RNPS No. 2143
ISSN 2072-6260
Versión Digital

SERIE GRIS

REPORTE TÉCNICO
Minería de Datos

Estado actual de la paralelización de algoritmos utilizando FPGAs para minería de texto

Ing. José Alejandro Somoza Chuay,
Dr. C. José Hernández Palancar

RT_014 **abril 2010**

CENATAV
Centro de Aplicaciones de Tecnologías de Avanzada
MINISTERIO DE LA INDUSTRIA BÁSICA

7ma. No. 21812 e/218 y 222,
Rpto. Siboney, Playa;
Ciudad de La Habana.
Cuba. C.P. 12200
www.cenatav.co.cu

RNPS No. 2143
ISSN 2072-6260
Versión Digital

SERIE GRIS

REPORTE TÉCNICO
Minería de Datos

Estado actual de la paralelización de algoritmos utilizando FPGAs para minería de texto

Ing. José Alejandro Somoza Chuay,
Dr. C. José Hernández Palancar

RT_014 **abril 2010**

CENATAV
Centro de Aplicaciones de Tecnologías de Avanzada
MINISTERIO DE LA INDUSTRIA BÁSICA

7ma. No. 21812 e/218 y 222,
Rpto. Siboney, Playa;
Ciudad de La Habana.
Cuba. C.P. 12200
www.cenatav.co.cu

Índice

Estado actual de la paralelización de algoritmos utilizando FPGAs para minería de texto

Ing. José Alejandro Somoza Chuay, Dr. C. José Hernández Palancar

Centro de Aplicaciones de Tecnología de Avanzada, 7a #21812 e/ 218 y 222, Siboney, Playa, Ciudad de La Habana, Cuba
jsomoza@cenatav.co.cu

RT_014 CENATAV
Fecha del camera ready: 26 de marzo de 2010

Resumen: La minería de texto se ha convertido en una rama importante de la minería de datos, la cual tiene como objetivo extraer conocimientos útiles a partir de textos no estructurados. Dado el incremento de la cantidad de información de este tipo que es almacenada en bases de datos y que transita por la web, se hace necesario que el procesamiento de la misma se realice de forma rápida y eficiente. Es por eso que la mayoría de las investigaciones en este campo tratan de buscar herramientas más eficaces para el desarrollo de aplicaciones cada vez mejores. La introducción de los FPGAs ha provisto a los investigadores de la herramienta ideal sobre la cual implementar sus diseños. El alto nivel de paralelismo presente en este tipo de dispositivo y su flexibilidad a la hora de implementar las aplicaciones son sus características más importantes. En este trabajo se realiza un estudio del estado del arte de paralelización de algoritmos utilizando FPGAs para la minería de texto y se brindan una serie de elementos teóricos importantes acerca de este tema.

Palabras clave: minería de texto, algoritmos paralelos, paralelismo, FPGAs

Abstract: Text mining has become an important branch of data mining which aims to extract useful knowledge from unstructured texts. Given the increase in the amount of information of this type that is stored in databases and passing through the web, it is necessary that the same processing is performed quickly and efficiently. That's why most research in this field trying to find more effective tools for developing better applications. The introduction of FPGAs has provided researchers an ideal tool on which to implement their designs. The high levels of parallelism in this type of device and its flexibility to deploy the applications are its most important features. This work makes a study of the state of the art of parallelizing algorithms using FPGAs for text mining and provides a number of important theoretical elements about this topic.

Keywords: Text Mining, Parallel Algorithms, Parallelism, FPGAs

1 Introducción

El creciente desarrollo que vienen teniendo la informática y las comunicaciones así como el crecimiento y evolución de Internet han provocado que hoy en el mundo se genere diariamente un enorme volumen de información de todo tipo. Se estima que alrededor del 80 % de esta información se genera en formato de texto y es almacenada y administrada en grandes bases de datos [1]. La información almacenada se convierte en una gran fuente de conocimientos que requiere del empleo de herramientas eficaces para su uso y propagación.

El desarrollo de la computación ha posibilitado el desarrollo de softwares que permiten automatizar el procesamiento de este tipo de datos, apareciendo la denominada minería de textos o text mining. La minería de textos ofrece la posibilidad de explorar grandes cantidades de textos, no organizados en forma de datos, establecer patrones y extraer conocimientos útiles.

Existen una gran cantidad de softwares que han sido diseñados para el trabajo con la minería de textos en diferentes ramas como la biología molecular, la ingeniería genética,

seguridad informática entre otras. Sin embargo, estas aplicaciones que corren sobre computadoras de propósito general (PC), en ocasiones presentan limitaciones en cuanto a la velocidad y en cuanto al volumen de información que son capaces de procesar.

Muchas veces los algoritmos utilizados por las aplicaciones que corren sobre PC tienen una complejidad operacional (cantidad de instrucciones lógicas, iteraciones y ciclos) tal, que no es posible realizar todas estas operaciones en un período de tiempo corto, lo que hace ineficiente el trabajo de estas aplicaciones. Una de las soluciones utilizadas por un grupo cada vez mayor de investigadores es la utilización de los FPGAs (Field Programmable Gate Arrays, en sus siglas en inglés). Los FPGAs se han convertido en una de las herramientas más utilizadas a la hora de implementar sistemas que trabajan la minería de texto. Su estructura permite desarrollar un alto grado de paralelismo en las tareas, lo que se traduce en realizar muchas más operaciones lógicas en un mismo lapso de tiempo, ganando en eficiencia.

Las aplicaciones realizadas con este tipo de hardware para solucionar problemas de la minería de texto han venido mostrando resultados muy interesantes desde el punto de vista de flexibilidad y la eficiencia. El número de investigadores que en el mundo se encuentran realizando estudios en este campo está en ascenso.

En este trabajo se presenta, en apretada síntesis, una panorámica del estado del arte del uso del paralelismo en los FPGAs para la minería de texto. La estructura de este documento estará dividida en las siguientes secciones: introducción, FPGAs y paralelismo, FPGAs y su aplicación a la minería de texto y las conclusiones.

2 FPGAs y paralelismo

2.1 Introducción al paralelismo

Hoy en día la gran mayoría de procesadores de propósito general (CPUs, en sus siglas en inglés), presentan relojes cuya frecuencia alcanza valores 50 veces más elevados que los que pueden alcanzar los de un FPGA. Sin embargo, los CPUs sólo pueden realizar una instrucción por ciclo de reloj, mientras que los FPGAs pueden ser configurados para funcionar como si existieran varios procesadores trabajando de manera paralela. La notable desventaja en cuanto a la diferencia de velocidad de los FPGAs frente a los CPUs, puede ser revertida utilizando el paralelismo en el diseño de sus estructuras y también en los algoritmos que emplean. En esta sección se analizarán algunos conceptos básicos muy importantes del paralelismo entre otros aspectos de interés.

Se han publicado muchas definiciones del significado de paralelismo. Sin embargo, generalmente se define como: la acción de realizar o ejecutar varias instrucciones o sentencias de programa en un mismo período de tiempo, de manera que su verdadero significado la acción de realizar más instrucciones en menos tiempo.

El objetivo principal del procesamiento paralelo es satisfacer la necesidad de lograr un mayor rendimiento, un bajo índice costo/rendimiento (el menor costo con mayor rendimiento posible), y permitir una escalabilidad adecuada del diseño.

2.2 Paralelismo temporal y espacial

El paralelismo se divide en dos grupos principales: el paralelismo temporal y el paralelismo espacial [2] [3]. El paralelismo temporal o tunelizado, hace referencia a la ejecución de una tarea como una cascada de sub-tareas. Existe aquí una unidad de procesamiento para llevar a cabo cada sub-tarea. Todas las unidades pueden trabajar al mismo tiempo de manera que se solapan las tareas. Puede definirse también como SIMD (Single Instruction stream, Multiple Data stream).

El *paralelismo espacial* consiste en la ejecución simultánea de las tareas mediante varias unidades de procesamiento. En un instante dado, esas unidades pueden estar ejecutando una misma tarea (o instrucción) o tareas diferentes. Este tipo de paralelismo puede definirse también como ***MIMD*** (Multiple Instruction stream, Multiple Data stream). Aquí la implementación física puede implicar dos variantes de organización de memoria: memoria compartida y memoria distribuida. [2]

2.3 Métodos de organización de la memoria

En cuanto a los métodos de organización de la memoria, en el paralelismo espacial es necesario definir que cuando se utiliza la arquitectura de memoria compartida, cada procesador tiene acceso directo a toda la memoria, mientras que en la memoria distribuida, cada procesador tiene acceso solamente a una memoria local y el intercambio de datos entre ellos ocurre mediante mensajes a través de una red de interconexión.

La diferencia básica entre esas dos arquitecturas se muestra en la figura 2.1, donde la memoria está representada por rectángulos y cuadrados, y los procesadores son representados por círculos pequeños.

La arquitectura de memoria compartida en más flexible; por ejemplo, es relativamente sencillo usar la memoria compartida para simular la arquitectura de memoria distribuida mediante la división de la memoria global en fragmentos, asignando cada uno de ellos a un procesador diferente.

Además, la memoria compartida facilita la programación debido a que los programadores no tienen que preocuparse por conocer cuál fragmento de memoria local del procesador contiene los datos, cómo minimizar el número de comunicaciones entre los procesadores requeridas por el algoritmo paralelo en la red de interconexión fundamental, entre otros aspectos. Sin embargo, la arquitectura de memoria compartida tiende a ser más cara y ocupa un mayor espacio en el diseño debido a que se le añaden módulos de memoria adicionales.

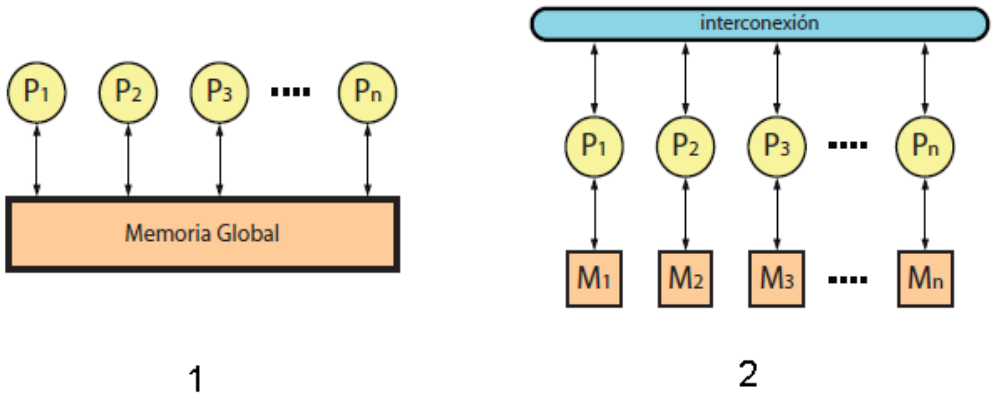

Fig. 2.1 Diferencias entre los modelos (1) memoria compartida y (2) memoria distribuida

2.4 Speed up, eficiencia y escalabilidad

Existen una serie de parámetros importantes y que deben de tenerse en cuenta a la hora de evaluar el rendimiento de un sistema paralelo. Los tres parámetros más usados son: la aceleración (Speed up, SP), la eficiencia (Ef), y el aumento de la escalabilidad (Scale up, Sc). El índice SP se calcula al dividir TI entre TP, donde TI es el tiempo que se toma el algoritmo secuencial más rápido en correr en un procesador (p) simple y TP es el tiempo tomado por el algoritmo paralelo en ejecutarse en un sistema con p procesadores en paralelo.

$$S_P = \frac{T_I}{T_P} \quad (1)$$

La escalabilidad se calcula teniendo en cuenta dos parámetros del sistema (el número de procesadores y la cantidad de memoria utilizada), y su consecuente grado de aplicación al problema o la aplicación específica.

$$S_C = \frac{T_{Large}}{T_{Small}} \quad (2)$$

En este caso, T_{large} es el tiempo que se tarda el sistema en procesar un problema de tamaño q con un sistema con procesamiento paralelo con p procesadores. T_{small} es el tiempo requerido para procesar un problema de tamaño q en un sistema pequeño, con procesamiento paralelo con p procesadores, tal que p > q. Idealmente S_C debería ser 1.

