

Ciudad de México, 18 de Octubre



ni.com/mexico



El Cambio hacia un Mundo Paralelo

Descubriendo el Poder de las Tecnologías Concurrentes

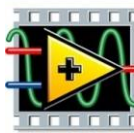
Rick Ary
Gerente de Ventas Internacional
México y Canadá

Diseño Gráfico de Sistemas



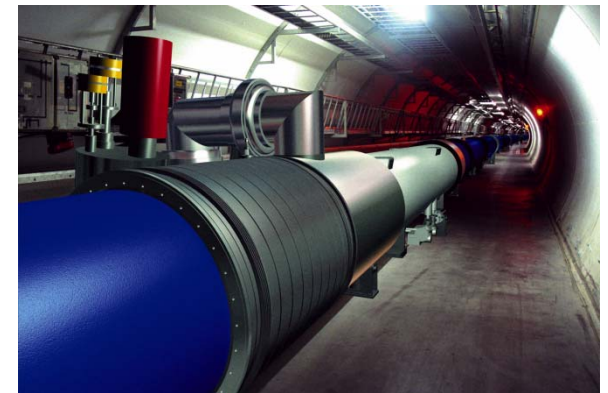
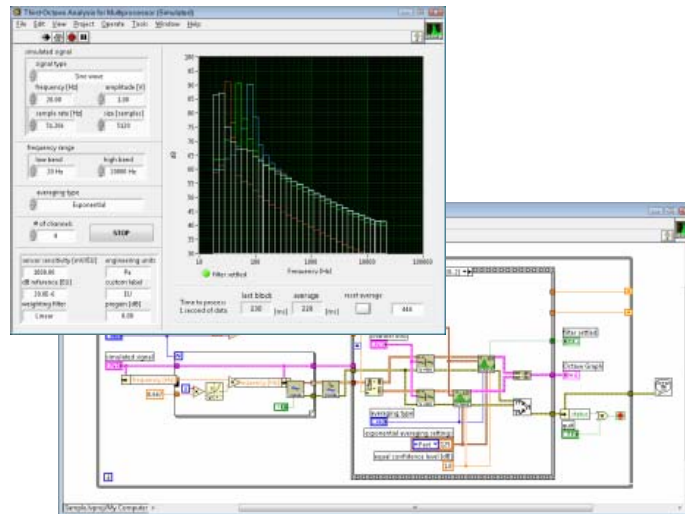
LEGO MINDSTORMS NXT
“el juguete más inteligente y emocionante del año”

POWERED BY



NATIONAL INSTRUMENTS

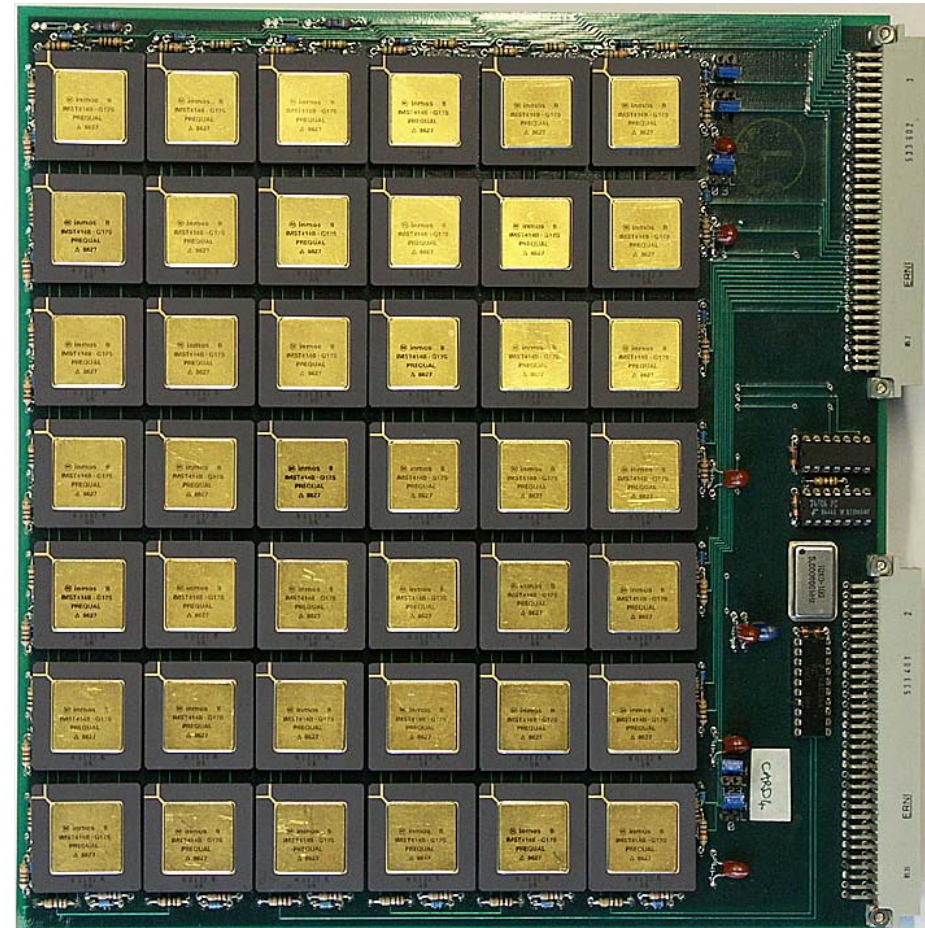
LabVIEW™



CERN Large Hadron Collider
“el instrumento más poderoso de la tierra”

Demanda Histórica por Soluciones Paralelas

El “transputador”
(computadora transistor):
primer microprocesador de
propósito general diseñado
específicamente para
utilizarse en sistemas de
cómputo paralelo.



El Transputador Inmos de 1979

Demanda Histórica por Soluciones Paralelas

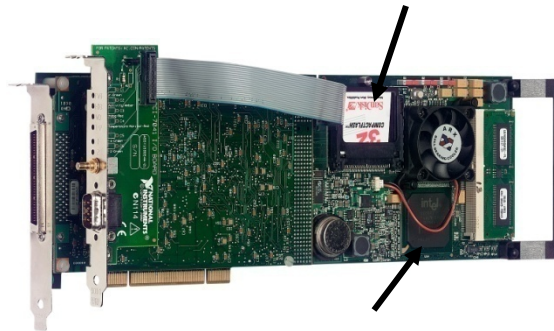
Dispositivos de Medición Paralela de National Instruments

DSP en Tarjeta



1993 AT-DSP Board

Disco Compact Flash



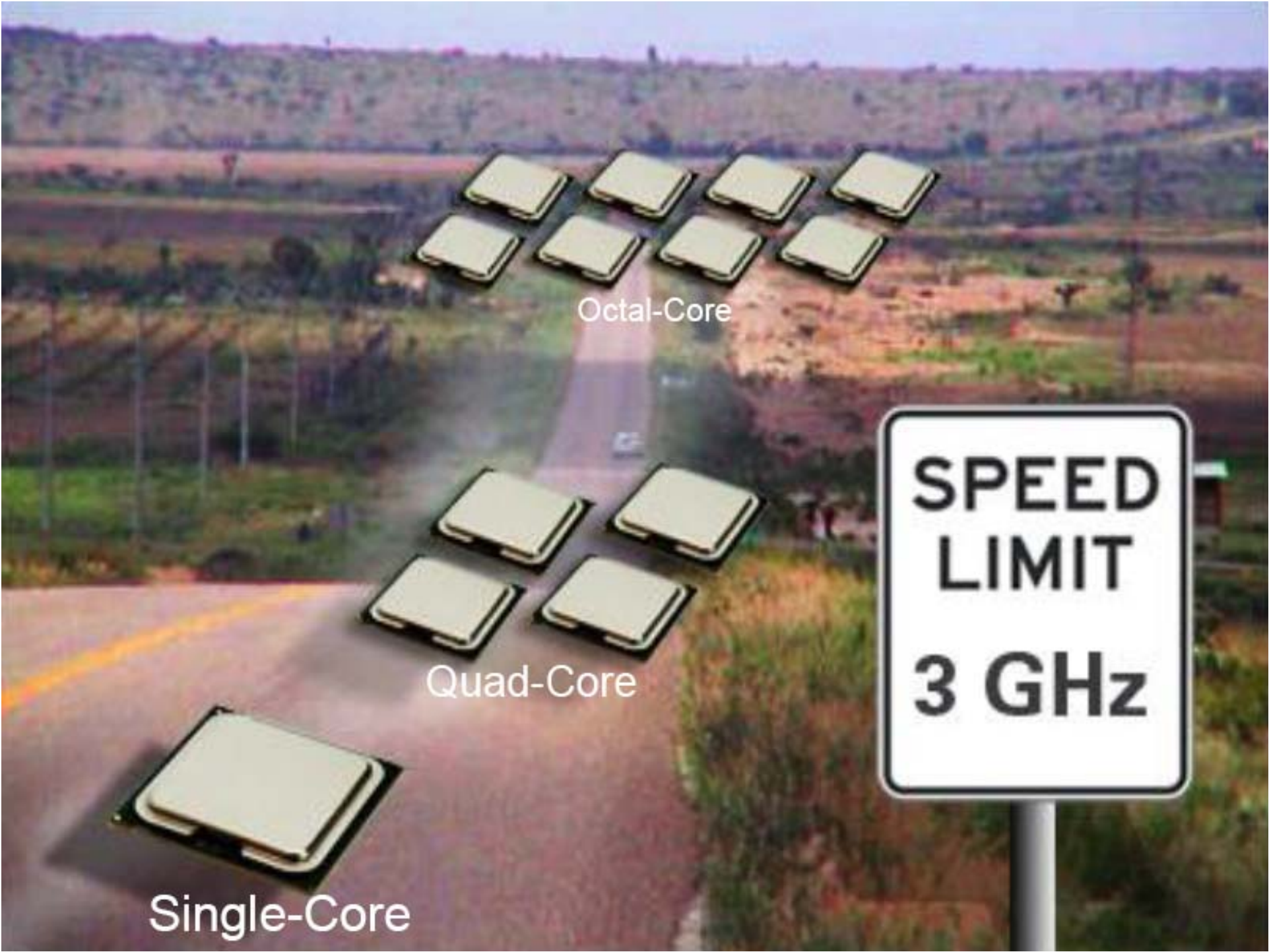
Procesador de hasta 1 GHz

1999 RT Board

Procesamiento en tarjeta con FPGA



2004 AWG



Octal-Core

Quad-Core

Single-Core

**SPEED
LIMIT
3 GHz**

Surgimiento de Tecnologías Paralelas

Núcleo Único	Multinúcleo
Hilo Único	Multihilos
ASICs	FPGAs
Bus Compartido	Bus Punto a Punto
Driver no Preparado para Ejecución en Hilos	Driver Listo para Ejecución en Hilos

