HERRAMIENTA DIDÁCTICA PARA EL APRENDIZAJE DEL DISEÑO FÍSICO DE CIRCUITOS INTEGRADOS

MARTA FERNÁNDEZ GONZÁLEZ Y SERAFÍN A. PÉREZ LÓPEZ
Departamento de Tecnología Electrónica. Universidad de Vigo. Campus de Lagoas (Marcosende) 36280-Vigo. España.

Presentamos un nuevo enfoque para la exposición docente de los conceptos asociados con el diseño físico de circuitos integrados. Está basado en una herramienta desarrollada para entornos Windows que permite seguir la evolución de los algoritmos más representativos de las principales fases del diseño físico, tales como particionado, posicionado y enrutado. El método ha sido implantado con éxito en las asignaturas de Diseño Microelectrónico de la titulación de Ingeniería de Telecomunicación de la Universidad de Vigo.

1. Introducción

Desde la aparición de la muy alta escala de integración (VLSI) en la década de los setenta, el diseño físico microelectrónico ha experimentado un avance tal que en la actualidad es factible la fabricación de circuitos integrados con decenas de millones de transistores. Este progreso no hubiese sido factible sin herramientas de diseño sofisticadas. Éstas deben ser potentes desde un punto de vista computacional para poder obtener resultados cercanos a los óptimos, dado que todas las fases del diseño físico constituyen problemas NP-completos, lo cual hace referencia a la imposibilidad de hallar la solución óptima en un tiempo razonable. El computador más potente podría precisar miles de años para explorar todos los posibles casos.

En los últimos treinta años la investigación en diseño físico ha sido intensa, como lo prueba la gran cantidad de publicaciones y herramientas desarrolladas [1,2,3,4]. Por ello ha resultado complicado crear en paralelo una metodología para clasificar y exponer los principales algoritmos y sus conceptos asociados.

Este trabajo ha sido concebido como ayuda al aprendizaje del funcionamiento de los principales algoritmos de diseño físico, no sólo mediante la exposición de su secuencia de operación y pseudo-código (método clásico), sino principalmente por la observación de su evolución para llegar a un adecuado particionado, posicionado o enrutado en un entorno de ventanas (método nuevo), programado mediante lenguajes orientados a objetos (C++ y Java).

Dividimos la herramienta desarrollada en varias secciones, cada una correspondiente a una fase diferente del diseño físico, enlazadas de tal forma que los resultados de una constituyen las entradas a la siguiente. Por ejemplo, el posicionado obtenido de la aplicación de un algoritmo de enfriamiento simulado (*simulated annealing*) es la información que se aplica a los algoritmos de enrutado global.

2. Aspectos fundamentales

La figura 1 muestra el ciclo de diseño físico en VLSI. Básicamente consta de cinco etapas: Particionado, *Floorpanning*/Posicionado, Enrutado, Compactación y Extracción/Verificación.

Debido a la limitación impuesta al tamaño de estas comunicaciones, resulta imposible la descripción de toda la herramienta desarrollada. De entre todas las etapas seleccionamos como ilustrativa la de posicionado o *placement*. Para una visión completa de todo el entorno educativo desarrollado remitimos al lector interesado al servidor ftp de nuestro Departamento [5].

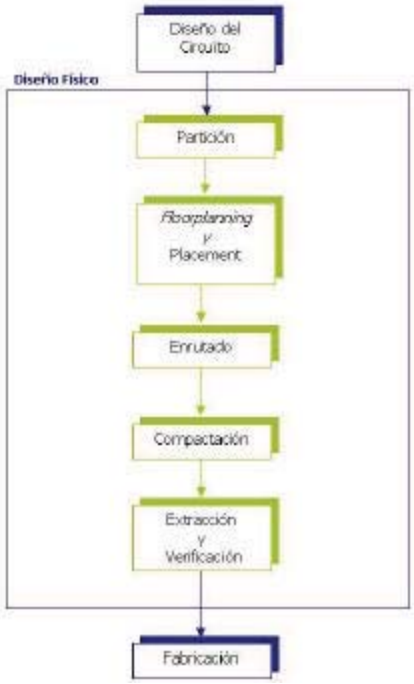

Figura 1: *Ciclo de diseño físico*

En la fase de *placement* se asigna una posición exacta a cada bloque para obtener una disposición de mínimo área con unas restricciones, fundamentalmente temporales. Para entrar se selecciona PLACEMENT en el menú principal. Ello conduce a la ventana que se muestra en la figura 2, que consta de tres opciones: GASP (*Genetic Algorithm for Standard Cell Placement*), SAGA (*Simulated Annealing and Genetic Algorithm for S.C. Placement*) and SA (*S.A. for Macrocells*), correspondientes a los tres algoritmos que se han codificado para ser simulados. Como ejemplo mostramos la evolución del primero.

