ESTUDIO DE ALGORITMOS EN COMPUTACIÓN PARALELA PARA HERRAMIENTAS DE DISEÑO ELECTRÓNICO

Walter Aróztegui[1] [2] *, Fernando Tinetti*[1] [2]*, Jorge Osio*[1]*, José Rapallini*[1]*, Antonio Quijano*[1]

Universidad Nacional de La Plata / Facultad de Ingeniería / Centro de Técnicas Analógico Digitales (CeTAD).[1] - Facultad de Informática / Instituto de Investigación en Informática LIDI (III-LIDI).[2]

waroz@graffiti.net; fernando@info.unlp.edu.ar; josio@gioia.ing.unlp.edu; josrap@ing.unlp.edu.ar; quijano@ing.unlp.edu.ar

RESUMEN

En el marco de desarrollo de herramientas de diseño electrónico, para sistemas de diseño de ciruitos integrados (ASICs), se hace necesaria la resolución de sistemas de ecuaciones lineales, tanto más grandes cuanto más complejidad tenga el circuito a diseñar. La ejecución de aplicaciones en forma paralela en varias computadoras aparece como una solución efectiva para conseguir la potencia de cálculo necesaria .

En este artículo se presenta una estrategia de paralelización de clusters del método de Gauss-Seidel para la solución de sistemas de ecuaciones ralas. Desde el punto de vista de la solución numérica para matrices de coeficientes con poca densidad de elementos no nulos, se siguen los lineamientos estándares, es decir, esquemas de almacenamientos especiales (solamente se almacenan los elementos no nulos) y métodos iterativos de búsqueda de solución por aproximaciones sucesivas.

Desde el punto de vista de la paralelización del procesamiento en clusters de computadoras, se siguen dos principios básicos: distribución de carga de procesamiento aún en el caso de clusters heterogéneos y utilización de mensajes broadcast para toda comunicación de datos entre procesos. La interconexión más usual en clusters es la provista por las redes Ethernet, y por lo tanto pueden implementar los mensajes broadcast a nivel físico con sobrecarga mínima.

Se presenta el análisis de rendimiento paralelo y además los resultados obtenidos en una red local de computadoras heterogénas no dedicadas. En este caso se utiliza una implementación de la biblioteca MPI (Message Pasage Interface) para la comunicación entre procesos.

1. INTRODUCCIÓN

La complejidad de los ASICs hace totalmente indispensable el uso de herramientas capaces de automatizar el diseño, al menos en las tareas mas tediosas y más complejas en cuanto a cálculo numérico, asociadas a éste. Tales herramientas pueden ser de varios tipos y de utilización en diferentes etapas de los procesos de diseño: verificaciones estáticas, verificaciones dinámicas con simuladores y síntesis de circuitos a diferentes niveles como síntesis VHDL, Verilog, de Vectores de Test o "Placement & Routing" (Ubicación y Conexionado) desde esquemáticos, particionamiento de sistemas implementados con técnicas de codiseño, etc..

Los simuladores eléctricos tipo Spice calculan la tensión v(t) y la i(t) en cada nodo del circuito planteando y resolviendo un sistema de ecuaciones diferenciales basadas en la aplicación de las reglas de Kirchoff.

Los dispositivos (transistores, en general) se modelan por sus ecuaciones de funcionamiento y mediante los parámetros tecnológicos suministrados por el fabricante. Si se utiliza un modelo muy complejo, los resultados de las simulaciones son muy precisos, pero el costo computacional y por lo tanto el tiempo empleado es muy alto.

Así como mencionamos, se ven involucrados un gran número de ecuaciones diferenciales (EDPs) y la manera más común de resolverlas es discretizándolas, es decir, aproximándolas por ecuaciones que contienen un número finito de incógnitas. Se producen de esta manera grandes sistemas de ecuaciones lineales ralos o dispersos (sparse) del tipo que se tratan más adelante en este trabajo.

Para conseguir la potencia de cálculo necesaria, desde hace unos pocos años los avances incluidos en los microprocesadores llamados "domésticos" o de oficina y una más que aceptable relación costo-beneficio determinan la tendencia a la utilización de computadoras paralelas formadas por redes o *clusters* de PCs ya sean homogéneos, cuando todas las PCs son del mismo tipo (que es la definición más comúnmente aceptada de cluster) o heterogéneos si se trata de diferente clases, como los que se pueden encontrar en ámbitos variados y cotidianos.

