HW/SW-Codesign ressourcenminimaler Systeme, Teil 2
Eine Prozessorarchitektur mit integrierter Debugunterstützung

Hagen Ploog, Jens Hildebrandt, Tino Rachui

Universität Rostock
Fachbereich Elektrotechnik und Informationstechnik
Institut für Angewandte Mikroelektronik und Datentechnik
Richard-Wagner-Str. 31, 18119 Rostock, Tel.: (0381) 498 35 34
email: {hp, hil}@e-technik.uni-rostock.de

Abstract: In diesem Beitrag stellen wir einen 4b-Mikroprozessor für Signalvorverarbeitung vor, der eine Hardware-Erweiterung zum Debuggen der zu entwickelnden Software innerhalb der Zielhardware enthält. Diese ermöglicht einer entsprechenden Entwicklungsumgebung über eine modifizierte JTAG-Schnittstelle den Zugriff auf die internen Registerstrukturen des Zielprozessors. Somit läßt sich nicht nur der eigentliche Prozessor, sonder auch die an ihn angeschlossene Peripherie in den Softwaretest mit einbeziehen.

1. Einleitung

Software für eingebettete Systeme kann derzeit vorab nur beschränkt simuliert werden. Eine vollständige Simulation einschließlich der gesamten Peripherie ist jedoch in einem frühen Stadium der Entwicklung zu Verifikationszwecken unbedingt wünschenswert.
Kostengünstige Entwicklungssysteme beschränken sich in der Regel auf Simulation der Prozessorarchitektur. Hardware-Unterstützung des Debuggings ist wesentlich teurer und besteht im einfachsten Fall aus einem ROM-Emulator. Größere Funktionalität bieten Spezialprozessoren mit von außen über zusätzliche Leitungen zugänglichen Registern, sogenannten ice-cap-Systeme. Das sind in geringen Stückzahlen gefertigte und damit um Größenordnungen teurere Versionen des entsprechenden Prozessors. Sie erlauben jedoch nur die Softwareentwicklung für die jeweils ausgewählte Prozessorarchitektur. Bei Integration der Debug-Unterstützung auf dem Standard-Prozessorchip ergibt sich durch die Massenfertigung ein Kostenvorteil. In diesem Fall sollte durch diese zusätzliche Funktionalität ein möglichst geringer Mehraufwand bezüglich Chipfläche und Pin-Anzahl verursacht werden.
An der Universität Rostock wurde der RUN4 entwickelt, ein 4b-Mikrocontroller, dessen vorrangiges Einsatzgebiet im Bereich intelligenter Sensoren liegt [1]. Für derartige Systeme erfolgt die Hardware-Entwicklung zeitlich parallel zur eigentlichen Programmerstellung. In diesem speziellen Fall ist es von besonderer Bedeutung, daß die Software auf der realen Zielhardware getestet werden kann. Prozessoren wie der RUN4 haben nach der Produktion ihr Programm in einem Masken-ROM mit auf dem Chip vorliegen, daher ist eine
Korrektur des Programms zu einem späteren Zeitpunkt nicht mehr möglich. In diesem Beitrag präsentieren wir, ausgehend von der Architektur des Prozessors, eine Debug-Komponente, die es der zum Prozessor zugehörigen Entwicklungssoftware ermöglicht [7], Prozessorzustände auszulesen oder zu modifizieren.
Dazu wird eine modifizierte JTAG-Schnittstelle mit auf dem Chip integriert, die einen Zugriff auf die internen Registerstrukturen des Zielprozessors ermöglicht.

2. Grundlegende Architektur

Der RUN4-Kern basiert auf einer modifizierten Harvard-Architektur (Abbildung 1). Er weist sowohl RISC-typische Elemente wie Multiregister-Struktur und einheitliches Befehlsformat als auch CISC-typische Eigenschaften wie die Verwendung komplexer Adressierungsarten für arithmetische Funktionen auf. Die Ausführungszeit beträgt für einen Befehl durchschnittlich zwei Takte. Der RUN4 stellt keine statische Implementation dar, sondern liegt als VHDL-Beschreibung vor. Dadurch kann er zusammen mit den anzusteuerenden Sensoren und Aktoren auf einem Chip integriert werden. Um unterschiedlichen Anforderungsprofilen gerecht zu werden, ist die Architektur des RUN4-Controllers parametrierbar. So läßt sich z.B. vor der Synthese, dem Abbilden einer VHDL-Beschreibung auf eine Netzliste, festlegen, ob ein Interrupt-

System samt der dazugehörigen Befehle vorhanden sein soll oder nicht. Weiterhin kann über einen Parameter bestimmt werden, ob drei oder vier Arbeitsregister implementiert werden sollen.

