****

**UNIVERSIDADE FEDERAL DE PERNAMBUCO**

**CENTRO DE INFORMÁTICA**

**ENGENHARIA DA COMPUTAÇÃO**



**CO-EMULAÇÃO HARDWARE/SOFTWARE DE SISTEMAS NO PDESIGNER**

**Trabalho de Graduação**

**Aluno:** Ciro Luiz Araujo Ceissler (clac@cin.ufpe.br)

**Orientador:** Cristiano Coelho de Araújo *(cca2@cin.ufpe.br*)

Recife, Dezembro de 2009

**UNIVERSIDADE FEDERAL DE PERNAMBUCO**

**CENTRO DE INFORMÁTICA**

**ENGENHARIA DA COMPUTAÇÃO**

**CO-EMULAÇÃO HARDWARE/SOFTWARE DE SISTEMAS NO PDESIGNER**

**Trabalho de Graduação**

Monografia apresentada à Universidade Federal de Pernambuco, como trabalho de conclusão do curso de Graduação em Engenharia da Computação, para a obtenção do título de Engenheiro da Computação.

**Aluno:** Ciro Luiz Araujo Ceissler (clac@cin.ufpe.br)

**Orientador:** Cristiano Coelho de Araújo *(cca2@cin.ufpe.br*)

Recife, Dezembro de 2009

**UNIVERSIDADE FEDERAL DE PERNAMBUCO**

**CENTRO DE INFORMÁTICA**

**ENGENHARIA DA COMPUTAÇÃO**

**CO-EMULAÇÃO HARDWARE/SOFTWARE DE SISTEMAS NO PDESIGNER**

**Trabalho de Graduação**

Monografia apresentada à Universidade Federal de Pernambuco, como trabalho de conclusão do curso de Graduação em Engenharia da Computação, para a obtenção do título de Engenheiro da Computação.

Aprovada em \_\_\_\_ de \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ de \_\_\_\_\_\_\_.

Prof. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Orientador)

Recife, Dezembro de 2009

“Escolha um trabalho que você ame e não terá de trabalhar um único dia na sua vida.”

Confúcio

**AGRADECIMENTOS**

Família, claro! Meu pai, que me influenciou a seguir a carreira de engenheiro, mas não só isso, ele ensinou muitas coisas importantes ao longo da vida e que ajudaram na minha formação. Minha mãe sempre carinhosa e atenciosa. Minha irmã mais velha, Cilla, arquiteta maluca! Minha irmã mais nova, Cindy, a multiplicadora. Os primos: Tharcio, Taís, Júnior, Dell, Elissa, Helaysa, Thainá, Lukas, Guilherme e os desconhecidos. Tios desmantelados, tias, avôs...

A turma de engenharia vida boa! Adelmo (fulião), Silvio Santos, Styve, Igino, Lúcio, Diogo, Rafael, TAVL, Biu, J.P., Libório, Bruno BPN, Theogenes, Cirdes, entre outros. Apesar de serem estressados, foram companheiros em provas, projetos e cachaça. Relaxem! Aos cientistas frequentadores ou não da Kelly: Galego, Vanessinho, Flávio Almeida, Macaxeira, Vanessão, Lucas Araújo, Arara, Leo Vilaça...

Aos membros e ex-membros do PDesigner: Williams, Millena, João Victor, Fernando, Danilo, Josiane, Denys, Luís Pedro, entre outros. Principalmente Williams e Millena, que me ajudaram muito nas minhas atividades de IC, tendo paciência para ensinar, compartilhar idéias e tornar o ambiente das *baias* mais divertido.

O pessoal da monitoria de Infra-de-Software, entre eles Alan o monitor chefe. Em Teresina, aos amigos: Tonton, André, Laécio, Kayro, Marcelo, Leo, Gustavo Presuntim, Rodrigo, Mayra, Tássia, Djamila, Andréia, entre outros.

Cristiano pela paciência como meu orientandor de IC e TG, ajudando nesta etapa da minha vida!

Cindy, novamente, e Copinho por tirarem esta monografia do mundo virtual.

Minha namorada Amanda.

**RESUMO**

Atualmente existe uma crescente demanda pelo desenvolvimento de sistemas embarcados mais complexos e com *time-to-market* reduzido. Estes sistemas se caracterizam por possuírem partes em hardware e partes em software, desta maneira impondo o desenvolvimento o mais cedo possível do software embarcado, mesmo sem que o hardware seja finalizado. Existe uma demanda para a criação de ferramentas e técnicas que auxiliem o desenvolvimento de sistemas, antecipando a validação do seu sistema antes dele ser colocado em produção. Uma solução é aumentar o nível de abstração na modelagem de um sistema através de metodologias de desenvolvimento de sistemas eletrônicos *Electronic System Level (*ESL*)*. Entretanto, um novo problema surge que é o *gap* entre sistemas modelados em ESL e o hardware alvo, tornando complexa a tarefa de síntese, que tem o objetivo de diminuir o esforço humano. O objetivo deste trabalho de graduação é criar e integrar ao PDesigner, que é uma ferramenta para especificação de sistemas embarcados, uma técnica de co-emulação através da interface ethernet que permita a comunicação entre módulos simulados em SystemC e uma plataforma base numa FPGA, diminuindo o *gap* entre níveis diferentes de sistema. Apesar da interface ethernet não ser a alternativa mais rápida, a interface PCI é uma solução com melhor desempenho, ela é mais flexível e oferece suporte a placa de desenvolvimento DE2-70.

**Palavras-chave**: ESL, co-emulação, SystemC, FPGA, MPSoC, plataforma virtuais.

**SUMÁRIO**

[Lista de Figuras 8](#_Toc248192356)

[Lista de Tabelas 9](#_Toc248192357)

[Lista de Abreviaturas 10](#_Toc248192358)

[1. Introdução 11](#_Toc248192359)

[1.1 Motivação 11](#_Toc248192360)

[1.2 Contexto 11](#_Toc248192361)

[1.3 Objetivo 14](#_Toc248192362)

[1.4 Estrutura do Trabalho 15](#_Toc248192363)

[2. Modelagem de Sistemas 16](#_Toc248192364)

[2.1 Verificação 16](#_Toc248192365)

[2.2 Platform Designer 17](#_Toc248192366)

[2.3 SOPC Builder 19](#_Toc248192367)

[2.4 Altera DE2-70 20](#_Toc248192368)

[2.5 Conclusão 23](#_Toc248192375)

[3. Co-Emulação 24](#_Toc248192376)

[3.1 Visão Geral 24](#_Toc248192379)

[3.2 Trabalhos Relacionados 25](#_Toc248192380)

[3.3 Conclusão 29](#_Toc248192381)

[4. Estratégia de Co-Emulação no PDesigner 30](#_Toc248192382)

[4.1 Visão Geral 30](#_Toc248192385)

[4.2 Arquitetura 31](#_Toc248192386)

[4.3 Protocolo de Comunicação 36](#_Toc248192387)

[4.4 Integração no PDesigner 37](#_Toc248192388)

[5. Estudo de Caso 40](#_Toc248192389)

[5.1 Visão Geral 40](#_Toc248192393)

[5.2 Arquitetura 41](#_Toc248192394)

[5.3 Protocolos 42](#_Toc248192395)

[5.4 Resultados 43](#_Toc248192396)

[6. Conclusões e Trabalhos Futuros 46](#_Toc248192397)

[7. Referências Bibliográficas 47](#_Toc248192398)

# Lista de Figuras

[Figura 1: Capacidade do CI versus Produtividade. 12](#_Toc248147945)

[Figura 2: Fluxo de Genérico de Desenvolvimento de SOC’s. 13](#_Toc248147946)

[Figura 3: Fluxo de Desenvolvimento de FPGA’s. 14](#_Toc248147947)

[Figura 4: Verificação de um Módulo através de um Sistema como Testbench. 17](#_Toc248147948)

[Figura 5: Interface gráfica do PDesigner baseada na IDE Eclipse. 18](#_Toc248147949)

[Figura 6: Arquitetura de uma FPGA. 19](#_Toc248147950)

[Figura 7: Exemplo de um Sistema Criado pelo SOPC Builder. 20](#_Toc248147951)

[Figura 8: Altera DE2-70 Digital Camera Development Platform. 21](#_Toc248147952)

