Proyecto 3 Microelectrónica

Leonardo Agüero Villagra
Escuela de Ingeniería Eléctrica
Universidad de Costa Rica
Carné: B70103

Gabriel Jiménez Amador
Escuela de Ingeniería Eléctrica
Universidad de Costa Rica
Carné: B73895

***Resumen*—En este proyecto se trabaja con la generación de mapas de congestión para el sistema *micro_micro_ucr_hash*. En la primera parte se hace un reporte de la cantidad de DRCs y tiempo de ejecución de las etapas de *Qflow* para los sistemas optimizados en área y velocidad. En la segunda parte para ambos sistemas se generan mapas de congestión de celdas y metal 4 mediante la librería `matplotlib` para el lenguaje de programación *Python* y la asistencia de las expresiones regulares de *Unix*. Finalmente como una tarea extra al proyecto se genera un mapa de congestión de pines.**

I. Introducción

El proyecto realizado corresponde a la continuación del trabajo del primer y segundo proyecto. En el primer proyecto se realiza un diseño de alto nivel del sistema optimizado para área y rendimiento. Este modelo permitió probar el funcionamiento lógico del sistema así como para verificar relaciones de entrada/salida del mismo. En la segunda parte se implementa el RTL mediante el lenguaje de descripción de hardware *Verilog* y mediante el programa *Qflow* se realizan las etapas de síntesis, *placement* y *static timing analysis*, del cual se reportan las métricas obtenidas de estas.

Para esta tercera parte se debe hacer un reporte de la cantidad de DRCs (si los tiene) y el tiempo de ejecución de las etapas del programa *Qflow*. Una vez realizado lo anterior se debe generar un mapa de congestión de celdas y metal 4 para los sistemas optimizados en área y velocidad

En este proyecto se crean los mapas de congestión de celdas, metal 4 y pines de un sistema que emplea cierta función hash llamada `micro_ucr_hash` para iterar sobre sus salidas para lograr generar salidas especiales según cierto bloque de entrada y *target* deseado. La vista general de bloques del sistema se observa en la Figura 1.

El sistema:

1. Recibe un bloque de 12 bytes, un target de 1 byte más una señal de inicio.
2. Concatena un nonce de 4 bytes.
3. Calcula la firma de la concatenación del bloque + nonce, haciendo uso de la función hash `micro_ucr_hash`.
4. Si los dos primeros bytes de la salida:
   - No son menores al target: debe intentar de nuevo modificar el nonce y volver a concatenarlo con el bloque.
   - Son menores al target: el sistema debe retornar el nonce que hace cumplir la condición.
5. Al terminar, el sistema retorna una señal de terminado.

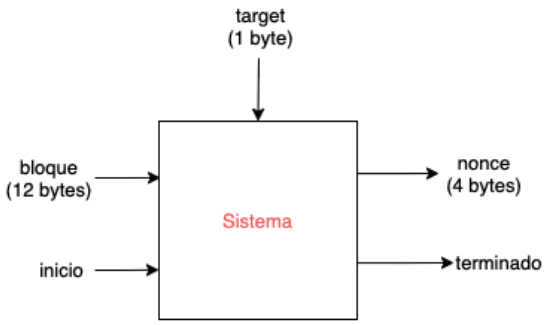

Figura 1. Sistema a generar mapas de congestión de celdas, metal 4 y pines.

Una implementación del sistema fue diseñada priorizando un reducido uso de área (i.e., menores componentes y lógica simplificada), y otra priorizando desempeño aunque utilice mayor área y más componentes, estas pueden ser consultadas en las secciones II y III. Las secciones IV, V y VI muestran el proceso de construcción de los mapas de congestión de celdas, metal 4 y pines respectivamente que consiste en leer y procesar la información de los archivos .def de manera que estos puedan ser leídos por un *script* de *Python* que interpretará estos datos para poder generar un mapa de calor de los elementos solicitados. Los resultados obtenidos de la construcción de los mapas y medición de los tiempos de síntesis se muestran en la sección VII de resultados y finalmente en la sección VIII se puede leer las conclusiones del trabajo realizado. El código del trabajo realizado puede ser explorado en https://github.com/leus8/micro-ucr-hash-rtl.

II. Arquitectura optimizada en área

El principio del diseño de una arquitectura del Sistema presentado en la figura 1 consiste en lograr el funcionamiento correcto mediante la menor cantidad de bloques posibles, en este caso se debe evitar utilizar bloques redundantes para el funcionamiento del sistema ya que un bloque significa mayor área necesitada para construir el ASIC del sistema.

El diagrama de bloques del diseño de la arquitectura optimizada para el área se puede observar en figura 2.

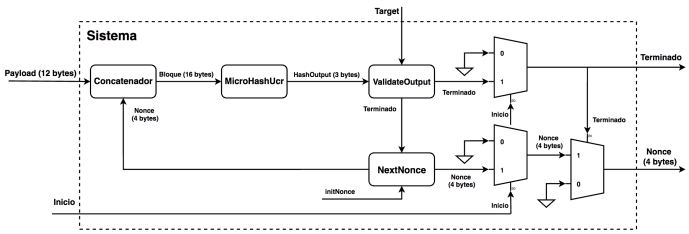

Figura 2. Arquitectura del sistema optimizada para área.

