

SÍNTESIS DE LÓGICA SECUENCIAL

- Se utilizan estos términos para definir diseños en los cuales existen elementos de almacenamiento, particularmente Flip Flops.

Lenguajes de Descripción de Hardware

INICIALIZACIÓN

- Cuando se trabaja con máquinas secuenciales se debe tener en cuenta que es necesario proveer un sistema de inicialización. No es estrictamente necesario para simulación ya que pueden usarse valores por defecto.
- Normalmente se utiliza una entrada de reset para estos propósitos.
- El comportamiento asíncrono puede lograrse sólo con procesos con lista de sensibilidad. El proceso debe reaccionar al clock y a la señal de reset.

Lenguajes de Descripción de Hardware

EJEMPLO

```
process ( clk, reset)
begin
    if (reset='1') then
        data <= 0 ;
    elsif (clk'event and clk = 1) then
        data <= input ;
    end if ;
end process ;
```

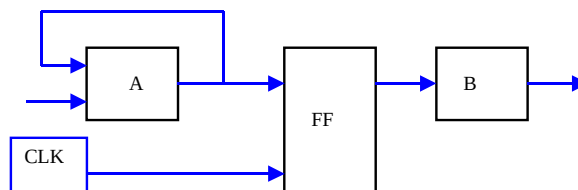
Lenguajes de Descripción de Hardware

SEÑALES EN PROCESOS SENSIBLES AL RELOJ

- Debe tenerse en cuenta que todas las señales internas a un proceso sensible al reloj tendrán asignado un flip flop. Debe hacerse un modelado cuidadoso ya que se pueden obtener resultados indeseados.

Ejemplo:

- Supongamos que se desea modelar el sistema siguiente:



Lenguajes de Descripción de Hardware

SEÑALES EN PROCESOS SENSIBLES AL RELOJ

- FORMA 1:

```
logic_A : process
begin
wait until clk'event and clk = 1;
:.....:
    DESCRIPCIÓN DE LÓGICA A
:.....:
end process logic_A;
```

```
logic_B : process (ST)
begin
:.....:
    DESCRIPCIÓN DE LÓGICA B
:.....:
end process logic_B;
```

- En este caso se lograría el comportamiento deseado ya que se inferirían FF sólo a la salida de la lógica A.

Lenguajes de Descripción de Hardware

SEÑALES EN PROCESOS SENSIBLES AL RELOJ

- FORMA 2

```
logic_AB : process
begin
wait until clk'event and clk = 1;
:.....:
    DESCRIPCIÓN DE LÓGICA A_B
:.....:
end process logic_AB;
```

- En este caso se inferirían FF también a la salida de la lógica B, lo cual conduce a un comportamiento indeseable.

Lenguajes de Descripción de Hardware

VARIABLES EN PROCESOS SENSIBLES AL RELOJ

- La implementación en hardware de variables depende de su utilización en el process. El lenguaje VHDL garantiza que las variables todavía conservan su valor cuando un process se ejecuta nuevamente.
- Los FFs son inferidos para variables sólo cuando son leídas antes de ser actualizadas.

Lenguajes de Descripción de Hardware

VARIABLES EN PROCESOS SENSIBLES AL RELOJ - EJEMPLOS

```
Var1 : process (clk)
variable tmp : integer ;
begin
    if (clk'event and clk = '1') then
        tmp := input * 2 ;
        outputA <= tmp + 1 ;
        outputB <= tmp + 2 ;
    end if ;
end process Var1;
```

- En este caso no se infieren FFs ya que el valor de la constante es actualizado previo a su utilización. En este caso un elemento de almacenamiento sería redundante.

Lenguajes de Descripción de Hardware

VARIABLES EN PROCESOS SENSIBLES AL RELOJ - EJEMPLOS

```
Var2 : process (clk)
variable tmp : integer ;
begin
    if (clk'event and clk = '1') then
        output <= temp + 1 ;
        temp := input * 2 ;
    end if ;
end process Var2;
```

- En este caso se infiere un banco de registros para almacenar a temp (se inferirá un registro de 32 bits ¡!!!)

