Uso do protocolo AMBA na metodologia VeriSC para melhorar o processo de verificação funcional de sistemas embarcados

Ricardo Augusto Pereira Franco – ricardofranco3@gmail.com
Orientadora: Dra. Karina Rocha Gomes da Silva – karinarg@eee.ufg.br

Escola de Engenharia Elétrica e de Computação

Palavras-Chave: AMBA AXI; barramento; protocolo de comunicação AMBA; verificação funcional; VeriSC; SystemC.

Revisado pelo orientador – Orientando: Ricardo A. P. Franco; Orientadora: Dra. Karina R. G. da Silva

I. INTRODUÇÃO

A tecnologia computacional está evoluindo, desde seus primórdios até os dias atuais, rapidamente. Segundo a Lei de Moore, enunciada na década de 70, “a partir deste momento a potência dos processadores dobraria a cada 18 meses”. O desenvolvimento é perceptível tanto em computadores pessoais (desktop), quanto para servidores e computadores embutidos.

Os computadores de uso doméstico são os principais influenciadores no desenvolvimento de novas tecnologias e seu custo [03]. Os componentes básicos de um computador comum (desktop) podem ser citados da seguinte forma: processador, memória principal, dispositivos de entrada e/ou saída (E/S) e estrutura de barramento. O microprocessador é conhecido como CPU ou UCP (Unidade Central de Processamento), onde se localizam as implementações de operações aritméticas, lógicas, de desvio e de transferência de dados. A memória principal (memória Random Access Memory - RAM) é a memória responsável pela leitura e a escrita de dados. Os dispositivos de entrada e saída são considerados todos aqueles dispositivos que possibilitam a entrada e/ou saída de dados. Os dispositivos de entrada codificam as informações que entram na forma de dados, para que possam ser processados pela CPU do computador; e os dispositivos de saída decodificam os dados em informações que podem ser entendidas pelo usuário. O barramento é o responsável pela comunicação de todos estes dispositivos que formam o computador, ele interliga a CPU, memória principal e os dispositivos de E/S.

O barramento (Fig.1) é um conjunto de linhas de comunicação entre dispositivos de um computador, por exemplo, processador e memória (fios que conectam as partes de um computador).

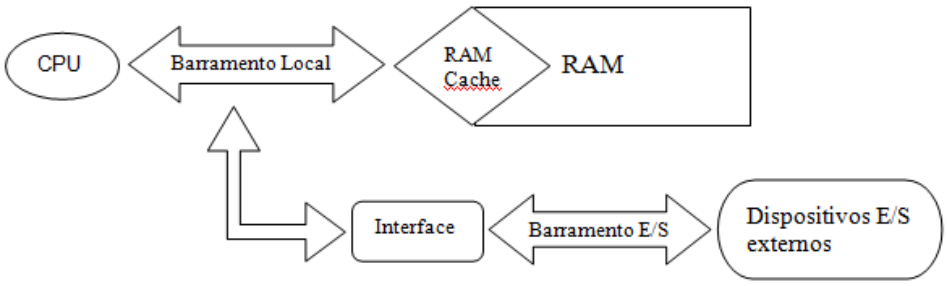

Fig.1. Exemplificação de um barramento

A composição do barramento é feita por alguns tipos de linhas: controle, dados e endereços – além de algumas linhas que distribuem a energia aos componentes. O barramento de controle é responsável pelo acesso e a utilização das linhas de dados e endereços através dos componentes do sistema. Ele pode ser destinado a enviar informações de temporizadores (clock) e/ou emitir comandos. O barramento de dados realiza transferências de dados entre módulos do sistema, onde são transferidos os bits. Os endereços solicitados para se aplicar a ação sobre os dados são carregados no barramento de endereços, os dados solicitados são encaminhados para o barramento de dados onde será executada a ação requerida, ou seja, o barramento de endereços é quem contém as informações dos locais para onde estes dados devem ser guiados ou entregues.

Para o funcionamento do barramento, existem dispositivos mestres, escravos, decodificadores e árbitros. Pode existir um ou mais dispositivos mestres, e são responsáveis por enviar as requisições e possuem o controle sobre o barramento. Os escravos recebem as requisições do barramento, as executam e retornam as respostas ao barramento para serem devolvidas ao dispositivo mestre. Quando existe mais de um mestre, é necessário um árbitro, para que ocorra um escalonamento de prioridade nas requisições e, assim, não haja perda de sinais e/ou corrompimento das requisições ou respostas.