La eficiencia es un parámetro que se calcula de igual manera sin embargo, sólo se hará referencia a la aceleración (SP) y la escalabilidad por ser los parámetros más utilizados a la hora de analizar el rendimiento de sistemas con paralelismo.

Existen tres estrategias principales para la explotación del paralelismo. La primera, es usar los llamados compiladores paralelizantes, los cuales automáticamente paralelizan los códigos secuenciales de los programas. Esto es conocido como paralelización automática. La segunda, es el poder paralelizar un algoritmo secuencial existente. Desde la perspectiva de los programadores de una aplicación, esta vía es mucho más compleja que la paralelizacion automática pero más simple que la tercera de las variantes, que consiste en poder diseñar algoritmos paralelos desde cero o desde el principio [2]. Este es, potencialmente, el método más prometedor para maximizar las potencialidades del paralelismo. En [4, 5, 6 y 7] se exponen numerosos ejemplos de la aplicación de estas estrategias.

2.5 Paralelismo en Hardware

Una gran cantidad de científicos e investigadores de diferentes ramas de las ciencias utilizan y explotan técnicas de paralelismo en hardware para desarrollar aplicaciones mucho más eficientes. Explotan las ventajas que brinda el procesamiento paralelo en cuanto a velocidad, escalabilidad y eficiencia. En la mayoría de los casos se combinan las técnicas de procesamiento paralelo en hardware con las de software.

Generalmente, las arquitecturas de los sistemas con procesamiento paralelo pueden agruparse según la distribución de la memoria y por la cantidad de procesadores que agrupan. De ahí se deriva que existan varias topologías en dependencia de sus aplicaciones, por ejemplo, cuando se utilizan en procesamiento de grandes cantidades de información; en operaciones computacionales complejas las cuales manejan grandes flujos datos, entre otras. Las arquitecturas más importantes son:

- Arreglos de Procesadores, también conocidas como máquinas MPP (Massively Parallel Procesing, más de 1000 CPU).
- Multiprocesadores (SMP) o máquinas de memoria compartida
- Multicomputadoras de Memoria Distribuida. (cantidad de CPU < = 1000), también se les conoce como máquinas de memoria distribuida (Distributed Memory Machines – DMM). Esta arquitectura a su vez puede ser dividida en:
  - Shared Nothing
  - Shared Disk
- Clusters de SMPs. A partir del año 2000 aparece el término de Constellation, refiriéndose a aquellos clusters con nodos construidos con hardware propietario

y con características especiales, por ejemplo: entre 4 y 8 procesadores por nodos, tecnología de interconexión de alta velocidad.

- Máquinas de Memoria compartida distribuida. Esta arquitectura es un híbrido entre la arquitectura de memoria compartida y la de memoria distribuida.

Existen muchos ejemplos de la utilización del procesamiento paralelo mediante un diseño eficiente del hardware. Estos diseños permiten elevar la capacidad de procesamiento de muchos sistemas que demandan grandes cantidades de recursos tales como memoria, velocidad, cantidad y tamaño de los datos a procesar, entre otros. Es muy común encontrarse el paralelismo en hardware en sistemas de:

- Procesamiento de textos
- Recuperación de información en grandes bases de datos
- Aceleradores de búsquedas de textos
- Reconocimiento de secuencias de ADN
- Detección de lenguajes en textos
- Aceleradores para el reconocimiento exacto e inexacto de cadenas de texto
- Detección de intrusiones en redes

Además de los procesadores en los últimos años han ido apareciendo diferentes dispositivos (hardwares) como las GPU (Graphics Processing Unit) y los FPGA (Field Programmable Gate Arrays) capaces de realizar una gran cantidad de operaciones lógicas en un mismo período de tiempo y ganando terreno en cuanto a su utilización en diversas aplicaciones. Estos dispositivos presentan un alto nivel de paralelismo en sus tareas así como la ventaja de ser programados para tareas desde las más generales hasta las más específicas.

Las GPU son potentes procesadores que se dedican principalmente al trabajo con imágenes principalmente en PCs (ver figura 2.2). La arquitectura de estas unidades presenta un elevado paralelismo que le permite realizar un riguroso tratamiento de las imágenes sin afectar el trabajo del procesador principal. Las frecuencias de reloj de los GPU son altas, incrementándose constantemente con el transcurso del tiempo.

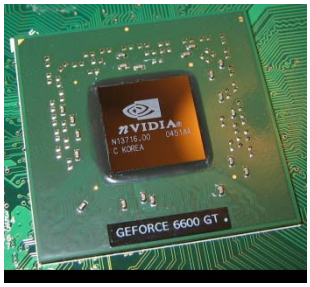

Fig. 2.2 Imagen de una GPU del tipo GEFORCE 6600 GT

Aunque en el inicio se destinaron para el desarrollo de los videos juegos o aplicaciones en 3D interactivas, hoy las GPU tienen una amplia aplicación. Gracias al paralelismo presente en ellas, se han visto aplicaciones en la minería de datos [8], en administración de bases de datos relacionales [9], en la biotecnología [10], simulación mecánica y criptografía, entre otras muchas. De los FPGAs se hará referencia en la próxima sección de este capítulo.

2.6 Características generales de los FPGAs

Los FPGAs (Field Programmable Gate Arrays) creados por Ross Freeman en 1984 pueden definirse como dispositivos lógicos de propósito general que pueden ser programados por los usuarios y que están compuestos de bloques lógicos que se comunican a través de conexiones programables. Una de sus características fundamentales es que pueden cambiar su configuración y ser reprogramados siempre que sea necesario.

Estos dispositivos surgen como consecuencia del desarrollo de los PLDs (Programmable Logic Devices). A diferencia de los PLDs, en los FPGAs la densidad de los elementos lógicos programables en una FPGA está en el orden de cientos de miles hasta millones. Los bloques lógicos son programados de manera tal que funcionen de forma coordinada en la ejecución de una o varias tareas las cuales se pueden ejecutar de forma paralela, de manera tal que su accionar se asemeja al de varios procesadores trabajando al unísono.

La estructura básica de los FPGAs consiste en un arreglo bidimensional de bloques lógicos (CLB, Configurable Logia Block) rodeados además de interconexiones programables y bloques de entrada/salida (IOBs). Los bloques lógicos o celdas lógicas contienen elementos de procesamiento para el desarrollo de una lógica combinacional simple como son los flip-flops, que se utilizan en la implementación de lógicas secuenciales. Como las unidades lógicas son frecuentemente sólo simples memorias (LUT) o un conjunto de multiplexores y compuertas, cualquier función Booleana combinacional o tal vez cinco o seis entradas pueden ser implementadas en cada bloque lógico. La estructura de interconexión general puede ser cableada arbitrariamente lo que significa que los elementos lógicos pueden ser conectados a voluntad del diseñador. En la figura que sigue se muestra la estructura interna de los FPGAs [11].

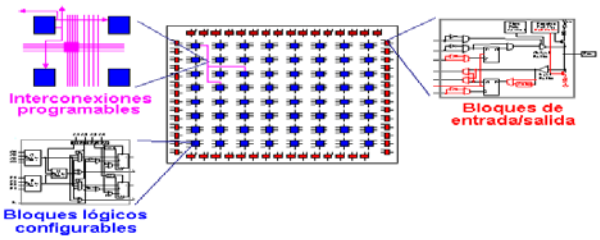

Fig. 2.3 Estructura interna de los FPGAs

Los elementos de interconexión y lógica son controlados mediante puntos programados los cuales pueden estar basados en anti-fusibles, flash o tecnología SRAM. Para lograrla son los más idóneos y de hecho, es el estilo de fabricación más usado para FPGAs en la industria electrónica de manera general. En la figura 2.4 se muestra una imagen de una tarjeta del tipo Spartan 3 que contiene un FPGA Xilinx.

Fig. 2.4 Tarjeta Spartan 3

Además de los elementos antes mencionados también pueden encontrarse sumadores, multiplicadores, registros entre otros [12].

El diseño de las FPGAs se realiza mediante la utilización de herramientas de programación o softwares descriptores de hardware. El lenguaje para el diseño de hardware es el HDL (Hardware Design Lenguaje) que abarca una amplia gama de variantes siendo las más usadas el VHDL, Verilog, HandelC, ABEL y está el LabVIEW que utiliza la variante gráfica. También existen otras herramientas con las cuales se realizan simulaciones como el MaxPlus II, el Simulink de Matlab, Active HDL y ModelSim.

La principal ventaja que tiene el uso de los FPGAs es sin dudas su capacidad para realizar operaciones o ejecutar tareas con un alto grado de paralelismo. Ellos pueden explotar los dos tipos de paralelismo fundamentales que son: el paralelismo temporal o tunelizado y el paralelismo espacial. De esta manera muchas aplicaciones se ejecutan a velocidades mayores que si se ejecutaran sobre PC. Aunque las frecuencias de reloj de los microprocesadores de las PCs en general son superiores a las de los FPGAs, el hecho de aplicar el paralelismo hace imperceptible esa desventaja. Lo anterior se explica puesto que la velocidad en que se ejecutan las instrucciones crece considerablemente. Las celdas o bloques lógicos se convierten en procesadores que trabajan en conjunto realizando una misma operación en el mismo intervalo de tiempo o tareas diferentes de manera solapada en un mismo intervalo de tiempo.

Otras ventajas del uso de este tipo hardware son:

- Capacidad de reconfiguración del diseño
- Verificación efectiva del diseño mediante simuladores o en el chip
- Ventajas de ser un producto de fabricación estándar
- Ventajas en el ciclo de vida de una aplicación
- Alta escala de integración en la aplicación (millones de compuertas)

Los FPGAs se pueden clasificar atendiendo a la tecnología de la memoria de programación en:

- *Volátiles*. Basadas en memorias RAM. Su programación se pierde al quitar la alimentación Requieren una memoria externa no volátil para configurarlas al arrancar (antes o durante el reset)
- *No volátiles*. Basadas en memorias ROM.
    - *Reprogramables*. Basadas en memorias EPROM (Erasable-Programmable ROM) o flash. Se borran y se pueden volver a programar unas 10000 veces.
    - *No reprogramables*. Basadas en fusibles. Sólo se pueden programar una vez.

Por las formas de programación se pueden clasificar en Static RAM (SRAM), Antifusibles y (E) EPROM.

3 FPGAs y su aplicación a la minería de texto

3.1 Introducción a la minería de texto

La minería de texto o minería textual (text- mining), surge como una tecnología emergente que sirve de soporte para el descubrimiento de conocimientos que poseen los datos almacenados. La minería textual está orientada a la extracción de conocimiento a partir de datos no estructurados en lenguaje natural almacenados en las bases de datos textuales, se identifica con el descubrimiento de conocimiento en los textos y se le denomina comúnmente Knowledge-Discovery in Text (KDT).

La minería de texto es la más reciente área de investigación del procesamiento de textos. Puede definirse también como el proceso de descubrimiento de patrones interesantes y nuevos conocimientos en una colección de textos, es decir, la minería de texto es el proceso encargado del descubrimiento de conocimientos que no existían explícitamente en ningún texto de la colección, pero que surgen de relacionar el contenido de varios de ellos [13].

Existe una relación muy estrecha entre la minería de datos y la minería de texto pero con una diferencia fundamental, mientras que la primera pretende obtener información a partir de los patrones y tendencias que pueden observarse en grandes volúmenes de información estructurada, la segunda busca un mismo objetivo, pero en información no estructurada.

