

*Semesterarbeit am Institut für Technische Informatik
WS 2003/04*

System Self-Test for RHWOS Platform

Bortis Dominik
6. Februar 2004

Assistent: Herbert Walder
Professor: Prof. Dr. Lothar Thiele

Inhaltsverzeichnis

1	Einleitung	1
1.1	XFBOARD und Infrastruktur	1
1.2	Ziel des Selftest	2
1.3	Aufbau/Ablauf	3
2	Bedienung und Einstellungen des Selftests	5
2.1	Vorbereitungen	5
2.1.1	Anschliessen der Peripheriegeräte	5
2.1.2	Öffnen des Hyperterminals	5
2.2	Konfigurieren des XFBoards	7
2.2.1	Netzwerkverbindung prüfen	7
2.3	Aufstarten des Selftest	8
2.4	Voreinstellungen	8
2.4.1	Settings	8
2.4.2	Progress	9
2.4.3	C-FPGA	9
2.4.4	R-FPGA	13
2.5	Starte Selftest	13
2.6	Rapport öffnen	13
3	Interpretation der Resultate/Fehler	14
4	Programm	19
4.1	Paketaufbau	19
4.2	Programmablauf	21
4.3	Programmierung der Applikation auf dem PC (Visual C++)	22
4.4	Programmierung des Selftests auf dem XFBoard (Xilinx Platform Studio)	23
4.5	Detaillierte Beschreibung der Einzeltests und ihren Funktionen	23
4.5.1	RAM	23
4.5.2	Switches	26
4.5.3	LED	27
4.5.4	Simple VGA	28

4.5.5	Keyboard	29
4.5.6	RS232	30
4.5.7	SelectMAP-Port	30
4.5.8	Temperatursensor	31
4.5.9	Ende des Selftests	32
4.6	Report	32
5	Schlusswort	33
A	CD	34

Kapitel 1

Einleitung

1.1 XFBOARD und Infrastruktur

In Embedded Systems (ES) werden immer häufiger rekonfigurierbare Komponenten (CPLDs, FPGAs, etc.) eingesetzt, um besonders performante Teilaufgaben effizient abzuarbeiten (Co-Prozessor). Die Kapazität heutiger SRAM-basierter FPGAs (Field Programmable Gate Arrays) übersteigt bei weitem die Grösse von einzelnen Schaltungsblöcken, wie z.B. Krypto-Algorithmen, Filter (FIR/IIR), Multimedia Encoder/-Decoder, sogar ganzer CPU-Soft-Cores. Zusätzlich lassen sich moderne FPGAs partiell rekonfigurieren, d.h. einzelne Teilbereiche können zur Laufzeit verändert werden, während die restlichen FPGA-Ressourcen parallel dazu weiterarbeiten.

Dies eröffnet neue Einsatzmöglichkeiten für FPGAs: Mehrere voneinander unabhängige Schaltungsblöcke (HW-Tasks) können zur Laufzeit auf FPGAs geladen, abgearbeitet und wieder entfernt werden (Multitasking). Somit wird ein FPGA zu einer dynamisch allozierbaren Ressource, die von einem neuartigen Betriebssystem (Reconfigurable Hardware Operating System, RHWOS) verwaltet werden kann.

In einem OS-Umfeld werden Applikationen als kooperierende Tasks implementiert. HW-Tasks sind eigenständige Schaltungsblöcke, die eine Teilfunktion einer Applikation ausführen und über eine wohldefinierte Schnittstelle (Standard Task Interface, STI) mit OS-Elementen bzw. mit anderen Tasks kommunizieren.

In einigen vorangegangenen Semester- und Diplomarbeiten am TIK wurde basierend auf dem XESS-Board und Xilinx Virtex FPGAs ein lauffähiger Prototyp eines RHWOS und einige HWTasks konzipiert, entwickelt und implementiert. Der Funktionsumfang des OS war jedoch beschränkt und einige Funktionen konnten aufgrund von Einschränkungen bzw. der Architektur des XESS-Boards nicht realisiert werden.

Mit dem XF-Board, das ebenfalls im Rahmen einer Semesterarbeit am TIK entwickelt wurde, steht seit kurzem eine Plattform zur Verfügung,

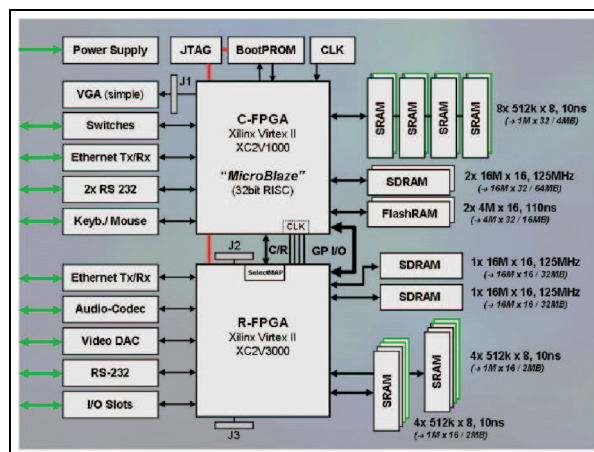


Abbildung 1.1: Blockschema des XFBoards

die den oben genannten Anforderungen zur Implementation eines RHWOS Rechnung trägt. Das XF-Board besteht im wesentlichen aus zwei miteinander gekoppelten FPGAs, dem C-FPGA und dem R-FPGA. Der C-FPGA ersetzt die CPU (gemäss Abb. 1.1), beinhaltet eine MicroBlaze-Soft-CPU sowie sämtliche Glue-Logic zur Ansteuerung der externen Komponenten, wie SRAM, SDRAM, FlashRAM, Ethernet- Driver, Keyboard/Mouse-Inteface, RS-232, simple VGA-Interface und Temperatursensoren. Der C-FPGA kontrolliert den Konfigurationsport des R-FPGAs und kann den R-FPGA mit Clock-Signalen versorgen. Am R-FPGA sind eine Reihe von externen I/O- und Speicherkomponenten angeschlossen: SRAM, SDRAM, Ethernet-Driver, Audio-Codec, Video-DAC, RS-232-Driver und Temperatursensoren.

1.2 Ziel des Selbsttest

Der Selbsttest soll es ermöglichen das XFBoard auf alle seine Komponenten zu testen und so die Funktionsfähigkeit des Boards zu überprüfen. Ziele bei der Implementierung des Selbsttests waren Einfachheit, Effizienz, Schnelligkeit, Übersichtlichkeit, Flexibilität, Benutzerfreundlichkeit, Sicherheit und aussagekräftige Resultate. Zusätzlich soll der Selbsttest dem Benutzer den Aufbau des Boards mit den einzelnen Komponenten näher bringen.

Das XFBoard-Projekt wurde erst in den letzten Semestern entwickelt und verfügt deshalb nur über einzelne Boards. Man sollte jedoch in Betracht ziehen, dass bei einer grossen Produktion von Boards die Möglichkeit besteht, diese in einer einfachen Weise zu testen.

Die Tests sollten zudem auch von nicht akademischen Arbeitskräften durchgeführt werden können. Es ist also wichtig, dass der Arbeiter möglichst schnell fähig ist, das Programm zu nutzen und somit die Boards auswerten

kann.

1.3 Aufbau/Ablauf

In diesem Abschnitt soll der Aufbau des Systemtests aufgezeigt werden. Zudem wird der Ablauf in einigen wichtigen Punkten aufgelistet, was später in den weiteren Kapiteln noch detaillierter dokumentiert wird.

Aufbau: Zur Durchführung des Systemtests werden ein PC und das zu testende Board benötigt. Diese beiden Geräte werden mit dem Netzkabel und dem JTAG verbunden. Das Netzkabel ermöglicht die Kommunikation über TCP/IP zwischen den beiden Komponenten. Die JTAG-Verbindung ermöglicht die Konfiguration des Boards.

Dies ist die Grundausstattung damit der Systemtest durchgeführt werden kann (Siehe Abbildung 1.2). Um die Schnittstellen zu externen Komponenten am Board zu testen werden noch weitere Peripheriegeräte wie Bildschirm, Tastatur, Maus und die RS232-Verbindung verlangt.

Ablauf: Nachdem die einzelnen Geräte angeschlossen und eingeschaltet sind, kann der Test beginnen.

1. Konfigurieren des XFBoards über die JTAG-Verbindung mit Hilfe der `download.bit`-Datei.
2. Starten der Test-Applikation auf dem PC. Hier werden die gewünschten Testroutinen ausgewählt.
3. Wird nun die **Start Test**-Taste gedrückt, so entnimmt das Programm der Oberfläche das sogenannte Bitmuster, das die einzelnen Funktionen aufruft.
4. Nun tauschen die beiden Teilsysteme (PC, Board) UDP-Datenpakete über die Netzwerkverbindung miteinander aus.
5. Wurde die Instruktion vom Board behandelt, so wird wiederum via UDP ein Antwort-Paket an den PC zurückgesendet.

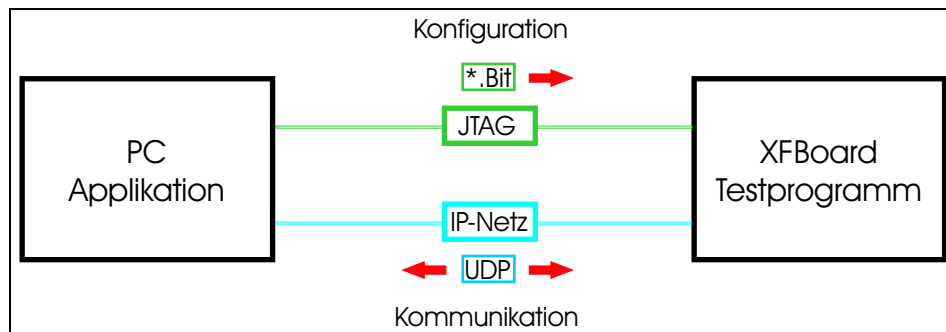


Abbildung 1.2: Aufbau und Ablauf des Selftests

Kapitel 2

Bedienung und Einstellungen des Selftests

2.1 Vorbereitungen

In diesem Kapitel wird Ihnen die Bedienung der Oberfläche und das Starten des Selftest Schritt für Schritt erklärt. Bevor Sie den Computer einschalten, müssen die einzelnen Peripheriegeräte angeschlossen und das Board mit dem PC verbunden werden.

2.1.1 Anschliessen der Peripheriegeräte

Schliessen Sie Netzgerät, Bildschirm, Tastatur, RS232-, JTAG- und Netzwerk-Kabel wie in der Abbildung 2.1 angezeigt an das XFBoard.

