6.3 ROUTING

Constituye la fase más importante de todo el flujo de diseño. El rutado de las líneas que componen la interfaz se realiza mediante la herramienta FPGA *Editor*, el resto de la lógica se ruta de forma automática usando la herramienta *par*.

El objetivo es rutar las líneas que componen la interfaz diseñada en las fases anteriores entre los grupos de lógica definidos, usando unos recursos físicos que se especifiquen.

El control del rutado de la interfaz se fija hasta el nivel de asignar a cada *net* un recurso físico (cable) que enlaza las zonas fija y dinámica, y sólo éste puede realizar el enlace. Ninguna línea que resulte tras el rutado automático del resto de la lógica puede ocupar recursos físicos de la zona dinámica, y cada *net* de la interfaz sólo puede unir ambas zonas mediante la línea física especificada.

Las siguientes acciones deben de realizarse para poder obtener el rutado de la interfaz entre la zona fija y la zona dinámica de la FPGA con las anteriores condiciones:

- Tras haber realizado el *placement,* se necesita saber en que lugar han sido colocados los CLBs y sus correspondientes pins que son salida o entrada de las líneas de la interfaz, para cada grupo de lógica. Esta información se encuentra en el fichero de texto *Floorplan* (*.fnf*).

- Se utiliza la herramienta FPGA *Edito*r. El rutado automático de las líneas que componen la interfaz se concreta realizando un fichero *script* que contiene las órdenes necesarias para rutar cada una ellas recurso físico que se especifique.

Hay que realiza la elección de que tipo de recursos físico de rutado es asignado a cada una de las *nets*. Debe de ser un tipo de línea física que cruce toda la FPGA de un lado a otro sin hacer uso de las matrices de interconexión y con mínimo retraso.

Además es necesario tener una correspondencia entre cada una de las *nets* y su línea física. La elección de la línea física debe hacerse teniendo en cuenta factores como la distancia a los grupos de lógica que pretende enlazar.

En una arquitectura FPGA Virtex, el tipo de línea física que más se adecua es el *h_long line*. Discurre de lado a lado de la FPGA y existen 12 líneas por cada fila de CLBs, lo que suele ser suficiente.

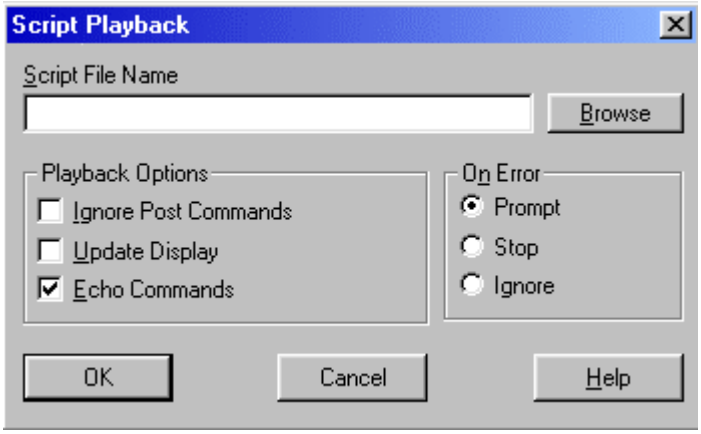

Figura 6-13: Menú FPGA *Editor* para ejecución de *scripts*

Para la realización del fichero *script* se desarrolla una pequeña aplicación en un entorno de programación (Visual C, Visual Basic, etc). Dicha utilidad debe de extraer la información del *placement* necesaria del fichero *floorplan*, basándose en el formato fuertemente estructurado del fichero y el nombre de cada una de las nets que componen la interfaz

La aplicación tiene como entrada una lista de *nets*, de líneas físicas a usar y el *Floorplan File*.

El fichero *script* generado tiene que reproducir la siguiente estructura para cada una de las *nets*:

```
# net         mcore0/apbi<0><PWDATA><9> (comentario)
unselect -all
select pin CLB_R24C52.S1.X        (pin de una zona de la FPGA)
select wire H_HLONG/x:8845/y:4196 (línea física a usar)
select pin CLB_R20C4.S1.BX        (pin de la otra zona)
route                             (orden de rutado manual)

unselect -all
select net mcore0/apbi<0><PWDATA><9>
autoroute                         (rutado automático del resto
                                  de pins conectados a la net)
```

La estructura del *script* hace que la herramienta FPGA *Editor* rute entre ambas zonas de la FPGA (zona del diseño principal y zona para la lógica de "relleno") usando sólo los recursos físicos (cables) especificados para cada *net*. Además cumple la condición de ser esa línea física la única que enlaza entre ambas zonas.