Lenguajes de Descripción de Hardware

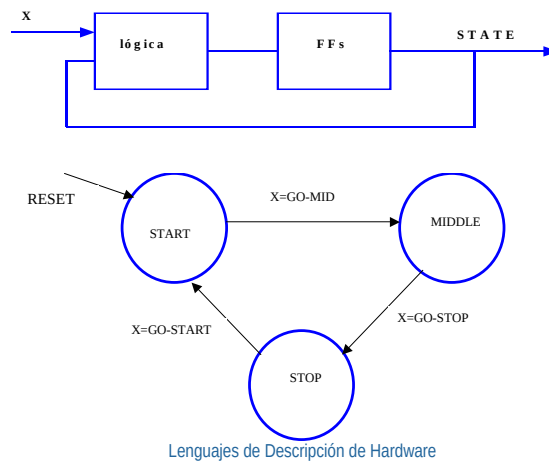
MÁQUINAS DE ESTADOS FINITOS

- En general las máquinas de estados finitos pueden ser descritas por un código único o dividido en dos partes, uno para la sección combinacional y la otra para los registros.
- Los estados de la máquina son descritos mediante tipos de datos enumerados, lo cual facilita la codificación VHDL.
- Hay tres tipos básicos de máquinas de estado finito: Máquinas de Moore, Máquinas de Mealy y Máquinas de Medvedev.

Lenguajes de Descripción de Hardware

DESCRIPCIÓN DE MÁQUINAS

▪ CON UN PROCESS



DESCRIPCIÓN DE MÁQUINAS: UN PROCESS

```

fsm_ff: process(clk, reset)
begin
    if reset = '1' then
        state <= start;
    elsif clk'event and clk='1' then
        case state is
            when start =>
                if x = go_mid then
                    state <= middle;
                end if;
            when middle =>
                if x = go_stop then
                    state <= stop;
                end if;
            when stop =>
                if x = go_start then
                    state <= start;
                end if;
            when others =>
                state <= start;
            end case;
        end if;
    end process fsm_ff;
    
```

Lenguajes de Descripción de Hardware

DESCRIPCIÓN DE MÁQUINAS:DOS PROCESS

```
fsm_ff : process (clk, reset)
begin
    if (reset = '1') then
        state <= start ;
    elsif clk'event and clk = '1' then
        state <= next _state;
    end if;
end process fsm_ff;
fsm_logic : process (state, x)
begin
    case state is
        when start =>
            if x = go_mid then
                next_state <= middle ;
            end if;
        when middle => .....
        :::::::::::
        when others => next state <= start ;
    end case;
end process fsm_logic ;
```

Lenguajes de Descripción de Hardware

DECISIONES SOBRE LA CANTIDAD DE PROCESOS

- ESTRUCTURA Y LEGIBILIDAD DEL CÓDIGO
 - El código debiera representar de una manera clara al hardware. La lógica combinacional y los registros del autómata son estructuras muy diferentes. Desde este punto de vista convendría utilizar dos process.
 - Sin embargo desde el punto de vista del comportamiento sólo es necesaria una comprensión al nivel brindado por el diagrama de burbuja. Desde este punto de vista convendría usar un solo process.
 - El usuario debe decidir entre estas dos posibilidades.

Lenguajes de Descripción de Hardware

DECISIONES SOBRE LA CANTIDAD DE PROCESOS

- SIMULACIÓN
- La descripción en dos process da la posibilidad de acceder a señales intermedias lo cual es importante para la depuración del circuito. Desde este punto de vista es mejor.

Lenguajes de Descripción de Hardware

DECISIONES SOBRE LA CANTIDAD DE PROCESOS

- SÍNTESIS
- Cada herramienta posee sus propios algoritmos de síntesis. Es imposible dar una regla general. Un grupo importante de herramientas da mejores síntesis cuando se describe en dos process (desde el punto de vista de la cantidad de recursos).

Lenguajes de Descripción de Hardware

CODIFICACIÓN DE ESTADOS

- Una máquina de estados es una representación de hardware. Los estados deben codificarse en binario. Esto puede ser hecho a mano por el usuario o la herramienta de síntesis mapeará los nombres de los estados a representaciones binarias.
- La mayoría de las herramientas toman el binario natural por defecto.
- Es posible hacer inferir a la herramienta otros tipos de codificación. Si se dan restricciones de velocidad el compilador utiliza por lo general el código “uno entre n”.
- Deben tenerse en cuenta los estados que no se utilizan. Si necesitamos una máquina de tres estados son requeridos dos bits para la codificación. Por lo tanto es posible que la máquina vaya a este estado en forma accidental. Este es un estado no seguro y se deben dar los medios para salir de él.