Na linguagem de hardware, os barramentos possuem um protocolo de comunicação, ou seja, é um padrão que controla e possibilita uma comunicação, conexão ou transferência de dados entre sistemas computacionais. Os protocolos de comunicação variam em relação à necessidade do projeto, sendo que o protocolo usado nesse projeto foi o Protocolo Advanced Microcontroller Bus Architeture (AMBA), e a especificação deste é o Advanced eXtensible Interface (AXI).

O principal intuito deste trabalho é a inserção do protocolo de comunicação no ambiente de verificação da metodologia VeriSC, aplicado a um projeto de hardware, o DPCM (differential pulse-code modulation).

Para a realização desse trabalho foi usada a metodologia VeriSC. A metodologia VeriSC [05] é usada em todo o desenvolvimento do projeto e possui o foco na verificação funcional do dispositivo. A verificação funcional é necessária para verificar

se o dispositivo sendo projetado (Design Under Verification –DUV) está de acordo com a especificação.

A metodologia VeriSC é uma forma de verificação que propõe a implementação do *testbench* (ambiente de testes) antes mesmo do próprio dispositivo, pois quando for iniciar a implementação do dispositivo, o ambiente de testes já não contém falhas.

> "A metodologia VeriSC é composta de um novo fluxo de verificação, que não se inicia pela implementação do DUV. Nesse fluxo, a implementação do testbench e do Modelo de Referência antecedem a implementação do DUV. Para permitir que o testbench seja implementado antes do DUV, a metodologia implementa um mecanismo para simular a presença do DUV com os próprios elementos do testbench, sem a necessidade da geração de código adicional que não é reutilizado depois. Com esse fluxo, todas as partes do testbench podem estar prontas antes do início do desenvolvimento do DUV."

O testbench é o ambiente que contém o mecanismo que irá gerar os estímulos para testar o dispositivo e o modelo de referência possui a mesma funcionalidade do DUV e pode ser escrito em qualquer linguagem de programação.

II. OBJETIVOS

O objetivo do projeto é a implementação de modelos – em sua fase de composição de módulos – de um IP-core utilizando o protocolo de comunicação AMBA AXI.

Para atingir esse objetivo, realizou-se um estudo bibliográfico sobre as ferramentas necessárias e as seguintes etapas foram cumpridas:

- Estudar todos os perfis do protocolo AMBA.
- Estudar a metodologia VeriSC.
- Definir qual o perfil do protocolo AMBA é mais adequado para a metodologia VeriSC.
- Integrar o protocolo AMBA na metodologia VeriSC.

III. METODOLOGIA

A. Protocolo AMBA

O Protocolo AMBA é um padrão de comunicação on-chip para projetar microcontroladores embarcados de alto desempenho e foi desenvolvido pela ARM.

Existem quatro tipos específicos de barramentos que podem ser utilizados na especificação AMBA, a diferença entre eles se baseia na aplicação que se deseja para o barramento, os tipos são:

- Advanced High-performance Bus (AHB);
- Advanced System Bus (ASB);
- Advanced Peripheral Bus (APB);
- Advanced Extensible Interface (AXI).

Os protocolos AMBA AHB, ASB e AXI, são utilizados para conectar processadores, memórias externas e dispositivos DMA (Direct Memory Acess), esses são os barramentos principais, ou seja, um sistema baseado no protocolo AMBA é tipicamente projetado com um barramento backbone. Para a comunicação de periféricos, utiliza-se o barramento APB. No protocolo AMBA, o barramento APB se conecta a um barramento backbone principal permitindo assim a comunicação dos dispositivos de E/S para que solicitem ou executem uma ação.

A principal diferença entre os barramentos é o nível de desempenho desejado. Para dispositivos de E/S, o barramento APB é menos complexo, pois ele é otimizado para o baixo consumo de energia, além de possuir uma interface de baixa complexidade. O AHB é um barramento utilizado quando se exige um alto desempenho com altas frequências de clock. O ASB é utilizado para módulos de alto desempenho, e quando os requisitos de desempenho do AHB não são necessários. O protocolo AXI é utilizado para altas performances, designs de sistemas de alta freqüência e uma série de características de interconexão de alta velocidade. Um exemplo da forma de composição em um barramento utilizando o protocolo AMBA é demonstrado na Fig. 2 (figura retirada de AMBA Specification Rev 2.0).

