RAFAEL SCHEFFER VARGAS

SISTEMAS EMBARCADOS: ACOPLAMENTO DO SOFT-CORE PLASMA AO BARRAMENTO OPB DE UM POWERPC 405

Monografia apresentada ao Curso de Ciências da Computação como requisito parcial à obtenção do grau de Bacharel em Ciências da Computação.

Universidade Federal de Santa Catarina.

Orientador: Prof. Dr. Antônio Augusto Fröhlich.

FLORIANÓPOLIS, 2007

RAFAEL SCHEFFER VARGAS

SISTEMAS EMBARCADOS:
ACOPLAMENTO DO SOFT-CORE PLASMA AO
BARRAMENTO OPB DE UM POWERPC 405

Esta monografia foi julgada adequada à obtenção do grau de Bacharel em Ciências da Computação e aprovada em sua forma final pelo Curso de Ciências da Computação da Universidade Federal de Santa Catarina.

Florianópolis – SC, fevereiro de 2007.

_____

Prof. Dr. Antônio Augusto Fröhlich.
Universidade Federal de Santa Catarina

_____

BsC. Hugo Marcondes
Universidade Federal de Santa Catarina

_____

MsC. Rafael Luiz Cancian
Universidade Federal de Santa Catarina

Sábio o homem que inventou a cerveja.

Platão

RESUMO

Barramento é um conjunto de linhas que permite a comunicação entre componentes, como memória, processadores e demais periféricos. Eles conectam diferentes partes para que ocorra a transmissão de dados. O objetivo deste trabalho é transformar o Plasma em um co-processador ligado ao barramento *On-chip Periferal Bus* (OPB) para aumentar a capacidade de processamento de um sistema. Para a implementação, é utilizada a linguagem VHDL. A proposta é que, após sintetizado, este sistema possa ser rodado em FPGA.

Palavras-chave: Barramento; System-on-a-Chip; Co-processador.

ABSTRACT

Bus is a set of wires connecting the various functional units in a computer, such as memory, processor and peripheral devices. The bus connects different parts to allow data transmission. This project aims to increase the processing capability of a system by transforming Plasma into a co-processor connected to the *On-chip Periferal Bus* (OPB). It is used the VHDL language to do the implementation. The proposal is to synthesize the system, and, after that, run it in FPGA.

Keywords: Bus; System-on-a-Chip; Co-processor.

Sumário

LISTA DE ABREVIATURAS

ASIC: Application Specific Integrated Circuit

CPU: Central Processing Unit

EDK: Embedded Development Kit

FPGA: Field Programmable Gate Array

HDL: Hardware Description Language

IP: Intellectual Property

OPB: On-chip Periferal Bus

RISC: Reduced Instruction Set Computer

SoC: System-on-a-Chip

UART: Universal Asynchronous Receiver-Transmitter

VHDL: VHSIC Hardware Description Language

VHSIC: Very-High-Speed Integrated Circuit

1 Introdução

Os sistemas embarcados, cada vez mais comuns na sociedade, estão ficando complexos e poderosos graças aos avanços da tecnologia. Estes avanços possibilitam que estes sistemas possuam uma maior capacidade de processamento, memória e adaptação a diferentes necessidades, podendo ser reconfigurados de acordo com a aplicação a qual se destinam.

O aumento desta flexibilidade e complexidade impacta diretamente no esforço de desenvolvimento destes sistemas. Uma das tendências tecnológicas é o desenvolvimento de todo um sistema em um único chip. Os SoCs (do ingles, *System-on-a-Chip*), como são chamados, consistem no hardware e também no software que o controle.

Em sua maioria, SoCs são criados a partir de componentes pré-validados, juntamente com os drivers que controlam sua operação. Estes componentes são agrupados utilizando-se de ferramentas CAD e seu software utilizando-se de um ambiente para desenvolvimento de software.

A chave no projeto de SoCs é a simulação, que se obtém utilizando ferramentas tanto de software quanto de hardware. No caso de simulação de software, tem-se, de modo geral, a validação comportamental do sistema. Na simulação com hardware, geralmente utiliza-se FPGAs, as quais imitam o comportamento final do SoC, onde se pode testar e depurar o software e o hardware em uma velocidade próxima da total de operação do sistema. Mais da metade do esforço e tempo de projeto de um SoC é gasto com a verificação e validação.