La arquitectura consiste en las dos entradas que debe tener el sistema que es el bloque de 12 bytes y la señal de inicio. En caso de que la señal de inicio sea 0 el circuito estará apagado y no se tendrá ninguna señal en la salida.

Al poner la señal de inicio en 1 entra los 12 bytes de la entrada (en esta implementación se le llamará *Payload*) donde entrará al bloque `Concatenador` que concatena el *payload* de 12 bytes de la entrada y el *nonce* respectivo para la iteración (para la primera iteración el nonce inicial será [0,0,0,0]) y producirá el bloque de 16 bytes que entra a la función `MicroHashUcr`.

La función `MicroHashUcr` producirá una salida de 3 bytes que en la implementación realizada se nombra como *HashOutput* que entra al bloque `ValidateOutput` el cual revisa que el valor de los dos primeros bytes sea menor que el *Target* que recibe en la entrada. Si `ValidateOutput` valida que la salida producida por `MicroHashUcr` es correcta se coloca la señal de *Terminado* en 1 y se envía a la salida el *nonce* con el que se ha validado el resultado. En caso de ser incorrecta la salida de `MicroHashUcr` se procede al siguiente *nonce* calculado por el bloque `NextNonce` y se vuelve a realizar el proceso hasta que se valide un *HashOutput* de `MicroHashUcr`.

III. Arquitectura optimizada en velocidad

Para la arquitectura enfocada en desempeño y velocidad en la generación de nonces válidos se tomó el diseñó preliminar de área y se decidió paralelizar ciertos componentes en lugar de realizar todo serialmente. Claro, esto significa mayor cantidad de módulos en su implementación en un circuito integrado, y por lo tanto mayor área del ASIC que habría que cubrir.

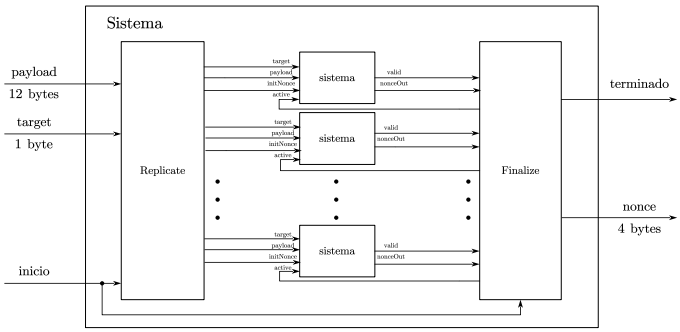

Figura 3. Arquitectura del sistema optimizada para desempeño.

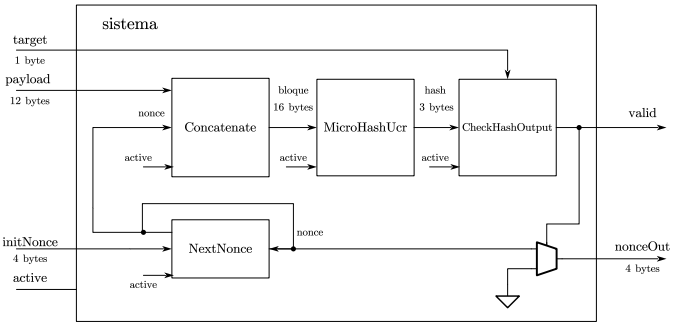

Figura 4. Submódulos paralelizados de Sistema.

El diagrama general de bloques propuesto para el sistema se observa en la Figura 3. En él se observan N submódulos `sistema` que correrán y procesarán nonces diferentes en paralelo. Estos submódulos son descritos por el diagrama de la Figura 4.

La funcionalidad de este diseño en teoría es la misma que la de la optimizada en área, salvo que logrará encontrar un nonce válido mucho más rápido gracias a la paralelización del proceso principal.

IV. Mapa de congestionamiento de celdas

Para generar el mapa de congestionamiento de celdas se trabajó con los archivos `sistema_area.def` y `sistema_velocidad.def`.

La información necesaria para generar el mapa de congestión se encuentra en un bloque de información con una estructura similar a la siguiente:

```
COMPONENTS 13663 ;
- NOR3X1_6 NOR3X1 + PLACED ( 20920 25050 ) FN ;
- FILL_0_NAND3X1_150 FILL + PLACED ( 21560 25050 ) FN ;
- FILL_1_NAND3X1_150 FILL + PLACED ( 21640 25050 ) FN ;
- NAND3X1_150 NAND3X1 + PLACED ( 21720 25050 ) FN ;
- FILL_0_AOI21X1_87 FILL + PLACED ( 22040 25050 ) N ;
- AOI21X1_87 AOI21X1 + PLACED ( 22120 25050 ) N ;
- FILL_0_INVX1_52 FILL + PLACED ( 22440 25050 ) FN ;
- INVX1_52 INVX1 + PLACED ( 22520 25050 ) FN ;
- FILL_0_AND2X2_244 FILL + PLACED ( 22680 25050 ) FN ;
- AND2X2_244 AND2X2 + PLACED ( 22760 25050 ) FN ;
END COMPONENTS
```

Cabe resaltar que el extracto anterior fue construido como ejemplo con los archivos `sistema_area.def` y `sistema_velocidad.def` pero no representan su bloque de información completo. Para generar el mapa de congestionamiento basta con leer las coordenadas dentro de los paréntesis pero para poder hacerlo se debe eliminar la información que está de sobra y para poder realizarlo se utilizan el comando `sed` en conjunto con las expresiones regulares de Unix para poder extraer las coordenadas necesitadas.

