Technische Universität München
Fakultät für Informatik
ETI-Großpraktikum
SS 2013

Entwicklung eines Logic Analyzers in VHDL

ETI-GP 03

Themensteller:

Georg Acher
ETI-Großpraktikum
Fakultät für Informatik
Technische Universität München
Boltzmannstraße 3, 85748 Garching

Bearbeitet von:

Sven Hertle, Markus Engel, Thomas Czok

Abgabetermin: 23. August 2013

Inhaltsverzeichnis

1. Was ist ein Logikanalyzer?

Mit einem Logikanalyzer kann der zeitliche Verlauf von digitalen Signalen aufgenommen werden.
Ein Logikanalyzer stellt also nur Low- und High-Pegel dar, wobei manche Geräte zusätzlich auch anzeigen können, wenn der aufgenommene Wert undefiniert ist. Im Gegensatz zu einem Oszilloskop hat ein Logikanalyzer eine große Zahl an Kanälen, teilweise sind Geräte mit mehreren hundert Eingängen verfügbar. Logikanalyzer werden zum Debuggen von digitalen Schaltungen benötigt. Für diesen Zweck müssen sie mit einem großen Speicher ausgestattet sein oder sehr gute Möglichkeiten zum Triggern besitzen, da man kaum periodische Vorgänge untersuchen will.
Inzwischen werden Logikanalyzer oft durch Simulationen ersetzt. Diese können einen Logikanalyzer aber nicht komplett ersetzen. Auch gegenüber den Diagnoseschnittstellen, die oft in integrierten Schaltkreisen vorhanden sind, hat ein Logikanalyzer Vorteile: Dieser ist oft schneller und genauer.

2. Verwendete Hardware

Der Logikanalyzer wurde mit dem FPGA-Board GOP_XC3S200 realisiert. Dieses ist ausgestattet mit einem Xilinx Spartan 3 XC3S200, der mit 49.152 MHz getaktet wird. Außerdem befinden sich auf dem Board zwei Taster und acht ansteuerbare LEDs. Diese wurden aber nicht verwendet, da noch weitere sieben Taster an das Board angeschlossen sind. Das FPGA-Board ist mit zwei Stiftleisten direkt in einem Steckbrett angebracht. Darüber sind die weiteren Ein- und Ausgänge sowie die Stromversorgung angeschlossen.
Die aufgezeichneten Daten können über VGA an einem Bildschirm angeschaut werden. Für den VGA-Ausgang werden die Signale HSYNC, VSYNC sowie analoge Werte für die Farben rot, grün und blau benötigt. Für diese analogen Werte werden jeweils zwei digitale Ausgänge über passende Widerstände zusammengeführt. Dadurch können 64 verschiedene Farben dargestellt werden.
Der Dateneingang ist 8 Bit breit, es können also acht Kanäle aufgezeichnet werden. Dazu befindet sich auf dem Steckbrett ein Zählbaustein, der durch einen Quarzoszillator mit 25 MHz Taktfrequenz hochgezählt wird. Dieser dient dem Logikanalyzer als beispielhafte Signalquelle.

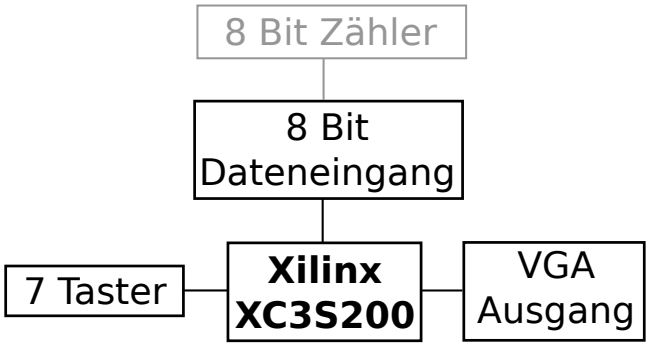

Abb. 1: Schematische Darstellung der Hardware

3. Spezifikation der Funktionen

Unser Logikanalyzer kann acht Kanäle aufzeichnen. Die Steuerung geschieht über die sieben Taster. Die VGA-Ausgabe erzeugt ein 640x480 Pixel Bild bei 60 Hz.
Es gibt grundsätzlich zwei Aufzeichnungsarten:

Continuous Der Speicher wird als Ring verwendet, d. h. die Messdaten werden ständig überschrieben. Sobald der Speicher voll ist, wird vorne wieder angefangen zu schreiben.

One Shot Hier wird der Speicher einmal vollgeschrieben, danach stoppt die Aufzeichnung. Es passen insgesamt 24.576 Messwerte in den Speicher. Dieser Modus ist für die Analyse der Signale meist sinnvoller.

