Grafische Oberfläche für SIRIUS Prozessorkern auf FPGA

Andreas Kreker, Marc Durrenbeger,
Daniel Bau, Florian Zowislok, Dirk Jansen
Hochschule Offenburg, Badstraße 24
0781/ 205 179, akreker@stud.fh-offenburg.de

Zusammenfassung: Auf dem Markt existiert eine Vielzahl an PDAs. Alle haben einen sehr hohen Funktionsumfang und übertreffen sich von Generation zu Generation und erfordern einen hohen Entwicklungsaufwand von ganzen Entwicklerteams.
Der in dieser Arbeit entwickelte PDA mit seiner Hard- und Software soll kein Konkurrenzprodukt darstellen, sondern aufzeigen, was mit haus-internen Mitteln der Hochschule Offenburg möglich ist und gegebenenfalls eine Benutzer-oberfläche für bestehende oder noch kommende Projekte bilden.
Das hier entstandene Gerät ist im Akkumulator-Betrieb autonom und kann als eigenständiges System betrieben werden. Als Herzstück dient das Softcore SIRIUS Mikroprozessorsystem, das als VHDL-Modell in einem FPGA emuliert wird.
Zum Darstellen des grafischen Betriebsystems, welches speziell für dieses PDA entwickelt wurde, wird ein AMOLED-Display verwendet. Dieses besitzt ein Touchpanel, welches zur Steuerung des Systems genutzt wird. Softwareseitig sind Grundfunktionen zur Darstellung von Bildern und Texten entstanden, sowie Beispielanwendungen, die diese benutzen. Das grafische Betriebssystem ist modular und ermöglicht die direkte Weiterentwicklung von Anwendungen für das System.

1. Hardware: PDA-Platine

Die 6-Lagen Platine des PDAs wurde mit dem ALTIUM-Designer entwickelt. Sie wird einseitig bestückt und die Außenmaße betragen BxH: 52 x 54 mm. Die vollständig bestückte Platine ist in Bild 1 dargestellt. Das Herzstück bildet ein FPGA von Altera. Die Konfigurationsdaten des FPGA sind in einem Configuration Device gespeichert, welcher unterhalb des µSD-Karten-Slots zu sehen ist. Rechts daneben ist der 16 MBit-SRAM (Static RAM) von Cypress. Direkt darüber befinden sich im gelben Kasten die RTC mit ihrem 32 kHz – Quarz und im hell-grünen Kasten der Powertaster und ein RS-Flip-Flop. Im blauen Kasten ist der FPC – Connector (Flexible Printed Circuit) des Displays und der Touchcontroller zu sehen.

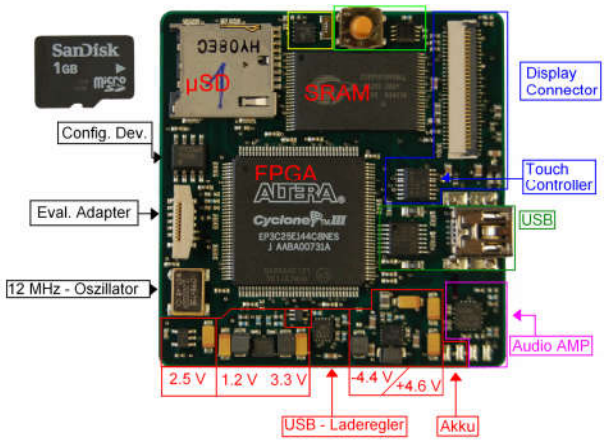

Bild 1: PDA-Platine

Der dunkel-grüne Kasten zeigt die USB-Buchse und den USB-Transceiver. Der pinke Rahmen beinhaltet den Audio-Verstärker und die Kontakte zum Lautsprecher. Neben diesen befinden sich die Kontakte, an denen der Akku angeschlossen wird. Alles was in Bild 1 rot umrandet ist, gehört zum Versorgungsteil. Oberhalb des 2,5 V – Linearreglers ist der 12 MHz Oszillator. Und darüber ist die Verbindungs-Buchse zum Evaluierungs-Adapter zu finden.