Para a definição dos componentes de hardware, geralmente utiliza-se de linguagens de descrição de Hardware (HDL, do inglês *Hardware Description Language*). As mais comuns são VHDL e Verilog, sendo que este trabalho utilizará somente VHDL.

1.1 Motivação

Os sistemas embarcados têm importância tecnológica, econômica e social. São utilizados em controladores de vôo, telefones, equipamentos de redes de computadores, impressoras, drives, calculadores e eletrodomésticos. As pesquisas nessa área são pertinentes no contexto atual, onde há competição mercadológica, principalmente no ramo de produtos eletrônicos.

Um dos diferenciais dos equipamentos eletrônicos é a sua capacidade de processamento de informação. Quanto mais rápido e de modo mais preciso se obtém um dado, mais fácil é a ação e a gestão de conhecimento. Por isso, as empresas têm investido com freqüência em pesquisas tecnológicas que visem o aumento da capacidade de processamento dos seus sistemas. Em 2006, a Intel anunciou que investira aproximadamente US$ 6 bilhões em esquisa e desenvolvimento. Segundo dados da IT Web, em 2007, a concorrente Samsung Electronics, uma das maiores fabricantes de processadores, vai investir US$ 1,9 bilhões em processadores.

Para se ter uma idéia do desenvolvimento dos processadores nos últimos 30 anos, segundo Wagner (2006), em 1970 os processadores possuíam poucos milhares de transistores em um chip. Em 2005, este número aumentou para dezenas a poucas centenas de milhões de transistores e, em 2010 a expectativa é que se tenha um total de mais de um bilhão de transistores em um chip.

Um dos *softcores* disponíveis no mercado é o Plasma, um processador de código aberto criado em linguagem VHDL e que implementa o conjunto de instruções MIPS I. Esse fato torna possível utilizar o GCC para gerar códigos para o Plasma, o que facilita o desenvolvimento de um software.

Este é o ponto-chave deste trabalho: como utilizar uma tecnologia livre, o Plasma, para criar uma alternativa de processamento que incentive futuras pesquisas e contribua para o desenvolvimento tecnológico dos sistemas embutidos?

1.2 Objetivos

1.2.1 Objetivo geral

Este trabalho objetiva transformar o Plasma em um co-processador ligado ao barramento *On-chip Periferal Bus* (OPB) para aumentar a capacidade de processamento de um sistema.

1.2.2 Objetivos específicos

- Estudar o desenvolvimento de um sistema embarcado, utilizando a ferramenta de desenvolvimento para sistemas embarcados da Xilinx (EDK) e adicionando componentes personalizados para que, finalmente, possa-se obter um SoC baseado somente nos componentes personalizados, não necessitando então do PowerPC, gerado pelo EDK.
- Sintetizar o soft-core Plasma na FPGA presente na ML-310, um kit de desenvolvimento diferente do qual o soft-core foi projetado, para ser sintetizado com um IP de acoplamento ao barramento OPB. E então acoplar o este IP ao barramento OPB do PowerPC.
- Alterar a estrutura interna de acesso à memória do Plasma para permitir que um barramento externo acesse sua memória sem atrapalhar o funcionamento do processador.
- Estudar a arquitetura do barramento OPB e criar um IP que possa ser acoplado a este barramento, sendo que este IP funcione como mestre e também como escravo. A proposta é permitir que o processador possa ler a partir do barramento sempre que necessário, e este que consiga também ler sua memória sem a necessidade de interrupção.

2 Sistemas embarcados e linguagem de descrição de hardware

2.1 Sistemas embarcados

Sistemas embarcados ou embutidos são aqueles que manipulam dados dentro de sistemas ou produtos maiores, não possuem interface com o usuário e executam uma função específica. Geralmente são projetados para realizar uma função ou uma gama de funções e não para serem programados pelo usuário final, como os computadores pessoais. Eles interagem com o ambiente em que se encontram e coletam dados de sensores para modificar o ambiente utilizando atuadores. Devem ser à prova de falhas, visto que interagem com o meio e causam impactos.