Surgimiento de Tecnologías Paralelas

Núcleo Único

Hilo Único

ASICs

Bus Compartido

Drivers no Preparado para
Ejecución en Hilos

Multinúcleo

Multihilos

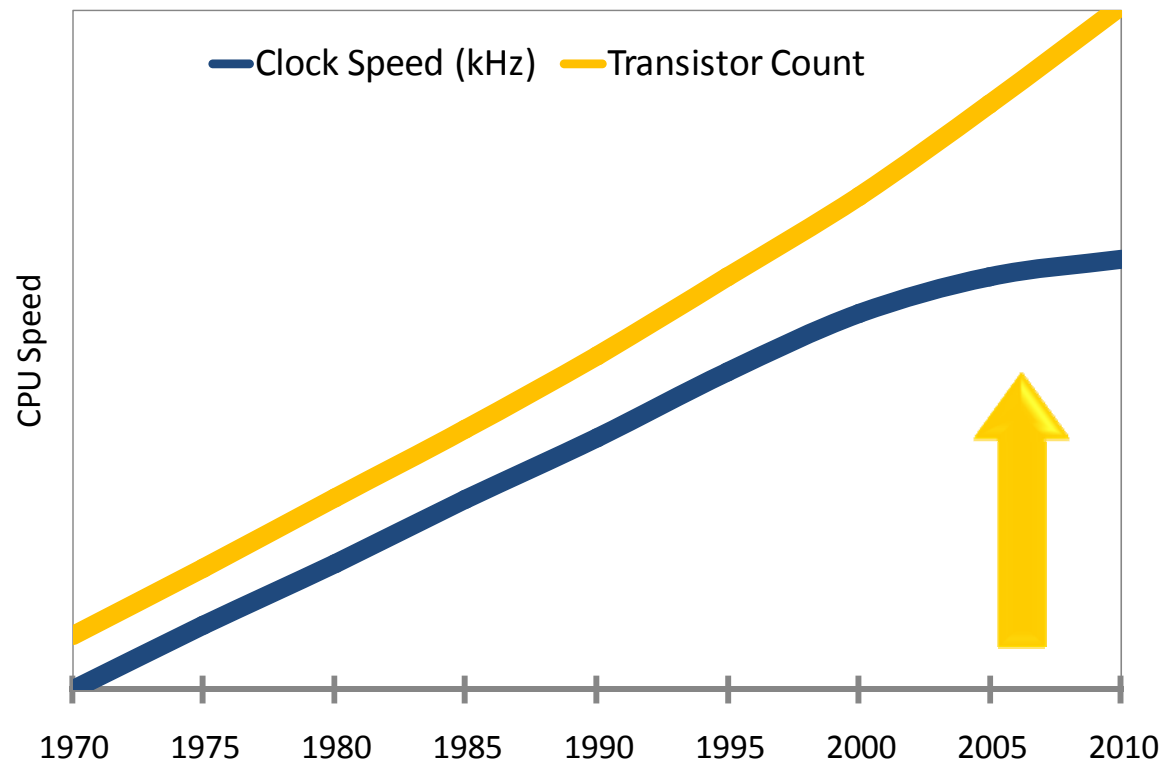
FPGAs

Bus Punto a Punto

Drivers Listo para
Ejecución en Hilos

Las Arquitecturas Paralelas Impulsan el Desempeño

Procesadores más rápidos → Procesadores Multinúcleo



Procesador de Cuatro Núcleos
Intel QX6700

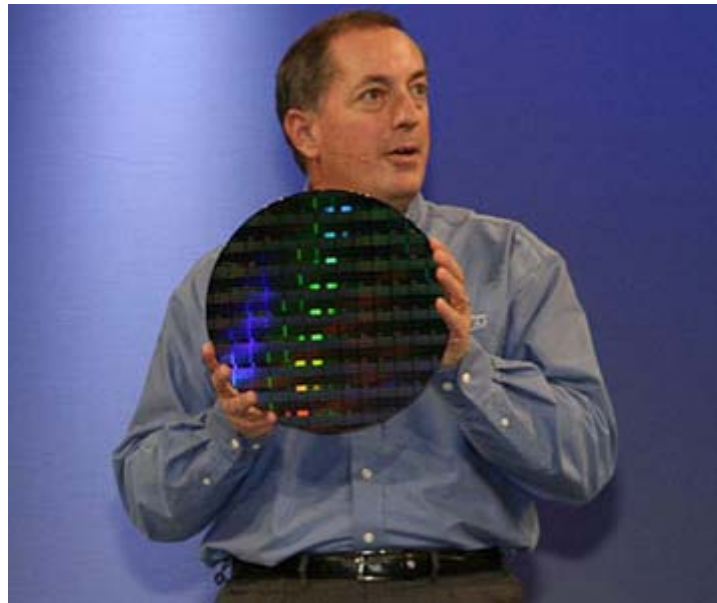
- 4 procesadores (un par de chips Core 2)
- Velocidad de Reloj 2.66 GHz

Acelerando el Cambio a Multinúcleo

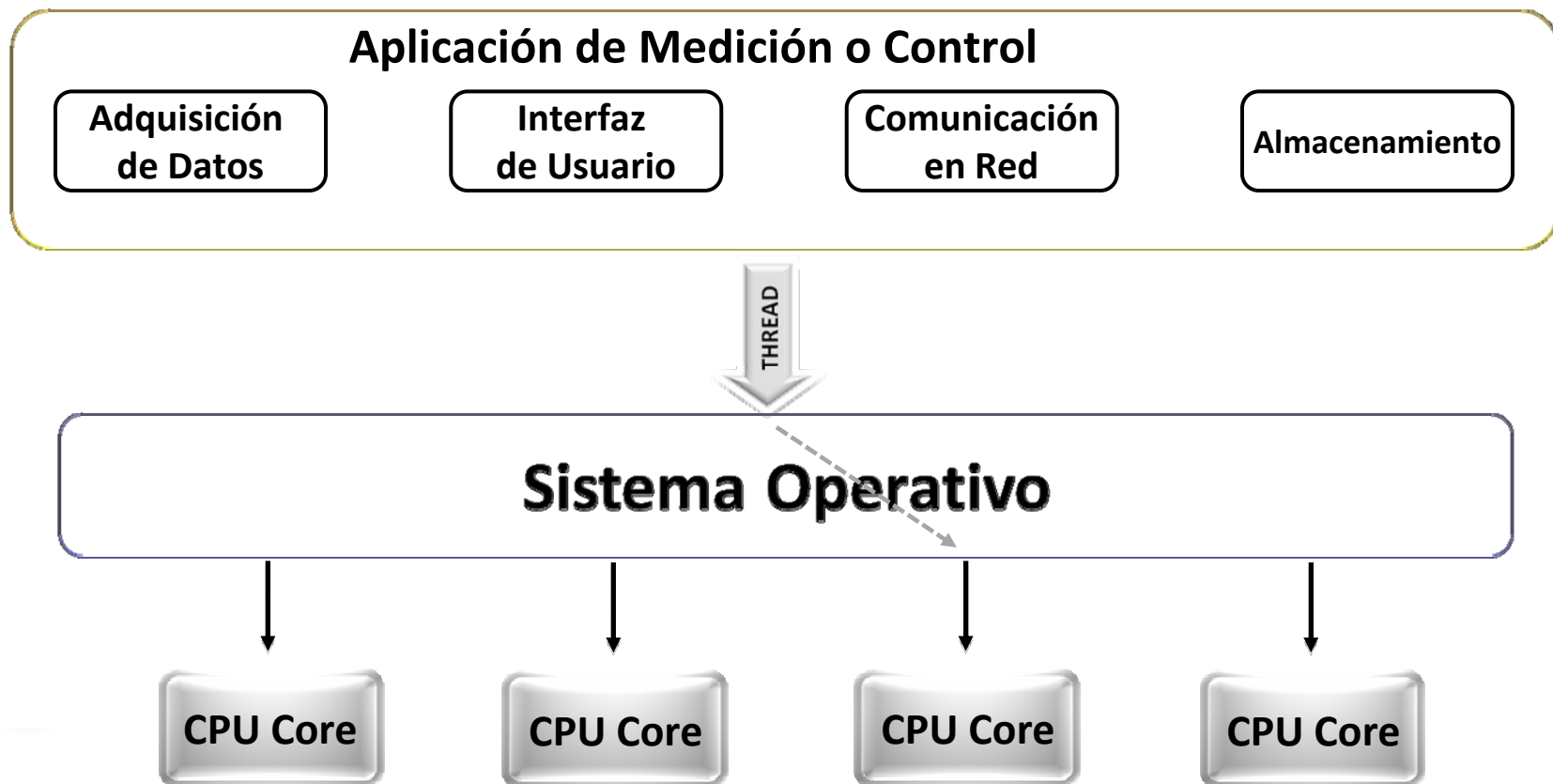
“Intel pronostica 80 núcleos en cinco años”

