UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de São José do Rio Preto

Renan Galeane Alboy

Geração de simulador de eventos discretos aplicado a circuitos sequenciais

São José do Rio Preto
2016

Renan Galeane Alboy

Geração de simulador de eventos discretos aplicado a circuitos sequenciais

Monografia apresentada ao Departamento de Ciências de Computação e Estatística do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", como parte dos requisitos necessários para aprovação na disciplina Projeto Final.

Orientador:

Prof. Dr. Aleardo Manacero Jr.

São José do Rio Preto
2016

Renan Galeane Alboy

Geração de simulador de eventos discretos aplicado a circuitos sequenciais

Monografia apresentada ao Departamento de Ciências de Computação e Estatística do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", como parte dos requisitos necessários para aprovação na disciplina Projeto Final.

Prof. Dr. Aleardo Manacero Jr.

Renan Galeane Alboy

Banca Avaliadora:

Prof.ª Dr.ª Renata Spolon Lobato

Prof. Dr. Rodrigo Capobianco Guido

São José do Rio Preto
2016

"Não enfrente um monstro sob pena de tornares um deles, e se contempla o abismo, a ti o abismo também contempla."

-Friedrich Nietzsche

Agradecimentos

Primeiramente gostaria de agradecer aos meus pais e irmão, Celso, Marcia e Felipe, pelo apoio dado duranto todo processo até o ponto atual. Também os meus avós, parentes (Antônio, Lourdes, Antónia e Célia) e a Vitória. Essas pessoas que sempre deram força nos momentos desde os momentos mais tranquilos até os mais turbulentos.

Agadeço também ao Professor Aleardo e a Professora Renata, pela orientação tanto da iniciação científica quanto ao trabalho de conclusão de curso, fornecedo estrutura laboratorial, auxiliando no desenvolvimento do trabalho e também puxando a orelha quando necessário. Também gostaria de agradeçer aos amigos de laboratório e da faculdade, Bruno (boxexa), Matheus, Fernanda, Chen, Mario (Jabuca), Lucas (Tesouro) e Fábio.

Renan Galeane Alboy

Resumo

Simuladores são amplamente usados como ferramentas para analises, avaliacão ou validação de diferentes sistemas ou ambientes. Seu uso reduz o tempo e custo necessário para desenvolver um novo produto porque podemos simular desde as primeiras fases do projeto. Entretanto, a maioria dos simuladores disponíveis não são fáceis de usar ou não são projetados para necessidades específicas. Neste trabalho é apresentado uma ferramenta que pode criar um simulador de eventos discretos, com interface gráfica, a partir de formulação relativamente simples, dada por seu usuário. A especificação dos parâmetros exigidos para criar simuladores de sistemas baseados em funções de transferência tais como circuitos sequenciais, é apresentado aqui.

Palavras-chave: Simulação. Circuitos sequenciais.

Abstract

Simulators are widely used as tools to analyse, evaluate or validate different systems or even environments. Their use can reduce the time and coast needed to develop a new product since simulations can be performed at very early phases of project. However, most of the available simulation tools are not easy to use or tailored to specific needs. In this work we present a tool that can create a discrete-event simulator, with a GUI-based interface, from relatively simple formulation given by its user. The specification of the required parameters to create simulators systems characterized by transfer function, such as sequential circuits, is present here.

Keywords: Simulation. Sequential circuits.

Sumário

Lista de Figuras

Capítulo 1

Introdução

A simulação de sistemas discretos é uma área de grande importância em computação. Isso decorre do fato de inúmeras áreas do conhecimento fazerem uso de simulação para a avaliação de produtos, ambientes ou mesmo situações. Em muitas áreas os sistemas a serem avaliados são descritos ou modelados por meio de caracteristicas discretas, ou por estados discretos, ou seja, sem evolução contínua no tempo. Sistemas desse tipo são simulados por simuladores de eventos discretos.

Entretanto, em muitas situações quem deseja analisar um sistema através de um simulador não é um especialista em computação. Logo é interessante que os simuladores disponíveis a ele possam ser utilizados de forma simples, sem necessidar codificação. Infelizmente essa não é a realidade em muitas áreas nas quais a modelagem depende da escrita de códigos ou em linguagens tradicionais de programação ou orientadas para simulação.

Este trabalho faz parte de um projeto de uma ferramenta que gera simuladores de fácil uso, com interfaces icônicas. Essa ferramenta também funciona com interfaces gráficas para o usuário gerar o simulador a partir de características dos sistemas a serem simulados.

1.1 Objetivos

O trabalho tem como objetivo o desenvolvimento de parte do Yasc (Yes, a simulator compiler), que é um gerador de simuladores para diferentes contextos. Em particular será tratada a geração de simuladores cuja interação entre os objetos é modelada por transição de estados, como ocorre na simulação de circuitos sequenciais.

Com isso, o desenvolvimento desta ferramenta visa permitir ao usuário a construção

de seu próprio simulador de maneira simples e que este ainda seja fácil de usar.

1.2 Motivação

Atualmente o uso de ferramentas de simulação para avaliar o funcionamento de sistemas, das mais diversas áreas, é muito comum. O uso de simuladores permite reduzir o custo do projeto, desenvolvimento e verificação de sistemas, uma vez que podem ser usados até mesmo antes da construção de um protótipo.

Entretanto, o uso de simuladores é dificultado por diferentes motivos. Um problema é que parte das ferramentas é difícil de usar, exigindo conhecimento técnico de programação na elaboração dos modelos avaliados. Um outro problema é que nem sempre tratam dos aspectos que necessitam ser avaliados, ou exigindo informações demais ou não tratando alguns parâmetors. Um último incoveniente é a disponibilidade das ferramentas, uma vez que muitos dos simuladores descritos na literatura ou não são mais encontrados ou são ferramentas comerciais de alto custo.

1.3 Justificativa

Os problemas apontados na seção anterior justificam a criação de uma ferramenta que permite a um analista, de qualquer área do conhecimento, criar um simulador ajustado aos seus propósitos. Isso é verdade também em muitas áreas em que o sistema seja modelado por transição de estado, como é o caso de circuitos sequenciais digitais, que serão o caso de estudo desse trabalho.

1.4 Exequibilidade

O projeto é exequível pois os conceitos necessários fazem parte de diversas disciplinas da graduação. Além disso não há necessidade de equipamentos que já não estejam disponíveis no Laboratório de Sistemas Paralelos e Distribuídos, onde ele é desenvolvido.

1.5 Metodologia

Para o desenvolvimento do projeto serão utilizados a linguagem Java e o sistema operacional Linux. A implementação dos componentes necessários será feita seguindo-se a metodologia de projeto orientado a objeto e sua fundamentação teórica vem do estudo de simuladores, técnicas de compilação e de sistemas baseados em transições de estados.

Os testes de validação ocorrerão em duas etapas. Uma para validar a implementação e outra para verificar a usabilidade da ferramenta, que ocorrerá com testes com alunos do curso de computação, que deverão gerar simuladores de circuitos digitais.

1.6 Organização do texto

Além deste capítulo introdutório, o capítulo 2 traz uma análise teórica dos simuladores estudados, do iSPD, de técnicas de simulação por templates e uma revisão sobre circuitos sequenciais, geração de simuladores baseados em transição de estados, que é o foco deste projeto.

Capítulo 2

Revisão Bibliográfica

Neste capitulo será descrito a fundamentação teórica para o desenvolvimento do trabalho. Contando com as principais definições, como conceitos de sistemas, modelos, os tipos de simulação, suas vantagens e desvantagens. Também abordadno o uso de templates-based na simulação, exemplificado alguns aplicativos que já o utilizam.

2.1 Simulação

Simulação é a técnica de imitar o comportamento de um sistema por meio de uma situação análoga para obter informações de maneira mais eficiente (Choi and Kang, 2013). Assim, a simulação é a imitação de um processo do mundo real ao longo de um dado intervalo de tempo. Neste sentido a simulação envolve a criação de uma história artificial de um sistema e a observação do que acontece com a história ficcional para fazer inferências e obter características do sistema real (Banks et al., 2010).

O uso de simulação apresenta diversas vantagens, como por exemplo:

- Formas de trabalho, políticas, regras, organizações de processos podem ser planejadas e testadas sem afetar o sistema em funcionamento;
- Podem ser obtidos importantes dados em relação ao desempenho de um sistema;
- Testes com hardwares podem ser feitos sem a necessidade de gastos com os mesmos;
- A simulação pode ser usada como um meio de ensino;

A partir ddesses conceitos é importante caracterizar dois entes definidores de uma simulação, que são o sistema a ser simulado e o modelo que o representa.

2.1.1 Sistema

Sistema é o objeto de estudo de uma simulação (Banks et al., 2010), podendo ser definido como uma coleção ordenada logicamente e relacionada a princípios, fatos e objetos (Pooch and Wall, 1993). Os elementos deste sistema interagem entre si, ou trabalham individualmente, para realizarem uma ação. Um exemplo de sistema pode ser observado em uma fábrica de automóveis, em que cada máquina realiza a colocação de uma peça, interagindo entre si, para que no fim tenha um carro completo.

Neste caso a coleção de objetos que seriam usados na simulação desta fábrica seriam as máquinas, e com o decorrer da simulação poderiam ser gerados resultados como o tempo total para a produção do carro ou outras métricas de saída.

2.1.2 Modelo

Modelo pode ser definido como a representação mais simples de um sistema. Sendo uma representação do sistema, um modelo poede não representar exatamente ele todo, mas sim, apenas a parte que será estudada (Banks et al., 2010). Porém, mesmo se tratando de uma parte do todo ele deve fornecer detalhes suficientes para que possam ser obtidas informações com condições de serem aplicadas ao sistema real.

2.2 Simulação baseada em *templates*

No processo de simulação é necessário resolver dois problemas: identificação de características relevantes do sistema e criação do modelo adequado. Enquanto o primeiro-problema é conceitual e depende do conhecimento do analista sobre a área tratada, o segundo depende da ferramenta de simulação usada. Infelizmente, como apontado no capítulo 1, a maioria dos simuladores existentes apresentamproblemas para a modelagem, ou pela necessidade de programação ou pela inadequação dos objetos disponíveis para criar o modelo.

Uma abordagem para reduzir esse problema é fazer uso do que se chama "simulação baseada em *templates*"(Guru and Savory, 2004), criando o conceito de uma simulação genérica. Assim, a simulação genérica ou baseada em modelos consiste em oferecer uma série de recursos prontos para o usuário, sejam esses recursos módulos, objetos já modelados ou situações comuns para o tipo de simulação abordada. Isso permite que o usuário ative ou desativo parâmetros do template, adequando o modelo a melhor forma de uso para a simulação que será realizada.

Desta maneira, a simulação baseada em templates elimina a necessidade de recriar modelos para situações semelhantes, permitindo ao usuário maior praticidade e agilidade para a realização da simulação.

2.2.1 Ferramentas baseadas em templates

O uso de templates se aproxima da idéia de gerar simuladores de interface icônica adaptados ao problema que o usuário quer resolver. Assim são apresentados a seguir alguns simuladores propostos com essa abordagem.

- **KanbanSIM:** Sua finalidade é simular rotas de entrega de produtos, escalonamento de transporte, simulando, inclusive, o impacto de possíveis atrasos (KanbanSIM, 2016).

  É um ferramenta usada com um template genérico de simulação. Utiliza interface em Excel, sendo também reconfigurável.

- **PDSIM (Program Development SIMulator):** É uma ferramenta mais específica, com foco em resolver problemas de gestão de recursos em empresas de engenharia e gerenciamento de programas para grandes empresas.

  Assim como o KanbanSIM, ele é um simulador baseado em modelo (templates). O simulador permite que o modelo seja facilmente reconfigurado para qualquer usuário, com pouco esforço e baixo custo, quando comparado a desenvolver um simulador do zero (PDSIM, 2016).

- **PowerTrainSIM:** Ele é voltado para a simulação da manufatura e testes de componentes de tração automotivos. O uso de templates é possível, apesar das diferenças físicas entre fábricas, porque a fabricação e montagem dos componentes é semelhante (PowerTrainSIM, 2016).