El primer paso consistió en eliminar las líneas con las palabras *COMPONENTS* aunque fueron de gran ayudar para poder encontrar esta información estas se tuvieron que eliminar como siguiente paso en el procesamiento de las coordenas y se termina con el siguiente archivo:

```
- NOR3X1_6 NOR3X1 + PLACED ( 20920 25050 ) FN ;
- FILL_0_NAND3X1_150 FILL + PLACED ( 21560 25050 ) FN ;
- FILL_1_NAND3X1_150 FILL + PLACED ( 21640 25050 ) FN ;
- NAND3X1_150 NAND3X1 + PLACED ( 21720 25050 ) FN ;
```

```
- FILL_0_AOI21X1_87 FILL + PLACED ( 22040 25050 ) N ;
- AOI21X1_87 AOI21X1 + PLACED ( 22120 25050 ) N ;
- FILL_0_INVX1_52 FILL + PLACED ( 22440 25050 ) FN ;
- INVX1_52 INVX1 + PLACED ( 22520 25050 ) FN ;
- FILL_0_AND2X2_244 FILL + PLACED ( 22680 25050 ) FN ;
- AND2X2_244 AND2X2 + PLACED ( 22760 25050 ) FN ;
```

El siguiente paso fue eliminar las celdas *FILL* ya que estas no son de interés para la construcción del mapa, el archivo obtenido tiene la siguiente forma:

```
- NOR3X1_6 NOR3X1 + PLACED ( 20920 25050 ) FN ;
- NAND3X1_150 NAND3X1 + PLACED ( 21720 25050 ) FN ;
- AOI21X1_87 AOI21X1 + PLACED ( 22120 25050 ) N ;
- INVX1_52 INVX1 + PLACED ( 22520 25050 ) FN ;
- AND2X2_244 AND2X2 + PLACED ( 22760 25050 ) FN ;
```

Se vuelve a eliminar información siendo esta toda la que no son las coordenadas y se termina con un archivo con la siguiente forma:

```
20920 25050
21720 25050
22120 25050
22520 25050
22760 25050
```

Finalmente para que el script de *Python* que genera el mapa pueda leer la información este tiene que estar en formato .csv, por lo tanto, el último paso de preparación de la información fue cambiar los espacios por comas y se obtiene un archivo con extensión .csv con el siguiente formato:

```
20920,25050
21720,25050
22120,25050
22520,25050
22760,25050
```

A continuación se muestra los comandos de Unix utilizados que resuelven la tarea anterior:

```
sed -n -e '/COMPONENTS.*;/,/END COMPONENTS/ p' archivo.def # Busca el bloque
sed -i '1d;$$d' datos.csv # Elimina demarcador de bloque COMPONENTS.
sed -i '/- FILL/ d' datos.csv # Elimina las celdas FILL.
sed -i 's/.*( \(.*\) ).*/\1/' datos.csv # Elimina lo que no sea coordenadas.
sed -i 's/ /,/' datos.csv # Cambia espacios por ,
```

Los comandos anteriores no fueron exactamente los utilizados pero son muy parecidos, en la implementación lo que cambian son las rutas y nombres de archivos.

Una vez en formato .csv el *script* lee este archivo de la forma (x,y). Por la posición en que se coloca el chip en las etapas de síntesis este no se encuentra centrado en (0,0), por lo tanto, se debe ajustar estas coordenadas para que puedan ser interpretadas de forma correcta por la librería que crea el mapa de congestionamiento. Seguidamente la librería utilizada `matplotlib` se encarga de tomar estos puntos, crear una matriz (histograma 2D) y cuantificar que tantas coordenadas se encuentran dentro de cada área y finalmente generar el mapa de congestionamiento de celdas.

V. Mapa de congestionamiento de metal 4

Para obtener el mapa de congestamiento de metal 4 se debe primero identificar los tracks correspondientes al metal 4. En el inicio del archivo DEF se denotan los siguientes tracks:

```
TRACKS Y -300 DO 947 STEP 100 LAYER metal1 ;
TRACKS X -320.0 DO 1239 STEP 80 LAYER metal2 ;
TRACKS Y -300 DO 947 STEP 100 LAYER metal3 ;
TRACKS X -320.0 DO 1239 STEP 80 LAYER metal4 ;
TRACKS Y -300 DO 947 STEP 100 LAYER metal5 ;
TRACKS X -320.0 DO 620 STEP 160 LAYER metal6 ;
```

De donde sale que para el metal 4:

- Existen 1239 tracks totales
- Los tracks están espaciados 80 dbuPerMicron (i.e. 800 nm)
- Están orientados verticalmente empezando en -320

