Plano de Verificação Funcional

MUSA

Fazemos Qualquer Negócio Inc.

Compilação 1.0

Histórico de Revisões

Data	Descrição	Autor(es)
23/10/2014	Criação do documento.	Terseu Hunter
08/12/2014	Ajustes iniciais	jadsonfirmo
12/12/2014	Inserção de dados e alterações	jadsonfirmo & lucasmorais
15/12/2014	Visão Geral do DUT e ajustes	jadsonfirmo
16/12/2014	Revisão parcial e ortográfica	gordinh

CONTENTS

1. Introdução

1.1. Propósito do Documento

Este documento tem como objetivo detalhar o plano de verificação para o projeto Core-MUSA. Este detalhamento passa pelo ambiente de verificação utilizado para realizar a verificação do processador, pela lista de funcionalidades, lista de testes, *assertions*, recursos e por fim, pelo cronogograma.

1.2. Stakeholders

Nome	Papéis/Responsabilidades
Jadson, Kelvin e Odívio	Verificação e Análise
Filipe, Lucas, Matheus e Wagner	Testes
Diego e Victor	Implementação

1.3. Siglas e Abreviações

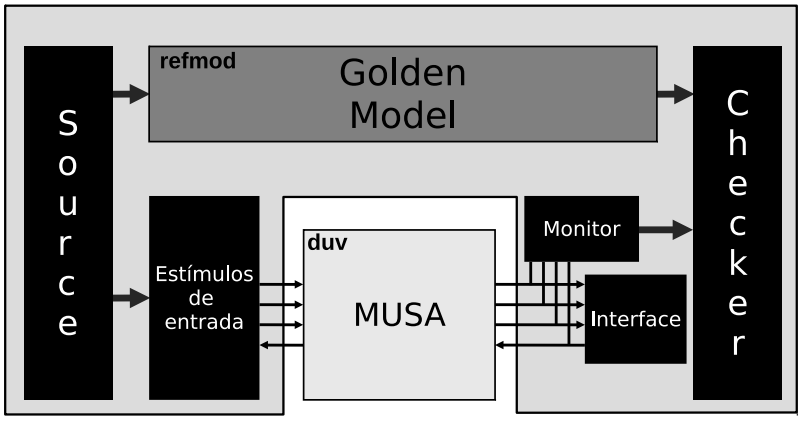
Sigla	Descrição
DUT	Design Under Test
IF	Interface
DUV	Design under verification

2. Visão Geral do DUT

- Cada *testbench* deve instanciar o projeto que é esperado para testar. Este projeto é normalmente referido como o "projeto em teste" (DUT).
- Antes do DUT ser instanciado, cada uma de suas entradas e saídas devem ser declaradas no *testbench*.
- As entradas para o DUT são declarados como "reg" e saídas são declarados como *wires* (fios). Nota-se que as saídas do DUT são entradas para o *testbench*, lembrando que qualquer entrada deve ser um *wire* (fio).
- As entradas para DUT são os impulsos gerados no *testbench*. Os estímulos são, normalmente, gerados em um bloco inicial sempre processual ou no *testbench*.
- Após a implementação do DUT, temos a simulação do DUV e o *testbench*.
- Espera-se que o DUT seja capaz de facilitar os testes a serem realizados a cerca do projeto, de modo que através da captação de dados e dos *logs* de simulação possa haver análise dos resultados bem como simplificar a refatoração, caso seja necessária.

3. Ambiente de Verificação

A metolologia de verificação adotada pelo projeto é baseada em *testbench*, compondo parte das análises por meio de verificação baseada em *waveform*. Situações especiais serão verificadas apartir de verificações baseadas em *assertions*. A interface do DUT será responsável por coletar os dados do MUSA e enviá-los para o *monitor*, no qual estarão declarados todos os *assertions*. A Figura abaixo apresenta um modelo conceitual do ambiente de verificação.



3.1. Design Under Test Interface

O DUT IF promove a interface entre o *monitor* e o DUT. Esta interface é responsável por controlar as informações trocadas entre o ambiente de verificação e o DUT. Dessa forma, ela deve conter instâncias de todos os sinais do DUT a serem utilizados ao longo do processo de verificação.

A interface do DUT possui tabém a implementação dos *assertions*. Estas estruturas têm como objetivo garantir que o comportamento dos sinais internos do DUT estão sendo produzidos e manipulados de maneira correta. Esta interface é instanciada na entidade *top level* do ambiente de verificiação e seus sinais são conectados aos sinais provenientes do DUT.

3.2. Monitor e Checker

O *monitor* é reponsável por observar o comportamento do DUT e coletar as suas saídas, de modo a verificar se as instruções estão funcionando da maneira desejada. O *monitor* observa o comportamento dos sinais de controle e, quando necessário, captura os dados armazenados na memória de instruções e no banco de registradores.

