



Performancemaximierung mit der Intel® Multi-Core Architektur

von Jean-Charles Grabiaud, Ardence, Inc.

Intel® Multi-Core Architekturen sorgen für eindeutige Performancevorteile gegenüber Single-Core und sogar Multiprozessor Konfigurationen. Sobald sie mit den deterministischen Windows-Erweiterungen (RTX) von Ardence kombiniert werden, können kritische Applikationen an einen Core gebunden werden um sein volles Potential auszuschöpfen, unbeeinflusst vom Betriebssystem oder anderen Applikationen. Die Vorteile sind eine Optimierung der Leistungsfähigkeit mit verringerten Latenzzeiten, sowie eine verbesserte Zuverlässigkeit.

Herausforderungen

Multiprozessorsysteme gibt es seit einigen Jahren, aber seitdem die Taktfrequenzgrenze durch zu hohe Leistungsaufnahme erreicht wurde, wird die Multi-Core Architektur von Intel immer populärer – und zwar aus gutem Grund. Verglichen mit Einzelprozessor- (UP) oder Multiprozessorchips (MP) liefern Multi-Core Architekturen eine bessere Performance pro Watt, eine höhere Rechenleistung durch Parallelverarbeitung, eine höhere Systemdichte und eine reduzierte Wärmeabstrahlung.

Was ist mit existierenden Programmen?

Parallelverarbeitung ist nichts Neues – die Softwareentwickler arbeiten seit einiger Zeit damit. Aber viele der Firmen, die sich auf embedded Lösungen spezialisiert haben, haben einfach nicht die notwendige Erfahrung um damit umzugehen und die meisten davon haben bereits in die Entwicklung von Programmen, in erster Linie für Einzelprozessorarchitekturen, investiert. Diese Firmen benötigen eine Lösung, die es ihnen erlaubt, mit den bestehenden Programmen und einem minimalen Portierungsaufwand die Ressourcen der Multi-Core Systeme bestmöglich auszunutzen

“Power is nothing without control”

Determinismus ist noch immer eine große Herausforderung in embedded Lösungen und, obwohl verbesserte Performance den Durchsatz der Prozesse und die durchschnittlichen Reaktionszeiten dramatisch verbessern kann, wird dadurch aus einem nichtdeterministischen System noch lange kein deterministisches. Ja nicht einmal die worst-case Reaktionszeiten werden dadurch immer verbessert. Einige Firmen werden ihre Anforderungen als weiches Echtzeitverhalten beschreiben, andere als hartes Echtzeitverhalten. Technisch gesehen ist es entweder Echtzeitverhalten oder eben nicht.

Das Softwaredesign ist ein Knackpunkt bei der Umsetzung von Performance auf Multi-Core Systemen und Überlegungen bezüglich Software fangen beim Betriebssystem an.

Die Notwendigkeit von "kontrolliertem" Windows

Die Popularität und der Marktanteil der Microsoft® Windows® Betriebssysteme (OS) in industriellen Umgebungen wuchsen in den letzten Jahren dramatisch. Gründe dafür sind:

- Der Industriestandard HMI (**H**uman **M**achine **I**nterface) und die große Vielzahl an verfügbaren Funktionalitäten.
- Die ohne weiteres verfügbaren, erfahrenen Entwickler und die Menge der existierenden Applikationen.
- Der zuverlässige Support und die weit reichende Roadmap von Microsoft.

