

Laboratório de Sistemas Digitais

Aula Teórico-Prática 8

Ano Letivo 2024/25

Modelação, simulação e síntese de
Máquinas de Estados Finitos

Aspetos gerais e
Modelo de *Moore*

Conteúdo

- Revisão sobre Máquinas de Estados Finitos (MEF ou *FSM*)
 - Estrutura-base; modelos de Mealy e **Moore**
- Revisão sobre Diagramas de Estado (DDE)
 - Especificação inicial → DDE → MEF
 - Tradução completa e **rigorosa**
 - Exemplos práticos
 - Métodos de construção
 - Manual
 - Com o editor do *Quartus Prime*
- Procedimentos de síntese e implementação
 - Descrição em VHDL de MEF
 - Estrutural (registo + blocos combinatórios)
 - Comportamental (tradução do DDE): abordagem com 2 processos (Moore)
 - Codificação (e síntese) automática a partir do DDE
 - Simulação de MEF

Síntese de FSMs (unidades de controlo e circuitos sequenciais em geral)

- Passagem da especificação à implementação
 - Conceção de uma solução para um problema concreto (muitas vezes) inicialmente descrito em linguagem natural ou na forma de “use-cases”
 - Formalização/abstracção segundo o modelo de Máquina de Estados Finitos (MEF/FSM)
 - Processo composto por diferentes passos de modelação, optimização e geração de hardware
 - Solução frequentemente
 - Não única
 - Sub-ótima

Instrumentos/Ferramentas de Modelação e Síntese de FSMs

- **Diagramas de Estado**
- Cartas ASM: *Algorithmic State Machines*
- Tabelas de Estado/Saídas
- Tabelas de Transição/Saídas
- Tabelas de Excitação
- Diagramas temporais
- **Linguagens de Descrição de Hardware**
 - VHDL, Verilog, ...
- ...

Especificação inicial → MEF

– Problema concreto

- Descrição inicial (frequentemente) incompleta/imprecisa
 - linguagem natural
 - “use-cases”

– Formalização / abstracção

- Especificação (forçosamente) completa/rigorosa

Diagramas
de Estados

– Modelação

- Manual
 - Automática
- } em VHDL

– Simulação (testbench)

- Revisão / correcção / optimização

– Implementação (FPGA)

- Verificação
- Solução pode ser não única / sub-ótima

Formalização (esp. inicial → DDE)

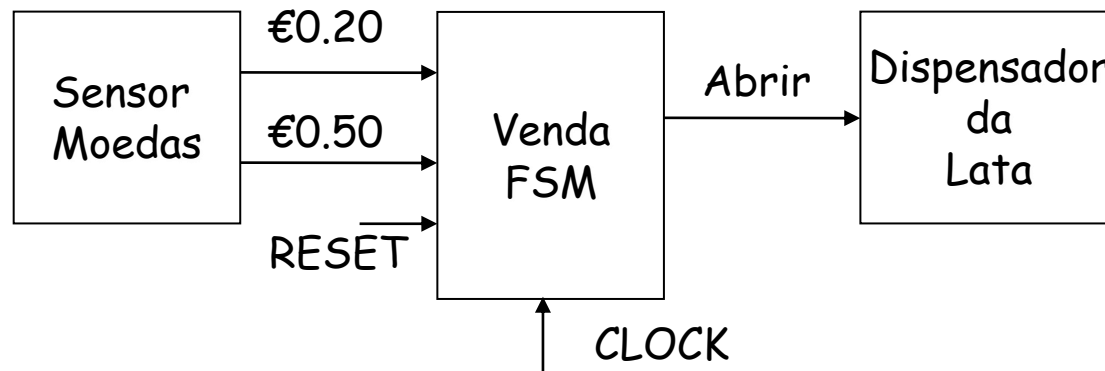
Problema principal (aspecto mais importante)

- Esclarecer ao máximo a especificação inicial
- Identificar as entradas e as saídas (E/S)
- Identificar o comportamento E/S
- Definir estados
 - Estabelecer o significado de cada estado
- Analisar “use cases”

Seguem-se exemplos...

Exemplo 1 - Especificação

- Máquina de venda de bebidas
 - Requisitos gerais:
 - entrega lata de cerveja (sem álcool 😊) após depósito de € 0.60
 - uma única entrada para moedas de € 0.20 e € 0.50
 - a máquina não dá troco
 - Passo 1: perceber o problema (fazer um desenho / diagrama de blocos!...)

