Adaptación del núcleo IP de un procesador tipo MC6805 para operar en un ambiente multiprocesador y multitarea

Guillermo A. JAQUENOD
Facultad de Ingeniería, UNCPBA.
Olavarría, Buenos Aires, ARGENTINA.
chipi@netverk.com.ar

Horacio A. VILLAGARCÍA
CICPBA - Facultad de Informática, UNLP.
La Plata, Buenos Aires, ARGENTINA.
hvw@info.unlp.edu.ar

Oscar N. BRIA
CONICET – Facultad de Informática, UNLP.
La Plata, Buenos Aires, ARGENTINA.
onb@info.unlp.edu.ar

Marisa R. DE GIUSTI
CICPBA - Facultad de Informática, UNLP.
La Plata, Buenos Aires, ARGENTINA.
marisadg@volta.ing.unlp.edu.ar

Palabras clave (Keywords): procesamiento distribuido, procesamiento paralelo, lógica programable, multiprocesadores empotrados, system-on-a-chip, núcleos IP.

RESUMEN

La disponibilidad de dispositivos de Lógica Programable de alta densidad de integración permite buscar soluciones integradas en un dispositivo SOPC (System On a Programmable Chip).

Un tema de creciente interés son los procesadores empotrados, siendo usual un único procesador y un sistema operativo con capacidad de multitarea.

Sin embargo, debe considerarse como alternativa insertar varios procesadores, no necesariamente idénticos, que pueden a su vez atender varias tareas. En un SOPC, como diferencia fundamental con los casos tradicionales de multiprocesamiento y multitarea, las tareas a realizar son conocidas antes de comenzar el diseño, por lo tanto hardware como software se pueden configurar a medida de la aplicación, combinando la velocidad propia del primero, con la versatilidad del segundo.

Este artículo describe las modificaciones de hardware realizadas al núcleo IP (Intellectual Property) de un procesador, de modo de permitir la inclusión de un administrador de tareas por hardware y de canales de comunicación interprocesadores.

1. INTRODUCCIÓN

Los criterios de diseño empleados para definir la arquitectura de un sistema de procesamiento [8][10][13], dependen de modo fundamental del grado de conocimiento que se tenga de los problemas a resolver:

- si las tareas que deberán ser resueltas son desconocidas, y pueden ser de tipo muy variado, la solución más razonable es el empleo de un procesador de propósito general; tal es el caso, por ejemplo, de un computador personal.
- si el sistema a desarrollar se orientará a un tipo de tareas específico pero aún indefinidas (por caso, tratamiento de imágenes), dicha elección suele orientarse a un procesador especializado (tal como un DSP) con amplios recursos adicionales (memoria, I/O, periféricos).
- si la aplicación es totalmente conocida, el diseño puede hacerse "a medida", usando los recursos de hardware más apropiados para esa aplicación, y según el caso, es posible pensar en el diseño de un ASIC (Application Specific Integrated Circuit).

La necesidad de integrar sistemas cada vez más complejos en dimensiones más reducidas, y con tiempos de desarrollo cada vez menores (TTM: Time To Market) junto con el vertiginoso crecimiento de las densidades de integración, ha motivado al mercado a la búsqueda de soluciones que integren en un único chip todo un sistema dando origen a los denominados SOC (System On a Chip). Como consecuencia de ello, las nuevas metodologías de diseño basadas en SOC se fundamentan en la existencia de librerías con bloques pre-diseñados y pre-verificados denominados "bloques IP (Intellectual Property)". La reusabilidad de los bloques IP es uno de los tópicos que facilita la creación de un nuevo SOC [6][7][9][12]. Dentro del marco de la lógica programable, la existencia de dispositivos programables cada vez más poderosos ha llevado a las empresas líderes a proponer soluciones SOC en dispositivos programables SOPC (System On a Programmable Chip), asi como se proponen bloques IP de determinadas funcionalidades [4].