  É outro simulador baseado em modelos, ou seja, templates. Além disso, ele também permite a construção de uma aplicação personalizada para uma situação específica de maneira fácil, rápida e com menor custo do que uma começada do zero.

2.3 O iSPD

O iSPD (Menezes et al., 2012) é o simulador de grades computacionais desenvolvido pelo Grupo de Sistemas Paralelos e Distribuidos. É uma ferramenta desenvolvida com

a linguagem Java que tenta trazer para o usuário maior facilidade na modelagem do sistema a ser simulado, com o uso de uma interface icônica. O iSPD é dividido em quatro módulos básicos, figura 2.1, que são:

- **Interface Icônica:** Em que o usuário modela o sistema usando ícones que representam máquinas, clusters e canais de comunicação. Nessa interface o usuário configura os parâmetros dos objetos que compõe a grade e também a carga de trabalho a qual será submetido.

- **Interpretador de modelos:** Este módulo faz a conversão de modelos entre a interface icônica e o motor de simulação que usa uma linguagem de filas. Adicionalmente este módulo faz a conversão de modelos de outros simuladores de grade para executar no iSPDpermitindo o aproveitamento daqueles modelos.

- **Motor de simulação:** Realiza a simulação propriamente dita. Ele faz a leitura do modelo de filas gerando os resultados e disponibilizando-os ao usuário pela interface icônica. Seu funcionamento é baseado no atendimento de eventosem centros de serviço sendo que os eventos são gerados ou pelo atendimento das filas ou por alocação gerada pelos meta-escalonadores da grade.

- **Gerados de escalonadores:** Na verdade é composto por dois componentes, um que gerencia os escalonadores disponíveis para a grade e outro que gera novos escalonadores. Deve-se notar que a possibilidade de controlar quais políticas de escalonamento serão utilizadas nos meta-escalonadores da grade é de vital importância para a abrangência de sua modelagem.

Assim, para melhor explicar o iSPD, segue a figura 2.1 em que é mostrado um diagrama do simulador. Nele é exibidos eus módulos e de que maneiras se relacionam.

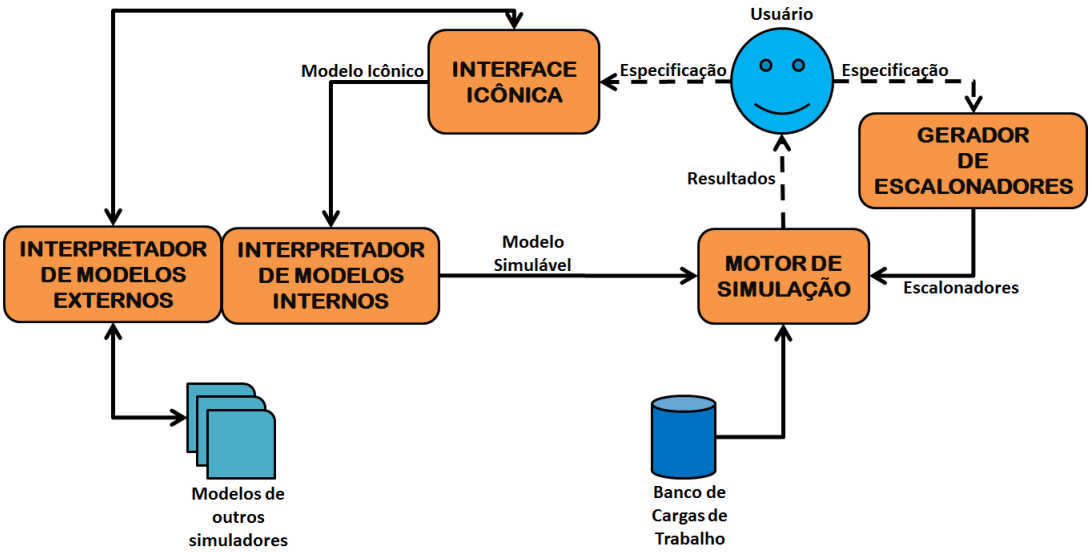


Figura 2.1: Diagrama conceitual do iSPD (Menezes et al., 2012)

2.4 Simulação de circuitos sequenciais

Os simuladores aqui descritos, tanto os baseados em templates quanto o iSPD, apresentam como característica comum o fato de seus modelos poderem ser transformados em sistemas de filas. Porém existem aplicações em que o modelo de filas nãao é aplicavel. Nesse caso o modelo de funcionamento do sistema é baseado em transições de estados instantâneas, ou quase, sem a ocorrência de filas. Uma área em que isso ocorre é a de circuitos digitais, tanto combinacionais como sequênciais.

Como o foco do projeto é possibilitar a geração de simuladores desta classe e como caso de testes será feita a geração de um simulador de circuitos sequenciais, apresenta-se a seguir alguns simuladores encontrados na literatura.

- **Qucs (Quite circuit simulator):** É uma ferramenta de simulação gratuita de circuitos sequenciais. Também apresenta seu código aberto e não se encontra em seu estado definitivo, pois seu desenvolvimento ocorre de maneira colaborativa. A modelagem de um circuito pode ser feita tanto pela interface gráfica como por linha de comando (circuit simulator, 2015).

- **Circuit Maker:** É uma ferramenta para a simulação na área de eletrônicos, incluindo a simulação de circuitos sequênciais. É distribuido de maneira gratuita para uso e de código aberto (Maker, 2015).

- **SPICE (Simulation Program for Integrated Circuits Emphasis):** É uma ferramenta para propósitos gerais de simulação de circuitos, utilizada tanto para o designing de circuitos como para prever o comportamento deles. É distribuida de maneira gratuita e está disponível tanto para download como para uso online via browser (SPICE, 2016).

  Apesar de aparentar uma modelagem simples apresenta a necessidade de conhecimentos específicos para a caracterização do objeto e desecrição da operação que deseja ser realizada. Já que tal descrição é realizada através de códigos.

- **TINA:** É uma ferramenta que funciona como um motor de geração para simuladores descritos na linguagem Verilog. Assim, é capaz de realizar simulação de circuitos sequênciais. Entretanto, exige que o usuário tenha conhecimento da linguagem de programação que será utilizada (TINA, 2016).

2.5 Considerações finais

Neste capítulo apresentou-se os conceitos utilizadas para construir uma base teorica para o desenvolvimento do projeto. Estudou-se também ferramentas de simulação baseada em template e de circuitos sequenciais já existentes, assim, colocando sua função e especificidade a mostra. O capitulo 3 falará das atividades desenvolvidas ao longo do projeto.

Capítulo 3

Desenvolvimento do Yasc

Neste capítulo é apresentado o trabalho desenvolvido dentro do projeto do Yasc. Será dada ênfase aos aspectos relativos aos componentes para simulação de sistemas baseados em transições de estado.

3.1 O YASC (Yes, a simulator compiler)

O Yasc é um gerador de simuladores de sistemas de eventos discretos que, a partir de informações sobre eventos e componentes de uma classe de sistemas gera um simulador com interface icônica para o contexto desta classe. Como sistemas de eventos discretos incluem duas grandes classes de modos de execução, ou seja, sistemas baseados em filas e sistemas baseados em transições de estado, o projeto do Yasc deve atender aos dois modos. Neste trabalho são tratados sistemas baseados em transições de estado, incluíndo por exemplo a simulação de circuitos digitais sequenciais.

Como o objetivo do Yasc é ser uma ferramenta de fácil uso, adotou-se um padrão de interfaces gráficas para a interação com o usuário. Assim, para que seja gerado um simulador qualquer o usuário é guiado através de um conjunto de janelas, nas quais informa que objetos tipicamente aparecem em sistemas do tipo a ser simulado, quais são os seus parâmetros característicos e como ocorrem relações entre eles.

A arquitetura geral do Yasc é apresentada na Figura 3.1, sendo que seus principais componentes são:

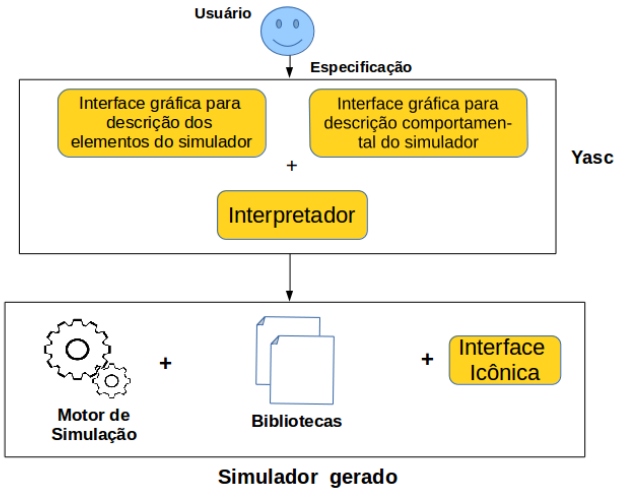


Figura 3.1: Arquitetura do Yasc Furlanetto (2016).

- **GUI para descrição de elementos gráficos do simulador:** É a interface inicial para criação dos objetos relevantes ao sistema a ser simulador. Nela o usuário nomeia o objeto, podendo também fornecer uma representação gráfica (um ícone) para utilização no simulador a ser gerado.

- **GUI para descrição comportamental do simulador:** Nesta interface se faz a descrição comportamental do objeto criado pela interface anterior. Nela se permite ao usuário descrever como o objeto se comporta, como por exemplo, se possuirá ou não filas, se apresentará transições instantâneas ou durativas, dentre outras possibilidades.

- **Interpretador:** Uma vez concluídas as descrições de todos os objetos ou eventos que possam ocorrer num sistema a ser modelado, o módulo interpretador cria o simulador que se quer gerar, sendo que o simulador é também composto por três componentes principais:

- **Biblioteca de objetos:** Contendo rotinas para o tratamento de cada tipo de evento e ou objeto do sistema a ser modelado.

- **Motor de simulação:** É gerado a partir de componentes de uma biblioteca pré-definida e dos objetos descritos pelo usuário, sendo responsável por executar as simulações de fato.

- **Interface icônica:** Também é gerada a partir de componentes pré-definidos, com a inserção dos objetos característicos do sistema a ser modelado e simulado.

O desenvolvimento do Yasc atacou inicialmente a simulação de sistemas baseados em filas. Isso direcionou a parametrização dos elementos das duas GUIs que compoem a ferramenta e, mais ainda, a forma de atuação do motor de simulação gerado. Isso ocorreu principalmente porque adotou-se o motor de simulação usado no iSPD Menezes et al. (2012), que é baseado em filas, como fonte dos componentes a serem montados para gerar o motor de simulação da aplicação para a qual se constrói o simulador. Para este trabalho é preciso incluir elementos que permitam a geração de simuladores de sistemas de transição de eventos, modificando a versão inicial do Yasc. Essas modificações são apresentadas na próxima seção.

3.2 Implementação do trabalho

Nesta seção se descreve o trabalho de implementação realizado no Yasc, abordando desde as interfaces para recebimento de parâmetros até a parte do motor que realiza a simulação proposta no trabalho.

3.2.1 Generalização do motor de simulação

O motor de simulação do iSPD era direcionado a filas e também continha componentes que tratavam explicitamente grades computacionais. A partir disso se aplicou um processo de reengenharia do motor de simulação de modo a separar componentes funcionais do motor e alterar os componentes que estivessem restritos a grades computacionais. Desta maneira, as partes que poderiam ser reutilizadas foram separadas, identificando primeiramente suas dependências e eliminando as suas particularidades.

Os componentes do motor de simulação para transições de estados serão implementados usando-se o conceito de tabela de estados. Assim, serão implementadas rotinas para manipulação de tabelas de estados, as quais deverão basicamente obedecer funções de transferência definidas na biblioteca gerada pelo Yasc. Com isso, o motor de simulação deverá armazenar as tabelas de estado, que serão modificadas sempre que um evento de transição ocorrer. Por exemplo, em um circuito sequencial os eventos de transição correspondem aos pulsos de relógio e a tabela de transição aos estados nas saídas/entradas de cada porta lógica. Assim, a cada evento “pulso” o motor de simulação avalia as entradas de cada porta modelada e altera sua saída atendendo à função de transferência definida para aquela porta.

