Centro de Engenharia Elétrica e Informática
Departamento de Engenharia Elétrica

José Iuri Barbosa de Brito

Aurora - Ferramenta de geração integrada de Fluxo de Referência

Campina Grande
2020

José Iuri Barbosa de Brito

Aurora - Ferramenta de geração integrada de Fluxo de Referência

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Eletrônica

Orientador:
Marcos Ricardo de Alcântara Morais, D. Sc.

Campina Grande
2020

José Iuri Barbosa de Brito

Aurora - Ferramenta de geração integrada de Fluxo de Referência

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Aprovado em / /

Marcos Ricardo de Alcântara Morais, D. Sc.
UFCG

Antônio Marcus Nogueira Lima, D. Sc
Professor Convidado
UFCG

Campina Grande

2020

Este trabalho é dedicado a todos que me acompanharam até aqui, por toda a ajuda e colaboração, e a meus pais Antônio Macêdo de Brito (in memoriam) e Glaucijane Ferreira Barbosa sem o apoio deles jamais conseguiria qualquer coisa.

AGRADECIMENTOS

”Estamos todos num mesmo barco, em mar tempestuoso, e devemos uns aos outros uma terrível lealdade.” Com essa frase de Chesterton gostaria de iniciar essa sessão de agradecimentos. Primeiramente agradeço a Deus pelo dom da vida e meu padroeiro S. José cujo exemplo de trabalho muito me inspirou. Em seguida gostaria de agradecer aos meus pais e familiares que sempre me apoiaram em minhas empreitadas e sonhos. Também agradeço a minha namorada Laryssa, pois sem seu apoio e sua cobrança, em especial essa última parte, provavelmente não existiria esse texto ou esse trabalho.

Aos meus colegas de turma cuja coincidência nos uniu na turma do 2015.1. Sem esses colegas com certeza a jornada pela graduação não seria a mesma e certamente seria bem mais complicada.

Aos meus companheiros do Programa de Educação Tutorial (PET) agradeço imensamente na pessoa do professor Wamberto. Os anos que passei junto dessas incríveis pessoas com certeza moldaram minha atuação profissional e meus princípios acadêmicos.

Deixo meus agradecimentos também entusiastas, assim como eu, do capítulo estudantil RAS UFCG, que tive o prazer de estar entre os fundadores. Sempre torço pelo sucesso merecido de todos que compõe esse excepcional grupo e também o laboratório eRobótica, em especial os colegas Yuri Loya, Davi, Felipe, Arthur e Emanoel, este que sempre nos socorreu em nossas maluquices.

Gostaria de agradecer a todos os professores do Departamento de Engenharia Elétrica, em especial aos professores Antônio Marcus, Marcos Morais, Gutemberg Júnior e também do Departamento de Sistemas e Computação, professor Elmar Melcher, cujo contato se deu de forma mais próxima desenvolvendo muitas atividades juntos.

Aos membros do Laboratório de Excelência em Microeletrônica do Nordeste (XMEN), muito obrigado pelo companheirismo e competência nesse mundo tão minúsculo quanto os transistores que compõe nossos *chips*. Em especial gostaria de agradecer aos Pinguins da Verificação, grupo mais próximo a mim do laboratório, trabalhar ao lado de vocês é sempre uma honra.

Ao longo do desenvolvimento desse trabalho muitas das comunidades *online* que faço parte me fizeram companhia nas manhãs e noites. Por isso gostaria de agradecer ao Gustavo Gomes (Flock) e ao Victor Vasconcelos (ViktorKav) e toda a comunidade de Runeterra da Twitch.

Também gostaria de agradecer a todos aqueles que não foram citados aqui, mas que contribuíram para este trabalho.

"Fairy tales do not tell children the dragons exist. Children already know that dragons exist. Fairy tales tell children the dragons can be killed."

(Gilbert Keith Chesterton)

RESUMO

A diversidade de problemas e complexidades que envolvem o desenvolvimento de projetos de *hardware* motivam cada vez mais pesquisas e metodologias para superação desses desafios impostos. Dessa forma, este trabalho tem o objetivo de desenvolver uma ferramenta para a geração integrada de fluxo de referência para projetos de *hardware*. A ferramenta deve ser uma habilitadora da adoção da metodologia de *test driven development* por meio da utilização de propriedades formais, assim como estar em consonância com os padrões já estabelecidos na indústria, IEEE 1800 e IEEE 1800.2. A ferramenta aqui desenvolvida foi feita utilizando a linguagem Python com a prática de orientação a objetos. Além disso foi feito um desenvolvimento de um caso de uso em um módulo aritmético com uma interface AMBA *Advanced Peripheal Bus*, em que observou-se a aplicação da metodologia e o fluxo gerado pela ferramenta, comprovando a eficácia da mesma para os objetivos propostos.

Palavras-chave: Desenvolvimento de *hardware*, Fluxo de projeto, *Hardware Design*, *Test Driven Development*, Verificação.

ABSTRACT

The diversity of problems and complexities that involve the development of hardware projects increasingly motivate research and methodologies to overcome challenges imposed challenges. Thus, this work aims to develop a tool for the integrated generation of reference flow for hardware projects. The tool should be an enabler for the adoption of the test-driven development methodology through the use of formal properties, as well as being in line with the standards already adopted in the industry, IEEE 1800 and IEEE 1800.2. The tool developed here was made using a Python language with the practice of object orientation. In addition, a use case was developed in an arithmetic module with an AMBA Advanced Perifeal Bus interface, in which the application of the methodology and the flow generated by the tool were observed, proving the effectiveness of the tool for the proposed objectives.

Keywords: Hardware Design, Hardware Development, Project flow, Test Driven Development, Verification.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

UFCG	Universidade Federal de Campina Grande
DEE	Departamento de Engenharia Elétrica
VLSI	*Very large-scale integration*
HDL	*Hardware Description Language*
HVL	*Hardware Verification Language*
ASIC	*Application Specific Integrated Circuits*
TDD	*Test Driven Development*
UVM	*Universal Verification Methodology*
SVA	*SystemVerilog Assertions*

SUMÁRIO

1 INTRODUÇÃO

O crescente avanço do projeto de circuitos eletrônicos integrados e dos processos de VLSI (*Very large-scale integration*) permitiu que nós pudéssemos ser capazes de integrar bilhões de transistores em um único *chip* para realizar uma, ou várias funções. Desde a produção de microprocessadores, memórias, ou até mesmo ASICs (*Application Specific Integrated Circuits*), fez-se necessário um contínuo aprimoramento de técnicas de projeto e de processos de desenvolvimento.

Quando se trata de desenvolvimento de projeto de circuitos integrados, é comum que haja a divisão em três grandes frentes, sendo elas: *design*, verificação e implementação física (*backend*). Na frente de *design* é feita a construção da arquitetura do circuito digital e implementada sua lógica, comumente em uma linguagem de descrição de *hardware* (HDL). Na verificação é feita a checagem se a descrição do circuito digital atinge os requisitos funcionais definidos e se a correta "intenção" (*design intent*) foi atingida. Por fim, o *backend* implementa fisicamente a lógica do circuito, ou seja, convertendo o circuito lógico em um circuito físico com os transistores.

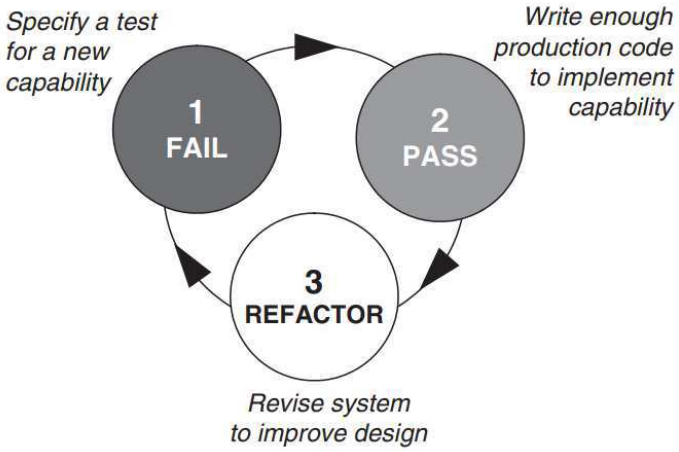
Os processos de desenvolvimento de *hardware*, principalmente no âmbito de microeletrônica, são diversos e muitos deles são aplicáveis a apenas um tipo de projeto. Nos últimos anos ocorreram diversos esforços no sentido de estabelecimento de alguns padrões para a indústria em termos de unificar metodologias e facilitar a reusabilidade, como é o exemplo do padrão IEEE 1800-2017 (IEEE..., 2018), em que a linguagem SystemVerilog passou a ser adotada tanto para *design* como para verificação e do padrão IEEE 1800.2-2017 (IEEE..., 2020), onde é padronizada a metodologia UVM (*Universal Verification Methodology*). Sempre com a criação de um novo padrão, sua adoção por parte da indústria torna-se um processo extremamente custoso, pois além de esperar que as ferramentas de EDA (*Eletronic Design Automation*) deem suporte ao novo padrão, existe o processo de formação de recursos humanos capazes de executar os projetos dentro dos requisitos exigidos pelo padrão. A concepção e a execução de um projeto de circuito integrado, envolve, então, a adoção de práticas da indústria e a integração entre as equipes para que os módulos desenvolvidos sejam confiáveis e possibilitem a correta execução da funcionalidade desejada.

Com o estabelecimento dos novos padrões de desenvolvimento de *hardware*, a existência de ferramentas de automação que possibilitem a rápida integração desses novos padrões, dentro de uma metodologia comum, evita problemas de integração e erros de compreensão da especificação do produto por parte da equipe de desenvolvimento. Dentro dessas necessidades surge a ferramenta descrita nesse trabalho, a Aurora. Com a

sua utilização é possível acelerar o processo de *deploy* do desenvolvimento de *hardware* e com a adoção das práticas sugeridas torna-se possível desenvolver circuitos mais seguros contra erros e menos suscetíveis a erros na implementação lógica.

A base de criação da ferramenta Aurora partiu da necessidade de integração entre as equipes de *design* e verificação que devem atuar em conjunto para entregar os módulos devidamente construídos e livres de erros. Portanto, além de uma ferramenta de automação do fluxo, a ferramenta acelera a adoção de práticas de desenvolvimento como o TDD (*test driven development*) tornando a adaptação mais suave para a equipe. Com a adoção da prática do TDD, que é mais comum em desenvolvimento de *software*, cada módulo desenvolvido pela equipe, ou partes dele, tornam-se também alvo de procedimentos de teste, de forma que a própria execução do teste auxilie o desenvolvedor a encontrar a correta implementação dos requisitos do *design*.

Figura 1 – Fluxo tradicional de TDD no desenvolvimento de *software*



Fonte: (ERDOGMUS; MELNIK; JEFFRIES, 2010)

A adoção do TDD em um contexto de desenvolvimento de *hardware* foi repensado para a correta adequação ao fluxo. Diferente do desenvolvimento de *softwares* em que os módulos são independentes entre si, a concepção de um módulo de *hardware* está em sua maior parte integrada a uma arquitetura do sistema como um todo. Devido a essa característica, pensou-se, na adoção da verificação formal dos blocos e das especificações, portanto a partir de uma especificação do projeto é possível extrair requisitos formais e escrever os testes correspondentes a esses comportamentos, assim cada um desses requisitos pode ser atendido serialmente pelo projetista do circuito de modo que, ao fim do processo, todo o módulo esteja completo.

É possível observar soluções para gerenciamento de fluxo e também de geração de *testbenchs*, como o UVM Framework e o System Testbench Generator da Cadence. A ferramenta aqui proposta, difere dessas, pois promove a integração entre as equipes de

desenvolvimento de *hardware*. A adoção dessa ferramenta modifica a forma como se dá o desenvolvimento tanto do *frontend* como da verificação.

O presente trabalho está dividido em cinco seções, a primeira é a introdução, onde também são expostos os objetivos do projeto. A segunda seção é o embasamento teórico no qual foi alicerçado o desenvolvimento da ferramenta. A terceira parte trata da ferramenta propriamente dita, em que é descrito sua estrutura e seu funcionamento, além dos conceitos por trás de cada tipo de construção. Na quarta seção é exemplificado um caso de uso da ferramenta para a implementação de um bloco que realiza operações aritméticas com uma interface *APB* (*Advanced Peripheal Bus*). Por fim na quinta seção são feitas as considerações finais do projeto. Além das seções que compõe o projeto, ao fim do documento a seção de Anexos contêm os códigos-fonte e o endereço do repositório da ferramenta para que sua funcionalidade possa ser aperfeiçoada futuramente.

1.1 OBJETIVOS

1.1.1 OBJETIVO GERAL

A execução desse trabalho objetiva o desenvolvimento de uma ferramenta de geração integrada de fluxo de referência para execução de projetos de desenvolvimento de *hardware*.

1.1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos para o desenvolvimento desse projeto são:

- Geração automática do arcabouço do projeto baseado em uma simples descrição de arquitetura do sistema;
- Facilitação de adoção de práticas e metodologias por meio da utilização da ferramenta proposta;
- Aceleração de desenvolvimento a partir dos códigos fontes gerados pela ferramenta;
- Unificação de *scripts* de execução de testes e simulações;
- Abertura do código fonte permitindo contribuições externas futuras;
- Exemplificar o uso da ferramenta através de um caso de uso.

2 EMBASAMENTO TEÓRICO

O desenvolvimento de uma ferramenta de automação de processos envolve a compreensão e entendimento dos métodos e procedimentos que devem ser automatizados. No caso da AURORA, o foco é a aceleração do fluxo de *design* e de verificação, portanto necessita-se o embasamento sobre os padrões IEEE 1800-2017, em que rege a descrição da linguagem SystemVerilog e SystemVerilog Assertions, utilizada para verificação formal, e o padrão IEEE 1800.2-2017, em que é descrito a metodologia de verificação funcional UVM.

Como a proposta da ferramenta é a integração entre as equipes através da adoção de práticas do TDD (*Test Driven Development*) também será descrito os principais conceitos relativos a essa prática de desenvolvimento.

2.1 LINGUAGEM SYSTEMVERILOG E SVA

A linguagem SystemVerilog, muitas vezes enquadrada na categoria de HDVL (*hardware design and verification language*, é uma linguagem que surgiu em meados do fim da década de 1990 como iniciativa da Accelera, um consórcio de companhias de EDA e usuários que objetivavam a criação de uma nova geração para a já consolidada linguagem Verilog (SPEAR, 2008).

No ano de 2005, o SystemVerilog alcançou o patamar de padrão IEEE. Passando por diversas revisões, o padrão mais atual que rege as funcionalidades e as regras da linguagem é o Padrão IEEE 1800-2017. Com o avanço desse padrão e sua constante revisão, a adoção da linguagem ganhou grande aceitação pela comunidade de *hardware*, sendo utilizado em aproximadamente 53% dos projetos de ASIC/IC para *design* e 78% para verificação (FOSTER, 2020).

2.1.1 PROPRIEDADES DA LINGUAGEM

A linguagem SystemVerilog possui um conjunto de características que a torna flexível tanto para o uso em *design* como em verificação. Devido a vasta extensão da linguagem só será descrita nessa seção algumas de suas características fundamentais para o entendimento da ferramenta exposta, características que são utilizadas em ambas as frentes de projeto. Para expandir o que será descrito aqui, sugere-se que o leitor consulte (SUTHERLAND; DAVIDMANN; FLAKE, 2006) para a sua utilização em *design* e modelagem e (SPEAR, 2008) para sua utilização em verificação.

Tipos de Dados

A linguagem Verilog contêm tipos de dado de 4 estados, sendo eles 0, 1, X e Z. Adicionalmente a esses tipos padrões de dados, a linguagem SystemVerilog passou a dar suporte aos tipos de dados de dois estados, onde só são considerados os valores 0 e 1, essa adição permite que as simulações possam ser feitas com maior velocidade.

Com os tipos de 2 estados são formados os agrupamentos de estruturas de dados. O primeiro tipo de 2 estados é o tipo *bit* que é um único valor binário, assim como o tipo *reg* do Verilog, o novo tipo pode ser agrupado em intervalos maiores de múltiplos *bits*. O tipo *int* é um inteiro de *32 bits* com sinal bastante similar ao seu homônimo na linguagem C. Os tipos com tamanho fixo (inteiro e suas variantes), em geral, são úteis para o uso em índices e em laços de repetição.

A linguagem trouxe como novidade a capacidade de utilização de tipos de ponto flutuante, ou *floats*. Assim como outras linguagens o SystemVerilog passou a dar suporte ao padrão IEEE 754 que define o uso da aritmética de ponto flutuante. Essa adição permitiu a aplicação da linguagem em contextos analógicos, em adição a já utilizada palavra-chave *net* em Verilog AMS, assim como expandiu a capacidade de modelagem em alto nível da linguagem. O tipo *real*, é a palavra chave utilizada pela linguagem para a utilização de tipos de ponto flutuante.

A utilização de *strings*, *queue* e *arrays* associativas também está disponível no SystemVerilog.

Módulos e Programas

Em SystemVerilog, a utilização de módulos se dá como na sua geração anterior o Verilog. Esses componentes em geral são utilizados para encapsular um circuito lógico e em sua construção são declarados os sinais de entrada e saída. No contexto dos módulos é que são permitidos a utilização das construções *always* do Verilog, com a adição do três novos tipos diferentes de blocos '*always*' que são introduzidos no SystemVerilog, ou seja, '*always_ff*', '*always_comb*' e '*always_latch*'. A adição desses novos tipos permite que as ferramentas de síntese façam a diferenciação de lógicas sequenciais das combinacionais mais facilmente.

Os programas, em SystemVerilog, são, em via de regra, onde estão contidas as partes ativas de um *testbench*. A palavra-chave *automatic* foi introduzida para fazer com que o armazenamento de variáveis se desse de forma mais parecida com as demais linguagens de programação que o Verilog tradicional. Com a adição de ferramentas de armazenamento mais voltadas para *software* foi possível a criação de variáveis dentro de contextos,

como as variáveis locais, e também valores armazenados globalmente, disponíveis para todos os processos.

Functions* e *Tasks

Assim como nas demais linguagens de programação, o SystemVerilog possui compatibilidade com o uso de funções. Na linguagem a utilização de chamadas externas ao escopo de código pode ser feito de duas formas, *function* e *task*. O uso das funções se assemelha ao uso em outras linguagens, cada função deve ter um tipo específico de retorno, assim como há a possibilidade de utilização de parâmetros. As *tasks* são um tipo especial de chamada externa em que é possível consumir tempo de simulação. Dessa forma, as funções são executadas instantaneamente e as *tasks* podem consumir tempo de simulação.

Interface de Sinais

Uma interface, em SystemVerilog, é uma coleção de definições de sinais. Os módulos agora podem se conectar as interfaces, ao invés, de realizarem conexões uns com os outros. Dessa forma as definições dos sinais só precisam ser feitas uma vez e não precisam ser refeitas em cada módulo. A direção dos sinais pode ser especificada pela criação de *modports* dentro da interface.

Também é possível a utilização tanto de funções como *tasks* dentro do escopo de uma interface. As funções e *tasks* podem ser externas a interface habilitando seu uso em módulos que estão conectados a interface. Apesar de bastante úteis, a maior parte desses construtores e funções externos a interface, em via de regra, não são compreendidos pela ferramenta de síntese, portanto seu uso acaba restrito ao fluxo de verificação.

Funções Externas

Desde o Verilog era possível utilizar chamadas de funções externas a linguagem por meio da PLI (*Progam Language Interface*). O SystemVerilog expandiu essa capacidade com a nova tecnologia do DPI, com sua utilização é possível realizar chamadas de funções da linguagem C no SystemVerilog de forma mais simples e direta.

Algumas empresas também dão suporte a formas de acesso a funções externas através de tecnologias proprietárias ou abertas. O principal exemplo de interface com funções externas é o UVM Connect da Mentor Graphics (ERICKSON, 2012) que é uma ferramenta aberta e independente. Outro exexmplo é o DirectC, ferramenta proprietária da Synopsys utilizada em seu simulador VCS (SYNOPSYS, 2006).

Mailbox

O uso de *Mailbox* é uma forma flexível de habilitar a comunicação entre duas entidades do *testbench*. Seu uso pode ser comparado a utilização de FIFOs. Sua capacidade de armazenamento de valores é expansível.

Orientação a Objetos

O SystemVerilog também introduziu uma das principais ferramentas de outras linguagem de programação, a orientação a objetos. Com essa característica, é possível a sua utilização em ambientes de verificação, assim como facilitou o reuso e a possibilidade de existência de diversas bibliotecas. As classes podem conter diversos componentes, membros e funções, assim como é possível a utilização das capacidades de outras linguagens OOP, como a hierarquia e o polimorfismo.

A Figura 2 mostra a evolução das palavras-chaves e capacidades da linguagem SystemVerilog ao longo de suas versões. A revisão de 2017 não trouxe novas funcionalidades a linguagem, entretanto, houve mudanças na clareza e como algumas construções devem ser utilizadas.

2.1.2 SYSTEMVERILOG ASSERTIONS

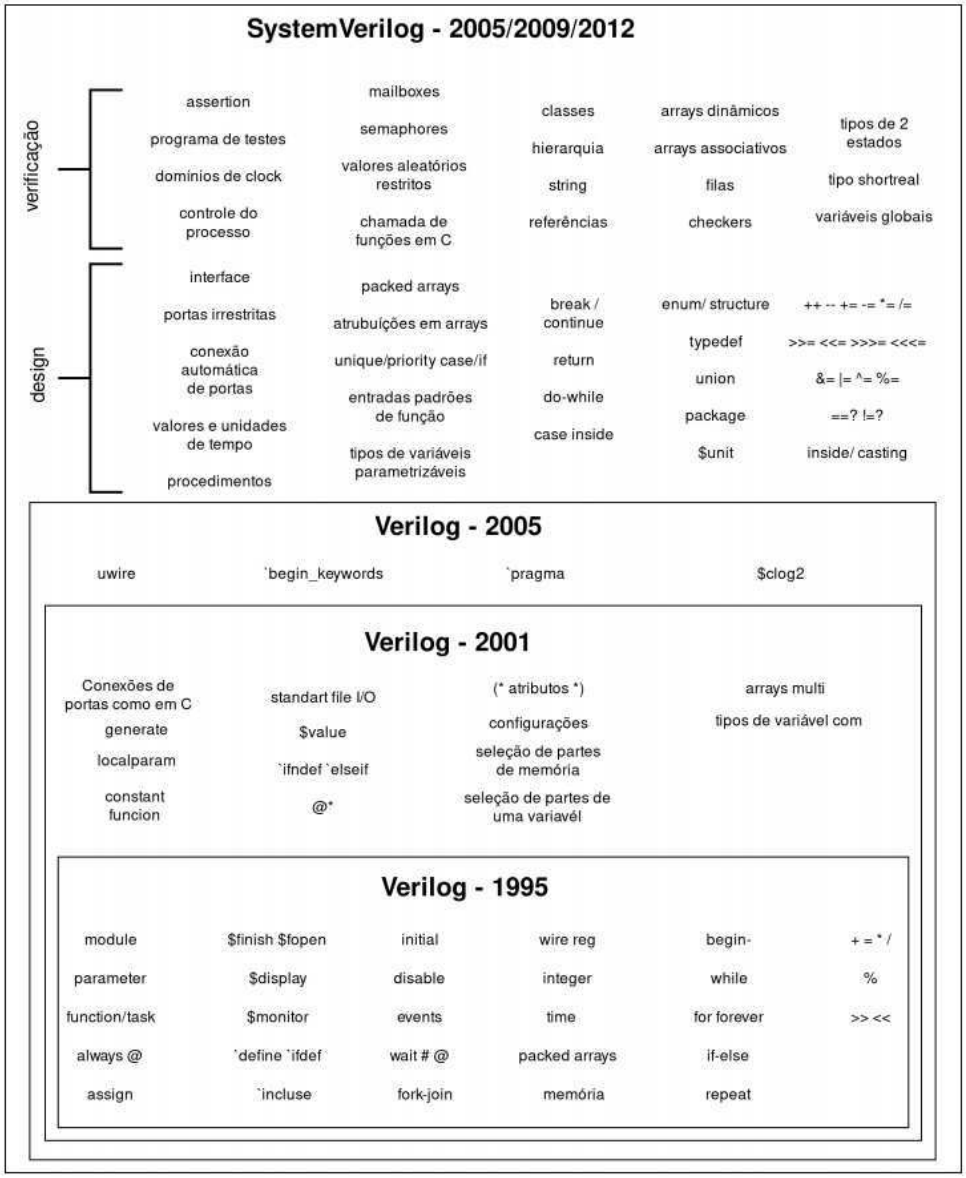
Enquanto *assertions* são parte de desenvolvimento de *software* há anos, a adoção do ABV (*Asserion Based Verification*) como instrumento de verificação de *hardware* só vem se tornando popular nos últimos tempos (VIJAYARAGHAVAN; RAMANATHAN, 2006). Entretanto, as propriedades que deseja-se utilizar na verificação de *hardware* são fundamentalmente diferente daquelas utilizadas em *software*.

A principal diferença entre os modelos desenvolvidos em *software* e em *hardware* é a utilização de recursos temporais. Linguagens do tipo HVL tem mecanismos para representar a passagem de tempo, enquanto que as linguagens procedurais (C, C++, Java, etc.) não possuem. Portanto a utilizações de *assertions* num contexto de verificação de *hardware* permite a escrita de assertivas temporais e sequências que não são possíveis em *software*. Com a utilização de SVA, os projetistas podem codificar o comportamento temporal desejado dos *designs* e checá-los através de ferramentas funcionais e formais. Além da utilização em verificação é possível a documentação dessas características a fim de esmiuçar o comportamento temporal de determinado barramento.

Sequências

Em qualquer modelo de *design*, sua funcionalidade é representada através de uma sequência de eventos lógicos (VIJAYARAGHAVAN; RAMANATHAN, 2006). Os eventos

Figura 2 – Evolução dos elementos do Verilog para o SystemVerilog



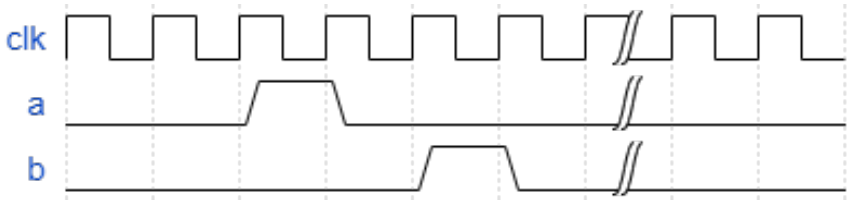
Fonte: (LIMA, 2019)

podem ser simples expressões booleanas, avaliadas no mesmo intervalo do pulso de *clock*, ou uma grande sequência de eventos que duram um período de múltiplos ciclos de *clock*. A sintaxe de SVA fornece uma palavra-chave para a descrição desses eventos temporais chamada de *sequence*. A utilização dessa ferramenta permite a escrita dos eventos temporais de acordo com sua sucessão. Um exemplo de descrição de uma sequência é mostrado a seguir:

```
1 sequence seq;
2   @(posedge clk) a ##2 b;
3 endsequence
```

Essa sequência é a expressão do seguinte comportamento temporal:

Figura 3 – Exemplo de sequência temporal descrita em SVA



Fonte: Próprio Autor

Propriedades

Quando um conjunto de sequências são utilizadas para compor um comportamento de um determinado conjunto de sinais há a formação de uma propriedade. Em SVA uma propriedade, cuja palavra chave é *property*, é utilizada como uma composição de sequências temporais que implicam em outra sequência temporal descrevendo assim um comportamento finito no tempo de um conjunto de sinais.

A sintaxe básica de uma propriedade em SystemVerilog Assertions é mostrada a seguir:

```
1 sequence seq;
2   a ##2 b;
3 endsequence
4
5 property p;
6   @(posedge clk) seq;
7 endproperty
```

Assertions

As propriedades descritas para um *hardware* precisam ser afirmadas para poderem ter sua validade conferida por meio de alguma ferramenta ou metodologia. Em SVA quando se quer checar a validade de uma propriedade utiliza-se a palavra-chave *assertion* de modo que a partir do momento de simulação sempre que ocorrer um evento as sequências descritas na propriedade avaliada serão checadas.