Dazu können verschiedene Samplingraten gewählt werden, d. h. der Abstand zwischen zwei Aufnahmen. Dabei sind folgende Werte möglich: Maximum (20 ns), 1 ms, 10 ms, 100 ms, 1s.
Nach der Aufnahme können die Daten genauer betrachtet werden. Dazu ist Scrollen und Zoomen möglich. Dabei wird selbstverständlich eine Zeitskala und der zeitliche Abstand (Timebase) zwischen zwei Strichen angezeigt. Zum Überfliegen der Messdaten steht schnelles Scrollen zur Verfügung, zum Ausrichten an den Zeitmarkierungen kann dann auch langsam gescrollt werden. Beim Zoomen entspricht 100% einem Messwert pro Pixel. Es sind die Zoomstufen 25%, 50%, 100%, 200% und 400% möglich. Die Timebase wird für jede Zoomstufe und Samplingrate korrekt angezeigt.
Der Logikanalyzer ist auch mit einfachen Triggern ausgestattet. Pro Kanal kann als Bedingung gewählt werden, dass eine logische 1, 0, eine steigende oder fallende Flanke anliegen soll. Wenn alle Bedingungen wahr sind, startet die Aufzeichnung.

4. Bedienungsanleitung für Anwender

Der Logikanalyzer lässt sich über den vorhandenen VGA-Anschluss an einen handelsüblichen Monitor anschließen.

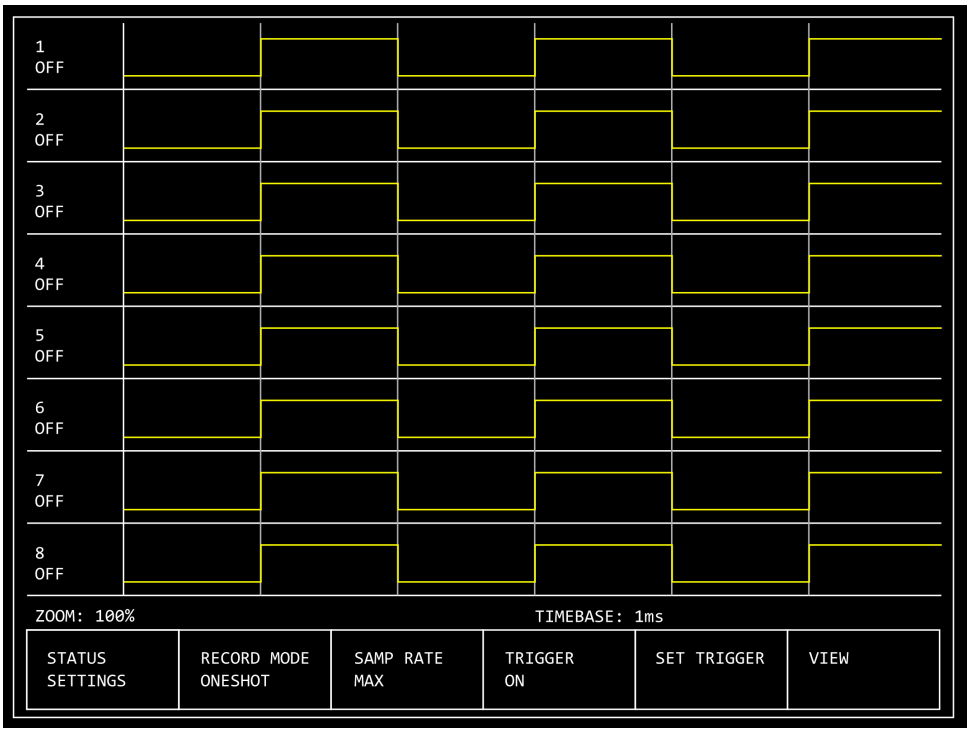

Abb. 2: Bildschirmansicht des Logikanalyzers

4.1. Beschreibung des Benutzerinterfaces

Im oberen Teil der Anzeige befindet sich die Darstellung der einzelnen Kanäle, links stehen die Kanalnummer und eventuell eingestellte Trigger. Unter der Anzeige des aktuell gewählten Zoomfaktors und der Timebase (der vergangenen Zeit zwischen zwei aufeinanderfolgenden senkrechten Strichen) liegt die Menüzeile.
Nach dem Einschalten befindet sich der Logikanalyzer zunächst im Zustand „Settings“, in dem die Einstellungen am Aufnahmemodus, der Samplingrate und den Triggern vorgenommen werden können.