Schalten Sie nun den PC und den Strom ein. Der Schalter für das Netzgerät befindet sich auf der Rückseite des Gerätes.

2.1.2 Öffnen des Hyperterminals

1. Öffnen Sie im Menü **Start/Programme/Zubehör/Kommunikation** eine neue Hyperterminal-Verbindung.
2. Nennen Sie die Verbindung **Test** und klicken dann **OK**.
3. Wählen Sie die Schnittstelle, in der sie das RS232-Kabel angeschlossen haben (COM1 oder COM2) und klicken Sie wiederum **OK**.
4. Setzen Sie die Baud-Rate (Bits per Second) auf 9600 und Flow Control auf **none**. Die restlichen Einstellungen können in den Default-Werten belassen werden. Klicken Sie **OK** und die Hyperterminal-Verbindung ist aufgebaut.

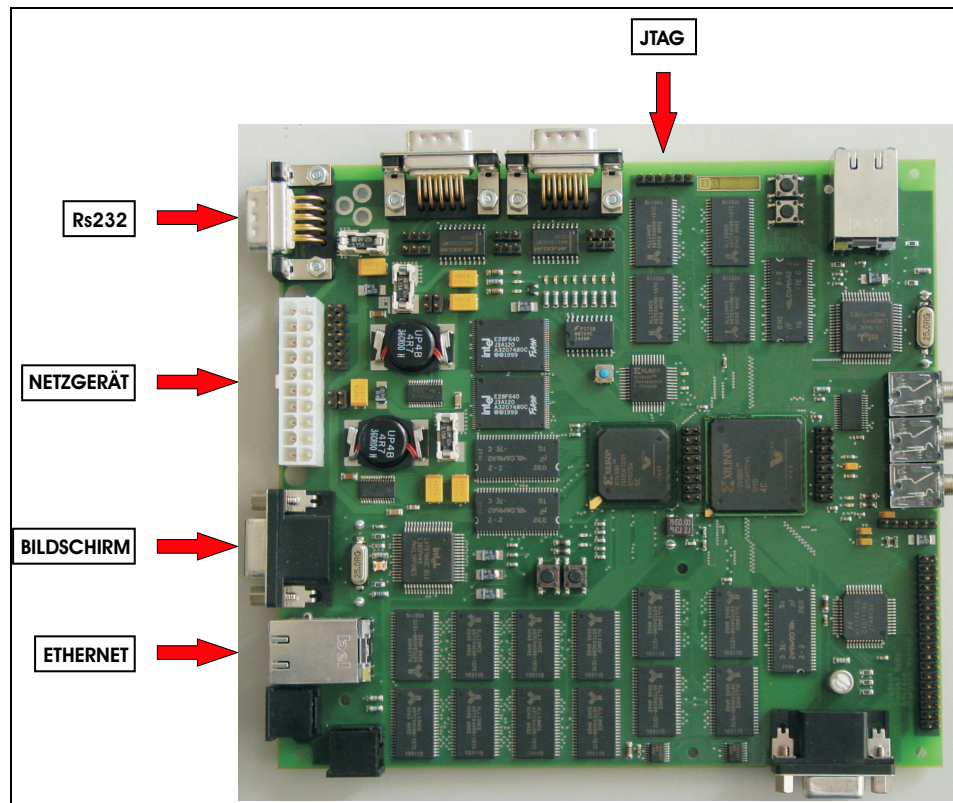


Abbildung 2.1: Anschliessen der verschiedenen Peripheriegeräte

2.2 Konfigurieren des XFBoards

Konfigurieren Sie nun das XFBoard.

1. Starten Sie `iMPACT` unter `Xilinx ISE/Accessoires`.
2. Es werden nun zwei Fenster geöffnet. Beim ersten Fenster drücken Sie jeweils `next` und zum Schluss `finish`. Das Programm sucht nun nach dem Board.
3. Sobald die FPGAs erkannt werden, zeigt das Programm 2 FPGAs an und öffnet ein Fenster, in dem Sie das Programm für die Konfiguration auslesen können. Drücken Sie beim ersten FPGA auf `Cancel` und beim zweiten FPGA wählen Sie die `download.bit`-Datei aus.
4. Klicken Sie mit der rechten Maustaste auf den zweiten FPGA und wählen `program` aus. Das Board wird konfiguriert.
5. Wurde die Konfiguration korrekt abgeschlossen erscheint auf blauem Hintergrund `Programming succeeded`

Es bestehen nun zwei Möglichkeiten um zu testen, ob das XFBoard richtig konfiguriert wurde. Im ersten Fall wird die Konfiguration mit dem Hyperterminal getestet. Sobald das Board konfiguriert ist, erscheint `Ethernet Package XFBoard ready . . .`. Wird im Hyperterminal nichts angezeigt, so wurde der FPGA nicht richtig konfiguriert.

Zweitens kann die Konfiguration mit einem Ping getestet werden. Pingen Sie das XFBoard an. Erhalten Sie einen `Reply Reply from 192.168.1.x`, so ist das Board korrekt konfiguriert. Falls `Request timed out` erscheint müssen Sie das Board nochmals konfigurieren.

2.2.1 Netzwerkverbindung prüfen

1. Klicken Sie unter dem Menü `Start` auf `Ausführen`. Es erscheint ein Fenster mit einer Edit-Box.
2. Geben Sie `cmd` in die Kommandozeile ein und klicken `OK`.
3. Eine Konsole wird geöffnet, in der sie die Zeile `ping 192.168.1.x` eingeben, wobei für `x` die Boardnr steht.
4. Drücken Sie nun die Enter-Taste und beobachten den Output.

2.3 Aufstarten des Selbsttest

1. Legen Sie die CD in das CD-ROM-Laufwerk.
2. Starten Sie die Anwendung `XFBoardTest.exe`. Die Applikation (Abbildung 2.2) wird gestartet.

Die Benutzeroberfläche des System Selbsttests ist in vier verschiedene Bereiche unterteilt: **C-FPGA**, **R-FPGA**, **Progress** und **Settings**. Zudem wird in der unteren rechten Ecke der Benutzeroberfläche das XFBoard angezeigt. Die einzelnen Teile des GUIs (Graphische User Interface) werden in den nächsten Abschnitten detailliert betrachtet.

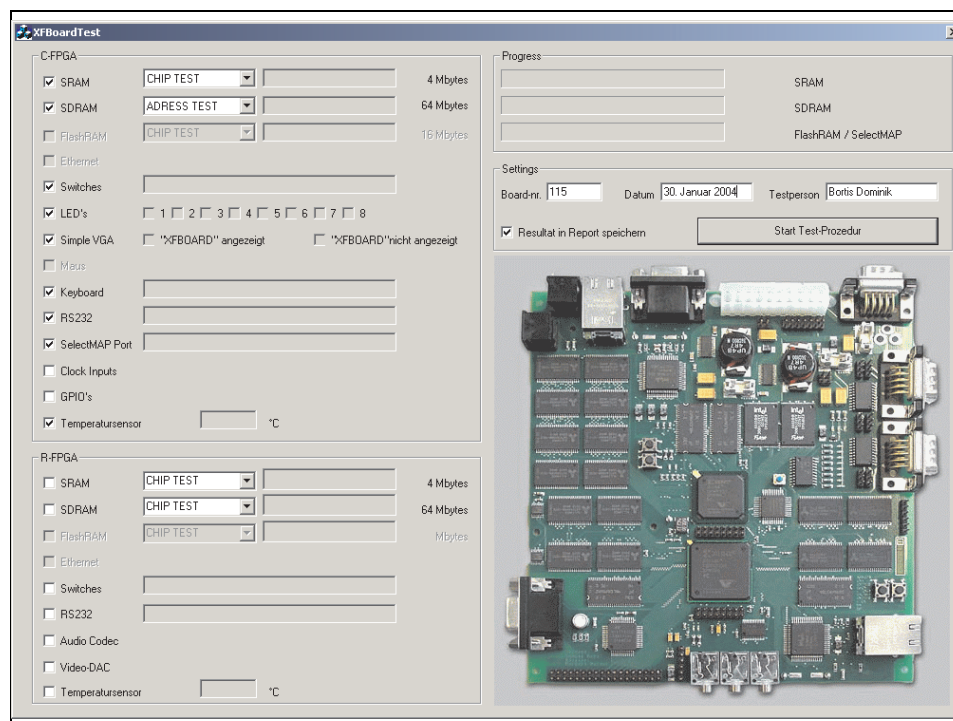


Abbildung 2.2: Graphische User Interface

2.4 Voreinstellungen

2.4.1 Settings

Der Bereich **Settings** besteht aus drei Eingabe-Feldern in denen die Boardnummer, die Testperson und das Datum eingetragen werden. Diese Angaben werden am Schluss im Report aufgeführt. Füllen Sie nun die einzelnen Felder aus. Als Boardnummer geben Sie die letzte Zahl der IP-Adresse an.

2.4.2 Progress

Progress enthält drei Statusanzeigen für die Funktionen SRAM, SDRAM, FlashRAM und SelectMAP Port. Wird eine dieser Funktionen gestartet, kann der Status in der jeweiligen Progressanzeige angegeben werden. Die Anzeige wurde nur für diese vier Funktionen implementiert, da die RAM-Tests und der SelectMAP Port-Test etwas länger dauern.

2.4.3 C-FPGA

Als nächstes werden die Testfunktionen des C-FPGA ausgewählt. Hierzu noch einige Bemerkungen zu den einzelnen Tests:

RAM-Test: Es werden nun die drei Tests SRAM, SDRAM und FlashRAM als ein Test beschrieben, da alle Tests dieselben Einstellungen benötigen. Wählen Sie die Speicher (SRAM, SDRAM und FlashRAM) aus, welche getestet werden sollen. Dies erfolgt indem Sie das kleine Quadrat anklicken und ein "Häcklein" erscheint. Es kann nun jeweils zwischen 3 verschiedenen Tests ausgewählt werden, die sich nur in ihrer Granularität unterscheiden.

Mit dem Chiptest testen Sie jedes Bauelement, mit dem Adresstest jede einzelne Adresse auf allen Chips und zuletzt dem Bittest jedes einzelne Bit im ganzen Speicherbereich. Umso genauer die Granularität wird, desto länger dauert der Test. Dies ermöglicht jedoch die Fehler besser zu detektieren. Wählen Sie nun die gewünschte Granularität aus.

