可組態即時嵌入式作業系統

廖筱芸、熊博安

國立中正大學資訊工程學系 嵌入式系統實驗室

一、簡介

SIMD(signal instruction, multiple data)指令一開始被使用在向量超級處理器(vector supercomputer)，而自 1970 年被 Cray(超級電腦製造商)普及化後，“單一指令、多資料流”(SIMD)是近代計算機組織很重要的特徵，本文章可組態即時嵌入式作業系統是設計與實作一套專門針對具有原生 SIMD 指令集的微處理器，例如 PLX，的開放程式碼、可行的、有效率的可組態即時嵌入式作業系統(RTOS)。

我們的目標不僅是設計一套可行的即時作業系統，而且是一個盡量利用 PLX 微處理器特性以及可在面積、效能與功耗之間做取捨的作業系統。事實上，我們並非只設計一套固定的即時作業系統，而是設計一個根據 PLX 組態的機器定義檔(machine definition file) 自動生成該 PLX 組態專屬的即時作業系統。如此一來，微處理器與作業系統之間就會有更佳的整合。

下面各部份將分別介紹要實作的平台—PLX，現存可配置的即時作業系統的例子，還有在嵌入式系統中所關注的現存能源管理策略、安全議題以及最後的結論。

二、PLX平台特性

PLX 是普林斯頓大學電機系 Ruby Lee 教授所發展的小型具有 wordsize 可展延、平行次字彙(subword-parallel，native SIMD)指令集架構的微處理器，用以設計處理多媒體，如圖 1 所示。

1. PLX 是精簡指令集(Reduced Instruction Set Computer，RISC)的架構
2. 以功能特性(functional)分成五種指令格式。其中所有指令都是可被預測的，可以減少條件分支(conditional branches)。
3. 為了快速處理媒體而設計的完全次字彙(subword)指令集架構。
4. PLX 具資料路徑延展性(datapath scalability)，其資料路徑有 32、64、128 位元寬。PLX 可以實作此三種不同位元架構，不會更改到指令集架構。
5. PLX 指令集架構充分利用多媒體應用的兩種特性：
   甲、 大量資料平行化。
   乙、 低精準資料(low-precision data)的廣泛應用。

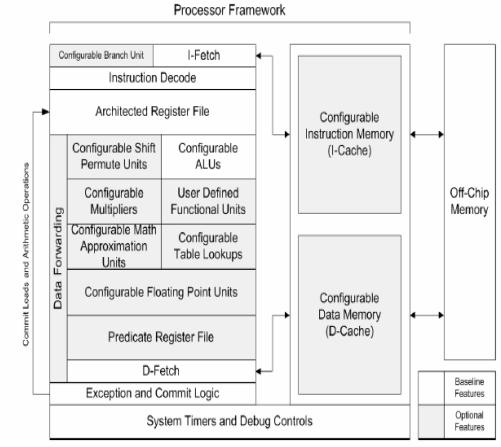

圖 1. PLX 處理器架構

PLX 有固定長度 32 位元的指令，最多有二個運算元、一個結果。指令集架構可以分成三類主要部份：ALU 指令、位移與排列指令(Shift and Permute Unit，SPU)、乘法指令(圖 2)。所有指令都是 32 位元，次字彙(subword)的大小則分成 1、2、4、8 位元組(圖 3)。可參考 Refining instruction set architecture for high-performance multimedia processing in constrained environments