- Paul Otellini, Intel CEO

Intel Developer Forum, September 2006

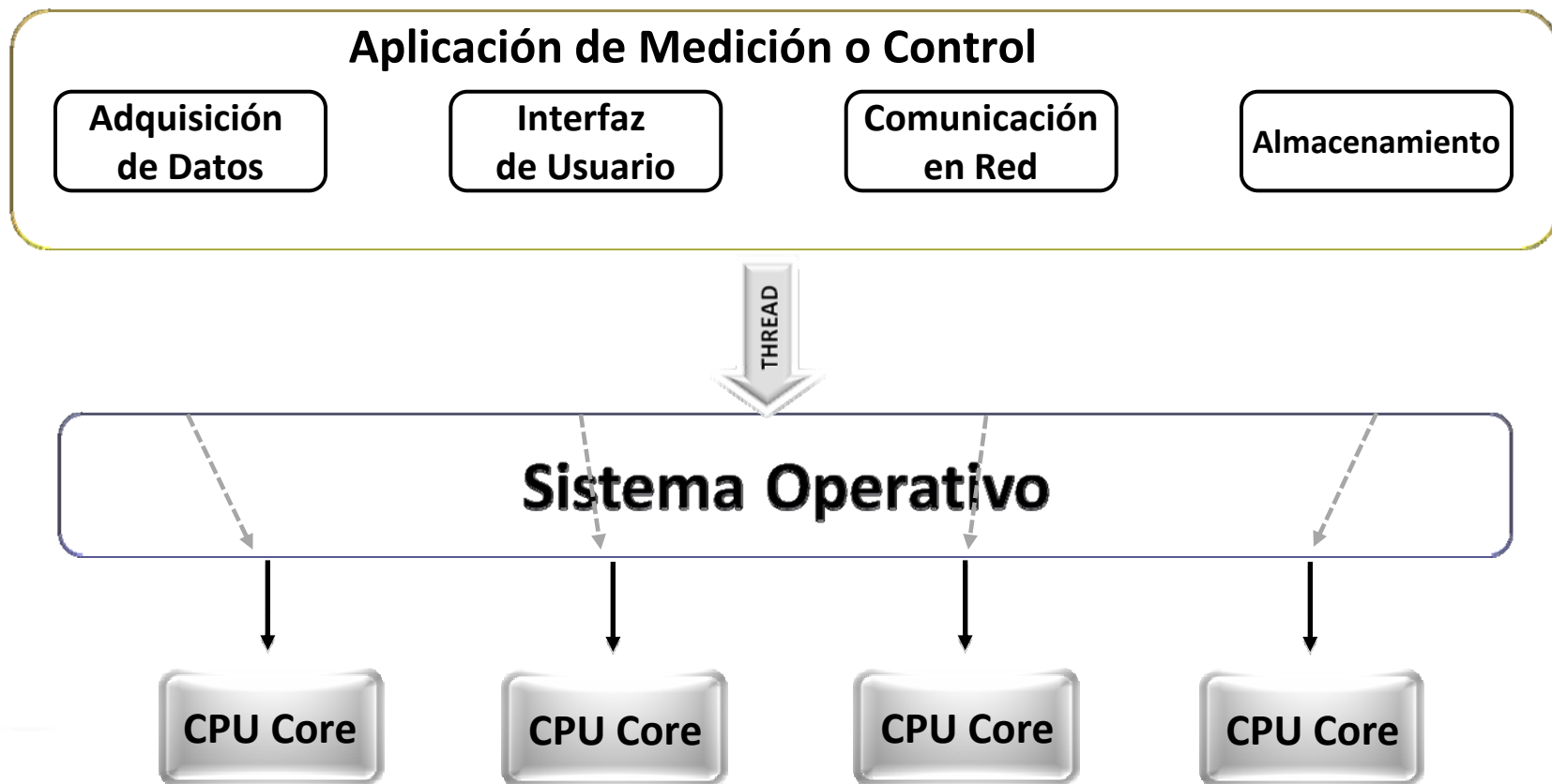


Impacto en Ingenieros y Científicos



Impacto en Ingenieros e Investigadores

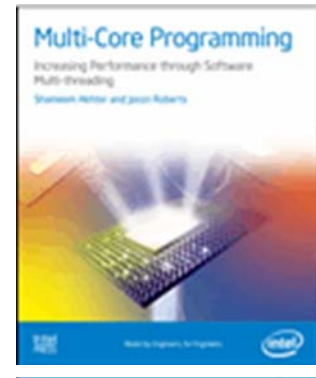
Usted **debe** crear aplicaciones multihilo para maximizar el beneficio de procesadores multinúcleo.



Programar Hilos es Difícil

Los desarrolladores deben utilizar herramientas convencionales para:

- Crear y destruir hilos
- Comunicación entre hilos
- Sincronizar hilos
- Depurar hilos



Ejemplo: Programación Multihilo en Ambientes de Texto

```
#include <windows.h>
#include <stdio.h>
#include <conio.h>

// CriticalSection.cpp
#include <windows.h>
#include <stdio.h>
#include <process.h>
CRITICAL_SECTION g_criticalSection;
bool g_bThreadOneFinished = false;
bool g_bThreadTwoFinished = false;
int g_iResult = 0;
void ThreadOne( void *)
{
    for ( int i = 0; i < 10000; i++)
    {
        // Request ownership of the critical section
        EnterCriticalSection(&g_criticalSection);
        // Access the shared resource
        g_iResult += 1;
        // Release ownership of the critical section
        LeaveCriticalSection(&g_criticalSection);
    }
    // Finished
    g_bThreadOneFinished = true;
    _endthread();
}
```

```
void ThreadTwo( void *)
{
    for ( int i = 0; i < 10000; i++)
    {
        // Request ownership of the critical section
        EnterCriticalSection(&g_criticalSection);
        // Access the shared resource
        g_iResult += 1;
        // Release ownership of the critical section
        LeaveCriticalSection(&g_criticalSection);
    }
    // Finished
    g_bThreadTwoFinished = true;
    _endthread();
}

int main()
{
    // Initialize the critical section
    InitializeCriticalSection(&g_criticalSection);
    // Start the threads
    _beginthread(ThreadOne, 0, NULL);
    _beginthread(ThreadTwo, 0, NULL);
    // Wait for the threads to finish
    while (( false == g_bThreadOneFinished)
        || ( false == g_bThreadTwoFinished))
    {
        Sleep(1);
    }
    // Release resources used by the critical section
    DeleteCriticalSection(&g_criticalSection);
    // Print the result
    printf("Result: %i\n", g_iResult);
}
```

Ejemplo: Programación Multihilo en Ambientes de Texto

```
#include <windows.h>
#include <stdio.h>
#include <conio.h>
```

```
// CriticalSection.cpp
```

```
#include <windows.h>
#include <stdio.h>
#include <process.h>
```

```
CRITICAL_SECTION g_criticalSection;
```

```
bool g_bThreadOneFinished = false;
```

```
bool g_bThreadTwoFinished = false;
```

```
int g_iResult = 0;
```

```
void ThreadOne( void *)
```

```
{
```

```
    for ( int i = 0; i < 10000; i++)
```

```
    {
```

```
        // Request ownership of the critical section
        EnterCriticalSection(&g_criticalSection);
```

```
    // Access the shared resource
```

```
        g_iResult += 1;
```

```
    // Release ownership of the critical section
```

```
        LeaveCriticalSection(&g_criticalSection);
```

```
    }
```

```
    // Finished
```

```
        g_bThreadOneFinished = true;
```

```
        _endthread();
```

```
    }
```

```
}
```

Hilo 1

Hilo 2

```
void ThreadTwo( void *)
```

```
{
```

```
    for ( int i = 0; i < 10000; i++)
```

```
    {
```

```
        // Request ownership of the critical section
        EnterCriticalSection(&g_criticalSection);
```

```
    // Access the shared resource
```

```
        g_iResult += 1;
```

```
    // Release ownership of the critical section
```

```
        LeaveCriticalSection(&g_criticalSection);
```

```
    }
```

```
    // Finished
```

```
        g_bThreadTwoFinished = true;
```

```
        _endthread();
```

```
    }
```

```
int main()
```

```
{
```

```
    // Initialize the critical section
```

```
    InitializeCriticalSection(&g_criticalSection);
```

```
    // Start the threads
```

```
    _beginthread(ThreadOne, 0, NULL);
```

```
    _beginthread(ThreadTwo, 0, NULL);
```

```
    // Wait for the threads to finish
```

```
    while (( false == g_bThreadOneFinished)
```

```
        || ( false == g_bThreadTwoFinished))
```

```
    {
```

```
        Sleep(1);
```

```
    }
```

```
    // Release resources used by the critical section
```

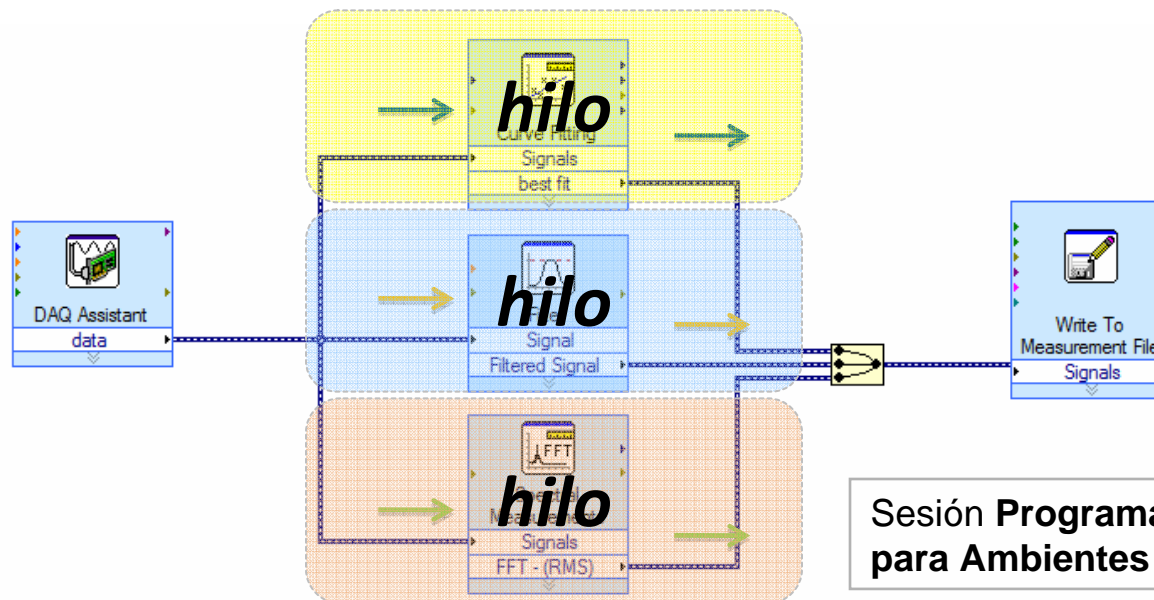
```
    DeleteCriticalSection(&g_criticalSection);
```

```
    // Print the result
```

```
    printf("Result: %i\n", g_iResult);
```