Segundo Wolf (2001), os sistemas embarcados apresentam características em comum com os sistemas computacionais, mas não possuem a mesma uniformidade. Cada aplicação apresenta requisitos diferentes de desempenho, consumo de potência e área ocupada. Isso pode acarretar em uma combinação diferente de módulos de hardware e software para atender a esses requisitos. Para Yaghmour (2003), os sistemas embutidos podem ser classificados segundo quatro critérios: tamanho, conectabilidade, requisitos de tempo e interação com o usuário.

Oliveira (2006) registra o histórico dos sistemas embarcados, que foram usados a partir de 1960, mas, até a década de 80, ainda necessitavam de uma série de componentes externos para funcionar provendo memória de armazenamento e de trabalho. Após a década de 80, com a queda do custo do microcontrolador, os sistemas embarcados puderam substituir componentes físicos, como potenciômetros e capacitores variáveis. Hoje estão em quase todo equipamento eletrônico. Oliveira (2006) afirma que se calcula nos Estados Unidos uma média de oito dispositivos baseados em microprocessadores por pessoa. Apesar de esse parecer um número muito grande, basta observar os equipamentos eletrônicos: aparelhos de TV e de DVD, tocadores de MP3, telefones, sistemas de alarme, relógios e celulares são alguns dos exemplos de aparelhos que utilizam sistemas embarcados.

2.2 FPGA

Giorgini (2001) afirma que na indústria eletrônica, o tempo de desenvolvimento e de produção deve ser cada vez mais reduzido para limitar o risco financeiro presente no desenvolvimento de um novo produto. Para solucionar essa questão, segundo os autores, foi criado o FPGA (Field Programmable Gate Array), cuja estrutura pode ser configurada pelo usuário final, o que permite uma rápida fabricação e construção de protótipos a custos muito reduzidos. A escolha dos FPGAs para implementar um co-processador deve-se aos requisitos de alto desempenho, baixo custo e facilidade de reconfiguração.

O FPGA, introduzido em 1985 pela empresa californiana Xilinx Inc., é um circuito integrado específico para construção de sistemas digitais implementado como uma matriz de blocos funcionais em uma rede de interconexão e cercado por blocos de entrada e saída (I/O blocks). Segundo Giorgini (2001), o termo *field* (campo) indica que a programação do FPGA pode ser feita no lugar de aplicação. É o usuário quem especifica, no momento da programação, a função de cada bloco lógico do sistema e as suas conexões. Na atualidade existem muitos tipos de FPGA lançados por empresas como Actel, Altera, AMD, Crosspoint Solutions, Atmel entre outras.

Os FPGAs contêm atualmente mais de 2,5 milhões de portas lógicas. Por isso, desenvolver projetos utilizando apenas diagramas esquemáticos pode ser uma tarefa muito difícil. Os projetistas estão adotando cada vez mais o projeto de FPGAs baseado em HDL (Hardware Description Language). Giorgini (2001) afirma que a utilização dessas ferramentas permite um aumento de produtividade, pois o projeto pode ser feito em um nível de abstração mais alto (RTL, do inglês *register transfer level*), ao invés de nível lógico booleano ou de portas.

2.3 VHDL

Segundo Navabi (1998), VHDL ou VHSIC Hardware Description Language é uma linguagem de descrição de hardware comumente usada como linguagem de entrada para FPGAs na automação do projeto de circuitos eletrônicos. Criada originalmente para o Departamento de Defesa dos EUA para documentar o comportamento dos ASICs que eram incluídos nos equipamentos que comprava, substituía manuais grandes e complexos.

Posteriormente, foram criados simuladores de hardware para a leitura do VHDL. Também surgiram os sintetizadores lógicos que liam VHDL e tinham como saída uma definição física da implementação do circuito.

A sintaxe do VHDL é pesadamente baseada em ADA, porém seus meios de especificar o paralelismo inerente ao projeto de hardware (processos) são diferentes dos meios de ADA (tarefas).

Existem também outras linguagens para descrição de hardware, como Verilog, porém o VHDL tem como vantagens o fato de ser uma linguagem fortemente tipada. Polpeta (2006) diz que em Verilog não há o conceito de pacotes, assim como não existem diretivas para a configuração ou replicação de estruturas e o recurso de parametrização é muito pobre, sendo assim, necessário sobrescrever constantes ao longo do processo de síntese para que, deste modo, os componentes sejam pré-processados corretamente.

