Forschungsbericht SS 2010

Embedded Linux auf FPGA-basierten Systemen mit freien Prozessor-IPs (2. Teil)

Unterthema: Booten von Embedded Linux-Systemen

Prof. Dr.-Ing. Rainer Bermbach

Einleitung

Auf allen Computern muss nach dem Einschalten zuerst die Hardware (Prozessor, Speichercontroller, Peripherie) initialisiert werden. Daran schließt sich das Laden eines ersten Programms an, das typischerweise weitere Software, wie z.B. das Betriebssystem aus nicht-flüchtigen Speichern (im PC die Festplatte) lädt. Das Betriebssystem startet dann die eigentliche Anwendung. Dieses Hochfahren des Systems nennt man *booten*, die Programme, die das Betriebssystem (oder manchmal direkt die Applikation) laden *Bootloader*. Auch Embedded Systeme benötigen diese Funktionsschritte und verwenden daher häufig ebenfalls einen Bootloader, insbesondere, wenn ein Betriebssystem wie Linux gestartet werden soll.

Im PC übernimmt das sog. BIOS die Hardwareinitialisierung und lädt einen ersten Teil des Bootloaders aus dem Master Boot Record der Festplatte, der seinerseits den eigentlichen Loader von der Platte startet. Unter Windows tritt der Bootloader meist nicht in Erscheinung, nur bei Systemen mit mehreren Betriebssystemen. Unter Linux kennt man z.B. GRUB (Grand Unified Bootloader) oder LILO (Linux Loader), die das Betriebssystem oder eines der Betriebssysteme hochfahren.

In Embedded Systemen mit Betriebsystem liefern manchmal die Board-Hersteller entsprechende Bootsoftware mit, ansonsten muss man sie selber erstellen. Dies kann sehr aufwändig sein, da detaillierte Hardwarekenntnisse des Boards und ein tiefes Verständnis der Betriebssystemsoftware von Nöten sind. Jede neue Hardware erfordert die erneute Durchführung dieses Entwicklungsprozesses.

Im Rahmen des Projektes wurde der Bootprozess für Embedded Linux-Systeme auf FPGA-basierten Systemen untersucht. Standardmäßig war bislang das Booten von Linux-Images nur über einen laufenden Debugger interaktiv möglich. Um die Embedded Systeme aber autonom starten zu können, wird ein Bootloader nötig, damit ohne Entwicklungssystem und ohne Benutzerinteraktion das Betriebssystem und die Anwendung gestartet werden können. Der Vorteil der FPGA-basierten Systeme liegt in ihrer leichten Anpassbarkeit an den aktuellen Hardwarebedarf eines konkreten Projekts. Neue Hardwarekomponenten bedeuten aber immer Anpassungen auch in der Software. Ziel war es daher, ein möglichst allgemeines Vorgehen für das Booten von Linux zu entwickeln, das den Adaptionsaufwand für neue Hardware in Grenzen hält bzw. so weit wie möglich minimiert. Da auf den betrachteten FPGA-basierten Systemen verschiedenste Prozessoren und Peripheriekomponenten zum Einsatz kommen können, ist dieses allgemeine Vorgehen hier besonders notwendig und hilfreich. Beispielhaft wurde der Bootprozess für die Architekturen PowerPC405 und MicroBlaze auf unterschiedlichen FPGA-Boards untersucht und implementiert.

Einsatz des Bootloaders U-Boot

Untersucht man verfügbare Bootloader, so stößt man schnell auf das Projekt „Das U-Boot“, einen Open Source Bootloader für verschiedene Prozessorarchitekturen. Obwohl sehr viele Bootloader existieren, sind diese zum größten Teil sehr speziell und teilweise auch nicht für die benötigten Prozessorarchitekturen verfügbar. Eine Portierung eines Bootloaders auf eine neue Architektur schied aus Aufwandsgründen aus. Das U-Boot unterstützt dagegen neben vielen anderen auch die Prozessoren PowerPC 405 und MicroBlaze der FPGA-Boards und ermöglicht es, mit überschaubarem Aufwand ein Bootprogramm zu erhalten, sofern das jeweilige Board und seine Hardware von „Das U-Boot“ unterstützt werden.