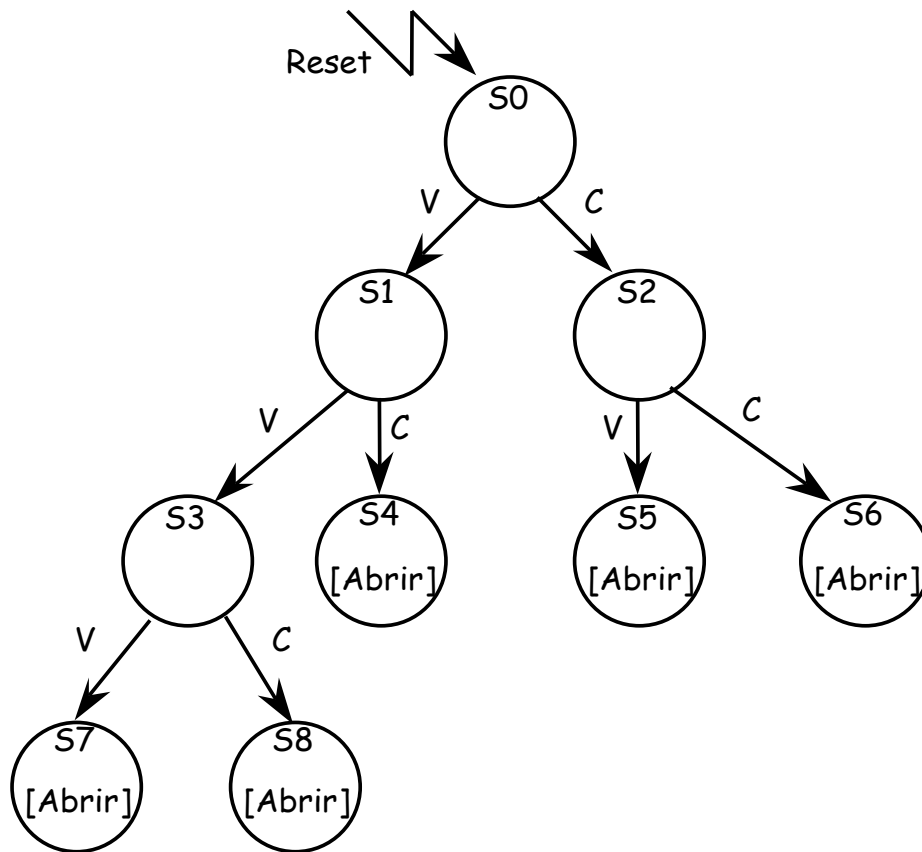


Exemplo 1 – Análise de Requisitos

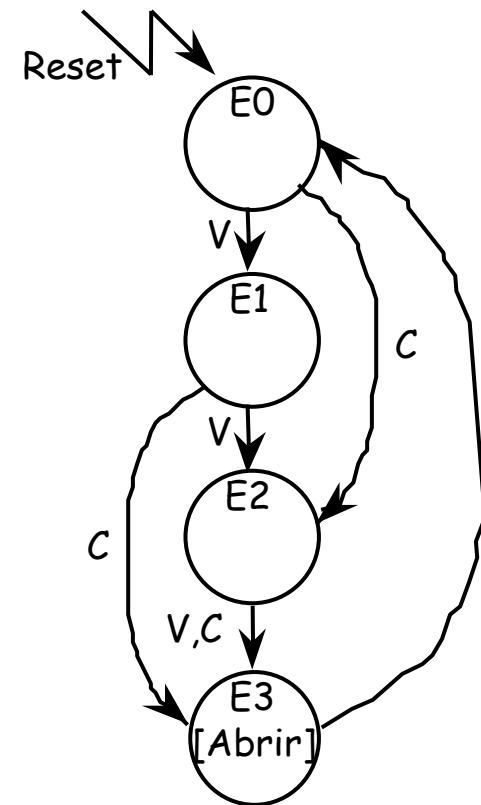
- Começar por identificar as sequências de entradas que levam diretamente à abertura
 - 3 moedas de €0.20
 - 1 moeda de €0.20 + 1 moeda de €0.50
 - 1 moeda de €0.50 + 1 moeda de €0.20
 - 2 moedas de €0.50
 - 2 moedas de €0.20 + 1 moeda de €0.50
- Identificar entradas e saídas:
 - Entradas:
 - V (sensor ativo para €0.20)
 - C (sensor ativo para €0.50)
 - Saída
 - Abrir

Exemplo 1 – Diagrama de Estados

- Diagrama de estados primário (inicial)