A linguagem VHDL pode ser empregada para descrever hardware através de três abordagens distintas: a abordagem estrutural, a data flow e a comportamental. Geralmente emprega-se uma mistura das três abordagens para formar um projeto por diferentes seções, expressas de diferentes modos.

Para Giorgini (2001), a linguagem VHDL oferece uma ampla variedade de níveis de abstração e possibilita mesclar esse níveis durante a simulação, tornando possível a adoção de um estilo de projeto verdadeiramente *top-down*. Essa linguagem possui também recursos excepcionais para modelagem de temporização em hardware e provê mecanismos para construção de modelos genéricos.

3 Plasma

Rhoads (2007) é o criador do Plasma, um *soft-core*, espécie de processador que pode ser sintetizado a partir de uma linguagem de descrição de hardware (HDL), neste caso VHDL. Sua vantagem é que ele implementa uma arquitetura de von Neumann utilizando apenas uma memória física para armazenar dados e instruções. Suporta praticamente todas as instruções especificadas pelo MIPS I Instruction Set, documentada por Price (1995), com exceção de apenas duas: carga e armazenamento não alinhado na memória não são suportados (são patenteados). Também não são suportadas exceções, apenas interrupções.

O compilador GCC para o MIPS não costuma gerar instruções de acesso não alinhado à memória, uma vez este tem suporte limitado na maioria das CPUs RISC. Para garantir que no programa não haja estas instruções, pode-se gerar uma lista com as utilizadas pelo programa compilado e procurar por: LWL, LWR, SWL ou SWR.

3.1 Funcionamento do microprocessador

Patterson e Hennessy (1997) descrevem a execução de uma instrução em um processador RISC com pipeline de cinco estágios, sendo que esta se dá por meio dos seguintes passos e ações. Cada instrução necessita de três a cinco dos passos descritos a seguir.

1. **Busca da instrução**

Busca a instrução da memória, apontada por PC, e calcula o endereço da próxima instrução somando quatro ao valor atual de PC.

2. **Decodificação da instrução e busca dos registradores**

São lidos os registradores identificados por `rs` e `rt` (e gravados em registradores temporários A e B, respectivamente), supondo que a instrução seja do tipo R, pois, caso não seja, os valores desnecessários podem ser simplesmente descartados. Também é calculado o possível valor da próxima instrução, caso esta seja um *branch*, que também pode ser ignorado caso não seja.

3. **Execução, computação do endereço de memória, ou término do *branch***

Neste passo a ALU (do inglês *Arithmetic and Logic Unit*) executa uma de quatro funções, dependendo do tipo da instrução, podendo esta função ser um dos itens abaixo.

- Endereço de memória: o resultado será A somado do valor do campo "imediato" com o sinal extendido para 32 bits.

- Cálculo lógico-aritmético: o resultado será A (operação) B, onde (operação) pode ser qualquer operação da ALU.

- *Branch*: subtrai B de A, e caso o valor seja igual a zero (valores iguais) atualiza o valor de PC para o endereço calculado no ciclo anterior.

- *Jump*: concatena os 4 bits de mais alta ordem do PC com os 26 bits de mais baixa ordem da instrução deslocados 2 bits para a esquerda. E o valor resultante é escrito em PC.

4. **Acesso à memória ou término de instrução lógico-aritmética**

No caso de acesso á memória o endereço é determinado pela saída da ALU no ciclo anterior. Para leitura, o valor lido é salvo para o próximo ciclo, e para gravação, é gravado o dado de B.

Já no caso de instrução lógico-aritmética (tipo R) o registrador identificado por `rd` é escrito com o valor da saída da ALU no ciclo anterior.

5. **Término de leitura de memória**

Aqui, o valor lido da memória no ciclo anterior é escrito no registrador identificado por `rd`.

A Figura 1, de Rhoads (2007), ilustra os componentes do Plasma e as ligações entre eles.