Para poder encarar soluciones programables por software, estas empresas han comenzado a ofrecer productos donde dentro de un chip conviven un único procesador, un sistema operativo de tiempo real (RTOS) con capacidad de multitarea, y un conjunto de recursos programables aptos para distintas aplicaciones:

- ATMEL ofrece un procesador RISC de 8 bits (AVR), con abundante RAM y ROM de programa y un bloque programable de entre 10K y 40K compuertas.
- TRISCEND ofrece un ARM7DMI de 32 bits, con memorias cache internas, interfase a memorias externas, periféricos (timers, UARTs, interrupts) y una conexión a una matriz programable de complejidad equivalente a 40K compuertas.
- ALTERA ofrece una alternativa Softcore llamada NIOS [14], configurable a 16 o 32 bits de ancho de datos, que puede ser sintetizado en chips de las familias APEX, FLEX o ACEX. Como alternativa Hardcore, dentro de la familia Excalibur, se ofrecen tambien 3 modelos de ARM922T y 3 modelos de MIPS32 4Kc, que difieren entre sí por la capacidad de memoria y de lógica programable interna [15].
- XILINX ha anunciado una alternativa Softcore llamada MicroBLAZE, de 32 bits, que puede ser sintetizado en la familia VirtexII, con UART, timer, entrada/salida paralela, controlador de interrupciones, árbitro multimaster, e interfases a memoria FLASH, y distintos tipos de RAM.

Todas estas soluciones se basan en un único procesador muy poderoso (salvo el AVR, todos los otros casos son de 32 bits), periféricos propios, y recursos de interconexión con la lógica programable; internamente los procesadores poseen buses de alta performance (100MHz o más) usando formatos estándar tipo AMBA (ARM), CoreConnect (IBM), Wishbone (SILICORE), o

soluciones propias [11]. Para el desarrollo de sistemas todas las empresas ofrecen paquetes de diseño que combinan las herramientas EDA propias del diseño lógico con compiladores y depuradores del software, y poderosos sistemas operativos multitarea de tiempo real (RTOS).

Como alternativa, una línea de investigación en expansión analiza la inclusión, dentro de un chip, de varios procesadores no necesariamente idénticos que operan como "agentes de procesamiento" [1][2], y donde cada procesador puede a su vez atender varias tareas; se obtendría un MPOC (Multi-Processors on a Chip). En el diseño de un MPOC, se presenta una diferencia fundamental con los casos tradicionales de multiprocesamiento y multitarea, pues las tareas a realizar son conocidas "a priori", es decir antes de comenzar el diseño, y por lo tanto hardware como software se pueden configurar a medida de la aplicación, combinando la velocidad propia del primero con la versatilidad del segundo; en cambio, en un sistema de multiprocesamiento convencional este conocimiento es "a posteriori".

En este artículo se describen las modificaciones de hardware realizadas al núcleo IP de un procesador MC6805 de 8 bits [3], de modo de permitir la inclusión de un administrador de tareas por hardware y de canales de comunicación interprocesadores.

2. LA PROPUESTA MPOC

La propuesta MPOC (***MultiProcessors On a Chip***) [5] está orientada a aplicaciones dedicadas donde el factor costo es crítico. La idea básica de esta propuesta es que el uso de múltiples procesadores simples y no necesariamente idénticos puede aprovechar de modo más eficiente sus diferentes habilidades, y disminuir las latencias y el *overhead* que impone un RTOS en un sistema monoprocesador. En esta propuesta se sugiere una metodología estructurada para construir aplicaciones multitarea, donde existen dos niveles de partición, a nivel de procesadores y de métodos de comunicación entre tareas:

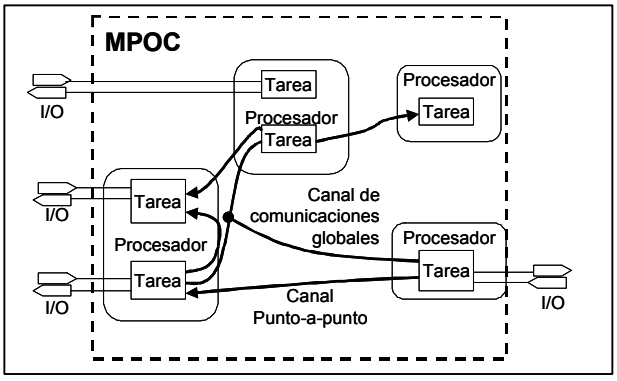
- A nivel de tareas, aquellas que atienden procesos de tipo similar pueden convivir en un mismo procesador, en tanto aquellas que atienden procesos de diferente tipo pueden estar en procesadores separados.
- A nivel de comunicaciones, dos tareas que intercambian entre sí una cantidad importante de información pueden usar como medio de comunicación áreas de memoria compartida, FIFOS, o similares, en tanto aquellas que están sólo ligeramente acopladas pueden usar canales más simples (por ejemplo, canales seriales tipo TLINK [17]).

Para que un procesador pueda operar en un contexto MPOC debe poseer ciertas características:

- Si va a atender una cantidad predefinida de tareas conocidas, debe poseer facilidades para administrar esas tareas con poco sobrecosto de software.
- Si va a interactuar con los otros procesadores debe poder comunicarse con ellos de un modo eficiente, que consuma pocos recursos de hardware (silicio y cableado interno).

En base a estos requerimientos un MPOC puede ser visto como una estructura jerárquica de procesadores, tareas, canales y puertas de entrada/salida.

La figura siguiente muestra el esquema de un hipotético MPOC, tal como es descripto en [5]; en este sistema varios procesadores atienden varias tareas (algunos sólo una, otros más de una), y se comunican entre sí mediante canales punto a punto, o empleando algún recurso de *broadcast*.



Asimismo, ciertas tareas pueden actuar sobre el mundo externo a través de líneas propias de entrada/salida, en tanto ciertas tareas pueden ser sólo de procesamiento interno.

Por ejemplo, considérese el caso de la necesidad de diseñar el computador de control de un automóvil.

En este caso existen tareas claramente diferenciadas:

- A nivel del motor: control de inyección, de encendido, de temperatura del motor, monitoreo de variables de operación (presión de aceite, gases de escape).
- A nivel estructural: suspensión activa, control de frenos (ABS) y de tracción, airbags.
- A nivel general: control del acondicionamiento de aire, computadora de navegación (estimación de consumo, promedio de velocidad, etc.), elementos de confort (audio, radio, CD, etc.), alarma antirobo-antiasalto, levantavidrios, control centralizado de cierre, control de luces, desempañador, limpiaparabrisas, etc.

Un análisis superficial muestra que:

- las tareas a nivel de motor están fuertemente interrelacionadas entre sí, y que su conexión con las de nivel general es casi nula; asimismo, estas tareas requieren cómputo aritmético intenso, la evaluación de modelos de operación del motor, y pueden ser resueltas con más facilidad por procesadores para el tratamiento de señales (tipo DSP).
- de igual modo los sistema de control de suspensión, tracción, frenos y airbags forman un bloque funcional compacto que comparte información de sensores propios (sensores de giro de las ruedas, acelerómetros) y que comanda actuadores propios (frenos y airbags), y en la bibliografía se muestra la conveniencia de su atención mediante sistemas basados en reglas (tipo *fuzzy*).
- finalmente, las tareas de tipo general requieren fundamentalmente un intenso manejo de entrada/salida a nivel de bits, mucho poder de decisión, recursos para múltiples temporizaciones, y canales de comunicación a periféricos (por ejemplo hacia el sistema de audio), lo que sugiere un procesador de propósito general.

3. MODIFICACIONES A UN NÚCLEO IP DEL MC6805 PARA ATENDER MULTITAREAS

El microprocesador MC6805 es ampliamente usado en aplicaciones de bajo costo, y sus características detalladas pueden encontrarse en los manuales técnicos [18]; sin embargo, a fines de coherencia de este artículo, se detallan los aspectos más relevantes.

Es un procesador de punto fijo, con bus de datos de 8 bits, y arquitectura Von Neumann. La CPU cuenta con pocos registros internos: un acumulador (A) y un registro índice (X) de 8 bits, un puntero de pila (SP) de sólo 5 bits variables, un registro de códigos de condición de 5 bits, y un contador de programa (PC) de ancho parametrizable entre 8 y 16 bits.