Diese Architektur wurde um eine modifizierte JTAG-Schnittstelle erweitert, die neben den Pin-Zuständen auch die internen Registerinhalte auslesen und beschreiben kann. Nach Beenden der Software-Entwicklung kann die Debug-Komponente bei Bedarf wieder entfernt werden.

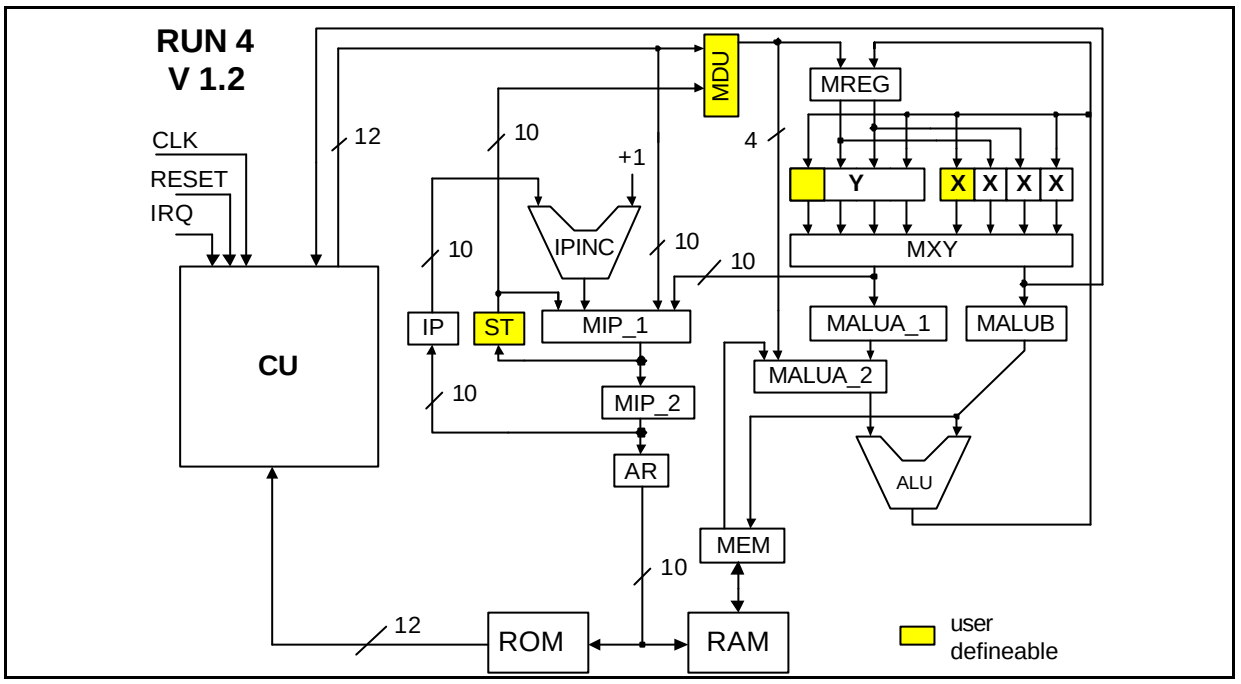

Abbildung 1 : Basisarchitektur des RUN4

Die JTAG-Architektur [2], die ursprünglich für Boundary-Scan [3] entwickelt wurde, sieht zwischen den Pins eines Schaltkreises und seinem Kern sogenannte Scan-Zellen vor, die über spezielle Pfade zu einem Schieberegister verkettet sind. Zusätzlich beinhaltet die JTAG-Schnittstelle noch weitere Register, das Instruction-, Bypass und User-Data-Register. Das Instruction-Register bestimmt über seinen Inhalt, welche Scan-Kette zwischen den seriellen Ein- und Ausgang geschaltet wird. Das User-Data-Register steht im allgemeinen für Nutzerdaten zur Verfügung und wird in der RUN4-Architektur dafür benutzt, um die aktuelle Konfiguration des Prozessorkerns von außen lesbar zu machen. Hierdurch erhält die Entwicklungssoftware Informationen über den konfigurierbaren Teil der Architektur und kann auf dieser Basis eine Prüfung des Sourcecodes auf nicht von der Hardware unterstützte Befehle vornehmen. Das Konzept wurde um zusätzliche Scanpfade für die internen Register erweitert (Abbildung 2).