Lenguajes de Descripción de Hardware

EXTENSIÓN DE LA SENTENCIA CASE

- Es la forma más obvia de evitar los estados no deseados. En general lo que se pretende es reinicializar la máquina ante esta condición.

```
type state_type is ( start, middle, stop);  
signal state : state_type;  
.....  
case state is  
when start => .....  
When middle => .....  
when stop => .....  
when others => .....  
end case;
```

- VHDL es un lenguaje muy estricto: sólo pueden ser accedidos para simulación los estados declarados. No es posible simular este tipo de anomalías.
- Muchas herramientas ignoran directamente la sentencia “when others” y por lo general conducen a máquinas de estados inseguras.
- La utilización de “when others” es consecuentemente poco recomendable.

Lenguajes de Descripción de Hardware

EXTENSIÓN DE LA DECLARACIÓN DE TIPO

- Es una alternativa para poder simular los estados erróneos de la máquina de estados.
- Para ello se debe extender la declaración a todos los estados posibles a los que pueda ir la máquina.
- Es poco recomendable cuando se utilizan restricciones de alta velocidad ya que la sobreasignación de recursos de hardware es muy marcada. (cuando se utiliza codificación “one shot”).
- Es sumamente tediosa ya que el usuario debe definir todos los estados posibles manualmente.

Lenguajes de Descripción de Hardware

CODIFICACIÓN MANUAL

- El usuario decide cuál es el código que se utilizará para representar a los estados de la máquina.
- Se utilizan vectores de bits y constantes para modelar el comportamiento.
- Todos los estados erróneos pueden ser simulados.
- Presenta el problema de que es muy tediosa para el usuario.

Lenguajes de Descripción de Hardware

EJEMPLO- CODIFICACIÓN MANUAL

```
subtype state_type is std_logic_vector (1 downto 0) ;
signal state : state_type;

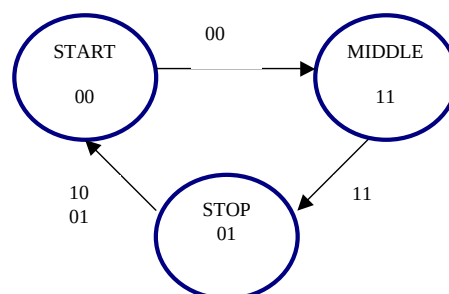
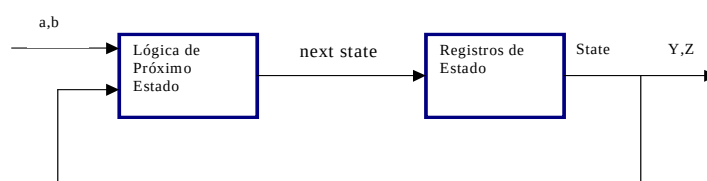
constant start : state_type := "01";
constant middle : state_type := "11" ;
constant stop : state_type := "00" ;
:.....
:.....

case state is
when start =>:.....
when middle => :.....
when stop => :.....
when others => :.....

end case;
```

Lenguajes de Descripción de Hardware

MÁQUINAS DE MEDVEDEV



Lenguajes de Descripción de Hardware

MÁQUINAS DE MEDVEDEV

```

architecture rtl of medvedev is
    subtype state_type is std_logic_vector (1
        downto 0);
    constant start : state_type := "00" ;
    constant middle : state_type := "11" ;
    constant stop : state_type := "01" ;
    signal state, nextstate, : state_type ;

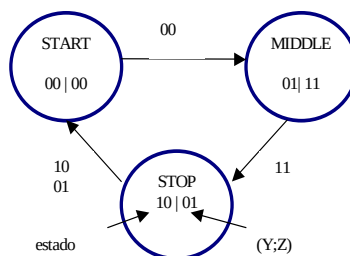
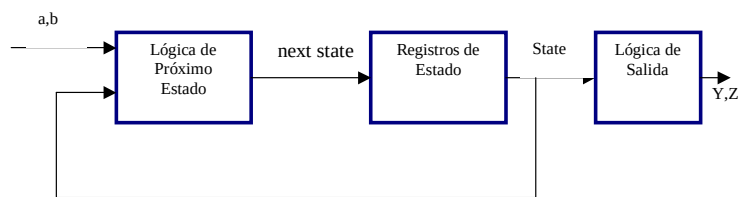
begin
    reg : process (clk, reset)
    begin
        if reset = '1' then
            state <= start ;
        elsif clk'event and clk = '1' then
            state <= nextstate ;
        end if ;
    end process reg ;

    cmb : process (a, b , state)
    begin
        nextstate <= state ;
        case state is
            when start =>
                if (a or b) = '0' then
                    nextstate <= middle ;
                end if ;
            when middle =>
                if (a and b) = '1' then
                    nextstate <= stop ;
                end if ;
            when stop =>
                if (a xor b) = '1' then
                    nextstate <= start ;
                end if ;
            when others =>
                nextstate <= start ;
            end case ;
        end process cmb;
        (y,z) <= state ;
    end rtl;