Além da palavra-chave o SystemVerilog também utiliza a palavra-chave *assume* para indicar que uma propriedade deve ter seu comportamento aceito *a priori*. Esse tipo de ferramenta é largamente utilizada no fluxo de verificação formal pois indica a ferramenta comportamentos que serão observados, de modo que a prova formal das *assertions* torna-se mais simples pois o espaço de busca é reduzido. Um exemplo do uso das *assertions* e *assumes* é quando um bloco de protocolo escravo é verificado, costuma-se assumir

que o comportamento dos sinais que viriam do mestre serão aceitos como corretos, diminuindo o espaço de busca para possíveis violações no barramento escravo. Um exemplo do uso de *assertions* é mostrado abaixo:

```
1 sequence seq;
2   @(posedge clk) a ##2 b;
3 endsequence
4
5 property p;
6   not seq;
7 endproperty
8 a_1: assert property(p);
```

2.2 TEST DRIVEN DEVELOPMENT

O *Test Driven Development*, ou TDD, é uma técnica de desenvolvimento incremental. Nenhum código é escrito sem antes haver um teste para verificar a corretude de sua implementação em relação a especificação (GRENNING, 2011). Baseando-se em testes pequenos e automatizados é possível fazer com que as unidades que compõe o sistema possam ser integradas de forma fácil e sem muitos *bugs*. Portanto é possível que os códigos inseridos no projeto tenham uma maior qualidade, facilitando o *debug* e a manutenção.

Apesar do nome o TDD não é uma técnica de testagem, apesar da escrita de muitos testes. A sua utilização visa solucionar problemas comuns no desenvolvimentos de projetos tanto de *software* como de *hardware*. A medida que os projetos crescem em complexidade, suas unidades passam a se tornarem pontos de fragilidade no sistema. Uma simples operação de adição, calculada incorretamente, pode desencadear uma reação catastrófica em termos de funcionalidade. A utilização do TDD visa, justamente, fortalecer as pequenas unidades de código diminuindo a fragilidade do sistema.

De acordo com (ASTELS, 2003), o TDD é uma metodologia que:

- Você mantém um conjunto exaustivo de testes;
- Nenhum código entra em produção a menos que tenha testes associados;
- Você escreve os testes primeiro;
- Os testes determinam qual código você precisa escrever;

Partindo do primeiro conceito, a existência de uma larga quantidade de testes habilita um conjunto de características nos códigos escritos. A primeira dela é que a existência de testes direcionados ao código escrito facilita, em termos de clareza, a real

intenção de cada um deles, facilitando dessa forma seu desenvolvimento por parte do *designer*. Outro fator importante e que está diretamente ligado ao desenvolvimento dos testes é a perspectiva deles. Os testes passam a ser direcionados ao desenvolvimento e não a funcionalidade, portanto, mesmo que dentro de um bloco de aritmética as operações de registro sejam invisíveis, deve haver testes que tem como escopo a escrita e leitura dos registradores.

O segundo ponto do TDD versa sobre os códigos que serão efetivamente incorporados ao projeto. Dentro de um fluxo dirigido por testes, todo o código deve ser efetivamente exercitado para evitar falhas. A principal função dessa premissa é que com códigos completamente testáveis permite que o desenvolvedor possa refatorar o código escrito sem medo de causar problemas de integração já que o mesmo será testado novamente. Portanto a utilização de códigos que foram testados permite que nenhuma mudança nos códigos causem problemas indesejados. Esse tipo de ferramenta é especialmente útil em um ambiente de desenvolvimento de *hardware* pois muitas modificações são realizadas em termos de otimizações de potência e área e não especificamente voltadas para a funcionalidade.

O terceiro ponto é o fundamento habilitador da metodologia. A escrita dos testes antes do desenvolvimento possibilita que a metodologia seja aplicada como método de desenvolvimento. Como cada funcionalidade e unidade do projeto estará associada a um teste, a equipe de teste do projeto saberá com mais clareza quais os testes deverão ser escritos e quais os requisitos devem ser exercitados.

Assim como o terceiro ponto facilita o trabalho da equipe de testes, o quarto ponto facilita o fluxo do desenvolvimento. Como o que deve ser desenvolvido está intimamente relacionado a um teste, a existência do segundo implica também que o objeto a ser testado deva existir. Outro ponto é a limitação do desenvolvimento, com a utilização dessa técnica o desenvolvedor apenas escreve o mínimo necessário para que o objeto passe no teste, de modo que evita o trabalho em funcionalidades não especificadas e também divide o trabalho em pequenos passos incrementais. Um exemplo dessa aplicação é a seguinte, imagine que um somador de dois números seja desenvolvido, o primeiro passo é testar se é possível inserir dois números no módulo, dessa forma o desenvolvedor deve escrever um código com apenas duas entradas, em sequência é testado a soma de dois números positivos, implicando no desenvolvimento da lógica de aritmética com números positivos, depois os testes com números negativos, expandindo o código para aritmética com números negativos e assim o somador foi completamente desenvolvido ao mesmo tempo que foi testado.

Assim como a existência dos testes guiam o desenvolvimento, no TDD, o ferramental possui uma importância fundamental no fluxo do projeto. O princípio do "Deixe a ferramenta lhe dizer o que deve ser feito"é bastante aplicável nesse contexto. Por exemplo ao escrever um teste em que eu precise da existência de um sinal de *valid* ao tentar simular

um bloco com esse teste a ferramenta provavelmente irá retornar um erro com o seguinte formato:

```
valid cannot be resolved or is not a signal.
```

Esse tipo de retorno da ferramenta guia o desenvolvedor, de modo que, ele insira apenas os componentes necessários no código, dessa forma, mantendo os códigos limpos e sem excesso de variáveis ou sinais.

Portanto, o desenvolvimento utilizando o TDD segue um processo cíclico como demonstrado na Figura 4. Cada iteração sempre associa um teste a um código desenvolvido mantendo a estrutura do fluxo completo.

Figura 4 – Ciclo de desenvolvimento do TDD.

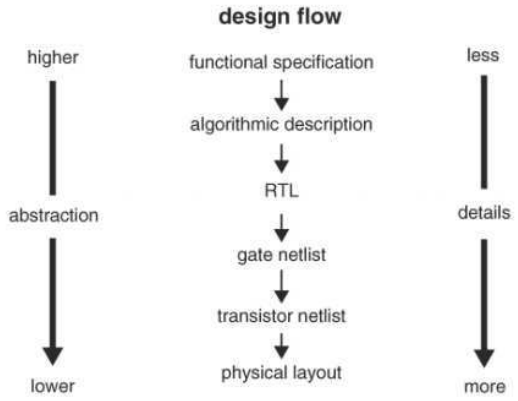
Fonte: Próprio Autor

2.3 HARDWARE DESIGN

O *design* de um sistema digital é um processo de transformação de um conjunto de especificações em uma implementação (LAM, 2005). Em implementações modernas, a densidade dos chips pode chegar a casa das centenas de milhões de transistores, tornando impossível o projeto desses dispositivos nesse nível. Dessa forma para se projetar um circuito digital é necessário que o sistema seja descrito em diferentes níveis de abstração (CHU, 2006). A Figura 5 mostra os níveis de abstração em relação a quantidade de detalhes escritos.

O processo de transformar as implementações mais abstratas em representações mais concretas é chamado de refinamento (CHU, 2006). Os dois primeiros níveis são

Figura 5 – Níveis de abstração de cada tipo de implementação de um *design* digital.



Fonte: (CHU, 2006)

escritos como o conjunto de especificações do bloco a ser construído. Muitas vezes nessa etapa é comum a utilização de uma linguagem de programação em que é modelado o comportamento em alto nível do bloco. O primeiro nível de implementação em termos de *hardware* é o *Register-transfer Level*, ou RTL. Nesse nível, o bloco é descrito em termos de unidades funcionais, como somadores, comparadores e registros. Nesse nível é comum a representação de dados de forma abstrata, assim como são também descrita as máquinas de estados finitos (FSM).

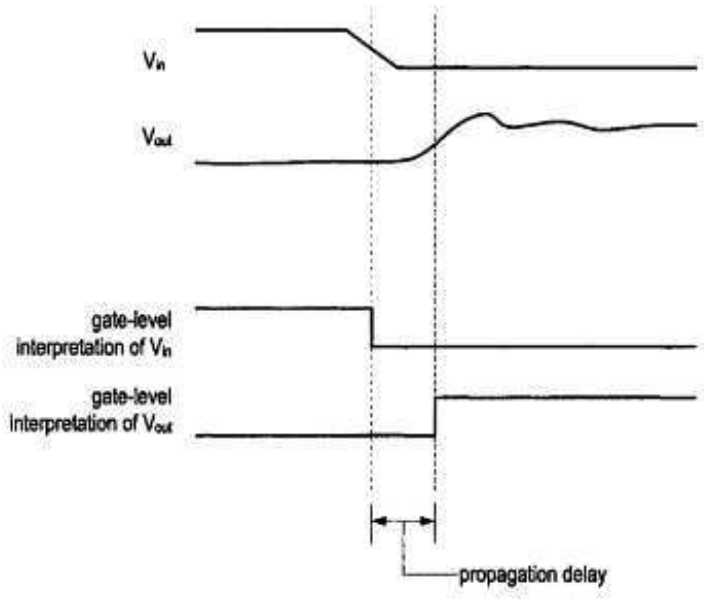
Ainda no nível de registro, em circuitos síncronos, os eventos discretos acontecem sincronizados com o sinal de relógio (*clock*) do bloco. As transições comuns de sinais são abstraídas e o modelo dos eventos apenas leva em consideração a amostragem dos dados nas bordas do sinal de *clock*, efeitos como atraso de propagação e *glitches* são ignorados nesse nível de abstração.

O próximo nível de abstração é o nível de *gates*. Nesse estágio o circuito é modelado como um conjunto de portas lógicas, portas *and*, *xor* e multiplexadores, e elementos de memória básicos, como *flip-flops* e *latches*. Apesar de estar em um nível de circuito lógico, os valores não são modelados de forma continua e sim se a tensão em cada porta está abaixo ou acima de um valor limítrofe. Portanto nesse nível ainda existe a modelagem em sinais discretos. É possível nesse nível modelar o sistema a partir de equações *booleanas*. Em relação a questões temporais, a transição de sinais é modelada como simples atrasos, em que é considerado o tempo de propagação até a obtenção de um valor estável de tensão, como exemplificada na Figura 6. Outras informações físicas podem ser retiradas desse nível de abstração, como a quantidade de *gates*, que em geral, fornece uma boa estimativa da quantidade final de transistores e a área ocupada pelo circuito.

Por fim o nível mais básico de implementação é o nível de transistor (excetuando os níveis de *layout* e demais que acontecem durante o processo de fabricação). Nesse

nível os elementos básicos que constitui os modelos são elementos de circuito eletrônicos como transistores, capacitores, resistores, etc. O comportamento do circuito nesse nível é modelado como um conjunto de equações diferenciais. A partir desse nível os sinais são tratados como analógicos e passam a poder assumir qualquer valor, e seu transitório passa a ser considerado para as simulações. É comum a utilização de *softwares* de simulação analógica como o SPICE para extrair as características do circuito projetado.

Figura 6 – Comparação de sinais entre os níveis de abstração de *gate level* e *transistor level*.



Fonte: (LAM, 2005)

2.4 VERIFICAÇÃO

Dentro de um fluxo de projeto de circuitos integrados, a verificação é a frente de trabalho que busca garantir que as implementações feitas pelo *design* cumprem os requisitos e a especificação. Se tratando de escopo, a verificação pode ser definida em dois tipos básicos, a equivalência lógica (*logical equivalence*) e a equivalência de modelo (*model checking*). O tipo de verificação que busca a equivalência lógica é aquele que compara duas implementações com o mesmo nível de abstração e busca provar que elas são logicamente equivalentes, para tal, basta demonstrar que para todo o conjunto de entradas possíveis as implementações sob verificação produzem a mesma saída.

O nível de verificação que trata de comparar duas diferentes implementações em níveis de abstração diferentes é aquela que busca a equivalência de modelo. Ao longo do processo de refinamento, como descrito na seção anterior, é possível que ocorram problemas e erros funcionais. Portanto, o objetivo desse tipo de verificação é garantir que

a "intenção funcional"do circuito não foi perdida quando o nível de abstração foi alterado, diz-se então que as implementações são "funcionalmente equivalentes". O processo de verificação, nesse sentido, é feito de forma oposta ao fluxo de *design*, busca-se verificar a corretude das implementações em relação ao nível superior de abstração.

O fluxo tradicional de verificação começa com a entrega da implementação. A partir de então os verificadores podem comparar os objetos em diferentes níveis de abstração. Algumas propostas atuais de fluxo de verificação permitem que o desenvolvimento seja feito em paralelo entre o *design* do *hardware* e sua verificação, principalmente a partir de modelos de referência utilizando SystemC (SILVA et al., 2005), ou substituindo partes do RTL por BFMs (WANG, 2015).

Como já discutido, a verificação pode ser de dois tipos básicos, com relação ao *model checking* é importante ressaltar que se o documento de especificação puder ser abstraído em termos de expressões lógicas e temporais, torna-se possível checar o comportamento do *design* em relação a satisfatibilidade dessas expressões. Essa técnica é conhecida como verificação formal, estática, ou *property checking*. Portanto o esquema de divisão da verificação pode ser ilustrado na Figura 7.

2.4.1 VERIFICAÇÃO FUNCIONAL E UVM

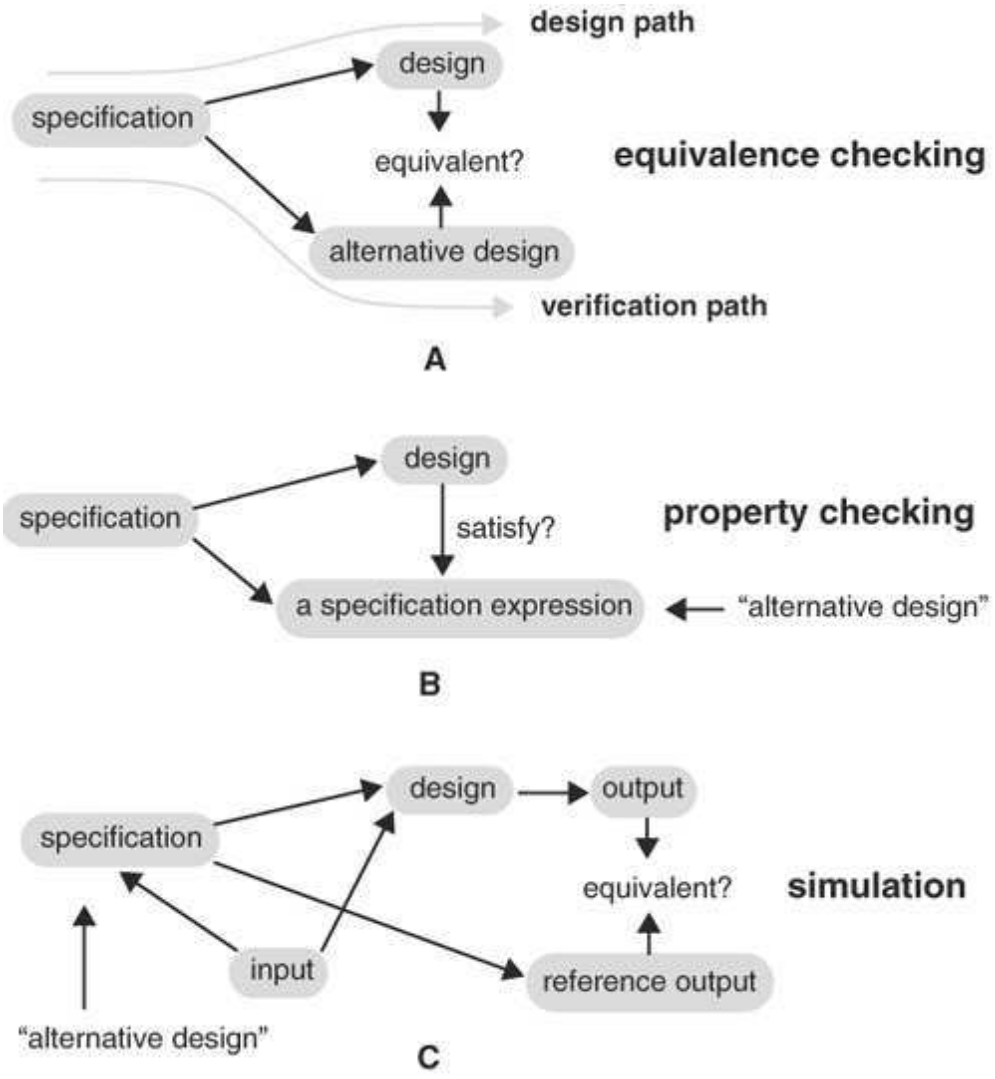
A verificação funcional pode ser definida como o processo usado para demonstrar a funcionalidade correta de um circuito (BERGERON, 2012). O argumento a favor desta técnica vem do fato de que se tanto o *hardware* descrito quanto os modelos abstratos que reproduzem a mesma funcionalidade foram feitos de forma independente e todos produzem as mesmas respostas para o mesmo teste, eles são extremamente prováveis de estarem corretos. Portanto, a tarefa da verificação funcional é escrever a estrutura dos testes que permitem a comparação correta entre os modelos em diferentes níveis de abstração estabelecidos.

Para o estabelecimento de uma estrutura de verificação funcional é necessário quatro principais componentes: Uma fonte geradora de estímulos, um monitor para capturar as saídas produzidas pelo objeto testado, um modelo de referência que gera as saídas desejadas e por fim um comparador que verifica se os critério de semelhança entre as saídas produzidas foi satisfeito. A Figura 8 demonstra uma estrutura básica de um ambiente de verificação funcional.

Universal Verification Methodology

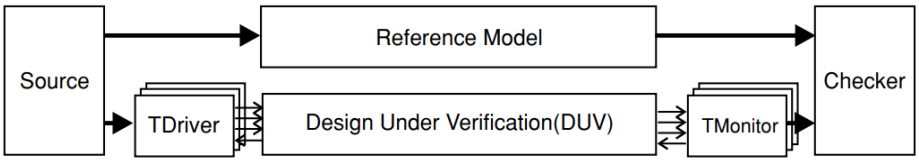
A metodologia de verificação universal (UVM) é uma metodologia padronizada para verificar *designs* digitais. Os principais objetivos do UVM são a reutilização de componentes para reduzir o tempo de lançamento no mercado, ser direcionado para ve-

Figura 7 – Tipos possíveis de verificação.



Fonte: (LAM, 2005)

Figura 8 – Estrutura de um ambiente de verificação funcional.



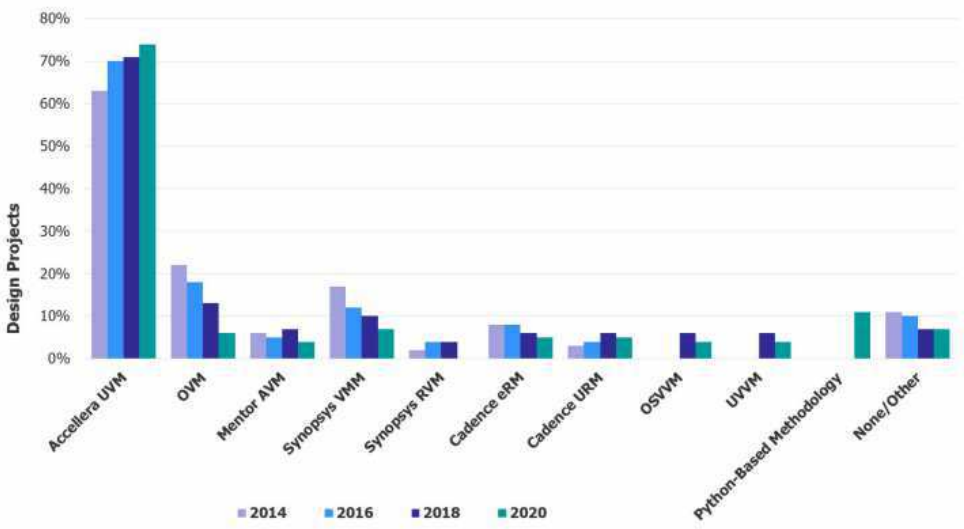
Fonte: (SILVA et al., 2005)

rificar sistemas pequenos a grandes e estabelecer um padrão na indústria e engenharia de verificação. Os principais benefícios da metodologia são um amplo suporte de biblioteca de classes bases, que alcançaram o patamar de padrão IEEE 1800.2-2017, poder realizar verificação aleatória orientada por cobertura funcional, também fornecer fluxo de

verificação prático e eficiente reutilizando casos de teste e *testbenchs* de IPs já desenvolvidos, além de que a padronização é independente do fornecedor de EDA (PAVITHRAN; BHAKTHAVATCHALU, 2017).

A utilização da biblioteca UVM tornou-se padrão na indústria. Segundo (FOSTER, 2020) 73% dos projetos de *hardware* utilizam de alguma forma a metodologia, Figura 9. Graças ao esforço do consórcio Acellera, muitas das ferramentas desenvolvidas para verificação dão suporte ao UVM, assim como muitas ferramentas de simulação e emulação das companhias de EDA. A ampla utilização dessa metodologia motivou a adoção da mesma como parte integrante do fluxo proposto pela ferramenta Aurora, descrita nesse documento.

Figura 9 – Histórico de utilização de metodologias de verificação em pesquisa espontânea com desenvolvedores.



Fonte: (FOSTER, 2020)

2.4.2 VERIFICAÇÃO FORMAL

A verificação baseada em métodos formais, ou verificação formal, é uma metodologia que, diferentemente das baseadas em simulação, não necessitam da geração de vetores de entradas. Sua aplicação como metodologia baseia-se em computação simbólica e em algoritmos de implicação de espaço de estados finitos capazes de enumerá-los e percorre-los (LAM, 2005).

Na utilização de métodos formais, é escrito uma propriedade formal, que é uma especificação parcial do *design* testado, e uma ferramenta atesta ou contesta se o *hardware* em questão possui a propriedade ou não. A ideia por trás disso é a busca em todos os espaços de estados uma situação em que a propriedade falha e então nesse ponto é gerado um contra-exemplo, caso contrário a propriedade é satisfeita. A sua utilização na indústria

só foi possível graças ao avanço da computação simbólica que permitiu a checagem de diversos pontos do espaço de estados de uma única vez, aumentando significativamente a eficiência do método.

Apesar de extremamente eficaz, a verificação formal sofre de algumas problemáticas limitando seu uso, elas são:

1. Erros na especificação. Uma implementação de uma verificação em que há algum problema nas especificações possivelmente não será descoberto nesse tipo de verificação, já que a não existência do modelo de referência não evidenciará essa falha.

2. Propriedades que não cobrem o funcionamento completo do *design*. Em muitos casos, quando não há a cobertura completa do *design* através da escrita das propriedades, partes essenciais do objeto testado ficam descobertas e podem camuflar erros funcionais.

3. Erros na escrita das propriedades. Um propriedade mal escrita pode provocar uma representação incorreta da especificação e sobrecarregar os testes do *design*.

4. *Bugs* nas ferramentas de verificação formal. *Bugs* nas ferramentas de verificação formal pode provocar um falso-positivo que não será descoberto no processo de verificação.

Portanto a utilização de verificação formal dentro de um fluxo de projeto possibilita alcançar alguns resultados mais rapidamente, além de poder obter uma prova formal das propriedades analisadas pelas ferramentas. Apesar do custo computacional, todas as vantagens envolvidas com o uso dessa forma de verificação são justificadas pela sua versatilidade e segurança.

3 FERRAMENTA DE GERAÇÃO DE FLUXO DE REFERÊNCIA

Nesta seção será descrita a ferramenta de geração de fluxo de referência Aurora, objeto de estudo e desenvolvimento desse trabalho. A ferramenta surge como uma forma de possibilitar a utilização de metodologias baseadas no TDD para o desenvolvimento de um fluxo de projeto de *hardware* digital. Devido a essa característica que o produto final gerado pela ferramenta é um fluxo de referência, com a utilização dos arquivos gerados é possível aplicar a metodologia adequada de desenvolvimento. Caso o usuário da ferramenta não deseje utilizar o TDD em seu projeto, a ferramenta também pode ser utilizada, mas apenas como um gerador de *testbench* funcional.

A seção está dividida em três partes, a primeira trata da descrição da ferramenta enquanto *software* em si. A segunda parte trata da utilização da ferramenta para a geração de fluxo de referência. A terceira e última parte fala sobre algumas considerações e adaptações do uso do TDD em projetos de *hardware*.

3.1 DESCRIÇÃO DA FERRAMENTA

A ferramenta Aurora foi desenvolvida como objeto desse Projeto de Engenharia Elétrica. Sua utilização é livre e distribuído sob a licença MIT. A linguagem utilizada no desenvolvimento da ferramenta foi o Python. A lista de pacotes e dependências que foram utilizadas na ferramenta é listada a seguir:

1. Python 3.X
2. Packages:
    a) os
    b) sys
    c) getopt
    d) colorama
    e) pathlib

Para seu funcionamento o usuário deve fornecer dois arquivos de configuração. Esses arquivos são lidos e interpretados pela ferramenta para gerar a configuração adequada fornecida pelo usuário. Esses arquivos são divididos em dois, pois cada frente de trabalho

tem uma configuração própria, e assim um utilizador da equipe de *frontend* pode gerar apenas a parte relativa ao seu fluxo e outro utilizador da equipe de verificação pode fazer o mesmo, garantindo a interoperabilidade do fluxos e a correta localização dos arquivos e referências.

A ferramenta tem como ponto de principal de funcionamento a existência de arquivos de *templates* que são modificados de acordo com a configuração escolhida pelo usuário e as necessidades do *hardware* a ser desenvolvido. Os arquivos de configuração são interpretados pela ferramenta e as informações extraídas são utilizadas no preenchimento dos *templates*. A estrutura desses arquivos são mostradas a seguir, é possível notar a inserção de identificadores que seguem o padrão:

|-IDENTIFICADOR-|

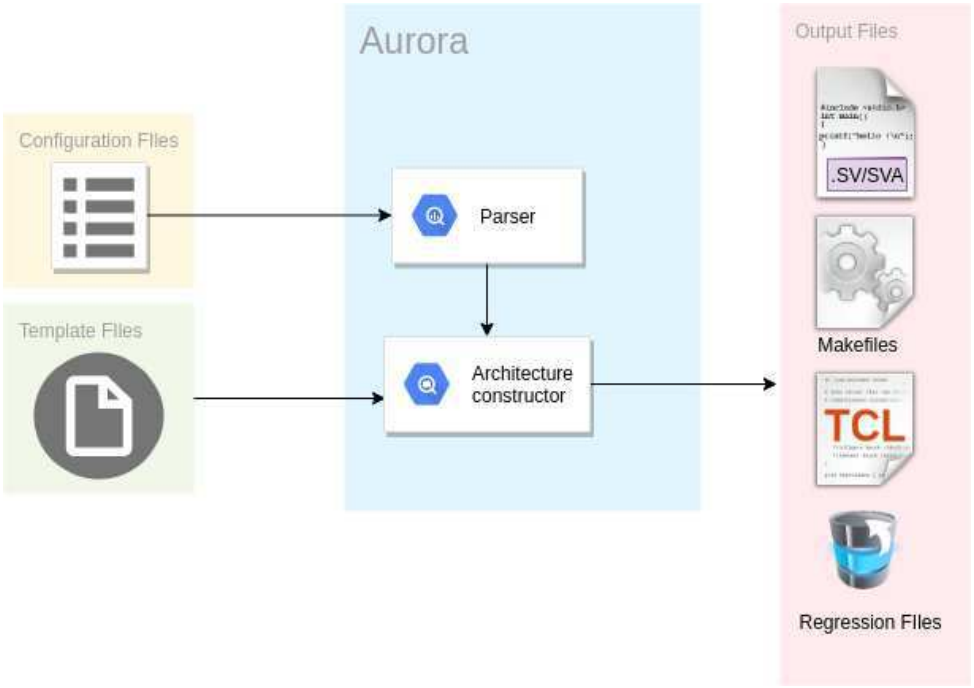
A inserção desses identificadores servem para que a ferramenta insira os trechos de códigos nas posições corretas.

```
1  class |-ENV-| extends uvm_env;
2
3      |-AGENT-|
4      |-SCOREBOARD-|
5      |-VIP-|
6
7      `uvm_component_utils(|-ENV-|)
8
9      function new(string name, uvm_component parent = null);
10         super.new(name, parent);
11     endfunction
12
13     virtual function void build_phase(uvm_phase phase);
14         super.build_phase(phase);
15         |-AGENT_CREATION-|
16         |-SCOREBOARD_CREATION-|
17         |-VIP_CREATION-|
18     endfunction
19
20     virtual function void connect_phase(uvm_phase phase);
21         super.connect_phase(phase);
22         |-CONNECTIONS-|
23     endfunction
24
25     virtual function void end_of_elaboration_phase(uvm_phase phase);
26         super.end_of_elaboration_phase(phase);
27     endfunction
28
29 endclass
```

Listing 3.1 – Exemplo de um *template* utilizado na ferramenta.

