Workshop VHDL B143a
Fachhochschule Wedel

Guillaume Fournier-Mayer (Tinf 101922)
Jan Ottmüller (Tinf101737)

19. September 2018

Inhaltsverzeichnis

Kapitel 1

Einleitung

Diese Dokumentation dokumentiert und beschreibt den Workshop VHDL (B143a). Die Aufgabe dieses Semesters mit dem Namen „Crypto Notepad“ erfordert das Auslesen einer PS2 Tastatur, die Ver- und Entschlüsselung sowie Speicherung der Tastaturausgaben und das Ausgeben des Speichers auf einem Zwei-Zeilen-Textdisplay. Die Beschreibung soll auf dem Altera DE2 board mit Cyclone 2 FPGA laufen.

Kapitel 2

Aufgabenanalyse

Für die Implementierung haben wir uns folgende Ziele gesetzt:

1. Gute Modularisierung zur Wiederverwendbarkeit der einzelnen Module
2. Klare Namensgebung der Signale und Prozesse
3. Die Top-Level-Entity soll im RTL-Viewer übersichtlich und strukturiert sein.

Ausgehend von der Top-Level-Entity (TLE) sollen die einzelnen Komponenten eingebunden und verbunden werden.
Die Top-Level-Entity CryptoNotepad enthält somit Schnittstellen zum Board und führt die einzelnen Module zusammen.
Hier die Grobe Strukturierung der Entities auf TLE:

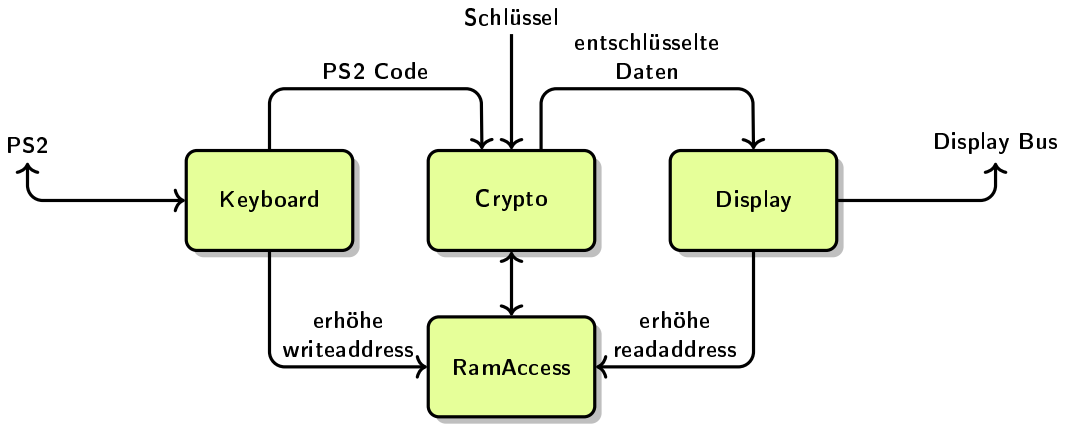

Abbildung 2.1: Vereinfachtes Blockschaltbild TLE

Die Entity RamAccess dient als Schnittstelle zum, mit dem MagicWizzard automatisch generiertem, Ram.

Die Ver- und Entschlüsselung der Zeichen soll in einer separaten Entity stattfinden um das Problem besser zu modularisieren und die Keyboard Entity möglichst wiederverwendbar zu machen. Diese gibt über ihren Ausgang die empfangenen Scancodes aus. Ein andere Möglichkeit wäre, die PS2 zu ASCII Konvertierung bereits im Keyboard Modul durchzuführen. Jedoch ist diese Umwandlung auch eine Art Verschlüsselung und findet deshalb in der Crypto Entity Platz.
An vielen Orten sollen Zustandsautomaten mit zwei- oder drei-Prozess-Beschreibung verwendet werden. Spezielle Zustandstypen sollen die Übersichtlichkeit fördern.
Die Keyboard Entity wird in Transmitter und Receiver aufgeteilt.
Das Display ist deutlich langsamer als der FPGA. Um dies zu berücksichtigen, können zwei Strategien verfolgt werden. Entweder kann das Busy Signal vom Display ausgelesen werden. Wenn das Display bereit ist, wird dieses ausgeschaltet und es können neue Daten gesendet werden. Ein anderer Ansatz macht sich zu Nutzen, dass wenn man lange genug wartet, das Display immer bereit ist. So wird das Display nach 2,5 ms (400 Hz) immer fertig mit der Verarbeitung des letzten Befehls sein. In der Initialisierungsphase kann das Busy Signal jedoch nicht ausgelesen werden, weshalb der zweite Ansatz gewählt wird.

