PRÁCTICA 1. Introducción al Software Xilinx ISE versión 6.

1. Introducción.

Debido a los requerimientos de funcionamiento, la complejidad que están alcanzando los diseños digitales aumenta día a día. Estos diseños implementan multitud de funciones, las cuales deben realizarse de forma rápida y precisa, por lo que resulta necesario el empleo de circuitos integrados digitales programables con mayor complejidad que los ampliamente difundidos PLD (Programmable Logic Device). Los dispositivos PLD ofrecen una respuesta muy rápida, pero están fuertemente restringidos por su capacidad, y principalmente por su estructura, ya que está completamente fijada y ha de ser nuestro diseño el que se adapte al PLD. Otra posibilidad de implementación la podemos encontrar en los circuitos semi-custom. Estos circuitos permiten la implementación de cualquier sistema digital por complejo que sea, pero tienen algunas desventajas: En primer lugar nos encontramos con un coste elevado, únicamente rentabilizado si el volumen de producción es alto, en segundo lugar, el tiempo de desarrollo es largo (un diseño puede tardar varios meses en comercializarse), esto hace que no resulte interesante debido a la pérdida de competitividad en un mercado que innova constantemente.

Con los dispositivos FPGA (Field Programmable Gate Array) se ha alcanzado un punto intermedio entre los dispositivos PLD y los circuitos semi-custom. Un FPGA puede alcanzar una alta densidad de integración en un solo circuito integrado (hasta 1.000.000 de puertas lógicas equivalentes aproximadamente, y cada año aumenta su complejidad), y con velocidades de tratamiento de la información de entrada muy altas; cada día se continua mejorando la eficiencia de estos dispositivos tanto en velocidad como en complejidad. Aunque su estructura está completamente fijada a nivel de silicio, la flexibilidad en su programación es grande ya que un FPGA está formado por células independientes que se pueden programar para realizar funciones sencillas, pero debido a los amplios recursos de interconexión de que disponen, estas células se pueden conectar entre ellas para generar unas funciones lógicas de salida complejas. La programación de este tipo de dispositivos se realiza en cuestión de minutos, con lo que se hacen altamente recomendables para prototipos o bajas producciones. Los problemas de un FPGA frente a la tecnología semi-custom son principalmente la menor velocidad y optimización del chip, pues a menudo, en un FPGA quedan recursos sin utilizar debido a la falta de canales de conexión; además, los canales introducen retardos en la señal, y a veces es necesario un canal de gran longitud (mayor retardo) para interconectar las células requeridas. Todo esto hace que el dispositivo pierda eficiencia, pero estos factores se ven subsanados por la rapidez con la que pueden ser reprogramados, adaptándose rápidamente a las necesidades del mercado. Actualmente, los dispositivos FPGA son empleados en todo tipo de aplicaciones tanto científicas como de consumo; por ejemplo, actualmente se emplean en reproductores de CD, tarjetas de expansión para PC's, dispositivos para telecomunicaciones, tareas de control de dispositivos, etc.

Si el número de unidades a fabricar está por debajo de 10.000, la fabricación con tecnología semicustom no resulta rentable, y la mejor opción es un FPGA.

Se puede pensar que si el número de unidades es bajo, podría emplearse lógica estándar, pero el tamaño total del circuito y el encarecimiento de la corrección de errores en una placa de circuito impreso vuelven a hacer aconsejable el empleo de FPGA. Si nuestro diseño se desea implementar con los métodos tradicionales, después de las especificaciones se han de separar diferentes bloques funcionales, diseñando cada uno de ellos por separado para posteriormente integrarlos todos en una placa prototipo, esto provoca que a menudo se produzcan "cuellos de botella" que habrá que corregir. Mientras se fabrican las placas definitivas, pueden sugerirse nuevos cambios que repercutirán en alteraciones de la placa, lo que provocará un retraso en el lanzamiento del producto y un mayor gasto económico, por desgracia, estas alteraciones en el diseño son usuales. El diseño completo puede llevar de 6 a 12 meses para diseños que necesiten de 500 a 1500 circuitos integrados de lógica estándar, y si se incluyen las alteraciones de la placa puede llegarse a un tiempo total de 2 años. Una vez implementado el diseño correctamente, para reducir costes se puede pasar a implementarlo en tecnología semicustom, lo que requeriría otro año más.

Actualmente, un nuevo sistema de diseño tiende a implantarse. Una vez hechas las especificaciones, el sistema es dividido en grandes bloques como pueden ser memorias, microprocesador, PLD y FPGA, se realiza una descripción de alto nivel a través de un esquema o lenguaje de descripción lógica para posteriormente simular el diseño, cuando la simulación es correcta se realiza el diseño de la placa que contendrá los diferentes elementos, mientras la placa está siendo fabricada, se depura el diseño, pero en la mayoría de los casos esto no afecta a la placa sino a los circuitos programables incluidos en ella.

El entorno de diseño Xilinx ISE consiste en una herramienta que permite realizar un diseño completo basado en lógica programable (tanto CPLD como FPGA), es decir, incluye todas las etapas necesarias como son:

- La entrada de diseño, bien a través de captura esquemática, lenguajes de descripción hardware como ABEL, VHDL o Verilog, o representación gráfica de diagramas de estado (***StateCAD/State Bencher***).
- Herramientas de verificación para la obtención de una simulación del sistema, tanto a nivel funcional como de estimación de retardos. La herramienta empleada se denomina ModelsimXE. Por otro lado, también se facilita la generación de bancos de prueba para la verificación mediante la herramienta ***HDL Bencher.***
- Herramientas de implementación donde se permite la especificación de restricciones o indicaciones para realizar una implementación óptima sobre el dispositivo lógico programable especificado. Esta herramienta incluye tres etapas principales en el diseño: ***Translate, Map, Place & Route.***
- Herramientas de programación, para permitir descargar el diseño sobre el dispositivo físico, ya sea en una placa de evaluación o bien en la placa definitiva mediante la programación in-situ (en sistema) a través de la programación JTAG. De este modo, es posible probar y depurar el sistema sobre el hardware de forma rápida y flexible, permitiendo tantos cambios como sean necesarios.

En la figura 1 se muestra de forma global los procesos que se llevan a cabo en durante el diseño de sistemas basados en lógica programable.

Este entorno permite combinar las diferentes técnicas de diseño para facilitar la labor de descripción del diseño. Además, se permite la inclusión de restricciones para optimizar el proceso de implementación y adaptarlo a las necesidades del diseño, como ejemplo, inclusión de restricciones temporales para determinadas señales, restricciones de ubicación de la lógica en una determinada zona del array programable, o bien inclusión de opciones de particularización en elementos hardware como asignación de patillas, líneas de reloj específicas, etc.

Por otro lado, el conocimiento de este tipo de entornos permite ser capaces de emplearlo en multitud de diseños, ya que independientemente de la complejidad del diseño, si éste está destinado a un dispositivo programable, ya sea CPLD o FPGA, será posible realizarlo mediante el mismo software, por lo que una vez dominado, el proceso de diseño de nuevos sistemas resulta mucho más rápido.

