos externos do usuário, foi escolhida a plataforma Altera DE2. Essa plataforma é um sistema computacional completo, com hierarquia de memória, barramento, processamento, portas de comunicação, entre outros elementos. Ao todo são 20 módulos em sua versão mais básica, a qual foi usada nesse **TG**. Para melhor compreensão do funcionamento desse sistema, nas seções seguintes será feita uma abordagem compacta com a maior amplitude possível de cada um desses módulos. Vale ressaltar que todos esses componentes são implementados no paradigma soft-core [27], ou seja, são descritos em software, não existindo fisicamente, a partir de sua compilação podem ser baixados (carregados) em um **FPGA**, e passam a operar.

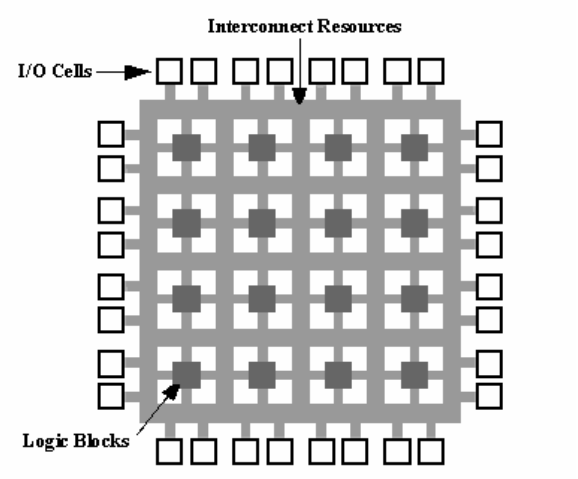


Figura 2. Estrutura básica de um FPGA

A tecnologia FPGA corresponde a um chip programável, onde o projetista pode descrever o comportamento ou a estrutura de um sistema e programar esse chip para funcionar como se fosse um hardware projetado fisicamente. Esses dispositivos são compostos por blocos lógicos e células de I/O, como mostrado na Figura 2. Essa tecnologia é viável quando se quer um sistema para uso rápido e produção de poucas unidades e o desempenho não seja fator crítico. Do contrário a tecnologia ASIC seria a mais indicada, pois representa os circuitos projetados em grande escala e produzidos fisicamente em fábricas especializadas. Para efeitos de comparação, na Figura 3, encontra-se um comparativo de ***custo x volume*** produzido entre chips desenvolvidos na tecnologia ASIC e FPGA.

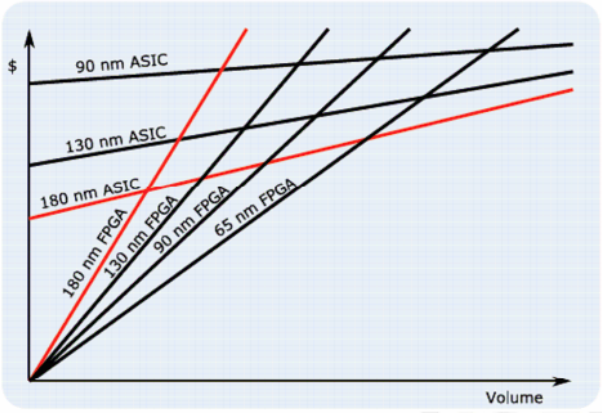


Figura 3. Gráfico comparativo de custo x volume entre FPGA e ASIC

Dessa forma, o FPGA é mais indicado para o ambiente acadêmico, pois possibilita o reuso de componentes e testes em curto espaço de tempo, com baixo custo e boa eficiência. Por outro lado, a ideia das ferramentas ESL é estar um pouco acima da representação comportamental ou estrutural, fornecendo a possibilidade de o projetista descrever apenas a estrutura abstrata do sistema e suas conexões. Nesse sentido o FPGA é interessante, pois insere flexibilidade no desenvolvimento dessas ferramentas.

Sendo assim, as descrições dos componentes de uma plataforma básica para o sistema Altera DE2 estão nas seções seguintes, e o seu diagrama de blocos desse sistema é apresentado na Figura 4.

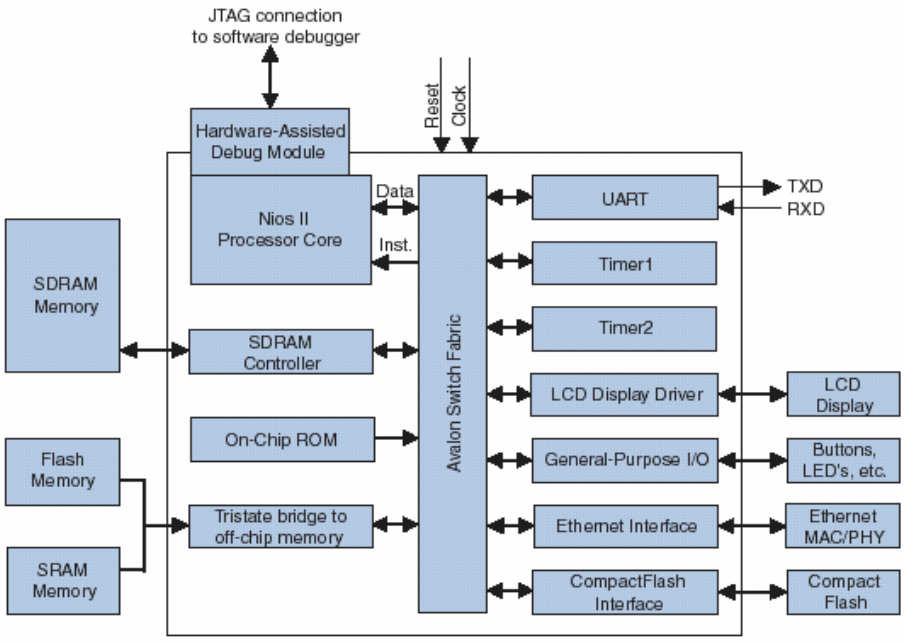


Figura 4. Esquema de uma plataforma básica da Altera

A família de FPGA utilizada nos experimentos desse TG foi o Altera Cyclone II, e o modelo foi o **EPC2630**.

2.1. Processador NIOS

É processador de propósito geral e surgiu em 2001 e representou o primeiro processador comercial viável criado especificamente para sistemas embarcados projetados em FPGA. Desde então, milhares de usuários adotaram o NIOS e o NIOS II da Altera.

A versão usada na plataforma desse projeto foi o NIOS II, que é um processador soft-core de 32 bits, desenvolvido na arquitetura **RISC**, que propõe um número reduzido de instruções e transfere para o compilador e posteriormente para o programador a tarefa de realizar processamento de instruções mais complexas.

Essa unidade de processamento possui freqüência de varia entre 165 a 200 MHz, pipeline de 5 ou 6 estágios, dependendo da implementação. Ele também conta com 32 registradores de propósito geral (r0 a r31), 5 registradores de controle (ct10 a ct14), 32 fontes de interrupções externas.

O NIOS II aceita instruções para o cálculo de operações de multiplicação e divisão, inclusive com ponto flutuante, embora não possua registradores específicos para cálculo com decimais. Outra característica é a possibilidade realizar instruções de deslocamento, realizar acesso a periféricos on-chip, interface para memórias fora do chip, permitindo adicionar mais níveis de hierarquia de memória, e periféricos a partir de interrupções.

Ele também possui um módulo de debug de hardware que habilita o processador a iniciar, parar e continuar gradualmente o processamento controlado por uma IDE. Possui unidade de MMU opcional para suporte a sistemas operacionais que necessitam desse tipo de um conversor de endereços lógicos para endereços físicos.

Esse processamento conta também com uma unidade de MPU opcional, e desempenho de até 250 **DMIPS**, podendo realizar também operações lógicas e relacionais. Outro aspecto seu é o gerenciamento da memória cachê feito por software e tratamento de exceções sem vetor de

interrupção. Devido a todas essas características o NIOS II pode ser comparado a um micro-controlador, com periféricos conectados e diversas interfaces, ele usa o conjunto de instruções consistente e um modelo de programação consistente.

2.2. Barramento Avalon

O Barramento Avalon é uma das partes mais importantes de um sistema computacional **SOPC** baseado na plataforma Altera DE2. É um barramento criado com o propósito de priorizar a taxa de transmissão, fazendo o uso de conexões em paralelo [29]. Ele representa o principal caminho de comunicação entre os componentes desse tipo e plataforma [15]. O Barramento Avalon reúne todos os controles, sinais de dados, endereços e combinações lógicas que conectam os periféricos para montar um sistema computacional completo. Esse barramento possível uma arquitetura flexível, permitindo adaptações para se adequar aos módulos implementados pelos projetistas de sistemas embarcados.

Em um projeto de sistema o Barramento Avalon pode ser gerado automaticamente pelo SOPC Builder [13], que se responsabiliza por conectar todos os módulos lhes atribuindo também os devidos endereços, como exibido na Figura 5. O SOPC Builder automatiza a integração dos módulos já existentes ao Barramento Avalon, e a menos que o projetista determine que seu uso seja discreto, a sua inserção no sistema acontece de forma bastante transparente. Sendo assim, a visão que o desenvolvedor tem do barramento se restringe basicamente as portas de conexão que podem ser usadas para interligar os periféricos do sistema, podendo ser considerado um componente lógico ativo e não apenas passivo como outro barramento qualquer. Sendo assim, o Avalon pode conectar todos os módulos que um barramento passivo conecta, fornecendo a possibilidade também de resguardar o projetista da obrigação de especificar características elétricas ou físicas de um barramento **hard-core**.

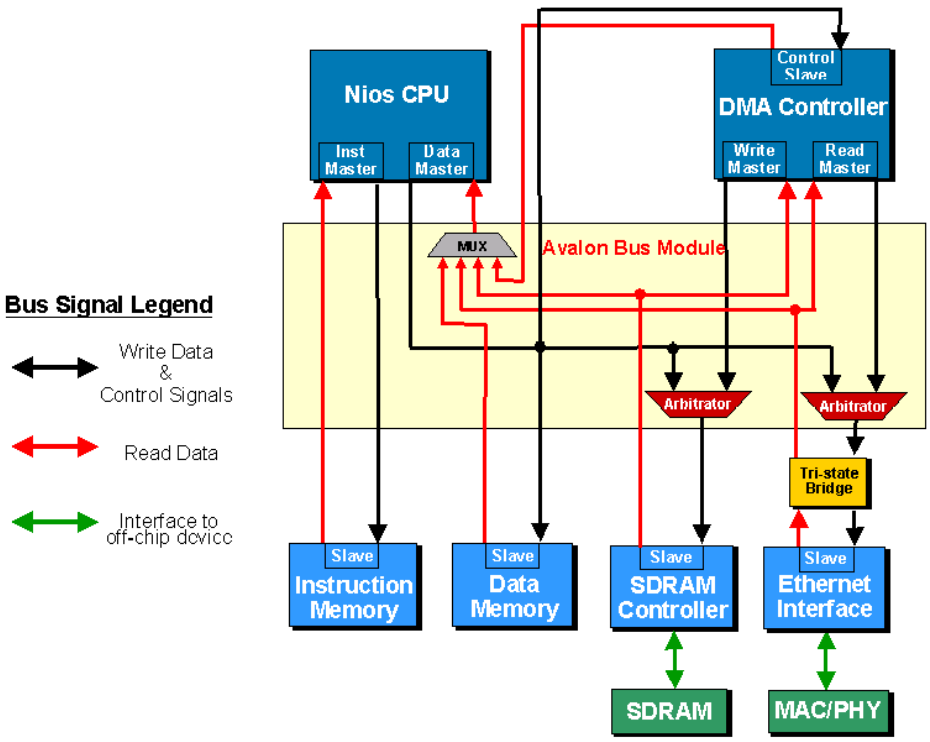


Figura 5. Aspecto estrutural de uma plataforma da Altera

O Barramento Avalon fornece os seguintes serviços aos módulos conectados a ele, que são:

- *Multiplexação de Data-path* - é feita por multiplexadores existentes no Avalon que transferem os dados de um periférico escravo para seu respectivo periférico mestre.
- *Decodificação de endereços* - é a tradução de endereços internos produzindo sinais que habilitam o chip (chipselect) para cada periférico. Esta implementação simplifica o projeto porque o próprio periférico não precisa interpretar e decodificar os sinais de endereço.
- *Produção do estado de espera* - estende as transferências de dados do barramento por um ou mais ciclos, para acomodar o funcionamento de periféricos com necessidades de sincronização especiais. Esse recurso é usado em casos quando os sinais de read-enable e write-enable tem que ser modificados ou para manter requisitos temporais.
- *Dimensionamento dinâmico* - é um procedimento que esconde os detalhes de interfaceamento entres os periféricos e o Avalon. Fornecendo, por exemplo, conversão do tamanho de leituras e escritas entre periféricos mestre e escravos, reduzindo a complexidade do software nos

periféricos mestre, desde que é necessário qualquer precaução quanto a natureza física do periférico escravo.

- *Atribuição de prioridade de interrupção* - é a transferência de prioridade entre os periféricos, quando existe concorrência por algum recurso, solicitados através de interrupções.
- *Transferência de capacidade latente* - é a capacidade fornecer transferência com latência entre os módulos dentro do Avalon.
- *Capacidade de fluxo de leitura e escrita* - é mais um recurso contido dentro do Avalon.

O Avalon é um barramento com espaço de endereçamento de até 4 GB, ou seja, a memória e os periféricos podem ser mapeados em um range de até 32 bits. O Avalon também possui tanto periféricos on-chip (conexão gerada em tempo de configuração) como aceita off-chip (periféricos ligados aos pinos do barramento). Esses periféricos podem ser classificados em módulos do tipo inside e outside respectivamente. Esse barramento também conta com 3 categorias de portas: mestre, escravo e a híbrida mestre-escravo, cada uma conectando os seus respectivos periféricos que possuem categorias com os mesmos nomes das portas.

O Barramento Avalon possui uma série de outras restrições que podem ser conferidas no manual da Altera [28]. No entanto, as informações mais importantes para construir uma ferramenta ESL são os sinais do barramento que estão detalhadas na Tabela 1:

Tipo de sinal		**Largura**		**Direção**		**Necessária**		**Descrição**	
◪	◧	◪	◧	◪	◧	◪	◧	◪	◧
clk		1		in		sim	não	Sinal global de clock para os módulos e o barramento. Apenas portas escravas podem omitir o clock.	Sinal global de clock para os módulos e o barramento. Todas as transições do barramento são síncronas com o clock
reset		1		in		não		Sinal global de reset. A sua implementação depende do módulo	
Chip select	-	1	-	in	-	sim	-	É o sinal que indica que a porta escrava está ativa	-
address		1-32		in	out	não	sim	Recebe o endereço dos módulos	
byteenable		0,2,4		in	out	não		Este sinal habilita bytes específicos durante transferências para a memória com comprimento maior que 8 bits.	

Tipo de sinal		Largura		Direção		Necessária		Descrição	
read		1		in	out	não		Sinal de leitura para o escravo. Se esse sinal nunca enviar dados ao mestre ele não é necessário, do contrário o sinal readdata também deve ser usado	Sinal de leitura do mestre. Se o Avalon nunca realizar leitura esse sinal não é necessário, do contrário o sinal readdata deve também ser usado
readdata		1-32		out	in	não		Representa um fluxo de dados a ser lido pelo Avalon. Esse processo depende do sinal read estar ativo	
write		1		in	out	não		Sinal de escrita para o escravo. Se esse sinal nunca receber dados do mestre ele não é necessário, do contrário o sinal writedata também deve ser usado	Sinal de escrita do mestre. Se o Avalon nunca realizar escrita esse sinal não é necessário, do contrário o sinal writedata deve também ser usado
waitrequest		1		out	in	não	sim	Esse sinal é usado para fazer o Avalon esperar um tempo, quando a porta escrava não está habilitada para responder imediatamente	Esse sinal força a porta mestre a esperar até que o Avalon esteja pronto para realizar a transferência
irq		1		out	in	não		Esse sinal é usada pela porta escrava precisa de algum serviço da porta mestre	
reset request	-	1	-	out	-	não	-	Esse sinal permite um periférico resetar o módulo inteiro	-
Begin transfer	-	1	-	out	-	não	-	Esse sinal é atribuído durante o primeiro ciclo de cada nova transferência do Avalon	-
-	irq number	-	6	-	in	-	não	-	Esse sinal indica a prioridade de interrupção, quanto menor o valor maior a prioridade
-	end of packet	-	1	-	in	-	não	-	Esse sinal indica uma condição de termino de transferência
-	read data valid	-	1	-	in	-	não	-	Esse sinal indica se os dados presentes na porta slave readdata são válidos
-	flush	-	1	-	out	-	não	-	Esse sinal pode limpar qualquer leitura pendente

Porta Slave Porta Master

Tabela 1. Sinais do Barramento Avalon

2.3. Trisatate Bridge

Esse módulo serve para mediar a conexão de componentes off-chip com o barramento. E ele cria sinais de **I/O** que devem se conectar aos pinos do FPGA [30]. O tristate bridge cria pinos de dados e endereço que podem ser compartilhados com outros módulos externos. O tristate bridge deve ser usado nos seguintes casos:

- O módulo externo possui pinos bidirecionais de dados;
- Módulos externos compartilham barramento de endereço, dados ou ambos.