3.2.2 Interfaces de recebimento de parâmetros

A definição do comportamento de cada elemento do sistema a ser modelado é feita através de parâmetros inseridos pelo especialista na área. Para viabilizar essa inserção foram implementadas diversas interfaces, que recebem os dados do usuário e transformam esses dados em um conjunto de regras. Estas regras são posteriormente interpretadas para a geração do simulador. As interfaces são trabalhadas sequencialmente, de forma a conduzir o processo de produção de cada elemento do futuro simulador. A primeira destas interfaces é mostrada na Figura 3.2, na qual se faz a opção pelo tipo de função de transferência que será usada para um dado elemento. No caso da simulação de sistemas baseados em transição de estados essas funções de transferência podem assumir o formato de expressão algébrica (uma fórmula) ou de tabela-verdade. Na interface vista em 3.2 são mostradas as opções para iniciar a criação do componente do simulador e também deve ser definida qual forma da representação da função de tranferência do mesmo.

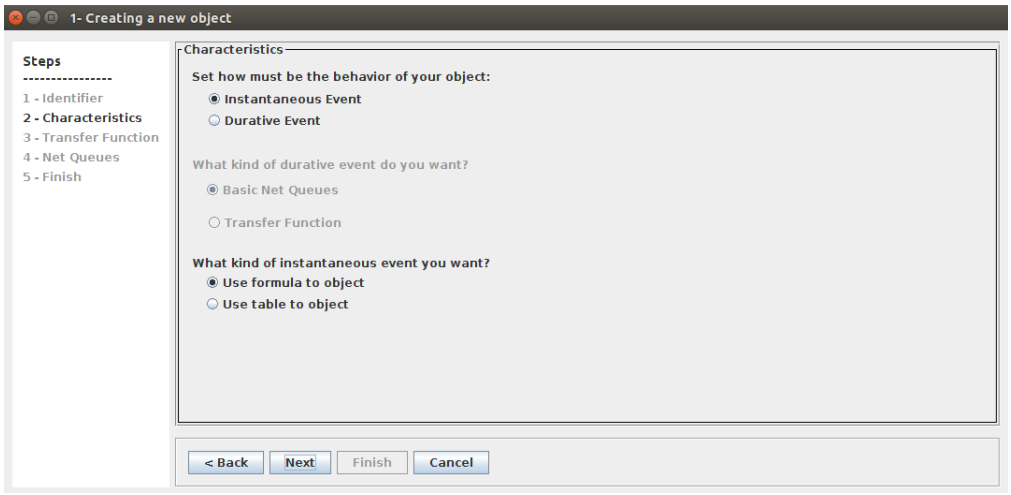


Figura 3.2: Interface de escolha do tipo de função de transferência.

Caso se opte por inserir a função de transferência por meio de expressão algébrica o usuário vai receber a interface vista na Figura 3.3. Nesta interface, o usuário pode criar váriaveis, escolhendo seu nome e tipo, e constantes para compor a função de transferência que traduz o comportamento do componente que está sendo criado. A expressão inserida pelo usuário será interpretada, possivelmente com a inclusão de operadores lógicos, para gerar um conjunto de rotinas que a implementem computacionalmente. Essa operação segue os procedimentos básicos de um compilador, que produzirá como "código executável" um conjunto de rotinas do simulador.

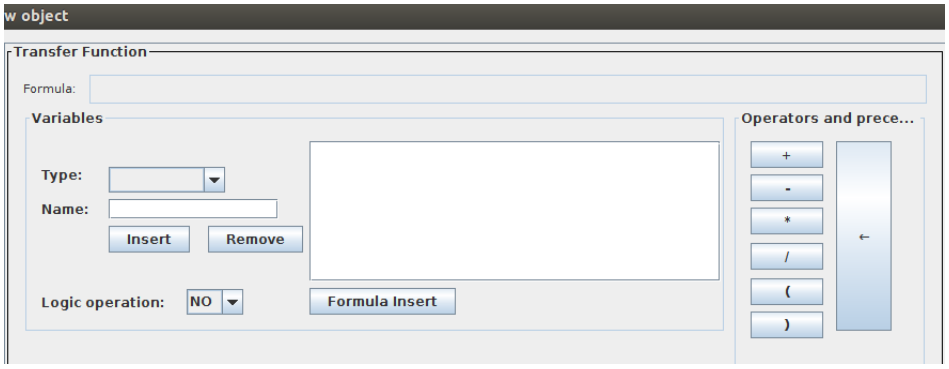


Figura 3.3: Interface de definição da fórmula.

Por outro lado, se a opção for por definir a função de transferência como sendo uma tabela-verdade, o usuário será levado para a interface vista na Figura 3.4. Neste caso o usuário definirá duas listas de variáveis, uma com as entradas do elemento e outra com as suas saídas. Tendo isso definido o usuário inserirá a tabela-verdade correspondente ao conjunto de entradas e saídas e sua função de transferência. Durante a interpretação o Yasc irá associar a tabela ao elemento que está sendo definido.

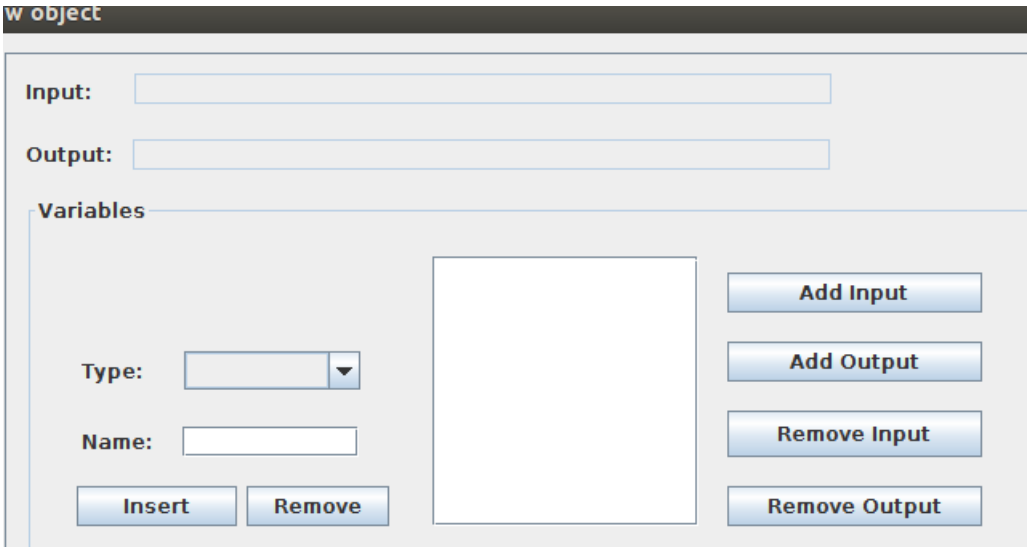


Figura 3.4: Interface de definição tabela-verdade.

3.2.3 Interpretador de parâmetros

Após o usuário finalizar a criação de seus objtos, os parâmetros passadas para cada um como sua fómula, ou tabela verdade, serão interpretados para a escrita do arquivo que será a biblioteca de simulação. Nela estará contida todas as caracteristicas dos objetos

criados, como por exemplo, a fórmula ou tabela que o representa e suas variáveis. A partir dos parâmetros descritos pelo usuário é criado o arquivo que representa todas as informações passadas de forma simbólica na biblioteca de simulação.

```
3 y Id_0 - (a+b); 2
b boolean a
b boolean b
6 s
```

Figura 3.5: Representação do objeto criado com fórmula como função de transferência.

Na imagem 3.5 é mostrado o resultado da descrição feita pelo usuário em um objeto que é definido por uma fórmula. Os itens descritos na imagem do arquivo são:

- **Identificados do tipo de objeto:** O número da primeira linha representa o tipo do objeto, sendo identificado por ”3”os objetos criados por transição de objetos;
- **Idenficados de operação lógica e Identificador do objeto:** É representada pelas letras ”y”ou ”n”e registra se o objeto realiza operação lógica ou não. Logo após este identificador encontra-se o nome do objeto;
- **Função de transferência:** Mostra a função de transferência escolhida para o objeto, seguido pelo número de variáveis que o objeto possui. Na imagem é representada por ”(a + b); 2”;
- **Descrição das variáveis:** Representada pelas linhas com a letra ”b”, marcam os tipos e os nomes das variáveis;
- **Parâmetro relevante:** É representado pelo número ”6”no inicio da linha e seguido pela variável quie vai representa-lo.

```
3 Id_1 - (ab); (c); 3
b boolean a
b boolean b
b boolean c
t a b c
t 1 1 1
t 1 0 0
t 0 1 0
6 sa
```

Figura 3.6: Representação do objeto criado com tabela verdade como função de transferência.

Na imagem 3.6 é mostrado o resultado da descrição feita pelo usuário em um objeto que é definido por uma tabela verdade. Os itens descritos na imagem do arquivo são:

- **Identificados do tipo de objeto:** O número da primeira linha representa o tipo do objeto, sendo identificado por ”3”os objetos criados por tabela verdade;
- **Identificador do objeto:** Nome do objeto;
- **Função de transferência:** Neste tipo de objeto é dividido em função de entrada (input) e saída (output), seguido do número total de variáveis. Na imagem os parâmetros função de entrada, saida e número de variáveis são representados por ”(ab); (c); 3”;
- **Descrição das variáveis:** Representada pelas linhas com a letra ”b”, marcam os tipos e os nomes das variáveis;
- **Descrição da tabela verdade:** Representado com a letra ”t”no inicio da linha, marca a descrição feita da tabela verdade contendo suas variáveis de entrada e saida, assim como, seus possíveis valores;
- **Parâmetro relevante:** É representado pelo número ”6”no inicio da linha e seguido pela variável que vai representa-lo.

A interpretação dos objetos na simulação ocorre inicialmente dentro do centro de serviços que ordena a execução da simulação, sendo tanto as bibliotecas definidas por fórmulas como as por tabela-verdade executadas pelo centro de serviço instantaneo. O que diferencia cada tipo de execução é o tipo de análise que é feita com os parâmetros passados para os interpretadores. Ambos os interpretadores possuem uma parte em comum, esta responsável por definir qual tratamento será dado ao objeto analisado, como por exemplo se é um objeto de origem, destino ou passagem, também organizando seus dados para a obtenção correta de valores para a passagem de resultados de um objeto para o outro.

A chamada do analisador do objeto ocorre em seu atendimento, em que são passados os parâmetros necessários para retornar um resultado. Assim como ocorre com o centro de serviço, cada tipo de objeto também possui seu analisador.

O analisador para objetos descritos com fórmula como função de transferencia recebe como parâmetro o nome do objeto, a sua tabela de variáveis e a tabela dos valores correspondentes as variáveis. A partir destes dados é feita a leitura da fórmula e realizada a separação em parte referênte ao valor das variáveis e os sinais que a fómula

possui. Após a separação são obtidos dois vetores que através da comparação de seus elementos é obtido o valor que será retornado a outro elemento como entrada ou como resposta da simulação.

O outro analisador, para objetos descritos com tabelas verdade como função de transferência, recebe como parâmetro o nome do objeto, tabela de variáveis e de valores, as partes com os valores de entrada e saída da tabela referente ao objeto. Assim como na analise descrita no parágrafo anterior é formado um vetor com os valores passados para as variáveis, sendo realizado a comparação entre este vetor e as linhas da parte de entradas da tabela. Localizando a linha correspondente, é possível localizar na parte de valores de saída o valor correspondente atravez do número da linha e obtendo o valor que deve ser retornado.

3.3 Sincronização de disparos

Dentre as partes implementadas no trabalho uma das mais importantes e a sincronização entre a execução dos objetos. Tal importancia se deve ao fato de que o resultado da análise de um objeto pode ser influenciado pelos disparos anteriores, assim, caso os disparos não ocorressem em uma determinada ordem o resultado final da simulação pode ser incorreto.

Para a implementação da sincronização foi considerado a forma com que o motor de simulação do YASC executata suas ações. Após a modelagem feita pelo usuário o motor procura por uma origem e um destino e estabelece uma sequencia de eventos com inicio e término nos pontos definidos, também estabelecendo as relações entre os objetos atraves do link que pode ser colocado para indicar o caminho. Assim, por padrão existia uma sequencia de eventos pré determinada para ser executada, porém não necessariamente respeitava a ordem de execução necessária para o tipo de simulação que este trabalho necessitava.