Simplificando la Programación Multihilo con Ambientes Gráficos

- Enfoque gráfico expone intuitivamente las oportunidades de crear hilos
- LabVIEW crea automáticamente múltiples hilos (se introdujo multihilo en 1998)



Demo:

Multihilo con LabVIEW en Sistema Multinúcleo

GV8

Slide 17

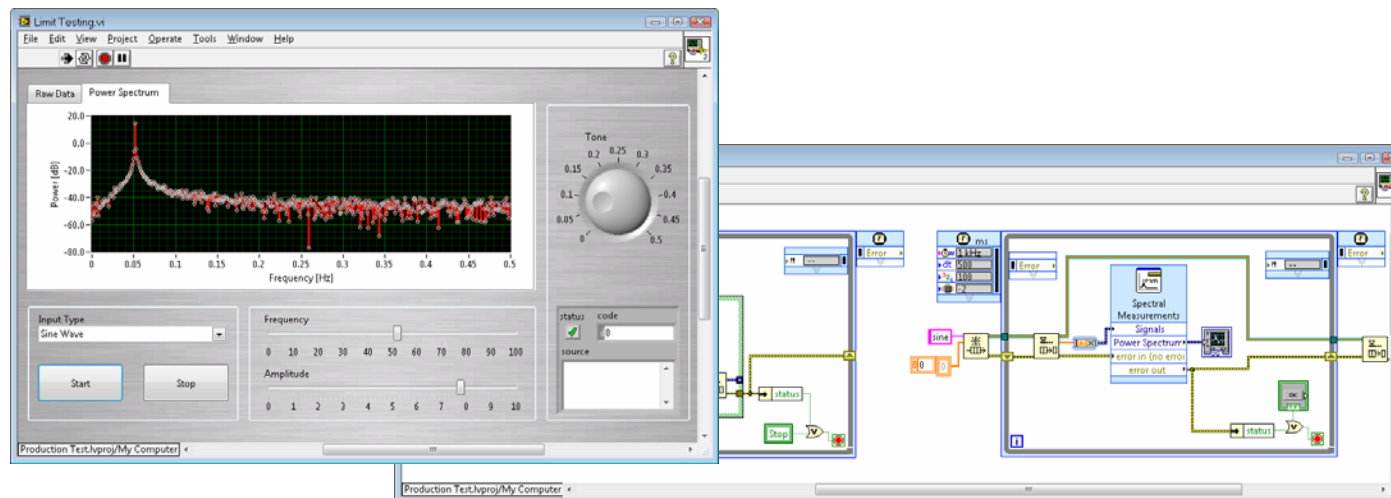
GV8

remove slide

Gustavo Valdes, 10/12/2007

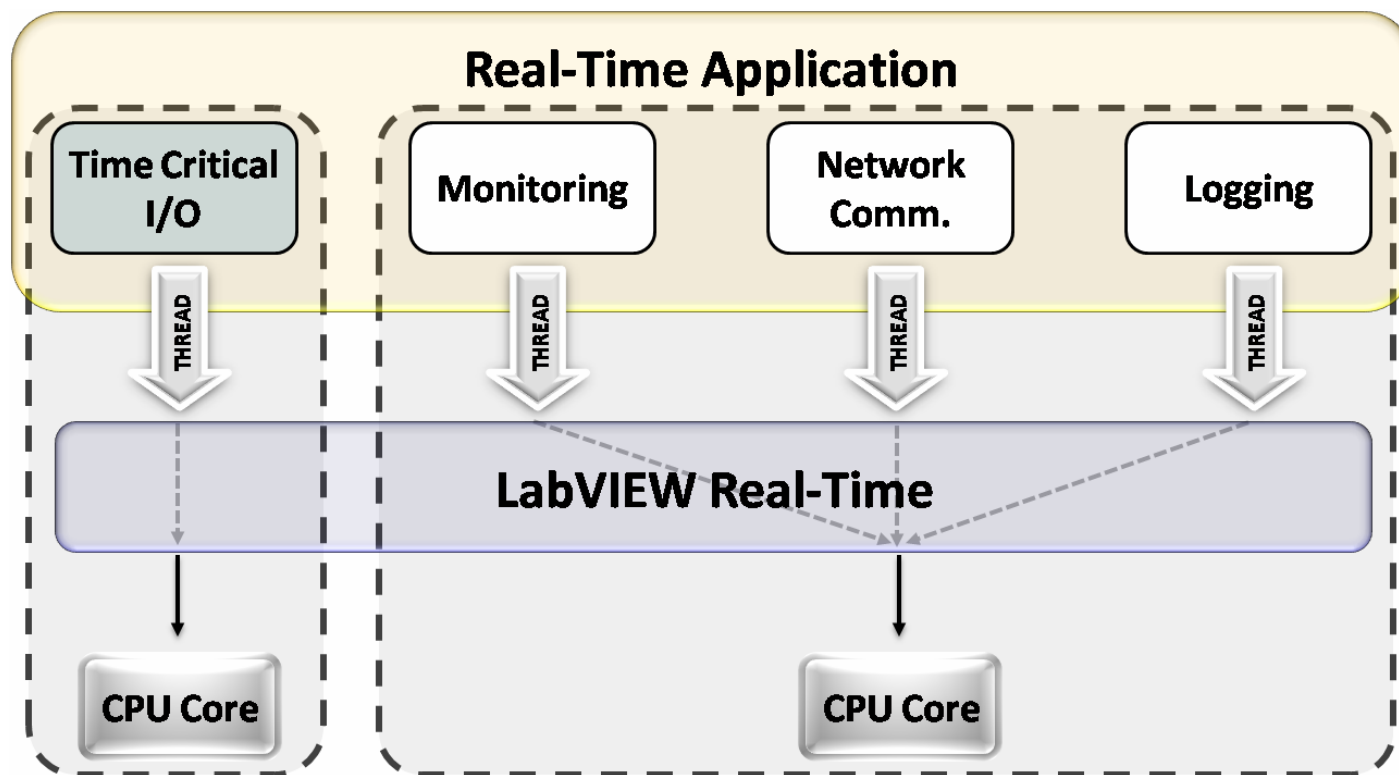
Nuevos Avances en LabVIEW 8.5 para Programación Multihilo

- Crea más hilos con más núcleos disponibles
- *Drivers* de hardware listos para ejecución en multihilo (bloqueo y reentrante)
- Capacidad de asignar código a un núcleo en específico