Mediante el proceso de análisis, (separación, evaluación, validación, comparación) se le agrega valor a la información para convertirla en conocimientos. En el caso de la información textual, sólo las computadoras pueden manipular rápidamente la inmensa masa de datos y producir conocimientos valiosos que apoyen la toma de decisiones. Sin embargo, en los últimos años se ha hecho necesario aumentar la velocidad en que se procesan esos datos así como la cantidad de los mismos. Entre las etapas más significativas del proceso de minería de textos se encuentran [14]:

- Adquisición de los datos.
- Normalización de los textos.
- Filtrado: identificación de textos relevantes mediante un análisis de presencia de palabras predeterminadas.
- Análisis: establecimiento de relaciones entre textos con base en los términos y categorías.
- Visualización: uso de gráficos y diagramas.

El procesamiento de grandes volúmenes de texto libre no estructurado para extraer conocimiento requiere la aplicación de una serie de técnicas de análisis ya utilizadas en la Recuperación de Información (RI), el Procesamiento del Lenguaje Natural (PLN) y la Extracción de Información (EI), tales como la identificación y extracción de patrones, análisis de clustering, clasificación o visualización de datos.

La minería de texto se aplica en ramas tan diversas y peculiares que su campo crece ostensiblemente con el tiempo. Es común encontrarse con una gran cantidad de sistemas que utilizan métodos de la minería de texto en:

- Sistemas de recuperación de información
- Sistemas de Detección de Intrusiones en Redes (NIDS - "Network Intrusion Detection Systems")
- Biología Molecular y Bioinformática
- Web mining
- Análisis y organización de noticias
- Detectores de lenguajes
- Buscadores webs
- Aplicaciones de inteligencia militar

3.2 ¿Por qué FPGAs en minería de texto?

Como se expresó en la sección anterior, las computadoras han sido tradicionalmente las encargadas de realizar los análisis (en la minería de texto) de la información con el objetivo de convertirla en conocimientos valiosos que apoyen la toma de decisiones en diversas ramas y campos de aplicación. Sin embargo, el desarrollo tecnológico y la evolución de los medios de comunicación han propiciado un aumento en el volumen de información textual que se genera hoy en el mundo. Todo esto, unido a la disminución de los costos de fabricación de los dispositivos de almacenamiento y a la aparición de tipos de datos cada vez más complejos, ha demostrado que las PC ya no brindan todo el poder computacional que se necesita para procesar y manipular la información.

Las aplicaciones que corren sobre microprocesadores de propósito general presentan muchas veces limitaciones en una serie de aspectos que dan al traste con su correcto funcionamiento. Entre las principales limitaciones se encuentran:

- Velocidades de procesamiento bajas
- Limitación en cuanto al tamaño y la cantidad de los datos que procesan
- Embotellamiento en los procesos de entrada y salida de los datos
- Embotellamiento en la ejecución de instrucciones

Una solución eficaz para eliminar esas limitaciones se produjo con la introducción de los FPGAs. Con la introducción de este tipo de hardware muchas de las aplicaciones concebidas hasta hoy en el terreno de la minería de texto han ido mejorando sus prestaciones de manera considerable. Valdría la pena, entonces, preguntarse cuáles son las características que han

hecho de los FPGAs una solución tan eficaz. Para poder dar respuesta a la interrogante anterior hay que hacer referencia obligada a algunas de sus características. Estas son [11]:

- Permite su reconfiguración
- Explota el paralelismo tanto de software como de hardware
- Relativamente sencillos de diseñar y programar
- Versátiles, adaptables y eficientes

La propiedad de la reconfiguración en los FPGAs cobra una vital importancia. Normalmente, el diseño de estos dispositivos se realiza mediante la programación de cada una de sus estructuras directamente en el circuito integrado, pudiendo este ser transformado por el diseñador. Esto quiere decir que los FPGAs puede ser programados y reprogramados varias veces. En ocasiones la reprogramación de estos elementos se realiza sencillamente para corregir algún error en el diseño original que genera comportamientos no deseados o con el objetivo de añadirle nuevas características que posibiliten que el mismo aumente en su rendimiento. Otras veces, pueden ser reprogramados para nuevas tareas o incluso el dispositivo puede ser reconfigurado durante la ejecución de operaciones permitiendo que una simple pieza de silicio realice de manera simultánea el trabajo de numerosos procesadores de propósito general.

Es quizás el alto nivel de paralelismo implícito en los diseños de los FPGAs el elemento más tenido en cuenta por los diseñadores por la gran cantidad de beneficios que aporta. Los FPGAs pueden explotar los dos principales tipos de paralelismo que existen: el paralelismo temporal o tunelizado y el paralelismo espacial. La realización de las operaciones lógicas con un alto nivel de tunelizado permite que el sistema creado pueda realizar conjuntos de tareas de una manera solapada (que pueden o no tener relación), de modo que todas estas operaciones se realizan en un mismo instante de tiempo, logrando retornar N resultados por ciclo de reloj, luego de una latencia inicial. El paralelismo espacial se aplica cuando las operaciones lógicas son realizadas al mismo tiempo pero en unidades de procesamiento diferentes como lo pueden ser multiplicadores, sumadores y otros.

Estas posibilidades de los FPGAs, unidas al paralelismo presente en la mayoría de los algoritmos que se emplean en las aplicaciones de la minería de texto, conllevan a que estos sistemas trabajen a una gran velocidad y, por demás, que su eficiencia sea mayor que en aplicaciones similares que corren sobre procesadores de propósito general.

En términos de eficiencia, el uso de los FPGAs realiza importantes aportes. Además del factor velocidad, la eficiencia juega un papel importante. Los FPGAs consumen muy baja potencia en comparación con las PC y los ASICs. El consumo de potencia se limita a unos cuantos watts aunque está en dependencia del número de componentes presentes en el integrado. También la escalabilidad juega un papel muy importante de manera que los diseños eficientes optimizan la cantidad de elementos (compuertas lógicas) que van a estar presentes en el chip, evitando gastos innecesarios.

Con todos estos elementos no es difícil inferir que los FPGAs constituyen una de las principales plataformas para el desarrollo de aplicaciones para la minería de texto.

3.3 Principales campos de aplicación

Como se había mencionado en la sección 3.1, existe una amplia variedad de campos de aplicación de la minería de texto. Sin embargo, de manera general, es muy común encontrarse aplicaciones de este tipo de minería en sistemas de recuperación de la información, en la bioinformática, en sistemas detectores de intrusiones en redes y en sistemas clasificadores de lenguajes de textos. Esto se puede apreciar en la figura 3.1.

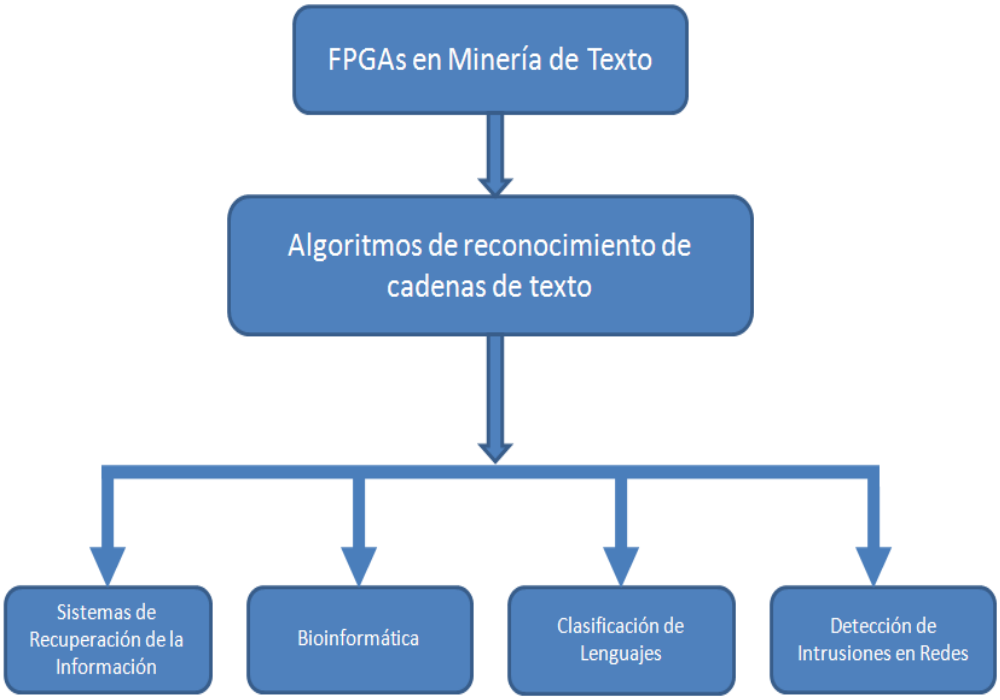

Fig. 3.1 Principales aplicaciones

Una gran cantidad de aplicaciones concebidas para problemas concernientes con la minería de texto basan su funcionamiento en algoritmos de reconocimiento de cadenas, ya sea de manera exacta como inexacta. Estos algoritmos, establecen la búsqueda o comparación de los caracteres de tres formas fundamentales: por prefijo, por sufijo y por factor.

3.3.1 Sistemas de recuperación de información

Los sistemas de recuperación de información textual son muy utilizados hoy en día a nivel mundial. Su uso se ha generalizado debido al auge y crecimiento que ha tenido Internet. Es muy común encontrarse una gran cantidad de información en bases de datos contenidas en grandes servidores (webs o no) situados en lugares cercanos o distantes. Acceder lo más rápido y eficientemente posible constituye entonces un reto debido a que muchas veces el volumen de información almacenada dificulta el proceso de localización y otras el volumen de información a transferir es tan alto que este proceso demora considerablemente además de generar los llamados cuellos de botella. El uso de los FPGAs para solucionar problemas de este tipo se ha convertido en una variante muy aceptada por muchos investigadores a escala global.

Entre las principales limitaciones que presentan las aplicaciones para minería de texto que, como se ha comentado, generalmente basan su funcionamiento en algoritmos de reconocimiento de cadenas para la recuperación o acceso a la información, y que corren sobre procesadores de propósito general, se encuentra una relativamente baja velocidad en la búsqueda de la información textual y un volumen de información a procesar (ancho de banda de los datos), relativamente bajo.

Entre las causas de esta situación se encuentra que muchos de estos algoritmos de reconocimiento son altamente secuenciales, y por lo tanto, las operaciones a realizar por el procesador tienen un bajo nivel de paralelismo. El tiempo que toma la ejecución de cada etapa de estos algoritmos es mayor que si se aplicara el paralelismo o tunelizado de cada una

de ellas. La otra causa fundamental es la falta de una plataforma sobre la cual poder implementar el paralelismo de manera más amplia o sea, no solamente explotar el procesamiento paralelo a través de algoritmos más eficientes y menos secuenciales, sino crear una infraestructura sobre la cual explotar también el paralelismo en el hardware.

La plataforma ideal para lograr lo expresado anteriormente son los FPGAs. La introducción de estos dispositivos no solamente permite el desarrollo del paralelismo en hardware o software de manera individual, sino que permite, también, desarrollar ambos tipos de paralelismo en la misma arquitectura.

Un ejemplo del uso de los FPGAs para acelerar el proceso de acceso a la información contenida en discos duros se aprecia en [15]. Aquí se crea un acelerador para el intercambio de datos entre disco duro y procesador basado en FPGAs. Este prototipo llamado Mercury está diseñado para trabajar en clusters de computadoras donde el volumen de información a transmitir desde y para los discos duros es muy elevado.

La idea básica del diseño es dividir el proceso de minería en dos fases. Un componente de bajo nivel que abarca operaciones simples, o sea, varias operaciones que deben realizarse para examinar el total de datos. Esta operación se realiza dentro de la misma unidad de almacenamiento realizando un filtraje efectivo del flujo total de los datos previo a la segunda etapa. La segunda etapa o de alto nivel, abarca el proceso de filtrado de las salidas atendiendo a la información semántica. Esto permite la posibilidad de modelar o crear bases de conocimientos y modelarlas.

