Análisis y Extensión de una Herramienta de Simulación Multi-DSP para procesamiento paralelo

Lic. Fernando G. Tinetti[1]
Lic. Hugo D. Ramón[2]
Lic. Claudia C. Russo[3]
Ing. Armando De Giusti[4]

Laboratorio de Investigación y Desarrollo en Informática[5]
Departamento de Informática - Facultad de Ciencias Exactas
Universidad Nacional de La Plata

1. Resumen

El área de investigación y desarrollo en programación paralela es una de las que despierta mayor interés en la Informática actual. Es de particular importancia la especificación, implementación y verificación de algoritmos aplicados al procesamiento de imágenes, así como la evaluación de arquitecturas y soporte de comunicaciones para los mismos [DEG93].

En particular se ha trabajado sobre tres modelos de arquitecturas: una red heterogénea de procesadores vinculados por un soporte de comunicaciones y scheduling ad-hoc sobre la que se desarrolla software en lenguajes convencionales [TIN93] [HER93]; una red de procesadores RISC (transputers) programable en OCCAM y C Paralelo [CSA90], y la modelización de las arquitecturas CISC tipo DSP en un ambiente de simulación [MOT90]. Estos últimos modelos permiten la evaluación de soluciones desde el punto de vista del hardware, del software de administración de recursos y de la implementación concreta de algoritmos ejecutables. Los procesadores dedicados tipo DSP (Digital Signal Processor) se constituyen una alternativa en sí misma (aún sin tener en cuenta el procesamiento con más de un DSP), para el procesamiento de grandes volúmenes de datos.

---

[1] Becario de Estudio CIC. Centro de técnicas Analógico-Digitales (CeTAD), UNLP.

[2] Ayudante Diplomado Dedicación Exclusiva. Dpto. de Informática, Fac. de Ciencias Exactas, UNLP.

[3] JTP Diplomado Dedicación Exclusiva. Dpto. de Informática, Fac. de Ciencias Exactas, UNLP.

[4] Director del LIDI. Investigador Principal CONICET. Prof. Titular dedicación Exclusiva del Dpto. de Informática, Fac Cs. Exactas, UNLP.

[5] LIDI. Laboratorio de Investigación y Desarrollo en Informática, Dpto. de Informática, Fac. de Cs. Exactas, UNLP.
Calle 50 y 115 - 1er piso - 1900 La Plata - Buenos Aires - Argentina.
Tel / Fax: 54 - 21 - 22 7707
E-Mail: lidi @ ada.info.unlp.edu.ar

2. Introducción

Los procesadores DSP fueron diseñados especialmente para el procesamiento digital de señales [HWA94], donde se deben respetar las restricciones de cálculo en tiempo real que las aplicaciones imponen. Las características sobresalientes de estos procesadores son [RAM94]:

- Arquitectura Harvard, con tres bloques de memoria independientes: memoria de programa P, y dos memorias dedicadas a datos: X e Y.
- Operaciones adaptadas al procesamiento de señales implementadas en hardware, evitando el microcódigo.
- Modos de direccionamiento adaptados.
- Características de los microcontroladores, tales como: ROM programables, funciones de I/O integradas, timers integrados, etc.

La arquitectura Harvard de base, con memorias y buses independientes permite, por ejemplo, hacer acceso simultáneo a distintos bancos de memoria mientras se realizan operaciones de ALU. Dentro de las instrucciones adaptadas, se pueden mencionar MAC (Multiply and Accumulate), MACR (Multiply, Accumulate and Round), instrucciones de ajuste de precisión, y Hardware Loops. En los DSPs se han incluido también diferentes modos de direccionamiento para facilitar las operaciones específicas de procesamiento digital de señales tales como el direccionamiento "mariposa" o bit reverse para el cálculo de FFTs, y direccionamiento circular para el manejo de buffers de datos [RAM94]. Teniendo en cuenta que para el procesamiento de señales se utilizan conversores Analógico-Digitales (A/D) y Digitales-Analógicos (D/A), se han incorporado a los DSPs funciones de hardware similares a las de los microcontroladores.