O tristate bridge possui 2 modos de funcionamento que são:

- Registrado - este modo adiciona registradores a todos os pinos de entrada do FPGA.
- Não registrado - este modo não adiciona registradores entre os pinos de saída da memória do dispositivo e sistema conectado ao barramento.

Dessa forma, se constata que registrando os sinais de entrada e saída, é diminuído o atraso entre os registradores no percurso entre a memória do dispositivo e o FPGA resultando em aumento de desempenho. No entanto, esse processo adiciona um ciclo a mais de latência em cada direção, mas como os registradores não afetam as restrições de tempo de transferência, não existe perda de desempenho [21].

2.4. CFI flash

É uma memória flash que pode ser usada em associação com outras memórias e sua função é melhorar o desempenho do sistema realizando o papel das memórias não voláteis como a ROM. Seu barramento de endereços possui 23 bits, enquanto o barramento de dados possui 8 bits. Esse módulo está conectado a 3 sinais que são: chipselect, read_n, write_n e uma porta slave.

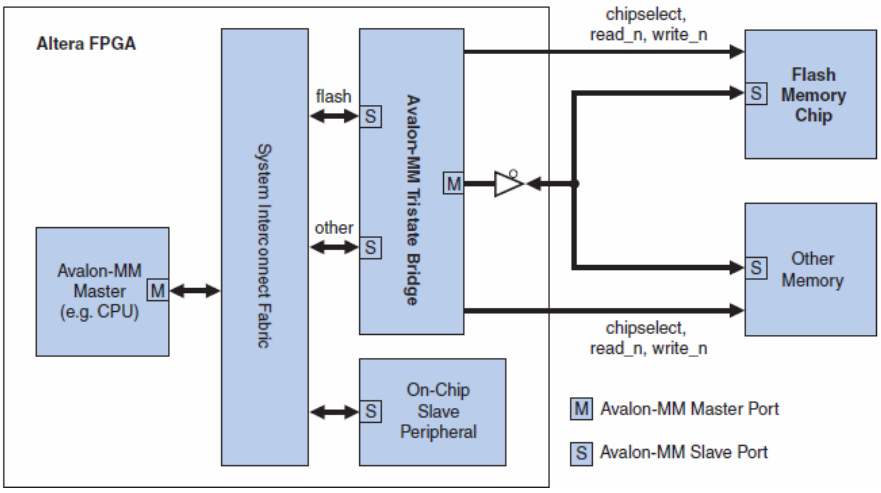


Figura 6. Diagrama de blocos de um sistema com memória flash

O controlador dessa memória, conhecido como CFI [31], é um hardware pequeno. Ele é uma porta tristate configurada com waitstates, hold time e setup apropriado para o chip flash escolhido. Essa porta tristate do Barramento Avalon é capaz de realizar leitura e escrita de dados, e seu aspecto geral é exibido na Figura 6.

2.5. SDRAM

Corresponde a memória principal do sistema [32] com capacidade de até 8 Mb. Nesse componente, todos os sinais com exceção do clock são fornecidos pelo controlador de memória como mostra a Figura 7:

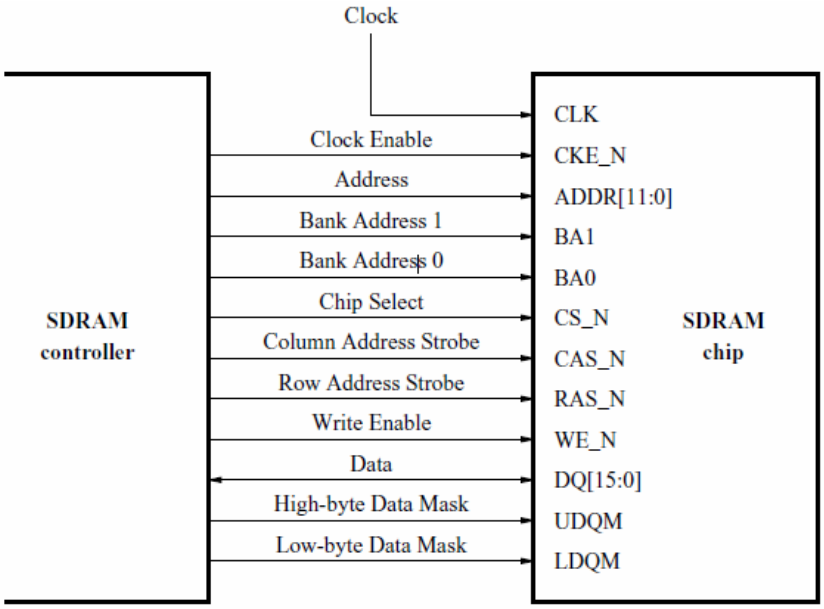


Figura 7. Sinais de comunicação entre SDRAM e seu controlador

As descrições dos demais sinais são:

- Clock enable - habilita o uso do clock;
- Address - possui 11 bits e representa o barramento de endereços.
- Bank Address 0 e 1 - são regiões da memórias utilizadas exclusivamente para armazenar endereços de instruções.
- Chipselect - é o sinal que ativa o uso da memória.
- Column Address Strobe (CAS) - é o pino da memória no qual os dados são apresentados na memória, realizando a transição entre o momento em que o comando de leitura é executado e a chegada do dado.
- Row Address Strobe (RAS) - é o pino da SDRAM que realiza o acesso as linhas de armazenamento da memória.
- Write enable - é pino que habilita a escrita na SDRAM.
- Data - é um barramento por onde os dados efetivamente são transmitidos, lidos e escritos na memória.
- High-byte data mask - é o pino que ativa o acesso dos dados contidos nos bytes mais significativos.
- Low-byte data mask - é o pino que ativa o acesso dos dados contidos nos bytes menos significativos, esse recurso pode ser usado quando se quer acessar apenas uma parte dos dados e deixar a outra inalterada.

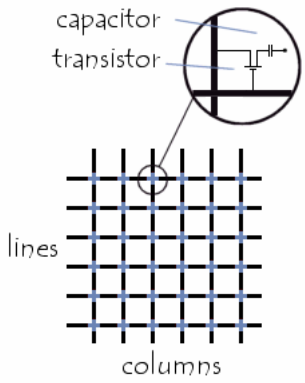


Figura 8. Representação de uma região de memória

2.6. Controlador EPCS

EPCS são todos aqueles módulos que realizam algum tipo de comunicação serial, por exemplo, memória flash. Por sua vez o controlador EPCS [33], juntamente com o Barramento Avalon fornecem ao NIOS II acesso a dispositivos EPCS seriais. O Altera equipou um NIOS II com um sistema de biblioteca **HAL**, permitindo módulos EPCS ler e escrever usando uma **API** comum para equipamentos flash.

O uso de um controlador como esse, fornece a capacidade ao NIOS II de guardar o código de um programa em um módulo EPCS, permitindo acessos mais rápidos a determinadas funcionalidades, bem como armazenar dados não voláteis, e gerenciar dados de configuração de equipamentos, por exemplo, receber novas configurações de um sistema embarcado de rede e usar um módulo EPCS para aplicar essas modificações como mostrado na Figura 9.

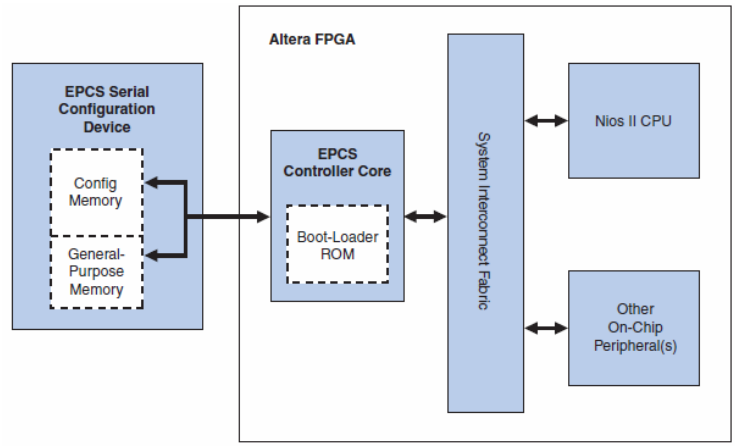


Figura 9. Diagrama de blocos de interação do EPCS com o Avalon

Nesse esquema é possível ver que o controlador EPCS possui uma memória ROM para fazer a inicialização de um programa que use esse recurso. O controlador é ligado diretamente ao Barramento Avalon, bem como o NIOS II e outros periféricos. O EPCS está conectado a um dispositivo EPCS serial, que pode ser uma memória flash. Para o FPGA Cyclone II, usado nos experimentos nesse TG, são requisitados 512 bytes da memória ROM desse controlador.

2.7. JTAG UART

O **JTAG** é um padrão criado por essa entidade que acabou levando o nome da mesma, e tem o propósito de facilitar a realização de teste em placas de circuito integrado (PCB), recebendo mais tarde o nome de **IEEE** 1149.1. A implementação do módulo Altera JTAG UART representa um dispositivo que juntamente com o Barramento Avalon implementa uma forma para fazer a comunicação serial entre o PC e um FPGA com o utilização do SOPC Builder. Em diversos projetos a JTAG UART elimina a necessidade de se ter uma conexão serial RS-232 para entrada/saída de caracteres. Esse dispositivo proporciona uma interface Avalon que esconde a complexidade da interface JTAG do projetista. Dessa forma, periféricos máster podem se comunicar lendo e escrevendo em registradores de dados e controle.

O módulo JTAG UART [34] usa as implementações do JTAG presente nos FPGAs da Altera para fornecer comunicação para os computadores. Essa conexão pode ser feita usando um cabo USB-Blaster. A Altera fornece todas as ferramentas necessárias para a comunicação, como JTAG terminal, que gerencia, decodifica os dados e mostra os caracteres na tela.

O funcionamento de todo JTAG se dá através de 5 pinos que são: TDI (entrada de dados), TDO (saída de dados), TCK (clock), TMS (modo de seleção) e TRST (reset). O JTAG UART se comunica com um módulo controlador que trata esses sinais nos FPGAs Altera, como mostra a Figura 10.

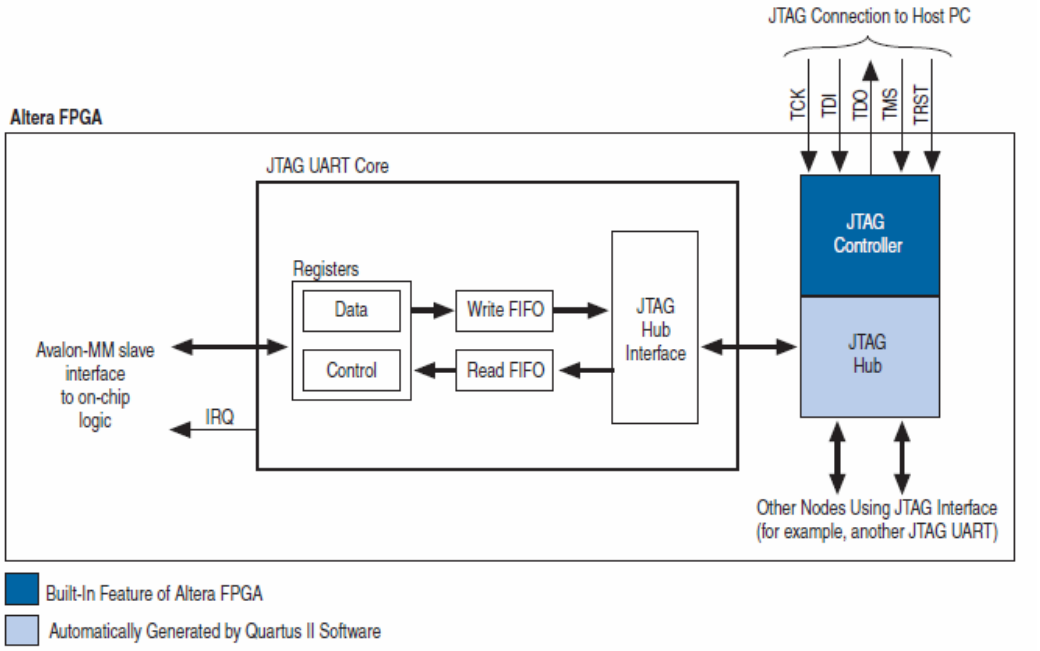


Figura 10. Diagrama de blocos de um JTAG UART

2.8. Timer 0 e Timer 1

São dois temporizadores semelhantes aos encontrados em microcontroladores ou microprocessadores e podem ser usados como gerador de pulsos periódicos ou em um sistema para monitorar possíveis travamentos em software conhecido como “watchdog”. Seu contador possui 32 bits e seu funcionamento é comandado por software que escreve em seus registradores [35]. Esses temporizadores têm a capacidade de gerar interrupções que podem ser mascadas por um bit interno de controle.

Esses temporizadores possuem 1 registrador de status e 1 registrador de controle que serão detalhados a seguir:

Registrador de status:

- Valor ‘0’ (to) - quando o registrador está com esse valor indica que o contador do temporizador está zerado, o seu valor pode também ser alterado por software.
- Valor ‘1’ (run) - quando o registrador está com esse valor indica que o contador está operando. Os comandos de início e parada são determinados pelo registrador de controle.

Registrador de controle:

- Valor ‘0’ (ito) - habilita interrupção para um timeout. Se o ito é ‘1’ o temporizador gera uma interrupção (coloca ‘1’ no IRQ), quando o registrador de status for ‘1’, se o ito é ‘0’, o IRQ é ‘0’.
- Valor ‘1’ (cont) - habilita o modo contínuo, que determina como o contador deve se comportar quando ele for zerado.
- Valor ‘2’ (start) - habilita o contador a iniciar ou continuar uma contagem.
- Valor ‘3’ (stop) – determina a parada do temporizador.

2.9. LCD 16.207

Esse módulo presente na plataforma básica da Altera juntamente com o Avalon fornecem uma interface de hardware e um driver em software necessários para o NIOS II mostrar caracteres na tela de **LCD** Optrex 16.207 [36], presente na placa Altera DE2 usada nesse TG. Os devices drivers necessários são fornecidos por uma biblioteca HAL e o NIOS realiza o acesso a essa tela usando rotinas da **ANSI** C, como por exemplo, printf().

O módulo LCD 16.207 possui 11 sinais que são conectados a um controlador do LCD, que por sua vez se conectam ao Barramento Avalon, eles são:

- E - Enable (saída);
- RS - Register Select (saída);
- R/W - Read or Write (saída);
- DB0 trough DB7 – Data Bus (bidirecional).

Esses sinais são mostrados na Figura 11, bem como o controlador.