4.2. Steuerung des Menüs

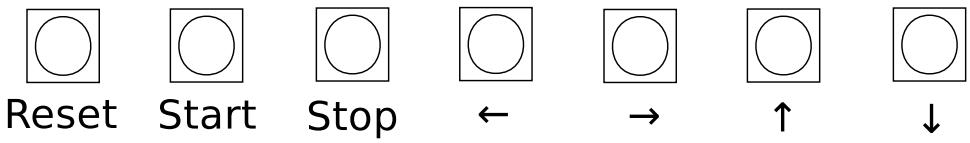

Abb. 3: Belegung der Taster

Die Bedienung erfolgt über die acht zur Verfügung stehenden Taster (vgl. Abb. 4.2.). Der momentan aktive Menüpunkt wird durch die Pfeiltasten nach links und rechts gewählt und durch einen roten Rahmen markiert. Mit den Pfeiltasten hoch und runter kann die Einstellung des aktiven Menüpunkts verändert werden.

4.3. Setzen von Triggern

Werden Trigger im Menü aktiviert, besteht die Möglichkeit, über den Menüpunkt „Set Trigger“ in das Trigger-Untermenü zu wechseln. Dieses befindet sich am linken Rand jeweils unterhalb der Kanalnummern. Der momentan aktive Kanal wird durch einen Pfeil gekennzeichnet und kann mit den Pfeiltasten gewechselt werden. Durch die Pfeiltasten nach links und rechts kann der Trigger eingestellt werden.
Wird nach dem Setzen der Einstellungen das Menü verlassen und die Aufnahme gestartet, so beginnt diese tatsächlich erst, wenn das momentan anliegende Signal alle Bedingungen erfüllt, die durch die Trigger vorgegeben wurden.

4.4. Ansicht der Messwerte

Nachdem die Aufnahme von selbst gestoppt hat (Oneshot) oder vom Benutzer durch den Stop-Button unterbrochen wurde (Continuous), besteht die Möglichkeit im „View“-Modus mit dem Pfeiltasten nach links und rechts zu scrollen. Für schnelles Scrollen muss zusätzlich die Stop-Taste gedrückt werden. Um den Zoomfaktor zu verändern, können die Pfeiltasten hoch und runter verwendet werden.
Zum Wechseln in die Einstellungen kann die Reset-Taste gedrückt werden. Um dann wieder in den „View“-Modus zu kommen, gibt es im Menu die Option „View“, die durch die Stop-Taste bestätigt wird.

5. Dokumentation für Entwickler

Der Logikanalyzer wurde mit VHDL implementiert. Dazu wurde die Software Xilinx ISE WebPACK verwendet.

5.1. Komponenten

Der Logikanalyzer ist in verschiedene Komponenten unterteilt. Diese werden in Abb. 5.1. dargestellt.

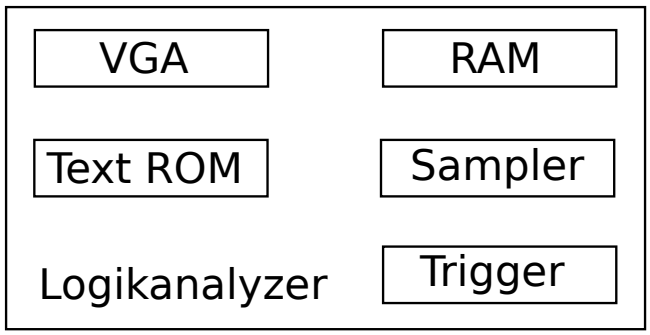

Abb. 4: Unterteilung des Logikanalyzers in Komponenten

5.1.1. Logikanalyzer

Die Komponente Logikanalyzer übernimmt die Steuerung aller anderen Komponenten. Sie beinhaltet alle Signale, die den Status der Logikanalyzers speichern. Die zur Steuerung verwendeten Taster werden hier ausgewertet. Die Signale werden dann entsprechend verändert, sodass die vom Benutzer gewünschte Aktion ausgeführt wird.

