

Systematischer Entwurf und Test eingebetteter Systeme am Beispiel eines ATM-Diagnosesystems

Karlheinz Weiß, André Hergenhan, Wolfgang Rosenstiel

Forschungszentrum Informatik an der Universität Karlsruhe (FZI)

Systementwurf in der Mikroelektronik (SIM)

Haid- und Neu-Straße 10-14

Email: weiss@fzi.de, hergenhan@fzi.de, rosenstiel@informatik.uni-tuebingen.de

Kurzfassung

Der folgende Beitrag analysiert die Problemfelder beim Entwurf eingebetteter Systeme und beschreibt als Lösungsansatz eine effiziente und praxisorientierte Prototyping-Methode. Grundlage bildet dabei die Nutzung einer modularen Prototyping-Umgebung, die eine schnelle Evaluierung verschiedener Zielarchitekturen eingebetteter Systeme ermöglicht. Sie besteht aus einem Field Programmable Gate Array-Modul (FPGA-Modul) für die Emulation von anwendungsspezifischer Hardware und einem eingebetteten Mikrocontroller-Modul zur Implementierung der Software. Eine ebenfalls FPGA-basierte, frei programmierbare Schnittstellenstruktur ermöglicht eine flexible Kopplung der beiden Module. Zusätzlich ermöglicht sie die Einbindung von vorgefertigter, dedizierter Hardware. Die vorgestellte Methode für den Entwurf und Test von eingebetteten Systemen wird am Beispiel eines ATM-Diagnosesystems demonstriert.

1. Einleitung

Obwohl eingebettete Systeme¹ sehr weit verbreitet sind, hat sich bis heute eine allgemeingültige Definition nicht durchgesetzt [1]. In diesem Beitrag wird davon ausgegangen, daß ein eingebettetes System aus einem anwendungsspezifischen Softwareteil, der auf einem Mikrocontroller-Kern abgearbeitet wird, und einem anwendungsspezifischen Hardwareteil besteht, welcher z.B. als FPGA oder ASIC realisiert werden kann. Beide Teile kommunizieren untereinander und mit der Umwelt. Insbesondere die Umwelt gibt die Randbedingungen vor (Zeitverhalten, Leistungsaufnahme, Baugröße, usw.).

1. Die Arbeit wurde teilweise mit Mitteln der Deutschen Forschungsgemeinschaft unter dem Geschäftszeichen 3221040 im Rahmen des Schwerpunktprogramms „Entwurf und Entwurfsmethodik eingebetteter Systeme“ gefördert.

Der Entwurfsprozeß wird heute überwiegend sequentiell durchgeführt. Nach einer Spezifikationsphase werden die Funktionen auf verschiedene Systemkomponenten verteilt. Dieser Prozeß beruht hauptsächlich auf der Erfahrung der Entwickler. Danach wird ein spezifischer Hardware-Prototyp erstellt und getestet. Anschließend erfolgen die Entwicklung der Anwendungssoftware und der Systemtest. Stellt sich heraus, daß die Randbedingungen nicht eingehalten werden können, kann ein zeitaufwendiges Redesign notwendig werden.

Bei komplexeren Entwicklungen, die durch den starken Innovationsdruck in immer kürzeren Entwurfszeiträumen erstellt werden müssen, ist dieser Weg nicht mehr praktikabel. Gesucht werden daher Methoden für einen systematischen Entwurf, um frühzeitig Probleme erkennen und getrennt bearbeiten zu können. Ein Lösungsansatz basiert auf der Nutzung einer entsprechenden Entwicklungsumgebung, die eine Co-Emulation von Hardware und Software ermöglicht.

In [2] [3] werden Entwicklungsumgebungen vorgestellt, welche aus einem fest vorgegebenen Mikrocontroller-Kern und FPGAs für anwendungsspezifische Hardware bestehen. Die Konfigurationsdaten für die FPGAs und der Programmcode für den Prozessor werden über Lade-Verbindungen vom Entwicklungsrechner geladen. Die Architektur besteht in [2] aus nur einem FPGA, in [3] ist sie dagegen sehr komplex. Sie besteht aus vier quadratisch angeordneten FPGAs. Zusätzlich läßt sich deren Verbindungsstruktur durch sogenannte FPICs frei wählen. Die beiden Entwicklungsumgebungen sind jedoch nicht modular aufgebaut, ermöglichen keine freie Auswahl des Mikrocontrollers und haben keine geeigneten Schnittstellen zur Einbindung weiterer, dedizierter anwendungsspezifischer Hardware (z.B. ASICs).

Da geschlossene Entwurfsmethoden für alle Einsatzgebiete eingebetteter Systeme nicht angegeben werden können, konzentrieren sich unsere Verfahren und Aussagen speziell auf die Bereiche der industriellen Automation und Kommunikation. Die folgende Arbeit analysiert zunächst in Kapitel 2 die wichtigsten Problemfelder beim Entwurf solcher eingebetteter Systeme. Kapitel 3 stellt eine systematische Entwurfsmethode zur Lösung der genannten Problemfelder vor und leitet daraus Anforderungen an entsprechende Entwurfswerkzeuge ab, welche in Kapitel 3.1 und 3.2 vorgestellt werden. In Kapitel 4 wird die Methode am Beispiel eines ATM-Diagnosesystems demonstriert. Kapitel 5 faßt die wichtigsten Aussagen zusammen.

