

SÍNTESIS

- Es el proceso de transformación de una descripción VHDL en una descripción más detallada.
- Por ejemplo:
 - El operador + es transformado en una netlist de compuertas.
 - IF A=B THEN ... se convierte en un comparador que controla un multiplexor.

Lenguajes de Descripción de Hardware

SÍNTESIS

- Debe tenerse en consideración que sólo un subset de las sentencias son sintetizables.
- El circuito final dependerá de la herramienta de síntesis empleada y de las bibliotecas de celdas que esta disponga.
- Las operaciones de comparación y decisión son por lo general fácilmente sintetizables a una descripción de compuertas, pero hay otras operaciones como las matemáticas que se mapean a una librería de macroceldas propia de la herramienta.
- Diferentes herramientas implican diferentes soportes de lenguaje. Esto producirá diferentes implementaciones.

Lenguajes de Descripción de Hardware

EJEMPLO

```
process (clk)
begin
    if clk = '1' then
        q <= d ;
    end if ;
end process ;
```

- Si la herramienta soporta lista de sensibilidad entonces se infiere un flip flop, caso contrario se infiere un latch transparente.

Lenguajes de Descripción de Hardware

RESTRICCIONES ADICIONALES

- Además de las restricciones impuestas por las capacidades de la herramienta y la tecnología de implementación, el usuario puede imponer otras:
- Velocidad de operación.
- Recursos de hardware requeridos.
- Las descripciones son transformadas para poder cumplir con las restricciones, lo cual puede hacerse mediante modelos puramente matemáticos o mediante diferentes mapeos de las ecuaciones booleanas sobre la tecnología utilizada.

Lenguajes de Descripción de Hardware

RESTRICCIONES ADICIONALES

- Las optimizaciones requieren numerosas iteraciones aunque puede suceder que no se alcancen a cumplir las restricciones propuestas. Si esto sucede quedan las siguientes alternativas:
 - Alterar las condiciones de operación.
 - Alterar las restricciones.
 - Alterar la jerarquía.
 - Alterar la descripción VHDL.

Lenguajes de Descripción de Hardware

PROBLEMAS CON LAS HERRAMIENTAS DE SÍNTESIS

- La información del layout que impacta sobre el timing falta durante el proceso de síntesis.
- El árbol de reloj debe ser generado luego del proceso de síntesis (NO FPGA)
- Los esquemas de reloj complicados son difíciles de sintetizar (clocks negados, diferentes clocks, etc).
- Las memorias son difíciles de generar.
- Cuando se disponen de diferentes PADS de entrada salida debe hacerse una selección manual, ya que no existe un algoritmo de instanciación general.

Lenguajes de Descripción de Hardware

ESTRATEGIA DE SÍNTESIS

- El estilo de codificación VHDL tiene un gran impacto sobre los resultados de la síntesis. Esto debe ser tenido en cuenta desde los estadios iniciales de diseño. Algunas consideraciones al respecto son las siguientes:
- Los algoritmos de síntesis trabajan mejor en bloques que no superen los varios miles de compuertas. Por lo tanto debe plantearse una división en bloques no tan grandes.
- Dentro de un mismo diseño pueden coexistir distintos criterios de optimización, los cuales pueden ser utilizados para realizar la partición antes de la síntesis.

Lenguajes de Descripción de Hardware

EL ESTILO RTL

- En el estilo RTL el diseño se divide en una sección que describe la lógica combinacional y otra que hace lo propio con los registros. Por lo general se utilizan dos process:
 - Un process combinacional que describe la lógica de próximo estado
 - Un process sensible al reloj que describe los registros.

Lenguajes de Descripción de Hardware

EL PROCESS COMBINACIONAL

- LA LISTA DE SENSIBILIDAD
- La lista de sensibilidad consiste de todas las señales que son leídas dentro del process. Las herramientas de síntesis normalmente ignoran a esta lista pero no así las de simulación. Es importante no olvidar señales si se pretende que la simulación sea similar al comportamiento del hardware resultante.

Lenguajes de Descripción de Hardware

EJEMPLO : MULTIPLEXOR

```
process (a,b,select)
begin
    if ( select = 1 ) then
        out <= a ;
    else
        out<= b ;
    end if ;
end process ;
```

Si la señal select no se incluyera en la lista de sensibilidad:

- La herramienta de síntesis generaría igualmente un multiplexor
- La herramienta de simulación no simularía un multiplexor: habría cambios en la salida sólo si hay cambios en las señales a o b.
- Consecuentemente, la funcionalidad de la entrada select no puede ser simulada.

