Simulador de Máquina de Estados Finitos

Bruno Del Monde (Nº 10284712)
Kaique da Cunha Lima (Nº 10368980)
Matheus Carvalho Raimundo (Nº 10369014)
Paulo Renato Campos Barbosa (Nº 9779475)

Orientador: Prof. Mauricio Acconcia Dias

USP - São Carlos

Dezembro/2017

Índice Analítico

1. Considerações Iniciais

Neste projeto do fim de semestre, foi proposto a construção de um Simulador de Máquina de Estados Finitos (Moore e Mealy). Tal simulador deve aceitar um arquivo que descreve a máquina (em um formato pré-determinado) ou permitir que o usuário construa através da própria interface. Durante a simulação, deve desenhar os estados e realizar as animações de transição entre eles. O objetivo do trabalho é inserir no âmbito do curso o hardware, de forma a utilizar conceitos (como a própria interface gráfica) que serão extremamente úteis na profissão de um computeiro.

Este grupo produziu uma interface gráfica para a web, utilizando de HTML 5, CSS 3 e Javascript. A interface funciona em qualquer navegador moderno que respeite os padrões da W3C. Os navegadores mais utilizados e conhecidos o fazem.

2. História da Web e Conhecimentos Gerais

Antes de 1957 computadores só executavam um processo por vez, o que era conhecido como “batch processing”. Devido ao crescimento em tamanho dos computadores, estes passaram a ser instalados em salas especiais refrigeradas e, já que o operador não podia mais ficar na mesma sala devido à ocorrência de *bugs*, fez-se necessário a criação de um tipo de acesso remoto a essas máquinas. Foi durante este período também que o conceito de “time-sharing” começou a ser desenvolvido, nesse tipo de processamento vários usuários dividem o tempo de processador de um mesmo computador. Foi essa união do acesso remoto com o *time-sharing* que possibilitou o surgimento dos primeiros servidores.

Quando a União Soviética lançou seu satélite, o Sputnik, surgiu nos Estados Unidos o medo do surgimento de mísseis intercontinentais; e esse medo fez com que a DARPA (Defense Advanced Research Project Agency) fosse criada. Para evitar vazamento de informações, a DARPA começou a planejar uma enorme rede de computadores interconectados que ficou conhecida como Arpanet. Uma das principais características do funcionamento da Arpanet era que um “computador-interface” (IMP) cuidava do acesso a um outro mais poderoso, conhecido como *Mainframe*, onde a inicialização dos processos se dava. Para assegurar a correta comunicação entre os computadores-interface, foram criados protocolos de comunicação: NCP, e depois TCP.

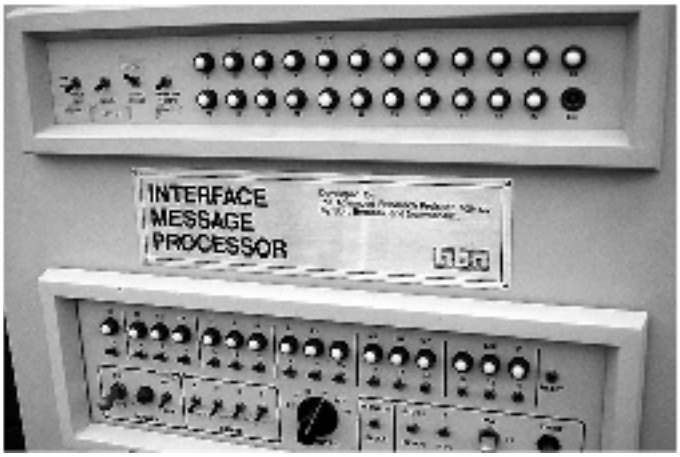

Figura 1: Exemplo de computador-interface (IMP).

A Arpanet não foi a única tentativa de se criar uma rede de computadores; outras já estavam funcionando durante sua concepção e algumas foram criadas depois. As principais são: NPL, Cyclades e RAND. Sendo suas contribuições: a divisão dos pacotes transmitidos em pedaços menores pela NPL, devido ao medo do congestionamento da rede; a descentralização dos servidores pela RAND, para evitar a queda por completo da rede devido a um ataque militar; a interconectividade entre pequenas redes pela Cyclades. Foi por causa dessa rede da Cyclades que o termo “inter-net” surgiu.

