# 利用快速原型化及模型化基礎設計來加速感測器開發

By Martin Hein, Hella Fahrzeugkomponenten GmbH

汽車代工廠在購置新的感測器系統之前，多半會希望先看到感測器在即時車輛環境下的原型，以評估其性能或變更規格。舉例來說，我們往往需要展示一個早期版本的感測器，這通常包含執行於 FPGA 或微型處理器的演算法及邏輯。為了符合這項要求，我們利用一個客製的快速控制原型化平台以及模型化基礎設計，為一個仍在開發階段初期的新設計建立即時原型。

與其為了一個 ASIC 實現等待長達兩年的時間，我們其實可以在短短幾個月內先製造一個具備最終產品 80%功能的 A-sample。A-sample 幫助我們在開發階段早期與客戶一同精煉感測器的功能並評估程式碼大小、模組的劃分、及硬體要求。測試團隊利用 A-sample 來設置測試環境及測試套件，因而能夠在以 ASIC 或微型處理器實現的產品樣品準備好時立刻開始進行測試。

## 建立一個具備靈活度的原型環境

由於市面上可取得的替代品並不符合我們對靈活度的要求，我們自行建立了 Hella Fahrzeugkomponenten GmbH Rapid Control Prototyping (HFK RCP)單元。大部分現成的原型化系統只支援 ECU 軟體開發，但感測器的設計可能還會需要包含 VHDL®程式碼與離散電子零件。市售系統的另一項限制為其僅提供固定的介面。在 Hella，我們必須支援各式各樣的通訊協定與介面硬體，包含 SPI、$I^2C$、LIN、XCP，CAN、與 SENT。

透過模型化基礎設計以及我們的客製的原型化環境，我們可以依照需求增加新的介面、協定、功能。我們可以把微型處理器與 FPGA 訂為目標。當我們在開發規格，並且利用原型化環境來延伸或改善演算法時，演算法已經在生產處理器上實現了。

## 從要求條件到設計

我們的開發過程按照 V-model 進行，共包含五個主要步驟，分別為：要求條件分析、演算法設計、生產程式碼的生成、程式碼驗證、以及測試。在條件分析階段，我們與客戶合作，一同定義在 IBM® Rational® DOORS®的系統條件。接著在 Simulink®建立一個設計的初始模型(圖 1)。

