TP4 – FUNDAMENTOS DE SISTEMAS DIGITAIS
40% DO TOTAL DA NOTA DO TRABALHO PRÁTICO

V1.1 - 14/junho/2019 Moraes

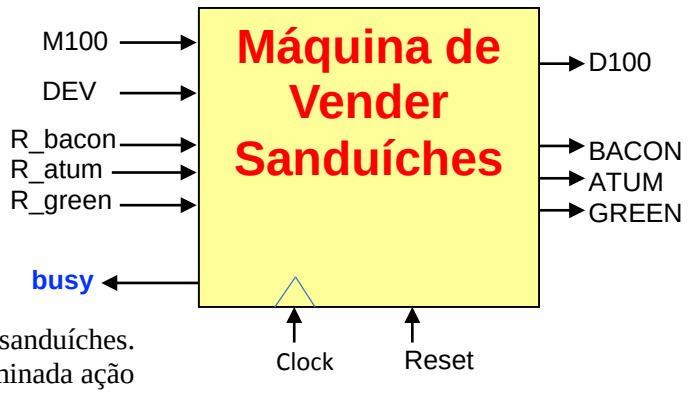
Especificação de uma Máquina de Vender Sanduíches

A máquina deve ser capaz de fornecer três tipos de sanduíches, denominados ATUM, BACON, GREEN. Estes estão disponíveis a partir de três teclas com o nome dos sanduíches. O preço de cada sanduíches é:

GREEN=R$ 2,00 ATUM= R$ 3,00 BACON=R$ 4,00

A interface da Máquina de Vender Sanduíches com o mundo externo é apresentada na figura abaixo.

- Entradas (7 sinais de 1 bit):
  - **M100**: informação relativa à inserção de uma moeda de 1 R$.
  - **DEV**: solicitação de devolução do saldo inserido.
  - **R_bacon, R_atum, R_green**: requisição de um tipo de sanduíche;
  - **clock**, **reset**: sinais de controle.
- Saídas (5 sinais de 1 bit):
  - **D100**: devolução de moeda
  - **BACON, ATUM, GREEN**: liberação dos sanduíches.
  - **busy**: aviso de tratamento de uma determinada ação realizada pelo usuário (*M100*, *DEV*, *R_bacon*, *R_atum*, *R_green*).



Na máquina existe uma fenda para inserir moedas, com um circuito capaz de reconhecer apenas moedas de R$ 1,00, e capaz de devolver qualquer outro tipo de moeda ou objeto não reconhecido. Além disso, o usuário pode desistir da transação e apertar a tecla **DEV** que devolve as moedas inseridas até o momento. Logo, esta máquina ao devolver moedas, irá devolver a soma correspondente ao número de moedas de R$ 1,00 inseridas, descontando o valor do sanduíche se for o caso.

Condições de devolução de todo o saldo (haverá um pulso em D100 a cada moeda devolvida):

- Usuário pressionar DEV;
- Condição de erro, a qual ocorre quando o usuário solicitar simultaneamente 2 ou mais tipos de sanduíche. Dica, usem o código abaixo indicar condição de erro:

```
erro <= (R_bacon and R_atum) or (R_bacon and R_green) or (R_atum and R_green);
```

- Solicitação de sanduíche com saldo insuficiente;
- Após a entrega de um sanduíche caso o saldo for maior que zero.

Sinal de sistema ocupado - *busy*: após selecionar-se uma das 5 entradas (*M100*, *DEV*, *R_bacon*, *R_atum*, *R_green*) o sinal *busy* sobe, impedindo uma nova entrada. Este processo de subir o *busy* é importante para evitar o acionamento de uma das entradas durante o processamento da entrada em curso. O sinal *busy* desce ao terminar o tratamento da ação solicitada.

Tarefa 1: modelar o comportamento deste circuito segundo uma máquina de estados finita (*FSM* – do inglês, finite state machine). Desenhe a máquina de estados que execute o comportamento especificado acima para a máquina de sanduíches. **Sugestão** de estados (podem ser outros, ou mais estados sejam necessários):

- *ACTION*: estado de inicialização. Fica-se neste estão aguardando-se o a seleção de uma das 5 entradas (*M100*, *DEV*, *R_bacon*, *R_atum*, *R_green*), ou a ocorrência de um erro. Este estado é o único em que a saída *busy* é igual a zero, permitindo a entrada de um novo comando.
- *SOMA*: estado que incrementa o valor inserido pelo usuário ao detectar-se uma inserção de moeda.
- *Sbacon*, *Satum*, *Sgreeen*: estados que ativam a devolução de um dado sanduíche quando a tecla relacionada ao sanduíche for pressionada e a soma armazenada for igual ou maior que o valor do

sanduíche. Nestes estados deve-se diminuir do saldo o valor do sanduíche.

- NULO: verifica se a soma acumulada chegou a zero para voltar ao estado ACTION.
- DEVOL: devolve uma moeda ativando o sinal D100. Este estado deve iterar com o estado NULO, até que o saldo zere.

Tarefa 2: Descrever a máquina de sanduíches em VHDL. **Importante**: a descrição da FSM deve seguir o modelo de dois processos e nenhuma saída deve ser ativada dentro da FSM. A FSM deve ser do tipo Moore.

Tarefa 3: Simule o circuito implementado utilizando incialmente o *testbench* fornecido, e depois modifique-o com uma sequência de eventos diferentes. Simular condições realistas, tais como tentativa de tirar sanduíche sem crédito suficiente, tentativa de solicitar devolução de moedas com crédito nulo, condição de erro, etc.

Entregar:

1. **Relatório: (a) desenho da máquina de estados, explicando seu comportamento; (b) simulação comentada. A simulação comentada deve conter pelo menos 15 eventos (deve aparecer condição de erro). A sequência de eventos deve ser diferente da fornecida pelo professor.**
2. **Códigos VHDL (circuito e *test bench*), wave.do, script de simulação.**

Test bench O *test bench* opera lendo um "programa" como o abaixo, que é ilustrado nas formas de onda:

```
constant program : input_data :=   --   M100  DEV  R_green R_atum R_bacon
    ( "10000",   -- M100
      "10000",   -- M100
      "10000",   -- M100
      "10000",   -- M100
      "10000",   -- M100
      "10000",   -- M100
      "10000",   -- M100    --   R$ 7 reais
      "00001",   -- R_bacon
      "10000",   -- M100
      "10000",   -- M100    --  R$ 2 reas
      "00011",   -- R_atum R_bacon  -- erro
      "10000",   -- M100
      "10000",   -- M100    --  R$ 2 reas
      "01000",   -- devolução
      "10000",   -- M100
      "10000",   -- M100
      "10000",   -- M100    --   R$ 3 reais
      "00010",   -- R_atum
      "10000",   -- M100
      "10000",   -- M100
      "10000",   -- M100    --   R$ 3 reais
      "00100",   -- R_green
      "10000",   -- M100
      "00001",   -- R_bacon   -- pede bacon sem saldo
      "00000"    -- nada
    );
```

É esta sequência de eventos que se deve modificar.

EXEMPLO DE SIMULAÇÃO – entregar no relatório uma figura semelhante a simulação abaixo (a simulação do programa acima continua com mais eventos, com um pedido sem saldo e a correspondente devolução). No final da simulação é impressa a mensagem: ***Failure: A SIMULACAO TERMINOU!*** **(710 ns)**