DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Diseño de Sistemas Digitales con FPGA

Automátas como herramienta de modelado y de control

Primer Cuatrimestre 2023

Diseño de Sistemas Digitales con FPGA
DC - UBA

Introducción

La clase de hoy y la del jueves próximo van a involucrar formalismos, herramientas y métodos que nos permiten razonar sobre el **comportamiento de un sistema a través del tiempo**.

Estructura de las clases

En la clase hoy vamos a ver:

- Estado, trazas y valuaciones.

Estructura de las clases

En la clase hoy vamos a ver:

- Estado, trazas y valuaciones.
- Máquinas de estado, **Moore y Mealy**.

Estructura de las clases

En la clase hoy vamos a ver:

- Estado, trazas y valuaciones.
- Máquinas de estado, **Moore y Mealy**.
- ¿Circuitos secuenciales o controladores?

Estructura de las clases

En la clase hoy vamos a ver:

- Estado, trazas y valuaciones.
- Máquinas de estado, **Moore y Mealy**.
- ¿Circuitos secuenciales o controladores?
- Implementación de un controlador con estados en `VHDL`.

Estructura de las clases

En la clase que viene veremos:

- Model checking.

Estructura de las clases

En la clase que viene veremos:

- Model checking.
- Lógicas temporales y μ-cálculo.

Estructura de las clases

En la clase que viene veremos:

- Model checking.
- Lógicas temporales y μ-cálculo.
- Sistemas reactivos.

Estructura de las clases

En la clase que viene veremos:

- Model checking.
- Lógicas temporales y μ-cálculo.
- Sistemas reactivos.
- Assertion based verification (`SVA`, `PSL`, `SystemVerilog`).

Estructura de las clases

En la clase que viene veremos:

- Model checking.
- Lógicas temporales y μ-cálculo.
- Sistemas reactivos.
- Assertion based verification (`SVA`, `PSL`, `SystemVerilog`).
- Un ejemplo de verificación con PSL.

Estructura de las clases

En la clase que viene veremos:

- Model checking.
- Lógicas temporales y μ-cálculo.
- Sistemas reactivos.
- Assertion based verification (`SVA`, `PSL`, `SystemVerilog`).
- Un ejemplo de verificación con PSL.
- Verificación y síntesis de especificaciones.

Definiciones

Estado

¿A qué llamamos estado?

- ¿Es el estado del circuito o del sistema?

Estado

¿A qué llamamos estado?

- ¿Es el estado del circuito o del sistema?
- ¿Se refiere sólo a los componentes de memoria o a las señales (impermanentes)?

Estado

¿A qué llamamos estado?

- ¿Es el estado del circuito o del sistema?
- ¿Se refiere sólo a los componentes de memoria o a las señales (impermanentes)?
- ¿Si son ambas, cuándo cambia el estado? ¿Cuando cambia el valor de la señal, de la memoria o de ambas?

Estado

¿A qué llamamos estado?

- ¿Es el estado del circuito o del sistema?
- ¿Se refiere sólo a los componentes de memoria o a las señales (impermanentes)?
- ¿Si son ambas, cuándo cambia el estado? ¿Cuando cambia el valor de la señal, de la memoria o de ambas?
- ¿Tenemos un estado distinto para cada cambio de flanco?

Estado

¿A qué llamamos estado?

- ¿Es el estado del circuito o del sistema?
- ¿Se refiere sólo a los componentes de memoria o a las señales (impermanentes)?
- ¿Si son ambas, cuándo cambia el estado? ¿Cuando cambia el valor de la señal, de la memoria o de ambas?
- ¿Tenemos un estado distinto para cada cambio de flanco?
- ¿Los cambios son de valores discretos o contínuos?

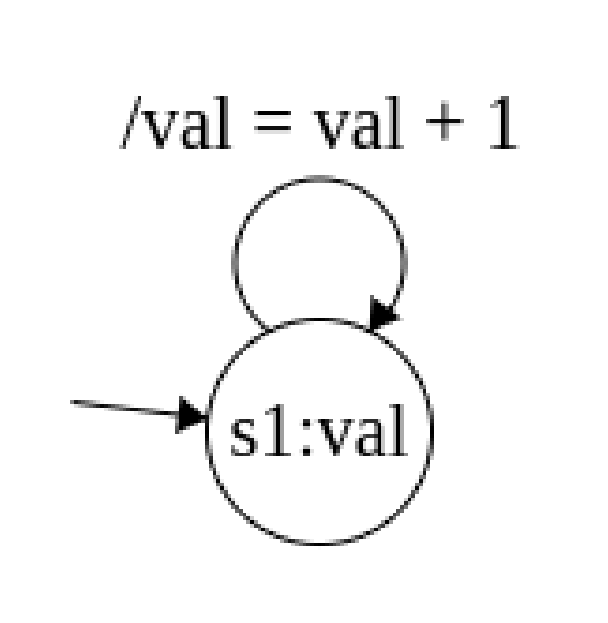
Estado

Las respuestas van a depender del modelo que utilicemos y del uso que le queramos dar, pero definen categorías a observar de nuestros modelos.

Ejemplo

¿Cómo modelaríamos con una máquina de estado un circuito que incrementa el valor de un registro interno en cada pulso de reloj?

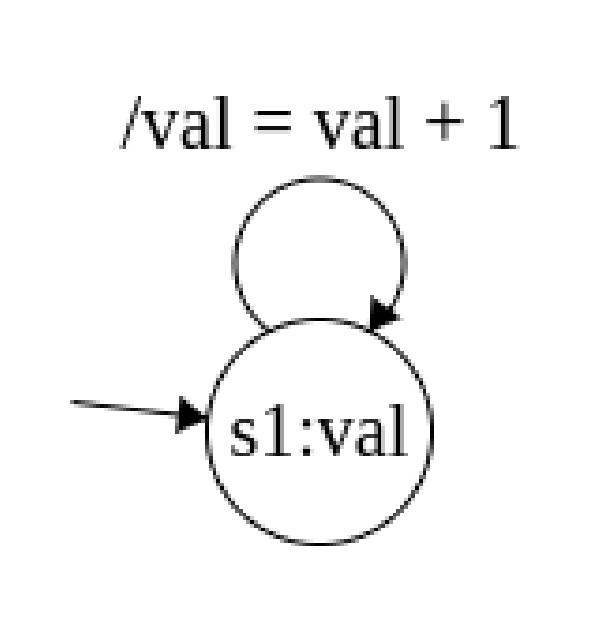
Ejemplo



Observando este modelo:

- ¿Qué tipo de ejecuciones permite?