Luego de obtenida esta información, se pasa a localizar las definiciones de metal 4 en el archivo DEF. Estan se encuentran localizadas en la sección de `NETS` de la siguiente forma, de donde se dibujó manualmente flechas apuntando a las conexiones de metal 4:

```
NETS 19878 ;
+ ROUTED metal1 ( 48800 56600 ) M2_M1
  NEW metal2 ( 48800 56600 ) ( * 57000 ) M3_M2
  NEW metal3 ( 48800 57000 ) ( 46960 * ) M4_M3
  NEW metal4 ( 46960 57000 ) ( * 60400 ) M4_M3 <-
  NEW metal3 ( 46960 60400 ) ( 40880 * ) M3_M2
  NEW metal2 ( 40880 60400 ) ( * 61500 ) M2_M1
  NEW metal1 ( 40880 61500 ) ( 40800 * )
  NEW metal1 ( 54560 50600 ) M2_M1
  NEW metal2 ( 54560 50600 ) ( * 51000 ) M3_M2
  NEW metal3 ( 54560 51000 ) ( 50640 * ) M4_M3
  NEW metal4 ( 50640 51000 ) ( * 53700 ) M4_M3 <-
```

Los dos ejemplos de metal 4 anteriores están *routed* de forma vertical:

1. `NEW metal4 ( 46960 57000 ) ( * 60400 )` Indica que:
   - Empieza en 46960 y termina en 46960 del eje X
   - Empieza en 57000 y termina en 60400 del eje Y
2. `NEW metal4 ( 50640 51000 ) ( * 53700 )` Indica que:
   - Empieza en 50640 y termina en 50640 del eje X
   - Empieza en 51000 y termina en 53700 del eje Y

Para efectos del procesamiento de cada uno de estos, se:

1. Extrajeron todas las líneas que contenían “NEW metal4”
2. Eliminaron casos especiales como la creación de vias.
3. Eliminó el inicio y final de cada línea tal que quedara solamente los paréntesis con la ubicación de los metales.
4. Eliminaron los paréntesis para quedara solamente los números y asteriscos separados por coma.

Los comandos de `sed` de Unix que cumplen lo anterior tienen un formato similar a:

```
sed -n -e '/NEW metal4/ p' archivo.def > datos.csv
sed -i '/viagen/ d' datos.csv
sed -i 's/^\(  NEW metal4 \)*//' datos.csv
sed -i 's/M.*//' datos.csv
sed -i 's/[^0-9* ]*//g' datos.csv
sed -i 's/^\( \)*//' datos.csv
sed -i 's/  */ /g' datos.csv
sed -i 's/ /,/g' datos.csv
```

Al final de todo el procesamiento se crea un archivo .csv con la siguiente forma:

```
46960,57000,*,60400,
50640,51000,*,53700,
58960,58700,*,51000,
63840,55500,*,58700,
67360,45600,*,55500,
55760,31700,*,51000,
46240,18900,*,27900,
```

El script de Python es corrido sobre los valores separados por coma de forma tal que completa los puntos del eje Y

faltantes que comprende un segmento de metal. Es decir si procesa `46960,57000,*,60400` llenará:

- Eje X: `[46960, 46960, 46960, ..., 46960]`
- Eje Y: `[57000, 57001, 57002, ..., 60400]`

Luego de procesar cada uno de las entradas del archivo CSV procede a crear y guardar el mapa de congestionamiento con un histograma 2D utilizando la librería `matplotlib` y definiendo la cantidad de *cuadrados* o *bins* que tendrá el mapa.

Con esta cantidad de *bins* se calcula el promedio de tracks por *bin* para el eje X (puesto que para el metal 4 los tracks están definidos verticalmente).

$$\text{Promedio tracks} = \frac{\text{Tracks totales}}{\text{Cantidad de bines del eje X}}$$

Luego, se puede calcular la utilización de tracks para cada cuadrado. Primero se define el máximo de utilización en un sólo cuadrado:

$$\text{Utilización máx.} = (\Delta Y) \cdot \text{Promedio tracks}$$

Donde ΔY se refiere a la longitud Y que miden los cuadrados.

Seguidamente se calcula el porcentaje de utilización por cuadrado:

$$\text{Utilización Bin } (\%) = \frac{\text{Cantidad de metal en el Bin}}{\text{Utilización máx.}} \cdot 100$$

Donde "Cantidad de metal en el Bin" se extrae de la información provista por el histograma 2D creado de `matplotlib`.

VI. Mapa de congestionamiento de pines

Para generar el mapa de congestionamiento de pines igual que con los puntos anteriores se trabajó con los archivos `sistema_area.def` y `sistema_velocidad.def`.

Para generar este mapa se debe preparar la información contenida dentro de estos archivos de manera que pueda ser trabajada posteriormente con el *script* de *Python*. Las herramientas utilizadas fueron los comandos de Unix `sed` y `grep` en conjunto con las expresiones regulares.

El bloque de información necesario para generar el mapa tiene la siguiente estructura:

```
PINS 165 ;
- payload[0] + NET payload[0]
  + LAYER metal3 ( -15 -15 ) ( 15 15 )
-240 24700 ) N ;
- payload[1] + NET payload[1]
  + LAYER metal2 ( -15 -15 ) ( 15 15 )
  + PLACED ( 18000 46300 ) N ;
- payload[2] + NET payload[2]
  + LAYER metal2 ( -15 -15 ) ( 15 15 )
  + PLACED ( 12480 46300 ) N ;
- payload[3] + NET payload[3]
  + LAYER metal2 ( -15 -15 ) ( 15 15 )
  + PLACED ( 21920 -200 ) N ;
END PINS
```