O fluxograma de utilização da ferramenta é mostrado na Figura 10. Como é possível observar na figura o produto final da ferramenta é um conjunto de arquivos, devidamente organizados em seus respectivos repositórios, esses arquivos são de diversos tipos, sendo eles: arquivos fonte SystemVerilog, SystemVerilog Assertions, Makefiles, arquivos de *scripting Tcl* e de regressão VSIF. Dessa forma é possível notar que além da geração dos códigos-fontes para um projeto de *hardware* a ferramenta produz um outro conjunto de ferramentas de automação que possibilita a utilização do TDD.

Figura 10 – Fluxograma de funcionamento da ferramenta Aurora.



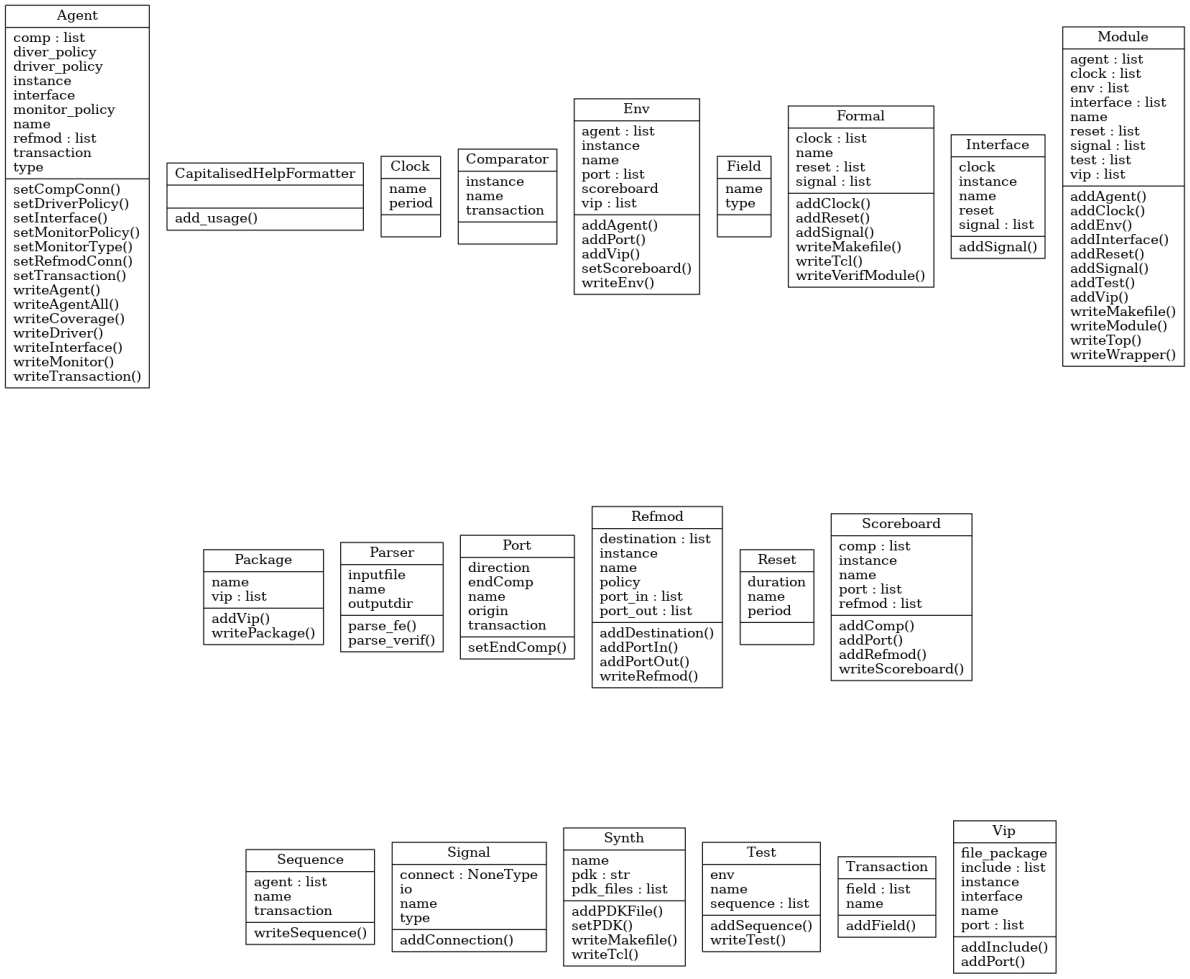
Fonte: Próprio autor

Para o desenvolvimento da ferramenta foi-se utilizado a orientação a objetos da linguagem Python. A estrutura de classes é mostrada na Figura 12.

Observa-se no diagrama de classes a presença de classes próprias para a construção dos elementos de projetos como por exemplo a classe "*Module*"e "*Interface*". Algumas classes abrangem instâncias e listas de outras, com o intuito de flexibilidade, muitos dos componentes podem existir com múltiplas instâncias, por isso a preferência por incluir membros na classe com o tipo lista de forma que a estrutura do componente pode ser adequada a quantidade de instâncias dos módulos que o compõe.

A classe "*Parser*"é a responsável por extrair a estrutura do projeto a partir dos arquivos de configuração. Sua função é basicamente dividir as seções do arquivo fonte de modo a separar as configurações específicas de cada componente. Em linhas gerais, a primeira tarefa da classe é agrupar os componentes de configuração de acordo com seu tipo. A próxima etapa da função do *parser* é extrair os dados de cada um dos

Figura 11 – Diagrama de classes da ferramenta Aurora.



Fonte: Próprio autor

componentes, separando mais uma vez cada um deles e agrupando os do mesmo tipo. Uma vez cada componente corretamente identificado e seus dados também agrupados é possível construir a arquitetura do projeto.

Uma vez construída a arquitetura do projeto, as classes são criadas e suas funções *write** são chamadas gerando os arquivos no diretório correto.

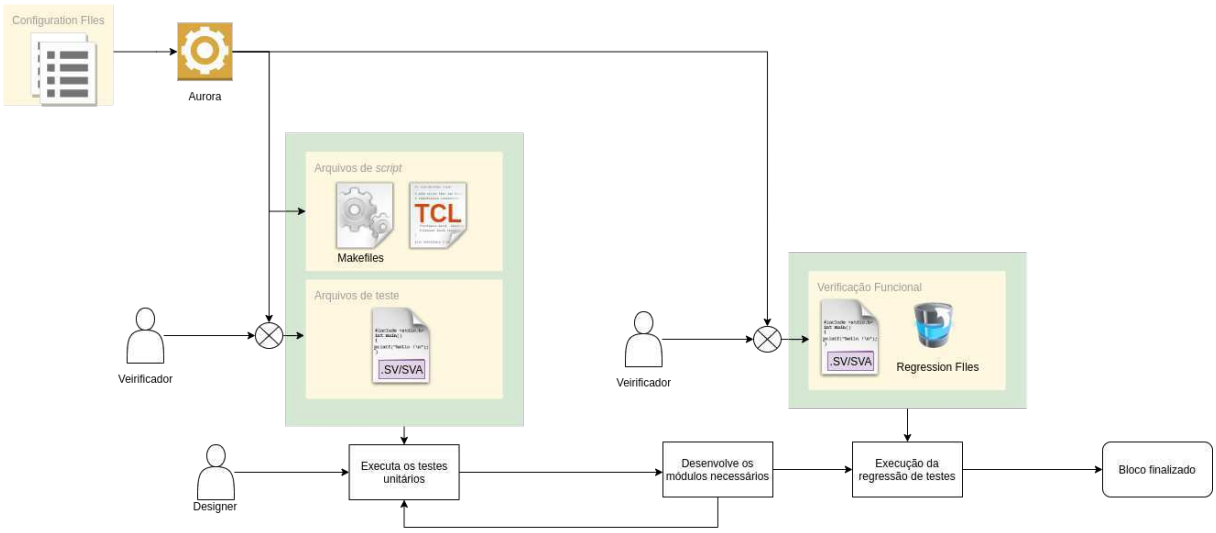
Atualmente, a ferramenta conta com as seguintes funcionalidades:

- Geração semi-automática de *testbenches* funcionais, com suporte a construção arquiteturas flexíveis de *testbench* como os modelos hierarquicos e decompostos apresentados em (OLIVEIRA et al., 2010);
- Possibilidade de integração de VIPs no ambiente de verificação funcional;
- Geração semi-automática de propriedades formais e arquivos correlacionados (e.g. *bindings*);

- Geração dos *scripts* e Makefiles para o fluxo de projeto, sendo eles os de simulação RTL, síntese lógica, Aplicativos de provas formais de propriedades, simulação gatelevel, etc;
- Geração dos arquivos contendo as propriedades formais de cada teste descrito para utilização de TDD.

O produto final entregue ao usuário da ferramenta é os arquivos que habilitam o fluxo de projeto utilizando o TDD. Os arquivos gerados facilitam a atividade do desenvolvedor e integram os fluxos de projetos, tanto de *frontend* quanto de verificação. A unificação dos *scripts* de simulação e teste aceleram a atividade de simulação e *debug*. A Figura **??** demonstra a aplicação da ferramenta em um fluxo de projeto.

Figura 12 – Fluxo de referência proposto com os arquivos gerados pela ferramenta.



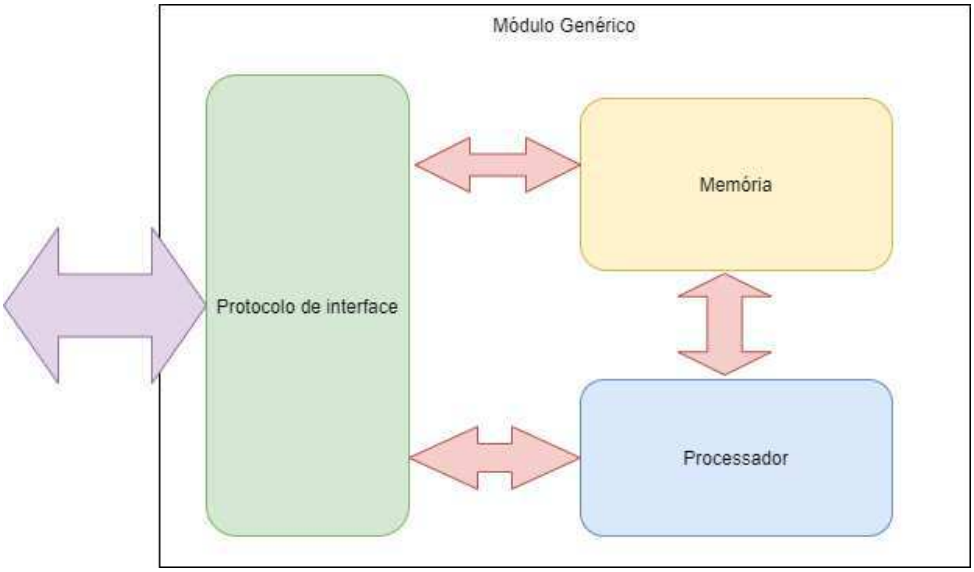
Fonte: Próprio autor

3.2 UTILIZAÇÃO DA FERRAMENTA

Com relação a utilização da ferramenta para a geração semi-automática dos arquivos, é necessário a adoção de práticas de TDD para que toda a capacidade da ferramenta seja aproveitada. A primeira etapa para utilizar a ferramenta é a correta especificação do módulo a ser desenvolvido, a macro-arquitetura do bloco e a arquitetura do testbench funcional. O exemplo a seguir ilustrará como aproveitar ao máximo da utilização da ferramenta.

Suponha que desejamos desenvolver um bloco qualquer que é composto dos módulos ilustrados na Figura 13.

Figura 13 – Diagrama de representação de um bloco genérico.



Fonte: Próprio autor

Primeiramente estabelecemos a estrutura de testes que deve ser utilizada é a que deve existir um conjunto de propriedades que descreva cada um dos módulos que compõe nosso modelo genérico, ou seja:

- Conjunto de *assertions* que descreva as operações no barramento externo de acordo com o estabelecido pelo protocolo adodado, e.g. AMBA AXI, PCIe, etc.;
- Conjunto de *assertions* que descreva as operações de leitura e escrita no bloco de memória;
- Conjunto de *assertions* que descreva algumas operações, com resultados conhecidos, no processador em que se observe uma sequência esperada na saída.

Portanto, inicia-se a escrita do arquivo de configuração como mostrado abaixo:

```
1 module{
2 name=genericModule
3 formal=true
4 }
5
6 clock{
7 name=clock
8 period=90
9 }
10
11 reset{
12 name=reset
13 period=3e12
14 duration=180
15 }
```

```
16
17 property{
18 name=protocolCheck
19 signal=i_valid
20 signal=i_rw
21 signal=i_addr
22 signal=i_data
23 signal=o_valid
24 signal=o_data
25 }
26
27 property{
28 name=processorCheck
29 signal=i_opperand
30 signal=i_op
31 signal=o_valid
32 signal=o_data
33 }
34
35 property{
36 name=memoryCheck
37 signal=rw
38 signal=addr
39 signal=i_data
40 signal=o_data
41 }
```

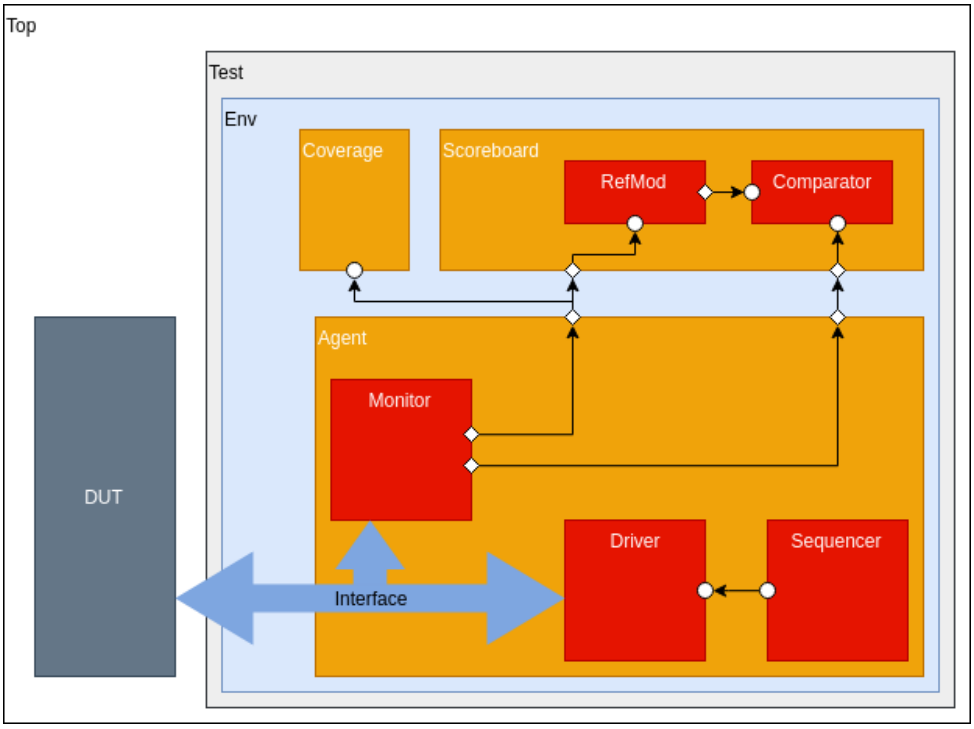
Com essa descrição a ferramenta já irá criar os arquivos onde deverão ser escritos as propriedades com seus respectivos nomes, assim como as demais ligações com o ambiente de frontend e o módulo RTL propriamente dito.

Agora podemos também indicar no nosso arquivo de configuração uma estrutura de *testbench* a ser utilizada. Tomando como base a ilustrada na Figura 14

Adicionando ao mesmo arquivo anterior:

```
1 signal{
2 name=in_data
3 type=logic [8]
4 io=input
5 connect=in_data
6 }
7
8 signal{
9 name=out_data
10 type=logic [8]
11 io=output
12 connect=out_data
13 }
14
15 clock{
16 name=clock
17 period=90
18 }
19
20 reset{
```

Figura 14 – Representação do ambiente de verificação a ser construído.



Fonte: Próprio autor

```
21 name=reset
22 period=3e12
23 duration=180
24 }
25
26 interface{
27 name=agentDummy
28 signal=in_data
29 signal=out_data
30 clock=clock
31 reset=reset
32 }
33
34 if_instance{
35 type=agentDummy
36 con=in_data,in_data
37 con=out_data,out_data
38 instance=agtDummy
39 }
40
41
42 field{
43 name=in_data
44 type=int
45 }
46
47 field{
48 name=out_data
49 type=int
```

```
50 }
51
52 transaction{
53 name=dummyTransa
54 field=in_data
55 field=out_data
56 }
57
58 agent{
59 name=agentDummy
60 instance=instanceDummy
61 interface=agtDummy
62 transaction=dummyTransa
63 driver_policy=
64 monitor_policy=
65 type=bidirectional
66 refmod=dummyRfm
67 comp=dummyComp
68 }
69
70 refmod{
71 name=dummyRefmod
72 instance=dummyRfm
73 refmod_policy=
74 comp=dummyComp
75 connect=
76 }
77
78
79 comp{
80 name=comp
81 instance=dummyComp
82 transaction=dummyTransa
83 }
84
85 sequence{
86 name=dummySequence
87 agent=instanceDummy
88 transaction=dummyTransa
89 }
90
91 test{
92 name=dummyTest
93 sequence=dummySequence
94 }
```

Finalizando assim o arquivo de configuração para o fluxo de verificação.

No fluxo de *frontend* descreve-se o módulo e os sinais que devem ser gerados, além de opções adcionais para criação de *scripts* de síntese lógica.

```
1 module{
2 name=testeIO
3 formal=true
4 }
5
```

```
signal{
name=in_data
type=logic [8]
io=input
connect=in_data
}

signal{
name=out_data
type=logic [8]
io=output
connect=out_data
}

clock{
name=clock
period=90
}

reset{
name=reset
period=3e12
duration=180
}

pdk{
name=xfab
dir=/example/xfab/test
file=bloblo/blabla
file=blablo/blablu
}
```

Por fim executa-se o script da seguinte forma:

```
python3 ../scripts/aurora.py (verif|fe) -i <input_configuration_file> -o <output-
    directory>
```

O uso da ferramenta ainda está estrita a utilização em conjuntos com os *softwares* proprietários da Cadence Design System devido a disponibilidade das licenças para o uso, portanto os *scripts* de síntese lógica são compatíveis com a ferramenta *Genus*, os *scripts* de simulação de RTL com o *XCelium* e os de Verificação Formal com o *JasperGold*.

3.3 CONSIDERAÇÕES SOBRE O USO DE TDD PARA PROJETOS DE HARDWARE

A utilização do TDD para projetos de *Hardware* como proposto nesse trabalho por meio da utilização da ferramenta Aurora requer algumas adaptações e modificações de certas práticas dada as limitações do tipo de projeto. Como a metodologia foi pensada para o desenvolvimento de *software* seria natural a necessidade de tais adaptações.

Como nesse trabalho é proposto a adoção de técnicas formais para servir como os testes unitários da metodologia, as limitações das técnicas formais acabam sendo transferidas como limitações de casos de testes. Portanto algumas especificações não podem ser testadas de maneira formal devido a sua complexidade computacional, notadamente as operações de processamento de dados e de sinais. Para contornar esse problema, como na seção anterior sugere-se que sejam escritas algumas *assertions* com um conjunto limitado de vetores de entradas em que é conhecida sua saída, assim é possível testar minimamente a funcionalidade, entretanto dessa forma não é garantida a sua cobertura funcional, que só será garantida quando executada todo o processo de verificação funcional.

Rigorosamente, as práticas de verificação formal não podem ser considerada uma técnica de testagem de um *hardware*. Dessa forma outra terminologia similar para a metodologia aqui proposta poderia ser *Property Driven Development*. Entretanto, sem perda de sentido, a checagem das propriedades formais funcionam como método de checagem de corretude funcional da implementação feita pelo desenvolvedor.

Uma das práticas mais comuns da utilização de TDD para *software* é o princípio que o mesmo desenvolvedor que escreve os testes é aquele que efetivamente implementa o código. De um ponto de vista de verificação de *hardware* faz-se necessário que o código de verificação seja escrito por outra pessoa além daquela que fez a implementação, essa necessidade garante que não haverá vícios de interpretação de documentação por parte de um dos desenvolvedores, adicionando mais uma camada de redundância ao processo. Além do fator de segurança, a diferenciação entre o verificador e o desenvolvedor também viabiliza uma dupla revisão na especificação, já que no processo do TDD, uma especificação mal compreendida ou documentada pode causar problemas funcionais graves.

O nível de abstração dos testes também deve ser considerado quando se adota esse tipo desenvolvimento. Como os blocos que compõe os sistemas são muitos, a escrita de testes para cada um deles é extensa e custosa, de modo que o desenvolvimento seria bastante impactado, já que só poderia iniciar com a finalização dos testes unitários. Recomenda-se a abstração do *hardware* a ser desenvolvido em unidades de funcionamento maiores e que façam sentido a escrita de um teste, por exemplo, não faz sentido a verificação de um somador de um *bit* quando se pode escrever testes para um somador de 32 bits que será o efetivamente implementado. Ao mesmo tempo que não se deve abstrair alguns componentes das escritas dos testes, pois dessa forma a metodologia não irá garantir a qualidade do código ou sua correta funcionalidade.

4 CASO DE USO

Com o intuito de mostrar a ferramenta Aurora como habilitadora de projetos de desenvolvimento de *hardware* utilizando o TDD como uma metodologia de desenvolvimento foi-se demonstrado seu funcionamento em um módulo de *design* como prova de conceito. Nesta seção será descrito os passos adotados para a realização do experimento e o desenvolvimento do módulo a partir de um conjunto de especificações.

4.1 ESPECIFICAÇÃO DO HARDWARE

O *hardware* utilizado como prova de conceito foi chamado de Arith, pois se trata de um simples módulo que realiza operações de soma e multiplicação de números inteiros de 32 bits operando com uma interface AMBA APB3 (Advanced Peripheal Bus) com extensão de sinal de "SLVERR"(ARM, 2003). Abaixo segue a tabela com a descrição dos sinais de interface.

Nome	**Tamanho**	**I/O**	**Descrição**
pclk	1	I	Sinal de clock que opera a 11 MHz
presetn	1	I	Sinal de reset assíncrono baixo ativo
paddr	32	I	Sinal de endereçamento
pwdata	32	I	Sinal de dado de entrada
psel	1	I	Sinal de seleção, o mestre indica ao periférico que ele foi selecionado para a operação
penable	1	I	Sinal de habilitação, esse sinal deve ser subsequente ao psel e indica o início da transferência
pwrite	1	I	Sinal de seleção do modo de transferência, se ativo a seleção é para realização de uma escrita
prdata	32	O	Sinal de dado de saída
pready	1	O	Sinal de ready do dado, ao ativar indica a finalização da transferência
pslverr	1	O	Sinal de erro, ao ativar indica uma transferência mal sucedida

Tabela 1 – Descrição da interface de sinais do bloco Arith

O bloco deve possuir um banco de registrados com 4 registros de 32 bits, nesses registros é feito tanto a entrada de dados como a de instruções. A descrição do banco de registradores, assim como as permissões de cada um deles é feita na Tabela 2.

Endereço	**Permissão**	**Descrição**
0x00	R/W	Operando 1
0x04	R/W	Operando 2
0x08	R/W	Registro de controle
0x12	R	Resultado da operação

Tabela 2 – Descrição do banco de registradores

O registro de endereço 0x08 é utilizado para controle. Ele é composto de um *bit* de indicação de início da operação e um bit de indicação de qual operação ele deve fazer, sendo o valor 0 para adição e o valor 1 para multiplicação.

4.2 ESCRITA DOS TESTES

Para a utilização correta da metodologia, após a escrita dos arquivos de configuração, foi feita a escrita das propriedades formais do bloco a ser desenvolvido. Dada a natureza do bloco optou-se por escrever o seguinte conjunto de testes:

- Propriedades de sanidade de sinais (Verificação se eles em algum momento assumem valores Z ou X após o reset;
- Propriedades para verificar o comportamento dos sinais apos a aplicação do *reset*;
- Propriedades para verificar se as operações de transferência ocorrem de acordo com o descrito no protocolo;
- Propriedades para verificar violações de endereços (escritas e leituras em endereços não permitidos);
- Propriedades para verificar escrita e leitura nos registradores;
- Propriedades para verificar a correta realização das operações.

Todas as propriedades escritas para esse módulo podem ser consultadas no Apêndice C.

Após essa escrita também foi feito a construção do ambiente de verificação funcional seguindo a mesma arquitetura mostrada na Figura 14.