2.1 Zusatzaufgabe

Um kleinere Fehler in der Implementierung auszugleichen, soll eine Zusatzfunktion implementiert werden, welche die Aufgabe erweitert.
Eine Erweiterung, welche implementiert wird, ist das Setzen der Status-LEDs. Hierfür muss der PS2-Transmitter funktionieren, da Kommandos an die Tastatur verschickt werden müssen. Diese Funktion soll im Keyboard Modul Platz finden.

Kapitel 3

Benutzerhandbuch

3.1 Reset

Die Schaltung kann über den Schalter SW17 zurückgesetzt werden. Hierbei werden alle Zustände der Schaltung zurückgesetzt. Als letztes muss der PS2 Stecker neu eingesteckt werden, damit auch die Status LEDs an der Tastatur zurückgesetzt werden.
Ein Zurücksetzen des Speichers, und somit des Displayinhaltes geschieht nicht. Jedoch wird der Cursor wieder auf das Erste Feld im Display gesetzt, wenn der PS2 Stecker neu eingesteckt wird.

3.2 Anwendung

Nachdem das Programm auf das Altera DE2 Board geladen wurde, wird die Tastatur angeschlossen. Danach können Zeichen über die angeschlossene Tastatur eingegeben werden. Sie werden auf das Display geschrieben. Zulässig sind die Tasten a-z sowie 0-9 und 'LEERZEICHEN'. Bei nicht bekannten Zeichen wird ein „?“ ausgegeben. Beim Drücken auf die Tasten CAPS-LOCK, SCROLL-LOCK, und NUM-LOCK gehen die entsprechenden Status LEDs auf der Tastatur an und bei erneutem Drücken wieder aus. Einen Einfluss auf die geschriebenen Zeichen gibt es nicht.
Über die Schalter SW0 bis SW7 kann der zu verwendende Schlüssel eingegeben werden, mit dem die Daten vor dem Speichern verschlüsselt, und beim Lesen entschlüsselt werden. Die Verschlüsselung ist eine einfache XOR Verknüpfung. Der Schlüssel kann nach belieben geändert werden. Die Zeichen auf dem Display werden entsprechend des eingestellten Schlüssels entschlüsselt und auf dem Display angezeigt.

Kapitel 4

Implementierung

In diesem Kapitel werden die Entities Bottom-Up beschrieben.

4.1 DoubleEdgeDetector

Diese Entity erledigt die Einsynchronisierung der PS2 Takt und Datenleitung. Hierfür werden hintereinander geschaltete Flip Flops verwendet, sodass Metastabilität im System vermieden wird, wenn das Signal zu einem ungünstigen Zeitpunkt empfangen wird. Als weitere Ausgänge verfügt der Edge Detector über zwei Signalausgänge. Das Signal fedge_o wird bei Fallender Flanke des Eingangs für einen Takt auf 1 gesetzt. Das Signal redge_o bei steigender Flanke.