La estructura del *script* se basa en el algoritmo de rutado que realiza la herramienta FPGA *Editor*. El comando *route* ruta los objetos seleccionados en el mismo orden, si no tienen conexión directa realiza el rutado automático necesario para unirlos. El comando *autoroute* entiende que la parte ya rutada es fija y no la modifica. Así se pueden rutar de forma automática el resto de conexiones de cada *net*.

- Por último, se realiza el rutado del resto del diseño usando el método *Incremental Routing* de la herramienta PAR. Respeta el rutado ya existente y ruta el resto de señales del diseño.

La elección de este modo de rutado frente a la opción normal se debe a dos efectos observados durante el desarrollo:

- Para preservar el rutado existente en el modo normal es necesario bloquear las *nets* (atributo *lock on*) en el *script*. Esto aumenta el tamaño del mismo y el tiempo de ejecución.

- Si se bloquean todas las líneas ya rutadas, en diseños con una gran cantidad de líneas implementados en dispositivos con pocos recursos de rutado, puede que la herramienta PAR no encuentre forma de rutar el diseño por completo.

La opción *Incremental Routing* consigue el rutado completo en estos casos, a costa de modificar alguna *net* ya rutada. Esta variación es más fácil de resolver en la siguiente fase de corrección.

6.3.1 *ROUTING*: LEON

Se aplican las anteriores acciones al resultado del *placement* del procesador LEON obtenido.

Como recursos de rutado se utilizan las líneas *h_long lines* que posee Virtex. Existen 12 líneas por fila y un total de 672 líneas *h_long*. Suficientes para rutar toda la interfaz AMBA, ya que ésta consta de 310 *nets* incluyendo las líneas IRQ.

La asignación de *h_long lines* a cada *n*et se hace dando prioridad a las más cercanas a la lógica que conectan, pero siempre de forma continuada para tenerlas agrupadas y no dispersas. En total se hacen tres grupos de *nets*, APB, AHB esclavo y AHB maestro, dejando varias filas de líneas *h_long* entre ellos como separación.

Para hacer uso de las líneas *h_long* hay que denominarlas con la siguiente notación:

```
H_HLONG/x:????/y:????
```

6.3.1.1 Generador de *scripts*

Para poder automatizar la generación del fichero *script* es necesario crear una aplicación que usa la información contenida en el fichero *floorplan*. La estructura que sigue este fichero es la que permite realizar el *scrip*t de forma automática.

Por cada *net*, para que funcionen correctamente los comandos de FPGA *Editor* y se obtengan las líneas rutadas correctamente entre las zonas dinámica y fija de la FPGA, es necesario conocer el pin del CLB cuya salida es esa *net (driver pin)*, y un pin de entrada de la otra zona (*load pin*).

En el fichero *floorplan*, las *nets* se encuentran numeradas y cada CLB usado en el *placement* tiene indicado que *net* le ha sido asignada a cada pin.

Una vez encontrado un CLB que está conectado a una *net* de la interfaz, se puede extraer la información de si es una salida o entrada, un generador de funciones o *flip-flop*, y por tanto, la denominación del pin necesario para el comando de selección.

```
n 389 mcore0/ahbmo<1><HWDATA><7>
n 390 mcore0/ahbmo<1><HWDATA><8>
n 391 mcore0/ahbmo<1><HWDATA><9>

s C13265 FGF c 1
 place CLB_R26C51.S0.F
 p I0 I 744
 p I1 I 628
 p I2 I 2564
 p I3 I 389
 p O O 646
```

De las líneas anteriores se obtendría el siguiente comando tras extraer la información necesaria antes mencionada:

```
select pin CLB_R26C51.S0.F4
```

Si una *net* tiene dirección zona dinámica – zona fija es necesario encontrar en el fichero *floorplann* tras el *placement*, el pin de salida en la zona dinámica y un pin de entrada en la zona fija. Para la dirección contraria se realiza de forma análoga.

Para facilitar el algoritmo de búsqueda se utilizan las restricciones de área que se impusieron a los distintos grupos de lógica. Estas restricciones aparecen el fichero *.fnf* encabezando todos los CLBs que pertenecen a es grupo, esto facilita el punto de comienzo de una búsqueda intensiva.