3. Ejemplo de algoritmo de posicionado o *placement*

Figura 2: *Ventana principal del posicionado*

GASP [2] es un algoritmo genético de posicionado, elaborado siguiendo las ideas propuestas por Holland en los años 70. En él cada solución (*cromosoma* o individuo de la población) al posicionado de las celdas se representa como se indica en la figura 3, esto es, cada *gen* (celda) consta de 4 campos:

- Identificador de la celda
- Coordenada X de la esquina superior izquierda
- Coordenada Y de la esquina superior izquierda
- Identificador de la ranura asignada a la celda

La *calidad* de cada *cromosoma* viene dada por la siguiente función de coste:

$$f = \frac{1}{\sum_{nets} x_i W_H + y_i W_V} \qquad (1)$$

donde *x(i)* e *y(i)* representan el ancho y alto del menor rectángulo que abarca los terminales de la *net* i. $W_H(i)$ y $W_V(i)$ son los pesos asignados a dichas dimensiones.

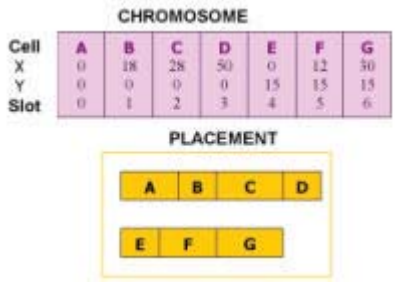

Figura 3: *Representación genética de cada individuo*

GASP hace uso de los 4 operadores genéticos básicos: selección, cruce, mutación e inversión. Comienza con una población inicial aleatoria en la que cada individuo tiene un coste asociado de acuerdo a la ecuación anterior. A continuación entra en la fase de iteraciones, en la cual se generan nuevos descendientes por aplicación de los operadores mencionados. La nueva población debe reducirse para mantenerse constante. La probabilidad de que cada individuo sobreviva es proporcional a su función de coste asociada. De este modo la población va mejorando por cada iteración o era. Cuando se alcanza el número de iteraciones preestablecido, se elige como solución el mejor individuo de la población.

La implementación que hemos realizado resuelve el posicionado de 75 celdas de una librería. La *netlist* se genera aleatoriamente y se ha supuesto un área disponible de 400x440 unidades.

Se diseñaron 4 ventanas para la especificación y seguimiento de la evolución del posicionado:

1. Ventana de control. Permite seleccionar los parámetros de trabajo (figura 4 a la izquierda).
2. Evolución del algoritmo. Esta ventana muestra el porcentaje de ejecución del algoritmo así como el número de iteraciones. Puede verse en la figura 4 a la derecha.
3. Visualización del posicionado. Muestra el posicionado asociado al mejor individuo de la población. La figura 5 muestra un ejemplo. Se puede seleccionar el período de actualización. Un código de colores representa la longitud de las *nets*: rojo las de más de 700 unidades, azul las intermedias (? 700 y ? 400) y gris las más cortas (< 400 unidades).
4. Evolución del coste. Esta última muestra la evolución del coste promedio así como el coste del mejor individuo, conforme el algoritmo va evolucionando entre generaciones.