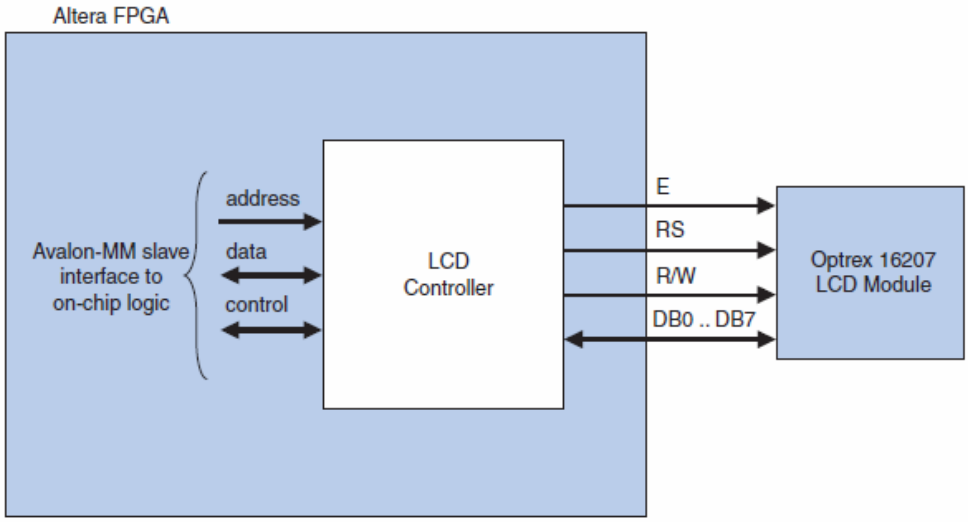


Figura 11. Diagrama de blocos do controlador de LCD

2.10. LEDs vermelho e verde

São módulos simples que recebem um sinal do Barramento Avalon e acionam os LEDs presentes na placa Altera DE2, que ao todo são 18 LEDs vermelhos e 9 verdes.

2.11. Botão PIO

Os botões paralelos de entrada e saída (PIO) são controlados por um módulo chamado Button_pio presente na plataforma básica e fornecem uma forma simples de I/O para controle da placa. Alguns exemplos de funções dos botões PIO são: controle de LEDs, aquisição de dados a partir do barramento, controle do display, configuração e comunicação com equipamentos off-chip. O seu

funcionamento é baseado na ativação de interrupções, leitura e escrita em registradores mapeados no Avalon, podendo prover até 32 portas de I/O [37].

2.12. Switch PIO

É um módulo que fornece informações a dispositivos PIO integrados com o Barramento Avalon. Esses switchs podem conectar portas dos tipos:

- Bidirecionais (tristate) - nesse modo os bits compartilham um pino para capturar e direcionar os dados. Essa direção é selecionada individualmente para cada pino. Para habilitar a função tristate nos pino de I/O do FPGA, é necessário configurar a direção de entrada dos dados.
- Entrada - nesse modo apenas os dados de entrada são capturados.
- Saída - nesse modo apenas os dados de saída são capturados.
- Portas de entrada e saída - nesse modo os barramentos das portas de entrada e saída são separados em barramentos unidirecionais de n bits de largura.

2.13. Seg7_display

É mais um módulo feito em HDL, e pode ser usado para acessar o display de 7 segmentos presente da placa Altera DE2, possuindo aspecto representado na Figura 12.

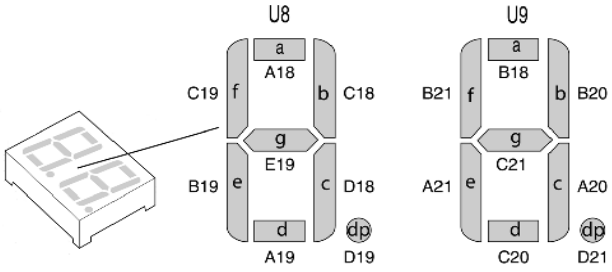


Figura 12. Representação de um display de 7 segmentos

2.14. SRAM

É um módulo feito em Verilog que integra a plataforma básica da Altera. Esse módulo implementa a memória principal usada pelo sistema e é manipulado por um controlador que acessa a **RAM** estática **CMOS** presente da placa Altera DE2.

Segundo as especificações técnicas, os dados se localizam em registradores e de 8 ou de 16 bits, dessa forma, são necessários 2 ciclos de latência para leitura e um ciclo de latência para a escrita. Sendo assim, o tratamento dos dados de 32 bits é o seguinte: o Barramento Avalon quebra esses dados em 2 blocos de 16 bits, dessa forma são necessários 5 ciclos para leitura e 2 ciclos para a escrita dos dados na SRAM [38]. Esse módulo trabalha a uma frequência de 50 MHz.

2.15. DM9000A

É um módulo que implementa um controlador para o protocolo IEEE 802.3 (Fast Ethernet), presente na placa Altera DE2. Esse barramento fornece taxas de transferência que chegam a 100 Mbits/seg, com interface conectada ao Barramento Avalon com 8 ou 16 bits e faz uso e se comunica com a SRAM do FPGA [39].

2.16. VGA

É um módulo em HDL que provê a possibilidade de se ligar um monitor de vídeo diretamente no FPGA. A geração dessa imagem é possível graças a um sinal temporizado incluindo tanto sincronização horizontal quanto vertical. Esse sinal produz em um monitor de vídeo resolução de até 640x480 pixels com freqüência de 60 **FPS**. O funcionamento do módulo VGA [40] requer um clock de 25 MHz, que é proporcionado por um PLL, desde que o clock geral do FPGA é de 50 MHz.

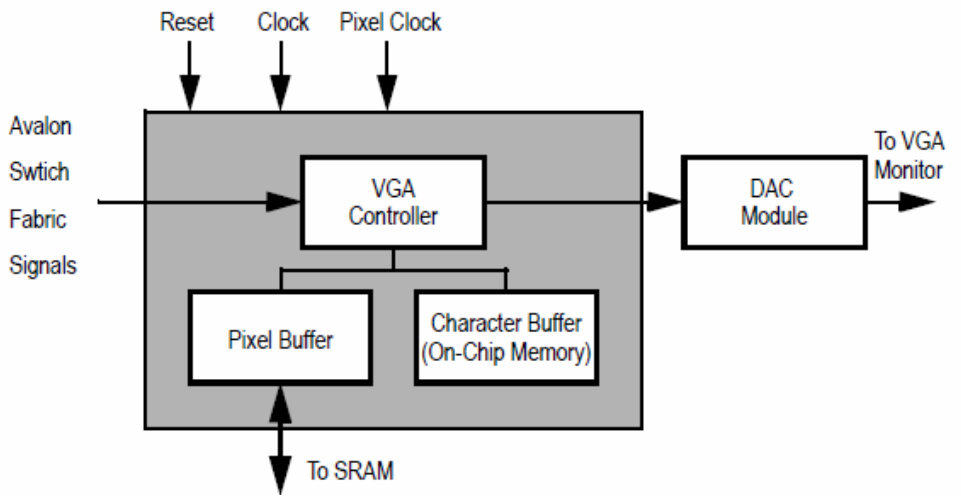


Figura 13. Diagrama de blocos de um controlador de VGA

O módulo VGA presente na plataforma básica da Altera está representado na Figura 13 em forma de diagrama de blocos e contempla os 3 modos de operação do dispositivo de vídeo da placa Altera DE2, esses modos são:

- **Pixel** - nesse modo o dispositivo fica habilitado por uma interface própria com o Barramento Avalon. Ele faz uso da memória SRAM para bufferização os pixels a serem exibidos, além de permitir algumas opções de cor e resolução. O modelo de colorização fornecido é o RGB, podendo ter saída colorida por escala de cinza, e a resolução vai de 40 x 30 até 640 x 480 pixels.
- **Caracter** - nesse modo o dispositivo também possui interface direta com o Avalon. O princípio de operação se baseia em caracteres, ou seja, um dispositivo pode enviar caracteres ASCII para a sua interface o Avalon, que internamente esses caracteres serão devidamente convertidos em pixels. Quando o VGA é inicializado ou reinicializado nesse modo, todos os símbolos são convertidos para caracteres “ ” (espaço). Esse procedimento leva em media 5000 ciclos de clock para ser realizado o que não chega a afetar o desempenho do FPGA que opera a 50 MHz (50 milhões de ciclos por segundo). O usuário também pode limpar a tela simplesmente escrevendo ‘1’ na posição de memória mais significativa da interface com o Avalon. Esse modo de operação fornece modelo de colorização RGB com resolução 80 x 60 e 40 x 30 pixels.
- **Sobreposição de caracteres** - esse é um modo híbrido que aceita tanto caracteres quanto pixels, com resoluções de modo de colorização herdados dos 2 outros modos apresentados.

2.17. Audio

É mais um módulo implementado em HDL que integra a plataforma básica da Altera. Esse componente fornece uma interface para entrada e saida de audio e realiza a sincronização dos canais R e L, forçando um dos canais esperar até que o sinal tenha sido recebido pelo outro canal. Dessa forma, o seu funcionamento faz uso do clock do Avalon, retirando uma amostra baseada na taxa do audio, e a largura dos registrados de dados podem ser 16, 20, 24 ou 32 bits [41].

2.18. SD_Dat

O SD (Secure Digital) Dat é um módulo que implementa um barramento que faz a transferência de dados para o dispositivo SD/MMC que fornece uma interface bidirecional com o leitor de cartão [42].

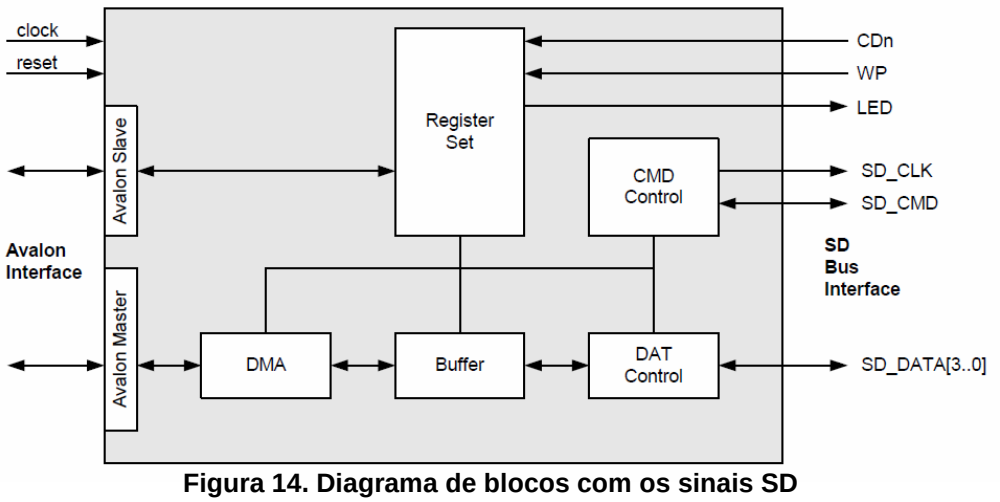


Figura 14. Diagrama de blocos com os sinais SD

2.19. SD_CMD

O SD_CMD é um módulo que implementa os controles do componente SD/MMC presente na placa Altera DE2 [42], de acordo com a Figura 14.

2.20. SD_CLK

É um módulo que proporciona o sinal de clock, a partir da parametrização do clock da Altera DE2, para o dispositivo SD/MMC [43]. Essa freqüência determina a taxa de transmissão dos dados e é determinada por software.

3. Ferramentas usadas no TG

Para a implementação desse TG foram utilizadas diversas ferramentas de hardware e software, entre elas, Eclipse, Altera Quartus II, SOPC Builder, Altera DE2-35 [44]. Elas serão descritas em linhas gerais a seguir.

3.1. Eclipse

O Eclipse é uma **IDE** em código aberto (opensource) criada pela IBM [45] em 1990, como uma alternativa à ferramenta Microsoft Visual Studio [46]. Por volta de 1998 a comunidade de software livre estava surgindo e começou a adotá-la, e atualmente é umas das IDEs mais usadas. O Eclipse pode ser usado tanto para o desenvolvimento de aplicações Java como em linguagens C++, PHP, JSP, Python, entre outras. Nesse projeto foi escolhido o Eclipse, juntamente com a linguagem Java, devido a possibilidade de uma posterior integração com a plataforma PDesigner [12]. Outro fator de escolha da linguagem Java, foi a ampla quantidade de documentação e algoritmos existente, possibilitando o reuso de certas estruturas de dados, devido a seu paradigma de orientação a objetos, que foi bastante útil no momento da abstração do sistema.

Retomando, o Eclipse atualmente está na versão Galileo, e é uma ferramenta muito versátil e portável, presente tanto em sistemas Linux, como no Windows e uma imagem do aspecto atual dessa IDE é exibida na Figura 15. Alguns exemplos de ferramentas que são plugins do Eclipse são: QNX [47], NIOS II IDE [48], Google App Engine [49], entre outros. As facilidades do Eclipse começam no uso, pois o mesmo não precisa ser instalado nem no Windows, nem no Linux. Ele necessita apenas da máquina virtual Java no sistema operacional, no qual vai ser executado.

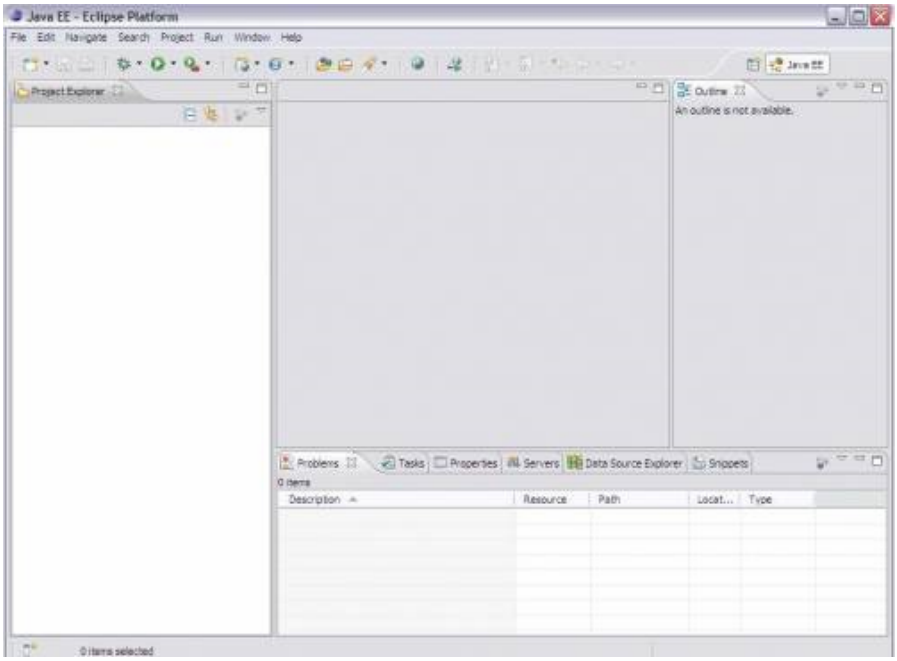


Figura 15. IDE Eclipse

O Eclipse fornece muita versatilidade ao programador, pois uma vez que determinada linguagem tenha seu pacote instalado nessa ferramenta, a própria IDE auxilia na identificação de erros de sintaxe, sugerindo bibliotecas de correção, sem falar no seu excelente depurador, que fornece relatórios em tempo real durante o processo de depuração.

3.2. Altera Quartus II

O Quartus II é uma ferramenta desenvolvida pela Altera e tem como finalidade principal criar projetos de sistemas embarcados. Ele possui versões tanto para Windows, como Linux. Está na versão 9.1sp2, a mesma usada nesse TG. E conta com versões de licença livre e paga. Contudo, na modalidade livre, a geração do arquivo com projeto sintetizado, conhecido como SOF, é restrita, sendo impossível baixar no FPGA o sistema. É um ambiente bastante versátil, pois permite que os módulos, a serem criados pelo usuário, possam ser descritos tanto no nível comportamental (HDL), como lógico (esquemático). O Quartus II realiza todas as etapas do fluxo de prototipação de um circuito integrado como síntese de alto nível, roteamento do circuito, posicionamento de componentes, (otimização para redução de potência dissipada) fitter, análises temporais, síntese lógica, entre outras. Na Figura 16 encontra-se uma representação RTL feita no Quartus II para efeito de contextualização.

