# MO601 - Projeto 1

**Casio Pacheco Krebs - ra264953**

[1]Instituto de Computação - IC
Universidade Estadual de Campinas - UNICAMP
Campinas, Brasil

## 1. Introdução

O objetivo deste projeto é implementar um simulador de circuitos lógicos, que permita gerar o histórico dos valores dos sinais de entrada e de saída para cada porta lógica. Onde serão projetados dois fluxos de simulação, o primeiro considerando que o tempo de propagação da resposta a partir da variação de alguns dos sinais de entrada seja de 0 ciclos, e outro sendo de 1 ciclo de atraso [1].

A descrição desse relatório está separado em Execução (Seção 2), onde são apresentados as dependências e o seu procedimento de instalação e execução, Implementação (Seção 3), onde são detalhados os métodos e funções utilizadas, Testes (Seção 4), ande são apresentadas as metodologias dos testes executados, e por fim, Conclusão (Seção 5), onde são apresentados os aprendizados tirados por este projeto.

## 2. Execução

Nessa seção estão descritos as etapas da execução do projeto, Para a sua implementação foi utilizando a linguagem de programação *Python* em sua versão 3.9.

### 2.1. Dependências

Para a execução do projeto deve-se ter instalado as seguintes bibliotecas:

- git: utilizado para realizar o download dos arquivos a partir do GitHub;
- python3.9: para realizar a execução do script desenvolvido;
- csv: Utilizado para a escrita dos sinais em arquivos .csv;
- os.path: Utilizado para a manipulação de diretórios;

### 2.2. Como executar

O ambiente de execução por ser configurado a partir do DockerFile entregue pelo GOOGLE Classroom, o qual se encontra descrito na Figura 1:

O código desenvolvido se encontra no repositório, gerado pelo GitHub, acessível através do link: https://github.com/CPKrebs/MO601-P1.git . No repositório se encontram o código-fonte do simulador, assim como exemplos de circuitos utilizados para a realização de testes de execução do simulador.

Para a correta execução do simulador desenvolvido, as representações de cada circuito, assim como a lista de estímulos, devem estar dispostas em sub-pastas no diretório $/MO601 - P1/test/$. O simulador foi projetado para realizar o procedimento com todas as sub-pastas existentes nesse diretório, dessa forma,

```
FROM ubuntu

RUN export DEBIAN_FRONTEND=noninteractive \
 && apt-get update \
 && apt-get install -y python3.9

COPY Projeto_1_Casio_Krebs.py /

RUN mkdir test
COPY test/ test/

RUN python3 Projeto_1_Casio_Krebs.py
```

**Figura 1.** **Código do DockerFile implementado**

caso alguma subpasta não conste com os arquivos de descrição das portas lógicas (*circuito.hdl*) ou das listas de estímulos (*estimulos.txt*), será apresentada uma mensagem de erro para o usuário.

O DockerFile realiza a cópia dos arquivos existentes no diretório */MO*601 – *P*1/*test*/ e o scripy **Projeto-1-Casio-Krebs.py** para dentro do contêiner, e em seguida aciona automaticamente o script. Dessa forma, apos a etapa de construção da imagem Docker, é necessário apenas realizar a cópia dos resultados das simulações para o Host.

A lista de etapas para a realização dos experimentos pode ser vista a baixo:

- Antes de realizar a construção da imagem Docker, é necessário realizar a cópia dos novos testes de simulação dentro da pasta "test".
- Na pasta do repositório execute: docker build -t casio_p1 .
- Em seguida: docker run -d –name casio_p1_exec casio_p1
- Por fim: docker cp casio_p1_exec:/test/. test/.

### 2.3. Dados de Saída

A saída do programa baseia-se na criação de dois arquivos CSV, denominados de *saida*0.*csv* e *saida*1.*csv*. O primeiro apresenta o histórico dos valores dos sinais de entrada e de saída das portas logicas, ordenadas alfabeticamente, para cada unidade de tempo, quando o atraso de propagação das portas lógicas é de 0 unidades de tempo, enquanto a segunda apresentada o mesmo histórico, mas para 1 unidades de tempo, o atraso de propagação das portas lógicas.

## 3. Implementação

Nessa seção serão descritos as etapas das implementações do simulador, separadas em rotinas gerais e específicas para cada nível de atraso.

### 3.1. Leitura de dados

Inicialmente é realizada a leitura das sub-pastas existentes na pasta "*test*/"através do da função os.listdir(). Em seguida é feita a leitura do circuito existente no

arquivo *circuito.hdl*, onde os sinais de entrada das portas logicas são marcadas para a identificação dos sinais de entrada geral do circuito, i.e. sinais de entradas de portas logicas que estão conectadas às saídas de outras portas logicas são desmarcadas.