Conociendo estos pins, el rutado actúa correctamente al rutar entre ellos sólo la línea física que se especifica y ninguna otra líneas entre ambas zonas.

La figura 6-14 muestra la aplicación desarrollada para obtener el fichero *script*. Realiza un algoritmo de búsqueda basándose en la estructura antes planteada. Es necesario indicarle cuatro ficheros: un fichero de salida (*.scr*), el fichero *floorplan* (*.fnf*), y dos ficheros de texto que contienen las *nets* que componen la interfaz y la correspondientes líneas *h_long*.

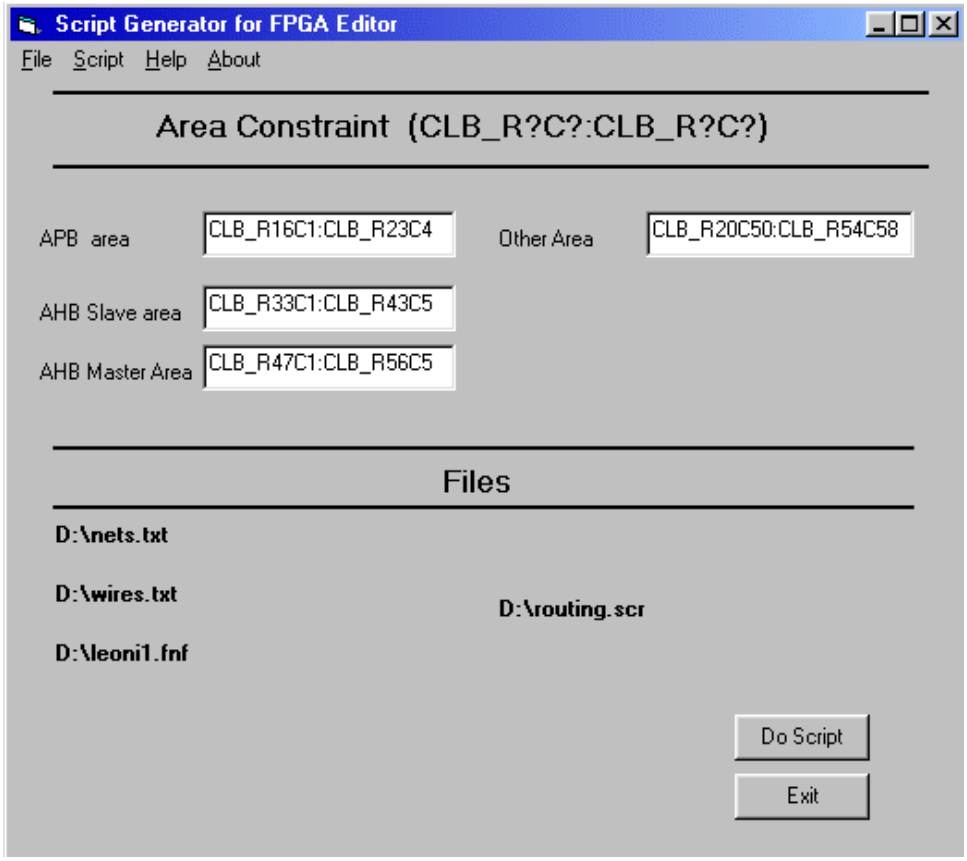

Figura 6-14: Aplicación *genscript*

Para la anterior net, se obtienen los siguientes comandos:

```
# net         mcore0/ahbmo<1><HWDATA><7>
unselect -all
select pin CLB_R26C51.S0.F4
select wire H_HLONG/x:8847/y:-11154
select pin CLB_R50C4.S0.XQ
route
```

```
unselect -all
select net mcore0/ahbmo<1><HWDATA><7>
autoroute
```

Antes de aplicar el comando `autoroute` a cada *net* es necesario haber aplicado el rutado parcial a todas las líneas de la interfaz, para evitar que el rutado automático use líneas *h_long* que vayan a ser usadas más adelante en el *script*.

6.3.1.2 Edición

Tras obtener el *script,* se inicia el FPGA Editor y se ejecuta sobre el fichero de la implementación que tiene el *placement* realizado.