Los fundamentos de la necesidad de desarrollar una herramienta de simulación se basan principalmente en dos conceptos: flexibilidad y bajo costo. Con respecto a la flexibilidad, se puede destacar que la prueba de distintas alternativas no implica cambios en el hardware y además permite tener la visión global del sistema en cualquier momento (por ejemplo se puede calcular exactamente la cantidad de ciclos de reloj que se utilizan para una tarea dada).

Con respecto al costo, se pueden simular distintas arquitecturas para la evaluación de performance sin la necesidad de disponer del hardware y sin riesgo de dañarlo (en áreas tales como el control de dispositivos, una falla en el software o en el tiempo de respuesta implica la posibilidad de destruir el dispositivo que se está controlando).

Además de disponer del paralelismo que proveen las arquitecturas tipo Harvard [RAM94] [RUS95] se ha decidido conectar varios DSP para tener paralelismo a nivel de procesos. Inicialmente se cuenta con la posibilidad de ejecutar un proceso en cada DSP a diferencia de los mecanismos provistos por los lenguajes concurrentes puros (OCCAM) y/o los extendidos (Parallel C) donde se pueden asignar varios procesos por procesador.

3. Modelos de Comunicación entre Procesos

En las arquitecturas tipo red (network architectures) que se utilizan cada vez con mayor frecuencia [AND91], los procesadores comparten un canal de comunicaciones para cuya utilización se necesita implementar una interface que permita a los procesos leer y escribir en el canal, estas interfaces se denominan message-passing primitives.

El modelo de comunicación por pasaje de mensajes, se basa en la utilización de canales. Un canal es una abstracción de la comunicación física (unidireccionales o bidireccionles) y provee un camino para la comunicación entre procesos. Se puede acceder a los canales a través de las primitivas **send** y **receive**. Para iniciar la comunicación, un proceso envía un **send** y es de esperar que otro proceso adquiera el mensaje desde el canal a través de un **receive**. El canal es típicamente el único objeto compartido por los procesos, esto indica que no se debe proveer ningún mecanismo de exclusión mutua.

La comunicación a través de canales se puede realizar en forma asincrónica (non-blocking) o sincrónica (blocking). En la forma non-blocking, la primitiva **send** no causa que el proceso se detenga, que permite que los procesos se ejecuten en forma independiente y el mensaje puede ser recibido arbitrariamente después de ser enviado. En el método blocking, cada canal provee un link directo con los procesos involucrados y si un proceso realiza un **send** queda bloqueado hasta que otro proceso ejecuta un **receive** para recibir el mensaje. De esta forma, no solamente se realiza el pasaje de mensajes, sino que también se sincronizan los procesos. La ausencia de variables compartidas permite ejecutar diferentes procesos en procesadores independientes, por esta razón a los sistemas que usan pasaje de mensaje se los denomina programas distribuidos (distributed programs).

El modelo de concurrencia utilizando memoria compartida provee una solución a los problemas de comunicación de los algoritmos paralelos [COD92]. La propuesta multi-DSP se ajusta al modelo distribuido el cual posee las siguientes ventajasrespecto del uso de memoria compartida:

- Cuando el número de procesadores crece, la memoria compartida tiende a ser cuello de botella, debido a la cantidad de accesos a memoria y consecuentemente aumenta la probabilidad de conflictos.
- Crece la complejidad de la estructura de intercomunicación física entre los procesadores, incrementando el costo de la conexión.
- La presencia de problemas de sincronización tanto a nivel de datos como de procesos hace que las técnicas de programación con memoria compartida no sean verificables formalmente y por esta razón se generan mayor cantidad de errores.

4. Comunicación en los DSP

En general, los DSP poseen periféricos on-chip diseñados para minimizar el hardware asociado a la comunicación con el exterior [TIN95]. Para los periféricos se han dedicado posiciones específicas de memoria e instrucciones especiales para el acceso a tales posiciones de memoria (MOVEP). Cada periférico posee sus

registros de control, estado y datos, además se minimiza el overhead de atención pudiendo asociar a cada fuente de interrupción tiene su propia rutina de atención.

En particular el DPS96002 posee dos ports de expansión con el exterior (Port A y Port B), cada uno de ellos posee un registro de control BCRx (donde x puede ser A o B) en donde podemos especificar entre otras cosas la cantidad de ciclos de espera para acceso a memoria externa.