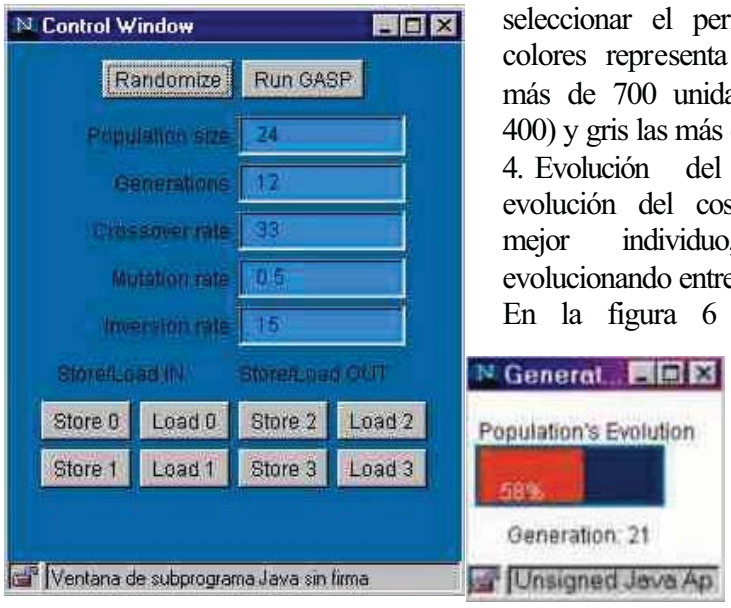

Figura 4: *Ventanas de control y de evolución de ejecución del algoritmo*

En la figura 6 pueden verse un par de ventanas correspondientes a la ejecución del mismo ejemplo con diferentes tamaños de población y número de iteraciones. Puede observarse que la reducción del coste es más significativa en las primeras fases. Ello justifica la importancia del número de iteraciones. Si es pequeño o la tasa de cruce alta, puede ocurrir que el algoritmo finalice aunque no haya sido capaz de optimizar aún el mejor

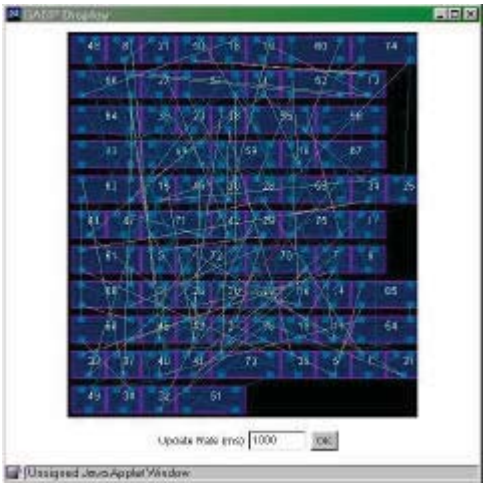

Figura 5: *Ventana de visualización del posicionado*

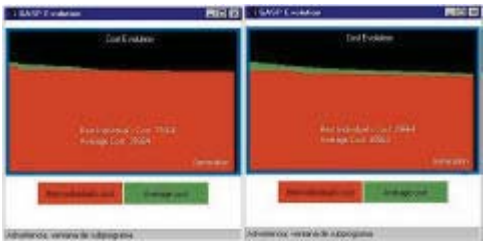

Figura 6: E*volución del coste*

individuo. En el otro extremo, si el número de iteraciones es grande o la tasa de cruce baja, puede ocurrir que el algoritmo se bloquee y el resultado no mejore.

Otro parámetro importante es el tamaño de la población. A medida que aumenta, la solución final mejora aunque ello implica un período de ejecución mayor porque se procesan más individuos y se generan más descendientes.

4. Conclusiones

En este trabajo hemos mostrado una parte reducida de la herramienta desarrollada para facilitar el aprendizaje del diseño físico en VLSI. Por problemas de extensión nos hemos centrado únicamente en uno de los algoritmos de posicionado. Se implementaron algunos más, así como para las fases de partición y enrutado, fundamentalmente.

Se accede a la herramienta mediante un navegador convencional de Internet. Cada sección está relacionada con una fase del diseño físico microelectrónico y contiene la implementación de los algoritmos seleccionados y la exposición de los fundamentos de cada uno. Los algoritmos de particionado se codificaron en C++ y todos los demás en Java, de tal forma que prevalece el carácter didáctico e intuitivo, en tanto que se observa en detalle su evolución.

La aceptación entre el alumnado ha sido más que satisfactoria. Ha contribuido a aumentar el interés por el diseño físico respecto a cursos anteriores. Actualmente estamos desarrollando una serie de mejoras, tales como la inclusión de las fases de *floorplaning* y compactación así como algoritmos para circuitos de altas prestaciones y enrutado tridimensional.

Referencias

[1] N.Sherwani. *Algorithms for VLSI Physical Design Automation*. Kluwer A.P., 1999.

[2] P.Mazumder y E.M.Rudnick. *Genetic Algorithms for VLSI Design, Layout and Test Automation*. Prentice Hall International, 1999.

[3] D.W.Jespen y C.D.Gelatt. *Macro Placement by Monte Carlo Annealing*. Proceedings of IEEE International Conference on Computer-Aided Design :495-498, 1983.

[4] C.Sechen. *VLSI Placement and Global Routing using Simulated Annealing*. Kluwer A.P., 1997.

[5] ftp://www.dte.uvigo.es/pub/VLSI