對稱多處理或虛擬化

最大化軟體控制架構的價值與能力

IntervalZero

概要

多核心技術的持續進步是軟體創新的催化劑，這種創新孕育了新一代嵌入式系統。這些系統因為能並列運行多項任務，所以開發成本更低，性能更高，還具有擴充性。

這些新系統的理想架構在先前的一份白皮書「軟控制架構：複雜系統硬即時設計的突破」中有所描述。本篇白皮書已假設讀者具備基本的了解，對於軟體控制架構如何取代 FPGA/DSP/PowerPC，並透過一個微軟 Windows 的硬即時外掛程式—專用 RTOS 解決方案—來提供突破性的成本、效能和可擴充效益。

系統開發人員已開始利用軟體控制架構在多樣化的市場像是工業自動化、醫療系統、檢測，以及數位媒體中獲得持續的競爭優勢。

本篇白皮書探討了兩種普遍且競相提供軟體控制架構價值的多處理方法。分別是對稱多處理（SMP）和非對稱多處理（ASMP），後者常被稱為虛擬化或虛擬機監控（hypervisor）技術。

這兩種方法各有其特點和挑戰，但對於那些試圖最大化擴充性並透過充分利用 x86 多核心的力量（4 核、6 核、8 核及以上）來最小化延遲的開發者來說，SMP 顯然更具優勢。

我們將會看到，虛擬化/ASMP 確實在初期成本降低方面提供了短期成效，但對於需要擴充性、精準度和效能，且要求最低延遲的真正即時系統來說，卻是一種死胡同的技術。

目前虛擬化/ASMP 雖然在單晶片上是主流技術，但這種鏡像式的現狀極為局限，且缺乏靈活性、抑制效能，當開發人員想跨越雙核心時也阻礙了可擴充性。

由於支援 SMP 的架構是動態的—與虛擬化/ASMP 的靜態架構不同—因此能為開發人員提供更廣泛的選擇以簡化和精簡開發流程，同時充分利用多核心處理能力，開發出能夠改變競爭格局的系統。

這對於那些開發具有複雜人機介面（HMI）和嚴苛硬即時與控制要求的嵌入式系統開發人員來說尤其如此。大型醫療系統如核磁共振（MRI）和數位媒體混音控台，就是具有這些需求的例子。

如上所述，對於軟體控制架構的詳細描述可參考早期的白皮書，但為了提供相關背景資料給本篇文章的比較資訊，我們在此涵蓋了該白皮書的重點概要。

長期以來，FPGA 和 DSP 在運動控制和其他複雜、高精確度和高效能系統的硬即時市場上佔有主導地位。但現在情況已經不同了，隨著科技進步，OEM 廠商能夠部署軟體控制架構，從而取代大部分的專用硬體。

長期以來，FPGA 和 DSP 在運動控制和其他複雜、高精確度和高效能系統的硬即時市場上佔有主導地位，但現在情況已經不同了。

有利於採用支援 SMP 軟體控制架構的主要趨勢包括:

- 日益強大的 x86 處理器技術；
- 朝向更多商用現成（COTS）硬體和軟體的發展趨勢；
- 以太網總線的進步及可用性；
- 系統設計中組件的融合；以及
- 以觸控為中心的可用性—特別是多點觸控—和運動感測技術。

利用這些趨勢所形成的軟體控制架構採用了多核 x86，可以在單個多核心的晶片上同時運行 Windows（包括帶有觸控和手勢技術的 Windows 7）和支援對稱多處理的硬即時外掛程式例如英特蒙的 RTX64。

打造軟體控制架構可使用不同的技術，但為了實現最大價值，開發人員必須牢記八個關鍵的成功特徵：

1. 通用的整合開發環境（IDE）和世界級的圖形使用者介面（GUI）—Microsoft Visual Studio 和 Windows 作業系統，包括 Windows 10 和 Windows Embedded Standard 7；