O *checker* é responsável por executar o modelo de referência com o mesmo programa usado pelo DUT e comparar os dados armazenados na memória de dados e no banco de registradores. Se qualquer mal funcionamento for identificado, o *checker* deve reportar uma mensagem de erro.

Quando a execução do programa chega ao fim, o *monitor* deve invocar o *checker*. O *monitor* identifica o final da execução do programa a partir de uma sequência de seis instruções NOP consecutivas.

O teste que será executado no modelo de referência deve ser definido no arquivo `sim/tb/defines.sv`. Para executar o teste no DUT, o procedimento deve ser realizado no arquivo de memória de instruções, `rtl/instruction_memory.v`, a partir da alteração do caminho especificado na função `read_memh`.

3.3. Modelo de Referência

Tendo em vista garantir que o processador executará as instruções corretamente, foi desenvolvido um modelo de referência, capaz de simular o comportamento do processador MUSA. Este modelo é capaz de executar todas as instruções suportadas pelo MUSA. O arquivo do modelo de referência está localizado no diretório `sim/model/`.

3.4. Especificações de Projeto do Ambiente de Verificação

Componente	Descrição
Nome do Documento	Plano de Verificação do MUSA
Versão e data do documento	Versão 1.0, 23 de outubro de 2014
Autor(es) / Proprietário(s)	Terseu Hunter
Metodologia de Verificação	Top-Down
Métodos de Verificação	Simulation and Formal Verification
Aplicação	ModelSim ALTERA Edition
Linguagens	System Verilog
Ambiente de verificação	Custom testbench
Arquivos de teste	No diretório: sim/tests
Tecnologias	FPGA Cyclone 3 Development Board

4. Lista de Funcionalidades

Feature Número	*Feature* Descrição	Prioridade
MUSA_F1	Sinais são ativados baseados na instrução.	10
MUSA_F2	Comunicação com a memória de instrução.	9
MUSA_F3	Operações de leitura e escrita para a Memória de Dados.	9
MUSA_F4	Operações de leitura e escrita para os arquivos de registradores.	10
MUSA_F5	Todos os protocolos de interface devem funcionar corretamente.	9

5. Lista de Testes

Número do Teste	Descrição	Método	Nível	Funcionalidade Verificadas	Prioridade	Proprietário	Situação
MUSA_T1	Execução de todas as instruções da categoria aritmética.	Sim	Unit	MUSA_F1, MUSA_F4	5	Filipe e Diego	47%
MUSA_T2	Execução de todas as instruções de transferência de dados.	Sim	Unit	MUSA_F1, MUSA_F3, MUSA_F4	5	Victor	87%
MUSA_T3	Execução de todas as instruções da categoria lógica.	Sim	Unit	MUSA_F1, MUSA_F4	5	Filipe e Diego	47%
MUSA_T4	Execução de todas as instruções da categoria salto condicional.	Sim	Unit	MUSA_F1, MUSA_F4	5	Matheus e Wagner	87%
MUSA_T5	Execução de todas as instruções da categoria salto incondicional.	Sim	Unit	MUSA_F1, MUSA_F2	5	Matheus e Wagner	87%
MUSA_T6	Acesso à memória de dados	Assertion	Unit	MUSA_F3	7	Lucas e Jadson	100%
MUSA_T7	Acesso à memória de instruções	Assertion	Unit	MUSA_F4	9	Lucas e Jadson	100%
MUSA_T8	Execução de programas completos sob a arquitetura.	Sim	Unit	MUSA_F3, MUSA_F4	8	Odívio e Kelvin	0%

continua na próxima página

continuação da página anterior

Número do Teste	Descrição	Método	Nível	Funcionalidade Verificadas		Prioridade	Proprietário	Situação
MUSA_T9	Teste de todos os protocolos de interface.	Assertion	Unit	MUSA_F5		8	Filipe e Victor	0%

6. Assertions

Número	Critério	Status
MUSA_A1	Assertion para a busca correta das instrução.	Em andamento
MUSA_A2	Assertion para verificar a operação de decodificação	Em andamento
MUSA_A3	Assertion para verificar a operação do bloco de execução.	Em andamento
MUSA_A4	Assertion para leitura da memória de dados e write back.	Em andamento
MUSA_A5	Assertion para branches e instruções de salto.	Em andamento
MUSA_A6	Assertion para verificar os protocolos de interface.	Em andamento

7. RecursosRequirements

Recursos	Quantidade	Descrição	Início	Duração
Recursos de Engenharia				
Engenheiro de Verificação	03	Filipe, Jadson e Lucas	13/11	N.A.
Recursos Computacionais				
Computador	Dell	Core I7	N.A.	N.A.
Recursos de Software				
ALTERA Quartus	1	WEB Edition	13/11	36 dias
ALTERA ModelSIM	1	ALTERA WEB Edition	17/11	32 dias