El bus de direcciones de 8 a 16 bits referencia un espacio común para variables, instrucciones y entrada/salida, y en este espacio existe un tratamiento distinto a la página cero (bit 8 y superiores del bus igual a cero) que al resto, pues para operar en página 0 existen dos modos de direccionamiento dedicados, llamados directo (DIR) e indexado sin offset (IX).

En el 6805 coexisten variados modos de direccionamiento como: Inherente, Inmediato, Directo y Extendido, Indexado, Relativo al SP, Relativo al PC y Vectorizado.

Además de estos modos básicos, existe la posibilidad de referirse a un bit en particular dentro de un dado byte, conformando un modo mixto de direccionamiento.

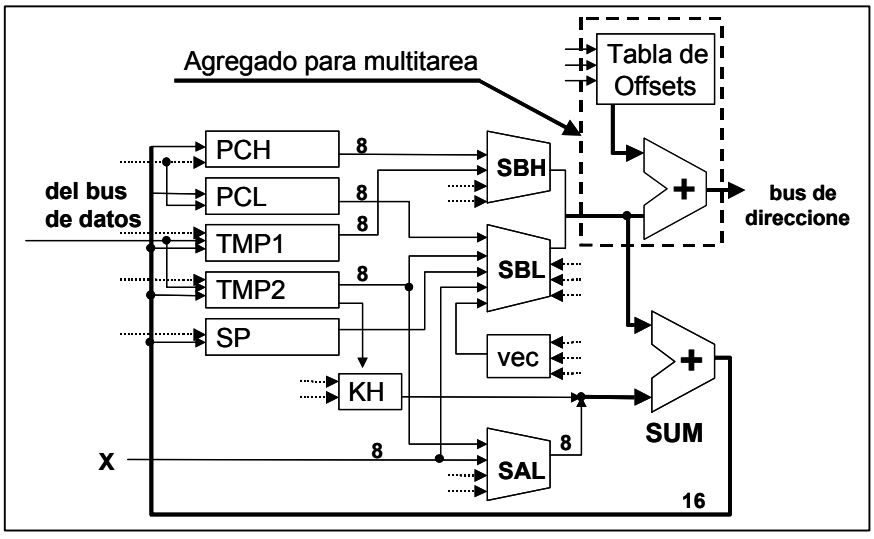
En [3] se muestra detalladamente cómo es posible diseñar un procesador MC6805 monotarea usando lógica programable FLEX10K de ALTERA, de modo que resulte eficiente en velocidad y haga uso mínimo de recursos (entre 500 y 550 elementos lógicos).

Para poder atender varias tareas es necesario poder realizar fácilmente el cambio de contexto, es decir disponer de un medio para que la tarea saliente almacene todos aquellos valores necesarios para poder continuar en una instancia posterior.

Esto implica la salvaguarda de dos recursos:

- De las áreas propias (privadas) de memoria de datos.
- De los registros del procesador.

La salvaguarda de las áreas propias (privadas) de memoria puede aprovechar la circunstancia que el tamaño de código y de datos variables usados por cada tarea es conocido “a priori”. Por ello puede usarse un espacio lineal de memoria y asignar a cada tarea una dada región, cuyo origen está marcado por un desplazamiento (offset) constante.



La figura muestra las modificaciones a realizar a unidad de cálculo de direcciones de un MC6805 monotarea [3], y evidencia que sólo es necesario agregar un sumador y una tabla de constantes que genere esos offsets.

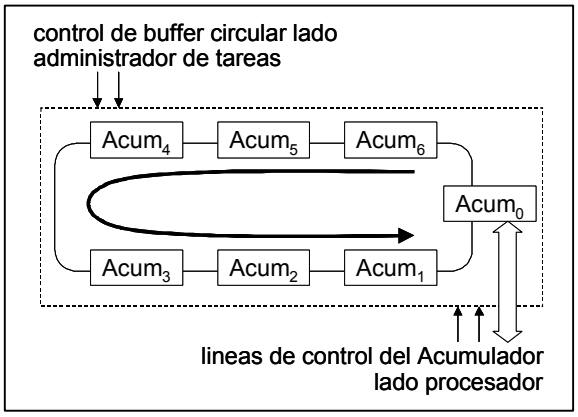
.