2. Problemfelder beim Entwurf eingebetteter Systeme

Die wesentlichen Problemfelder beim Entwurf eingebetteter Systeme liegen in der Weiterentwicklung der Mikrocontroller zu „System-On-A-Chip“-Lösungen, der Einbindung hochintegrierter ASICs, technologiebedingter Probleme bei der Systemfertigung sowie in der Entwurfsstrategie der Entwickler selbst.

2.1 Problemfeld Mikrocontroller

Wesentliches Kriterium bei der Architekturbestimmung für ein eingebettetes System ist die Auswahl des Mikrocontrollers. Dabei ist nicht nur die reine Rechenleistung des CPU-Kerns ausschlaggebend, sondern vielmehr die Gesamtfunktionalität aus Prozessor-Kern und „On-Chip“-Peripherie. Diese reicht von typischen Funktionen wie Interruptcontroller, Timer und seriellen Ports (z.B. beim MC8051) bis hin zu „System-On-A-Chip“-Lösungen [6] (z.B. beim MPC860). Solche Bausteine verfügen über ausgeprägte Buskontroll- und Schnittstellenfunktionalitäten. Dadurch können fast alle verfügbaren Speichertechnologien mit unterschiedlichen

Busbreiten- und Geschwindigkeiten ohne zusätzliche Hardware (dynamic bus sizing) an den Mikrocontroller angeschlossen werden. Externe Hardware muß dafür nicht mehr entworfen werden und die Komponentenanzahl für ein System sinkt deutlich. Das Problem verlagert sich vom eigentlichen Hardware-Entwurf in die optimale Konfiguration und Ausnutzung der „On-Chip“-Ressourcen.

Insbesondere die Testverfahren verändern sich erheblich. Mußte der Systementwickler früher hauptsächlich Signale auf der Board-Ebene beobachten, so muß er nun bei einem „System-On-A-Chip“ die Signale im Controller verfolgen. Dazu sind neue Verfahren wie Background Debug Mode (BDM) notwendig. Diese nutzen im Chip integrierte Einheiten und erfordern keine aufwendigen externen Emulatoren.

2.2 Problemfeld ASIC¹-Einbindung

Mit Hilfe der VLSI-Technologie kann immer mehr Funktionalität in einem ASIC integriert werden. Funktionen, die in der Vergangenheit ganze Baugruppen füllten, stehen heute als preisgünstige Chips (z.B. AAL-Controller für B-ISDN) zur Verfügung. Sie können als in sich abgeschlossene Systeme betrachtet werden und kommunizieren mit ihrer Umgebung über gemeinsame asynchrone Pufferspeicher. Die Beschreibung dieser Pufferspeicher (Anfangsadressen und Puffergröße, Zugriffsrechte, Kommunikationsparameter, usw.) wird in sogenannten Deskriptoren festgelegt, welche ebenfalls in einem gemeinsamen Speicherbereich stehen. Zusätzlich können sie noch viele Initialisierungsparameter besitzen, die über ein Prozessor-Interface im ASIC programmiert werden müssen. Diese gemeinsamen Datenstrukturen und Parameter müssen von einem Host richtig initialisiert und während des Betriebs kontrolliert werden. Dazu ist eine sehr anwendungsspezifische Software notwendig, welche man auch als Firmware auffassen kann, die einen solchen ASIC-Baustein erst zur gewünschten Funktion bringt. Das bedeutet auch, daß ein ASIC nicht für sich alleine (z.B. beim Test) betrieben werden kann, er benötigt immer die Anbindung an einen Hostrechner.

Die Testeigenschaften solcher ASICs sind aber größtenteils nicht so gut ausgeprägt (z.B. durch Rücklesemöglichkeiten der Initialisierungsparameter oder BDM). Sind alle Start-up Bedingungen erfüllt (Deskriptoren und Register initialisiert, gemeinsamer Speicher und Board funktionsfähig), dann funktioniert der Chip, andernfalls ist seine Funktion nicht vorhersagbar. Diese „geht-geht nicht“ Mentalität ohne geeignete Analysemöglichkeiten erschwert die Einbindung und den Test in komplexen Systemen.

2.3 Problemfeld Technologie

Weitere Probleme erwachsen auch aus der Technologie für die Systemfertigung. Mit den „On-Chip“-Funktionen steigt auch die Zahl der I/O-Pins der Bausteine. SMD-Gehäuse mit bis zu 300 Pins (z.B. 0,5 mm Finepitch) oder Ballgrid-Arrays mit bis zu 600 Pins werden bereits eingesetzt. Durch hohe Taktfrequenzen entstehen weitere, häufig schwer zu lokalisierende und nicht deterministische Fehler. Diese Bausteine werden auf Boards mit bis zu zehn Signallagen integriert. Diese Randbedingungen führen zu Fehlern in der Systemfertigung, die nur schwer erkannt und selbst bei Prototypen kaum behoben werden können. Flexible Probeaufbauten, wie bei TTL-Bausteinen üblich, sind bei hochintegrierten Bausteinen nicht mehr möglich.

1. Unter ASIC verstehen wir bereits am Markt verfügbare VLSI-Bausteine, die sehr spezielle und komplexe Teilfunktionen innerhalb ihres Einsatzgebietes ausführen.

2.4 Problemfeld Entwickler

Um die beschriebenen Komponenten zu nutzen, muß der Entwickler bereit sein, in jedem Projekt neue Bausteine einzusetzen, die eine geforderte Gesamtfunktion möglichst optimal erfüllen. Zu beobachten ist jedoch, daß diese Freiheitsgrade heute noch nicht ausgenutzt werden. Entwickler bleiben lange Zeit bei einer Komponentengeneration, was häufig dazu führt, daß vorhandene und bessere Lösungen nicht genutzt werden.