Sin embargo, las operaciones de bajo nivel se han desarrollado mediante FPGA, vinculándolos directamente con la unidad del disco. Los FPGAs pueden explotar de manera eficiente el paralelismo operacional que facilita el mantenimiento del flujo de datos del disco duro y la habilidad de la reconfiguración permite el cambio hacia nuevas variantes con nuevas funcionalidades para las búsquedas de la información.

El proceso de localización de la información textual contenida en disco se realiza con la utilización de algoritmos de reconocimiento (exacto e inexacto) de cadenas de texto como el desarrollado por R. Baeza- Yates y G. Gonet, conocido por el nombre de método de Shift-Add [16]. Este método es muy eficaz para el reconocimiento exacto de cadenas para patrones relativamente pequeños. Debido a esto se adoptó la variante creada por S. Wu y U. Manber [17] el cual incluye el manejo de errores, inserciones, sustituciones y eliminaciones de caracteres. Las principales ventajas que tiene el algoritmo son su simplicidad (sólo realiza operaciones lógicas muy simples), procesamiento en tiempo real y facilidad para su ubicación en hardware.

La arquitectura de este diseño fue concebida para evitar la ocurrencia de los llamados cuellos de botella (bottleneck) que se producen en el bus de E/S cuando se produce un colapso por el volumen de datos que están siendo transferidos. En ocasiones el volumen supera el ancho de banda del bus. Además, existe también el problema de la eficiencia, la cual disminuye cuando se ejecutan a la vez varias órdenes de acceso a la información contenida en diferentes discos duros. Cada vez que esto ocurre, el microprocesador lanza un comando o instrucción el cual habilita al disco duro para tomar los datos y que los mismos sean transferidos a la memoria cache, donde son procesados por el microprocesador. El número de transferencias para la ejecución de todas estas operaciones crece considerablemente lo que provoca que la eficiencia disminuya.

Para solucionar los problemas descritos anteriormente, los autores crearon una arquitectura en hardware que consta de tres partes: un Data Shift Register (DSR), una unidad de lógica reconfigurable y un microprocesador. El DSR recibe el flujo de datos proveniente del disco; la lógica reconfigurable se encarga de las funciones de verificación de bajo nivel y el microprocesador se encarga de controlar las funciones de la lógica reconfigurable. El conjunto conforma un motor de búsquedas que permite el desarrollo del reconocimiento de bajo nivel sin afectar al microprocesador principal, devolviendo solamente los resultados para el procesamiento a alto nivel. En la figura 3.2 se muestran las características del sistema creado.

Se realizaron varias simulaciones usando como herramienta Xilinx Virtex FPGA que, por ejemplo, admite (para reconocimiento exacto) cadenas con una longitud mayor a 32 caracteres, lo que significa que se realizan 32 copias en el FCL (Fine Comparison Logic) permitiendo la entrada de 8 caracteres por ciclo de reloj. El resultado muestra el paralelismo operacional de 256 (8 x 32) comparaciones concurrentes (al mismo tiempo) utilizando un solo FPGA. El diseño soporta un flujo de datos de 500 MB/seg, mucho mayor en comparación al rango soportado por un procesador individual de un clúster.

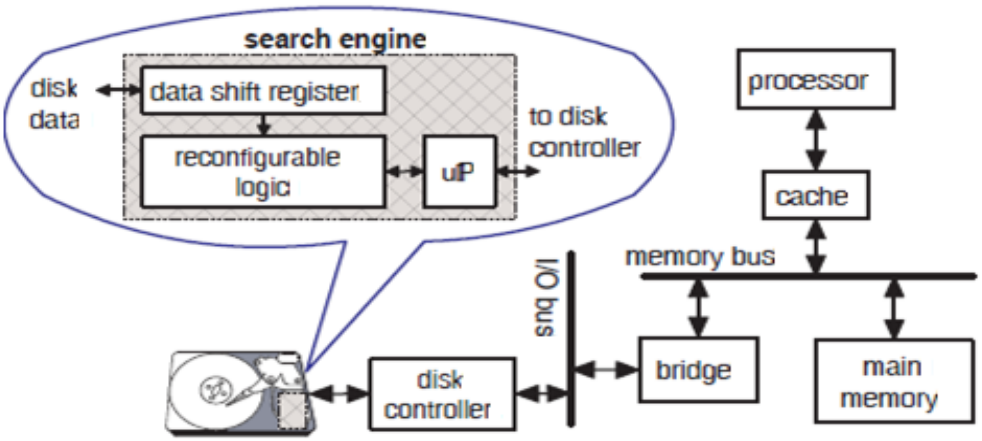

Fig. 3.2 Arquitectura creada para Mercury

Sin embargo, se utilizó como banco de pruebas una plataforma cuyo nombre es The Field Programmable Port Extender (FPX). FPX contiene a su vez dos FPGAs los cuales son: the Network Interface Device (NID) y el Reprogrammable Aplication Device (RAD). El NID introduce un sendero de comunicación entre el interior y el exterior del RAD. El RAD es el módulo en el cual se programan los algoritmos de reconocimiento a utilizar por el prototipo creado. Tampoco pudo embeberse el módulo dentro de la unidad del disco sino que se agregó al bus de I/O.

Luego de realizar las simulaciones se crearon los prototipos RAD FPGA en una plataforma de Xilinx (V1000E-FG680-7) cuya frecuencia de trabajo final es de 100 MHz y procesando un caracter por ciclo de reloj, lo que arroja un flujo total de datos de 100 MB/seg. La utilización del FPGA fue de sólo de un 10% del total, o sea, solamente 1257 compuertas lógicas de un total de 12288 presentes en el circuito (aquí se ve claramente que el diseño pudo haber incrementado su rendimiento con elementos adicionales). Esto último nos indica que se pudo haber utilizado un modelo más pequeño de FPGA.

Aplicaciones como esta muestran como la aplicación del paralelismo trae aparejado un aumento significativo de la velocidad. Por ejemplo, en este caso los resultados mostraron que el sistema creado es de 12.5 a 31.25 veces más rápido que los sistemas típicos.

Por otro lado, se han creado otras aplicaciones capaces de identificar el objetivo de búsquedas en tiempo real a través de la asociación de palabras claves. En [18] se desarrolló una tarjeta PCI llamada PRESENCE II la cual contiene un FPGA del tipo Virtex II. Además del FPGA la tarjeta cuenta con un procesador digital de señales (DSP), 4 GB de memoria SDRAM, dos unidades de 1 GB de memoria ZBT SRAM y además 8 canales bidireccionales de señales de bajo voltaje (LVDS). La estructura del diseño se muestra en la figura 3.3.

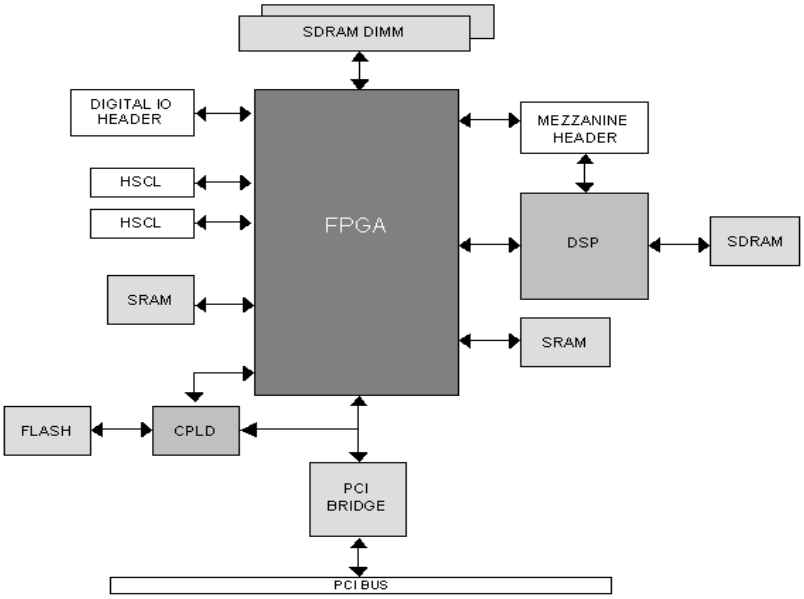

Fig. 3.3 Estructura del diseño de [15]

En este diseño se aplicó el paralelismo de manera exitosa mediante el manejo eficiente de los diseños del circuito integrado en general, y muy especialmente, en el uso y distribución de las memorias. Se hizo uso de tablas de hash para el agrupamiento o indexación de las palabras. Los resultados preliminares de esta aplicación muestran cómo el paralelismo o canalización de los procesos permitieron que la tarjeta PRESENCE II presentara velocidades superiores de 4 a 29 veces en comparación con aplicaciones similares aplicaciones a base de softwares.

Una de las potencialidades más importantes y menos utilizadas de los FPGAs es sin dudas la posibilidad de su reconfiguración. Este tema ha sido estudiado ampliamente por muchos investigadores pero en ocasiones han sido poco utilizadas en aplicaciones prácticas porque se necesita mucha destreza para implementar esos diseños.

Un ejemplo exitoso de la aplicación de la reconfiguración en aplicaciones de minería de texto se brinda en [19]. Se construyó una plataforma llamada The Exegy A2000. Aquí se incluyen dos procesadores de propósito general y un FPGA. La aplicación es capaz de realizar búsquedas de tres tipos: exactas, aproximadas (incluye sustitución de caracteres) y de expresiones regulares. De esta forma, cuando el usuario realiza la elección del tipo de búsqueda que desea, el sistema carga de forma dinámica la configuración apropiada en el FPGA y hace correr los datos a través del mismo para ejecutar las instrucciones del algoritmo de búsqueda correspondiente. La arquitectura del sistema se muestra en la figura 3.4.

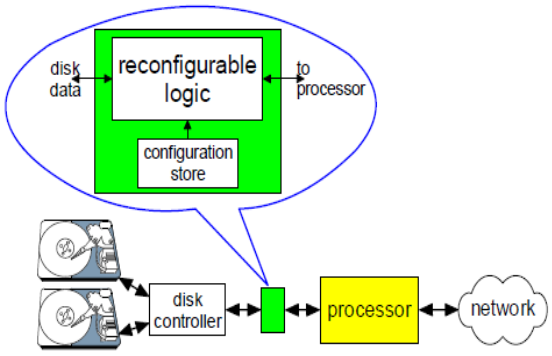

Fig. 3.4 Arquitectura del sistema Exegy

Otro trabajo de interés es el realizado por Christopher Layer en 2007, donde se desarrolla el prototipo de un coprocesador para búsquedas asociativas de textos llamado ACE (Associative Computing Engine) [20]. Este sistema se dedica a la recuperación de información digital por medio de procedimientos de aproximación. De esta manera se obtuvieron resultados relevantes en cuanto a la escalabilidad del diseño y al ancho de banda de los datos que maneja.

Por último, en [21] el autor, Zachary Baker, propone el desarrollo de algoritmos eficientes para diferentes aplicaciones junto con la aplicación de los FPGAs. Se hace referencia a algoritmos de reconocimiento de cadenas como el KMP (Knuth-Morris-Pratt) [22] y Shift and Compare. Se propone la aplicación de los mismos en problemas de recuperación de la información entre otras.

Como posibles temas a desarrollar se encuentran la paralelización de los algoritmos que posibilitan acelerar los procesos de acceso a la información contenida en grandes bases de datos y de esta forma estudiar la posibilidad de implementarlos en FPGAs. También investigar sobre la factibilidad de utilizar las llamadas memorias CAM (Content Addressable Memory) que posibilitan hacer más dinámicas estas operaciones.

3.3.2 Bioinformática

Otro de los campos donde es muy frecuente el uso de las técnicas de la minería de texto es la bioinformática dado, en gran medida, al desarrollo vertiginoso que ha tenido la ingeniería genética y la biotecnología. Hoy en día, es muy común encontrarse con un gran número de aplicaciones que utilizan técnicas y algoritmos relacionados con la minería de texto. Las aplicaciones más frecuentes son las que están relacionadas con proyectos de ingeniería genética, donde es necesario encontrar similitudes entre secuencias de ADN, ARN o proteínas.