El resultado puede observarse en la figura 6-15. Entre la zona que se fijó como dinámica (lado izquierdo) y la zona fija (lado derecho), no existe ninguna línea rutada salvo las de la interfaz, y éstas usan los recursos físicos que se especificaron.

Se pueden distinguir los tres grupos de líneas que existen: APB, AHB esclavo, AHB maestro.

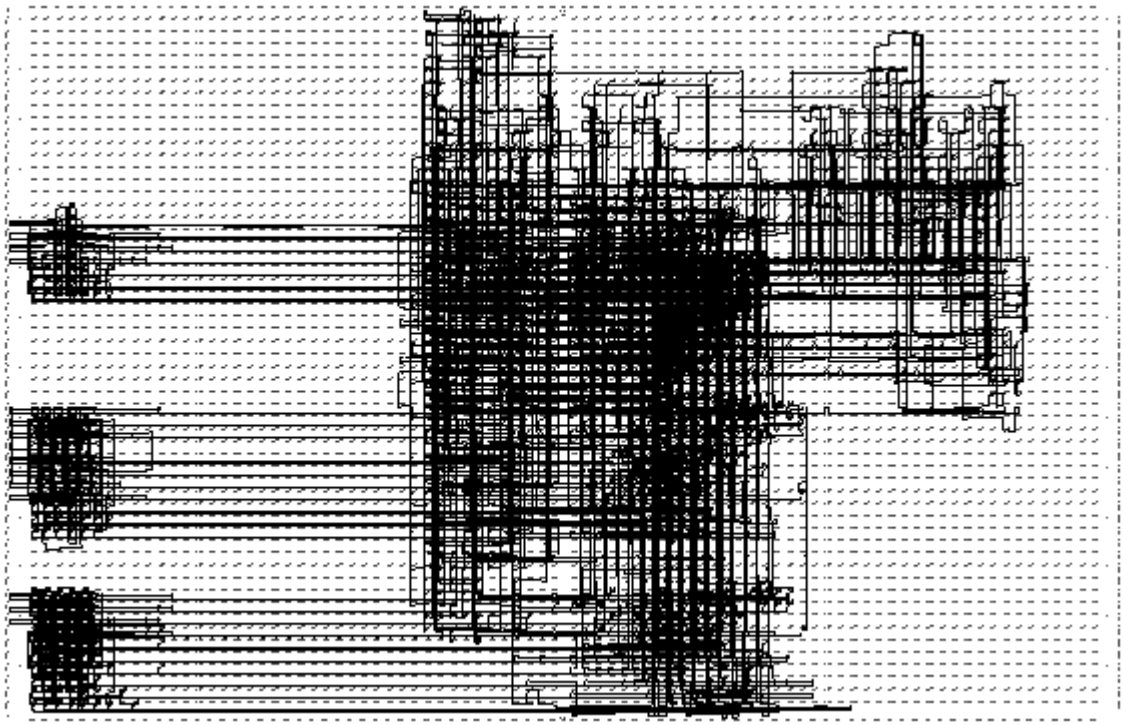

Figura 6-15: Interfaz rutada tras la ejecución del *scrip*t

La información generada en el *script* deberá ser utilizada en un futuro para la obtención de los periféricos que se diseñen para ser conectados en la interfaz rutada.

6.3.2 PAR AUTOMÁTICO

Se realiza el rutado del resto del diseño usando el método *Incremental Routing* (opción –k) de la herramienta PAR.

Esta opción respeta las señales ya rutadas y realiza el rutado de forma automática del resto de señales del diseño.

```
par  -w -ol 4 -k -p  leoni1.ncd leoni1r.ncd leoni1.pcf
```

A continuación se muestra parte del informe que proporciona la herramienta PAR tras realizar el rutado:

```
1484 connection(s) routed; 28769 unrouted active, 114 unrouted
       PWR/GND.
Starting router resource preassignment
Completed router resource preassignment. REAL time: 25 mins 26 secs
Starting iterative routing.
Routing active signals.
.....................................................
End of iteration 1
30253 successful; 0 unrouted active,
   114 unrouted PWR/GND; (0) REAL time: 39 mins 42 secs
.....................................................
Power and ground nets completely routed.
Total REAL time: 53 mins 30 secs
Total CPU  time: 53 mins 29 secs
End of route.  30367 routed (100.00%); 0 unrouted.
No errors found.
Completely routed.
```

Para el procesador LEON en Virtex XCV800, la herramienta PAR ruta todas las líneas del diseño pero modifica dos del total que componen la interfaz (315 líneas).

