Einbindung eines generischen Mikrocontrollers inklusive analoger und digitaler Peripherieelemente in einer FPGA-basierten Hardware-in-the-Loop Simulationsumgebung für Leistungselektronik.

Hersteller von Leistungselektronik (kurz LE) modellieren und simulieren die Hardware (bspw. Schaltnetzteiltopologien wie einen Abwärtswandler) und die Software (bspw. Regelungskonzepte), welche auf einem Mikrocontroller (kurz µC) ausgeführt wird, um damit früh in einem Projekt Schwächen oder Fehler eines Konzeptes zu detektieren und gegebenenfalls zu beheben. Neben den daraus folgenden Kostenersparnissen ermöglicht eine Modellierung des Systems, Toleranzen von elektronischen Bauteilen und Temperatureinflüssen ohne großen Aufwand zu berücksichtigen. Die entwickelten Modelle können darüber hinaus zur Optimierung oder Fehlersuche von Prototypen / Produkten verwendet werden, da interne Systemgrößen im Modell ohne Probleme gemessen werden können. Zudem wird durch die Modellierung des Systems eine parallele Entwicklung von Hardware und Software ermöglicht, was zu einer drastischen Reduzierung der Entwicklungszeit und somit der Entwicklungskosten führt. Um all die erwähnten Vorteile nützen zu können, müssen die Modelle der Hardware und der Software möglichst detailgetreu und zueinander kompatibel sein.

Stand der Technik

Die vorhandenen Produkte und die Ansätze in der Literatur können nicht verwendet werden, da sie mindestens eines der folgenden Probleme haben:

- Sie unterstützen weder Simulink noch PLECS (bestehende Simulationsmodelle können nicht genützt werden).
- Sie berücksichtigen die Peripherie des µC nicht.
- Es können nur bestimmte Mikrocontroller verwendet werden.
- Es sind viele Arbeitsschritte nötig, um eine Simulation zu starten.
- Die Simulationsgeschwindigkeit ist sehr gering.
- Die Kosten für die Simulationsumgebung (Hardware- und Lizenzkosten) sind zu hoch, um die Simulationsumgebung mehrfach einzusetzen.

Lösungsansatz

Im gewählten Konzept (illustriert in Abbildung 1) wird der µC-Prozessor inklusive der digitalen Peripherie auf einem FGPA oder SoC modelliert. Über ein Bus-System tauscht der Prozessor Daten mit der entwickelten digitalen Peripherie aus. Die Kommunikation zwischen dem FPGA und Simulink, auf der die LE und die analogen Peripherien modelliert werden, ist über die PCIe-Schnittstelle und einem S-Function-Block in Simulink realisiert worden. Wie im Hardware-in-the-Loop Ansatz, wo die LE in Simulink ausgeführt wird, generiert Simulink den Takt für den µC-Prozessor und die digitale Peripherie. Aufgrund des variablen Solvers entsteht ein variierender Takt. Der gewählte Prozessor und die Peripherie sollten deshalb unempfindlich gegenüber Taktänderungen sein.

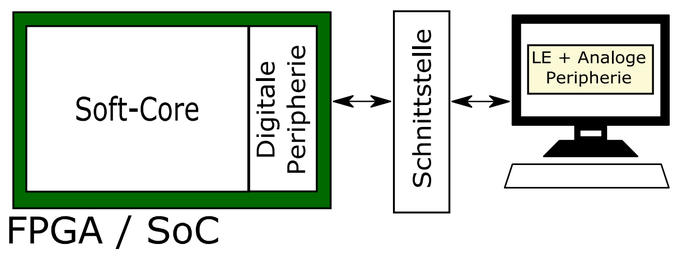

Abbildung 1: Konzeptblockschaltbild

Soft-Core

Der NIOS II/e von Intel wurde als Soft-Core IP ausgewählt. Tests mit diesem Prozessor zeigten, dass der NIOS II/e unempfindlich gegenüber einer Variation der Taktfrequenz ist. Neben dem Prozessor wird ein RAM benötigt, auf dem die abzuarbeitende Software platziert wird. Über das Bus-System (Avalon-MM) können der RAM und selbst generierte digitale Peripherien angeschlossen und verwendet werden.

Analoge Peripherie

Die Modellierung der analogen Peripherie geschieht auf der Funktionsebene. Als Modellierungsumgebung wurde PLECS ausgewählt, in der mittels C-Script Blöcken ein Endlicher Automat (kurz FSM) aufgebaut wird. Mit dieser FSM wird das Verhalten eines Komparators (kurz COMP) und eines Analog-Digital-Umsetzers (kurz ADC) nachgebildet. Die entwickelte analoge Peripherie behandelt statische und dynamische Signale / Parameter. Dynamische Signale / Parameter haben einen Signaleingang in das Modell, um direkt in die FSM einzuwirken. Die statischen Parameter können von Hand oder mit einem entwickelten Prozess automatisch aus den vorhandenen Peripherie-Konfigurationen in die Eingabemaske geladen werden.