Para hallar similaridades entre cadenas tanto de ADN como de proteínas se utilizan algoritmos de reconocimiento de cadenas de texto. El reconocimiento de cadenas de texto puede presentarse de dos maneras, de forma exacta y de forma inexacta. Sin embargo, es el reconocimiento aproximado (Approximate Matching, en inglés) el que más se utiliza en este campo de la ciencia. Existe una gran cantidad de algoritmos para el AM, pero de manera general, la mayoría tiene como base la búsqueda de la menor distancia entre la cadena patrón P y el fragmento de texto T. Esta distancia es conocida comúnmente como distancia de edición o de Levenshtein que también se puede definir como el número de operaciones (inserciones, eliminaciones o sustituciones de caracteres) que se necesita para transformar una cadena en otra.

Los algoritmos para el trabajo con secuencias de ADN necesitan necesariamente ser eficientes. La inserción de los FPGAs como herramienta para la aceleración de estos

algoritmos así como la utilización de la programación dinámica para aumentar la eficiencia de los mismos ha propiciado grandes avances en la materia.

Una aplicación interesante en este campo fue realizada por Tom Van Court y Martin C. Herbordt del Departamento de Ingeniería Eléctrica y Computación de la Universidad de Boston [23]. El artículo aborda la temática del uso de la programación dinámica (DP) para el reconocimiento aproximado de cadenas de texto y su aplicación en la biotecnología. La DP para AM comprende una larga familia de algoritmos que varían significativamente en sus aplicaciones, complejidad y utilización de hardware.

Varias implementaciones de estos algoritmos han reportado significativos aumentos de la velocidad de procesamiento de los datos, pero éstas han sido típicamente soluciones a problemas muy específicos. Sin embargo, la aparición de nuevas técnicas como BLAST y su frágil implementación en hardware provocaron su poca utilización. Algoritmos como el Needleman-Wunsch (NW) [24] y el Smith-Waterman (SW) [25] que emplean formas diferentes para la verificación de cadenas se han utilizado en el desarrollo de aceleradores a base de hardware reconfigurable.

Un hecho significativo es el uso de las llamadas matching cells o células de reconocimiento. En [23] los autores emplean un algoritmo basado en un arreglo bidimensional (2D) el cual llaman subtitution matrix o matriz de sustitución, que almacena los resultados de la comparación entre cada elemento de las cadenas.

Sin embargo, existen diferencias en estos algoritmos en cuanto a las reglas que definen los tipos de caracteres dentro de la cadena; las matching cells, las cuales forman unidades para almacenar los valores de los resultados de la comparación de las cadenas y por último el secuenciador, que controla el flujo básico de cadenas de datos y los resultados del reconocimiento a través del sistema.

Se realizaron varios prototipos de este diseño utilizando un FPGA de Xilinx del tipo XC2VP70 con los cuales se ejecutaron varias simulaciones obteniendo buenos resultados en cuanto a velocidad y eficiencia. Por ejemplo, se pudo comprobar que la velocidad del prototipo creado supera de 186 a 510 veces a aplicaciones que corren sobre computadoras fabricadas en 2004.

Otra importante aplicación fue la realizada en el año 2007 precisamente en el antes mencionado Departamento de Ingeniería Eléctrica y Computación de la Universidad de Boston [26]. En esta ocasión se desarrollan nuevas versiones para acelerar el proceso de reconocimiento de cadenas o secuencias de ADN o aminoácidos.

Según lo expresado en [26], los métodos basados en la programación dinámica o DP son ideales puesto que utilizan m processing cells o células de procesamiento (siendo m el número de caracteres de la cadena patrón), su complejidad es proporcional al flujo de datos que maneja el sistema. Sin embargo, la fragilidad de las plataformas creadas en cuanto al limitado uso por parte de los usuarios primarios ha restringido su utilización.

Por su parte las implementaciones con el método BLAST (Basic Local Alignment Search Tool) son sustancialmente más veloces y con mejor adaptabilidad a los sistemas bien establecidos. En comparación con la DP, BLAST tiene dos inconvenientes, uno de ellos es que necesita de múltiples pasos a través de la base de datos o del total de los mismos, mientras que la DP sólo necesita de uno. El segundo inconveniente es el orden en que se realizan las operaciones llamadas indels (inserciones y eliminaciones de caracteres). Otra diferencia entre estos dos métodos lo constituye el hecho de que mientras BLAST devuelve cualquier cifra de elementos correctamente alineados, DP devuelve solamente uno, o al menos, un número menor.

Los autores presentan entonces una nueva variante para el sistema FPGA BLAST. Esta nueva variante es la FPGA Tree BLAST. Tree BLAST se asimila al BLAST en cuanto a los parámetros de su programación, los cuales fueron concebidos para tener un máximo de sensibilidad sin afectar su rendimiento. Las operaciones de inserción y eliminación de caracteres son manejadas de manera independiente. Esta variante fue implementada en una tarjeta del tipo Xilinx Virtex-II Pro XC2VP70. Los resultados de esta aplicación son muy

superiores a los sistemas que corren sobre PC, sobre todo porque en aplicaciones como el reconocimiento de cadenas de ADN y proteínas tienen un rendimiento muy superior.

En esta sección se demuestra cuán importante es tener en cuenta la naturaleza de los algoritmos así como la posibilidad siempre abierta de conjugar los elementos que brinda la programación dinámica con diseños eficientes de hardware. Un elemento importante y que debe de tenerse en cuenta para próximas investigaciones lo constituye la aplicación de la DP en los algoritmos y su implementación en los FPGAs.

3.3.3 Sistemas de detección de intrusiones en redes

El aumento de la vulnerabilidad de las redes debido al incremento de la cantidad de ataques de virus y gusanos (worms) es un problema latente hoy en día. Los elementos maliciosos poseen en su mayoría una estructura formada por cadenas de caracteres. La acción de estos elementos indeseados puede conllevar al colapso de importantes sistemas de bases de datos o servidores. El desarrollo de sistemas de protección como el (NIDS) Sistema de detección de intrusos en una Red que trabaja en la búsqueda y detección de anomalías que inicien un riesgo potencial, tales como ataques de denegación de servicio, escaneadores de puertos o intentos de entrar en un ordenador, analizando el tráfico en la red en tiempo real son contribuciones importantes en esta rama. Para ello, analiza todos los paquetes, buscando en ellos patrones sospechosos. Los NIDS no sólo vigilan el tráfico entrante, sino también el saliente o el tráfico local, ya que algunos ataques podrían ser iniciados desde el propio sistema protegido. A pesar de la vigilancia, su influencia en el tráfico es casi nula.

Sin embargo, los sistemas que realizan este monitoreo y que corren sobre procesadores de propósito general muchas veces son ineficientes. Las operaciones de monitoreo del tráfico en las redes necesitan una alta complejidad computacional para poder realizar un correcto análisis de los paquetes que transitan por ella en tiempo real. Este análisis se basa en el reconocimiento de cadenas de texto para las búsquedas de elementos maliciosos. Generalmente, el flujo de paquetes a analizar es muy superior a la capacidad de escaneo de los sistemas corrientes. Es por ello que se han introducido nuevos métodos para acelerar este proceso. Los FPGAs permiten elevar la velocidad de ejecución de las operaciones de reconocimiento de cadenas así como el flujo de datos a procesar.

Un ejemplo de lo expresado puede verse en [27]. En la publicación el artículo describe el diseño de un sistema basado en FPGA que permite elevar la velocidad de escaneo de paquetes en sistemas de detección de intrusos en redes como el SNORT. Este escaneo de paquetes se realiza mediante un proceso de reconocimiento de cadenas de caracteres, donde las reglas se comparan con cada paquete que entra a la Red.

Cada regla tiene una estructura bien definida en forma de cadenas de caracteres con una longitud que puede variar en dependencia del tipo. Un ejemplo de una regla de SNORT es: alert tcp any any ->192.168.1.0/24 111(content: “idc}3a3a}”; msg: “mountd access”). Las reglas contienen campos que pueden especificar protocolos de paquetes sospechosos, direcciones IP, puertos, contenidos y otros. El campo ¨content¨ (idc|3a3a|) contiene el patrón que será macheado y puede encontrase en caracteres ASCII, hexadecimal o en formato mixto. El problema de la verificación de cadenas se hace importante cuando la velocidad en que se transmiten los datos por la red es grande, por ejemplo 10 Mbps.

Muchos de los sistemas de reconocimiento de cadenas en NIDS que se han implementado últimamente a base de FPGA usan autómatas finitos para la búsqueda de los textos. Estos diseños generalmente son bastante baratos puesto que utilizan entre 1 y 1.5 elementos lógicos por caracter de búsqueda. Sin embargo, la operación del autómata finito se ve limitada a solamente un carácter por ciclo de reloj. Para lograr aumentar el ancho de banda del escaneo, los autores de [27] han propuesto la utilización del paralelismo, mediante el cual se producen múltiples copias de los paquetes con los que trabaja el autómata, realizando el mismo número de operaciones en menos tiempo. La inconveniencia que presenta lo anteriormente expresado es que muchas veces el tamaño de los paquetes que se copian no son los mismos.

Para resolver el problema anterior los autores utilizan comparadores de banda ancha para las búsquedas. Cuando los comparadores (N) trabajan en conjunto sobre los elementos de entrada (el mismo paquete), se hace más factible el incremento del ancho de banda del sistema puesto que se incrementa la cantidad de datos a procesar por unidad de tiempo mediante la adición de más elementos de hardware trabajando en paralelo. Además, para suplir la desventaja que trae el incremento de elementos de hardware se utiliza una canalización (pipelined) extensiva de las operaciones lo cual permite aumentar las frecuencias de operación. Se direccionan directamente el abanico de paquetes hacia múltiples motores de búsquedas. Luego, el diseño va a permitir procesar N caracteres por ciclo. En la figura 3.5, se puede apreciar la configuración paralela de los comparadores.

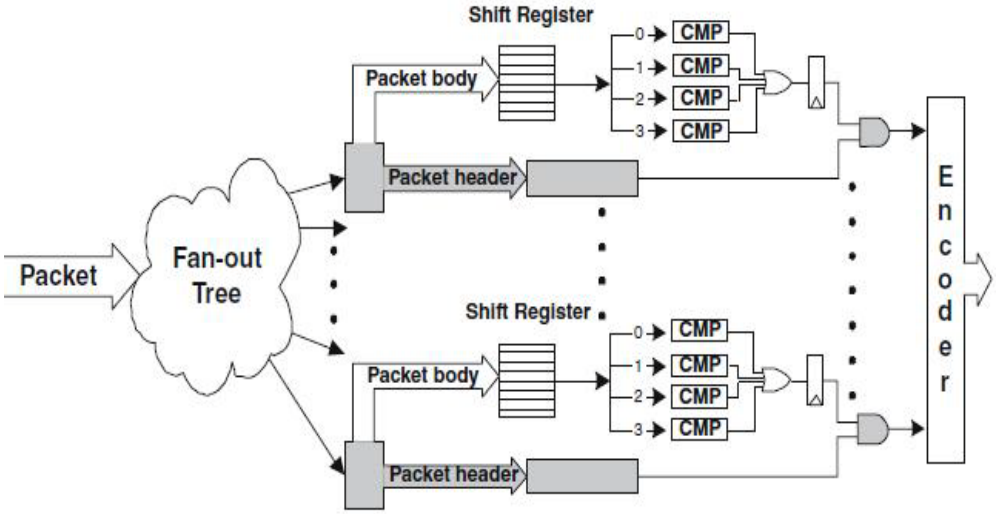

Fig. 3.5 Sistema FPGA-NIDS. Los paquetes arriban y son conectados a las unidades de reconocimiento. N comparadores en paralelo procesan N caracteres por ciclo de reloj y los resultados del reconocimiento son codificados para determinar que acción tomar

Para la simulación del proyecto se utilizaron las herramientas de Xilinx (ISE 4.2i) y el diseño fue montado sobre los modelos Virtex 1000–6, Virtex E 1000–8, Virtex2 1000–5, Virtex E 2600-8 and Virtex2 6000–5. Los resultados obtenidos luego de realizar diferentes simulaciones (estas simulaciones se realizaron experimentando con diferentes números de reglas) las cuales arrojaron que para reglas con un número bajo de caracteres por patrón los resultados son buenos (para plataformas pequeñas) puesto que el ancho de banda se mantiene en el orden de los 6 a los 12 Gbps. Por otro lado, para reglas largas se tiene que el ancho de banda oscila entre 6.5 y 8 Gbps. A continuación, en la tabla 1, se muestran los resultados de las simulaciones.