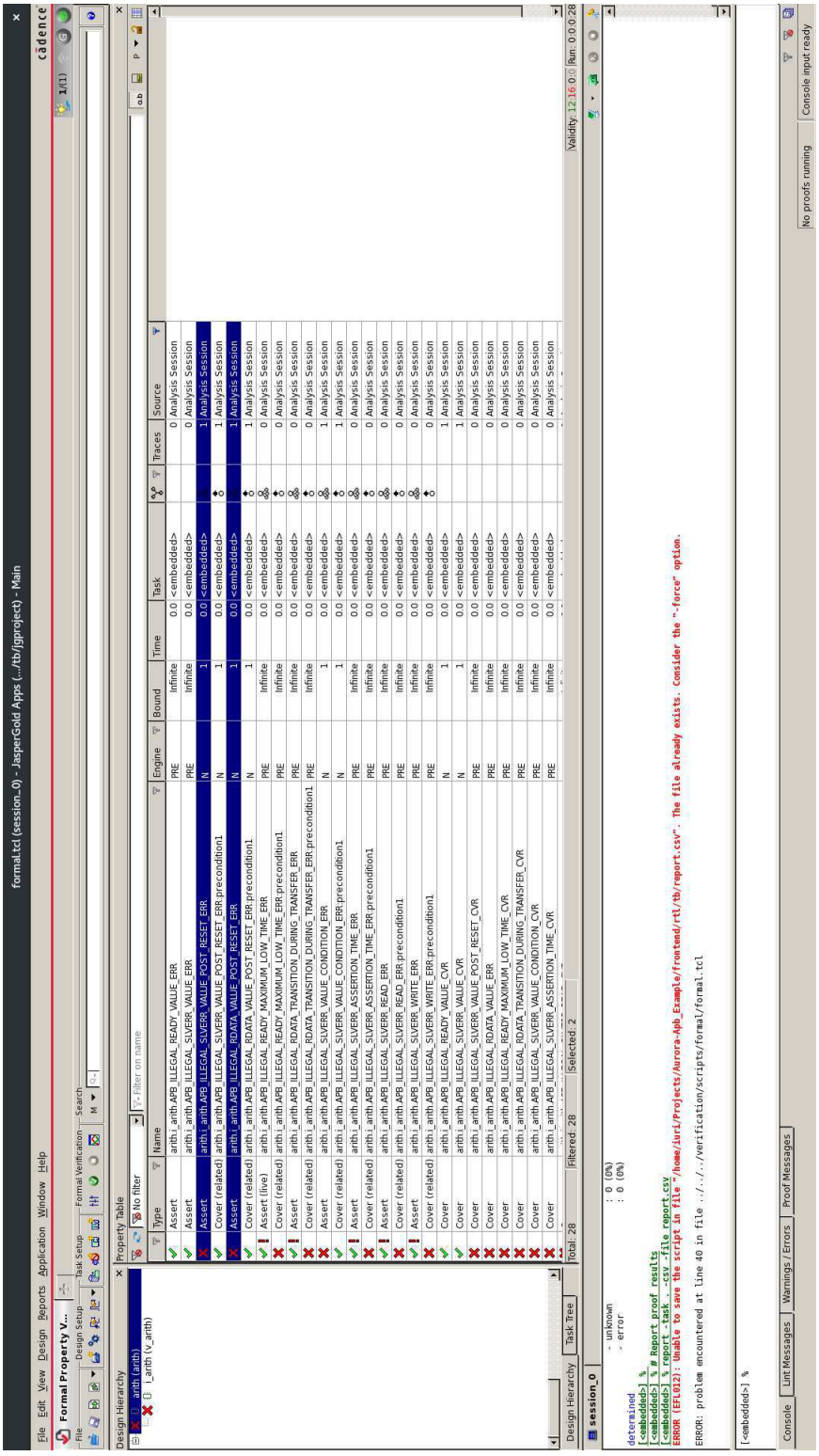
4.3 DESENVOLVIMENTO DO MÓDULO

Com os testes já escritos, o próximo passo é desenvolver os códigos fontes em si. A primeira etapa é ver se todos os testes estão falhando (primeira etapa do ciclo de iteração do TDD). A Figura 15 mostra a execução da ferramenta JasperGold quando não há nenhuma *assertion* passando.

Após essa etapa escreve-se os códigos para fazer com que os testes passem nas primeiras *assertions* que tratam da estabilidade dos sinais e seus valores após a aplicação do sinal de *reset*. A Figura 16 mostra os testes com as *assertions* passando.

O próximo passo foi implementar a primeira etapa do ciclo de produção e também da montagem da máquina de estados finitos do nosso *hardware*. Após o *reset* deve

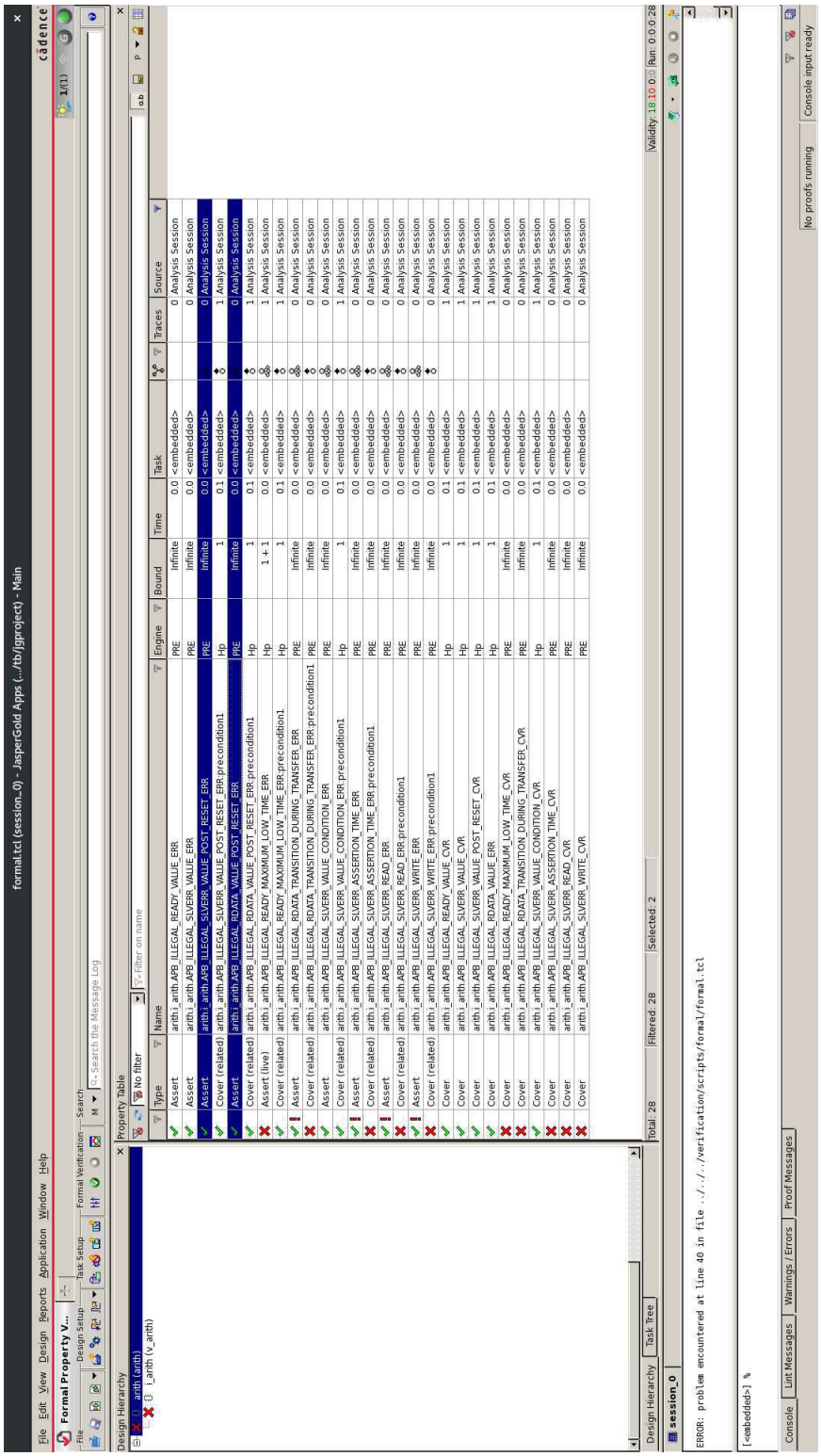
Figura 15 – Primeira execução dos testes na ferramenta.



Fonte: Próprio autor

haver alguma condição para levar o bloco ao estado de início de transferência, portanto implementou-se o primeiro estado que se chamou *IDLE* enquanto o bloco não e habilitado. O próximo passo foi atender as *assertions* de checagem de protocolo, principalmente a

Figura 16 – Primeira iteração de testes concluída com a aceitação da implementação pela ferramenta.

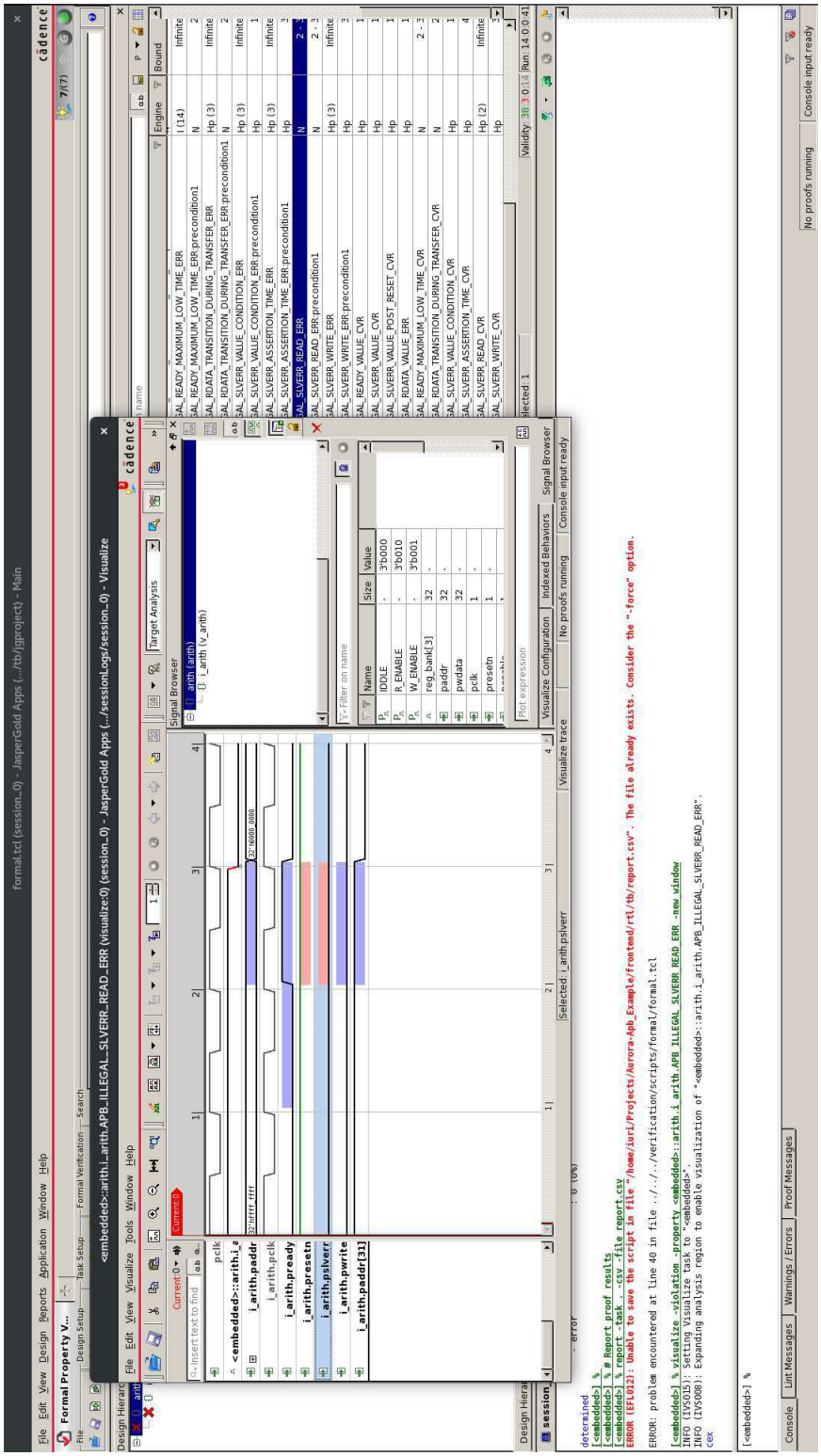


Fonte: Próprio autor

parte de transferências e a obediência aos valores de estabilidade de cada um dos sinais durante uma transferência. Em sequência a essa última etapa, para atender as *assertions* de leitura nos registros, criou-se mais um estado na máquina, chamado de "R_ENABLE".

Nota-se que alguns dos erros de implementação são indicados pela própria ferramenta como mostrado na Figura 17

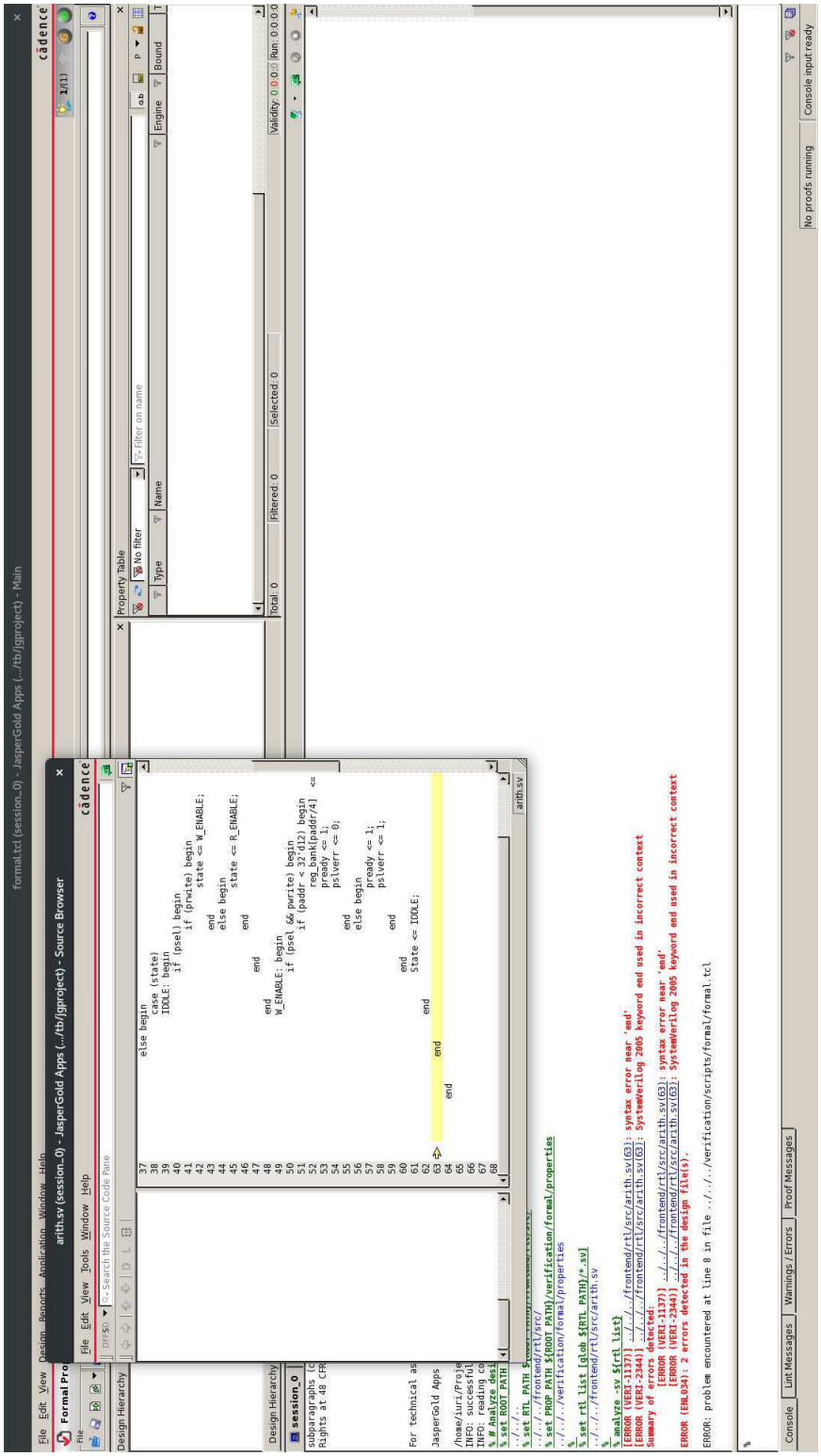
Figura 17 – Contra-prova da falha da *assertion* que indica uma operação incorreta do sinal *pslverr* durante uma leitura.



Fonte: Próprio autor

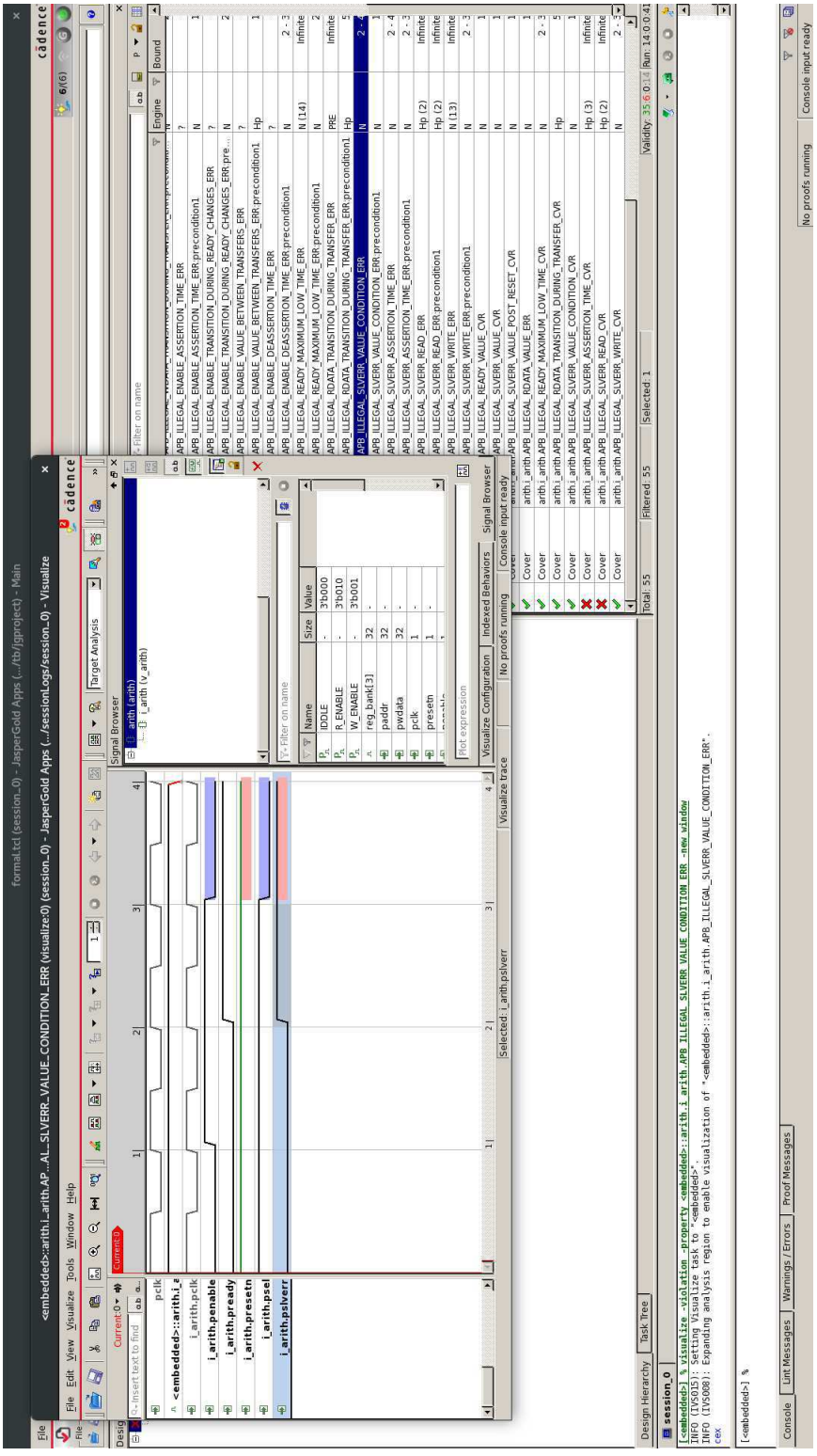
Após a etapa de atender as checagens de leitura, adicionou-se mais um estado a FSM, o "W_ENABLE". Com isso foi atendida também as *assertions* de escrita nos registros. Por fim para atender aos testes de operação dentro desse mesmo estágio foi feito a lógica combinacional que realiza a operação aritmética correspondente se o endereço de escrita foi o registro de comando e o bit de *start* estava ativo. A Figura 18 mostra um erro de sintaxe indicado pela ferramenta. A Figura 19 mostra um teste falho para o sinal *pslverr* em uma operação de escrita e por fim a Figura 20 mostra a finalização do desenvolvimento com os testes passando e corretamente cobertos pela ferramenta.

Figura 18 – Indicação de um erro de sintaxe. Nessa caso a palavra chave *end* foi utilizada incorretamente, já que a implementação correta utilizaria a palavrachave *endcase*.



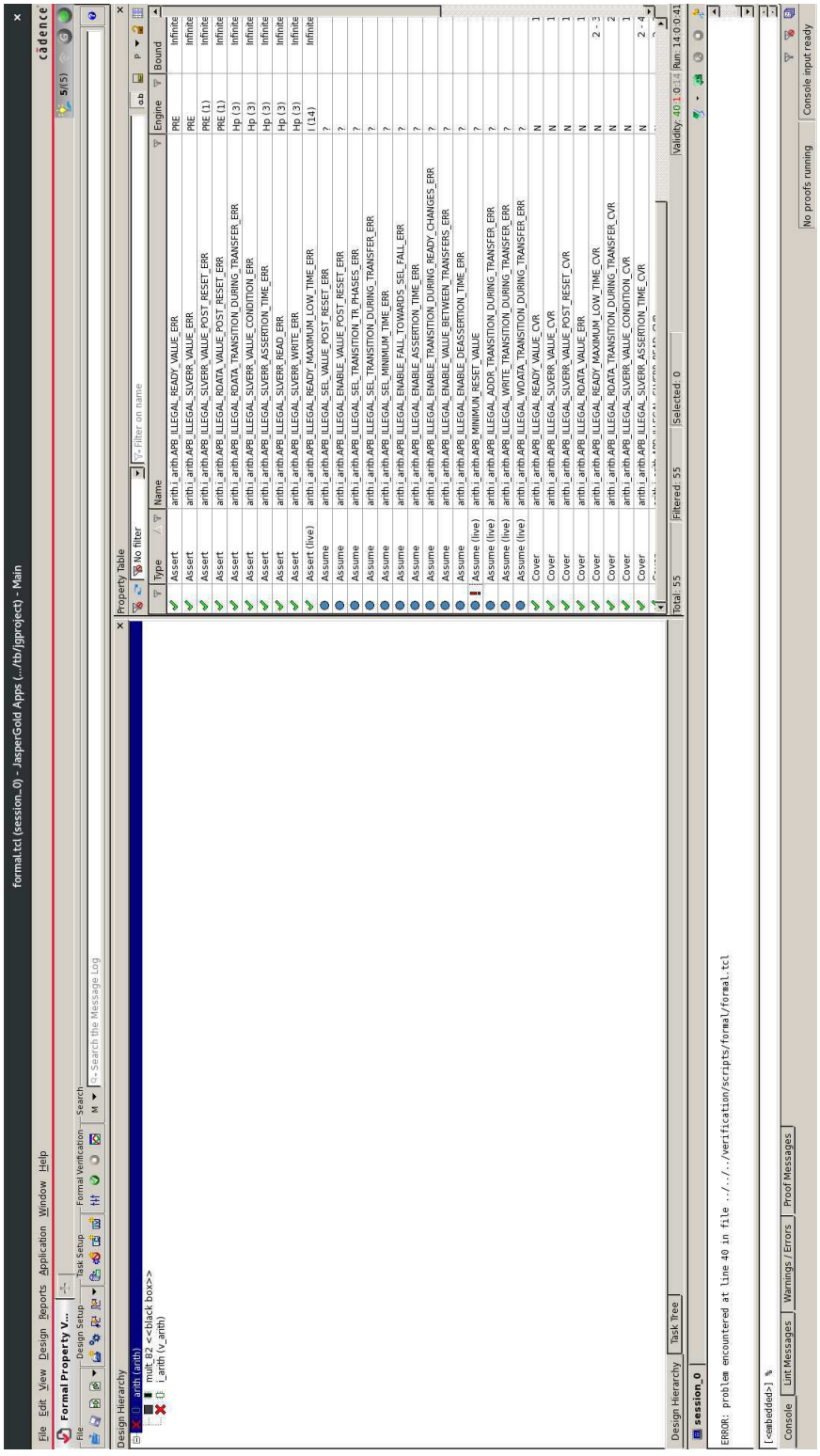
Fonte: Próprio autor

Figura 19 – Falha no sinal *pslverr* durante uma operação de escrita.



Fonte: Próprio autor

Figura 20 – Indicação da ferramenta que todos os testes foram bem-sucedidos.



Fonte: Próprio autor

5 CONSIDERAÇÕES FINAIS

A partir dos pressupostos indicados na seção de Embasamento Teórico foi possível desenvolver uma ferramenta habilitadora de utilização dos princípios do Desenvolvimento Dirigido por Testes (TDD). A ferramenta Aurora é capaz de gerar de maneira satisfatória o fluxo de referência adequado para as necessidades de um projeto de *hardware* acelerando o processo de desenvolvimento e organização da metodologia adotada.

A utilização da ferramenta Aurora permite que, aliada as práticas propostas nesse trabalho, o fluxo de desenvolvimento de *hardware* possa entregar produtos mais robustos, seguros e de maior qualidade escrita. A quantidade exaustiva de testes que são inerentes as práticas de TDD permitem que o desenvolvedor possa refatorar seu código e aperfeiçoa-lo sem o temor de problemas futuros de integração, ao mesmo tempo permite que os verificadores consigam aumentar o escopo dos testes realizados e também atinjam um maior nível de cobertura funcional. Com unidades já testadas previamente, a atividade de integração dos blocos se torna mais rápida e mais fácil, já que problemas dessa categoria serão facilmente percebido pelos testes. Além disso, a tarefa de *debug*, especialmente custosa no processo do desenvolvimento, também torna-se mais facilitada já que com os testes unitários as fontes dos problemas funcionais podem ser facilmente encontrada e corrigida.

É importante ressaltar que a prática do TDD em projetos de desenvolvimento de *hardware* possui suas limitações e requer alguns níveis de adaptação. O julgamento do engenheiro de verificação sobre quais as unidades precisam ser testadas tem sua importância aumentada, pois esse tipo de decisão irá guiar o desenvolvimento do produto, a elaboração de um plano de testes incompleto, provavelmente irá causar implementações incompletas e que provavelmente não serão testadas.

A utilização de práticas de automação de testes, aliadas ao TDD, tornam o desenvolvimento ainda mais produtivo e robusto. Essa tarefa de automação e execução de testes, assim como a adoção de praticas de *Continuos Integration* podem ser possibilidades de avanço da ferramenta descrita nesse trabalho. Além da adoção dessas práticas, outro ponto de avanço é a adição do suporte a mais ferramentas de EDA, já que esse trabalho fornece suporte apenas as ferramentas da Cadence Design Systems. Outra adição possível a ferramenta é a possibilidade de adicionar testes unitários durante a execução do projeto, de modo que estes se acoplem aos demais já desenvolvidos.

Por fim, o trabalho aqui apresentado encontra-se dentro de um contexto de desenvolvimento de *hardware* em especial a gerência de processos. A pesquisa e a criação de novas práticas de desenvolvimento, assim como o aperfeiçoamento das práticas já existentes, permite que toda a área avance em direção a produção de dispositivos cada vez mais

robustos, úteis e seguros impactando diretamente a sociedade como um todo.