Digitale Peripherie

Eine spezifische Komponente wie ein Timer besitzt einen Aufbau wie in Abbildung 2. Die Logik, die das Verhalten der Timer beschreibt (Task Logic), wurde mittels SystemVerilog in Form einer Verhaltensbeschreibung entwickelt. Damit die Timer mit dem Soft-Core kommunizieren können, sind ein Register-File (Register file) und eine Avalon-Schnittstelle (Avalon Slave Interface) nötig. Das Register-File beinhaltet für jedes Register des Timers eine spezifische Adresse (in Form eines Offsets). Die Avalon-Schnittstelle wandelt die Signale vom Avalon-Bus so um, dass die Register des Timers beschrieben oder gelesen werden. Durch diesen modularen Aufbau ist es möglich, dass eine spezifische Peripherie mit einem anderen Prozessor kommunizieren kann, indem die Avalon Slave Interface Komponente ausgewechselt wird.

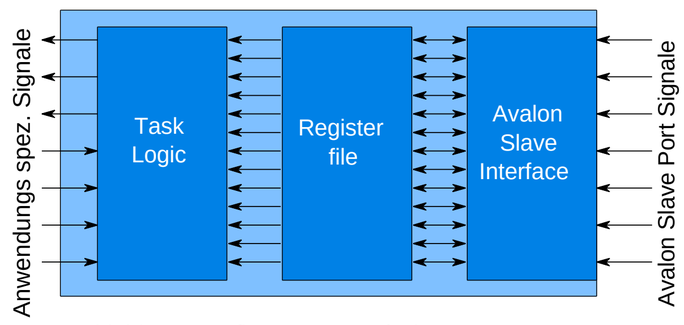

Abbildung 2: Aufbau einer spezifischen Komponente

Verifikation

Als Modell für die LE wird ein Abwärtswandler in PLECS entwickelt. Der Mittelwert des Laststroms soll durch das Setzen des maximalen Spitzenstroms in der Induktivität (iL_ipeak in Abbildung 3) oder durch die Dauer der Öffnung des Schalters (tdead) vom Benutzer bestimmt werden.

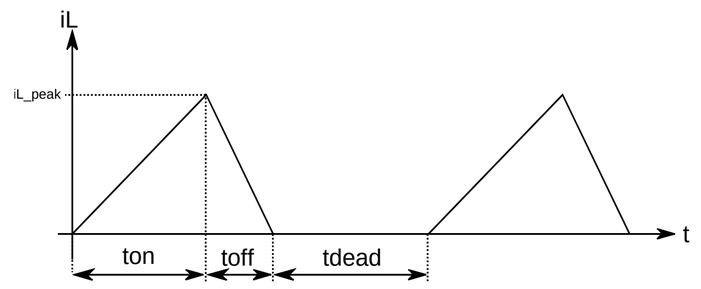

Abbildung 3: Gewünschter Stromverlauf durch die Induktivität

Der modellierte COMP gibt den zwei implementierten Timern die Steuersignale, um entsprechend den Benutzereingaben den Schalter anzusteuern. Abbildung 4 zeigt den Ausgangsstrom bei verschiedenen iL_ipeak-Werten (Wert vor dem /) und verschiedene tdead-Werte (Wert nach dem /).

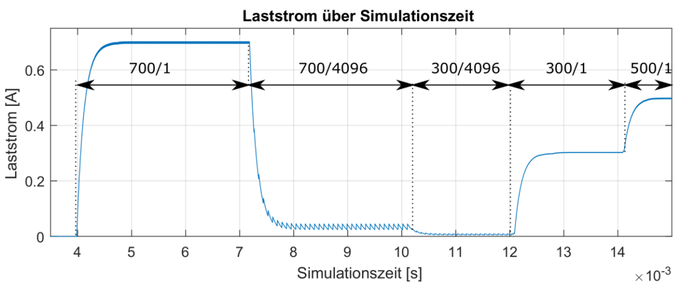

Abbildung 4: Ausgangsstrom

Master
Mechatronics
2017

Studierender
Stefan Stark, BSc.

Betreuende Lehrperson der FHV
Dr. Andre Mitterbacher

Auftraggeber
Mathias Dünser, BSc.
Tridonic GmbH & Co KG

FH Vorarlberg
University of Applied Sciences