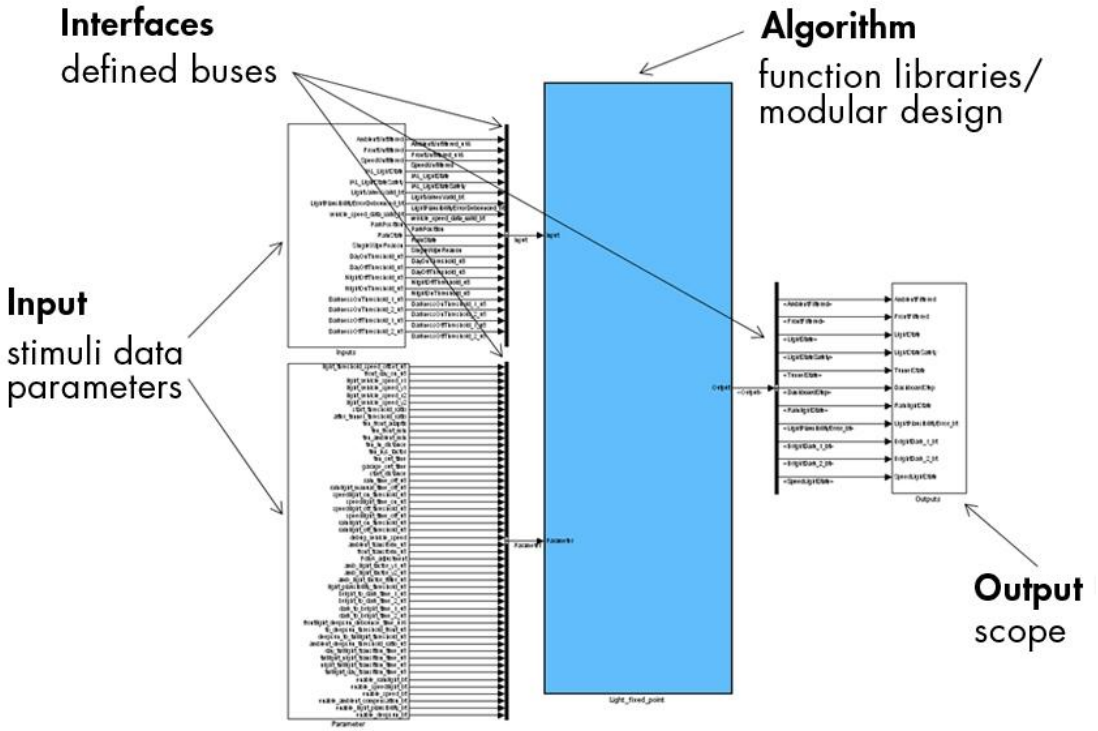


圖 1. Simulink 系統模型

我們利用 Simulink 驗證及有效性檢測模塊組(Simulink Verification and Validation™ )將 DOORS 的要求條件繪製於模型的每個元件，讓這些條件可以被雙向追蹤。

建立模型時，我們使用 Simulink 的 Model Advisor 功能來確保我們遵照 MathWroks Automotive Advisory Board (MAAB)演算法建模準則，也以 Model Advisor 依照在 Hella 內部制訂的準則進行檢驗。

為了對初步的浮點設計進行早期功能驗證，我們在 Simulink 執行模擬，以從一個相似的感測器所蒐集到的測試數據，或由 Simulink 模塊產生的測試數據來刺激模型。在這些模型迴圈測試(model-in-the-loop test)之後，我們檢查由 Simulink 驗證及有效性檢測模塊組產生的模型覆蓋報告來辨識模型中尚未經過測試的元件，並在必要時更新測試來增加覆蓋。

在快速原型化平台測試的準備階段，我們建立了通訊介面模型，讓感測器演算法可以執行於車輛。與 MathWorks 顧問合作，我們開發了一個 Simulink 區域互聯網路(local interconnect networking, LIN)模塊組，讓我們能夠擴充原型化系統的能力來支援 LIN。

# 從建模到原型化

在模型經過內部的設計檢視之後，我們將設計移植到 HFK RCP 單元(圖 2)。HFK RCP 內含 TI 的 C2000™ 微型處理器、Xilinx® FPGA、車用匯流排與感測器之間的連接器、以及一個離散電子元件區塊，支援各種具備標準化元件組的設計結構。