4.2 Transmitter

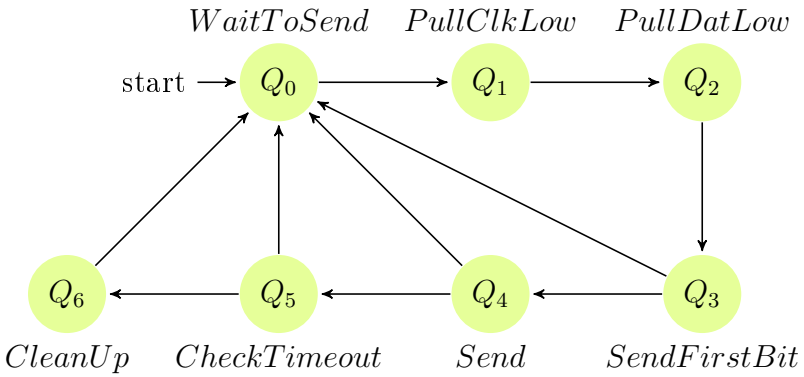

Abbildung 4.1: Zustandsautomat Transmitter

Beim Senden werden die PS2-Clockimpulse von der Tastatur erzeugt. Zum Senden muss der Host die Leitungen in den Request-To-Send Zustand bringen, indem die Clock Leitung für 100μs auf Low zieht und danach die Datenleitung auch auf Low zieht und beide wieder freigibt.
Die Tastatur generiert nun Clockimpulse bei denen die Daten vom Host gesendet werden. Das ändern der Daten geschieht bei Low Pegel. Das Auslesen von der Tastatur geschieht bei High Pegel. Hierbei können zwei Timeouts auftreten. Entweder Die Tastatur fängt erst später als 15ms an, den Takt zu generieren, oder die Übertragung des Datenpaketes dauert mehr als 2ms. Diese Zeitmessung wird bei der Implementierung in der Entity berücksichtigt.
Der Kern dieser Entity ist ein Zustandsautomat in Zwei-Prozess-Beschreibung.

4.3 Receiver

Im Gegensatz zum Senden werden beim Empfangen die Daten bei Fallender PS2 Clock Flanke vom Host gelesen. Bei diesem Ereignis wird der anliegende Pegel des PS2 Datensignals in das Schieberegister *regs* geschrieben, welches danach weitergeschoben wird. Sind alle 11 Bits eines PS2 Paketes empfangen, so wird geprüft, ob die logische Verknüpfung mit dem Ergebnissignal *error* einen Fehler signalisiert. Entsprechend der Auswertung werden die Daten dann an den Datenausgang der Entity geschrieben und das Empfangen eines neuen Paketes signalisiert.

4.4 Transceiver

Der Transceiver implementiert das Senden und Empfangen von Daten über die PS2 Schnittstelle. Hierfür werden die Komponenten Transmitter und Receiver eingebunden. Die PS2 Signale werden mit Hilfe von DoubleEdgeDetectoren ein-synchronisiert. Da die ein-synchronisierten Signale, sowie die Flankendetektoren, wie in Abb. 4.2 mit gestrichelten Linien zu sehen, sowohl im Receiver als auch im Transmitter benötigt werden, sind Die Flankendetektoren auf dieser Ebene implementiert.

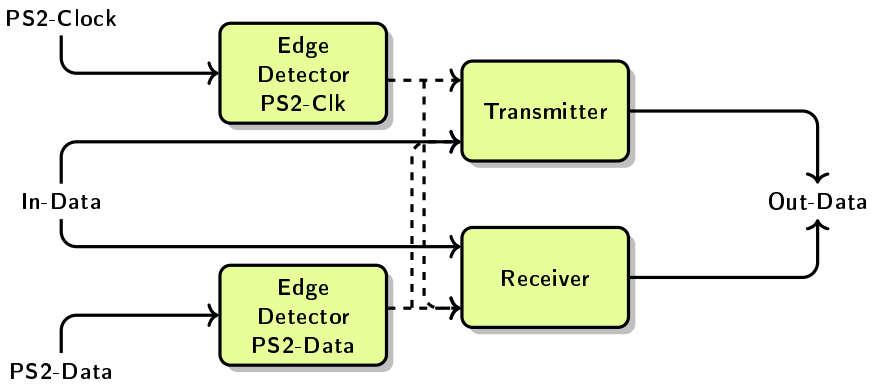

Abbildung 4.2: Blockschaltbild Transceiver

4.5 Keyboard

4.6 Crypto

Die Entity Crypto dient zum Ver- und Entschlüsslen der Daten. Sie Enthält keine internen Signale oder Speicherzustände und bedarf deswegen keinem Takt und Reset Signal. Die Umsetzung der Make codes in ASCII Zeichen findet ebenfalls hier statt. Dies mag auf den ersten Blick keine Verschlüsselung sein, doch genauer Betrachtung ist diese Umwandlung auch nur eine Art der Verschlüsselung. Anschließend wird das ASCII Zeichen mit dem übergebenen Schlüssel XOR verknüpft und zurückgegeben. Bei der Entschlüsselung findet wieder eine XOR Verknüpfung mit dem anliegenden Schlüssel statt.