Los datos necesitados corresponden a las coordenadas mencionadas en las líneas con la palabra *PLACED*. Por lo tanto como primer paso se debe eliminar toda las líneas que no correspondan a las que contiene la palabra *PLACED* y se termina con un archivo con la siguiente estructura:

```
  + PLACED ( -240 24700 ) N ;
  + PLACED ( 18000 46300 ) N ;
  + PLACED ( 12480 46300 ) N ;
  + PLACED ( 21920 -200 ) N ;
```

El siguiente paso corresponde eliminar todos los caracteres que no corresponden a los números de las coordenadas, al final del procesamiento se obtiene la siguiente estructura:

```
-240 24700
18000 46300
12480 46300
21920 -200
```

Finalmente se cambian los espacios por comas de manera que se pueda generar un archivo con extensión .csv que pueda ser procesado por el *script*. Se obtiene un archivo .csv con la siguiente estructura:

```
-240,24700
18000,46300
12480,46300
21920,-200
```

De forma parecida a los programas anteriores, el *script* va interpretar los datos de este archivo de la forma (x,y) y hará un ajuste a las coordenadas como fue explicado anteriormente y finalmente mediante la librería `matplotlib` se creará un histograma 2D que cuantifica que tantas coordenadas se encuentran dentro de cada área y generará el mapa de congestionamiento de pines.

VII. Resultados

VII-A. Área

Se inicia ejecutando el proceso de síntesis con una densidad inicial del 0,7 (esto es necesario para crear los mapas de congestión) calculando la cantidad de DRCs y los tiempos de ejecución de todas las etapas dentro del programa *Qflow* para el sistema optimizado en área. El cuadro I muestra los resultados obtenidos.

Cuadro I
Reporte sobre cantidad de DRCs y tiempos de ejecución de las etapas de síntesis de Qflow para el sistema optimizado en área

Medición	**Resultado**
Cantidad de DRCs	0
Tiempo de ejecución	**Duración (min:s)**
Síntesis	00:08
Placement	01:15
Static timing analysis	00:03
Routing	04:05
Post-Route STA	00:04
Migration	00:42
DRC	00:06
LVS	00:02
GDS	00:09

Del cuadro I se puede observar que como no hubo errores de DRCs se logró ejecutar todas las etapas de síntesis con tiempos cortos donde el más largo fue la etapa de *routing* y la más corta fue el LVS.

Posteriormente se procede a crear los mapas de congestión para el sistema optimizado en área. El mapa de congestión de celdas se observa en la figura 5

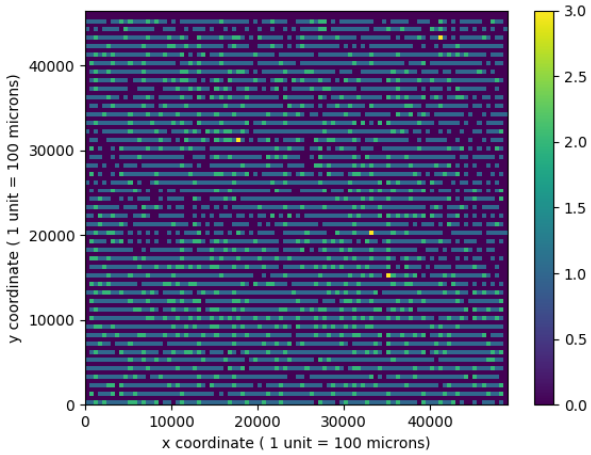

Figura 5. Mapa de congestión de celdas para el sistema optimizado en área.

De la figura 5 se observa que la distribución de las celdas en el sistema están distribuidas mediante líneas horizontales uniforme y existen muy pocos puntos donde exista un mayor congestionamiento.

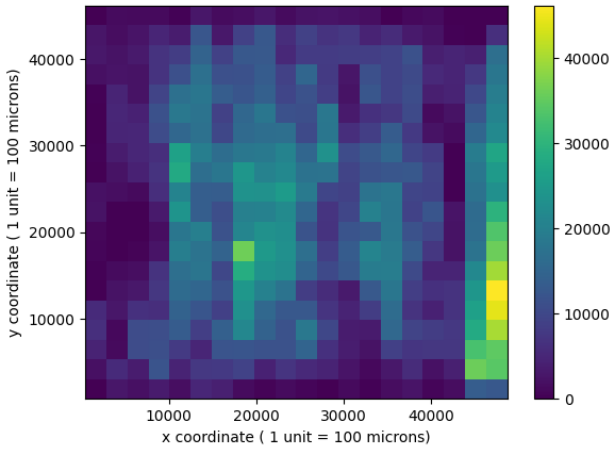

Figura 6. Mapa de congestión de metal 4 para el sistema optimizado en área.

Para el mapa de congestión, promedio y utilización del metal 4 se eligió 20x20 cuadrados iniciales principalmente para obtener una matriz relativamente pequeña de utilización y poder desplegarla en el reporte. Del mapa de congestión de la figura 6 se observa que existe una mayor congestión cerca del borde derecho del *chip*. Este comportamiento que también se observa en la utilización de tracks calculado y desplegado en la terminal como en la figura 7. Además se observa que en promedio existen 30.7 tracks por cuadrado.