REFERÊNCIAS

ARM. *APB protocol specification v1. 0 2003*. 2003. 41

ASTELS, D. *Test driven development: A practical guide*. [S.l.]: Prentice Hall Professional Technical Reference, 2003. 21

BERGERON, J. *Writing testbenches: functional verification of HDL models*. [S.l.]: Springer Science & Business Media, 2012. 26

CHU, P. P. *RTL hardware design using VHDL: coding for efficiency, portability, and scalability*. [S.l.]: John Wiley & Sons, 2006. 23, 24

ERDOGMUS, H.; MELNIK, G.; JEFFRIES, R. *Test-Driven Development*. 2010. 13

ERICKSON, A. Introducing uvm connect. *Mentor Graphics Verif. Horiz*, v. 8, n. 1, p. 6–12, 2012. 17

FOSTER, H. *2020 Wilson Research Group functional verification study*. [S.l.], 2020. 15, 28

GRENNING, J. W. *Test Driven Development for Embedded C*. [S.l.]: Pragmatic bookshelf, 2011. 21

IEEE Standard for SystemVerilog–Unified Hardware Design, Specification, and Verification Language. *IEEE Std 1800-2017 (Revision of IEEE Std 1800-2012)*, p. 1–1315, 2018. 12

IEEE Standard for Universal Verification Methodology Language Reference Manual. *IEEE Std 1800.2-2020 (Revision of IEEE Std 1800.2-2017)*, p. 1–458, 2020. 12

LAM, W. K. *Hardware Design Verification: Simulation and Formal Method-Based Approaches (Prentice Hall Modern Semiconductor Design Series)*. [S.l.]: Prentice Hall PTR, 2005. 23, 25, 27, 28

LIMA, J. W. F. Implementação em hardware de um gerador de ruído branco de alta performance. *Universidade Federal de Campina Grande. Centro de Engenharia Elétrica e Informática. Departamento de Engenharia Elétrica*, 2019. 19

OLIVEIRA, H. F. d. A. et al. Bvm: Reformulação da metodologia de verificação funcional verisc. Universidade Federal de Campina Grande, 2010. 33

PAVITHRAN, T.; BHAKTHAVATCHALU, R. Uvm based testbench architecture for logic sub-system verification. In: IEEE. *2017 International Conference on Technological Advancements in Power and Energy (TAP Energy)*. [S.l.], 2017. p. 1–5. 28

SILVA, K. R. D. et al. A methodology aimed at better integration of functional verification and rtl design. *Design Automation for Embedded Systems*, Springer, v. 10, n. 4, p. 285–298, 2005. 26, 27

SPEAR, C. *SystemVerilog for verification: a guide to learning the testbench language features*. [S.l.]: Springer Science & Business Media, 2008. 15

SUTHERLAND, S.; DAVIDMANN, S.; FLAKE, P. *SystemVerilog for Design Second Edition: A Guide to Using SystemVerilog for Hardware Design and Modeling*. [S.l.]: Springer Science & Business Media, 2006. 15

SYNOPSYS, I. *VCS DirectC Interface User Guide*. [S.l.], 2006. 17

VIJAYARAGHAVAN, S.; RAMANATHAN, M. *A practical guide for SystemVerilog assertions*. [S.l.]: Springer Science & Business Media, 2006. 18

WANG, R. Wrapping verilog bus functional model (bfm) and rtl as drivers in customized uvm vip using abstract classes. 2015. 26

