18/04/2017

Práctica 1

MODELOS VHDL DE BIESTABLES Y DE REGISTROS ENTRADA PARALELO-SALIDA PARALELO PARA SU SIMULACIÓN

RAÚL CARBAJOSA GONZÁLEZ. Y160311.

MÁXIMO GARCÍA MARTÍNEZ. Y160242.

Esta práctica tiene dos partes:

1. Modelado de un biestable tipo de activado por flanco de subida de la señal de reloj y señal clear asíncrona activa a nivel alto.
2. Modelado de un registro con entrada en paralelo y salida en paralelo.

**Parte 1: Modelado de un biestable tipo de activado por flanco de subida de la señal de reloj y señal clear asíncrona activa a nivel alto.**

Consiste en la modelización en VHDL de un biestable tipo D, utilizando como arquitectura flujo de datos. Tendrá como señales:

* Entrada D
* Entrada de reloj: CLK
* Entrada de clear asíncrona: CLR
* Salida Q

El objetivo de esta primera práctica es desarrollar el programa de [*biD\_grupo22.vhd*](https://github.com/onmax/sistemasDigitales/blob/master/practica2/biD_grupo22.vhd), un biestable activado por flanco de subida de la señal reloj y por una señal clear asíncrona activa a alto nivel.

Para ello, primeramente, debemos escribir el código. Para ello tenemos dos archivos. Uno de ellos llamado “[*biD\_grupo22*](https://github.com/onmax/sistemasDigitales/blob/master/practica2/biD_grupo22.vhd)*.vhd*”, donde hemos escrito el código principal. Primeramente, importamos la librería “ieee” que contiene definiciones estándar para VHD. Luego establecemos la entidad (“biestable\_D\_con\_Clr”), esta parte del código sirve para declarar las entradas y salidas. Por último, tenemos la arquitectura (“*flujo\_asin*”) donde establecemos una arquitectura de flujo (Figura 1). El otro archivo, “[*biD\_grupo22*](https://github.com/onmax/sistemasDigitales/blob/master/practica2/biD_grupo22.vhd)*\_tb.vhd*”, sirve para establecer el test que queremos llevar a cabo. En este caso, el ejercicio nos pedía establecer una simulación de 50ns, por lo que nosotros hemos decidido hacer un test que cada 25ns, cambiase de estado. El test empieza con el estado “1” y cada 10ns cambia hasta llegar a los 50ns. Se puede ver los diferentes estados que hemos establecido para el test en la Figura 3.

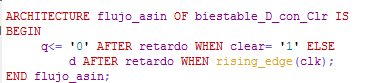


Figura 1: Arquitectura de [biD\_grupo22](https://github.com/onmax/sistemasDigitales/blob/master/practica2/biD_grupo22.vhd).vhd

Una vez escrito todo ello, ejecutamos la simulación con el comando “run 50 ns” y podemos ver lo siguiente:

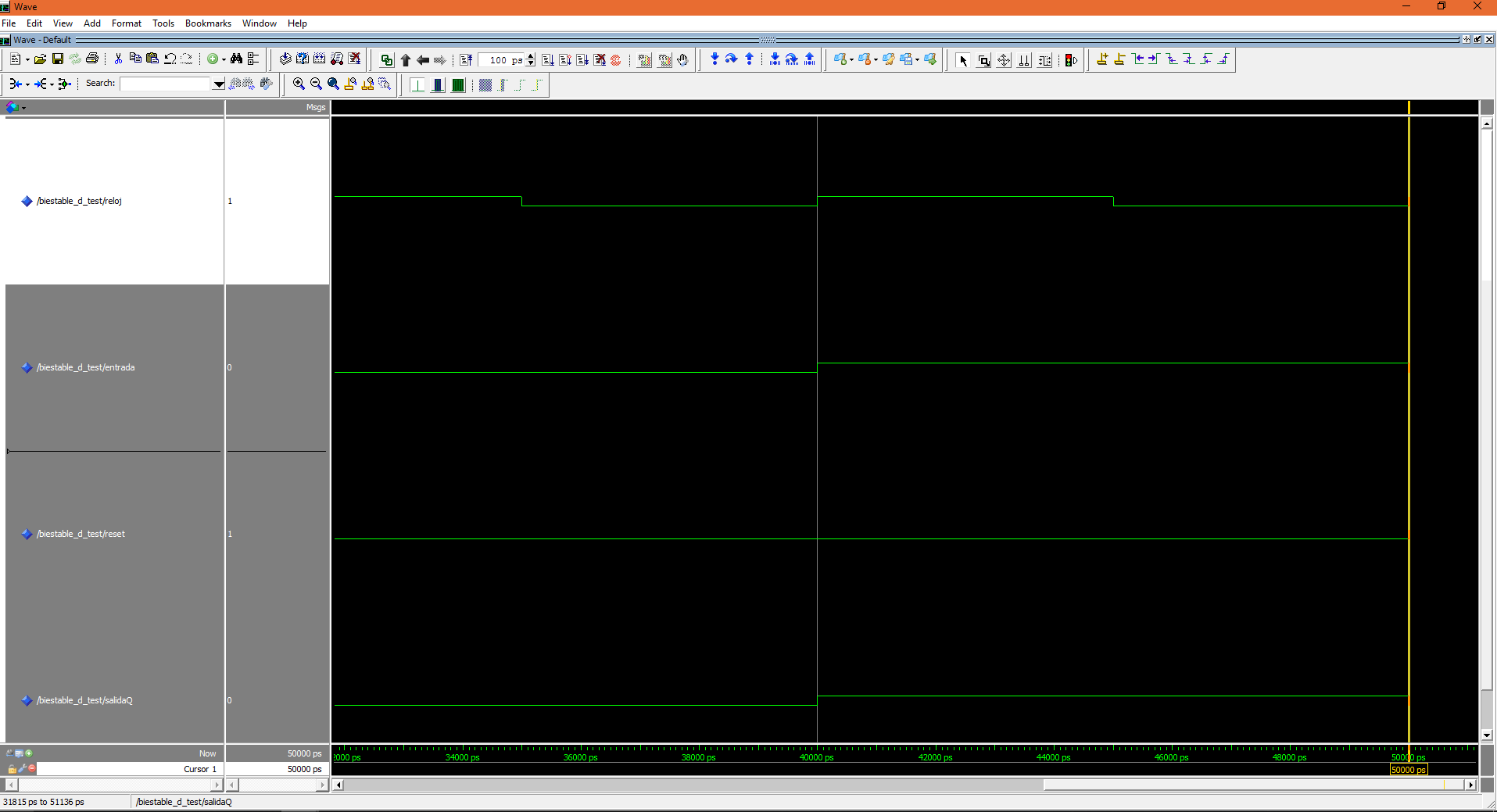


Figura 2. El orden de las entradas de arriba a abajo: Reloj, entrada, reset y salidaQ

En la gráfica de la simulación podemos observar como varían las señales *“reloj”, “entrada”* y *“salidaQ”.*

Las entradas van cambiando según pasa el tiempo según hemos establecido en el test “[*biD\_grupo22*](https://github.com/onmax/sistemasDigitales/blob/master/practica2/biD_grupo22.vhd)*\_tb.vhd*”. La entrada reloj cambia de valor cada 5 ns, el valor de *entradaD*, cambia cada 10 ns, *reset* a los 25 ns y la *salidaQ* cambia llegados los 40 ns. El código empleado para realizar el test se puede ver a continuación.

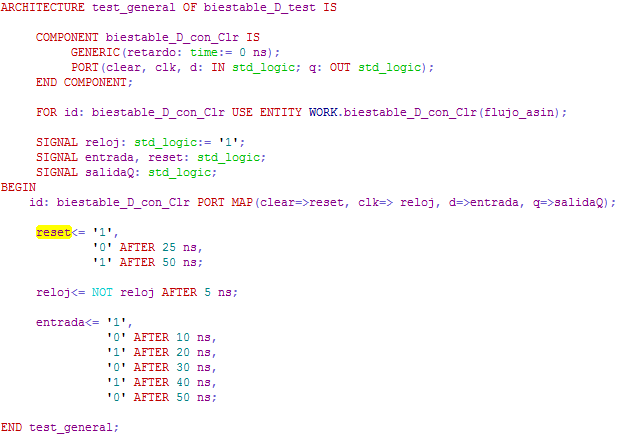


Figura 3: Arquitectura del test en [biD\_grupo22](https://github.com/onmax/sistemasDigitales/blob/master/practica2/biD_grupo22.vhd)\_tb.vhd

La señal de salida “*salidaQ*” que depende del valor “*clear*” tiene el valor “1” durante los últimos 40 ns porque coincide que tanto el “*reset*” tiene un valor igual a 1 y que el valor de entrada es ‘0’.

**Parte 2: Modelado de un registro con entrada en paralelo y salida en paralelo.**

En esta segunda parte de la práctica se realizará un modelo VHDL de un registro con entrada paralelo y salida paralela de 5 bits con arquitectura en estilo estructural, utilizando el modelo de biestable desarrollado en la primera parte.

Las señales de esta segunda parte serán :

* Entradas de datos: *ent\_datos*
* Entrada de reloj: *clock*
* Entrada de clear asíncrona: *clear*
* Salidas de datos: *sal\_datos*

La entrada *ent\_datos* y la salida son de tipo *std\_logic\_vector* mientras que las entradas *clock* y *clear* son de tipo *std\_logic*.