Eine genauere Untersuchung von U-Boot zeigte, dass die beiden zur Verfügung stehenden FPGA-Entwicklungsboards XUPV2P und XUPV5 zwar nicht (mehr) explizit im U-Boot-Entwicklungszweig aufgeführt sind, dort aber erfreulicherweise im Rahmen von Vereinfachungen und Vereinheitlichungen durch Prozessorspezifische, generische Dateien (microblaze-generic/ppc405-generic) unterstützt werden.

Für einen autonomen Bootvorgang muss die Hardware normalerweise über einen vom Prozessor zum Startzeitpunkt verfügbaren Festwertspeicher (ROM, EPROM, Flash) verfügen. Dies ist beim XUPV2P nicht der Fall. Das XUPV5 verfügt über einen NOR-Flash-Speicher, der prinzipiell als Bootspeicher verwendbar ist. Beide Boards sind mit sog. Platform Flashs ausgestattet, die aber nur die sog. Bit Files für die Hardwarekonfiguration aufnehmen; als Bootspeicher sind sie ungeeignet. Beide Boards besitzen noch ein Interface für CompactFlash-Karten (CF Cards), die als Festplattenersatz größere Datenmengen aufnehmen können. Ein Boot daraus ist jedoch nicht direkt möglich. Der auf den Boards vorhandene SystemACE Controller erlaubt es aber (s.u.), von einer CF Card sowohl die Konfiguration des FPGAs (Hardware) als auch Software für den Prozessor zu laden, wobei letztere im RAM des Boards gespeichert wird. Dadurch wird ein Boot aus dem RAM ermöglicht. Im Sinne einer Verallgemeinerung der Vorgehensweisen wurde auf ein Booten aus dem NOR Flash des XUPV5 verzichtet und für beide Boards das Starten aus der CompactFlash-Karte untersucht.

Um mit den Boards und beiden Prozessorarchitekturen U-Boot/Linux starten zu können, sind bestimmte Hardwarevoraussetzungen nötig. So müssen neben dem Prozessor mit Cache, MMU etc. u.a. die Speichercontroller, der SystemAce Controller, serielle Schnittstelle und Timer konfiguriert werden. Die gesamte genutzte Peripherie muss interruptfähig sein. Für den Zugriff auf die CF Card mit U-Boot muss auch ein Barrel Shifter aktiviert werden. Mit dem PowerPC405 auf dem XUPV2P gibt es allerdings Schwierigkeiten. Der PPC405 besitzt keinen Barrel Shifter, weshalb beim Betrieb von U-Boot zurzeit kein Laden von Programmen aus der CompactFlash-Karte möglich ist. Vermutlich handelt es sich noch um eine Inkompatibilität in der Toolchain, die noch beseitigt werden muss.

Die Hardware eines Systems muss wie üblich (vgl. WS 2009/2010-1) erstellt werden und die Konfiguration für das FPGA als sog. Bit File zur Verfügung stehen.

Um U-Boot (und später Linux) Board- und Prozessorspezifisch erstellen zu können, muss auch die sog. Toolchain (Compiler etc., vgl. WS 2009/2010-2) vorhanden sein bzw. zugeschnitten installiert werden. Diese wird für den PowerPC 405 mittels CrossTools NG erstellt und, da CrossTools NG den MicroBlaze derzeit nicht unterstützt, für den MicroBlaze die vorbereitete Toolchain von Xilinx eingesetzt.

Damit U-Boot auf die benötigte Hardware zugreifen und sie z.B. vorab initialisieren kann, muss die (genutzte) Hardwareausstattung natürlich U-Boot bekannt gemacht werden. Hierfür existiert ein Programm, das in der Version 10.1.3 der Xilinx-Entwicklungsumgebung eingebunden wird und dort die benötigten Parameterdateien mit den jeweiligen Hardwareparametern (xparameters.h, config.mk) nach der Hardwareerstellung automatisch erzeugt. In neueren Versionen (z.B. 12.3) ist das Programm nicht einsatzfähig. (Die alte Version der Entwicklungsumgebung ist nach wie vor wichtig, weil das XUPV2P von neuen Versionen nicht mehr unterstützt wird.) Da das Ein-

