1 Reglas de estilo

En esta sección se van a presentar una serie de reglas de codificación de obligado cumplimiento en las prácticas de la asignatura. Los profesores del laboratorio vigilarán y evaluarán que el código entregado satisface estas reglas.

1.1 Estilo

Reg 1. Todo proceso debe ir acompañado de un comentario que describa su funcionalidad, entradas y salidas.

Reg 2. Todo el código debe ir en letras minúsculas.

Reg 3. No usar tabuladores para sangrar el código. En su lugar utilizar tres espacios. En casi todos los editores de texto se puede configurar que la tabulación se realice insertando un número arbitrario espacios.

Reg 4. Ninguna línea de código VHDL debe exceder los 80 caracteres.

1.2 Nomenclatura

Reg 1. El nombre de la señal de reloj siempre será clk.

Reg 2. Si en el diseño hay varias señales de reloj todas ellas deberán comenzar con el sufijo clk_ seguido de un nombre representativo del reloj. Por ejemplo, si el diseño posee dos relojes de 50 MHz y 25 MHz entonces sus nombres deberían ser: clk_50mhz y clk_25mhz.

Reg 3. El nombre de la señal de reset siempre será rst.

Reg 4. Si en el diseño hay varias señales de reset entonces todas ellas comenzaron con el sufijo rst_ seguido de un nombre representativo del reset. Por ejemplo, si hay una señal de reset síncrono y otra señal de reset asincrona entonces sus nombres podrían ser rst_sync y rst_async.

Reg 5. Cada definición de una señal debe ir en una línea separada. En la misma linea debe incluirse un comentario descriptivo de la señal.

Reg 6. Si un objeto tiene más de 1 bit entonces el bit msb debe ir primero y tener un valor igual o superior al bit lsb.

Reg 7. Los nombres de los proceso deberán comenzar con el prefijo p_.

Reg 8. Los nombres de los estados deben finalizar con el sufijo _st

Reg 9. Los nombres de las instancias deben comenzar con el prefijo i_

Reg 10. Los nombres de las señales activas a nivel bajo deben finalizar con el sufijo _n

Reg 11. Los nombres de los registros deben finalizar con el sufijo _reg

Reg 12. Los nombres de los genericos deben comenzar con el prefijo g_

1.3 Codificación procesos

Reg 1. En un mismo proceso no puede definirse lógica combinacional y lógica secuencial.

Reg 2. La lista de sensibilidad de un proceso combinacional debe contener todas las señales leidas en el proceso.

Reg 3. La lista de sensibilidad de un proceso secuencial sólo debe contener la señal de reloj y la señal de reset.

Reg 4. En un único proceso no deben definirse lógica secuencial con dos relojes distintos.

Reg 5. Todos los archivos de un diseño deben contener comentarios. Siempre que sea posible hay que incluir una breve descripcion de cada proceso o bloque combinacional encima del código en cuestión. Comentarios adicionales deberán aparecer junto a las líneas de código en la que se esté llevando a cabo funciones importantes a fin de proporcionar información adicional al diseñador.

Reg 6. Todos los diseños deben ser síncronos. El diseño síncrono por lo general reduce el esfuerzo de satisfacer los requisito de temporización.

Reg 7. Todos los elementos de memoria en un diseño deben ser síncronos.

Reg 8. El uso de latches está terminantemente prohibido

1.4 Instancias

Reg 1. La asignación de señales en una instancia debe hacerse de forma explícita. Está terminantemente prohibido utilizar la asignación implícita.

Reg 2. Cada asignación deberá ir en una línea separada.

Reg 3. Todos las asignaciones de una instancia deben ir acompañados de un comentario que indica su función.

1.5 Organización del código

Reg 1. En cada archivo sólo el estará contenida la definición de una entidad y su arquitectura

Reg 2. El nombre del archivo debe coincidir con el nombre de la entidad declarada del archivo.

GUÍA DE BUENAS PRÁCTICAS PARA LA SÍNTESIS DE CIRCUITOS DESCRITOS CON VHDL

Evita ser demasiado creativo en la utilización del lenguaje VHDL. Debemos ser innovadores a la hora de diseñar el hardware pero no a la hora de describirlo.

- Sentencias concurrentes:
  - Evitar lazos cerrados de realimentación (a <= a+b;)
  - Piensa en las sentencias concurrentes (when-else y with-select), así como en las asignaciones, como una estructura de cableado más que como una estructura secuencial
  - Una asignación concurrente genera una ruta con prioridad, un gran número de when-else con condiciones muy dispares produce una cadena de multiplexores y lógica de gran tamaño.
- Sentencias dentro de un process:
  - Las variables deben ser usada con mucho cuidado, siempre que se pueda es preferible utilizar una señal (signal).
  - Excepto para el valor de la señal por defecto (el que se escribe al principio del process) debe evitarse la sobreasignación de un señal en distintos puntos del process.
  - Piensa en las sentencias if-else como una estructura de cableado más que como una estructura secuencial
  - Una estructura if-else o case genera una ruta con prioridad, un gran número de condiciones muy dispares y produce una cadena de multiplexores y lógica de gran tamaño.
  - Piensa en un loop como una manera fácil de describir una ruta, una estructura loop siempre se va a desenrollar por completo en el proceso de síntesis y tiene que ser independiente de los valores de entrada al circuito.
- Lógica combinacional dentro de un process:
  - Conviene añadir todas las señales de entrada en la lista de sensibilidad para evitar comportamientos inesperados.
  - Toda sentencia if tiene que acabar con un else, para que no se produzcan latches
  - Todas las señales de salida tienen que tener asignado un valor en cada rama if-else para evitar la aparición de latches. Es por lo tanto una buena práctica de diseño asignar un valor por defecto a las señales de salida al principio del process