Tabla 1. Resultados de las simulaciones: frecuencia de operación, flujo de datos a procesar y porcentaje de retardo (por conexiones) con flujo de datos máximo

	Rule Set # Patterns (rules) Average Patterns Size (characters)	Synth 10 10 10	Synth 16 16 16	Web attacks 47 10.4	Web-all 210 11.7
Virtex 1000-6	MHz Wire Delay Gbps	193 56.7% 6.176	193 45.2% 6.176	171 61.9% 5.472	
Virtex 1000-8	MHz Wire Delay Gbps	272 54.6% 8.707	254 57.5% 8.144	245 49.8% 7.840	
Virtex2 1000-5	MHz Wire Delay Gbps	396 37.4% 12.672	383 54.1% 12.256	344 58.7% 11.008	
VirtexE 2600-8	MHz Wire Delay Gbps				204 70.2% 6.528
Virtex2 6000-5	MHz Wire Delay Gbps				252 69.7% 8.064

En comparación con otras aplicaciones creadas con el mismo objetivo, como la Lockwood, Gokhale, Cho, Clark [28,29, 30 y 31] y otras, ésta presenta ventajas importantes. En la tabla 2 se ofrecen algunos datos comparativos entre la aplicación presentada en [27] y otras similares.

Tabla 2. Comparación entre diferentes aplicaciones a base de FPGAs

Description	Input Bits/ c.c	Device	Freq. MHz	Throughput (Gbps)	Logic Cells	Logic Cells /char	# Patterns x # Characters
Sourdis-Pnevmatikatos Discrete Comparators	32	Virtex 1000 VirtexE 1000 Virtex2 1000 VirtexE 2600 Virtex2 6000	193 171 272 245 396 344 204 252	6.176 5.472 8.707 7.840 12.672 11.008 6.524 8.064	1.728 8.132 1.728 7.982 1.728 8.132 47.686 47.686	17.28 16.64 17.28 16.33 17.28 16.64 19.40 19.40	10x10 47x10.4 10x10 47x10.4 10x10 47x10.4 210x11.7 210x11.7
Lockwood FSM+Counter	32	VirtexE 1000	119	3.808	98	8.9	1x11
Gokhale Dis.Comparators	32	VirtexE 1000	68	2.176	9.722	15.2	32x20
Cho Dis.Comparators	32	Altera EP20K	90	2.880	17000	10.55	105x15.3
Clark NFAs SharedDecoders	8	Virtex 1000	100	0.800	19660	1.1	$1500x11.7^6$

Hay que destacar que el haber concebido una arquitectura basada en el paralelismo mediante el uso audaz de las herramientas de diseño del hardware y además la utilización racional de los elementos lógicos, permitió obtener altos rendimientos en cuanto a ancho de banda, frecuencia de trabajo y números de unidades lógicas por carácter.

Otra aplicación interesante relacionada precisamente con la utilización de los FPGAs para prevenir amenazas en redes es presentada en [32]. En este caso los autores trabajan con vistas a aumentar la velocidad de procesamiento de los datos en NIDS como el SNORT debido a que muchas veces esta no supera los 60 Mbps. Esto puede traer como consecuencia que se puedan insertar en las redes algunos fragmentos de tráfico con alguna contaminación y que las mismas no puedan ser detectadas.

Para revertir esta situación, los autores han propuesto utilizar como herramienta los FPGAs de manera que se puedan aplicar nuevos algoritmos, capaces de aumentar la velocidad de procesamiento de los datos y mejorar de esta manera la eficiencia de los sistemas.

En [32] se emplea un algoritmo llamado Shift-OR. Este provee una metodología efectiva para el reconocimiento de cadenas por su simpleza y flexibilidad. En este caso el SNORT asocia cada patrón a una memoria ROM (de sólo lectura) y a un registro de cambio para su comparación, los cuales son diseñados de acuerdo al tamaño del patrón. En la figura 3.6, se muestra la estructura del algoritmo llevada al hardware.

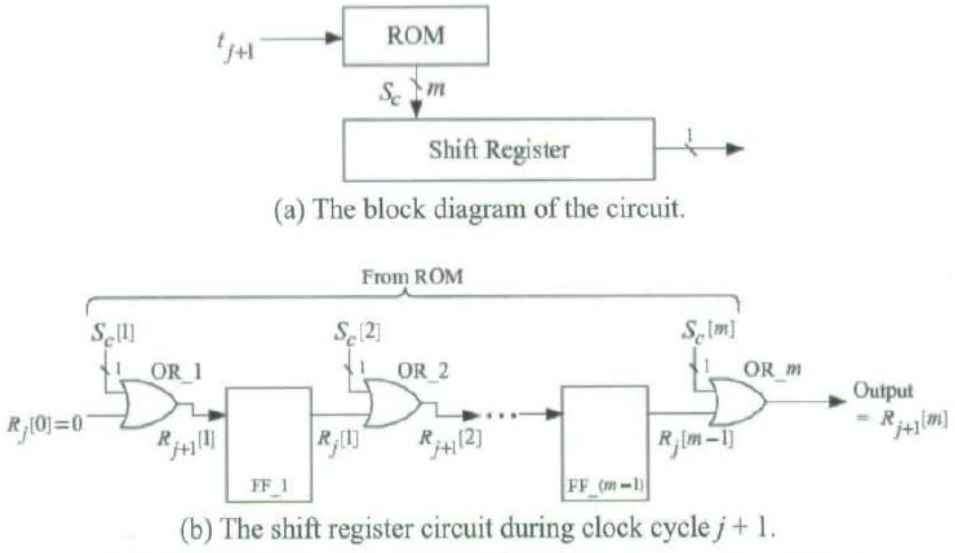

Fig. 3.6 Circuito básico de cada módulo (M) de reconocimiento de cadenas

Cada módulo M, dentro de la arquitectura diseñada, se corresponde con la cantidad de reglas de reconocimiento implementadas en el diseño. Cada módulo es responsable de la verificación de patrones mediante una sola regla. Los codificadores reciben las alarmas de intrusiones emitidas por cada módulo y transfieren la información al administrador del sistema.

La arquitectura propuesta (el diseño se realizó con el software de Altera Stratix FPGA devices) permite que este diseño pueda operar a una mayor frecuencia de reloj (a más velocidad) comparada con otros diseños similares debido a la flexibilidad del algoritmo con que trabaja. A su vez, el número de compuertas lógicas se reduce debido a que la memoria ROM está embebida en el mismo bloque que la RAM al FPGA. El procesamiento de los caracteres solamente puede realizarse uno por uno ya que de lo contrario se encarecería bastante el diseño.

El éxito de este diseño es, precisamente, el hecho de haber logrado procesar un número mayor de caracteres por ciclo de reloj agrupando q caracteres consecutivos (en este caso en grupos de 2). Después de realizar ajustes, sustituyendo las memorias ROM previstas originalmente por memorias ROM dual-port (compartidas) la estructura del diseño final queda como se muestra en la figura 3.7.

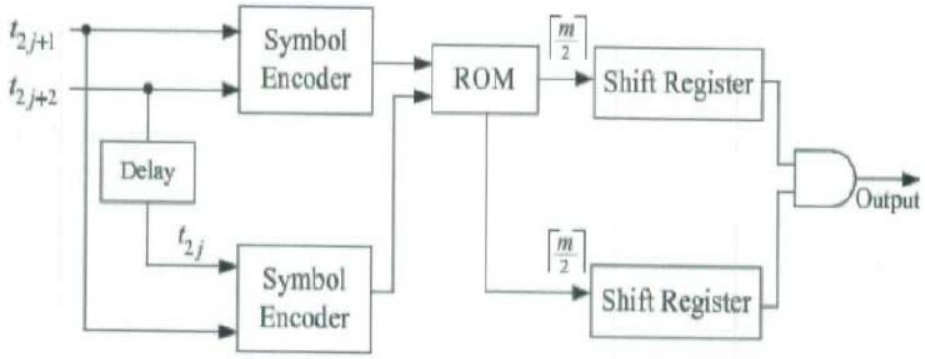

Fig. 3.7 Estructura del módulo de ampliación del ancho de banda a dos caracteres por ciclo (q=2) con memoria ROM dual compartida

No obstante, después de haber realizado un estudio pormenorizado de los rendimientos del sistema propuesto en la figura 3.6, junto a dos nuevas variantes que comprenden en su estructura la no compartición de la memoria ROM (la primera), y la variante con una no compartición de la memoria ROM, y además con un solo carácter por ciclo de reloj, se llegó a la conclusión de que la variante que no toma en cuenta la memoria compartida presenta un rendimiento más elevado que las demás. En la tabla 3, se ilustra lo expresado anteriormente.

Tabla 3. Comparación entre diferentes aplicaciones a base de FPGAs propuestas

Design	Device	Throughput (GB/s)	No. Characters	Logics Cells /char
Proposed architecture (q =1)	Altera Stratix EP1S40	2.25	5004	0.96
Proposed architecture (q =2) without ROM sharing	Altera Stratix EP1S40	5.14	1568	1.09
Proposed architecture (q =2) with ROM sharing	Altera Stratix EP1S40	4.65	1568	1.08

En los últimos tiempos se han realizado varias aplicaciones en las cuales se han introducido elementos que contribuyen a elevar la velocidad de algunos procesos. En este sentido se aprecia como la aplicación de las memorias CAM ha permitido elevar la velocidad de las búsquedas de los patrones aumentando el rendimiento de estos sistemas. Por ejemplo, en [33] se utilizaron arreglos de memorias CAM para el reconocimiento de cadenas de texto. Con este tipo de diseño los algoritmos tradicionales existentes elevan sus prestaciones considerablemente.

Puede destacarse que el incluir este tipo de memorias en los diseños, encarece un tanto el producto final debido a que se necesitan elementos adicionales para su implementación. Sin embargo, las ventajas que ofrecen son a menudo significativas. Después de realizar el diseño del sistema en un dispositivo del tipo Virtex 2VP30 los resultados muestran que la velocidad de procesamiento varia, según la cantidad de reglas, de 2.36 Gb/s a 6.41 Gb/s, la cantidad de celdas lógicas por carácter de 0.7 a 2.2.

Los sistemas vistos en esta sección resaltan la importancia que tiene, en este tipo de sistemas, combinar las técnicas y diseños de hardware con los algoritmos desarrollados en software con el objetivo de dotar a los diseños de múltiples ventajas. Las investigaciones más importantes en este sentido están encaminadas a seguir incrementando la cantidad de datos con los que pueden trabajar estos sistemas así como la velocidad de procesamiento de los mismos.

3.3.4 Sistemas para la clasificación de textos por el idioma

La detección de idiomas en documentos de textos es un elemento primordial en sistemas de clasificación de textos, filtraje de spam, sistemas de recuperación de textos, minería de texto y otros. Un gran número de aplicaciones detectoras de lenguaje corren sobre ordenadores de propósito general (PC). Sin embargo, el crecimiento del volumen de los documentos que diariamente son situados en la red de redes y bases de datos ha provocado la aparición de nuevos métodos para el procesamiento de textos y la detección automática del idioma de los mismos. Uno de estos métodos se basa en el uso de los FPGAs, que suministran una plataforma sobre la cual se logran diseños eficaces, con una arquitectura que combina el paralelismo con algoritmos específicos. Entre las técnicas usadas para la clasificación del lenguaje en documentos de textos es la N-Gram (n secuencias de caracteres en documentos de texto).