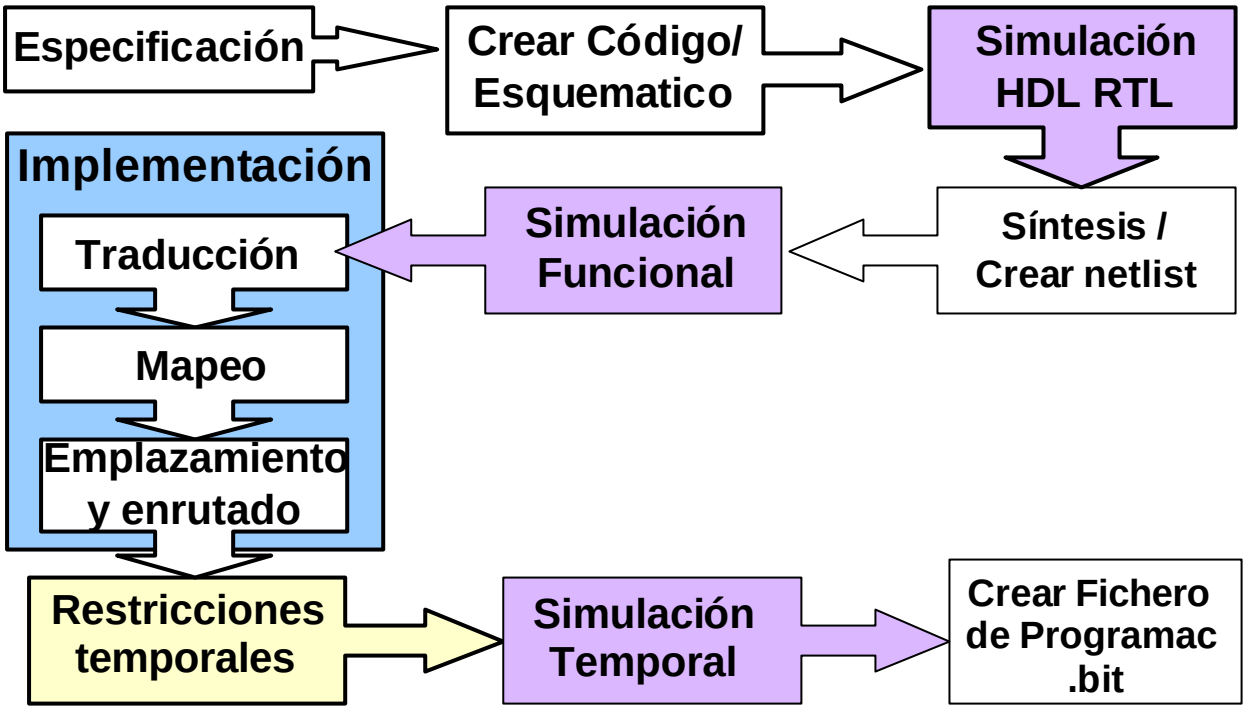

Figura 1. Diagrama de evolución de los procesos involucrados en el diseño de sistemas basados en lógica programable.

2. Descripción del entorno de desarrollo.

El entorno de desarrollo ISE de Xilinx posee un aspecto similar al de los entornos de programación actuales como puede ser Visual Basic o Visual C, es decir, posee diversas ventanas para visualización de tareas específicas sobre cada una de ellas. En este caso existen cuatro tipos de ventanas (figura 3):

1. Ventana de ficheros fuente. En esta ventana se muestran los ficheros fuente utilizados en el diseño y las dependencias entre ellos. También es aquí donde se elige el tipo de dispositivo donde se desea implementar el diseño. Esta ventana posee diversas solapas para visualizar diferentes tipos de información relativa a las fuentes de diseño empleadas.
2. Ventana de Procesos. Esta ventana muestra todos los procesos necesarios para la ejecución de cada etapa de diseño. La lista de procesos se modifica dinámicamente dependiendo del tipo de fuente seleccionado en la ventana de ficheros fuente.
3. Ventanas de edición. Al hacer doble clic sobre un fichero fuente de la ventana de ficheros fuente se abre una ventana de edición para modificar el fichero (en caso de lenguaje HDL), o bien se ejecuta el programa que permite editar el diseño (en caso de diseños esquemáticos ó máquinas de estado).
4. Ventana de información, situada en la parte inferior. Muestra mensajes de error, aviso o información emitidos por la ejecución de los programas de compilación, implementación, etc.

Tanto en la ventana de procesos como en la de ficheros fuente es posible modificar las opciones de cada elemento a través del botón derecho del ratón, o bien a través de los menús del entorno de diseño; estos menús se modifican dependiendo del tipo de selección realizada en las ventanas de ficheros fuente y de procesos. Cada elemento mostrado en la ventana posee un icono diferente dependiendo del tipo de acción o fichero que se trate, en la ventana de procesos por ejemplo, nos indica si un elemento es un documento texto , una acción a ejecutar por el entorno ISE o una acción a ejecutar por un programa adicional como puede ser ModelSim a la hora de simular. También muestra información sobre el estado que ha resultado tras la ejecución del proceso, es decir, si ha sido satisfactorio , si ha tenido errores , o ha generado avisos . Las imágenes de la figura 2 muestran un ejemplo de los diferentes tipos de información mostrados en estas dos ventanas. Para la ventana de ficheros fuente, se indica si un fichero es de código , de vectores de test , si es un paquete , o una selección de dispositivo .

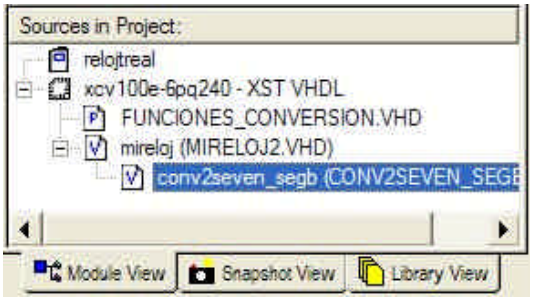

Figura 2. Detalle de las ventanas de ficheros fuente y de procesos.

