# Implementação de um Filtro de Tráfego de Dados

**Leonardo Schilling**

Curso de Ciência da Computação
Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)

Porto Alegre – RS – Brazil

leonardo.schilling@edu.pucrs.br

## 1. Introdução

Este relatório aborda a implementação do módulo **traffic_filter**, que tem como objetivo detectar determinados padrões em um fluxo de entrada de dados serial. O módulo é projetado para identificar esses padrões, que correspondem a "ataques", e notificar sua ocorrência. A primeira parte do trabalho trata da definição e diagramação da máquina de estados, e a segunda parte consiste na apresentação de um cenário de operação, no qual serão programados quatro padrões diferentes, pelo menos três padrões serão detectados e, em seguida, um novo padrão será reprogramado e detectado.

## 2. Máquina de Estados

A máquina de estados possui 8 estados que serão apresentados após o diagrama.

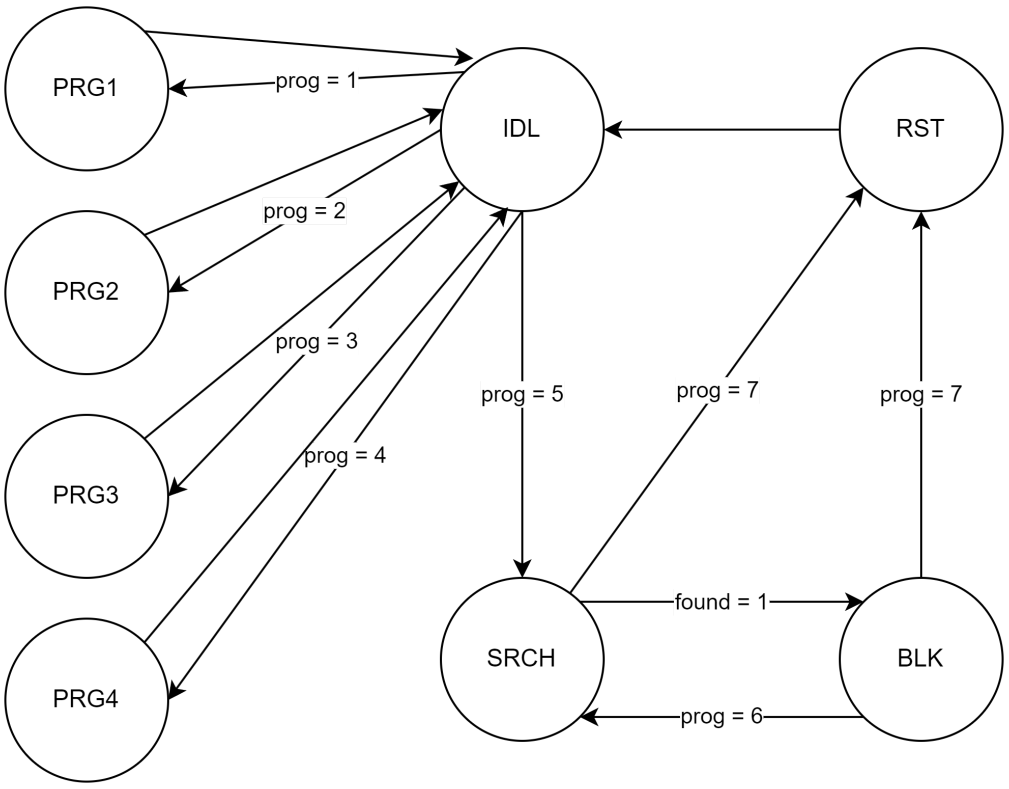



**Diagrama da Máquina de Estados**

## 2.1 Descrição textual da Máquina de Estados

Abaixo serão descritos em detalhes o comportamento, os estados e as respectivas transições que foram apresentadas no diagrama.

**IDL**

Aguardando que um padrão de ataque seja programado.

**PRG1, PRG2, PRG3, PRG4**

Nesses estados, a máquina de estados recebe comandos de programação que definem os padrões de ataque a serem detectados. Dependendo do valor de *prog*, padrões são armazenados e habilitados para comparação.

**SRCH**

Nesse estado, a máquina de estados busca no fluxo de entrada por qualquer um dos padrões de ataque programados. Se um padrão de ataque for detectado (*found*=1), a máquina de estados muda para **BLK**.

**BLK**

Neste estado, o alarme é ativado e a saída de dados é bloqueada. Dependendo do valor de *prog*, a máquina de estados pode voltar para **SRCH**(se *prog*=6) ou para **RST** (se *prog*=7).

**RST**

Neste estado, a máquina de estados zera os registradores *alarme_int* e *sel*, e então retorna para o estado **IDL**.

## 3. Cenário de Operação

O cenário a ser apresentado consiste na definição de quatro padrões, e, posteriormente, a programação e detecção de um quinto padrão.