Lenguajes de Descripción de Hardware

LA SENTENCIA WAIT

- Es posible modelar con la sentencia “WAIT ON” comportamientos similares a los modelados con la lista de sensibilidad.
 - El process obtenido es equivalente.
 - Se debe utilizar lista de sensibilidad o wait on : son mutuamente excluyentes.
 - El wait debe ir al final del process.
 - En el caso de lista de sensibilidad, el process es ejecutado cuando se produce un cambio en alguna de las señales, y se ejecuta nuevamente solo si se producen nuevos cambios.
 - En el caso del wait el process es ejecutado una vez y se suspende hasta que la condición especificada en el wait se satisfaga.

Lenguajes de Descripción de Hardware

EJEMPLO: DOS PROCESS EQUIVALENTES

```
-----  
process1  
begin  
  if sel = '1' then  
    z<=a ;  
  else  
    z<=b ;  
  end if ;  
wait on a, b , sel ;  
end process1 ;
```

```
-----  
process2 ( a, b , sel )  
begin  
  if sel = '1' then  
    z<=a ;  
  else  
    z<=b ;  
  end if ;  
end process2 ;
```

Lenguajes de Descripción de Hardware

ASIGNACIONES INCOMPLETAS EN PROCESS INCONDICIONAL.

- Cuando se modela lógica combinacional debe tenerse especial cuidado para evitar la generación de registros indeseados.

Lenguajes de Descripción de Hardware

Ejemplo

<pre>entity mux is port (a, b, select : in std_logic; z : out std_logic) ; end mux ;</pre>		
<pre>architecture mal of mux is begin process (a,b,select) begin if select = '1' then z <= a ; end if ; end process ; end mal;</pre>	<pre>architecture ok1 of mux is begin process (a,b,select) begin z <= b ; if select = '1' then z <= a ; end if ; end process ; end ok1 ;</pre>	<pre>architecture ok2 of mux is begin process (a,b,select) begin if select = '1' then z <= a ; else z <= b ; end if ; end process ; end ok2 ;</pre>

- El código de la izquierda no tiene una rama else incondicional. El valor de z es preservado en el caso de que select sea cero.
- Consecuentemente la herramienta de síntesis generará un registro para que el circuito sea capaz de cumplir con lo anterior, y no se logra la funcionalidad de un multiplexor.
- Los otros dos códigos serán sintetizados como multiplexores.

Lenguajes de Descripción de Hardware

PROCESOS SENSIBLES AL RELOJ

- DETECCIÓN DE FLANCOS DE RELOJ.
 - Las listas de sensibilidad son normalmente ignoradas por las herramientas de síntesis.
 - Las sentencias WAIT son generalmente no sintetizables.
 - Fue necesario debido a lo anterior generar una solución para el modelado de elementos de almacenamiento.
 - Los sintetizadores buscan segmentos de código para inferir estos elementos.
 - Existen diferentes alternativas para modelar flip flops.
 - No todas son soportadas por las herramientas de la actualidad.
 - La primera de las opciones es la generalmente soportada.
 - La simulación da idénticos resultados.

Lenguajes de Descripción de Hardware

PROCESOS SENSIBLES AL RELOJ

- Las alternativas se enumeran en la tabla siguiente.

IF	WAIT UNTIL
Clock_signal_name'event and Clock_signal_name= '1'	Clock_signal_name'event and Clock_signal_name = '1'
Clock_signal_name = '1' and Clock_signal_name'event	Clock_signal_name = '1' and Clock_signal_name'event
Not Clock_signal_name'stable And Clock_signal_name = '1'	Not Clock_signal_name'stable And Clock_signal_name = '1'
Clock_signal_name = '1' and Not Clock_signal_name'stable	Clock_signal_name = '1' and Not Clock_signal_name'stable
Rising edge (Clock_signal_name)	Rising edge (Clock_signal_name)
	Clock_signal_name = '1'

Lenguajes de Descripción de Hardware

INFERENCIA DE REGISTROS

- EJEMPLO : Modelado de un contador decimal de 1 dígito.

```
library IEEE;
use IEEE.std_logic_1164.all ;
entity counter is
port ( clk : in std_logic ;
      q : out integer range 0 to 15);
end counter ;

architecture RTL of counter is
signal count : integer range 0 to 15;
begin
process (clk)
begin
    if clk'event and clk = '1' then
        if count >= 9 then
            count <= 0 ;
        else
            count <= count + 1 ;
        end if ;
    end if ;
end process;

q <= count ;
end RTL ;
```

OBSERVACIONES :

- El contador no es reseteable. No hay problemas en la simulación ya que es posible inicializarlo a los valores por defecto de las señales. Es recomendable agregar un mecanismo de reset si se lo va a sintetizar.
- El rango de la señal entera ha sido restringido. Observar que no se lo ha restringido a 9 sino a 15. Esto evita potenciales problemas con las herramientas de síntesis.
- Se ha declarado la señal interna count además de puerto del salida q. Esto es debido a que q se declara como puerto de salida y no puede ser leído dentro de la arquitectura.
- Los flips flops se infieren únicamente en procesos sensibles al clock. Cada señal que puede ser actualizada en un proceso sensible a reloj recibe un registro.