Posteriormente se experimentó con mayor cantidad de cuadrados para poder mostrar un mapa de mejor resolución como el de la figura 13, en el que se visualiza el comportamiento vertical del metal 4. Se confirma que como tal el diseño optimizado en área no presenta mucho congestionamiento,

```
Average tracks per bin: 30.7
Track usage per bin (%):
  0   1   1   1   0   3   1   2   3   2   3   3   3   2   0   1   1   0   0   0
  3   2   3   6   5  17   4  12  17   8  12   9  10   6   8   4   3   0   0   8
  4   3   4   9  11  20  12  17  18   7  11  11  11  12  13  16   5   6   5  23
  2   3   3   9  16  24  16  16  19  12  21  11   3  16  13  14   7   6  10  26
  0   6   3  13  24  25  19  17  19  13  16  11   3  17  13  15   5   6  14  27
  0   7   6  16  26  25  12  21  26  16  15  26   4   9  10  14   1   3  17  29
  0   6   7  15  21  24  14  22  23  23  14  26   8  11  18  11   3   1  21  31
  0   5   7   8  38  28  23  26  26  29  21  33  15  18  21  13  11   0  27  38
  0   3   6   9  39  23  26  34  29  30  25  25  13  16  17  17  13   0  26  35
  2   2   1   8  29  19  19  33  33  36  27  13  12  25  26  13  13   0  24  33
  3   0   0   4  34  19  15  28  28  31  21  10  13  28  25  13  17   3  22  42
  1   0   0   2  26  28  16  30  33  32  16   9  17  18  26  16  13   3  23  47
  0   0   0   3  28  25  20  51  35  33  23  11  18  29  27  19  10   3  25  51
  0   1   2   9  23  25  14  40  33  32  21  14  17  27  27  14   6   6  31  57
  0   3   4  10  22  21  16  34  28  25  11   8   4  18  25  12   9   9  34  66
  7   3   9   8  20  17  12  33  22  17  18   9   6  11  23   9   5   9  37  63
  8   1  15  15  18  12  20  30  20  16  26  14   5   4  22  13  11  10  40  57
  6   1  15  15  11   6  17  17  16  16  21   9   5   3  12  10  10  10  47  49
  2   7   5  17   6  10  10  13   6   9   7   6   3   1   4   8   3   5  51  48
  0   3   2   4   2   6   5   1   0   1   0   0   1   0   2   3   0   2  19  17
```

Figura 7. Promedio y utilización de tracks por cuadrado para 20x20 cuadrados en el sistema optimizado en área.

sólo un poco en el borde derecho. Para este mapa se calculó 1.535 tracks por cuadrado. Sin embargo no se muestra su matriz de utilización de metal 4, debido a su tamaño extenso (400x400).

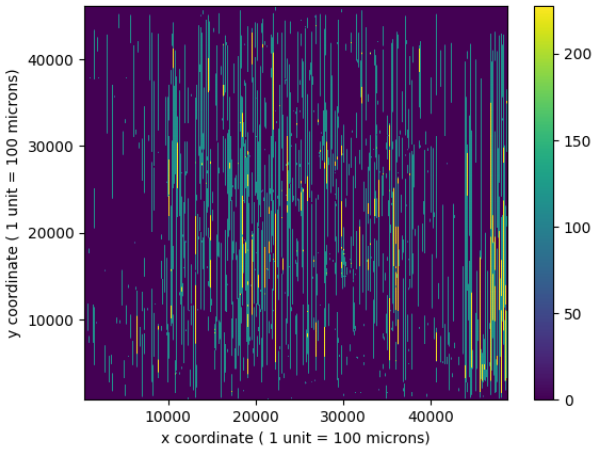

Figura 8. Mapa de congestión de metal 4 con 400x400 cuadrados para el sistema optimizado en área.

Finalmente como un extra al trabajo realizado se obtiene el mapa de congestión de pines, que se puede observar en la figura 9.

En la figura 9 los espacios blancos corresponde a una congestión de 0. Se obtiene un resultado esperado puesto que el congestionamiento se encuentra en los bordes del sistema, de esta figura se observa que los bordes laterales presentan el mayor congestionamiento de pines.

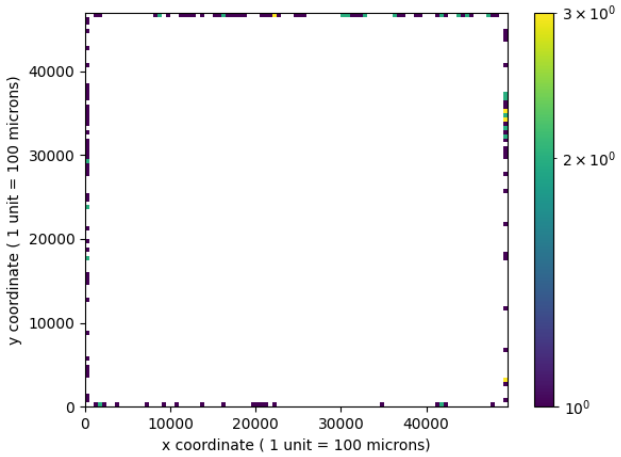

Figura 9. Mapa de congestión de pines para el sistema optimizado en área.