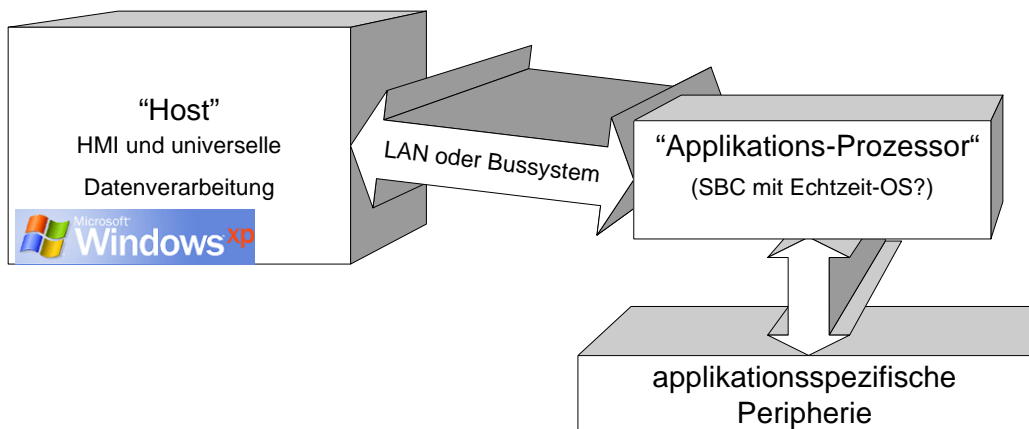
Wegen zunehmender Komplexität und zunehmenden Wartungskosten für eine heterogene Computerumgebung sind die Firmen bestrebt, Windows als Betriebssystem auf allen Ebenen Ihrer Organisation einzusetzen. Der Einsatz in einem Netzwerksystem oder einem Arbeitsplatzrechner ist dabei einfach verständlich, da dies genau die Umgebungen sind, für die Windows entworfen wurde. Es gibt allerdings auch eine Motivation es in anderen Umgebungen einzusetzen, wie dem Installationskanal, medizinischen Geräten und Geräten für Simulation, Test und Kommunikation. Ein gemeinsames Merkmal dieser Umgebungen ist, dass sie oft hartes Echtzeitverhalten erfordern.

Kann Windows diesen Bedarf erfüllen? Die Antwort ist... **nicht von der Stange.**

Microsoft Windows wurde als universelles Betriebssystem entworfen. Die Einschränkungen für Echtzeitanwendungen wurden gründlich ermittelt (zu wenig Threads und Prioritäten, teilweise unverständliches und nichtdeterministisches Scheduling, Prioritätsumkehr...) und führen uns zu folgenden Beobachtungen:

- Es bietet nicht genügend Kontrollmöglichkeiten, um Zielapplikationen soweit wie nötig zu priorisieren.
- Kritische Applikationen laufen nicht weiter, wenn ein fataler Ausnahmezustand auftritt.
- Entwickler sind daran gewöhnt, einen einfachen Zugriff auf die Hardware zu haben, und das Treibermodell von Windows ist dafür zu kompliziert.
- Windows wurde nicht zum Managen harter Echtzeitanforderungen entworfen.

Die "zeitwürdige" Lösung...



Traditionell würden Entwickler ein zweites, echtzeitfähiges Gerät in ihr System einbauen. Aber die offensichtlichen Nachteile sind höhere Hardwarekosten, eine begrenzte Integration und höhere Kosten und Anforderungen an die Entwicklungstools bezüglich der Unterscheidung zwischen Host- und Applikationsprozessor.

Bedenkt man die aktuell durch die Intel Multi-Core Architekturen verfügbare Rechenleistung, so ist es unglücklich, dass Windows sie im embedded Bereich nicht effektiv nutzt.

Die Forderung

Für embedded Märkte besteht die Forderung nach einer kostengünstigen Alternative zu kundenspezifischer Hardware (PC basiert, Standard-Betriebssystem) und nach den Annehmlichkeiten von Windows, aber mit Echtzeitattributen.

Ardence RTX Lösung

Windows XP ist ein Massenprodukt – nicht einfach für Nischenapplikationen anpassbar, wie z.B. Echtzeitanwendungen. Der korrekte Weg Windows XP echtzeitfähig zu machen, ist mittels einer Erweiterung oder eines Plug-Ins zum ursprünglichen Produkt. Ardence RTX ist diese Echtzeiterweiterung.