Como refinamiento adicional, podría detectarse el acceso a RAM o ROM, y en base a él generar offsets distintos, de modo de operar sobre espacios de memoria diferentes y optimizar su uso; esta separación es necesaria en el caso de memorias RAM/FLASH externas.

Este esquema multi-offset también puede ser de utilidad si se predefinen ciertas áreas del mapa de memoria para su uso compartido.

Para el caso peor de una solución multitarea de hasta 16 tareas de longitud de código variada, la generación de las tablas de offset requiere tantos elementos lógicos como ancho tenga el bus de direcciones final, mas otro tanto para realizar el sumador. Por ejemplo, dadas 8 tareas, cada una de ellas con un bus de direcciones de menos de 12 bits, la generación del bus final de 15 bits requiere agregar 30 elementos lógicos.

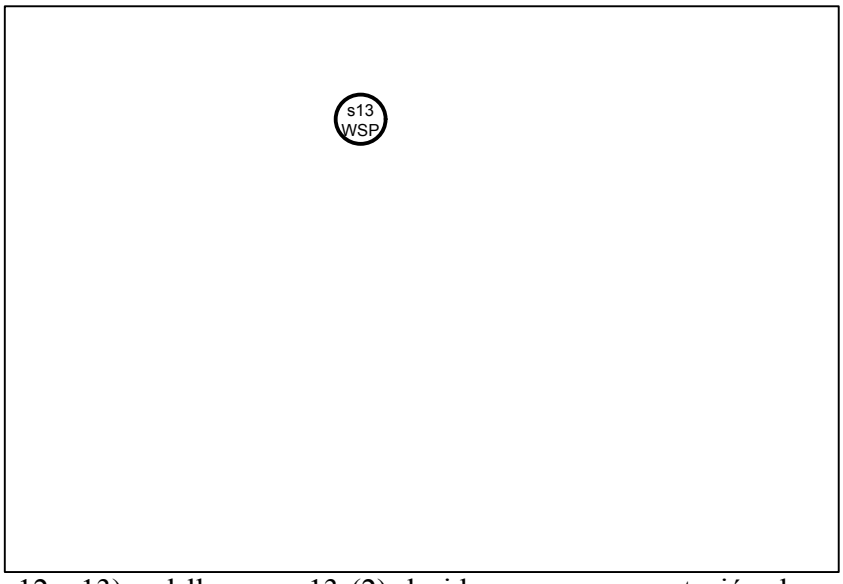
La salvaguarda de registros puede resolverse de forma paralela o secuencial.



En el caso paralelo, cada registro propio del procesador mono-tarea es reemplazado por un banco de registros de igual tamaño organizado en forma de buffer circular, uno por cada tarea a atender. En este caso, cuando una tarea se desactiva se hace rotar en buffer hasta que en la puerta activa del buffer queda el registro propio de la nueva tarea que es activada.

La figura muestra el caso hipotético de un procesador que atiende 7 tareas, donde se evidencia cómo la puerta activa del registro (lado procesador) es independiente del control de selección de cuál es el registro activo, tarea que realiza el administrador de tareas (*scheduler*). Si se considera la necesidad de salvaguardar a los registros A, X, CCR, SP y PC, y suponiendo un mapa de ROM de 12 bits, cada tarea adicional implica el agregado de 8+8+5+5+12=38 elementos lógicos.

En el caso secuencial, puede aprovecharse que el MC6805 realiza automáticamente la salvaguarda de registros en la pila al atender una interrupción, que los recarga desde la pila en un retorno de interrupción, y donde el único registro que no es salvado es el puntero de pila. Para esta solución puede usarse un buffer circular solo para salvaguardar el puntero de pila, y realizar el cambio de contexto mediante una secuencia de interrupción, conmutación de puntero de pila, y retorno de interrupción.