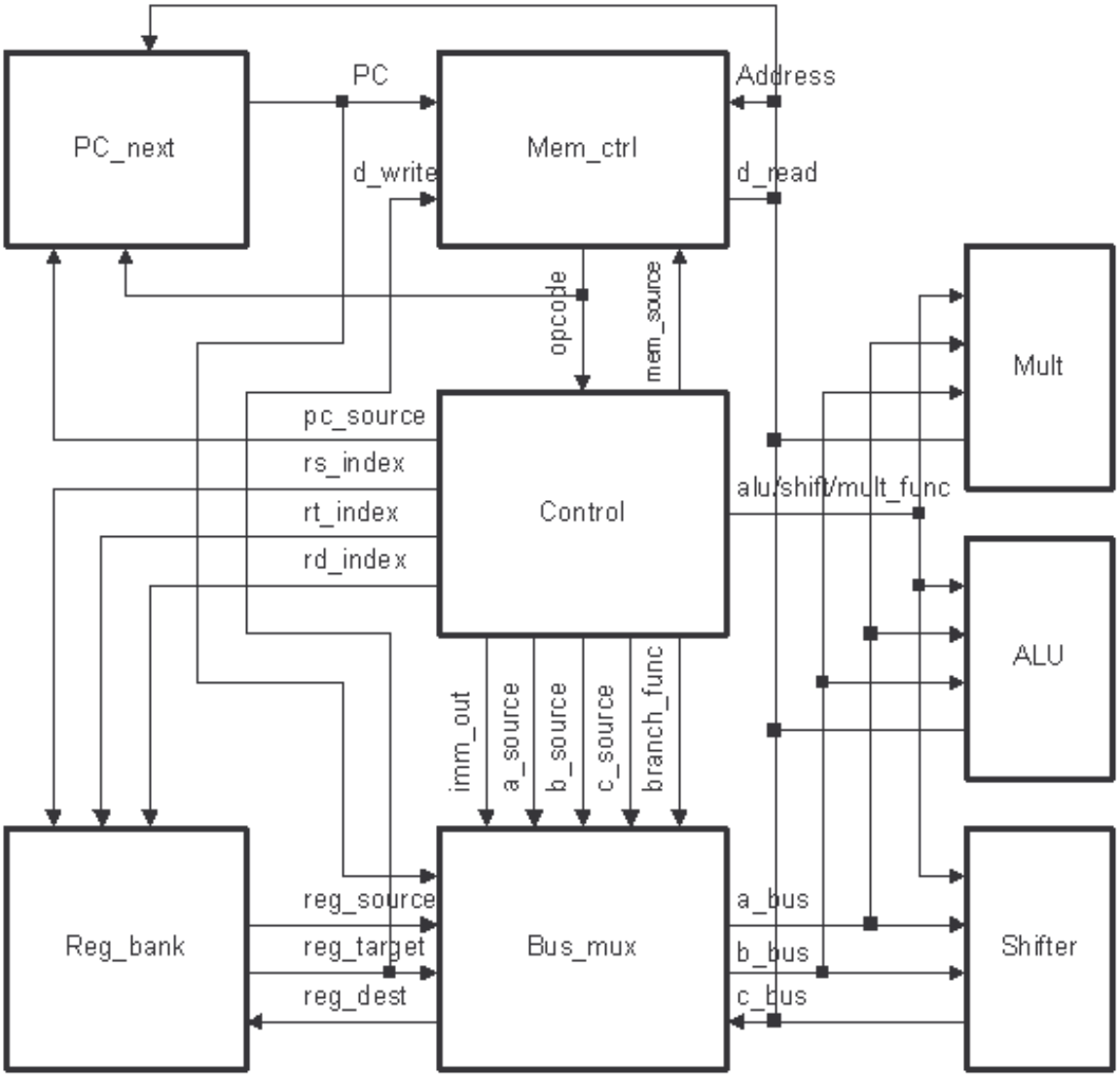

Figura 1: Diagrama de blocos do Plasma.
Fonte: Rhoads, 2007.

3.2 Periféricos

Juntamente com o hardware do microprocessador, o Plasma inclui um UART (do inglês *Universal Asynchronous Receiver-Transmitter*), possibilitando a comunicação serial bidirecional. Também agrega um controlador de interrupção e um timer com interrupção.

O UART, tipo de circuito mais comum usado nos modems de computadores pessoais, é um módulo geralmente composto de um único circuito integrado que contém, ao mesmo tempo, os circuitos de transmissão e recepção necessários para as comunicações seriais assíncronas.