RTX besteht aus einer Zusammenstellung von Bibliotheken (statisch und dynamisch), einem Echtzeit-Subsystem (RTSS), das als Windows XP Gerätetreiber im Kernel implementiert wurde und einer erweiterten Hardware Abstraktionsschicht (HAL - siehe nächste Grafik). Das RTSS implementiert die zuvor erwähnten Echtzeitobjekte und den Scheduler. Die Bibliotheken erlauben den Zugriff auf das Subsystem über eine Echtzeit-API, der so genannten RtWinAPI. Die Funktionen der RtWinAPI können dabei in der normalen Win32-Umgebung, wie auch im RTSS-Kontext aufgerufen werden. Wenn die RtWinAPI aus Win32 genutzt wird, liefert sie nicht den im RTSS verfügbaren Determinismus, aber es ermöglicht einen Großteil der Applikationsentwicklung in der benutzerfreundlicheren Win32 Programmierumgebung durchzuführen, anstelle in der durch das DDK bereitgestellten. Um aus einem Win32 Programm ein RTSS Programm zu machen, ist es nur notwendig es mit anderen Bibliotheken neu zu linken. Der Windows XP Service Control Manager lädt RTX Prozesse und ausführbare DLLs in den nicht auslagerbaren Speicher des Kernelbereichs - wie RTX selbst.

Die Schlüsselbausteine von RTX sind die HAL-Erweiterung, der RTSS-Scheduler, der IPC-Mechanismus (Interprozesskommunikation), die hochgenauen Takte und die verfügbaren Entwicklungswerkzeuge.

Der HAL (Hardware Abstraction Layer) ist ein Teil des Microsoft Windows XP Systems, für den der Source Code zur Modifikation und Erweiterung verfügbar ist. RTX hat den HAL aus 3 Gründen abgeändert:

- Um eine Trennung der Interrupts zwischen Windows XP und RTSS Threads zu ermöglichen.
- Um hochgenaue Takte und Timer zu implementieren.
- Um Routinen zum geregelten Herunterfahren (Shutdown) des Systems zu implementieren.

Ein wichtiges Architekturmerkmal von RTX ist seine sperrenfreie, interruptgesteuerte Schnittstelle zwischen Windows XP und RTSS mit der ein Local Procedure Call (LPC) Mechanismus implementiert wurde. Diese klare architektonische Trennung erlaubt Anpassungen von RTSS an unterschiedliche Umgebungen (z.B. das Multiprozessor RTX Produkt) und stellt dabei eine schnelle und robuste Implementierung sicher.

Mögliche *Taktperioden von RTX* sind 100, 200, 500 und 1000 Mikrosekunden. Da die präzise Ausführung von Ereignissen für ein Echtzeitsystem entscheidend ist, liefert RTX drei Takte auf denen Event Timer basieren. Die Zeitauflösung der Takte ist bis auf 0,001 Nanosekunden genau, ohne irgendeine Verschiebung (Drift), abhängig vom genutzten Takt.

Microsoft Visual Studio IDE wird voll unterstützt. Durch das mitgelieferte, umfassende Paket an Werkzeugen (Tools), die sich darin glatt einfügen, können Software-Entwickler die benötigte Zeit für Entwicklung und Fehlersuche signifikant verkürzen.

Mit diesen Werkzeugen kann man die realisierte Applikation in Echtzeit betrachten und damit die Interaktionen zwischen Hardware, RTX und RTX Applikation verstehen und das Verhalten der Applikation einfach debuggen und analysieren.