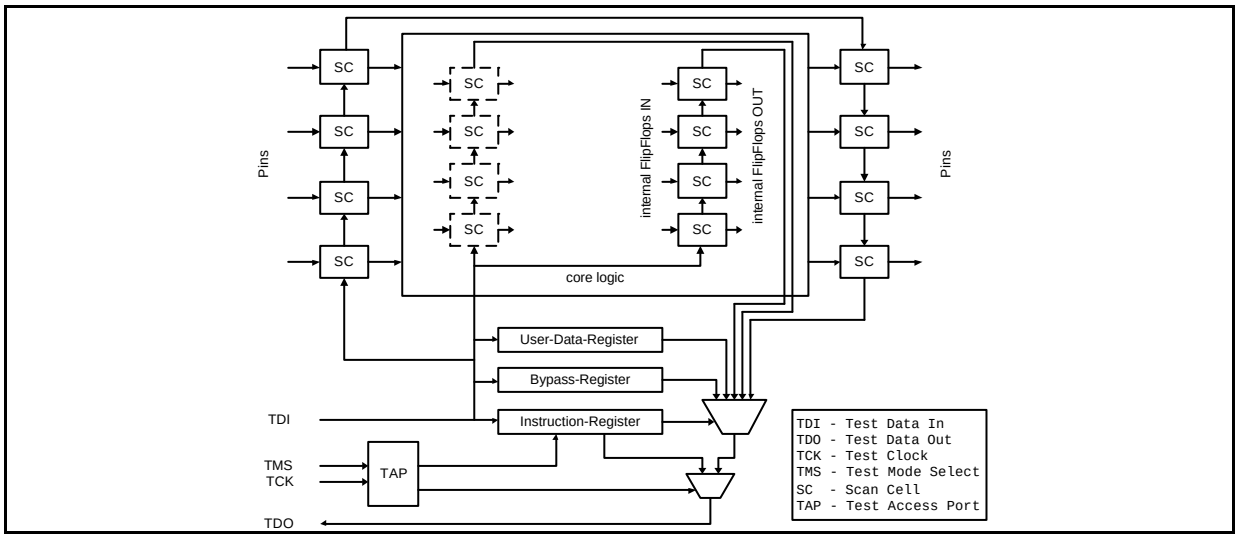

Abbildung 2 : modifizierte JTAG-Architektur

Dafür wurden die ursprünglichen FlipFlops gegen Scan-Zellen ausgetauscht, die sich aus Sicht des Prozessors wie normale Register verhalten, auf die jedoch genau wie auf die Scan-Zellen der Pins über die JTAG-Schnittstelle zugegriffen werden kann. Gesteuert wird der Zugriff über eine Zustandsmaschine, den TAP-Controller. Dieser ändert seine Zustände, die den einzelnen Phasen der Kommunikation entsprechen, in Abhängigkeit vom Pegel des TMS-Pin bei steigender Flanke am TCK-Pin. Durch die Anwendung des seriellen JTAG-Protokolls für die Debug-Schnittstelle wird die Architektur nur um vier zusätzliche Pins erweitert.

3. Funktionsweise der Debugeinrichtung

Zum Debuggen eines Programms auf dem RUN4 muß dieser mit dem Computer, auf dem die Entwicklungsumgebung abläuft, verbunden werden (Abbildung 3). Dafür ist ein spezielles Interface notwendig, das eine Verbindung zwischen einem der Standardports des PCs und der JTAG-Schnittstelle des RUN4 herstellt. Im vorliegenden Fall wurde dazu der Parallelport verwendet [4].