[Figura 9: Protocolo IEEE 802. 22](#_Toc248147953)

[Figura 10: Protocolo Ethernet. 23](#_Toc248147954)

[Figura 11: Comparativo entre as Técnicas de Execução do Sistema. 25](#_Toc248147955)

[Figura 12: Esquema do Sistema de Co-verificação. 27](#_Toc248147956)

[Figura 13: Diagrama de Co-emulação. 27](#_Toc248147957)

[Figura 14: Diagrama de um Sistema de Co-emulação HW/SW. 28](#_Toc248147958)

[Figura 15: Arquitetura Co-emulação. 31](#_Toc248147959)

[Figura 16: Esquema do Controlador Ethernet DM9000A. 32](#_Toc248147960)

[Figura 17: Arquitetura Tradutor de Protocolos. 33](#_Toc248147961)

[Figura 18: Comunicação pin-accurate com o Tradutor de Protocolos. 34](#_Toc248147962)

[Figura 19: Operação de Escrita utilizando o Protocolo Avalon. 34](#_Toc248147963)

[Figura 20: Operação de Leitura utilizando o Protocolo Avalon. 35](#_Toc248147964)

[Figura 21: Identificação da Pasta da Altera no PDesigner. 38](#_Toc248147965)

[Figura 22: Conexão entre uma Plataforma no PDesigner com uma Plataforma FPGA. 39](#_Toc248147966)

[Figura 23: Fórmula da Distância de Mahalanobis. 40](#_Toc248147967)

[Figura 24: Imagem Real vs Imagem Segmentada. 41](#_Toc248147968)

[Figura 25: Plataforma em SystemC. 42](#_Toc248147969)

[Figura 26: Plataforma no Sistema Co-emulado. 42](#_Toc248147970)

[Figura 27: Análise de Pacotes através do Wireshark. 44](#_Toc248147971)

# Lista de Tabelas

[Tabela 1: Resultados. 45](#_Toc248148069)

# Lista de Abreviaturas

IP – Intellectual Property

LAN – Local Area Network

CRC – Code Cyclic

FIFO – First-In-First-Out

FFT – Fast Fourier Transform

SCE-MI – Standard Co-emulation Modeling Interface

SOF – SRAM Object File

MAN – Metropolitan Area Network

MAC – Medium Access Control

RTL – Register Transfer Level

ESL – Electronic System Level

ISS – Instruction Set Simulator

FPGA – Field Platform Gates Array

SOPC – System on a Programmable Chip

IEEE – Institute of Electrical and Electronics Engineers

MPSoC – Multi-processor System-on-Chip

TLM – Transaction Level Modeling

# Introdução

## Motivação

Sistemas embarcados são cada vez mais úteis no nosso dia-a-dia, às vezes não percebemos mas a maioria dos produtos à nossa volta contém algum módulo embarcado, como: telefones celulares, sistemas de controle nos carros, televisão, câmera fotográficas digitais, entre outros.

## Contexto

A complexidade dos sistemas embarcados vem aumentado cada dia mais, isto ocorre pelo aumento da capacidade dos circuitos integrados, seguindo as previsões da lei de Moore [11], a cada 18 meses o número de transistores em um circuito irá dobrar. Apesar deste aumento, outro problema aparece relativo ao *gap* de produtividade, ou seja, apesar da capacidade estar aumentando não conseguimos produzir o suficiente para supera-lá. Na figura 1, está ilustrado o crescimento da capacidade em comparação com a produtividade.

Figura 1: Capacidade do CI versus Produtividade.

O aumento da mão-de-obra neste setor não é a solução para este problema. O aumento da equipe pode torna o projeto mais lento e complexo, devido ao problema de gerenciamento e comunicação da equipe [21].

A criação de ferramentas que aumentem a produtividade, automatizando processos e tornando mais fácil a análise do sistema, facilitando a exploração de arquitetura do sistema para encontrar o ponto ideal em relação ao custo e desempenho no projeto. Uma tendência para aumentar a produtividade é a elevação do nível de abstração através da adoção de uma metodologia conhecida como ESL[14]. Esta metodologia é muito associada a SystemC[22] e TLM[22].

SystemC é uma linguagem de descrição de hardware e software baseado em C++, orientada a hardware através de uma biblioteca própria implementada no padrão de C++. SystemC provê modelagem de plataformas e componentes em no nível de sistema de forma rápida, sendo muito utilizado para descrição de barramentos e dispositivos, além de aplicações específicas para plataformas, aumentando performance.

TLM é uma abstração da comunicação que é baseada em transações, encapsulando detalhes da troca de comunicação. O foco dela é na funcionalidade da comunicação, para onde os dados serão transferidos e de qual localização, ao invés da implementação em baixo nível. A sua associação com SystemC permite separar detalhes da comunicação entre módulos e sua implementação. Na figura 2, o fluxo genérico de desenvolvimento SOC’s, SystemC e TLM.

.

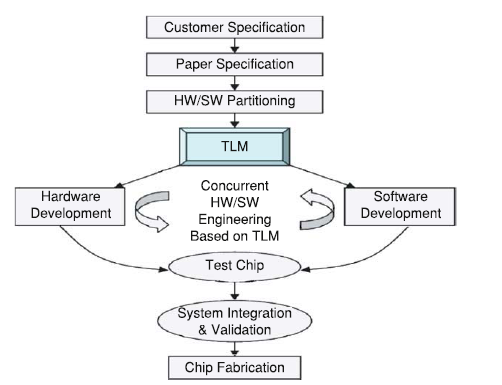


Figura 2: Fluxo de Genérico de Desenvolvimento de SOC’s.

No fluxo acima após a validação e integração do sistema, a próxima etapa é a fabricação do chip. Na figura 3, pode-se observar este fluxo para desenvolvimento de uma FPGA. FPGA são chips utilizados em várias aplicações que incluem: processamento digital de sinais, prototipação de ASIC, visão computacional, criptografia, sistemas de defesa do exército, entre outros.

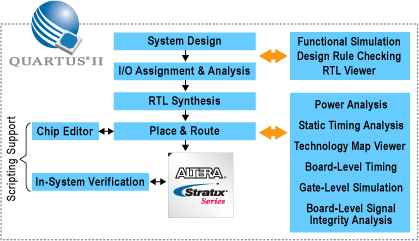


Figura 3: Fluxo de Desenvolvimento de FPGA’s.

A co-emulação é a junção dos dois fluxos de desenvolvimento, permitindo a partição do projeto. Isso permite que módulos em alto nível de abstração não sintetizáveis ou módulos de verificação, que não serão sintetizados, executem em paralelo com plataformas bases na FPGA, que já foram validadas anteriormente. Conseqüentemente, a validação do sistema será mais rápida, pois uma parte do sistema já se encontra executando no hardware alvo.

## Objetivo

Este trabalho integrará ao *framework* PDesigner [5] uma técnica de co-emulação, que permita ao usuário comunicar plataformas virtuais com plataformas reais implementadas em FPGA [8] sem se preocupar como a comunicação será realizada, evitando erros durante o desenvolvimento do projeto.

## Estrutura do Trabalho

Este documento está divido em seis capítulos. O capítulo 2 descreverá duas ferramentas de modelagem de sistemas. O capítulo 3 falará sobre a estratégia de co-emulação e alguns trabalhos relacionados nesta área. O capítulo 4 abordará uma forma de realizar a co-emulação e como integrar a mesma numa plataforma virtual, chamada PDesigner. O capítulo 5 irá mostrar o estudo de caso desenvolvido e alguns resultados comparativos entre duas abordagens.

# Modelagem de Sistemas

Neste capítulo, primeiro será mostrado o conceito de verificação na modelagem de sistemas. Depois, será abordado duas ferramentas de modelagem de sistemas que possuem características complementares, uma voltada para o desenvolvimento de sistemas de alto nível e outra para sistemas de baixo nível com foco em FPGA. Depois é descrito as características de uma FPGA utilizada para modelagem. No final, será feito um comparativo entre as duas ferramentas.

## Verificação

A verificação funcional de um projeto SOC é uma fase que garante  
a execução do projeto de acordo com suas especificações. É uma etapa complexa e demorada, realizada através da conversão das especificações em uma combinação de estímulos e resultados esperados, verificando os resultados esperados quando os estímulos são aplicados.