Para criação de uma entrada pseudo aleatória, alterei a semente do *tb.vhd*:

```
constant seed : std_logic_vector(GP-1 downto 0) := "1101101110110001010";
```

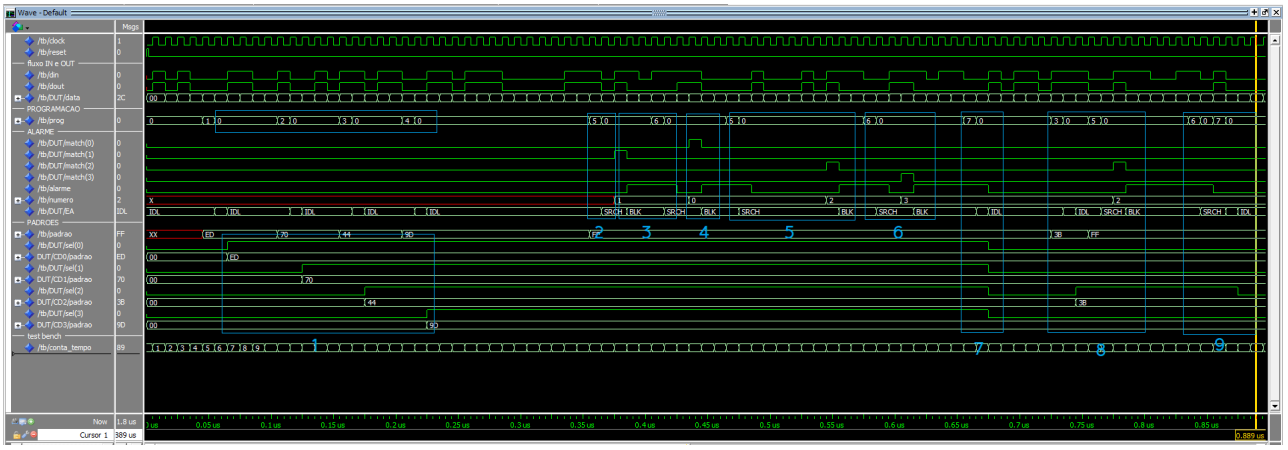
Destaco abaixo a escolha dos quatro novos padrões:

```
constant padrao_de_teste : padroes := (
 (t =>  4, prog => "001", padrao => x"ED"),
 (t =>  10, prog => "010", padrao => x"70"),
 (t =>  15, prog => "011", padrao => x"44"),
 (t =>  20, prog => "100", padrao => x"9D"),
```

## 4. Análise das ondas e descrição dos eventos

Abaixo apresento um print do gráfico de ondas, com alguns pontos destacados que serão descritos abaixo.

A imagem também estará disponível em anexo no momento da entrega.



**Cenário de operação no ModelSim**

**4.1.** *prog* {1,2,3,4} programando os padrões e habilitando a comparação. Nota-se que {*sel*(0), *sel*(1), *sel*(2), *sel*(3)} só voltam a zerar no evento 7.

**4.2.** Início da busca (SRCH na figura) quando *prog*=5.

**4.3.** Match com o padrão 2, com a máquina indo para o estado BLK. Percebe-se que a saída fica zerada e o alarme é ativado.

**4.4.** *Prog*=6, permitindo comparar novamente (habilitando a saída e desativando o alarme). Logo em seguida houve um match com o padrão 1, novamente bloqueando a saída e ativando o alarme.

**4.5.** Voltou-se novamente para o estado SRCH (nota-se que prog = 6), e o tráfego é liberado novamente, desativando o alarme, habilitando a saída e, logo em seguida, detectando o padrão 3, bloqueando e ativando o alarme novamente.

**4.6.** Valor de *prog*=6 novamente, permitindo seguir com a busca. Em seguida, há um match com o padrão 4, novamente bloqueando a saída e ativando o alarme.

**4.7.** Reset (prog = 7), desativando os seletores, habilitando a saída que estava bloqueada e em seguida há um retorno para o estado IDL.

Os eventos 1 a 7 correspondem aos eventos apresentados no enunciado do trabalho, e, os eventos a seguir (8 e 9) tratam da programação de um novo padrão e sua detecção, e serão descritos a seguir.

**4.8.** Redefinição do sel(3) (prog = 3), com um novo padrão. Em seguida ocorre a detecção deste padrão e o comportamento de bloqueio acontece.

O comportamento neste caso ocorre pois houve a alteração no tb.vhd para programação de um novo padrão, conforme solicitado.

A alteração foi a seguinte:

```
(t =>  72, prog => "011", padrao => x"3B"),    -- Programação de um novo padrão
(t =>  75, prog => "101", padrao => x"FF"),    -- ATIVA A COMPARAÇÃO (5)
(t =>  83, prog => "110", padrao => x"FF"),    -- reinicia a comparação (6)
(t => 85, prog => "111", padrao => x"FF")     -- reinicializa (7)
```

**4.9.** O circuito volta para o estado de busca (prog = 6), e em seguida é resetado (prog = 7), desativando o alarme, retomando a saída e desativando o sel(2), e logo em seguida retorna para o estado IDL.

## 5. Conclusão

O circuito proposto foi implementado com sucesso em linguagem VHDL, com a correta modelagem da entidade, dos sinais internos, dos módulos compara_dado, dos registradores dependentes da FSM e do registrador de deslocamento, bem como a atribuição correta das saídas.

Na segunda parte do relatório, foi apresentado um cenário de operação que contemplou a programação de quatro padrões diferentes. Durante os testes realizados, foi possível observar a detecção de todos esses padrões. Além disso, foi realizada com sucesso a reprogramação de um novo padrão, demonstrando a flexibilidade do sistema. A detecção desse novo padrão foi verificada, confirmando a capacidade do traffic_filter em identificar corretamente as igualdades com os padrões definidos.