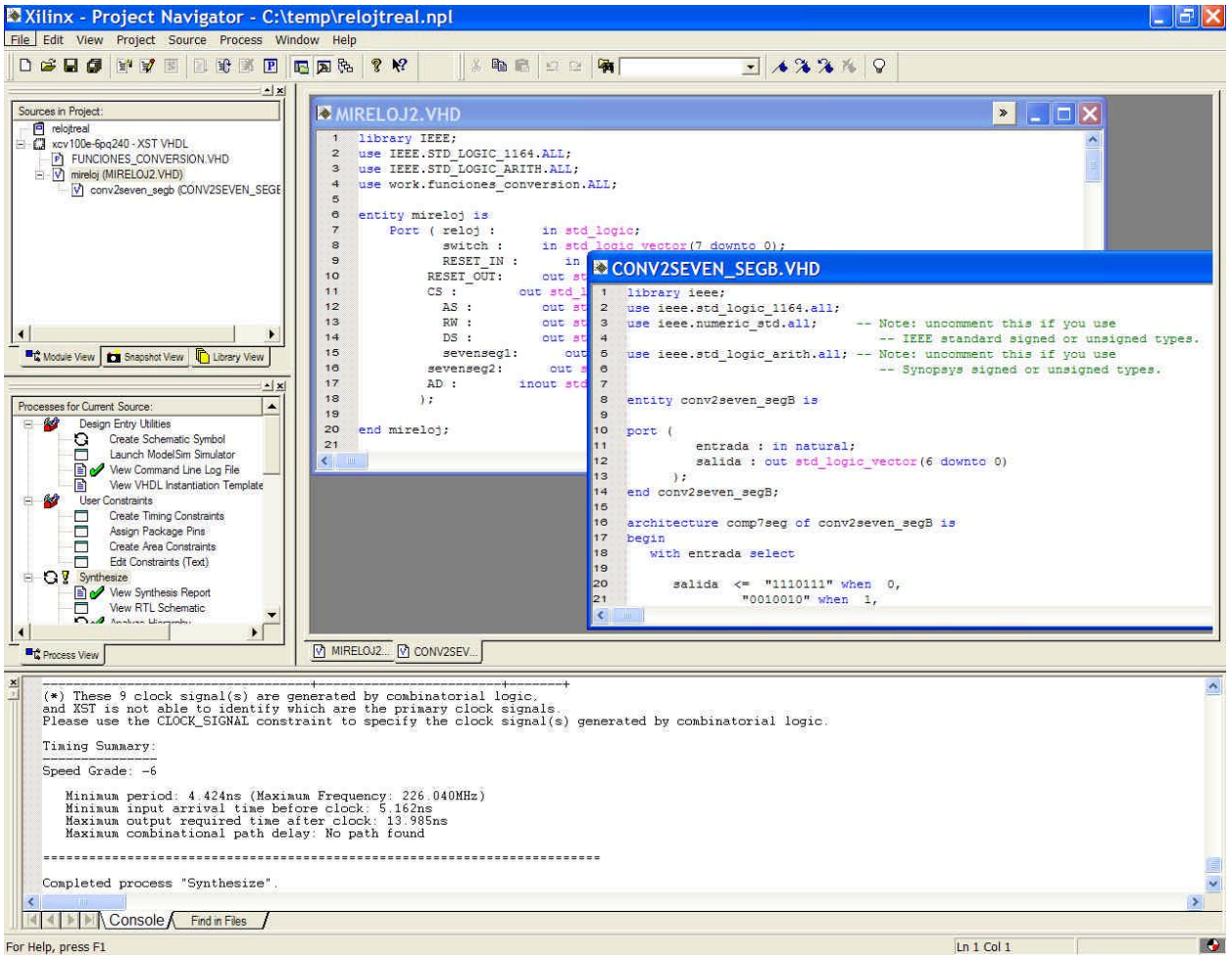

Figura 3. Pantalla principal del entorno Xilinx ISE.

En concreto, la ventana de procesos incorpora todas las opciones necesarias para realizar todos los pasos de implementación de sistemas en lógica programable, incluyendo la edición y verificación. La figura 4 muestra el diagrama de flujo de diseño en Xilinx ISE, y la figura 5 muestra las diversas partes en que se divide la ventana de procesos dependiendo de la tarea a realizar.

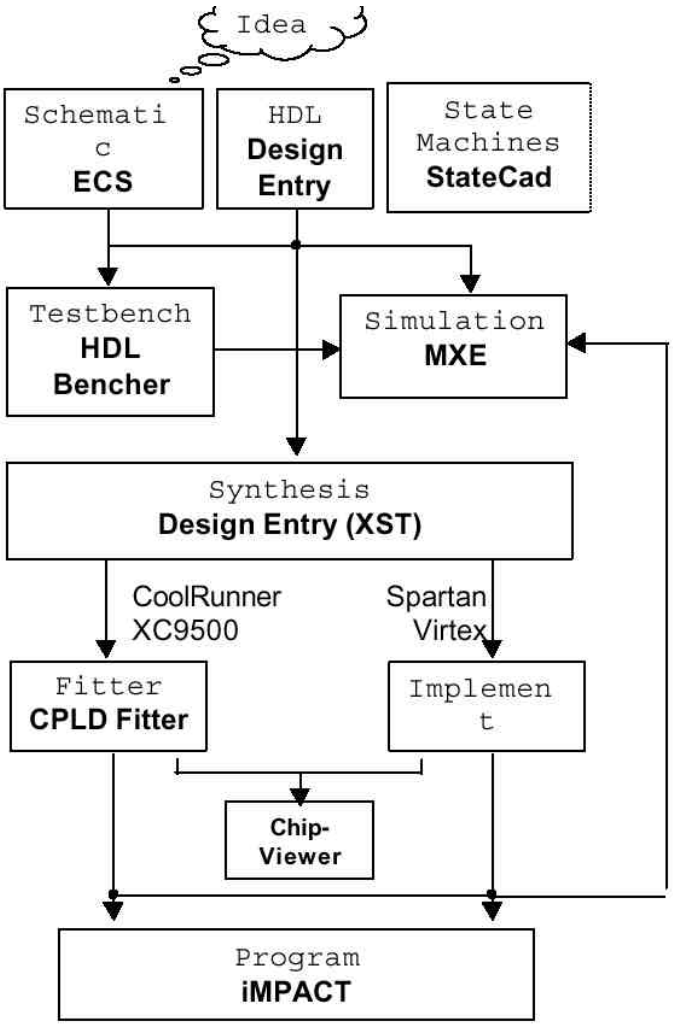

Figura 4. Proceso de diseño en Xilinx ISE.

Paso 1:Diseño

Paso 2: Síntesis

Paso 3: Implementación

Algunos procesos se muestran varias veces para indicar que se puede acceder a ellos desde diferentes etapas de diseño

Paso 4: Configuración

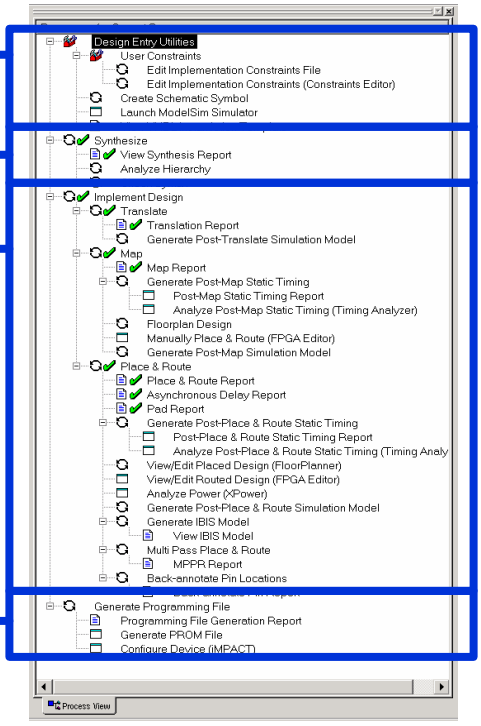

Figura 5. División de tareas dentro de la ventana de procesos.

De manera resumida, el proceso de diseño resulta sencillo y se realiza en tres pasos, el primer paso consiste en añadir los ficheros fuente, en el segundo paso se selecciona el fichero de más alto nivel que se quiere implementar, y finalmente se hace doble clic sobre el último proceso al que se desea llegar, de este modo se ejecutarán todos los procesos intermedios necesarios para llegar al proceso seleccionado en último lugar (figura 6).

Figura 6. Proceso simplificado para el desarrollo de un diseño en Xilinx ISE.

2. Objetivo de la práctica.

El objetivo de esta sesión de laboratorio consiste en realizar una primera aproximación al software Xilinx ISE a través de un diseño guiado para que el alumno conozca las diversas posibilidades que ofrece el programa y se puedan observar todas las etapas de diseño, incluyendo la descarga del programa realizado en una placa de evaluación. En esta primera práctica se analizarán las herramientas de entrada de diseño mediante esquemáticos (ECS) y mediante diagramas de estados (StateCAD).

En primer lugar, para acceder al software es necesario ejecutar el icono denominado

Project Navigator 

Al que también se accede desde: ***Inicio -> programas -> Xilinx ISE Webpack -> Project Navigator***

3. Creación de proyectos.

Cada vez que se desea realizar un nuevo diseño es necesario crear un proyecto nuevo: por cada proyecto se crea automáticamente una carpeta donde se almacenan todos los ficheros relacionados con dicho proyecto, no sólo los ficheros de entrada de diseño, sino todos los ficheros generados durante la ejecución de los pasos intermedios de compilación y verificación. La información relacionada con cada proyecto se guarda en un fichero con extensión ***.npl***.

