Forschungsbericht WS 2009/2010

Embedded Linux auf FPGA-basierten Systemen mit freien Prozessor-IPs (1. Teil)

Prof. Dr.-Ing. Rainer Bermbach

Ein Vorläuferprojekt untersuchte die Eignung verschiedener freier IPs von 32-Bit-Prozessoren in Hinsicht auf ihre Verwendbarkeit in Embedded Prozessorsystemen basierend auf programmierbarer Logik in FPGAs. Um leistungsfähige Systeme sinnvoll realisieren zu können, muss man mit einem passenden Betriebssystem auf der Softwareseite arbeiten. Hier entwickelt sich Embedded Linux zu einem Quasi-Standard. Die Vielfalt der verfügbaren, freien Software und Applikationen ermöglicht es, selbst komplexere Systeme in überschaubarer Zeit zu erstellen.

Da die aus dem PC-Bereich bekannten Linux-Distributionen aus verschiedenen Gründen nicht eingesetzt werden können, muss man andere Wege gehen. Es existiert eine Reihe von eigenständigen Ansätzen bzw. spezifischen Distributionen, die versprechen, Embedded Systeme unter Linux zu erstellen. Daneben gibt es auch die Möglichkeit, sich selbst „sein“ Embedded Linux zusammenzustellen.

Im Rahmen des geplanten Projektes wurden verfügbare Distributionen und Vorgehensweisen auf ihre Eignung hinsichtlich der flexiblen Anpassung von Systemen untersucht und Linux auf den FPGA-Boards mit verschiedenen Prozessorsystemen implementiert.

Um ein Embedded Linux für ein Zielsystem zu erstellen, benötigt man verschiedene Werkzeuge und Programmpakete, die für das System angepasst vorliegen oder dafür adaptiert werden müssen. Die erste Voraussetzung ist eine Softwareentwicklungsumgebung (Cross Toolchain), die auf dem eigenen Rechner (Host) läuft und Code für das Zielsystem (Target) erstellen kann. Dazu gehören ein sog. Cross Compiler und weitere Werkzeuge.

Die *GNU Toolchain* umfasst neben der

- *GNU Compiler Collection* (GCC) noch
- *GNU Make* (ein Programm das sog. Makefiles – Scripts – ausführt, mit denen eine Vielzahl von manuellen Schritten automatisch durchgeführt werden können), die
- *GNU Binutils* (eine Sammlung von diversen Programmierwerkzeugen z.B. Assembler, Linker, Objdump), den
- *GNU Debugger* (GDB), eventuell um ein grafisches Frontend ergänzt, sowie das
- *GNU Build System* (Autotools), das weitere benötigte Software enthält.

Um die Cross Toolchain zu erzeugen, benötigt man zuerst einmal eine native Toolchain auf einem laufenden Linux-System (typ. PC).

Auch hier kann es schon Probleme geben, so dass je nach Distribution verschiedene Anpassungen (Patches) vorgenommen werden müssen. Auf diesen Rechner lädt man sich meist eine Skriptsammlung wie CrossTools-NG und zugehörige Pakete, mit deren Hilfe man dann die Cross Toolchain erstellt. (Das Entwickeln einer eigenen Cross Toolchain von Grund auf erfordert sehr viel Wissen und Zeit und wird daher meist nicht durchgeführt.)

Beim Aufbau der Werkzeuge ist auch zu beachten, mit welcher C-Bibliothek man arbeiten will. Die standardmäßige *glibc* ist für viele Embedded Systeme zu umfangreich, so dass andere Libraries wie die *uClibc* Verwendung finden

Der nächste Schritt erzeugt ein sog. *Kernel Image*, das einen zugeschnittenen Linux-Kern enthält. Ausgehend vom reinen Linux Kernel (Vanilla Kernel) oder schon angepassten (patched) Kernels erfolgt seine Konfiguration mittels „make config“ (textbasiert), „make menuconfig“ (semigrafisch) oder „make xconfig“ (grafisch). Mit der daraus erhaltenen Konfiguration, den Kernel-Header-Dateien und bestimmten, erzeugten Verknüpfungen kann man nun Kernel-Image und Kernel-Module „bauen“.

In Linux-Systemen benötigte Dienstprogramme, das sog. Root File System und die Device Files etc. bilden das als nächstes zu erzeugende *Userland*, also einen Bereich im Linux-System, der dem normalen Benutzer zugänglich ist. Häufig setzt man auch hier fertige Pakete und Skriptsammlungen ein, um das Userland zu erzeugen, beispielsweise *Buildroot*, das u.a. *Busybox* nutzt und die uClibc unterstützt. Dabei ist auch zu beachten, wo das Userland im fertigen System gespeichert sein soll, beispielsweise komplett mit Linux im RAM, FlashEPROM oder ausgelagert auf einem „Massenspeicher“ wie z.B. einer CompactFlash Card.

Die eigentlichen Anwendungsprogramme eines Embedded Linux müssen nun ebenfalls, eventuell mit zugehörigen Bibliotheken, konfiguriert und dann übersetzt werden.