Na figura 4, está um exemplo de *testbench*, que é um sistema utilizando vários módulos em TLM, para validar o funcionamento do módulo em RTL, com o nome de DUT.

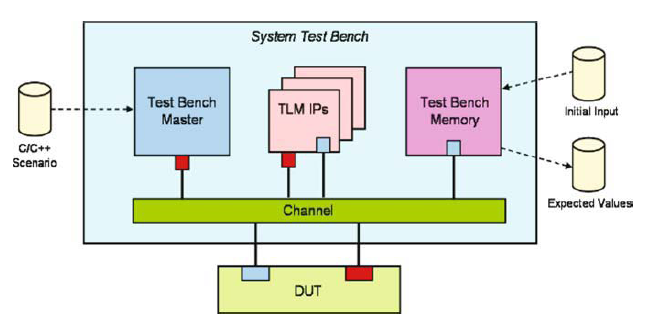


Figura 4: Verificação de um Módulo através de um Sistema como Testbench.

## Platform Designer

O PDesigner é uma ferramenta na área de eletrônica de sistema, ESL, que permite a criação e simulação de plataformas virtuais multiprocessadas, MPSoC’s, através de uma interface gráfica que permite a manipulação dos componentes. O seu desenvolvimento foi realizado pelo Grupo de Engenharia da Computação (Greco) do Centro de Informática da Universidade Federal de Pernambuco (Cin-UFPE).

As linguagens suportadas por esta ferramenta são SystemC[22], Verilog[4] e ArchC[23]. O SystemC é uma linguagem para desenvolvedores de SoCs utilizando o padrão de C/C++, sua utilização serve como base para modelarmos os sistemas MPSoCs. O Verilog é uma linguagem de descrição de hardware, utilizada para descrição em nível RTL ou porta lógica, ou seja, permite a criação de módulos com o nível de abstração mais baixo e executá-lo de forma abstrata através do simulador Icarus Verilog[24] utilizando uma técnica de cossimulação. Os processadores utilizados na ferramenta são modelados em ArchC, uma linguagem de descrição de processadores baseada em SystemC, e um conjunto de ferramentas que suportam a descrição de processadores em diferentes níveis de abstração, permitindo a o desenvolvimento paralelo de hardware e software.

O framework é baseado também no formato de distribuição de IP-cores do Spirit Consortium[24], um padrão IEEE utilizado para facilitar a manipulação automática de IP’s em plataformas virtuais.

O seu desenvolvimento foi realizado criando plugins para o ambiente de desenvolvimento integrado Eclipse[25] e permite que a plataforma forma seja montada através da interface gráfica mostrada na figura 5. O PDesigner possui vários componentes já integrados na biblioteca tais como modelos de processadores, memórias e barramentos, além de permiti que o usuário inclua seu componente através do IPZip.

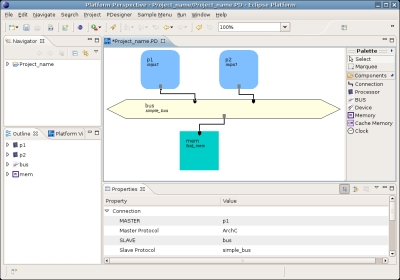


Figura 5: Interface gráfica do PDesigner baseada na IDE Eclipse.

## SOPC Builder

O SOPC Builder[6] é uma ferramenta, criada pela Altera, para o desenvolvimento e geração de sistemas de uma maneira muita mais rápida que o métodos manuais de integração. O objetivo deste programa é montar uma plataforma em Verilog ou VHDL e sintetizar para uma FPGA.

FPGA é um hardware reconfigurável cuja funcionalidade pode ser modificada de acordo com a necessidade, ou seja, é um chip que suporta a implementação de circuitos lógicos, consistindo num grande arranjo de blocos lógicos configuráveis, blocos de entrada e saída, e canais de roteamento. Na figura 6, pode- se ver a arquitetura de uma FPGA.

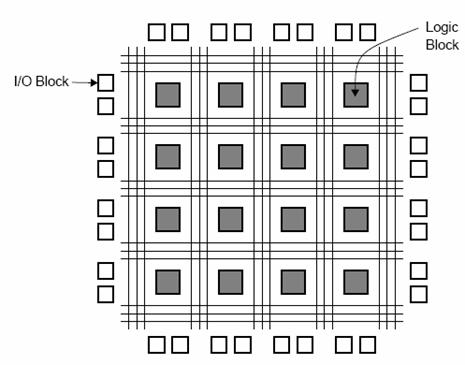


Figura 6: Arquitetura de uma FPGA.

O SOPC Builder possui uma biblioteca de componentes que permite a construção de plataformas na FPGA. Os componentes são o microprocessador Nios II, controladores de memória, interfaces e periféricos. Uma plataforma pode conter algum microprocessador, que permite a criação de um software para interagir com este sistema. Todos os componentes incluídos nessa plataforma serão, por padrão, conectados ao barramento Avalon da Altera[26]. Na figura 7, pode-se ver um exemplo de uma plataforma.

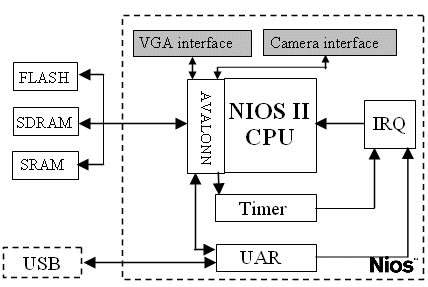


Figura 7: Exemplo de um Sistema Criado pelo SOPC Builder.

## 2.4 Altera DE2-70

A placa de desenvolvimento utilizada, neste trabalho, foi a Altera DE2-70[3], que vem equipada com aproximadamente 70.000 elementos lógicos na FPGA Cyclone II EP2C70. Além disso, esta placa contém interface de entrada e saída como 10/100 ethernet, RS-232, porta infravermelha, entre outros. Ela possui uma memória de 64 MB de SDRAM, 2 MB de SSRAM e 8 MB de Flash, além do suporte a cartões de memória SD. As freqüências geradas pela placa ou ciclos de relógio são 50 Mhz e 27 Mhz, além de ter uma interface para a utilização de *clock* externo.

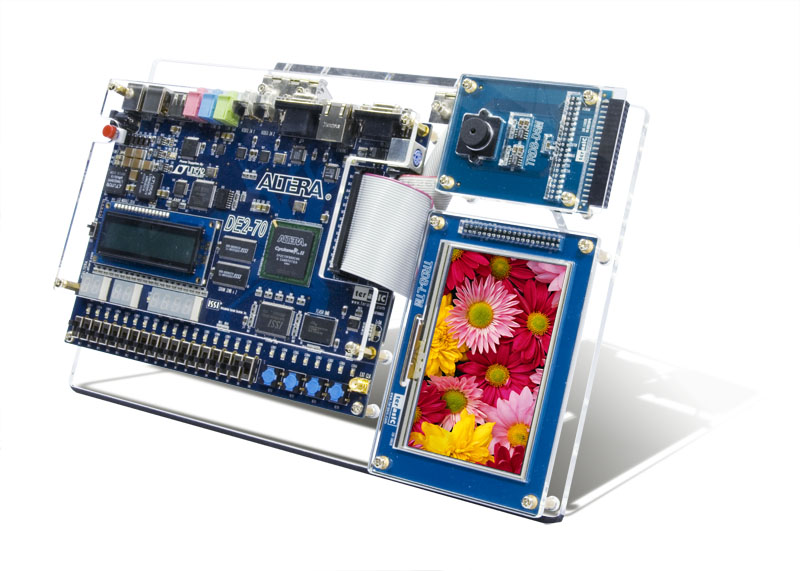


Figura 8: Altera DE2-70 Digital Camera Development Platform.

Este kit de desenvolvimento possui alguns periféricos como o controlador DM9000A, que permite a comunicação utilizando o protocolo ethernet.

**2.4.1 Protocolo Ethernet**

## O padrão IEEE 802 é uma norma que tem como objetivo padronizar LAN que são redes locais e MAN que são redes metropolitanas nas camadas física e enlace do modelo OSI para padrões de redes.