4.7 RamAccess

Die Entity RamAccess dient als Schnittstelle zum Ram. Es werden hilfreiche Schnittstellen zur Traversierung des Speichers bereitgestellt.
Da sowohl das Schreiben, als auch das Lesen des Speichers in einer definierten Reihenfolge stattfindet, kümmert sich diese Entity um diese Aufgabe. Bei Steigender Flanke an den Signalen
ramaccess_increase_read_address und *ramaccess_increase_write_address* werden die entsprechenden Adressen erhöht. Zur Detektierung der Flanken wird der bereits implementierte DoubleEdgeDetector benutzt, da die Signale für mehrere Taktzyklen an sind, aber die Adresse nur einmal inkrementiert werden soll. Die leichte Verzögerung des Signals durch den Kantenerkennung ist nicht störend. Beim Inkrementieren der Schreibadresse wird mit Einschalten des Signals ram_write_enable der Inhalt des Dateneingangs in den Speicher geschrieben.

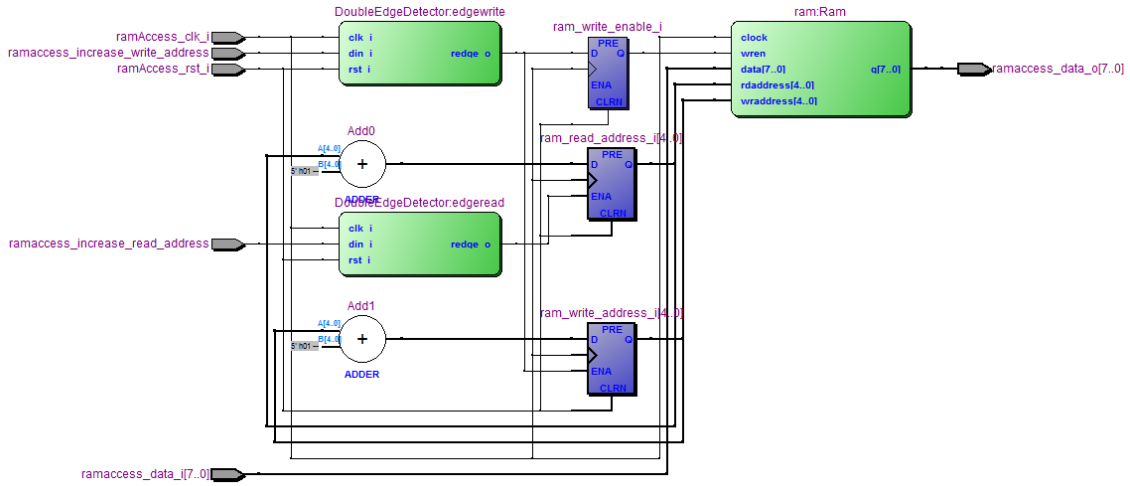

Abbildung 4.3: RamAccess Modul RTL Ansicht

4.8 Ram

Das Ram Modul ist ein, mit dem Magic Wizzard automatisch generierter, Speicherbaustein. Dieser besitzt mit 8Bit x 32Bit genauso viele Bytes wie das Display Zeichen hat. So kann man eine direkte Abbildung des Speichers auf dem Display realisieren. Leider verfügt das Ram Modul nicht über einen Asynchronen Reset, welcher den ganzen Speicher leert. Dies wäre praktisch, um den Display Inhalt bei einem Reset zu leeren.