Una vez que se abre el entorno de diseño, ya estamos en disposición de crear nuestros proyectos, para ello, ejecutamos **File -> New Project,** donde nos aparece la siguiente ventana:

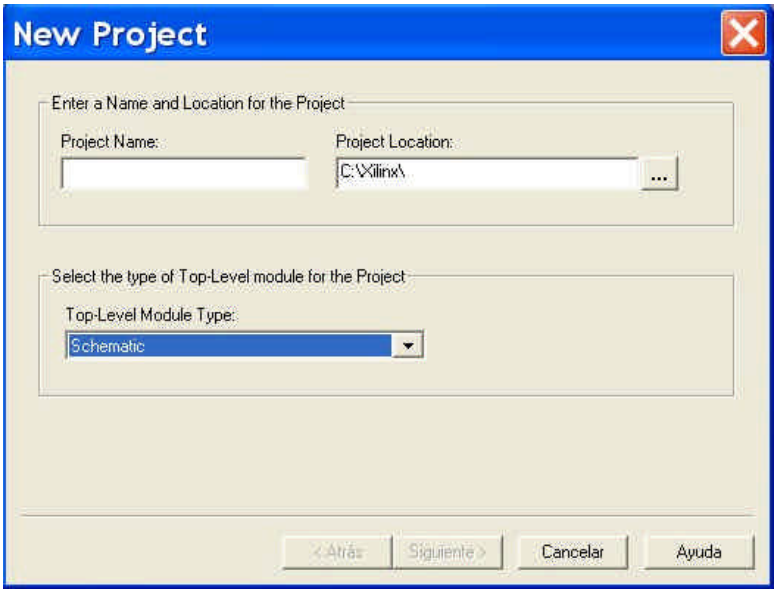

Figura 6. Ventana de diálogo para la creación de un nuevo proyecto Xilinx ISE.

En esta ventana se asigna un nombre al proyecto y se elige la carpeta de destino donde se almacenarán todos los ficheros relacionados con el proyecto. También se elegirá el tipo de módulo que será el de más alto nivel (en caso de diseño jerárquico), que en este caso será de tipo esquemático.

Seguidamente aparece una nueva ventana donde se debe seleccionar la familia de dispositivos sobre la que se va a realizar el diseño, el dispositivo en particular, el tipo de herramienta de síntesis, que en nuestro caso será siempre ***XST (VHDL/Verilog)*** y el simulador a emplear (ModelSim). La figura 7 muestra las opciones a seleccionar.

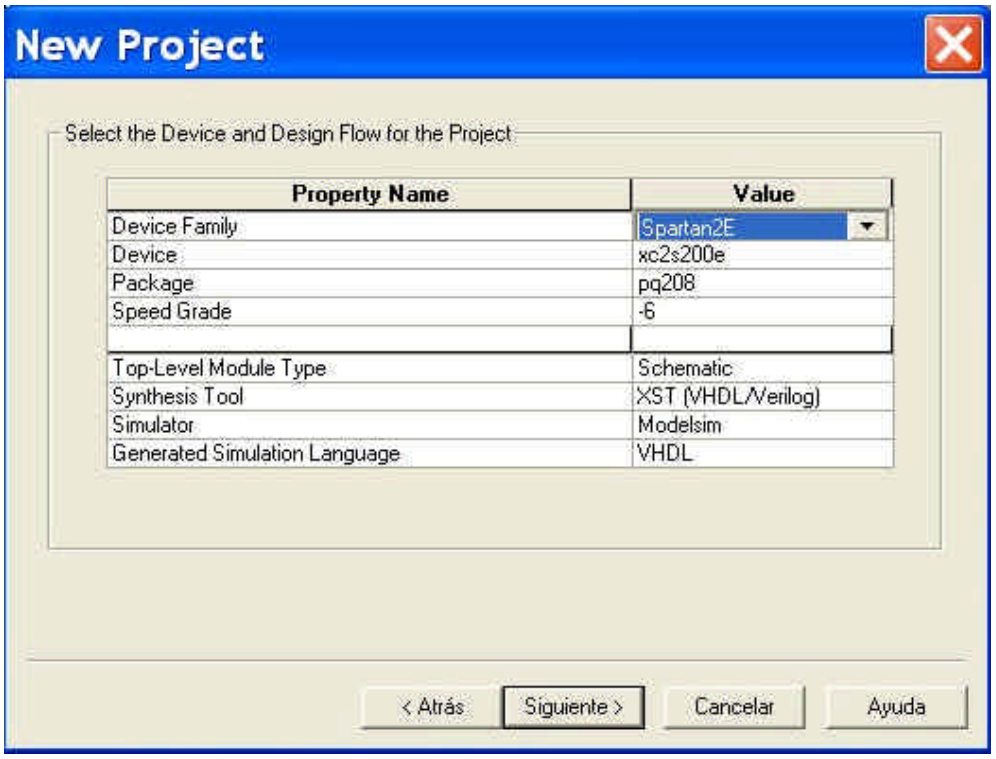

Figura 7. Ventana de diálogo para la selección de opciones en un nuevo proyecto Xilinx ISE.

4. Diseño esquemático.

El objetivo principal es conocer cómo crear y añadir nuevas hojas de esquemas en nuestro diseño, cómo emplazar y conectar elementos y componentes, y cómo editar componentes. Para crear una nueva hoja de

esquemas, se selecciona **Project** ? **New Source,** donde se elige **Schematic** y se le asigna un nombre y una carpeta de destino (generalmente la misma donde se ubica el proyecto), se verifica que la casilla de ***"Add to Project"*** está marcada y se pulsa ***Next*** hasta que se finalice el asistente, momento en el que automáticamente se abrirá el programa ECS de entrada de esquemas (podría ocurrir que se ECS apareciera oculto tras el Project Navigator).

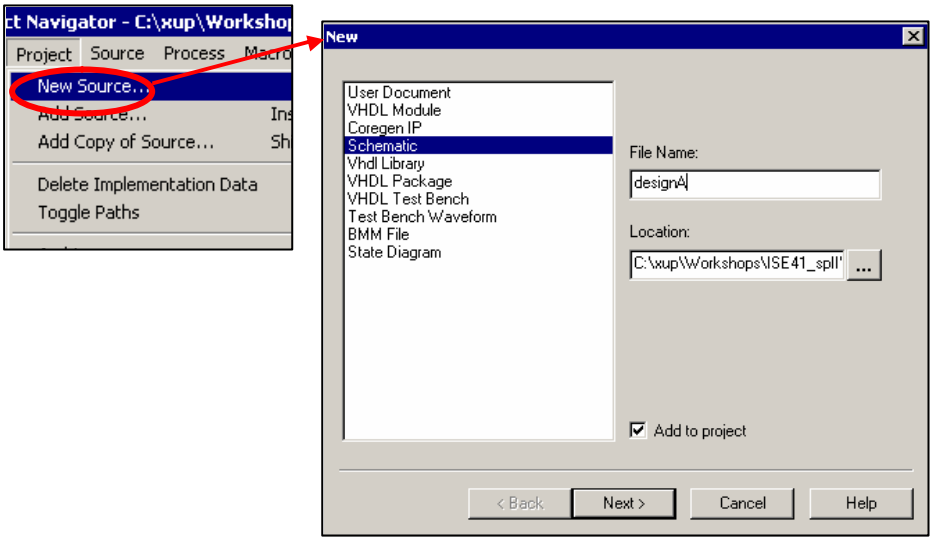

Figura 8. Proceso de creación de un nuevo fichero fuente para el proyecto.

Una vez en ECS (figura 9), vemos que tenemos en la parte izquierda dos solapas con los nombres ***Options*** y ***Symbols***. La solapa ***Options*** se particulariza dependiendo de la acción a realizar y permite asignar las propiedades deseadas a cada elemento empleado, y la solapa ***Symbols*** permite insertar todos aquellos elementos funcionales existentes en las librerías que el programa incluye además de los elementos de librería que uno mismo pueda crear.