Foi graças a todas essas contribuições que a internet se tornou o que se conhece hoje. Em 1990 o hardware da Arpanet foi removido, mas graças a descentralização dos servidores e a todas as suas outras características, a rede continuou funcionando.

Contudo, a transferência de informações ainda não era algo trivial. É aí que entra os navegadores e o HTML.

Tim Berners-Lee foi o pioneiro do hipertexto. Ele criou o navegador WorldWideWeb em 1989, o primeiro navegador de internet e primeiro editor WYSIWYG (“What You See Is What You Get”) de HTML. Após o WorldWideWeb, outro navegador marcante foi o Netscape. O Netscape foi o primeiro lançado oficialmente para o Microsoft Windows, o que fez com que reinasse nos computadores pessoais. Contudo, a Microsoft lança o Internet Explorer, e em 1997 passa a incluir este dentro do Windows. Isso levou a decadência do Netscape, e como reação este teve seu código aberto a comunidade. Apesar disso, abriu-se portas ao surgimento de muitos outros projetos, como o Firefox, Opera e Chrome. Todos esses navegadores têm, porém, um ponto em comum. Todos fazem a conversão de uma determinada linguagem de marcação em texto e gráfico. Essa linguagem de marcação é o HTML.

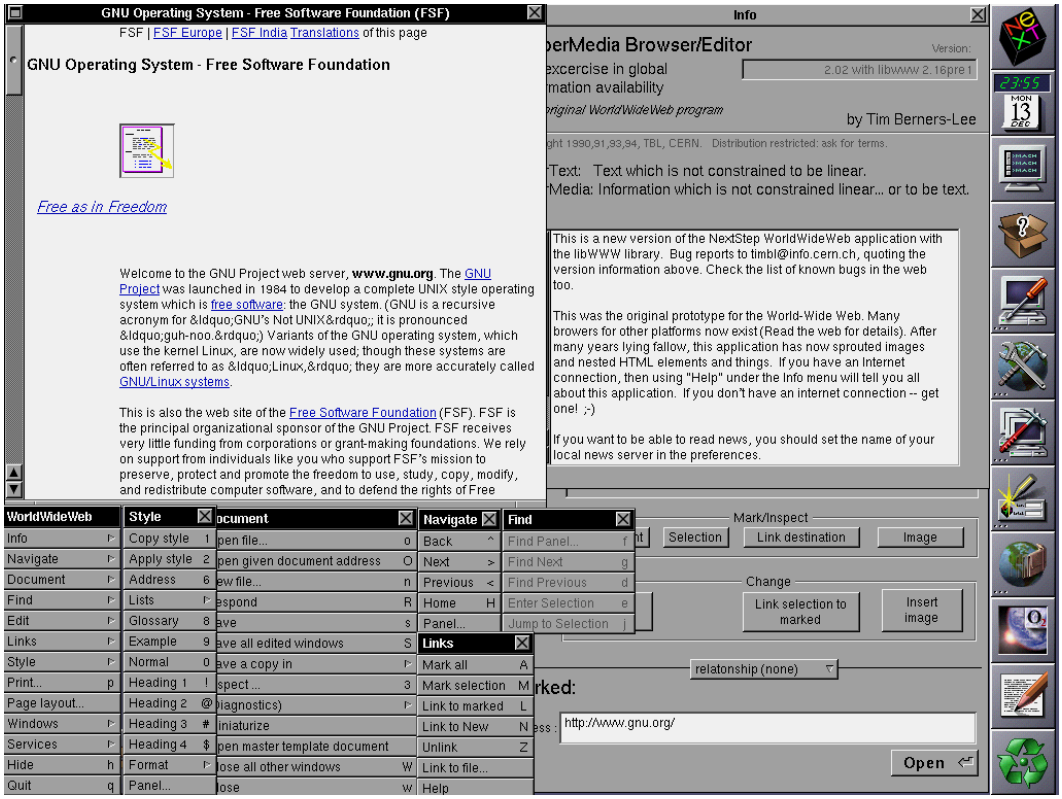

Figura 2: WorldWideWeb, primeiro navegador e editor WYSIWYG de HTML.