binden des Skripts immer etwas Aufwand bedeutet, es ohnehin in neueren Versionen nicht einsetzbar ist und die benötigten Hardwareinformationen eh bei der Hardwareerstellung eingegeben werden müssen, wurde letztlich entschieden, im Sinne einer allgemeinen Vorgehensweise die beiden benötigten Dateien xparameters.h und config.mk aus vorbereiteten Templates jeweils manuell aufzubauen. Der Aufwand hält sich in Grenzen, da nur Änderungen der Hardware berücksichtigt werden müssen und daher nur geänderte Adressen eingetragen, entfallene Hardware auskommentiert und lediglich völlig neu hinzu gekommene Komponenten einmalig im Template nachgetragen werden müssen.

Diese Parameterdateien sind Voraussetzung, damit U-Boot compiliert werden kann. Sie müssen an die richtige, von der Prozessorarchitektur (PowerPC405, MicroBlaze) abhängige Stelle des vorher installierten U-Boot-Softwarebaums (/board/xilinx/microblaze-generic/… bzw. board/xilinx/ppc405-generic/…) kopiert werden. In einer weiteren Datei /include/configs/microblaze-generic.h bzw. /include/configs/xilinx-ppc405-generic.h, prozessorabhängig) müssen ebenfalls Einträge vorgenommen werden, beispielsweise Zuordnung der Konsole zur seriellen Schnittstelle, benötigte Umgebungsvariablen, IP- und MAC-Adresse, (Boot-) Skripte und die Aktivierung der Monitorbefehle, die in U-Boot zur Verfügung stehen sollen. Da diese Änderungen auch meist einmalig und sonst allenfalls geringfügig sind, wurde auf die Entwicklung eines Skriptes o.ä. verzichtet. Mit der Compilierung von U-Boot entsteht eine ausführbare Datei (u-boot.elf), die eigentliche Bootdatei.

Zugriff auf die CompactFlash-Karte

Was hier vergleichsweise einfach klingt, war in der Realität eine aufwändige Untersuchung und Entwicklung, die häufig in Sackgassen führte. Beispielsweise musste einiges „Lehrgeld“ gezahlt werden, bis klar war, dass U-Boot an das Ende des Adressbereiches geladen werden muss. Anderenfalls überschreibt ein zu ladendes Linux-Image U-Boot und das gesamte System stürzt ab.

Ein anderes Problem stellte das Laden aus der CompactFlash-Karte dar. Wie schon erwähnt, besitzen die Boards ein Interface für CF Cards, mit dessen Hilfe die Hardwarekonfiguration für die FPGAs (Bit Files) geladen werden können. Nach einigen Untersuchungen fand sich das Skript *genace*, das aus einer Datei einen sog. ACE Container erzeugt. Dazu wandelt das Skript die Datei in eine SVF-Datei (Serial Vector Format), die über die JTAG Chain der Xilinx Boards in die entsprechenden Bauteile geladen werden kann. Weitere Untersuchungen ergaben, dass dieses Skript auch dazu verwendet werden kann, das Bit File für das FPGA (*.bit) mit der Bootdatei u-boot.elf in einen ACE Container (eine gemeinsame Datei) zu packen. Vorab müssen in der Skriptdatei *genace* einmalig einige Parameter mit Board-spezifischen Daten (FPGA-Typ, JTAG-Chain-Position …) gesetzt werden. Der SystemAce Controller, der die Bedienung des CF Card Interface managet, ist in der Lage, die Dateien im ACE Container unterschiedlich zu behandeln. So liest er von der CF Card den ACE Container und lädt das Bit File darin in das FPGA und die Bootdatei an eine angegebene Adresse im RAM, die Startadresse des Bootloaders U-Boot.

Nach dem Restart des FPGA Boards lädt der ACE Controller die ACE-Datei, konfiguriert das FPGA und lädt die U-Boot-Datei in den Speicher (RAM) und startet sie. Dann wartet U-Boot auf weitere Kommandos bzw. kann automatisch z.B. vom Netzwerk oder von der CF Card (derzeit nur beim MicroBlaze) ein Linux-Image nachladen und starten.