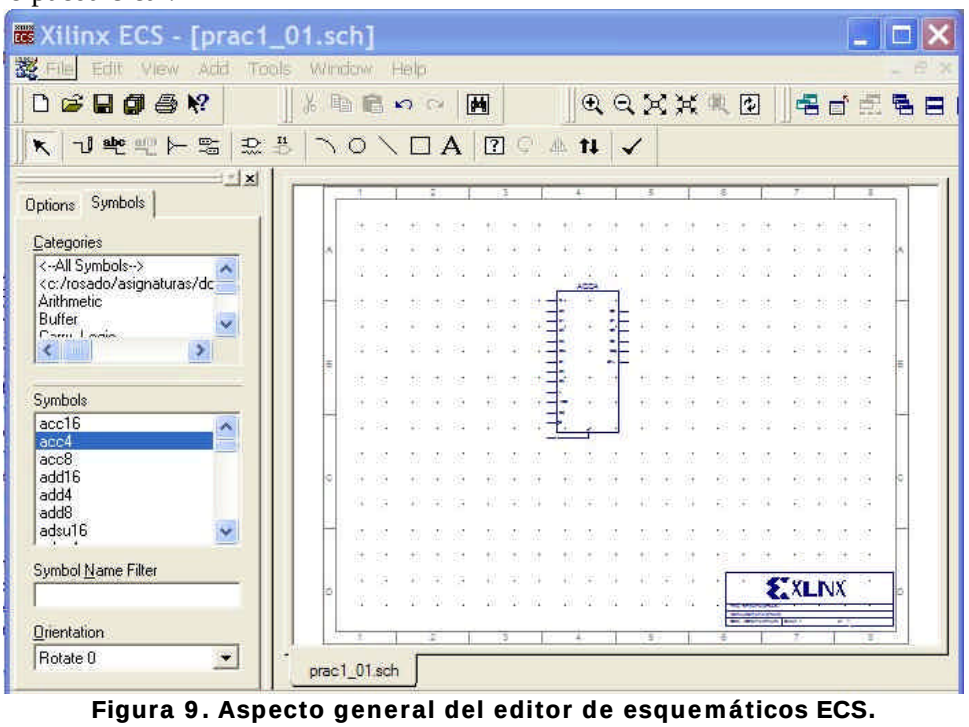

Figura 9. Aspecto general del editor de esquemáticos ECS.

Para realizar las opciones básicas de cableado (***Add Wire***), nombrado de líneas (***Add Net Name***), inclusión de uniones de bus (***Add Bus Tap***, sirve para individualizar las señales múltiples contenidas en un bus) e inclusión de terminales de entrada/salida (***Add I/O Marker***), se usan los iconos siguientes:

La única diferencia entre una línea sencilla y un bus es el nombre; ambas se trazan con ***Add Wire***, pero para nombrar un línea sencilla se le asigna un nombre cualquiera, y para nombrar un bus se debe poner un nombre y entre paréntesis el índice inicial y final del bus (por ejemplo, datos(3:0)). Es importante remarcar la necesidad de nombrar todas y cada una de las líneas de señales externas y todas aquellas que se separan de un bus. Para revisar posibles fallos en el esquema se utiliza **Tools -> Check Schematic**.

5. Actividades a realizar en el editor de esquemas.

5.1. Parte I.

En este caso, se pretende realizar un contador binario que cuente entre 0 y 59 en decimal. Las tareas a realizar son:

- Comprobar que el diseño de la figura 10 se corresponde con este contador (Nótese que los módulos CD4CE y CB4CE al llamarse de modo diferente, su funcionalidad es diferente).
- Implementar este circuito en ECS.
- Simular el diseño con ModelSim (ver apartado 6). Se verá que no funciona bien del todo. Proponed una solución.
- Simular el diseño con ModelSim, pero generando estímulos gráficos. Ver apartado 6.1.

5.2. Parte II.

Después de haberlo simulado el módulo individual, generar un elemento de librería para realizar un nuevo esquemático que contenga dos contadores de este tipo controlados por las mismas señales de entrada (reloj, reloj_ena y clear) (figura 11).

Para crear un nuevo elemento de librería, una vez completado el esquema, se cierra ECS y desde la ventana de procesos de ***Project Navigator,*** con el esquema de la ventana de fuentes seleccionado, se ejecuta la acción ***"Create Schematic Symbol***". Una vez finalizado el proceso, si todo es correcto, se crea un nuevo esquema dentro del proyecto y para insertar nuestro símbolo dentro de este nuevo esquema, en la solapa "Symbols" se marca la librería que tiene como nombre la carpeta del proyecto que actualmente tenemos, pudiendo seleccionar como elemento de esa librería el nuevo símbolo que acabamos de crear. Simular el nuevo sistema.

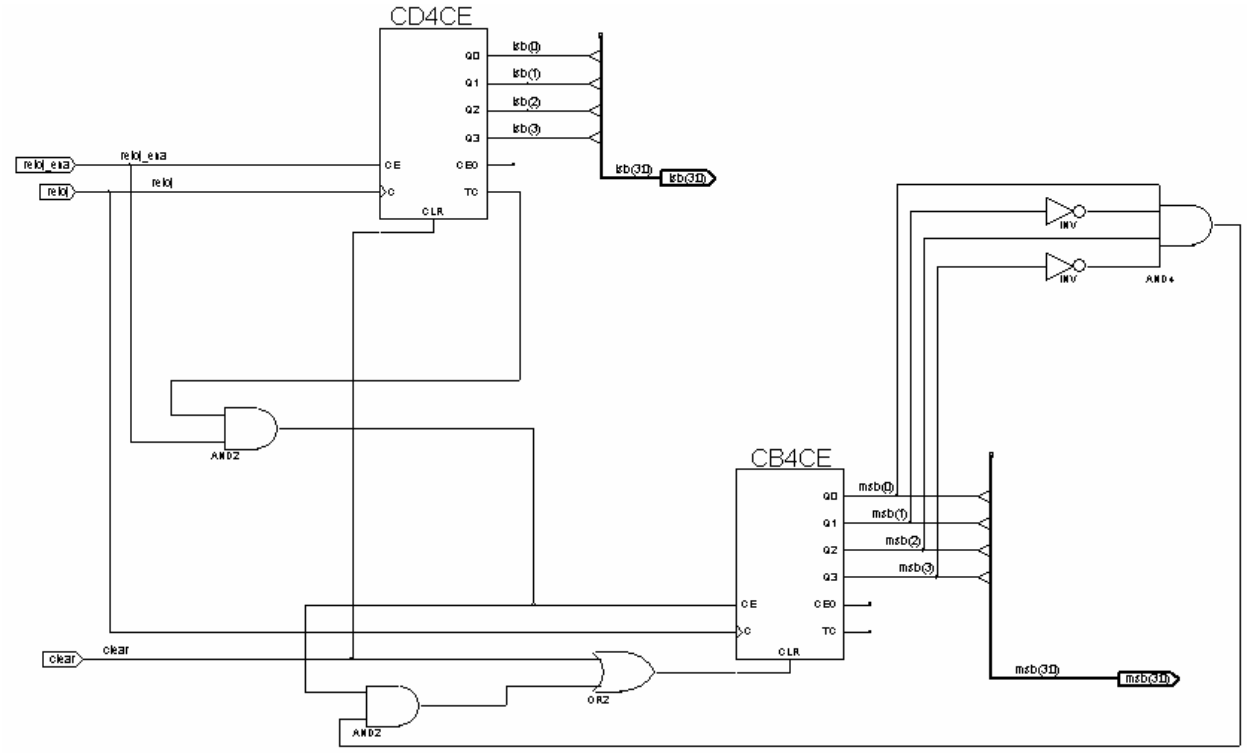

Figura 10. Esquema correspondiente a un contador de 0 a 59.