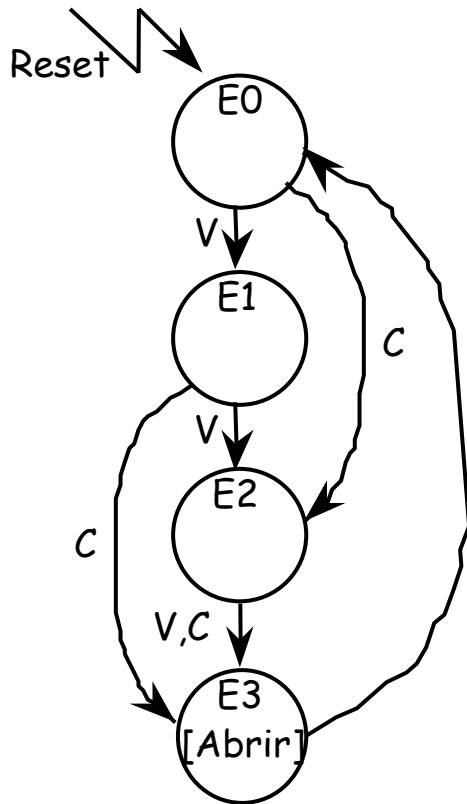


- Diagrama de estados correto com reutilização de estados

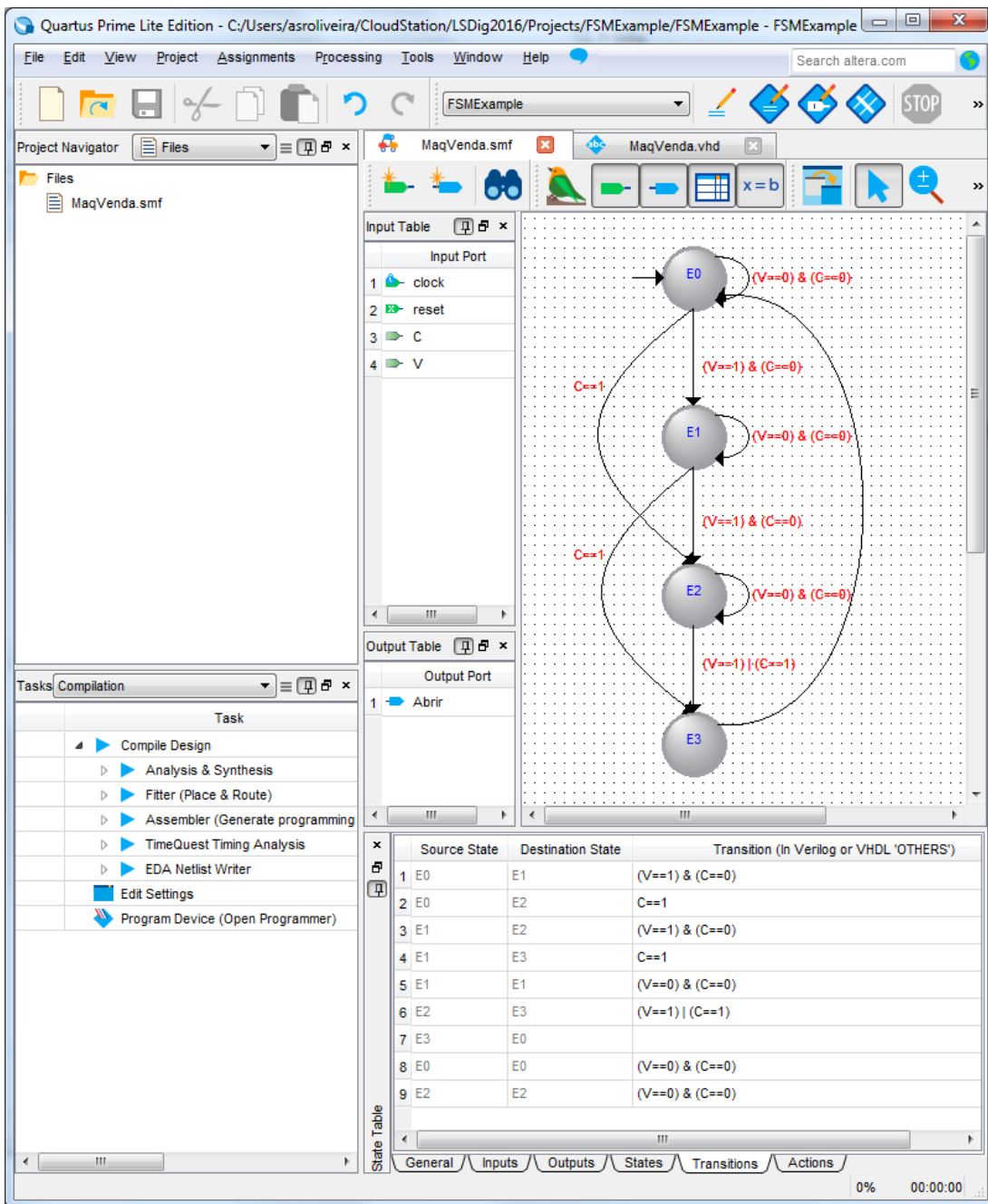


Exemplo 1 – Tabela de Estados

- Tabela de Estados/Saídas decorre diretamente do DE



PState	Inputs		NState	Output Abrir
	V	C		
E0	0	0	E0	0
	0	1	E2	0
	1	0	E1	0
	1	1	X	X
E1	0	0	E1	0
	0	1	E3	0
	1	0	E2	0
	1	1	X	X
E2	0	0	E2	0
	0	1	E3	0
	1	0	E3	0
	1	1	X	X
E3	X	X	E0	1



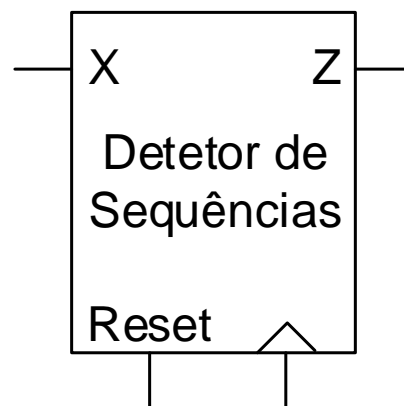
Modelação com o Editor de FSMs do Quartus Prime

- As **condições** das transições têm de ser mutuamente exclusivas (boa prática mesmo nos diagramas construídos com “papel e lápis”)
- As **condições não especificadas** correspondem à **manutenção do estado**
- Geração de VHDL a partir do ficheiro SMF

TPC: analise o código VHDL gerado automaticamente (disponibilizado no site)

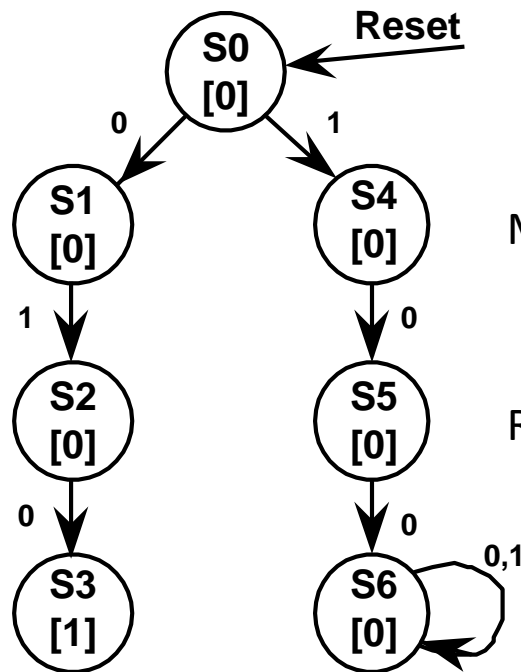
Exemplo 2: Detector de Sequências

- Reconhecimento de padrões em frases de comprimento finito. Exemplo:
 - Um reconhecedor de frases finitas tem uma entrada (X) e uma saída (Z). A saída é ativada sempre que a sequência de entrada ...010...é observada, desde que a sequência 100 nunca tenha surgido.
 - Exemplo do comportamento entrada/saída:
 - X: 011010000010...
 - Z: 00000100000...
 - X: 00101010010...
 - Z: 00010101000...



Detector de Sequências: Diagrama de Estados

- Desenhar o diagrama de estados para os padrões que devem ser reconhecidos i.e., 010 e 100.