El DSP96002 provee una interface paralela de 32 bits para comunicarse con un procesador que actúa como Master (Host) o Slave en cada uno de sus ports. Se puede poner como host un controlador de DMA, teniendo el DSP los registros necesarios para el control de DMA. Cada host interface (HI) tiene su propio registro de control, estado y datos que se direccionan como memoria de I/O del DSP96002, también posee bits de control para habilitar/deshabilitar interrupciones.

La HI provee soporte de funciones host para la operación de un DSP96002 en un ambiente multiprocesador, el dispositivo externo que requiere estas operaciones se llama procesador host y puede ser otro DSP96002 o cualquier procesador de 32 bits. Las funciones que el HI provee son:

- Transferencia de datos desde un procesador host hacia o desde el DSP96002 sin utilizar memoria compartida.
- Posibilidad de interrumpir al DSP96002 utilizando múltiples vectores de interrupción.
- Transferencia de bloque de datos utilizando DMA.

La HI se conecta al exterior a través del port de expansión que consta de:

- Bus de datos de 32 bits bidireccional.
- 5 líneas de control.
- Líneas de dirección A2-A5.

Posee además registros de transmisión y recepción de datos double-buffered para permitir al slave y al host transferir datos a alta velocidad. La HI posee dos modelos de programación, una para programar al DSP96002 y otro para el host [RAM95].

5. Simulación de un Ambiente Multi-DSP

El simulador es una herramienta de desarrollo de programas para el DSP96002. Esta herramienta provee toda la funcionalidad del chip, incluyendo las operaciones sobre periféricos, las modificaciones de registros y direcciones de memoria, además de la actividad de excepciones, mientras se ejecuta el código de un programa. Se simula exactamente la actividad de pipeline del bus, lo cual permite al usuario del simulador tener la medida exacta de tiempo utilizado por la aplicación, crítico en algunos casos. El código ejecutable generado para el simulador es escrito en C o assembler.

Las funciones que se pueden realizar en el simulador son, entre otras:

- Debugging, utilizando breakpoints condicionales o incondicionales.

- Se puede modificar el programa cambiando directamente las líneas del assembler.
- Visualización de la instrucción a ser ejecutado como así también el contador del tiempo.
- Archivos ASCII para Input/Output de periféricos.
- Definición de macros.
- Posibilidad de revisar y modificar los obtenidos de los registros y las memorias.
- Grabar y cargar archivos desde o hacia los diferentes bancos de memorias.

Se permite la simulación de varios DPSs pero no se pueden realizar comunicaciones entre ellos, no hay manera de "conectar" desde un port a otro, además las funciones de transferencia a través de los ports tienen dificultades en cuanto a la actualización de los diferentes bits de control.

Debido a las estas dificultades, se debió modificar el código fuente del simulador para realizar el control del acceso a los ports cada ciclo de reloj, verificando si se realizo algún acceso, actualizando desde el simulador los bits de los registros correspondientes para la comunicación. Por ejemplo, el bit de control que indica que el registro de transferencia se encuentra lleno, bit de habilitación de DMA, bits de control de interrupciones, funciones del host como leer memoria X/Y/P del slave, etc.

6. Extensiones del Lenguaje para comunicaciones

Teniendo la posibilidad de simular un ambiente multi-DSP con las comunicaciones resueltas a nivel físico, se debería tener la capacidad de transmitir información entre los distintos procesadores simulados. Una alternativa consiste en utilizar el modelo de programación de las puertas de comunicación del DSP desde el lenguaje mismo del ensamblador, teniendo acceso a todos los modos de comunicación utilizables (palabra por palabra o por DMA, con o sin interrupciones, etc.). Esta alternativa no sólo es factible (teniendo el simulador multi-DSP), sino también tiene a su favor la flexibilidad en los métodos y formas de comunicación. Asimismo se debe notar que tiene en su contra el hecho de estar restringido a trabajar con el lenguaje del ensamblador, con todas sus características negativas en términos de desarrollo y puesta a punto del software. Desde el punto de vista del desarrollo de software no solamente se tiende a evitar la utilización del lenguaje del ensamblador por sus características negativas sino que se debe tener en cuenta también el peso de desarrollar aplicaciones concurrentes, que traen consigo sus propias complicaciones relacionadas a la sincronización, comunicación y corrección de cada proceso que se ejecuta sobre un DSP.