Lenguajes de Descripción de Hardware

SET / RESET ASÍNCRONO

- Si se desea emplear una estrategia de reset sincrónico, se debe tratar a la señal de reset como cualquier señal de control, es decir que la señal de reloj será la única en la lista de sensibilidad del proceso. Esto también puede implementarse con una estructura wait-until.
- Las señales de control asíncronas se pueden modelar solamente con procesos con lista de sensibilidad. Todas las señales que puedan disparar el proceso deben ser listadas.
- El proceso consta de una estructura IF en donde las señales asíncronas son verificadas primero seguidas por la detección del flanco de reloj. Consecuentemente, las señales asíncronas son tratadas con un nivel de privilegio superior.
- Un ELSE incondicional está estrictamente prohibido ya que las sentencias que tienen que ser procesadas cuando no está el flanco de reloj no tienen representación física.

Lenguajes de Descripción de Hardware

EJEMPLO:

```
library IEEE;
use ieee.std_logic_1164.all;

entity ASYNC_FF is
port (    d, clk, set, rst : in std_logic;
      q : out std_logic);
end ASYNC_FF ;

architecture rtl of ASYNC_FF is
begin

    process (clk,rst,set)
    begin
        if rst ='1' then
            q<='0';
        elsif set='1' then
            q<='1';
        elsif (clk'event and clk='1')
        then
            q<=d;
        end if;
    end process;
end rtl;
```

Lenguajes de Descripción de Hardware

LÓGICA COMBINACIONAL

- LAZOS DE REALIMENTACIÓN
 - Cuando se modela lógica combinacional es necesario evitar lazos de realimentación. Un lazo se dispararía a si mismo todo el tiempo.

- EJEMPLO

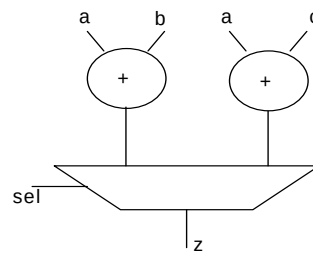
```
architecture ejemplo of realimentación is
signal b, x : integer range 0 to 99 ;
begin
    process ( x, b )
    begin
        x <= x + b ;
    end process ;
...
...
end example ;
```

Lenguajes de Descripción de Hardware

INFLUENCIA DEL ESTILO DE CODIFICACIÓN

- Implementación directa

```
process ( sel , a , b , c )  
begin  
  if sel = '1' then  
    z <= a + b ;  
  else  
    z <= a + c ;  
  end if ;  
end process ;
```

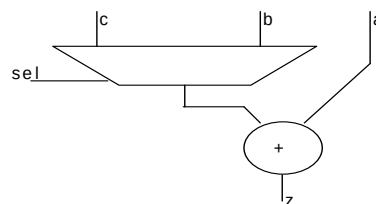


Lenguajes de Descripción de Hardware

INFLUENCIA DEL ESTILO DE CODIFICACIÓN

- Distribución Manual de Recursos

```
process (sel, a , b , c)  
  variable tmp : bit ;  
begin  
  if sel = '1' then  
    tmp := b ;  
  else  
    tmp := c ;  
  end if ;  
  z <= a + tmp ;  
end process ;
```



Lenguajes de Descripción de Hardware

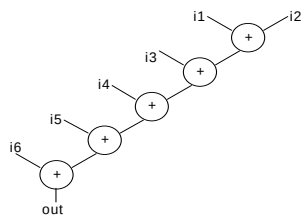
INFLUENCIA DEL ESTILO DE CODIFICACIÓN

Optimización del código fuente

- Una operación puede ser descrita eficientemente para síntesis.

Ejemplo:

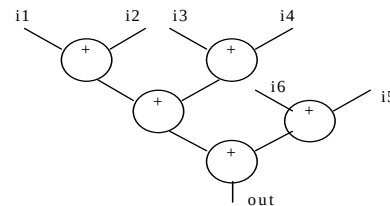
$$\text{out} = i1 + i2 + i3 + i4 + i5 + i6$$



- En la descripción de arriba la señal en el peor caso debe propagarse por cinco sumadores, pudiendo conducir a demoras inaceptables.

- Otra posibilidad de describir la misma operación :

$$\text{out} = (i1 + i2) + (i3 + i4) + (i5 + i6)$$



- Esta descripción genera un camino crítico de 3 sumadores.