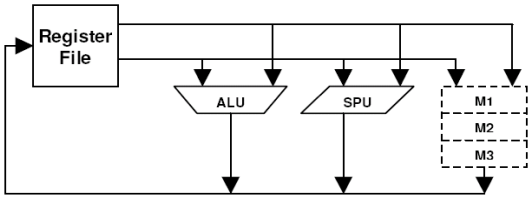

圖 3. PLX 指令集架構

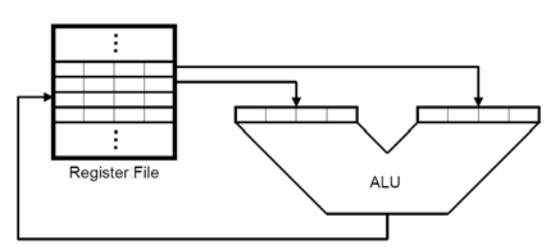

圖 2. 次字彙平行

下表 1 是基本的 ALU 指令，主要是可以平行做加減動作、平行做位移與加、平行做平均及找最大最小值、邏輯及比較的指令。

Instruction	Description
padd	$c_i = a_i + b_i$
padd w/ saturation	$c_i = a_i + b_i, \quad c_i \in [L,H]$
psubtract	$c_i = a_i - b_i$
psubtract w/ saturation	$c_i = a_i - b_i, \quad c_i \in [L,H]$
paverage	$c_i = average(a_i, b_i)$
psubtract average	$c_i = average(a_i, -b_i)$
pshift left add	$c_i = (a_i << n) + b_i$
pshift right add	$c_i = (a_i >> n) + b_i$
pmaximum	$c_i = \max(a_i, b_i)$
pminimum	$c_i = \min(a_i, b_i)$
logical operations (and,or, not,xor,and complement)	$c = a\ op\ b$, where *op* is one of the logical operations
cmp (compare)	$P_{d1} = rel(a,b);\ P_{d2} = !P_{d1}$
cmp.pw1 (compare parallel write one)	*see Section 3*

表 1. 基本 ALU 指令

[PLX 1.1 ISA Reference**]**

乘法指令(表 2)對某些不需要太精確的運算使用 pmultiply shift right 指令先乘完再位移；pmultiply even 或 pmultiply odd 只對奇數或偶數做乘法。

Instruction	Description
pmultiply shift right	$c_i = [(a_i * b_i) >> n]_{lowerhalf}$
pmultiply even	$[c_{2i}, c_{2i+1}] = a_{2i} * b_{2i}$
pmultiply odd	$[c_{2i}, c_{2i+1}] = a_{2i+1} * b_{2i+1}$

表 2. 乘法指令

PLX 具有位移與排列指令(表 3)，可以共同(parallel)平移及次字彙排列的指令。共同平移指令可以依照立即值或在暫存器的數值做左右平移的動作。shift right pair 指令可以串連兩個資源指令並且向右平移它們在暫存器中的值。被平移過的較低半部被放置在目的地暫存器，而此指令當暫存器相同時可以達到旋轉(rotation)的動作。

上述指令相對於一般的機械指令更能對於多媒體的運算有所助益，利用平行化的方式能達到對具有資料量大的多媒體快速運算。雖然 PLX 設計簡單，但是其指令集架構對於多媒體運算卻是具有高度效能。

Instruction	Description
pshift left	$c_i = a_i << n$
pshift left variable	$c_i = a_i << b$
pshift right	$c_i = a_i >> n$
pshift right variable	$c_i = a_i >> b$
shift right pair	$c = [[a,b] >> n]_{lowerhalf}$
mix left/right	*see text*
permute	*see text*
permute variable	*see text*

表 3. 位移與排列指令

三、可配置的即時作業系統

可組態即時嵌入式作業系統將採用模組化設計策略如圖 4，其中實作的微核心僅含一個基本的即時排程器。其他的模組可視需要動態執行時載入。核心中的程序管理者將負責高效能低功耗多重執行緒之間的溝通與排程。可載入式模組將包含區塊組態式記憶管理、輸出入管理、驅動程式、功耗管理、檔案系統管理、資訊安全協定介面、通訊介面及多媒體標準。

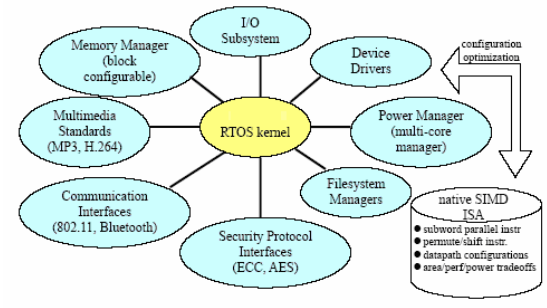

圖 4. RTOS 的可配置模組化架構

下面舉出幾種現在存在的可配置的即時作業系統，如 MQX RTOS 提出可配置的部份：處理器、快取記憶體(cache)、匯流排(bus)、計時

器(timers)等。MQX 依照平台及選項的選擇，可以使所佔空間小到 6KB，而其可以配置的項目如下圖 5 所示，包含內核(kernel)、中斷、信號(semaphore)、佇列及記憶體管理等。

