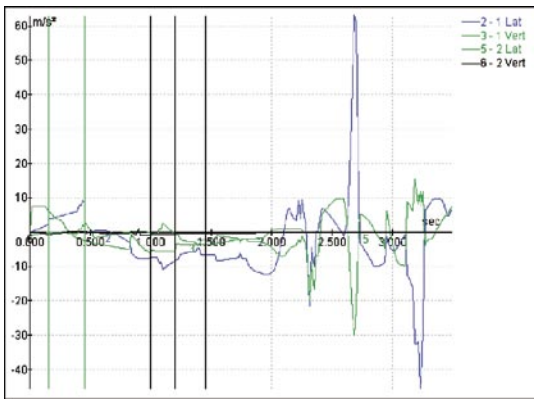
Surviving a Vehicle Rollover

Vehicle rollovers are a serious concern in automobile safety and can be fatal. They account for approximately one quarter of all road accident deaths in the US each year. Delphi's patented WinGAMR rollover detection algorithm detects a rollover and triggers protection. Delphi used model-based design to implement WinGAMR. TargetLink from dSPACE was used for automatic production code generation. TargetLink's code profiling techniques helped to analyze the code and significantly improve the efficiency of the production code.

- **Delphi's developments for passenger safety systems**
- **Rollover detection algorithm implemented and now in production**
- **Significant code improvements achieved using TargetLink's code profiling techniques**



▲ *Simulated sensor signals show the lateral and vertical accelerations to be analyzed by a rollover detection algorithm.*

Background

Rollovers are complex events. They can have several causes, such as over correction by a drowsy or distracted driver. They can also occur when a vehicle loses positive traction, skids sideways, and encounters a low obstacle. Vehicle-to-vehicle collisions can also cause rollovers.

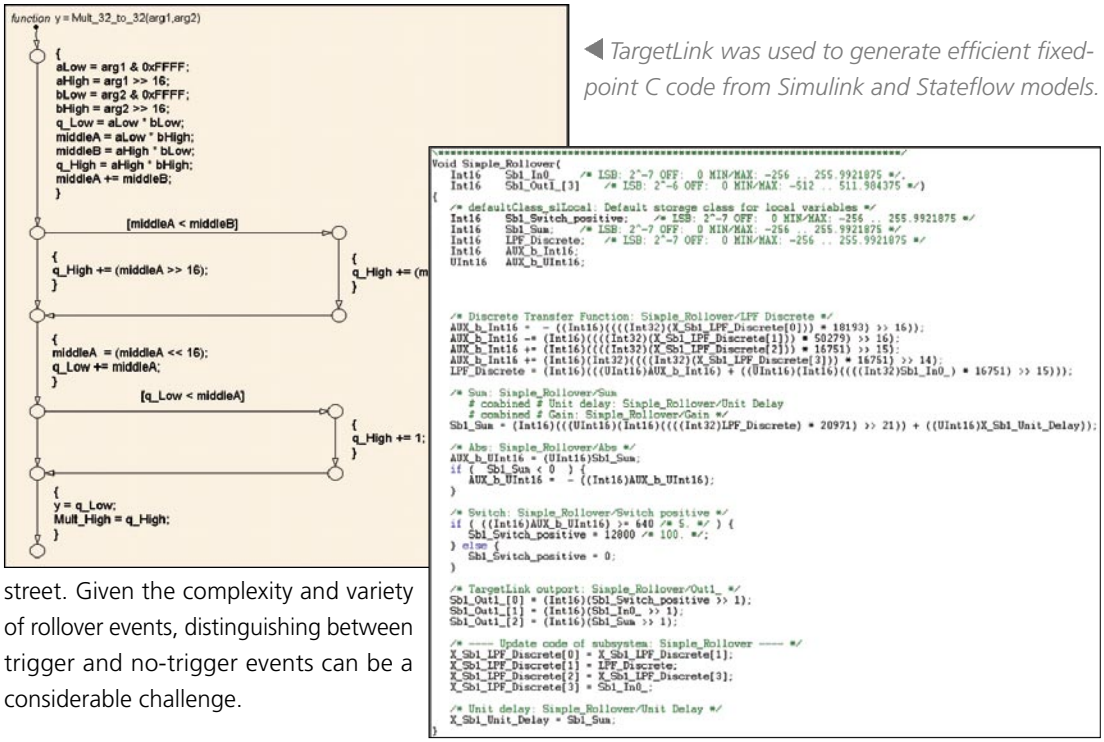
Protecting Vehicle Occupants

With front airbags now covering the driver and passenger, and with side airbags available on many vehicles, rollovers are the next big market for crash safety systems. There are three common ways of protecting vehicle occupants: removing seat belt slack (actively tensioning the seat belt) to reduce occupant ejection; deploying head-rest roll bars for convertibles; and side window curtain airbags to reduce head injury and prevent ejection. Post-rollover actions may also be required, such as cutting off the fuel flow and sending a distress call to the emergency services.

Rollover Detection

Knowing when to trigger such safety measures is the task of a rollover detection algorithm. Data from inertial sensors such as gyros and accelerometers is typically processed in a rollover-sensing module to make the trigger-or-not decision. The design of the rollover detection algorithm has to cover a very wide, dynamic range of events, from a gradual drift into a highway ditch to a rapid curb trip on a city

▲ *In situations like these, the WinGAMR rollover detection algorithm triggers safety measures.*



◀ *TargetLink was used to generate efficient fixed-point C code from Simulink and Stateflow models.*

street. Given the complexity and variety of rollover events, distinguishing between trigger and no-trigger events can be a considerable challenge.

