Tecnología embebida, multinúcleo y virtualización. ¿Está al corriente?

Por Casey Weltzin, director de producto de National Instruments

www.ni.com/es/

El mundo del diseño embebido ha cambiado notablemente en la última década y su evolución no muestra signos de desaceleración. El procesamiento multinúcleo (en forma de multiproceso simétrico (SMP) y multiproceso asimétrico (AMP)) se está convirtiendo en habitual, gracias a las CPUs multinúcleo embebidas se espera que los ingresos crezcan en un factor de x6 desde 2007 a 2011 (Venture Development Corporation). Además, las FPGAs (Field Pogrammable Gate Arrays) han crecido en capacidad y disminuido en su costo, proporcionando la funcionalidad de alta velocidad que solo podía conseguirse con circuitos integrados de aplicación específica (ASICs). Por último, la virtualización está difuminando la conexión entre el hardware y software al permitir que múltiples sistemas operativos puedan funcionar sobre un único procesador. Con la rápida evolución de estas tecnologías, ¿que posibilidades tienen los desarrolladores de sistemas embebidos de estar al corriente? Este artículo explicará brevemente lo que significan estas tecnologías para los diseños embebidos y cómo se pueden aprovechar estos cambios a la vez que se mantiene el tiempo de desarrollo al mínimo.

Huelga decir que el procesamiento multinúcleo representa un enorme cambio en el diseño embebido. Con la presencia de un solo núcleo de procesador en un chip, los diseñadores de sistemas embebidos han sido capaces tradicionalmente de utilizar los lenguajes de programación secuencial como C, incluso para las más complejas aplicaciones. Sin embargo, la presencia de múltiples núcleos de procesamiento en un chip físico complica considerablemente el proceso de diseño. Puesto que la mayoría de los compiladores comerciales no han avanzado para analizar automáticamente las secciones de código que se pueden ejecutar en paralelo, los diseñadores de sistemas embebidos que buscan sacar provecho de los procesadores multinúcleo deben hacer uso de las API de programación paralela que añaden una sobrecarga al código y son difíciles de depurar. Además, los programas secuenciales hacen que sea muy difícil visualizar las rutinas de tipo paralelo, creando un gran problema para los códigos más antiguos de los diseñadores (o agobiándoles con sus propias aplicaciones complejas). Si la programación paralela de hoy en día es difícil para los diseñadores, ¿cómo les irá cuando sufran el desafío de la próxima generación de procesadores (con 16 o más núcleos)?

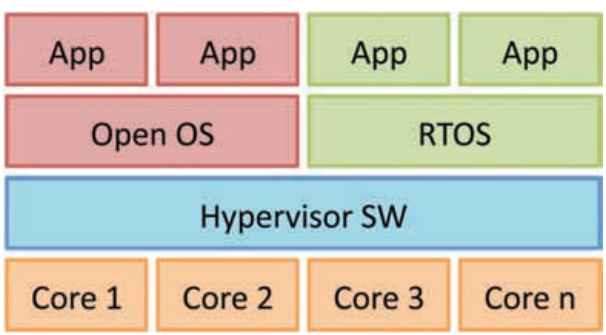

Figura 1. La utilización de un lenguaje de flujo de datos como LabVIEW puede acelerar el desarrollo de aplicaciones embebidas de tipo paralelo.

La solución más obvia para este problema es utilizar mejor las herramientas y los métodos de programación para quitarle complejidad al hardware multinúcleo. Si bien, APIs como OpenMP y POSIX se han convertido en algo común para las aplicaciones en paralelo, las nuevas APIs, tales como MCAPI (Multicore Communications API) prometen ser más escalables y soportar una amplia variedad de arquitecturas de hardware paralelo (PSM y AMP). Además, los nuevos paquetes de herramientas, tales como Intel Parallel Studio proporcionan mejores herramientas de depuración que las anteriormente disponibles. Por último, los lenguajes gráficos de flujo de datos, como NI LabVIEW, proporcionan un modelo de programación paralelo intrínseco para el SMP que puede reducir enormemente el tiempo de lanzamiento al mercado. La pregunta es, ¿por qué programar en serie cuando se supone que la aplicación debe funcionar en paralelo? Mediante el análisis automático de las secciones paralelas del código y el mapeado de esas secciones en varios hilos, los lenguajes de flujo de datos permiten centrarse en la principal tarea: el rápido y conciso desarrollo del código.

Imaginemos el típico proceso de diseño de software embebido para un proyecto. Una gran aplicación embebida se inicia probablemente con un diagrama de flujo y a continuación, las piezas del diagrama de flujo se traducen a código y se implementan. Con la programación mediante flujo de datos, se puede saltar un paso, el código puede ser ejecutado en paralelo tal como se establece en el diagrama de flujo, sin traducción a un lenguaje secuencial. De esta forma, la inversión en herramientas de programación paralela (incluidas las nuevas APIs y IDEs que soportan los lenguajes de flujo de datos) ayudarán a sacar el máximo provecho de los avances de la tecnología multinúcleo en los diseños embebidos.