Lenguajes de Descripción de Hardware

EJEMPLO : SÍNTESIS DE UN MULTIPLICADOR DE DOS BITS

- El multiplicador tendrá dos entradas de dos bits y una salida de cuatro.

```

entity multiplicador is
port ( a0 : in bit ;
       a1 : in bit ;
       b0 : in bit ;
       b1 : in bit ;
       c1 : out bit ;
       c2 : out bit ;
       c3 : out bit ;
       c4 : out bit ) ;
end multiplicador ;
  
```

a0	a1	b0	b1		c1	c2	c3	c4
0	0	0	0		0	0	0	0
0	0	0	1		0	0	0	0
0	0	1	0		0	0	0	0
0	0	1	1		0	0	0	0
0	1	0	0		0	0	0	0
0	1	0	1		0	0	0	1
0	1	1	0		0	0	1	0
0	1	1	1		0	0	1	1
1	0	0	0		0	0	0	0
1	0	0	1		0	0	1	0
1	0	1	0		0	1	0	0
1	0	1	1		0	1	1	0
1	1	0	0		0	0	0	0
1	1	0	1		0	0	1	1
1	1	1	0		0	1	1	0
1	1	1	1		1	0	0	1

Lenguajes de Descripción de Hardware

EJEMPLO : SÍNTESIS DE UN MULTIPLICADOR DE DOS BITS

- La representación más compacta se obtiene de los mapas de Karnaugh

Código VHDL utilizando la tabla de función.

```
architecture RTL1 of multiplicador is
    signal a_b : bit vector ( 3 downto 0 ) ;
begin
    a_b <= a1 & a0 & b1 & b0 ;
    process ( a_b )
    begin
        case a_b is
            when "0000" => ( c3,c2,c1,c0 ) <="0000"
            when "0001" => ( c3,c2,c1,c0 ) <="0000"
            when "0010" => ( c3,c2,c1,c0 ) <="0000"
            when "0011" => ( c3,c2,c1,c0 ) <="0000"
            when "0100" => ( c3,c2,c1,c0 ) <="0000"
            when "0101" => ( c3,c2,c1,c0 ) <="0001"
            ...
            when "1111" => ( c3,c2,c1,c0 ) <="1001"
        end case ;
    end process ;
end RTL1 ;
```

Lenguajes de Descripción de Hardware

EJEMPLO : SÍNTESIS DE UN MULTIPLICADOR DE DOS BITS

- Código VHDL utilizando los minitérminos

```
architecture RTL2 of multiplicador is
begin
    c0 <= a0 and b0

    c1 <=(a0 and not a1 and b1)or
        (a0 and not b0 and b1) or
        (not a0 and a1 and b0) or
        (a1 and b0 and not b1) ;

    c2 <=(a1 and b1 and not b0) or
        (a1 and not a0 and b1) ;

    c3 <= a1 and a0 and b1 and b0 ;
end RTL2 ;
```

Lenguajes de Descripción de Hardware

EJEMPLO : SÍNTESIS DE UN MULTIPLICADOR DE DOS BITS

- Realización con enteros

```
library IEEE ;
use IEEE.numeric_bit.all ;

architecture RTL3 of multiplicador is
signal a_vec : unsigned ( 1 downto 0 );
signal b_vec : unsigned ( 1 downto 0 );
signal a_int : integer range 0 to 3 ;
signal b_int : integer range 0 to 3 ;
signal c_vec : unsigned ( 3 downto 0 ) ;
signal c_int : integer range 0 to 9 ;

begin
a_vec <= a1 & a0 ;
a_int <= to_integer ( a_vec ) ;
b_vec <= b1 & b0 ;
b_int <= to_integer ( b_vec ) ;
c_int <= a_int * b_int ;
c_vec <= to_unsigned ( c_int, 4 );

( c3, c2, c1, c0 ) <= c_vec ;
end RTL3 ;
```

- El paquete numeric_bit provee todas las funciones necesarias para convertir vectores de bits a valores enteros y viceversa.
- Se utilizan señales internas para generar vectores de bits y representaciones enteras de las señales de puertos. Los vectores de bits serán tratados como valores binarios no signados.
- Las señales de entrada tipo bit único son concatenadas para convertirlas en vectores y posteriormente convertirlas nuevamente en tipos de datos enteros.
- La multiplicación es realizada utilizando el operador estándar VHDL.
- Se debe especificar el tamaño del vector objetivo cuando se convierten nuevamente los enteros a vectores de bits.
- Finalmente los elementos del vector de bits se asignan a los puertos de salida.

Lenguajes de Descripción de Hardware