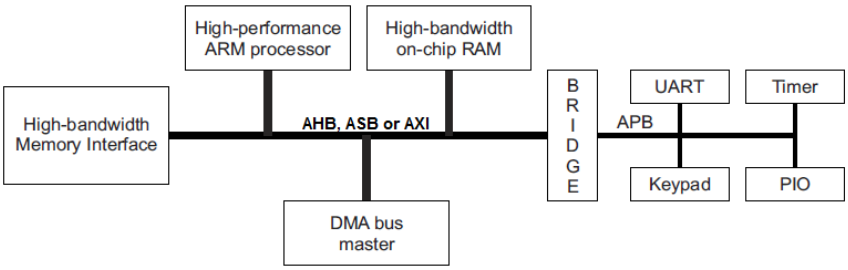

Fig. 2. Representação de um sistema AMBA típico

B. Funcionalidades do AMBA Advanced eXtensible Interface

Como prescrito na especificação AMBA, o barramento AXI possui funcionalidades mais simples, dentro dos outros protocolos AMBA, para o estudo realizado nesse trabalho e foi o perfil escolhido para a implementação do estudo de caso.

Este protocolo é da terceira geração de interfaces AMBA, e possui as seguintes características:

- Possui fase de dados, comandos de endereço ou controle separados;
- Realiza transações em burst após a emissão do endereço;
- Aconselhável para projetos que necessitem de uma alta largura de banda e baixa latência;
- Possui flexibilidade na implementação de arquiteturas intercomunicáveis;
- É compatível a comunicação com a interface dos protocolos AMBA AHB e APB;
- Fácil adição de estágios de registros;
- Possui a capacidade de emitir vários endereços pendentes;
- Possui canais de endereçamento, dados de leitura, resposta da leitura, dados de escrita e resposta da escrita.

O protocolo AXI foi criado com o objetivo de, além de ter alto desempenho e freqüência, utilizar sua alta freqüência sem utilizar pontes complexas, atender aos requisitos de interfaces de um amplo conjunto de componentes, realizar transações em rajadas com apenas o endereço inicial emitido, separar a leitura e escrita dos canais de dados para possibilitar o low-cost (pouco consumo) da DMA e realizar transações

alternadas.

Toda a transação possui informações de controle e endereço propagadas no canal de endereço que descreve o tipo de dado a ser transferido. O dado é transferido entre o mestre e o escravo usando o canal de escrita de dados, caso seja para o escravo, ou canal de leitura, caso seja para o mestre.

C. Arquitetura do AMBA AXI

O protocolo AXI se baseia em operações em rajada (*burst*). Todas as transações possuem um endereço e informações de controle, ambas enviadas pelo canal de endereço, que fornecerá a natureza do dado a ser transferido. O dado é transferido entre o mestre e o escravo utilizando o canal de leitura de dados para o mestre se for operação de leitura ou, se utiliza o canal de escrita de dados para o escravo para operações de escrita de dados. A Fig. 3 mostra os canais de leitura entre o mestre e o escravo (Fig.3 e Fig.4 retirada de AMBA AXI Protocol v1.0 Specification).

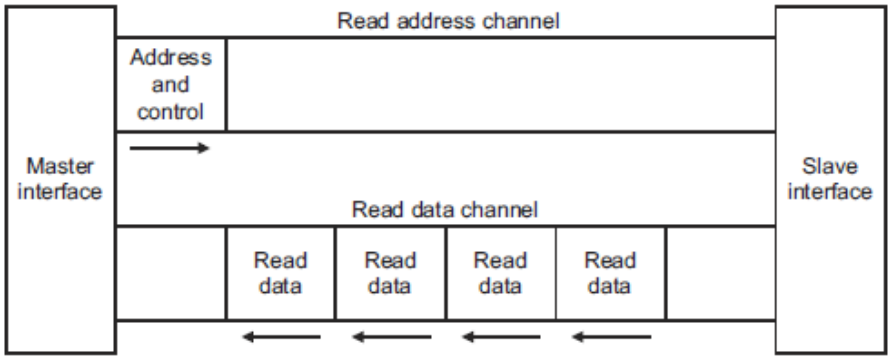

Fig.3. Arquitetura de leitura do Protocolo AMBA AXI

Em operações de escrita de dados, na qual todos os dados vão do mestre para o escravo, o protocolo AXI tem um canal de resposta adicional que notifica o mestre que a operação de escrita foi concluída com sucesso, pode-se notar este canal na próxima figura (Fig.4).