O Hypertext Markup Language (HTML) é uma linguagem de marcação padronizada para criar aplicativos e páginas Web. Com ele, é possível que os navegadores transformem código em texto, links, imagens, formulários, e qualquer outra forma de multimídia. O HTML trabalha em conjunto com o Cascading Style Sheets (CSS) e o JavaScript. O CSS é uma linguagem de estilos que define toda a aparência e estilo das páginas, enquanto o JavaScript é uma linguagem de programação que permite adicionar dinamicidade e interatividade ao conteúdo.

```
<!DOCTYPE html>
<html>
<body>

<h2>What Can JavaScript Do?</h2>

<p>JavaScript can change HTML attributes.</p>

<p>In this case JavaScript changes the src (source) attribute of an image.</p>

<button onclick="document.getElementById('myImage').src='pic_bulbon.gif'">Turn on the light</button>

<img id="myImage" src="pic_bulboff.gif" style="width:100px">

<button onclick="document.getElementById('myImage').src='pic_bulboff.gif'">Turn off the
light</button>

</body>
</html>
```

What Can JavaScript Do?

JavaScript can change HTML attributes.

In this case JavaScript changes the src (source) attribute of an image.

Turn on the light | Turn off the light

Figura 3: Exemplo de página HTML com CSS + Javascript, e seu código-fonte.

Como existiam anteriormente muitos navegadores com formas diferentes de interpretar o HTML, foi necessário estabelecer padrões, para que uma mesma página possa ser exibida da mesma forma por diferentes navegadores. Para estabelecer esses padrões, foi fundado a World Wide Web Consortium (W3C). Em 28 de outubro de 2014, após anos de desenvolvimento, a W3C incluiu em seu acervo de

padrões e recomendações a primeira versão do HTML 5, que trazia consigo recursos como reprodução de mídia, vetores gráficos escaláveis, desenho em *canvas*.

A cada ano novos recursos são adicionados a especificação, tornando o HTML a linguagem de marcação mais usada no mundo. Ele é usado não apenas na web, mas também na construção de diversos aplicativos para Android/Windows, em e-mails, em alguns editores de texto, e embarcado no projeto de muitos softwares.

3. Requerimentos

- Dispositivo com resolução de tela maior ou igual a 600:400 pixels (largura:altura);
- Navegador de internet moderno, capaz de processar HTML 5, CSS 3 e Javascript em suas últimas versões, e que respeite os padrões da W3C.

Exemplos de ambientes de execução compatíveis:

★ Google Chrome 63, instalado no Microsoft Windows 10, executando em um notebook atual de 15 polegadas;
★ Mozilla Firefox 57, instalado no Ubuntu, executando em uma TV de 32 polegadas;
★ Opera 49, instalado no Microsoft Windows 8.1, executando em um Tablet-PC de 10 polegadas.

4. O Simulador de Máquina de Estados Finitos

O **Finite State Machine WebApp** permite que o usuário desenvolva uma máquina de estados finitos de Moore ou Mealy e faça uma simulação com as entradas, saídas e estados dessa máquina. O usuário também pode carregar um arquivo de texto que descreve a máquina.

A interface possui tela de inicialização (*Splash Screen*), efeitos de transição entre as telas e adota uma paleta de cores azul-escuro (*Celurean #0064A8*).

A priori, é apresentada a tela de descrição da máquina de estados finitos.

4.1 Descrição da Máquina

Figura 4: Página inicial, onde a máquina é descrita.

Nessa tela, espera-se que o usuário digite todas as entradas, saídas e estados da máquina. Selecione o tipo de máquina (Moore/Mealy) e clique nas caixas de texto para editar, inserir, remover ou mover itens. Vale lembrar que qualquer tipo de nome pode ser usado, desde que seja composto por caracteres alfanuméricos (sem acentuação e sem espaços, permitido *underscore*). Isso significa que não é necessário chamar os estados de “000” ou “010”, por exemplo (apesar de ser possível), mas pode-se usar denominações genéricas, como “EstadoInicial” ou “Estado_X”. O mesmo se aplica para as entradas e saídas. Mas é necessário inserir todas as entradas, saídas e estados, pois nas próximas telas não será possível adicionar ou remover estes. Quando pronto, clique em “Próximo” para continuar.