Por fim, é feita a leitura dos estímulos existente no arquivo *estimulos.txt* onde, ocorre o tratamento da notação simplificada, e para os casos de indicadores de passagem de tempo maiores que 1, é feito um tratamento adicional de conversão, por exemplo, para a representação de "+3" é realizada a conversão para três indicadores "+1" seguidos, para simplificar a execução com o atraso 0 e 1.

Para a realização da simulação, é inicialmente feita uma passagem pelas portas para verificar se alguma porta é afetada pelos sinais intermediários quando o circuito é iniciado com todos os sinais em N.L. 0, e também, o marcador de tempo é iniciado com o valor 0. Em seguida é iniciada a etapa de processamento dos estímulos, onde é configurada uma pilha que armazena os IDs de todas as portas lógicas atingidas diretamente pelos estímulos delimitada pelo primeiro indicador de passagem de tempo, essa pilha é denomina de "Modificado".

### 3.2. Atraso 0

Para a realização da simulação com atraso 0, é programado um loop que percorre por toda a lista de estímulos, realizando todas as modificações dos sinais de entrada geral do circuito e armazenando os IDs na pilha "Modificado". Quando identificado um indicador de passagem de tempo, o marcador de tempo é acrescido por 1 unidade de tempo, e em seguida, é iniciado um laço de repetição onde, todas as portas lógicas marcadas são executadas e os valores de saída atualizados. Caso o novo valor de saída difira do atual, é analisado se essa saída está diretamente conectada a outra porta lógica, se estiver, o ID dessa nova porta logica é adicionado na pilha de modificados, para também ser tratada.

O código se mantêm no laço até que não haja nenhuma porta lógica para ser atualizada. Devolvendo em seguida o controle, para o loop de processamento dos estímulos. A simulação se encerra quando não há nenhum estímulo não processado e quando a pilha de modificados estiver vazia.

### 3.3. Atraso 1

Para a realização da simulação com atraso 1, quando identificado um indicador de passagem de tempo, o marcador de tempo também é acrescido por 1 unidade de tempo, e em seguida, também é iniciado um laço de repetição onde, todas as portas lógicas marcadas são executadas e os valores de saída atualizados. Porém, ao contrário da simulação com atraso 0, quando o novo valor de saída difere do atual, o ID de todas as portas logicas afetadas diretamente, são adicionadas em uma pilha auxiliar. Dessa forma, apenas são tradadas as portas lógicas marcadas previamente, no mesmo marcador de tempo.

Quando encerrado o segundo laço, todos os IDs salvos na pilha auxiliar são adicionados na pilha principal, e então o controle é devolvido para o para a função de processamento dos estímulos. Assim, quando a lista de estímulo se encerra, apenas são tratados os casos de modificações geradas pelos sinais intermediários

do circuito. Encerando a simulação quando não houver mais sinais intermediários para serem propagados.

## 4. Metodologias de testes e avaliação

Para a realização da corretude do simulador foram testadas hipóteses onde o circuito aplicado não apresenta memória e hipóteses onde apresenta memória.

Circuitos lógicos que não apresentam memória, são circuitos onde o valor de saída depende apenas dos valores presentes nas suas entradas, dessa forma para verificar a corretude do simulador, foram aplicados circuitos sequenciais como o apresentado na Figura 2, por exemplo, composto por todas as portas lógicas suportadas.

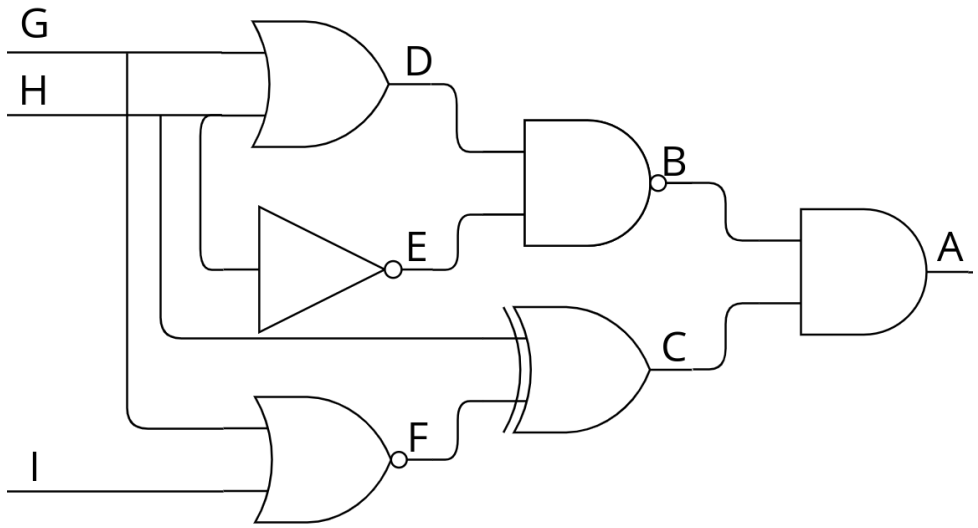


**Figura 2.** **Exemplo de circuito que não apresentam memória usado para testes**

Para todos os exemplos sequenciais testados, foram aplicados uma lista de estímulos que apresentavam todas as combinações possíveis de entrada, separadas pela marcação de uma passagem de unidade de tempo, e em seguida era realizado a checagem manual de todas as portas lógicas simuladas para cada combinação específica. O procedimento executado foi aplicado tanto para a simulação com atraso de propagação 0, quanto para a simulação com atraso de propagação 1, permitindo assim verificar a corretude das portas lógicas.