La figura muestra las modificaciones a realizar a la máquina de estados de control de un MC6805 monotarea [3], y evidencia que sólo es necesario agregar un nuevo estado (s30-SCHED) a los 30 estados pre-existentes (s0 a s29) para poder realizar la conmutación de contexto.

En este caso, cuando el scheduler decide conmutar una tarea, genera una interrupción (1) que fuerza el apilado de los registros (secuencia s9, s10, s11, s12, s13), al llegar a s13 (2) decide pasar a conmutación de registros (s30) en lugar de buscar un vector como sería en una interrupción normal (s14); ya en s30 salva el puntero de pila de la tarea

saliente en un buffer circular y deja activo el puntero de pila de la tarea entrante, cambia los offsets para apuntar a los mapas de memoria de esta nueva tarea, y en el ciclo siguiente realiza un retorno de interrupción (4) ya en el nuevo contexto (secuencia s16, s17, s18, s19, s20, s21, s8). Esta ampliación requiere mínimo agregado de hardware, consistente en 5 elementos lógicos por cada nueva tarea (para almacenar el puntero de pila) y de 5 elementos lógicos más para el agregado del estado s30 en sí mismo.

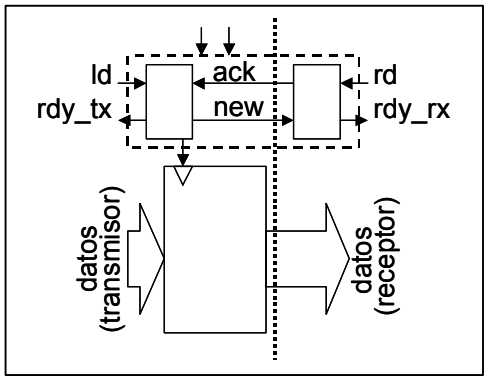
4. LOS CANALES DE COMUNICACIÓN

Desde el punto de vista del MPOC, un canal de comunicación es un objeto virtual de hardware, que describe enlaces entre procesadores. Desde el punto de vista de cada procesador, cada conexión a un canal está asociada a un periférico, ya sea un transceptor serial, una puerta paralela, o un elemento más complejo, tal como una área de RAM compartida o un buffer FIFO.

Una transacción a través de un canal implica que una tarea transmita un dato y otra tarea lo reciba, y este proceso puede ser utilizado como mecanismo para sincronizar ambas tareas entre sí; la transacción puede ser iniciada por el transmisor (que escribe un dato) y cerrada cuando el receptor lo lee, o iniciada por el receptor (que queda pendiente a la espera de datos) y cerrada cuando el transmisor envía ese dato.

Este proceso puede demorar tiempo, y es entonces razonable definir medios para que ese tiempo pueda ser aprovechado para atender otras tareas, y por ello en [5] se proponen modelos de puertas de canal donde de ambos lados del canal aparecen señales de sincronización (rdy).

Por ejemplo, una puerta paralela constituye el esquema más simple de hardware para intercomunicar tareas, donde el transmisor usa registros para almacenar los datos y una pequeña máquina de estados para sincronización, y el receptor sólo otra máquina de estados.



- Del lado transmisor, cuando se escriben nuevos datos (***ld*** activa) la línea ***rdy_tx*** se desactiva, indicando que se pasa a un período de espera, y la presencia de datos en el canal se avisa mediante ***new*** activo; cuando el receptor lee esos datos (mediante ***rd***) se activa ***ack***, con lo que ***rdy_tx*** se activa y ***new*** se desactiva.
- Si, en cambio, el receptor trata de leer datos (***rd*** activo) cuando no los hay (***new*** inactivo) la señal ***rdy_rx*** se desactiva y queda en ese estado hasta que el transmisor escriba un nuevo dato.