3. Systematische Entwurfsmethode

Im Bereich der industriellen Automation und Kommunikation zeichnen sich Architekturen dadurch aus, daß sie aus wenigen Komponenten bestehen. Im Mittelpunkt steht dabei ein Mikrocontroller mit seinen Speicherbausteinen (Mikrocontroller-Kern) und ein ASIC-Baustein. Auf dem Mikrocontroller werden die datenorientierten Teile, auf dem ASIC die steuerungorientierten Teile implementiert. Diese Komponenten sind jedoch hochintegrierte Bausteine mit sehr komplexen On-Chip-Funktionen. Bei der Partitionierung einer Gesamtfunktion auf einzelne Komponenten ist man bestrebt, bereits am Markt verfügbare Komponenten einzusetzen, die die Anforderungen der Spezifikation erfüllen. Erst wenn keine geeigneten Komponenten (ASICs oder Mikrocontroller) zur Verfügung stehen, muß man selbst Hardware entwerfen. Diese Notwendigkeit bildet das typische Einsatzfeld von anwenderprogrammierbaren Bausteinen (FPGAs und CPLDs).

Unsere Entwurfsmethode für eingebettete Systeme versucht, die aufgezeigten Problemfelder möglichst getrennt zu behandeln, um die Entwicklung und den Test zu beschleunigen. Abbildung 1 zeigt den vorgeschlagenen Entwurfsablauf.

Nach der Spezifikation wird das System auf einzelne Komponenten partitioniert. Dieser Prozeß wird sich in Zukunft wesentlich verändern. In der Vergangenheit war die Menge eingesetzter Bausteine sehr eingeschränkt (z.B. MC80C51, TTL und PAL-Bausteine). Die Anzahl der Bausteine pro System war hoch bei eingeschränkter Funktionalität eines einzelnen Chips. In der Zukunft wird es weniger Bausteine pro System mit mehr Funktionalität auf den verbleibenden Chips geben. Diese innovativen Komponenten der Chip-Hersteller müssen in jede neue Entwicklung einbezogen werden, um kostenintensive Eigenentwicklungen zu minimieren. Das bedeutet, Systementwickler müssen häufig Komponenten wechseln, um optimale Gesamtsysteme zu entwerfen. Dieser Punkt steht in Zusammenhang mit dem Problemfeld Entwickler in Kap. 2.4. Ein Ansatz, um Systementwicklern den schnellen Wechsel von Bausteinen zu erleichtern, besteht in der nachfolgend diskutierten Entwurfsumgebung.

Nach der Partitionierung werden zwei getrennte und parallele Pfade für Hard- und Software durchlaufen. Unter Hardware verstehen wir den Entwurf und Test des steuerungorientierten Teils, welcher mit Hilfe eines am Markt verfügbaren ASICs oder durch Eigenentwicklung in einem FPGA/CPLD realisiert werden kann. Dieser Pfad dient zur Isolierung des Problemfeldes ASIC-Einbindung in Kap. 2.2. Um die Funktionsweise eines ASICs für sich alleine zu testen, braucht man ein Entwicklungswerkzeug, welches die Schnittstellen zur Umwelt entsprechend stimuliert und insbesondere die Schnittstelle zum Mikrocontroller durch einen geeigneten Ersatz adaptiert, da ASICs in der Regel nicht ohne Anbindung an einen Host betrieben werden können. Im Falle von eigener Logik-Entwicklung muß die Entwicklungsumgebung geeignete Ressourcen in Form von programmierbaren Bausteinen (FPGAs, CPLDs) zur

Verfügung stellen. Diese Anforderungen werden von unserem Werkzeug FPGA-Modul erfüllt, welches in Kap. 3.1 beschrieben wird.

Der Software-Pfad betrifft in erster Linie den datenorientierten Teil. Die Entwurfsumgebung muß dazu einen entsprechenden Mikrocontroller-Kern zur Verfügung stellen. Neben dem Entwurf und Test der eigentlichen Algorithmen für die Anwendung muß sie grundlegende Einblicke in das Systemverhalten ermöglichen. Darunter verstehen wir Leistungsanalysen des Mikrocontrollers in Abhängigkeit von bestimmten Systemparametern (Taktrate, angeschlossene Speicher- Busbreiten und Zugriffsgeschwindigkeiten, Caches, usw.). Dieser Pfad ermöglicht die Isolierung des Problemfeldes Mikrocontroller in Kap. 2.1. Zur Durchführung haben wir das Werkzeug Mikrocontroller-Modul entwickelt, welches in Kap. 3.2 beschrieben wird. Dadurch ist auch ein Software-Test vor dem eigentlichen Integrationstest möglich. Das ist eine wesentliche Erweiterung zu der bereits in der Einleitung beschriebenen sequentiellen Methode (siehe Abbildung 1).