Ejemplo



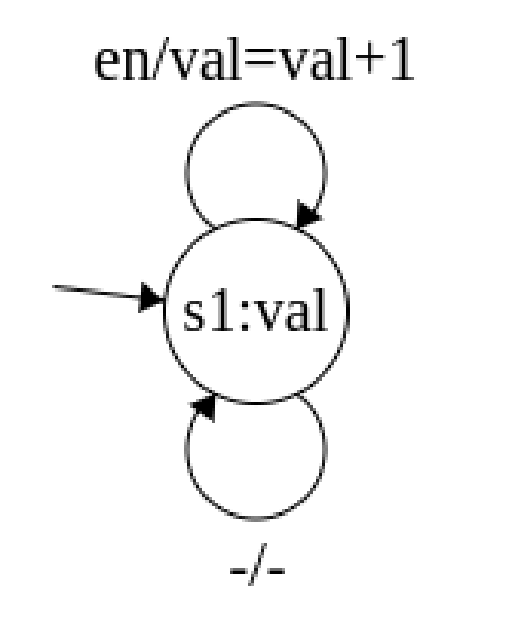
Observando este modelo:

- ¿Qué tipo de ejecuciones permite?
- ¿Hay señales que no hayamos tenido en cuenta?

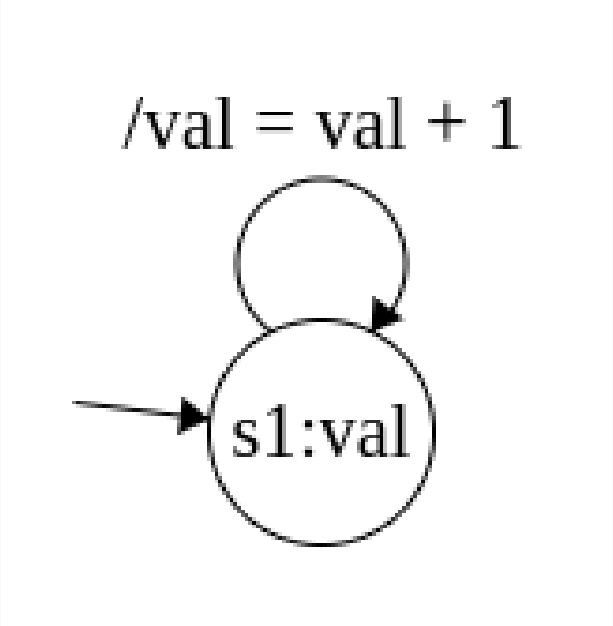
Ejemplo

¿Cómo modelaríamos con una máquina de estado un circuito que incrementa el valor de un registro interno en cada pulso de reloj solamente cuando la señal `en` está activada?

Ejemplo



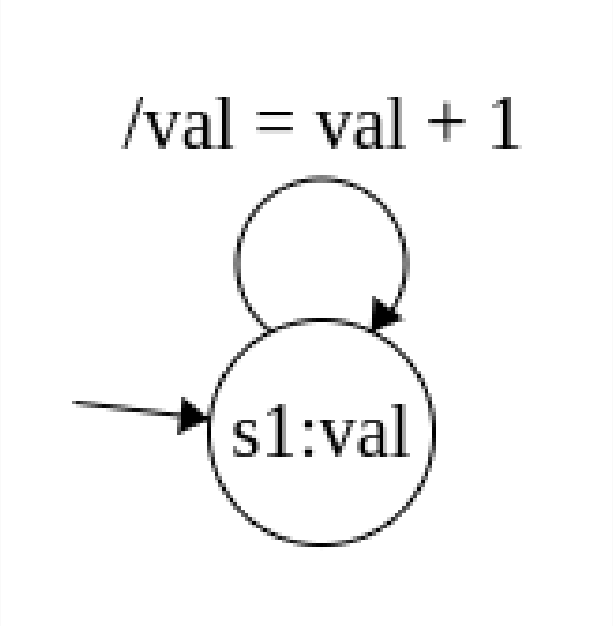
Ejemplo



Observando este modelo:

- ¿Qué tipo de ejecuciones permite?

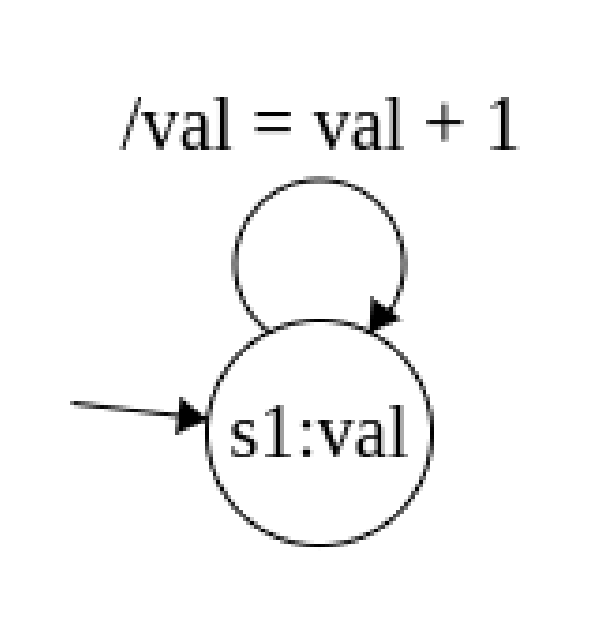
Ejemplo



Observando este modelo:

- ¿Qué tipo de ejecuciones permite?
- ¿Hay señales que no hayamos tenido en cuenta?

Ejemplo



Ejecuciones del modelo (a esto llamaremos trazas):

$(val : 0, en : 0), (val : 0, en : 0), (val : 1, en : 1),$
$(val : 2, en : 1), (val : 2, en : 0), (val : 3, en : 1) \ldots$

Máquinas de estado, Moore y Mealy

Controlador de memoria

Vamos a comenzar la sección sobre máquinas de estado con un ejemplo. Supongamos que queremos diseñar un **controlador** de memoria para un procesador. El controlador va a recibir tres posibles comandos (señales de control de entrada):

- *mem*: que indica en el valor alto que queremos operar con la memoria.

Controlador de memoria

Vamos a comenzar la sección sobre máquinas de estado con un ejemplo. Supongamos que queremos diseñar un **controlador** de memoria para un procesador. El controlador va a recibir tres posibles comandos (señales de control de entrada):

- *mem*: que indica en el valor alto que queremos operar con la memoria.
- *rw*: que indica con el valor alto indica que queremos escribir y con el bajo que queremos leer.

Controlador de memoria

Vamos a comenzar la sección sobre máquinas de estado con un ejemplo. Supongamos que queremos diseñar un **controlador** de memoria para un procesador. El controlador va a recibir tres posibles comandos (señales de control de entrada):

- *mem*: que indica en el valor alto que queremos operar con la memoria.
- *rw*: que indica con el valor alto indica que queremos escribir y con el bajo que queremos leer.
- *burst*: que indica con el valor alto si queremos realizar cuatro lecturas consecutivas.