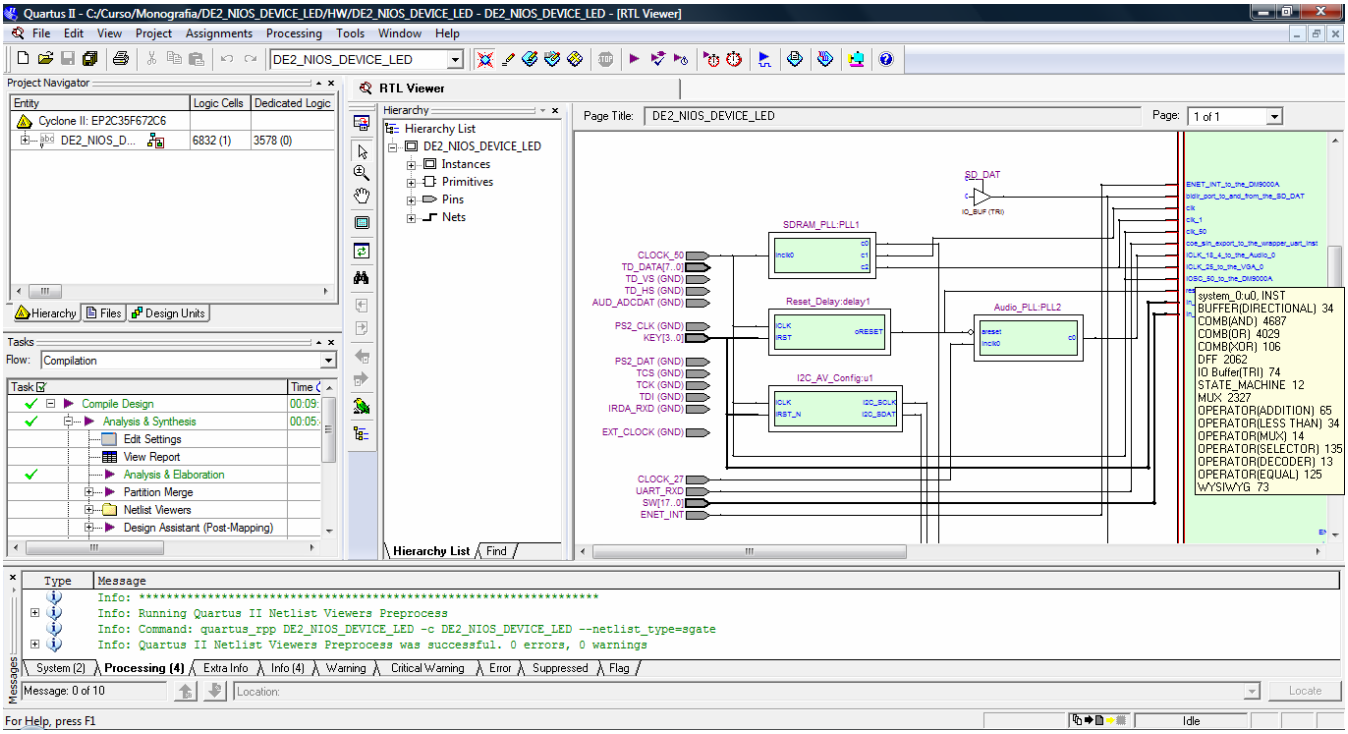


Figura 16. Representação RTL na ferramenta Quartus II

Essa ferramenta também dá suporte a um analisador lógico o Signal Tap [50], que fornece uma ambiente de simulação verificando os sinais reais no FPGA. Outra forma de simulação no Quartus é a realizada com Waveforms, que representa uma espécie de linha do tempo, onde o usuário pode conferir o fluxo dos bits em um barramento ou componentes em uma faixa continua de tempo.

O Quartus II oferece também suporte ao aplicativo SOPC Builder, que será descrito na próxima seção, possibilitando a criação de plataformas com um esquema completo dos componentes (barramento, processador, memória, entre outros).

Outra funcionalidade interessante é a possibilidade de acessar seus recursos por meio de scripts e que viabilizou a implementação da ferramenta Power ESL, desenvolvida nesse TG. Para acessar a funcionalidade script, é necessário montar um arquivo de entrada, com extensões .TCL, .QSF, por exemplo, e lançar algum dos comandos disponíveis na documentação da Altera, que podem realizar todas as etapas do fluxo de prototipação de um componente embarcado, inclusive o download do arquivo de descrição .SOF diretamente no FPGA.

Sendo assim, devido ao fato de Quartus ser da Altera, implica logicamente que ele possui um suporte bastante grande aos FPGAs Altera, e também uma documentação bastante extensa. Por outro lado, essa ferramenta possui desempenho e resultado inferiores na sua síntese quando comparado com a Symplify da Synplicity [51], mas facilidade de abstração e uso dão credibilidade ao Quartus.

3.3. SOPC Builder

É uma ferramenta da Altera usada para construir sistemas baseados no processador NIOS II, sendo executado em um FPGA Altera. Em linhas gerais, o SOPC Builder é um software para implementar dispositivos SOPC.

O SOPC Builder permite criar sistema simplesmente selecionando unidades funcionais e especificando seus parâmetros. É uma ferramenta que necessita de uma certa experiência em

hardware, pois o mesmo fornece uma grande quantidade de restrições a serem determinadas pelo usuário.

A princípio, logo que iniciado, o SOPC Builder realiza uma varredura em busca de sistemas ou plataforma previamente configurados. Nesse TG foi usada a plataforma básica, que é um software fornecido pela Altera que integra o conjunto do FPGA, e seus componentes já foram descritos.

O SOPC Builder tem como entrada no modo gráfico um arquivo com extensão .SOPC, e no modo interpretador de comandos um arquivo com extensão .PTF. Esses arquivos são gerados pela ferramenta Power ESL e serão detalhados nas próximas seções. A seguir encontra-se a Figura 17, com a representação dos processos realizados pelo SOPC Builder

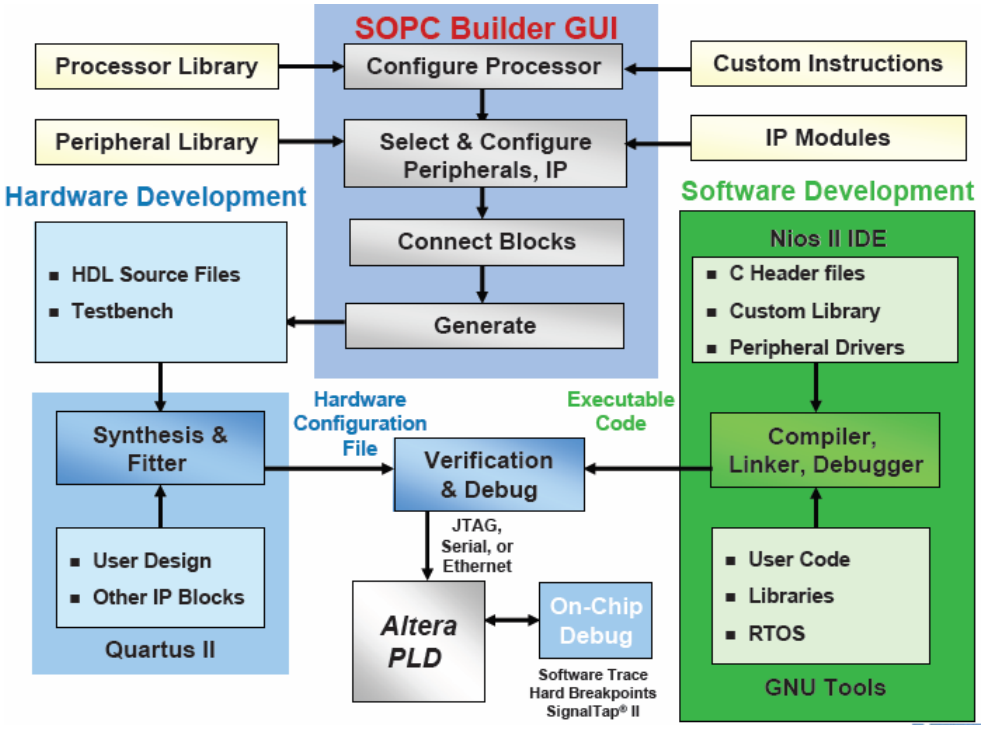


Figura 17. Diagrama de etapas das ferramentas Altera

O SOPC Builder, nada mais é, que um editor desses arquivos, transformando a abstrações de componentes, com ícones que podem ser clicados e arrastados, em um formato padrão de dados que pode ser entendido pelo computador para programar o FPGA e também o NIOS II IDE, responsável pela criação do software.

O SOPC Builder organiza todos os módulos do sistema fornecendo opções para conexão com o Barramento Avalon, atribuição automática de endereços e clock, e atribuição de restrições padrão para alguns componentes, bem como a determinação da quantidade e tipos de portas de entrada e saída dos módulos, através do utilitário Component Editor.

No contexto ESL o SOPC Builder é muito útil, pois quando é utilizada uma plataforma base, como nesse TG, esse software gera diversos arquivos HDL, scripts de compilação e os arquivos de definição de plataforma, abstraindo bastante a complexidade de um sistema computacional completo, deixando apenas a tarefa de definir o módulo externo para o usuário.

3.4. FPGA Altera DE2-35

O Altera DE2-35 é o FPGA usado para implementar os módulos usados no Estudo de Caso desse TG. Os FPGAs são chips programáveis formados por chaves controladas por software, que simulam o comportado de um componente hard-core. Uma descrição um pouco mais detalhada desse dispositivo já foi feita na seção 2.

No Altera DE2 todos os componentes são conectados aos pinos do chip, permitindo que o usuário configure a conexão entre eles. Esse FPGA permite a emulação tanto de experimentos simples, como contadores e temporizadores, como sistemas processados, que fazem uso de hierarquia de memória. Para exibição de informações e dados sobre o comportamento, ele conta com 27 LEDs, uma tela de LCD e display de 7 segmentos. Para a comunicação possui barramento serial RS-232, USB, Ethernet RJ-45, saída de vídeo composto e VGA, porta PS/2 para teclado, entrada para microfone e saída para alto-falante, porta USB Blaster, leitor de cartão, conexão infra-vermelho, entre outros. Uma visão geral do FPGA Altera DE2 é exibida na Figura 18 e um diagrama de blocos desse dispositivo encontra-se na Figura 19 a seguir.

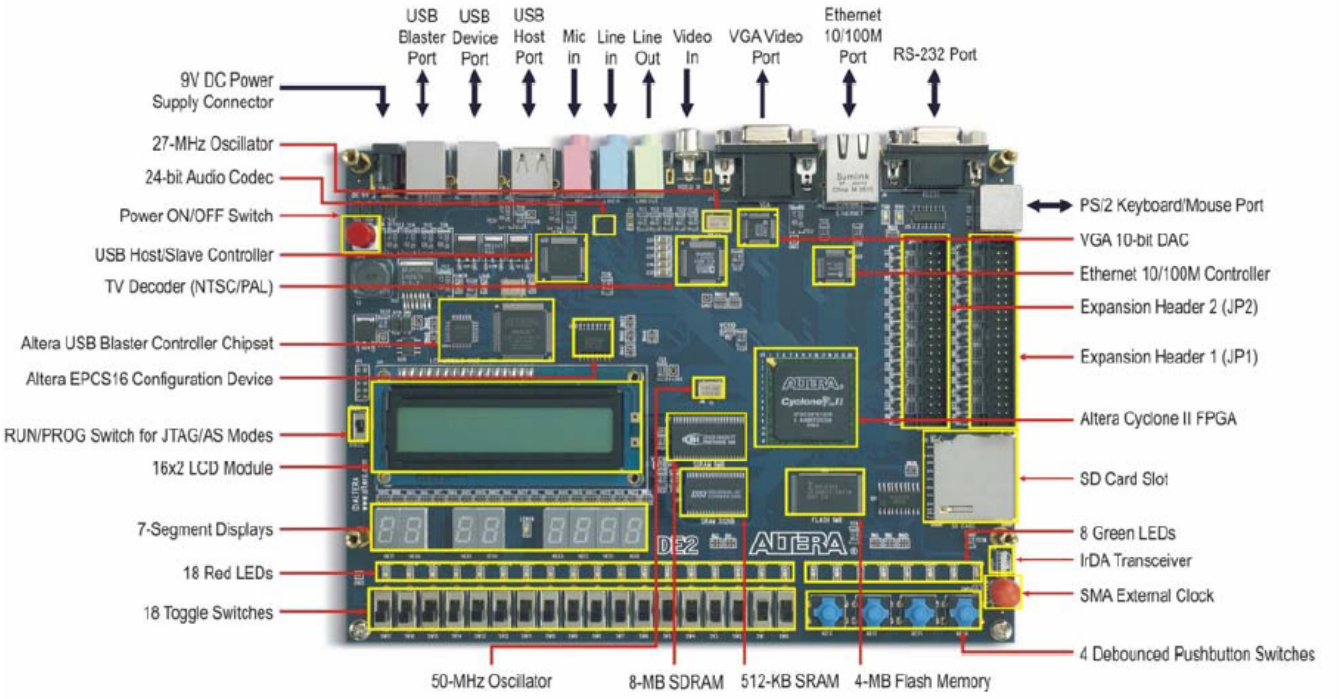


Figura 18. FPGA Altera DE2-35

Esse dispositivo possui clocks internos de 50 MHz e 27 MHz. Dessa forma, ele pode ter todos os seus componentes instanciados pelo Altera Quartus II, possibilitando que esses recursos sejam usados em projetos do usuário. O número que acompanha o modelo desse tipo de dispositivo da Altera, no caso em questão '35' (DE2-35), faz referência a quantidade aproximada de elementos lógicos presentes no equipamento.

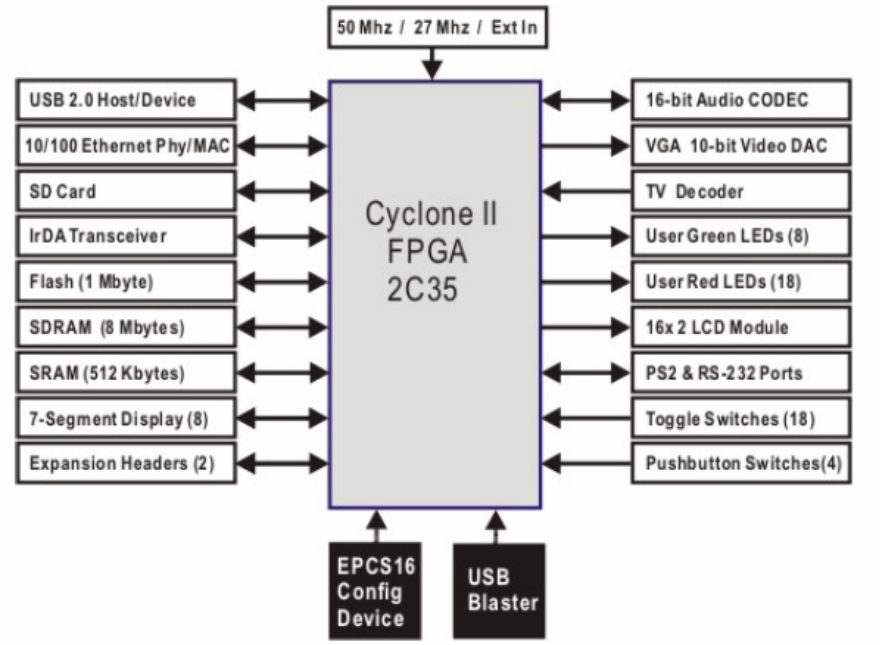


Figura 19. Diagrama de blocos do FPGA Altera DE2-35

4. Soluções ESL existentes

Os métodos tradicionais de desenvolvimento de hardware necessitam descrição e depuração manuais do RTL, consumindo muito tempo e abrindo espaço para possíveis erros imperceptíveis diante da grande complexidade que um sistema grande de hardware pode ter. Nesse sentido, é que foram criadas as ferramentas ESL, entre elas destacamos como forma de comparação a Mentor Catapult C Synthesis e a CoWare Platform Architect, não como forma de mostrar a superioridade de uma ou outra ferramenta, nem mesmo compará-las sistema desenvolvido nesse TG. Dessa forma, a descrição feita aqui, apenas introduz conceitos e tenta tornar mais claro o universo dessas ferramentas.

4.1. Mentor Catapult C Synthesis

É uma ferramenta EDA que pode ser usado para fazer síntese de alto nível, síntese algorítmica ou síntese ESL. Essa ferramenta suporta as linguagens ANSI C/C++ e SystemC e gera código RTL para FPGA ou ASIC.