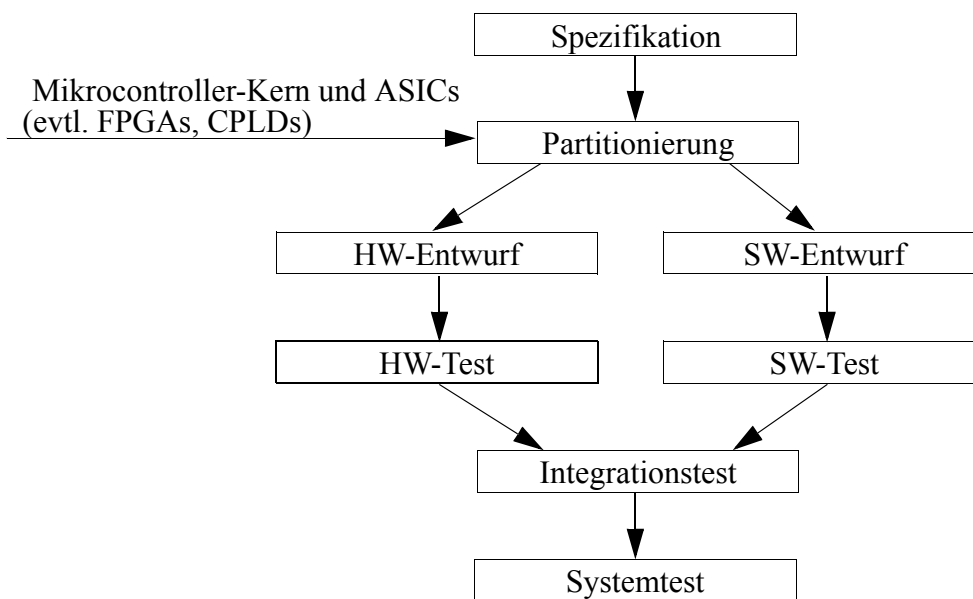


ABBILDUNG 1. Systematischer Entwurfsablauf

Der Integrationstest führt die beiden parallelen Pfade wieder zusammen. Die Entwicklungsumgebung muß diesen Schritt durch ein systematisches Zusammensetzen der Werkzeuge unterstützen. Geeignete Testunterstützung ist ebenfalls notwendig, um das komplexe Gesamtsystem zu kontrollieren.

Der Systemtest schließt den Entwurfsprozeß ab. Dazu muß es möglich sein, die Umwelt an das Gesamtsystem anzuschließen. Zusätzlich braucht man den Anschluß von geeigneten Testrechnern, um die anfallende Datenmenge während des Tests zu analysieren.

3.1 FPGA-Modul

Hauptforderungen bei der Spezifikation dieses Werkzeugs waren die Anbindung des PC an der Mikrocontroller-Schnittstelle eines ASICs (als Ersatz für einen Mikrocontroller) und die Bereitstellung technologieübergreifender, anwenderprogrammierbarer Bausteine. Abbildung 2 zeigt das Blockdiagramm.

3.1.1 PC als Ersatz für Mikrocontroller

Bei der Inbetriebnahme eines komplexen ASIC-Bausteins muß man sehr spezifische Software (Firmware, siehe auch Kap. 4) entwickeln, welche einen solchen Baustein zur Funktion bringt. Um diese Probleme (siehe auch Kap. 2.2) losgelöst vom restlichen System bearbeiten zu können, muß man einen Ersatz für den Mikrocontroller haben. Wir benutzen dazu den PC, welcher über den AT-ISA Bus an einem SRAM-basierten FPGA (XC4010E) angeschlossen ist. Auf der anderen Seite wird über einen leistungsstarken Extension-Port der eigentliche ASIC (hier in Form eines ATM-Moduls) angeschlossen. Im XC4010E-FPGA kann eine flexible Interface-Komponente installiert werden, welche den PC mit dem ATM-Modul verbindet. Die Benutzung des PC zur Inbetriebnahme und Test dieser dedizierten Hardware hat folgende Vorteile:

- Nutzung des PC als Software-Entwurfs- und Testplattform (C-Compiler und Debugger), Laden und Beobachten der Test-Software auf einem Zielsystem entfällt
- Einfachere Handhabung als Test-Software auf einem Mikrocontroller (insbesondere Nutzung der Infrastruktur eines PC wie Bildschirm/Tastatur zur Anzeige von Zuständen, usw.)
- Isolierte Bearbeitung der Problemfelder ASIC-Einbindung (Kap. 2.2)
- Begrenzung der technologischen Probleme (Kap. 2.3) auf ein kleines Ansteckmodul als Träger des ASICs (z.B. das ATM-Modul). Die restliche Hardware (PC und FPGA-Modul) steht als getestete und wiederverwendbare Hardware zur Verfügung.

Die Vorteile dieser Test-Architektur wird in Kap. 4 anhand eines Beispiel demonstriert.