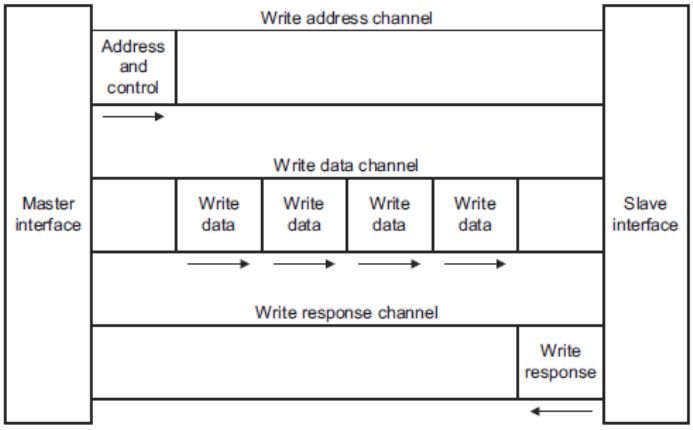

Fig. 4. Arquitetura de escrita do Protocolo AMBA AXI

Esse protocolo também suporta transações múltiplas pendentes, realização de transações fora de ordem e informações a respeito do endereço a ser emitido antes da transferência de dados reais.

D. Verificação funcional VeriSC

Na verificação funcional VeriSC, para se criar um ambiente é necessário que estabeleça as devidas especificações dos requerimentos do dispositivo. A criação e a modelagem deste ambiente irão definir os testes a serem aplicados.

"Verificação funcional é um processo usado para demonstrar que o objetivo do projeto é preservado em sua implementação." [5].

O testbench é o ambiente na qual o DUV será inserido, o ambiente irá gerar estímulos específicos, primeiramente, para o modelo de referência e produzindo, assim, respostas que serão comparadas com as respostas ideais. Esses estímulos serão enviados para o modelo de referência e para o DUV, e como o modelo de referência é criado, testado e garantido que não há falhas, as respostas deverão ser as mesmas, se forem diferentes o DUV não está implementado corretamente.

Os estímulos são enviados e comparados usando o dispositivo FIFO (*first-in, first-out)* para que não exista nenhum problema para comparar os resultados finais, portanto, as FIFO´s são responsáveis pelo sincronismo e seqüenciamento dos dados que entram e saem delas.

Esse ambiente deverá, preferencialmente, ser implementado em um nível alto de abstração (nível de transação), mas deve possuir um padrão para realizar a conversão do nível de transação para o nível de sinais antes de enviar os dados para o DUV, pois este "trabalha" apenas no nível de sinais.

A finalização da construção do *testbench* possibilita a implementação do DUV e, em seguida, aplicá-lo aos testes para a verificação de erros.

Pode definir a metodologia VeriSC em alguns passos (Fig. 5, retirada de [5]):

- Levantamento e especificação dos requisitos do DUV;
- Implementação dos mecanismos do *testbench* juntamente com a cobertura requerida;
- Desenvolvimento do modelo de referência;
- Geração de estímulos (testes) em todo o *testbench* (e modelo de referência);
- Implementação do DUV;
- Simulação de todos os módulos – *testbench*, modelo de referência e DUV;
- Análise das respostas e da cobertura através de itens de cobertura e logs de simulação;
- Iteração até a satisfação de todos os requisitos levantados e especificados.