La alternativa más atractiva desde el punto de vista académico consiste en el desarrollo de un lenguaje concurrente completo tal como fue desarrollado OCCAM para el entorno de multiprocesamiento con transputers. En este caso, el lenguaje no solamente se encarga de la comunicación y sincronización de los procesos ejecutándose en distintos procesadores, sino también de tareas tales como creación y eliminación de procesos. La característica negativa de esta alternativa es, justamente, el desarrollo de un compilador completo para que luego se puedan desarrollar y probar aplicaciones concurrentes sobre multi-DSP.

Quizás la alternativa más simple a la utilización del lenguaje del ensamblador y sin la necesidad de desarrollar un lenguaje completo sea aumentar o extender un lenguaje disponible con un conjunto de funciones básicas (biblioteca) que se encarguen de realizar las comunicaciones [HEE91]. Estas funciones se pueden desarrollar y validar lo suficiente como para ser utilizadas desde las aplicaciones sin la necesidad de controlar que funcionen correctamente cada vez que se utilizan. Esta idea ya ha sido tomada en el entorno de multiprocesamiento con transputers, y en general se ha elegido el lenguaje de programación C [KER88], dada su tendencia a la utilización de bibliotecas dedicadas y su amplia difusión [3L91], [COM90]. Además se dispone de un compilador C para el DSP96002 de Motorola que es directamente utilizable.

Como ejemplo de funciones de biblioteca de comunicación, se tomaron las funciones del C paralelo 3LC [3L91] para transputers tales como

void chan_out_message(int **len**, char ***b**, CHAN ***chan**);
void chan_in_message(int **len**, char ***b**, CHAN ***chan**);

que se utilizan para enviar/recibir **len** bytes a partir de la posición indicada por **b** a través del canal apuntado por **chan**. Del mismo tipo son las funciones que se han definido en LSC, también para transputers [COM90]

void ChanOut(Channel *c, char *ptr, int n);
void ChanIn(Channel *c, char *ptr, int n);

que se utilizan para enviar/recibir **n** bytes a partir de la posición indicada por **ptr** a través del canal apuntado por **c**.

Para resolver las comunicaciones en el entorno multi-DSP, las funciones que se definieron son las siguientes:

void SendMsg(PortType Port, void *Buffer, unsigned Size);
void ReceiveMsg(PortType Port, void *Buffer, unsigned Size);

que se utilizarán para enviar/recibir **Size** palabras (32 bits) a partir de la posición de memoria apuntada por **Buffer** por el port indicado por **Port**. Se deben destacar al menos dos características positivas de estas funciones que aportan claridad al código separándolo de las características propias de los DSP:

- El programador de aplicaciones no debe saber en qué memoria (X o Y) del DSP se encuentran los datos a transmitir. Esto se debe a que el compilador de C utilizado tiene una distribución conocida de todos los datos y el programador se desentiende de detalles tales como la distribución de los datos en memoria.
- El programador de aplicaciones no debe saber si el DSP sobre el que ejecutará su proceso es el Master o Slave del port a utilizado. Eventualmente se pueden agregar funciones para cambiar el "estado" (Master o Slave) del procesador con respecto al port.

7. Trabajo Futuro

Existen dos objetivos concretos para completar el desarrollo y verificar la utilidad de la extensión multi-DSP:

- Definir una aplicación y paralelizarla realizando mediciones (speed-up) con diferentes alternativas de implementación (por ejemplo utilizando redes heterogéneas, transputers, DSP, multi-DSP, etc.). De esta manera se pueden obtener valores de comparación reales en base a aplicaciones concretas.
- Probar la extensión sobre otros DSP utilizando los mecanismos de comunicación que ellos proveen para comparar alternativas de hardware de base.

Como mejoras a lo ya realizado se podrían agregar funciones que controlen el tiempo de establecimiento de la comunicación (control de time out), utilización de DMA en la implementación de las funciones de comunicación para transferencia de bloques de datos, y funciones de cambio de Master a Slave y de Slave a Master para cambiar dinámicamente la topología de comunicación de los distintos DSP.