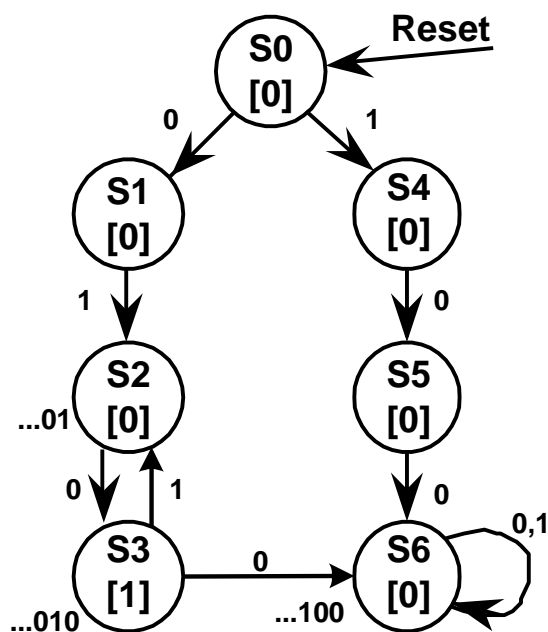


Modelo de Moore

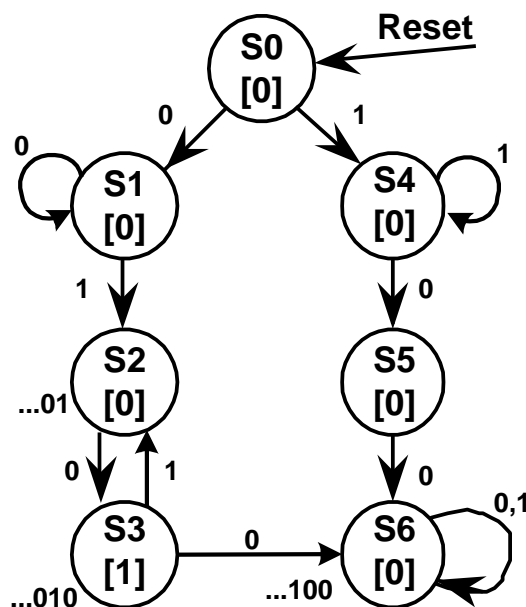
Reset “leva” a máquina para o estado S0

Detector de Sequências: Diagrama de Estados

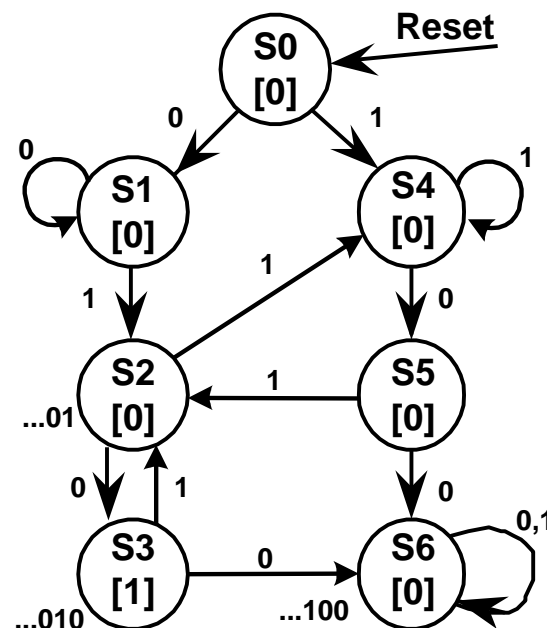
- Completar o diagrama analisando as condições de transição de cada estado



Transições em S3



Transições em S1 e S4



Transições em S2 e S5

Detector de Sequências

Revisão do procedimento

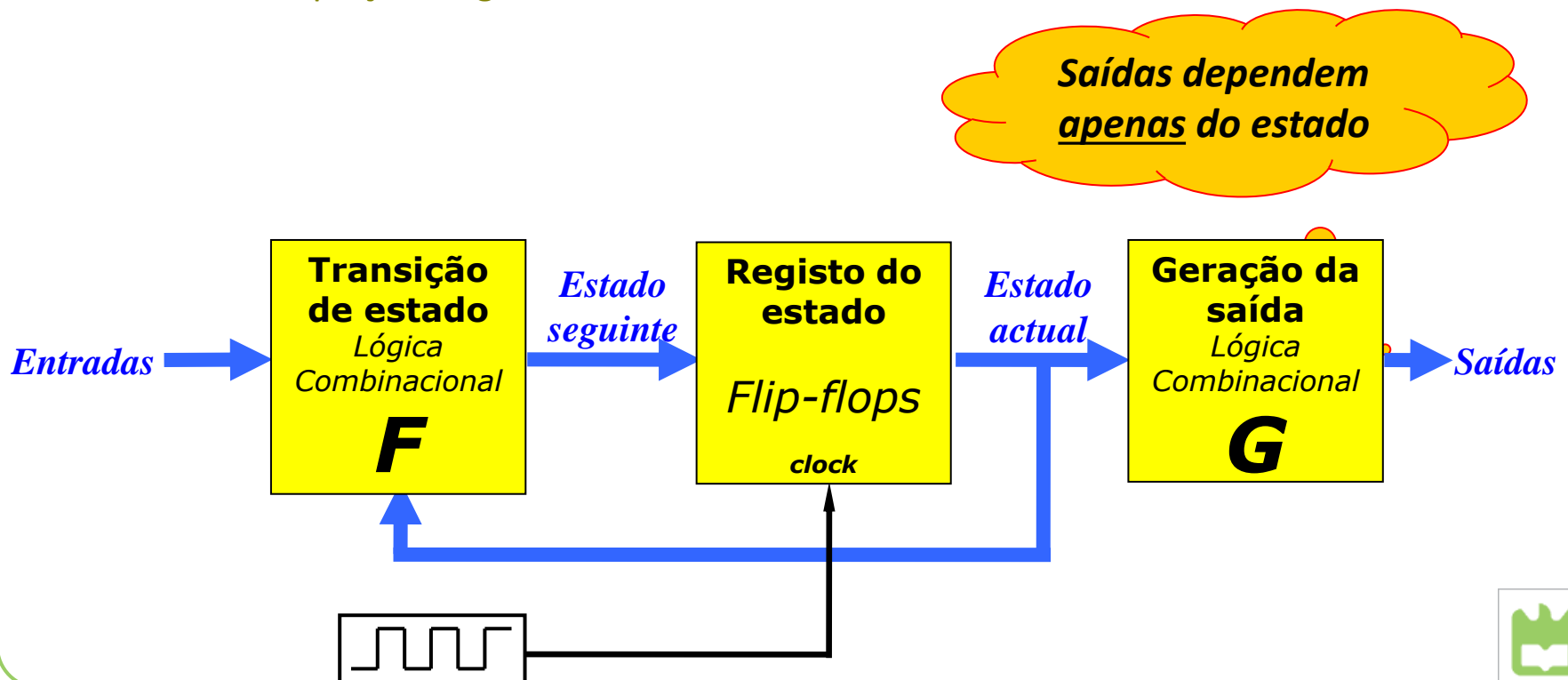
1. Escrever sequências de teste com as entradas/saídas para perceber a especificação
2. Criar uma sequência de estados e transições para as sequências que se pretende ver reconhecidas
3. Acrescentar transições em falta; reutilizar o mais possível os estados existentes
4. Verificar o comportamento E/S do diagrama de estados para assegurar que funciona como pretendido

Agora que vimos alguns exemplos de passagem da especificação para o diagrama de estados, vamos ver como descrever eficientemente uma FSM em VHDL...

