E.T.S. de Ingenieros de Telecomunicación
Departamento de Ingeniería Electrónica

Herramientas para el Diseño Electrónico
Máster Universitario en Ingeniería de Sistemas Electrónicos

Curso 2010-2011

Proyecto Final, 1ª Parte **Fecha de entrega: Viernes, 11 de Abril de 2011**

Colocación de objetos puntuales

1. Introducción

Los objetivos de este proyecto son (1) practicar escribiendo programas en C de cierta complejidad, y (2) diseñar un programa de colocación (*placement*) básico capaz de colocar objetos puntuales en una rejilla rectangular arbitraria. El programa debe hacer lo siguiente:

- Leer un fichero de entrada donde se describe la *netlist*, o lista en conexionado (qué módulos hay y qué redes los conectan). Como suponemos que es un problema de colocación en un chip, dicho fichero describirá también la localización de los *pads* de entrada/salida (éstos son fijos) y a qué redes están conectados.
- Generar una colocación en una rejilla para esos módulos. El programa utilizará una técnica muy sencilla para esto. Primero, debe generar un *placement* aleatorio. Después debe hacer una mejora iterativa aleatoria usando la métrica del “semiperímetro” para medir la longitud total del conexionado para cada colocación. Utilizará sólo mejora *"downhill"*, es decir, sólo aceptará aquellas perturbaciones que lleven a una menor longitud de conexionado. No intentes nada "mejor" por ahora.
- Escribir el *placement* resultante en un fichero. Cada línea especificará ahora los valores X, Y de la localización de cada módulo en la rejilla.

2. Detalles de implementación

Todo el código será escrito en C o C++, y estará dividido en 4 partes:

1. *"Parser"* de entrada: Esta parte debe ser capaz de leer el fichero de entrada y construir las estructuras de datos para almacenar esa información (recuerda las estructuras del artículo sobre *"partitioning"* de Fiduccia-Mattheyses). Después de llamar a esta rutina, tendréis el número de redes, células, pads, tamaño de la rejilla y las estructuras de datos para redes y células.
2. Colocación aleatoria: Esta parte colocará cada pad en su sitio correcto y cada célula en una posición aleatoria. Hay muchas formas de hacer esto. Si de verdad no se te ocurre ninguna, pregunta.

3. Mejora de la colocación: Aquí es donde se realiza la mejora iterativa utilizando la métrica "semiperímetro". Recuerda, sólo acepta cambios que mejoren la longitud total del conexionado. Establece el criterio de parada como quieras, por ejemplo, cuando consideres que aquello ya no mejora más.
4. Generación de la salida: Esta parte escribirá las localizaciones X, Y de todas las células, así como la longitud del conexionado total (según métrica "semiperímetro").

3. Modelo para el problema de colocación

Lo primero que debemos modelar es la rejilla. Dos números la definen: sus dimensiones X e Y. Supongamos que decimos X = 8 e Y = 7. La rejilla debe ser como en la figura:

Y = 6	p	p	p	p	p	p	p	p
5	p	.	.	.	.	.	.	p
4	p	.	.	.	.	.	.	p
3	p	.	.	.	.	.	.	p
2	p	.	.	.	.	.	.	p
1	p	.	.	.	.	.	.	p
Y = 0	p	p	p	p	p	p	p	p
X =	0	1	2	3	4	5	6	7

Cada "p" en la rejilla es un punto donde debe ser colocado un pad. Cada "." es el sitio donde puede colocarse una célula. Células, pues, sólo pueden ir en puntos (x, y) tal que $1 < x < 6$, $1 < y < 5$.

Deberás seguir el sistema de coordenadas de la figura. Observa que la rejilla es uniforme y de distancia entre cada uno de sus puntos de "1" (tanto en el eje X como en el eje Y). O sea, la distancia Manhattan entre dos puntos será siempre:

$$|X_1 - X_2| + |Y_1 - Y_2|$$

donde los puntos se suponen colocados en (X_1, Y_1) y (X_2, Y_2).