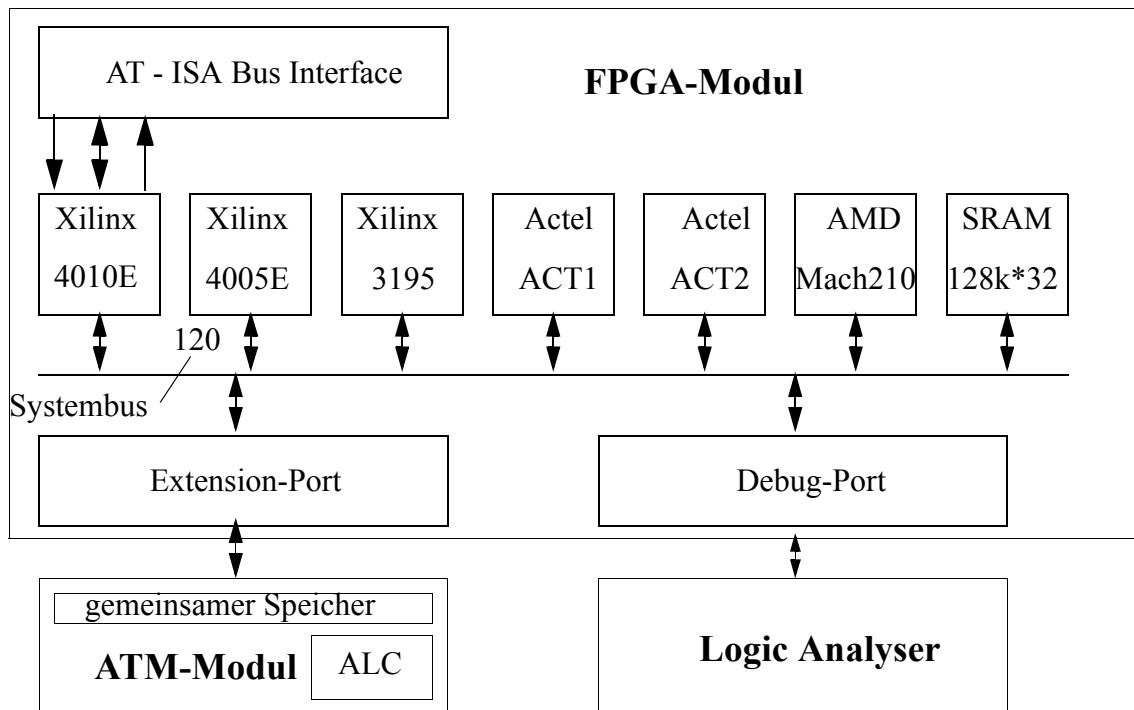


ABBILDUNG 2. Blockdiagramm des FPGA-Moduls mit angeschlossenem ATM-Modul

3.1.2 Technologieübergreifende FPGA/CPLD-Ressourcen

Fordert die Spezifikation einer Anwendung den Entwurf von eigener Logik, benötigt man anwenderprogrammierbare Bausteine. Eine Entwurfsplattform, welche dem Zielsystem möglichst nahe kommt, muß verschiedene, programmierbare Bausteine zur Verfügung stellen, um deren spezifische Eigenschaften zu nutzen. Das FPGA-Modul stellt technologieübergreifend neben SRAM-basierten FPGAs auch Antifuse- und CPLD-Bausteine zur Verfügung. Da reguläre Speicherstrukturen zur Zeit in FPGAs nicht effizient nachgebildet werden können, steht auch eine SRAM-Bank zur Verfügung. Die einzelnen Bausteine sind untereinander über einen leistungsstarken Systembus verbunden. Die Verbindung mit dem Mikrocontroller-Modul oder anderen Modulen mit dedizierter Hardware erfolgt über den Extension-Port. Der Debug-Port dient zum Anschluß von Testwerkzeugen.

Auf die Vor- und Nachteile der verschiedenen Technologien kann hier nicht eingegangen werden. Im Rahmen des ATM-Beispiels (siehe Kap. 4) wurde nur der Baustein XC4010E (Interface-Komponente) und der MACH 210 (schneller Arbiter) benutzt. Das FPGA-Modul wurde bereits in weiteren, praxisrelevanten Anwendungen [4] [8] eingesetzt, bei denen auch die anderen Bausteine benötigt wurden.

3.2 Mikrocontroller-Modul

Abbildung 3 zeigt das Blockdiagramm des Mikrocontroller-Moduls. Hauptkriterium bei der Definition der Architektur war die Tatsache, daß abhängig von der Anwendung der Mikrocontroller wechseln kann. Der Mikrocontroller-Steckplatz bietet die Möglichkeit, das eigentliche Mikrocontroller-Chip auszutauschen. Das Modul an sich stellt nur die Infrastruktur (unterschiedliche Speicher, Debug- und I/O-Ports, usw.) zum Betrieb eines 8, 16 oder 32 Bit Mikrocontrollers zur Verfügung. Bereits integriert wurde ein Stecksocket für einen 32 Bit PowerPC Embedded Controller (PPC403GA von IBM) [5].