4 O barramento OPB

O barramento de periféricos *on-chip* (OPB, do inglês *On-chip Periferal Bus*) foi projetado para permitir uma conectividade fácil entre dispositivos periféricos em um mesmo chip. Segundo dados da IBM (2001), são as principais características do OPB:

- endereçamento de até 64 bits;
- barramento de dados de 32 ou 64 bits;
- totalmente síncrono;
- suporte a escravos de 8, 16, 32 ou 64 bits;
- suporte a mestres de 32 ou 64 bits;
- definição de tamanho dinâmico, com transferências de *byte, halfword, fullword* e *doubleword*;
- suporte a protocolo de endereços seqüenciais;
- *timeout* de 16 ciclos provido pelo árbitro de barramento; e
- supressão de *timeout* pelo escravo.

Dangui (2006) afirma que o barramento OPB é síncrono e projetado para a conexão de dispositivos de baixa performance e baixo consumo de energia, não sendo indicado para ser conectado ao processador central de um sistema.

5 IP de acoplamento ao barramento OPB

O *soft-core* que deve ser acoplado ao barramento irá realizar operações leitura e escrita no OPB. Na sua arquitetura, este processador possui uma memória interna de 8Kb e uma interface de memória externa. É nesta interface de memória externa que Plasma solicitará leituras e escritas no barramento. E em sua memória interna o barramento poderá solicitar leitura e escrita, possibilitando assim que os programas que devem ser executados sejam passados à memória do microprocessador.

Devido à natureza das operações que o processador realiza, é necessário que este atue no barramento OPB, tanto como mestre, nas operações nas quais ele requisita leitura ou escrita, através da interface de memória externa, nos periféricos conectados a este barramento; quanto como escravo, nas operações nas quais o barramento solicita a leitura ou escrita na memória interna do Plasma. Seria então necessário implementar um módulo que pudesse:

- acoplar-se ao barramento OPB como mestre e como escravo;
- conectar-se à interface de memória externa do Plasma;
- acessar a memória interna do Plasma, tanto para leitura quanto para a escrita;
- mapear para o barramento OPB as porta de GPIO do Plasma; e
- mapear o sinal de reset do Plasma para o barramento.

6 Transformação do Plasma em um co-processador ligado ao barramento *On-chip Periferal Bus* (OPB)

6.1 Passo 1: a escolha dos componentes

Para desenvolver o projeto, inicialmente foram escolhidos os elementos necessários na construção do sistema com co-processador, os quais eram:

- circuito integrado (FPGA);
- dispositivo de comunicação com a FPGA; e
- software para interpretação da linguagem e síntese do hardware.

A seguir são detalhados os requisitos de cada um desses elementos bem como informados quais componentes foram escolhidos.

6.1.1 Circuito integrado (FPGA)

Para a realização do projeto é necessária a utilização de um FPGA que comporte o sistema completo. Este *chip* é uma matriz de portas lógicas, cujo número é limitado, de acordo com o modelo. A princípio, foi cogitado o uso da Xilinx Spartan-3, contida no kit de desenvolvimento *Spartan-3 starter kit*. Porém, após alguns testes, verificou-se que só o Plasma ocupava 90% da capacidade. Desta forma, optou-se pelo uso de uma FPGA mais robusta.

O modelo escolhido foi a Xilinx Virtex II pro, que vem no kit de desenvolvimento ML 310. Neste segundo modelo, o Plasma ocupou 13% das portas lógicas e, então, observou-se que com esta FPGA era possível sintetizar o sistema com o PowerPC, o barramento e o Plasma.

6.1.2 Dispositivo de comunicação com a FPGA

Neste projeto, são necessários dois tipos de comunicação: um para verificar os resultados através da saída do sistema e outro para programar a FPGA.

Para observar os resultados, foi utilizado o cabo serial RS-232, pois é um modelo facilmente encontrado no mercado e que permite uma comunicação simples, eficiente e bilateral. Sua implementação não é complexa.

Para programar a FPGA normalmente utiliza-se um cabo JTAG. De acordo com o manual do kit de desenvolvimento da Xilinx ML 310, observou-se que o único modelo compatí-

vel é o JTAG4. No entanto, não foi possível obter este cabo e, na ausência dele, foi decidido que a programação dar-se-ia através do cartão *compact flash*, presente no kit.

6.1.3 Software para interpretação da linguagem e síntese do hardware

Para programar a FPGA utiliza-se uma linguagem de descrição de hardware, geralmente VHDL ou Verilog. Neste projeto, optou-se pela VHDL, por ser a linguagem em que o Plasma estava implementado e devido ao conhecimento prévio do autor do projeto.