Design

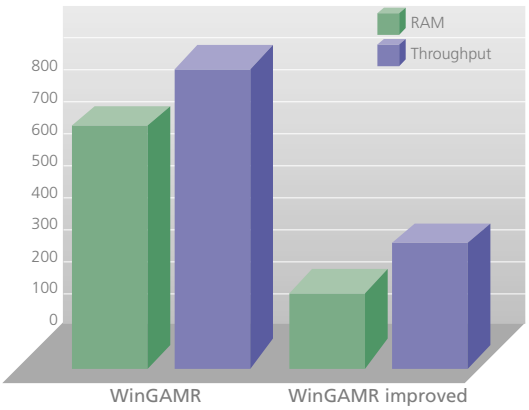
Validating rollover detection algorithms and performing tolerance studies on their calibration to a specific vehicle platform are greatly facilitated by math-based design methods. Once an algorithm has been validated, it is ready to be autocoded for implementation on a fixed-point microprocessor. To ensure the reliability of the final system, it is essential to verify that the C code accurately represents the same performance as the math models.

Project and Process

The patented WinGAMR algorithm, invented at Delphi Electronics & Safety, was first implemented using Simulink® and Stateflow® models. TargetLink was then used to develop the range and resolution for each variable, and to generate code. The fixed-point target C code was compared with the original model to clarify the tradeoffs involved in memory and throughput, and to optimize the system. Comparing target C code (PIL simulation) to the original validation file (MIL simulation) then verified the performance of the final product.

Profiling and Optimization

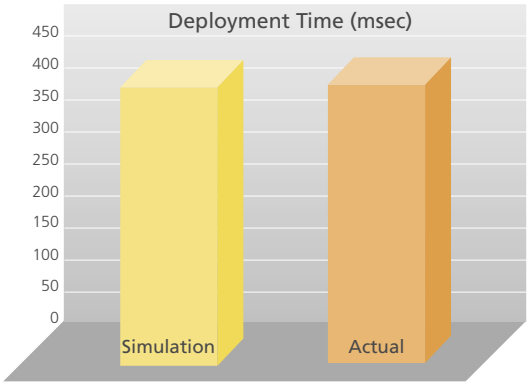
Using the execution time metric produced with TargetLink and a target processor evaluation board, a sub-optimal implementation of the algorithm was identified. Within minutes, the cause of the high RAM and throughput was evident. By making a very modest change in the algorithm implementation, both RAM and throughput were reduced by 75%. This remarkable improvement may not have been noticed for weeks or months using the traditional approach. Bringing the consequences of algorithm implementations to the awareness of the algorithm designer enabled quick cycles of learning that led to rapid improvement in performance. Besides speeding up the time to market, the wasteful expenditure of a considerable amount of engineering effort was entirely avoided.



▲ *Insights into the algorithm gained from using TargetLink's code analysis tools and an evaluation board led to huge improvements in RAM consumption and throughput (execution time).*

Implementation

The auto-generated code was passed to the software integration engineer. Integration took less than 2 days. His comments were: *"The generated code was easy to understand. Every comment and variable name was a real help, and in my opinion it saved a lot of time. It is a good base for developing target C code. The main backbone of code was almost unchanged."* The word "almost" is a reference to a digital high-pass filter that used 16-bit variables. The software integration engineer expanded these to 32-bit variables, which improved accuracy in the final results.



▲ *The deployment time measured for the autocoded algorithm matched the simulation.*

Handcode vs. Autocode

The autocoded algorithm was a new development; so direct comparisons with handcode are not available. However, an earlier handcoded version of the rollover detection algorithm consumed more throughput, more RAM, and more ROM than the autocoded algorithm. The autocoded algorithm also improved field performance. This comparison involves other factors, but it can be stated that autocode did not cause any undesirable metrics in the newly-created rollover detection algorithm.

Live Testing

Live testing of the algorithm was performed to compare results to predictions. Three tests were run. The two near-rollover tests were successful, and did not trigger. The measured deployment time for the rollover test matched the simulation exactly (10 ms sampling rate). Moreover, a professional racing driver was unable to produce any undesirable behavior on a rigorous test course. The results were a success and this algorithm is now in production.

Future Outlook

Further development of rollover detection algorithms continues to use MATLAB/Simulink/Stateflow for development and calibration, and TargetLink for automatic production code generation. The re-use blocks from WinGAMR allow the next generation of rollover detection algorithms to be developed much more rapidly and efficiently.

Peter J. Schubert, PhD
Technical Fellow
Systems Methodology Advocate
Delphi Electronics & Safety
USA

Essence

Autocode Success Story

- Code-profiling led to handcode efficiency; RAM and throughput down by 75%
- Integrated in 1.5 days
- In production
- Statements from software engineers:
  "The generated code was easy to understand. Every comment and variable name was a great help."
  "In my opinion it saved a lot of time. It is a good base for developing target C code. The main backbone of code was almost unchanged."

Equipment and Methods

- TargetLink, Target Optimization Module and Motorola HC12 Evaluation Board for code profiling
  – Throughput (execution time)
  – RAM (including stack)
  – ROM
- Back-to-back-tests (MIL, SIL and PIL simulations) at earliest stage

Autocode Reduces Risks

- No transcription errors
- No misinterpretation of specifications
- Match with model performance