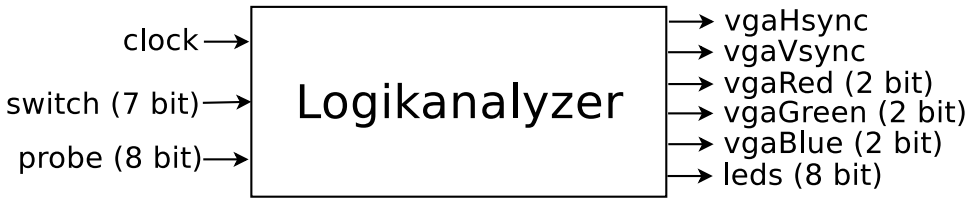

Abb. 5: Ein- und Ausgänge der Komponente Logikanalyzer

Die Komponente Logikanalyzer hat diese Ein- und Ausgänge (vgl. Abb. 5.1.1.):

clock Eingang für den Takt des Bausteins (`std_logic`)

switch 7 Bit breiter Eingang der externen Taster (`std_logic_vector(1 to 7)`)

probe 8 Bit breiter Eingang für die Messwerte (`std_logic_vector(7 downto 0)`)

vgaHsync HSync-Ausgang für VGA-Signal (`std_logic`)

vgaVsync VSync-Ausgang für VGA-Signal (`std_logic`)

vgaRed Ausgang für rote Farbkomponente des VGA-Signals (`std_logic_vector(1 downto 0)`, vgl. Kapitel 2.)

vgaGreen Ausgang für grüne Farbkomponente des VGA-Signals (`std_logic_vector(1 downto 0)`)

vgaBlue Ausgang für blaue Farbkomponente des VGA-Signals (`std_logic_vector(1 downto 0)`)

leds 8 Bit breiter Ausgang für LEDs (wird nur auf 0 gesetzt, `std_logic_vector(7 downto 0)`)

5.1.2. RAM

Zum Speichern der Messwerte wird ein Dualport Block RAM verwendet. Dieser hat den Vorteil, dass gleichzeitig Daten gelesen und geschrieben werden können. Dies ist nötig, da gleichzeitig Messwerte aufgenommen und Daten auf dem Bildschirm angezeigt werden müssen. Der RAM wurde mit dem Core Generator aus Xilinx ISE generiert. Für Änderungen ist daher mindestens Xilinx ISE Version 14.6 nötig.

5.1.3. Sampler

Der Sampler hat die Aufgabe, die Messwerte im RAM zu speichern. Dabei muss die gewählte Samplingrate sowie der Aufnahmemodus beachtet werden. Außerdem müssen die aktuell an den Eingängen anliegenden Messwerte an den Sampler weitergegeben werden. Dieser braucht die aktuellen Daten, um zu entscheiden, ob die Aufnahme gestartet werden soll.

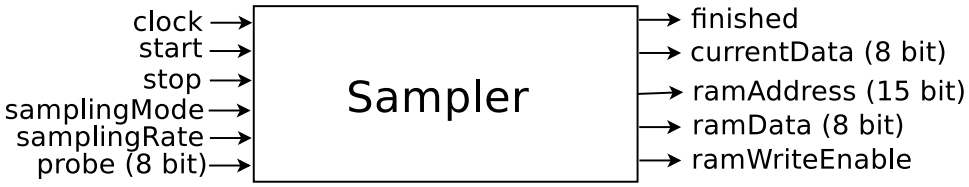

Abb. 6: Ein- und Ausgänge der Komponente Sampler

Die Komponente Sampler hat diese Ein- und Ausgänge (vgl. Abb. 5.1.3.):

clock Eingang für den Takt des Bausteins (`std_logic`)

start Eingang: Aufnahme wird gestartet, wenn eine 1 anliegt. Diese darf nur einen Takt lang anliegen, sonst beginnt die Aufnahme wieder von vorne (`std_logic`)

stop Eingang zum Stoppen der Aufnahme. Dies ist im Oneshot-Modus nicht nötig, da die Aufnahme automatisch stoppt (`std_logic`)

finished Ausgang: 1, wenn der Sampler mit der Aufzeichnung fertig ist. V. a. für Continuous Modus wichtig (`std_logic`)

samplingMode Eingang für zu verwendenden Sampling Modus (`SamplingMode`, vgl. 5.2.6.)

samplingRate Eingang für zu verwendende Samplingrate (`SamplingRate`, vgl. 5.2.5.)

probe 8 Bit breiter Eingang für die Messwerte (`std_logic_vector(7 downto 0)`)

currentData Ausgang für aktuelle Messwerte am Eingang unter Beachtung der Sampling Rate und des Modus (`std_logic_vector(7 downto 0)`)

ramAddress Ausgang der Adresse für schreibenden RAM-Zugriff (`std_logic_vector(14 downto 0)`)

ramData Ausgang für Daten, die in den RAM geschrieben werden sollen (`std_logic_vector(7 downto 0)`)

ramWriteEnable Ausgang: 1, wenn im nächsten Takt in der RAM geschrieben werden soll (`std_logic_vector(0 downto 0)`)

5.1.4. Trigger

Der Trigger entscheidet, ob die Aufnahme gestartet werden soll. Grundlage dafür sind die Einstellungen des Benutzers und die aktuellen Messwerte.