4.9 Display

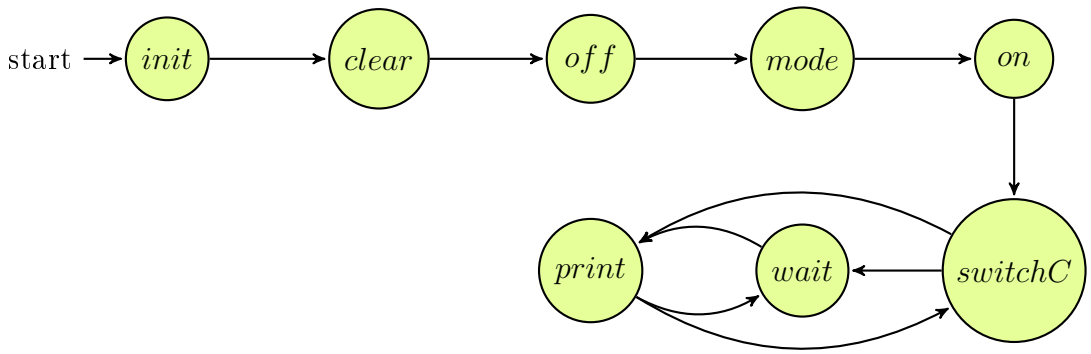

Abbildung 4.4: Vereinfachter Zustandsautomat ohne Wartezustände

Der Kern der Display Entity ist ein Zustandsautomat der, wie in der Aufgabenanalyse beschrieben, mit einem Takt von 400 Hz getaktet ist.

4.9.1 Prozesse

Der Zustandsautomat wird mit einer drei-Prozess-Beschreibung implementiert. So entstehen die drei Prozesse *store*, *transition* und *output*. Der vierte Prozess *displayClock* dient der Generierung des Display Taktes.

4.9.2 Displaytakt

Dieser Prozess generiert langsame Impulse mit 400 Hz, indem die Taktzyklen von lcd_clk_i gezählt werden.
Bei 62500 gezählten Taktflanken wird das Signal clk_displayEnable für einen Takt auf 1 gesetzt, was bei 50 MHz Basistakt zu 400 Hz führt.

4.9.3 Initialisierung

In der Initialisierungsphase wird zunächst für 10ms das Function Set Signal an den parallelen Bus angelegt. das Signal lcd_rs steht auf 0, da die Daten auf das Controll Register und nicht auf das Datenregister geschrieben werden sollen. Im Function Set wird der Display Controller auf einen 8-Bit Bus mit einem Zweizeilen Display initialisiert.
Danach wird das Display ausgeschaltet und der Inhalt gelöscht. Jetzt folgen die entsprechenden Befehle, um den Cursor automatisch weiter zu setzen und den Cursor anzuzeigen. Abschließend wird das Display wieder eingeschaltet.

4.9.4 Zeichen schreiben

Nach der Initialisierungsphase können Zeichen als ASCII Code an das Display gesendet werden. Hier wechseln sich nun immer die Zustände printChars und waitState ab, es sei denn, der Cursor befindet sich am Ende des Displays und muss die Displayzeile wechseln. im Zustand PrintChars wird über lcd_rs das Datenregister zum schreiben ausgewählt und die Daten vom Datenbus lcd_ram_data_i werden in den Speicher des Displays geschrieben. Im waitState wird dann über die ausgehende Schnittstelle das Ram_Access Modul dazu angewiesen, die Leseadresse des Speichers zu erhöhen.

4.10 CryptoNotepad

Diese Entity ist die Top-Level-Entity. Von hier aus werden die einzelnen Module verknüpft und die Pins an den FPGA verschaltet.

Node Name	Direction	Location
clk_i	Input	PIN_N2
crypto_key[7]	Input	PIN_C13
crypto_key[6]	Input	PIN_AC13
crypto_key[5]	Input	PIN_AD13
crypto_key[4]	Input	PIN_AF14
crypto_key[3]	Input	PIN_AE14
crypto_key[2]	Input	PIN_P25
crypto_key[1]	Input	PIN_N26
crypto_key[0]	Input	PIN_N25
display_data_bits[7]	Output	PIN_H3
display_data_bits[6]	Output	PIN_H4
display_data_bits[5]	Output	PIN_J3
display_data_bits[4]	Output	PIN_J4
display_data_bits[3]	Output	PIN_H2
display_data_bits[2]	Output	PIN_H1
display_data_bits[1]	Output	PIN_J2
display_data_bits[0]	Output	PIN_J1
lcd_blon	Output	PIN_K2
lcd_en	Output	PIN_K3
lcd_on	Output	PIN_L4
lcd_rs	Output	PIN_K1
lcd_rw	Output	PIN_K4
ps2_clk_io	Bidir	PIN_D26
ps2_dat_io	Bidir	PIN_C24
rst_i	Input	PIN_V2