O Catapult C [5] dá suporte a síntese em pipeline e possibilita a descrição de subsistemas multi-blocos em C/C++. Nessa ferramenta o usuário determinar apenas as restrições de tempo e área, determinar o clock e o tipo de dispositivo (FPGA ou ASIC), que se quer gerar. A Mentor Graphics disponibiliza uma biblioteca C completa para quem for desenvolver sistemas nessa plataforma. Sendo assim essa plataforma não possui bibliotecas C/C++ opensource.

A linguagem C usada nessa IDE é semelhante a normalmente encontrada nos demais projetos de software, com ponteiros, arrays e estruturas, com a adição de comandos que permite a criação e reuso de componentes RTL. O Catapult C também suporte a síntese lógica e possui interface gráfica para mostrar o aspecto estrutural do hardware construído.

Essa ferramenta também possui 3 opções de testbench: cíclica, RTL e gate-level. Ela permite ao projetista explorar micro-arquiteturas e opções de interface. Possui também opções de otimização automática de redução de potência. O Catapult C fornece uma visão completa do desempenho do

sistema e controle dos processos de síntese. Na Figura 20, é exibido um diagrama de etapas no funcionamento dessa ferramenta.

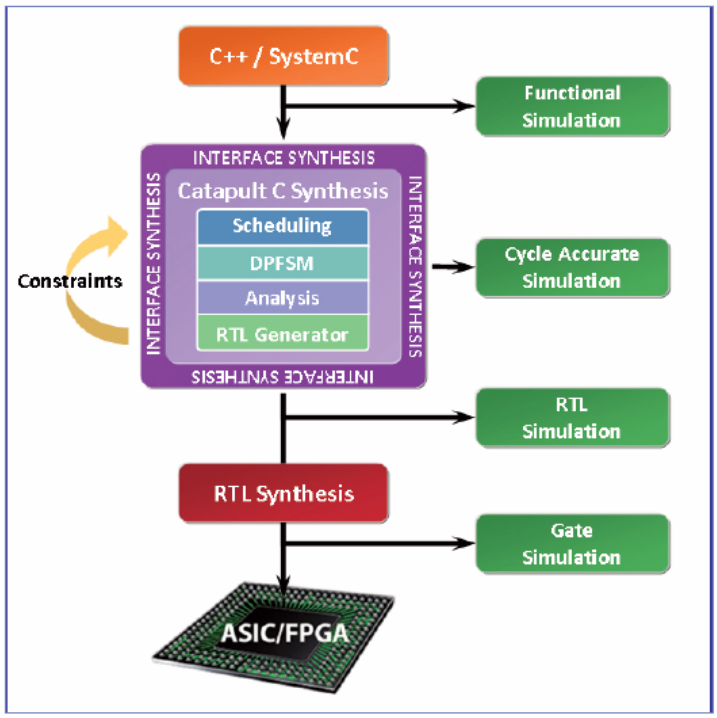


Figura 20. Diagrama de etapas da ferramenta Catapult C

4.2. CoWare Platform Architect

É uma ferramenta da CoWare, empresa que produz sistemas de software ESL, usada para prototipar sistemas embarcados abstraindo a complexidade e restrições físicas do hardware. Esse software proporciona uma rápida configuração para plataformas SoC. Ele também realiza análises de desempenho para arquiteturas feitas em SystemC, permite a uma eficiente abstração da complexidade de conexão de módulos de memória. O CoWare Platform Architect [6] habilita também simulação, depuração e análise dos sistema embarcados que estão sendo criados, com uma integração automatizada dos blocos RTL dentro do sistema **TLM**.

Essa IDE também favorece o reuso de componentes e periféricos definidos em SystemC, bem como uso de unidades de teste. Ela também pode ser integrada com ambientes de simulação HDL e verificação RTL de empresas com Cadence, Mentor e Synopsys.

5. Ferramenta Power ESL

Nessa seção são apresentados o contexto, objetivo e arquitetura da ferramenta desenvolvida nesse TG.

5.1. Projeto de síntese de comunicação

Para compreender o funcionamento da ferramenta Power ESL, desenvolvida nesse TG, é preciso explicar que ela pode ser usada para completar o fluxo de geração automática de SoC de um outro projeto, trata-se do Projeto de Síntese de Comunicação da Mestranda Millena de Andrade Almeida Gomes (maag@cin.ufpe.br). Esse projeto visa a produção de sistemas embarcados, a partir de uma descrição de alto nível, identificando dessa forma, o caráter desse projeto como ESL. A estratégia geral de desenvolvimento é mostrada na Figura 20.

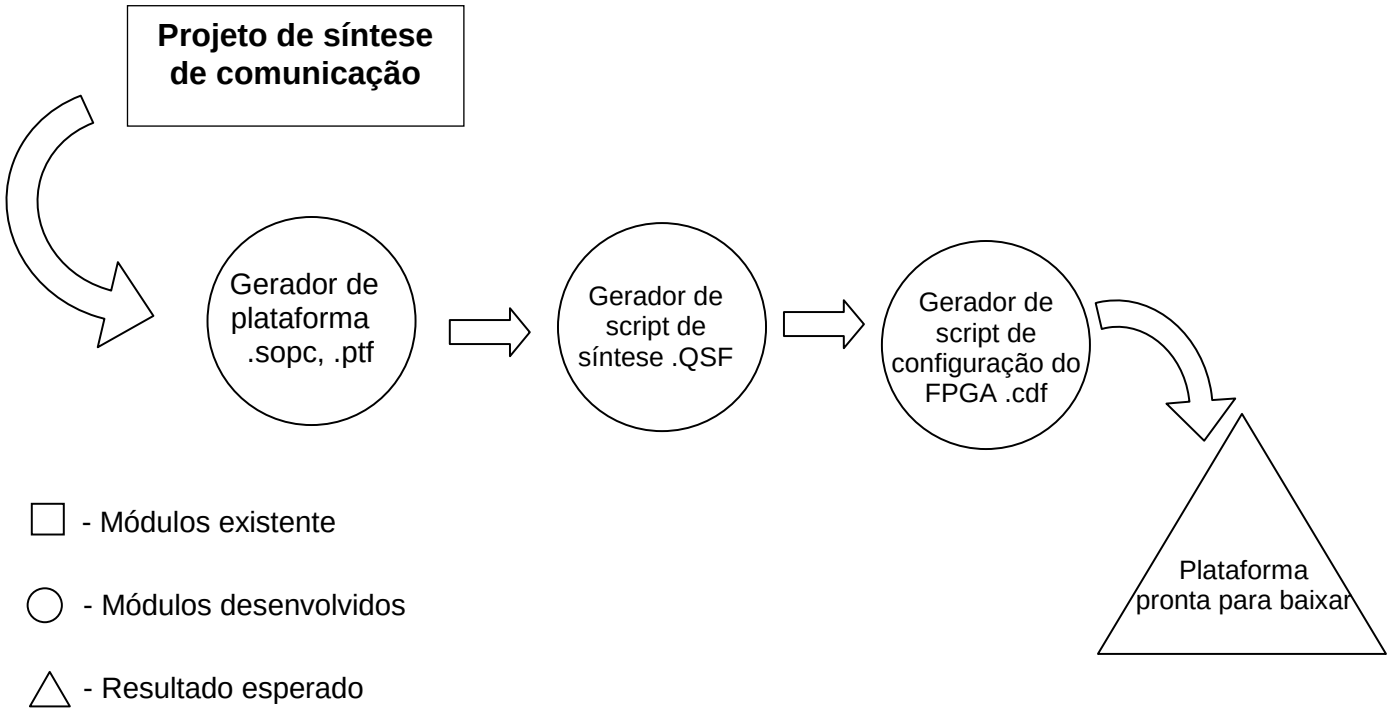


Figura 21. Esquema geral dos experimentos

O projeto de Síntese de Comunicação dessa Mestranda utiliza UML para descrever os módulos e determinar algumas restrições e relações entre os componentes do sistema embarcado. Uma ferramenta de interpretação e tradução conhecida como parser é usada nesse projeto, lendo as informações da linguagem UML, e produzindo arquivos HDL (.v, .vhdl). Esses HDLs descrevem o

funcionamento do sistema, já as restrições, tais como tempo de espera para leitura e escrita no módulo, latência, número de interrupção, entre outros, são escritas em um arquivo XML.

5.2. Objetivo

O sistema desenvolvido nesse TG tem o objetivo de automatizar todas as tarefas de configuração, feitas em ambientes gráficos como Quartus II, abstraindo toda a complexidade, retendo, configurando e aproximando restrições e resultados de uma IDE convencional. Essa ferramenta assume tarefas que teriam que ser feitas por um usuário com um conhecimento relativamente alto em características físicas dos módulos. Para fazer isso a Power ESL utiliza como entrada justamente os arquivos de saída do projeto de Síntese de Comunicação mencionado anteriormente, ou seja, uma descrição XML e os arquivos HDLs. Essa ferramenta também dá suporte à manipulação gráfica do sistema embarcado, se desejada pelo usuário, e fornece os mesmo relatórios de exceções da ferramenta tradicional, no caso desse TG o Altera Quartus II.

5.3. Arquitetura

Nessa seção é mostrada a arquitetura geral da ferramenta, mostrando os módulos e suas interligações. Antes de mais nada, é preciso ter uma visão geral do resultado esperado, após a execução da Power ESL, isso ajuda a contextualizar a sua atuação, dessa forma, um esboço da arquitetura de uma plataforma básica Altera, com alguns módulos integrados é mostrada na Figura 22 a seguir:

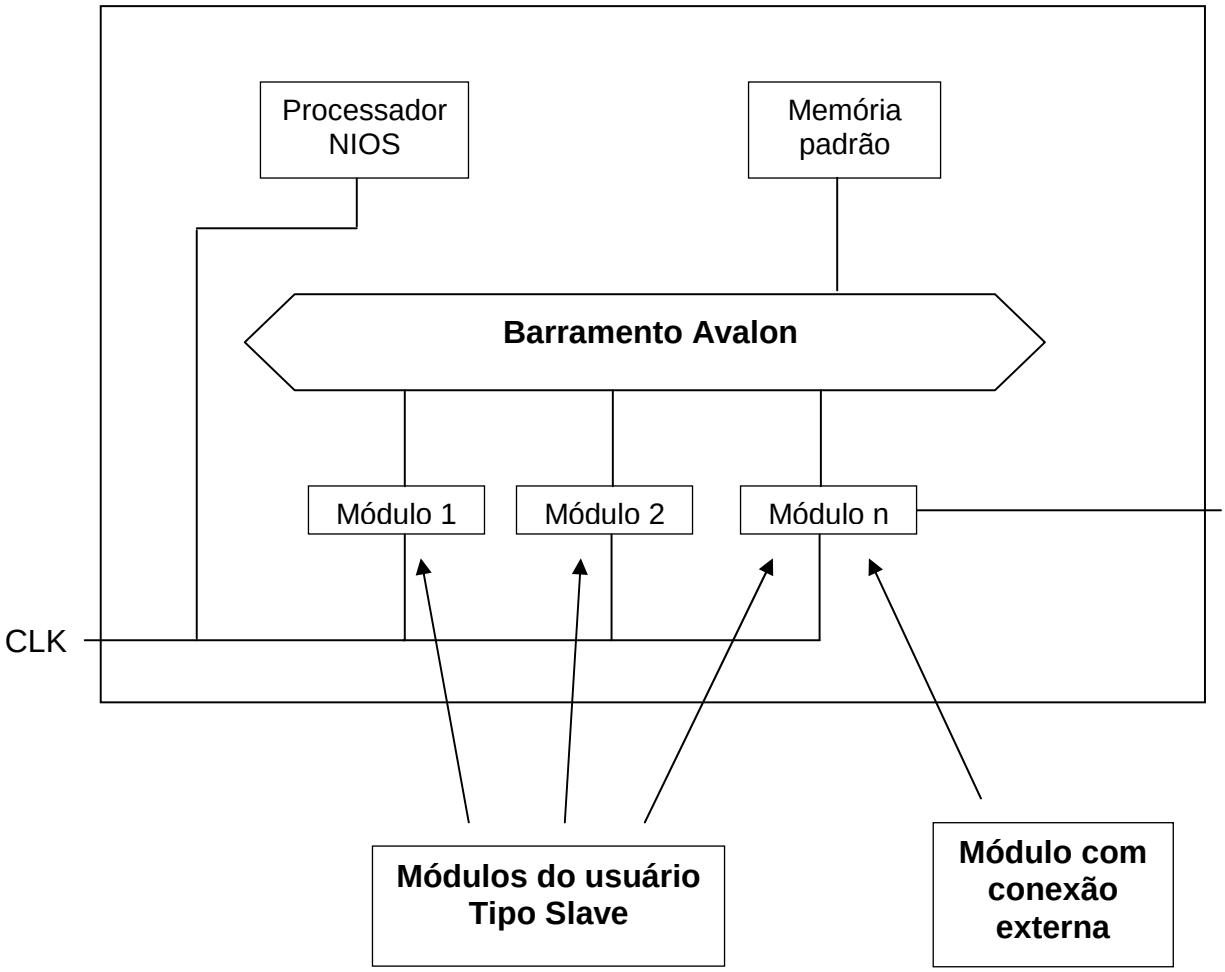


Figura 22. Arquitetura básica com os módulos do usuário adicionados

No cenário mostrado, apenas os componentes da arquitetura de Von Neuman (barramento, processador, memória e I/O), foram mostrados, alguns módulos foram omitidos, mas estão presentes na plataforma básica e já detalhados nesse TG.

É importante notar que os módulos geralmente podem ser de 2 tipos:

- Módulos com conexão externa - tipo UART, saída de vídeo composto;
- Módulos sem conexão externa - tipo memória, co-processador.

Esses fatores foram pensados e foi decidido colocar a mesma estrutura de XML para ambos, a diferenciação é feita pela presença ou não da informação de porta externa, dessa forma, o parser XML ficou mais simples.

Para atingir esse resultado a Power ESL foi construída em camadas para facilitar a manutenção e depuração de seus módulos. Sua programação foi toda feita em Java, orientada a objetos, com o interesse de facilitar uma futura integração com a plataforma PDesigner.

Na verdade, o algoritmo desenvolvido trabalha exclusivamente com a semântica existente dentro dos arquivos de sistemas usados pelo Quartus II e o SOPC Builder. Essa ferramenta também organiza as chamadas a determinados aplicativos da Altera, ou seja, a Power ESL funciona gerenciando as informações fornecidas pelo projeto de síntese de comunicação, modificando arquivos existentes na plataforma básica e atribuindo comandos que geram novos arquivos. Esse processo se repete até a geração do .SOF, arquivo que quando utilizado configura todo o FPGA, num processo conhecido como “baixar”. Para tentar introduzir e contextualizar a forma de execução da Power ESL, um fluxo de funcionamento é mostrado na Figura 23 a seguir.