En la fase de V&C (Verificación y Corrección) se analiza la causa y se realizan las modificaciones necesarias para volver a obtener la interfaz rutada por completo.

6.4 VERIFICACIÓN Y CORRECCIÓN (V&C)

Es necesario realizar una verificación del resultado obtenido tras el *routing* automático, y si existen desviaciones del objetivo final realizar las correcciones oportunas. El proceso de V&C se encarga de esto.

- Se comprueba con la herramienta FPGA *Editor* que la interfaz mantiene la situación física asignada antes de realizar el rutado automático del resto de conexiones (PAR automático).

Si no ha sido modificada ninguna *net* rutada que forme parte de la interfaz, entonces ya se ha obtenido el diseño completamente rutado y cumpliendo todas las restricciones que se le han impuesto. El último paso consiste en generar los bits necesarios para la configuración de la FPGA (*bitgen*).

La modificación de una *net* rutada por parte de la herramienta PAR se produce sólo en ciertos casos con unas circunstancias de diseño muy especiales.

Si la interfaz rutada contiene un gran número de líneas y el dispositivo FPGA utilizado contiene el diseño con un gran porcentaje de utilización, puede ocurrir que alguna línea cambie su rutado inicial con la opción *Incremental Routing*. Es debido a la escasez de recursos de rutado en el dispositivo FPGA y al gran número de líneas.

Si se hubiesen bloqueados las *nets* tras haber ejecutado el *scrip*t (atributo *lock on*), no se modifica el rutado existente, pero el diseño puede quedar sin rutar por completo. La situación que existe con esta posibilidad, *nets* sin rutar que no forman parte de la interfaz, es más complicada de corregir que si las *nets* modificadas forman parte de la interfaz rutada y el resto del diseño ha sido completamente rutado.

La corrección de esta situación se resuelve aplicando las siguientes acciones:

- Corrección en el *placement* del diseño una vez rutado por completo pero con alguna línea de la interfaz sin usar los recursos físicos que se especificaron. El problema se debe a la saturación de la matriz de conexiones que existe en cada CLB, y en particular la de un CLB que se conecta a la línea modificada. El CLB afectado se intercambia a una localización cuya matriz de rutado tenga una menor densidad de conexiones, siempre dentro de la zona asignada (restricción) al correspondiente grupo de lógica.

- Rutado de la línea de la interfaz modificada con el CLB en la nueva localización. Los comandos se obtienen del *script*, con la única modificación de la nueva denominación del pin del CLB afectado.

- El resto de conexiones modificadas con el cambio que no forman parte de la interfaz se rutan usando el comando *autoroute* aplicado al CLB seleccionado

Tras las acciones de corrección, se ha obtenido el diseño completamente rutado, y con la *net*s pertenecientes a la interfaz en los recursos físicos que se han especificado.

6.4.1 V&C: LEON

En el ejemplo de aplicación, el dispositivo FPGA Virtex XCV800 puede quedar escaso de recursos de rutado.

Se observa en los resultados una línea cuyo rutado inicial ha sido modificado por la herramienta PAR:

```
mcore0/ahbmo<1><HADDR><18>
```

La modificación de esa línea es debida ha haber fijado 315 *nets* en unos recursos fijos (H_LONG), y sin tener en cuenta los posibles caminos para el resto de líneas que quedan sin rutar. Observando el diseño se puede ver que el problema se debe por la agrupación de CLBs, que provocan escasez de recursos de rutado en ciertas zonas de la FPGA XCV 800.