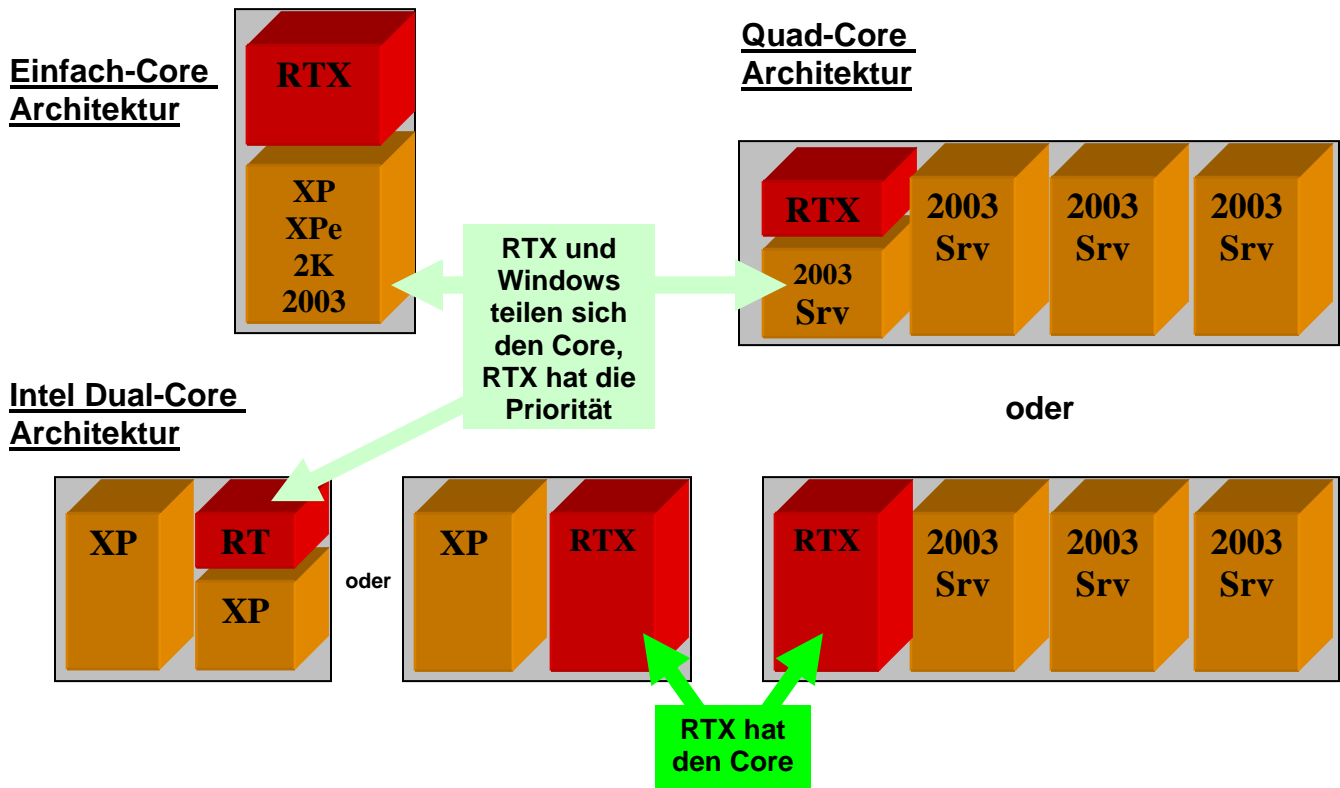
Ardence RTX für Mehrprozessor und Intel Multi-Core Systeme

Vor 1999 lief RTX nur auf Einprozessorsystemen. Aber seit über 7 Jahren, seit diese eingeführt wurden, laufen RTX Versionen auf Multiprozessor und Multi-Core Systemen.

Durch die Intel Multiprozessor Spezifikationen können Interrupts durch einen Advanced Programmable Interrupt Controller (APIC) kontrolliert werden, der in ein Mehrprozessorsystem passt. Durch diesen APIC können Interrupts an verschiedene Gruppen von Prozessoren geleitet werden.

