Übersetzen und Optimieren von Simulink Modellen

Markus Degen
Universität Freiburg
degen@informatik.uni-freiburg.de

Zusammenfassung

Simulink ist ein Tool zur Erstellung und Simulation von Modellen für z.B. Regelungssysteme. Diese graphisch erstellten Modelle sollen für eingebettete Systeme übersetzt und optimiert werden. Dies ist ein Arbeitsbericht über einen Compiler für Simulink Modelle, der momentan implementiert wird.

Schlüsselwörter: Compiler, Simulation, Optimierung.

1 Einleitung

Simulink ist ein anerkanntes Tool um Modelle, besonders für eingebettete Systeme, zu entwickeln und zu simulieren.

Die so entwickelten Systeme sollen für eingebettete Controller übersetzt werden, so dass die entworfenen Modelle nicht von Hand nochmals implementiert werden müssen. Besonders wichtig hierbei ist eine effiziente Übersetzung, die den Anforderungen einzelner eingebetteter Systeme angepasst werden kann. Neben der klassischen Laufzeitoptimierung spielen hierbei Speicherknappheit, Energieeffizienz und die Codegröße eine wichtige Rolle.

2 Simulink

Simulink[2] ist ein graphisches Entwicklungstool von MathWorks, das auf der mathematischen Umgebung Matlab aufsetzt. Mit Simulink werden in vielen Bereichen Regelungs-, Signalverarbeitungs-, Kommunikations- und andere Systeme entworfen. Die Modelle werden dabei graphisch durch Zusammenfügen verschiedener Blöcke und deren Verbindungen erstellt.

Die so entworfenen Modelle werden dann mit Hilfe des darunter liegenden Matlab simuliert. Ihr Verhalten kann also beobachtet und das System entsprechend der Anforderung angepasst und korrigiert werden.

Zur Erstellung komplexer Systeme steht dem Benutzer eine große Bibliothek vorgefertigter Blöcke zur Verfügung. Ein Block stellt dabei eine Operation auf den ihm zur Verfügung stehenden Signalen dar. Neben einfachen algebraischen Blöcken, wie z.B. der Addition zweier Eingänge, gibt es Speicher- bzw. Verzögerungsblöcke, logische Blöcke zur Berechnung einfacher logischer Operationen (z.B. AND) und Steuerblöcke, die andere Blocke ein- und ausschalten bzw. Signale auswählen. Zusätzlich können all diese Blöcke hierarchisch zusammengefasst werden. Die Blöcke werden mit Signalleitungen verbunden. Diese haben unterschiedliche Datentypen. Neben den Standardtypen `double`, `integer` und `boolean` gibt es Fixpunkttypen, aber auch Vektoren verschiedener Typen sind zulässig. Simulink inferiert die Typen so weit wie möglich selbst, für unspezifizierte Typen, die nicht aus anderen Bedingungen inferiert werden können, wird `double` als Standardtyp fest gelegt.

Die Simulation erfolgt grundsätzlich auf einem kontinuierlichem Zeitmodell, d.h. alle Signalleitungen haben zu jeder Zeit ein definiertes Signal anliegen. Für die Simulation werden die Werte natürlich mit möglichst kleinem Delta der Zeit berechnet. Insgesamt ist diese kontinuierliche Darstellung jedoch meist nicht nötig. Daher gibt es in Simulink eine Gruppe von Blöcken, die ein diskretes Verhalten simulieren. Da generell nur Einzelwerte berechnet werden können, beschränken wir die Übersetzung auf solche diskrete Blöcke. Zusätzlich wird zur Vereinfachung die Zeit stets um einen konstante Wert erhöht. Dies kann in Simulink mit der Einstellung auf 'fixed-step' direkt mit angegeben werden.

Einzelne diskrete Blöcke können jedoch dennoch eine unterschiedliche Abtastrate haben, d.h. ein Block berechnet unter Umständen nicht zu jedem Zeitschritt seinen Ausgang neu. Da die Signale aber prinzipiell kontinuierlich sind und keine Leitung einen undefinierten Wert haben soll, halten diese Blöcke ihr zuletzt berechnetes Ergebnis am Ausgang solange vor, bis ein neuer Wert errechnet und angelegt wird.

Schleifen in dem Signalfluss werden nur dann zugelassen, wenn auf dem Weg ein Verzögerungsblock liegt, sogenannte algebraische Schleifen sind also ausgeschlossen. Diese algebraischen Schleifen führen zu Berechnungs- und Definitionsproblemen die durch ihren Ausschluss vermieden werden. Auch diese Einschränkung wird von Simulink direkt unterstützt und sollte auch unabhängig der Übersetzung beachtet werden.

Eine Hauptproblem der Übersetzung liegt in der mangelhaften Spezifikation von Simulink. Leider wird keine vollständige formale Definition gegeben. Lediglich Beschreibungen der gewünschten Verhalten aus den Hilfedateien können herangezogen werden. Generell soll die Übersetzung keine Abweichung zu der Si-

mulation von Simulink aufweisen. Dies formal zu zeigen scheitert aber an der fehlenden Spezifikation.