4. Formatos

4.1. Formato de entrada

Aquí hay un ejemplo de la sintaxis del fichero de entrada al programa:

```
grid 6 6
results num num num ... num -1
pad pad z0 coord 1 5 pins 24 5 0
pad pad z1 coord 2 5 pins 24 5 0
           .....
gate gate0 g4 0 1 7 60  -1 -1 -1
gate gate1 g4 2 1 9  8  -1 -1 -1
gate gate2 g5 5 0 6     -1 -1 -1
           .....
stop
```

Observa que hay 5 clases de líneas en este formato:

1. La línea que empieza con la palabra clave `grid` nos indica las dimensiones de la rejilla (6x6). Recuerda el modelo de rejilla explicado más arriba.
2. La línea que empieza con la palabra clave `results` contiene información sobre la lista de conexionado después de que se ha realizado la colocación. Esta información está compuesta por una serie de enteros que acaban con un "-1". Por el momento, lo único que tendremos que hacer es almacenar esos valores. Volveremos sobre ello más tarde.
3. Las líneas que empiezan con la palabra clave `pad` especifican los pads de entrada/salida con el siguiente formato:

   ```
   pad PadName  coord  X Y  pins NetNum Number Number
   ```

   donde `PadName` es el nombre del pad (único para cada pad) y X e Y son las coordenadas donde va colocado. La red a la que va conectado queda especificado por `NetNum`. Ignorad los otros números.
4. Las líneas que comienzan con `gate` indican los objetos que debemos colocar. El formado es:

   ```
   gate GateName  GateType  NetNum NetNum ... NetNum -1 X Y
   ```

   donde `GateName` es el nombre del objeto (único para cada puerta) y `GateType` es el tipo de puerta (se puede ignorar por el momento). Los siguientes enteros hasta el -1 son las redes a las que ésta puerta está conectada. Los enteros X e Y después del -1 son las coordenadas donde la puerta está colocada (si esta colocación está prefijada). Si su valor es negativo, se supone que no existe esta restricción previa.
5. La línea con la palabra `stop` indica el final del fichero.

IMPORTANTE: Los números en los nombres de las puertas NO significan nada. Así mismo, los números que indican redes no tienen por qué ser consecutivos.

4.2. Formato de salida

Es muy fácil. Es el mismo que el de entrada. Solo que ahora deberán aparecer todos los valores X, Y de todas las puertas. Por otra parte, el valor de la longitud total de conexionado que calcula tu programa (recuerda, métrica "semiperímetro") debe aparecer en la línea que comienza con `results`. Usa el siguiente formato:

```
results  HALF_PERIM_NETLENGHT  -1
```

Observa que para imprimir todos estos datos en el fichero de salida bastará con recorrer tus estructuras de datos.

5. Evaluación de la solución

La longitud del conexionado total obtenido es una buena medida de cómo de bien funciona tu programa. Otra es el tiempo que tarda en obtener dicha solución. A fin de unificar criterios, mediremos el tiempo en el que el programa realiza 10.000 intercambios de células aleatorios (haz que tu programa sea fácilmente modificable para medir esto). Esta medida se realizará sobre un fichero de entrada estándar que se te suministrará.

Pero también nos interesa saber cuánto tiempo tarda el programa en obtener lo que tú consideras "un buen placement". Mídelo para ese mismo fichero.

6. Requerimientos y fecha de entrega

De acuerdo con todo lo anterior, a continuación se resume lo que ha de ser tu trabajo:

1. Escribe un programa que pueda colocar objetos puntuales en una rejilla arbitraria.
2. Realiza el *placement* de los ejemplos que se te suministren.
3. Mide el tiempo para 10.000 intercambios de células (tiempo CPU). Di cuál es la longitud de conexionado obtenida.
4. Escribe un breve informe sobre lo que has hecho y cómo lo has hecho. **La calidad de este informe es de extrema importancia**.
5. La fecha máxima de entrega es el **Viernes, 15 de Abril de 2011**.