2. 直接在多個指定處理器（不是多個個體）執行能支援 SMP 的即時子系統

3. 所有即時行程都能監看硬體狀態；

4. 能夠在多個處理器上排程即時執行緒，或將某些邏輯專用於特定核心並利用 hook API 達到負載平衡；

5. 直接存取共享資料/記憶體，無需額外副本和行程間通訊；

6. 硬體需求最小化—處理器、記憶體、功率和佔用空間；

7. 能夠跨核心進行除錯；

8. 一次編碼和自動擴充的能力。

在 SMP 能夠完全滿足這八項成功要素的同時，虛擬化/ASMP 只能部分實現一個特性—透過將原本需要兩台 PC 的系統整合為單一 PC 以最小化硬體需求（第 6 點）。

虛擬化/ASMP 確實讓許多不同的作業系統/應用程式存在於同一個多核硬體上，並共享周邊資源如 I/O、序列埠、以太網、USB 等，也因此似乎是一種降低系統成本和複雜度的好方法。

過去在不同 PC 系統上運行的應用程式，現在已經能整合到單一 PC 中的一塊矽晶片上，這樣的方式能降低物料成本、保留現有編碼、也不需要重寫和重新設計架構。然而，這正是事情陷入死胡同的地方，虛擬化/ASMP 架構的靜態本質限制了自身的擴充性，因此無法成為可行的長期解決方案。

例如，在虛擬化/ASMP 中，多處理器配置需要的所有元素—獨立的即時作業系統（RTOS）、獨立的程式庫、獨立的開發工具和獨立的開發團隊—都會鏡像到單一系統上，客戶管理的個體並沒有減少，反而是增加了一個用來管理的 hypervisor。

簡而言之，雖然虛擬化/ASMP 在硬體成本降低上有所收穫，但並沒有相應的軟體突破—像 SMP 那樣—能讓開發人員在多核上垂直擴充並打造在市場上具備差異化的系統。

例如，隨著嵌入式開發人員追求利用越來越多的核心，透過將較大的功能分散到不同核心以提高效能；或將核心分散為相關功能；或指派核心進行特定的計算和數學功能—這些都能透過 SMP 達成—而虛擬化/ASMP 的鏡像架構則會帶來重大挑戰。

虛擬化/ ASMP 目前雖然在單晶片上是主流技術，但這種鏡像式的現狀極為局限，且缺乏靈活性、抑制效能，當開發人員跨越雙核心時也阻礙了可擴充性。

讓我們再深入探討其他七種成功特徵，就能看出 SMP 的相對優勢，和虛擬化/AMSP 在提供硬即時軟體控制架構方面的弱點。

1. 通用的整合開發環境（IDE）和世界級的圖形使用者介面（GUI）— Microsoft Visual Studio 和 Windows 作業系統，包括 Windows 10 和 Windows Embedded Standard 7

雖然虛擬化/ASMP 技術可能提供，也可能不提供一個通用的開發環境，但虛擬化並不會搭配一個整合環境。每個作業系統的個體都必須有自己的代碼，並對在其他核心上運行的隔離環境中的代碼一無所知。虛擬化的缺乏整合性會導致管理成本急劇上升。

2. 直接在多個指定處理器（不是多個個體）執行能支援 SMP 的即時子系統

由支援 SMP 的即時外掛程式（如英特蒙的 RTX64）提供的輔助排程器，能夠清楚地檢視所有處理器，這確保了核心之間的正確使用和同步。

SMP 架構透過擁有另一個輔助排程器—除了 Windows 排程器之外獨立的排程器—而獲得很大程度上的好處，並專用於處理所有客戶端應用程式的核心。

由支援 SMP 的即時外掛程式（如英特蒙的 RTX64）提供的輔助排程器，能夠清楚地檢視所有處理器，這確保了核心之間的正確使用和同步。

這能讓使用者實行一個監督行程，用於監控各個核心上的活動。根據環境需求或使用者指令，排程器可使用各種 API 呼叫來分配或整合核心之間的功能，這對於最佳化吞吐量和透過閒置核心最小化耗電量來說非常有用。

作為範例，我們來看看一個四核心的 x86 設備，其中一個核心分配給 Windows，剩下的三個核心分配給 SMP

子系統。在負載較重的情況下，所有分配給 SMP 的三個核心可能都會100%滿載，這是硬體的最佳使用方式。然而，如果 100% 的負載只在 20% 的工作週期內發生呢？這樣就變成多核架構的低效使用，會浪費電力和效能。SMP API能讓使用者編寫監控行程或負載平衡器，將功能整合到單一核心上，使得系統能閒置/停用兩個核心以減少電力消耗。