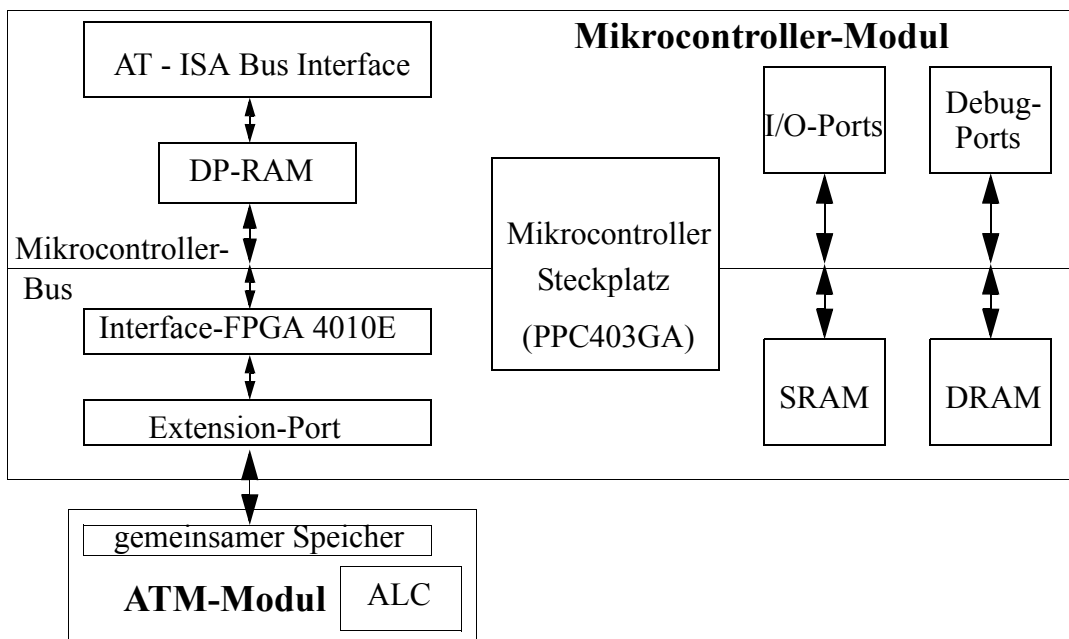


ABBILDUNG 3. Blockdiagramm des Mikrocontroller-Moduls mit angeschlossenem ATM-Modul

Weitere Adaptionen von 8 Bit Mikrocontroller der MC80C51-Serie oder des hyperstone E1-32 sind vorgesehen. Der gesamte Mikrocontroller-Bus kann über ein Xilinx Interface-FPGA mit dem FPGA-Modul oder weiterer Hardware verbunden werden, um individuelle Anpassungen zu ermöglichen (z.B beim Integrationstest). Die Karte wurde als PC-AT Einsteckkarte realisiert und paßt mechanisch und elektrisch zum FPGA-Modul.

4. Rapid Prototyping eines ATM-Diagnosesystemes

Anhand der Entwicklung eines ATM-Diagnosesystems für B-ISDN soll die Entwurfsmethodik demonstriert werden. Zur Diagnose von ATM-Netzen wird für alle virtuellen Pfadkennungen (VCI) eine Statistik über die Anzahl der empfangenen Datenpakete in Abhängigkeit der beim Empfang erkannten Fehlerzustände auf AAL5-Ebene benötigt (Monitor-Mode). Die komplette Bearbeitung der AAL5-Ebene erfüllt ein sogenannter Adaption Layer Controller (ALC), der hier in Form von dedizierter Hardware (ASIC) auf dem ATM-Modul zur Verfügung steht.

Abbildung 4 zeigt die zum Senden benötigte Datenstruktur, auf die in einem gemeinsamen Speicher von beiden Seiten zugegriffen wird. Vor Beginn des Betriebs muß diese Datenstruktur vom Host initialisiert werden. Während des Betriebs regelt diese Datenstruktur die Kommunikation zwischen Host und dem ALC. Der Transmit Descriptor besteht aus einem 8*32 Bit Feld und beinhaltet neben Zeigern auf die Nutzdaten und Circuit Reference Tables viele Einzelbit-Einträge. Diese steuern die Verarbeitung (Segmentation) der Nutzdaten und die Fehlerbehandlung. Die Circuit Reference Table ist ein 4*32 Bit Feld und enthält Verkehrsinformationen und Parameter, die im Zellkopf der ATM-Zelle weitergegeben werden. Die Transmit Pending Queue steuert die Verarbeitung mehrerer Pakete in einer Warteschlange und die Transmit Buffer Release Queue gibt bearbeitete Speicherbereiche wieder frei. Eine ähnliche Datenstruktur gibt es auch noch für den Empfangspfad. Zusätzlich müssen bis zu 60 Register vom Host direkt im ALC programmiert werden. Eine tiefergehende Erklärung kann hier aus Platzgründen nicht gegeben werden und ist in [7][9] zu finden.

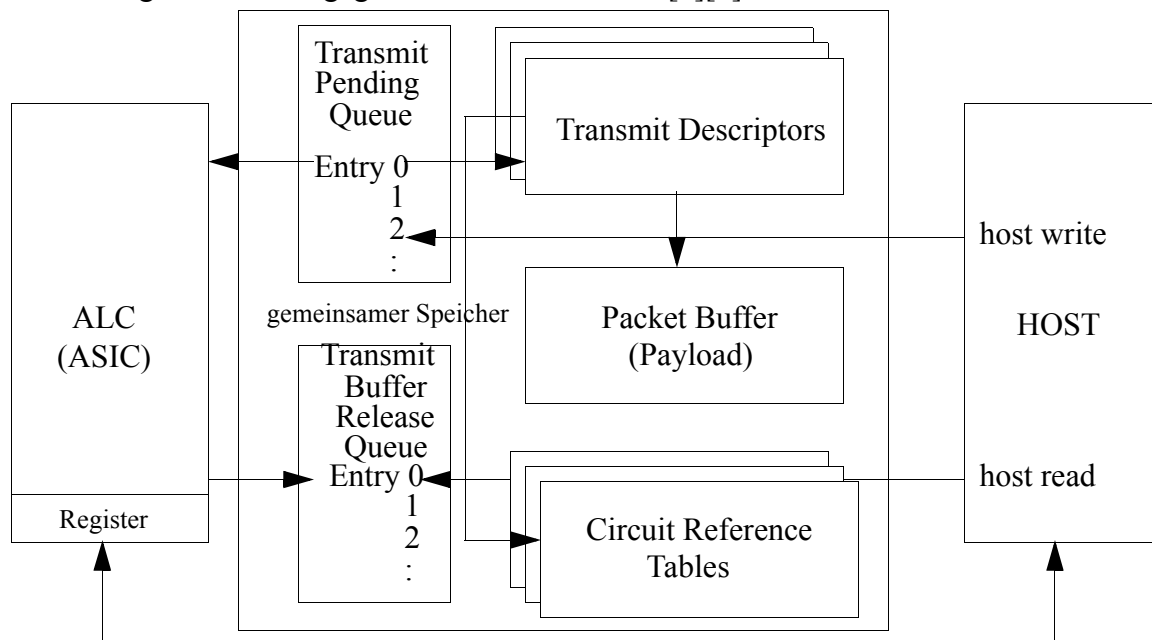


ABBILDUNG 4. Transmit-Datenstruktur (Quelle: FUJITSU)