Ao avançar, será mostrado a tela da tabela verdade. Nessa tela será descrito, para cada estado, suas saídas e próximos estados.

4.2 Tabela-Verdade

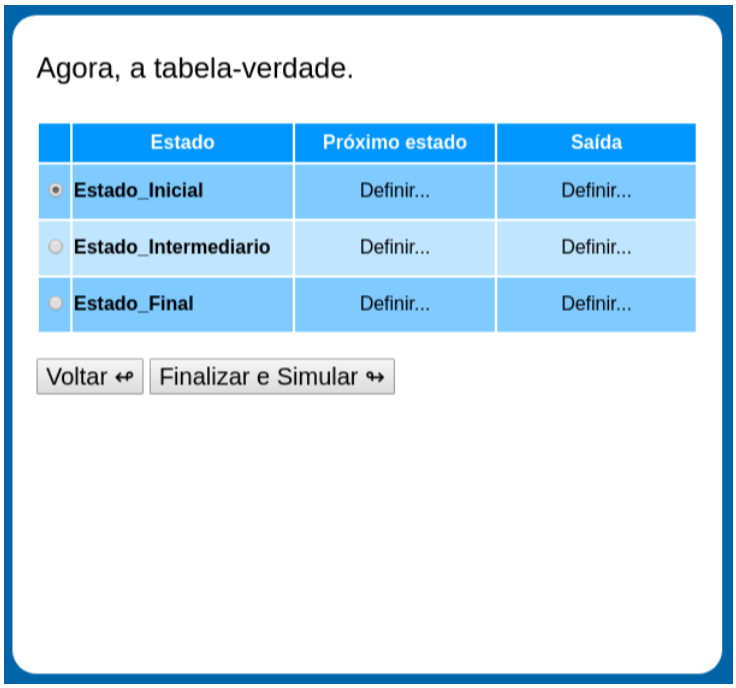

Figura 5: Tabela-verdade da máquina.

Você pode selecionar o estado inicial/reset com as caixas de seleção na primeira coluna da tabela. Além, para cada estado, clique em “Definir...” para definir seus próximos estados e saídas. Para definir um próximo estado, é necessário escrever a condição (Equação Verilog) que leve a este estado. Vale lembrar que a equação Verilog deve utilizar os próprios nomes das entradas, que será substituído durante a simulação pelo seu valor. Por exemplo, se o estado ‘X’ vai para o estado ‘Y’ só quando a entrada ‘Ir’ for ‘1’ e a entrada ‘Pausa’ for ‘0’, você deve definir como próximo estado de ‘X’ uma equação Verilog “Ir & ~Pausa” que leva a ‘Y’.

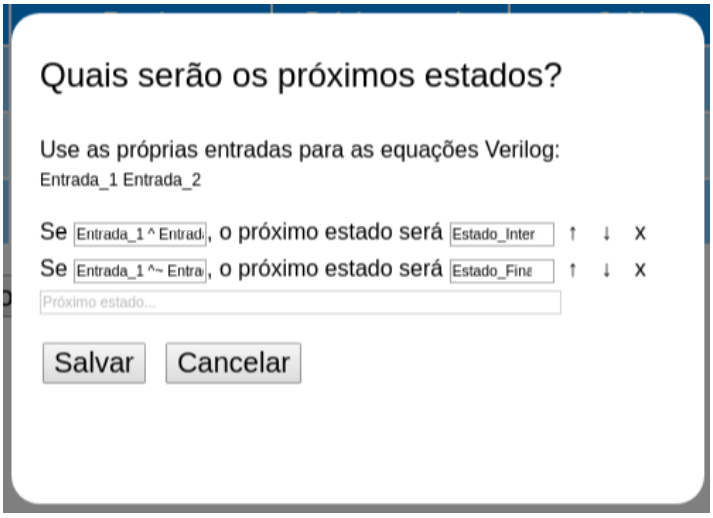

Figura 6: Descrição dos próximos estados (a partir de um estado qualquer).