相同的API呼叫也能在單個核心超載的情況下，將執行緒/行程分配到利用率較低的核心，從而提高處理能力和吞吐量。

虛擬化/ASMP 架構在執行管理各核心之間的行程監督功能或負載平衡方面則存在著很大的困難，因為每個核心都有單獨的應用程式映像和即時作業系統。每個應用程式只了解自己內部的排程並且是透過基於硬體設計的行程間通訊，這意味著為了在不同核心之間正確移動和排程執行緒/行程，原始應用程式必須預先建置所有必要的通訊和支援功能以達到負載平衡，但這通常不在單核設備（如 DSP）的設計規範中。而這些負載平衡API對SMP排程器來說即是原生的API。

此外，虛擬化/ASMP 架構承襲了傳統多處理器設計的限制，例如複製的設計可能在CPU、記憶體、I/O等方面存在一些效能限制，這些限制來自於該設計使用的DSP。

這些限制不一定會被帶到新的多核設備，但由於應用程式是直接導入到虛擬化/ASMP 設計中，因此這些限制仍會存在，結果就是：設計不夠理想。

3. 所有即時行程都能監看硬體狀態

在 SMP 架構中，客戶端應用程式直接運行在核心上，並能無障礙地存取 I/O，簡而言之，所有即時行程都能看到硬體資源的狀態。在虛擬化/ASMP 架構中則不是這樣。

在SMP架構中，客戶端應用程式直接運行在核心上，並能無障礙地存取I/O，簡而言之，所有即時行程都能看到硬體資源的狀態。

由於虛擬化/ASMP 架構只是直接導入曾在專用設備上運行的獨立應用程式，因此並沒有考慮到主機和客戶端應用程式要如何充分利用新設備的功能和特性，這種缺乏協調的狀況最終會導致大量的硬體競爭。

讓我們來看看將一個Windows設備和兩個DSP整合到單個四核心x86的例子：每個不同的應用程式都有其專屬的硬體/周邊設備，包括從序列埠、乙太網控制器，再到 USB 埠，這裡就是虛擬化/ASMP 架構需要一個 hypervisor 或抽象層來管理和仲裁所有不同應用程式之間周邊設備的地方，而這對不同的應用程式產生了一種獨佔特定硬體的錯覺。儘管可行，但這種仲裁會導致系統延遲，隨著應用程式數量的增加，競爭越多，延遲也會越大。

SMP 擁有在核心之間移動執行緒/行程所需的 hook 介面/API，這是負載平衡和正確使用多核架構的必要基石。負載平衡是確保最佳吞吐量和核心利用率的關鍵。

讓我們再次看看將一個 Windows 設備和兩個 DSP 整合到單個四核心 x86 的範例—這次使用支援 SMP 的 Windows 即時外掛程式。重新架構 SMP 的第一步就是將所有時間關鍵或 CPU 密集型的執行緒/行程安排到輔助 SMP 排程器，同時保留 Windows 排程器的功能。接著，設計師確定周邊設備/驅動程式的位置：非時間關鍵的 Windows 子系統或 SMP 即時子系統，一旦驅動程式被指派，該特定子系統就會對該周邊設備擁有不受阻礙的獨佔許可權。這種架構非常具有擴充性，因為隨著核心數量的增加，SMP 排程器就能通過簡單的同步機制（號誌、互斥鎖等）輕鬆控制任何核心上的執行緒/行程和驅動程式。

在某些情況下驅動程式還能共享，而在這些情況下，子系統之間可以匯出驅動程式 API 以共用周邊設備和同步呼叫。這說明 SMP 不僅提供了極大的效能，並且在打造可擴充和並行性的系統方面也提供了高度的靈活性。隨著 Intel 和 AMD 多核設備的快速發展，擴充性和並行性變得越來越重要。

4. 能夠在多個處理器上排程即時執行緒，或將某些邏輯專用於特定核心並利用 hook API 達到負載平衡

SMP 擁有在核心之間移動執行緒/行程所需的 hook API，這是負載平衡和正確使用多核心架構的必要基石。

負載平衡是確保最佳吞吐量和核心利用率的關鍵。

如上所述，虛擬化/ ASMP 在負載平衡方面有困難，因為原始系統分割是沿用於複製的舊有設計，如果沒有額外的副本和在不同核心之間進行十分複雜的協調，就無法輕鬆地在處理器之間移動需要的功能。