APÊNDICES

A CÓDIGO FONTE DA FERRAMENTA

```
1  ##########################################################################
2  #  Project             : Aurora Integrated Reference Flow Generation
3  #
4  #  File Name           : aurora.py
5  #
6  #  Author              : Jose Iuri B. de Brito (XMEN LAB)
7  #
8  #  Purpose             : Main file of the application. Used to call the
9  #               Other files and functions.
10 ##########################################################################
11
12 import os
13 import sys, getopt
14 import argparse
15 from colorama import Fore
16 from pathlib import Path
17 import copy
18
19 class Clock:
20     def __init__(self, name, period):
21         self.name = name
22         self.period = period
23
24 class Reset:
25     def __init__(self, name, period, duration):
26         self.name = name
27         self.period = period
28         self.duration = duration
29
30 class Signal:
31     def __init__(self, name, type, io):
32         self.name = name
33         self.type = type
34         self.io = io
35         self.connect = None
36
37     def addConnection (self, connect):
38         self.connect = connect
39
40 class Field:
41     def __init__(self, name , type):
42         self.name = name
43         self.type = type
44
45 class Interface:
46     def __init__(self, name, instance, clock, reset):
47         self.name = name + '_interface'
48         self.instance = instance
49         self.signal = []
50         self.clock = clock
51         self.reset = reset
```

```
52
53      def addSignal(self, signal):
54          self.signal.append(signal)
55
56
57  class Transaction:
58      def __init__(self, name):
59          self.name = name + '_transaction'
60          self.field = []
61
62      def addField(self, field):
63          self.field.append(field)
64
65  class Agent:
66          def __init__(self, name, instance, interface, transaction, driver_policy,
        monitor_policy, type):
67              self.name = name
68              self.instance = instance
69              self.interface = interface
70              self.transaction = transaction
71              self.driver_policy = driver_policy
72              self.monitor_policy = monitor_policy
73              self.type = type
74              self.refmod = []
75              self.comp = []
76
77          def setInterface(self, Interface):
78              self.interface = Interface
79
80          def setTransaction(self, transaction):
81              self.transaction = transaction
82
83          def setDriverPolicy(self, diver_policy):
84              self.diver_policy = diver_policy
85
86          def setMonitorPolicy(self, monitor_policy):
87              self.monitor_policy = monitor_policy
88
89          def setMonitorType(self, type):
90              self.type = type
91
92          def setRefmodConn(self, refmod):
93              self.refmod = refmod
94
95          def setCompConn(self, comp):
96              self.comp = comp
97
98          def writeInterface(self, outputdir):
99
100             with open(os.path.dirname(os.path.realpath(__file__)) + '/../src/
        templates/verification/general_agent/general_interface.tb', 'r') as file:
101                 tbInterface=file.read()
102
103             tbInterface = tbInterface.replace('|-INTERFACE-|', self.interface.
        name)
104
105             tbInterface = tbInterface.replace('|-CLOCK-|', self.interface.clock.
```

```
name)
106
107             tbInterface = tbInterface.replace('|-RESET-|', self.interface.reset.
    name)
108
109             # SIGNALS
110
111             for uSignal in self.interface.signal:
112                 tbInterface = tbInterface.replace('|-SIGNAL_NAME-|', uSignal.type
     + ' ' + uSignal.name + ';\n\t|-SIGNAL_NAME-|')
113
114             # Cleaning residual tags
115             tbInterface = tbInterface.replace('|-SIGNAL_NAME-|', '')
116
117             interface_file = open(outputdir + '/' + self.interface.name + ".sv",
    "wt")
118             n = interface_file.write(tbInterface)
119             interface_file.close()
120
121         def writeTransaction(self, outputdir):
122
123             with open(os.path.dirname(os.path.realpath(__file__)) + '/../src/
    templates/verification/general_agent/general_transaction.tb', 'r') as file:
124                 tbTransaction=file.read()
125
126             tbTransaction = tbTransaction.replace('|-TRANSACTION-|', self.
    transaction.name)
127
128             # FIELDS
129
130             for uField in self.transaction.field:
131                 tbTransaction = tbTransaction.replace('|-FIELD_NAME-|', uField.
    type + ' ' + uField.name + ';\n\t|-FIELD_NAME-|') # INPUT DECLARATION
132                 tbTransaction = tbTransaction.replace('|-FIELD_MACRO-|','`
    uvm_field_int(' + uField.name + ', UVM_ALL_ON|UVM_HEX)\n\t\t|-FIELD_MACRO-|')
    #MACRO DECLARATION
133                 tbTransaction = tbTransaction.replace('|-FIELD_NAME_S-| = %h,',
    uField.name + ', |-FIELD_NAME_S-| = %h,') # convert2string DECLARATION
134                 tbTransaction = tbTransaction.replace('|-FIELD_NAME_S-|,', uField
    .name + ', |-FIELD_NAME_S-|,') # convert2string DECLARATION
135
136             # Cleaning residual tags
137             tbTransaction = tbTransaction.replace('|-FIELD_NAME-|', '')
138             tbTransaction = tbTransaction.replace('|-FIELD_MACRO-|', '')
139             tbTransaction = tbTransaction.replace(', |-FIELD_NAME_S-| = %h,', '')
140             tbTransaction = tbTransaction.replace(', |-FIELD_NAME_S-|,', '')
141
142
143             transaction_file = open(outputdir + '/' + self.name + "_transaction.
    sv", "wt")
144             n = transaction_file.write(tbTransaction)
145             transaction_file.close()
146
147         def writeAgent(self, outputdir):
148
149             with open(os.path.dirname(os.path.realpath(__file__)) + '/../src/
    templates/verification/general_agent/general_agent.tb', 'r') as file:
```

```
150                 tbAgent=file.read()
151
152             tbAgent = tbAgent.replace('|-AGENT-|', self.name)
153             tbAgent = tbAgent.replace('|-TRANSACTION-|', self.transaction.name)
154
155             if (self.type == 'input'):
156                 tbAgent = tbAgent.replace('uvm_analysis_port #(|-TRANSACTION-|)
    ap_resp;', '')
157                 tbAgent = tbAgent.replace('ap_resp = new("ap_resp", this);', '')
158                 tbAgent = tbAgent.replace('mon.resp.connect(ap_resp);', '')
159             elif (self.type == 'output'):
160                 tbAgent = tbAgent.replace('uvm_analysis_port #(|-TRANSACTION-|)
    ap_req;', '')
161                 tbAgent = tbAgent.replace('ap_req = new("ap_req", this);', '')
162                 tbAgent = tbAgent.replace('mon.req.connect(ap_req);', '')
163                 tbAgent = tbAgent.replace('mon.req.connect(cov.collected_port);',
     '//OUTPUT COVERAGE')
164             else:
165                 pass
166
167
168             agent_file = open(outputdir + '/' + self.name + "_agent.sv", "wt")
169             n = agent_file.write(tbAgent)
170             agent_file.close()
171
172         def writeDriver(self, outputdir):
173             with open(os.path.dirname(os.path.realpath(__file__)) + '/../src/
    templates/verification/general_agent/general_driver.tb', 'r') as file:
174                 tbDriver=file.read()
175
176             tbDriver = tbDriver.replace('|-AGENT-|', self.name)
177             tbDriver = tbDriver.replace('|-TRANSACTION-|', self.transaction.name)
178             tbDriver = tbDriver.replace('|-INTERFACE-|', self.interface.name)
179             tbDriver = tbDriver.replace('|-INTERFACE_INSTANCE-|', self.interface.
    instance)
180
181             tbDriver = tbDriver.replace('|-DRIVER_POLICY-|', self.driver_policy.
    replace('\n','\n\t'))
182
183             driver_file = open(outputdir + '/' + self.name + "_driver.sv", "wt")
184             n = driver_file.write(tbDriver)
185             driver_file.close()
186
187         def writeMonitor(self, outputdir):
188             with open(os.path.dirname(os.path.realpath(__file__)) + '/../src/
    templates/verification/general_agent/general_monitor.tb', 'r') as file:
189                 tbMonitor=file.read()
190
191             tbMonitor = tbMonitor.replace('|-AGENT-|', self.name)
192             tbMonitor = tbMonitor.replace('|-INTERFACE-|', self.interface.name)
193             tbMonitor = tbMonitor.replace('|-INTERFACE_INSTANCE-|', self.
    interface.instance)
194
195             tbMonitor = tbMonitor.replace('|-MONITOR_POLICY-|', self.
    monitor_policy.replace('\n','\n\t\t'))
196
197             if (self.type == 'input'):
```

```
198                 tbMonitor = tbMonitor.replace('this.resp.write(transCollected);',
     '')
199                 tbMonitor = tbMonitor.replace('uvm_analysis_port #(|-TRANSACTION
    -|) resp;', '')
200                 tbMonitor = tbMonitor.replace('this.resp = new("resp", this);', '
    ')
201             elif (self.type == 'output'):
202                 tbMonitor = tbMonitor.replace('this.req.write(transCollected);',
    '')
203                 tbMonitor = tbMonitor.replace('uvm_analysis_port #(|-TRANSACTION
    -|) req;', '')
204                 tbMonitor = tbMonitor.replace('this.req = new("req", this);', '')
205             else:
206                 pass
207
208             tbMonitor = tbMonitor.replace('|-TRANSACTION-|', self.transaction.
    name)
209
210             monitor_file = open(outputdir + '/' + self.name + "_monitor.sv", "wt"
    )
211             n = monitor_file.write(tbMonitor)
212             monitor_file.close()
213
214         def writeCoverage(self, outputdir):
215
216             with open(os.path.dirname(os.path.realpath(__file__)) + '/../src/
    templates/verification/general_agent/general_coverage.tb', 'r') as file:
217                 tbCoverage=file.read()
218
219             tbCoverage = tbCoverage.replace('|-AGENT-|', self.name)
220             tbCoverage = tbCoverage.replace('|-TRANSACTION-|', self.transaction.
    name)
221
222             coverage_file = open(outputdir + '/' + self.name + "_coverage.sv", "
    wt")
223             n = coverage_file.write(tbCoverage)
224             coverage_file.close()
225
226         def writeAgentAll(self, outputdir):
227             self.writeInterface(outputdir)
228             self.writeTransaction(outputdir)
229             self.writeAgent(outputdir)
230             self.writeDriver(outputdir)
231             self.writeMonitor(outputdir)
232             self.writeCoverage(outputdir)
233             self.writeAgent(outputdir)
234
235 class Port:
236     def __init__(self, name, direction, origin, transaction, endComp):
237         self.name = name
238         self.direction = direction
239         self.origin = origin
240         self.transaction = transaction
241         self.endComp = endComp
242
243     def setEndComp(self, endComp):
244         self.endComp = endComp
```

```
245
246 class Refmod:
247     def __init__(self, name, instance, policy, port_out, destination):
248         self.name = name + '_refmod'
249         self.instance = instance + '_rfm'
250
251         self.port_in = []
252         self.port_out = [port_out]
253
254         self.policy = policy
255
256         self.destination = [destination]
257
258
259     def addPortIn (self, port):
260         self.port_in.append(port)
261
262     def addPortOut (self, port):
263         self.port_out.append(port)
264
265     def addDestination (self, destination):
266         self.destination.append(destination)
267
268     def writeRefmod(self, outputdir):
269         with open(os.path.dirname(os.path.realpath(__file__)) + '/../src/
    templates/verification/scoreboard/refmod.tb', 'r') as file:
270             tbRefmod=file.read()
271
272         tbRefmod = tbRefmod.replace('|-REFMOD-|', self.name)
273
274         for idx, uPort in enumerate(self.port_in):
275             tbRefmod = tbRefmod.replace('|-INPUT_TRANSA-|', uPort.transaction.
    name + ' ' +  'req_' + str(idx) + ';\n\t|-INPUT_TRANSA-|')
276
277         for idx, uPort in enumerate(self.port_out):
278             tbRefmod = tbRefmod.replace('|-OUTPUT_TRANSA-|', uPort.transaction.
    name + ' ' +  'resp_' + str(idx) + ';\n\t|-OUTPUT_TRANSA-|')
279
280         for idx, uPort in enumerate(self.port_in):
281             tbRefmod = tbRefmod.replace('|-TRANSA_CREATION-|', 'req_'+ str(idx) +
     ' = new("req_' + str(idx) + '")' + ';\n\t\t|-TRANSA_CREATION-|')
282
283         for idx, uPort in enumerate(self.port_out):
284             tbRefmod = tbRefmod.replace('|-TRANSA_CREATION-|', 'resp_'+ str(idx)
    + ' = new("resp_' + str(idx) + '")' + ';\n\t\t|-TRANSA_CREATION-|')
285
286         for idx, uPort in enumerate(self.port_in):
287             tbRefmod = tbRefmod.replace('|-INPUT_PORT-|', 'uvm_tlm_analysis_fifo
    #(' + uPort.transaction.name + ') ' + 'from_' + uPort.origin + ';\n\t|-
    INPUT_PORT-|')
288
289         for idx, uPort in enumerate(self.port_out):
290             tbRefmod = tbRefmod.replace('|-OUTPUT_PORT-|', 'uvm_analysis_export
    #(' + uPort.transaction.name + ') ' + uPort.origin + '_to_'+ uPort.direction +
     ';\n\t|-OUTPUT_PORT-|')
291
292         for idx, uPort in enumerate(self.port_in):
```

```
293             tbRefmod = tbRefmod.replace('|-PORT_CREATION-|', 'from_' + uPort.
    origin + ' = new("' + 'from_' + uPort.origin + '", this)' + ';\n\t\t|-
    PORT_CREATION-|')
294 
295         for idx, uPort in enumerate(self.port_out):
296             tbRefmod = tbRefmod.replace('|-PORT_CREATION-|', uPort.origin + '_to_
    '+ uPort.direction + ' = new("' + uPort.origin + '_to_'+ uPort.direction + '",
     this)' + ';\n\t\t|-PORT_CREATION-|')
297 
298         tbRefmod = tbRefmod.replace('|-REFMOD_POLICY-|', self.policy.replace('\n'
    ,'\n\t\t'))
299 
300         # Cleanup
301         tbRefmod = tbRefmod.replace('|-INPUT_TRANSA-|', '')
302         tbRefmod = tbRefmod.replace('|-OUTPUT_TRANSA-|', '')
303         tbRefmod = tbRefmod.replace('|-TRANSA_CREATION-|', '')
304         tbRefmod = tbRefmod.replace('|-INPUT_PORT-|', '')
305         tbRefmod = tbRefmod.replace('|-OUTPUT_PORT-|', '')
306         tbRefmod = tbRefmod.replace('|-PORT_CREATION-|', '')
307 
308         refmod_file = open(outputdir + '/' + self.name + ".sv", "wt")
309         n = refmod_file.write(tbRefmod)
310         refmod_file.close()
311 
312 class Comparator:
313 
314     def __init__(self, name, instance, transaction):
315         self.name = name
316         self.transaction = transaction
317         self.instance = instance
318         self.name = 'uvm_in_order_class_comparator #('+ self.transaction.name + '
    )'
319 
320 class Scoreboard:
321 
322     def __init__(self, name, instance):
323         self.name = name
324         self.instance = instance
325         self.port = []
326         self.refmod = []
327         self.comp = []
328 
329     def addPort(self, port):
330         self.port.append(port)
331 
332     def addComp(self, comp):
333         self.comp.append(comp)
334 
335     def addRefmod(self, refmod):
336         self.refmod.append(refmod)
337 
338     def writeScoreboard(self, outputdir):
339 
340         with open(os.path.dirname(os.path.realpath(__file__)) + '/../src/
    templates/verification/scoreboard/scoreboard.tb', 'r') as file:
341             tbScoreboard=file.read()
342 
```

```
343         tbScoreboard = tbScoreboard.replace('|-SCOREBOARD-|', self.name)
344 
345         for idx,uRefmod in enumerate(self.refmod):
346             tbScoreboard = tbScoreboard.replace('|-REFMOD-|', uRefmod.name + ' '
    +  uRefmod.instance + ';\n\t|-REFMOD-|')
347 
348         for idx,uComp in enumerate(self.comp):
349             tbScoreboard = tbScoreboard.replace('|-COMP-|', uComp.name + ' ' +
    uComp.instance  + ';\n\t|-COMP-|')
350 
351         for idx,uPort in enumerate(self.port):
352             tbScoreboard = tbScoreboard.replace('|-PORTS-|', 'uvm_analysis_port
    #(' + uPort.transaction.name + ') ' + uPort.origin + '_to_' + uPort.direction
    + ';\n\t|-PORTS-|')
353 
354         for idx,uRefmod in enumerate(self.refmod):
355             tbScoreboard = tbScoreboard.replace('|-REFMOD_CREATION-|', uRefmod.
    instance + ' = ' + uRefmod.name + '::type_id::create("' + uRefmod.instance + '
    ", this)' +  ';\n\t\t|-REFMOD_CREATION-|')
356 
357         for idx,uComp in enumerate(self.comp):
358             tbScoreboard = tbScoreboard.replace('|-COMPARATOR_CREATION-|', uComp.
    instance + ' = ' + uComp.name + '::type_id::create("' + uComp.instance + '",
    this)' +  ';\n\t\t|-COMPARATOR_CREATION-|')
359 
360         for idx,uPort in enumerate(self.port):
361             tbScoreboard = tbScoreboard.replace('|-PORTS_CREATION-|', uPort.
    origin + '_to_' + uPort.direction + '= new("' + uPort.origin + '_to_' + uPort.
    direction + '", this)' + ';\n\t\t|-PORTS_CREATION-|')
362 
363         for idx,uPort in enumerate(self.port):
364             if uPort.endComp == 0:
365                 tbScoreboard = tbScoreboard.replace('|-CONNECTIONS-|', uPort.
    origin + '_to_' + uPort.direction + '.connect(' + uPort.direction + '.' 'from_
    ' + uPort.origin + ');\n\t\t|-CONNECTIONS-|')
366             elif uPort.endComp == 1:
367                 tbScoreboard = tbScoreboard.replace('|-CONNECTIONS-|', uPort.
    origin + '_to_' + uPort.direction + '.connect(' + uPort.direction + '.' '
    before_export' + ');\n\t\t|-CONNECTIONS-|')
368             else:
369                 for idx,uRefmod in enumerate(self.refmod):
370                     if uPort.direction + '_rfm' == uRefmod.instance:
371                         tbScoreboard = tbScoreboard.replace('|-CONNECTIONS-|',
    uPort.origin + '_to_' + uPort.direction + '.connect(' + uPort.direction + '
    _rfm.' 'from_' + uPort.origin + '.analysis_export);\n\t\t|-CONNECTIONS-|')
372                 for idy,uComp in enumerate(self.comp):
373                     if uPort.direction == uComp.instance:
374                         tbScoreboard = tbScoreboard.replace('|-CONNECTIONS-|',
    uPort.origin + '_to_' + uPort.direction + '.connect(' + uPort.direction + '.'
    'before_export' + ');\n\t\t|-CONNECTIONS-|')
375 
376         # Refmod Port Creation
377         for idx,uPort in enumerate(self.port):
378             for idy,uRefmod in enumerate(self.refmod):
379                 if uPort.direction == uRefmod.instance:
380                     port_aux = Port('rfm_in', uPort.direction, uPort.origin ,
    uPort.transaction, 0)
```

```
381                     self.refmod[idy].addPortIn(port_aux)
382
383         for idx,uRefmod in enumerate(self.refmod):
384             for idz, uPort_out in enumerate(uRefmod.port_out):
385                 if (uPort_out.endComp == 0):
386                     for idy,uRefmod_out in enumerate(self.refmod):
387                         if (uPort_out.direction + '_rfm') == (uRefmod_out.
    instance):
388                             port_aux = Port('rfm_in', uPort_out.direction,
    uPort_out.origin , uPort_out.transaction, 0)
389                             self.refmod[idy].addPortIn(port_aux)
390
391         # Refmod -> Comp connection
392         for idy,uRefmod in enumerate(self.refmod):
393             for idx,uPort in enumerate(uRefmod.port_out):
394                 if idx == 0:
395                     tbScoreboard = tbScoreboard.replace('|-CONNECTIONS-|',
    uRefmod.instance + '.' + uPort.origin + '_to_'+ uPort.direction + '.connect('
    + uPort.direction + '.' 'after_export' + ');\n\t\t|-CONNECTIONS-|')
396                 else:
397                     tbScoreboard = tbScoreboard.replace('|-CONNECTIONS-|',
    uRefmod.instance + '.' + uPort.origin + '_to_'+ uPort.direction + '.connect('
    + uPort.direction + '_rfm.' + 'from_' + uPort.origin + ');\n\t\t|-CONNECTIONS
    -|')
398
399         # CLEANUP
400         tbScoreboard = tbScoreboard.replace('|-REFMOD-|', '')
401         tbScoreboard = tbScoreboard.replace('|-COMP-|', '')
402         tbScoreboard = tbScoreboard.replace('|-PORTS-|', '')
403         tbScoreboard = tbScoreboard.replace('|-REFMOD_CREATION-|', '')
404         tbScoreboard = tbScoreboard.replace('|-COMPARATOR_CREATION-|', '')
405         tbScoreboard = tbScoreboard.replace('|-PORTS_CREATION-|', '')
406         tbScoreboard = tbScoreboard.replace('|-CONNECTIONS-|', '')
407
408         scoreboard_file = open(outputdir + '/' + self.name + "_scoreboard.sv", "
    wt")
409         n = scoreboard_file.write(tbScoreboard)
410         scoreboard_file.close()
411
412 class Vip:
413     def __init__(self, name, instance, interface, file_package):
414         self.name = name + '_env'
415         self.interface = interface
416         self.instance = instance
417         self.file_package = file_package
418         self.port = []
419         self.include = []
420
421     def addPort(self, tlm_port):
422         self.port.append(tlm_port)
423
424     def addInclude(self, include_file):
425         self.include.append(include_file)
426
427 class Env:
428     def __init__(self, name, instance, port, scoreboard):
429         self.name = name + '_env'
```

```
430         self.instance = instance
431         self.port = [port]
432         self.scoreboard = scoreboard
433         self.agent = []
434         self.vip = []
435 
436     def addPort(self, port):
437         self.port.append(port)
438 
439     def setScoreboard (self, scoreboard):
440         self.scoreboard = scoreboard
441 
442     def addAgent(self, agent):
443         self.agent.append(agent)
444 
445     def addVip(self, vip):
446         self.vip.append(vip)
447 
448     def writeEnv(self, outputdir):
449 
450         with open(os.path.dirname(os.path.realpath(__file__)) + '/../src/
    templates/verification/env.tb', 'r') as file:
451             tbEnv=file.read()
452 
453         tbEnv = tbEnv.replace('|-ENV-|', self.name)
454 
455         for idx,agent in enumerate(self.agent):
456             tbEnv = tbEnv.replace('|-AGENT-|', agent.name + '_agent ' +  agent.
    instance + ';\n\t|-AGENT-|')
457 
458         tbEnv = tbEnv.replace('|-SCOREBOARD-|', self.scoreboard.name + '
    _scoreboard ' +  self.scoreboard.instance  + ';\n\t|-SCOREBOARD-|')
459 
460         for idx,agent in enumerate(self.agent):
461             tbEnv = tbEnv.replace('|-AGENT_CREATION-|', agent.instance + ' = ' +
    agent.name + '_agent' + '::type_id::create("' + agent.instance + '", this)' +
     ';\n\t\t|-AGENT_CREATION-|')
462 
463         tbEnv = tbEnv.replace('|-SCOREBOARD_CREATION-|', self.scoreboard.instance
     + ' = ' + self.scoreboard.name + '_scoreboard' + '::type_id::create("' + self
    .scoreboard.instance + '", this)' +  ';\n\t\t|-SCOREBOARD_CREATION-|')
464 
465         for idx,uVip in enumerate(self.vip):
466             tbEnv = tbEnv.replace('|-VIP-|', uVip.name + ' ' +  uVip.instance + '
    ;\n\t|-VIP-|')
467             tbEnv = tbEnv.replace('|-VIP_CREATION-|', uVip.instance + ' = ' +
    uVip.name + '::create("' + uVip.instance + '", this)' +  ';\n\t\t|-
    VIP_CREATION-|')
468 
469         for uAgent in (self.agent):
470             if (uAgent.type == 'input'):
471                 envPortAgtRfm = Port(uAgent.instance + '_to_' + uAgent.refmod,
    uAgent.refmod, uAgent.instance, uAgent.transaction, 0)
472                 self.scoreboard.addPort(envPortAgtRfm)
473                 tbEnv = tbEnv.replace('|-CONNECTIONS-|', uAgent.instance + '.' +
    'ap_req' + '.connect(' + self.scoreboard.instance + '.' + uAgent.instance + '
    _to_' + uAgent.refmod + ');\n\t\t|-CONNECTIONS-|')
```

```
474             elif (uAgent.type == 'output'):
475                 envPortAgtComp = Port(uAgent.instance + '_to_' + uAgent.comp,
    uAgent.comp, uAgent.instance, uAgent.transaction, 1)
476                 self.scoreboard.addPort(envPortAgtComp)
477                 tbEnv = tbEnv.replace('|-CONNECTIONS-|', uAgent.instance + '.' +
    'ap_resp' + '.connect(' + self.scoreboard.instance + '.' + uAgent.instance + '
    _to_' + uAgent.comp + ');\n\t\t|-CONNECTIONS-|')
478             else:
479                 envPortAgtRfm = Port(uAgent.instance + '_to_' + uAgent.refmod,
    uAgent.refmod, uAgent.instance, uAgent.transaction, 0)
480                 self.scoreboard.addPort(envPortAgtRfm)
481                 tbEnv = tbEnv.replace('|-CONNECTIONS-|', uAgent.instance + '.' +
    'ap_req' + '.connect(' + self.scoreboard.instance + '.' + uAgent.instance + '
    _to_' + uAgent.refmod + ');\n\t\t|-CONNECTIONS-|')
482                 envPortAgtComp = Port(uAgent.instance + '_to_' + uAgent.comp,
    uAgent.comp, uAgent.instance, uAgent.transaction, 1)
483                 self.scoreboard.addPort(envPortAgtComp)
484                 tbEnv = tbEnv.replace('|-CONNECTIONS-|', uAgent.instance + '.' +
    'ap_resp' + '.connect(' + self.scoreboard.instance + '.' + uAgent.instance + '
    _to_' + uAgent.comp + ');\n\t\t|-CONNECTIONS-|')
485
486         for uVip in self.vip:
487             for uPort in uVip.port:
488                 envPortVipC = Port(uPort.origin + '_to_' + uPort.direction, uPort
    .direction, uPort.origin, uPort.transaction, 3)
489                 self.scoreboard.addPort(copy.copy(envPortVipC))
490                 tbEnv = tbEnv.replace('|-CONNECTIONS-|', uVip.instance + '.' +
    uPort.name + '.connect(' + self.scoreboard.instance + '.' + uPort.origin + '
    _to_' + uPort.direction + ');\n\t\t|-CONNECTIONS-|')
491
492
493         # CLEANUP
494         tbEnv = tbEnv.replace('|-AGENT-|', '')
495         tbEnv = tbEnv.replace('|-VIP-|', '')
496         tbEnv = tbEnv.replace('|-REFMOD-|', '')
497         tbEnv = tbEnv.replace('|-SCOREBOARD-|', '')
498         tbEnv = tbEnv.replace('|-SCOREBOARD_CREATION-|', '')
499         tbEnv = tbEnv.replace('|-AGENT_CREATION-|', '')
500         tbEnv = tbEnv.replace('|-VIP_CREATION-|', '')
501         tbEnv = tbEnv.replace('|-SCOREBOARD_CREATION-|', '')
502         tbEnv = tbEnv.replace('|-CONNECTIONS-|', '')
503
504
505         env_file = open(outputdir + '/' + self.name + ".sv", "wt")
506         n = env_file.write(tbEnv)
507         env_file.close()
508
509 class Sequence:
510
511     def __init__(self, name, agent, transaction):
512         self.name = name + '_sequence'
513         self.agent = [agent]
514         self.transaction = transaction
515
516     def writeSequence(self, outputdir):
517
518             with open(os.path.dirname(os.path.realpath(__file__)) + '/../src/
```

```
    templates/verification/sequence.tb', 'r') as file:
519                 tbSequence=file.read()
520
521             tbSequence = tbSequence.replace('|-SEQUENCE-|', self.name)
522             tbSequence = tbSequence.replace('|-TRANSACTION-|', self.transaction.
    name)
523
524             sequence_file = open(outputdir + '/' + self.name + ".sv", "wt")
525             n = sequence_file.write(tbSequence)
526             sequence_file.close()
527
528 class Test:
529     def __init__(self, env, name):
530         self.name = name + '_test'
531         self.env = env
532         self.sequence = []
533
534     def addSequence(self, sequence):
535         self.sequence.append(sequence)
536
537     def writeTest(self, outputdir):
538
539         with open(os.path.dirname(os.path.realpath(__file__)) + '/../src/
    templates/verification/test.tb', 'r') as file:
540             tbTest=file.read()
541
542         tbTest = tbTest.replace('|-TEST-|', self.name)
543
544         tbTest = tbTest.replace('|-ENV-|', self.env.name + ' env_h;')
545
546         for idx,sequence in enumerate(self.sequence):
547             tbTest = tbTest.replace('|-SEQUENCE-|', sequence.name + ' seq_' + str
    (idx) + ';\n\t|-SEQUENCE-|')
548
549         tbTest = tbTest.replace('|-ENV_CREATION-|', 'env_h' + ' = ' + self.env.
    name + '::type_id::create("env_h", this)' +  ';\n\t\t|-ENV_CREATION-|')
550         for idx,sequence in enumerate(self.sequence):
551             tbTest = tbTest.replace('|-SEQUENCE_CREATION-|', 'seq_' + str(idx) +
    ' = ' + sequence.name + '::type_id::create("'+ 'seq_' + str(idx) +'", this)' +
      ';\n\t\t|-SEQUENCE_CREATION-|')
552             for agent in sequence.agent:
553                 tbTest = tbTest.replace('|-SEQUENCE_START-|', 'seq_' + str(idx) +
     '.start(env_h.' + agent.instance + '.sqr)' + ';\n\t\t|-SEQUENCE_START-|')
554
555         # CLEANUP
556         tbTest = tbTest.replace('|-TEST-|', '')
557         tbTest = tbTest.replace('|-ENV-|', '')
558         tbTest = tbTest.replace('|-SEQUENCE-|', '')
559         tbTest = tbTest.replace('|-ENV_CREATION-|', '')
560         tbTest = tbTest.replace('|-SEQUENCE_CREATION-|', '')
561         tbTest = tbTest.replace('|-SEQUENCE_START-|', '')
562
563
564         test_file = open(outputdir + '/' + self.name + ".sv", "wt")
565         n = test_file.write(tbTest)
566         test_file.close()
567
```

```
568 class Module:
569     def __init__(self, name):
570         self.name = name
571         self.clock = []
572         self.reset = []
573         self.signal = []
574         self.interface = []
575         self.env = []
576         self.agent = []
577         self.vip = []
578         self.test = []
579 
580     def addSignal(self, signal):
581         self.signal.append(signal)
582 
583     def addInterface(self, interface):
584         self.interface.append(interface)
585 
586     def addClock(self, clock):
587         self.clock.append(clock)
588 
589     def addReset(self, reset):
590         self.reset.append(reset)
591 
592     def addEnv(self, env):
593         self.env.append(env)
594 
595     def addAgent(self, agent):
596         self.agent.append(agent)
597 
598     def addVip(self, vip):
599         self.vip.append(vip)
600 
601     def addTest(self, test):
602         self.test.append(test)
603 
604     def writeWrapper(self, outputdir):
605         with open(os.path.dirname(os.path.realpath(__file__)) + '/../src/
    templates/verification/wrapper.tb', 'r') as file:
606             tbWrapper=file.read()
607 
608         tbWrapper = tbWrapper.replace('|-MODULE-|', self.name)
609 
610         for idx,uAgent in enumerate(self.agent):
611             tbWrapper = tbWrapper.replace('|-INTERFACE-|', uAgent.interface.name
    + ' ' + uAgent.interface.instance + '_if' +',\n\t|-INTERFACE-|')
612 
613         for idx,uVip in enumerate(self.vip):
614             tbWrapper = tbWrapper.replace('|-INTERFACE-|', uVip.interface.name +
    ' ' + uVip.interface.instance + '_if' +',\n\t|-INTERFACE-|')
615 
616         for idx,uClock in enumerate(self.clock):
617             tbWrapper = tbWrapper.replace('|-INTERFACE-|', 'input ' + uClock.name
    +',\n\t|-INTERFACE-|')
618 
619         for idx,uReset in enumerate(self.reset):
620             tbWrapper = tbWrapper.replace('|-INTERFACE-|', 'input ' + uReset.name
```

```
 +',\n\t|-INTERFACE-|')
621
622         for idx,uAgent in enumerate(self.agent):
623             for idy,busSig in enumerate(uAgent.interface.signal):
624                     if busSig.connect is not None:
625                         tbWrapper = tbWrapper.replace('|-CONNECTIONS-|', '.'+
    busSig.connect + '(' + uAgent.interface.instance + '_if.' + busSig.name +') '
    +',\n\t\t|-CONNECTIONS-|')
626                     else:
627                         tbWrapper = tbWrapper.replace('|-CONNECTIONS-|', '.'+
    busSig.name + '(' + uAgent.interface.instance + '_if.' + busSig.name +') ' +'
    ,\n\t\t|-CONNECTIONS-|')
628
629         for idx,uVip in enumerate(self.vip):
630             for idy,busSig in enumerate(uVip.interface.signal):
631                     if busSig.connect is not None:
632                         tbWrapper = tbWrapper.replace('|-CONNECTIONS-|', '.'+
    busSig.connect + '(' + uVip.interface.instance + '_if.' + busSig.name +') ' +'
    ,\n\t\t|-CONNECTIONS-|')
633                     else:
634                         tbWrapper = tbWrapper.replace('|-CONNECTIONS-|', '.'+
    busSig.name + '(' + uVip.interface.instance + '_if.' + busSig.name +') ' +',\n
    \t\t|-CONNECTIONS-|')
635
636         for idx,uClock in enumerate(self.clock):
637             tbWrapper = tbWrapper.replace('|-CONNECTIONS-|', '.' + uClock.name +
    '(' + uClock.name + ')' +',\n\t\t|-CONNECTIONS-|')
638
639         for idx,uReset in enumerate(self.reset):
640             tbWrapper = tbWrapper.replace('|-CONNECTIONS-|', '.' + uReset.name +
    '(' + uReset.name + ')' +',\n\t\t|-CONNECTIONS-|')
641
642         #CLEANUP
643         tbWrapper = tbWrapper.replace(',\n\t|-INTERFACE-|','')
644         tbWrapper = tbWrapper.replace(',\n\t\t|-CONNECTIONS-|','')
645
646         wrapper_file = open(outputdir + '/' + self.name + "_wrapper.sv", "wt")
647         n = wrapper_file.write(tbWrapper)
648         wrapper_file.close()
649
650     def writeTop(self, outputdir):
651         with open(os.path.dirname(os.path.realpath(__file__)) + '/../src/
    templates/verification/top.tb', 'r') as file:
652             tbTop=file.read()
653
654         tbTop = tbTop.replace('|-PACKAGE-|', self.name)
655         tbTop = tbTop.replace('|-MODULE_NAME-|', self.name)
656
657         for idx,uClock in enumerate(self.clock):
658             tbTop = tbTop.replace('|-CLOCK-|', 'logic ' + uClock.name +';\n\t|-
    CLOCK-|')
659             tbTop = tbTop.replace('|-CLOCK_PERIOD-|', 'localparam P_' + uClock.
    name.upper() + ' = ' + str(uClock.period) +'ns;\n\t|-CLOCK_PERIOD-|')
660             tbTop = tbTop.replace('|-CLOCK_CH-|', 'always #(P_' + uClock.name.
    upper() + '/2)' + uClock.name + '= ~' + uClock.name +';\n\t|-CLOCK_CH-|')
661
662         for idx,uReset in enumerate(self.reset):
```

```
663             tbTop = tbTop.replace('|-RESET-|', 'logic ' + uReset.name +';\n\t\t|-
    RESET-|')
664             tbTop = tbTop.replace('|-RESET_PERIOD-|', 'localparam P_' + uReset.
    name.upper() + ' = ' + str(uReset.period) + 'ns;\n\t|-RESET_PERIOD-|')
665             tbTop = tbTop.replace('|-RESET_CH-|', 'always begin\n\t\t#(P_' +
    uReset.name.upper() + ') ' + uReset.name +' = 0; '+ '\n\t\t#(' + str(uReset.
    duration) + ') ' + uReset.name + ' = 1;' +'\n\t|-RESET_CH-|')
666
667         for idx, uAgent in enumerate(self.agent):
668             tbTop = tbTop.replace('|-INTERFACE-|', uAgent.interface.name + ' ' +
    uAgent.interface.instance + '_if_top (\n\t\t.' \
669             + uAgent.interface.clock.name + '(' + uAgent.interface.clock.name + '
    ), .' + \
670             uAgent.interface.reset.name + '(' + uAgent.interface.reset.name + ')\
    n\t);'+ '\n\t\t|-INTERFACE-|')
671             tbTop = tbTop.replace('|-INTERFACE_CONNECTION-|', '.' + uAgent.
    interface.instance + '_if (' + uAgent.interface.instance + '_if_top)' +',\n\t\
    t|-INTERFACE_CONNECTION-|')
672
673         for idx, uClock in enumerate(self.clock):
674             tbTop = tbTop.replace('|-INTERFACE_CONNECTION-|', '.' + uClock.name +
     '(' + uClock.name + ')' +',\n\t\t|-INTERFACE_CONNECTION-|')
675             tbTop = tbTop.replace('|-INITIAL_CLOCK-|', uClock.name + ' = 0;' +'\n
    \t\t|-INITIAL_CLOCK-|')
676
677         for idx, uReset in enumerate(self.reset):
678             tbTop = tbTop.replace('|-INTERFACE_CONNECTION-|', '.' + uReset.name +
     '(' + uReset.name + ')' +',\n\t\t|-INTERFACE_CONNECTION-|')
679             tbTop = tbTop.replace('|-INITIAL_RESET-|', uReset.name + ' = 1;' +'\n
    \t\t|-INITIAL_RESET-|')
680
681         for idx, uAgent in enumerate(self.agent):
682             tbTop = tbTop.replace('|-INTERFACE_CDB-|', \
683                         'uvm_config_db#(virtual ' + uAgent.interface.name + ')::
    set(null, "*.env_h.' + uAgent.instance + '.*", "VIRTUAL_IF", ' \
684                         + uAgent.interface.instance + '_if_top)' +';\n\t\t|-
    INTERFACE_CDB-|')
685
686         for idx, uVip in enumerate(self.vip):
687             tbTop = tbTop.replace('|-INTERFACE_CDB-|', \
688                         'uvm_config_db#(virtual ' + uVip.interface.name + ')::set
    (null, "*.env_h.' + uVip.instance + '.*", "VIRTUAL_IF", ' \
689                         + uVip.interface.instance + '_if_top)' +';\n\t\t|-
    INTERFACE_CDB-|')
690
691
692         #CLEANUP
693         tbTop = tbTop.replace('|-CLOCK-|', '')
694         tbTop = tbTop.replace('|-RESET-|', '')
695         tbTop = tbTop.replace('|-CLOCK_PERIOD-|', '')
696         tbTop = tbTop.replace('|-RESET_PERIOD-|', '')
697         tbTop = tbTop.replace('|-INITIAL_RESET-|', '')
698         tbTop = tbTop.replace('|-INITIAL_CLOCK-|', '')
699         tbTop = tbTop.replace('|-CLOCK_CH-|', '')
700         tbTop = tbTop.replace('|-RESET_CH-|', 'end')
701         tbTop = tbTop.replace('|-INTERFACE-|', '')
702         tbTop = tbTop.replace(',\n\t\t|-INTERFACE_CONNECTION-|', '')
```

```
703         tbTop = tbTop.replace('|-INTERFACE_CDB-|', '')
704         tbTop = tbTop.replace('\n\n', '\n')
705 
706 
707         top_file = open(outputdir + '/' + self.name + "_top.sv", "wt")
708         n = top_file.write(tbTop)
709         top_file.close()
710 
711     def writeModule(self, outputdir):
712         with open(os.path.dirname(os.path.realpath(__file__)) + '/../src/
    templates/frontend/module.fe', 'r') as file:
713             tbModule=file.read()
714 
715         tbModule = tbModule.replace('|-MODULE-|', self.name)
716 
717         tbModule = tbModule.replace('|-MODULE_NAME-|', self.name)
718 
719         for idx,uClock in enumerate(self.clock):
720             tbModule = tbModule.replace('|-SIGNALS-|', 'input ' + uClock.name +'
    ,\n\t\t|-SIGNALS-|')
721 
722         for idx,uReset in enumerate(self.reset):
723             tbModule = tbModule.replace('|-SIGNALS-|', 'input ' + uReset.name +'
    ,\n\t\t|-SIGNALS-|')
724 
725         for uSignal in self.signal:
726                 tbModule = tbModule.replace('|-SIGNALS-|', uSignal.io + ' ' +
    uSignal.type + ' ' + uSignal.name + ',\n\t\t|-SIGNALS-|')
727 
728 
729         tbModule = tbModule.replace(',\n\t\t|-SIGNALS-|', '')
730 
731         module_file = open(outputdir + '/rtl/src/' + self.name + ".sv", "wt")
732         n = module_file.write(tbModule)
733         module_file.close()
734 
735     def writeMakefile(self,  outputdir):
736         with open(os.path.dirname(os.path.realpath(__file__)) + '/../src/
    templates/verification/makefile_verif.tb', 'r') as file:
737             tbMakefile=file.read()
738 
739         tbMakefile = tbMakefile.replace('|-MODULE-|', self.name)
740 
741         aux_files = []
742         for idx, uAgent in enumerate(self.agent):
743             if uAgent.interface.name not in aux_files:
744                 tbMakefile = tbMakefile.replace('|-INTERFACE-|', '../../tb/' +
    uAgent.interface.name + '.sv \ \n\t|-INTERFACE-|')
745 
746             aux_files.append(uAgent.interface.name)
747 
748         for idx, uVip in enumerate(self.vip):
749             for idy, uInclude in enumerate(uVip.include):
750                 tbMakefile = tbMakefile.replace('|-INTERFACE-|', uInclude +' \ \n
    \t|-INTERFACE-|')
751 
752         test_string = """|-TEST-|:
```

```
753 \t@xrun -64bit -uvm +incdir+$(RTL_SRC) $(PKGS) $(IF) $(RTL) $(WRAPPER) ../../tb
    /|-MODULE-|_top.sv +UVM_TESTNAME=|-TEST-| -covtest |-TEST-|-$(SEED) -svseed $(
    SEED) -defparam top.min_cover=$(COVER) -defparam top.min_transa=$(TRANSA) $(
    RUN_ARGS_COMMON) -xmlibdirpath ../../workspace/rtlsim $(RUN_ARGS)"""
754
755         for idx, uTest in enumerate(self.test):
756             aux_string = test_string.replace('|-TEST-|', uTest.name)
757             aux_string = test_string.replace('|-MODULE-|', self.name)
758             tbMakefile = tbMakefile.replace('|-TEST-|', aux_string + '\n\n|-TEST
    -|')
759
760
761         tbMakefile = tbMakefile.replace('\n\n|-TEST-|', '')
762         tbMakefile = tbMakefile.replace(' \ \n\t|-INTERFACE-|', '')
763
764         module_file = open(outputdir + "/../scripts/rtlsim/Makefile", "wt")
765         n = module_file.write(tbMakefile)
766         module_file.close()
767
768
769 class Synth:
770     def __init__(self, name):
771         self.name = name
772         self.pdk = ''
773         self.pdk_files = []
774
775
776     def setPDK(self, pdk_dir):
777         self.pdk = pdk_dir
778
779     def addPDKFile(self, pdk_file):
780         self.pdk_files.append(pdk_file)
781
782     def writeTcl(self, output_dir):
783         with open(os.path.dirname(os.path.realpath(__file__)) + '/../src/
    templates/frontend/synth.fe', 'r') as file:
784             tbTcl=file.read()
785
786         tbTcl = tbTcl.replace('|-MODULE-|', self.name)
787         tbTcl = tbTcl.replace('|-TECHLIBASE-|', self.pdk)
788
789         for idx,uFile in enumerate(self.pdk_files):
790             tbTcl = tbTcl.replace('|-TECHLIBPATH-|', '$TECHLIBBASE' + uFile + ' \
     ' + '\n                        |-TECHLIBPATH-|')
791
792         tbTcl = tbTcl.replace(' \ \n                        |-TECHLIBPATH-|', '')
793
794
795         tcl_file = open(output_dir + '/scripts/synth_logic/synth.tcl', "wt")
796         n = tcl_file.write(tbTcl)
797         tcl_file.close()
798
799     def writeMakefile(Self, output_dir):
800         with open(os.path.dirname(os.path.realpath(__file__)) + '/../src/
    templates/frontend/makefile_synth.fe', 'r') as file:
801             tbMake=file.read()
802
```

```
803         mk_file = open(output_dir + '/scripts/synth_logic/Makefile', "wt")
804         n = mk_file.write(tbMake)
805         mk_file.close()
806 
807 class Package:
808     def __init__(self, name):
809         self.name = name
810         self.vip = []
811 
812     def addVip(self, vip):
813         self.vip.append(vip)
814 
815     def writePackage(self, outputdir):
816         pkg = """package {MODULE}_pkg;
817     `include "uvm_macros.svh"
818     import uvm_pkg::*;
819     |-IMPORT-|
820 
821     |-FILES-|
822 endpackage""".format(MODULE=self.name)
823 
824         f = []
825         for (dirpath, dirnames, filenames) in os.walk(outputdir):
826             f.extend(filenames)
827 
828         for filename in f:
829             if '_top.sv' not in filename and '_wrapper.sv' not in filename and '
    _interface.sv' not in filename:
830                 pkg = pkg.replace('|-FILES-|', '`include ../../tb/"' + filename +
    '"\n\t|-FILES-|')
831 
832         for uVip in self.vip:
833             import_name = uVip.file_package.replace('.sv', '')
834             if '/' in import_name:
835                 string = import_name.rsplit("/", 1)
836             pkg = pkg.replace('|-IMPORT-|', 'import ' + string[1] +'::*\n\t|-
    IMPORT-|')
837 
838         pkg = pkg.replace('|-FILES-|', '')
839         pkg = pkg.replace('|-IMPORT-|', '')
840 
841         pkg_file = open(outputdir + '/' + self.name + "_pkg.sv", "wt")
842         n = pkg_file.write(pkg)
843         pkg_file.close()
844 
845 class Formal:
846     def __init__(self, name):
847         self.name = name
848         self.clock = []
849         self.reset = []
850         self.signal = []
851 
852     def addSignal(self, signal):
853         self.signal.append(signal)
854 
855     def addClock(self, clock):
856         self.clock.append(clock)
```

```
857
858     def addReset(self, reset):
859         self.reset.append(reset)
860
861     def writeVerifModule(self, output_dir):
862         with open(os.path.dirname(os.path.realpath(__file__)) + '/../src/
    templates/verification/module_assertions.tb', 'r') as file:
863             tbVMod=file.read()
864
865         tbVMod = tbVMod.replace('|-MODULE-|', self.name)
866
867         for idx,uClock in enumerate(self.clock):
868             tbVMod = tbVMod.replace('|-SIGNALS-|', 'input logic' + uClock.name +'
    ,\n\t\t|-SIGNALS-|')
869
870         for idx,uReset in enumerate(self.reset):
871             tbVMod = tbVMod.replace('|-SIGNALS-|', 'input logic' + uReset.name +'
    ,\n\t\t|-SIGNALS-|')
872
873         for uSignal in self.signal:
874             tbVMod = tbVMod.replace('|-SIGNALS-|', 'input' + ' ' + uSignal.type +
     ' ' + uSignal.name + ',\n\t\t|-SIGNALS-|')
875
876
877         tbVMod = tbVMod.replace(',\n\t\t|-SIGNALS-|', '')
878
879         vmodule_file = open(output_dir + '/../formal/properties/v_' + self.name +
     '.sva', "wt")
880         n = vmodule_file.write(tbVMod)
881         vmodule_file.close()
882
883         with open(os.path.dirname(os.path.realpath(__file__)) + '/../src/
    templates/verification/bindings.tb', 'r') as file:
884             tbBind=file.read()
885
886         tbBind = tbBind.replace('|-MODULE-|', self.name)
887
888         for idx,uClock in enumerate(self.clock):
889             tbBind = tbBind.replace('|-SIGNALS-|', '.' + uClock.name + '(' +
    uClock.name + ')' +',\n\t\t|-SIGNALS-|')
890
891         for idx,uReset in enumerate(self.reset):
892             tbBind = tbBind.replace('|-SIGNALS-|', '.' + uReset.name + '(' +
    uReset.name + ')' +',\n\t\t|-SIGNALS-|')
893
894         for uSignal in self.signal:
895             tbBind = tbBind.replace('|-SIGNALS-|', '.' + uSignal.name + '(' +
    uSignal.name + ')' +',\n\t\t|-SIGNALS-|')
896
897         tbBind = tbBind.replace(',\n\t\t|-SIGNALS-|', '')
898
899         bidings_file = open(output_dir + '/../formal/properties/bindings.sva', "
    wt")
900         n = bidings_file.write(tbBind)
901         bidings_file.close()
902
903     def writeTcl(self, output_dir):
```

```
904
905         with open(os.path.dirname(os.path.realpath(__file__)) + '/../src/
    templates/verification/fpv.tb', 'r') as file:
906             tbTcl=file.read()
907
908         tbTcl = tbTcl.replace('|-MODULE-|', self.name)
909
910         for idx,uClock in enumerate(self.clock):
911             tbTcl = tbTcl.replace('|-CLOCK-|', 'clock ' + uClock.name +'\n|-CLOCK
    -|')
912
913         for idx,uReset in enumerate(self.reset):