Defina todos os próximos estados e clique em “Salvar”. Após isso, você deve definir as saídas.

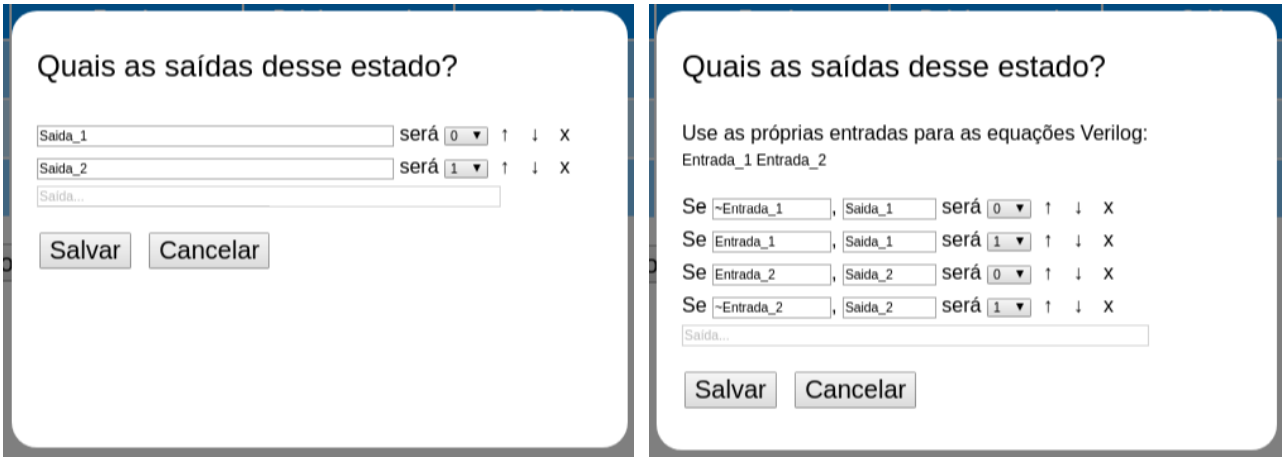

Figura 7: Descrição das saídas (de um estado qualquer), em Moore (esquerda) e Mealy (direita).

Se for uma máquina de Moore, basta escolher 0 ou 1 para cada saída. Se for uma máquina de Mealy, você deve entrar com uma condição (Equação Verilog), e escolher se uma determinada saída será 0 ou 1 quando essa condição for satisfeita. A equação Verilog funciona da mesma maneira que nos próximos estados.

Quando terminar de descrever toda a tabela verdade de sua máquina, clique em “Finalizar e Simular” e a tela principal da simulação será aberta.

4.3 Carregar Arquivos

Na primeira tela quando iniciado o aplicativo, há também a opção “Carregar arquivo”. Com isso, é possível enviar um arquivo que descreve a máquina e a tabela verdade, pulando todas as etapas acima. Esse arquivo deve estar no formato descrito pela especificação do projeto. O formato é:

[0 para Mealy, 1 para Moore]
[Número de estados] [Número de entradas] [Número de saídas]
[Tabela Verdade, onde cada linha contém: os bits de codificação do estado, todas as entradas, os bits de codificação do próximo estado, e todas as saídas]

Neste arquivo, o aplicativo web vai ignorar “Don’t Care” (X) ou transformar “Don’t Care” em ‘0’.

```
1
3 2 2
0 0 0 1 0 1 0 1
0 0 1 0 0 1 0 1
0 0 0 0 1 0 0 1
0 0 1 1 1 0 0 1
0 1 0 0 0 1 0 0
0 1 0 1 1 0 0 0
0 1 1 0 0 1 0 0
0 1 0 1 0 1 0 0
1 0 0 0 1 0 0 0
1 0 0 1 1 0 0 0
1 0 1 0 0 0 0 0
1 0 1 1 1 0 0 0
```

Figura 8: Descrição de uma máquina Moore de 3 estados, 2 entradas e 2 saídas através de um arquivo de texto.