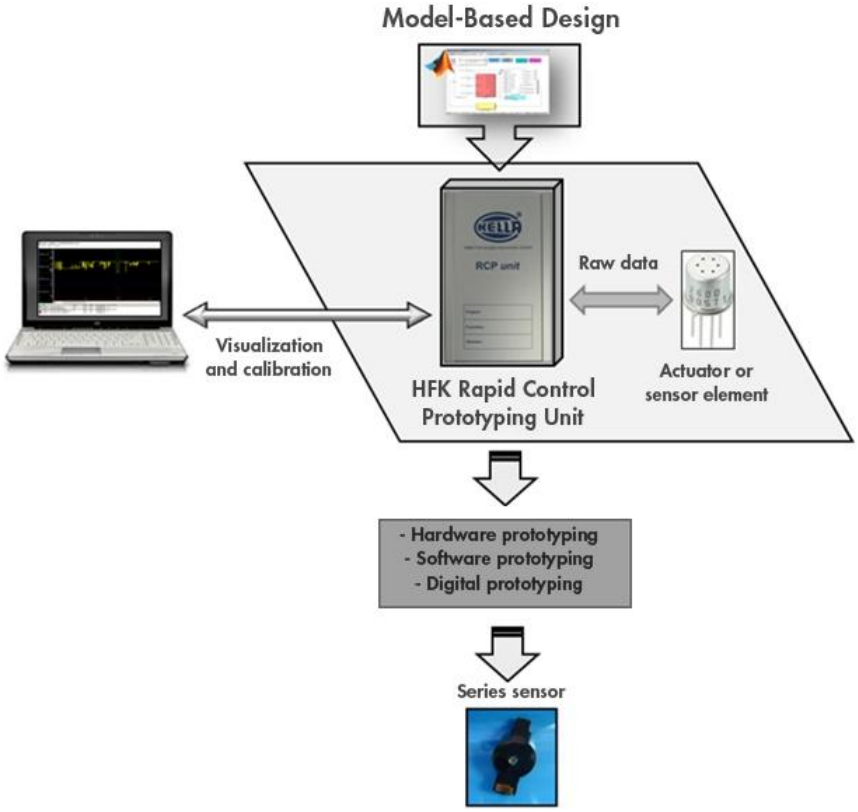


圖 2. 運用 HFK RCP 單元以及模型化基礎設計的 Hella 工作流程

我們利用 Embedded Coder™，從 Simulink 模型產生針對微處理器的設計的程式碼，並轉檔佈署到 HFK RCP 單元上的 TI C2000 處理器。如果設計需要用到 FPGA，我們則使用 HDL Coder™ 從模型中產生 VHDL 程式碼，轉檔佈署到 Xilinx 的 FPGA。

# 從原型化到生產

以 HFK RCP 單元驗證過作為概念驗證用的 A-sample 之後，我們準備了要在系列處理器上實現的設計。我們利用定點設計工具箱(Fixed-Point Designer™)中的 Fixed-Point Advisor 將浮點模型轉換為一個初始的定點設計。Fixed-Point Advisor 幫助我們對程式碼的大小、記憶體使用量、以及定點縮放最佳化。我們接著利用驗

證及有效性檢測模塊組(Simulink Verification and Validation)產生模型覆蓋與迴路複雜性衡量資料來重複確認設計是否符合建模準則，並重複進行模擬。我們比較從浮點及定點模型各自產生的結果，確保在轉換過程當中沒有出錯。

我們利用軟體迴圈(software-in-the-loop, SIL)測試來驗證以 C 編程的演算法的執行情況，並以處理器迴圈(processor-in-the-loop, PIL)測試來驗證在即時硬體上的演算法。透過這樣的方式，我們能夠確保經過驗證的模型在執行時不會發生錯誤。

在軟體迴圈測試階段，我們以 Simulink Coder™從我們的 Simulink 模型中的感測器演算法元件產生 C 程式碼。接下來，我們將模型中的元件以帶有所產生的 C 程式碼的 S-function 取代，並且重新執行模擬。然後我們再一次將模擬結果與稍早產生的測試結果作比較，驗證軟體的執行。

最近我們與 MathWorks 公司負責開發了以 78K 系列的微型控制器為目標的嵌入式程式碼轉碼器的顧問合作，促成了在我們 Renesas® 78K 微型控制器的 PIL 測試。現在，我們可以利用嵌入式程式碼轉碼器產生程式碼，並且可以將這些程式碼轉檔佈署到裝置上以進行 PIL 及車輛測試。

從原型化到生產的途徑因原型化的方式而異。如果我們利用硬體描述語言轉碼器產生 FPGA 的 VHDL 程式碼，我們會將設計及所產生的 VHDL 程式碼交給外部的合作夥伴，讓他們依據原型產生 ASIC。因為我們已經徹底的驗證模型及其 HDL 執行，我們的 ASIC 需要重複的工作就大幅減少了，也能節省成本並避免計畫延遲。

但如果我們以微型控制器當作原型化的目標，則會是繼續在 Hella 內部進行生產。我們利用嵌入式程式碼轉碼器從定點 Simulink 模型產生 ANSI C 程式碼，並設定生產微型控制器目標。產生出的程式碼必須經過 Hella 嚴格的測試流程，包含利用 Polyspace Client™ for C/C++以及 Polyspace Server™ for C/C++進行整合測試與靜態分析。ANSI C 程式碼最終會以 IBM Rational Rhapsody (C/C++)整合到目標。

藉由調整程式碼生成的設定以及遵循已建立的建模準則，我們能夠產生比同等的手寫程式碼更加簡潔的生產程式碼。透過重複利用我們的原型化模型來產生生產程式碼，我們縮短了大約 60%的開發時間。再加上有了模型化基礎設計以及 HFK RCP 單元，我們可以在開發階段的初期執行測試，讓我們可以提早幾個月檢測要求的有效性及驗證設計決策。