914             tbTcl = tbTcl.replace('|-RESET-|', 'reset ~' + uReset.name +'\n|-
    RESET-|')
915
916         tbTcl = tbTcl.replace('|-CLOCK-|', '')
917         tbTcl = tbTcl.replace('|-RESET-|', '')
918
919         tcl_file = open(output_dir + '/../scripts/formal/formal.tcl', "wt")
920         n = tcl_file.write(tbTcl)
921         tcl_file.close()
922
923     def writeMakefile(Self, output_dir):
924         with open(os.path.dirname(os.path.realpath(__file__)) + '/../src/
    templates/verification/formal_mk.tb', 'r') as file:
925             tbMake=file.read()
926
927         if 'verification' in output_dir:
928             mk_file = open(output_dir + '/../scripts/formal/Makefile', "wt")
929             n = mk_file.write(tbMake)
930             mk_file.close()
931         else:
932             tbMake = tbMake.replace('-fpv ', '-fpv ../../../verification/scripts/
    formal/')
933             mk_file = open(output_dir + '/rtl/tb/Makefile', "wt")
934             n = mk_file.write(tbMake)
935             mk_file.close()
936
937 class Parser:
938
939     def __init__(self, name, inputfile, outputdir):
940         self.name = name
941         self.inputfile = inputfile
942         self.outputdir = outputdir
943
944     def parse_verif(self):
945         with open(self.inputfile, 'r') as file:
946             tbConfig=file.read()
947
948         tbConfig_Split = "".join([s for s in tbConfig.splitlines(True) if s.strip
    ("\r\n")])
949
950         tbConfig_Split = tbConfig_Split.splitlines()
951
952         tbSplit_agent = []
953         tbSplit_comp = []
954         tbSplit_signal = []
```

```
955         tbSplit_field = []
956         tbSplit_clock = []
957         tbSplit_reset = []
958         tbSplit_interface = []
959         tbSplit_transa = []
960         tbSplit_refmod = []
961         tbSplit_module = []
962         tbSplit_test = []
963         tbSplit_sequence = []
964         tbSplit_if_instance = []
965         tbSplit_vip = []
966 
967         for line in tbConfig_Split:
968             if 'agent' in line and '=' not in line:
969                 current_analysis='agent'
970             if 'comp' in line and '=' not in line:
971                 current_analysis='comp'
972             if 'signal' in line and '=' not in line:
973                 current_analysis='signal'
974             if 'field' in line and '=' not in line:
975                 current_analysis='field'
976             if 'clock' in line and '=' not in line:
977                 current_analysis='clock'
978             if 'reset' in line and '=' not in line:
979                 current_analysis='reset'
980             if 'interface' in line and '=' not in line:
981                 current_analysis='interface'
982             if 'transaction' in line and '=' not in line:
983                 current_analysis='transaction'
984             if 'refmod' in line and '=' not in line:
985                 current_analysis='refmod'
986             if 'module' in line and '=' not in line:
987                 current_analysis='module'
988             if 'test' in line and '=' not in line:
989                 current_analysis='test'
990             if 'sequence' in line and '=' not in line:
991                 current_analysis='sequence'
992             if 'if_instance' in line and '=' not in line:
993                 current_analysis='if_instance'
994             if 'vip' in line and '=' not in line:
995                 current_analysis='vip'
996             if '}' in line and '=' not in line:
997                 current_analysis=current_analysis
998 
999             if current_analysis == 'agent':
1000                 tbSplit_agent.append(line)
1001             if current_analysis == 'comp':
1002                 tbSplit_comp.append(line)
1003             if current_analysis == 'signal':
1004                 tbSplit_signal.append(line)
1005             if current_analysis == 'field':
1006                 tbSplit_field.append(line)
1007             if current_analysis == 'clock':
1008                 tbSplit_clock.append(line)
1009             if current_analysis == 'reset':
1010                 tbSplit_reset.append(line)
1011             if current_analysis == 'interface':
```

```
1012                tbSplit_interface.append(line)
1013            if current_analysis == 'transaction':
1014                tbSplit_transa.append(line)
1015            if current_analysis == 'refmod':
1016                tbSplit_refmod.append(line)
1017            if current_analysis == 'module':
1018                tbSplit_module.append(line)
1019            if current_analysis == 'test':
1020                tbSplit_test.append(line)
1021            if current_analysis == 'sequence':
1022                tbSplit_sequence.append(line)
1023            if current_analysis == 'if_instance':
1024                tbSplit_if_instance.append(line)
1025            if current_analysis == 'vip':
1026                tbSplit_vip.append(line)
1027            if current_analysis == 'NONE':
1028                pass
1029
1030        list_agent = []
1031        list_comp = []
1032        list_signal = []
1033        list_field = []
1034        list_clock = []
1035        list_reset = []
1036        list_interface = []
1037        list_if_instance = []
1038        list_transa = []
1039        list_refmod = []
1040        list_test = []
1041        list_sequence = []
1042        list_vip = []
1043
1044        for line in tbSplit_module:
1045            if 'module' in line and '=' not in line:
1046                auxName = ''
1047
1048            if 'name' in line:
1049                string = line.split("=", 1)
1050                auxName = string[1]
1051                auxName = auxName.replace(" ","")
1052
1053            if '}' in line:
1054                moduleName = auxName
1055
1056        for line in tbSplit_clock:
1057            if 'clock' in line and '=' not in line:
1058                auxName = 'NONE'
1059                auxPeriod = 'NONE'
1060
1061            if 'name' in line:
1062                string = line.split("=", 1)
1063                auxName = string[1]
1064                auxName = auxName.replace(" ","")
1065
1066            if 'period' in line:
1067                string = line.split("=", 1)
1068                auxPeriod = string[1]
```

```
1069                auxPeriod = auxPeriod.replace(" ","")
1070
1071            if '}' in line:
1072                auxClock = Clock(auxName, auxPeriod)
1073                list_clock.append(auxClock)
1074
1075        for line in tbSplit_reset:
1076            if 'reset' in line and '=' not in line:
1077                auxName = 'NONE'
1078                auxPeriod = 'NONE'
1079                auxDuration = 'NONE'
1080
1081            if 'name' in line:
1082                string = line.split("=", 1)
1083                auxName = string[1]
1084                auxName = auxName.replace(" ","")
1085
1086            if 'period' in line:
1087                string = line.split("=", 1)
1088                auxPeriod = string[1]
1089                auxPeriod = auxPeriod.replace(" ","")
1090
1091            if 'duration' in line:
1092                string = line.split("=", 1)
1093                auxDuration = string[1]
1094                auxDuration = auxDuration.replace(" ","")
1095
1096            if '}' in line:
1097                auxReset = Reset(auxName, auxPeriod, auxDuration)
1098                list_reset.append(auxReset)
1099
1100        for line in tbSplit_signal:
1101            if 'signal' in line and '=' not in line:
1102                auxName = 'NONE'
1103                auxType = 'NONE'
1104                auxIo = 'NONE'
1105                auxConnect = 'NONE'
1106
1107            if 'name' in line:
1108                string = line.split("=", 1)
1109                auxName = string[1]
1110                auxName = auxName.replace(" ","")
1111
1112            if 'type' in line:
1113                string = line.split("=", 1)
1114                auxType = string[1]
1115                auxType = auxType.replace(" ","")
1116
1117            if 'io' in line:
1118                string = line.split("=", 1)
1119                auxIo = string[1]
1120                auxIo = auxIo.replace(" ","")
1121
1122            if '}' in line:
1123                auxSignal = Signal(auxName, auxType, auxIo)
1124                list_signal.append(auxSignal)
1125
```

```
1126        for line in tbSplit_field:
1127            if 'field' in line and '=' not in line:
1128                auxName = 'NONE'
1129                auxType = 'NONE'
1130
1131            if 'name' in line:
1132                string = line.split("=", 1)
1133                auxName = string[1]
1134                auxName = auxName.replace(" ","")
1135
1136            if 'type' in line:
1137                string = line.split("=", 1)
1138                auxType = string[1]
1139                auxType = auxType.replace(" ","")
1140
1141
1142            if (auxName is not 'NONE') and (auxType is not 'NONE'):
1143                auxField = Field(auxName, auxType)
1144                list_field.append(auxField)
1145
1146        for line in tbSplit_interface:
1147
1148            if 'interface' in line and '=' not in line:
1149                auxName = ''
1150                auxInstance = ''
1151                auxSignal_list = []
1152
1153            if 'name' in line:
1154                string = line.split("=", 1)
1155                auxName = string[1]
1156                auxName = auxName.replace(" ","")
1157
1158            if 'clock' in line:
1159                string = line.split("=", 1)
1160                auxClock_name = string[1]
1161                auxClock_name = auxClock_name.replace(" ","")
1162
1163                for idx,uClock in enumerate(list_clock):
1164                    if (uClock.name == auxClock_name):
1165                        auxClock = uClock
1166                        break
1167
1168            if 'reset' in line:
1169                string = line.split("=", 1)
1170                auxReset_name = string[1]
1171                auxReset_name = auxReset_name.replace(" ","")
1172
1173                for idx,uReset in enumerate(list_reset):
1174                    if (uReset.name == auxReset_name):
1175                        auxReset = uReset
1176                        break
1177
1178            if 'signal' in line:
1179                string = line.split("=", 1)
1180                auxSignal_name = string[1]
1181                auxSignal_name = auxSignal_name.replace(" ","")
1182
```

```
1183                for idx,uSignal in enumerate(list_signal):
1184                    if (uSignal.name == auxSignal_name):
1185                        auxSignal = uSignal
1186                        break
1187
1188                auxSignal_list.append(auxSignal)
1189
1190
1191            if '}' in line:
1192                for line in tbSplit_if_instance:
1193                    if 'if_instance' in line and '=' not in line:
1194                        auxInstance = ''
1195                        auxName_con = ''
1196                        auxSig_if = []
1197                        auxDut_if = []
1198                        auxSignal_sig_list = []
1199                        auxInterface = None
1200                        for uSignal in auxSignal_list:
1201                            auxSignal_sig_list.append(uSignal)
1202
1203                    if 'type' in line:
1204                        string = line.split("=", 1)
1205                        auxName_con = string[1]
1206                        auxName_con = auxName_con.replace(" ","")
1207
1208                    if 'instance' in line and 'if_instance' not in line:
1209                        string = line.split("=", 1)
1210                        auxInstance = string[1]
1211                        auxInstance = auxInstance.replace(" ","")
1212
1213                    if 'con' in line:
1214                        string = line.split("=", 1)
1215                        auxConnect_name = string[1]
1216                        auxConnect_name = auxConnect_name.replace(" ","")
1217                        connection = auxConnect_name.split(",", 1)
1218                        auxSig_if.append(connection[0])
1219                        auxDut_if.append(connection[1])
1220
1221                    if '}' in line:
1222                        if auxName_con==auxName:
1223                            auxInterface = Interface(auxName, auxInstance,
    auxClock, auxReset)
1224                            for idx,uSignal in enumerate(auxSignal_list):
1225                                signal_aux = copy.copy(uSignal)
1226                                for idy, uSignal_aux in enumerate(auxSig_if):
1227                                    if uSignal_aux == signal_aux.name:
1228                                        signal_aux.addConnection(auxDut_if[idy])
1229                                auxInterface.addSignal(signal_aux)
1230
1231                            list_interface.append(auxInterface)
1232                        else:
1233                            pass
1234
1235        for line in tbSplit_transa:
1236
1237            if 'transaction' in line and '=' not in line:
1238                auxName = ''
```

```
1239                auxField_list = []
1240
1241            if 'name' in line:
1242                string = line.split("=", 1)
1243                auxName = string[1]
1244                auxName = auxName.replace(" ","")
1245
1246
1247            if 'field' in line:
1248                string = line.split("=", 1)
1249                auxField_name = string[1]
1250                auxField_name = auxField_name.replace(" ","")
1251
1252                for idx,uField in enumerate(list_field):
1253                    if (uField.name == auxField_name):
1254                        auxField = uField
1255                        break
1256
1257                auxField_list.append(auxField)
1258
1259
1260            if '}' in line:
1261
1262                auxTransaction = Transaction(auxName)
1263                for idx,uField in enumerate(auxField_list):
1264                    auxTransaction.addField(uField)
1265
1266                list_transa.append(auxTransaction)
1267
1268        for line in tbSplit_comp:
1269
1270            if 'comp' in line and '=' not in line:
1271                auxName = ''
1272                auxInstance =''
1273
1274            if 'name' in line:
1275                string = line.split("=", 1)
1276                auxName = string[1]
1277                auxName = auxName.replace(" ","")
1278
1279            if 'instance' in line:
1280                string = line.split("=", 1)
1281                auxInstance = string[1]
1282                auxInstance = auxInstance.replace(" ","")
1283
1284            if 'transaction' in line:
1285                string = line.split("=", 1)
1286                auxTransaction_name = string[1]
1287                auxTransaction_name = auxTransaction_name.replace(" ","")
1288
1289                for idx,uTransaction in enumerate(list_transa):
1290                    if (uTransaction.name == (auxTransaction_name + '_transaction
    ')):
1291                        auxTransaction = uTransaction
1292                        break
1293
1294            if '}' in line:
```

```
1295
1296                auxComp = Comparator(auxName, auxInstance, auxTransaction)
1297                list_comp.append(auxComp)
1298
1299        for line in tbSplit_refmod:
1300
1301            if 'refmod' in line and '=' not in line:
1302                auxName = ''
1303                auxInstance =''
1304                auxPort_out = []
1305
1306            if 'name' in line:
1307                string = line.split("=", 1)
1308                auxName = string[1]
1309                auxName = auxName.replace(" ","")
1310
1311            if 'instance' in line:
1312                string = line.split("=", 1)
1313                auxInstance = string[1]
1314                auxInstance = auxInstance.replace(" ","")
1315
1316            if 'refmod_policy' in line:
1317                string = line.split("=", 1)
1318                auxRefmod_policy = string[1]
1319                auxRefmod_policy = auxRefmod_policy.replace(" ","")
1320
1321            if 'comp' in line:
1322                string = line.split("=", 1)
1323                auxComp = string[1]
1324                auxComp = auxComp.replace(" ","")
1325
1326            if 'connect' in line:
1327                string = line.split("=", 1)
1328                auxConnect_name = string[1]
1329                auxConnect_name = auxConnect_name.replace(" ","")
1330                connection = auxConnect_name.split(",", 1)
1331                for uTransaction_out in list_transa:
1332                    if uTransaction_out.name == (connection[1] + '_transaction'):
1333                        aux_Transa = uTransaction_out
1334                        break
1335
1336                auxPort = Port(auxName, connection[0], auxInstance, aux_Transa,
    0)
1337                auxPort_out.append(auxPort)
1338
1339            if '}' in line:
1340
1341                for idx,uComp in enumerate(list_comp):
1342                    if (uComp.instance == auxComp):
1343                        auxTransa_comp = uComp.transaction
1344                        break
1345
1346                port_comp = Port('rfmtocomp', auxComp, auxInstance,
    auxTransa_comp, 1)
1347
1348                auxRfm = Refmod(auxName, auxInstance, auxRefmod_policy, port_comp
    , auxComp)
```

```
1349                if not auxPort_out:
1350                    pass
1351                else:
1352                    for idx, uConnect in enumerate(auxPort_out):
1353                        auxRfm.addPortOut(uConnect)
1354
1355                list_refmod.append(auxRfm)
1356
1357        for line in tbSplit_agent:
1358
1359            if 'agent' in line and '=' not in line:
1360                auxName = ''
1361                auxInstance = ''
1362
1363            if 'name' in line:
1364                string = line.split("=", 1)
1365                auxName = string[1]
1366                auxName = auxName.replace(" ","")
1367
1368            if 'instance' in line:
1369                string = line.split("=", 1)
1370                auxInstance = string[1]
1371                auxInstance = auxInstance.replace(" ","")
1372
1373            if 'type' in line:
1374                string = line.split("=", 1)
1375                auxType = string[1]
1376                auxType = auxType.replace(" ","")
1377
1378            if 'driver_policy' in line:
1379                string = line.split("=", 1)
1380                auxDriver_policy = string[1]
1381                auxDriver_policy = auxDriver_policy.replace(" ","")
1382
1383            if 'monitor_policy' in line:
1384                string = line.split("=", 1)
1385                auxMonitor_policy = string[1]
1386                auxMonitor_policy = auxMonitor_policy.replace(" ","")
1387
1388            if 'transaction' in line:
1389                string = line.split("=", 1)
1390                auxTransaction_name = string[1]
1391                auxTransaction_name = auxTransaction_name.replace(" ","")
1392
1393                for idx,uTransaction in enumerate(list_transa):
1394                    if (uTransaction.name == (auxTransaction_name + '_transaction
    ')):
1395                        auxTransaction = uTransaction
1396                        break
1397
1398            if 'interface' in line:
1399                string = line.split("=", 1)
1400                auxInterface_instance = string[1]
1401                auxInterface_instance = auxInterface_instance.replace(" ","")
1402                for idx,uInterface in enumerate(list_interface):
1403                    if (uInterface.instance == auxInterface_instance):
1404                        auxInterface = uInterface
```

```
1405                        break
1406
1407            if 'refmod' in line:
1408                string = line.split("=", 1)
1409                auxRefmod_name = string[1]
1410                auxRefmod_name = auxRefmod_name.replace(" ","")
1411
1412            if 'comp' in line:
1413                string = line.split("=", 1)
1414                auxComp_name = string[1]
1415                auxComp_name = auxComp_name.replace(" ","")
1416
1417            if '}' in line:
1418
1419                auxAgent = Agent(auxName, auxInstance, auxInterface,
    auxTransaction, auxDriver_policy, auxMonitor_policy, auxType)
1420
1421                if (auxAgent.type == 'input'):
1422                    auxAgent.setRefmodConn(auxRefmod_name + '_rfm')
1423                elif (auxAgent.type == 'output'):
1424                    auxAgent.setCompConn(auxComp_name)
1425                else:
1426                    auxAgent.setRefmodConn(auxRefmod_name + '_rfm')
1427                    auxAgent.setCompConn(auxComp_name)
1428
1429                list_agent.append(auxAgent)
1430
1431
1432        for line in tbSplit_vip:
1433
1434            if 'vip' in line and '=' not in line:
1435                auxName = ''
1436                auxInstance = ''
1437                auxInterface = ''
1438                auxFile_package = ''
1439                auxInclude = []
1440                auxPort = []
1441                auxPort_destination = []
1442                auxPort_transa = []
1443
1444            if 'name' in line:
1445                string = line.split("=", 1)
1446                auxName = string[1]
1447                auxName = auxName.replace(" ","")
1448
1449            if 'instance' in line:
1450                string = line.split("=", 1)
1451                auxInstance = string[1]
1452                auxInstance = auxInstance.replace(" ","")
1453
1454            if 'interface' in line:
1455                string = line.split("=", 1)
1456                auxInterface_instance = string[1]
1457                auxInterface_instance = auxInterface_instance.replace(" ","")
1458                for idx,uInterface in enumerate(list_interface):
1459                    if (uInterface.instance == auxInterface_instance):
1460                        auxInterface = uInterface
```

```
1461                        break
1462
1463            if 'file_package' in line:
1464                string = line.split("=", 1)
1465                auxFile_package = string[1]
1466                auxFile_package = auxFile_package.replace(" ","")
1467
1468            if 'include' in line:
1469                string = line.split("=", 1)
1470                auxInclude_name = string[1]
1471                auxInclude_name = auxInclude_name.replace(" ","")
1472                auxInclude.append(auxInclude_name)
1473
1474            if 'tlm_port' in line:
1475                string = line.split("=", 1)
1476                auxConnect_name = string[1]
1477                auxConnect_name = auxConnect_name.replace(" ","")
1478                connection = auxConnect_name.split(",", 1)
1479                auxPort.append(connection[0])
1480                connection2 = connection[1].split(",", 1)
1481                auxPort_destination.append(connection2[0])
1482                auxPort_transa.append(connection[1])
1483
1484            if '}' in line:
1485
1486                auxVip = Vip(auxName, auxInstance, auxInterface, auxFile_package)
1487                for idx,uPort in enumerate(auxPort):
1488                    auxTransa = Transaction(auxPort_transa[idx])
1489                    port_create = Port(uPort, auxPort_destination[idx],
    auxInstance, copy.copy(auxTransa), 0)
1490                    auxVip.addPort(copy.copy(port_create))
1491
1492                for idx,uInclude in enumerate(auxInclude):
1493                    auxVip.addInclude(uInclude)
1494
1495                list_vip.append(copy.copy(auxVip))
1496
1497        # Creating ENV
1498
1499        scoreboard = Scoreboard(moduleName, moduleName +'_scb')
1500
1501        for uComp in list_comp:
1502            scoreboard.addComp(uComp)
1503
1504        for uRefmod in list_refmod:
1505            scoreboard.addRefmod(uRefmod)
1506
1507        env = Env(moduleName, 'env', [], scoreboard)
1508
1509        for uAgent in list_agent:
1510            env.addAgent(uAgent)
1511
1512        for uVip in list_vip:
1513            env.addVip(uVip)
1514
1515        env.writeEnv(self.outputdir)
1516        env.scoreboard.writeScoreboard(self.outputdir)
```

```
1517
1518        for refmod in env.scoreboard.refmod:
1519            refmod.writeRefmod(self.outputdir)
1520
1521        different_agent_name = []
1522        different_agent = []
1523
1524        for agent in env.agent:
1525            if agent.name not in different_agent_name:
1526                different_agent_name.append(agent.name)
1527                different_agent.append(agent)
1528            else:
1529                pass
1530
1531        for agent in different_agent:
1532            agent.writeAgentAll(self.outputdir)
1533
1534        # Creating Tests and Sequences
1535
1536        for line in tbSplit_sequence:
1537            if 'sequence' in line and '=' not in line:
1538                auxName = 'NONE'
1539
1540            if 'name' in line:
1541                string = line.split("=", 1)
1542                auxName = string[1]
1543                auxName = auxName.replace(" ","")
1544
1545            if 'transaction' in line:
1546                string = line.split("=", 1)
1547                auxTransaction_name = string[1]
1548                auxTransaction_name = auxTransaction_name.replace(" ","")
1549
1550                for idx,uTransaction in enumerate(list_transa):
1551                    if (uTransaction.name == (auxTransaction_name + '_transaction
    ')):
1552                        auxTransaction = uTransaction
1553                        break
1554
1555            if 'agent' in line:
1556                string = line.split("=", 1)
1557                auxAgent_instance = string[1]
1558                auxAgent_instance = auxAgent_instance.replace(" ","")
1559
1560                for idx,uAgent in enumerate(list_agent):
1561                    if (uAgent.instance == auxAgent_instance):
1562                        auxAgent = uAgent
1563                        break
1564
1565            if '}' in line:
1566                auxSequence = Sequence(auxName, auxAgent, auxTransaction)
1567                list_sequence.append(auxSequence)
1568
1569
1570        auxSequence = []
1571        for line in tbSplit_test:
1572            if 'test' in line and '=' not in line:
```

```
1573                auxName = 'NONE'
1574
1575            if 'name' in line:
1576                string = line.split("=", 1)
1577                auxName = string[1]
1578                auxName = auxName.replace(" ","")
1579
1580            if 'sequence' in line:
1581                string = line.split("=", 1)
1582                auxSequence_name = string[1]
1583                auxSequence_name = auxSequence_name.replace(" ","")
1584
1585                for idx,uSequence in enumerate(list_sequence):
1586                    if (uSequence.name == (auxSequence_name + '_sequence')):
1587                        auxSequence.append(uSequence)
1588                        break
1589
1590            if '}' in line:
1591                auxTest= Test(env, auxName)
1592                for uSequence in auxSequence:
1593                    auxTest.addSequence(uSequence)
1594
1595                list_test.append(auxTest)
1596
1597
1598        for sequence in list_sequence:
1599            sequence.writeSequence(self.outputdir)
1600
1601        for test in list_test:
1602            test.writeTest(self.outputdir)
1603
1604        Dut = Module(moduleName)
1605
1606        for uClock in list_clock:
1607            Dut.addClock(uClock)
1608
1609        for uReset in list_reset:
1610            Dut.addReset(uReset)
1611
1612        for idx,uAgent in enumerate(list_agent):
1613            Dut.addAgent(uAgent)
1614
1615        for idx,uVip in enumerate(list_vip):
1616            Dut.addVip(uVip)
1617
1618        for uSignal in list_signal:
1619            Dut.addSignal(uSignal)
1620
1621        for uTest in list_test:
1622            Dut.addTest(uTest)
1623
1624        Dut.addEnv(env)
1625
1626        Dut.writeWrapper(self.outputdir)
1627        Dut.writeTop(self.outputdir)
1628        Dut.writeMakefile(self.outputdir)
1629
```

```
1630            pkg = Package(moduleName)
1631
1632            for idx,uVip in enumerate(list_vip):
1633                pkg.addVip(uVip)
1634
1635            pkg.writePackage(self.outputdir)
1636
1637            formal = Formal(moduleName)
1638
1639            for uSignal in list_signal:
1640                formal.addSignal(uSignal)
1641
1642            for uClock in list_clock:
1643                formal.addClock(uClock)
1644
1645            for uReset in list_reset:
1646                formal.addReset(uReset)
1647
1648
1649            formal.writeMakefile(self.outputdir)
1650            formal.writeTcl(self.outputdir)
1651            formal.writeVerifModule(self.outputdir)
1652
1653        def parse_fe(self):
1654            with open(self.inputfile, 'r') as file:
1655                tbConfig=file.read()
1656
1657            tbConfig_Split = "".join([s for s in tbConfig.splitlines(True) if s.strip
        ("\r\n")])
1658
1659            tbConfig_Split = tbConfig_Split.splitlines()
1660
1661            tbSplit_signal = []
1662            tbSplit_clock = []
1663            tbSplit_reset = []
1664            tbSplit_module = []
1665            tbSplit_pdk = []
1666
1667            for line in tbConfig_Split:
1668                if 'signal' in line and '=' not in line:
1669                    current_analysis='signal'
1670                if 'clock' in line and '=' not in line:
1671                    current_analysis='clock'
1672                if 'reset' in line and '=' not in line:
1673                    current_analysis='reset'
1674                if 'module' in line and '=' not in line:
1675                    current_analysis='module'
1676                if 'pdk' in line and '=' not in line:
1677                    current_analysis='pdk'
1678                if '}' in line and '=' not in line:
1679                    current_analysis=current_analysis
1680
1681                if current_analysis == 'signal':
1682                    tbSplit_signal.append(line)
1683                if current_analysis == 'clock':
1684                    tbSplit_clock.append(line)
1685                if current_analysis == 'reset':
```

```
1686                tbSplit_reset.append(line)
1687            if current_analysis == 'module':
1688                tbSplit_module.append(line)
1689            if current_analysis == 'pdk':
1690                tbSplit_pdk.append(line)
1691            if current_analysis == 'NONE':
1692                pass
1693
1694        list_signal = []
1695        list_clock = []
1696        list_reset = []
1697        list_pdk_files = []
1698        list_synth_Script = []
1699
1700        for line in tbSplit_module:
1701            if 'module' in line and '=' not in line:
1702                auxName = ''
1703
1704            if 'name' in line:
1705                string = line.split("=", 1)
1706                auxName = string[1]
1707                auxName = auxName.replace(" ","")
1708
1709            if '}' in line:
1710                moduleName = auxName
1711
1712        for line in tbSplit_clock:
1713            if 'clock' in line and '=' not in line:
1714                auxName = 'NONE'
1715                auxPeriod = 'NONE'
1716
1717            if 'name' in line:
1718                string = line.split("=", 1)
1719                auxName = string[1]
1720                auxName = auxName.replace(" ","")
1721
1722            if 'period' in line:
1723                string = line.split("=", 1)
1724                auxPeriod = string[1]
1725                auxPeriod = auxPeriod.replace(" ","")
1726
1727            if '}' in line:
1728                auxClock = Clock(auxName, auxPeriod)
1729                list_clock.append(auxClock)
1730
1731        for line in tbSplit_reset:
1732            if 'reset' in line and '=' not in line:
1733                auxName = 'NONE'
1734                auxPeriod = 'NONE'
1735                auxDuration = 'NONE'
1736
1737            if 'name' in line:
1738                string = line.split("=", 1)
1739                auxName = string[1]
1740                auxName = auxName.replace(" ","")
1741
1742            if 'period' in line:
```

```
1743                string = line.split("=", 1)
1744                auxPeriod = string[1]
1745                auxPeriod = auxPeriod.replace(" ","")
1746
1747            if 'duration' in line:
1748                string = line.split("=", 1)
1749                auxDuration = string[1]
1750                auxDuration = auxDuration.replace(" ","")
1751
1752            if '}' in line:
1753                auxReset = Reset(auxName, auxPeriod, auxDuration)
1754                list_reset.append(auxReset)
1755
1756        for line in tbSplit_signal:
1757            if 'signal' in line and '=' not in line:
1758                auxName = 'NONE'
1759                auxType = 'NONE'
1760                auxIo = 'NONE'
1761                auxConnect = 'NONE'
1762
1763            if 'name' in line:
1764                string = line.split("=", 1)
1765                auxName = string[1]
1766                auxName = auxName.replace(" ","")
1767
1768            if 'type' in line:
1769                string = line.split("=", 1)
1770                auxType = string[1]
1771                auxType = auxType.replace(" ","")
1772
1773            if 'io' in line:
1774                string = line.split("=", 1)
1775                auxIo = string[1]
1776                auxIo = auxIo.replace(" ","")
1777
1778            if '}' in line:
1779                auxSignal = Signal(auxName, auxType, auxIo)
1780                list_signal.append(auxSignal)
1781
1782        for line in tbSplit_pdk:
1783            if 'pdk' in line and '=' not in line:
1784                auxName = 'NONE'
1785                auxDir = 'NONE'
1786                auxFile = 'NONE'
1787
1788            if 'name' in line:
1789                string = line.split("=", 1)
1790                auxName = string[1]
1791                auxName = auxName.replace(" ","")
1792
1793            if 'dir' in line:
1794                string = line.split("=", 1)
1795                auxDir = string[1]
1796                auxDir = auxDir.replace(" ","")
1797
1798            if 'file' in line:
1799                string = line.split("=", 1)
```

```
1800                auxFile = string[1]
1801                auxFile = auxFile.replace(" ","")
1802                list_pdk_files.append(auxFile)
1803
1804            if '}' in line:
1805                auxSynth = Synth(moduleName)
1806                auxSynth.setPDK(auxDir)
1807                for file_s in list_pdk_files:
1808                    auxSynth.addPDKFile(file_s)
1809
1810                list_synth_Script.append(auxSynth)
1811
1812
1813        Dut = Module(moduleName)
1814
1815        for uClock in list_clock:
1816            Dut.addClock(uClock)
1817
1818        for uReset in list_reset:
1819            Dut.addReset(uReset)
1820
1821        for uSignal in list_signal:
1822            Dut.addSignal(uSignal)
1823
1824        Dut.writeModule(self.outputdir)
1825
1826        for uSynth in list_synth_Script:
1827            uSynth.writeTcl(self.outputdir)
1828            uSynth.writeMakefile(self.outputdir)
1829
1830        formal = Formal(moduleName)
1831
1832        formal.writeMakefile(self.outputdir)
1833
1834
1835
1836 def display_title_bar():
1837
1838     # Clears the terminal screen, and displays a title bar.
1839     print(Fore.BLUE + "#####################################################")
1840     print(Fore.BLUE + "
                                ")
1841     print(Fore.BLUE + "
                    ")
1842     print(Fore.BLUE + "
                           ")
1843     print(Fore.BLUE + "                                                    ")
1844     print(Fore.BLUE + "
                ")
1845     print(Fore.BLUE + "
                ")
1846     print(Fore.BLUE + "
                  ")
1847     print(Fore.BLUE + "#####################################################")
1848     print(Fore.BLUE + "#####################################################")
1849     print(Fore.BLUE + "# This script is used to automate generation of    ")
1850     print(Fore.BLUE + "# Reference flow for hardware projects in Frontend")
```

```
1851     print(Fore.BLUE + "# and Verification.                                    ")
1852     print(Fore.BLUE + "#################################################")
1853     print(Fore.BLUE + "#################################################")
1854     print(Fore.BLUE + "# Aurora Integrated Reference Flow Generation ")
1855     print(Fore.BLUE + "# Author: Jose Iuri Barbosa de Brito             ")
1856     print(Fore.BLUE + "# MIT License                                    ")
1857     print(Fore.BLUE + "# Copyright: Copyright (c) 2020, XMEN Lab - Universidade
         Federal de Campina Grande")
1858     print(Fore.BLUE + "# Credits: ")
1859     print(Fore.BLUE + "# Version: 0.2")
1860     print(Fore.BLUE + "# Maintainer: Jose Iuri Barbosa de Brio")
1861     print(Fore.BLUE + "# Email: jose.brito@embedded.ufcg.edu.br")
1862     print(Fore.BLUE + "# Status: In Progress")
1863     print(Fore.BLUE + "#################################################\n\n")
1864
1865 class CapitalisedHelpFormatter(argparse.HelpFormatter):
1866     def add_usage(self, usage, actions, groups, prefix=None):
1867         if prefix is None:
1868             prefix = 'Usage: '
1869         return super(CapitalisedHelpFormatter, self).add_usage(
1870             usage, actions, groups, prefix)
1871
1872     def _format_action_invocation(self, action):
1873         if not action.option_strings or action.nargs == 0:
1874             return super()._format_action_invocation(action)
1875         default = self._get_default_metavar_for_optional(action)
1876         args_string = self._format_args(action, default)
1877         return ', '.join(action.option_strings) + ' ' + args_string
1878
1879 def main(argv):
1880
1881     inputfile = ''
1882     outputdir = './'
1883
1884     mode = ''
1885
1886     help_s = """Aurora Integrated Workflow v0.2 (c) Copyright 2020, XMEN Lab -
         Universidade Federal de Campina Grande"""
1887
1888
1889     parser_arg = argparse.ArgumentParser(description=help_s, allow_abbrev=False,
         formatter_class=CapitalisedHelpFormatter)
1890
1891     parser_arg._positionals.title = 'Positional arguments'
1892
1893     parser_arg._optionals.title = 'Optional arguments'
1894
1895     parser_arg.add_argument("mode", type=str, help="Choose the mode for
         generation (verif|fe)")
1896
1897     parser_arg.add_argument("-i","--ifile", type=str, required=True, help="Select
          the input configuration file", metavar="input_file")
1898
1899     parser_arg.add_argument("-o","--odir", type=str, required=False, help="Select
          the output directory", default='./', metavar="output_dir")
1900
1901     args = parser_arg.parse_args()
```

```
1902     mode = args.mode
1903     inputfile = args.ifile
1904     outputdir = args.odir
1905 
1906     if (mode == 'verif') :
1907 
1908         verification_path =  Path(outputdir)
1909 
1910         display_title_bar()
1911 
1912         verification_path = verification_path / 'verification'
1913         Path(verification_path).mkdir(parents=True, exist_ok=True)
1914         doc_path = verification_path / 'docs'
1915         Path(doc_path).mkdir(parents=True, exist_ok=True)
1916         tb_path = verification_path / 'tb'
1917         Path(tb_path).mkdir(parents=True, exist_ok=True)
1918 
1919         Path(verification_path / 'logs').mkdir(parents=True, exist_ok=True)
1920         Path(verification_path / 'reports').mkdir(parents=True, exist_ok=True)
1921         Path(verification_path / 'scripts').mkdir(parents=True, exist_ok=True)
1922 
1923         Path(verification_path / 'scripts' / 'gatesim').mkdir(parents=True,
     exist_ok=True)
1924         Path(verification_path / 'scripts' / 'rtlsim').mkdir(parents=True,
     exist_ok=True)
1925         Path(verification_path / 'scripts' / 'verif_manager').mkdir(parents=True,
      exist_ok=True)
1926         Path(verification_path / 'scripts' / 'formal').mkdir(parents=True,
     exist_ok=True)
1927 
1928         Path(verification_path / 'src').mkdir(parents=True, exist_ok=True)
1929         Path(verification_path / 'vplan').mkdir(parents=True, exist_ok=True)
1930         Path(verification_path / 'workspace').mkdir(parents=True, exist_ok=True)
1931 
1932         Path(verification_path / 'formal').mkdir(parents=True, exist_ok=True)
1933         Path(verification_path / 'formal' / 'properties').mkdir(parents=True,
     exist_ok=True)
1934 
1935         print(Fore.BLUE + "# GENERATING DIRECTORIES IN " + str(verification_path)
      + "\n")
1936         print(Fore.BLUE + "#######################################################\n\n
     ")
1937 
1938         outputdir = outputdir + '/verification/tb'
1939         uParser = Parser('Parser', inputfile, outputdir)
1940 
1941         print(Fore.BLUE + "# GENERATING FILES \n")
1942         print(Fore.BLUE + "#######################################################\n\n
     ")
1943         uParser.parse_verif()
1944 
1945     elif (mode == 'fe'):
1946 
1947         frontend_path =  Path(outputdir)
1948 
1949         display_title_bar()
1950 
```

```
1951         frontend_path = frontend_path / 'frontend'
1952         Path(frontend_path).mkdir(parents=True, exist_ok=True)
1953         doc_path = frontend_path / 'docs'
1954         Path(doc_path).mkdir(parents=True, exist_ok=True)
1955 
1956 
1957         Path(frontend_path / 'logs').mkdir(parents=True, exist_ok=True)
1958         Path(frontend_path / 'logs' / 'gatesim').mkdir(parents=True, exist_ok=
    True)
1959         Path(frontend_path / 'logs' / 'lec').mkdir(parents=True, exist_ok=True)
1960         Path(frontend_path / 'logs' / 'power').mkdir(parents=True, exist_ok=True)
1961         Path(frontend_path / 'logs' / 'rtlsim').mkdir(parents=True, exist_ok=True
    )
1962         Path(frontend_path / 'logs' / 'synth_logic').mkdir(parents=True, exist_ok
    =True)
1963 
1964         Path(frontend_path / 'reports').mkdir(parents=True, exist_ok=True)
1965         Path(frontend_path / 'reports' / 'gatesim').mkdir(parents=True, exist_ok=
    True)
1966         Path(frontend_path / 'reports' / 'lec').mkdir(parents=True, exist_ok=True
    )
1967         Path(frontend_path / 'reports' / 'power').mkdir(parents=True, exist_ok=
    True)
1968         Path(frontend_path / 'reports' / 'rtlsim').mkdir(parents=True, exist_ok=
    True)
1969         Path(frontend_path / 'reports' / 'synth_logic').mkdir(parents=True,
    exist_ok=True)
1970 
1971         Path(frontend_path / 'scripts').mkdir(parents=True, exist_ok=True)
1972         Path(frontend_path / 'scripts' / 'gatesim').mkdir(parents=True, exist_ok=
    True)
1973         Path(frontend_path / 'scripts' / 'lec').mkdir(parents=True, exist_ok=True
    )
1974         Path(frontend_path / 'scripts' / 'power').mkdir(parents=True, exist_ok=
    True)
1975         Path(frontend_path / 'scripts' / 'rtlsim').mkdir(parents=True, exist_ok=
    True)
1976         Path(frontend_path / 'scripts' / 'synth_logic').mkdir(parents=True,
    exist_ok=True)
1977 
1978         Path(frontend_path / 'parasitics').mkdir(parents=True, exist_ok=True)
1979 
1980         Path(frontend_path / 'rtl').mkdir(parents=True, exist_ok=True)
1981         Path(frontend_path / 'rtl' / 'src').mkdir(parents=True, exist_ok=True)
1982         Path(frontend_path / 'rtl' / 'tb').mkdir(parents=True, exist_ok=True)
1983 
1984         Path(frontend_path / 'software').mkdir(parents=True, exist_ok=True)
1985         Path(frontend_path / 'software' / 'api').mkdir(parents=True, exist_ok=
    True)
1986         Path(frontend_path / 'software' / 'apps').mkdir(parents=True, exist_ok=
    True)
1987 
1988         Path(frontend_path / 'structural').mkdir(parents=True, exist_ok=True)
1989 
1990         Path(frontend_path / 'switching').mkdir(parents=True, exist_ok=True)
1991 
1992         Path(frontend_path / 'timing').mkdir(parents=True, exist_ok=True)
```

```
1993
1994            Path(frontend_path / 'workspace').mkdir(parents=True, exist_ok=True)
1995            Path(frontend_path / 'workspace' / 'gatesim').mkdir(parents=True,
        exist_ok=True)
1996            Path(frontend_path / 'workspace' / 'lec').mkdir(parents=True, exist_ok=
        True)
1997            Path(frontend_path / 'workspace' / 'power').mkdir(parents=True, exist_ok=
        True)
1998            Path(frontend_path / 'workspace' / 'rtlsim').mkdir(parents=True, exist_ok
        =True)
1999            Path(frontend_path / 'workspace' / 'synth_logic').mkdir(parents=True,
        exist_ok=True)
2000
2001            print(Fore.BLUE + "# GENERATING DIRECTORIES IN " + str(frontend_path) + "
        \n")
2002            print(Fore.BLUE + "#################################################\n\n")
2003
2004            outputdir = outputdir + '/frontend'
2005
2006            print(Fore.BLUE + "# GENERATING FILES \n")
2007            print(Fore.BLUE + "#################################################\n\n")
2008
2009            uParser = Parser('Parser', inputfile, outputdir)
2010            uParser.parse_fe()
2011
2012        else:
2013            pass
2014            # print (help_s)
2015            # sys.exit(2)
2016
2017
2018 if __name__== "__main__" :
2019     main(sys.argv[1:])
```