3 Übersetzung

Um die Übersetzung zu vereinfachen, werden nur solche Modelle angenommen, die in Simulink akzeptiert und simuliert werden. Zusätzlich muss darauf geachtet werden, dass keine algebraischen Schleifen vorhanden sind, und die Simulation mit fester diskreter Schrittweite der Zeit erfolgt. Für diese Punkte gib es entsprechende Voreinstellungen in Simulink, die für die Übersetzung vorausgesetzt werden.

Die Menge der übersetzten Blöcke ist entsprechend auf den diskreten Teil der Bibliothek ein zu schränken. Zu Beginn wird auch diese Menge auf die Basisblöcke beschränkt, weitere Blöcke werden aber nach und nach folgen.

Vor der Übersetzung wird eine Typinferenz auf den Signalleitungen durchgeführt. Die Typen der Leitungen werden entweder direkt vom Benutzer angegeben, oder sie ergeben sich aus den Blöcken und den dazugehörigen Typen. Alle nicht auf diese Weise getypten Signale werden, wie in Simulink auch, auf `double` als Standardtyp gesetzt.

Die Schrittweite der Zeit errechnet sich für das Modell aus den einzelnen Abtastraten der beteiligten Blöcke. Hierfür wird der ggT aller Blöcke berechnet. Dies ist möglich, da lediglich Integerwerte für die Abtastraten zulässig sind. In der Praxis liegt im Regelfall eine Schrittweite von 1 vor.

Die Initialisierung erfolgt durch die Initialisierungswerte der entsprechenden Blöcke. Alle Verzögerungs-, Speicher- und Konstantenblöcke haben einen Initialwert. Dadurch haben alle Leitungen die benötigt werden zu Beginn einen definierten Wert anliegen. Die Initialwerte müssen ggf. vor Simulationsstart durch algebraische Blöcke propagiert werden.

Durch den Ausschluss von algebraischen Schleifen muss jeder Block pro Zeitschritt nur genau ein Mal berechnet werden. Die Reihenfolge der Berechnung kann statisch durch eine topologische Sortierung festgelegt werden.

Je nach Abtastrate und aktueller Zeit berechnen die einzelnen Blöcke ihren Ausgang neu oder lassen das alte Ergebnis anliegen. Es sind, je nach Anforderung an das System, zwei verschiedene Übersetzungen für die Abtastrate vorgesehen: Zum einen kann ein Zähler hochgezählt werden, die Berechnung des Ausganges erfolgt dann bei einem definierten Wert; oder die globale Zeit wird modulo der Abtastrate betrachtet, eine Berechnung erfolgt, falls der modulo gleich Null ist.

Jeder Speicher-, und Verzögerungsblock benötigt schon durch seine Definition einen Speicherplatz um die abgelegten Signale halten und wieder ausgeben zu können. Wiederum wegen der Abtastraten muss aber auch für algebraische Blöcke eine Speicherzelle vorgehalten werden. Solange der Ausgang nicht neu berechnet wird, muss der Inhalt der Speicherzelle ausgegeben werden.

4 Optimierung

Zu der klassischen Laufzeitoptimierung kommen bei eingebetteten Systemen Anforderungen bezüglich Speicherbedarf, Energiebedarf und Codegröße. Die Anforderungen können hierbei von Controller zu Controller variieren. Daher soll die Optimierung der Übersetzung auch entsprechend dieser Anforderungen angepasst werden können.

Es gibt eine Reihe von möglichen Optimierungen. So können z.B. algebraische Blöcke, die in dem Modell hintereinander stehen, zusammengefasst werden. Die für die Abtastrate eingebauten Zähler können in vielen Fällen wieder entfernt werden. Dazu sollen auch Blöcke mit gleicher Abtastrate so weit wie möglich gruppiert werden. Zusätzlich können bekannte Verfahren aus der partiellen Auswertung eingesetzt werden.

5 Weitere Arbeiten

Das Tool S2L[3] übersetzt Simulink Modelle mit ähnlichen Einschränkungen nach Lustre. Während S. Tripakis et al. auf eine Übersetzung setzen, in der das System ggf. verifiziert und geprüft werden kann, soll der Schwerpunkt unseres Übersetzers auf der Optimierung des compilierten Codes liegen.

MathWorks selbst stellt einen kommerziellen Übersetzer[1] nach C für Simulink zur Verfügung.

Literatur

[1] The MathWorks. Real-time workshop 6.4.1. `http://www.mathworks.com/products/rtw/`, May 2006.

[2] The MathWorks. Simulink. `http://www.mathworks.com/products/simulink/`, May 2006.

[3] S. Tripakis, C. Sofronis, P. Caspi, and A. Curic. Translating discrete-time simulink to lustre. *Trans. on Embedded Computing Sys.*, 4(4):779–818, 2005.