Controlador de memoria

El chip de memoria tiene dos señales de control a su entrada:

- *oe*: que indica en el valor alto que queremos leer un dato.

Controlador de memoria

El chip de memoria tiene dos señales de control a su entrada:

- *oe*: que indica en el valor alto que queremos leer un dato.
- *we*: que indica con el valor alto que queremos escribir un dato.

Controlador de memoria

El chip de memoria tiene dos señales de control a su entrada:

- *oe*: que indica en el valor alto que queremos leer un dato.
- *we*: que indica con el valor alto que queremos escribir un dato.

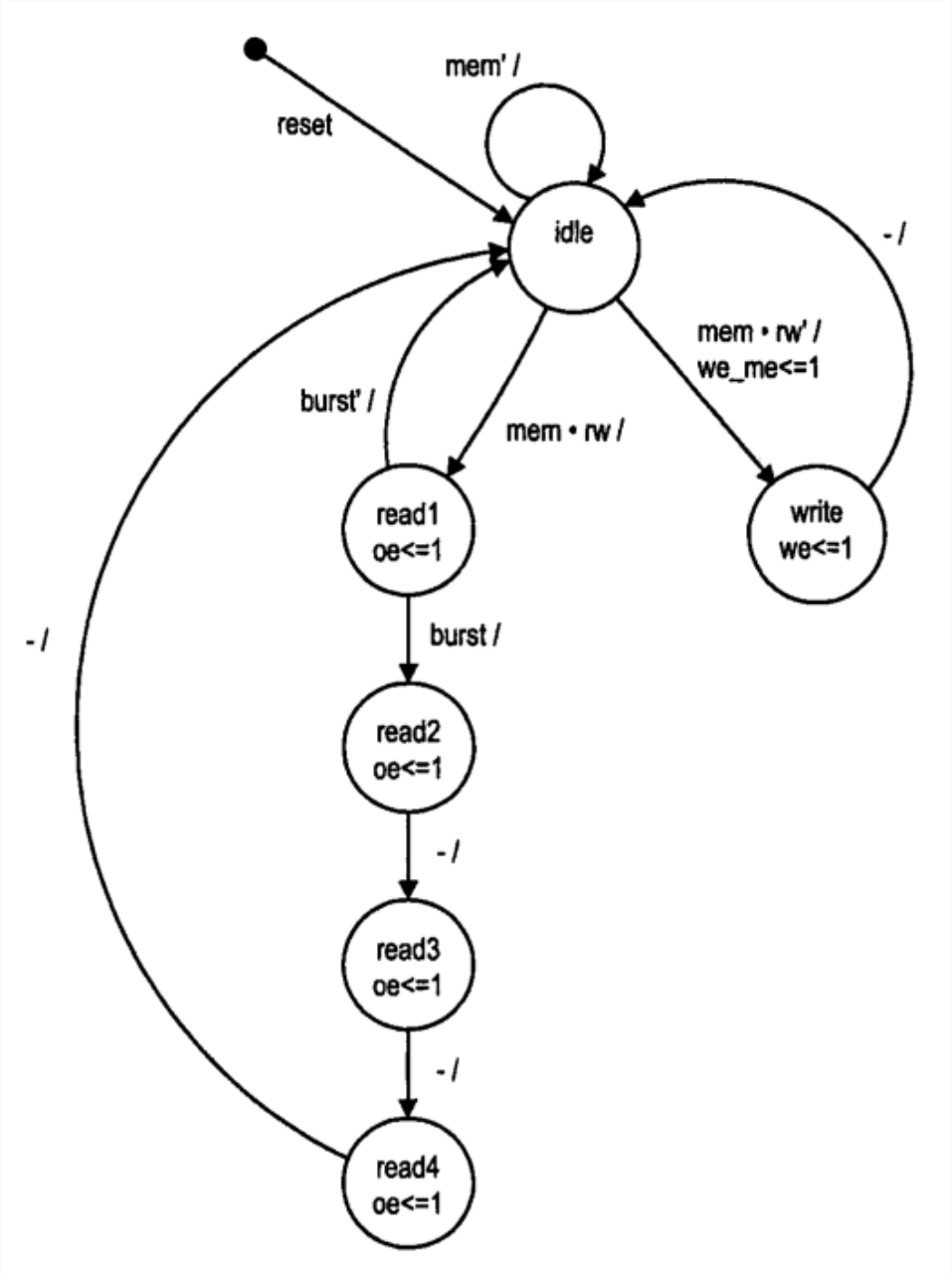
¿Cómo sería una máquina de estados de este controlador?

Controlador de memoria

- $\leftarrow$ *mem*: que indica en el valor alto que queremos operar con la memoria.
- $\leftarrow$ *rw*: que indica con el valor alto indica que queremos escribir y con el bajo que queremos leer.
- $\leftarrow$ *burst*: que indica con el valor alto si queremos realizar cuatro lecturas consecutivas.
- $\rightarrow$ *oe*: que indica en el valor alto que queremos leer un dato.
- $\rightarrow$ *we*: que indica con el valor alto que queremos escribir un dato.

¿Qué ponemos en cada estado? ¿Qué representa una transición?

Controlador de memoria



Formalismos

Veamos un poco los formalismos de dos tipos de máquinas de estado, empezando con Moore.

Máquina de Moore

Una máquina de Moore define sus salidas solamente en base a su estado actual y se define como una tupla:

$$M = (Q, q_0, \Sigma, O, \delta, G)$$

Donde:

- Q es un conjunto de estados.

Máquina de Moore

Una máquina de Moore define sus salidas solamente en base a su estado actual y se define como una tupla:

$$M = (Q, q_0, \Sigma, O, \delta, G)$$

Donde:

- Q es un conjunto de estados.
- $q_0 \in Q$ es el estado inicial.

Máquina de Moore

Una máquina de Moore define sus salidas solamente en base a su estado actual y se define como una tupla:

$$M = (Q, q_0, \Sigma, O, \delta, G)$$

Donde:

- Q es un conjunto de estados.
- $q_0 \in Q$ es el estado inicial.
- Σ es el alfabeto de entrada.

Máquina de Moore

Una máquina de Moore define sus salidas solamente en base a su estado actual y se define como una tupla:

$$M = (Q, q_0, \Sigma, O, \delta, G)$$

Donde:

- Q es un conjunto de estados.
- $q_0 \in Q$ es el estado inicial.
- Σ es el alfabeto de entrada.
- O es el alfabeto de salida.

Máquina de Moore

Una máquina de Moore define sus salidas solamente en base a su estado actual y se define como una tupla:

$$M = (Q, q_0, \Sigma, O, \delta, G)$$

Donde:

- Q es un conjunto de estados.
- $q_0 \in Q$ es el estado inicial.
- Σ es el alfabeto de entrada.
- O es el alfabeto de salida.
- $\delta : Q \times \Sigma \to Q$ es la función de transición.