A transformação da VHDL no formato de entrada da FPGA é feita pelo software de síntese. Neste projeto foi escolhido o pacote de desenvolvimento da Xilinx, o ISE, por ser compatível com o *chip* e por estar disponível gratuitamente na Web.

Além desse software, foi utilizado também o EDK (do inglês *Embeded Development Kit*), um kit de desenvolvimento da Xilinx para sistemas embarcados, responsável por gerar o PowerPC, o barramento e o dispositivo de comunicação serial.

6.2 Sintetização do Plasma

Após a definição dos elementos necessários, houve uma tentativa de sintetização do Plasma na Xilinx Virtex II pro. Como ocorreram imprevistos neste processo, optou-se, num primeiro momento, por sintetizar o Plasma na FPGA Xilinx Spartan-3. A escolha obteve resultado positivo, e foi possível testar a modificação do acesso à memória do processador.

Foi modificado o acesso de memória do Plasma para que ele funcionasse na metade do *clock* da memória. Isso possibilitou que o barramento externo, neste caso o IP de acoplamento, acessasse a memória interna do processador no *clock* oposto ao *clock* que o Plasma acessa a memória. Este procedimento mostrou que a ação de acesso do Plasma funcionou corretamente. Porém, como a Xilinx Spartan-3 tem capacidade limitada, não foi possível verificar o perfeito funcionamento do barramento, o qual pode ser observado apenas na etapa de simulação de software.

Após a modificação, o Plasma estava funcionando na freqüência de 12,5MHz enquanto a memória continuava a funcionar à 25Mhz.

6.3 Implementação do IP

Para realizar a implementação do IP cumprindo os requisitos propostos, foi necessário modificar a estrutura de memória interna do Plasma para que este passasse a funcionar apenas com metade do *clock* que rodava originalmente enquanto a memória continuaria a funcionar com o *clock* original do sistema. Deste modo, um barramento externo, o OPB neste caso, poderia acessar a memória interna do *soft-core* de maneira completamente transparente, sem a necessidade de solicitar interrupções ao processador.

Pronta a modificação no acesso à memória do Plasma, fora então necessário implementar o módulo de acoplamento do barramento OPB em VHDL. A ferramenta de projeto de sistemas embarcados da Xilinx, o EDK, gera templates para IPs de acoplamento ao barramento, o qual utiliza de um IP da Xilinx para a tradução dos sinais do barramento.

O template foi gerado tendo três portas de entrada como escravo. Estas portas foram ligadas à memória interna do Plasma, à porta de GPIO e ao sinal de *reset* do microprocessador. O lado mestre do barramento, conectado à interface de memória externa, fora implementado de modo que, sempre que o Plasma solicitar uma leitura na memória, será feita uma leitura no barramento e, da mesma forma, analogamente, para a escrita.

7 Resultados e conclusão

A expectativa inicial era ter em um *chip* uma configuração composta de um processador PowerPC, gerado pelo EDK, ligado a um barramento PLB. Uma bridge PLB-OPB ligaria os dois barramentos (PLB ao OPB). Ao OPB estariam conectados o dispositivo de comunicação serial e o Plasma – este último por meio do IP de acoplamento do barramento OPB.

Após o desenvolvimento e a validação através de simulação de software dos componentes em separado, foi feita a tentativa de síntese e validação em FPGA do sistema. Para isto foi criada uma configuração padrão dentro do EDK, que contém:

- um processador PowerPC;
- um árbitro para o barramento PLB, no qual liga-se o processador principal;
- 64Kb de memória BRAM (do inglês *Block Random Access Memory*, bloco de memória interno à FPGA);
- um árbitro para o barramento OPB;
- uma *brigde* PLB-OPB para possibilitar a comunicação de mestres no PLB com escravos no OPB;
- uma *bridge* OPB-PLB para possibilitar a comunicação de mestres no OPB com escravos no PLB; e
- o barramento OPB ao qual está ligado 8Kb de memória BRAM, um Timer e um dispositivo de comunicação serial (UART).