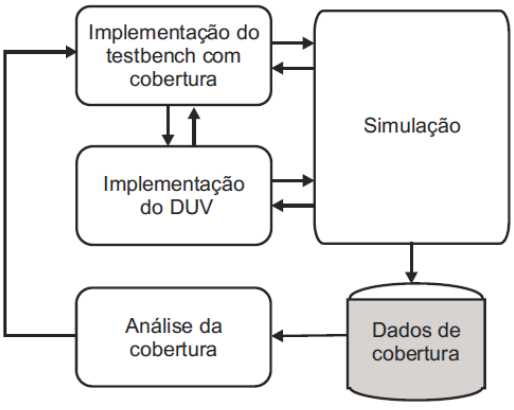

Fig. 5. Fluxo de verificação

O testbench é composto pelos seguintes módulos: Source, TDriver(s), TMonitor(es), Modelo de Referência e Checker. O Source é responsável pela criação de estímulos para a simulação, são criados em nível de transações e são escolhidos para satisfazer aos critérios da cobertura. O Checker é o responsável por comparar as respostas fornecidas pelo modelo de referência e do DUV – através do TMonitor (será explicado a frente). O TDriver é responsável por converter as transações, emitidas pelo Source, a nível de sinais e enviá-los para o DUV, existe um TDriver para cada interface de entrada do DUV. O TMonitor converte todos os sinais para o nível de transações e em seguida submetê-las para o Checker, cada interface de saída do DUV possui um TMonitor.

O DUV é o projeto a ser desenvolvido, implementado em RTL (nível de sinais) e que através da conversão do TDriver e TMonitor recebe e emite transações (Transaction Level Modelling). O modelo de referência deve receber os estímulos e produzir as respostas corretas, que serão comparadas com as respostas do DUV.

O protocolo de comunicação é executado nos módulos TDriver e TMonitor através do nível de sinais. O protocolo AMBA AXI é aplicado entre estes módulos e o DUV.

E. SystemC

SystemC é uma linguagem de descrição de hardware, implementada como uma biblioteca da linguagem de programação C++.

"SystemC acrescenta conceitos importantes do C++, tais como simultaneidade (múltiplos processos executando simultaneamente), cronometragem de eventos e tipos de dados. SystemC adiciona uma biblioteca de classes C++ para expandir as capacidades do C++". [4]

A biblioteca SystemC fornece conceitos necessários para um projetista de hardware como portas, sinais e módulos, e também fornece conceitos de processos e modificações de sinais pelas bordas (positiva ou negativa) do clock. A linguagem fornece uma simulação do kernel, permite ao projetista simular o executável desenvolvido por ele. A simulação semântica é definida pelo SystemC.

IV. RESULTADOS

Os resultados obtidos durante o período do desenvolvimento de dispositivos foi a implementação de um módulo de IP-core com o protocolo de comunicação AMBA AXI realizando a comunicação entre o módulo e todo o ambiente externo de testes, que simula partes de um computador.

O dispositivo implementado foi um DPCM (Differential Pulse-Code Modulation), na linguagem SystemC. As variáveis referentes ao protocolo AMBA AXI realizaram a parte de comunicação do DUV com o ambiente de testes, permitindo que fosse implementada a comunicação dos dados.

Um DPCM é um procedimento para converter um dado de sinal analógico para um dado de sinal digital. O sinal analógico é amostrado e então é calculada a diferença entre o valor real e o valor predito (o valor predito é o valor anteriormente obtido na subtração, que deve ser guardado), sendo que o valor inicialmente predito é igual a zero. Os dados do DPCM desenvolvido vão de -60 até 60. Quando os dados excedem esse valor, eles devem ser truncados para os valores máximos definidos, ou seja, -80 ou 80.

Foi desenvolvido um módulo para realizar os cálculos do DPCM. Neste módulo, foram adicionados alguns sinais de informações do mecanismo de handshake, os sinais de Ready e Valid. Estes sinais também são responsáveis por ser parte do protocolo de comunicação, na qual os sinais Valid são usados quando um dado válido ou uma informação de controle está disponível no canal. Os sinais Ready são usados para mostrar quando os dados estão prontos para serem capturados. Em ambos os canais, tanto de leitura quanto de escrita de dados, foram incluídos estes dois sinais para auxiliarem as transações. A Fig. 6 representa a definição dos sinais criados para realizar a comunicação do módulo.

```
21  sc_in<bool> sample_in_valid; //Definição dos sinais do barramento
22  sc_out<bool> sample_in_ready;
23  sc_out<bool> sample_out_valid;
24  sc_in<bool> sample_out_ready; //Fim da definição
```

Fig.6. Definição dos sinais de comunicação

O módulo foi desenvolvido como uma máquina de estados na qual permite que ele sempre esteja em um estado bem definido. Os quatro possíveis estados para o

módulo são: Parado (*Idle*), Recebido, Validado, Terminado. O estado Parado (Fig.7) é um loop infinito que é executado quando o dispositivo não está sendo utilizado, no momento em que o DUV receber sinal de dado válido é encaminhado para o estado Recebido que inicia a execução do módulo do DPCM.

```
37      case IDLE: //caso parado
38          sample_in_ready=1; //quando o DUV receber sinal de dado válido
39          estado = RECEBIDO; //envia para estado Recebido
40
41          break;
```

Fig.7. Implementação do estado 'Parado'

Após o estado Recebido o dado é enviado para sua validação (estado Validado) e então, se o dado está nas condições impostas (os dados desenvolvidos entre os valores de -60 até 60) é passado para o último estado (Terminado) que termina a execução do módulo e encaminha para o estado inicial (Parado).

