TeraSoft 鈦思科技
www.terasoft.com.tw

利用 Simulink 即時及多核心目標硬體進行同作即時執行

By Abhishek Bhat, MathWorks

即時執行一個高逼真度(fidelity)的受控體模型作為硬體迴圈(hardware-in-the-loop, HIL)模擬有助於減少硬體原型的必要性，這同時也代表了開發時間及成本的降低。然而，當模型的逼真度愈高，CPU 超限運轉(*overrun*)的風險也就愈高 — 也就是目標電腦的 CPU 超載。

您可以使用同作即時執行(*concurrent execution)*來降低這種風險。在同作即時執行中，模型是被分割的，而且每一個分割的部分都平行地在一個多核心的機器執行。同作即時執行透過讓目標電腦的多核心之間的運算負載達到平衡來改善即時系統性能，因為這增加了可在特定取樣時間內執行的程式碼量。本文將以一電動車模型為例，講解一個使用 Simulink 即時控制工具(Simulink Real-Time™)及一個 Speedgoat 雙核心目標電腦進行同作即時執行的工作流程。

我們將從在單一核心執行模型開始來獲得一個即時性能的基準測量值。接著，我們配置模型來利用多核心並執行一個同作即時執行。從單核心與多核心的模擬結果比較，我們可以看出 CPU 是否有超限運轉的風險。

高逼真度受控體模型

在這個範例中，我們將使用一個透過 Simulink®、傳動模擬模塊組(Simscape™ Driveline™)、電子模擬模塊組(Simscape Electronics™)、以及物理模型模擬模塊組(Simscape™)建立的電動車模型(圖 1)。

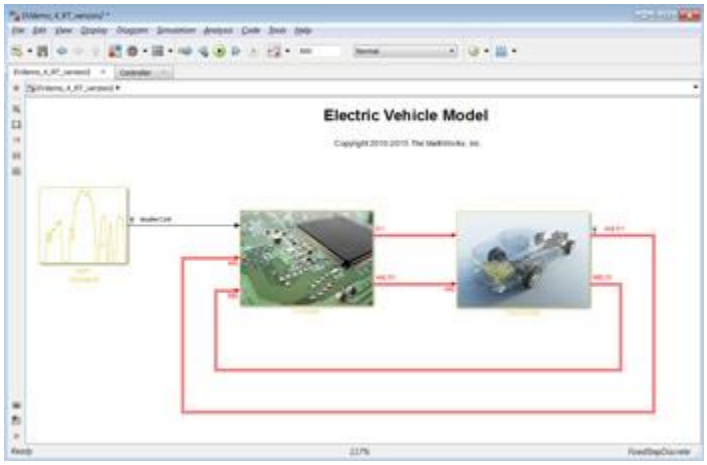

圖 1：電動車基準模型

這個模型包含了下列幾個部分：

- 測試輸入- 駕駛週期(本測試使用 US06 駕駛週期)
- 運算子及動力管理子系統
- 電池、電力驅動、及車輛模型

模型建立目的是為了能夠在 HIL 模擬器執行這個高逼真度的受控體模型，所以我們可以將真實的 ECU 連結到模擬器並測試其性能。

執行一個基準模擬

我們開始在一個單核心即時機器執行這個電動車模型來計算整個執行時間(即時執行在模型上所需要的時間)。我們將利用執行時間作為基準來將單核心及多核心的執行結果做比較。

為了要建構 Simulink Real-Time 模型，我們會進行以下步驟：

1. 選擇解算器以及取樣階躍時間—在這個案例為離散解算器及 Ts=0.001。
2. 到 Configuration Parameters>Code Generation 將系統目標檔案設定為” slrt.tlc”。
3. 至 Configuration Parameters>Code Generation>Verification> Code Profiling 選項，啟動” Measure task execution time”並且在儲存的選單下選擇” All measurement and analysis data”。

在這之後，我們建立並下載這個模型到即時的機器之後執行 10 秒鐘，接著利用以下指令產生一個程式碼執行剖析報告：

```
% Matlab Code-

profileInfo.modelname = 'EVdemo_4_RT'; % your model name

profData = profile_xpc(profileInfo);
```

這個剖析報告提供了一個模型執行時間的摘要(圖 2)。

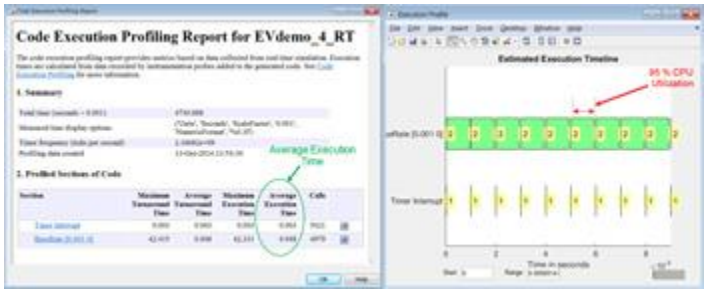

圖 2：基準模型的程式碼執行報告及執行剖析。

對於基準模型，所有的模塊在產生程式碼時以任務進行分組。因此，剖析報告只顯示兩個任務：計時器中斷與基本比率。

報告的右側顯示模型任務的平均的執行時間為 0.948ms，而維持觸發模型執行的計時器中斷所需要的時間為~ 0.003 ms。

這個執行剖析顯示任務已連續地執行(圖 3)。

圖 3：基準模型的執行剖析

整個執行時間是維持計時器中斷所需要的時間以及執行模型需要的時間的加總—在本案例為 0.948+0.003 =~0.951ms。這代表了我們已經使用超過 95%的核心能力。任何模型動態造成的執行時間中的變化都會導致 CPU 超限運轉並且中斷模擬。為了避免這種情況發生，我們將採用同作即時執行。

