Uso de instrumentos modulares con el software de la librería de programación de entrada/salida de VISA y una DLL de traducción

Por Andy Purcell, Agilent Technologies, Inc.

Agilent Technologies
www.agilent.com

Muchos sistemas de prueba utilizan instrumentos modulares, los cuales ofrecen numerosas ventajas, entre las que podemos destacar su capacidad para acortar el tiempo de prueba y minimizar las dimensiones del sistema de prueba. Ello puede observarse sobre todo en los últimos instrumentos modulares que usan PCI-Express (PCIe) cableado. Dichos instrumentos son conocidos como instrumentos PXIe y AXIe. PXIe [1] es el estándar PCIe ampliado para instrumentos. AXIe[2] es un estándar basado en AdvancedTCA con ampliaciones para instrumentos. Los tiempos de latencia para operaciones de registro en dispositivos PXIe y AXIe pueden ser de tan sólo 1,5 microsegundos. Con una conexión de 4 caminos (x4) entre el PC y un chasis PXIe, las velocidades de transferencia pueden alcanzar 2 GBytes/s. Así, tanto en términos de latencia como de producción, PCIe cableado supera a los sistemas LAN, USB y todos los demás buses utilizados hoy en día en el sector de medida y prueba.

Actualmente existen estándares para software de E/S específicos para dispositivos modulares PXIe y AXIe. El estándar industrial VISA, disponible en http://www.ivifoundation.org/, define una API para desarrolladores de aplicaciones que puede ser utilizada para comunicarse con dispositivos PXIe y AXIe. Las aplicaciones que integran el estándar VISA son fáciles de escribir y admiten mayor soporte.

Este artículo examinará la API VISA existente para dispositivos modulares e ilustrará cómo utilizar un sistema basado en una DLL de traducción para poder usar con mayor libertad dispositivos modulares de diversos fabricantes.

El estándar VISA y los dispositivos modulares

La función VISA *viMapAddress()* permite a una aplicación mapear los registros de los módulos PXIe/AXIe en la memoria de la aplicación (espacio de usuario). Por tanto, la aplicación puede escribir en el registro de un módulo o leer el registro de un módulo directamente. Es una manera de controlar el conmutador electrónico de un módulo o hacer una simple transferencia de datos. La aplicación tiene que ejecutar *viUnmapAddress()* una vez que se ha completado la operación.

La función VISA *viMoveOut32()* permite a una aplicación trasladar un bloque de datos hasta un módulo. El estándar VISA hace lo mismo que *viMapAddress()*, transfiere los datos de una memoria de aplicación al dispositivo y, luego, hace lo mismo que *viUnmapAddress()*. La función *viMoveIn32()* funciona del mismo modo, pero la transferencia se hace del dispositivo a la memoria de aplicación. Para mover realmente los datos, VISA puede hacer una llamada a un controlador del kernel o bien dejar que el controlador del kernel mueva los datos mediante acceso directo a memoria (DMA), o VISA puede mover los datos mediante la CPU (conocido también como modo PIO).

El modo PIO (Programmed IO) realiza un uso intensivo de CPU y no es tan rápido transfiriendo datos como el modo DMA.

Por tanto, el DMA es el modo preferido para mover datos. Sin embargo, para usar el modo DMA es necesario que VISA realice una llamada al controlador del kernel para ese módulo, ya que VISA no dispone de conocimientos suficientes. Sólo el controlador del kernel del módulo conoce los registros hardware DMA específicos del módulo. Además, VISA sólo puede llamar a controladores del kernel si conoce la API del controlador del kernel. Por lo general, esto sólo es posible cuando VISA y los controladores del kernel son creados por la misma empresa. Si la empresa XYZ suministra el software VISA, entonces VISA sólo puede realizar el acceso directo a memoria de los módulos suministrados por la empresa XYZ.

Evidentemente, se requiere un entorno más abierto para poder habilitar el acceso directo a memoria para todos los módulos.

Algunos pueden decir que como VISA ofrece un modo de mapear registros de módulos, una capa de software por encima de VISA podría mapear registros DMA y ofrecer acceso directo a memoria desde el espacio de usuario. Este enfoque puede ser adoptado para trabajar usando *viMemAlloc()* y algunas otras funciones VISA. Sin embargo, un enfoque más conveniente, y más tradicional, sería dejar que un controlador del kernel realizara acceso directo a memoria y dejar que un controlador del kernel manejara la interrupción DMA al finalizar el acceso DMA. Al fin y al cabo, es el controlador del kernel el que entiende las direcciones físicas necesarias para configurar los descriptores de DMA scatter-gather. Es el controlador del kernel el que puede responder con mayor rapidez a las interrupciones DMA.

Para crear un entorno más abierto que permita controlar el bus DMA en módulos de diversos fabricantes, y para habilitar otras funciones de valor añadido de desarrolladores de módulos, es posible usar un sistema basado en una DLL de traducción. La API de la DLL de traducción hace de puente entre la capa de software VISA y el controlador del kernel de un módulo. Véase la Figura 1.

Sistema basado en DLL de traducción

El sistema basado en DLL de traducción consiste en entradas de registro del sistema operativo y una API implementada en una DLL. Las entradas de registro, si las hay, permiten a VISA entender que el sistema basado en la DLL de traducción debe ser usado para un determinado módulo. Las entradas de registro se encuentran en la parte del registro específica del módulo. Dan nombre a las DLL de traducción de 32 y 64 bits, y proporcionan las rutas hasta las DLL. VISA puede así cargar dinámicamente la DLL y buscar las direcciones de las distintas API.