In einem Mehrprozessorsystem (MP) kann RTX auf folgende Arten konfiguriert werden:

- **Shared Mode** – Ein Prozessor bearbeitet RTX und Windows Prozesse; alle anderen Prozessoren sind für Windows Prozesse vorbehalten.
- **Dedicated Mode** – Ein Prozessor ist für RTSS Threads vorbehalten (dediziert), während die anderen Prozessoren Microsoft Windows XP Threads abarbeiten. Dies verringert die Latenzzeit von Echtzeit-Threads dramatisch und verhindert ein "Verhungern" von Windows XP Threads, das auf einem Einprozessorsystem möglich ist.



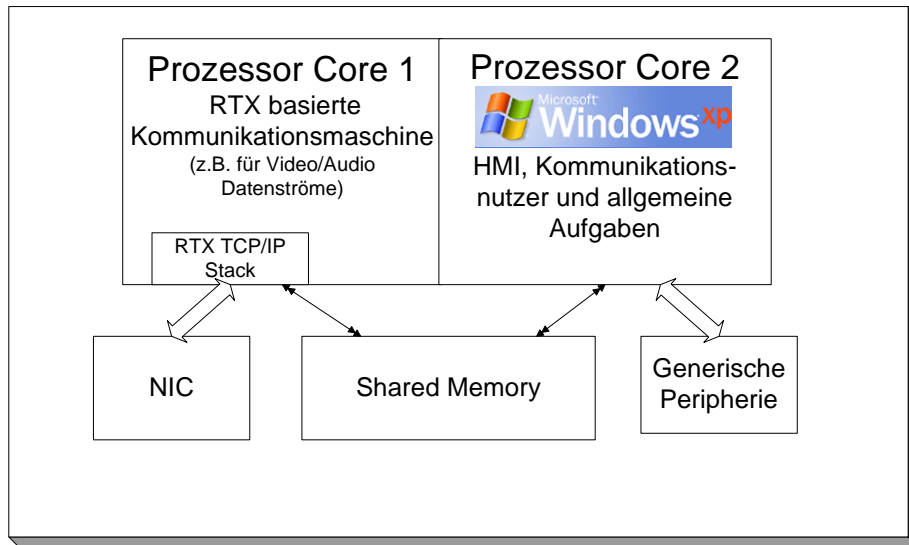
Das optimale Design...

- Ein Intel Multi-Core Prozessor PLUS
- Ardence RTX, das im "Dedicated Mode" läuft:
 - Ein Core ist für Ihre Applikation vorbehalten. Windows behält den Rest.
 - Alle Prozesse Ihrer Applikation haben Vorrang vor Windows Prozessen durch Nutzung eines eigenen Schedulers.
 - Ihre Applikation läuft weiter, auch wenn unter Windows eine fatale Ausnahme (Exception) auftritt.
 - Sie können Windows Multithreading zusammen mit RTX Multithreading weiternutzen.
 - Windows und RTX Prozesse können über gemeinsame Speicherbereiche (Shared Memory) kommunizieren.
 - Ihre Applikation kann auf Hardware zugreifen ohne über das Windows Treibermodell zu gehen.
 - Sie können mit Visual Studio für Windows und RTX entwickeln.
 - RTX liefert "harte" Echtzeit, falls erforderlich.