Switches: Es werden keine Voreinstellungen benötigt. Führt das Programm den Switch-Test durch, wird eine Konsole (siehe Abbildung 2.3) geöffnet. Sie werden aufgefordert Switch 1 und Switch 2 nacheinander zu drücken. Tun Sie dies sobald das Fenster erscheint und drücken Sie anschließend die OK-Taste um den Switch-Test zu beenden.

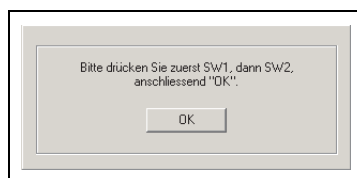


Abbildung 2.3: Switch-Fenster

Leds: Wählen Sie im Teil C-FPGA den LED-Test an, indem Sie das Quadrat des LED-Test anklicken. Sobald der LED-Test gestartet wird, erscheint ein weiteres Fenster (Abbildung 2.4). Bringen Sie die LEDs mit der **START**

LED-Taste zum Leuchten. Beobachten Sie gleichzeitig ob alle LEDs hintereinander aufleuchten. Durch mehrmaliges Drücken dieser Taste können Sie sich vergewissern welche LEDs wirklich brennen. Versehen Sie die Kästchen der intakten LEDs mit einem Hacken. Danach bitte OK drücken um den Test zu beenden, und die Daten zu speichern.

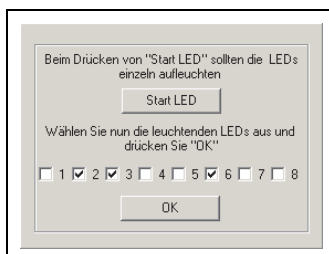


Abbildung 2.4: LED-Fenster

Simple VGA: Der Monitor muss eingeschaltet sein. In der Mitte des Bildes erscheint nach der korrekten Konfiguration ein weißer Strich. Nach dem Start des Simple VGA-Test wird ein Fenster (Abbildung 2.5) geöffnet, in dem Sie aufgefordert werden, die Anzeige auf dem Bildschirm zu überprüfen. Es erscheint nun XFBOARD auf dem Bildschirm. Die Schrift bewegt sich unter dauerndem Farbenwechsel eine gewisse Zeit nach oben. Bestätigen Sie diese Angaben mit der Taste JA falls XFBOARD angezeigt wird, ansonsten mit NEIN.

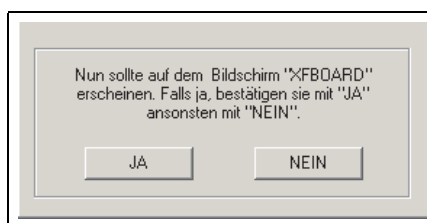


Abbildung 2.5: Simple VGA-Fenster

Keyboard: Das Programm öffnet eine Konsole, mit der Sie die Funktion des Keyboards testen (Abbildung 2.6). Schreiben Sie einen beliebigen Text. Der Text wird ins geöffnete Fenster übertragen. Nach einer gewissen maximalen Anzahl von Anschlägen wird der Text gelöscht, um einen Overflow vorzubeugen. Falls der von Ihnen eingegebene Text erscheint, betätigen Sie bitte die JA-Taste und anderenfalls die Nein-Taste.

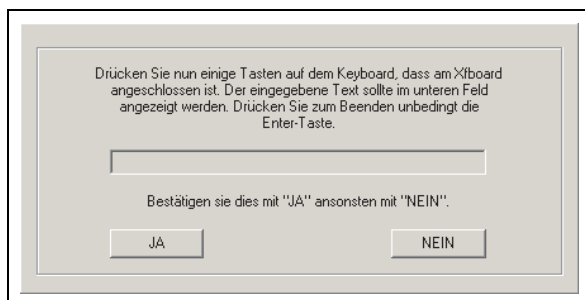


Abbildung 2.6: Keyboard-Fenster

RS232: Öffnen Sie eine Hyperterminal-Verbindung, falls diese noch nicht besteht. Bei der Ausführung des RS232-Test wird ein Fenster (Abbildung 2.7) angezeigt, in dem Sie aufgefordert werden das Hyperterminal zu betrachten.

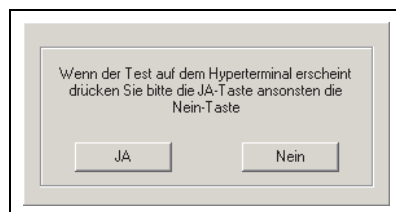


Abbildung 2.7: RS232-Fenster

Auf dem Hyperterminal wird der untenstehende Text des RS232-Testes angezeigt.

RS232-Test: Falls sie diesen Text sehen, koennen sie sicher sein das die RS232-Schnittstelle funktioniert.

Druecken Sie bitte die JA-Taste

Kontrollieren Sie dies und bestätigen Sie mit JA oder NEIN. Der Test wird somit beendet.

SelectMAP-Port: Beim Start des Tests füllt sich die Statusanzeige im Teil **Progress**. Diese zeigt wieviele Daten des Testprogramms bisher ins SRAM geschrieben wurden. Sobald der Datentransfer beendet ist, wird eine Konsole (Abbildung 2.8) eingeblendet. Mit der **JA**-Taste oder **NEIN**-Taste können Sie das Leuchten der LEDs bestätigen.

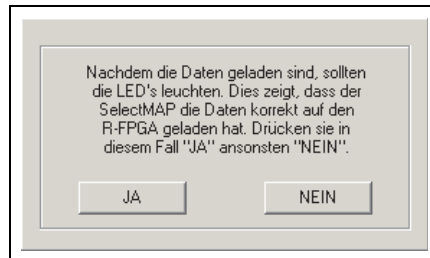


Abbildung 2.8: SelectMAP Port-Fenster

Temperatursensor: Der Test benötigt keine Voreinstellungen. Beim Start der Applikation erscheinen beim Temperatursensor schon die Initialwerte 0 C. Sobald der Test für den Temperatursensor durchgeführt wird, werden die Werte auf die momentane Temperatur gesetzt.

2.4.4 R-FPGA

Temperatursensor: Führen Sie den Temperaturtest des C-FPGAs durch, wird ebenfalls die Temperatur des R-FPGAs gemessen. Aus diesem Grund brauchen Sie nur den Temperaturtest des C-FPGAs anzuwählen.

Ende des Selftest: Wurde der Selftest komplett abgearbeitet, erscheint das letzte Fenster (Abbildung 2.9) in der Mitte der Applikation für ungefähr 3 Sekunden. Dieses Fenster zeigt Ihnen, dass der Test korrekt und vollständig durchgeführt wurde.

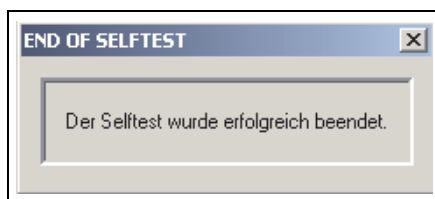


Abbildung 2.9: End of Selftest-Fenster

2.5 Starte Selftest

Alle Einstellungen wurde nun festgelegt. Um den Selftest nun zu starten, drücken Sie die Taste **START TEST**. Der Selftest testet nun die gewählten Funktion mit den eingegebenen Parametern selbständig durch. Zugleich wird im XFBoard-Bild jeweils die momentane Testfunktion angezeigt. Folgen Sie nun den Anweisungen des Programms.

2.6 Rapport öffnen

Zur Datensicherung wird, nachdem der gesamte Selftest durchgeführt wurde, ein Rapport erstellt. Sie können die Resultate aus der Datei `Report.txt` entnehmen. Die Datei befindet sich im Verzeichnis `C:/report.txt`.

Kapitel 3

Interpretation der Resultate/Fehler

RAM:

- Die Progress-Anzeige bewegt sich nicht: Das Problem liegt sicherlich momentan nicht am RAM. Als erstes sollte sichergestellt werden, ob das Board richtig konfiguriert wurde.

Öffnen Sie deshalb eine neue Hyperterminal-Verbindung. Das Erstellen einer Hyperterminal-Verbindung wurde im Abschnitt 2.1.2 beschrieben. Konfigurieren Sie das Board. Wird kein Text im Hyperterminal angezeigt, wurde das Board nicht richtig konfiguriert. Schalten Sie die Versorgung für einige Sekunden aus und versuchen Sie es noch einmal.

Erscheint im Hyperterminal `Ethernet Package XFBoard ready ...` so wurde sichergestellt, dass das Board richtig konfiguriert wurde. Es ist deshalb möglich, dass der Ethernet-Transceiver defekt sein könnte, da alle Tests auf der Netzwerkkommunikation basieren.

Versuchen Sie das Board anzupingen. Pinggen des Boards wurde im Abschnitt 2.2.1 beschrieben. Falls Sie Ping-Replys empfangen, bestätigt dies, dass der Ethernet-Transceiver funktioniert. Ansonsten besteht die Möglichkeit, dass der Ethernet-Transceiver defekt ist.

Haben Sie nun die Konfiguration mit dem Hyperterminal und den Ethernet-Transceiver mit pinggen getestet und beide Resultate positiv ausgefallen sind, so könnte es sein, dass die IP- und MAC-Adressen nicht richtig eingestellt sind. Starten Sie den Ethereal und sniffen Sie die Netzwerkverbindung. Kontrollieren Sie die Pakete auf Absender-IP, Absender-MAC, sowie Empfänger-IP und Empfänger-MAC.

- Einzelne Adressen sind falsch: Falls die defekten Adressen unsystematisch auftauchen, ist es sehr wahrscheinlich, dass die angezeigten Adressen defekt sind. Verifizieren Sie die Angaben, indem Sie den

RAM-Test noch einmal durchlaufen. Sollten dieselben Adressen angezeigt werden, so ist der Speicher an diesen Stellen defekt.

- Die defekten Adressen erscheinen in einem gewissen Muster: Das heisst, dass zum Beispiel jede 2.,jede 4.,jede 1.und 2. oder jede 128. Adresse angezeigt wird. Es kann sein, dass zwei Pins einen Kurzschluss bilden oder ein Pin nicht korrekt angelötet ist.

Ersteres bedeutet, dass die beiden Pins den gleichen Wert haben und somit nicht alle Adressen beschrieben werden können. Es bilden jeweils nur zwei benachbarte Pins einen Kurzschluss. Deshalb wird es einfach herauszufinden sein, welche zwei Pins denselben Wert besitzen. Sind Pin 1 und 2 kurzgeschlossen, so kann in keine Adresse geschrieben werden in denen Pin 1 und 2 unterschiedliche Werte besitzen. Somit sind immer 2 aufeinanderfolgende Adressen falsch. Bei Pin 2 und 3 werden 4 aufeinanderfolgende Adressen angezeigt und bei Pin 3 und 4 immer 8. Dies entspricht 2 hoch die niedrigere Pinzahl falsche Adressen. In Abbildung 3.1 wird dieses Verhalten für 4 Pins veranschaulicht.