Los métodos de resolución de sistemas de ecuaciones lineales pueden clasificarse en directos e indirectos, en los métodos directos, las matrices que representan a los sistemas a calcular son llevados mediante transfor-

maciones a formas más simples y éstas calculadas directamente. Un Ejemplo característico es el método de eliminación gaussiana que transforma la matriz que representa el sistema original en una de tipo triangular cuya resolución es simple y directa. Los métodos iterativos o indirectos dan lugar a una sucesión de vectores que idealmente convergen a una solución del sistema mediante la repetición de un proceso sencillo. El cálculo se detiene cuando se cuenta con una solución aproximada con cierto grado de precisión especificado de antemano o después de un determinado número de iteraciones [1],[3],[5].

En el caso especial de sistemas ralos o dispersos, en los que una proporción grande de los elementos de la matriz del sistema son ceros, los métodos iterativos son particularmente eficientes. Más aún cuando se emplean esquemas de almacenamiento especiales para matrices ralas, esto se puede verificar al considerar un sistema con una baja densidad de elementos no nulos. Utilizando un método directo, las transformaciones sobre ceros pueden producir elementos distintos de cero, por lo que debe trabajarse con el sistema como si fuera denso. Al utilizar un método iterativo, los coeficientes nulos se mantendrán invariables y puede prescindirse de ellos, con el resultante ahorro de espacio de memoria y tiempo de procesamiento al no tener que acceder ni utilizar operandos nulos.

En las secciones que siguen se mostrarán brevemente los métodos de almacenamiento de matrices ralas y el método iterativo de Gauss-Seidel para resolución de ecuaciones lineales, se introduce la paralelización del método iterativo directamente orientada a clusters interconectados por redes Ethernet y se muestran los resultados obtenidos en la experimentación sobre un cluster en particular.

2. SISTEMAS RALOS: ALMACENAMIENTO Y MÉTODO DE GAUSS-SEIDEL

En esta sección inicialmente se muestra cómo ahorrar espacio en el almacenamiento de matrices con una gran proporción de elementos nulos. Posteriormente se muestra cómo se resuelven los sistemas de ecuaciones con el método de Gauss-Seidel y en qué medida el método se ve afectado por el sistema de almacenamiento de matrices ralas.

De los esquemas de almacenamiento mencionados, probablemente el más popular en matrices ralas, se conoce como ***CSR*** *(Compressed Sparse Row)* o formato de fila rala comprimida (Fig. 1), consta de una estructura de datos que tiene tres arreglos con:

Un arreglo real conteniendo los elementos no-cero de la matriz almacenados por fila.

Un arreglo entero conteniendo los índices de columna respectivos.

Un arreglo entero conteniendo los punteros al inicio de cada fila.

De esta manera se tienen dos arreglos de longitud Nz y otro de longitud igual a n +1 con n igual al número de filas y siendo el último elemento de éste el que indica la cantidad total de elementos no nulos más uno.

Otras formas de almacenamiento de matrices ralas son:

Coordinado, CSC (Compressed Sparse Column), que es una forma análoga a la CSR, y que utiliza la herramienta MatLab por ejemplo, **MSR** (Modified Sparse Row), por **diagonales** y **Ellpack-Itpack** [4],[7].

Como sabemos, los métodos iterativos intentan encontrar la solución del sistema a partir de un vector solución inicial $x_1^{(0)}, x_2^{(0)},\ldots, x_n^{(0)}$ que generalmente se toma como cero, de manera que se producen paso a paso valores mejorados de los x_i . Este proceso se repite hasta llegar a lo que se considera una buena aproximación. Asumiendo que los coeficientes del sistema de ecuaciones son denotados como $a_{i\,j}$ y b_i es el resultado buscado de cada ecuación del sistema, una expresión más general para este procedimiento es:

$$x_{(k)}^{i} = \frac{a^{ii}}{1}\left(b^{i} - \sum^{i \neq j} a^{ij} x_{(k-1)}^{j} \right) \qquad (1)$$

que como mínimo requiere que $a_{i\,i} \neq 0$ y constituye el método de Jacobi, en el que cada aproximación se calcula con los datos de la iteración anterior.