As características físicas e eletrônicas dos dispositivos são descritas na camada física. Ela é responsável por permitir a transmissão da informação através algum meio, podendo este ser cabeado ou sem-fio. A sua principal função é estabelecer a conexão e terminá-la, ele também deve transformar o sinal digitalizado recebido na camada acima, para o formato definido no canal de comunicação, por exemplo, o sinal digital é modulado antes de ser transmitido através do ar.

A camada de enlace é responsável pela transmissão e recepção de quadros, podendo detectar erros provenientes da camada física e, opcionalmente, corrigir esses erros. Além disso, ela realiza o controle de fluxo. No padrão IEEE 802 esta camada é divida em duas: controle de ligação lógica e controle de acesso ao meio físico, conhecido como MAC. A primeira fornece uma interface para a camada de rede e a outra acessa diretamente o meio físico e controla a transmissão. Na figura 9, pode-se observar que a área sombreada corresponde ao protocolo especificado, e sua localização tendo o modelo OSI como referência, localizado na esquerda.

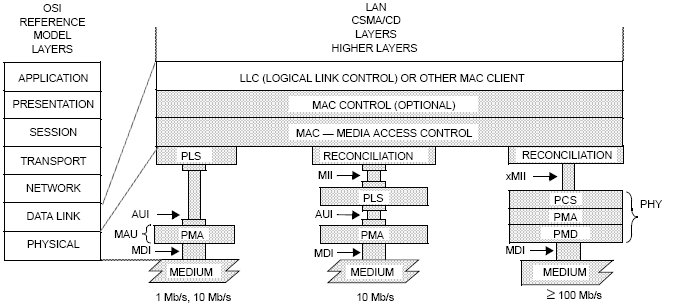


Figura 9: Protocolo IEEE 802.

Os primeiros oito bytes deste protocolo corresponde ao preâmbulo, que serve para sincronizar os receptores. O cabeçalho corresponde, ao endereço MAC destino e fonte, cada um utilizando seis bytes, além de dois bytes para identificação do tipo do frame. Após o cabeçalho, estará o dado do frame, cujo tamanho varia entre 46 e 1500 bytes. E por último, temos quatro bytes para identificação de erro na transmissão, conhecido como código de redundância cíclica ou CRC. Na figura 10, mostra-se como está divido o pacote do protocolo ethernet.Quadro ethernet: preambulo, end. destino,
end. fonte, tipo, dados, CRC

Figura 10: Protocolo Ethernet.



## Conclusão

As ferramentas descritas acima possuem focos diferentes. A primeira, o PDesigner, permitir a criação de sistemas com um alto nível de abstração para permitir uma validação mais rápida, permitindo assim um exploração de arquitetura. Já a outra ferramenta, o SOPC Builder, tem o foco voltado para sistemas voltados para FPGA, ou seja, está voltado para implementação em baixo nível de abstração, esta fase é complexa e possui um alto custo computacional, mas é a fase final do desenvolvimento se o hardware alvo for uma FPGA.

Um problema observado foi a falta de integração de módulos de hardware não sintetizáveis ou módulos de verificação que não serão sintetizados com a plataforma na FPGA. A possibilidade de integração de módulos de hardware não sintetizáveis permite uma exploração de arquitetura mais eficiente, além da possibilidade de identificação de problemas nas primeiras etapas do desenvolvimento. A Outra possibilidade é a realização de verificação do sistema, através de estruturas de testes em alto nível. Isso permiti a criação de testes com a capacidade de cobertura muito maior e mais simples de serem desenvolvidos, além de poder utilizar aplicativos gráficos para facilitar a identificação de erros.

# Co-Emulação

Neste capítulo, será explicado o que é co-emulação e depois serão mostrados alguns trabalhos relacionados, detalhando a abordagem utilizada.



## Visão Geral

A definição de emulação é a capacidade reproduzir o comportamento de algum dispositivo desenvolvido para uma plataforma específica em outra plataforma, por exemplo, o jogo Donkey-Kong feito para rodar no Commodore 64, pode ser executado no Microsoft Windows através de um emulador deste vídeo game[2].

Já a co-emulação é a comunicação de forma abstrata entre processos de software e dispositivos de hardware, ou seja, duas emulações em paralelo que estão trocando informações. Os processos de software interpretarão as informações obtidas do hardware como se fossem enviada por um processo de software, e vice-versa.

A co-emulação é utilizada como uma forma de diminuir a complexidade das estruturas para realizar teste e verificação, isso é necessário devido a dificuldade de realizar testes de sistemas de hardware, necessitando de uma interface entre o hardware e o software que facilite a criação de mecanismos para validar todo o sistema.

Outra possibilidade é fazer testes mais rápidos. A complexidade de alguns sistemas faz com que as simulações sejam muito demorada, por exemplo uma simulação que demora uma hora em uma FPGA, poderá dura mais de um ano em um simulador ISS e até um milênio num simulador de portas lógicas, como mostrado na figura 11. A integração de módulos em hardware com simulações permite conseguir tempos próximos do real durante a execução do sistema final.

Figura 11: Comparativo entre as Técnicas de Execução do Sistema.

Além disso, ao permite identificar erros de projeto mais cedo. Por exemplo, a interface entre um módulo que está sendo simulado e um barramento Avalon na FPGA. A co-emulação facilita a identificação de erro do módulo pois o barramento já foi validado antes e todas as suas características estão sendo levadas em conta na interface com o módulo.

## Trabalhos Relacionados

O primeiro trabalho é um sistema de co-verificação portável que gera automaticamente *testbench*[18], que foi desenvolvido na Universidade Tokai no Japão. Ele permite a criação do *testbench* em C++ e uma geração automática das estruturas para realizar a interface com o módulo descrito em uma linguagem HDL.

Co-verificação é uma técnica de verificação que utiliza a vantagem do software e hardware. Um modelo em alto nível, cuja carga de processamento é alta, pode ser implementando com a mesma lógica em hardware para realizar uma verificação mais rápida. Outra possibilidade é quando o hardware está incompleto, na qual pode-se substituir a parte faltosa por um modelo em alto nível equivalente aquela parte.

As técnicas de co-verificação utilizadas neste trabalho são a co-simulação e a co-emulação. A primeira permite utilizar testbench e módulos de software que simulam algum hardware em linguagens HDL. A segunda técnica permite a comunicação entre um testbench e uma FPGA que contém a implementação do hardware.

Na co-simulação, o hardware será convertido através do Verilator[28] para C++ e o compilador gerará um executável unindo o hardware e o *testbench*. Já na co-emulação, será gerado um binário para o arquivo de *testbench* que irá se comunicar com o hardware mapeado numa FPGA. Isso faz com que tenha-se uma economia no esforço de verificação, porque a interface com o código em C++ será gerada automaticamente. Na figura 12, pode-se ver um esquema do sistema de co-verificação.

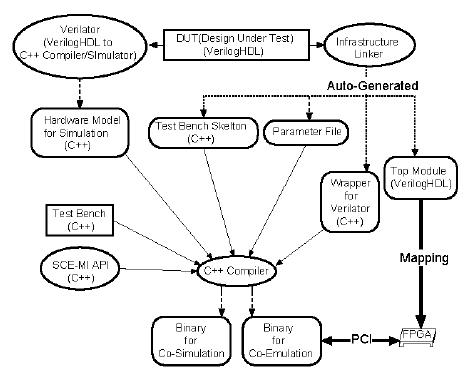


Figura 12: Esquema do Sistema de Co-verificação.

A co-emulação é feita utilizando um interface PCI, que é comum e permite altas velocidades de comunicação. O dispositivo utilizado é uma FPGA com interface PCI. O padrão de comunicação utilizado foi o SCE-MI que é uma sugestão de padronização para interfaces co-emuladas da Accellera[27].

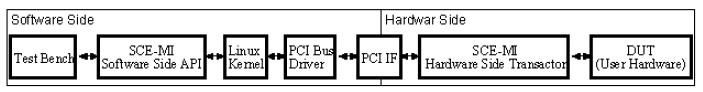


Figura 13: Diagrama de Co-emulação.

Nos testes realizados, observa-se que a técnica de co-emulação permite uma verificação mais rápida, em torno de 50 vezes mais rápida que a co-simulação. Porém, o sistema encontra-se limitado pela freqüência de operação do barramento PCI que é 33MHz.