2. Konfiguration des FPGAs

2.1. Auslastung des FPGAs

Der verwendete FPGA (EP3C25) [9] gehört zur Familie der Cyclone III FPGAs von Altera. Er hat eine 144-Pin EQFP- Gehäuseform (Enhanced Quad Flat Pack). 83 der 144 Pins sind für den Benutzer frei verfügbar. Diese sind im PDA-Design, bis auf zehn Clock – Eingänge, alle verwendet worden. In Tab.1 sind weitere Daten zu den verwendeten Ressourcen des FPGAs.

Tab.1: Auslastung des FPGAs

	Verwendet	Verfügbar	Auslastung [%]
Logische Zellen	9.191	24.624	37
Pins	73	83	88
Speicher-Bits	525.312	608.256	86
PLL	1	4	25

2.2. SIRIUS Softcore

Das Herz des gesamten Systems ist der SIRIUS Softcore Prozessor. Er ist in VHDL (Very high speed integrated circuit Hardware Desciption Language) geschrieben und wird im FPGA emuliert. Neben dem SIRIUS existieren noch andere emulierte Controller wie der SPI- und USB- Controller. Fast die gesamten Speicherzellen des FPGAs werden für den internen 64 kB RAM des SIRIUS verwendet. In Bild 2 ist die gesamte Struktur des FPGA und seiner angeschlossenen Geräte dargestellt.

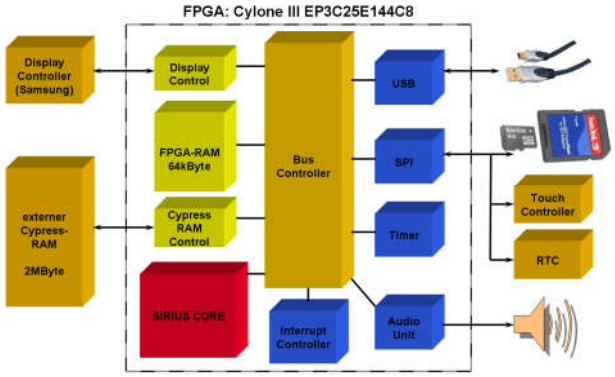

Bild 2: FPGA-Blockschaltbild

Neuerungen sind unter anderem die neue Audio-Unit, die es nun ermöglicht, Wave-Dateien auf dem Lautsprecher auszugeben. Sie entstand im Zuge einer Studienarbeit am IAF und ist mittels eines Delta-Sigma-Wandlers realisiert. Trotz dieses neuen Audio Interfaces ist das alte AHI (Accustic Human Interface) immer noch vorhanden und kann zur Ausgabe von Systemtönen benutzt werden. Da der gleiche Lautsprecher verwendet wird, kann jeweils nur eines der beiden Systeme genutzt werden.

Weitere Neuerungen sind die Komponenten externer SRAM und das Display, welches parallel über den Parallelbus-Controller angebunden ist. Am SPI-Bus befinden sich eine RTC (Real Time Clock), der Touch-Controller sowie die µSD-Karte.

Die Ergebnisse der Synthese des VHDL-Codes sind in Tab. 2 dargestellt.