Boot direkt mit ACE File

Die mit *genace* erzeugten ACE Container können auch auf andere Weise genutzt werden. Da eine in einen ACE Container gepackte, ausführbare Datei ins RAM geladen und gestartet wird, ist dieser Weg nicht nur für den Bootloader U-Boot einsetzbar. Prinzipiell können beliebige Programme so gestartet werden, sofern sie in der Lage sind, die Hardware passend anzusprechen bzw. zu initialisieren. Dies nutzt Xilinx für einen sog. Wrapper, ein Rahmenprogramm, das ein erstelltes Programm, insbesondere ein Linux-Image „verpackt“ (wrapped). In einen ACE Container

gepackt, initialisiert der Wrapper die nötigste Hardware vorab, bevor es Linux selbst startet. Linux initialisiert dann die restliche Hardware und fährt komplett hoch. Um den Wrapper nutzen zu können, muss man einen sog. Device Tree Generator in der Xilinx-Entwicklungsumgebung einbinden, der die benötigten Hardwareinformationen aus dem Design extrahiert. Zuvor trägt man noch die *bootargs* (z.B. Lage des Root File System, eingesetzte serielle Schnittstelle für die Konsole) ein. Der Device Tree Generator erzeugt ein sog. DTS File, das manuell in den entsprechenden Zweig des Linux-Verzeichnisbaums (arch/<Prozessortyp>/boot/dts/…) kopiert werden muss. Die Erstellung eines Linux-Images folgt ansonsten den schon beschriebenen Wegen (vgl. WS 2009/2010-2).

Soll U-Boot doch verwendet werden, so ist mit den entsprechenden Befehlen beim Make des Linux-Systems auch das Programm *mkimage* von U-Boot einzusetzen, was sicherstellt, dass ein sog. uImage, ein U-Boot-kompatibles Image erzeugt wird. Dieses spezielle Image enthält einen Header u.a. mit Informationen über Prozessorarchitektur, Betriebssystem, Kompressionsverfahern, Startpunkt etc. Dieses Image kann dann U-Boot automatisch oder durch Benutzerinteraktion starten.

Das vorgestellte Verfahren versagt derzeit noch seinen Dienst beim Einsatz des Softcore-Prozessors Leon3 (SPARC-kompatibel), da noch kein Weg gefunden wurde, die dort entstehenden ELF Files in einen ACE Container zu packen. Dies bleibt zukünftigen Arbeiten vorbehalten.

Zusammenfassung

Im Rahmen des Projektes erfolgte nach einer generellen Analyse der benötigten Schritte im Bootvorgang von Linux-Systemen eine Untersuchung von verschiedenen Bootloadern und ihrer Arbeitsweise, insbesondere des Bootloaders „Das U-Boot“. Da dieser für alle verwendeten Prozessoren geeignet ist und universell einsetzbar schien, wurde dieser Ansatz weiterverfolgt. Es folgte die Untersuchung und mögliche Implementierungen des Bootens auf den beiden FPGA Boards mit den beiden Prozessorarchitekturen PowerPC405 und MicroBlaze mit unterschiedlichen Entwicklungssystemversionen. Der Leon3 konnte noch nicht tiefer untersucht werden.

Da ein Zugriff auf die CF Card über U-Boot beim PowerPC (noch) nicht möglich ist, wurde die Einsatzmöglichkeit des Wrappers in der Xilinx-Entwicklungsumgebung weiter untersucht und implementiert. Dieses bildet letztlich ein komfortables, allgemein einsetzbares Verfahren, das bei beiden Prozessoren auf beiden Boards und unter den verschiedenen Entwicklungssystemversionen erfolgreich funktioniert.

Kontaktdaten

Ostfalia Hochschule für angewandte Wissenschaften
Fakultät Elektrotechnik
Prof. Dr.-Ing. Rainer Bermbach
Salzdahlumer Straße 46/48
38302 Wolfenbüttel
Telefon: +49 (0)5331 939 42620
E-Mail: r.bermbach@ostfalia.de
Internet: www.ostfalia.de/pws/bermbach