3.3.1 Implementação da sincronização

Para a implementação da sincronização de disparo foram armazenados a sequencia de execução dos objetos. Também foram armazenadas outras informações como se é um objeto de origem, destino ou apenas de passagem, que é aquele que não é origem nem destino. Outro dado importante foi obtido das conexos de saída e entrada, em que foi possível construir uma tabela que estabeleça uma relação de caminho de tal forma que seja possível identificar para qual objeto o resultado deve ser encaminhado e qual

variável o resultado do seu antecestral irá substituir.

A partir destes dados armazenados foi possível estabelecer uma relação com quais tarefas deveriam ser executadas primeiro e permitindo a organização da fila de eventos a serem executados. Tal organização prioriza colocar no início da fila os objetos identificados como sendo origens para então utilizar as relações estabelecidas com os dados obtidos das conexões de entrada e saída para ordenar as execuções dos objetos que seriam de passagem. Sendo adicionados por último os objetos identificados como destino e esta fila passada para a execução no motor de simulação.

Desta maneira, é garantido que os disparos ocorram em uma ordem correta sem o problema de ser executado sem que seus ancestrais já tenham sido resolvidos. Assim, a propagação do resultado ocorre de forma correta entre os objetos e a sincronização das execução é garantida.

3.4 Considerações finais

Neste capítulo foi descrita a arquitetura do Yasc. Por meio dela, pode-se observar que o gerador de simuladores é dividido em módulos, dentre eles estão a interface gráfica para criação e descrição comportamental do simulador criado, o gerador e interpretador de bibliotecas de configuração, a interface icônica e o motor de simulação.

Também foi apresentado como ocorreu o desenvolvimento de cada uma das adpatações feitas no motor de simulação para que fosse possível realizar a simulação de maneira genérica em todos os simuladores criados a partir do Yasc.

Capítulo 4

Testes e validações

4.1 Introdução

Neste capítulo são descritos os testes realizados sobre o Yasc. Foram executados dois testes, cada um com objetivos distintos. O primeiro consistem em analisar a eficácia da ferramenta atraves do uso feito por usuários, assim, considerando as críticas e sugestões para melhoria, como será mostrado na seção 4.2. No segundo, buscou-se comparar um simulador gerado pelo Yasc com o de outro simulador, como será mostrado na seção 4.3. Por fim, na seção 4.4 serão feitas algumas considerações finais sobre os testes.

4.2 Testes de usabilidade e o estudo de caso na área de circuitos sequenciais

Para verificar a eficácia oferecida pelo Yasc, subteu-se a ferramenta a testes com usuários e comparativos com outra ferramenta da área, com estudo de caso na área de circuitos sequenciais.

4.2.1 Procedimento para a realização dos testes

O teste realizado para a validação do Yasc foi dividido em etapas. Na primeira etapa, foi analisado qual seria o espaço amostral de avaliadores e definido qual o perfil dos usuários que participariam dele. Na segunda etapa, foi elaborado um plano de testes, do qual foi definido que o avaliador deveria fazer uso da ferramenta de maneira a criar um simulador de circuitos sequenciais e responder a um questionário com perguntas sobre o desempenho do software que estava sendo testado. Este procedimento permite

que possa ser observado o funcionamento da ferramenta para a criação de uma nova ferramenta de simulação. Após o uso da ferramenta, na terceira etapa o usuário respondeu a um questionário elaborado na etapa anterior e disponibilizou o simulador gerado para testes futuros. Na quarta etapa, os dados obtidos atraves dos questionários foram avaliados e realizados testes comparativos. As etapas serão detalhadas a seguir.

Definição do espaço amostral e perfil dos avaliadores

Para os testes foi adotado um espaço amostral de 39 usuários com o perfil de já possuirem conhecimento na área de circuitos sequenciais combinacionais. Sendo passoda a eles a pesquisa de um circuito conversors paralelo/serial e serial/paralelo para ser modelado no Yasc.

Elaboração do plano de testes e questionário

Após a definição da quantidade de avaliadores, foi elaborado o plano de testes e também o questionário que deveria ser respondido ao final do procedimento.

Com o plano de testes, cuja descrição completa encontra-se no Apêndice A, teve início o estudo de caso, utilizando o Yasc para a geração de um simulador de circuitos sequenciais e a utilização para a construção de um modelo.

Por último, cada usuário teve que responder um questionário, que encontra-se no Apêndice B, no qual relata sua experiência e em sequência entregar o simulador criado.

Execução dos testes

Após o termino da elaboração do plano de testes, foi dado início a sua execução. Os avaliadores foram submetidos aos mesmos processos. Eles realizaram todas as etapas descritas, sendo elas responder o questionário e entregar o simulador junto com o modelo nele criado para os procedimentos seguintes, descritos na seção 4.3.

4.2.2 Analise dos testes de usabilidade

Após realizadas as avaliações programadas para o teste de usabilidade, os resultados obtidos foram analisados e serão apresentados nesta seção.

O questionário respondido por cada usuário foi composto por duas questões discursivas e por sete afirmações, nas quais cada resposta representava um grau de concordância com o que era afirmado, podendo este grau ser:

1. Discordo totalmente
2. Discordo em boa parte
3. Nem concordo nem discordo

4. Concordo em boa parte

5. Concordo integralmente

A partir das afirmações foram gerados gráficos para análise da avaliação, como pode ser visto nas Figuras 4.1, 4.2,4.3, 4.4, 4.5, 4.6 e 4.7. Em todas elas pode-se observar qual foi a afirmação feita para o usuário e as quantidades apresentadas para cada uma das cinco possíveis concordancias.

Na figura 4.1, em que a afirmativa feita feito foi "O Yasc é fácil de usar.", pode-se concluir que, obteve uma boa qualificação com relação a sua facilidade de uso, porém com dificuldade de uso de alguns usuários. Isto foi confirmado pelos 69,23% dos validadores que concordam em boa parte ou integralmente com a informação.

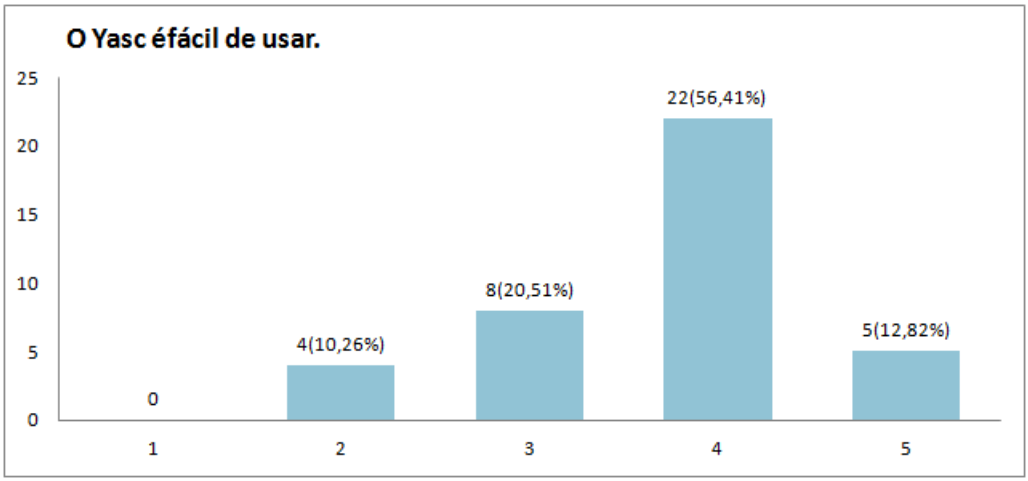


Figura 4.1: Resultado obtido para a primeira afirmativa do questionário

Na Figura 4.2, cuja afirmação era: "As informações na interface do Yasc estão bem organizadas.",71,8% dos usuários responderam positivamente, concordando totalmente ou parcialmente com isso.

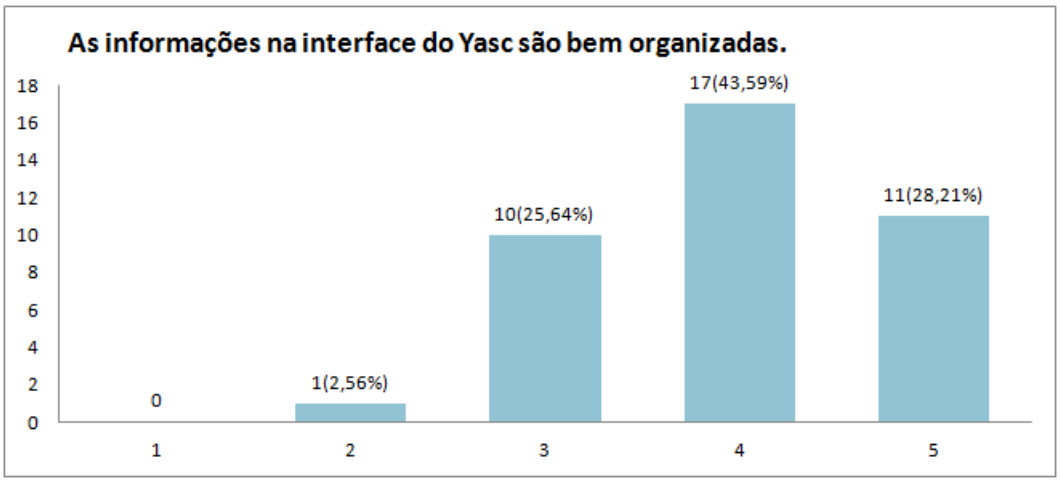


Figura 4.2: Resultado obtido para a segunda afirmativa do questionário

Na Figura 4.3, em que a afirmação feito foi: "A aparência das telas do Yasc é bastante clara.", 82,06% dos avaliadores confirmaram o que foi perguntado, concordando em boa parte ou integralmente com a afirmação.

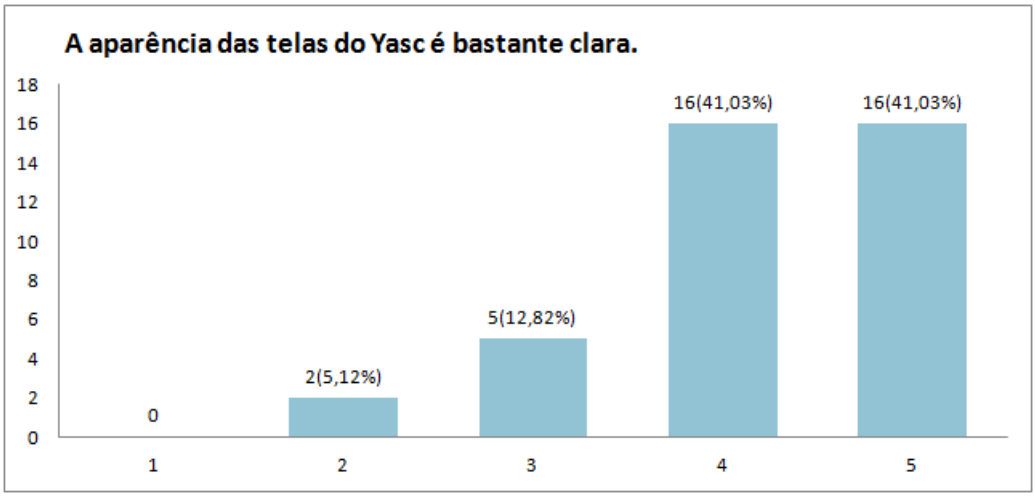


Figura 4.3: Resultado obtido para a terceira afirmativa do questionário

Na Figura 4.4, a afirmação foi: "A nomenclatura utilizada nas telas é fácil de compreender.", 74,36% dos usuários concordaram integralmente ou em boa parte com isso.