Máquina de Moore

Una máquina de Moore define sus salidas solamente en base a su estado actual y se define como una tupla:

$$M = (Q, q_0, \Sigma, O, \delta, G)$$

Donde:

- Q es un conjunto de estados.
- $q_0 \in Q$ es el estado inicial.
- Σ es el alfabeto de entrada.
- O es el alfabeto de salida.
- $\delta : Q \times \Sigma \to Q$ es la función de transición.
- $G : Q \to Q$ es la función de salida.

Controlador como máquina de Moore

Intentemos escribir el controlador de memoria como una instancia de la tupla anterior.

Controlador como máquina de Moore

Nos ha quedado

- $Q = \{idle, read1, read2, read3, read3, read4, write\}$

Controlador como máquina de Moore

Nos ha quedado

- $Q = \{idle, read1, read2, read3, read3, read4, write\}$
- $q_0 = idle$

Controlador como máquina de Moore

Nos ha quedado

- $Q = \{idle, read1, read2, read3, read3, read4, write\}$
- $q_0 = idle$
- $\Sigma = \{mem, burst, rw\}$

Controlador como máquina de Moore

Nos ha quedado

- $Q = \{idle, read1, read2, read3, read3, read4, write\}$
- $q_0 = idle$
- $\Sigma = \{mem, burst, rw\}$
- $O = \{oe, we\}$

Controlador como máquina de Moore

Nos ha quedado

- $Q = \{idle, read1, read2, read3, read3, read4, write\}$
- $q_0 = idle$
- $\Sigma = \{mem, burst, rw\}$
- $O = \{oe, we\}$
- $\delta = \{(idle, -, idle), (idle, \{mem, rw\}, read1), \ldots\}$

Controlador como máquina de Moore

Nos ha quedado

- $Q = \{idle, read1, read2, read3, read3, read4, write\}$
- $q_0 = idle$
- $\Sigma = \{mem, burst, rw\}$
- $O = \{oe, we\}$
- $\delta = \{(idle, -, idle), (idle, \{mem, rw\}, read1), \ldots\}$
- $G = \{(read1, \{oe\}), (read2, \{oe\}), \ldots\}´$

Máquina de Mealy

Ahora vamos a ver el formalismo de Mealy, que define sus salidas en base a sus entradas y el estado actual.

Máquina de Mealy

Una máquina de Mealy se define como una tupla:

$$M = (S, s_0, \Sigma, \Lambda, T, G)$$

Donde:

- S es un conjunto de estados.

Máquina de Mealy

Una máquina de Mealy se define como una tupla:

$$M = (S, s_0, \Sigma, \Lambda, T, G)$$

Donde:

- S es un conjunto de estados.
- $s_0 \in Q$ es el estado inicial.

Máquina de Mealy

Una máquina de Mealy se define como una tupla:

$$M = (S, s_0, \Sigma, \Lambda, T, G)$$

Donde:

- S es un conjunto de estados.
- $s_0 \in Q$ es el estado inicial.
- Σ es el alfabeto de entrada.

Máquina de Mealy

Una máquina de Mealy se define como una tupla:

$$M = (S, s_0, \Sigma, \Lambda, T, G)$$

Donde:

- S es un conjunto de estados.
- $s_0 \in Q$ es el estado inicial.
- Σ es el alfabeto de entrada.
- Λ es el alfabeto de salida.

Máquina de Mealy

Una máquina de Mealy se define como una tupla:

$$M = (S, s_0, \Sigma, \Lambda, T, G)$$

Donde:

- S es un conjunto de estados.
- $s_0 \in Q$ es el estado inicial.
- Σ es el alfabeto de entrada.
- Λ es el alfabeto de salida.
- $T : S \times \Sigma \to S$ es la función de transición.

Máquina de Mealy

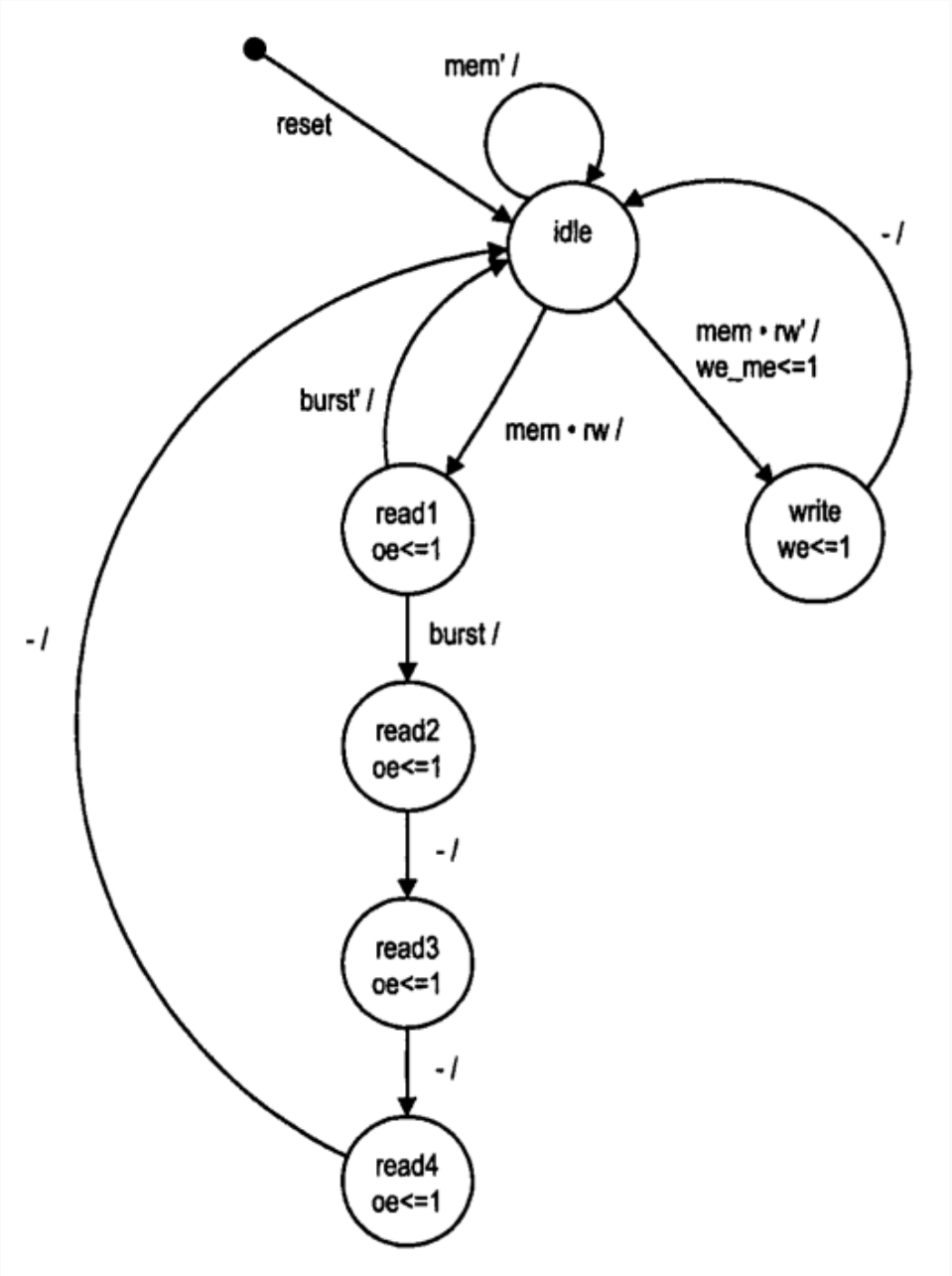
Una máquina de Mealy se define como una tupla:

$$M = (S, s_0, \Sigma, \Lambda, T, G)$$

Donde:

- S es un conjunto de estados.
- $s_0 \in Q$ es el estado inicial.
- Σ es el alfabeto de entrada.
- Λ es el alfabeto de salida.
- $T : S \times \Sigma \to S$ es la función de transición.
- $G : Q \times \Sigma \to \Lambda$ es la función de salida.

¿Un controlador de memoria Mealy o Moore?



Circuitos secuenciales o controladores

Controladores

Vamos a referirnos a un controlador cuando hablemos de un circuito que controla la operación de uno o varios otros circuitos, por ejemplo, el controlador de memoria va a controlar el acceso a los datos del chip de memoria, que se especializa en una función más general (por ej. no tiene el burst).

Controladores

La unidad de control de un procesador va a controlar a los componentes que toman parte del datapath.

Sobre los formalismos

Ahora, ¿pero por qué tenemos dos formalismos tan parecidos?

¿Qué ventajas tiene cada uno?

Sobre los formalismos

Ahora, ¿pero por qué tenemos dos formalismos tan parecidos? ¿Qué ventajas tiene cada uno?

Va a cobrar un poco de sentido al intentar implementar estas máquinas con HDL.

Pregunta

¿Por qué tenemos dos formalismos tan parecidos? ¿Qué ventajas tiene cada uno?

Por lo general Mealy va a tener menos estados y podemos utilizarlo si no está controlando un circuito que esté sincronizado con el clock.

Pregunta

¿Por qué tenemos dos formalismos tan parecidos? ¿Qué ventajas tiene cada uno?

Para Moore, si se emplea en un circuito sincronizado hay que validar que los tiempos de propagación no traigan problemas.

Circuitos secuenciales como controladores

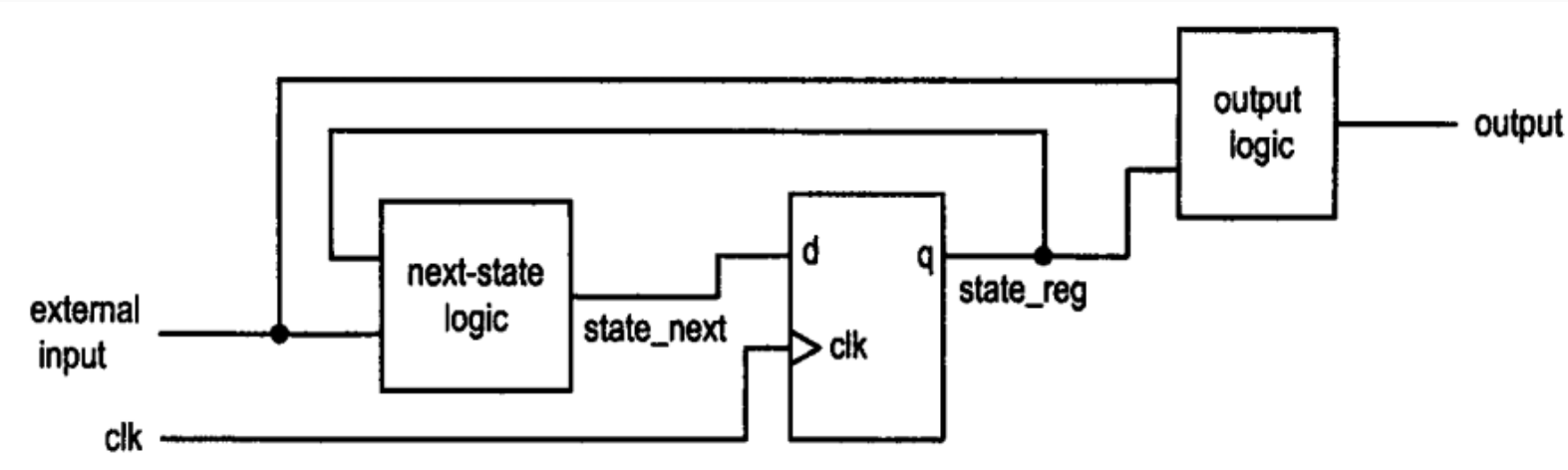
Hasta ahora veníamos diseñando circuitos combinacionales y secuenciales sin reparar mucho en si dependían o no del ambiente en el cual se ejecutan y si debían representar explícitamente su estado.

Circuitos secuenciales como controladores

Vamos a decir que un circuito cuyo estado depende de las entradas observadas es un **reactivo** en oposición a uno que no depende de éstas. Si un circuito tiene registros destinados a representar el estado en el cuál se encuentra, más allá de las señales de entrada o salida, diremos que es de representación **explícita**.