En el método de **Gauss-Seidel** este proceso iterativo se modifica de manera que las ecuaciones subsiguientes utilicen los valores de x calculados en las ecuaciones anteriores, correspondientes a la corriente iteración, produciendo una considerable mejora en la velocidad de convergencia.

Se expresa entonces de la forma:

$$x_i^{(k)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j<i} a_{ij} x_j^{(k)} - \sum_{j>i} a_{ij} x_j^{(k-1)} \right) \quad (2)$$

se debe notar que ninguno de estos dos métodos se ve alterado por la forma de almacenamiento de la matriz de coeficientes [1],[3],[5].. Por supuesto que el acceso a los datos en el caso del almacenamiento CSR para matrices ralas debe ser realizado utilizando los vectores mencionados antes, pero esto no cambia las características numéricas del algoritmo (cantidad de iteraciones y valores del vector solución) sino la cantidad de operaciones realizadas dado que no se utilizan los valores nulos.

Se ha presentado la paralelización del método de Gauss-Seidel por varias razones:

Es uno de los más representativos entre los algoritmos iterativos más simples.

Es una evolución del método de Jacobi, con mejor tiempo de convergencia, y puede tomarse como un caso particular de otro procedimiento muy conocido como es el de sobrerrelajación SOR (Successive Overrelaxation)

La dependencia de datos en los cálculos es suficientemente fuerte como para no estar en el mejor de los casos para los requerimientos de cómputo, como es

el caso de Jacobi, donde una vez que se tiene una solución, los cálculos para la siguiente solución son todos independientes entre sí y por lo tanto su paralelización es inmediata.

3. PARALELIZACIÓN DE GAUSS-SEIDEL

Como se menciona antes, la paralelización del método de Gauss-Seidel no es inmediata dada la dependencia de datos que existe para calcular los valores de cada iteración Ya dijimos que el método de almacenamiento elegido es el CSR (Compressed Sparse Row). La partición o distribución de los datos a procesadores entre los procesos o procesadores se realiza por bloques de filas en partes iguales (en este caso, el clúster se toma como homogéneo). Si por ejemplo se realiza la partición de un sistema de ecuaciones entre tres procesos P_0 , P_1 y P_2 , el proceso P_0 tendrá asignado el primer tercio de ecuaciones (filas de la matriz de coeficientes), el proceso P_1 el segundo tercio y el proceso P_2 tendrá el tercer y último tercio de filas de la matriz de coeficientes del sistema de ecuaciones. Debe recordarse que, por un lado, esta partición es independiente del almacenamiento CSR de una matriz rala, y por otro lado que, dado un bloque de filas, queda automáticamente determinado el conjunto de variables que el proceso puede calcular. Más espe-cíficamente, cada proceso puede calcular el valor de las variables que están asociadas a la diagonal principal de la matriz (rala) de coeficientes, tal como puede derivarse de la Ec. (2).

Una vez identificadas las filas de cada proceso, cada uno se encarga de dividir las filas asignadas por el elemento correspondiente a la diagonal principal respectiva. El sistema quedará:

$$x_i = b_i^* - \sum_{j \neq i} a_{ij}^* x_j \quad \text{con } a_{ij}^* = \frac{a_{ij}}{a_{ii}} \text{ y } b_i^* = \frac{b_i}{a_{ii}}$$

A partir de ahora empieza el proceso iterativo, que se puede separar en varias etapas, y que conviene describir desde la primera iteración:

1.- Inicialmente, es decir con un valor inicial para las incógnitas, el proceso con el primer bloque de filas puede comenzar con el cálculo de las variables asociadas. Es decir que el proceso P_0 con las primeras bs filas puede determinar el valor de las variables $x_1^{(1)},.....,x_{bs}^{(1)}$, directamente a partir de $x_1^{(0)},.....,x_n^{(0)}$. Los demás procesos solamente pueden llevar a cabo los resultados intermedios que corresponden a la segunda sumatoria que aparece en la Ec. (2), es decir que dependen de los valores $x_i^{(0)}$.