```

Lenguajes de Descripción de Hardware

MÁQUINA DE MOORE



Lenguajes de Descripción de Hardware

MÁQUINA DE MOORE

```
architecture rtl of moore is
    subtype state_type is std_logic_vector(1 downto 0);
    constant start:state_type:= "00";
    constant middle:state_type:= "01";
    constant stop:state_type := "10";
    signal state, nextstate:state_type;
begin
    reg : process (clk, reset)
    begin
        if reset = '1' then
            state <= start;
        elsif clk'event and clk='1' then
            state <= nextstate;
        end if ;
    end process reg ;
```

Lenguajes de Descripción de Hardware

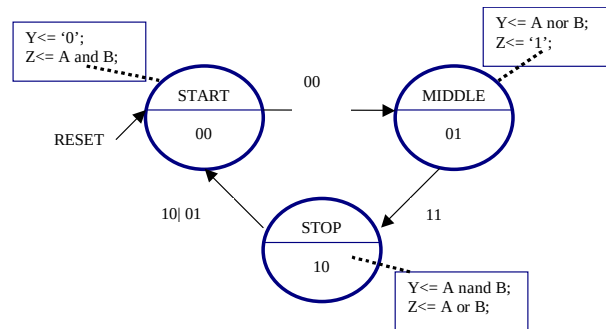
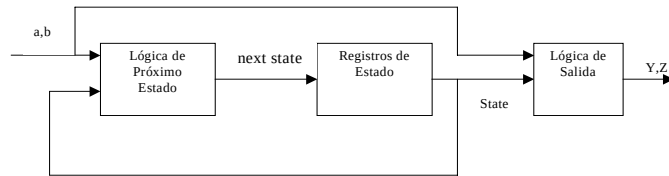
MÁQUINA DE MOORE

```
cmb : process (a, b, state)
begin
    nextstate <= state ;
    case state is
        when start =>
            if (a or b)='0' then
                nextstate <= middle;
            end if ;
        when middle =>
            if (a and b)='1' then
                nextstate => stop ;
            end if ;
        when stop =>
            if (a xor b) = '1' then
                nexstate <= start ;
            end if ;
        when others =>
            nextstate <= start ;
    end case ;
end process cmb ;

y <= '1' when state=middle
else '0';
z <= '1' when state=middle or
state=stop else '0' ;
end rtl ;
```

Lenguajes de Descripción de Hardware

MÁQUINA DE MEALY



Lenguajes de Descripción de Hardware

MÁQUINA DE MEALY

```
architecture rtl of mealy is
  subtype state_type is std_logic_vector (1 downto 0);
  constant start : state_type := "00" ;
  constant middle : state_type := "01" ;
  constant stop : state_type := "10" ;
  signal state, nextstate, : state_type ;
begin
  reg: process (clk, reset)
  begin
    if reset = '1' then
      state <= start ;
    elsif clk'event and clk = '1' then
      state <= nextstate ;
    end if ;
  end process reg ;
end architecture rtl;
```

Lenguajes de Descripción de Hardware

MÁQUINA DE MEALY

```
cmb: process (a, b, state)
begin
  EN ESTE PROCESS SE PONE LO MISMO QUE EN LOS
  ANTERIORES
end process cmb ;

y <= '1' when
  (state=middle and (a or b)='0') or (state=stop and (a
    and b)='0')
else '0' ;

z = '1' when
  (state=start and (a and b)='1') or (state=middle) or
  (state = stop and ( a or b) = '1')
else '0' ;
end rtl ;
```

Lenguajes de Descripción de Hardware