Tab.2: Syntheseergebnisse

Komponenten	Logische Zellen	Memory Bits
SIRIUS Core	3547	-
USB	3843	1024
Interruptcontroller	415	-
Audio-Unit	356	-
FPGA-RAM	188	524288
SPI	158	-
Buscontroller	144	-
Timer	102	-
Restliche Komponenten	302	-

2.3. Speicheraufteilung des Systems

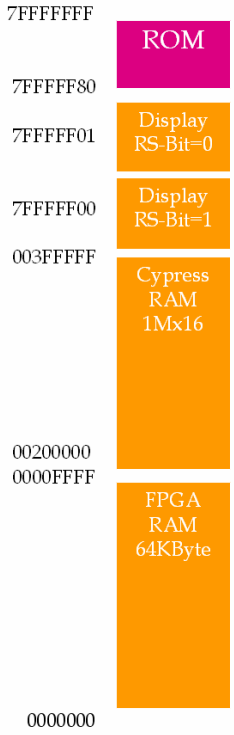

Bild 3: Speicheraufteilung

In Bild 3 ist die Speicher-Aufteilung des Systems dargestellt.

2.4. Externe SRAM

Für die externe RAM (CY62167DV30) sind zusätzlich 20-Adress-Bits nötig, um diesen adressieren zu können. Dies folgt aus der Größe des RAMs: 2^20 * 16 Bit = 16777216 Bit, also 16777216/(1024*1024) = 16 MBit. 16 MBit/8 = 2 MByte. Der SIRIUS ist prinzipiell ein 32-Bit Prozessor, dennoch wurde das alte Betriebssystem mit einem 16-Bit Compiler kompiliert. Um auch den oberen Speicherbereich adressieren zu können, muss dieser auf 32-Bit umgestellt werden. Das zweite Problem liegt darin, dass die externe RAM eine Latenzzeit von 45 ns besitzt und da der SIRIUS mit einem Takt von 48 MHz läuft, also für einen Takt 20,84 ns benötigt, ist er zu schnell für die RAM. Um nicht den gesamten Systemtakt zu vermindern, wurde ein Ready-Signal im SIRIUS Prozessor eingeführt. Dieses wird von einer Ansteuereinheit gesetzt, sobald die RAM-Operationen durchgeführt werden. Zum Beispiel lässt sie den SIRIUS bei einem externen RAM-Zugriff auf die Daten

des RAMs warten. In Bild 4 ist die Simulation des Ready-Signals zu sehen.

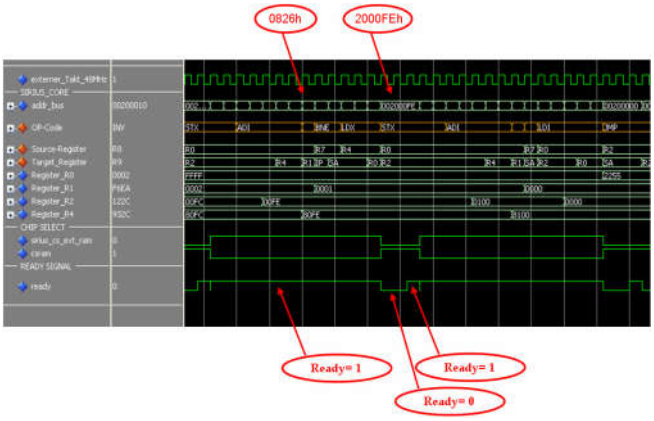

Bild 4: Simulation

Wird, wie das Beispiel in Bild 4 zeigt, auf die Adresse 0826h zugegriffen, handelt es sich um eine Adresse die kleiner 10000h ist, liegt sie also im internen RAM. Daraus folgt, dass das Ready-Signal auf logisch “1“ ist und somit der Prozessor ohne Unterbrechung weiterarbeiten kann. Bei Operationen die auf die interne RAM zugreifen, zum Beispiel arithmetische Operationen, läuft der SIRIUS mit seinem vollen Takt. Handelt es sich um eine Adresse die größer als eine 16-Bit Adresse ist, zum Beispiel um 2000FEh, ist das Ready-Bit auf logisch “0“ und der Ablauf wird um drei Takte verzögert. Der Vorteil liegt darin, dass das System nur verzögert wird, wenn auf das externe RAM zugegriffen wird.