Modelação de MEF em VHDL

Abordagem estrutural

- Registo
 - Codificação de estados
- Bloco combinacional de transição de estado
 - Equações lógicas
- Bloco combinacional de geração das saídas
 - Equações lógicas

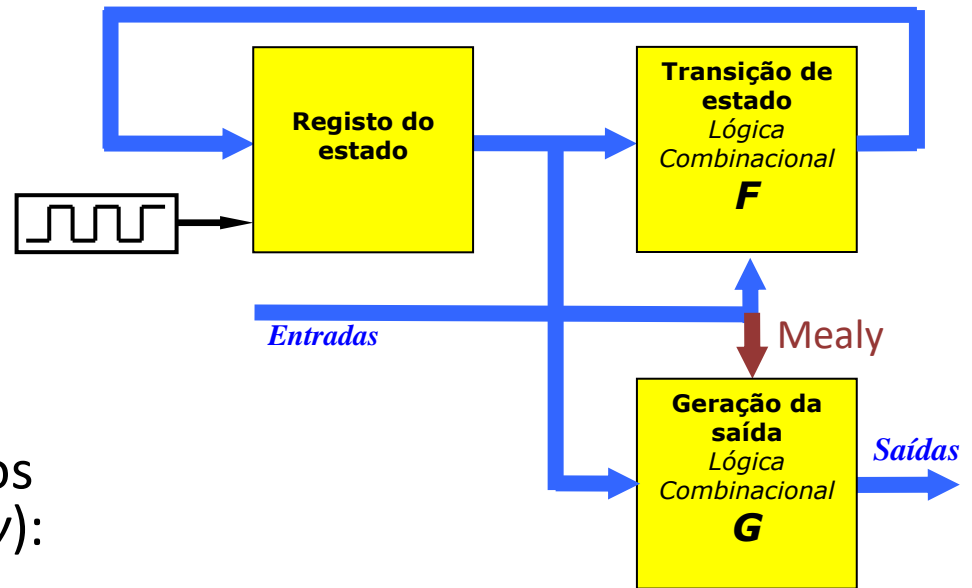


Modelos Comportamentais Textuais de FSMs

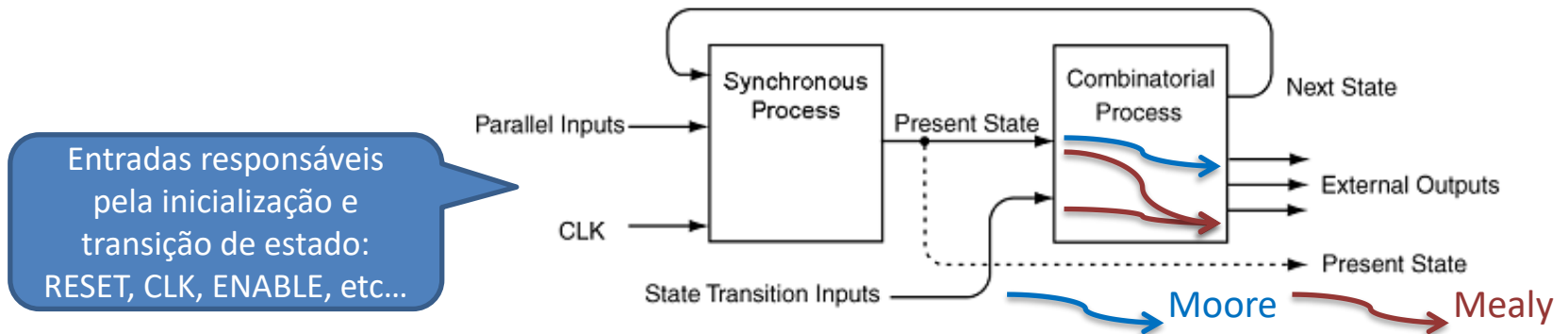
- A linguagem VHDL proporciona descrições de alto nível (comportamentais) de FSMs
 - Muito próximas dos diagrama de estados
 - Vários estilos de escrita possíveis em VHDL (mais frequentemente com 2 ou 3 processos)
 - Tradução direta do diagrama de estados para VHDL no editor de texto do IDE (Quartus Prime em LSD)
 - Por omissão é a ferramenta de síntese (compilador) que determina a codificação dos estados com base em estados simbólicos
 - Não são necessárias equações lógicas para explicitar saídas e o “próximo estado” (equações de excitação)
 - Minimização lógica realizada pelo compilador (ferramenta de síntese)

MEF em VHDL comportamental: método '2 processos'

Modelo de MEF revisitado:

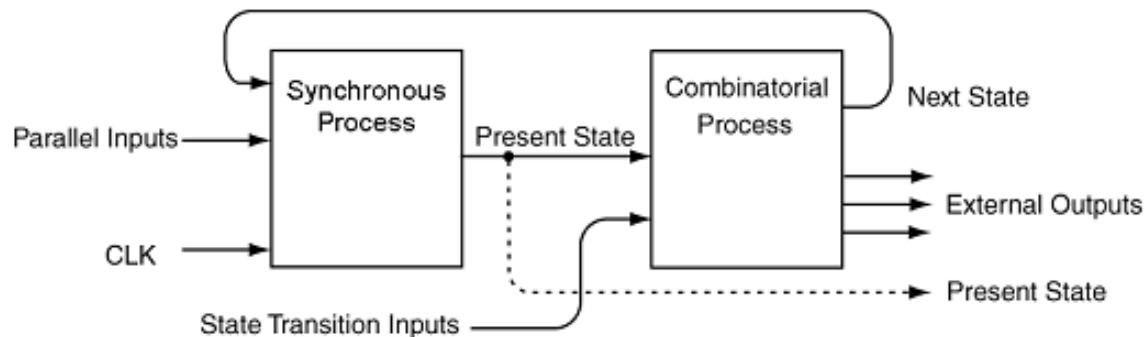


Reorganização em dois processos
(compatível com *Moore* e *Mealy*):