O segundo trabalho relacionado propõe um sistema de co-emulação de HW/SW de SOC para testes e verificação baseado em fluxo de dados para diminuir a sobrecarga na comunicação[19], que foi desenvolvido pela Universidade de Ciência e Tecnologia Eletrônica da China.

Essa técnica foi proposta para substituir os modos usuais de implementação da co-emulação que é baseado em interrupção, ou seja, esperava o dado chegar para receber o próximo. A arquitetura deste sistema é baseada em um simulador em software e uma FPGA com interface PCI emulando o hardware, como se pode ver na figura 14.

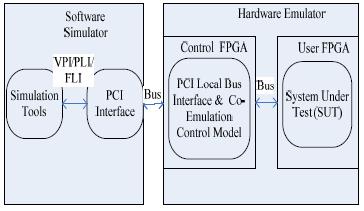


Figura 14: Diagrama de um Sistema de Co-emulação HW/SW.

A sobrecarga da comunicação é reduzida utilizando duas FIFOs no módulo em hardware, uma para transmitir os dados do para o lado do software e outra para armazenar a informação recebida.

Um dos estudos de casos utilizados foi o cálculo da FFT. Os resultados da técnica baseada em fluxos de dados em comparação com o modo de interrupção são em torno de cinco vezes mais rápidos, permitindo que o hardware co-emulado chegue próximo da velocidade de operação real.

## Conclusão

A co-emulação é uma técnica muita utilizada para realizar verificação de sistemas de uma forma mais rápida e simples. As técnicas propostas utilizam barramento PCI que é uma solução para realizar interface direta com o PC, sendo rápido. Entretanto, o barramento PCI não é flexível, pois precisa que cada máquina tenha instalado uma placa FPGA.

Os kits de desenvolvimentos nem sempre possuem interface PCI, sendo necessário uma interface alternativa. Alguns kits com interface PCI, como o PCI-X *Development Board*[29], são voltados para projetos que no final utilizarão esta interface para se comunicar, e não como um solução para realizar co-emulação. A utilização de outra interface, como a ethernet pode permite o uso muito mais rápido e simples por várias máquinas.

# Estratégia de Co-Emulação no PDesigner

Neste capítulo, será detalhado como a estratégia de co-emulação foi realizada no PDesigner, detalhando a arquitetura definida, o protocolo de comunicação escolhido e as modificações realizadas na ferramenta para executar esta técnica.



## Visão Geral

A necessidade de diminuir o gap entre sistemas modelados em ESL e as plataformas alvos numa plataforma virtual, fez necessária a criação de um técnica que permiti-se a comunicação entre eles. A solução proposta foi a co-emulação.

A co-emulação foi implementada utilizando o conceito de transações entre os módulos em nível de abstração diferente, comunicando-se através de controladores ethernet. No ambiente do simulador, como ilustrado na figura X, pode-se observar uma plataforma modelada no PDesigner utilizando componentes como processadores, memória, barramento e outros dispositivos, que possuem variados níveis de abstrações sendo simulados em linguagens como SystemC e ArchC. O simulador irá enviar e receber informações através de uma interface ethernet, que estará conectada ao outro ambiente relativo ao dispositivo hardware.

O dispositivo de hardware é uma FPGA, que irá se comunicar com o simulador através do controlador ethernet DM9000A contido da placa de desenvolvimento DE2-70, onde terá um módulo tradutor responsável por montar o pacote de envio e decodificar o pacote de retorno.

## Arquitetura

A arquitetura proposta para implementação do sistema pode ser vista na figura 15, esta contém uma plataforma virtual criada no PDesigner utilizando qualquer componente na sua biblioteca, PDLibrary, e instanciando com a FPGA através de um módulo com o nível de abstração FPGA. A outra parte da arquitetura são os controladores ethernet responsáveis por fazer a comunicação entre os dois módulos através de um cabo de par trançado utilizando conectores 8P8C. Já no outro ambiente, tem uma plataforma montada para FPGA instanciando um tradutor, responsável por realizar a comunicação entre controlador ethernet DM9000A e a plataforma FPGA.

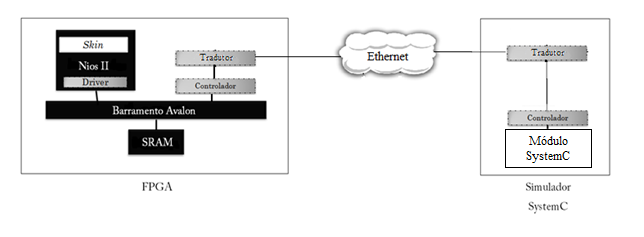


Figura 15: Arquitetura Co-emulação.

**4.2.1 Ambiente Simulado**

A plataforma virtual contém um controlador de transações, responsável por tratar as requisições recebidas de acordo com o protocolo definido. Além disso, a interface com o controlador ethernet será feita através do sistema operacional, no qual a plataforma está executando, o Linux. Este sistema possui uma API que permitir tratar pacotes de forma simples, além de possuir *device driver* para o controlador ethernet, facilitando boa parte da dificuldade de comunicação com o mesmo.

**4.2.2 Controlador Ethernet DM9000A**

O controlador ethernet é responsável por implementar o protocolo IEEE 802.3, que é um padrão para definir a comunicação na camada física.

Na placa de desenvolvimento, o controlador ethernet é o DM9000A é um controlador que implementa o protocolo Fast Ethernet, que permite chegar a velocidades de até 100Mb/s, com uma interface para processadores de uso geral, esta interface pode ser de oito ou 16 bits, e uma memória SRAM de 4KB. Na figura 16, pode-se ver a arquitetura do controlador.

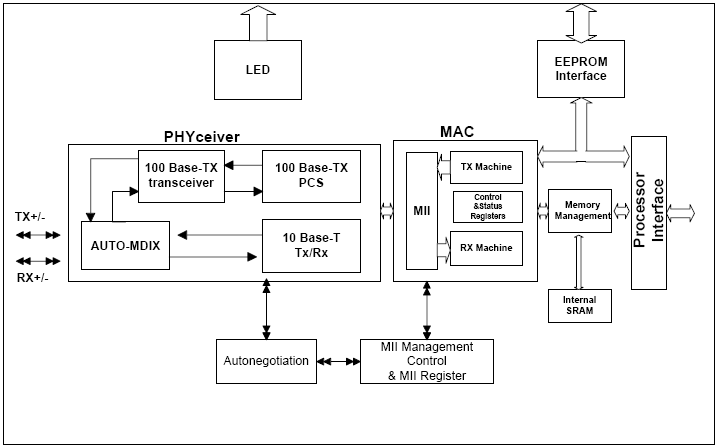


Figura 16: Esquema do Controlador Ethernet DM9000A.

**4.2.3 Tradutor de Protocolos**

O tradutor de protocolos é responsável por transforma pacotes ethernet em instruções de leitura e escrita no barramento Avalon, e vice-versa. Ele trabalha com duas freqüências de operação, a primeira é fixada em 25Mhz, devido a limitação do controlador ethernet DM9000A. A segunda freqüência de operação é variável e depende do barramento Avalon.

Apesar da primeira freqüência ser um limitante para emulação de um módulo, pois o mesmo ficaria com a freqüência máxima restrita que poderia não ser o comportamento desejado, este problema é minimizado através da utilização de *buffers* de recepção e envio. Então, teremos a comunicação com o ethernet limitada a 25Mhz, mas dependendo do sistema a freqüência de comunicação com o Avalon pode ser emulada próximo da freqüência real. Na figura 17, pode-se ver a arquitetura do tradutor de protocolos.

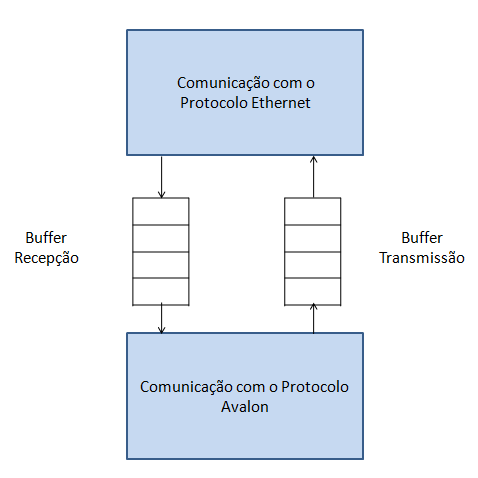


Figura 17: Arquitetura Tradutor de Protocolos.