aurora.py

B TEMPLATES

```
/**
  ******************************************************************************
  * File automatic generated by XGeneratorTB software
  ******************************************************************************
**/
class |-TEST-| extends uvm_test;
    |-ENV-|
    |-SEQUENCE-|

    `uvm_component_utils(|-TEST-|)

    function new(string name, uvm_component parent = null);
        super.new(name, parent);
    endfunction

    virtual function void build_phase(uvm_phase phase);
        super.build_phase(phase);
        |-ENV_CREATION-|
        |-SEQUENCE_CREATION-|
    endfunction

    task run_phase(uvm_phase phase);
        super.run_phase(phase);
        |-SEQUENCE_START-|
    endtask: run_phase

endclass:|-TEST-|
```

test.tb

```
module v_|-MODULE-|(
    |-SIGNALS-|
);

  //Properties

  // Assertion

  //Cover


endmodule
```

module_assertions.tb

```
RTL_SRC = ../../frontend/rtl/src/
WRAPPER = ../../tb/|-MODULE-|_wrapper.sv

IF = |-INTERFACE-|
RTL := $(shell find $(RTL_SRC) -name '*.sv')
REFMOD =
```

```
7  PKGS = ../../tb/|-MODULE-|_pkg.sv
8
9  SEED = 100
10 COVER = 100
11 TRANSA = 5000
12
13 RUN_ARGS_COMMON = -access +r -input shm.tcl \
14           +uvm_set_config_int=*,recording_detail,1 -coverage all -covoverwrite
15
16 |-TEST-|
17
18 clean:
19   @rm -rf INCA_libs waves.shm rtlsim/* *.history *.log rtlsim/* *.key mdv.log imc
       .log imc.key ncvlog_*.err *.trn *.dsn .simvision/ xcelium.d simv.daidir *.so
       *.o *.err
20
21 view_waves:
22   simvision waves.shm &
23
24 view_cover:
25   imc &
```

makefile_verif.tb

```
1  /**
2    ******************************************************************************
3    * File automatic generated by XGeneratorTB software
4    ******************************************************************************
5  **/
6  module |-MODULE-| (
7          |-SIGNALS-|
8      );
9
10
11
12
13 endmodule
```

module.fe

Demais arquivos disponíveis em https://github.com/JoseIuri/Aurora.

C BLOCO ARITMÉTICO

```
/**
  ******************************************************************************
  * File automatic generated by Aurora software
  ******************************************************************************
**/
module arith (
        input logic pclk,
    input logic presetn,
    input logic [31:0] paddr,
    input logic [31:0] pwdata,
    input logic  psel,
    input logic  pwrite,
    input logic  penable,
    output logic [31:0] prdata,
    output logic  pslverr,
    output logic  pready
    );

  localparam IDDLE = 3'd0,
         W_ENABLE = 3'd1,
         R_ENABLE = 3'd2;


  reg [31:0] reg_bank [0:3];

  reg [1:0] state;


  always @(posedge pclk or negedge presetn) begin
    if (presetn == 0) begin
      state <= IDDLE;
      prdata <= 32'd0;
      pslverr <= 0;
      pready <= 0;

      reg_bank[0] <= 31'd0;
      reg_bank[1] <= 31'd0;
      reg_bank[2] <= 31'd0;
      reg_bank[3] <= 31'd0;
    end

    else begin
      case (state)
      IDDLE: begin
        if (psel) begin
          if (pwrite) begin
            state <= W_ENABLE;
          end
          else begin
            state <= R_ENABLE;
          end
```

```
52            pslverr <= 0;
53            pready <= 0;
54            prdata <= prdata;
55          end
56        end
57        W_ENABLE: begin
58          if (psel && pwrite) begin
59            if (paddr < 32'd12) begin
60              reg_bank[paddr/4]  <= pwdata;
61              pready <= 1;
62              pslverr <= 0;
63            end
64            else begin
65              pready <= 1;
66              pslverr <= 1;
67            end
68
69            if (paddr < 32'd12 && paddr >= 32'd8) begin
70              if (reg_bank[2][31]) begin
71                if (reg_bank[2][0]) begin
72                  reg_bank[3] <= reg_bank[2] + reg_bank[1];
73                end
74                else begin
75                  reg_bank[3] <= reg_bank[2] * reg_bank[1];
76                end
77              end
78            end
79          end
80          state <= IDDLE;
81        end
82
83        R_ENABLE: begin
84          if (psel && !pwrite) begin
85            if (paddr < 32'd16) begin
86              prdata <= reg_bank[paddr/4];
87              pready <= 1;
88              pslverr <= 0;
89            end
90            else begin
91              pready <= 1;
92              pslverr <= 1;
93            end
94          end
95          state <= IDDLE;
96        end
97
98        endcase
99      end
100   end
101 endmodule
```

arith.sv

```
1  module v_arith(
2      input   pclk,
3    input   presetn,
4    input [31:0] paddr,
```

```
5   input [31:0] pwdata,
6   input  psel,
7   input  pwrite,
8   input  penable,
9   input  pslverr,
10  input [31:0] prdata,
11  input  pready
12 );
13
14  //Properties
15
16  property apb_minimum_reset_p;
17    @(posedge pclk)
18      $fell(presetn) |->  ##[1:$] presetn == 1;
19  endproperty
20
21  // Checks for X or Z
22
23  property apb_sel_valid_values_p;
24    @(posedge pclk) disable iff(!presetn)
25      $isunknown(psel) == 0;
26  endproperty
27
28  property apb_addr_valid_values_p;
29    @(posedge pclk) disable iff(!presetn)
30      $isunknown(paddr) == 0;
31  endproperty
32
33  property apb_write_valid_values_p;
34    @(posedge pclk) disable iff(!presetn)
35      $isunknown(pwrite) == 0;
36  endproperty
37
38  property apb_enable_valid_values_p;
39    @(posedge pclk) disable iff(!presetn)
40      $isunknown(penable) == 0;
41  endproperty
42
43  property apb_ready_valid_values_p;
44    @(posedge pclk) disable iff(!presetn)
45      $isunknown(pready) == 0;
46  endproperty
47
48  property apb_slverr_valid_values_p;
49    @(posedge pclk) disable iff(!presetn)
50      $isunknown(pslverr) == 0;
51  endproperty
52
53  //Checks for reset values
54
55  property apb_sel_post_reset_p;
56    @(posedge pclk)
57      $rose(presetn) |->  $past(psel) == 0;
58  endproperty
59
60  property apb_enable_post_reset_p;
61    @(posedge pclk)
```

```
62      $rose(presetn) |-> $past(penable) == 0;
63    endproperty
64
65    property apb_slverr_post_reset_p;
66      @(posedge pclk)
67        $rose(presetn) |-> $past(pslverr) == 0;
68    endproperty
69
70    property apb_rdata_post_reset_p;
71      @(posedge pclk)
72        $rose(presetn) |-> $past(prdata) == 0;
73    endproperty
74
75    //PROTOCOL Checks
76
77    property apb_sel_validity_during_transfer_phases_p;
78      @(posedge penable) (psel);
79    endproperty
80
81    property apb_sel_stability_during_transfer_p;
82      @(posedge pclk) disable iff(!presetn)
83        penable |-> $stable(psel);
84    endproperty
85
86    property apb_sel_minimum_time_p; // Sel must be asserted for 2 cycles minimum
87      @(posedge pclk) disable iff(!presetn)
88        $rose(psel) |-> not(##1 $fell(psel));
89    endproperty
90
91    property apb_enable_fall_towards_sel_fall_p; // De-assert enable and sel
        together
92      @(posedge pclk) disable iff(!presetn)
93        $fell(psel) |-> $past(penable) == 1 && penable == 0;
94    endproperty
95
96    property apb_addr_stability_during_transfer_p; // Stability of ADDR during
        transfer
97      @(posedge pclk) disable iff(!presetn)
98        (psel) and penable |-> ($stable(paddr) and ##[1:$] $fell(penable));
99    endproperty
100
101   property apb_write_stability_during_transfer_p; //Stability of write during
        transfer
102     @(posedge pclk) disable iff(!presetn)
103       (psel) and penable |-> ($stable(pwrite) and ##[1:$] $fell(penable));
104   endproperty
105
106   property apb_wdata_stability_during_transfer_p; //Stability of wdata during
        transfer
107     @(posedge pclk) disable iff(!presetn)
108       (psel) and penable and pwrite |-> ($stable(pwdata) and ##[1:$] $fell(
        penable));
109   endproperty
110
111   property apb_enable_assertion_time_p; //Enable 1 cycle after sel
112     @(posedge pclk) disable iff(!presetn)
113       (psel) and !penable |=> (penable);
```

```
114  endproperty
115
116  property apb_enable_stability_during_ready_changes_p; //Stability of enable
       during ready
117    @(posedge pclk) disable iff(!presetn)
118      !pready and penable |=> penable;
119  endproperty
120
121  property apb_enable_value_between_transfers_p;
122    @(posedge pclk) disable iff(!presetn)
123      !(psel) |-> !penable;
124  endproperty
125
126  property apb_enable_deassertion_time_p; // De-assert after transfer
127    @(posedge pclk) disable iff(!presetn)
128      (psel) and pready and penable |=> !penable;
129  endproperty
130
131  property  apb_ready_low_maximum_time_p; // Ready time
132    @(posedge pclk) disable iff(!presetn)
133      (psel) and $rose(penable) and !pready |-> ##[1:$] pready;
134  endproperty
135
136  property apb_rdata_stability_during_transfer_p; // Stability of RData during
       transfer
137    @(posedge pclk) disable iff(!presetn)
138      (!pwrite) and (psel) and penable and (pready==$past(pready)) and (!pslverr)
       |-> not($changed(prdata));
139  endproperty
140
141  property apb_slverr_value_condition_p; //Checks stability of pslverr
142    @(posedge pclk) disable iff(!presetn)
143      !(psel) or !penable or !pready |-> !pslverr;
144  endproperty
145
146  property apb_slverr_assertion_time_p;  //Time for pslverr
147    @(posedge pclk) disable iff(!presetn)
148      $past(pslverr) == 0 and $rose(pslverr) |=> $fell(pslverr);
149  endproperty
150
151  property apb_slverr_illegal_read_addr_p; //Try to read from illegal adress
152    @(posedge pclk) disable iff(!presetn)
153      !pwrite and $rose(pready) and paddr >= 16 |-> $rose(pslverr);
154  endproperty
155
156  property apb_slverr_illegal_write_addr_p; //Try to write in illegal adress
157    @(posedge pclk) disable iff(!presetn)
158      pwrite and $rose(pready) and paddr >= 12 |-> $rose(pslverr);
159  endproperty
160
161
162  //Asserions
163
164  APB_MINIMUN_RESET_VALUE: assume property (apb_minimum_reset_p);
165
166  // APB_ILLEGAL_SEL_VALUE_ERR: assert property (apb_sel_valid_values_p);
167
```

```
168  // APB_ILLEGAL_ADDR_VALUE_ERR: assert property (apb_addr_valid_values_p);
169
170  // APB_ILLEGAL_WRITE_VALUE_ERR: assert property (apb_write_valid_values_p);
171
172  // APB_ILLEGAL_ENABLE_VALUE_ERR: assert property (apb_enable_valid_values_p);
173
174  APB_ILLEGAL_READY_VALUE_ERR: assert property (apb_ready_valid_values_p);
175
176  APB_ILLEGAL_SLVERR_VALUE_ERR: assert property (apb_slverr_valid_values_p);
177
178
179  APB_ILLEGAL_SEL_VALUE_POST_RESET_ERR: assume property (apb_sel_post_reset_p);
180
181  APB_ILLEGAL_ENABLE_VALUE_POST_RESET_ERR: assume property (
       apb_enable_post_reset_p);
182
183  APB_ILLEGAL_SLVERR_VALUE_POST_RESET_ERR: assert property (
       apb_slverr_post_reset_p);
184
185  APB_ILLEGAL_RDATA_VALUE_POST_RESET_ERR: assert property (apb_rdata_post_reset_p
       );
186
187
188  APB_ILLEGAL_SEL_TRANSITION_TR_PHASES_ERR: assume property (
       apb_sel_validity_during_transfer_phases_p);
189
190  APB_ILLEGAL_SEL_TRANSITION_DURING_TRANSFER_ERR: assume property (
       apb_sel_stability_during_transfer_p);
191
192  APB_ILLEGAL_SEL_MINIMUM_TIME_ERR: assume property(apb_sel_minimum_time_p);
193
194  APB_ILLEGAL_ENABLE_FALL_TOWARDS_SEL_FALL_ERR: assume property(
       apb_enable_fall_towards_sel_fall_p);
195
196  APB_ILLEGAL_ADDR_TRANSITION_DURING_TRANSFER_ERR: assume property (
       apb_addr_stability_during_transfer_p);
197
198  APB_ILLEGAL_WRITE_TRANSITION_DURING_TRANSFER_ERR: assume property (
       apb_write_stability_during_transfer_p);
199
200  APB_ILLEGAL_WDATA_TRANSITION_DURING_TRANSFER_ERR: assume property (
       apb_wdata_stability_during_transfer_p);
201
202  APB_ILLEGAL_ENABLE_ASSERTION_TIME_ERR: assume property (
       apb_enable_assertion_time_p);
203
204  APB_ILLEGAL_ENABLE_TRANSITION_DURING_READY_CHANGES_ERR: assume property (
       apb_enable_stability_during_ready_changes_p);
205
206  APB_ILLEGAL_ENABLE_VALUE_BETWEEN_TRANSFERS_ERR: assume property(
       apb_enable_value_between_transfers_p);
207
208  APB_ILLEGAL_ENABLE_DEASSERTION_TIME_ERR: assume property(
       apb_enable_deassertion_time_p);
209
210  APB_ILLEGAL_READY_MAXIMUM_LOW_TIME_ERR: assert property (
       apb_ready_low_maximum_time_p);
```

```
211
212  APB_ILLEGAL_RDATA_TRANSITION_DURING_TRANSFER_ERR: assert property (
       apb_rdata_stability_during_transfer_p);
213
214  APB_ILLEGAL_SLVERR_VALUE_CONDITION_ERR: assert property (
       apb_slverr_value_condition_p);
215
216  APB_ILLEGAL_SLVERR_ASSERTION_TIME_ERR: assert property(
       apb_slverr_assertion_time_p);
217
218  APB_ILLEGAL_SLVERR_READ_ERR: assert property(apb_slverr_illegal_read_addr_p);
219
220  APB_ILLEGAL_SLVERR_WRITE_ERR: assert property(apb_slverr_illegal_write_addr_p);
221
222
223  // Cover
224
225  // APB_ILLEGAL_SEL_VALUE_CVR: cover property (apb_sel_valid_values_p);
226
227  // APB_ILLEGAL_ADDR_VALUE_CVR: cover property (apb_addr_valid_values_p);
228
229  // APB_ILLEGAL_WRITE_VALUE_CVR: cover property (apb_write_valid_values_p);
230
231  // APB_ILLEGAL_ENABLE_VALUE_CVR: cover property (apb_enable_valid_values_p);
232
233  APB_ILLEGAL_READY_VALUE_CVR: cover property (apb_ready_valid_values_p);
234
235  APB_ILLEGAL_SLVERR_VALUE_CVR: cover property (apb_slverr_valid_values_p);
236
237
238
239  // APB_ILLEGAL_SEL_VALUE_POST_RESET_CVR: cover property (apb_sel_post_reset_p);
240
241  // APB_ILLEGAL_ENABLE_VALUE_POST_RESET_CVR: cover property (
       apb_enable_post_reset_p);
242
243  APB_ILLEGAL_SLVERR_VALUE_POST_RESET_CVR: cover property (
       apb_slverr_post_reset_p);
244
245  APB_ILLEGAL_RDATA_VALUE_ERR: cover property (apb_rdata_post_reset_p);
246
247
248  // APB_ILLEGAL_SEL_TRANSITION_TR_PHASES_CVR: cover property (
       apb_sel_validity_during_transfer_phases_p);
249
250  // APB_ILLEGAL_SEL_TRANSITION_DURING_TRANSFER_CVR: cover property (
       apb_sel_stability_during_transfer_p);
251
252  // APB_ILLEGAL_SEL_MINIMUM_TIME_CVR: cover property(apb_sel_minimum_time_p);
253
254  // APB_ILLEGAL_ENABLE_FALL_TOWARDS_SEL_FALL_CVR: cover property(
       apb_enable_fall_towards_sel_fall_p);
255
256  // APB_ILLEGAL_ADDR_TRANSITION_DURING_TRANSFER_CVR: cover property (
       apb_addr_stability_during_transfer_p);
257
258  // APB_ILLEGAL_WRITE_TRANSITION_DURING_TRANSFER_CVR: cover property (
```

```
      apb_write_stability_during_transfer_p);
259
260   // APB_ILLEGAL_WDATA_TRANSITION_DURING_TRANSFER_CVR: cover property (
        apb_wdata_stability_during_transfer_p);
261
262   // APB_ILLEGAL_ENABLE_ASSERTION_TIME_CVR: cover property (
        apb_enable_assertion_time_p);
263
264   // APB_ILLEGAL_ENABLE_TRANSITION_DURING_READY_CHANGES_CVR: cover property (
        apb_enable_stability_during_ready_changes_p);
265
266   // APB_ILLEGAL_ENABLE_VALUE_BETWEEN_TRANSFERS_CVR: cover property(
        apb_enable_value_between_transfers_p);
267
268   // APB_ILLEGAL_ENABLE_DEASSERTION_TIME_CVR: cover property(
        apb_enable_deassertion_time_p);
269
270   APB_ILLEGAL_READY_MAXIMUM_LOW_TIME_CVR: cover property (
        apb_ready_low_maximum_time_p);
271
272   APB_ILLEGAL_RDATA_TRANSITION_DURING_TRANSFER_CVR: cover property (
        apb_rdata_stability_during_transfer_p);
273
274   APB_ILLEGAL_SLVERR_VALUE_CONDITION_CVR: cover property (
        apb_slverr_value_condition_p);
275
276   APB_ILLEGAL_SLVERR_ASSERTION_TIME_CVR: cover property(
        apb_slverr_assertion_time_p);
277
278   APB_ILLEGAL_SLVERR_READ_CVR: cover property(apb_slverr_illegal_read_addr_p);
279
280   APB_ILLEGAL_SLVERR_WRITE_CVR: cover property(apb_slverr_illegal_write_addr_p);
281
282 endmodule
```

v_arith.sva