Multihilos Determinísticos en LabVIEW Real-Time

Los usuarios pueden asignar y ejecutar código en núcleos específicos



Surgimiento de Tecnologías Paralelas

Núcleo Único Multinúcleo

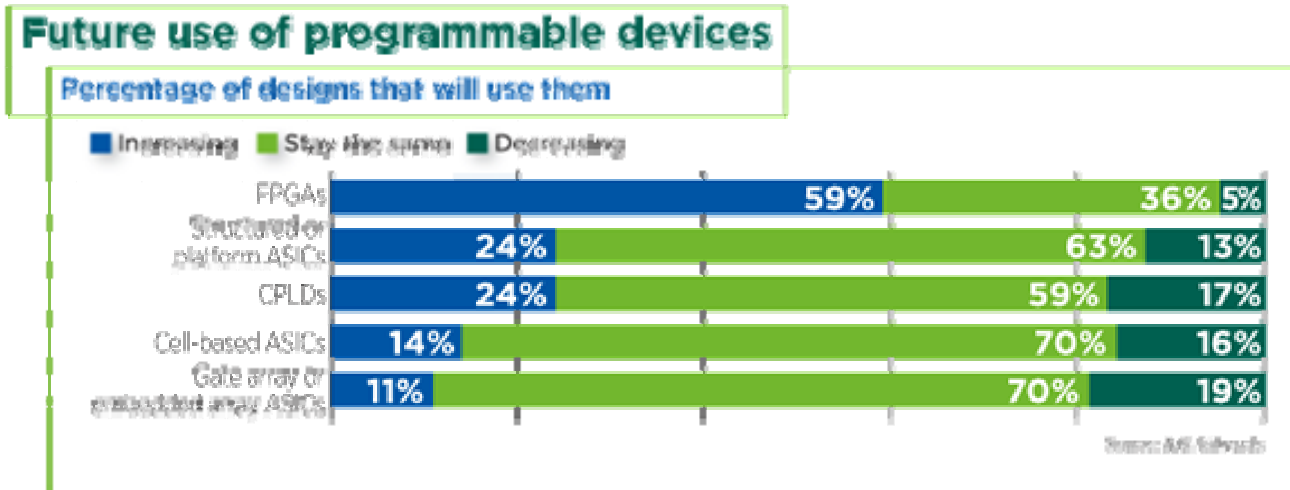
Hilo Único Multihilos

ASICs **FPGAs**

Bus Compartido Bus Punto a Punto

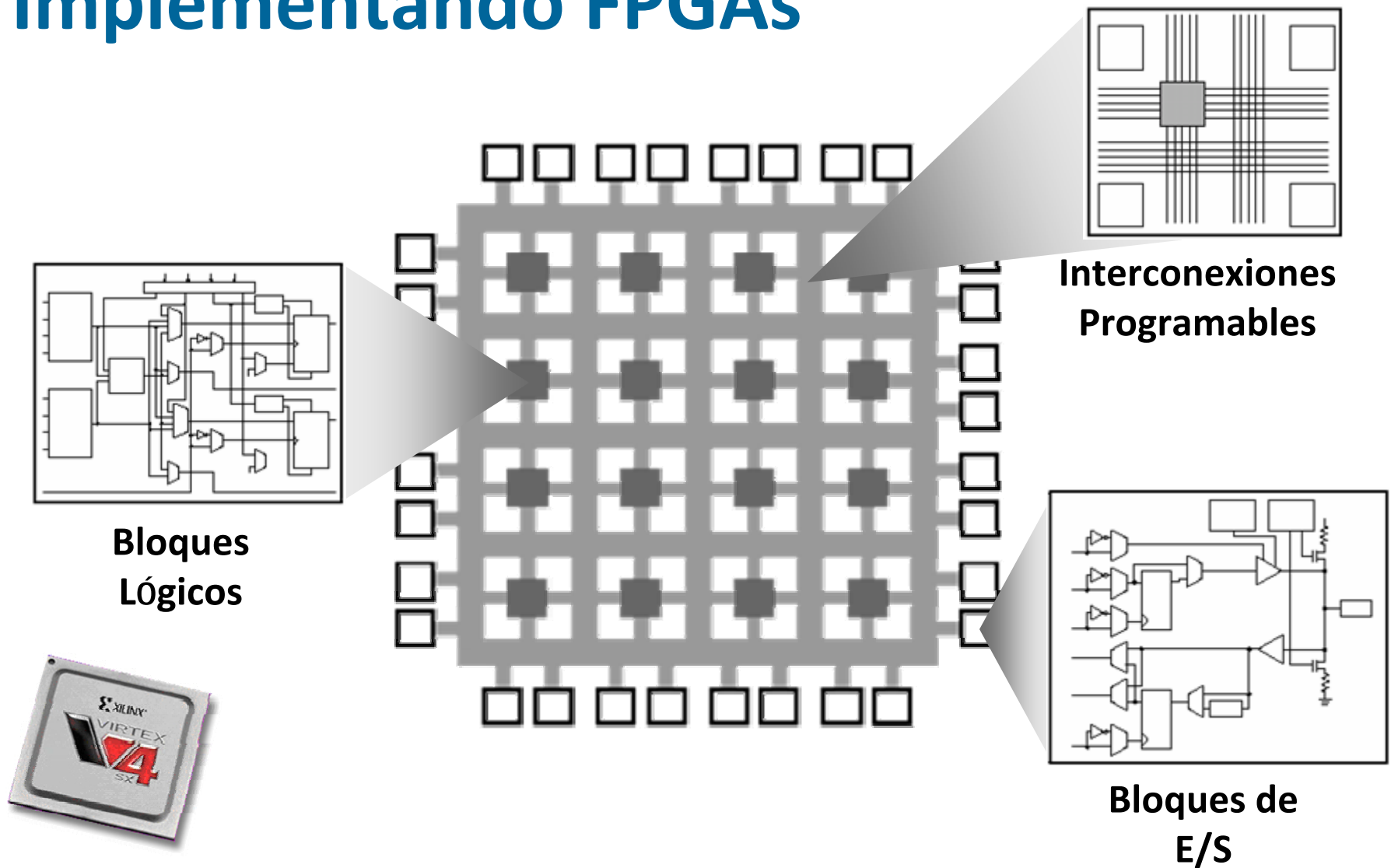
Drivers no Preparado para Drivers Listos para
Ejecución en Hilos Ejecución en Hilos

¿Porqué Cambiar a FPGAs?

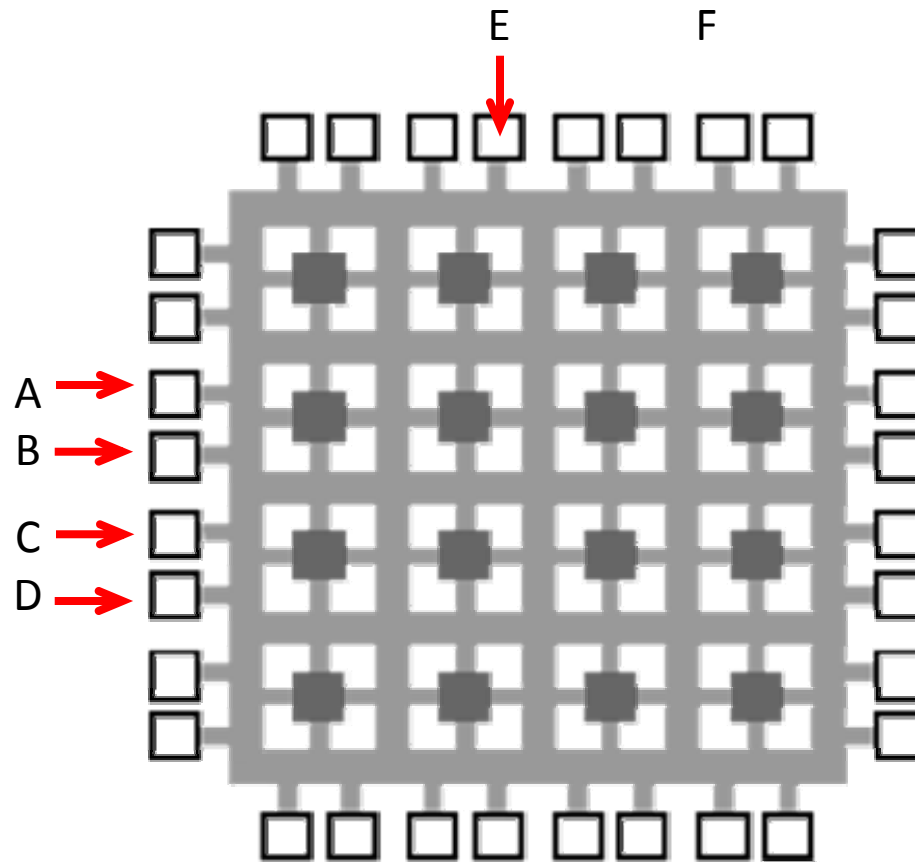


- Reconfigurable
- Verdadero paralelismo
- Alto determinismo
- Operación independiente
- Núcleos funcionales o de IP únicos