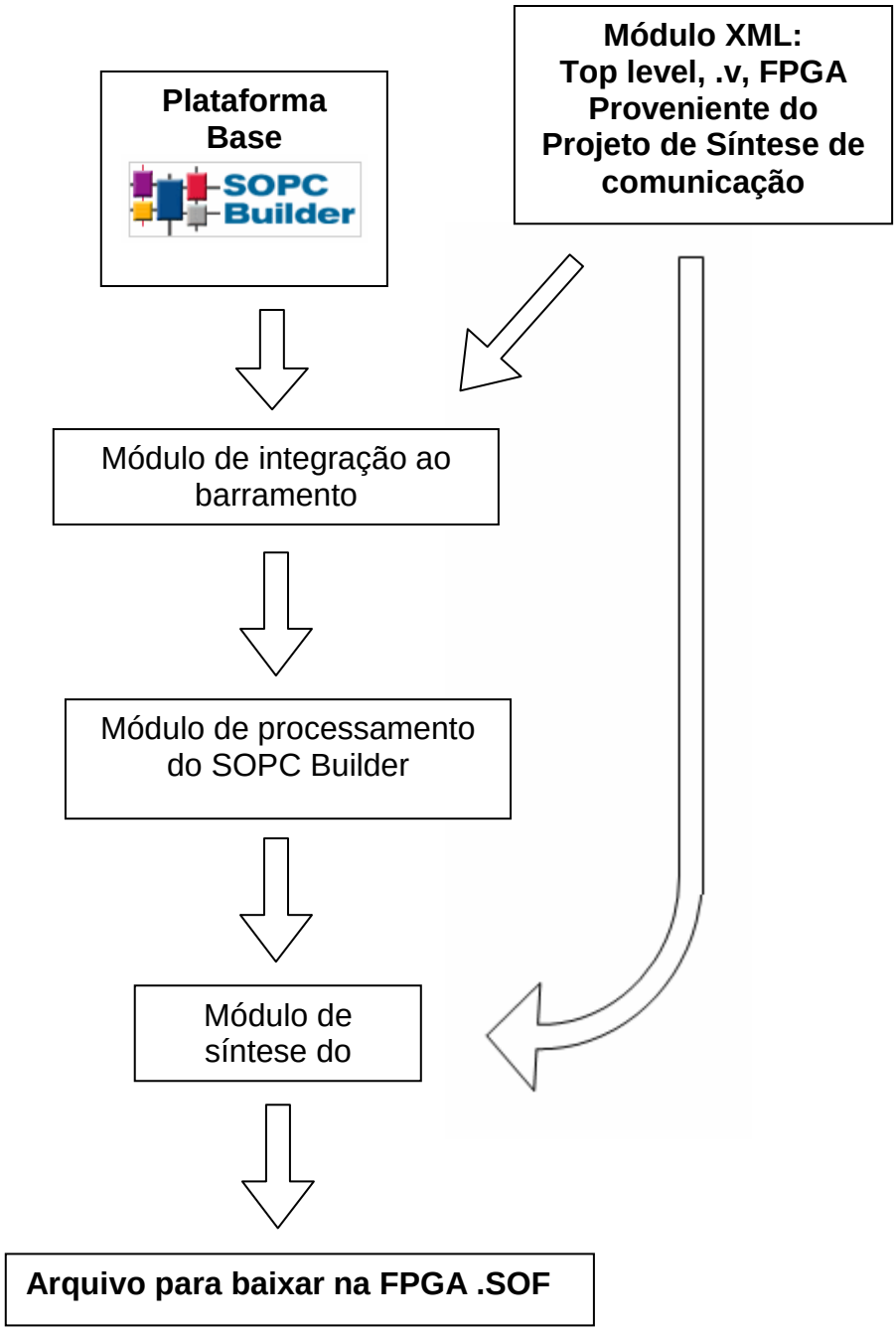


Figura 23. Fluxo de execução da ferramenta Power ESL

Módulo de integração ao barramento - recebe uma plataforma pronta e adiciona módulos HDL ao sistema, integrando ao Barramento Avalon, de acordo com as especificações de entrada do arquivo .XML.

Módulo de processamento do SOPC Builder - recebe um sistema integrado ao barramento no formato <.sopc> retornando um projeto do Quartus, com uma descrição em formato .QIP, e sistema

com Top level modificado de acordo com o XML, gerando plataforma pronta para NIOS IDE .PTF, bem como arquivo .SOPC, usado para modificação do projeto através do ambiente gráfico tradicional.

Módulo de síntese - realiza síntese do projeto através dos scripts QSF. Os QSF são compostos tantos por arquivos de descrição XML como por arquivos resultantes do processamento do SOPC Builder.

5.4. Funcionamento

Nessa seção serão detalhadas todas as etapas, tecnologias e estratégias de implementação da ferramenta Power ESL, analisando todos os arquivos de entrada e saída. Para iniciar a explicação, é mostrado, na Figura 24, um diagrama de classes da ferramenta com todas as abstrações da ferramenta feitas em Java.

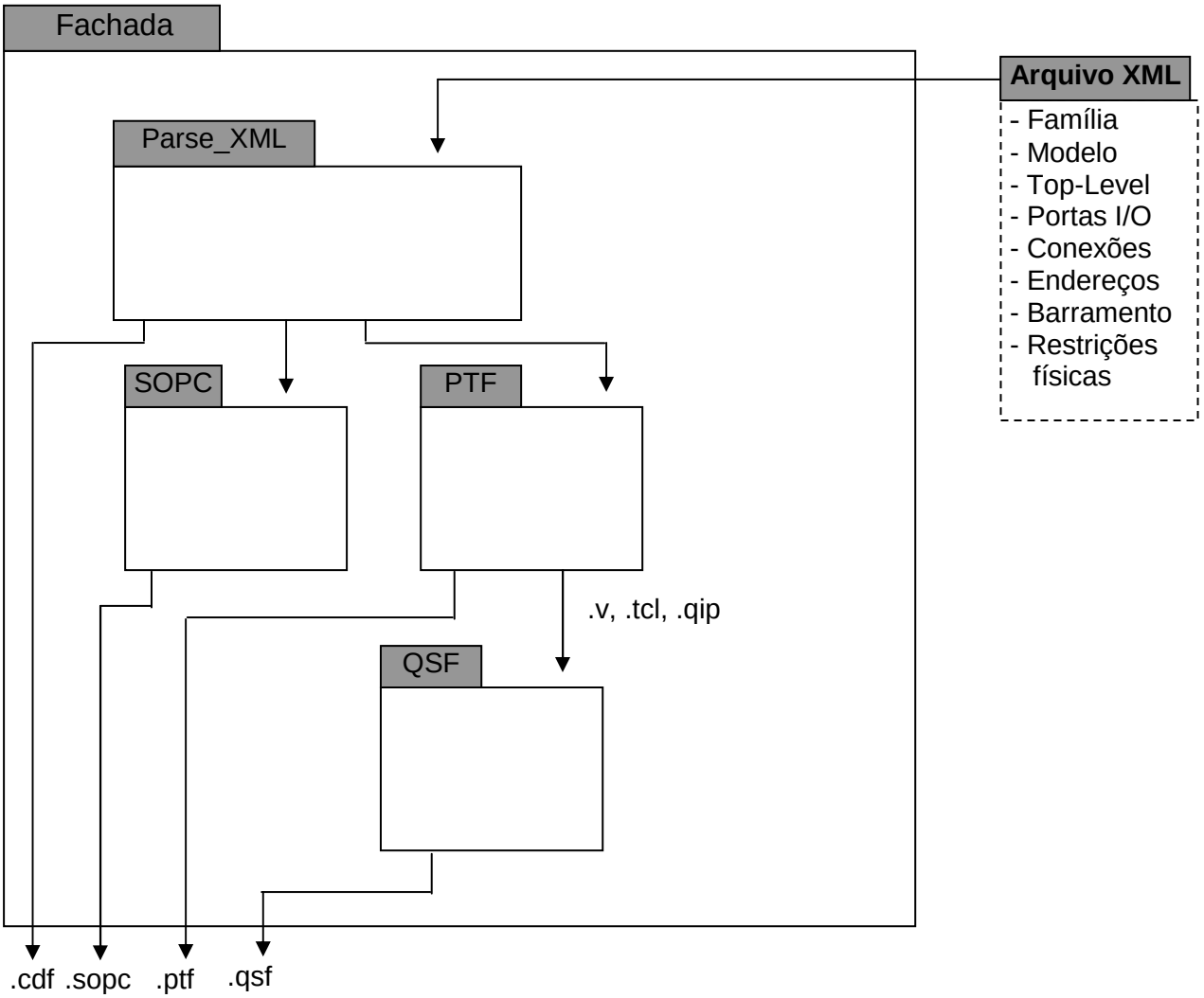


Figura 24. Diagrama de classes da ferramenta Power ESL

5.5. Parser_XML

O módulo principal da ferramenta, como foi mostrado na Figura 24 é representado por um arquivo principal, a classe Fachada, que possui como única finalidade instanciar as outras classes do sistema. Dessa forma, internamente temos a classe Parser_XML e como o próprio nome sugere, é a parte da ferramenta que executa um algoritmo de parser de um arquivo XML de entrada. Nesse arquivo, estão todas as informações do sistema a ser gerado como família e modelo do FPGA, top level do sistema e as informações dos módulos que serão detalhadas no decorrer da descrição das outras classes.

A forma de leitura desse arquivo é o DOM. Essa metodologia foi desenvolvida pela W3C e orienta os desenvolvedores como fazer uma varredura por dados em arquivos XML. Ela foi escolhida por permitir uma busca por qualquer percurso, uma vez que os dados ficam estruturados na memória. A tecnologia alternativa, o SAX possui varredura com percurso seqüencial, e que não seria interessante, pois os módulos do usuário não precisam necessariamente de uma seqüência de inclusão dentro de uma plataforma. No entanto, o SAX possui desempenho melhor em boa parte dos parsers.

Sendo assim, os módulos em Java da ferramenta Power ESL, foram implementados de forma a refletir a mesma organização dos dados no arquivo XML. Esse maneira de implementar tornou a compreensão do funcionamento mais simples e favoreceu uma depuração eficiente.

A classe Parser_XML também é responsável pela criação do arquivo .CDF. Essa extensão significa Chain Description File, e tem a função de automatizar o processo que “baixa” o arquivo .SOF diretamente no FPGA. Para fazer isso 3 informações são essenciais:

- O modelo do FPGA - essa informação é extraída do XML no campo *model*;
- O nome do arquivo .SOF - essa informação é retirada também do XML no campo *top_level*;
- A localização do .SOF - é a informação de caminho que é passada internamente pela ferramenta e indica o local onde o .SOF está para ser baixado e poder configurar o FPGA.

Nas seções seguintes serão descritas as 3 classes que produzem os arquivos utilizados na geração da plataforma do FPGA.

5.6. SOPC

Essa classe possui o mesmo nome da extensão do seu arquivo de saída. Esse arquivo realiza a integração de informações de uma plataforma que pode ser visualizada pelo SOPC Builder. O objetivo de produzir esse arquivo é justamente fornecer ao usuário a opção de poder visualizar o projeto na interface gráfica do SOPC Builder. Para poder adicionar um módulo dentro SOPC Builder é preciso editar o arquivo .SOPC, adicionando as seguintes informações:

a) Informações Avalon - essas informações descrevem como o elemento se ao Barramento Avalon. No XML a tag de identificação é <avalon_element>. Seus dados são:

- Nome da conexão - é o nome que identifica o módulo dentro do SOPC Builder. Para um reconhecimento do componente, existe um padrão para esse nome que é: <nome do módulo> + < _inst.avalon_slave_0>. Exemplo: wrapper_uart, tem nome de conexão, wrapper_uart _inst.avalon_slave_0;
- Referência do endereço - especifica o tipo de endereço do módulo que pode ser endereço base ou sortIndex;
- Endereço - é o valor utilizado para identificar o módulo;
- Tipo de endereço - especifica o tipo de valor do endereço.

b) Informações do módulo - essas informações são usadas para organizar a sequência dos módulos dentro do SOPC Builder e no XML possui a tag < simple_element>. Seus dados são:

- Nome do elemento - é o próprio nome do módulo;
- Referência do endereço - especifica o tipo de endereço do módulo que pode ser endereço base ou sortIndex;
- Número de sequência - é um número especifica uma ordem para organizar os módulos;

- Tipo de endereço - especifica o tipo de valor do endereço.

c) Identificação no Component Editor - são informações usadas para identificar os módulos dentro do aplicativo Component Editor. No XML é especificado pela tag <component>. Os dados são:

- Nome do módulo - é utilizado para identificar o módulo no Component Editor, no momento de especificar uma conexão. Seu padrão é completar o nome do módulo com a terminação <_inst>;
- Tipo de módulo - é o campo onde fica propriamente o nome do módulo;
- Versão - informa qual é a versão do módulo, geralmente todos os módulos tem o valor 1.0 nesse campo.
- Tag de habilitação - é campo indica de o módulo está habilitado para uso.
- Parâmetros - eventualmente o módulo pode possui parâmetros, que podem ser usados para especificar um barramento de dados ou endereço, informando também o tamanho desse barramento.

d) Informações de conexão com Avalon - representam as propriedades da conexão que cada módulo deve possui para se comunicar com o barramento. É identificado no XML pela tag <connection> e seus dados são:

- Tipo - essa informação especifica a que é o tipo de barramento o módulo está ligado, nesse caso todos estão ligados ao Avalon;
- Versão - especifica qual foi a versão do Quartus usada para construir a plataforma, é uma informação importante para questões de compatibilidade;
- Start - essa informação indica o ponto de partida da conexão, que é o NIOS II ou a saída de um PLL;

– End - especifica o ponto de chegada do sinal transmitido por essa conexão e deve conter o <nome do módulo> + < _inst.avalon_slave_0>;

– Parâmetros - são informações que especificam a prioridade de acesso a recursos desse módulo, e também o endereço base usado para identifica esse módulo dentro do sistema.

e) Informações de conexão com processador - representam as propriedades de conexão com o NIOS II. No XML é identificado também pela tag <connection>, mas seus dados possuem nomes diferentes da informações de conexão com o Avalon. Seus dados são:

– Tipo - especifica o tipo de clock, pois dentro da plataforma, com o uso de PLLs é possível ter diversas freqüências de clock;

– Versão - essa informação indica em qual versão do Quartus II foi implementado o módulo;

– Start - essa informação indica o ponto de partida da conexão, que é o NIOS II ou um PLL;

– End - especifica o ponto de chegada do sinal transmitido por essa conexão.

Essas são as informações para adicionar módulos básicos, outros componentes como display de 7 segmentos, memória de vídeos, precisam de bem mais dados e restrições para serem exibidos no SOPC Builder.

5.7. PTF

Essa classe também possui a mesmo nome da extensão do arquivo que ela produz. Esse arquivo é o responsável pela orientação do SOPC Builder durante o processo de geração da plataforma [52]. Durante esse processo, é criada uma série de arquivos de configuração, bem como HDLs. As principais extensões criadas são:

– .**v** ou .**vhdl** - são arquivos de descrição de hardwate;

– .**tcl** - são scripts que orientam a síntese interna de alguns módulos;

- .**qip** - é um arquivo de descrição de componentes, que instancia o processador e os módulos top_level do usuário.