An diesem Beispiel soll gezeigt werden, daß zum Betrieb des ALC eine sehr spezielle Software (Firmware), bestehend aus komplexen Bit-Kombinationen benötigt wird. Der ALC liefert dabei keine geeignete Testunterstützung. Würde man nun das Gesamtsystem, bestehend aus ALC und den Host-Mikrocontroller (z.B. PPC403GA) als ein einziger Prototyp aufbauen, würden sich alle beschriebenen Probleme in Kapitel 2 überlagern. Der Test würde wesentlich erschwert oder vielleicht unmöglich. Deshalb ermöglichen unsere Werkzeuge die Trennung der Problemfelder durch Überführung der Gesamtarchitektur in zwei Testarchitekturen.

Die erste Testarchitektur zielt auf die Entwicklung des steuerungsorientierten Teils, also der ALC-Firmware, und ist in Abbildung 2 dargestellt. Der ALC-ASIC wird zusammen mit einem 128kByte * 32 Bit großen gemeinsamen Speicher auf einem Board (ATM-Modul) mit dem FPGA-Modul verbunden. Im XC4010E-FPGA wird eine Interface-Komponente installiert, welche das 32 Bit Interface des gemeinsamen Speichers an das 8 Bit Interface des AT-ISA Bus adaptiert. Zusätzlich wird auch das Prozessor-Interface des ALC (60 Register) in den Speicherbereich des PC eingebunden. Diese Interface-Komponente wurde in VHDL beschrieben und mit Synopsys synthetisiert (ca. 10 min. auf einer SPARC 20). Das Platzierungs- und Verdrahtungswerkzeug (XACT von Xilinx) braucht auf der gleichen Maschine ca. 25 min zur Generierung der Programmierdaten. Der Zielbaustein XC4010E ist dabei etwa zu 35% ausgelastet. Das Laden der Programmierdaten von der Festplatte des Entwicklungsrechners ins FPGA erfolgt über den AT-ISA Bus vollautomatisch und dauert wenige Millisekunden. Dazu haben wir eigene Betriebssoftware entwickelt, welche die Konfiguration der FPGAs und die Steuerung des FPGA-Moduls unterstützt. Der MACH 210 ist ein schneller CPLD-Baustein und kontrolliert die Zugriffe (Arbiter) von beiden Seiten auf den gemeinsamen Speicher. Die Firmware-Entwicklung erfolgt auf dem PC und hat folgende Vorteile:

- Abtrennung der Entwurfsprobleme des ALC-ASICs vom restlichen System
- Lade-Prozeß entfällt (Entwicklungssystem = Zielsystem), Nutzung leistungsstarker Entwicklungswerkzeuge auf dem PC (Debugger, C-Compiler)
- Nutzung der Intrastruktur des PC (Bildschirm, Tastatur, Festplatte) zur Anzeige von Fehlerzuständen bei der Firmware-Entwicklung
- Begrenzung technologiebedingter Probleme auf das ATM-Modul (PC und FPGA-Modul stehen als getestete und wiederverwendbare Hardware zur Verfügung)

Auf den ersten Blick erscheinen die Entwicklung des ATM-Moduls und der Interface-Komponente zum PC als zusätzlicher Aufwand und somit nachteilig. Dieser Aufwand ist aber relativ gering, da die Interface-Komponente in VHDL beschrieben wurde. Sie kann mit wenigen Modifikationen auch als Verbindung zum späteren Mikrocontroller wiederverwendet werden. Die Fertigung des ATM-Moduls ist sicherlich ein Aufwand nur zu Entwicklungszwecken, bietet aber die Möglichkeit, die technologiebedingten Probleme (siehe auch Kap. 2.3) des ALC-ASICs (208 Pin SQFP, 0,5 mm Finepitch) auf eine kleine Platine zu begrenzen.

Die zweite Testarchitektur zielt auf die Entwicklung des datenorientierten Teils und kann parallel mit Hilfe des Werkzeugs Mikrocontroller-Modul in Abbildung 3 durchgeführt werden. Die Vorteile dieser Testarchitektur sind folgende:

- Abtrennung der Entwurfsprobleme moderner Mikrocontroller. Für diese Anwendung wurde ein Embedded PowerPC 403GA ausgewählt. Seine Rechenleistung kann in Abhängigkeit von DRAM bzw. SRAM mit unterschiedlichen Busbreiten evaluiert werden. Zusätzlich kann die Wirkung seiner Caches auf harte Echtzeitanwendungen untersucht werden.

- Die Portierung eines Betriebssystems sowie die Entwicklung und der Test von Algorithmen für die Anwendung kann noch vor dem Integrationstest auf dem Zielsystem erfolgen. Dadurch erreicht man eine Parallelisierung von Soft- und Hardware-Entwurf (siehe auch Abbildung 1).