8. Conclusiones

Con el desarrollo e implementación de esta librería se pueden implementar aplicaciones concurrentes sin tener el hardware de base necesario. Sólo es necesario el simulador extendido, la librería de comunicaciones y herramientas tales como el compilador C, assembler y el linker.

Además se pueden realizar estudios en relación a los dos niveles de concurrencia que se proveen. El primer nivel se refiere al acceso simultáneo a la memoria junto con las operaciones aritméticas (debido a la arqitectura Harvard) y el otro nivel en referencia a la concurrencia relacionada a la existencia de un proceso en cada DSP.

Con esta herramienta es posible implementar diferentes algoritmos paralelos ejecutándolos sobre diferentes tipos de arquitectura (Transputers, LAN heterogéneas, etc.) para comparar resultados.

9. Agradecimientos

Al Dr. Gregory Randall (Universidad de la República - Uruguay) por su asesoramiento técnico.

10. Bibliografía

[3L91] 3L Ltd., "Parallel C version 2.2.2. Release Note", Julio 1991.

[AND91] Gregory Andrews, "Concurrent Programming, Principles and Practice", The Benjamin/Cummings Publishing Company, Inc., 1991

[BUR93] Alan Burns, Geoff Davies, "Concurrent Programming", Addison-Wesley Publishing Company, 1993.

[CHA88] K. Mani Chandy, Joyadev Misra, "Parallel Program Design", Addison-Wesley Publishing Company, 1988.

[COD92] Brunp Codenotti, Mauro Leoncini, "Introduction to Parallel Processing", Addison-Wesley Publishing Company, 1992.

[COM90] Computer System Architects, "Logical Systems C for the Transputer: Version 89.1 User Manual", 1990.

[CSA90] "Transputer Architecture", Computer System Architects, 1990.

[DEG93] A. De Giusti, "Procesamiento Concurrente y Paralelo. Aplicaciones a tratamiento de Imágenes" . Proyecto LIDI-UNLP.

[GEH89] Narain Gehani, Andrew D. McGettrick, "Concurent Programming", Addison-Wesley Publishing Company, 1989.

[HEE91] Dieter W. Heermann, Anthony N. Burkitt, "Parallel Algorithms in Computational Science", Springer-Verlag Berlin Heidelberg, 1991.

[HER93] J. Hertzcowitz, A. Foster, "Soporte de comunicaciones y servicios para arquitectura distribuida", Trabajo de Grado de Licenciatura en Informática, UNLP 1993.

[HUL94] M. E. C.Hull, D. Crookes, P. J. Sweeney, "Parallel Processing. The Transputer and its Applications", Addison-Wesley Publishing Company, 1994.

[HWA84] K. Hwang, F. Briggs, "Computer Architecture and Parallel Processing", McGraw-Hill, 1984.

[KER88] Brian W. Kernighan and Dennis M. Ritchie, "The C Programming Language", Second Edition, Prentice-Hall, 1988.

[MOT89] Motorola Inc., "DSP96002 User's Manual, IEEE FLoating-Point, Dual-Port Processor", 1989.

[MOT90] Motorola Inc., "DSP96002 Advance Information. 96-bit General Purpose IEEE FLoating-Point, Dual-Port Processor", 1990.

[MOTA90] Motorola, "Intertools. 56001", Intermetrics, 1990.

[RAM94] Hugo D. Ramón, Claudia C. Russo, "Descripción de una Aplicación en DSP", Reporte Técnico LIDI, 1994.

[RUS95] Claudia C. Russo, Fernando G. Tinetti, Hugo D. Ramon, Marcelo Naiouf, "Procesamiento Paralelo: Experiencias con Transputers y DSP", First International Congress of Information Engeneering, 1994.

[TIN93] Fernando G. Tinetti, Armando De Giusti, "Una aplicación de procesamiento paralelo sobre LAN", Proc. Cuarto encuentro académico tecnológico, 1993.

[TIN95] Fernando Tinetti, Hugo Ramón, Claudia Russo, "Mecanismos de Comunicación del DSP96000", Reporte Técnico LIDI, 1995.