Abbildung 4.5: Pin Zuweisung im Pin Planner

Kapitel 5

Test

Für das Testen der Funktionalität der einzelnen Entities werden Testbenches geschrieben, welche in Modelsim simuliert werden. Zunächst werden die grundlegenden Entities getestet. Sobald diese als funktionstüchtig festgestellt sind, werden Testbenches für die Entities geschrieben, welche die grundlegenden Funktionen zusammenführen. So kann an dieser Stelle die Zusammenarbeit der einzelnen Entities getestet werden.

5.1 DoubleEdgeDetector_tb

Aufgrund der recht einfachen Funktionalität der Entity ist auch die Testbench recht klein. An den Eingang wird ein sich änderndes Signal angelegt. Mit Asserts werden dann die Zustände der eingetakteten Signale überprüft.

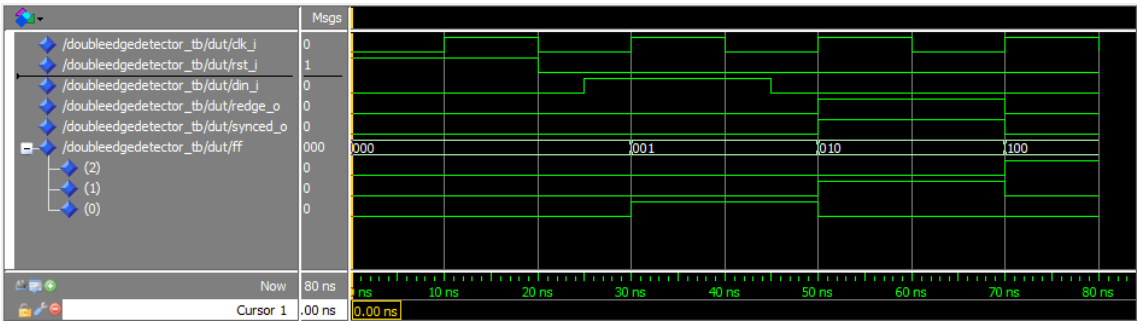

Abbildung 5.1: Signalverlauf DoubleEdgeDetector Testbench

5.2 Transceiver_tb_receiver

Zum Testen des PS2 Empfängers werden zwei Pakete Simuliert. Das erste ist ein aalides Paket. Mit Asserts wird geprüft, ob die entsprechenden Signale richtig gesetzt

sind. Auf das erste Paket folgend wird ein Paket mit fehlerhafter Parität gesendet. Daraufhin wird geprüft, ob das Fehlersignal an geht und nicht fälschlicher Weise signalisiert wird, dass neue Daten empfangen wurden. Der System und PS2 Takt werden in separaten Prozessen erzeugt.

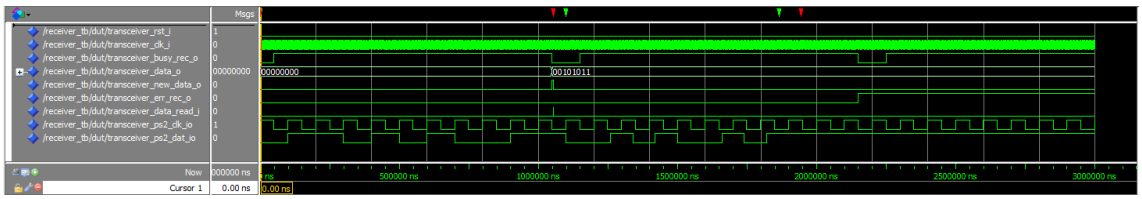

Abbildung 5.2: Signalverlauf DoubleEdgeDetector Testbench

5.3 Transceiver_tb_transmitter

Zum Testen des Transmitters wird das Versenden von zwei Paketen simuliert. Da beim Senden der Takt von der Tastatur erzeugt wird, muss der Takt in der Testbench simuliert werden.
Während des Sendevorgangs werden die PS2-Daten mitgeschnitten und am Ende des Tests mit den gesendeten Daten verglichen.
Danach wird ein Fehlerfall simuliert, indem der PS2 Takt als deutlich zu langsam simuliert wird. Hier wird nach 2ms ein Timeout ausgelöst.