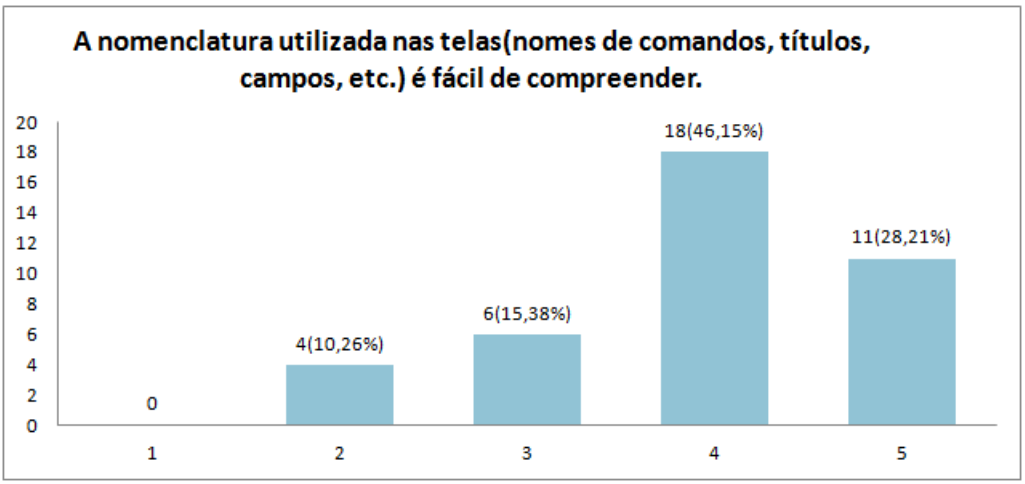


Figura 4.4: Resultado obtido para a quarta afirmativa do questionário

Encerrando as afirmações relacionadas com as interfaces gráficas do Yasc, na Figura 4.5, foi afirmado: "As mensagens do sistema são claras.". Questão que recebeu 69,33% das avaliações sendo positivas.

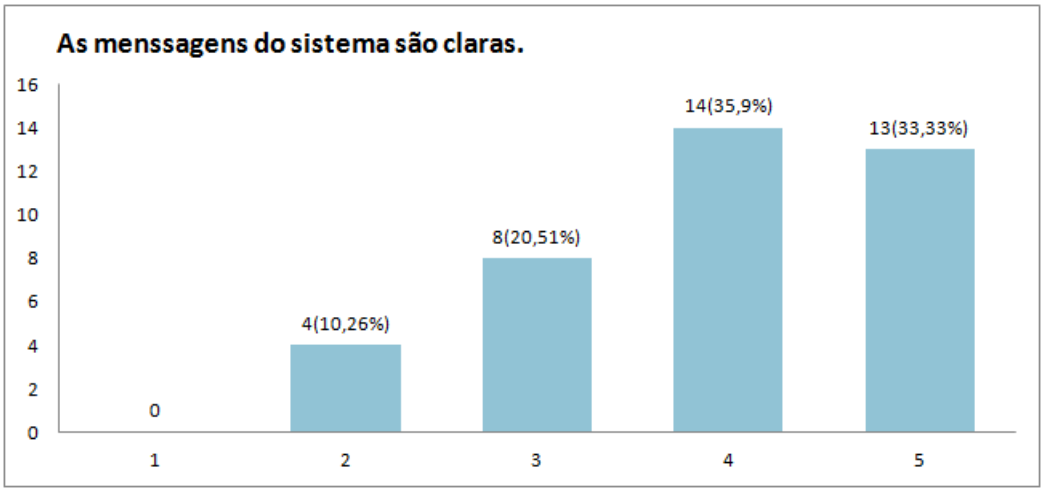


Figura 4.5: Resultado obtido para a quinta afirmativa do questionário

Desta forma, com as respostas das questões das Figuras 4.2, 4.3, 4.4 e 4.5, pode-se concluir que quanto a organização, clareza, nomenclatura e aparência, o Yasc agradou os usuários e foi eficiente.

Por fim, mais duas questões objetivas foram feitas. Na primeira, que aparece em 4.6, o enunciado foi: "No geral, a utilização do Yasc foi interessante". Dentre as respostas, 76,93% dos usuários confirmaram que o uso da ferramenta foi interessante e confirmaram que concordam completamete ou parcialmente com a afirmação colocada.

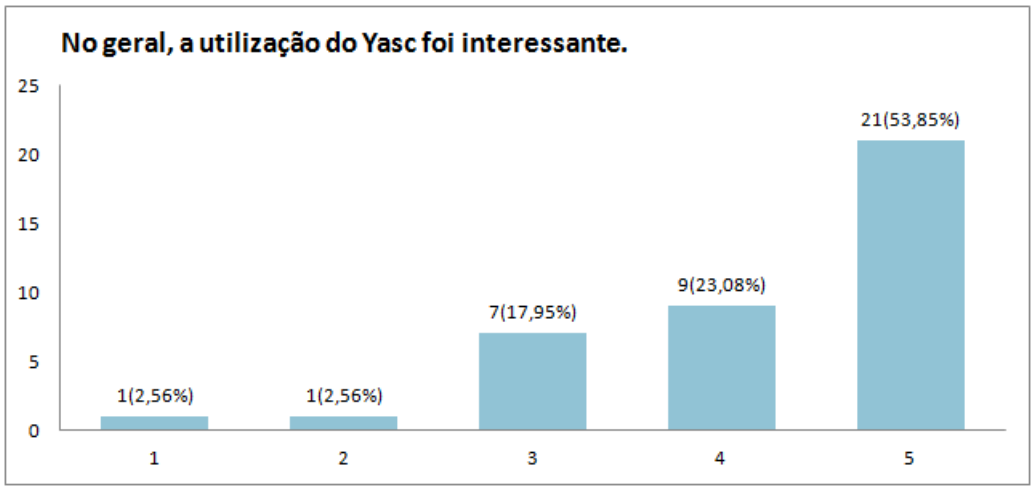


Figura 4.6: Resultado obtido para a sexta afirmativa do questionário

Na segunda, que aparece na Figura 4.7, foi perguntado: "O Yasc é uma ferramenta interessante para geração de simuladores", e 92,32% dos validadores concordaram em boa parte ou de forma integral com o Yasc sendo uma ferramenta interessante para a geração de simuladores.

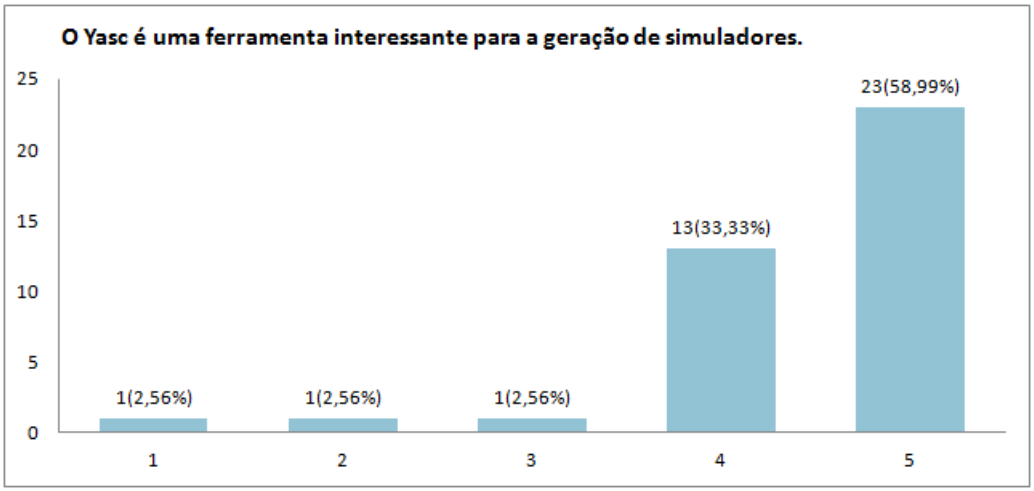


Figura 4.7: Resultado obtido para a sétima afirmativa do questionário

Com relação às questões discursivas, a primeira, "Aponte os pontos positivos que você encontrou ao utilizar o sistema", teve o objetivo de obter dos usuários quais pontos positivos foram encontrados durante a utilização do Yasc. Assim, dentre as respostas obtidas se destacam:

- A iteratividade e o ambiente agradável do Yasc.

- Facilidade da construção dos objetos.
- Possibilidade de criar uma simulação personalizada.
- O fato do Yasc ser multiplataforma

Enquanto isso, na segunda, que questionava "Aponte os pontos negativos que você encontrou ao utilizar o sistema", o propósito era questionar os mesmos usuários sobre os pontos negativos encontrados. Dentre as respostas estavam:

- Impossibilitar do usuário alterar o tamanho do icone que representa o objeto.
- A impossibilidade de adicionar novos componetes a biblioteca após ela estar criada.
- Falta de templates com exemplos.

4.3 Validação e resultados produzidos pelos simuladores gerados

A partir dos testes de usabilidade obteve-se 39 simuladores de circuitos. Classificou-se os simuladores gerados em corretos, parcialmente corretos e incorretos. Assim, segundo a classificação obteve-se 23 simuladores contruídos da maneira correta, 13 parcialmente correto e 3 incorretos.

Para o teste comparativo foi escolhido o simulador Qucs(Quite universal circuit simulator).A escolha do Qucs para comparação com o Yasc se deu devido ao fato de que este simulador aceita descrição de elementos em verilog, e também outras linguagens como vhdl, e também permite realizar a modelagem do simulador utilizando a biblioteca já pronta do software.

4.3.1 Tetses comparativos

Os parâmetros comparados entre os dois simuladores foram: personalização de objetos, ausência de codificação, automação e modelagem da simulação. A comparação entre estes parâmetros nos dois simuladores encontra-se na figura 4.8 seguido dos detalhamentos de cada tópico abordado.

	Yasc	Qucs
Personalização de objeto	+	+
Ausência de codificação	+	-
Automação	-	+
Modelagem	+	+

Figura 4.8: Tabela Comparativa entre Yasc e Qucs.

Personalização de objetos e ausência de codificação

No quesito personalização de objetos, ambos os simuladores apresentaram tal caracteristica. Assim, ambos permitem que o usuário crie bibliotecas de simulação para serem usadas. Porém a personalização dos objetos que farão parte da biblioteca são feitas de maneiras diferentes. Enquanto a criação dos objetos no Yasc é feita por meio de interfaces gráficas, permitindo inclusive a criação do objeto com uma função de transferencia baseada em tabela-verdade, a do Qucs apresenta a possibilidade de criação com a utilização de liguagens, como vhdl e Verilog.

Assim, apesar de ambos apresentarem a possibilidade de personalização, o Yasc apresenta a caracteristica da ausência de codificação enquanto o Qucs não. Tal caracteristica permite que o usuário crie seu simulador sem a necessidade de aprender, ou saber, uma linguagem de programação.

Automação

Em relação ao quesito automação foi levado em conta a possibilidade de utilização de botões ou mecanismos que alterem valores com apenas um clique, sem a necessidade de demais trabalhos manuais do usuário. Neste quesito, o simulador Qucs mostrou-se melhor em relação ao Yasc.

Isso ocorre devido ao fato de que o Qucs é um simulador específico para a simulação de componentes eletrônicos, assim dedicando-se a abordar situações especificas deste tipo de simulação. Enquanto que o Yasc apresenta em sua proposta ser um gerador de simulador para diferentes contextos. Assim, ao mesmo tempo que deve permitir que o usuário construa um simulador de circuitos sequencias também deve manter suas caracteriscas de poder gerar outros tipos de simuladores, como por exemplo dos simuladores baseados em filas já implementados.

Modelagem da simulação

No de vista ponto de modelagem ambos os simuladores apresentam interfaces claras, cada um considerando suas diferenças. Essa diferença encontra-se principalmente

na quantidade de elementos presentes na tela. Enquanto que o simulador Qucs apresenta todas suas possibilidades de bibliotecas prontas, o Yasc aprsenta somente uma biblioteca de cada vez na interface, por mais que o usuário tenha mais de uma.

Essa diferença pode proporcionar diferenças na curva de aprendizagem do usuário devido a quantidade de informações que o Qusc apresenta em sua interface de modelagem. Estando esta questão ligada diretamente ao quanto o usuário está acostumado ao simulador.

4.4 Considerações finais

Neste capítulo foram apresentados os resultados obtidos a partir da análise feita da ferramenta implementada, o Yasc.

Por meio dos testes de usabilidade realizados e da avaliação feita pelo usuário, percebeu-se facilidade no uso da ferramenta. Obteve-se como conclusão que o Yasc é uma ferramenta de fácil uso, devido a ausência de codificação e também mostrou-se que é interessante para a geração de simuladores.