Implementando FPGAs

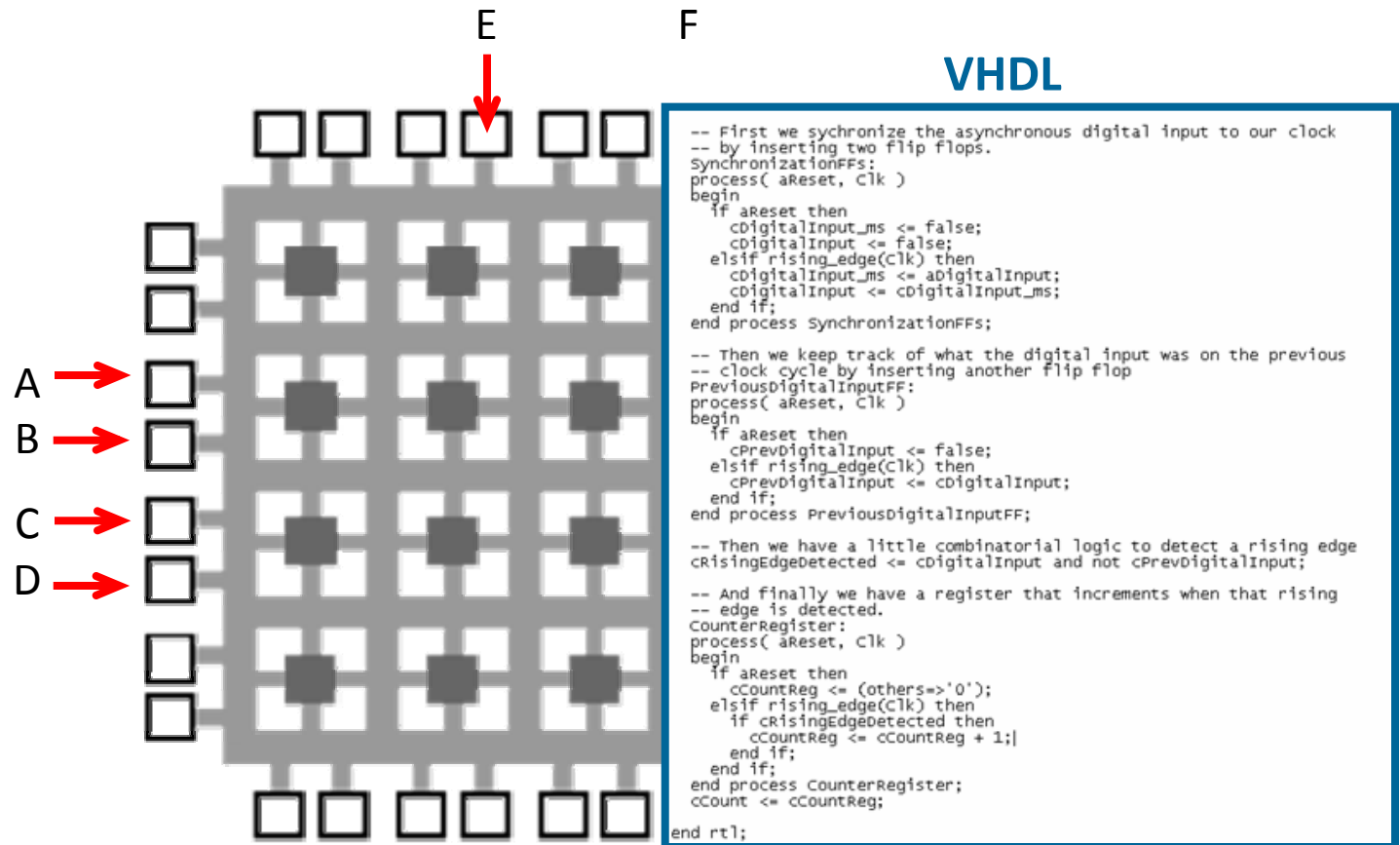


Implementado Lógica en FPGAs



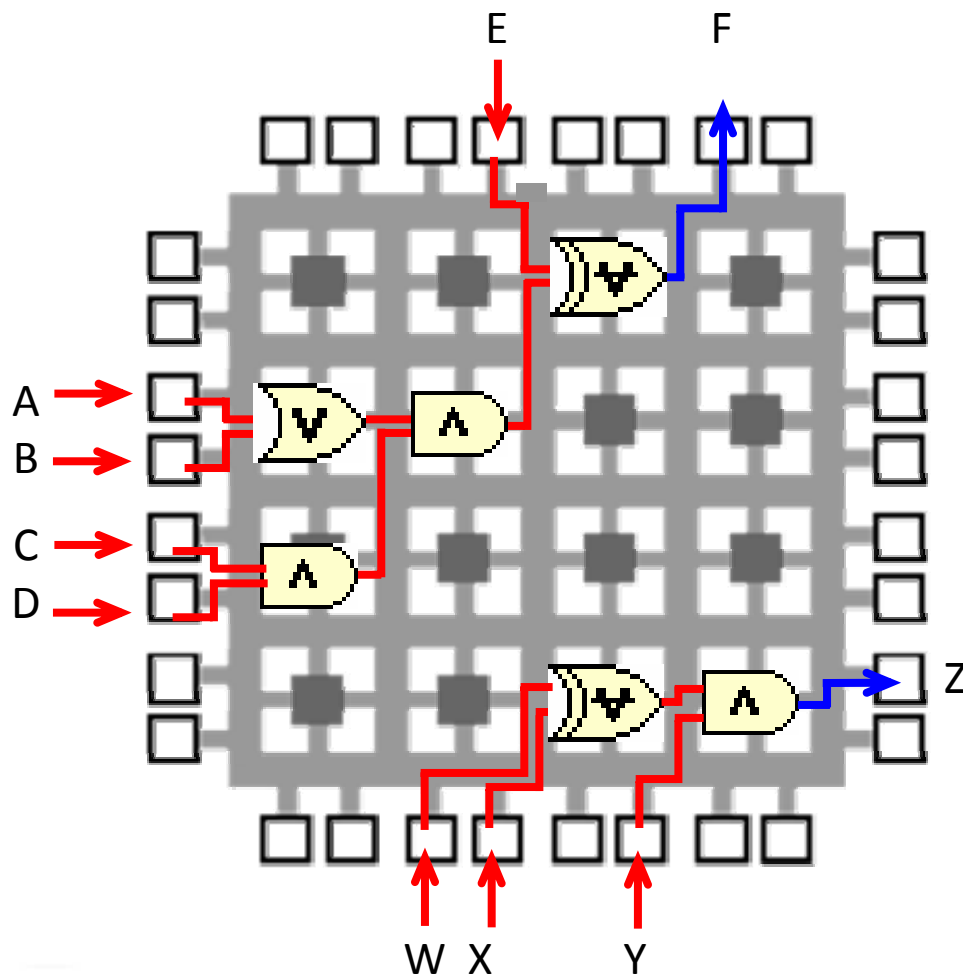
$$F = \{(A+B)CD\} \oplus E$$

Implementado Lógica en FPGAs

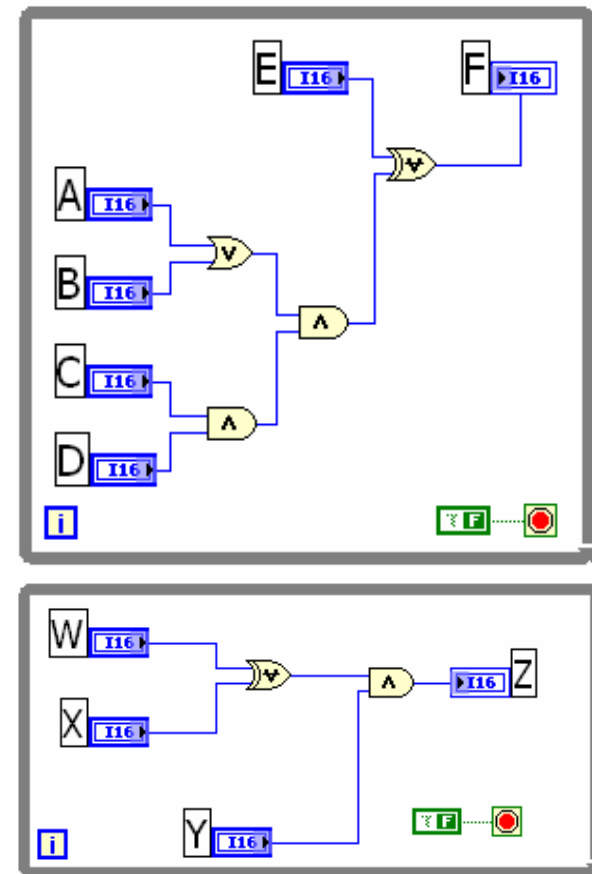


$$F = \{(A+B)CD\} \oplus E$$

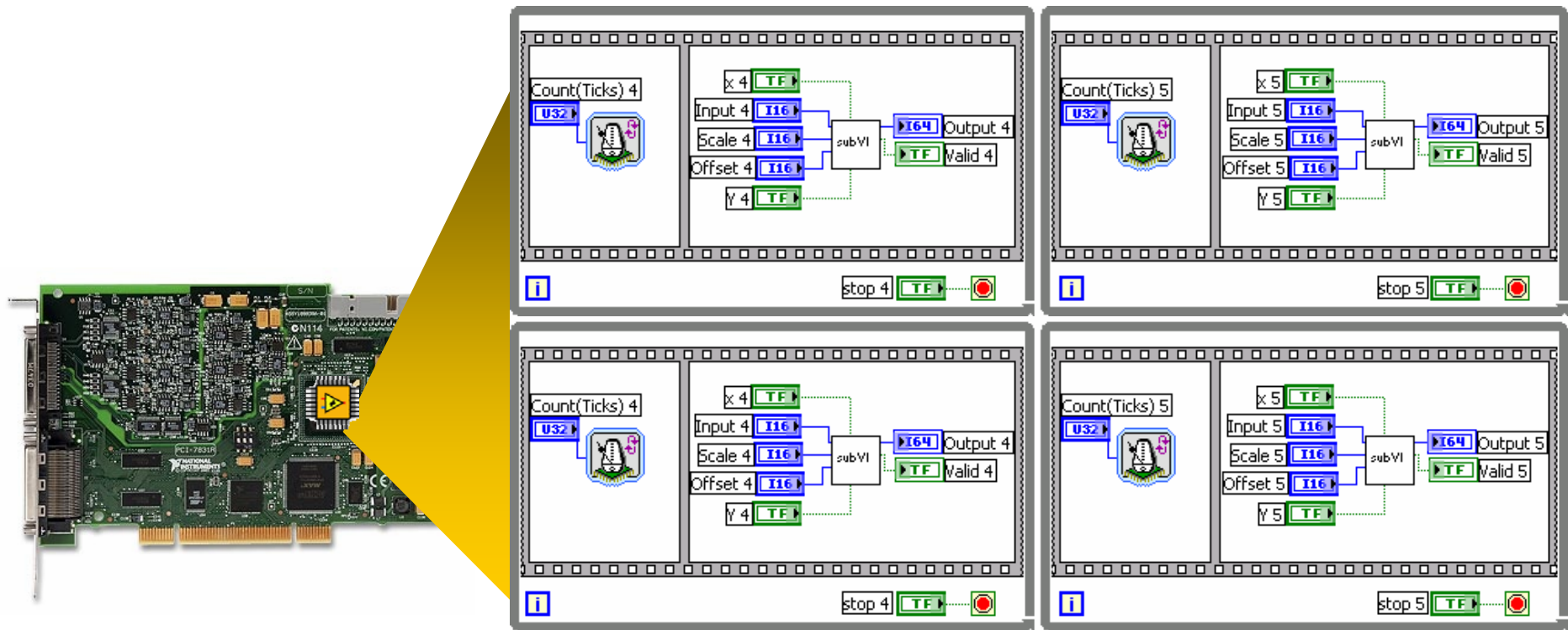
Implementado Lógica en FPGAs



$$F = \{(A+B)CD\} \oplus E$$



Creación Gráfica de Sistemas Paralelos



Surgimiento de Tecnologías Paralelas

Núcleo Único Multinúcleo

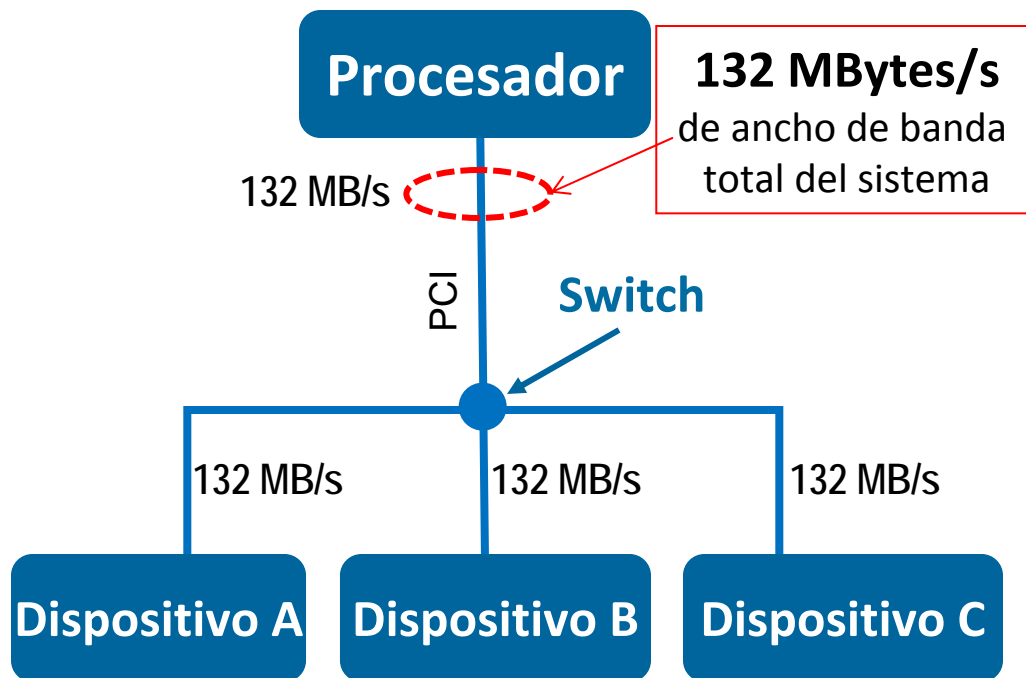
Hilo Único Multihilos

ASICs FPGAs

Bus Compartido Bus Punto a Punto

Drivers no Preparado para Drivers Listos para
Ejecución en Hilos Ejecución en Hilos