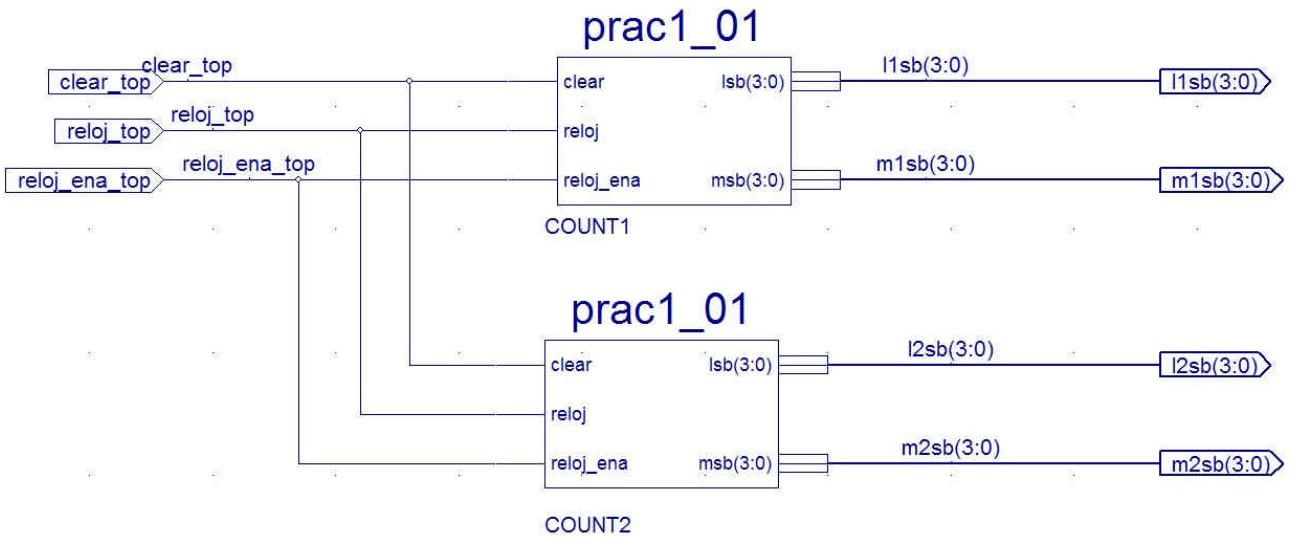

Figura 11. Esquema de dos contadores de 0 a 59.

6. Simulación con ModelSim.

Para arrancar el simulador ModelSim basta con marcar el fichero fuente que se quiere simular y hacer doble clic sobre el proceso "***Launch ModelSim Simulator***", entonces se arrancará ModelSim, compilará las fuentes del proyecto y a partir de ese momento basta con generar las señales de entrada para poder visualizar la salida. El proceso a seguir es el siguiente:

- En la ventana **signals** aparecen todas las señales involucradas en el diseño, marcar sólo aquellas que sean de utilidad para analizar el resultado.
- Una vez marcadas se elige el menú de la ventana signals: ***Add -> Wave -> Selected signals***
- Con esto aparecen las señales marcadas en la ventana **Wave** donde se mostrarán la evolución temporal de las señales.
- Seguidamente, todavía en la ventana **signals**, se marca cada señal una por una y se le asigna un estado lógico. Este valor se asigna mediante el menú ***Edit -> Clock*** si queremos que sea una señal cuadrada, o ***Edit -> Force*** para que tenga un valor fijo.
- Para comenzar la simulación propiamente dicha, nos vamos a la ventana principal de ModelSim y ejecutamos ***Simulate -> Run -> Run 100ps***, por ejemplo. Pulsando repetidamente avanzará la simulación.

6.1. Generador gráfico de estímulos (HDL Bencher).

Si se desea crear un fichero de estímulos para las entradas, de forma que no sea necesario definirlos cada vez que se ejecuta ModelSim, esta herramienta permite crear dicho estímulos muy rápidamente, y cuando se ejecuta ModelSim, automáticamente se ejecuta un tramo de simulación inicial.

Para crear un fichero con estas características, basta con añadir una nueva fuente al diseño (***New Source***), y a la hora de especificar el tipo de nueva fuente a crear, se le indica que es del tipo "***Test Bench Waveform***". El paso siguiente consiste en una pantalla donde se define la señal de reloj, u otras opciones si las hubiera (figura 12). Dado que nuestro diseño es síncrono, se deberá definir qué señal es la de reloj. Seguidamente aparece una ventana con las señales del diseño donde con el ratón podemos cambiar el valor que se va a aplicar a las señales. Una vez generadas las combinaciones de entrada que consideremos necesarias para simular los diferentes modos de funcionamiento (figura 13), guardamos y cerramos el fichero de estímulos.

Desde ***Project Navigator***, teniendo seleccionado el fichero de estímulos (que ahora aparece en la ventana de fuentes con extensión .tbw), ejecutaremos la opción de ***Simulate Behavioral VHDL Model***.

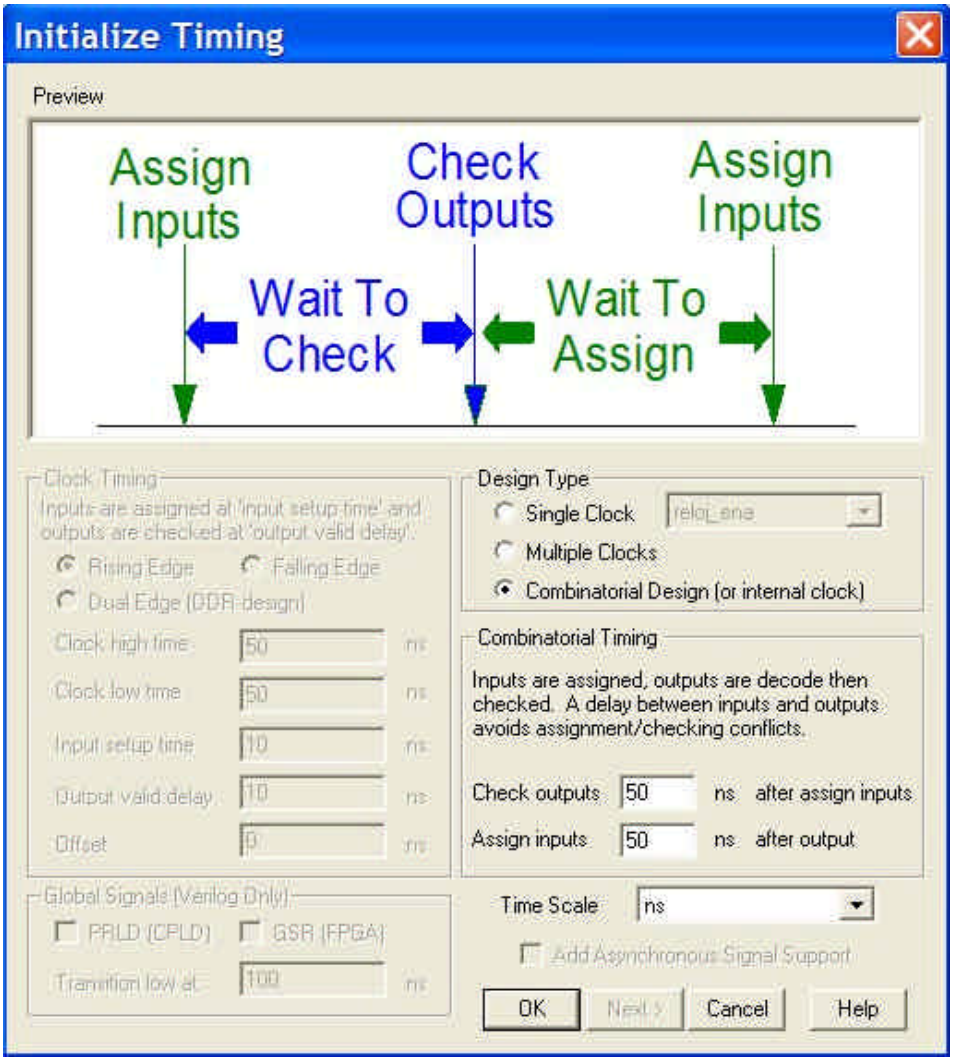

Figura 12. Esquema de dos contadores de 0 a 59.