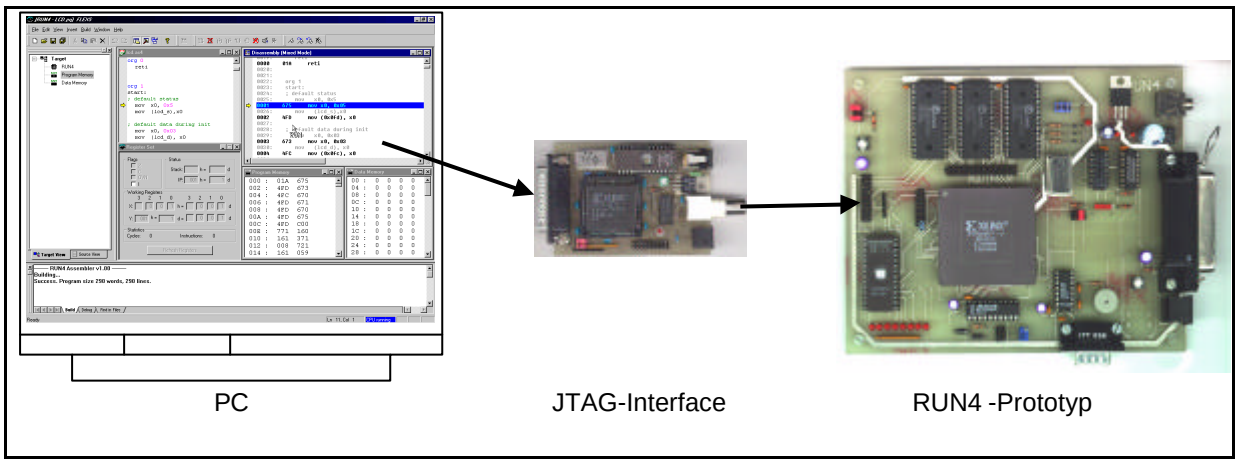

Abbildung 3 : Anbindung des RUN4 an die Entwicklungsumgebung

Dieses Parallelport-JTAG-Interface erhält von der Software-Entwicklungsumgebung Transferbefehle, die es selbsttätig in die entsprechenden JTAG-Steuersequenzen und Datenströme umsetzt. Das bedeutet, daß der TAP-Controller zunächst in den entsprechenden Zustand gebracht wird und im Anschluß daran dem PC signalisiert wird, daß nun die eigentlichen Daten übertragen werden können.
Prinzipiell werden über die JTAG-Schnittstelle lediglich folgende Grundoperationen ausgeführt, aus denen sich sämtliche gewünschten Zugriffe auf den Mikroprozessor zusammensetzen:

- Lesen / Schreiben des Instruction-Registers
- Lesen / Schreiben des ausgewählten Scan-Pfades

Eine weitere Vereinfachung der Ansteuerungssoftware ergibt sich aus der Nutzung des Parallelports im Enhanced-Parallel-Port-Modus (EPP-mode), in dem ein Hardware-Handshake durchgeführt wird. Somit müssen lediglich die Steuer- und Datenbytes für das JTAG-Interface auf die entsprechende Portadressen geschrieben bzw. von dort gelesen werden. Um Synchronität zwischen der Programmabarbeitung auf der Zielhardware und den Debug-Zugriffen zu gewährleisten, wird der Prozessor während des Scanbetriebes von dem JTAG-Interface mit dessen Takt betrieben. Dazu ist eine Taktumschaltlogik in den RUN4 integriert worden. Das hat den Effekt, daß beim Datenaustausch zwischen den eigentlichen Registern und den Scan-Zellen keine Metastabilitäten und damit keine undefinierten Zustände auftreten können. Durch die Bereitstellung des Taktes durch das JTAG-Interface ist es möglich, vor dem Auslesen oder Beschreiben der Registern den Prozessor anzuhalten. Damit während der Debug-Phase der Prozessor mit seiner vorgesehenen Arbeitsfrequenz betrieben werden kann, läßt sich das Taktsignal des Interfaces in weiten Bereichen variieren (10 kHz bis 20 MHz).

