衡量微控制器不僅取決於匯流排寬度

當談到微控制器時，今日嵌入式開發人員有著大量的選擇，但是匯流排寬度不是唯一絕對的性能衡量標準。

■作者：Matt Saunders/
Silicon Labs 微控制器和無線產品區域市場行銷總監

自從半導體製造商開始轉移 32 位元架構至微控制器，一些技術論壇中的人們就已經開始預測 8 位元晶片的滅亡。現實中，8 位元晶片的使用量確實已有下降，他們可能不再佔據主導地位，但 8 位元晶片真的離滅亡不遠了嗎？實際上現在的製造商仍然一直開發和擴展他們的 8 位元系列產品，甚至包括那些正提供 32 位元產品的供應商。

當提到該如何在 8 位元和 32 位元 MCU 之間選擇時，或許爭論的焦點其實就在於他們的彈性；畢竟，單一個裝置就能衍生許多應用。但是如果 MCU 被設計得如此具有彈性，為什麼仍然有如此多的變種呢？對於該問題的多數答案將是周邊而非核心，但是實際上核心和它的周邊緊密相關。

還有一種看法是：8 位元架構和相對應的 32 位元架構相較之下是過時的，但是實際上的比較結果或許出人意料。雖然他們的指令集可能已經被完整地建構，但是大多數 8 位元核心在他們的生命週期中有不止一次的「升級」，同時就像任何其他元件一樣，他們也受益於製程的發展。因此，認為兩種架構在某個方面可相互媲美，這應當是合理的。

基本差異

除了明顯的匯流排寬度差異之外，8 位元晶片通常比 32 位元晶片更「小」，特別是和核心整合在一起的儲存容量，以及相關的平均售價。同樣的，如果需要完全整合系統級功能 (例如 LCD 控制器 / 驅動器)，那麼這些功能更可能出現在 32 位元解決方案中。

一般而言，如果系統需要的代碼儲存容量大於 65kbyte，則需要選擇 32 位元解決方案，如果需要的代碼儲存容量小於 8kbyte，那麼 8 位元解決方案更可行。當然，就其本質而言，8 位元元件對於簡單操作可能需要更少的代碼空間，但是 32 位元指令集的單條指令可以完成更多的工作，因此，在較大和更複雜的應用中，更複雜的指令集實際上可能獲得更好的整體代碼密度。

對於代碼容量在這兩個極端之間的應用，或者僅僅需要「標準」MCU 周邊，選擇判斷標準不再顯而易見，需要基於實際的應用選擇。透過花些時間分析應用，工程師便能夠快速確定何種架構最適合他們的需求。

基準性能

當然，大多數工程師可能會說 8 位元和 32 位元的主要區別完全在於性能，但這只能根據具體的情況才能這樣說，要看具體的要求。「應用性能」才是真正要考量的問題。

舉例來說，對比 8051 和 Cortex-M0+；8051 是完全著眼於 8 位元應用領域的架構，這也是大多數工程師可能要進行的對比，因為它要用於嵌入式

領域。脫離具體環境直接進行資料手冊對比將是沒有意義的；在大多數情況下，Cortex-M0+ 裝置可能會「勝出」，但在真實的場景中，結果可能會大相徑庭。

較大核心的一個特點是不用太在意它們的資源使用情況；而在嵌入式系統中，這會引發問題，包括 8 位元架構開發人員一直避免的問題。舉例來說，考量圖 1 中的代碼。在基於 Cortex-M0+ 的裝置上編譯和執行代碼時，我們發現堆疊需要 48 位元組，而在 8051 上編譯和執行相同的代碼時僅需 16 位元組。儘管區別不是很大，但在 RAM 有限的系統中，這一點就變得非常重要。

圖 1 ··

```
int main(void){
   funcA(0xACED);
   while (1);
}

void funcA(uint32_t a){
    uint8_t i, j=0;
    for (i=0; i<3; i++){j = funcB(i, j);        }
}

uint16_t funcB(uint16_t testA, uint16_t testB){
   return  (testA * testB)/(testA - testB)
}
```

基於 8051 最初設計的原因，它一直採用非統一的儲存映射。在大多數情況下，這能夠提高效率，因為它使用不同的指令指向不同類型的儲存區 (例如：Flash、內部 RAM 或外部儲存)。不過，指令集還允許通用指標指向任何類型的儲存區，如此可提高代碼的再使用性，但代價是會稍稍影響執行的效率。ARM 架構有統一的儲存區管理，這意味著無需使用特殊指標，工作可能會變得更簡單。

低效是困擾嵌入式開發人員的一大問題，開發人員會想盡一切辦法避免這個問題，這凸顯了另一個問題：延遲。直覺上，工程師可能會認為 Cortex-M0+ 對中斷和函式呼叫有更快的反應時間，但實際上 8051 架構更快。ARM 核心透過 AMBA 高性能匯流排 (AHB) 在高階周邊匯流排 (APB) 上訪問周邊的事實使得情況變得更糟。

基於此原因，在簡單的系統中，8051 能夠顯示出其在中斷服務程式進 / 出時間上的優勢，但在更加複雜的系統或執行時間更長的服務程式中，優勢變得不再明顯。

應用適用性

一般來說，8 位元和 32 位元核心的另一個重要差異是處理控制任務時各自內在的優勢和劣勢，尤其是 8051 和 Cortex-M0+。8051 指令集在計算位元和位元組時表現卓越，而 Cortex-M 架構的優勢在於能夠流暢處理較大的資料塊，或使用廣泛的數學函數執行複雜的邏輯演算法。

在判斷何種架構最適合應用時，這種「控制 vs. 處理」的對比尤為有用，但這並不是一成不變的規則；雖然在一個主要實現 UART-to-SPI 橋接器的應用中，採用 ARM 元件可能會表現的更高效，但是 8 位元方案也能輕而易舉處理這種情況，而且可能會非常適合僅僅有 2kbyte 整合儲存容量的元件。

舉例來說，在一個應用中，它 10% 的時間用於執行 32 位元數學函數，25% 的時間用於處理控制函數，剩餘的 65% 處理時間則用於執行一般目的的活動。如果沒有清晰顯著的優先考量之架構，並且如果系統級要求是更小的代碼空間而不是執行速度，那麼可能更適合選用 8 位元產品。但是，如果要優先考量執行速度，則可能就要選用 32 位元產品了。

評估整體功耗時，也可做同樣的對比，一般情況是整體評估兩種選擇方案的占空比、活動功耗和休眠電流。現在，許多供應商提供硬體和軟體工具來協助工程師評估這些參數，尤其是那些同時有 8 位元和 32 位元產品組合的供應商，例如 Silicon Labs。

最後，如果在為某個應用選擇 8 位元或 32 位元產品時，如果沒有明顯優於對方的好處，那麼情況很有可能是：即使做出「錯誤」選擇，也真的不會有什麼大問題。8 位元架構在嵌入式開發中仍佔有重要位置，這就繼續要求工程師們仔細評估其選擇，而不是在今後一段時間裡理所當然地選擇單一的通用架構。 CTA