O protocolo de comunicação com Avalon é *pin-accurate*, realizado através de seis pinos: escrita, dado de escrita, leitura, dado de leitura, selecionador de chip e endereço. Já no protocolo ethernet, a comunicação também é *pin-accurate*, realizada através de sete pinos: selecionador de chip, leitura, escrita, *reset*, interrupção, comando, dados (bi-direcional). Na figura 18, as conexões com o tradutor de protocolos e os outros módulos, barramento Avalon e controlador ethernet DM9000A.

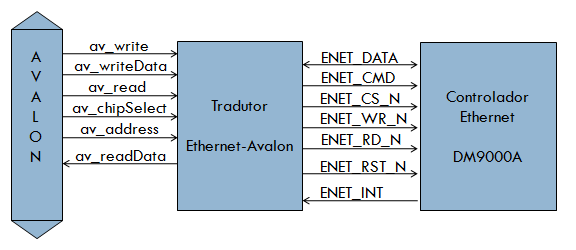


Figura 18: Comunicação pin-accurate com o Tradutor de Protocolos.

No protocolo Avalon, a operação de escrita é feita setando o pino de escrita e o selecionador de chip para alta voltagem, após ter sido indicado o endereço e o dado da escrita. Esta operação está demonstrada através de *waveforms* na figura 19.

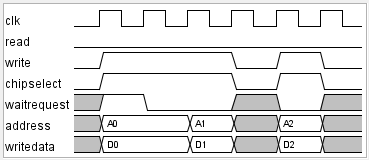


Figura 19: Operação de Escrita utilizando o Protocolo Avalon.

Já a operação de leitura é feita setando o pino de leitura e o selecionador de chip para alta voltagem, indicando o endereço que será obtido o dado. Após um ciclo de relógio, o dado estará disponível. Na figura 20, pode-se ver o *waveform* de execução da operação de leitura.



Figura 20: Operação de Leitura utilizando o Protocolo Avalon.

A comunicação com o protocolo ethernet é feita através do empacotamento de várias transações de escrita recebidas pelo barramento Avalon, que foram armazenadas no *buffer*. Cada frame enviado é composto por 80 bytes de dados mais 17 bytes do protocolo, como a largura de dados só é de 16 bits, várias operações de escritas são feitas ao controlador ethernet para forma um frame.

As operações de leitura são feitas através de interrupção, ou seja, no momento da chegada de um frame, o mesmo é interpretado através de várias operações de leitura do tradutor de protocolos, verificando tamanho e corretude do frame, e depois os dados são lidos e armazenados no *buffer* de recepção. A checagem de erro é feita automaticamente através do controlador, através de uma configuração prévia.

**4.2.4 Plataforma FPGA**

A plataforma FPGA terá o barramento Avalon como padrão para conseguir se comunicar com o tradutor de protocolos, além deste componente outros pertencentes a biblioteca do SOPC Builder ou criados pelos usuários podem ser adicionados ao sistema.

## Protocolo de Comunicação

O protocolo de comunicação utilizado foi baseado no padrão IEEE 802.3, conhecido como ethernet. O tipo de frame foi utilizado como um protocolo de transações entre as plataformas. Ela especifica qual operação será executada e quais parâmetros serão utilizados nela. Essa foi a única modificação em relação ao protocolo ethernet, especificado acima.

O envio de informações de uma plataforma em FPGA para uma no PDesigner é feito da seguinte forma:

1. Os dados serão armazenados em um buffer no tradutor de protocolos.
2. O tradutor de protocolos enviará os dados através da rede enquanto tiver dados disponível ou o controlador ethernet DM9000A estiver disponível.
3. No computador, verifica se os dados estão disponíveis, caso esteja repassa os dados para a plataforma no PDesigner, senão aguarda a chegada dos dados.

O caminho contrário, o envio de informações de uma plataforma no PDesigner para uma na FPGA, é feito da seguinte forma:

1. Os dados serão enviados para o controlador ethernet DM9000A a medida que uma solicitação for requerida.
2. O tradutor de protocolos aguarda uma interrupção, indicando que chegou dados, para armazenar no buffer de recepção.
3. O módulo que tentar executar uma leitura irá receber o dado mais antigo no buffer de recepção do tradutor de protocolos ou irá esperar a chegada de dados.

## Integração no PDesigner

A integração da co-emulação no PDesigner foi feita para tornar abstrato para o usuário, ao utilizar a ferramenta, as etapas fazer o download do projeto de hardware na placa e da aplicação em software, caso a plataforma na FPGA esteja utilizando o microprocessador Nios II.

Algumas modificações na estrutura da ferramenta foram necessárias para suportar esta técnica. A primeira delas foi a inserção de um campo para identificar a pasta onde se encontra os aplicativos “quartus\_pgm”, este é responsável por fazer o download do hardware na placa, e “nios2-download”, responsável por carregar um o binário da aplicação no microprocessador Nios II. Na figura 21, pode-se ver as modificações realizadas.

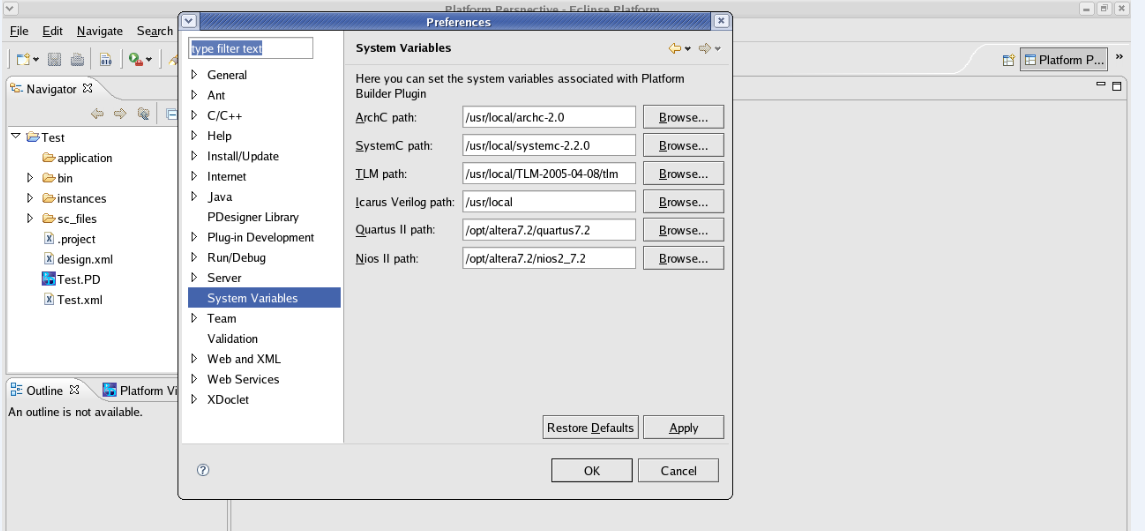


Figura 21: Identificação da Pasta da Altera no PDesigner.

A segunda modificação foi a inserção das funções para chamar automaticamente as atividades especificadas acima, através do *script* gerado pelo PDesigner, chamado de *generator.sh*. O script irá adicionar a chamada dos comandos, além do diretório que contém a plataforma em FPGA, realizando automaticamente as operações ao executar o projeto pelo PDesigner.

Após criar a plataforma na FPGA, o usuário tem que inserir no PDesigner através da criação de um arquivo XML, baseado no Spirit Consortium[25], que contém informações sobre ele, como protocolo utilizado e nível de abstração, e dois arquivos: um de extensão SOF e outro de extensão ELF, caso esteja utilizando algum microprocessador. Depois disso, o módulo deve ser adicionado a biblioteca, PDLibrary. Após esta etapa, pode-se selecionar o módulo inserido e conectar a outro módulo como mostrado na figura 22. Na execução, é abstraído para o usuário a co-emulação entre as duas plataformas.