Die Ausführung einer Debug-Operation beginnt damit, daß zuerst der entsprechende Code in das Instruction-Register geschrieben wird. Dadurch wird ein spezielles Scan-Register selektiert, das in folgenden Datentransfers beschrieben und/oder gelesen werden kann. Eine Ausnahme davon bildet die INSTEP-Anweisung. Diese bewirkt, daß der RUN4 jeweils genau einen Befehl ausführt. Dafür wird der Prozessortakt solange freigeschaltet, bis interne Steuersignale das Ende einer Befehlsabarbeitung anzeigen. Diese Anweisung stellt die Grundlage für des Debuggen dar, auf der ein kontrolliertes Ausführen eines Programms möglich wird.

4. Leistungsmerkmale

Mit der hier vorgestellten Architektur lassen sich nun die von Software-Simulatoren her gewohnten Funktionalitäten auch in der Zielhardware bereitstellen [5]:

- Wahlfreier Zugriff auf Prozessorregister
- Lesen und Verändern von Flags
- Setzen und Löschen von Breakpoints
- Beliebige Unterbrechung und Fortsetzung der Programmabarbeitung
- Schrittweises Ausführen des Programmcodes
- Animated run

Zusätzlich lassen sich aber noch weitere Funktionen realisieren:

- Lesen und Ändern der Speicherinhalte (RAM und ROM)
- Ausführen von beliebigen Befehlen außerhalb des regulären Programmablaufes

Diese werden von der Software durch Kombination einfacherer Befehle zusammengesetzt. Das Ändern des Speicherinhaltes wird durch Ausführen eines RUN4-Schreibbefehls (mov adr,x0) realisiert. Dadurch werden die Register und der Instruction-Pointer verändert, so daß diese zunächst durch Auslesen gesichert und nach der Befehlsausführung wieder zurückgeschrieben werden müssen [6].

5. Zusammenfassung

Die vorstehend beschriebene Prozessorarchitektur mit integrierter Debug-Unterstützung ist prototypisch auf einem FPGA der Firma XILINX implementiert worden. Die maximal erreichbare Taktrate beträgt 12 MHz. Mit der hier vorgestellten Komplettlösung aus debug-unterstützender Hardware und einer dazu gehörigen Softwareentwicklungsumgebung lassen sich sowohl die Software als auch die den Prozessor umgebende Peripherie weitgehend parallel entwickeln und testen. Damit wird der Entwicklungszyklus wesentlich verkürzt, so daß Produkte früher eine Marktreife erhalten. Damit, und aufgrund seines geringen Resourcenbedarfs und der Konfigurierbarkeit der Architektur, ist der RUN4 damit bestens für System-on-Chip-Applikationen (SoC) geeignet.

6. Literatur

[1] Ploog, H; Wassatsch, A.; Dolling, S.; Timmermann, D., "RUN4 - Flächenminimierter 4b-µP-Core als VHDL Modul für die Mikrosystemtechnik und Sensorelektronik", Fachtagung Informations- und Mikrosystemtechnik, Universität Madgeburg, März 1998

[2] Texas Instruments,"Primer - IEEE Std. 1149.1 (JTAG) Testability 1997", Texas Instruments Incorporated

[3] Texas Instruments, "Boundary-Scan-Logic 1998", Texas Instruments Incorporated

[4] Jens Hildebrandt, "Aufbau eines Parallelport-JTAG-Interface für IBM-kompatible Personalcomputer", Diplomarbeit, Universität Rostock, 1997

[5] Niels Reinhardt, "Konzeption, Entwicklung und Implementierung einer modifizierten JTAG-Schnittstelle für In-Target Online-Software-Debugging", Diplomarbeit, Universität Rostock, 1999

[6] Bert Katschke, "Entwicklung einer OnLine-Debug-Komponente für eine Mikroprozessorsimulationsumgebung auf Basis von COM/OLE", Diplomarbeit, Universität Rostock, 1999

[7] Tino Rachui, et.al., "HW/SW-Codesign ressourcenminimaler Systeme, Teil 1....", in diesem Tagungsband