	1. Adresse	2. Adresse	3. Adresse	4. Adresse	5. Adresse	6. Adresse	7. Adresse	8. Adresse
Pin 1	0	1	0	1	0	1	0	1
Pin 2	0	0	1	1	0	0	1	1
Pin 3	0	0	0	0	1	1	1	1
Pin 4	0	0	0	0	0	0	0	0

	9. Adresse	10. Adresse	11. Adresse	12. Adresse	13. Adresse	14. Adresse	15. Adresse	16. Adresse
Pin 1	0	1	0	1	0	1	0	1
Pin 2	0	0	1	1	0	0	1	1
Pin 3	0	0	0	0	1	1	1	1
Pin 4	1	1	1	1	1	1	1	1

2 Adressen falsch:

4 Adressen falsch:

8 Adressen falsch:

Abbildung 3.1: Kurzschluss von zwei Pins

Letzteres (nicht angeschlossenes Pin) bedeutet, dass das lose Pin irgendeinen Wert trägt. Hier gilt unter der Annahme, dass während des RAM-Test der Wert am losen Pin konstant bleibt, dass ebenfalls nicht an alle Adressen geschrieben wird. Somit wird eine Adresse angezeigt, wenn das erste Pin lose ist, 2 Adressen wenn beim 2. Pin , 4 beim 3. und so weiter. Es werden also 2 hoch Pinzahl minus 1 Adressen hintereinander als defekt angezeigt. Zudem kann es vorkommen, dass dieses Muster nur über einen gewissen Bereich auftaucht. Dies bedeutet, dass einer der beiden geschilderten Fälle genau auf einem Chip auftritt.

- Gruppen von Adressen sind falsch: Ist ein Chip defekt, werden beim Chiptest 2 aufeinanderfolgende Adressen detektiert. Beim Adresstest

wäre dann ein ganzer Bereich falsch, was für den SRAM-Test 1/8 der Adressen und beim SDRAM 1/2 der Adressen bedeutet, da der SRAM-Speicher aus 8 und der SDRAM-Speicher aus 2 Chips besteht.

- Alle Adressen sind falsch: Falls alle Adressen falsch ausgegeben werden, müssen Sie die Konfiguration des Boards nochmals überprüfen. Konfigurieren Sie das Board neu.

Switches:

- Einer der beiden Switches wird detektiert: Der Switch scheint nicht zu funktionieren. Prüfen Sie die Switches jedoch noch ein weiteres Mal, indem Sie den Test nochmals durchführen.
- Beide Switches sind defekt: Ist dies der erste Test, der ausgeführt wird, so kann es sein, dass das Board nicht richtig konfiguriert wurde. Prüfen Sie die Konfiguration mit einer Hyperterminal-Verbindung, den Ethernet-Transceiver mit einem Ping, wie es im Abschnitt 2.2.1 beschrieben wird.

LEDs:

- Die LEDs leuchten nur vereinzelt: Es ist klar, dass es ganz sicher an den LEDs liegt. Somit sind entweder die Leuchtdioden oder ein Vorwiderstand defekt. Prüfen Sie zudem die richtige Verlotung der LEDs.
- Alle LEDs leuchten nicht: Die Wahrscheinlichkeit, dass alle LEDs ausgebrannt haben, strebt gegen Null. Es ist möglich, dass der Fehler an einer anderen Stelle liegt. Sollte dies der erste Test sein, welcher ausgeführt wird, so besteht die Möglichkeit, dass das Board noch nicht richtig konfiguriert wurde. Prüfen Sie deshalb die Konfiguration mit einer Hyperterminal-Verbindung, den Ethernet-Transceiver mit einem Ping wie es im Abschnitt 2.2.1 beschrieben ist.
- Zwei LEDs leuchten zusammen auf: Die beiden LEDs sind kurzgeschlossen. Testen Sie die LEDs noch einmal und schauen Sie sich die Lötstellen genauer an.

Simple VGA:

- Bevor Sie den Test beginnen, vergewissern Sie sich, dass der Strich in der Mitte des Bildes angezeigt wird. Erscheint dieser nicht, wurde das Board nicht richtig konfiguriert. Prüfen Sie deshalb die Konfiguration mit einer Hyperterminal-Verbindung, den Ethernet-Transceiver mit einem Ping wie es im Abschnitt 2.2.1 beschrieben ist.

Keyboard:

- Der eingegebene Text wird nicht auf der Dialogbox angezeigt: Mögliche Gründe sind eine defekte Tastatur oder die fehlgeschlagene Konfiguration des Boards. Letzteres kann jedoch ausgeschlossen werden, wenn die vorherigen Test korrekt abgearbeitet wurden. Versuchen Sie den Test mit einer anderen Tastatur nochmals aus. Sollte auch in diesem Fall keine Anzeige auf der Applikation erscheinen, so ist der Fehler bei der Keyboard-Schnittstelle zu suchen.

RS232:

- Es erscheint keine Anzeige im Hyperterminal: Haben Sie zuvor den Keyboard-Test durchgeführt und vergessen die ENTER-Taste zum Beenden des Keyboard-Tests zu drücken? Wird die ENTER-Taste beim Keyboard-Test nicht betätigt, kann der Keyboard-Test auf dem Board nicht korrekt abgeschlossen werden. Das Board wartet weiterhin auf eine Eingabe auf der Tastatur und kann somit nicht auf ihre RS232-Anfrage reagieren.

Drücken Sie die Reset-Taste auf dem Board. Wird Ihnen der Text `Ethernet Package XBoard ready . . .` angezeigt, so sind Sie sicher, dass die RS232-Schnittstelle funktioniert.

SelectMAP Port:

- Die Statusanzeige bewegt sich nicht: Ist der Pfad der Datei richtig? Vergewissern Sie sich, dass sich die `download.bit`-Datei im richtigen Verzeichnis befindet, oder dass die Datei überhaupt existiert.

Haben Sie zuvor den Keyboard-Test durchgeführt und vergessen die ENTER-Taste zum Beenden des Keyboard-Tests zu drücken? Wird die ENTER-Taste nicht betätigt, kann der Keyboard-Test auf dem Board nicht korrekt abgeschlossen werden. Das Board wartet weiterhin auf eine Eingabe auf der Tastatur und kann somit nicht auf ihre SelectMAP Port-Anfrage reagieren. Starten Sie den Test noch einmal.

- Die Daten wurden übertragen aber die LEDs leuchten nicht: Testen Sie nochmal die LEDs. Falls die LEDs brennen, wurde der R-FPGA nicht richtig konfiguriert. Versuchen Sie es noch einmal, indem Sie das Board nochmals konfigurieren und nur diesen Test auswählen.

Temperatursensor:

- Haben Sie zuvor den Keyboard-Test durchgeführt und vergessen die ENTER-Taste zum Beenden des Keyboard-Tests zu drücken? Wird die ENTER-Taste nicht betätigt, kann der Keyboard-Test auf dem

Board nicht korrekt abgeschlossen werden. Das Board wartet weiterhin auf eine Eingabe auf der Tastatur und kann somit nicht auf ihre Temperatur-Anfrage reagieren. Starten Sie den Test noch einmal.

Kapitel 4

Programm

4.1 Paketaufbau

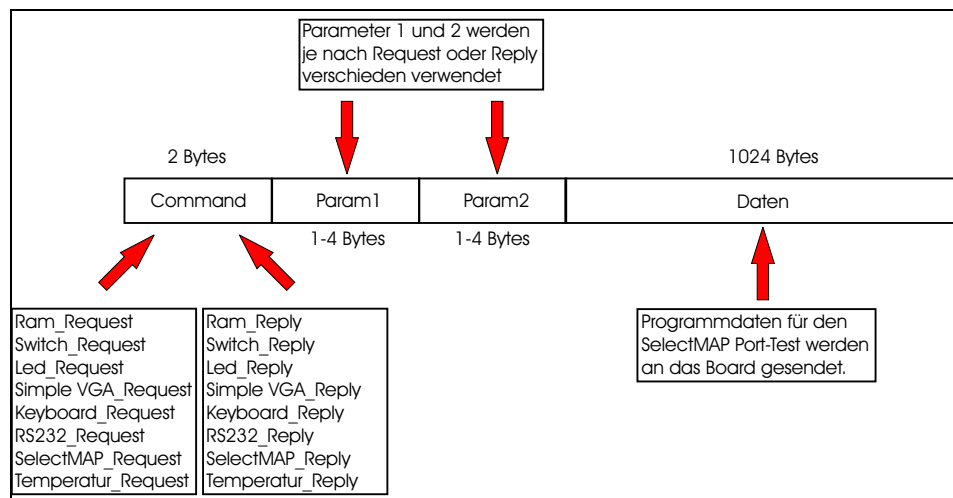


Abbildung 4.1: Paketformat

Zu Beginn der Modellierung musste ein Paketformat entwickelt werden, damit eine Kommunikation mit UDP-Protokoll überhaupt stattfinden kann. Dieses Problem wurde, wie in Abbildung 4.1 gezeigt, mit einem einfachen Aufbau gelöst. Die ersten 2 Bytes definieren den Header. Der Header ist entweder ein REQUEST oder ein REPLY, wobei REQUESTs (also Anfragen) nur vom PC und REPLYs (also Antworten) nur vom Board gesendet werden. Zudem wurde für jeden einzelnen Test ein REQUEST und ein REPLY definiert, was einem speziellen Zahlenwert entspricht. Die entsprechenden Befehle und Zahlenwerte sind in der folgenden Tabelle aufgeführt.