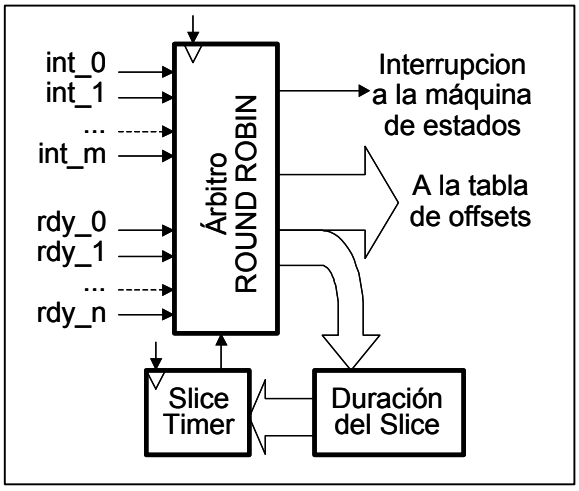
Estas señales (***rdy_rx*** y ***rdy_tx***) pueden ser usadas tanto dentro de una tarea, esperando la culminación de la transacción por encuesta, o usada por el administrador para conmutar de tarea.

5. ADMINISTRADOR DE MULTITAREAS

No existe un esquema predeterminado para la administración de tareas, por cuanto depende de si las tareas tienen o no igual prioridad o tiempo asignado, si el administrador gerencia las interrupciones externas, si existe una tarea especializada como *front-end* de interrupciones, u otras

opciones.

El caso más simple es el de un sistema multitarea, donde cada tarea tiene igual prioridad, caso en el que se emplea un esquema “round-robin”.



En este caso, el inicio de un proceso de conmutación de tareas puede tener distintas causas:

- que la tarea activa quede en espera (rdy inactivo), caso en que se pasa a la siguiente tarea disponible, siguiendo un orden de prioridad circular.
- que se haya agotado el tiempo disponible para una tarea (“slice time”) y haya otra tarea pendiente, con idéntico resultado.
- excepcionalmente, que deba ser atendida una interrupción externa, caso en el que se conmuta a la tarea responsable de atender esa interrupción.

En cualquiera de los casos, el administrador interumpe al procesador, y cuando éste pasa al estado SCHED conmuta los offsets y define el tiempo asignado a la nueva tarea; este proceso sigue así hasta que el tiempo se agota, el proceso se desactiva o llega una interrupción de mayor prioridad. Un árbitro Round Robin con hasta 8 tareas requiere alrededor de 50 elementos lógicos.

En el caso de un procesador multitareas atendiendo 8 tareas, cada una de ellas con un bus de direcciones de menos de 12 bits, se requieren 30 elementos lógicos para el manejo de áreas privadas de memoria, 45 elementos lógicos para el buffer de punteros de pila y la modificación de la máquina de estados, y 50 elementos lógicos para el árbitro, es decir un uso adicional de recursos de hardware de sólo un 25% en comparación al procesador monotarea.

Es importante considerar que este 25% adicional es de peor caso:

- Para este mismo núcleo del MC6805, si las áreas privadas de memoria se hacen de igual tamaño y de longitud 2^N, los 30 elementos lógicos requeridos para el manejo de áreas privadas de memoria desaparecen, por cuanto las 3 lineas altas del bus final de direcciones salen directamente del árbitro, lo que significa que el *overhead* pasa a ser menor al 20%; de igual modo, si los tiempos asignados a cada slice son iguales, el árbitro también se reduce.
- Para el caso de procesadores más poderosos y complejos, con buses de datos y direcciones mayores, y cuya síntesis requiere más recursos de hardware, la complejidad lógica del agregado para operar en multitareas casi no se modifica, por lo que resulta un porcentaje menor.

La tabla siguiente muestra los productos de IP ofrecidos en el marco del AMPP (ALTERA Mega Partners Program) [16].

Producto	Hardware
FLEX_6805_MT	**645 LEs**
VA 6502	920 LEs
BareCore 8052	1400 LEs
VA Z80	2028 LEs
CAST 8051	2656 LEs

Analizando dicha tabla, se comprueba que incluso con el agregado de manejo de multitareas, este diseño (FLEX_6805_MT) requiere muchísimos menos recursos que las otras alternativas (todas ellas monotareas).