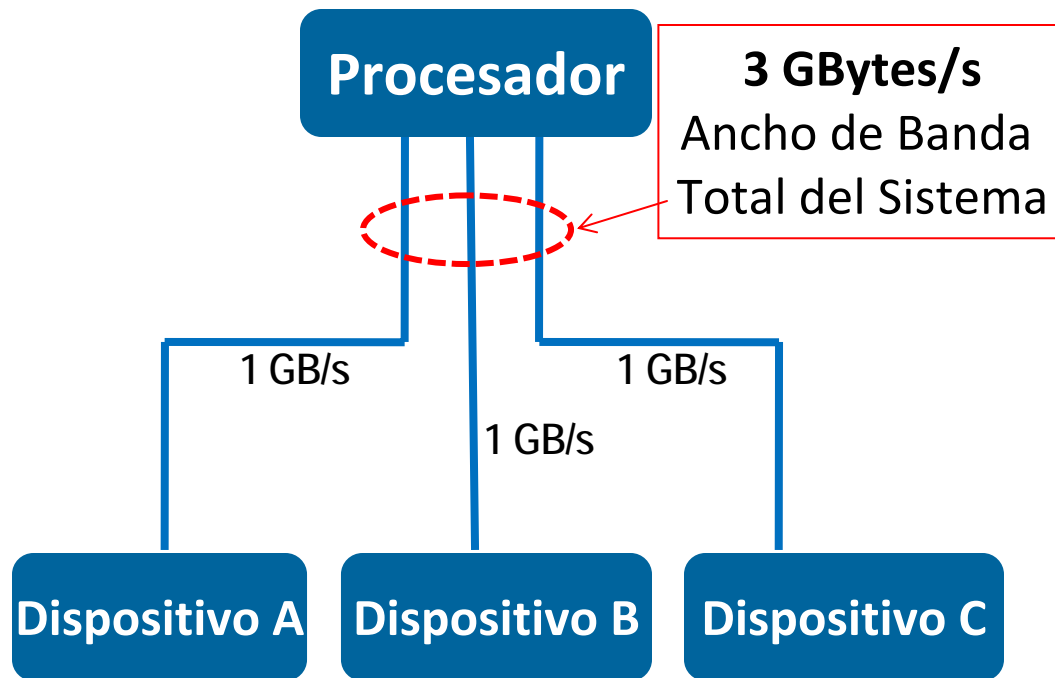
Cuello de Botella en Bus de Datos Compartido



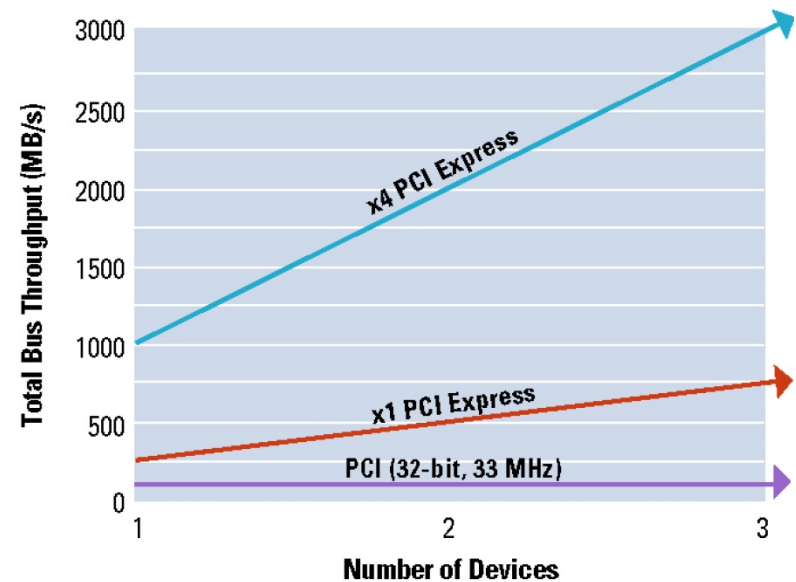
- Ancho de Banda Compartido
- Ejemplos*:
 - LAN
 - USB
 - PCI
 - GPIB
 - PXI

* Controlador de un solo "host"

Topología de Bus Punto a Punto



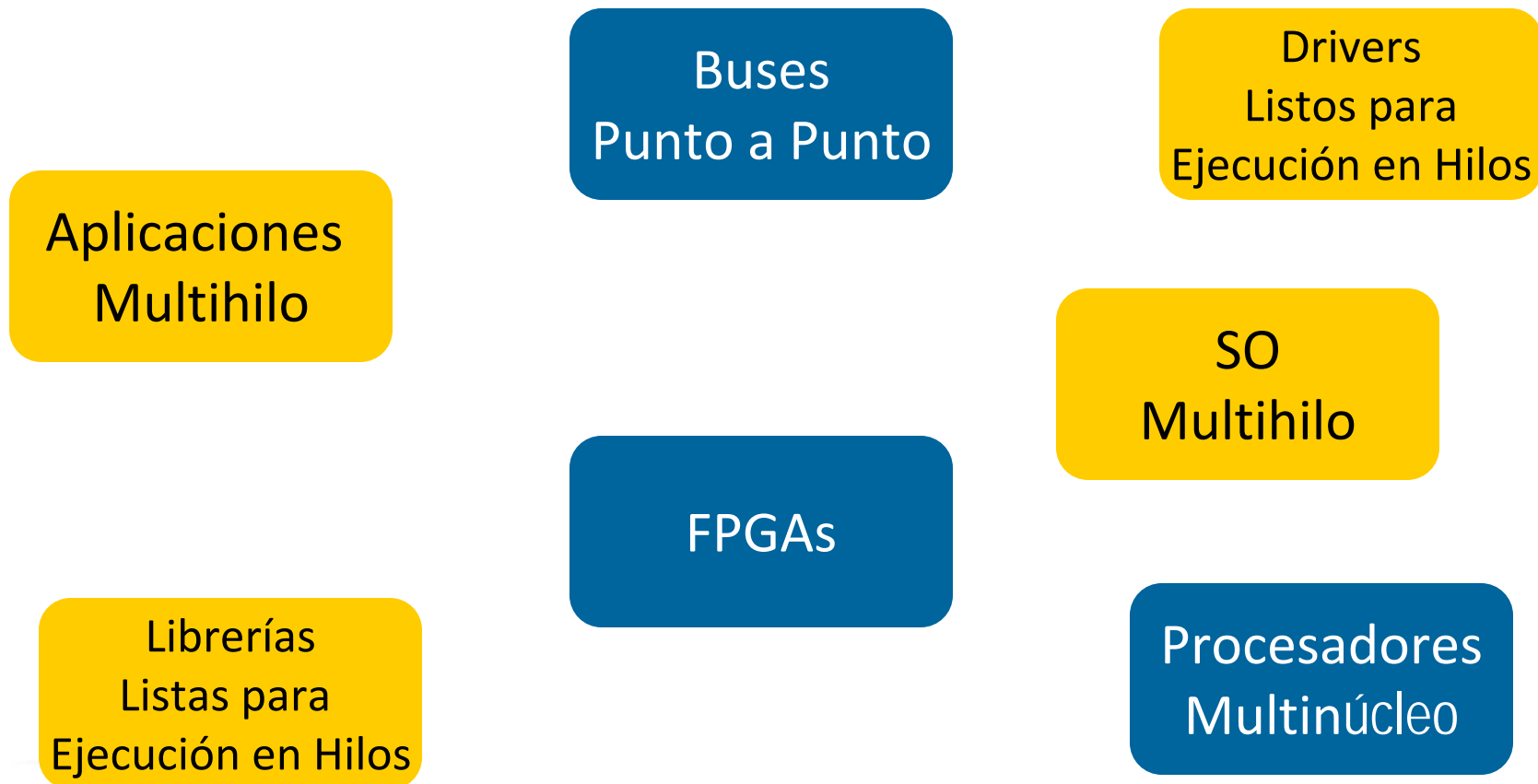
- Ancho de Banda Dedicado
- Ejemplo: PCI Express, PXIe



Soporte del *Stack* de Software Multihilo

Software		
Herramienta de Desarrollo	Soporte proporcionado en el sistema operativo de elección; la herramienta facilita hilos y optimización	✓ Ejemplo: Naturaleza multihilo de LabVIEW y estructuras que proporcionan optimización
Librerías	Ejecución preparada para hilos, librerías reentrantes	✓ Ejemplo: Librerías BLAS
Drivers de dispositivos	Preparados para ejecución en hilos y re-entrantes	✓ Ejemplo: Driver NI-DAQmx
Sistema Operativo	Sistema operativo soporta multihilo y multitarea y puede balancear la carga de tareas	✓ Ejemplo. Soporte para Windows, Mac OS, Linux® OS, y sistemas operativos de tiempo real

Herramientas Clave de la Tecnología Paralela



Plataforma PXI

PXIe-8106
Intel Dual Core

Bus
PXI Express

Serie R
PXI DAQ
3M Gate FPGA

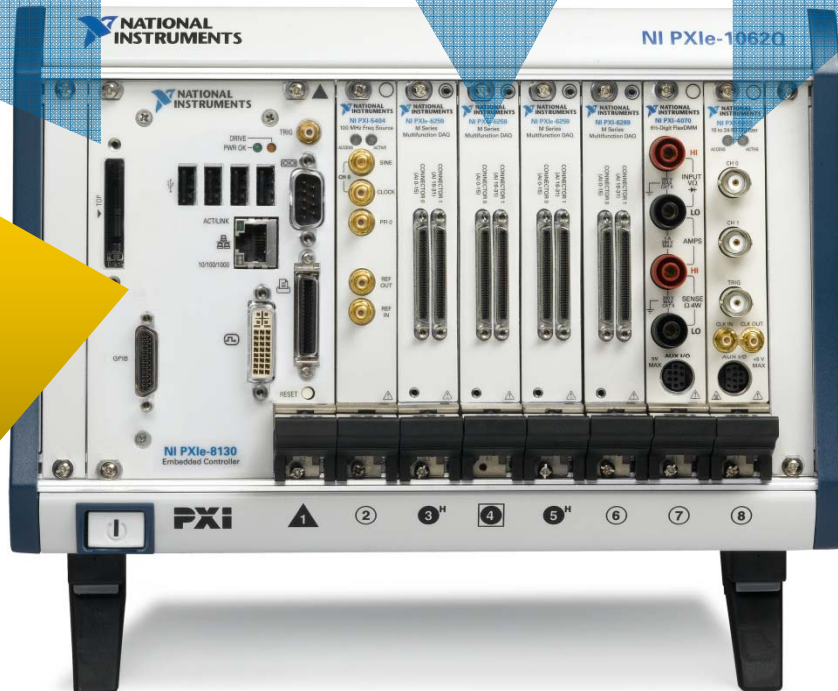
Instrumentos
Modulares

SO Windows o
RT

LabVIEW

Sound & Vibration
Toolkit, Vision

NI-DAQmx,
NI-DMM,
NI-Scope...