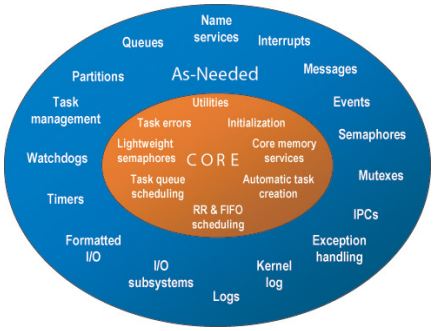

圖 5. MQX RTOS

[MQX RTOS home page

http://www.mqxembedded.com/products/mqx]

另一開放原始碼的嵌入式可配置作業系統 Rea Hat 的 eCos（Embedded Configureable Operating System）的特點是可配置性(configurable)、可攜性(portable)和即時性(real-time)。eCos 主要特色是可以讓使用者自行選定所需要的模組。在嵌入系統與記憶體的資源有限情形下，以達到使用者所需功能的最小化模組。應用程式設計者只要使用 eCos 所提供的配置工具(Configuration tool)即可以對 eCos 內模組加以選擇，打造一量身訂做的 OS，而不需要了解內部實作，見下圖 6。

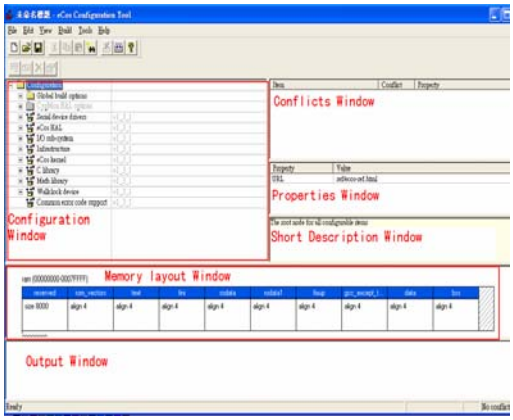

圖 6. eCos 配置工具

LibeRTOS 是 KURT-Linux 的企業化版本。KURT-Linux 是尚有爭議仍在研究中的版本，LibeRTOS 會獲得研究結果的更新，而 LibeRTOS 可以使 KURT-Linux 發展。LibeRTOS 具有下列可配置性詳細資訊可參考 LibeRTOS: A Congurable Single-OS Real-Time Linux Platform for Industrial Applications,

http://linuxdevices.com/articles/AT3829525232.html.

1. 分群排程(Group scheduling)
2. 在分群排程下，允許軟中斷(soft-IRQ)及小程式(tasklet)的控制是可以配置與整合的。
3. 可載入的模組能完全取代預設的 Linux 語意，變成使用者想要的程式模組。

上述各種現存的可配置的即時作業系統接有其可以配置的項目，而 RTOS 計畫中是要利用模組化的特性達到可配置的特色。

四、電源管理

對於嵌入式系統而言，有效控制能源使用，減少能源耗費是一項很重要的問題。因此藉由電源管理系統，降低非使用中的元件，來延長裝置的使用時間。電源管理有高階電源管理(Advanced Power Management，APM)、先進架構電源介面標準(Advanced Configuration and Power Interface，ACPI)、動態電源管理(Dynamic Power Management，DPM)。

高階電源管理由Microsoft與Intel公司針對監視器所開發出來的應用程式設計介面，可保存個人電腦系列和具有特殊電池供電的膝上型電腦之監視器的電力，由應用程式溝通相關的電力需求，不會對供應電力不再使用之硬體元件，以作到節省電源的要求。但是他對BIOS過度依賴，新舊BIOS系統之間會產生不相容的情形，無法判斷電源管理指令是由使用者發起還是由BIOS發起。現在只存在於Linux的一些可攜式裝置。微軟作業系統的筆記型電腦及手持裝置已經不在使用高階電源管理。