Ao longo dos testes o Yasc foi executado tanto nos sistemas operacionais Linux e Windowns, executando em ambos sem problemas. Assim, consolidando o Yasc como uma ferramenta multiplataforma.

Capítulo 5

Conclusão

Neste capítulo apresenta-se a conclusão sobre o trabalho e os próximos passos para a adição de novas funcionalidades na ferramenta.

Este trabalho teve como intuito o desenvolvimento de uma ferramenta para a geração de simuladores, de maneira simples e eficaz, com base em diferentes contextos. Porém com estudo de caso na simulação de circuitos sequenciais.

Para isto, foram feitos estudos relacionados tanto com simuladores de outras áreas como com simuladores da área de enfoque do trabalho. O estudo foi importante para entender o funcionamento de como a simulação é construida e obter um resultado satisfatório e eficaz para o software. Assim, buscando uma maneira de oferecer diretamente ao usuário um gerador de simuladores, ferramenta que possibilita a avaliação de sistemas.

Durante o desenvolvimento, as maiores dificuldades encontradas foram a caracterização dos objetos em relação ao centro de serviço que seria executado e a adapatação do motor de simulação para o caso de estudo que foi implementado. Ambas as dificuldades estão relacionadas com a estrutura do motor de simulação, em que para realizar a adição de novos módulos é necessário várias alterações em outros módulos que irão se relacionar com o que está sendo implementado.

A principal contribuição do Yasc é de fornecer a usuários leigos que não conheçam linguagem de programação, ou possuem dificuldades para trabalhar com elas, por não serem da área de computação, a possibilidade de desenvolverem simuladores e avaliar o desempenho deles através da ferramenta sem a necessidade de codificação.

5.1 Trabalhos futuros

Como trabalhos futuros destacam-se melhorias que podem ser feitas no Yasc como as que são mostradas a seguir:

- Reimplementação do motor para torna-lo mais modular. Assim, tornando-o mais flexível e permitindo que novos contextos sejam implementados com maior facilidade.
- Implementar alterações que permitam que o usuário incrmente a biblioteca após esta já estar criada.
- Trabalhar com a meneira que os objetos se conectam afim de permitir uma passagem de valores entre os objetos mais clara e fácil.

Referências Bibliográficas

Banks, J., Carson, J., Nelson, B., and Nicol, D. (2010). *Discrete-event system simulation.* Prentice Hall, 5th ed. edition.

Choi, B. K. and Kang, D. (2013). *Modeling and Simulation of Discrete Event Systems.* Wiley Publishing, 1st edition.

circuit simulator, Q. (2015). Quite circuit simulator. Disponível em <http://soyrceforge.net/index.html>, acessado em 06 de maio de 2016.

Furlanetto, G. C. (2016). Geração de simuladores de filas para diferentes contextos. Master's thesis, Universidade Estadual Paulista "Júlio de Mesquita Filho", Brasil.

Guru, A. and Savory, P. (2004). A template-based conceptual modeling infrastructure for simulation of physical security systems. In *Simulation Conference, 2004. Proceedings of the 2004 Winter*, volume 1, page 873.

KanbanSIM (2016). Kanbansim). Disponível em <http://focusedobjective.com/kanbansim_scrumsim/>, acessado em 16 de junho de 2016.

Maker, C. (2015). Circuit maker documantation. Disponível em <http://documantation.circuitmaker.com/>, acessado em 06 de maio de 2016.

Menezes, D., Manacero, A., Lobato, R. S., da Silva, D. T., and Spolon, R. (2012). Scheduler simulation using iSPD, an iconic-based computer grid simulator. In *Computers and Communications (ISCC), 2012 IEEE Symposium on*, pages 000637–000642.

PDSIM (2016). Pdsim. Disponível em <http://pdsim.sourceforge.net/index.html>, acessado em 16 de junho de 2016.

Pooch, U. W. and Wall, J. A. (1993). *Discrete Event Simulation: A Practical Approach.* CRC Press, Inc., Boca Raton, FL, USA.

PowerTrainSIM (2016). Powertrainsim). Disponível em <http://www.claytex.com/products/dymola/model-libraries/powertrain-dynamics-library/>, acessado em 16 de junho de 2016.

SPICE (2016). Spice. Disponível em <http://www.seas.upenn.edu/ jan/spice/spice.overview.html>, acessado em 16 de junho de 2016.

TINA (2016). Tina. Disponível em <http://pt.tina.com/digital_verilog_simulation>, acessado em 16 de junho de 2016.

Apêndice A

Questionário de validação

O objetivo deste questionário é coletar informações sobre a opinião do usuário a respeito do protótipo do Yasc (Yes, a simulator' s compiler). As informações fornecidas serão de extrema importância para a validação e o aprimoramento do sistema. Por favor, leia com atenção as questões a seguir e em caso de dúvida, solicite esclarecimento ao avaliador.

Nas questões a seguir indique o grau de concordância com a a firmação feita, sendo:

1. Discordo totalmente
2. Discordo em boa parte
3. Nem concordo nem discordo
4. Concordo em boa parte
5. Concordo integralmente

a)	O Yasc é fácil de usar	1	2	3	4	5
b)	As informações na interface do Yasc estão bem organizadas	1	2	3	4	5
c)	A aparência das telas do Yasc é bastante clara	1	2	3	4	5
d)	A nomenclatura utilizadas nas telas (nome de comandos, títulos, campos, etc.) é fácil de compreender	1	2	3	4	5
e)	As mensagens do sistema são claras	1	2	3	4	5
f)	No geral, a utilização do Yasc foi interessante	1	2	3	4	5
g)	O Yasc é uma ferramenta interessante para a geração de simuladores	1	2	3	4	5

Aponte os pontos positivos que você encontrou ao utilizar o sistema:

Aponte os pontos negativos que você encontrou ao utilizar o sistema:

Apêndice B

Manual de uso

1. **Introdução**

   O Yasc é uma ferramenta para geração de simuladores. Nele o projetista, por meio de uma interface gráfica, fornece descrição dos elementos que farão parte do simulador que será gerado e também detalha como este irá se comporta.

   A partir da descrição a ferramenta cria um simulador com interface icônica. O objetivo de usar interface icônica é facilitar a tarefa de criar modelos dos sistemas a serem simulados, evitando a necessidade de codificação dos ambientes.

   Este texto tem como finalidade orientar o usuário durante a geração e a utilização de simuladores. Para isto, na sequencia serão apresentados como exemplo passo a passo que devem ser seguidos para a criação e utilização de um simulador de circuitos sequenciais.

2. **Como utilizar a interface gráfica para criar um simulador**

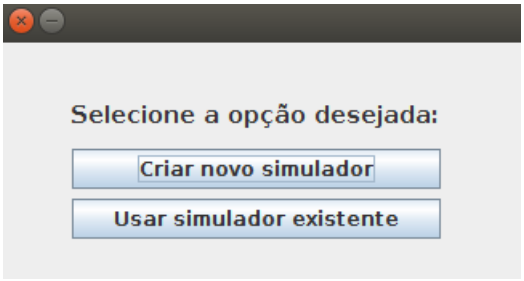
   Nesta seção do manual, será descrito como criar um simulador de circuitos sequenciais. Para isso os seguintes passos devem ser seguidos:

   1° execute a ferramenta

   Windows e Mac Os -> basta dar dois cliques em cima do ícone da ferramenta
   Linux -> abra um terminal e digite "java –jar Yasc.jar"

   Após realizado o primeiro passo, será solicitado ao usuário qual linguagem deseja e também aparecerá a tela de boas vindas ao simulador, bastando clicar "OK" em ambos para prosseguir.

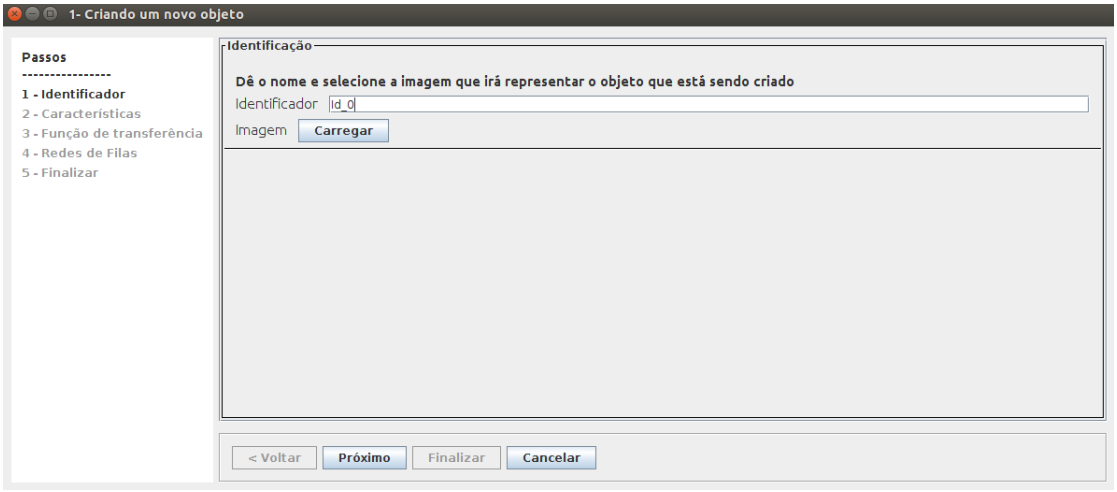
   2° Selecione a opção "Criar novo simulador"



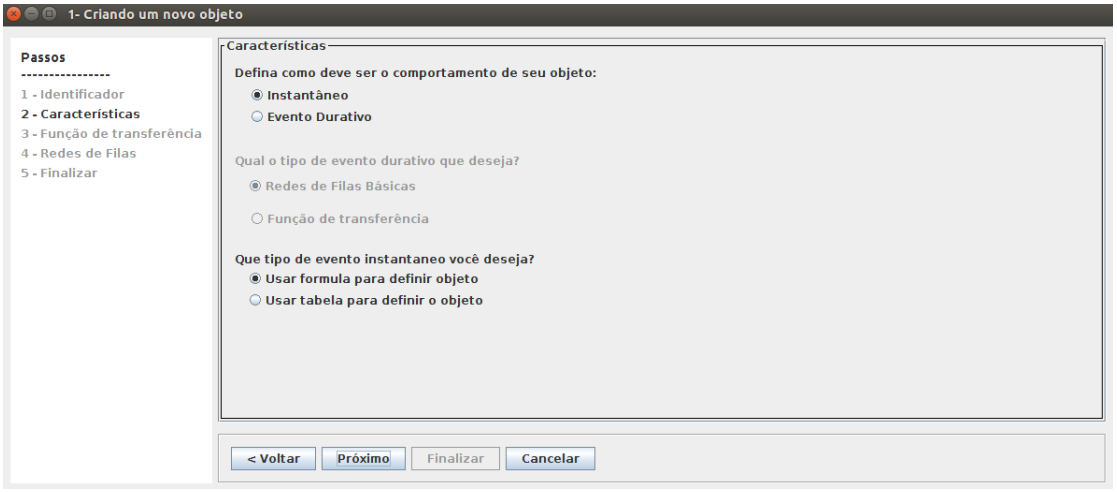
3° Crie os objetos

No segundo passo da etapa de criação, deverão ser projetados os objetos a serem disponibilizados no simulador criado. Neste caso, serão projetados objetos em etapas separadas para dois tipos de simulações diferentes. A primeira será de objetos cuja característica é de sua função de transição ser representada por uma fórmula, a segunda com a característica de sua função de transição ser uma tabela verdade.

I)Coloque o nome do objeto como o presente na tabela abaixo e clique em "Próximo".