2.- Una vez que se tienen calculados los valores $x_1^{(1)},.....,x_{bs}^{(1)}$, todos los demás procesos pueden utilizar estos valores para el cálculo de la primera sumatoria de la Ec. (2). Dado que el proceso P_0 es el que tiene los valores de estas variables y todos los demás los necesitan, es natural pensar en un broadcast desde el proceso P_0 a todos los demás. De hecho, en este segundo paso se completan todos los cálculos necesarios para los valores de $x_{bs+1}^{(1)},.....,x_{2bs}^{(1)}$ en el proceso P_1 que, a su vez, pueden ser enviados a todos los demás procesos para los cálculos que dependen de ellos de acuerdo con la Ec (2). Se debe notar que estos valores $x_{bs+1}^{(1)},.....,x_{2bs}^{(1)}$ se pueden utilizar también en el proceso P_0, pero para el cálculo de las variables de la siguiente iteración de Gauss-Seidel, es decir para el cálculo parcial de los valores $x_1^{(2)},.....,x_{bs}^{(2)}$.

3.- Los dos pasos anteriores se pueden continuar (cálculos parciales llegando a los valores de cada conjunto de variables en los procesos correspondientes) tomando como referencia los procesos P_i hasta el último proceso con el último bloque de filas/variables, donde:

Se tienen los valores para todas las variables de una iteración de Gauss-Seidel (la primera, siguiendo con ejemplo con k=1).

Si se calculan los errores junto con las variables de acuerdo a la definición del error de convergencia, se puede tener el error total correspondiente a la iteración actual de Gauss-Seidel y por lo tanto se puede determinar si seguir o no, de acuerdo con el valor de este error.

Desde el punto de vista de cada proceso, a partir de la inicialización se sigue una secuencia de recepción de datos provenientes de los demás procesos (vía broadcast), cálculos parciales con estos datos, envío de los valores de las variables que se manejan localmente una vez que se tienen completamente calculados (vía broadcast) y determinación de continuar o no de acuerdo al error o a una señal del último proceso que lo indica.

4. Experimentación y resultados obtenidos

En términos de hardware, la implementación se llevó a cabo en una red de PC's interconectadas con Ethernet de 10Mbps. Todas las PCs se utilizaron con el sistema operativo Linux y la implementación LAM (Local Area Multicomputer) de la biblioteca estándard MPI (Message Passage Interface) para pasaje de mensajes entre procesos. Para las comunicaciones sólo se usaron funciones Broadcast, dado que de hecho el algoritmo fue concebido para que esto sea posible, prescindiendo eventualmente de la utilización de un switch en la red, que es el único recurso con que otros métodos pueden eludir las colisiones que aumentan y se hacen importantes al crecer la dimensión de los sistemas a calcular.

A medida que los sistemas crecen en tamaño, el costo de comunicaciones también aumenta, y uno de los propósitos de la experimentación es justamente, determinar el peso relativo de las comunicaciones con respecto al cómputo para este algoritmo. Más específicamente, se debe determinar el rendimiento para distintos tamaños del sistema de ecuaciones con diferentes cantidades de computadoras. Se espera a priori obtener mayor rendimiento (speedup y eficiencia)

con mayor cantidad de máquinas utilizadas y tamaños relativamente grandes de sistemas de ecuaciones.

Resumiendo entonces los experimentos:

Se utilizaron entre una (procesamiento secuencial) y cinco computadoras en paralelo que es la máxima cantidad de PC disponibles en esta investigación, pero suficientes para observar la tendencia en los resultados.

Los tamaños de sistemas de ecuaciones variaron entre 200 y 5000 con sus respectivas incógnitas, y se presentan los resultados como más indicativos, de 1200 y 2400 ecuaciones. El mínimo está dado por la cantidad más chica que tiene sentido paralelizar (el tiempo secuencial es suficientemente grande). El máximo está relacionado con la cantidad más grande de datos que una computadora puede contener en memoria principal sin hacer uso de la memoria swap.

El speedup (métrica para la estimación del rendimiento) se calcula con respecto al procesamiento secuencial sin hacer uso de ninguna primitiva de pasaje de mensajes, es decir que no es afectado por ninguna sobrecarga que tenga relación con la paralelización del algoritmo. Además, el valor de referencia de tiempo de cómputo secuencial se toma en la más lenta de todas las computadoras utilizadas, es decir que se consideran todas como la de menor capacidad de cómputo (cluster homogéneo).