同作即時執行的模型分割

如果模型具備多種比率，您可以啟用“Allow tasks to execute concurrently on target”選項以在結構參數利用目標硬體的多核心，而不用重建模型。依據各比率完成的計算工作量多寡，有可能產生執行時間效益，也或許不會。這個步驟因為需要極小的配置，可以說是一個簡單的嘗試實驗。在我們的案例當中，整個模型在一個單一比率下執行，所以我們將會需要明確地分割這個模型已取得多核心的效益。

模型的分割有幾種方法。在高層級，模型以依據以下任何一項來分割：

- 比率 – 依據模型的多核心執行比率(比如慢比率、快比率)來分割
- 物理學 – 依據子系統的物理特性(例如引擎模型、傳輸模型、及電氣系統)來分割
- 功能 – 將 I/O 通道及受控體模型元件分離

由於測試模型是單一比率，而且我們不使用任何的 I/O 通道，我們因此依物理特性來分割模型如下：

- 機構元件 – 車輛模型
- 電氣元件 – 電池模型以及 DC/DC 轉換器
- 控制 – 運算子模型

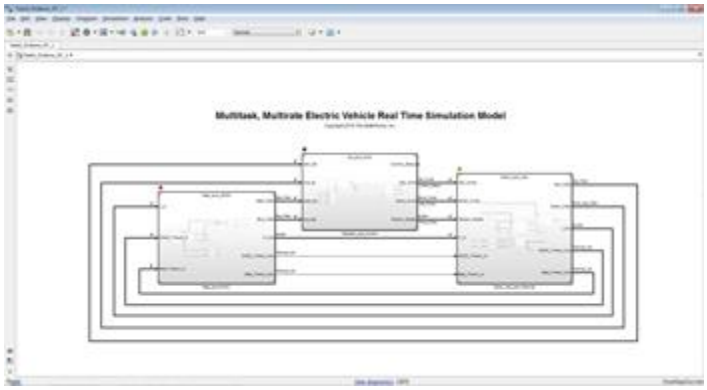

圖 4：同作即時執行的模型分割。

配置同作即時執行模型

為了配置同作即時執行的模型，我們進行以下步驟：

1. 在配置參數(Configuration Parameters)對話框的解算器設定內，啟用“Allow tasks to execute concurrently on target”並選擇“Configure Tasks”(圖 5)。

圖 5：分割模型的配置參數(Configuration parameters)

2. 建立 3 個分隔的任務，讓我們可以點選左上角的 **Add task** 來把每一個參照模型指派到個別任務(圖 6)。

圖 6：配置所有的模型任務。

3. 使用 **Task and Mapping** 標籤指派各次的參照模型(圖 7)。

圖 7：Task mapping

模型現在被配置來利用多核心。當模型被更新，指派的任務會在左上方以上色的圖示標記。這代表我們可以一直藉由查看頂層模型來檢視分配的任務。

要注意的是，以這種方式分配的任務藉由在任務之間的邊界增加單位延遲改變了模擬模型的行為。同作即時執行透過任務轉換列的註解讓您看到這些變化，也讓您模擬變化的行為。在模型分割的過程中，頻繁地模擬系統會是個好主意，因為單位延遲可能造成受控體模型功能的改變。

利用'Build and Download'按鈕，我們下載這個模型到 Speedgoat 即時目標電腦。

同作即時執行結果

在模型執行 10 秒之後，我們產生了一份新的剖析報告(圖 8)。

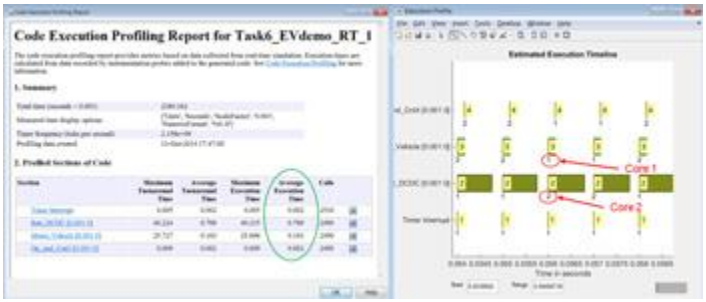

圖 8：同作即時模型的程式碼執行報告及執行剖析。

這份報告證實模型執行被切割為三個平行執行的任務，而不是基準模型中的單一任務。另外還有一個附加的計時器中斷任務的入口。

我們可以看出 Battery_DCDC 任務平均執行時間最長。這代表從計算方面來看，這個子系統的執行時間最高可達基準模型 0.948 ms 中的 0.789 ms。

因為模型任務現在已經被平行地安排，模型執行總共需要的時間只有 0.789+ 0.002 =~ 0.791 – 比原本的 0.951 ms 低了 16.82%。

要注意的是，利用雙核心並不會自動改善 50%的性能。實際的性能改善會因模型類型以及任務分割的方式而異。除此之外，資料在多核心之間的轉移會花上一些時間，所以會有一個與建立額外任務有關的懲罰。更多的任務幫助我們在多個核心之間分割模型，這是一個優點，但利用多核心也會增加核內(inter-core)溝通需要的架空(overhead)。如果要達到更進一步的改善，我們可以從 Speedgoat 利用高效能四分之一核心的目標電腦，而非雙核心機器。

總而言之，同作即時執行利用目標電腦的多核心幫助您即時執行高逼真受控體模型。我們看到總執行時間明顯降低，這給了我們足夠的信心證明即時模擬不會因 CPU 超限運轉而中斷。

減少模型模擬時間有益於 HIL 測試，因為這表示我們可以在更接近實際實體系統的高逼真度的受控體下測試控制器。