Com o intuito de se aprofundar na análise do efeito do atraso na propagação da resposta, foi aplicado no simulador desenvolvido, um circuito composto por 5 portas lógicas OR em sequências, conforme apresentado na Figura 3.

Em seguida, foi aplicado um estímulo em apenas um dos sinais de entrada da porta lógica de entrada (F), enquanto as demais (G-K) foi mantida em 0. Onde foi possível observar o sinal de entrada sendo propagado pelas portas lógicas com um atraso fixo. Por fim, foi aplicado no mesmo circuito, uma lista de estímulos composta pela variação do nível lógico do sinal F, com diferenças de 1 e 2 unidades de tempo, onde foi possível observar a corretude do simulador desenvolvido, para esses exemplos.

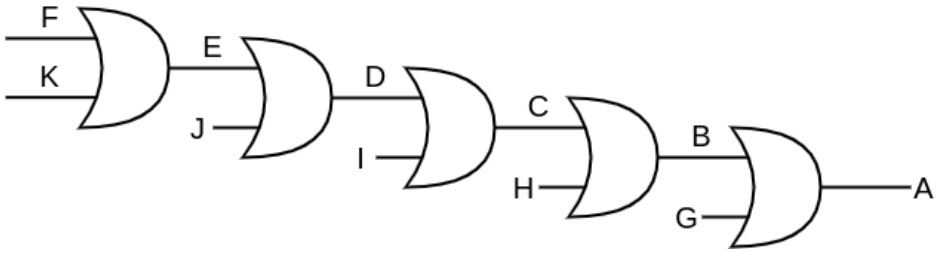


**Figura 3.** **Exemplo de circuito de portas lógicas encadeadas**

Para realizar a simulação de circuito lógicos apresenta memória, i.e. circuitos onde sua saída não depende apenas dos valores de entrada, mas também dos valores de passagem de tempo anteriores, foram aplicados circuitos que apresentam realimentação, como os que podem ser vistos na Figura 4.

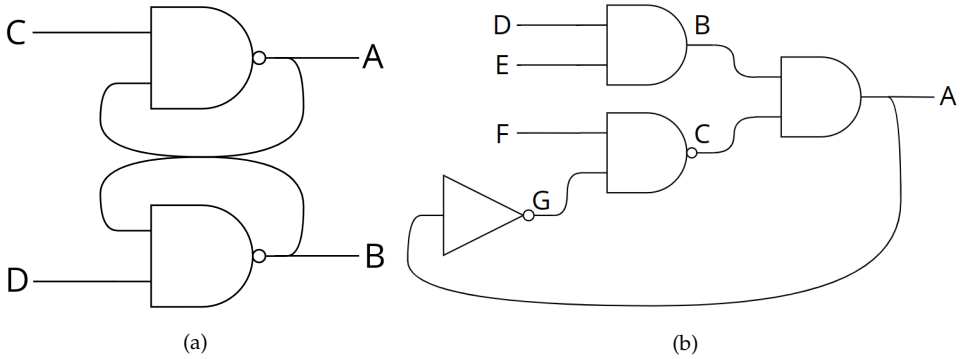


**Figura 4.** **Exemplo de circuito que apresentam memória usado para testes**

Como é possível observar o circuito (a) da Figura 4, representa o circuito Latch SR, onde a entrada C representa a entrada de SET, enquanto a entrada D representa a entrada de RESET, assim para verificar a corretude do simulador, foi aplicado o fluxo aonde, partindo do repouso tenta setar em 1, repousar e em seguida setar em 0, o sinal de saída A. O mesmo procedimento foi aplicado nas representações dos circuitos Latch D e Flip-Flop D. Com base nessa simulações, foi possível observar que o projeto desenvolvido estava apresentando o comportamento esperado para esses circuitos, tanto para o fluxo com atraso de propagação 0, quanto para o fluxo com atraso de propagação 1.

## 5. Conclusão

Durante o desenvolvimento deste projeto de simulador de portas lógicas, foi possível entender o efeito do atraso na propagação da resposta e estudar como que esse comportamento pode afetar o resultado da simulação. Para isso foi necessário planejar os circuitos de testes e os estímulos de entrada, a fim de validar a corretude do algorítimo desenvolvido.

## 6. Código

O DockerFile completo, será entregue juntamente com este relatório na plataforma GOOGLE Classroom.

## Referências

[1] https://www.ic.unicamp.br/ rodolfo/mo601/projeto1/, acessado em 19/03/2023