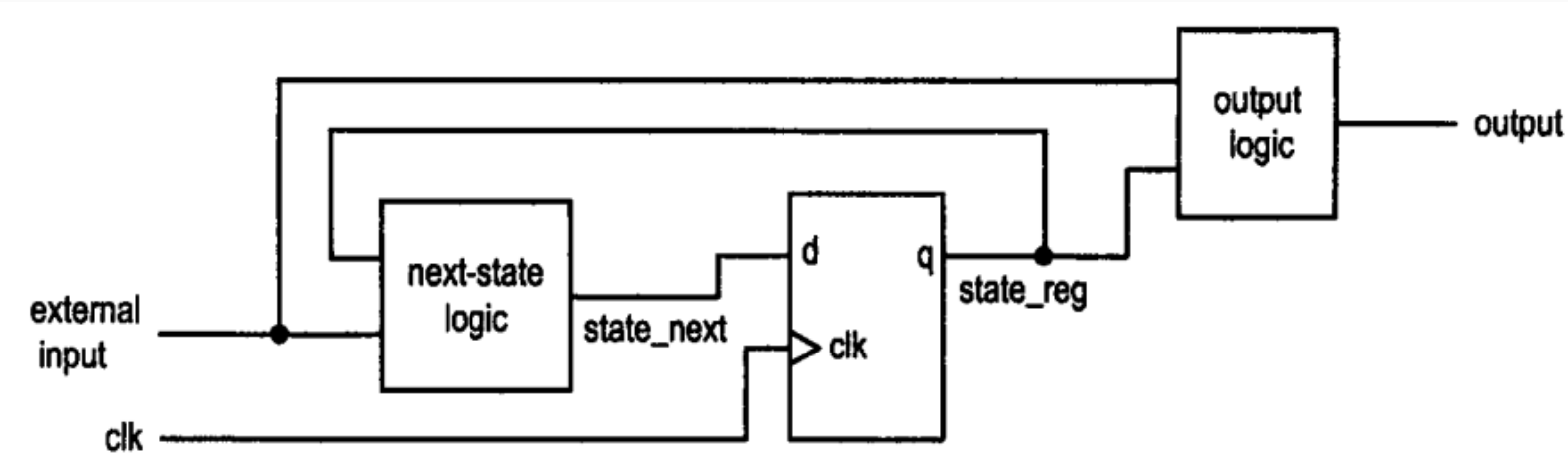
Categorías de un circuito secuencial

Repasemos el esquema de implementación de un circuito secuencial.



Categorías de un circuito secuencial

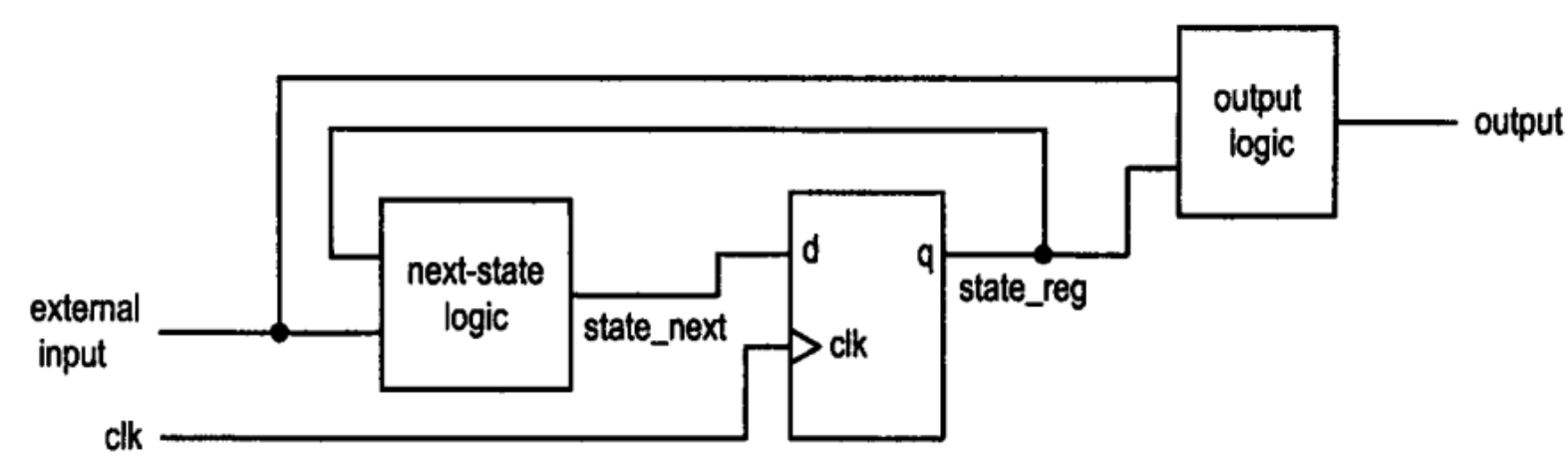
Repasemos el esquema de implementación de un circuito secuencial.



¿Podría ser **reactivo** o tener representación **explícita** de sus estados?

Categorías de un circuito secuencial

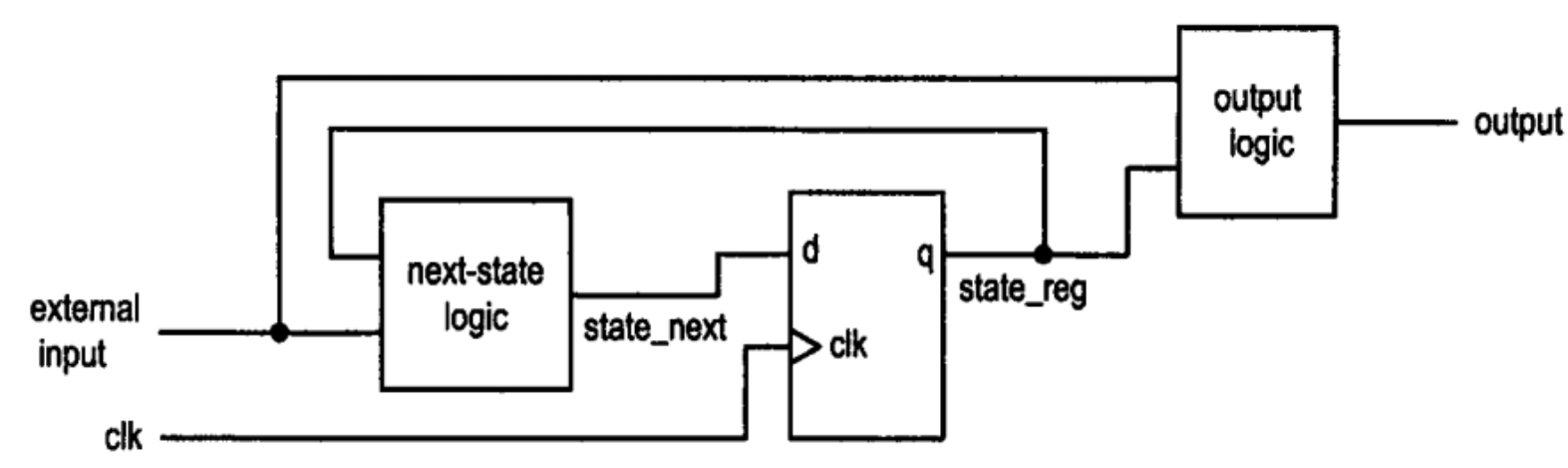
Repasemos el esquema de implementación de un circuito secuencial.



¿Puede implementar una máquina de Moore o de Mealy?