Typische Applikationen für den Ardence RTX "Dedicated Core" Modus

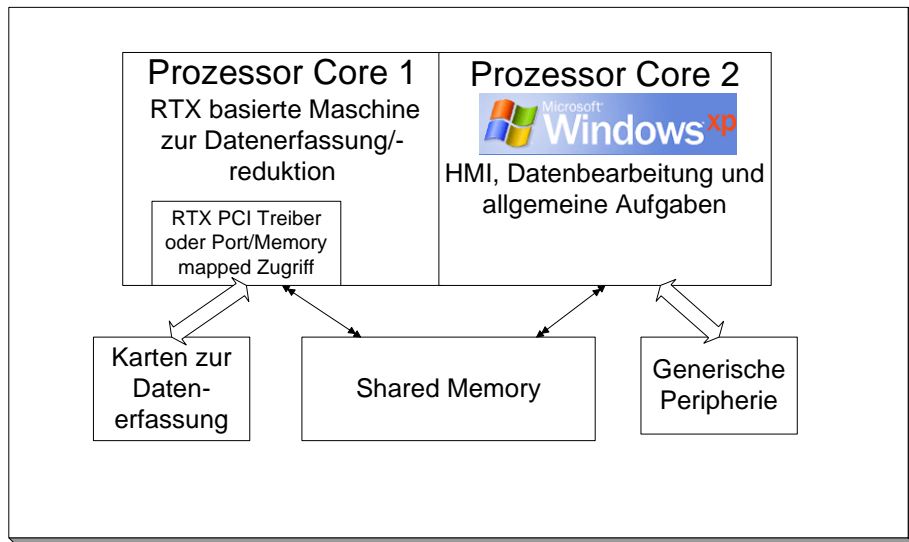
Dedizierte Kommunikationsmaschine

RTX kann alle Video/Audio Datenströme durch einen eigenen Stack auf einem einzelnen, dedizierten Core abarbeiten, während Windows die verbleibenden Aufgaben auf seinem eigenen Core bearbeitet. Die Kommunikation zwischen den Beiden ist durch den von RTX bereitgestellten IPC Mechanismus effizient gelöst.



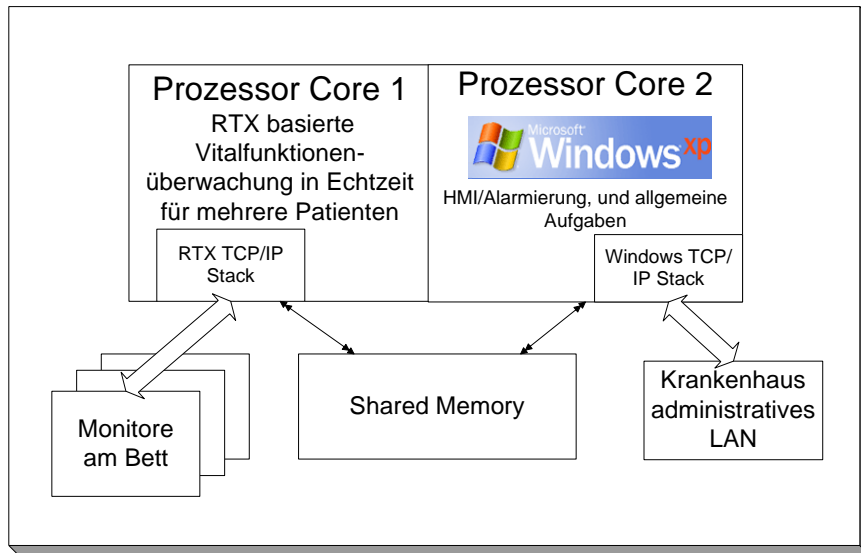
Maschine zur Datenerfassung/-reduktion

Im untenstehenden Beispiel, das in der industriellen Automatisierung sehr üblich ist, wird die Erfassung kritischer Daten und die Kontrolle von abgesetzten Geräten komplett von RTX übernommen, das die volle Leistung eines Cores nutzt, damit Latenzzeiten minimiert und sogar eine komplexere Bearbeitung der erfassten Daten ermöglicht. Windows, auf den anderen Cores, wird diese Tasks nicht beeinträchtigen, wird aber auch nicht durch sie "ausgehungert".