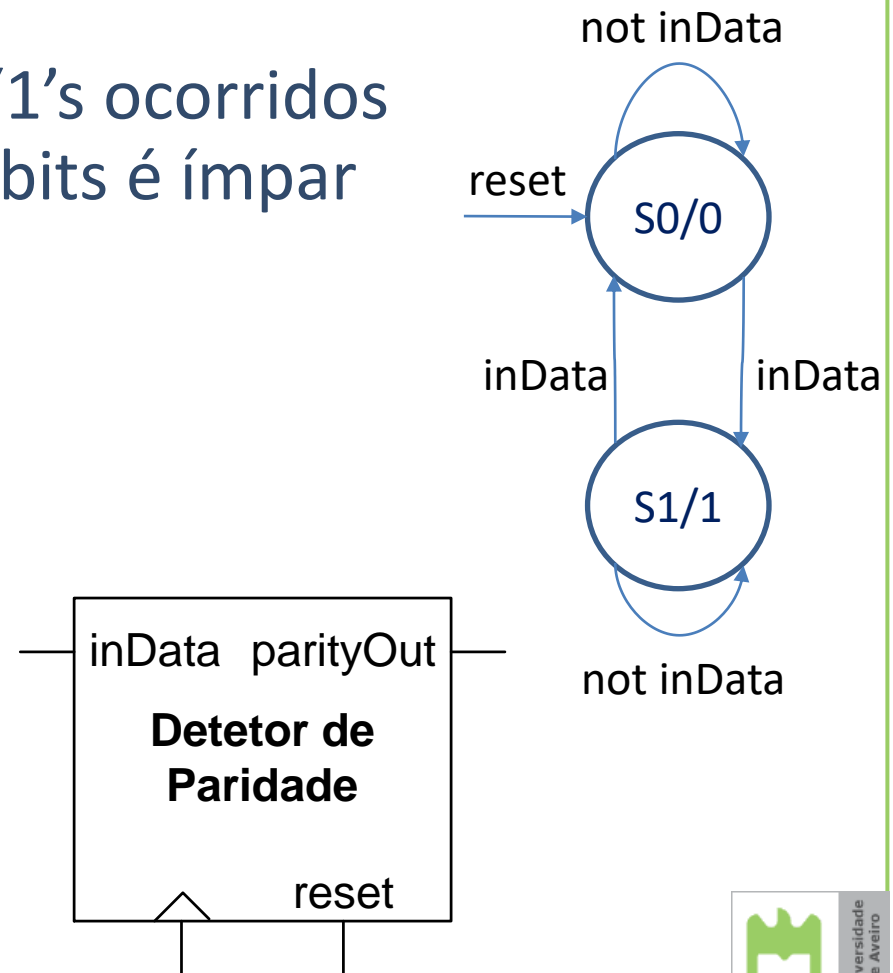
FSMs em VHDL - O Estilo de Codificação Baseado em “2” Processos

- Processo “*synchronous*” (apenas uma designação)
 - Atribuições dependentes dum evento de relógio e/ou de atribuição/inicialização assíncrona dos elementos de memória
- Processo “*combinatorial*” (apenas uma designação)
 - Atribuições relacionadas com a determinação de
 - Saídas
 - Estado seguinte
- Os 2 processos são interdependentes



Exemplo Trivial (segundo *Moore*) – Detetor de Paridade

- Detetor de paridade
 - Deteta se o número de ‘1’s ocorridos numa *stream* (série) de bits é ímpar (ou par)
 - Entradas
 - **clk**
 - **reset** (síncrono)
 - **inData**
 - Saída
 - **parityOut**



Exemplo Trivial (segundo *Moore*) – Detetor de Paridade

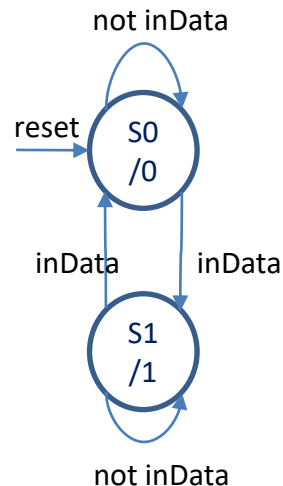
```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity ParityDetector is
    port(reset      : in  std_logic;
         clk        : in  std_logic;
         inData     : in  std_logic;
         parityOut  : out std_logic);
end ParityDetector;
```

```
architecture Behavioral of ParityDetector is
```

```
    type TState is (S0,S1);
    signal pState, nState: TState;
```

```
begin
    sync_proc : process(clk)
    begin
        if (rising_edge(clk)) then
            if (reset = '1') then
                pState <= S0;
            else
                pState <= nState;
            end if;
        end if;
    end process;
```

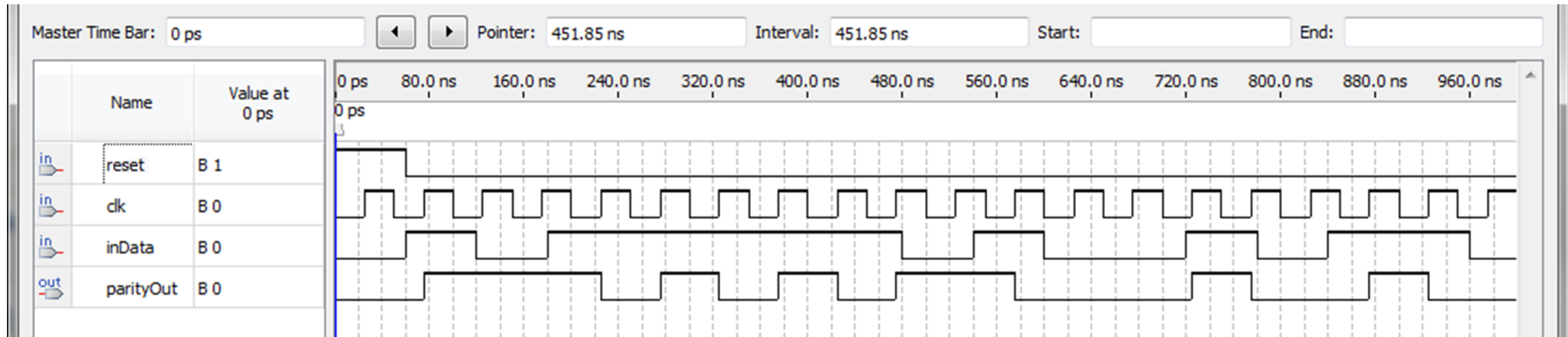


Criação dum
novo tipo de
dados
“enumerado”

```
    comb_proc : process(pState, inData)
    begin
        case pState is
            when S0 =>
                parityOut <= '0'; -- Moore output
                if (inData = '1') then
                    nState <= S1;
                else
                    nState <= S0;
                end if;
            when S1 =>
                parityOut <= '1'; -- Moore output
                if (inData = '1') then
                    nState <= S0;
                else
                    nState <= S1;
                end if;
            when others => -- “Catch all” condition
                nState <= S0;
                parityOut <= '0';
        end case;
    end process;
end Behavioral;
```