A continuación se enumeran las funciones requeridas para una DLL de traducción:

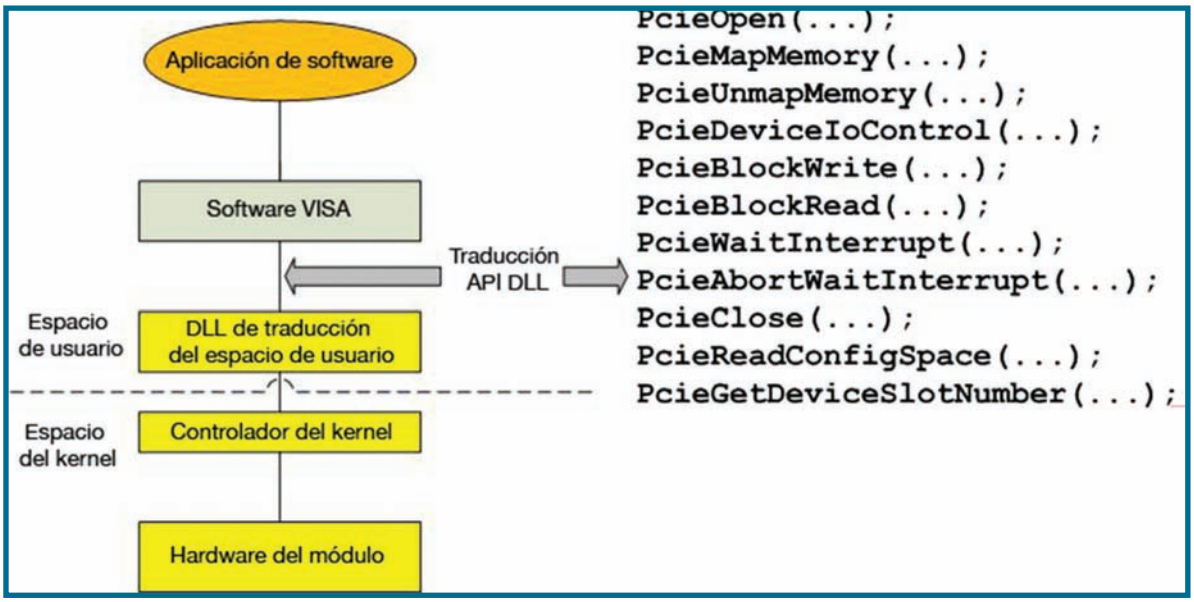

Figura 1. Sistema basado en DLL de traducción

- *DWORD PcieOpen(int intfc, int bus, int device, int function, HANDLE *pHandle);*
  Abre un identificador (handle) del dispositivo. Los dispositivos PCIe son identificados de forma unívoca por una interfaz, bus, dispositivo y función.
- *DWORD PcieMapMemory(HANDLE handle,BAR bar,IoIsPcieBarOffset offset,IoIsPcieLength length,void **pUserSpaceMem);*
  Mapea la memoria de registro de dirección base del módulo en el espacio de usuario.
- *DWORD PcieUnmapMemory(HANDLE handle,BAR bar,IoIsPcieBarOffset offset, IoIsPcieLength length,void *pUserSpaceMem);*
  Deshace el mapeado de memoria.
- *DWORD PcieDeviceIoControl(HANDLE h,DWORD dwIoControlCode,LPVOID lpInBuffer,DWORD nInBufferSize,LPVOID lpOutBuffer,DWORD nOutBufferSize,LPDWORD lpBytesReturned,DWORD dwTimeoutMilliseconds);*
  Realiza la acción específica del llamante para el código = dwIoControlCode. Es un mecanismo "pass-through" específico de VISA que permite a las aplicaciones comunicarse directamente con el controlador del kernel. En la implementación VISA de Agilent, la llamada es viSetAttribute (sess, VI_AGATTR_DEVICE_IO_CONTROL, ViAttrState);
- *DWORD PcieBlockWrite(HANDLE h,int timeoutMs,int useDma,void *mapHandle,IoIsPcieLength offset,int width,bool increment,void *writeBuffer,IoIsPcieLength count);*
  Efectúa una escritura por bloques. El argumento useDma es configurado de acuerdo con el atributo VI_ATTR_DMA_ALLOW_EN de VISA.
- *DWORD PcieBlockRead(HANDLE h,int timeoutMs,int useDma,void *mapHandle,IoIsPcieLength offset,int width,bool increment,void *readBuffer,IoIsPcieLength count);*
  Hace una lectura por bloques. El argumento usDma es configurado de acuerdo con el atributo VI_ATTR_DMA_ALLOW_EN de VISA.
- *DWORD PcieWaitInterrupt(HANDLE h);*
  Espera una interrupción.
- *DWORD PcieAbortWaitInterrupt(HANDLE h);*
  Cancela una espera de interrupción.
- *DWORD PcieClose(HANDLE handle);*
  Cierra el identificador del dispositivo.
- *DWORD PcieReadConfigSpace(HANDLE h, IoIsPcieBarOffset offset, int length, void *pBuffer);*
  Lee el espacio de configuración PCI.
- *DWORD PcieGetDeviceSlotNumber(HANDLE h, int *pSlotNumber);*
  Lee el número del slot donde está el módulo.

Conclusión

Es posible utilizar un nuevo sistema basado en una DLL de traducción de software que sirve de enlace entre el software VISA y software de bajo nivel específico de módulos creado por desarrolladores de módulos. Actualmente, existe un entorno más abierto que permite a todos los desarrolladores de módulos ofrecer completo soporte a sus módulos, incluida la optimización de transferencias de datos con DMA. El sistema basado en DLL de traducción está disponible en la familia de librerías de programación de entrada/salida 16.0 de Agilent (Agilent I/O Library Suite 16.0). Para más información, visite la web www.agilent.com y busque "Agilent IO Libraries Suite".