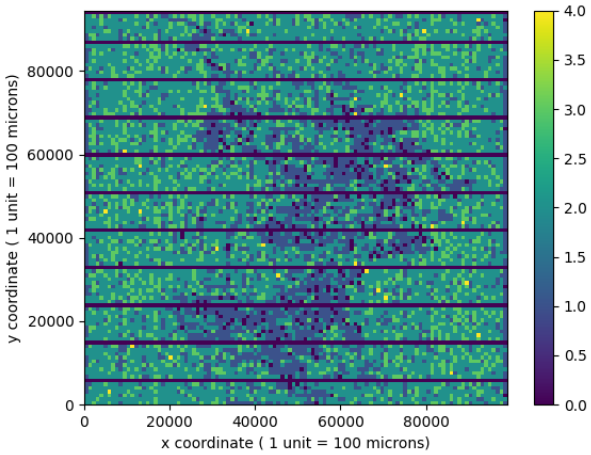

Figura 10. Mapa de congestión de celdas para el sistema optimizado en velocidad.

VII-B. Velocidad

Respecto al sistema optimizado en velocidad se inicia ejecutando el proceso de síntesis con una densidad inicial del 0,7 (esto es necesario para crear los mapas de congestión) calculando la cantidad de DRCs y los tiempos de ejecución de todas las etapas dentro del programa *Qflow*. El cuadro I muestra los resultados obtenidos.

Cuadro II
REPORTE SOBRE CANTIDAD DE DRCS Y TIEMPOS DE EJECUCIÓN DE LAS ETAPAS DE SÍNTESIS DE QFLOW PARA EL SISTEMA OPTIMIZADO EN VELOCIDAD

Medición	**Resultado**
Cantidad de DRCs	10
Tiempo de ejecución	**Duración (min:s)**
Síntesis	00:45
Placement	25:40
Static timing analysis	00:03
Routing	32:10
Post-Route STA	00:12
Migration	05:15
DRC	00:25

Esta vez se obtienen 10 errores de DRCs y por lo tanto el flujo de diseño de Qflow no podría avanzar más de esta etapa sin antes arreglarlos. Los tiempos de ejecución para este sistema son mucho más largos comparados al optimizado en área. Es de esperar, pero refleja el esfuerzo que realizan las herramientas que corren por debajo de Qflow para sintetizar e implementar sistemas más complejos. Para las etapas de mayor duración, se observa:

- Un incremento de 2005 % para la duración del placement.
- Un incremento de 787 % para la duración del routing.

Se procede a generar el mapa de congestión de celdas para el sistema optimizado en velocidad que se puede observar en la figura 10.

Se observa que de forma muy parecida a la figura 5 del sistema optimizado en área, las celdas fueron colocadas en tiras horizontales de manera uniforme. En este mapa si se puede observar que en las zonas centrales del chip se tiene un menor congestionamiento de las celdas.

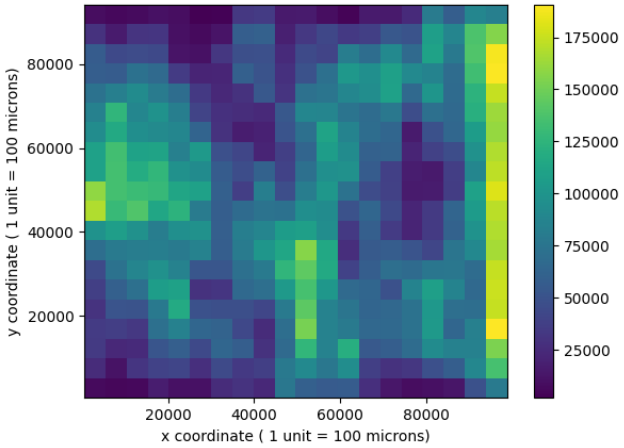

Figura 11. Mapa de congestión de metal 4 para el sistema optimizado en velocidad.

Para el mapa de congestión, promedio y utilización del metal 4 se inició nuevamente con 20x20 cuadrados iniciales y en el mapa de la figura 11 se observa que existe una mayor congestión cerca del extremo izquierdo central y la totalidad del extremo derecho con un mayor congestionamiento en general presente en el chip. Este comportamiento es reflejado en la utilización de tracks calculado y desplegado en la terminal como en la figura 12. Se observa que en promedio existen 61.95 tracks por cuadrado.

Posteriormente se experimentó nuevamente con mayor cantidad de cuadrados para poder mostrar un mapa de mejor

```
Average tracks per bin: 61.95
Track usage per bin (%):
  2   1   2   3   3   0   1  11  11   5   5   3   1   5   5   8  27  19  30  28
  9   5  10  10   3   2   4  19  20   6  11  14  11  12  18  17  36  30  47  53
 19  14  15  14   3   3  11  16  16   8  15  13  27  23  31  22  38  26  47  64
 19  19  24  22  11   7   7  20  17   9  16  24  35  32  37  31  29  23  45  65
 25  28  31  27  23  11   7  10  18  10  24  25  30  32  37  22  34  23  42  59
 28  43  30  33  28  13   9   8   8  18  30  30  25  23  26  15  24  22  41  56
 31  39  32  31  30  21   9   5   6  17  24  38  29  23  21   5  16  22  36  55
 37  45  37  38  28  35  16  14   7  12  21  36  31  20  20   7  13  15  33  57
 37  46  40  43  33  31  19  15  20  13  24  32  33  19  12   5   5  12  30  59
 54  46  44  44  38  37  20  12  28  15  27  35  25  15  13   5   5  12  37  62
 57  44  47  39  42  27  19  23  25  24  32  31  16  21  14   7  11  21  36  58
 33  36  33  28  32  33  20  28  30  36  37  32  11  20  11   7  11  20  33  57
 23  27  28  28  27  29  19  27  34  39  53  39   9  18  19  17  14  25  31  56
 15  25  30  33  29  18  18  24  27  43  49  39  13  19  24  23  26  26  23  59
 12  20  24  36  37   9  10  22  24  35  46  35  22  21  16  29  36  29  22  60
 10  17  17  26  40  16  16  16  14  27  49  29  25  21  22  25  35  24  23  59
 12  12  11  24  29  21  21  14   8  23  52  29  27  22  22  25  35  21  28  65
 11   7   8  17  20  14  21  19   9  22  43  29  41  18  19  23  31  28  28  52
  8   3  10  13  13   9  15  11  12  26  31  26  32  12   9  10  16  19  30  46
  2   1   2   5   3   3   8  10   7  26  20  19  19   9   4   2   3   4  16  27
```