Command	Code	Parameter 1	Parameter 2
SRAM_REQ	0x1	8Bit 0=chiptest 8Bit 1=adresstest 8Bit 2=bittest	
SDRAM_REQ	0x2	8Bit 0=chiptest 8Bit 1=adresstest 8Bit 2=bittest	
FlashRAM_REQ	0x3	8Bit 0=chiptest 8Bit 1=adresstest 8Bit 2=bittest	
ETHERNET_REQ	0x4		
SWITCHES_REQ	0x5	8Bit 1=start test 8Bit 2=end test	
LEDs_REQ	0x6		
SimpleVGA_REQ	0x7		
MAUS_REQ	0x8		
KEYBOARD_REQ	0x9	8bit 3=start test 8Bit 4=end test	
SELECTMAP_REQ	0x10	8Bit 0=data send 8Bit 1=konfig.	16bit paketnr 32bit total len
RS232_REQ	0x11		
TEMPERATUR_REQ	0x12		
SRAM_REPLY	0x100	8Bit 0=progress 8Bit 1=test end 8Bit 2=defekt	32bit mom. Adr. 32Bit def. Adr.
SDRAM_REPLY	0x100	8Bit 0=progress 8Bit 1=test end 8Bit 2=defekt	32bit mom. Adr. 32Bit def. Adr.
Flash_REPLY	0x100	8Bit 0=progress 8Bit 1=test end 8Bit 2=defekt	32bit mom. Adr. 32Bit def. Adr.
ETHERNET_REPLY	0x103		
SWITCHES_REPLY	0x104	8Bit 1= sw1 ok 8Bit 2=sw2 ok	
LEDs_REPLY	0x105		
SimpleVGA_REPLY	0x106		
MAUS_REPLY	0x107		
KEYBOARD_REPLY	0x108	8Bit Character	
SELECTMAP_REPLY	0x109		
RS232_REPLY	0x110		
TEMPERATUR_REPLY	0x111		

Ein weiteres Element sind die 2 Parameter. Diese haben je nach Test, der im Moment abgearbeitet wird, eine variable Länge. Die Länge kann zwischen 8 und 32 Bit variieren. Die genauen Angaben können aus den beiden Tabellen entnommen werden. Zum Schluss werden noch 1024 Bits Daten an das Paket gehängt. Dies Feature wird jedoch nur beim SelectMAP Port-Test benutzt und somit erhalten die üblichen Pakete eine Länge von 2 bis 10 Bytes.

4.2 Programmablauf

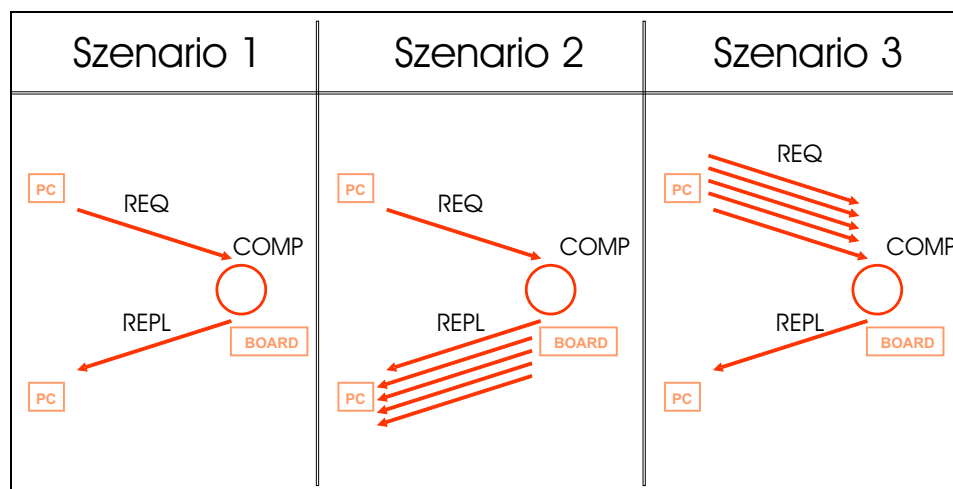


Abbildung 4.2: Szenarien zum Testen der Komponenten

Der Selbsttest besteht im Wesentlichen aus den verschiedenen Routinen auf dem PC und dem XFBoard. Der grösste Teil der Testprozedur fällt jedoch an die Kommunikation und Datenaustausch dieser beiden Systeme. In diesem Abschnitt soll nun genauer auf den Datenaustausch eingegangen werden. Es gibt 3 verschiedene Szenarien wie Komponenten getestet werden können. Die drei Varianten sind in der Abbildung 4.2 aufgelistet.

Die erste und einfachste Prozedur besteht aus einem REQUEST an das Board, der COMPUTING-Phase und dem REPLY zurück an den PC. Vertreter dieses Tests sind die LED, Simple VGA, RS232 und der Temperatursensor. Der PC sendet nun z.B. einen Temperatur-REQUEST, das Board misst die Temperatur und gibt sie in einem Temperatur-REPLY zurück.

Das zweite Szenario ist ein wenig komplexer. Vertreter dieses Typs sind RAM, Switch und Keyboard. Wie im Abschnitt 4.2 angesprochen, kann beim RAM-Test zwischen 3 Granularitäten ausgewählt werden. Der PC sendet nun z.B. als erstes einen RAM-REQUEST mit der Granularität als Parameter an das Board. Das Board beginnt in der COMPUTING-Phase die einzelnen Adressen zu testen. Nach einer festgelegten Intervallgrösse von

4.3 Programmierung der Applikation auf dem PC (Visual C++)

Adressen werden Status-Pakete an den PC zurückgesandt, die es ermöglichen eine Positionsanzeige auf dem GUI zu implementieren. Weiter werden Pakete an den PC gesandt, welche die defekten Adressen enthalten. Ist der Test nun beendet, kann das Board ein End-REPLY senden.

Das dritte Szenario entspricht genau dem Gegenteil des zweiten Szenario. Am Anfang werden mehrere REQUESTs an das Board gesendet und erhalten nur einen REPLY zurück. Der einzige Test dieses Typs ist der SelectMAP Port-Test. Zu Beginn werden vom PC Daten in mehreren Paketen an das Board gesendet und im SRAM gespeichert. Sobald alle Daten übertragen wurden, sendet der PC den Befehl zum Konfigurieren des R-FPGA. In der COMPUTING-Phase wird nun der R-FPGA konfiguriert und anschließend ein REPLY zurückgesendet.

4.3 Programmierung der Applikation auf dem PC (Visual C++)

Durch drücken der Start-Taste wird `OnStartTest()` aufgerufen. In dieser Funktion werden die ausgewählten Funktionen aus dem GUI (Graphisches User Interface) entnommen und in einen Bitmustervektor gespeichert, der für den entsprechenden Testablauf zuständig ist. Bei einem wiederholten Drücken der Start-Taste wird wiederum das Bitmuster eingelesen und die zusätzlichen Variablen des Programms zurückgesetzt.

Sobald die Werte im Bitmustervektor stehen, kann der eigentliche Test beginnen. Das Programm besteht aus verschiedenen Klassen. Es existiert je eine Klasse pro Dialogbox und zusätzlich noch eine Klasse für UDP. Die Hauptklasse `CXFBoardTestDlg` ist in 3 eigentliche Programmteile gegliedert. Dies wären die Funktion `Selftest()`, welche die jeweiligen Testfunktionen aufruft, der `PacketHandler()`, welcher die eingehenden Pakete vom Board entgegen nimmt, und die einzelnen Testfunktionen.

Die Funktion `OnStartTest()` ruft also nun die Hauptfunktion `Selftest()` auf, die anhand des Bitmuster die gewählten Funktionen startet. Mit der Variable `m_fkt_i` wird die momentan auszuführende Funktion im Sinne einer Nummer gespeichert. Nach dem Ausführen einer Testfunktion wird `m_fkt_i` inkrementiert und der nächste Test gestartet. Damit nun aber `m_fkt_i` nicht einfach dauernd inkrementiert wird, braucht es noch eine Variable `m_end_fkt`, die überprüft, ob die auszuführende Funktion schon beendet worden ist. Erst dann darf nämlich `m_fkt_i` inkrementiert werden. Wird eine Testfunktion aufgerufen, muss `m_end_fkt` auf 0 gesetzt werden. Beim Beenden der Funktion wird der Wert wieder zurück auf 1 gesetzt und erst dann kann `m_fkt_i` inkrementiert werden. Was die einzelnen Testfunktionen abarbeiten, sehen Sie im Abschnitt 4.5.

4.4 Programmierung des Selbstests auf dem XF-Board (Xilinx Platform Studio)

Das Testprogramm besteht aus den 2 c-Dateien `System.c` und `system_test.c`, wobei noch andere Dateien für den Ethernet Driver, das Keyboard und den SelectMAP Port hinzugezogen wurden. Diese sind jedoch für das Verständnis des Selbstests nicht von Bedeutung. Die Datei `System.c` ist als eine endlose WHILE-Schleife aufgebaut, in der auf eingehende Pakete gewartet wird. Dies musste so implementiert werden, da der MicroBlaze nicht auf Interrupts reagieren kann und somit kein Unterbrechen von Funktionen möglich ist. Es kann zum Beispiel nicht ein RAM-Test durchgeführt und zugleich noch auf einen Interrupt reagiert werden, wenn ein Paket erhalten wurde.

In dieser WHILE-Schleife wird geprüft, ob der Ethernet-Transceiver ein Paket erhalten hat.

Sodann setzt der Transceiver ein Flag und springt in den `PacketAnalyzer()`. Dieser überprüft das Paket und gibt die enthaltenen Daten in einem Pointer `ptrData` zurück. Als nächstes wird nun die Funktion `PacketHandler` in `system_test.c` aufgerufen.

Der `PacketHandler()` analysiert den Inhalt des Pakets. Das Paketformat wurde im Abschnitt 4.1 genauer beschrieben. Sind die Daten vom Paket entnommen, so kann die entsprechende Testroutine aufgerufen werden. Die einzelnen Routinen werden im Abschnitt 4.5 detaillierter beschrieben.

4.5 Detaillierte Beschreibung der Einzeltests und ihren Funktionen

4.5.1 RAM

Da der Ablauf der drei RAM-Tests ein und derselbe ist, wird in diesem Abschnitt der Programmteil allgemein als RAM-Test behandelt.

Beim Start der RAM-Tests wird von `Selftest()` die Funktion `sram()`, `sdram()` oder `flashram()` abgearbeitet. Bei allen Funktionen wird die jeweils zu testende Hardware auf dem XFboard-Bild mit einem Rechteck umrahmt.

Danach erhält das Programm mit der Abfrage `m_ComboRAM.GetCurSel()` den Wert der Granularität.

Der Rückgabe von `m_ComboSRAM.GetCurSel()` wird, abhängig von der Granularität, 0,1 oder 2 sein. Da die RAM-Tests etwas länger dauern, wird auf der Benutzeroberfläche der Stand des Testes angegeben. Anhand des Rückgabewertes von `m_ComboSRAM.GetCurSel()` kann somit die Länge der Progress-Anzeige gesetzt werden. Als weiteres sendet die Applikation ein `RAM_REQUEST`-Paket an das Board, in dem ebenfalls der Wert der Granularität als Parameter übergeben wird.