Tradução direta do
Diagrama de
Estados

Simulação de FSMs – Exemplo com Detetor de Paridade

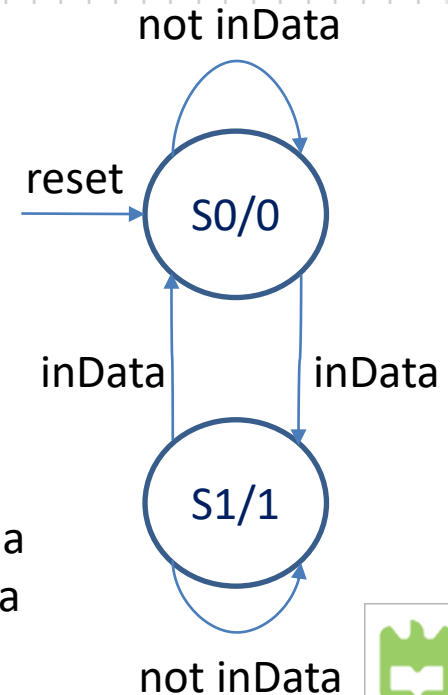


Simulação com uma *testbench* onde é instanciada a UUT (FSM)

- Testbench gerada a partir do ficheiro VWF
- Testbench gerada diretamente em VHDL de acordo com o *template* para componentes sequenciais
 - 1 processo para geração do sinal de relógio
 - 1 processo para inicialização e geração da entrada

- **Muito importante:**

Não comutar as entradas (variáveis independentes e reset) na vizinhança das transições ativas do sinal de relógio (refletir na simulação o cumprimento dos tempos de *setup* e de *hold*)



Máquina de Vendas

```
entity DrinksFSM is
    port(reset : in  std_logic;
          clk   : in  std_logic;
          v     : in  std_logic;
          c     : in  std_logic;
          abrir  : out std_logic);
end DrinksFSM;
```

architecture Behavioral of DrinksFSM is

```
type TState is (E0, E1, E2, E3);
signal pState, nState : TState;
```

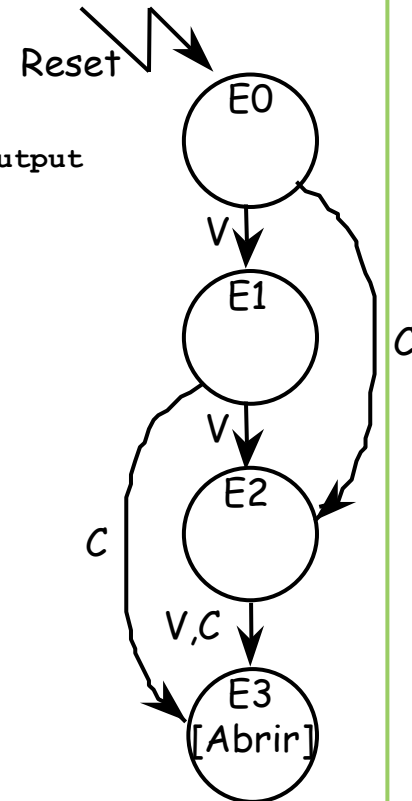
```
begin
sync_proc : process(clk)
begin
if (rising_edge(clk)) then
    if (reset = '1') then
        pState <= E0;
    else
        pState <= nState;
    end if;
end if;
end process;
```

```
comb_proc : process(pState, v, c)
begin
```

```
    case (pState) is
```

```
when E0 =>
    abrir <= '0'; --Moore Output
    if (v = '1') then
        nState <= E1;
    elsif (c = '1') then
        nState <= E2;
    else
        nState <= E0;
    end if;
```

```
when E1 =>
    abrir <= '0';
    if (v = '1') then
        nState <= E2;
    elsif (c = '1') then
        nState <= E3;
    else
        nState <= E1;
    end if;
end if;
```



Tradução direta do
Diagrama de
Estados

Máquina de Vendas

```
when E2 =>
  abrir <= '0';

  if (v = '1') or (c = '1') then
    nState <= E3;
  else
    nState <= E2;
  end if;

when E3 =>

  drink <= '1';

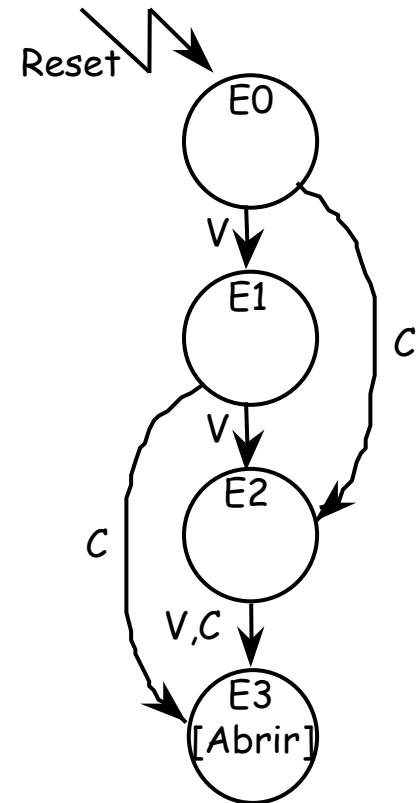
  nState <= E0;

when others =>
  abrir <= '0';
  report "Reach undefined state";

end case;

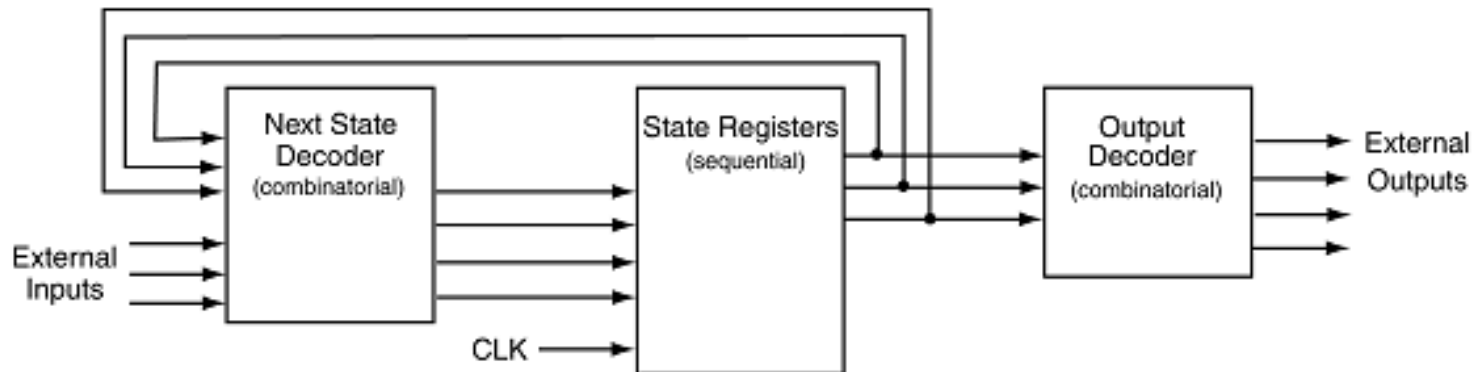
end process;

end Behavioral;
```



