Relatório *Traffic filter*:

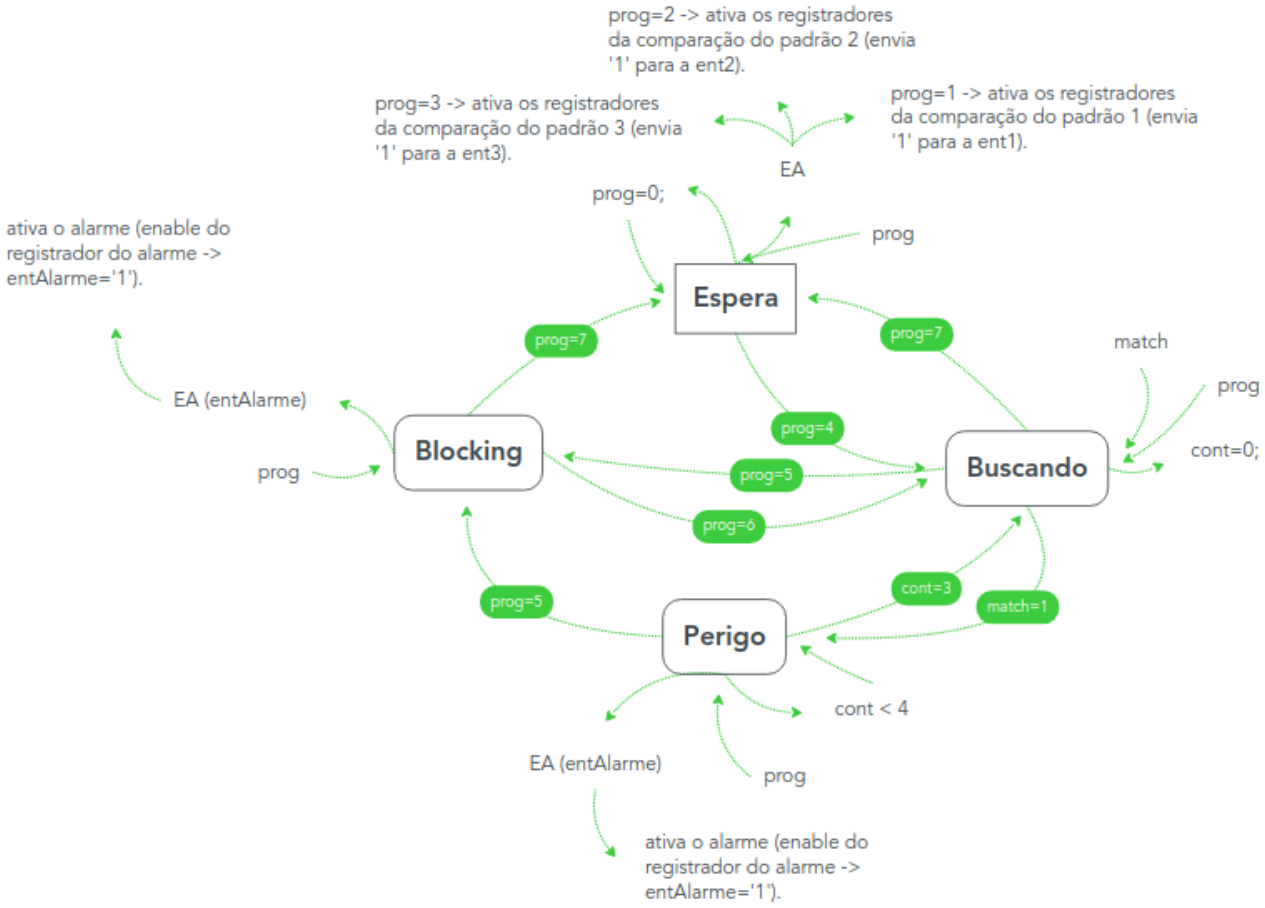
Alunos: Eduardo Bernardon e Filipe Bandeira

Engenharia de Computação.