ni.com

 **NATIONAL
INSTRUMENTS™**

Demo:

Transferencia de Video con LabVIEW y PXIe

Ejemplo de Aplicación: Pruebas de Comunicaciones

- AmFax Ltd. (Reino Unido)
- Crea sistema de pruebas inalámbricas para teléfonos de la siguiente generación, utilizando la técnica de “pipelining” de LabVIEW

“Con LabVIEW y el controlador embebido de doble núcleo, hemos logrado un ahorro de tiempo de 5 veces mejor que antes ... ”

Mark Jewell
BDM – Wireless
AmFax Ltd.



Ejemplo de Aplicación – Eaton Corporation

Eaton desarrolló un sistema portátil de pruebas en vehículo para transmisiones automotrices utilizando LabVIEW

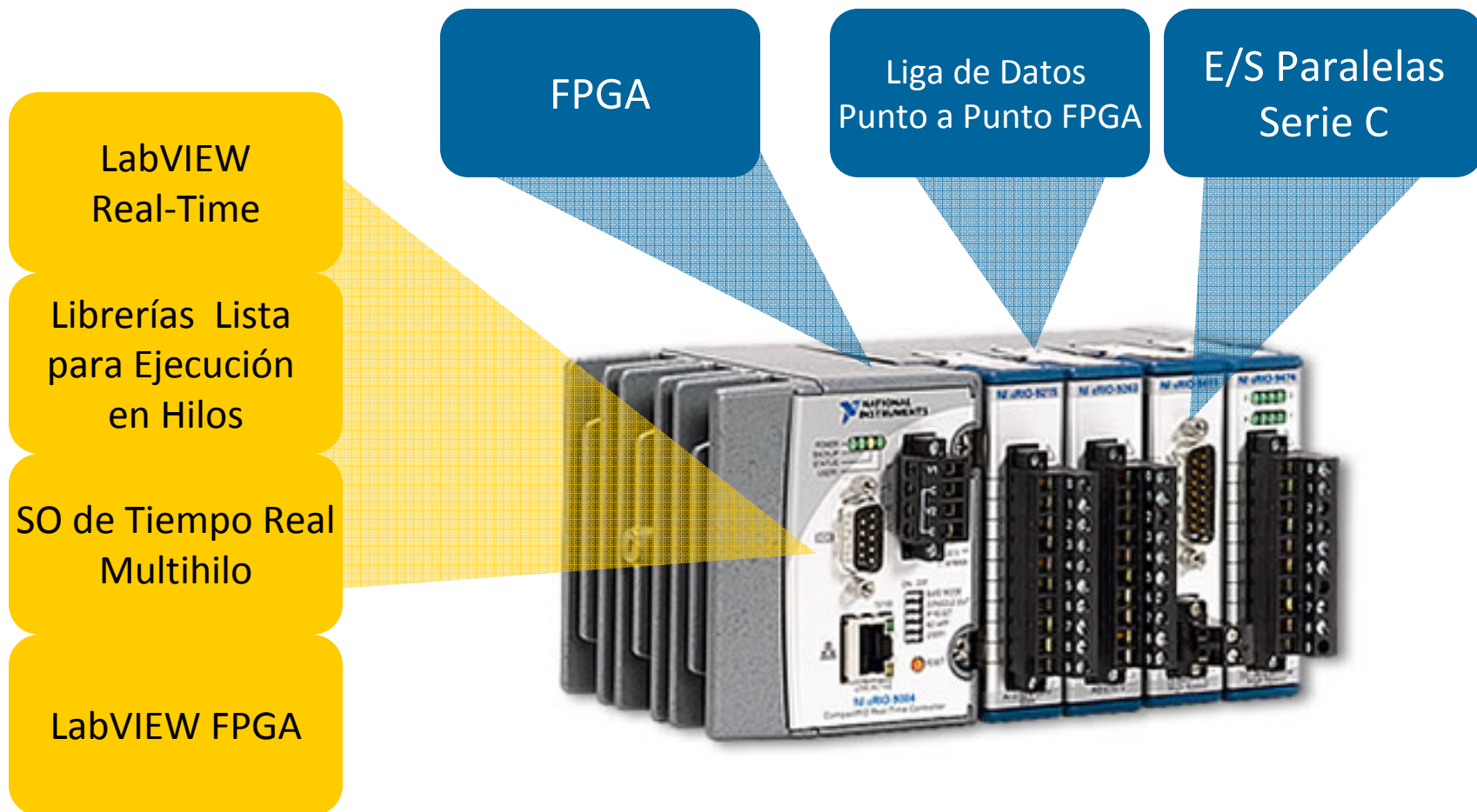
- Adquisición y análisis de 16 canales con un núcleo utilizando el driver multihilo NI-DAQmx
- Ahora adquieren y analizan más de 80 canales en multinúcleo

“No hubo necesidad de reescribir nuestra aplicación para las nuevas plataformas de procesamiento multinúcleo.”

Scott Sirrine
Ingeniero Líder en Diseño
Eaton División Camiones



Plataforma NI CompactRIO

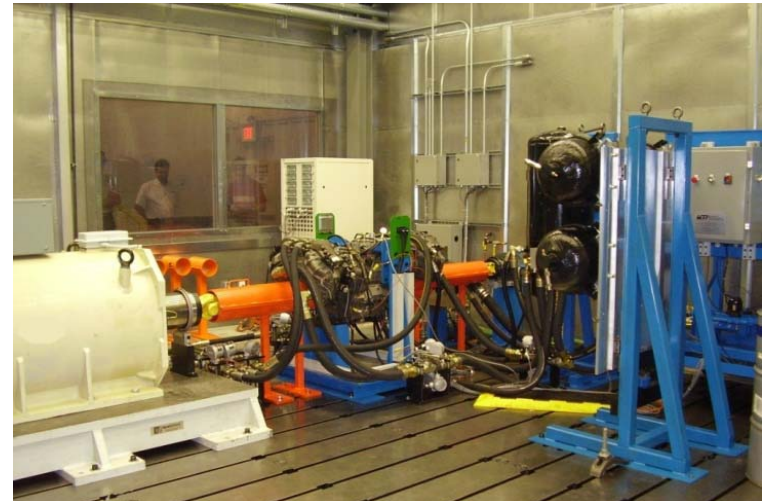


Aplicación Paralela: Prueba de Lazo de Control

*“Con LabVIEW Real-Time 8.5,
incrementamos nuestra velocidad de lazo en un
40% utilizando ambos núcleos del procesador...”*

Wineman Technology crea soluciones
de prueba de lazo cerrado:

- Simulación de “Hardware-in-the-loop”
- Prueba de Dinamómetro

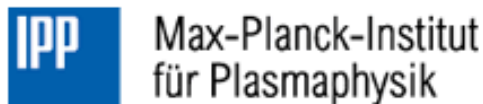


Aplicación Paralela: Control de Proceso

- Instituto Max Planck (Munich, Alemania)
- Control de Plasma en fusión nuclear tokamak con LabVIEW en sistema de ocho núcleos utilizando la técnica de paralelismo de datos.

“...con LabVIEW, obtuvimos un aumento en procesamiento de 20 veces más con una máquina de ocho núcleos sobre una de un solo núcleo...”

Louis Giannone



Sistema Paralelo en Grandes Sistemas Físicos – CERN



Preservando la Inversión del Usuario



Preparese para el Nuevo Mundo Paralelo

Inicio	Fin	Hands-On		
8:30 AM	9:00 AM	Registro		
9:00 AM	9:30 AM	Conferencia de Apertura		
9:30 AM	10:30 AM	¿Qué Hay de Nuevo en LabVIEW 8.5?	¿Qué es LabVIEW?	
10:30 AM	11:00 AM	Break		
11:00 AM	11:45 AM	Desarrollo de Aplicaciones con el Nuevo LabVIEW Statechart Module	Mediciones Físicas y Eléctricas con LabVIEW SignalExpress	HO: ¿Qué Hay de Nuevo en LabVIEW 8.5?
11:45 AM	12:30 PM	USB como bus para Aplicaciones en Ambientes Industriales	Introducción a LabVIEW FPGA y CompactRIO	HO: Introducción a LabVIEW
12:30 PM	2:00 PM	Lunch		
2:00 PM	3:00 PM	Acelere su Proceso de Diseño con Nuevas Herramientas Gráficas y Textuales		
3:00 PM	4:00 PM	Programación en LabVIEW para Ambientes Multinúcleo	Introducción a Mecatrónica: Herramientas para Diseño y Mantenimiento de Maquinaria	HO: Estrategias para Manejo y Administración de Datos con LabVIEW y DIAdem
4:00 PM	4:20 PM	Break		
4:20 PM	5:00 PM	Acceso a Datos desde Cualquier Dispositivo Industrial, Red y PLC con LabVIEW	Sistemas de Visión Embebidos: Nueva Smart Camera	HO: Introducción a LabVIEW
5:00 PM	6:00 PM	Exportando sus Aplicaciones de Windows a un Sistema Real-Time	Mediciones de Alta Velocidad y Procesamiento de Señales para Monitoreo de Condición de Maquinaria	HO: ¿Qué Hay de Nuevo en LabVIEW 8.5?
6:00 PM	6:30 PM	Clausura		