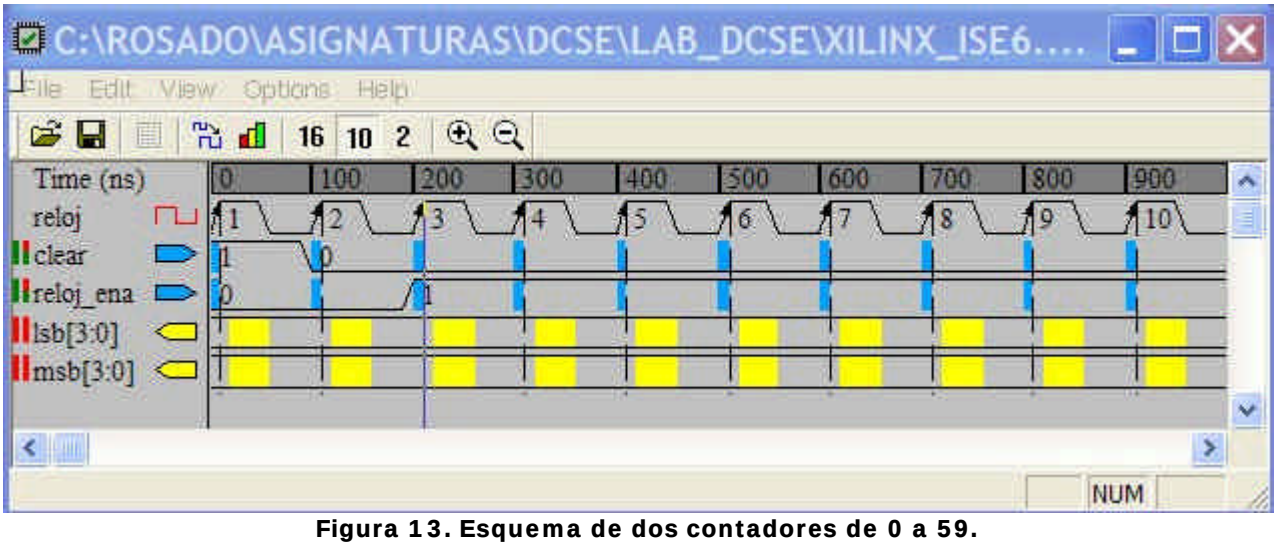

Figura 13. Esquema de dos contadores de 0 a 59.

7. Diseño de máquinas de estados.

El diseño de máquinas de estado se puede realizar mediante una herramienta muy visual llamada StateCAD, de modo que se realiza un diagrama de estados que luego se convierte automáticamente a un lenguaje HDL. Para crear un diagrama con StateCAD, se crea un proyecto nuevo (se pueden poner las mismas opciones de dispositivo que se pusieron en el diseño con esquemático), o se añade un nuevo diseño

al existente (opción **New Source**), y en la ventana de selección del tipo de diseño se selecciona **State Diagram**.

En esta parte de la práctica vamos a crear un diseño de un sumador con entrada serie y salida paralelo de 4 bits. Se reciben 4 bits de entrada mediante la entrada serie (***di***), se deja un ciclo de reloj sin adquirir dato, se reciben otros cuatro bits, y entonces se almacena el resultado de la suma de ambos en el acumulador (***ac***). El objetivo de esta parte de la práctica es aprender a dibujar máquinas de estados de forma gráfica, aunque no se pretende simularla.

El diagrama de estados correspondiente a este funcionamiento es el siguiente:

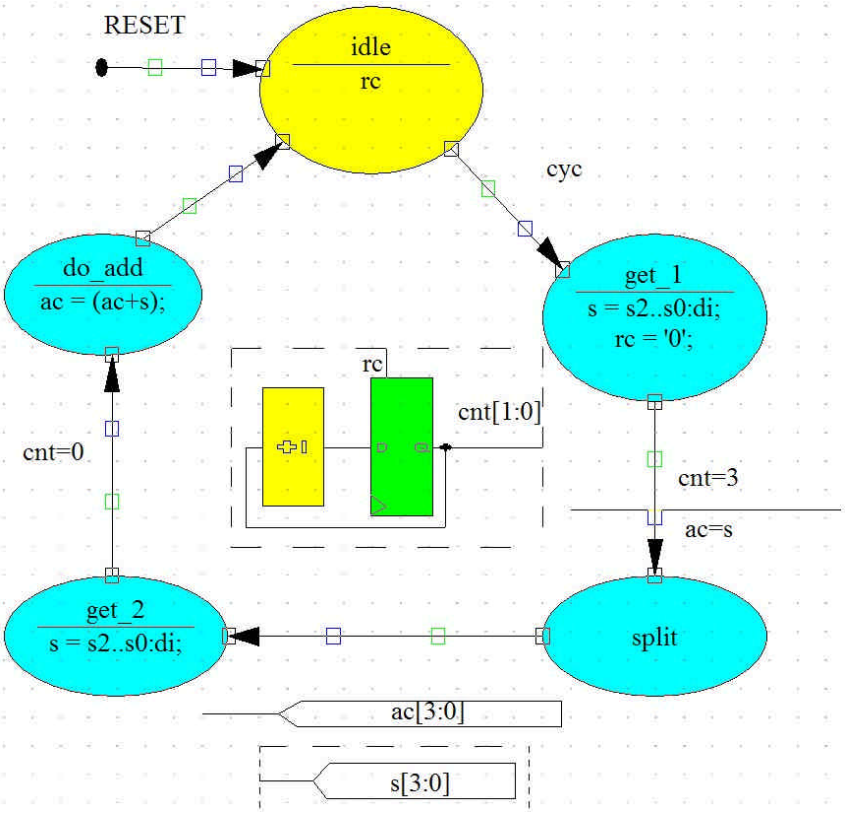

Figura 13. Máquina de estados del sumador de entrada serie y salida paralelo.

Uno de los modos más rápidos para crear una máquina de estados es mediante la utilización del asistente, que facilita la labor de creación de estados y transiciones. Para acceder a él se utiliza el botón de ***Draw State Machines***.

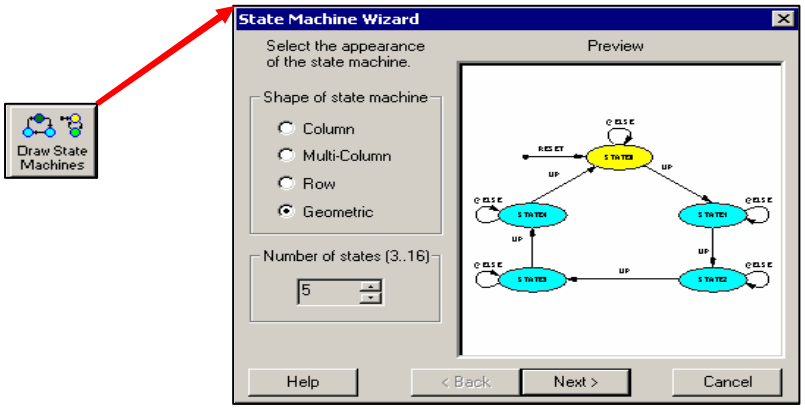

Figura 14. Inicio del asistente de máquina de estados.

- Como primer paso se selecciona la opción de dibujo geométrico y 5 estados.
- El segundo paso consiste en elegir el tipo de reset que se tiene, que en este caso haremos síncrono.