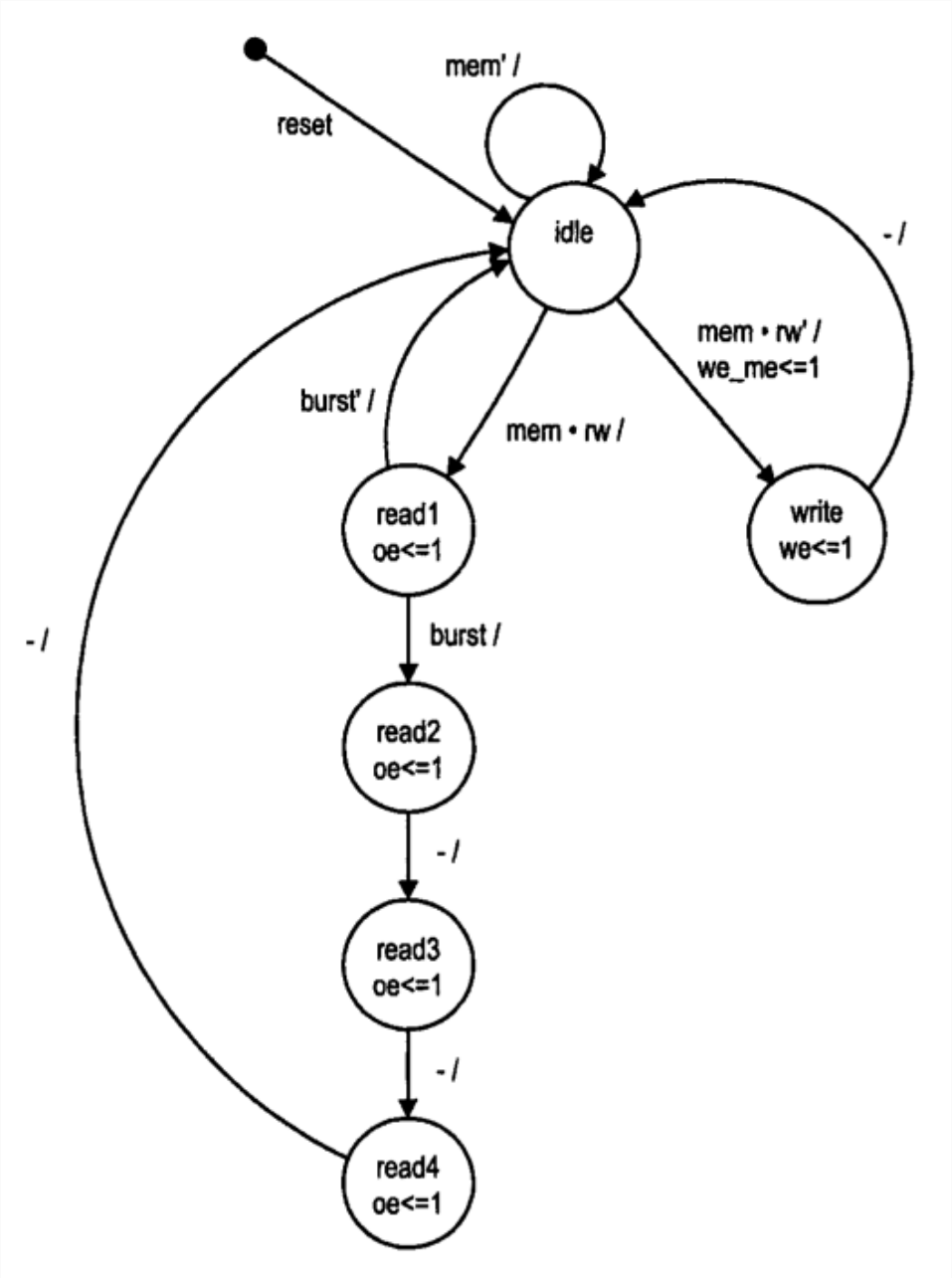
Categorías de un circuito secuencial

Repasemos el esquema de implementación de un circuito secuencial.



¿Cómo encaja nuestro controlador de memoria aquí?

Categorías de un circuito secuencial

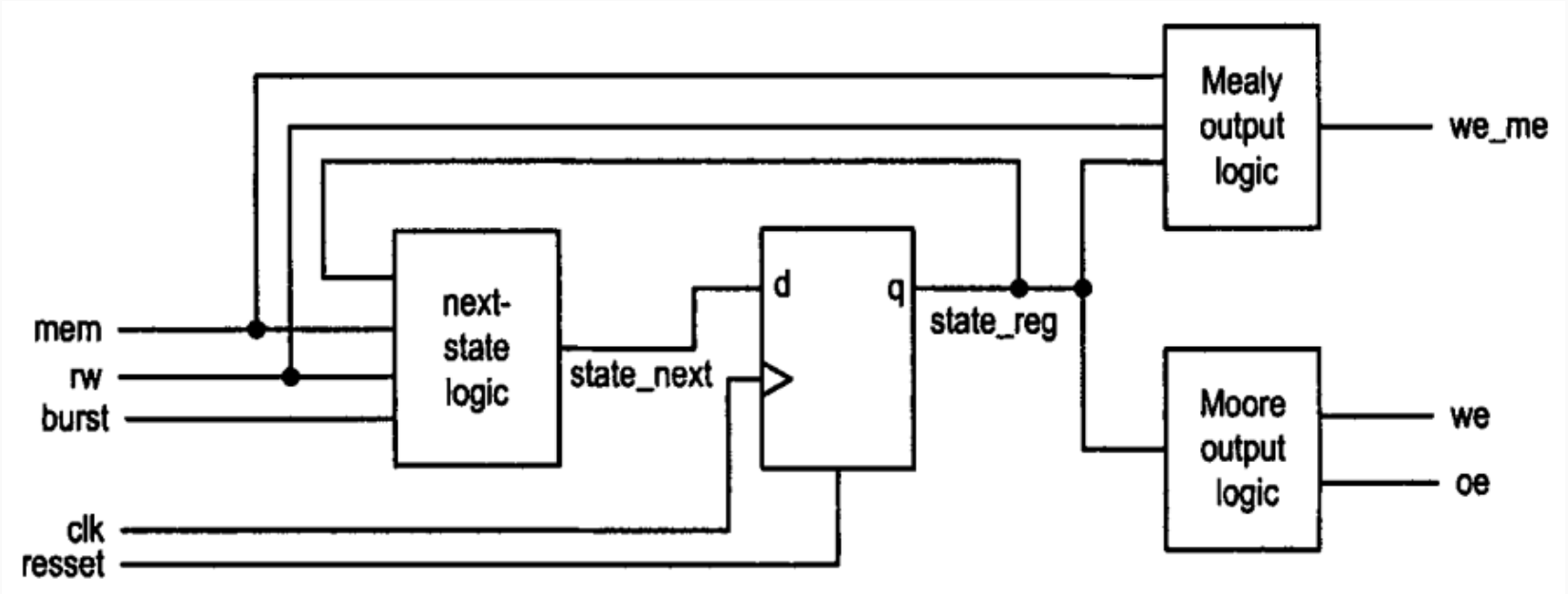


Categorías de un circuito secuencial

Una visión esquemática sería:

Categorías de un circuito secuencial

Una visión esquemática sería:



Implementación en VHDL

Controlador de memoria en VHDL

¿Cómo implementarían el contorlador de memoria en VHDL?