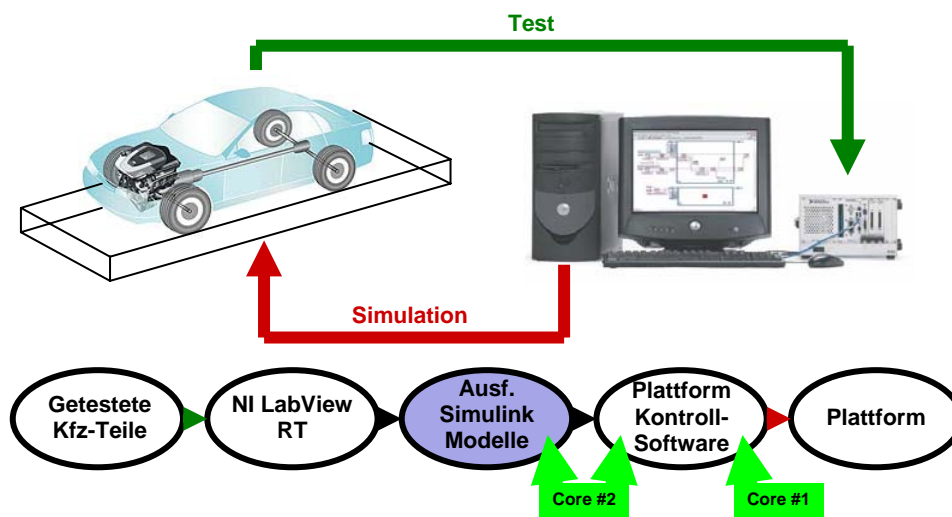
Überwachung der vitalen Funktionen für mehrere Patienten

Medizinische Applikationen sind ebenfalls vielfältig, aber dieses Beispiel hat den Schwerpunkt auf einem Überwachungssystem, in dem ein dediziertes Netzwerk komplett durch RTX bearbeitet wird, über eine eigene, getrennte Netzwerkschnittstelle. Die Daten sind natürlich auch für Windows verfügbar und jede Applikation im Krankenhausnetz kann sie sich holen und wunschgemäß nutzen, ohne die Überwachung an sich zu beeinträchtigen.



Automotive Simulator: Test und Simulation auf einem System

Diese Anforderung wird in der Automobilindustrie immer beliebter, bei der sich Systemlieferanten dadurch unterscheiden können müssen, dass sie Test und Echtzeitsimulation auf einem einzigen System anbieten und damit die Gesamtkosten der Lösung reduzieren. MathWorks Produkte werden in diesem Umfeld oft genutzt und RTX kann die sehr komplexen Simulationsmodelle im "Dedicated Mode" bearbeiten und damit zusätzliche Hardware- und Integrationskosten eines zweiten DSP-basierten Geräts wegrationalisieren.





Über Ardence

Ardence entwickelt und vertreibt Softwareplattformen für die "On-Demand" Welt. Unsere Software-Streaming Plattform ermöglicht es der IT durch die Bereitstellung einer virtuellen IT-Infrastruktur flexibler zu werden. Unsere Embedded OEM Entwicklungsplattformen ermöglichen es OEMs die Kontroll- und Echtzeit-Technologien von Ardence – und unsere Embedded Windows Erfahrung – zu nutzen, um Systeme mit höherer Leistungsfähigkeit und Managebarkeit zu entwickeln.

Ardence wurde 1980 gegründet, hat die Zentrale in Waltham, Massachusetts und operiert vorwiegend in Nordamerika, Europa und Asien. Ardence ist General Member der Intel® Communications Alliance. Mehr Informationen über Ardence, Inc. unter www.ardence.com.



Gold-level Member



NORTH AMERICA

266 2nd Avenue
Waltham, MA 02451
Toll-free: (800) 334-8649
Fax: (781) 647-3999
E-mail:
EmbeddedSales@Ardence.com
Web: www.Ardence.com

EUROPE

ABS – Porte de l'Arenas, Hall C
455 Promenade des Anglais
06299 Nice Cedex 3- France
Tel.: + 33 (0)4 89 06 60 10
Fax: + 33 (0)4 89 06 60 20
E-mail: fboisset@Ardence.com