Síntese de FSMs

- Realizada pelo compilador / ferramenta de síntese
 - Codificação de estados (com base nos estados simbólicos)
 - Geração do registo de estado (com o número de bits necessários em função da codificação de estados)
 - Determinação e otimização dos circuitos combinatórios de estado seguinte e das saídas

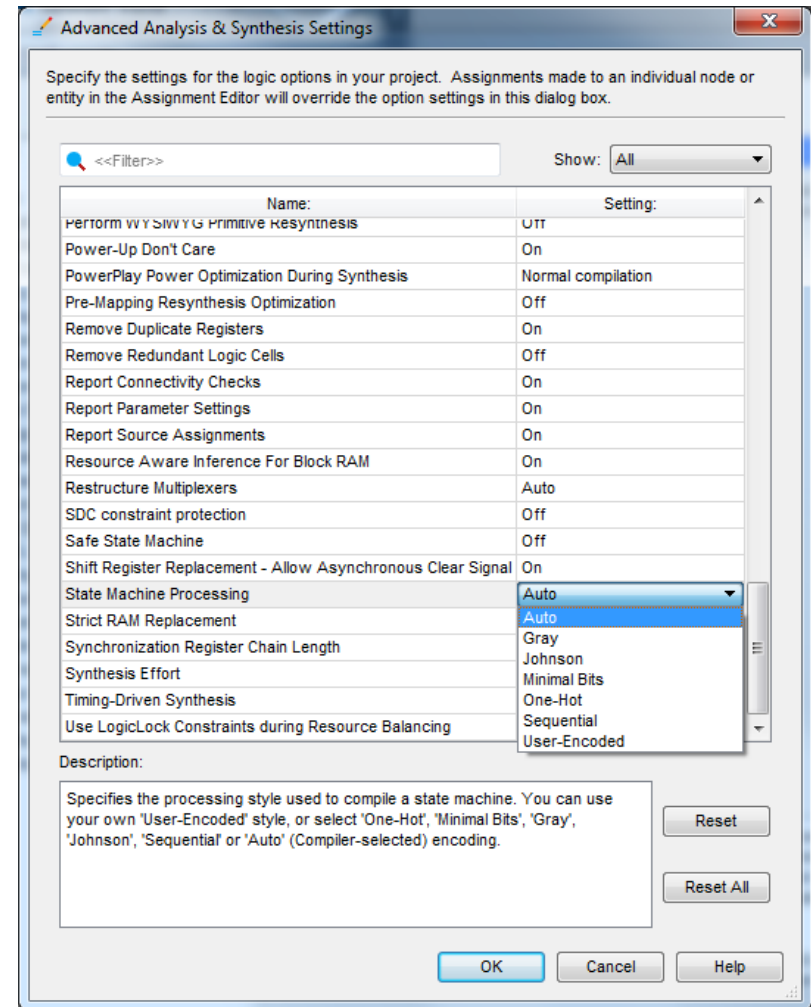


Codificação dos Estados

- Em VHDL o recurso a um tipo enumerado para codificar simbolicamente os estados
 - Permite uma descrição de alto-nível muito próxima do diagrama de estados
 - Confere ao sintetizador a tarefa da codificação binária dos estados
 - O hardware sintetizado inclui frequente mais *flip-flops* que o nº mínimo (quando a codificação é binária ou *gray*)
 - Codificação frequente – “*One-hot*”
 - Exemplo: 4 estados => 4 flip-flops (000**1**, 001**0**, 01**00**, 10**00**)
 - Adequada à implementação em FPGA (elevado número de *flip-flops* disponíveis mas LUTS com poucas entradas - tipicamente 4 a 6)
 - » Lógica de estado seguinte e de saída tendencialmente mais simples (com menos níveis e mais rápida)

Codificação dos Estados

- No “Quartus Prime” é também possível “forçar” a técnica de codificação de estado em “Assignments → Settings → Compiler Settings → Advanced Settings (Synthesis)”



Comentários Finais

- No final desta aula e do trabalho prático 8 de LSD, deverá ser capaz de:
 - Reconhecer o modelo de FSM como uma ferramenta adequada para formalizar e projetar sistemas sequenciais
 - Construir diagramas de estados com base na especificação de uma FSM
 - Conhecer os passos necessários à síntese de máquinas de estados finitos
 - Usar descrições comportamentais em VHDL próximas do diagrama de estados e de saídas
 - *Modelo de Moore*
 - Conceber *testbenches* para a simulação funcional das FSMs
 - Sintetizar, implementar em FPGA e testar FSMs
- ... bom trabalho prático 8, disponível no site da UC
 - elearning.ua.pt