- Máquina de estados:

A máquina de estados implementada possui quatro estados: espera, buscando, perigo e *blocking*. Cada estado recebe um valor de "prog" ou, enquanto no estado buscando, um "match" e executa determinada ação. Assim, gerando o funcionamento correto do sistema.

- **Estado espera:** esse estado é ativado quando o sistema está esperando um valor de "prog" para executar determinada ação, ou seja, enquanto o prog continua com valor zero, continuará esse estado ativo. Assim, quando o estado receber prog=1 ativará a comparação com o padrão 1 (ent1='1'), quando receber prog=2 ativará a comparação com o padrão 2 (ent2='1'), e quando receber prog=3 ativará a comparação com o padrão 3 do sistema (ent3='1').
- **Estado buscando:** é ativado no momento em que é recebido um prog=4 no estado espera. Enquanto estiver nesse estado, é verificado se ocorre alguma igualdade no sistema, assim, se ocorrer (recebe um match dos comparadores), é ativado o estado perigo. Caso receba um prog=5, é ativado diretamente o estado *blocking*. *S*e receber um prog=7 o sistema reativa o estado espera. Nesse estado, o contador de clocks (cont) está sempre em 0.
- **Estado Perigo:** É acionado, como explicado anteriormente, pelo match=1 recebido pelo estado buscando. Ele fica ativo até quatro *clocks* com o alarme ativado e *r*etorna, posteriormente, ao estado buscando, caso não receba prog=5 (ativaria o estado blocking) e zera o contador de *clocks (cont).*
- **Estado Blocking:** fica ativo quando recebe um prog=5 (ataque) no estado perigo ou buscando, os quais podem detectar um ataque, e ativa o alarme. Após o bloqueio dos dados, quando recebe um prog=7, o estado espera é reativado. Também, o estado blocking pode reativar o estado buscando quando receber um prog=6.



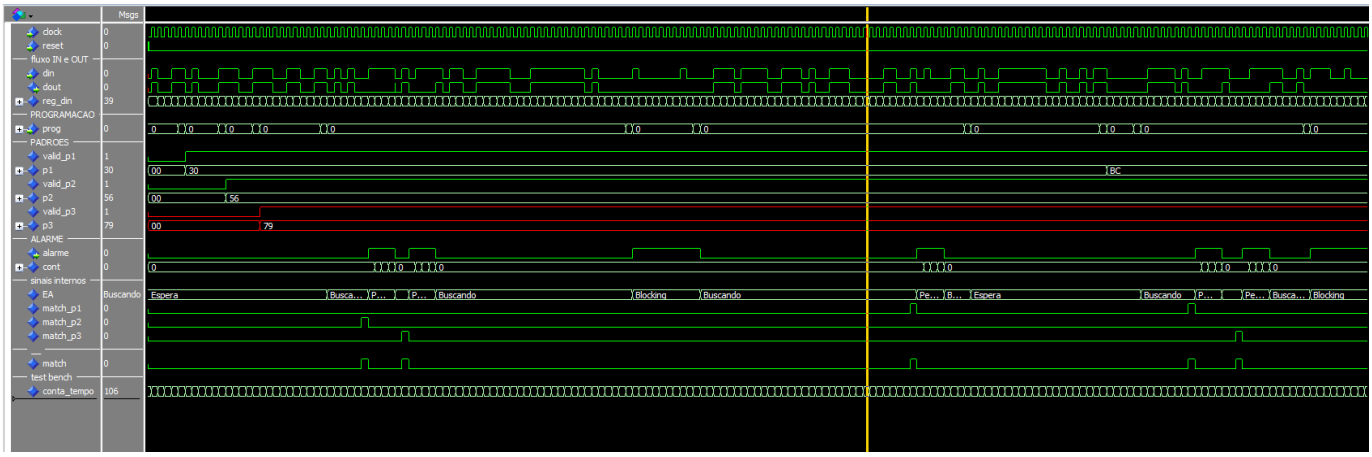
- Cenário de execução:

Para demonstrar o funcionamento da máquina de estados e do circuito utilizando o model sim, foi utilizado os seguintes padrões para gerar um determinado cenário de execução:

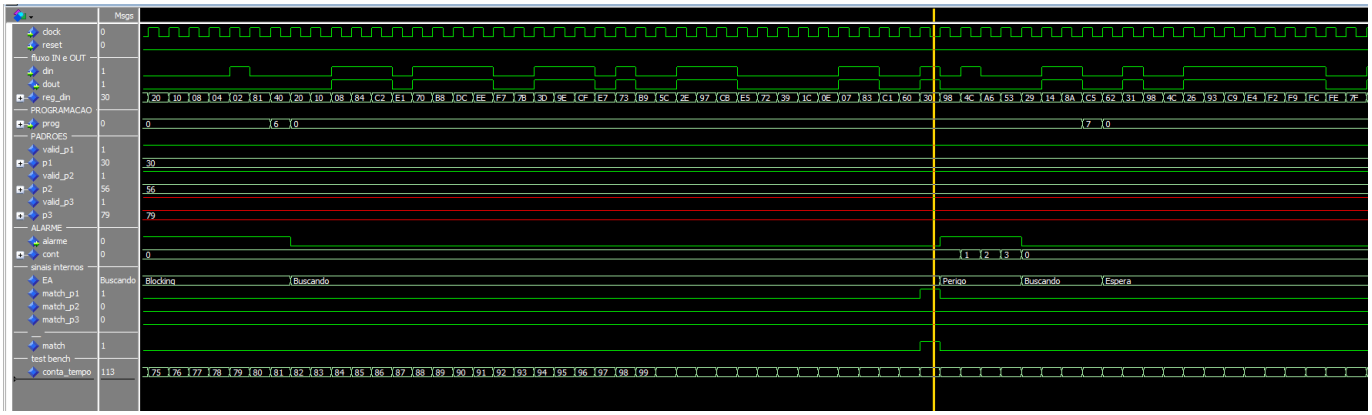
```
type padroes is array(natural range <>) of test_record;

 constant padrao_de_teste : padroes := (

  (t =>  4, prog => "001", padrao => x"30"),   -- Valor para a comparação 1.

   (t =>  10, prog => "010", padrao => x"56"), -- Valor para a comparação 2.

   (t =>  15, prog => "011", padrao => x"79"), -- Valor para a comparação 3.

   (t =>  25, prog => "100", padrao => x"FF"),    -- Ativa comparação

   (t =>  70, prog => "101", padrao => x"FF"),    -- Bloqueia

   (t =>  80, prog => "110", padrao => x"FF"),    -- Reinicia a comparação
```

```
    (t => 120, prog => "111", padrao => x"FF"),     -- Reinicializa
    (t => 140, prog => "001", padrao => x"BC"),    -- Novo valor para a comparação 1.
    (t=> 145, prog => "100", padrao => x"FF"),      -- Ativa a comparação
    (t =>  170, prog => "101", padrao => x"FF"),    -- Bloqueia
    (t =>  190, prog => "110", padrao => x"FF"),    -- Reinicia a comparação
    (t => 200, prog => "111", padrao => x"FF")      -- Reinicializa
  );
```

Nesse padrão de teste, foi trocado os valores padrões dos primeiros padrões de comparação (p1, p2 e p3) que são disponibilizados na especificação do trabalho. O padrão 1 passou a encontrar uma coincidência com o valor x“30”, o padrão 2 com o x“E0” e o padrão 3 com o x“79”. Além disso, foi adicionado outra comparação (nesse momento somente no p1 com o valor x “9B”) e posteriormente repetimos o funcionamento circuito completo, ou seja, foi ativada a comparação, bloqueado a passagem dos dados, retornado à busca de comparações e reinicializado. A *seed* foi alterada para "1101101110010110010" com a finalidade de gerar um cenário mais autêntico possível. Assim, gerou o sequente cenário do *traffic_filter:*