Entre los ejemplos más importantes que ilustran lo anterior se encuentra el desarrollado por [34] donde se expone la implementación en hardware (FPGA) de un sistema clasificador del lenguaje de documentos que transitan a través de la Red. Este sistema denominado Hardware-Accelerated Identification of Languajes (HAIL) es capaz de identificar de 4 a 255 idiomas diferentes presentes en textos y fue concebido teniendo en cuenta la ventaja que presentan los sistemas basados en lógica reconfigurable muchos de los cuales son capaces de procesar el trafico de las redes a velocidades mucho más grandes que las alcanzadas por los sistemas basados en microprocesadores.

El algoritmo presente en este sistema se basa en el uso de N-grams para determinar la presencia de un idioma en un texto determinado para lo cual debe ser entrenado con información de varias de estas lenguas. Los N-grams o N-Gramas son patrones secuenciales de una longitud n fija que se encuentran en documentos escritos. HAIL ha demostrado ser capaz de utilizar N-Grams de otras longitudes, sin embargo, los experimentos han demostrado que los N-Grams de longitud tres (tri-gramas) y cuatro (tetra-gramas) proveen mejores resultados. Si un n-grama aparece de una manera bastante frecuente en un documento, es muy probable que ese documento sea asociado con ese idioma.

El sistema fue implementado sobre la plataforma del Field-programable Port Extender (FPX) que aparece en la figura 3.8, ésta constituye una plataforma abierta que permite que los circuitos puedan ser rápidamente montados. La FPX es usada de forma extensiva para el procesamiento acelerado del flujo de datos provenientes de las redes.

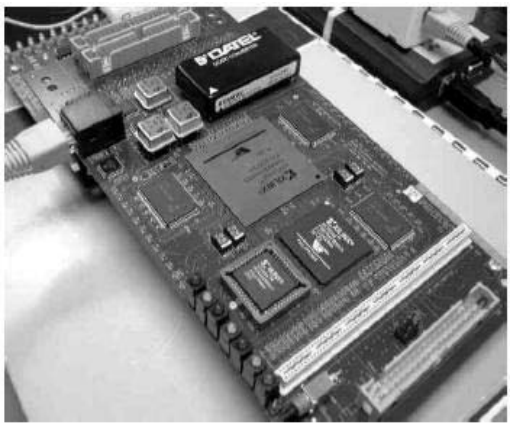

Fig. 3.8 Imagen del Field-programable Port Extender

El FPGA empleado en este caso fue el de Xilinx XCV2000E-8 sobre la plataforma antes mencionada FPX. Se pudo demostrar que los tetra-gramas pueden ser capaces de representar cada caracter del alfabeto latino con 5 bits.

Los resultados mostraron como esta aplicación presenta ventajas en comparación con sus similares que corren sobre microprocesadores de propósito general. Con solamente unos cuantos miles de palabras como entrenamiento este sistema presenta una precisión de más de un 99% y puede trabajar con un tráfico que tenga una velocidad aproximada de 2.5 Gigabits por segundos. En la tabla 4 se pueden observar otras de sus características.

Tabla 4. Recursos de hardware de HAIL

Resources	XCV2000E-8 Utilization	Utilization Percentage
Slices	5262 out of 19200	27%
4-imput LUTS	7157 out of 38400	14%
Flips Flops	5379 out of 38400	18%
Block RAMs	88 out of 160	55%

Otro ejemplo de aplicaciones similares a esta se muestra en el trabajo realizado por Arpith Jacob y Maya Gokhale en 2007, y que los mismos exponen en [35]. Ellos utilizan los Filtros de Bloom para realizar mejoras a otras aplicaciones clasificadoras de textos basadas en n-grams y basadas en los FPGAs y de esta forma lograr un diseño altamente escalable. Además el diseño se desarrolló sobre la plataforma XtremeData XD1000 que acopla excelentemente el microprocesador AMD Opteron con el FPGA Stratix II de Altera.

Los Filtros de Bloom son estructuras de datos probabilísticas utilizadas para probar pertenencias (de elementos o grupos de elementos), dentro de grandes conjuntos. Ellas usan múltiples funciones de hash y una simple memoria a base de arreglos de bits o bit-vector. Las funciones de hash son aplicadas a los elementos de entrada, generando valores de direcciones usados para referenciar el arreglo. Estos filtros son ideales para su implementación en hardware, debido a que permiten desarrollar el paralelismo: el hashing y el acceso a memoria pueden ser desarrollados independientemente. Se utilizan grandes cantidades de bloques de memoria RAM embebidas en las tarjetas donde se encuentran los FPGAs para almacenar los valores del bit-vector. Los Filtros de Bloom son implementados en los FPGAs, de manera eficiente, utilizando las ventajas de arquitectura de memoria distribuida.

Es importante destacar en esta aplicación que incrementando el número de bloques de memoria RAM embebidos en el integrado, se incrementa la precisión en la clasificación de los lenguajes pero también se reduce el número de lenguajes que pueden ser analizados simultáneamente.

En comparación con dos aplicaciones similares, el sistema creado ofrece ventajas en cuanto a velocidad y rendimiento. En la tabla 5, se puede apreciar la ventaja que le saca a aplicaciones como la Mguesser [36] y la HAIL [37], que también basan su trabajo de clasificación de lenguajes en N-Gram.

Tabla 5. Comparación entre aplicaciones que trabajan con n-gram

System	Type	Throughput (MB/sec.)
Mguesser	AMD Opteron Workstation	5.5
HAIL	Xilinx XCV 2000E-8 FPGA	324
BloomFilter	Altera EP2S180 FPGA	470

Los elementos más notables en la concepción del XtremeData XD1000, son sin dudas: la velocidad de transmisión de los datos, aproximadamente 500MB/s, y la no utilización de memorias SRAM externas, o sea, que no se le agregaron módulos externos de memorias que causan una sobredimensión del diseño en su conjunto. Además, es capaz de procesar 10 idiomas a una velocidad (teóricamente), de 1.4 GB/s. Es quizás la cantidad reducida de lenguajes (10 en total), la única desventaja que tiene con relación a sus dos similares.

En esta sección fueron abordados ejemplos que demuestran el gran número de aplicaciones que utilizan los FPGAs en la minería de texto. En cada uno de ellos se han mostrado métodos y algoritmos muy novedosos que permiten obtener un buen provecho a los sistemas basados en hardware reconfigurable y explotar al máximo el paralelismo.

3.4 Taxonomía de las aplicaciones

En esta sección se muestra una taxonomía de los métodos que aparecen en cada una de las aplicaciones presentes en este trabajo. Lo anterior se aprecia en la figura 3.9.

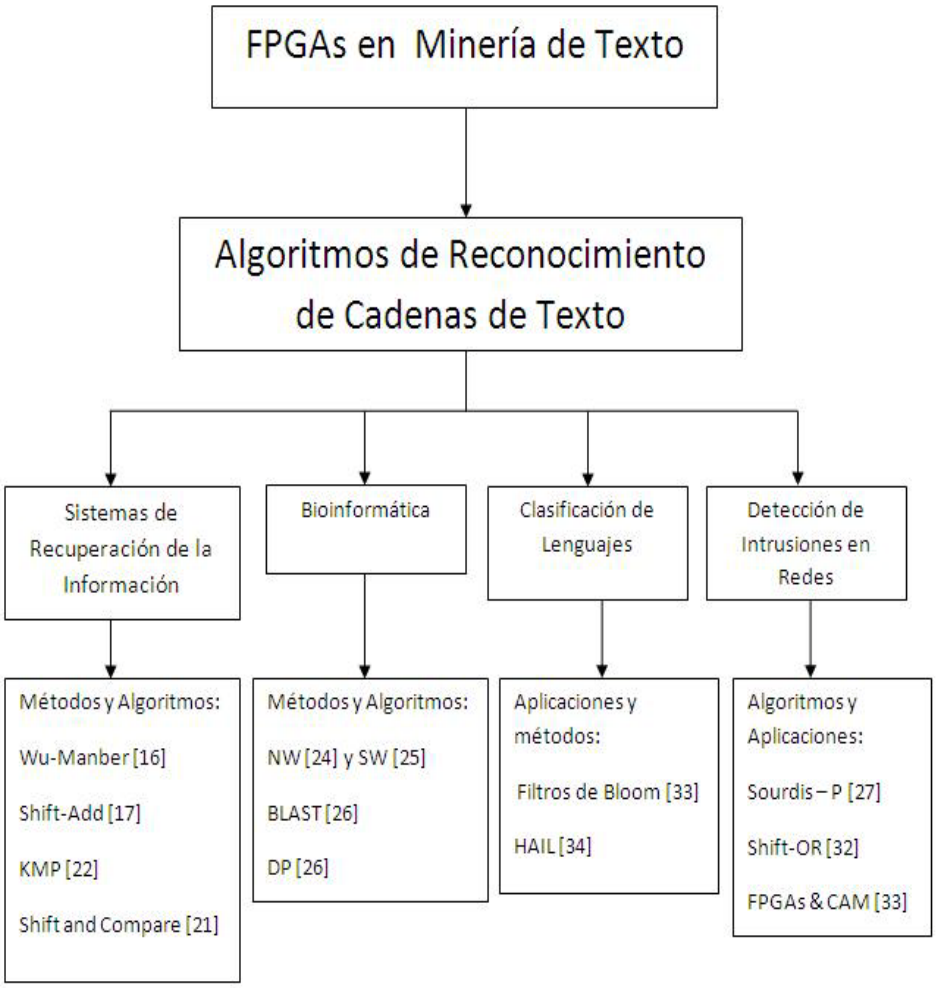

Fig. 3.9 Taxonomía de las aplicaciones métodos y algoritmos

3.5 Aspectos de interés para el futuro

Después de haber revisado los ejemplos de las diferentes aplicaciones del cómputo reconfigurable en la minería de texto se pueden realizar algunas observaciones interesantes acerca de diversos aspectos de interés. A continuación se brinda una breve panorámica de los mismos.

Entre los problemas detectados en las aplicaciones que se presentaron en este trabajo se encuentra que en la mayoría de los casos, no es bien aprovechado el total de compuertas lógicas presentes en cada dispositivo. El porciento de utilización del total del integrado no supera el 50%. Este aspecto, sin dudas, da al traste con la racionalidad de las aplicaciones

por lo que es recomendable estudiar técnicas que permitan elevar la racionalidad de los diseños.

En la mayoría de los trabajos se demuestra que a pesar que desde sus inicios la capacidad de reconfiguración de los FPGA fue teóricamente demostrada en gran medida desde la aparición de las memorias SRAM, sólo una pequeña parte de las aplicaciones utilizan esta característica que llega a constituir una gran ventaja sobre los sistemas ordinarios.

De igual forma, se han venido desarrollando con cierto éxito algunas aplicaciones en las cuales se han sustituido varios métodos y algoritmos como las tablas o mapas de hash o los llamados Arrays asociativos que agilizan los procesos de búsquedas a determinados datos de interés por métodos basados en configuraciones específicas de elementos de hardware como memorias de acceso rápido y las memorias CAM (Content-Addressable Memory). Sin embargo, la inserción de estos elementos suele traer aparejada varios inconvenientes importantes que deben ser tomados en consideración. Entre los aspectos importantes a tener en cuenta se encuentran:

- La aparición de latencias adicionales relacionadas con los nuevos elementos insertados (memorias y lógica de control adicional) a los cuales se tiene que acceder de forma frecuente cuando son ejecutadas funciones y subrutinas.
- Aumento del área que ocupan estos elementos dentro del integrado.
- Aumento del costo total del diseño.