2.5. Display

Bei der Ansteuerung des Displays ist eine direkte Adressierung nicht nötig. Hier reicht es, die Start- und Endposition des GRAM (Graphical RAM) per Datenbus dem Display-Controller (S6E63D6) zu übermitteln. Nach jedem geschriebenem Pixel inkrementiert der Display-Controller die RAM-Adresse automatisch. Dies ist möglich, da das GRAM selten wahllos beschrieben wird, sondern eher zeilenweise mit aufeinander folgenden Pixeln, die dann auch aufeinander folgende Adressen besitzen. Es ist am Parallel-Bus angeschlossen.

Das in Kapitel 2.4. beschriebene Problem mit den Antwortzeiten der Komponenten am Parallel-Bus wirkt sich beim Display-Controller genau so aus. Die Register- und RAM- Zugriffszeiten des Displaycontrollers variieren je nach Schreib- oder Lesevorgang zwischen 30 und 500ns. Bei einem Systemtakt von 48 MHz muss der SIRIUS Prozessor, um die Displayansteuerungszeiten einhalten zu können, angehalten werden. Auch hier kommt das READY Signal des SIRIUS zum Einsatz.

3. Software

3.1. Grafisches Betriebssystem

Wird der PDA gestartet, erscheint kurz das SIRIUS-Logo auf dem Display. Danach wird das Hauptmenu wie in Bild 5 angezeigt. Es zeigt maximal 20 Icons an, die bei Berührung eine Anwendung starten. Die Anwendungen können mit der PDA-Version der SIRIUS IDE kompiliert und als HEX-Datei auf die MicroSD-Karte kopiert werden. Das Icon für die Anwendung kann frei gewählt werden, sollte jedoch eine Auflösung von 48x48 Pixel haben und eine Farbtiefe von 32-Bit. Das Hauptprogramm, welches im “OS“ Ordner der SD-Karte liegt, erwartet die Anwendungen sowie die Icons im Wurzelverzeichnis. Um die beste Position für die Anwendung im Menü auszuwählen, ist eine alphabetische Einteilung realisiert, die es verlangt einen Buchstaben von A bis T sowohl für den Namen der Anwendung als auch für den des Icons zu wählen. Diese Aufteilung ist in Bild 6 dargestellt.

Bild 5: PDA-Menü

Aus dem Vergleich von Bild 5 und Bild 6 geht hervor, dass die Sonnenblume das Icon mit dem Namen “N.ICO“ sein muss, was auch bedeutet, dass die Anwendung, die mit dem Berühren dieser Sonnenblume gestartet wird, den Namen “N.HEX“ trägt. Die noch nicht belegten Platzhalter sind mit transparenten Icons versehen. Die HEX-Dateien der nicht benutzten Plätze müssen nicht vorhanden sein. Diese Art des Zugriffes auf die Anwendung ermöglicht eine leichtere Entwicklung von neuen Anwendungen, da das Hauptprogramm nicht erneut kompiliert werden muss.

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P
Q	R	S	T

Bild 6: Menu-Aufteilung

3.2. Beispielanwendung

Der Taschenrechner ist eine der Beispielanwendungen, die entstanden sind, um den Umgang mit den Funktionen und Treibern zu demonstrieren. Er ist auf die Grundrechenarten beschränkt und kann lediglich mit ganzen Zahlen rechnen. Der Aufbau des Taschenrechners ist einfach gestaltet. Die Oberfläche ist eine Bitmap-Datei, sie wird mit der bmp() – Funktion auf dem Display dargestellt und ist in Bild 7 zu sehen. Erfolgen Eingaben, so werden diese im Display des Taschenrechners dargestellt. Zum Aktualisieren der Anzeige wird aus Performancegründen nicht die ganze Bitmap erneut gezeichnet, sondern nur die durch die Eingaben veränderte Fläche.

Bild 7: Taschenrechner

Um die eingegebenen Zahlen und die Ergebnisse darzustellen, ist die Funktion writeString() zum Einsatz gekommen, welche Stringketten auf dem Display darstellt und zu der erstellten Schriftarten-Bibliothek gehört.