Nachdem das Simulieren in der Testbench funktionierte, wurde der Transceiver auf der Hardware getestet. Leider schlug dies fehl und die Tastatur begann zu Blinken, da ein Fehler bei der Übertragung auftrat. Deswegen wurden die PS2 Daten und Clock Leitungen auf zwei LED Ausgänge des Boards Herausgeführt, um sie mit dem Oszilloskop abzugreifen. Die Signalverläufe wurden dann mit der Testbench abgeglichen und entsprechend angepasst. Der Fehler war, dass die Tastatur direkt nach dem Versenden eines Paketes noch nicht bereit zum Empfangen eines neuen Paketes war. Als Lösung wird nun vor dem Versenden der Daten 400 μs gewartet

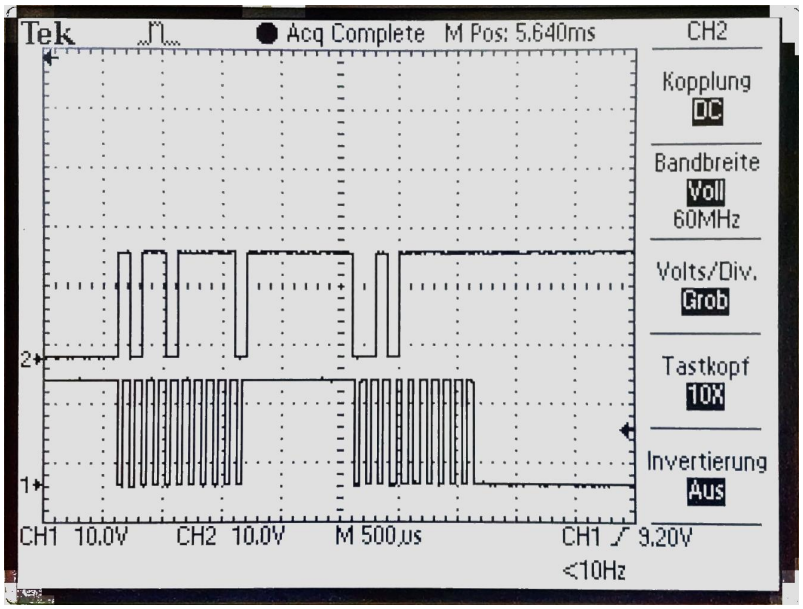

Abbildung 5.3: NumLock paket mit gesendetem “ 0xED“ paket zum Einstellen der LEDs

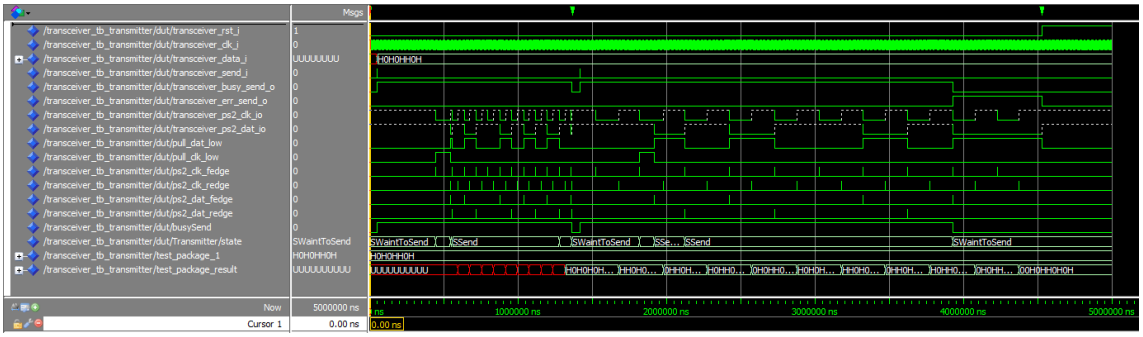

Abbildung 5.4: Signalverlauf Transmitter Testbench

5.4 Keyboard_tb

Da die Funktionalität das Transceivers schon getestet wurde, wird in der Keyboard Testbench das Einschalten der NumLock Taste simuliert.
Zunächst wird der Tastendruck der NumLock Taste simuliert. Danach wird das Taktsignal zum Senden des 0xED Paketes erzeugt, da dies sonst die Tastatur übernimmt. Nachdem das simulierte ACK Paket empfangen wurde wird wieder der Takt für das Paket gesendet, mit dem Die LED an der Tastatur eingeschaltet werden.