先進架構電源介面標準(圖7) 這是英特爾、微軟和東芝共同界定的電源管理標準，可以透過電腦鍵盤，將電源整合管理，也就是說，

使用先進架構電源介面標準電源管理，操作系統就可以不必藉助外來設備，並達到自行管理電源，靈活運用電源。管理電源的主要執行者由BIOS轉成作業系統。目前它對x86/IA-32BIOS架構十分依賴。

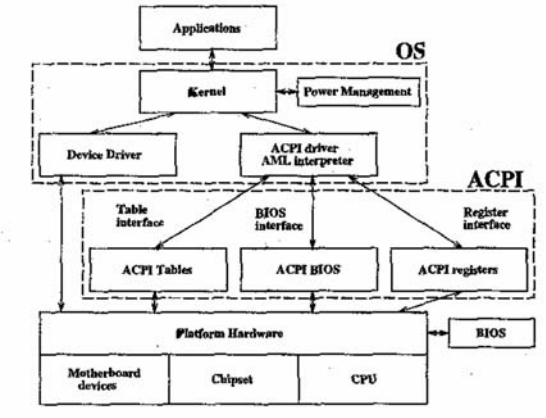

圖7. ACPI架構

Benini, L.; Bogliolo, A.; De Micheli, G., “A Survey of Design Techniques for System-Level Dynamic Power Management,” *IEEE Transactions on VLSI Systems*, Vol. 8, No. 3, pp. 299-316, June 2000

嵌入式系統通常沒有BIOS(在PC/AT的意義上)，並且無法奢侈的使用抽象層來將作業系統以及電源管理活動隔開。Embedded Linux採用動態電源管理(圖8)來對作業系統的核心及設備驅動做特殊干預。

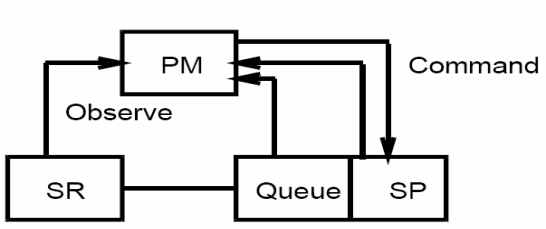

圖8. DPM概觀架構

E.-Y. Chung, L. Benini, A. Bogliolo, Y.-H. Lu, and G. De Micheli, “Dynamic Power Management for non-stationary service requests,” *IEEE Transactions on Computers*, Vol. 51, No. 11, pp. 1345-1361, Nov. 2002.

低階的動態電源管理駐留在作業系統的核心，而其電源管理策略是來自作業系統外(圖9)。這些策略以兩種方式與動態電源管理溝通：1.預先定義一些需要的方針策略予以使用。2.利用應用程式管理者或是方針策略集合管理者管理這些策略。

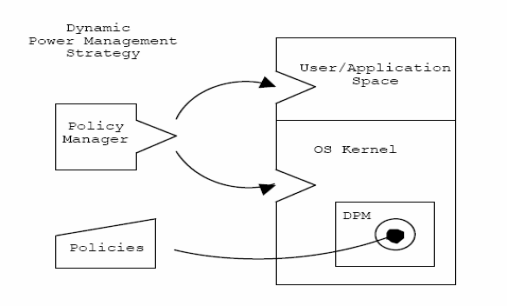

圖9. 高階DPM的觀點

B. Brock and K. Rajamani, “Dynamic Power Management for Embedded Systems,” In Procs. of the *IEEE International SOC Conference*, pp. 416-419, 2003.

六、結論

雖然現在模擬實驗版的空間大小已有大幅度的成長，然而卻不是無限制的增大，空間仍是有限的。對於使用者而言，可以針對自己所需要的特定目的而彈性地選擇其所需的配置在開發版上，可以縮小整個作業系統的空間。未來 RTOS 希望除了能夠 configurable 的特性發揮的淋漓盡致外，且希望結合電源管理，以期望未來使用者使用 RTOS 計畫開發應用出來的設備時，其能源耗費是較小的，且能量使用壽命時間是較長久的。