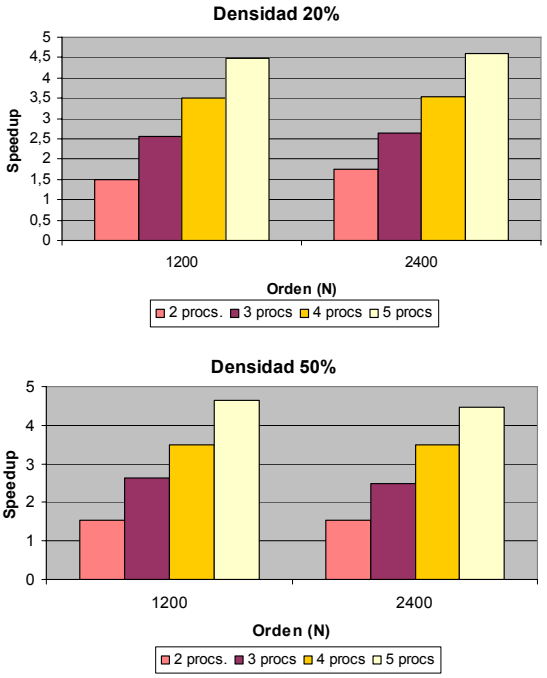

Fig2. Rendimiento con diferentes tamaños y cantidades de computadoras

Las densidades se fijaron en 20% y 50 %, es decir que las matrices de coeficientes tienen el 80% y 50% de valores iguales a cero, respectivamente.

En la figura 2 se muestran los resultados obtenidos en la experimentación de acuerdo a las condiciones antes mencionadas.

Los resultados obtenidos son casi idénticos para las dos densidades con las que se experimentó: 20% y 50%, con lo que se puede hacer el análisis de resultados sin hacer referencia a la densidad del sistema de ecuaciones. En todos los casos se puede ver que se supera el speedup óptimo esperable por el algoritmo, que está dado por p-1, asumiendo que p es la cantidad total de computadoras, esto significa que la reducción en los datos que cada computadora procesa, mejora la localidad y por lo tanto el rendimiento secuencial en cada una de ellas. Además, se debe recordar que las computadoras utilizadas son heterogéneas y que el tiempo de cómputo secuencial de referencia para el cálculo de speedup es el de la más lenta, y como consecuencia, cuando la computadora más lenta no interviene en los cálculos (es la que envía los valores de las variables a utilizar en todas las demás), el tiempo de cómputo de estos tiempos parciales puede ser menor que si se hicieran en la computadora más lenta.

5. CONCLUSIONES Y TRABAJO FUTURO

Se puede ver en los resultados de la experimentación que la implementación del algoritmo paralelo del método de Gauss-Seidel para matrices sobre una red local, presenta visibles ventajas en el cálculo de grandes sistemas de ecuaciones, proporcionando la potencia de cómputo que las actuales herramientas de asistencia al diseño de circuitos integrados necesitan.

Queda por analizar el rendimiento de esta estrategia de paralelización con mayor cantidad de computadoras, que dará más precisión en cuanto a escalabilidad, al impacto que las comunicaciones tienen frente al rendimiento general y estudiar su comportamiento en una red de computadoras heterogéneas, balanceando la carga computacional de acuerdo a las distintas capacidades individuales.

A partir de los resultados de esta línea de investigación, se pretende encarar el desarrollo de herramientas de diseño de Asic's que permitan reducir considerablemente los tiempos en el flujo total de creación de circuitos integrados.

Referencias:

[1]:Chapra-Canale, *"Métodos Numéricos para Ingenieros",* McGraw-Hill,1995.

[2]Foster, I., *"Designing and Building Parallel Programs",* Addison-Wesley, Reading, Massachusetts, 1995.

[3]:Kinkaid D.- Cheney W., "*Análisis Numérico (Las Matemáticas del Cálculo*

Científico)", Addison-Wesley Iberoamericana, 1994.

[4]: MatLab 6.0 release 12, Guide.

[5]:S.Nakamura, *"Métodos Numéricos aplicados con software"*

[6]: Koester D. **,"***Parallel Block-Diagonal-Bordered Sparse Linear Solvers for Power Systems Applications",* 1995

[7]Saad Y., "Iterative Methods for Sparse Linear Systems", 2000

[8]: Wilkinson, B., Allen, M., *"Parallel Programming, Techniques and Applications Using*

Networked Workstations and Parallel Computers",
Prentice Hall, Upper Saddle River, New Jersey, 1999.