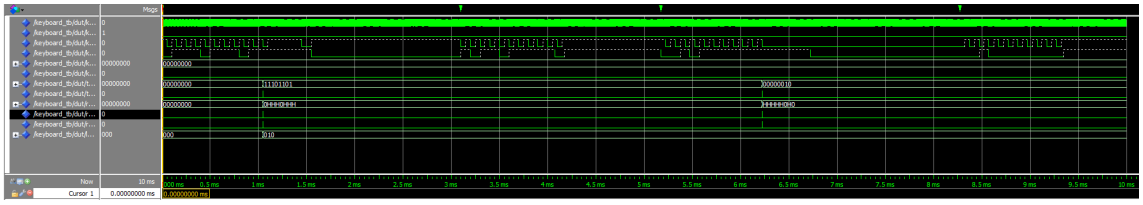

Abbildung 5.5: Signalverlauf Keyboard Testbench

5.5 RamAccess_tb

Zum Testen des RamAccess Moduls werden drei Bytes in den Speicher geschrieben. Danach werden sie sequenziell wieder ausgelesen und es wird geprüft, ob die Daten korrekt sind.

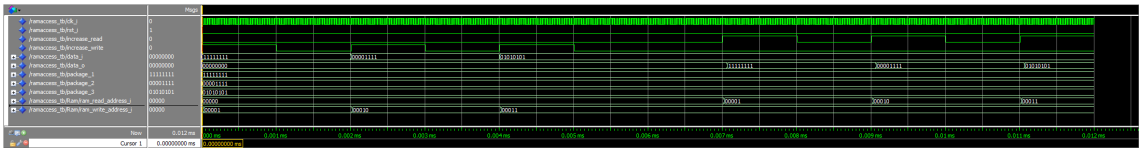

Abbildung 5.6: Signalverlauf RamAccess Testbench

5.6 CryptoNotepad_tb

In der TLE Testbench wird das Zusammenspiel aller Module getestet. Zuerst muss hierfür 20ms auf die Initialisierung des Displays gewartet werden. Danach werden zwei Tastenanschläge simuliert. Jeweils mit korrektem Break Paket und anschließender Wiederholung des Make Codes. Nach Empfangen des Tastendrucks werden die Daten wie erwartet in das Ram geschrieben und anschließend vom Display ausgelesen.

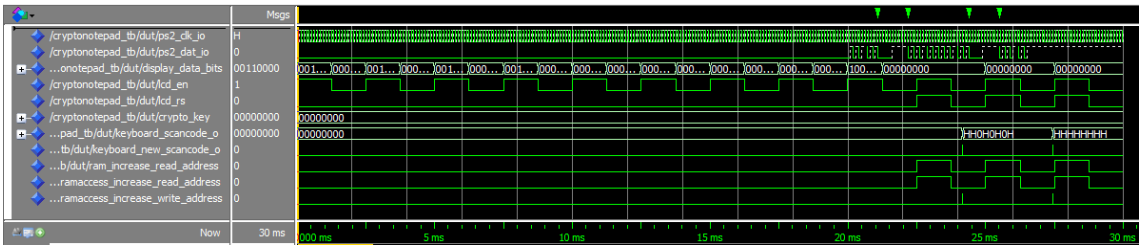

Abbildung 5.7: Signalverlauf TLE Testbench

Kapitel 6

Fazit

Leider sind die verwendeten Werkzeuge nicht zeitgemäß. Sowohl Quartus II als auch Modelsim werden in veralteten Versionen benutzt, welche den Arbeitsfluss extrem verlangsamen. Ohne VHDL Erfahrung war es zu Anfang recht schwer zu entscheiden, auf welcher Ebene modularisiert werden soll. Nach bestandener Klausur fiel es leichter.