Die Bedienung des Rechners erfolgt über das Touchpanel. Zur Auswertung der Koordinaten und Rückgabe der gedrückten Taste wird die Funktion get_calc_pos() verwendet. Sie gehört zum erstellten Touch-Treiber.

Durch Eingabe des ersten Operanden wird eine Berechnung eröffnet. Danach muss ein Operator folgen. Nach diesem wird der zweite Operand erwartet. Ist dieser eingegeben muss mit dem “Gleichheitszeichen“ das Ergebnis abgerufen werden. Dieses steht automatisch als neuer erster Operand für eine weitere Berechnung zur Verfügung. Soll eine vollkommen neue Berechnung durchgeführt werden, ist die “C-Taste“ zu betätigen. Die Funktion der Tasten “.“, “m-“ und “mr/mc“ sind nicht implementiert. Beendet wird das Programm mit der roten Ausschalttaste.

3.3. Treiber und Bibliotheken

Für die Kommunikation mit dem Display-Controller werden vier Routinen benötigt. Diese sind in Assembler geschrieben, im BIOS verankert und können aus jeder späteren Anwendung, sofern als externe Routine deklariert, aufgerufen werden.

Routine 1: setIndex(U16)

Mit dieser Routine wird im Display-Controller festgelegt, welches seiner Register momentan angesprochen werden soll. Sie erwartet einen vorzeichenlosen 16-Bit Wert als Übergabe, welcher die anzusprechende Adresse beinhaltet. Diese kann sowohl eines der Register als auch das GRAM ansprechen.

Routine 2: setData(U16)

Diese Routine setzt einen Wert in das momentan gewählte Register oder in den GRAM. Im Falle eines Registers wird hier die Konfiguration des Registers übertragen. Ist die gewählte Adresse das GRAM wird hiermit der 16-Bit Farbwert eines Pixels übertragen.

Routine 3: DisplayReadStatus()

Wählt ein Register aus, das gelesen werden soll.

Routine 4: DisplayReadData()

Hiermit kann die Konfiguration eines Registers des Display-Controllers oder ein Pixelwert wieder ausgelesen werden, abhängig von dem mit DisplayReadStatus() gewähltem Register oder GRAM.

Bevor das Display nach einem Neustart funktionieren kann, muss es initialisiert werden. Dies wird beim Bootvorgang des grafischen Betriebssystems erledigt.

Aufbauend auf den Display-Treiber sind unter anderem Bibliotheken entstanden zum Darstellen von Windows Bitmaps und Icons, verschiedenen Schriftarten und geometrischen Figuren wie Rechtecke und Kreise.

Um das System bedienen zu können ist ein Touch-Treiber mit Auswerte-Logik entstanden, die in Kapitel 3.1. beschrieben wird.

4. Eckdaten

Hardware	
Display	
Größe	2.8"
Technologie	AMOLED
Auflösung HxB	320x240 Pixel
Anzahl Farben	65536
CPU	
Prozessor	SIRIUS
Takt	48 MHz
Speicher	
internes RAM	FPGA 64 kB
externes RAM	Cypress 2 MB
Daten	SanDisk 1 GB micro SD
Steuerung	
Typ	resistives Touchpanel
Akku	
Akkutechnologie	Lithium-Ionen
Akkukapazität	1150 mAh
Dauerbetrieb	5h
Anschlüsse	
USB	Aufladen des Akkus und Verbindung mit USB-Konsole
Größe und Gewicht	
Abmessungen	107 x 60 x 10,5 mm
Gewicht	134 g

Software	
Betriebssystem	grafisches PDA-OS
Unterstützte Dateiformate	
Ausführtbar	HEX (Intel)
Bild	BMP (1,4,8,16,24-Bit), ICO (32-Bit)
Audio	WAV 44,1 kHz 16-Bit
Text	TXT