- El siguiente paso consiste en configurar las transiciones entre estados (**Setup Transitions**). Pulsa el botón **Default**, y todos los campos se rellenan con lo valores por defecto, pulsa entonces Finalizar para poder emplazar el esqueleto de la máquina de estados dentro del área de diseño.
- Ahora es necesario configurar las transiciones y los valores en cada estado:
  - Haz Doble-clic sobre ***STATE0*** para abrir el diálogo ***Edit State***. Escribe el nombre del estado (**idle**) en el campo **State Name**, y **rc** en el campo de salidas (**Outputs**). Haz Clic en **OK**.
  -
  - Haz Doble-clic sobre ***STATE1*** para abrir el diálogo ***Edit State***. Nombra el estado con **get_1**. Haz Clic en el botón **Output Wizard**. Aparecerá el asistente lógico (**Logic Wizard**) de la figura 15. Escribe los mismos valores que se muestran en la figura.
  - Adicionalmente, es posible añadir más acciones sobre el mismo estado, en este caso, también deseamos poner el valor **rc=0** en este estado, con lo que volveremos a editar el estado **get_1**, se ejecuta el **Output Wizard** y se elige como operación a realizar **constant**, como ***CONSTANT*** se marca **0**, y como ***DOUT*** **rc** con anchura de datos 1.
  - Tras pulsar **OK**, veremos que en el campo **Outputs** del diálogo **Edit State** se muestran las acciones que hemos configurado con el asistente.

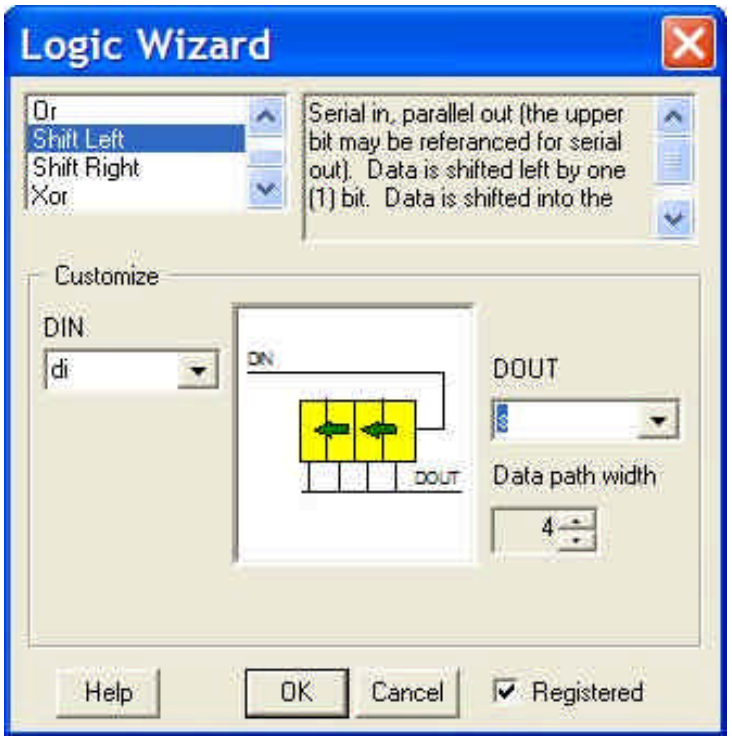

Figura 15. Inclusión de funcionalidad dentro de cada estado.

Para el resto de estados los pasos a seguir son equivalentes, a continuación se muestra cómo realizar los pasos necesarios:

- Doble-clic en **STATE2** nombra el estado con **split**. Edita el estado y ya está.
- Doble-clic en **STATE3** nombra el estado con **get_2**. Utiliza **Output Wizard** para marcar las mismas opciones que en el estado 1, pero sólo para el desplazador a izquierda.
- Doble-clic en **STATE4** nombra el estado con **do_add**. Utiliza **Output Wizard** para seleccionar las opciones que muestra la figura 16.

Ahora solo queda completar las condiciones de transición entre estados. Para añadir una transición es necesario hacer doble-clic sobre la línea de transición entre estados, de este modo aparecerá el diálogo **Edit condition**. Entre el estado **idle** y **get_1** pondremos la condición **cyc**. Del mismo modo, añadir el resto de condiciones tal y como aparecen en la figura 13.

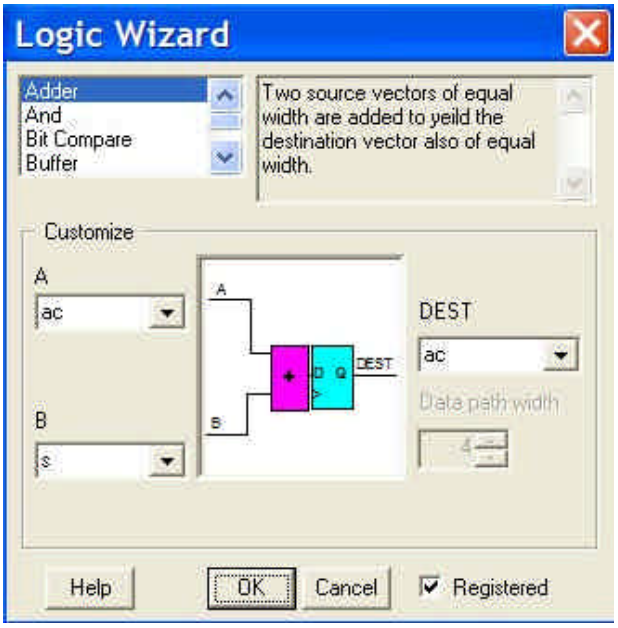

Figura 16. Asistente lógico para el estado do_add.

Otra posibilidad de StateCAD es la de incluir operaciones lógicas particularizadas, se pueden incluir sumadores, restadores, contadores, latches, puertas And, Or, Not, Xor, Nand, Comparadores, desplazadores, Multiplexores, etc.

Vamos a ver un ejemplo añadiendo un contador en el estado **get_1**:

- Haz clic en el botón **Random Logic**: 

- Al hacer clic en una posición de la hoja de dibujo del diagrama se abre el asistente lógico (**Logic Wizard**), selecciona el componente **Count Up**, en el campo COUNT asigna el nombre cnt y la dimensión del contador se elige de 4 bits. Cerramos el diálogo con OK.
- Aparece el diálogo **Edit Equation**. Este diálogo permite configurar a medida las señales del elemento a utilizar. En este caso las opciones a elegir son:

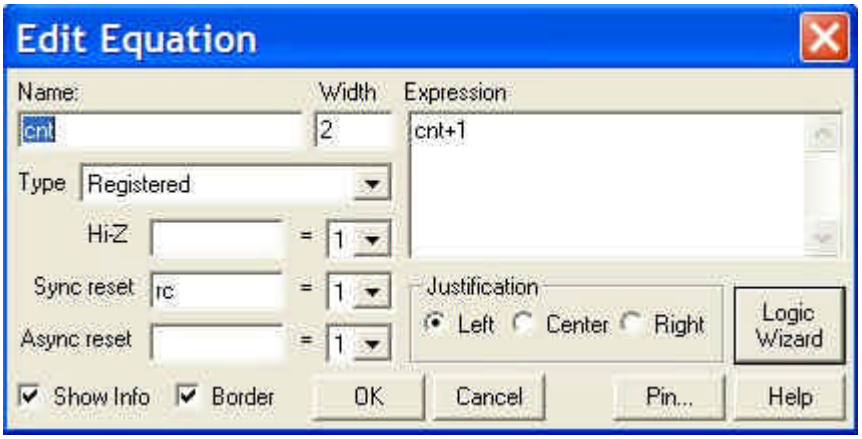

Figura 16. Cuadro de diálogo de configuración del elemento seleccionado.

De momento nos quedaremos en este paso y no simularemos la máquina de estados. En esta práctica el objetivo era saber manejar el editor de estados, posteriormente aprenderemos a generar código HDL basándonos en ella y a crear elementos de librería para utilizar en esquemas de alto nivel.