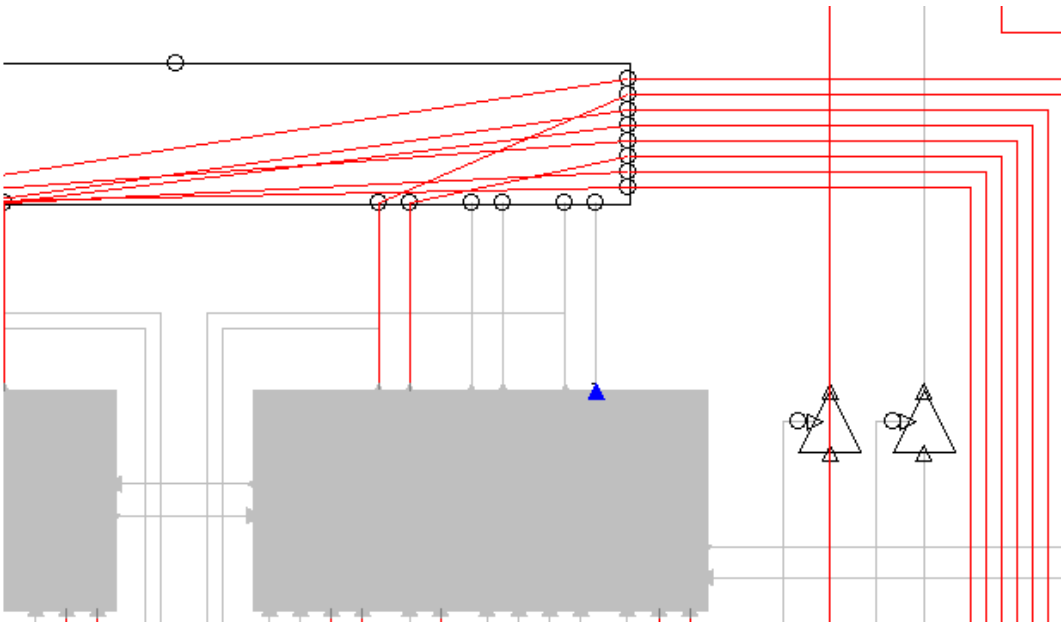

Figura 6-16: Escasez de recursos de rutado para *net* modificada

La matriz de conexiones que hay por encima del CLB cuya pin de salida pertenece a la *net* modificada está ya utilizada por completo y no permite nuevas conexiones (Figura 6-16).

Para obtener la net rutada por la línea *h_long* que se especificó se realizan las anteriores acciones.

- Modificación de la posición del CLB a una zona menos densa en conexiones
- Nuevo rutado de la *net* modificada
- Rutado del resto de conexiones del CLB

En la figura 6-17 puede observarse que ahora si están todas las conexiones del CLB rutadas al disponer de recursos de retado en la matriz de conexiones.

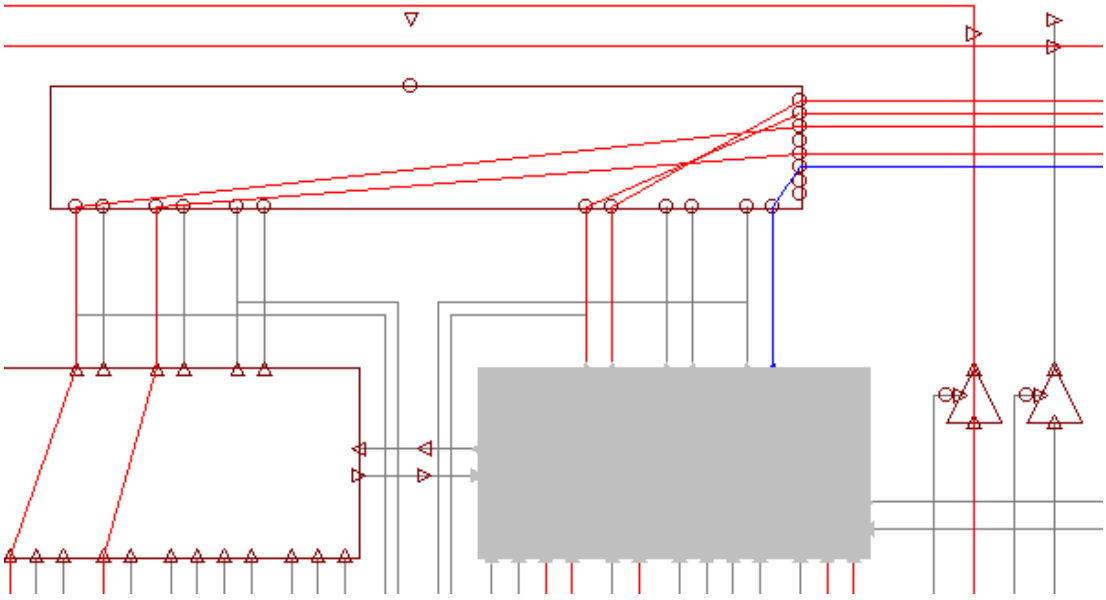

Figura 6-17: *Net* correctamente rutada en nueva localización

Como se señaló durante la fase de diseño de la interfaz, la línea `hbusreq` realiza la petición del control del bus al árbitro, y en caso de que no se haya conectado ningún maestro a la interfaz hay que fijar su valor permanentemente a 0 lógico para evitar falsos arbitrajes del bus.