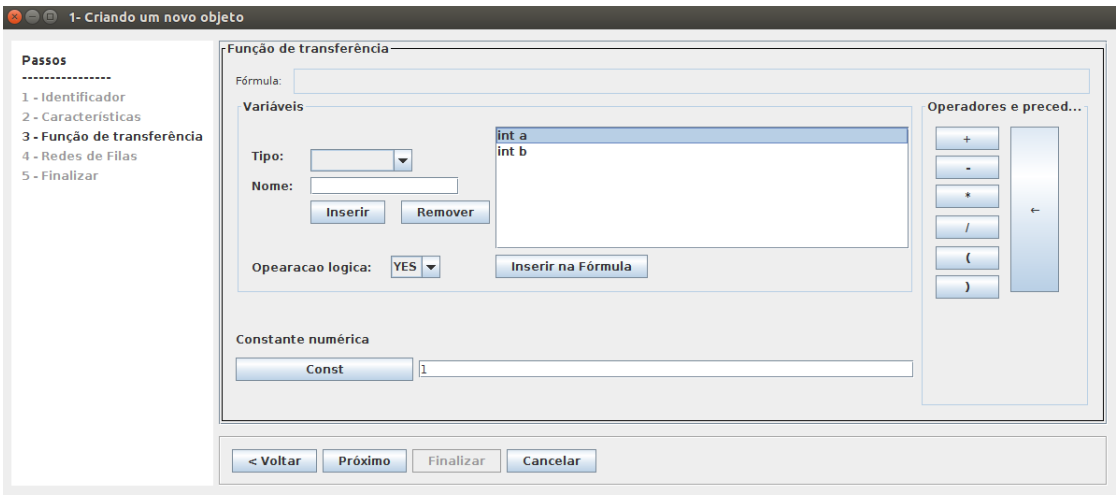


II) Selecione o tipo de comportamento que o objeto segue e clique em "Próximo".

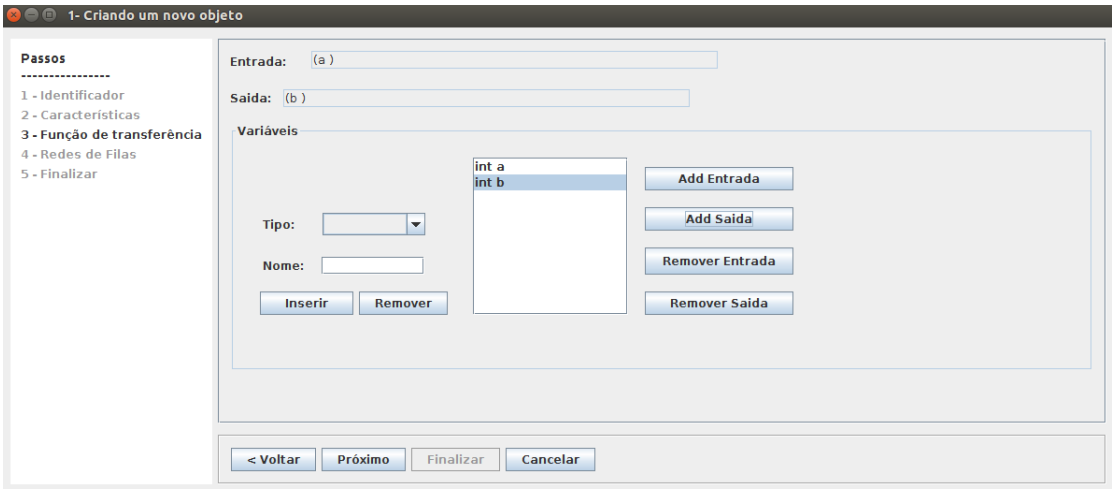


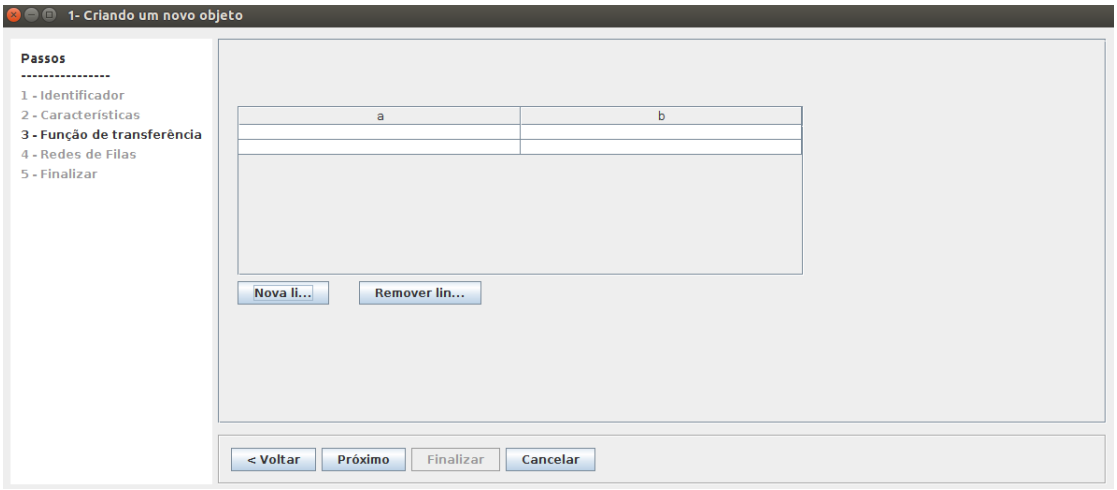
III) Crie a função de transição sem se esquecer de habilitar a função de operação lógica como ”SIM”. Na criação da função os operadores tem os seguintes significados:

- + : and
- - : or
- * : notAnd
- / : notOr



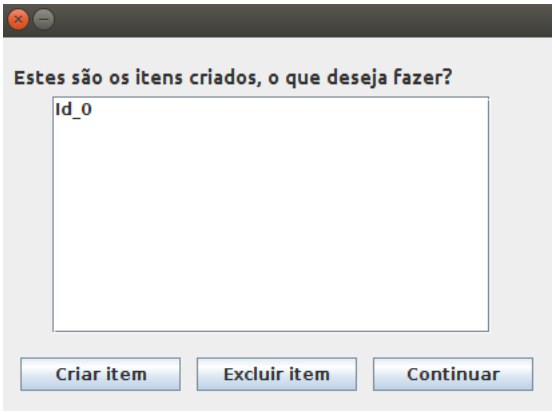
Caso o objeto que está sendo criado for definido por uma tabela definir qual variável será entrada e qual será saída e também qual a composição da tabela.



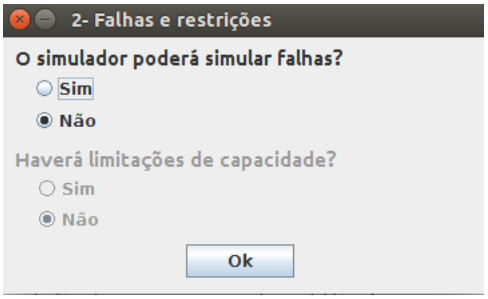


IV) Ao finalizar, aparecerá uma interface como mostrada abaixo, a qual é possível realizar o gerenciamento dos ícones criados. Clique em “Criar item” e inicie a criação para os itens desejados.

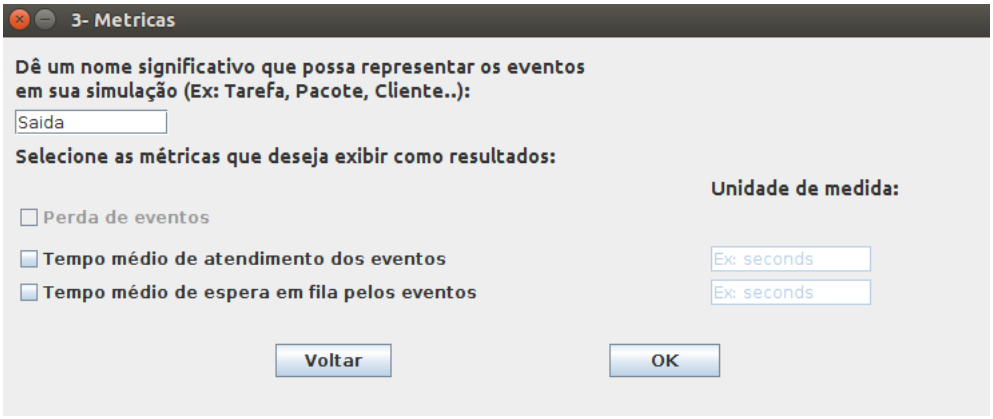
V) Após criar todos os objetos da lista, com suas respectivas funções de transição, clique em “Continuar”.



4° Selecione que não haverá falhas e clique em “Ok”.



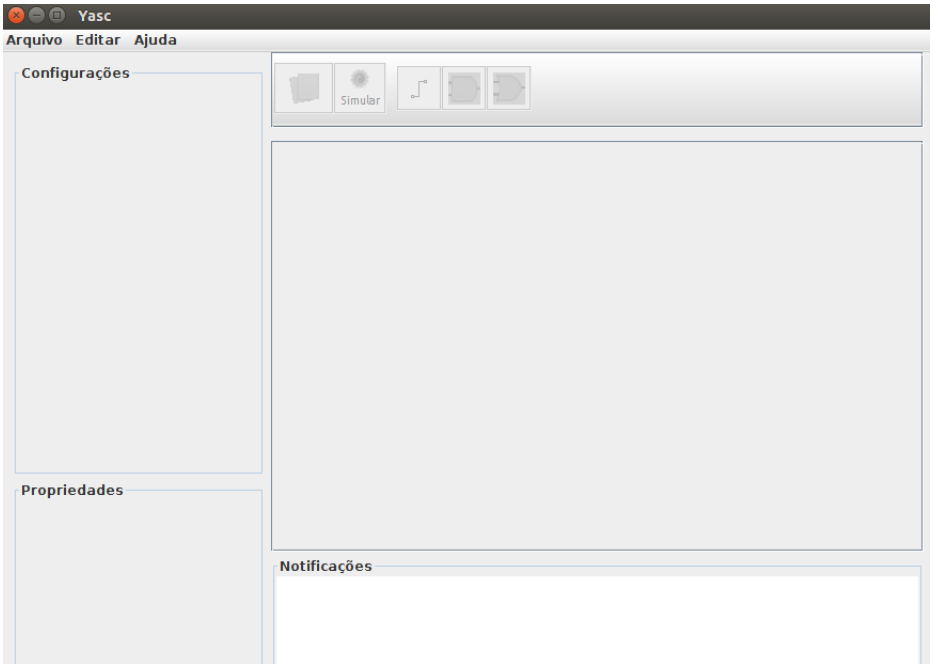
5° Forneça o nome dos eventos como “saída” e clique em “OK”.



6° Por fim, confirme que sua descrição de simulador está completa e este será carregado em sua tela.

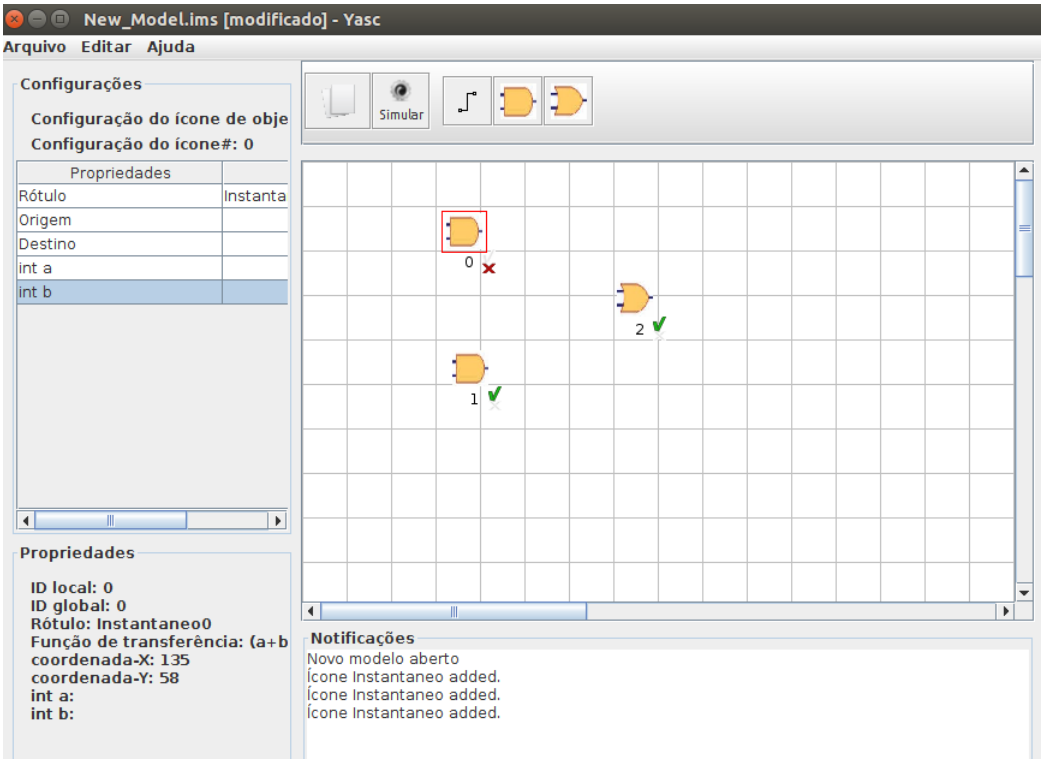
3. Como utilizar a interface icônica para criar um modelo