Der gesamte Mikrocontroller-Bus ist, wie in Abbildung 3 dargestellt, über ein Interface-FPGA mit dem Extension-Port verbunden. Dieser ist Pin-kompatibel mit dem des FPGA-Moduls. Dadurch wird ein schneller Integrationstest erleichtert. Das ATM-Modul kann nach Abschluß der getrennten Entwicklung einfach an das Mikrocontroller-Modul angesteckt (Abb. 3 unten) werden. Die Interface-Komponente kann an den Mikrocontroller angepaßt und in dem Interface-FPGA installiert werden. Die auf dem PC in C entwickelte Firmware wird mit Hilfe eines C-Compilers auf das Zielsystem portiert. Die modulare und kompatible Interface-Struktur der beiden Werkzeuge FPGA- und Mikrocontroller-Modul ermöglicht eine schnelle Integration des Gesamtsystems.

Der Systemtest schließt die Entwicklung ab. Dabei werden die im Integrationstest zusammengesetzten Funktionsteile im Bezug auf vorgegebene Randbedingungen getestet. Reale ATM-Netze wurden an das Diagnosesystem angeschlossen. Versuche zum Senden und Empfangen von ATM-Paketen konnten bereits erfolgreich abgeschlossen werden. Die Implementierung geeigneter Trigger- und Aufzeichnungsalgorithmen sind zur Zeit Gegenstand unserer Arbeiten.

5. Zusammenfassung

Im Bereich der industriellen Automation und Kommunikation bestehen Zielarchitekturen aus wenigen, dafür aber hochintegrierten Chips. Die großen Halbleiterhersteller bringen in immer kürzeren Abständen neue, innovative Bausteine (ASICs) auf den Markt. Bei nur einem Bruchteil der Kosten integrieren sie Funktionen auf einem Chip, die in der Vergangenheit noch vollständige Baugruppen benötigten. Auf der anderen Seite steigen die Entwurfsprobleme beim Einsatz solcher Komponenten drastisch an. Um dem Innovationsdruck zu begegnen, wird es insbesondere für mittelständische Unternehmen wichtig sein, diese Komponenten in immer kürzeren Entwurfszeiträumen in ihren Produkten zu nutzen.

Um die damit verbundenen Probleme bei der Systementwicklung zu beherrschen, ist eine systematische Entwurfsmethode notwendig. Diese basiert im wesentlichen auf der getrennten und parallelen Bearbeitung der Problemfelder. Um diese Methode durchführen zu können, haben wir Entwurfswerkzeuge entwickelt. Dabei handelt es sich um ein FPGA-Modul und ein Mikrocontroller-Modul sowie dedizierte, anwendungsspezifische Module. Diese wurden in mehreren, praxisorientierten Entwicklungsarbeiten [4][7][8] bereits getestet.

Die Vorteile können besonders gut am Beispiel unseres ATM-Diagnosesystems gezeigt werden. Dieses besteht aus einem ATM-spezifischen ASIC, welcher den gesamten steuerungsorientierten Teil realisiert und einem 32 Bit Mikrocontroller, auf dem der datenorientierte Teil implementiert wird. Diese Gesamtarchitektur wird mit Hilfe der Werkzeuge in zwei Testarchitekturen zerlegt. Die erste Testarchitektur adressiert die Lösung der ASIC-spezifischen Probleme (komplexe Firmware-Entwicklung), die zweite Testarchitektur die Lösung der Mikrocontroller-abhängigen Probleme (Evaluierung des Mikrocontrollers, Betriebssystem-Portierung, Entwicklung und Test der Algorithmen).

Die flexible und modulare Architektur der Werkzeuge ermöglicht nach Abschluß der Entwicklung beider Testarchitekturen eine schnelle Implementierung des Gesamtsystem. Im Rahmen des Integrationstests werden so die spezifischen Probleme der Bausteine schnell überwunden und man erreicht in kürzeren Entwurfszeiträumen ein funktionsfähiges Gesamtsystem.

Literatur.

- [1] W.Wolf: Hardware-Software Co-Design of Embedded Systems, Proceedings of the IEEE, Vol. 82, No.7, July 1994
- [2] M. Edwards, J. Forrest: A Development Environment for the Cosynthesis of Embedded Software/Hardware Systems. EDAC 1994, S. 469-473.
- [3] E.Mosanya, M. Goeke, J. Linder: A Platform for Co-design and Co-synthesis based on FPGA. Workshop on Rapid System Prototyping, S. 11-16, June 1996
- [4] V.Douridanova, K.Weiß: Architektur mehrkanaliger Hochleistungs-Master für ein Aktuator-Sensor-Interface (ASI). iNet Kongress Karlsruhe, Juni 1996
- [5] PPC403GA Embedded Controller - User Manual. IBM Corporation 1994
- [6] MPC860-Functional Design Specification. Revision 0.4, Motorola Inc., 1995
- [7] R. Kistner: Entwicklung eines Testgenerators/Monitors für ATM-Hochleistungsnetze. Diplomarbeit, Universität Karlsruhe, 1997
- [8] G.Haug: Konzeption und Implementierung einer 12MBaud fähigen PCMCIA-basierten Profibus-Monitor-Hardware. Diplomarbeit, Universität Tübingen, 1996
- [9] MB86687A - Adaption Layer Controller (ALC). Data Sheet Edition 2.0, FUJITSU 1996