La lógica que comanda la línea se modifica al nivel físico, mediante la utilidad *Block Editor* de *FPGA Editor*.

Una vez rutado por completo el diseño y tras haber realizado las posibles correcciones, es necesario modificar la lógica que tiene por salida `hbusreq`, para fijar su valor de salida en GND/ cero.

La siguiente figura muestra las modificaciones que se han realizado sobre el CLB cuya salida es `hbusreq`.

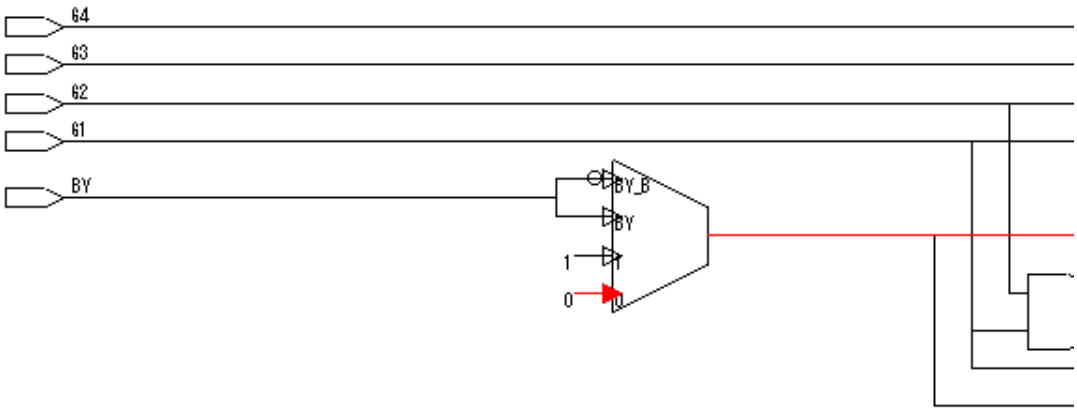

Figura 6-18: Modificación de CLB `hbusreq`

6.5 BITGEN

Para obtener la implementación del diseño es necesario obtener los bits de configuración del dispositivo objetivo, en este caso, la XCV800. La información acerca de la estructura de los bits de configuración es proporcionada por Xilinx.

Si se desea aplicar reconfiguración parcial al diseño hay que identificar las tramas de bits que corresponden a la zona fija y las tramas que corresponden a la zona dinámica. La unidad utilizada es una columna de la matriz de CLBs.

Una vez identificadas se puede obtener los bits de configuración dinámica, o en su caso aplicarlos durante el diseño de los periféricos reconfigurables.

6.6 RESULTADOS

La figura 6-19 muestra la implementación del procesador LEON en la Virtex XCV800 tras haber realizado el control del rutado de las líneas del bus AMBA.

Se puede observar como se ha conseguido agrupar el procesador LEON en una zona de la FPGA. Las líneas horizontales corresponden con las líneas físicas *h_long* que implementan el bus AMBA entre la zona fija y la zona dinámica.