4.4 Simulação

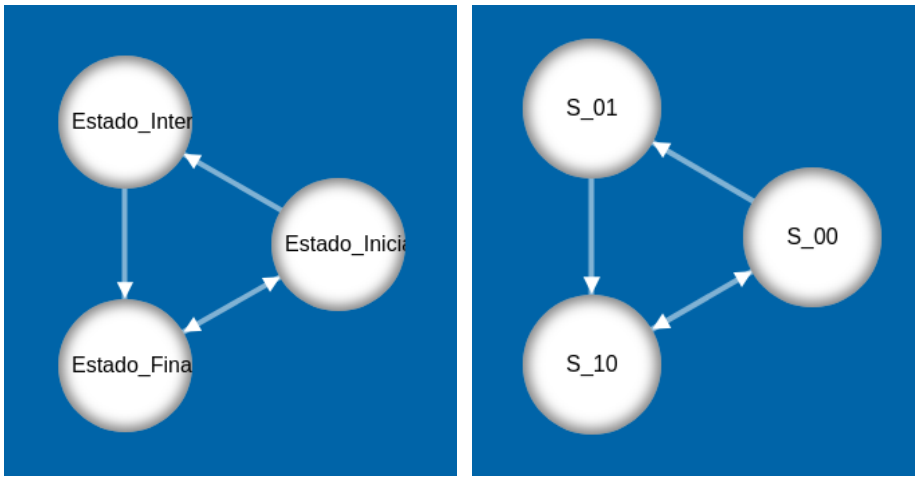

Figura 9: Simulação de uma mesma máquina descrita pela interface (esquerda) e pelo arquivo (direita).

Existem dois menus nessa tela. Na extrema esquerda, está o menu das entradas, e na extrema direita o menu das saídas. Passe o mouse sobre os menus para exibi-los. Você pode travar um menu (para que ele não fique oculto novamente) selecionando “Fixar”.

No menu das entradas você consegue definir se cada entrada será 0 ou 1, ativar um sinal de clock, resetar a máquina e escolher o modo como você vê ela desenhada na tela.

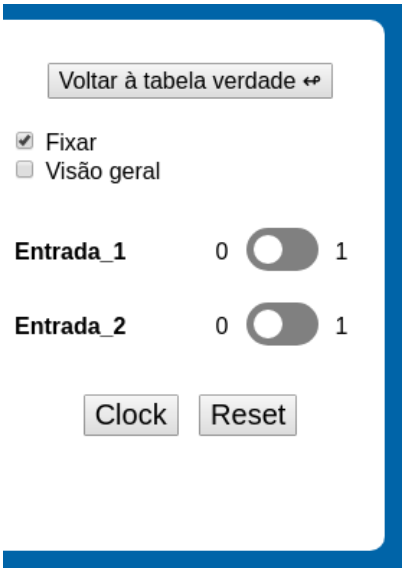

Figura 10: Menu de entradas na tela de simulação.

Se sua máquina for muito grande, a ponto de não caber na tela, será mostrado apenas o estado atual e estados adjacentes. A tela irá seguir o estado atual, ou seja, se o estado mudar, a tela irá mover de modo a mostrar sempre o estado atual em destaque. Mas se você deseja visualizar toda a máquina, você pode selecionar “Visão geral” e a máquina será reduzida de modo a caber na tela. A “Visão geral” não afeta máquinas pequenas ou que estejam já completamente visíveis na tela.

Durante a simulação, você pode colocar cada entrada em 0 ou 1, e clicar em clock para enviar um sinal de clock e visualizar possíveis mudanças/animações. Visando a agilidade, os links “0” e “1” ao lado do seletor de cada entrada possibilita alternar a entrada, respectivamente para 0 e 1, e logo imediatamente enviar um sinal de clock. O modo como a simulação é feita fica a gosto do usuário.