Figura 12. Promedio y utilización de tracks por cuadrado para 20x20 cuadrados en el sistema optimizado en velocidad.

resolución como el de la figura 13, en el que se visualiza el comportamiento vertical del metal 4. Asimismo se muestra el congestionamiento en el borde izquierdo y derecho. Para este mapa se calculó 3.0975 tracks por cuadrado. Sin embargo no se muestra su matriz de utilización de metal 4, debido a su tamaño extenso (400x400).

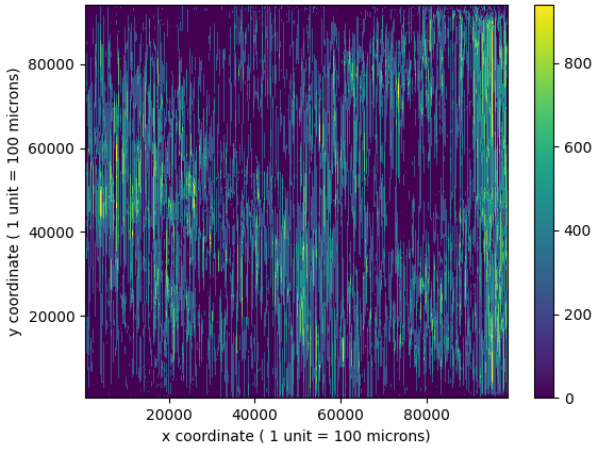

Figura 13. Mapa de congestión de metal 4 con 400x400 cuadrados para el sistema optimizado en velocidad.

Igualmente se realiza el trabajo extra para este sistema que es el mapa de congestionamiento de pines, este se puede observar en la figura 14.

Vemos que se obtiene el mismo resultado esperado de la figura 9 donde la congestión se encuentra en los bordes del chip y las zonas color blanco corresponde a una congestión de 0. Para este caso la sección con mayor congestión en el borde inferior.

VIII. Conclusiones

- Para el sistema optimizado en área no se obtienen errores de DRCs y logra completar el proceso completo de síntesis mediante tiempos de ejecución rápidos.

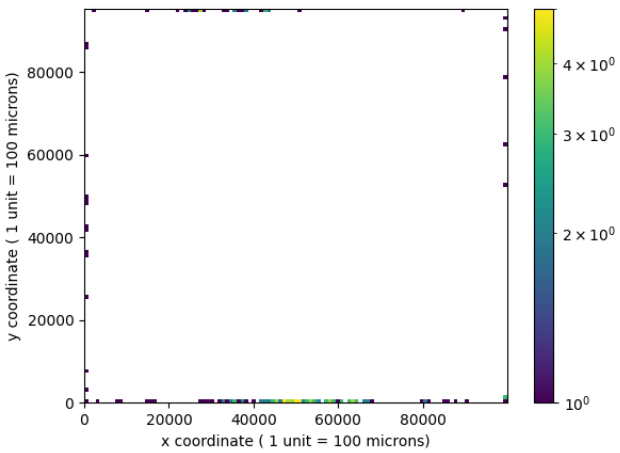

Figura 14. Mapa de congestión de pines para el sistema optimizado en velocidad.

- El sistema optimizado en desempeño diseñado por ahora presenta 10 errores de DRCs que se tendrán que corregir en el futuro para lograr terminar el flujo de diseño. Además tarda hasta 2005 % más en ejecutar el flujo comparado al sistema optimizado en área.
- El mapa de congestionamiento de celdas para el sistema optimizado en área presenta una distribución de tiras horizontales uniformes, lo mismo ocurre para el sistema optimizado en velocidad a diferencia que si se puede observar una menor congestión en el área central.
- Se denota un menor congestionamiento del metal 4 para el sistema optimizado en área. Con una mayor densidad cerca del borde derecho y un poco en el área central. Este mismo comportamiento es reflejado en la matriz de utilización de metal de los cuadrados del mapa.
- Se denota un mayor congestionamiento del metal 4 para el sistema optimizado en desempeño. Con una mayor densidad cerca del borde derecho, en el área central del borde izquierdo y un poco en el área central. Este mismo comportamiento es reflejado en la matriz de utilización de metal de los cuadrados del mapa.
- El mapa de congestionamiento de pines para ambos sistemas se manifiesta con una congestión en los bordes del chip.