Erkennt nun das Board das erhaltene Paket als `RAM_REQUEST`, wird durch das Modul `sram()`, `sdram()` oder `flashram()` der RAM-Test durchgeführt. Dieser ist in vier Schritte unterteilt, welche die folgenden wären:

1. Setzen der verschiedenen Konstanten anhand der gewählten Granularität.
2. Vollständiges Schreiben einer 32Bit langen 10-Sequenz in den Speicher.
3. Zurücklesen des Speichers und vergleichen der gelesenen Daten mit der 10-Sequenz.
4. Resultate an die Test-Applikation zurücksenden.

Das Spezielle am RAM-Test ist, dass die einzelnen Tests (Chiptest, Adresstest oder Bittest) die gleichen Schleifen zum Schreiben in, und Lesen aus dem Speicher benutzen. Dies verkürzt den Programmcode um ein Vielfaches. Damit dies implementiert werden kann, müssen zu Beginn der Routine noch diverse Variablen gesetzt werden. Anhand der gewählten Testgenauigkeit werden zum Beispiel die Iterationslänge der Schleifen oder die Anzahl der Schleifendurchläufe bestimmt. Mit Hilfe der Variablen `End` wird die Iterationslänge festgelegt. Für den Chiptest beträgt `End` 2, da pro Chip nur 2 Adressen geprüft werden, und für den Adresstest wird `End` der Grösse des RAM-Adressraumes gleichgesetzt, da jede Adresse auf dem Chip getestet werden soll.

Im Schleifenmanagement braucht es, wie oben erwähnt, noch eine weitere wichtige Variable `Chip`, welche die Anzahl der zu testenden Chips und somit die Anzahl der Schleifendurchgänge festlegt. Nur beim Chiptest ist die Variable `Chip` ungleich 0, da beim SRAM 8 Chips und beim SDRAM 2 Chips getestet werden.

Damit der Benutzer am PC den RAM-Test mittels Progress-Anzeige mitverfolgen kann, werden nach einer bestimmten Anzahl Iterationen `RAM_REPLY`-Pakete mit der momentanen Position zurückgesendet. Der erste Parameter ist auf 0 gesetzt, was einem Progress-Reply entspricht und der zweite Parameter enthält den erwähnten Status.

Die Intervalllänge wird mit der Variablen `Step` definiert. Beim Chiptest wird nach jeder Iteration ein Paket versendet, da nur 2 Adressen pro Chip getestet werden; somit ist `Step` gleich 0. Die Schrittweite des Adresstests hängt von der zu prüfenden Hardware ab. `Step` beträgt für das SRAM 4096 (0x1000). Beim SDRAM wird auf Grund seiner Grösse an jede 1048576 (0x100000.) Adresse ein Paket zurückgesendet. Je kleiner `Step` gewählt wird desto glatter ist der Verlauf der Anzeige, aber desto mehr Pakete müssen verschickt werden.

Weiter benötigt es für die Progress-Anzeige noch eine zweite Variable `Offset`. Diese wird nur beim Chiptest gebraucht, damit der zu sendende

Wert der Progress-Anzeige immer um 1 inkrementiert wird und so der Verlauf der Skala linear nach oben verläuft.

Bemerkung: Die Auswahl, nach wie vielen Adressen ein Paket gesendet werden soll, wurde am Anfang mit einer Modulo-Operation implementiert. Das heisst, dass bei $(\text{Adresse}) \bmod 10000 = 0$ nach 10000 Adressen ein Paket versendet wird. Nach mehrmaligem Testen wurde ersichtlich, dass die Modulo-Operation sehr zeitaufwendig war und deshalb durch eine simple IF-Schleife ersetzt wurde, was den Zeitaufwand für 64 Mbytes von einigen Minuten auf einige Sekunden reduzierte.

Der RAM Test kann nun beginnen. Als erstes schreibt der Prozessor den Wert 0xAAAAAAAA (binär: 32bit lange 10 Sequenz) in alle Adressen. Es sei nochmals erwähnt, dass beim Chiptest nur zwei Adressen pro Chip und beim Adresstest alle Adressen beschrieben werden. Der Wert 0xAAAAAAAA wurde so gewählt, dass es für die Hardware besonders "schwierig" wird den Wert korrekt zu schreiben oder besser ausgedrückt, dass es am wahrscheinlichsten ist, dass eine Fehlfunktion der Bausteine oder Zuleitungen erkannt wird.

Nachdem die Adressen mit der 10-Sequenz gefüllt wurden, kann jede Adresse in derselben Reihenfolge wieder zurückgelesen werden. Der Wert der Speicherzelle wird mit der 10-Sequenz verglichen. Sollten die Werte nicht übereinstimmen, wird direkt angenommen das die Adresse defekt ist. Deshalb wird ein RAM_REPLY-Paket mit den Parametern 1 = 2, welches einem Defekt-Reply entspricht, und dem Parameter 2, der die defekte Adresse enthält, an den PC gesendet. Am Schluss sendet das Board noch ein End-Paket, welches das Testende angibt. Dies geschieht indem der Parameter 1 auf 1 gesetzt wird.

Am PC werden die verschiedenen REPLY-Pakete vom `PacketHandler()` bearbeitet. Es werden mit Hilfe des Parameter 1 die 3 möglichen Pakettypen unterschieden.

1. Wird der Parameter 1 = 0 erhalten, bedeutet dies, dass ein Progress-Paket erhalten wurde, wobei der Parameter 2 der neuen Position entspricht, die an `m_progress_ram.SetPos(param2)` übergeben wird.
2. Ist der Wert vom Parameter 1 = 2, so wurde ein Paket mit einer defekten Adresse erhalten, die mit dem Parameter 2 übergeben wird.
3. Als letzte Möglichkeit kann der Parameter 1 noch gleich 1 sein. Dies bedeutet, dass der RAM-Test beendet wurde. Der nächste Test kann begonnen werden, indem `m_fkt_end` 1 gesetzt wird und somit `Selftest()` startet.

4.5.2 Switches

Um die Switches zu testen wird als erstes von der Funktion `Selftest()` der Klasse `CXFBoardTestDlg()` auf der PC-Seite die Funktion `switches()` aufgerufen. In `switches()` werden mit der Funktion `OnPaint()` zwei Pfeile (auf die beiden Switches zeigend) in das XFBoard-Bild gezeichnet. Als zweites sendet `switches()` ein Paket mit dem Kommando `SWITCHES_REQUEST` an das Board. Der erste Parameter ist 1, falls der Test gestartet werden soll und 0, falls der Switchtest beendet werden soll. Dies muss nur beim Switchtest und Keyboardtest ausgeführt werden.

Der Grund ist einfach. Nehmen wir an, dass der Switchtest in einer `WHILE`-Schleife implementiert wird, in der auf die beiden Signale der Switches abgewartet wird. Das heisst, die beiden Switches werden gedrückt, das Board erkennt dies und es kann die `WHILE`-Schleife verlassen und ein Antwortpaket zurücksenden.

Das Problem ist nun aber: Was passiert wenn ein Switch nicht funktioniert? Trifft dieser Fall ein, wartet das Board in der `WHILE`-Schleife auf ein Signal vom Switch. Dieser funktioniert aber nicht. Das Programm wäre in einer Endlos-Schleife und könnte keine Pakete zum Abbrechen des Testes empfangen. Aus diesem Grund wird die Funktion in der `WHILE`-Schleife des Ethernet-Drivers implementiert, in der abwechselnd abgefragt wird, ob ein Paket empfangen, oder ein Switch gedrückt wurde. Jedoch wird die Switch-Abfrage nur durchlaufen, wenn der PC das Startpaket für den Switchtest gesendet hat.

Wird nun das generierte Startpaket mit dem Parameter 1 gleich 1 vom PC an das Board gesendet, analysiert der `PacketHandler` das Paket. Ist es ein Switchaufruf, wird der Parameter 1 vom `PacketHandler` als Returnwert an `system.c` zurückgegeben. Dadurch, dass zum Beginnen des Test der Wert 1 gesendet wird, kann nun in die `IF`-Schleife gesprungen werden, um die Switches zu testen. Das Board befindet sich immer noch in der `WHILE`-Schleife und kann somit weiterhin Pakete empfangen, aber zusätzlich werden nun auch noch die Switches beobachtet.

Zur gleichen Zeit wird auf der PC-Seite ein weiteres Fenster für den Switchtest geöffnet, welches das weitere Vorgehen beschreibt. Diese Konsole besitzt seine eigene Klasse `CDlgSwitchesTestC()`. Das heisst, dass das Drücken einer Taste in diesem Fenster von der Klasse `CDlgSwitchesTest()` abgefangen wird und je nach dem die zugehörige Funktion aufgerufen wird.

Es müssen nun die beiden Switches auf dem Board gedrückt werden. Erkennt dies das Board, so wird ein Antwortpaket für den jeweiligen Switch mit dem Kommando `SWITCH_REPLY` zurückgesendet. Mit dem Parameter 1 gleich 1 wird angegeben, dass Switch 1 gedrückt wurde und mit Parameter 1 gleich 2 wird Switch 2 bestätigt.

Die erhaltenen Pakete werden auf dem PC vom `PacketHandler` analysiert

und im Array `m_switchtest[2]` wird entweder die erste oder zweite Stelle auf eins gesetzt.

Der Test kann nun mit der OK-Taste beendet werden. Als Folge schliesst sich das Fenster. Das Programm springt in die Funktion `OnOk()`. Diese Funktion wird in der Klasse `CDlgSwitchTestC()` aufgerufen, da das Dialogfenster dieser Klasse angehört.

Die Funktion `OnOk()` meldet den Abbruch des Testes mit dem Aufruf der Funktion `OnSwitchTestCOK()` der Klasse `CXFBoardTestDlg()`. In `OnSwitchTestCOK()` werden nun die Resultate vom Array `m_switchtest[2]` in die Ausgabefelder und in den Repport geschrieben. Zusätzlich sendet die Funktion noch das letzte Paket zum Abbrechen der Switchfunktion auf der Boardseite. Dies geschieht indem der Parameter 1 auf 2 gesetzt wird und somit nicht mehr in die IF-Schleife des Switchtests gesprungen werden kann. Die Variable `m_end_fkt` erhält den Wert 1 und damit kann der nächste Test abgearbeitet werden.