Nota: não há diferença visual entre máquinas de Moore e Mealy. Isso porque este aplicativo permite adicionar infinitésimos estados, com incontáveis entradas e saídas (não há um limite). Escrever todas essas equações Verilog na tela, junto com todas as saídas, não produziria um resultado agradável visualmente (a tela ficaria cheia de “lixo visual”). **Sendo assim, a diferença entre as máquinas Moore e Mealy é notada no próprio funcionamento**: as saídas na máquina de Moore só mudam com o Clock, enquanto na máquina de Mealy, elas mudam com qualquer alteração nas entradas.

No menu das saídas é possível visualizar o estado atual e as saídas.

Figura 11: Menu de saídas na tela de simulação.

Sinta-se a vontade para ajustar o tamanho e mover a janela do navegador a qualquer momento, pois a simulação é adaptável à janela.

5. Implementação

Na implementação deste trabalho, foram utilizadas algumas técnicas de programação web para dispor os elementos e as animações.

5.1 Leitura de Arquivos

Com o HTML 5, também surgiu a possibilidade de leitura e processamento de arquivos diretamente do JavaScript. Utilizando da biblioteca FileReader, o aplicativo faz a leitura do arquivo de texto selecionado pelo usuário, e utilizando de expressões regulares (*Regex*), interpreta-o.

5.2 Transição entre Telas

A experiência do usuário (UX) é um ponto que deve ser fortemente considerado em um projeto que utilize interfaces gráficas. Neste aplicativo, a interface é repleta de transições. Estas transições são feitas utilizando de uma biblioteca de JavaScript: o jQuery. O jQuery é uma biblioteca que tem como princípio a simplificação do JavaScript, e entre essas simplificações, se destaca o uso de animações. Entre as telas, é utilizada a transição de “fading”. Em janelas pop-up, é utilizada a transição “sliding”.

5.3 Parser de Verilog

Neste aplicativo, é utilizado equações Verilog para designar condições baseadas nas entradas. Para tal, foi necessário implementar um *Parser* de Verilog, ou seja, um algoritmo capaz de interpretar uma equação Verilog, verificar se é válida, e resolvê-la. O algoritmo implementado é próprio e foi criado do zero. Ele funciona a base de expressões regulares e substituição:

- ➔ Substituir na equação todos os valores das entradas. Por exemplo, se “Entrada_1” for igual a ‘0’, substitua todas as ocorrências de “Entrada_1” por ‘0’.
- ➔ Enquanto houverem expressões para serem resolvidas:
  - ◆ Resolver Negações (“not ~”).
  - ◆ Resolver Es (“and &”), Ous (“or |”), Ous exclusivos (“Xor ^”) e Ous exclusivos negados (“Xnor ^~ ~^”).
- ➔ Retornar como resultado ‘0’ ou ‘1’.

5.4 Disposição dos Estados

É importante pensar em uma disposição simples, mas eficiente, para os estados na tela. Isso porque, dependendo da forma como são colocados, alguns estados podem sobrescrever outros, e as setas de transição podem não ficar visíveis. Neste aplicativo, os estados são dispostos em uma circunferência. Isso faz com que os estados sejam igualmente espaçados, nenhum sobrescreva outro, e ainda por cima, todas as transições são visíveis.

5.5 Animação de Troca de Estado

A animação de troca de um estado para outro é feita também utilizando jQuery. Um pseudo-estado auxiliar é criado e toma a posição e características do estado atual. Depois disso, é animado para transladar para a posição do próximo estado. Após, este é removido da página e a animação está completa. Nesse meio termo, ambos o estado atual e próximo estado piscam (*blink*) de uma cor de destaque (amarelo).

6. Referências Bibliográficas

History of the Internet. Disponível em <https://www.youtube.com/watch?v=9hIQjrMHTv4>. Acessado em 28 de novembro de 2017.

História dos navegadores: do www ao Chrome. Disponível em <https://olhardigital.com.br/noticia/historia-dos-navegadores-do-www-ao-chrome/7285>. Acessado em 29 de novembro de 2017.

Tim Berners-Lee. Disponível em <https://en.wikipedia.org/wiki/Tim_Berners-Lee>. Acessado em 29 de novembro de 2017.

What is jQuery?. Disponível em <https://jquery.com/>. Acessado em 2 de dezembro de 2017.