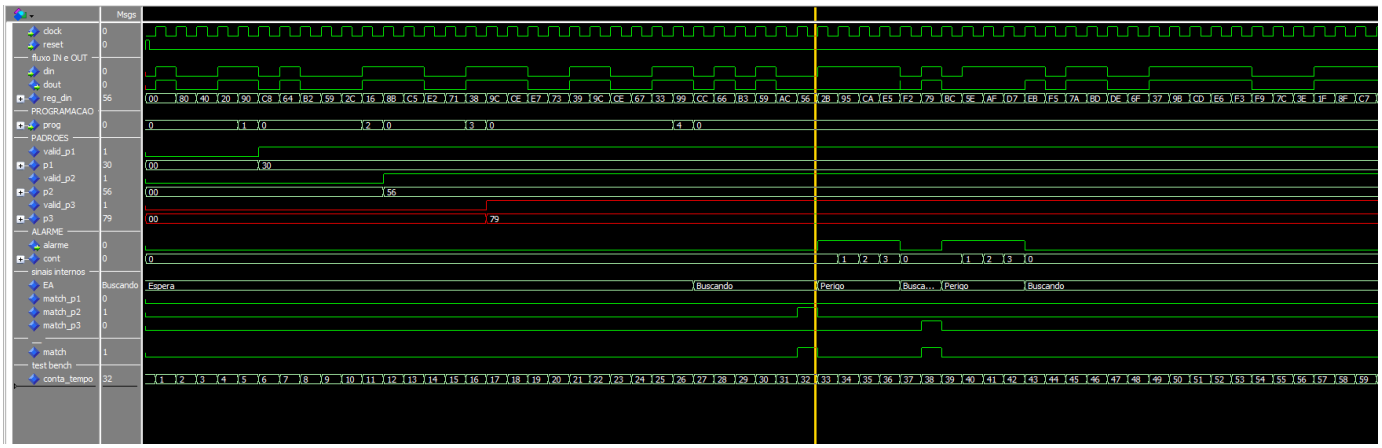


Com a alteração da *seed* do circuito, a onda “din” foi alterada em relação à disponibilizada na especificação do trabalho. Em decorrência, os valores “reg_din” (saída do registrador de deslocamento) também foram alterados, já que a sequência dos bits acumulados no registrador de deslocamento foi diferente.