Weitere Schritte vervollständigen das Root File System (im Wesentlichen kopieren von erzeugten Dateien in die entsprechenden Ordner), konfigurieren einen Bootloader o.ä. und erzeugen das eigentliche Image des Linux-Systems, das dann auf das Zielsystem geladen werden kann.

Die aufgeführten Prozesse können unter Nutzung einer kompletten Distribution für Embedded Linux-Systeme oder Teilen davon oder sogar ohne Distribution (Linux from Srcatch – Linux von Grund auf) ablaufen. Da viele Distributionen im Embedded Bereich nur spezielle Prozessoren und meist nur spezifische Boards unterstützen, gibt es in den meisten Fällen keine universelle Lösung. Die Variante *Linux from Scratch* bietet noch die größte Flexibilität, was aber auch bedeutet, dass fast alle benötigten Tools und Patches selbst zusammengetragen werden müssen.

Im Labor Datentechnik sind drei Typen von FPGA-Boards (mit Xilinx FPGAs) vorhanden, wovon nur zwei so ausgestattet sind, dass ein Linux darauf lauffähig ist: das XUPV2P (FPGA: Virtex II Pro) und das XUPV5 (FPGA: Virtex 5). Aus dem Vorläuferprojekt ergaben sich drei 32-Bit-Prozessor-Cores, die in diese Umgebung passen und entsprechend leistungsfähig sind: der Hard Core *PowerPC 405* (PPC405), der Soft Core *Microblaze* (MB) und der SPARC-kompatibe Soft Core *Leon3*. Der PowerPC ist auf den Virtex5-Bausteinen auf dem XUPV5 nicht verfügbar. Damit ergeben sich die folgenden möglichen Systeme:

- XUPV2P: PPC405, Microblaze, Leon3
- XUPV5: –, Microblaze, Leon3

Für diese fünf Konfigurationen wurden bislang drei verschiedene Linux-Varianten erstellt und getestet. Auf dem Leon3 wurde ein *SnapGear-Linux* entwickelt und installiert, das die von Aeroflex (vormals Gaisler) vorkonfigurierte Cross Toolchain verwendet und andere angepasste Teile des Linux-Systems beinhaltet. Das XUPV5 wird nicht von Aeroflex unterstützt, weshalb derzeit noch kein Linux darauf portiert wurde. Das Problem dabei liegt in der Erzeugung eines Boot Image und dem eigentlichen Bootvorgang, da für ein wichtiges, spezifisches Hardwareteil des Boards (ACE Controller) keine Treiber vorliegen und bei den anderen Prozessoren zur Zeit der Xilinx On-Board Debugger XMD die Images lädt. Mit der Verfügbarkeit eines universellen Bootloaders (U-Boot) am Ende des gerade laufenden Projekts (SS10) besteht eine realistische Chance diese Probleme zu lösen. Zu klären ist aber beispielsweise noch, ob die von BuildRoot genutzte Busybox die SPARC-V8-Achitektur des Leon3 unterstützt.

Auf beiden Plattformen konnten mit der Embedded Distribution *PetaLinux* von PetaLogix Linux-Systeme für den Microblaze entwickelt und installiert werden. PetaLinux ist augenblicklich die einzige Distribution, die den Microblaze vernünftig unterstützt. Die von Xilinx zur Verfügung gestellten Treiber für Microblaze und die anderen notwendigen IPs (z.B Ethernet-Schnittstelle) sind nicht Linux-konform und bedürfen noch umfangreicher Anpassung und Entwicklung (hoher Aufwand). Aus diesem Grund konnte im Projektzeitraum noch kein eigenes Linux für diesen Prozessor auf den beiden Boards erstellt werden. Weiterhin gab es auch Probleme bei der Erstellung der Cross Toolchain unter bestimmten Linux-Hostsystemen (z.B. Ubuntu).

Am interessantesten war die Arbeit mit dem PPC405 auf dem XUPV2P. Hier entstanden Linux-Systeme von Grund auf (Linux from Scratch) ohne Rückgriff auf bestehende Distributionen, was ein Vorgehen wie das oben beschriebene bedeutet. Dies zeigte, dass man unter bestimmten Voraussetzungen ein eigenes, der geplanten Applikation angepasstes Linux entwickeln kann, das nur die benötigten Komponenten enthält und damit den zur Verfügung stehenden Speicher sehr effizient nutzt. Zu den Voraussetzungen zählt, dass ein von Xilinx angepasster Kernel zur Verfügung steht, der einen Linux-konformen Treiber für die Ethernet-Schnittstelle enthält. Momentan ist diese wichtige Schnittstelle noch nicht im sog. Vanilla Kernel enthalten. Die Unterstützung anderer IPs wie z.B. die serielle Schnittstelle ist im Standard-Kernel sogar schon enthalten. Um die Problematik nicht verfügbarer Treiber angehen zu können, sollen im kommenden Wintersemester entsprechende Untersuchungen und Arbeiten stattfinden. Das Erstellen eigener Hardware und passender Linux-Treiber wird die Nutzung der Flexibilität der vorliegenden FPGA-Hardware in wesentlich größerem Umfang als bisher gestatten.

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