A comunicação é necessária também entre cada estado e não apenas entre o DUV e dispositivos externos, pois cada um define um determinado passo para que a execução do DUV ocorra corretamente.

V. DISCUSSÃO

A verificação funcional VeriSC conseguiu cobrir todos os cenários impostos pela especificação de verificação do hardware, demonstrando, então, que não havia falhas na cobertura.

Foram aplicadas os sinais sample_in_ready, sample_in_valid, sample_out_ready e sample_out_valid responsáveis pela consistência de dados enviados e recebidos pelo DUV. Os sinais 'ready' são responsáveis por decidir quando um dado está pronto para ser enviado, e os sinais variados de 'valid' decidem quando um dado que está pronto para ser recebido é válido.

Após todos os resultados satisfatórios dos testes aplicados no dispositivo, o resultado final foi um código que define um DPCM cuja funcionalidade pode ser implementado para um chip.

VI. CONCLUSÃO

Dispositivos SoC necessitam de um protocolo de comunicação padrão, pois possibilita a interligação entre dispositivos de empresas diferentes. Uma padronização é necessária, pois não apenas auxilia o consumidor, mas também não prejudica os desenvolvedores.

Todos os objetivos foram atingidos e os principais a serem citados são: o estudo sobre como ocorre à comunicação de dispositivos e sobre os tipos de protocolos AMBA existentes, a escolha de um Protocolo que alcançasse os objetivos sem perda de desempenho e a implementação do código de um DPCM.

A linguagem SystemC permitiu uma facilidade na implementação, porque é semelhante às linguagens C e C++ e, na qual um prévio conhecimento permite uma facilidade na utilização da mesma.

VII. REFERÊNCIAS BIBLIOGRÁFICAS

[1] AMBA Specification (Rev 2.0), ed. A. The AMBA protocol is an open standard. 13 maio 1999. Disponível em: http://www.arm.com/products/system-ip/amba/amba-open-specifications.php . Acesso em: 10 de ago. 2010.

[2] AMBA AXI Protocol (Ver 1.0), ed. B. The AMBA protocol is an open standard. 19 mar. 2004. Disponível em: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ihi0022c/index.html . Acesso em: 07 de out. 2010.

[3] HENNESSY J. L. e PETTERSON D. A., tradução de Vandenberg D. de Souza *Arquitetura de computadores: uma abordagem quantitativa*, 4.ed., Campus, 2003.

[4] BHASKER J. *A SystemC Primer*, 2.ed., Star Galaxy Publishing, 2002.

[5] SILVA, K. R. G. da, et al. A methodology aimed at better integration of functional verification and rtl design. *Design Automation for Embedded Systems*, 10(4):285–298, 2007.

[6] GRÖTKER T. et al. *System Design with SystemC*, New York: Kluwer Academic Publishers, 2002. Disponível em: http://site.ebrary.com/lib/ankos/ .Acesso em: 03 ago. 2010.

[7] WILE, B. et al. *Comprehensive Functional Verification: The Complete Industry Cycle.* San Francisco: Elsevier, 2005.

[8] DANGUI, Sandro C. *Modelagem e simulação de barramentos com SystemC*. Campinas: Universidade Estadual de Campinas, tese de mestrado. 2006. Disponível em: http://www.bibliotecadigital.unicamp.br/document/?code=vtls000401086&opt=1 .Acesso em: 14 set. 2010

[9] QUEIROZ, Daniel C. *Implementação do barramento on-chip AMBA baseada em computação reconfigurável.*São Carlos: ICMC/USP, tese de mestrado. 2005. Disponível em:http://www.teses.usp.br/teses/disponiveis/55/55134/tde-06042005-160026/pt-br.php .Acesso em: de set. 2010.