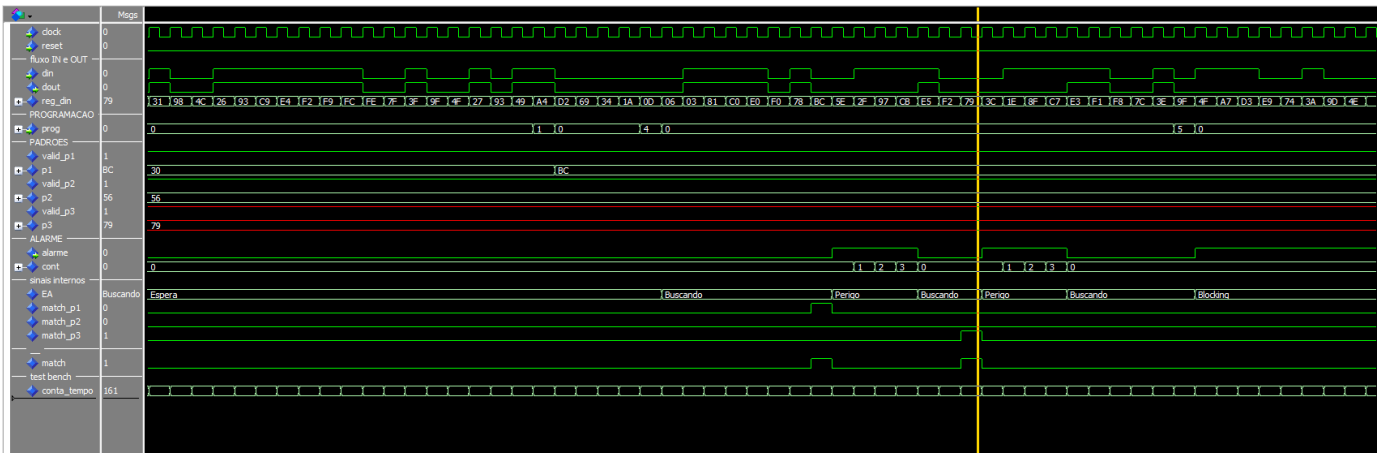


Após os números serem gerados, disponibilizados aos comparadores e o estado da máquina de estados tornar-se “buscando” (a máquina de estados recebeu um valor de prog = ’100’) é encontrado o valor correspondente ao disponibilizado no padrão de comparação 1, o x “30”. Quando identificado, o sistema entra no estado perigo, aciona o alarme e bloqueia a saída dos dados (a onda da variável “dout” fica em zero durante todo o estado perigo), enquanto o estado perigo fica ativado, retornando após quatro *clocks* contados pelo “cont” ao estado buscando. Assim, o sistema é permitido bloquear o tráfego de dados outras vezes com os outros valores inseridos nos padrões de comparação, como é mostrado a seguir:

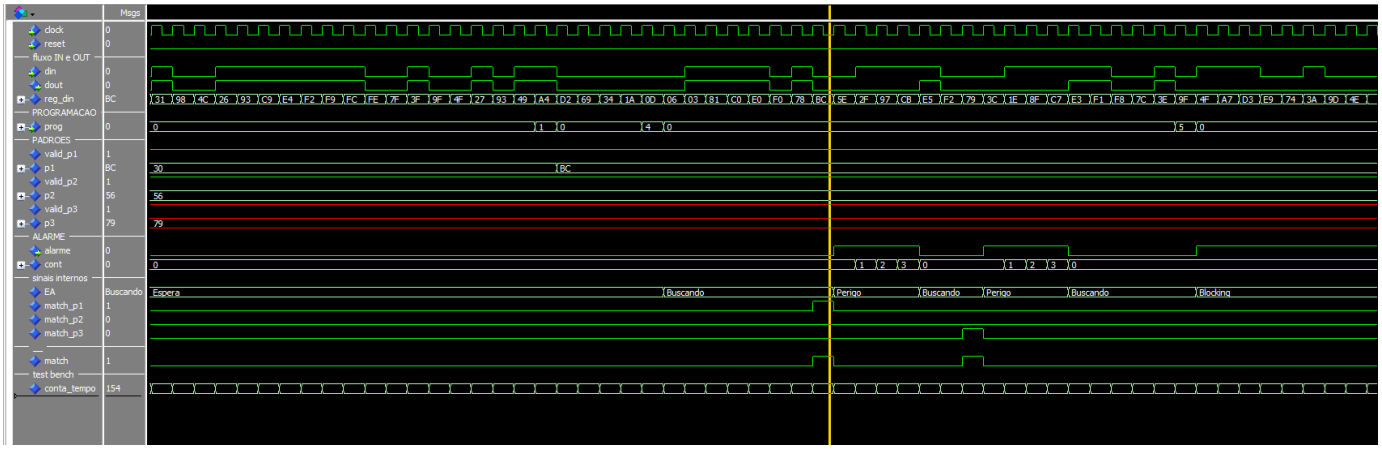
Padrão x"56" no p2:



Padrão x"79" no p3:



Além dos valores iniciais dos padrões, foi comentado anteriormente a alteração do valor do comparador 1, variável “p1”, durante a execução do sistema. Com isso, o valor x “30” alterado para x"BC" é encontrado durante a execução e bloqueado da mesma maneira:



Para encontrar esses valores padrões, é dado um “match”, ou seja, é enviado um bit pelos comparadores do sistema (p1, p2, p3), os quais ativam o estado perigo e disparam o alarme, como é possível ver nas ondas: “match”, “match_p1”, “match_p2”, “match_p3” e alarme. Além disso, essa comparação só é possível quando é ativado o registrador pela máquina de estados e o valid_p1 ou o valid_p2 ou o valid_p3 estão ativados para que possa passar o valor do “match” por uma porta AND e a máquina de estados receber um bit, ou seja, o match, se qualquer um dos comparadores sinalizar a comparação pela porta OR. Isso é possível visualizar pelas ondas valid_p1, valid_p2 e valid_p3, as quais estão em um bit somente após receberem o valor vão comparar.