Veamos la interfaz:

```
library ieee;
use ieee.std_logic_1164.all;
entity mem_ctrl is
  port(
    clk, reset: in std_logic;
    mem, rw, burst: in std_logic;
    oe, we, we_me: out std_logic;
  )
end mem_ctrl;
```

Controlador de memoria en VHDL

¿Cómo implementarían el contorlador de memoria en VHDL?

Veamos la interfaz:

```
library ieee;
use ieee.std_logic_1164.all;
entity mem_ctrl is
  port(
    clk, reset: in std_logic;
    mem, rw, burst: in std_logic;
    oe, we, we_me: out std_logic;
  )
end mem_ctrl;
```

¿Es un sistema reactivo? ¿Va a tener estado explícito?

Controlador de memoria en VHDL

¿Cómo implementarían el contorlador de memoria en VHDL?

Veamos la interfaz:

```
library ieee;
use ieee.std_logic_1164.all;
entity mem_ctrl is
  port(
    clk, reset: in std_logic;
    mem, rw, burst: in std_logic;
    oe, we, we_me: out std_logic;
  )
end mem_ctrl;
```

¿Es un sistema reactivo? ¿Va a tener estado explícito?

Va a ser reactivo y tendrá estados explícitos.

Controlador de memoria en VHDL - multi segmentos

Vamos a definir el comportamiento con un enfoque de multi segmentos, que significa que **vamos a descomponer la actualización de estado y de las salidas en bloques separados**. Vamos tener:

Vamos a definir el comportamiento con un enfoque de multi segmentos, que significa que **vamos a descomponer la actualización de estado y de las salidas en bloques separados**. Vamos tener:

- Bloque de próximo estado.

Controlador de memoria en VHDL - multi segmentos

Vamos a definir el comportamiento con un enfoque de multi segmentos, que significa que **vamos a descomponer la actualización de estado y de las salidas en bloques separados**. Vamos tener:

- Bloque de próximo estado.
- Bloque de salida de Moore.

Controlador de memoria en VHDL - multi segmentos

Vamos a definir el comportamiento con un enfoque de multi segmentos, que significa que **vamos a descomponer la actualización de estado y de las salidas en bloques separados**. Vamos tener:

- Bloque de próximo estado.
- Bloque de salida de Moore.
- Bloque de salida de Mealy.

Controlador de memoria en VHDL - estado

Definimos los registros de estado.

```
...
architecture mult_seg_arch of mem_ctrl is
  -- vamos a utilizar un enum por ahora
  type mc_state_type is
    (idle, read1, read2, read3, read4, write);
  -- y definimos nuestros estados explicitos
  signal state_reg, state_next : mc_state_type;
begin
  ...
end mult_seg_arch;
```

Controlador de memoria en VHDL - estado

Encabezado.

```
library ieee;
use ieee.std_logic_1164.all;
entity mem_ctrl is
  port(
    clk, reset: in std_logic;
    mem, rw, burst: in std_logic;
    oe, we, we_me: out std_logic;
  )
end mem_ctrl;
architecture mult_seg_arch of mem_ctrl is
type mc_state_type is (idle, read1, ...);
signal state_reg, state_next : mc_state_type;
begin
...
end mult_seg_arch;
```

Controlador de memoria en VHDL - estado

Actualización del registro de estado

```
architecture mult_seg_arch of mem_ctrl is
...
begin
...
process(clk, reset)
begin
  if(reset='1') then
    state_reg <= idle;
  elsif (rising_edge(clk)) then
    state_reg <= staten_next;
  end if;
end process;
...
end mult_seg_arch;
```

Actualización del registro de próximo estado

```
process(state_reg, mem, rw, burst)
begin
  case state_reg is
    when idle =>
      if mem = '1' then
        if rw = '1' then
          state_next <= read1;
        else
          state_next <= write;
        end if;
      else
        state_next <= idle;
      enf if;
    when write =>
      ...
  end case;
end process;
```

Asignación de salida de Moore.

```
process(state_reg)
begin
  we <= '0'; -- valores por defecto
  oe <= '0';
  case state_reg is
    when idle =>
    when write =>
      we = '1';
    when read1 =>
      oe => '1';
    ...
  end case;
end process;
```

Asignación de salida de Mealy.

```
process(state_reg, mem, rw)
begin
  we_me <= '0'; -- valores por defecto
case state_reg is
  when idle =>
    if(mem = '1') and (rw = '0') then
      we_me <= '1';
    end if;
    when write =>
    ...
    when read4 =>
  end case;
end process;
```

Con esto vimos una implementación típica de un controlador basado en máquinas de estado.

Cierre

Repaso de hoy

En la clase hoy vimos:

- Estado, trazas y valuaciones.

Repaso de hoy

En la clase hoy vimos:

- Estado, trazas y valuaciones.
- Máquinas de estado, **Moore y Mealy**.

Repaso de hoy

En la clase hoy vimos:

- Estado, trazas y valuaciones.
- Máquinas de estado, **Moore y Mealy**.
- ¿Circuitos secuenciales o controladores?

Repaso de hoy

En la clase hoy vimos:

- Estado, trazas y valuaciones.
- Máquinas de estado, **Moore y Mealy**.
- ¿Circuitos secuenciales o controladores?
- Implementación de un controlador con estados en `VHDL`.

Clase próxima

En la clase que viene veremos:

- Model checking.

Clase próxima

En la clase que viene veremos:

- Model checking.
- Lógicas temporales y μ-cálculo.

Clase próxima

En la clase que viene veremos:

- Model checking.
- Lógicas temporales y μ-cálculo.
- Sistemas reactivos.

Clase próxima

En la clase que viene veremos:

- Model checking.
- Lógicas temporales y μ-cálculo.
- Sistemas reactivos.
- Assertion based verification (`SVA`, `PSL`, `SystemVerilog`).

Clase próxima

En la clase que viene veremos:

- Model checking.
- Lógicas temporales y μ-cálculo.
- Sistemas reactivos.
- Assertion based verification (`SVA`, `PSL`, `SystemVerilog`).
- Un ejemplo de verificación con PSL.

Clase próxima

En la clase que viene veremos:

- Model checking.
- Lógicas temporales y μ-cálculo.
- Sistemas reactivos.
- Assertion based verification (`SVA`, `PSL`, `SystemVerilog`).
- Un ejemplo de verificación con PSL.
- Verificación y síntesis de especificaciones.

Máquinas de estado - Fin

Cierre y preguntas.