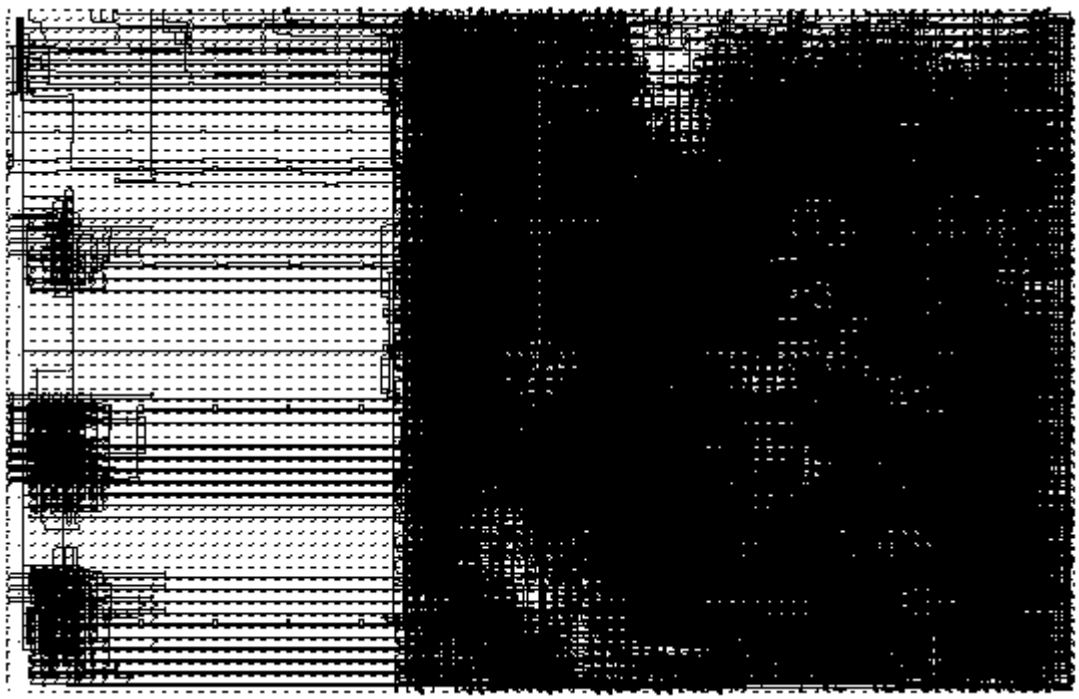

Figura 6-19: Implementación física del LEON con técnica de control del rutado en XCV800

No hay más líneas que ocupen la zona de la interfaz salvo las correspondientes de los bloques de memoria e IOBs que deben de situarse en la zona dinámica debido al tamaño de la XCV800.

Las líneas horizontales que no usan líneas físicas *h_long* y se encuentran en la zona de la interfaz corresponden con las líneas de CLK y *reset* del dispositivo. Este tipo de líneas usa recursos de rutado dedicados en la FPGA Virtex.

Se puede comparar los resultados obtenidos con la técnica de control del rutado desarrollada con la implementación del procesador LEON sin hacer uso de ella, mostrada en la figura 5-8.

Si la FPGA utilizada fuese más grande, XCV1000, por ejemplo, se podría haber obtenido una implementación con una mayor separación entre las dos zonas y sólo las líneas de la interfaz entre ellas.

En el momento de diseñar los periféricos que serán alojados en la zona dinámica es necesario tener en cuenta la siguiente consideración acerca del uso de líneas *h_long*. Cuando la herramienta PAR realiza el *routing* automático, habrá *nets* que no forman parte de la interfaz y que se implementen sobre líneas *h_long*. Por tanto, esa línea está ocupada a lo largo de toda la FPGA y no debe utilizarse en la implementación de los periféricos reconfigurables.

Las líneas *h_long* que han sido ocupadas durante el *routing* automático pueden observarse deshabilitando la opción *subtrimming* del FPGA *Editor*. Dicha opción permite mostrar en las *nets* rutadas sólo la parte de línea física utilizada o toda la línea.

6.6.1 ANÁLISIS TEMPORAL

En el análisis temporal que se realiza tras todas las etapas anteriores se obtienen los resultados que permiten determinar la frecuencia máxima de reloj:

El siguiente extracto del informe generado por la herramienta *trce* permite estimar la velocidad máxima del diseño, para posteriormente compararla con la obtenida en la implementación del diseño sin restricciones.

```
 Design statistics:
    Minimum period:  55.914ns (Maximum frequency:  17.885MHz)
    Maximum net delay:  17.490ns


 The Average Connection Delay for this design is:        3.063 ns
 The Maximum Pin Delay is:                              17.490 ns
 The Average Connection Delay on the 10 Worst Nets is:  13.706 ns

 Listing Pin Delays by value: (ns)

d < 3.00  < d < 6.00 < d < 9.00 < d < 12.00 < d < 18.00 d >= 18.00
--------   ---------   ---------   ---------   ---------   -------
 17808       9762       1966        528         303           0
```

La nueva frecuencia máxima de reloj es **17.88 Mhz**. La frecuencia máxima obtenida en la implementación del procesador sin restricciones de área y rutado fue de 17.51 Mhz**.**

Por tanto no se produce ninguna pérdida de prestaciones en velocidad debido al uso de la técnica de control del rutado en el diseño del procesador LEON. Las frecuencias máximas obtenidas, que son una estimación de la herramienta *trce*, son prácticamente iguales.