6. CONCLUSIONES

Si una aplicación es completamente conocida antes de iniciar el ciclo de diseño, tanto el hardware como el software pueden ser optimizados para los requerimientos de la aplicación.

Se ha mostrado cómo un núcleo de IP de un procesador convencional puede ser fácilmente ampliado, con mínimo agregado de recursos de hardware, para operar en un contexto de multiprocesamiento y multitarea. Este tipo de soluciones, sumado a los rápidos ciclos de diseño posibles con lógica programable, permiten un tiempo de desarrollo mínimo, una fácil depuración, y una rápida salida al mercado.

7. REFERENCIAS

[1]. Dömer, R. et al, “Specification and Design of Embedded Systems”, it+ti Magazine N° 3, Oldenbourg Verlag, Munich, Germany, June 1998.

[2]. Janka R.S., Wills L.M., “A Novel Codesign Methodology for Real-Time Embedded COTS Multiprocessor-Based Signal Processing Systems”, Proc. of the 8th. Intl. Workshop on Hardware/Software Codesign. San Diego, USA, May 2000, pp.157-161.

[3]. Jaquenod G., "Diseño de un microcontrolador MC6805 usando lógica programable FLEX de ALTERA". VI Workshop IBERCHIP, Sao Paulo, Brasil, Mar 2000, pp.130-139.

[4]. Jaquenod G., De Giusti M., "Diseño de microcontroladores empotrados mediante procesamiento serial: análisis usando FLEX10K para sintetizar un microcontrolador tipo COP8Sax”. VII Workshop IBERCHIP, Montevideo, Uruguay, Mar 2001, Anales en CD.

[5]. Jaquenod G., Villagarcía H., De Giusti M., “Towards a Field Configurable non-homogeneous Multiprocessors Architecture”. SCI 2001, Orlando, Florida, USA, Jul 2001, Proceedings Volume XIV pp.248-253.

[6]. Keating M., Bricaud P., “Reuse Methodology Manual For System-On-A-Chip Designs, Second Edition", Kluwer Academic Publishers 1999, USA, ISBN 0-7923-8558-6.

[7]. Meerwein M. et al, “Linking Codesign and Reuse in Embedded Systems Design”, Proc. of the 8th. Intl. Workshop on Hardware/Software Codesign. San Diego, USA, May 2000, pp.93-97.

[8]. Pollard L.H., “Computer Design and Architecture”, Prentice Hall 1990, USA, ISBN 0-13-167255-X.

[9]. Seepold R, Martinez Madrid N. (Editores), "Virtual Components Design and Reuse", Kluwer Academic Publishers 2000, USA, ISBN 0-7923-7261-1.

[10]. Smith M., “Application Specific Integrated Circuits”, Addison Wesley 1997, USA, ISBN 0-201-50022-1.

[11]. Usselmann R., “Open Cores SoC Bus Review”, Rev.1.0, http://www.opencores.org, Jan. 2001.

[12]. Villagarcía H., Bria O., “Diseño de bloques IP: Programabilidad y Reutilización”. WICC2001, San Luis, Argentina, May 2001, pp.2-5.

[13]. Wolf W., “Computer as Components: Principles of Embedded Computer Systems Design”, Morgan Kaufmann 2000, USA, ISBN 1-55860-541-X.

[14]. ALTERA Corp., “NIOS Soft core Embedded Processor Data Sheet. Version 1”. San José, CA, USA, 2000.

[15]. ALTERA Corp., “ARM-based Embedded Processor Device Overrview. Version 1.1”, “MIPS-based Embedded Processor Device Overrview. Version 1.1”. San José, CA, USA, 2000.
[16]. ALTERA Corp., “Intellectual Property Catalog”, M-CAT-AIPS-01, Altera Corp., 1999, USA.
[17]. INMOS Ltd., “Transputer Reference Manual”, Prentice Hall 1988, UK, ISBN 0-13-929001-X
[18]. MOTOROLA, “MC68HC705C8A/D, Rev.1”, Motorola Inc., 1996, USA.