4.5.3 LED

Die Hauptfunktion `Selftest()` ruft die Testfunktion `led()` auf. `led()` zeichnet mit Hilfe der Funktion `OnPaint()` ein Rechteck um die LEDs auf dem XFBoard-Bild. Zudem öffnet sie ein weiteres Fenster auf dem Bildschirm. Das Fenster besteht aus den 2 Tasten `Start LED` und `OK` sowie aus 8 Checkboxes, je eine für jede LED.

Dieses Fenster bildet für sich die Klasse `CDlgLedTestC()`. Die Elemente (Tasten und Checkboxes) dieses zusätzlichen Fensters gehören also ebenfalls zur Klasse `CDlgLedTestC()`. Die erste Taste `Start LED` ruft die Funktion `OnStartLED()` der Klasse `CDlgLedTestC()` auf.

Diese Funktion informiert in der Hauptklasse `CXFBoardTestDlg` die Funktion `OnStartLEDC()`, dass der Led-Test gestartet wurde. Nun wird ein Paket mit dem Kommando `LED_REQUEST` zum Board gesendet, welches das Board auffordert, die LEDs einzuschalten.

Empfängt das Board das Paket, wird vom `PacketHandler()` in der Datei `system_test.c` die Funktion `led()` gestartet. In dieser Methode werden mit einem leicht veränderten Zählers die LEDs reihenweise zum Leuchten gebracht. Danach ist diese Methode schon beendet. Sie kann jedoch durch mehrmaliges Drücken des `Start LED`-Buttons wiederholt aufgerufen werden. Der Ablauf ist derselbe, wie gerade beschrieben wurde.

Sobald die Person sicher ist, welche LEDs funktionieren, können die intakten LEDs in den Checkboxes ausgewählt werden und die OK Taste gedrückt werden. Als Folge schliesst sich das Fenster. Zudem wird die Funktion `OnOk()` der Klasse `CDlgLedTestC()` gestartet, in der die Resultate in das `led_check_array[8]` gespeichert werden. Damit die Ergebnisse auch angezeigt werden, wird das `led_check_array[8]` an die Funktion `OnLedTestOKC()` der Klasse `CXFBoardTestDlg` weitergeleitet. In `OnLedTestOKC()` werden die

Resultat im GUI angezeigt und in den Repport geschrieben.

4.5.4 Simple VGA

Wie bei allen Tests wird auch hier am Anfang der Funktion `SimpleVGA` die zu testende Komponente im Xfboard-Bild visualisiert. Weiter wird eine Konsole für den VGA-Test geöffnet und ein `SIMPLEVGA_REQUEST`-Datengramm an das Board gesendet, welches den Test auf dem Board startet.

Das erwähnte Fenster besteht aus den zwei Tasten `JA` und `NEIN`, die zur Klasse `CDlgSimpleVGATestC` gehören.

Das Board startet nach dem `SIMPLEVGA_REQUEST` die Routine `simple_VGA()` in der Datei `system_test.c`. Hier werden durch mehrere `FOR`-Schleifen die Buchstaben `X,F,B, O, A, R` und `D` an bestimmte Speicheradressen des VGA-Cores schreiben. Der VGA-Core scannt danach seinen Speicherbereich ab und gibt die enthaltenen Werte auf dem Bildschirm aus. Der Text wandert in ständigem Farbenwechsel nach oben. Dies wird mit den beiden Registern `lineRegister` und `colorRegister` ermöglicht. Für die genaue Benützung des VGA-Cores muss jedoch die jeweilige Dokumentation gelesen werden. Werden nun die oben erwähnten Tasten `JA` oder `NEIN` gewählt, wird (abhängig von der gepressten Taste) die Funktion `OnButtonJa()` oder `OnButtonNein()` aufgerufen.

Diese beiden Funktionen wiederum rufen die Methode `OnSimpleVGATestC()` der Klasse `CXFBoardTestDlg` auf, wobei von der Funktion `OnButtonJa()` der Wert 1 und von der Funktion `OnButtonNein()` der Wert 0 übergeben wird. Der übergebene Parameter gleich 1 entspricht einer Bestätigung, dass `XFBOARD` auf dem Bildschirm erschienen ist und der Parameter gleich 0 bedeutet somit das Gegenteil. Das Resultat kann von `OnSimpleVGATestC()` in der Applikation ausgegeben werden und dieses in den Rapport schreiben.

4.5.5 Keyboard

Nachdem das Rechteck um das Keyboard-Interface gezeichnet wurde, wird ein weiteres Fenster dargestellt.

Zusätzlich sendet die Applikation ein `KEYBOARD_REQUEST`-Datengramm an das Board. Das Keyboard-Testfenster enthält eine Eingabe-Box und die beiden Taste `JA` und `NEIN`. In der Eingabezeile werden die empfangenen Keyboard-Eingaben visualisiert.

Auf der Boardseite muss die Keyboard Routine speziell behandelt werden. Es besteht nämlich dasselbe Problem wie beim Switchtest. Dies sei hier nochmals schnell erwähnt. Das Problem entsteht, sobald der Test in einer `WHILE`-Schleife in der `system_test.c`-Datei implementiert werden muss. Wird ein Tastendruck vom Board erkannt, kann die `WHILE`-Schleife verlassen werden. Funktioniert jedoch die Tastatur nicht korrekt muss das Programm in der `WHILE`-Schleife verharren und das Board kann ohne `RESET` nicht mehr weiter arbeiten. Deshalb wird der Keyboard-Test, wie der Switch-Test, in der bereits bestehenden `WHILE`-Schleife der Datei `System.c` implementiert. Das Board kann jedoch den Keyboard-Test erst durchführen, sobald das `KEYBOARD_REQUEST`-Datengramm mit dem Parameter 1 gleich 3 erhalten wurde. Dieser Wert wird von `PacketHandler()` zurückgegeben und ermöglicht damit die Durchführung der `IF`-Bedingung für den Test.

Die Tastatur kann nun benützt werden. Mit Hilfe des Funktionsaufruf `kbd_getc()` vom Keyboard-Core kann aus dem erhaltenen Scancode der entsprechende Character generiert werden, welcher in der Variable `Buchstabe` gespeichert wird. Danach wird `Buchstabe` als Parameter 1 zurückgesendet. Es können so viele Eingaben erfolgen, wie erwünscht sind. Für jeden Tastendruck wird somit ein Paket gesendet.

Das Paket wird vom `PacketHandler()` in Empfang genommen. Von da wird der empfangene Character an die Funktion `CharReceive` der Klasse `CDlgKeyboardTestC` weitergegeben. In der Funktion `CharReceive(char m_buf)` wird der bestehende String aus der Eingabe-Box ausgelesen, der neu erhaltene Character angehängt und wieder am GUI angezeigt. Alle eingegebenen Zeichen erscheinen nun im zweiten Fenster.

Sobald der Test beendet werden soll, muss die Taste `JA` oder `Nein` gedrückt werden. Dadurch wird entweder die Methode `OnJa()` oder `OnNein()` aufgerufen.

Diese Funktionen bilden eine Verbindung zur Klasse `CDlgKeyboardTestC` dar. Es wird entweder `OnKeyboardJa()` oder `OnKeyboardNein()` aufgerufen. Zum Beenden des Keyboard-Tests auf der Boardseite wird kein weiteres Paket gesendet, sondern es muss die `ENTER`-Taste gedrückt werden. Die entsprechenden Resultate werden zum Schluss an die graphische Oberfläche und den Report übertragen.

4.5.6 RS232

Der RS232-Test umrahmt als erstes, wie die vorigen Tests, die Komponente, welche getestet wird. Als nächstes sendet die Funktion `rs232()` einen `RS232_REQUEST` an das Board. Es öffnet sich eine weitere Konsole der Klasse `CDlgRS232TestC`. Der Inhalt des RS232-Fensters besteht aus einem Text, der dem Benutzer die weiteren Schritte erklärt. Auf der Boardseite wird das Paket vom `PacketHandler` wie gewohnt bearbeitet und die `RS232-Routine` gestartet. Im Hyperterminal wird ein Text (siehe Abschnitt 2.4) auf der RS232-Schnittstelle ausgegeben. Auf dem PC kann als Bestätigung entsprechend die `JA-` oder `NEIN-Taste` gedrückt werden. Somit wird in der Klasse `CDlgRS232TestC` die Funktion `OnJa` respektive `OnNein` aufgerufen.

Diese Routinen starten zugleich die Funktion `OnRS232()` in der Klasse `CXFBoardTestDlg` und übergeben dieser den Parameter `resultat`. Wurde die `JA-Taste` gedrückt, beträgt der Wert von `resultat` 1, für die `Nein-Taste` 0. Das entsprechende Resultat kann am GUI (Graphische User Interface) ausgegeben werden. Zum Schluss wird die Variable `m_fkt_end` wieder auf 1 gesetzt, damit der nächste Test gestartet werden kann.

4.5.7 SelectMAP-Port

Beim `SelectMAP_Port-Test` wird bereits in der Funktion `Selftest()` die Komponente auf dem XFBoard-Bild gekennzeichnet. Als weiteres wird die Bitstream-Datei für die Konfiguration des R-FPGA geöffnet.

Sollte dies nicht möglich sein, wird ein `TRACE` ausgegeben und die Funktion `SelectMAP_Port()` kann somit nicht gestartet werden. Nach dem erfolgreichen Öffnen der Datei, wird die Funktion `SelectMAP_Port` ausgeführt.

In `SelectMAP_Port` werden als erstes 1024 Bit aus der zuvor geöffneten Datei gelesen. Die Variable `m_nBytesRead` gibt die genaue Anzahl der gelesenen Bits zurück. Die Variable `m_total_len` wird nun um diesen Wert vergrößert. Sie gibt an, wieviele Bits insgesamt bereits gelesen wurden. Beim Konfigurationsbefehl für den R-FPGA muss die genaue Länge des Programms angegeben werden, welche das Board aus dem SRAM lesen soll.

Solange `m_nBytesRead` einen Wert grösser als 0 besitzt, werden nun die eingelesenen Daten an das Board gesendet. Die Funktion `SelectMAP_Port()` wird nach dem Senden eines einzigen `SELECTMAP_REQUEST`-Paketes bereits beendet und wartet auf ein Acknowledgement des Boards, damit die Funktion ein weiteres Mal aufgerufen werden kann. Mit dem Parameter `m_pbuf[2]` wird angegeben, ob es sich um ein Datenpaket oder einen Konfigurationsbefehl handelt. Wurde der Parameter `m_pbuf[2]` auf 0 gesetzt, so handelt es sich um ein Datenpaket, beträgt er 1, so ist es ein Konfigurationsbefehl.