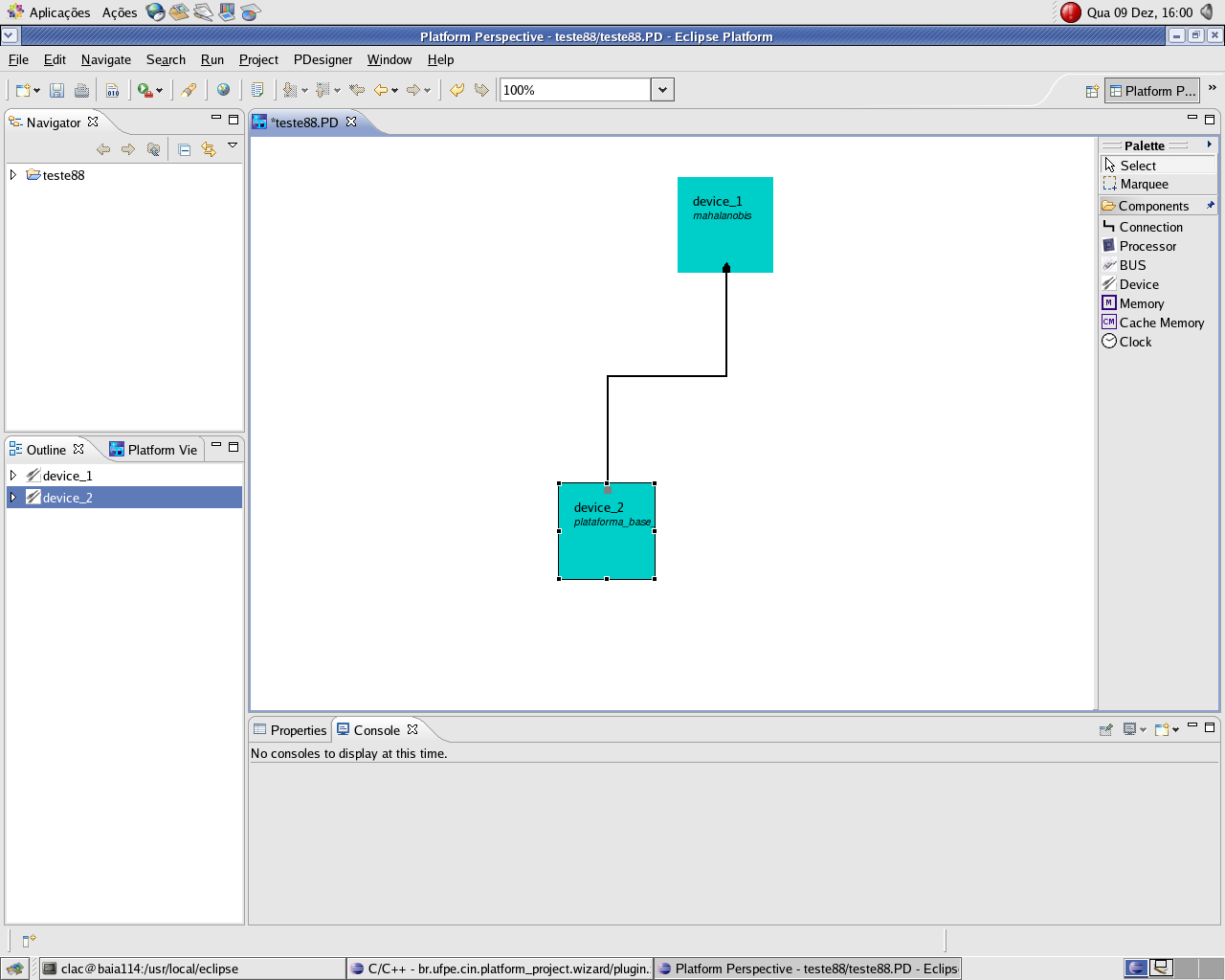


Figura 22: Conexão entre uma Plataforma no PDesigner com uma Plataforma FPGA.

# Estudo de Caso

Neste capítulo, será descrito o estudo de caso utilizado para realizar a segmentação de imagens, depois será descrito duas implementações, utilizando arquiteturas diferentes, e por último teremos um comparativo dos resultados obtidos.



## Visão Geral

A segmentação de imagem é uma das primeiras de muitas tarefas que um sistema de visão computacional tem que lidar para realizar reconhecimento de padrões. Ela consiste em dividir uma imagem em múltiplas regiões para simplificar ou mudar sua representação facilitando sua análise.

O estudo de caso é um sistema de segmentação de imagem de mãos para facilitar a identificação de padrões formados pela mesma através do cálculo da função distância de mahalanobis. Esta função é a distância medida baseada nas correlações entre as variáveis, na qual diferentes padrões podem ser identificados e analisados. Esta distância é calculada através do vetores multivariados *x*, a média de um grupo de valores e a matriz de covariância ∑.

x = ( x_1, x_2, x_3, \dots, x_p )^T

\mu = ( \mu_1, \mu_2, \mu_3, \dots , \mu_p )^T

D_M(x) = \sqrt{(x - \mu)^T \Sigma^{-1} (x-\mu)}.\,

Figura 23: Fórmula da Distância de Mahalanobis.

O sistema proposto é composto por dois módulos, um em software denominado *Skin* e outro em hardware denominado *Mahalanobis*. O módulo *Skin* é composto por um serviço denominado *calcMatrix*, que não possui nenhum entrada de dados e o resultado é uma matriz bidimensional. O módulo de hardware é composto por um serviço denominado *mahalanobisDistance*. Este serviço tem como entrada duas matrizes bidimensionais de inteiros e uma matriz bidimensional de booleanos como resultado. Na figura 24, pode-se ver o resultado do processo de segmentação.

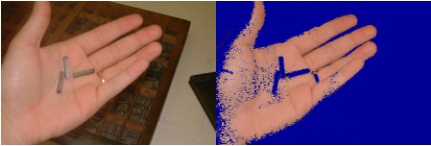


Figura 24: Imagem Real VS Imagem Segmentada.

## Arquitetura

A primeira arquitetura é uma plataforma no PDesigner, modelada em SystemC e ArchC, criada a partir do processo de síntese de comunicação. Ela contém um processador Mips modelado em ArchC, o barramento Simples Bus, uma memória rápida e o módulo do *Mahalanobis*. Alguns módulos são adicionados para permitir que a aplicação, *Skin*, que está rodando no processador se comunique através do driver com o módulo *Mahalanobis*, que terá um controlador de transações e uma interface com o barramento.



Figura 25: Plataforma em SystemC.

A outra arquitetura só manterá o módulo *Mahalanobis* e parte do controlador rodando no simulador em SystemC, o restante foi modificado para uma plataforma na FPGA. Pode-se observar que a plataforma da FPGA contém os seguintes módulos: um processador, Nios II, rodando a aplicação com um driver para acessar o Mahalanobis, o barramento Avalon, uma memória SRAM e diferente da outra arquitetura inserimos um tradutor de protocolos que converte do protocolo ethernet para o protocolo do controlador.

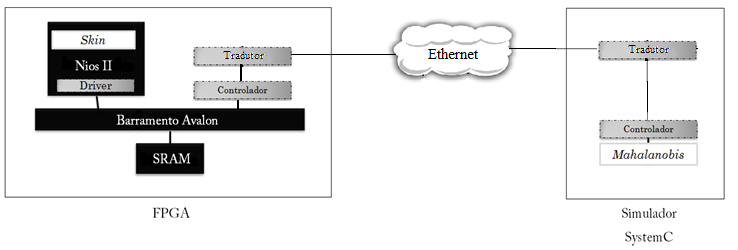


Figura 26: Plataforma no Sistema Co-emulado.

## Protocolos

O protocolo utilizado na plataforma virtual foi o seguinte:

* Uma solicitação é enviada para execução da função.
* Os parâmetros da função são repassados.
* Uma solicitação de escrita ou leitura é enviada, se for disponibilizado pela função.
* O status da solicitação é enviado.

Já o protocolo, utilizado pela plataforma na FPGA foi um pouco mais simples, ele foi definido da seguinte forma no dois bytes do protocolo ethernet:

* 0x00001: Transmissão de dados da FPGA para o dispositivo.
* 0x00002: Recepção de dados na FPGA.
* Outros valores: Frame será descartado.