Como continuación, las FPGAs han cambiado la forma en que se implementan los diseños embebidos de alta velocidad y ampliamente en paralelo y sin duda, continuarán evolucionando en el futuro. En el pasado, la implementación de las rutinas de procesamiento de señal personalizadas, como el filtrado digital basado en hardware, requería el diseño de un ASIC que suponía un gasto significativo en el diseño inicial. Si bien esto pudo haber sido rentable para aplicaciones de gran volumen de producción, en el caso de los diseños de sistemas embebidos de bajo volumen de producción se vieron obligados a utilizar una combinación de los existentes ASICs o a hacer funcionar el código de

procesamiento de señales sobre un procesador considerablemente más lento al estar basado en software. Las FPGAs han sido las que han cambiado el juego. Ahora, basta con descargar las aplicaciones de procesamiento de señal personalizadas a una FPGA y ejecutarlas en su hardware, a un costo de sólo unas decenas de dólares. Además, puesto que las FPGAs implementan las aplicaciones embebidas en el hardware, son por naturaleza ampliamente paralela. Con todas estas ventajas, ¿cómo se puede hacer un mejor uso de las FPGAs en los diseños embebidos y desarrollarlos en menos tiempo?

Uno de los principales retos a los que se enfrentan los desarrolladores de sistemas embebidos radica en la diferencia de las herramientas de diseño utilizadas para programar las FPGAs y los microprocesadores. Si bien muchos desarrolladores se sienten cómodos escribiendo el código C de alto nivel (al menos para aplicaciones secuenciales de microprocesadores), la programación de las FPGAs se suele hacer en un lenguaje de descripción de hardware (HDL), tal como VHDL. Esta diferencia fundamental en la comunicación entre los desarrolladores (hay que pensar que hablan lenguajes distintos), puede añadir un obstáculo importante en el ciclo de desarrollo, sobre todo cuando las FPGAs y los procesadores son utilizados en un único diseño. Para resolver este problema se han desarrollado una serie de herramientas para traducir las aplicaciones en C al código HDL (tal como Impulse CoDeveloper), que permiten especificar las aplicaciones de alto nivel y a continuación, destinar esas aplicaciones a la FPGAs. Además, los lenguajes gráficos de flujo de datos, tales como LabVIEW permiten desarrollar para FPGAs sin conocimientos específicos de HDL. Puesto que el flujo de datos proporciona un método inherentemente paralelo a la programación, también permite aprovechar la naturaleza ampliamente en paralelo de las FPGAs de forma automática. El mensaje es simple: el uso de las estrategias de diseño de alto nivel de las FPGAs (tales como los lenguajes de flujo de datos y C para traductores HDL)

Figura 2. La instalación de un hipervisor permite el multiprocesamiento asimétrico (AMP) en un conjunto homogéneo de núcleos de procesadores

puede maximizar la eficiencia del equipo de diseño y reducir el tiempo de lanzamiento del producto al mercado.

Por último, una de las más recientes tecnologías en entrar en la escena de los sistemas embebidos es la virtualización. La idea principal que subyace detrás de esta tecnología es la de hacer un mejor uso del hardware de procesamiento abstrayendo los detalles de la plataforma de hardware específica de los sistemas operativos y de las aplicaciones. En concreto, una forma de utilizar la virtualización en los diseños embebidos es instalar un software denominado hipervisor, que permitirá que varios sistemas operativos en paralelo funcionen simultáneamente. Esto termina por tener consecuencias positivas tanto en la capacidad general de un sistema embebido como en el uso del hardware multinúcleo. En un sistema homogéneo con múltiples núcleos de procesadores, un hipervisor hace que sea fácil de construir una arquitectura de software AMP, donde los sistemas operativos son asignados a uno o más núcleos. A alto nivel alto, se puede pensar que la tecnología de virtualización es como hacer que el hardware multinúcleo sea multitalento.

Aunque a menudo los diseñadores programan los sistemas embebidos enteros desde cero, la presión para reducir el tiempo de desarrollo (y por lo tanto el costo) ha dado lugar a un mayor uso de sistemas operativos en el dominio de los sistemas embebidos. Sin embargo, esto presenta un problema: ¿cómo hacer que los ingenieros equilibren la necesidad de los servicios y la interfaz de usuario proporcionada por un sistema operativo comercial con las prestaciones de tiempo real necesarias para una aplicación embebida? Imagínese, por ejemplo, que se está diseñando una máquina de imágenes médicas. ¿Cómo se pueden aprovechar las capacidades de la interfaz de usuario incorporada de un sistema operativo como Linux, al mismo tiempo que se procesan los datos de la imagen en tiempo real? El uso de un hipervisor puede hacer frente a estos desafíos. Ejecutando simultáneamente un sistema operativo comercial de buenas características y un sistema operativo en tiempo real en paralelo se puede reducir el tiempo de desarrollo para las aplicaciones embebidas, manteniendo al mismo tiempo el determinismo.

Como conclusión, aunque las tendencias de la tecnología de sistemas embebidos, que incluyen el procesamiento multinúcleo, FPGAs y virtualización, presentan una gran punto de partida con respecto a las tradicionales técnicas de desarrollo, es evidente que hay pasos que se puede dar muy claramente para aprovechar y mantener la competitividad. En primer lugar, la adopción de herramientas de programación que reduzcan la complejidad del hardware, como es el caso del procesamiento con múltiples núcleos o las puertas de FPGAs. Al concentrarse en la implementación del diseño, a la vez que se invierte menos tiempo en realizar los ajustes en la arquitectura del hardware subyacente, se pueden lanzar más rápidamente los productos embebidos al mercado. Los entornos de programación con características de depuración en paralelo, las nuevas APIs de programación en paralelo, la programación de flujo de datos y los convertidores de C a HDL pueden ayudar a lograr estos objetivos. Además, el empleo de la virtualización permite aprovechar el procesamiento en tiempo real y los servicios de los sistemas operativos comerciales para reducir el tiempo de desarrollo y aprovechar al máximo el hardware multinúcleo. En tanto que la siguiente generación de sistemas embebidos crezca con más potencia que nunca, el aprovechamiento de estas últimas tecnologías ayudará a que las compañías permanecezcan por delante de la curva.