No obstante, en la mayoría de las ocasiones las limitaciones que pueden presentar las aplicaciones debido a la latencia adicional se compensan fácilmente por el aumento en la velocidad de procesamiento de los datos. Existe un compromiso entre flujo de datos, la velocidad de procesamiento de los mismos y la frecuencia de trabajo de cada dispositivo.

Por último, en cuanto a la utilización de la programación dinámica (DP), se han implementado varias familias de algoritmos destinados al reconocimiento de cadenas de texto. Estos algoritmos varían significativamente en propósito, complejidad y utilización de hardware. Muchas de estas implementaciones han reportado aumentos en cuanto a la velocidad de ejecución de los algoritmos pero han sido generalmente, soluciones bastante puntuales a problemas específicos altamente especializadas que apuntan a solo una o a un conjunto del total de soluciones.

4 Conclusiones

En este trabajo se realizó una revisión del estado del arte de la aplicación del paralelismo en la minería de texto utilizando FPGAs. Se demostraron las múltiples ventajas que trae aparejada la utilización de los FPGAs en la elaboración de aplicaciones para la minería de texto más eficientes.

Se analizaron varios conceptos importantes acerca de la minería de texto, el paralelismo y los FPGAs, que permiten una rápida familiarización con estos temas. De esta forma, se pudo apreciar la estrecha relación que existe entre el procesamiento paralelo y los FPGAs.

Se pudo apreciar cómo la estructura interna de los FPGAs les permite realizar operaciones con un alto grado de paralelismo, propiciando que las mismas se realicen a mayor velocidad con relación a su ejecución en procesadores de propósito general, además de aumentar el flujo de datos con el cual es capaz de operar. También se pudo constatar cómo este tipo de hardware es capaz de combinar las ventajas de los algoritmos paralelos en software y en hardware.

Se revisó una gran cantidad de trabajos relacionados con la materia, donde se exponen nuevas arquitecturas realizadas con FPGAs, y que explotan el paralelismo en todas sus variantes. Destacan las aplicaciones dedicadas a los sistemas de recuperación de la información, detección de idiomas, bioinformática y seguridad en redes.

La mayoría de las aplicaciones estudiadas se basan en el reconocimiento de cadenas de texto, por lo que utilizan algoritmos de este tipo sobre hardware reconfigurable para acelerar

estos procesos. Esto trae como resultado que la mayoría de las aplicaciones creadas con FPGAs sean mucho más veloces (de decenas hasta miles de veces más veloces), eficientes y económicas que sus homólogas que corren sobre procesadores de propósito general.

Se prevé que las investigaciones futuras en este campo estén encaminadas a seguir perfeccionando los algoritmos paralelos para el desarrollo de aplicaciones cada vez mejores y, de esta manera, ampliar el espectro de aplicación de la minería de texto.

Los diseñadores han puesto su atención en el desarrollo de sistemas capaces de trabajar con grandes flujos de datos de forma rápida y eficiente, sacando provecho de las ventajas que brinda en este sentido el trabajo con FPGAs.

Actualmente en el mundo, una gran cantidad de grupos de investigación se dedican al desarrollo de sistemas para el trabajo en la biotecnología, con el objetivo de acelerar el proceso de búsqueda de secuencias de ADN en grandes bases de datos, así como al diseño de sistemas que aceleren el proceso de acceso a los datos contenidos en discos duros de gran tamaño.

El área de mayor interés, en la cual sería muy útil desarrollar una investigación más amplia, es la de Clasificación de Lenguajes. Esta temática ha sido abordada en menor medida que el resto y tiene una gran importancia para diversas áreas de trabajo y desarrollo de nuestro país.

De manera general los esfuerzos futuros en este tipo de aplicaciones están encaminados a lograr aumentar la cantidad de datos a procesar por cada ciclo de reloj así como el volumen de los mismos mediante nuevas y más eficientes técnicas y algoritmos. Cobra una importancia muy alta, el lograr que se solapen la mayor cantidad de tareas en instantes de tiempo similares sobre todo aquellas que tienen tiempos de espera relativamente altos y con un costo computacional elevado.

Referencias bibliográficas

1. Botta Ferret E, Cabrera Gato JE. ¨Minería de textos: una herramienta útil para mejorar la gestión del bibliotecario en el entorno digital¨. Acimed 2007; 16(4). Disponible en: http://bvs.sld.cu/revistas/aci/vol16_4_07/aci051007.htm
2. Alex A. Freitas and S. H. Lavington. ¨Mining Very Large Databases with Parallel Processing¨. Kluwer Academic Publishers, Norwell, MA, USA, 1997.
3. K. Hwang. ¨Advanced Computer Architectures: Parallelism, Scalability, Programmability¨. McGraw-Hill. 1993.
4. Quinn. ¨Designing Efficient Algorithms for Parallel Supercomputers¨. New York. McGraw-Hill. 1987.
5. M .J. Quinn. ¨Parallel Computing: theory and practice¨. McGraw-Hill. 1994.
6. V. Kumar, S. Shekhar and M. B. Amin. ¨A scalable parallel formulation of the back propagation algorithm for hypercube and related architecture¨. IEEE Trans. Parallel and Distributed Systems. Oct.1994.
7. S. G. Akl. ¨The Design and Analysis of Parallel Algorithms¨. Prentice-Hall.1989.
8. Christian Bohm, Robert Noll, Claudia Plant, Bianca Wackersreuther, and Andrew Zherdin. ¨Data Mining Using Graphics Processing Units¨. University of Munich, Germany. 2009.
9. Naga K. Govindaraju and Dinesh Manocha. ¨Efficient Relational Database Management using Graphics Processors¨. University of North Carolina at Chapel Hill.USA.2005.
10. Georgios A Pavlopoulos, Anna-Lynn Wegener and Reinhard Schneider. ¨A survey of visualization tools for biological network analysis¨. BioData Mining 2008.
11. Scott Hauck and André DeHon. ¨Reconfigurable Computing. The Theory and Practice of FPGA-Based Computation¨. 2008.
12. Jari Nurmi and Others. ¨Processor Design. System-on-chip computing for ASICs and FPGAs¨. Tampere University of Technology. Finland.2007.
13. Marti A. Hearst. ¨Untangling Text Data Mining¨, Proc. of ACL'99: The 37th Annual Meeting of the Association for Computational Linguistics, University of Maryland, June 20-26, 1999.
14. Sady Carina Fuentes Reyes y Marina Ruiz Lobaina. ¨Minería de Textos: Aplicación de Web Mining¨. IDICT. Cuidad de La Habana. Junio del 2007.

15. Qiong Zhang, Roger D. Chamberlain, Ronald S. Indeck, Benjamin M. West and Jason White. ¨Massively Parallel Data Mining Using Reconfigurable Hardware: Approximate String Matching¨. Center for Security Technologies. Washington University in St. Louis. USA. 2004.
16. R. Baeza-Yates and G. H. Gonnet. ¨A new approach to text searching¨. CACM, 35(10):74–82, Oct. 1992.
17. S. Wu and U. Manber. ¨Fast text searching allowing errors¨. CACM, 35(10):83–91, Oct. 1992.
18. Michael Freeman, Thimal Jayasooriya. ¨A Hardware IP-Core for Information Retrieval¨. Department of Computer Science, University of York, UK. 2006.
19. Benjamin C. Brodie, Roger D. Chamberlain, Berkley Shands, and Jason White, "Dynamic Reconfigurable Computing". in Proc. of 9th Military and Aerospace Programmable Logic Devices International Conference, September 2006.
20. Christophe Layer. ¨A Coprocessor for fast Searching in Large Databases: Associative Computing Engine¨. DISSERTATION submitted in partial fulfillment of the requirements for the degree of Dr.-Ing. to the Faculty of Engineering Science and Informatics of the University of Ulm. Germany. 2007.
21. Zachary Baker. ¨Efficient Algorithms and Architectures for High-speed Text Processing and Data Correlation¨. University of Southern California, Los Angeles.USA. 2004.
22. D.E. Knuth, J. Morris, and V.R. Pratt.¨ Fast Pattern Matching in Strings¨. In SIAM. Journal on Computing, 1977.
23. Tom Van Court & Martin C. Herbordt. ¨Families of FPGA-Based Accelerators for Approximate String Matching¨. Department of Electrical and Computer Engineering. Boston University.USA. 2005.
24. Needleman, S. B. and C. D. Wunsch. "A General Method Applicable to the Search for Similarities in the Amino Acids Sequences of Two Proteins". Journal of Molecular Biology 48:443-453. 1970.
25. Gusfield, Dan. ¨Algorithms in Strings, Trees, and Sequences¨. Cambridge University Press. Cambridge UK. 1997.
26. Martin C. Herbordt, Josh Model, Bharat Sukhwani, Yongfeng Gu, Tom VanCourt. ¨Single pass streaming BLAST on FPGAs¨. Department of Electrical and Computer Engineering, Boston University, Boston.USA. 2007.
27. Ioannis Sourdis and Dionisios Pnevmatikatos. ¨Fast, Large-scale String Match for a 10 Gbps FPGA-based NIDS ¨. Microprocessor and Hardware Laboratory, Electronic and Computer Engineering Department, Technical University of Crete and Institute of Computer Science (ICS),Foundation for Research and Technology-Hellas (FORTH), Vasilika Vouton, Heraklion. Greece. 2005.
28. Lockwood, J.W. ¨An open platform for development of network processing modules in Reprogrammable hardware¨. DesignCon '01. Santa Clara, CA, USA. 2001.
29. Gokhale, M., Dubois, D., Dubois, A., Boorman, M., Poole, S. Hogsett, V. Granidt: ¨Towards gigabit rate network intrusion detection technology¨. In: Proceedings of 12th International Conference on Field Programmable Logic and Applications. France. 2002.
30. Young, H. Cho, S.N. Mangione-Smith,W.: ¨Specialized hardware for deep network packet filtering¨. In: Proceedings of 12th International Conference on Field Programmable Logic and Applications. France. 2002.
31. Clark, C.R., Schimmel, D.E. ¨Efficient Reconfigurable Logic Circuit for Matching Complex Network Intrusion Detection Patterns¨. In: Proceedings of 13th International Conference on Field Programmable Logic and Applications (Short Paper). Lisbon, Portugal. 2003.
32. Huang-Chun Roan, Wen-Jyi Hwang, Wei-Jhih Huang and Chia-Tien Dan Lo. ¨Network Intrusion Detection Based on Shift-OR Circuit¨. Department of Computer Science and Information Engineering. National Taiwan Normal University. 2006.
33. Janardhan Singaraju, John A. Chandy. ¨FPGA based string matching for network processing applications¨. Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT, USA. 2007. Available online at www.sciencedirect.com
34. Charles M. Kastner, G. Adam Covington, Andrew A. Levine, John W. Lockwood. ¨HAIL: A Hardware-Accelerated Algorithm for Languaje Identification¨. 15th Annual Conference on Field Programmable Logic and Applications (FPL). August, 2005.
35. Arpith Jacob y Maya Gokhale. ¨Language Classification using N-grams accelerated by FPGA-based Bloom Filters¨. Publicado en HPRCTA en 2007.
36. Mguesser. http://www.mnogosearch.org/guesser/.

37. C. M. Kastner, G. A. Covington, A. A. Levine, and J. W. Lockwood. ¨HAIL: A Hardware-accelerated algorithm for language identification¨. 15th Annual Conference on Field Programmable Logic and Applications (FPL), Tampere, Finland, Aug. 2005.

RT_014, abril 2010
Aprobado por el Consejo Científico CENATAV

Editor: Lic. Lucía González Bayona
Diseño de Portada: DCG Matilde Galindo Sánchez
RNPS No. 2143
ISSN 2072-6260
Indicaciones para los Autores:
Seguir la plantilla que aparece en www.cenatav.co.cu
C E N A T A V
7ma. No. 21812 e/218 y 222, Rpto. Siboney, Playa;
Ciudad de La Habana. Cuba. C.P. 12200
Impreso en Cuba