Retomando, o .PTF não é um abreviatura e significa apenas plataforma e também é usado para configurar o NIOS IDE. Esse é um ambiente de desenvolvimento de software para o hardware construído pelo usuário. O .PTF é um arquivo bem longo e chega a mais de 6000 linhas de código na plataforma básica. Esse tamanho é devido ao nível de detalhamento requerido para produzir uma plataforma. As informações e detalhes necessários para adicionar um módulo na plataforma básica são mostrados a seguir:

a) Informações de entrada - são informações sobre os módulos mapeados em memória. No XML elas se encontram logo abaixo da tag <module> e seus dados são:

- Nome - é o nome do módulo acompanhado da terminação _inst. Essa terminação como já foi dito faz parte de um padrão usado para identificar os módulos no SOPC Builder.
- Endereço base - é usado para identificar o módulo.
- Endereço span - é usado para definir qual span o módulo ocupa.
- Is brigde - essa informação indica se o módulo é um barramento, possui o valor ‘0’, com exceção apenas se o campo bridge_to contiver algum valor. Essa informa está contida no trecho *SYSTEM_BUILDER_INFO*, que será explicado adiante.

b) Portas - representam os sinais de I/O do módulo. No XML são representados pela tag <input_ports> e <outpu_ports> para entrada e saída respectivamente. Os dados para representar uma porta são:

- Nome - especifica o nome da porta;
- Tipo - é o identificador dentro do SOPC Builder;
- Direção - especifica a direção para os dados da porta;
- Is enable - esse dado indica se a porta está habilitada.

c) Informações de construção do sistema (SYSTEM_BUILDER_INFO) - são restrições sobre o módulo para poder se configurar uma série de características e montar a tabela de mapeamento na memória. Essas informações são necessárias para a ferramenta Generator, poder construir o sistema e fornecer suporte ao usuário. Sendo assim, todas as informações dessa seção do .PTF são descritas a seguir:

- Bus type - especifica o barramento, em plataformas da Altera em geral é o Avalon;
- Write Wait State - representa o número de ciclos de clock que o módulo deve esperar para realizar uma escrita;
- Read Wait States - representa o número de ciclos de clock que o módulo deve esperar para realizar uma leitura;
- Hold time - esse sinal indica o número de ciclos de clock entre o momento em que o sinal de escrita é ativado e o momento em que o sinal chipselect é ativado.
- Setup time - é o tempo que o SOPC Builder precisa para mostrar um endereço válido antes de confirmar uma leitura ou escrita.
- Is Printable Device - esse sinal indica se o módulo uma escolha para depuração com a função printf();
- Address Alignment - esse sinal informa ao Barramento Avalon se ele deve fazer a correção de bits durante a transferência de dados em uma porta. Se esse dado for configurado como native, durante o casamento entre um dado de 32 bits com um dado de 16 bits, por exemplo, o Avalon vai receber um valor de 32 bits, mas considerando apenas o 16 bits menos significativos. No entanto, se for configurado como dynamic, durante a mesma conversão discutida acima o Avalon vai ler os 32 bits, preenchendo com 4 palavras de 8 bits;
- Well Behaved Waitrequest - esse sinal indica se o módulo precisa esperar algum tipo de resposta;
- Is Nonvolatile Storage - esse sinal informa se o módulo é um dispositivo de armazenamento volátil;

- Read Latency - é um valor que representa um cálculo de quantos ciclos serão necessários para o dado solicitado esteja pronto para leitura;
- Is Memory Device - esse dado se for ‘1’ indica que o módulo deve ser conectado ao barramento de dados e instruções, por outro lado, se for ‘0’ indica que deve ser conectado apenas ao barramento de dados;
- Maximum Pending Read Transactions - esse sinal indica se o módulo não tem sinal de readdata válido;
- Minimum Uninterrupted Run Length - esse sinal indica mínima quantidade de ciclos de execução ininterrupta;
- Accepts_Internal_Connections - indica se o módulo aceita conexões internas;
- Write Latency - é um valor que representa um cálculo de quantos ciclos serão necessários para o dado solicitado esteja pronto para a escrita;
- Is Flash - indica se o módulo é do tipo flash;
- Data Width - esse sinal representa quantos bits tem o barramento de dados do módulo;
- Address Width - esse sinal representa quantos bits tem o barramento de endereços do módulo;
- Maximum Burst Size - tamanho do maior dado transmitido em paralelo;
- Register Incoming Signals - se esse sinal for ‘1’ o módulo é ligado na entrada Avalon tristate, se for ‘0’ a ligação é no Avalon convencional;
- Register Outgoing Signals - se esse sinal for ‘1’ o módulo é ligado na saída Avalon tristate, se for ‘0’ a ligação é no Avalon convencional;
- Interleave Bursts - esse sinal indica se esse módulo permite transferência de dados com fluxo rápido.
- Linewrap Bursts - esse sinal informa se o módulo implementa o wrapper para conversão de dados nas transferências de dados.

- Burst On Burst Boundaries Only - esse sinal informa se as transferências de dados são garantidas ser nos endereços múltiplos do sinal Burst Size;
- Always Burst Max Burst - esse sinal indica a máxima transferência;
- Is Big Endian - esse sinal define se o conversor Endian é usado, o mesmo reduz o tempo de execução de algumas operações;
- Is enable - indica se o módulo está habilitado;
- Priority - indica o nível de prioridade do uso de CPU do módulo;
- Offset Address - é o endereço do módulo dentro do sistema;
- Base Address - é o menor endereço do módulo visível pelo barramento
- Address Group - esse sinal indica se o endereço está agrupado;
- Has IRQ - indica se o módulo tem porta de saída para requisição de interrupção;
- IRQ Number - é o valor que representa a prioridade de requisição de interrupção do módulo e deve ser 'NC' caso o módulo não tenha interrupção;

d) Informações de restrições do sistema - são informações utilizadas para caracterizar o módulo e no XML estão representadas pela tag <system_constraints>. Seus dados são:

- Class - esse sinal especifica o nome da classe do módulo, permitindo ao SOPC Builder e ao Generator extrair relevância de informações dentro do arquivo .PTF;
- Class version - esse sinal é usado pelo SOPC Builder e o Generator para determinar se o .PTF foi criado em um versão compatível com a que está executando atualmente, dependendo dessa informações é possível que nem todas os componentes sejam mostrados;
- GTF class name - é o nome do top_level do módulo;
- GTF class version - é a versão atual do módulo;

- Do Not Generate - esse sinal indica se o módulo não é gerado automaticamente pelo Generator;
- Instantiate In System Module - indica se o módulo está instanciado no sistema;
- Is brigde - essa informação indica se o módulo é um barramento, possui o valor ‘0’, com exceção apenas se o campo bridge_to contiver algum valor.
- Is enable - esse dado indica se a porta está habilitada.
- Clock Source - indica a qual clock o módulo está associado;
- Has Clock - esse sinal indica se tem clock no módulo;
- Simulation HDL Files - são arquivos gerados pelo SOPC Builder, relacionados ao módulo e são usados para fazer simulação.

5.8. QSF

É mais uma classe que como as outras tem o mesmo nome do arquivo de configuração que ela produz. O .QSF é o arquivo que orienta todo o processo de síntese, fitter, roteamento e análises (tempo e área) do módulo. Ele também é coordena a geração do .SOF, que programa todo o FPGA de acordo com o projeto. Os dados para adicionar um módulo na plataforma básica Altera são:

- Família - é a família do FPGA, nesse TG foi usado Cyclone II;
- Modelo - é o modelo do FPGA, nesse TG foi usado o EP2C35F672C6;
- Top level - é o top level do projeto, não confundir com o top level do módulo;
- Configuração - essa informação indica o tipo de síntese;
- Arquivo HDL - são os arquivos dos módulos criados.

5.9. Estrutura dos comandos

Nessa seção será mostrado através de um diagrama a estrutura dos comandos utilizados em Linux para chamar as ferramentas necessárias, para um sistema partir de uma descrição XML e poder configurar um FPGA baixando o .SOF.

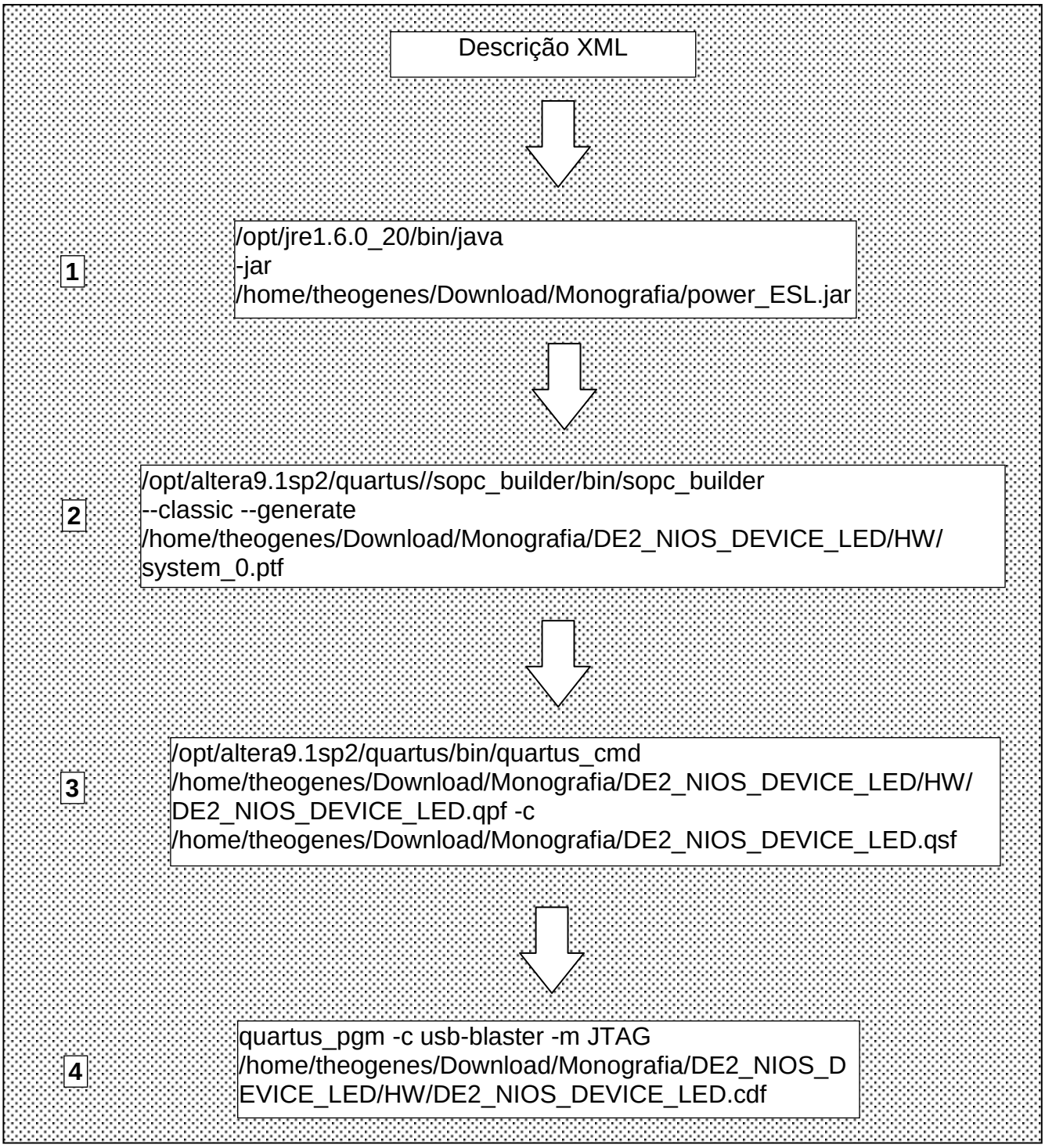


Figura 25. Estrutura de comandos da ferramenta Power ESL

Esse é o mesmo fluxo que a ferramenta gráfica da Altera realiza. É importante ressaltar ainda que todo esse processamento é orientado por um script de execução Linux .sh. Dessa forma, no passo 1, mostrado na Figura 25, é feita a chamada do executável Java da ferramenta Power ESL, gerando os arquivos já descritos acima. No passo 2, o sistema é gerado com o uso da ferramenta Generator, tendo como entrada um arquivo .PTF. No passo 3 é feita a compilação do sistema completo, com síntese, fitter, roteamento, entre outros processos, além da geração do .SOF. No 4º e último passo é onde acontece a configuração eletrônica do FPGA com o arquivo .CDF, passando o tipo de barramento de comunicação utilizado.

6. Estudo de Caso

Nesta seção está descritos os módulos usados como teste na ferramenta Power ESL. Esses módulos são uma UART e um módulo de memória.

6.1. UART

O UART é um circuito encontrado dentro das portas seriais que controla a transmissão e recepção de dados. Esse componente tem a capacidade de receber 8 sinais de dados simultaneamente de retornar um saída serial estruturada. O UART [53] também permite a transmissão e recepção ao mesmo tempo e a representação de um barramento serial é mostrada na Figura 26 a seguir.

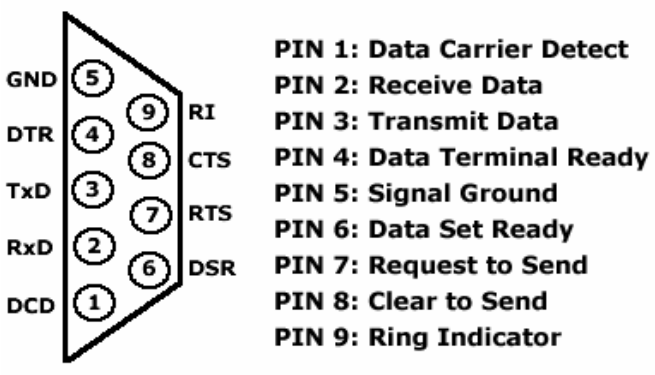


Figura 26. Distribuição de pinos em um barramento serial

O UART utilizado nesse TG é do tipo 16550 que tem a característica de oferecer transmissão e recepção bufferizada com 16 bits.

6.1.1. Arquitetura geral da UART modificada

A UART 16550 está descrita no datasheet que pode ser encontrado no endereço http://www.national.com/ds/PC/PC16550D.pdf , e sua arquitetura possui aspecto mostrado na Figura 27:

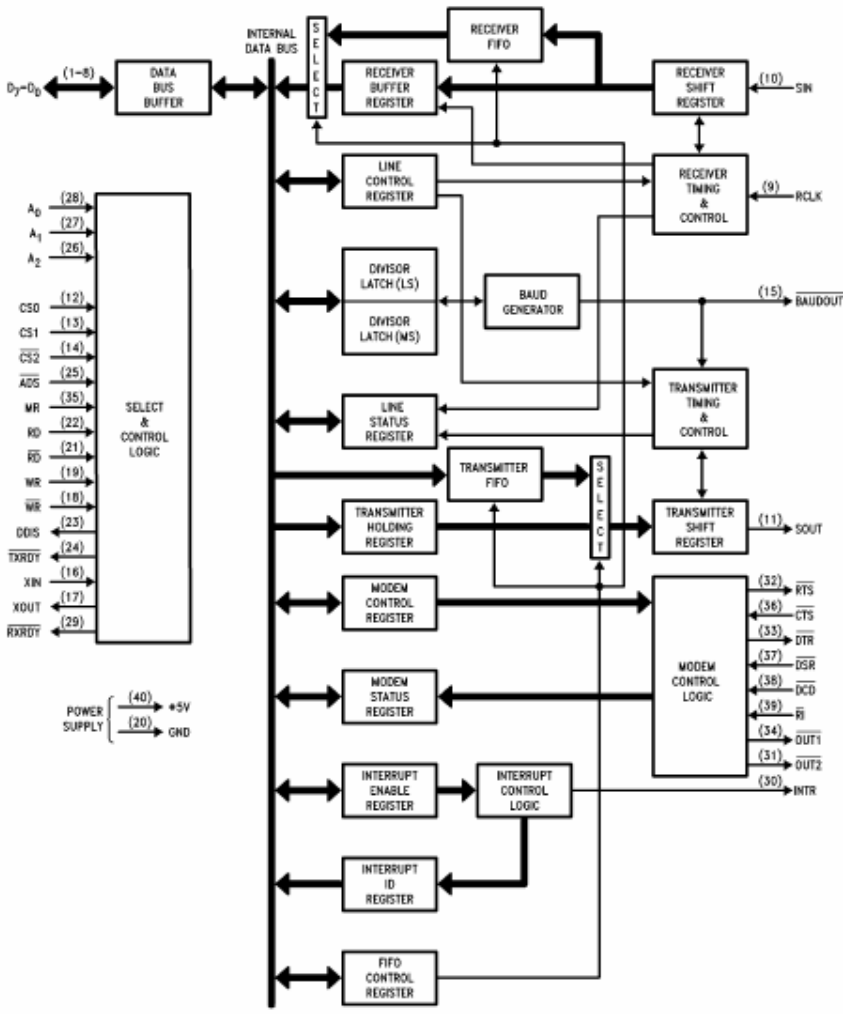


Figura 27. UART 16550 completa

Entretanto a arquitetura do projeto é mais restrita sem controle de modem, tri-state e sem capacidade de diagnóstico sendo definida da forma mostrada na Figura 28:

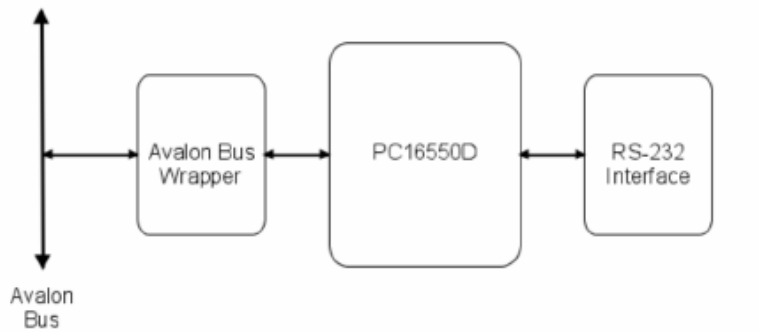


Figura 28. Diagrama de blocos da interligação entre UART e o Avalon

A arquitetura mais detalha da UART modificada se encontra na Figura 29 a seguir:

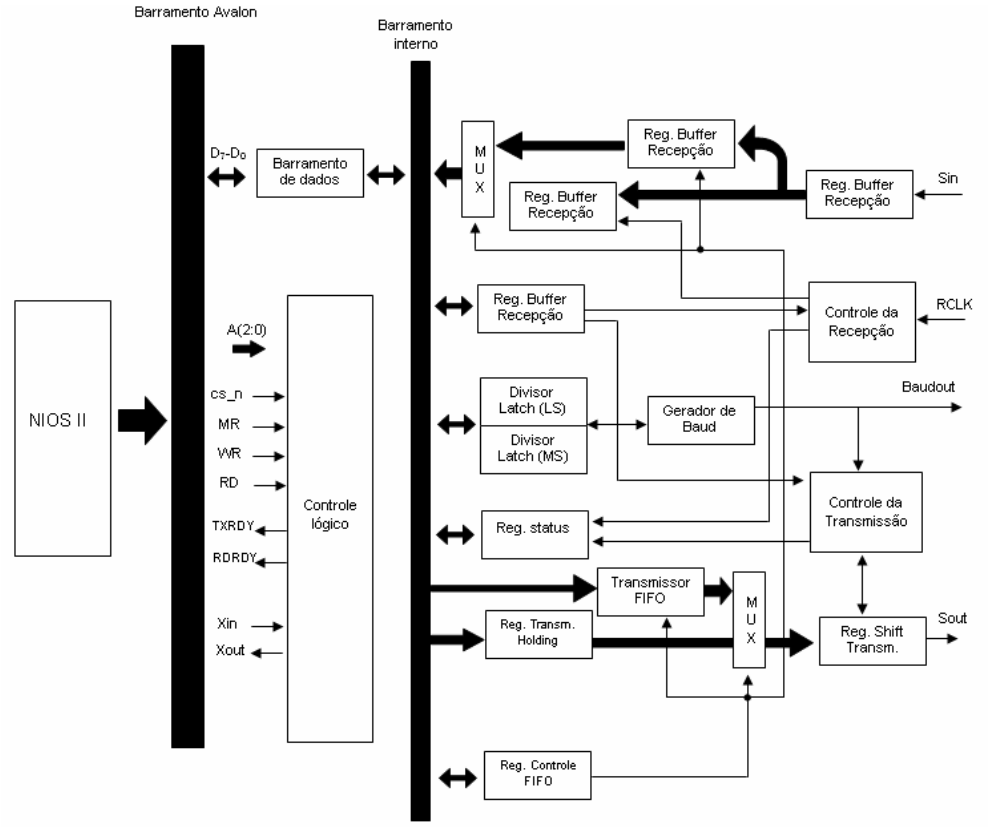


Figura 29. UART 16550 modificada

Possuindo menos pinos que a original, uma vez que as funções relativas ao modem e interrupções não foram implementadas. Devido ao fato da plataforma básica da Altera possui sua própria UART, foi necessário retirá-la para realizar os experimentos. O resultado foi bom, com a transmissão funcionando perfeitamente, no entanto a recepção teve problemas de restrições temporais e no sinal que controla o funcionamento da UART conhecido como baudrate. Contudo, o escopo desse TG é realizar um fluxo entre uma descrição XML e a configuração do FPGA, esse etapa funcionou normalmente.

6.2. Módulo de memória

É o outro módulo usado para testar a ferramenta Power ESL. Esse componente é uma memória principal simples, possuindo 7 portas que são: clock, endereço, dados, chipselect, writedata, write e read. Seu funcionamento é bem simples, e consiste na leitura de um dado e escrita no Barramento Avalon. O resultado no processo da ferramenta Power ESL com esse módulo, não apresentou problemas.

7. Conclusão

Analisando a proposta e o resultado é possível concluir que o a ferramenta implementada nesse TG realmente aumenta o nível de abstração no momento de se projetar um sistema embarcado, confirmando sua característica ESL. Evidentemente, esse trabalho é apenas um começo, e precisa de muitas otimizações para ser compatível com mais IDEs e também FPGAs. Posteriormente, outro desafio importante é aumentar a performance desse tipo de ferramenta. A evolução dos compiladores é um bom exemplo do caminho que um sistema de tradução de linguagens tem que percorrer para atingir uma qualidade de sistema compilado aceitável.

Por outro lado, pensar que um sistema construído com ESL vai ser mais eficiente do outro sistema feito em nível estrutural, ainda é precipitado. O que é mais provável é o aumento de produtividade. O aspecto de desempenho pode ser compensado com a evolução das tecnologias de hardware. Entretanto pensando nas vantagens da ferramenta Power ESL, é possível notar que seu processamento é mais rápido que o realizado pela interface gráfica, e também a forma de compreender a função do módulo dentro do sistema produzido é mais simples. Como desvantagem, existe o fato de ser mais complicado visualizar as interações dos componentes e também o relatório ser mais difícil de ser lido, pois todo ele é retornado no terminal do Linux.

De modo geral, esse TG foi bem interessante, pois abre a possibilidade de continuação no segmento de performance, software, relatórios, integração com a plataforma PDesigner, entre outros. Ou seja, é uma área de pesquisa bastante ampla e que tem perspectivas muito boas e uma quantidade de trabalho e estudo ainda maior.

8. Referências

[1] Closing the SoC Design Gap – online, acesso em 27/03/2010 na url:
http://www.inf.pucrs.br/~moraes/prototip/artigos/Closing%20the%20SoC%20Design%20Gap.pdf

[2] Assembly – Online, acesso em 27/03/2010 na url:
http://www.plugmasters.com.br/sys/materias/120/1/Assembly,-Primeiros-Passos

[3] C++ - Online, acesso em 27/03/2010 na url:
http://www.cplusplus.com/

[4] Java - Online, acesso em 27/03/2010 na url:
http://java.sun.com/

[5] Catapult C Synthesis, Full-Chip High-Level Synthesis – Online, acesso em 27/03/2010 na url:
http://www.mentor.com/products/esl/high_level_synthesis/catapult_synthesis/

[6] CoWare Platform Architect *SystemC Platform Capture and Analysis for Platform-driven ESL Design* - Online, acesso em 28/05/2010 na url:
http://www.europractice.stfc.ac.uk/vendors/coware_PlatformArchitect.pdf

[7] System Design Using SOPC Builder - online, acesso em 23/03/2010 na url:
http://www.altera.ru/Disks/Altera%20Documentation%20Library/literature/hb/qts/qts_qii51003.pdf

[8] Electronic System Level Models for Functional Verification of System-on-Chip - online, acesso em 23/03/2010 na url:
http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=04297576

[9] Design for Low-Power at the Electronic System Level - online, acesso em 23/03/2010 na url:
http://www2.itu.edu.tr/~orssi/dersler/SLD/ESL_Design_for_Low_Power_ChipVision.pdf

[10] Mundo Conectado - Online, acesso em 27/03/2010 na url:
http://mundoconectado.blogspot.com/2006/08/mas-afinal-o-que-convergncia-digital.html

[11] Dissertação de mestrado de Millena de Andrade Almeida Gomes (maag@cin.ufpe.br)

[12] PDesigner Framework – Online, acesso em 22/03/2010 na url: http://www.pdesigner.org

[13] Altera SOPC Builder – Online, acesso em 23/03/2010 na url: http://www.altera.com/literature/lit-sop.jsp

[14] Ambiente de desenvolvimento integrado Eclipse - Online, acesso em 22/03/2010 na url:
http://www.eclipse.org

[15] Altera Avalon Specification - Online, acesso em 23/03/2010 na url:
http://www.altera.com/literature/manual/mnl_avalon_spec.pdf

[16] Nios II Processor Reference Handbook - Online, acesso em 23/03/2010 na url:
http://www.altera.com/literature/hb/nios2/n2cpu_nii5v1.pdf

[17] Engineering Change Management with the Chip Planner - online, acesso em 23/03/2010 na url:
http://www.altera.com/literature/hb/qts/qts_qii52017.pdf

[18] Scripting with Tcl in the Quartus II software - online, acesso em 23/03/2010 na url:
http://www.altera.ru/Disks/Altera%20Documentation%20Library/literature/an/an195.pdf

[19] Create a simple Tcl script for Altera Quartus II - online, acesso em 23/03/2010 na url:
http://www.doulos.com/knowhow/fpga/Automating_Tool_Flows_with_Tcl/quartus.php

[20] Tcl Script to Automate Quartus II Compilation - online, acesso em 23/03/2010 na url:
http://fpgaforum.blogspot.com/2006/04/tcl-script-to-automate-quartus-ii.html

[21] Tcl Scripting - online, acesso em 23/03/2010 na url:
http://www.altera.com/literature/hb/qts/qts_qii52003.pdf

[22] Introduction to the Altera SOPC Builder Using VHDL Design - online, acesso em 23/03/2010 na url:
ftp://ftp.altera.com/up/pub/Tutorials/DE2/Computer_Organization/tut_sopc_introduction_vhdl.pdf

[23] Fundamentals of ESL Synthesis – online, acesso em 10/05/2010 na url
http://www.techonline.com/learning/course/217600425

[24] Electronic system-level design tools come up short - online, acesso em 10/05/2010 na url:
http://www.eetimes.com/news/latest/showArticle.jhtml?articleID=179103444

[25] A look inside electronic system level (ESL) design - online , acesso em 10/05/2010 na url:
http://www.embedded.com/columns/technicalinsights/18402916?_requestid=15391

26] Slow Adoption for ESL - online, acesso em 10/05/2010 na url:
http://chipdesignmag.com/sld/blog/2010/03/25/slow-adoption-for-esl/

[27] Soft-Core Processor Design - online, acesso em 20/05/2010 na url:
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.86

[28] Avalon Bus Specification, Reference Manual - online, acesso em 12/05/2010 na url:
http://stud4.tuwien.ac.at/~e0125015/nios/lit/avalon_bus_spec.pdf

[29] Barramento Avalon - online, acesso em 15/05/2010 na url:
http://webcache.googleusercontent.com/search?q=cache:FdwJbCiHUT0J:www.cin.ufpe.br/~maag/prototipacao_seminario_avalon.ppt+barramento+avalon&cd=1&hl=pt-BR&ct=clnk&gl=br

[30] SOPC Builder Memory Subsystem Development Walkthrough - online, acesso em 15/05/2010 na url: http://www.altera.com/literature/hb/qts/qts_qii54006.pdf

[31] Common Flash Interface Controller Core - online, acesso em 17/05/2010 na url:
http://www.altera.com/literature/hb/nios2/n2cpu_nii51013.pdf

[32] DDR and DDR2 SDRAM Controller Compiler User - online, acesso em 18/05/2010 na url:
http://www.altera.com/literature/ug/ug_ddr_sdram.pdf

[33] EPCS Device Controller Core - online, acesso em 18/05/2010 na url:
http://www.altera.com/literature/hb/nios2/n2cpu_nii51012.pdf

[34] JTAG UART Core - online, acesso em 20/05/2010 na url:
http://www.altera.com/literature/hb/nios2/n2cpu_nii51009.pdf

[35] Nios Timer - online, acesso em 20/05/2010 na url:
http://www.altera.com/literature/ds/ds_nios_timer.pdf

[36] Optrex 16207 LCD Controller Core - online, acesso em 20/05/2010 na url:
http://www.altera.com/literature/hb/nios2/n2cpu_nii51019.pdf

[37] PIO Core - online, acesso em 20/05/2010 na url:
http://www.altera.com/literature/hb/nios2/n2cpu_nii51007.pdf

[38] SOPC Builder, SRAM Controller forAltera's DE2/DE1 Boards - online, acesso em 20/05/2010 na url:
http://webcache.googleusercontent.com/search?q=cache:Wi8pAIM_qPsJ:ftp://ftp.altera.com/up/pub/University_Program_IP_Cores/SRAM_Controller.pdf+sram+altera&cd=3&hl=pt-BR&ct=clnk&gl=br

[39] Development and Education Board - online, acesso em 21/05/2010 na url:
http://users.ece.gatech.edu/~hamblen/DE2/DE2%20Reference%20Manual.pdf

[40] University Program IP Core Manual - online, acesso em 21/05/2010 na url:
ftp://ftp.altera.com/up/pub/University_Program_IP_Cores/VGA.pdf

[41] AUDIO CORE FOR ALTERA DE2/DE1 BOARDS - online, acesso em 21/05/2010 na url:
ftp://ftp.altera.com/up/pub/University_Program_IP_Cores/Audio.pdf

[42] SD Card Interface for SOPC Builder - online, acesso em 25/05/2010 na url:
ftp://ftp.altera.com/up/pub/Altera_Material/QII_9.0/University_Program_IP_Cores/Memory/SD_Card_Interface_for_SoPC_Builder.pdf

[43] SD/MMC SPI Core with Avalon Interface - online, acesso em 25/05/2010 na url:
www.altera.com/products/ip/ampp/elcamino/documents/sd-mmc-spi-core.pdf

[44] DE2 User Manual - online, acesso em 25/05/2010 na url:
ftp://ftp.altera.com/up/pub/Webdocs/DE2_UserManual.pdf

[45] A brief history of Eclipse - online, acesso em 25/05/2010 na url:
http://www.ibm.com/developerworks/rational/library/nov05/cernosek/index.html

[46] Microsoft Visual Studio - online, acesso em 25/05/2010 na url:
http://pt.wikipedia.org/wiki/Microsoft_Visual_Studio

[47] QNX - online, acesso em 28/05/2010 na url: www.qnx.com

[48] Introduction to NIOS II IDE - online, acesso em 28/05/2010 na url:
www.cse.unl.edu/~ylu/csce351/homework/lab2/lab2.pdf

[49] Google App Engine for Business - online, acesso em 29/05/2010 na url:
http://code.google.com/appengine

[50] Design Debugging Using the SignalTap II Embedded Logic Analyzer - online, acesso em 29/05/2010 na url: www.altera.com/literature/hb/qts/qts_qii53009.pdf

[51] Synopsys - online, acesso em 29/05/2010 na url: www.synopsys.com

[52] SOPC Builder PTF File, Reference Manual - online, acesso em 29/05/2010 na url: www.ittc.ku.edu/~dandrews/753/files/EPXA1Webpage/mnl_sopcptf.pdf

[53] RS-232 e Configuração da UART 16550A - online, acesso em 29/05/2010 na url: http://www.inf.pucrs.br/~eduardob/disciplinas/ProgPerif/sem04.2/Seminarios/JonisDemamann/RS%20232%20e%20UART%2016550.htm

[54] Soft-Core Processors for Embedded Systems - online, acesso em 30/05/2010 na url: http://www.reds.ch/share/cours/ReCo/documents/soft_core_processors.pdf

[55] Six Embedded-Processor Cores Challenge ARM, ARC, MIPS, and DSPs - online, acesso em 30/05/2010 na url: http://www.tensilica.com/uploads/pdf/MDR_DiamondPlus.pdf

[56] Sistemas Operacionais V, Gerência de Memória - online, acesso em 20/05/2010 na url: http://www.ppgia.pucpr.br/~maziero/lib/exe/fetch.php/so:so-cap05.pdf

[57] Nios II Processor Reference - online, acesso em 20/05/2010 na url: http://www.ece.mtu.edu/faculty/rmkieckh/cla/3173/Nios2-Proc-Ref-Handbk.pdf

[58] A memória viva (RAM ou memória PC) - online, acesso em 20/05/2010 na url: http://pt.kioskea.net/contents/pc/ram.php3

[59] Interfacing DDR & DDR2 SDRAM with Cyclone II Devices - online, acesso em 20/05/2010 na url: http://www.altera.com/literature/an/an361.pdf

[60] SDRAM Controller with Avalon Interface - online, acesso em 20/05/2010 na url: http://www.altera.com/literature/ds/ds_sdram_ctrl.pdf

[61] Using the SDRAM Memory on Altera's DE2 Board - online, acesso em 20/05/2010 na url: http://instruct1.cit.cornell.edu/courses/ece576/DE2/tut_DE2_sdram_verilog.pdf

[62] A Brief Introduction to the JTAG Boundary Scan Interface - online, acesso em 20/05/2010 na url: http://www.inaccessnetworks.com/projects/ianjtag/jtag-intro/jtag-intro.html

[64] Transaction Level Modeling in System Level Design - online, acesso em 01/06/2010 na url: http://*citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.* 12.38