Auf der Boardseite wird das erhaltene `SELECTMAP_REQUEST`-Paket auf den Parameter `m_pbuf[2]` untersucht und an die Routine `selectmap()` oder `configure_R_FPGA()` weitergeleitet.

Im ersten Fall werden, die im Pointer enthaltenen Daten, ins SRAM kopiert. Wurde das Paket erfolgreich in den Speicher geladen, sendet das Board ein Acknowledge-Paket mit der erhaltenen Paketnummer zurück an den PC.

Erkennt das Programm das Antwort-Paket als einen `SELECTMAP_REPLY`, wird innerhalb des `PacketHandler()` der Parameter `Paketnr` extrahiert und mit der gesendeten Paketnummer verglichen.

Stimmen die beiden Integers überein, wird die Statusanzeige inkrementiert und die Funktion `SelectMAP_Port()` erneut gestartet.

Das soeben beschriebene Prozedere läuft nun solange ab, bis die Variable `m_nBytesRead` den Wert 0 erhält. Trifft dieser Fall ein, wird vom PC der Konfigurationsbefehl gesendet, welcher mit dem Parameter `m_pbuf[2]` gleich 1 angezeigt wird. Als zweiter Parameter wird im Paket zudem `m_total_len` hinzugefügt. Zugleich öffnet sich ein zweites Fenster, welches zur Klasse `CDlgSelectMAPPortTestC` gehört. Das Fenster enthält die 2 Tasten `JA` und `NEIN` und einen erklärenden Text. Empfängt nun das Board den Konfigurationsbefehl, entnimmt dieses dem Paket die totale Länge der Datei und startet die Routine `configure_R_FPGA()`.

Durch den Aufruf der `selmap_Configure()`-Funktion wird die Datei aus dem SRAM gelesen und der R-FPGA über die SelectMAP-Schnittstelle konfiguriert. Wurde in der gesendeten Datei ein Zähler in den LEDs implementiert, kann die richtige Konfiguration visuell überprüft werden. Sobald die LEDs leuchten, kann die `JA`-Taste auf dem zusätzlichen Fenster gedrückt werden. Somit wird in der Klasse `CDlgSelectMAPPortTestC` die Funktion `OnJa()`. Diese wiederum startet `OnSelectMAP()`, die letzte Funktion des SelectMAP-Test. Wurde die `JA`-Taste gedrückt, erhält `OnSelectMAP()` den Wert 1, bei der `Nein`-Taste der Wert 0. Somit kann in `OnSelectMAP()` das entsprechende Resultat an der Oberfläche angezeigt werden. Damit der nächste Test gestartet wird, muss `m_fkt_end` auf 1 gesetzt werden.

4.5.8 Temperatursensor

Der Temperatursensor-Test ist ein Vertreter des ersten Szenarios wie in Abschnitt 4.2 vorgestellt. Entsprechend dem Szenario wird als erstes ein `TEMPERATUR_REQUEST` an das Board gesendet. Bei diesem Request werden keine Parameter übergeben. Das Board springt somit in die `temperatur()`-Routine.

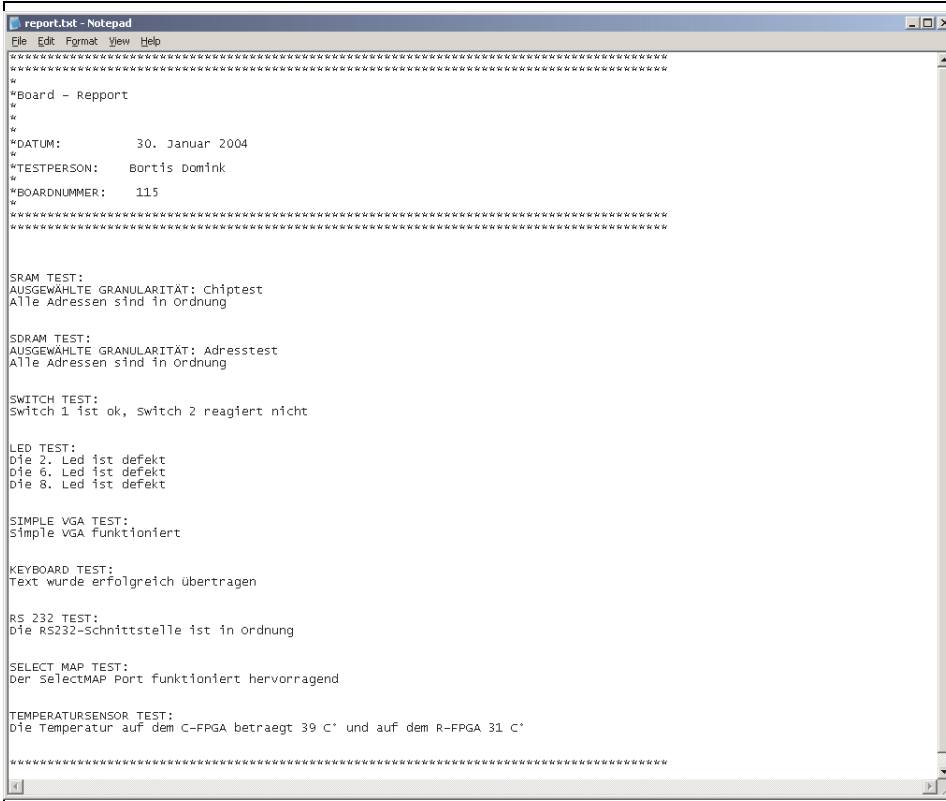
In dieser Funktion werden aus der Baseadresse des Temperatur-GPIOs die beiden Temperaturen für den C-FPGA und den R-FPGA gelesen. Diese befinden sich in derselben Adresse und müssen deshalb noch aus dem gespeicherten Wert extrahiert werden. Die beiden Temperaturen werden an den PC zurückgesendet und auf dem GUI angezeigt werden. Wie gewohnt muss noch `m_fkt_end` auf 1 gesetzt werden.

4.5.9 Ende des Selftests

Damit der Selftest korrekt abgeschlossen werden kann, muss dies dem Benutzer am Schluss angezeigt werden. Deshalb verfügt das Programm noch über eine weitere Routine, die das Ende des Tests signalisiert. Zum Schluss des Selftests wird die letzte Konsole angezeigt. Diese bildet die Klasse `CDlgEndTest`. Im Fenster wird das erfolgreiche Ende des Test signalisiert. Mit Hilfe des Timers wird das Fenster nach 3 Sekunden geschlossen und der Test ist somit beendet.

4.6 Report

Nach Beenden des Selftest wird vom Programm ein Report in einer txt-Datei generiert. Der Report beinhaltet als Header, die eingegebenen Werte, wie Boardnummer, Datum und Testperson. Als weiteres werden alle Resultate, die während dem Test generiert wurden, im Report festgehalten. Der Report wurde aus Gründen der Datensicherung hinzugefügt. Die Ergebnisse können für die verschiedenen Boards gespeichert oder ausgedruckt werden.



```
report.txt - Notepad
File Edit Format View Help
*****
*
*Board - Repport
*
*
*DATUM:          30. Januar 2004
*
*TESTPERSON:    Bortis domink
*
*BOARDNUMMER:   115
*
*****

SRAM TEST:
AUSGEWÄHLTE GRANULARITÄT: Chiptest
Alle Adressen sind in ordnung

SDRAM TEST:
AUSGEWÄHLTE GRANULARITÄT: Adresstest
Alle Adressen sind in ordnung

SWITCH TEST:
Switch 1 ist ok, Switch 2 reagiert nicht

LED TEST:
Die 2. Led ist defekt
Die 6. Led ist defekt
Die 8. Led ist defekt

SIMPLE VGA TEST:
Simple VGA funktioniert

KEYBOARD TEST:
Text wurde erfolgreich übertragen

RS 232 TEST:
Die RS232-Schnittstelle ist in ordnung

SELECT MAP TEST:
Der SelectMAP Port funktioniert hervorragend

TEMPERATURESENSOR TEST:
Die Temperatur auf dem C-FPGA betraegt 39 C' und auf dem R-FPGA 31 C'

*****
|
```

Abbildung 4.3: Report

Kapitel 5

Schlusswort

Die letzten 14 Wochen waren eine anstrengende Zeit. Ich habe mich in dieser Arbeit in ein ganz neues Gebiet gewagt und eingearbeitet. Ich denke die Resultate lassen sich sehen. Doch nicht nur Resultate in Papierformat und CDs sollen erwähnt werden. Ebenfalls meine Erfahrungen, die ich in dieser Zeit machen durfte, waren grossartig. Zum ersten habe ich einige Tools kennengelernt, wie zum Beispiel EDK (Programmieren des Boards) sowie Visual C++, mit dem ich die ganze Applikation auf dem PC realisieren konnte. Zudem konnte ich meine Kenntnisse in bezug auf Networking (TCP/IP) vertiefen. Hiermit verstehe ich den Aufbau eines Paketformates und Protokolls, das die beiden Teilsysteme koordiniert. Neben Networking auf TCP/IP durfte ich auch zwischenmenschliche Netzwerke aufbauen. Am XFBoard-Projekt waren einige weitere Gruppen beteiligt. Die Arbeiten waren voneinander abhängig und somit war Teamworking ein wichtiges Element in meiner Arbeit. Zum Schluss möchte ich mich beim Prof. Dr. Thiele für die Ermöglichung meiner Arbeit danken. Weiter möchte ich mich bei meinem Betreuer Herbert Walder für seine kompetente, hilfsbereite und intensive Unterstützung bedanken. Nicht zuletzt danke ich dem Institut für Technische Informatik, welches mir eine hervorragende Infrastruktur zur Verfügung gestellt hat. Merci.

Anhang A

CD

Dieses Kapitel gibt einen kurzen Überblick der Verzeichnisstruktur der zugefügten CD.

- **Aufgabenstellung** Eine pdf-Datei, in der das Ziel der Arbeit formuliert wird
- **AudioVideo_Testbench** Beinhaltet eine Testbench, welche die Video- und Audio-Driver des XFBoards testet.
- **XFBoard_Selftest** Beinhaltet die beiden Verzeichnisse `Board_Code` und `PC_Code`. Im Verzeichnis `Board_Code` sind alle Dateien zur Konfiguration des XFBoards vorhanden. Das Verzeichnis `PC_Code` enthält alle Dateien für die Applikation auf dem PC
- **Selftest_Präsentation** Beinhaltet die Schlusspräsentation dieser Semesterarbeit.
- **Selftest_Bericht** Enthält den Bericht der Semesterarbeit.