## Resultados

Para realização dos teste foi utilizado um máquina virtual, VMware Workstation 6.0.4, rodando o sistema operacional Ubuntu 9.1.0 com a seguinte configuração:

* Processador: Core 2 Duo T5750 2.0GHz
* Memória RAM: 804MB
* HD: 8Gb
* Ethernet 10/100Mbits

A verificação da comunicação entre as plataformas foi feita através do programa Wireshark[x]. Ele é um programa para análise de protocolo na rede, através do qual os dados são mostrados em tempo-real e disponibilizados para análise, como mostrado na figura 27.

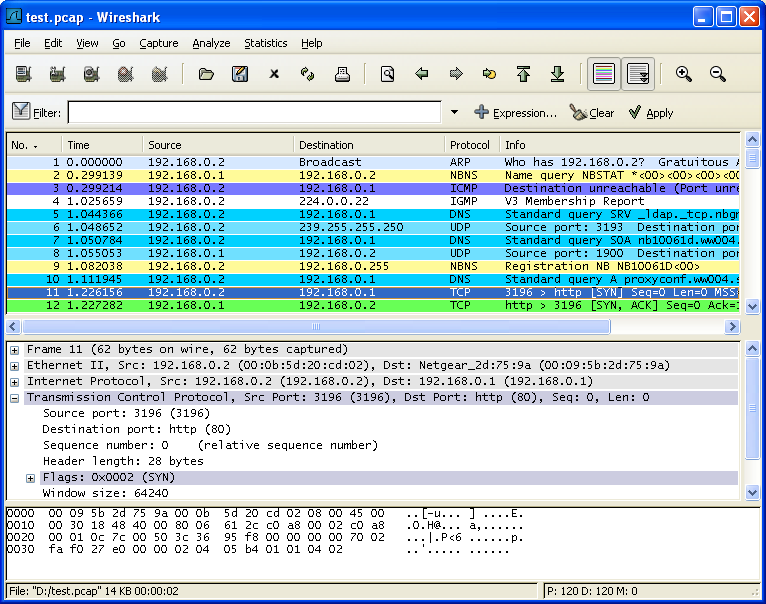


Figura 27: Análise de Pacotes através do Wireshark.

A análise do tempo foi medida através de duas funções do padrão C ANSI, dentro do arquivo “time.h”, *time* e *difftime*. A primeira pega o tempo atual do sistema, com precisão de segundos. A outra função é responsável por calcular a diferença entre os tempos obtidos na simulação entre o início e o fim, após a troca de dados e o processamento.

Os resultados obtidos foram:

|  |  |
| --- | --- |
| Plataforma Co-emulação | Plataforma Simulada |
| 5.0 segundos | 45.0 segundos |

Tabela 1: Resultados.

Percebe-se que a velocidade de simulação do sistema co-emulado foi nove vezes maior do que o sistema em SystemC. Isto ocorreu porque o custo computacional da simulação é alto, devido a necessidade de simular várias atividades realizadas como o ciclo de *clock*.

# Conclusões e Trabalhos Futuros

A técnica apresentada visa diminuir o *gap* entre plataformas com alto nível de abstração e o hardware, diminuindo a possibilidade de erro na fase de prototipação do sistema, além de permitir uma forma de verificação através da descrição de módulos descritos em alto nível de abstração, que são mais simples de serem implementados e permitem uma cobertura maior do testes.

Os trabalhos futuros serão voltados para automatização do processo, tornando a interação com o dispositivo simples para o usuário, permitindo explorar arquitetura de sistemas sem se preocupar com os detalhes da implementação. A criação automática de uma plataforma FPGA parametrizável, permitindo ao usuário especificar o sistema e definir o que será utilizado na FPGA. Algumas alternativas de comunicação serão exploradas, como padrões Gigabits, fibra óptica e USB, para realizar um estudo comparativo das vantagens e desvantagens de cada técnica.

# Referências Bibliográficas

**[1] SCE-MI** – Online, acesso em 01/12/2009 na url <http://www.eda.org/itc/scemi.pdf>

**[2] What is Emulation?** – Online, acesso em 01/12/2008 na url <http://www.kb.nl/hrd/dd/dd_projecten/projecten_emulatiewatis-en.html>

**[3] Altera DE2-70** – Online, acesso em 01/12/2009 na url <http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=226>

**[4] Verilog** – Online, acesso em 01/12/2009 na url <http://www.verilog.com/>

**[5] PDesigner Framework** – Online, acesso em 01/12/2009 na url <http://www.pdesigner.org/>.

**[6] Introduction to SOPC Builder** – Online, acesso em 01/12/2009 na url <http://www.altera.com/literature/hb/qts/qts_qii54001.pdf>

**[7] Design & Reuse** – Online, acesso em 01/12/2008 na url <http://www.design-reuse.com>

**[8] FPGA Central** – Online, acesso em 01/12/2009 na url <http://www.fpgacentral.com>

**[9] IEEE 802.3 Standard** – Online, acesso em 01/12/2009 na url <http://standards.ieee.org/getieee802/download/802.3-2008_section1.pdf>

**[10] Redes Locais: Ethernet** – Online, acesso em 01/12/2009 na url <http://www.pucsp.br/~paulino/as2001/aula9.html>

**[11] Lei de Moore** – Online, acesso em 01/12/2008 na url <ftp://download.intel.com/museum/Moores_Law/Video-Transcripts/Excepts_A_Conversation_with_Gordon_Moore.pdf>

**[13] Wireshark** – Online, acesso em 01/12/2009 na url <http://www.wireshark.org/>

**[14]** Bailey, B., Martin, G. and Piziali, A., **ESL Design and Verification: A Prescription for Electronic System Level Methodology.**  Editora Morgan Kaufmann/Elsevier, 2007.

**[15]** Tommy Baumann, Maik Hauguth, Horst Salzwedel. **Overcoming the Gap between Design at Electronic System Level (ESL) and Implementation for Networked Electronics.**

**[16]** Kurose, J. F. and Ross, K. W., **Redes de Computadores e Internet : uma abordagem top-down.** Editora Pearson Addison-Wesley, 2006.

**[17]** J. B. Mena, J.A. Malpica. **Color Image Segmentation Using the Dempster-Shafer Theory of Evidence for the Fusion of Texture.**

**[18]** Nakajima, T. and et al. **A Portable Co-Verification System Which Generates Testbench Automatically.** International Conference on[**Field-Programmable Technology, 2007.**](http://ieeexplore.ieee.org/xpl/RecentCon.jsp?punumber=4439213)

**[19]** Y. B. Liao, P. Li, A. W. Ruan, Y. W. Wang, W. C. Li, W. Li, **A Stream-Mode Based HW/SW Co-Emulation System for SOC Test and Verification.** Ninth International Workshop on Microprocessor Test and Verification, 2008.

**[20]** Schildt, H., **C Completo e Total.** Editora Makron São Paulo, 1990.

**[21]** Brooks, F. P., **The Mythical Man-Month.** Editora Addison-Wesley, 1995.

**[22] SystemC** – Online, acesso em 01/12/2009 na url <http://www.systemc.org/>

**[23] ArchC** – Online, acesso em 01/12/2009 na url <http://archc.sourceforge.net/>

**[24] Icarus Verilog Simulator** – Online, acesso em 01/12/2009 na url <http://www.icarus.com/eda/verilog/>

**[25] Spirit Consortium** – Online, acesso em 01/12/2009 na url

<http://www.spiritconsortium.org/home/>

**[25] Framework Eclipse** – Online, acesso em 01/12/2009 na url <http://www.eclipse.org/>

**[26] Altera Avalon** – Online, acesso em 01/12/2009 na url [www.altera.com/literature/manual/mnl\_avalon\_spec.pdf](http://www.altera.com/literature/manual/mnl_avalon_spec.pdf)

**[26] Projeto Fenix** – Online, acesso em 01/12/2009 na url <http://www.cin.ufpe.br/~greco/projects.php>

**[27] Accellera** – Online, acesso em 01/12/2009 na url <http://www.accellera.org>

**[28] Verilator** – Online, acesso em 01/12/2009 na url <http://www.veripool.org/wiki/verilator>

**[29] PCI-X Development Board** – Online, acesso em 01/12/2009 na url <http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=56&No=317>