5. 直接存取共享數據/記憶體，無需額外副本和行程間通訊

因為 SMP 架構只是和主機應用程式共享記憶體，所以在不同核心/應用程式之間共用資料是很自然的事，這也消除了任何資料複製的要求，因此能減少記憶體需求和提高效能。

虛擬化/ASMP 架構則必須對資料和程式代碼進行多次複製，因為原始設計的限制被承襲下來—例如行程間通訊以及程式和資料對獨佔記憶體的要求，而冗餘資料的緩衝就會增加延遲。

6. 硬體需求最小化—處理器、記憶體、功率和佔用空間

SMP 架構和虛擬化/ ASMP 都能整合硬體需求，減少 25-50% 的運算硬體成本。然而，虛擬化/ASMP 並沒有獲得滿分，因為硬體消耗並未最佳化，而且在應用程

式層級看不到的 hypervisor 層級可能會出現競爭和延遲。

相比之下，SMP 架構因為在核心之間以及兩個子系統 (Windows 和 SMP) 之間共用大量資料，而擁有更高的效率和更小的佔用空間，這減少了所需要的記憶體量，並進一步降低系統成本。此外，藉由在不同核心間組織執行緒/行程所達成的更高效率和性能，往往能減少對處理器的需求，進而降低系統成本。

7. 能夠跨核心進行除錯

這是非常重要的。

在虛擬化/ ASMP 架構中，競爭和延遲可能會發生在 hypervisor 層級，但在應用程式層級卻無法察覺，這尤其麻煩—使得除錯極其困難，並且使系統故障成為主要隱憂。

為了適當地除錯多核心系統，一套完整的工具必須要能識別所有作業系統並且觀察的到整個系統。在虛擬化/ASMP 架構中，因為原始應用程式和即時作業系統 (RTOS) 是直接導入的，IDE 就無法識別運行在其他核心的客戶端應用程式。

在這種情況下，需要使用不同的工具來除錯不同核心或應用程式間的問題，這在雙核心系統並不是特別困難，但隨著系統擴展到四核及以上的時候，就會變得越來越令人難以應付。

想像一下在除錯四核或六核系統時的複雜性，其中有更多四個或六個獨立的IDE 嘗試除錯不同的核心，儘管各個 IDE/除錯器本身很強大，但對其他環境的可見性卻非常有限。當處理 hypervisor 為了仲裁曾經專屬於不同應用程式的共享周邊設備而造成的額外延遲時，這種有限的除錯可見性問題就會變得更加嚴重。

在處理時間關鍵型應用程式時也值得注意的是，hypervisor 造成的延遲會產生一個在應用程式層級必須考慮進去的未知時間常數。

最後，除了除錯的複雜性以外，虛擬化/ASMP 架構通常要求使用者維護所有不同的工具和作業系統。隨著核心數量的增加，維護這些不同的工具會增加成本並帶來額外的支援負擔。

在虛擬化/ ASMP 架構中，競爭和延遲可能會發生在 hypervisor 層級，但在應用程式層級卻無法察覺，這尤其麻煩—使得除錯極其困難，並且使系統故障成為主要隱憂。

SMP 架構在開發工具和除錯方面採用了更簡化的方法。

SMP 應用程式建置在 Visual Studio IDE 中，並且能完全察覺 Windows 和 SMP 排程器。SMP 子系統使用一個輔助排程器來管理即時子系統中的所有核心，由於 SMP 子系統只有一個排程器，因此隨著核心/應用程式數量的增加，除錯的複雜度並不會跟著變大。Visual

Studio 的 RTX 除錯工具可以完全控制和查看所有核心以及兩個排程器—Windows 和 SMP。

在 SMP 架構中的應用程式能夠直接與其周邊設備互動並擁有控制權，從而消除了應用程式必須考慮的任何額外延遲。這種緊密的整合使開發和除錯變得簡單明瞭，使用者可以在 Visual Studio 中輕鬆地設置 Windows 和 SMP 子系統之間的斷點。SMP 使用單一的 IDE 大大簡化和精簡了除錯和工具維護，並且非常具有成本效益。