En esta parte tendremos que desarrollar una entidad (*registro\_PP*) y una arquitectura de tipo estructural (*con\_generate*), ambas las tendremos que incluir en un mismo archivo llamado *RPP\_gene\_grupo22.vhd*. La arquitectura *con\_generate* es la siguiente:

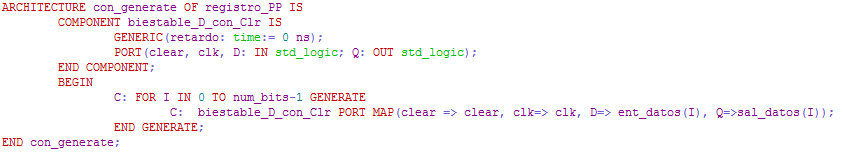


Figura 4: Arquitectura de [biD\_grupo](https://github.com/onmax/sistemasDigitales/blob/master/practica2/biD_grupo22.vhd)22.vhd

Después tenemos que desarrollar el test, donde se especificará el funcionamiento de la simulación. Dicho código estará escrito en un archivo llamado *RPP\_gene\_grupo22\_tb.vhd,* el cual, tendrá una entidad (*regPPtest*) y una arquitectura llamada *regPPtestArch* que se puede ver en la Figura 5. La simulación se llevará a cabo durante 100 ns, donde la señal de *reset* cambia de valor de alto nivel a bajo nivel y viceversa cada 25 ns y la entrada cada 10 ns hasta que termina la simulación. El código usado para el test se puede visualizar a continuación.

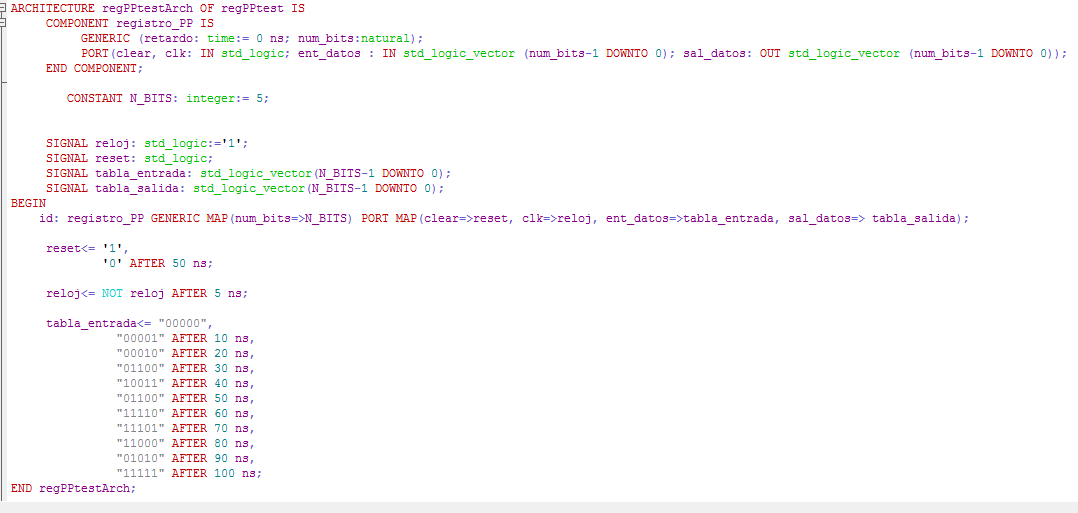


Figura 5: Arquitectura del test en [biD\_grupo22](https://github.com/onmax/sistemasDigitales/blob/master/practica2/biD_grupo22.vhd)\_tb.vhd

La gráfica del test se puede ver a continuación:

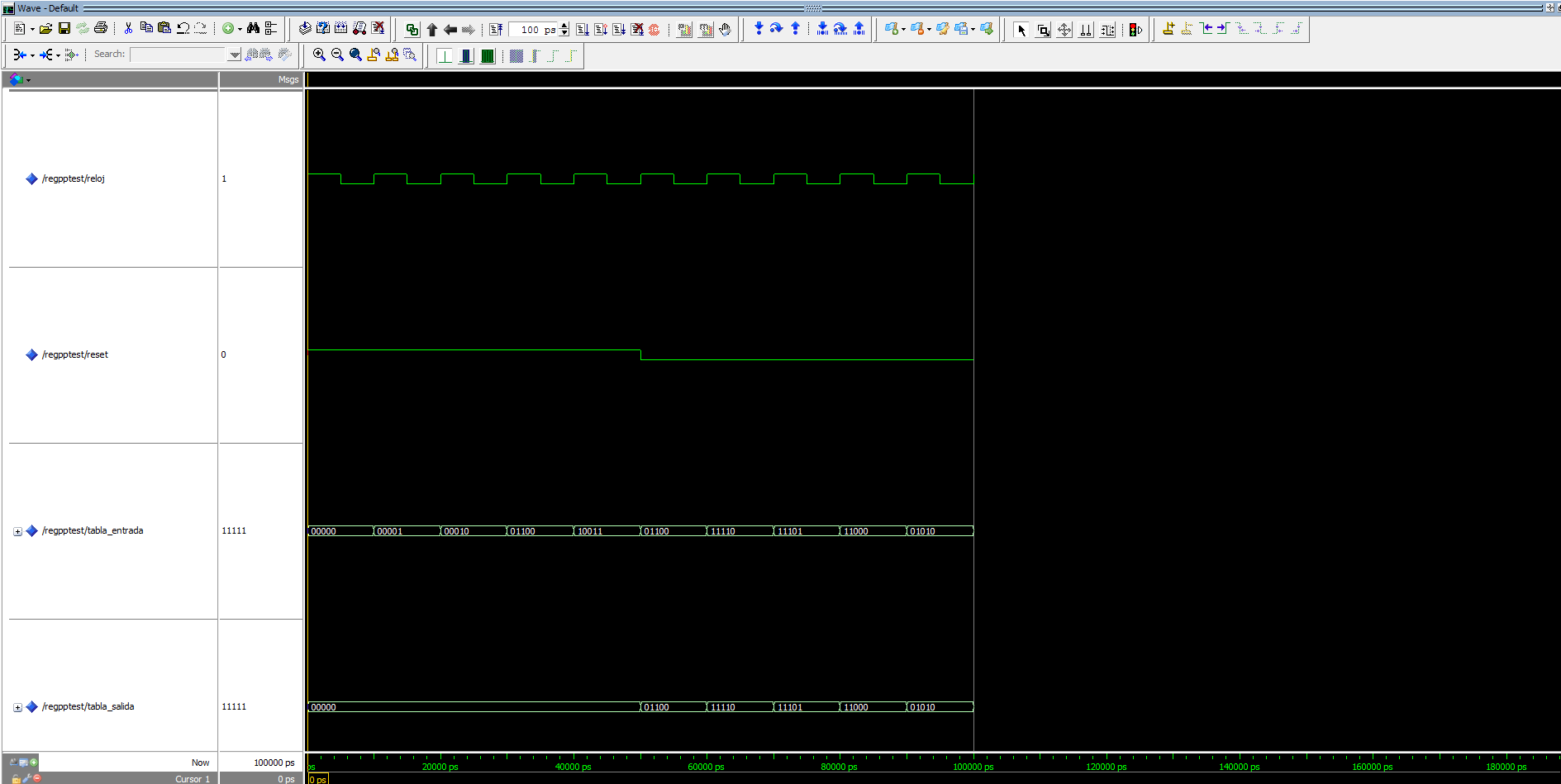


Figura 6: El orden de las entradas de arriba a abajo: reloj, reset, tabla\_entrada y tabla\_salida

En la gráfica observamos cómo va cambiando el valor de las variables: *reloj* cambia cada poco nanosegundos mientras que *reset* cambia de valor pasado 50 ns. También se puede observar cómo cambian los valores de las variables tipo *std\_logic\_vector* (*tabla\_entrada* y *tabla\_salida*) según hemos indicado nosotros en el test.

Por último, mostramos el contenido de la carpeta WORK que es la misma para las dos partes de la práctica:

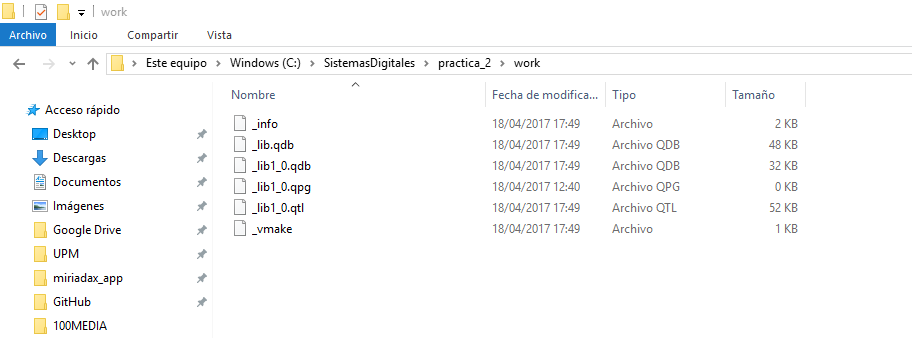


Figura 7: Captura de pantalla de la carpeta WORK usada para el proyecto.