7.1 Validação do IP de acoplamento ao barramento OPB

Nesta configuração padrão, a princípio, tentou-se a síntese do sistema sem o Plasma para validação do IP de acoplamento. O Plasma foi substituído por registradores dentro do componente, para que, ao ser feita tentativa de escrita e leitura, fosse possível obter dados sobre o funcionamento ou não deste IP, que foi adicionado à configuração padrão do EDK.

Fora então escrito um programa que deveria ser executado no PowerPC que tentasse escrever uma seqüência de bits em cada um dos registradores do IP e então tentasse ler o que havia sido escrito.

Após a síntese desta configuração e programação da FPGA foi possível observar o perfeito funcionamento do IP de acoplamento ao barramento OPB.

7.2 Síntese e validação do sistema

Uma vez que não foi possível verificar o funcionamento do Plasma na Virtex II Pro, e que a Spartan-3 não comporta o Plasma, com o barramento e o PowerPC, não se obteve sucesso na validação do sistema. Os motivos pelo qual não se obteve sucesso são, provavelmente, incompatibilidades entre a Spartan-3 e a Virtex II Pro quanto ao tempo de propagação de sinais, além de possíveis problemas com a ferramenta de síntese, pois foi observado que ao efetuar a síntese para a Spartan-3 utilizando o ISE na versão 7.1, o Plasma funcionava perfeitamente. Ao efetuar a síntese com a versão 8.2, o core de execução do Plasma parecia funcionar corretamente, porém o acesso à memória não funcionava corretamente.

7.3 Trabalhos Futuros

Como resultado, observou-se que é possível no futuro a criação de um sistema que consista em um barramento com vários processadores Plasma trabalhando em conjunto.

Também como sugestão para futuros projetos, a remoção da UART e talvez da memória interna do Plasma, fazendo com que este requeira ainda menos recursos da FPGA, e que mais processadores possam ser sintetizados em um mesmo chip.

8 Bibliografia

DANGUI, Sandro. *Modelagem e simulação de barramentos com SystemC*. 2006. 121f. Dissertação (Mestrado em Ciência da Computação) - Universidade Estadual de Campinas.

GIORGINI, André Linhares. *Implementação de um Controlador PID Digital para Robótica baseado em Computação Reconfigurável*. 2001. 108 f. Dissertação (Mestrado em Ciências de Computação e Matemática Computacional) - Universidade de São Paulo.

IBM (International Business Machines Corporation). *On-Chip Peripheral Bus: Architecture Specifications*. Versão 2.1. Carolina do Norte, 2001.

Intel. *Receita da Intel no segundo trimestre é de US$8 bilhões*. Disponível em: < http://www.intel.com/portugues/pressroom/releases/2006/0721a.htm?cid=rss-120769-c1-135463>. Acesso em: 03 dez 2006.

IT Web. *Samsung anuncia investimento de US$ 1,9 bi em chips*. Disponível em: <http://www.itweb.com.br/>. Acesso em: 22 jan. 2007.

NAVABI, Zainalabedin. *VHDL : Analysis and modeling of digital systems*. 2° Edição. McGraw Hill: Nova Iorque, 1998.

OLIVEIRA, Tadeu F. Desenvolvimento de aplicações para sistemas embutidos: um estudo da plataforma J2ME. 2006. 71 f. Dissertação (Bacharelado em Ciências da Computação) - Universidade Tiradentes.

PATTERSON, David; HENNESSY, John. *Computer Organization and Design: The Hardware/Software Interface*. 2° edição. Califórnia: Morgan Kaufmann, 1997.

POLPETA, Fauze V. *Uma Estratégia para a Geração de Sistemas Embutidos baseada na Metodologia Projeto de Sistemas Orientados à Aplicação*. 2006. 115 f. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Santa Catarina.

PRICE, Charles. *MIPS IV Instruction Set*. 3.2° Revisão. Setembro, 1995.

RHOADS, Steve. *Plasma CPU Core*. Disponível em: <http://www.open-cores.org>. Acesso em: 30 jan. 2007.

WAGNER, Flávio R. *Arquitetura e Organização de Processadores*. UFRGS: Programa de Pós-Graduação em Computação, 2006.

WOLF, Wayne. *Computers as Components*. Nova Iorque: McGraw-Hill, 2001.

YAGHMOUR, Karim. *Building Embedded Linux Systems*. Califórnia: O'Reilly, 2003