由於 SMP 子系統只有一個排程器，因此隨著核心/應用程式數量的增加，除錯的複雜度並不會跟著變大。

8. 一次編碼和自動擴充的能力

這點在上面已提及過，但由於並行處理是非常重要的，因此值得更詳細的解釋。

SMP 架構包括可以指派處理器親和性和理想處理器的 API，這些 API 讓開發人員只需要為理想的多核架構編寫一次程式，軟體就能隨著更多可用核心的增加而自動擴充到設備上。

相反的，從不同的多處理器轉換成單一晶片之後，在虛擬化/ ASMP 的環境中為各種系統設計適當利用不同核心倍數會變得困難。

如果沒有大幅度的重寫程式碼，虛擬化/ ASMP 設計必須在每個新增的核心上放置原始客戶端應用程式的多個副本/映像，但這樣就無法利用原本能藉由組合那些在特定核心上運行而獲益的執行緒/行程來提高效能和吞吐量，變成很多餘且浪費資源。

開發人員非常有理由擔憂虛擬化/ASMP 無法減輕傳統多處理器系統中存在的複雜性和低效率—多種工具環境、多個開發團隊和多個作業系統。

而從不同的晶片整合到周邊設備共享的多核架構時，資源競爭可能會成為各個開發團隊面臨的重大挑戰。

以前專用於每個應用程式的周邊設備現在變成共享，不可避免地意味著當需要的周邊設備正在忙碌時，有些操作就必須等待，這不僅效率低下，還和從多核硬體中獲得最大處理速度和性能的目標背道而馳。簡而言之，閒置的晶片是資源的浪費，開發人員要真正降低系統成本，必須時時刻刻將所有處理的潛力最大化。

如前所述，這些挑戰在 SMP 架構中根本不存在。

簡單來說，儘管虛擬化/ ASMP 可能適用於單一任務或較不複雜的環境—這也是桌面虛擬化容易被接受的原因—但並不適合用來解決必須具備硬即時、精確性、靈活性、加速上市時間和可擴充性的複雜多任務系統。事實上，要讓虛擬化/ASMP 成為複雜嵌入式系統的可靠選項，就需要一個 Windows hypervisor 擴充程式，也就是外掛一個即時 SMP 產品並擴展核心和周邊設備的存取。

有了 SMP，開發人員在重新思考和設計系統的時候會有很多選擇，不僅能充分利用多核心能力，還可以改善開發流程，實現長期經濟效益和擴充性。

透過使用支援SMP的硬即時子系統為Windows/x86系統增加第二個獨立的排程器，系統開發人員獲得了一個適合多處理能力最大化的軟體控制架構，同時提供超越系統整合的顯著經濟和性能效益。

結論

充其量而言，虛擬化/ASMP 代表了從多個系統到多核心的短期、低價值整合路徑。然而，對於一個真正簡化和精簡、高性能、可擴充、高效且具有長期價值的架構來說，支援 SMP 的軟體控制架構還是最佳選擇。

對稱多處理（SMP）vs. 虛擬化（ASMP）

Windows/X86 即時嵌入式解決方案的 8 個關鍵設計元素	優勢	SMP	ASMP
1. 通用的整合開發環境（IDE）和世界級的圖形使用者介面（GUI）—Microsoft Visual Studio 和 Windows 作業系統，包括 Windows 10 和 Windows Embedded Standard 7	整合 & 生產力	●	○
2. 直接在多個指定處理器（不是多個個體）執行能支援 SMP 的即時子系統	效能 & 可維護性	●	○
3. 所有即時行程都能監看硬體狀態	控制	●	○
4. 能夠在多個處理器上排程即時執行緒，或將某些邏輯專用於特定核心並利用 hook API 達到負載平衡	控制 & 效能	●	○
5. 直接存取共享資料/記憶體，無需額外副本和行程間通訊	品質 & 效能	●	○
6. 硬體需求最小化—處理器、記憶體、功率和佔用空間	效能	●	◐
7. 能夠跨核心進行除錯	效能	●	○
8. 一次編碼和自動擴充的能力	可擴充性	●	○