Com o simulador já pronto a seguinte tela será apresentada:

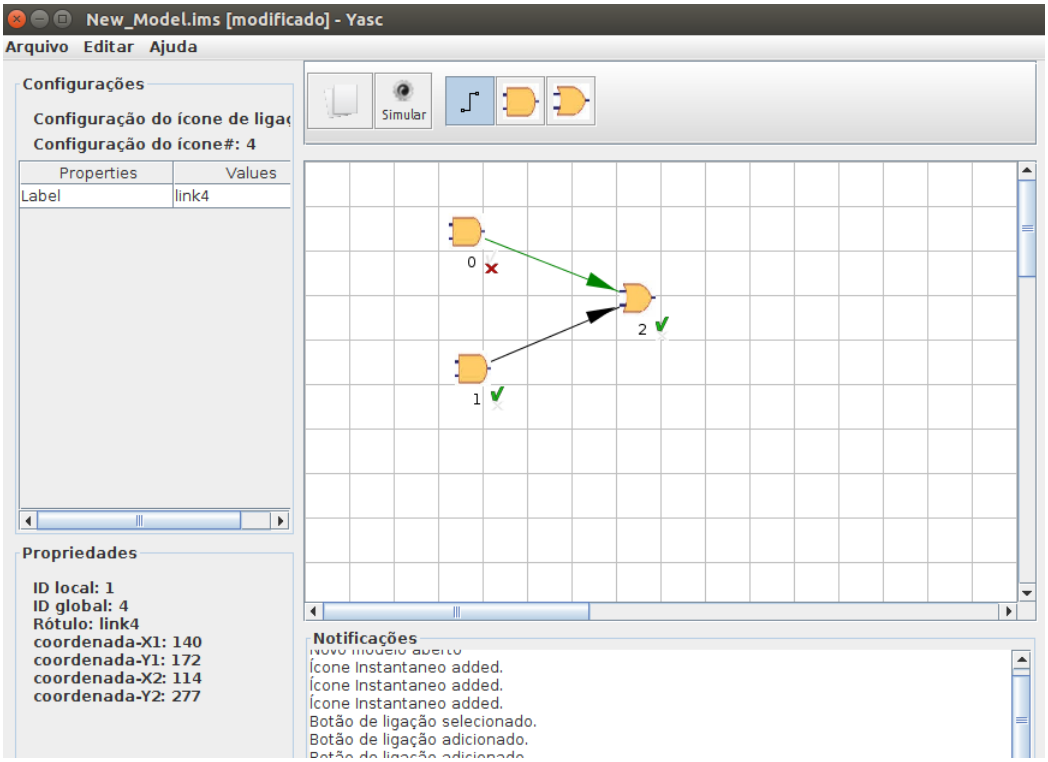


Ela representa a interface icônica do simulador criado, em que o usuário retratará o sistema real desejado por meio de um modelo, o qual será estimulado e retornará resultados relativos ao comportamento de tal sistema. Para uso da interface icônica os seguintes passos devem ser seguidos:

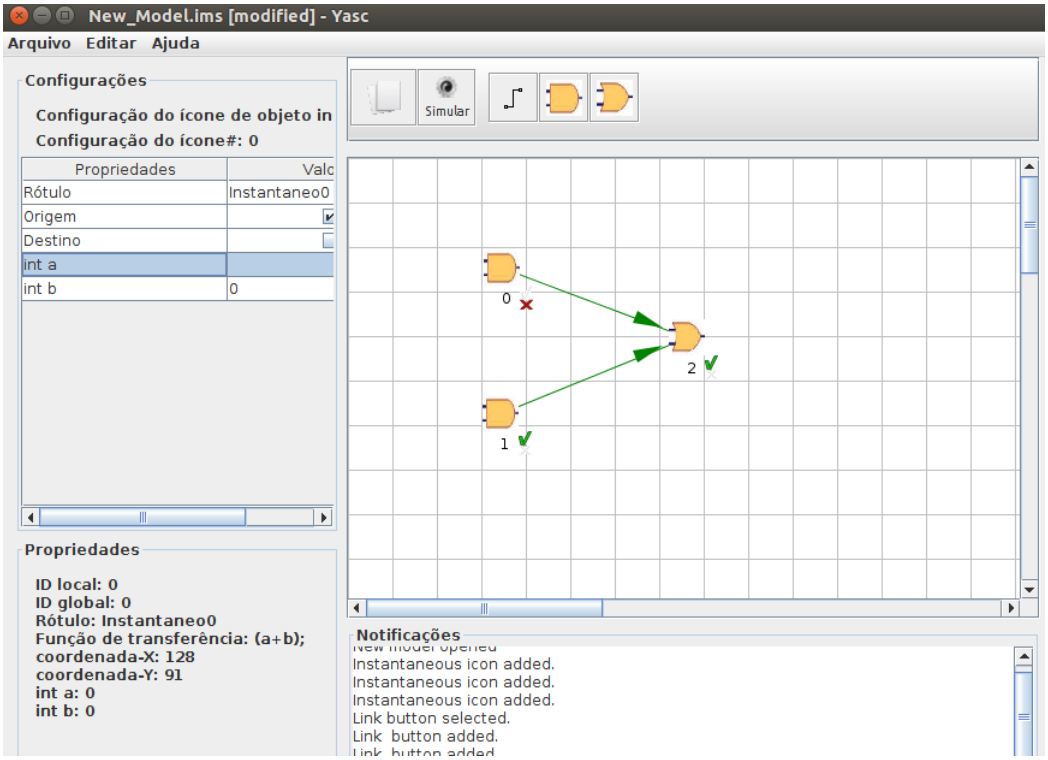
1° Com um modelo novo aberto selecione o ícone desejado dentre os que estão na parte superior, após a engrenagem, e os distribua na área quadriculada (Área de Desenho) da maneira desejada para melhor representar seu sistema



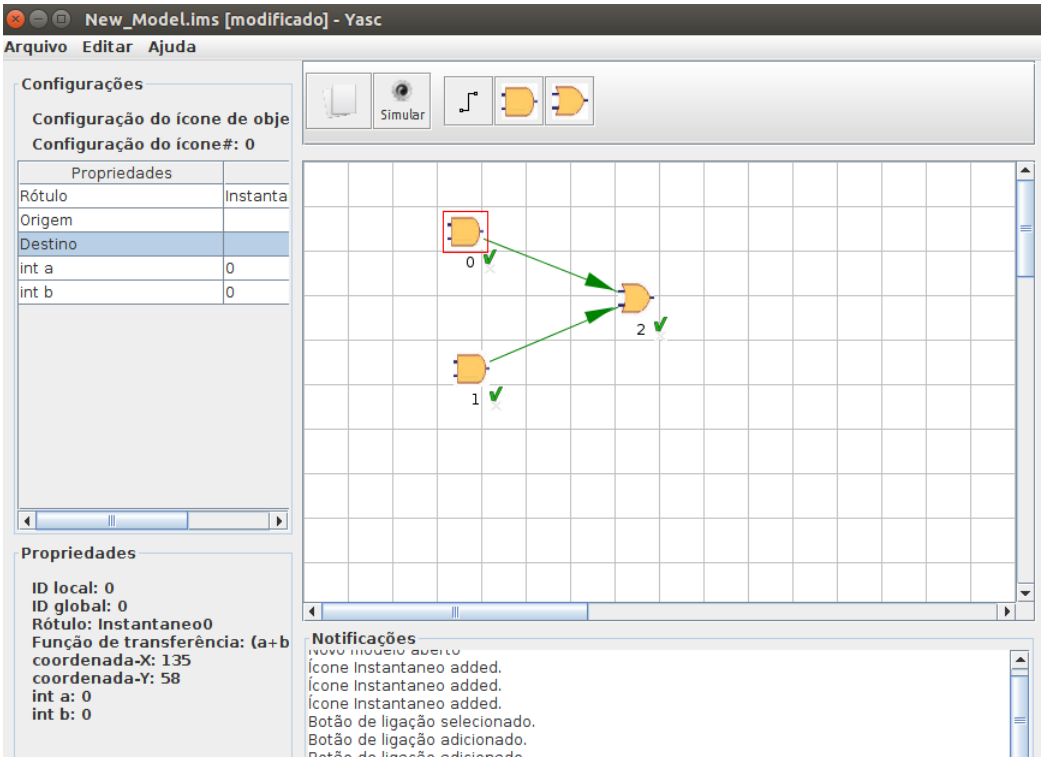
2° Ligue os ícones para indicar o sentido de propagação dos eventos, configurando o nome do ícone de link como uma das váriaveis do objeto que está recebendo o link



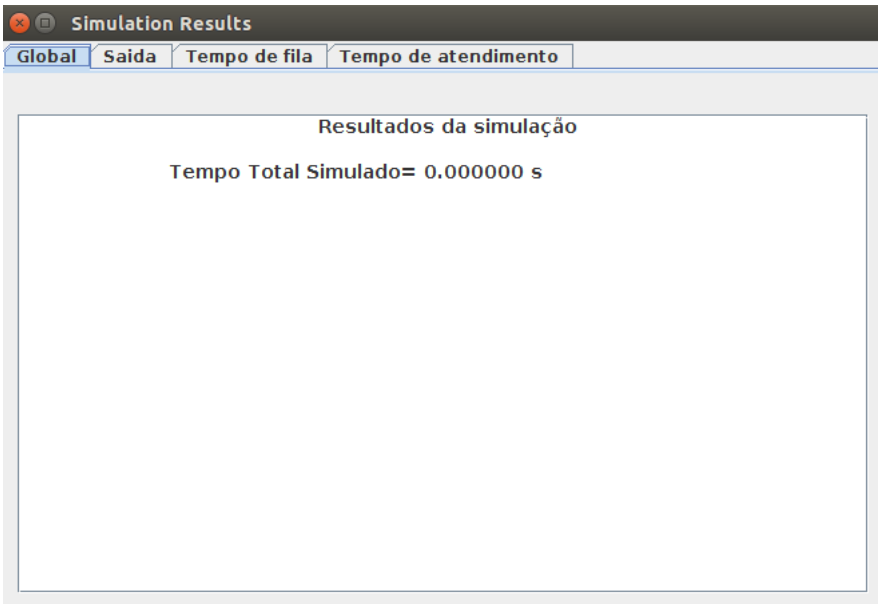
3° Para cada ícone adicionado, configure todos os seus parâmetros que aparecem ao lado esquerdo da Área de Desenho, assim, sua configuração será marcada como correta. Para ter certeza de que o ícone está configurado corretamente observe abaixo dele, caso haja um x, algum parâmetro está desconfigurado ou ausente, caso contrário, na presença de um pequeno "check", a configuração está correta.



4° Com todos os ícones configurados, selecione o botão de cargas e configure também os parâmetros destas no painel aberto



5° Por fim, clique em "Simular" e então ao término da simulação, caso não seja registrado nenhum erro no modelo fornecido, será apresentada uma tela com os resultados da simulação



4. Conclusão

O Yasc é um sistema que possui várias funcionalidades apesar de ainda estar em sua primeira versão. Por isto, torna-se uma tarefa muito complexa cobrir todos os seus casos de uso em um simples manual. Espera-se que com este texto explicativo, seja apresentado ao usuário uma introdução à ferramenta, sendo esta suficiente para a compreensão de como aplicar a ferramenta as mais diversas possibilidades, direcionando e facilitando o trabalho e a pesquisa.

Equipe de Desenvolvimento do GSPD (Grupo de Sistemas Paralelos e Distribuídos)