Abb. 7: Ein- und Ausgänge der Komponente Trigger

Die Komponente Trigger hat diese Ein- und Ausgänge (vgl. Abb. 5.1.4.):

clock Eingang für den Takt des Bausteins (`std_logic`)

state Eingang für die Einstellungen der Trigger für alle Kanäle (`AllTriggers`, vgl. 5.2.2.)

start Ausgang, der 1 ist, wenn die Trigger auslösen (`std_logic`)

reset Ausgang `start` auf 0 zurücksetzen (`std_logic`)

currentData Eingang für aktuelle Messwerte, die aufgenommen werden würden (`std_logic_vector(7 downto 0)`)

lastData Eingang für vorherige Messwerte für Flankenerkennung (`std_logic_vector(7 downto 0)`)

5.1.5. VGA

Die Komponente VGA erzeugt ein VGA-Signal mit 640x480 Pixeln und 60 Hz. Diese Größe und Frequenz wurde gewählt, da der Pixeltakt dann bei 25 MHz liegt, was grob dem halbem Takt des Bausteins (49.152 MHz) entspricht. Die Abweichung ist klein genug, sodass sie keine Rolle spielt.

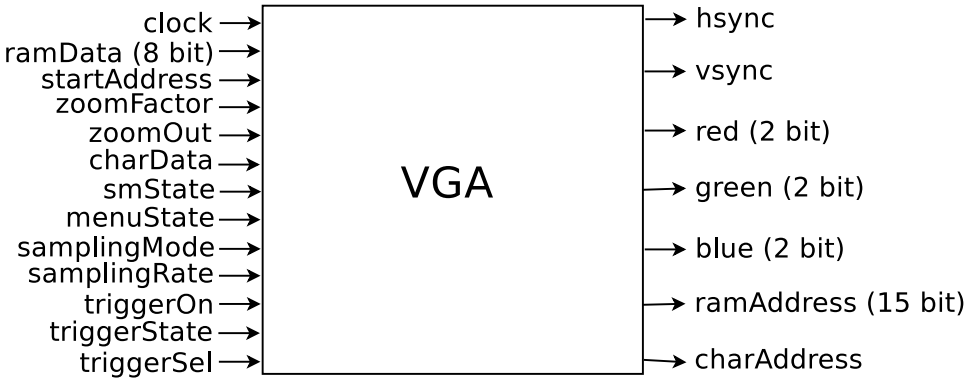

Abb. 8: Ein- und Ausgänge der Komponente VGA

Diese Komponente bekommt als Eingabe alle Informationen über das darzustellende Bild, vgl. die Eingänge des Bausteins.

Dementsprechend wird die Ausgabe dann gezeichnet. Da nicht genug Speicher für einen Bildspeicher verfügbar ist, wird in jedem Takt entschieden, welche Farbe der aktuelle Pixel hat. Dazu muss man für jeden Pixel, an den etwas gezeichnet werden soll, abfragen, ob dieser aktuell gezeichnet wird. Wenn dies der Fall ist, muss die passende Farbe auf die VGA Ausgänge gelegt werden. Es wurden Prozeduren geschrieben, die das Zeichnen von Linien, Rechtecken und Texten übernehmen:

setPixel(p: Point, c: Color) setzt einen einzelnen Pixel.

drawChar(p: Point, char : character) zeichnet einen Buchstaben. Die Position muss zwischen `(0,0)` und `(79, 59)` liegen (siehe unten).

drawString(p: Point, str: string) zeichnet einen String.

drawLine(fromP: Point, toP: Point, c:Color) zeichnet eine Linie.

drawRectangle(upperLeft: Point, lowerRight: Point, c: Color) Zeichnet ein Rechteck.

Hier wurden die Datentypen `Point` und `Color` verwendet. `Point` ist ein Tupel aus x und y-Koordinate. `Color` ist ein Tupel aus Rot, Grün und Blau-Wert, wobei jeder einzelne Wert 2 Bit hat. Einige Farben wurden als Konstanten definiert, z. B. `ColorBlack`. Die vollständige Liste findet man im Quellcode.
Damit das Zeichnen der Texte nicht zu aufwändig wird, wird dafür ein Bildspeicher verwendet: Die einzelnen Zeichen sind 8x8 Pixel groß und können nur in einem Raster mit 8 Pixel Abstand angeordnet werden. So können insgesamt 80x60 Zeichen dargestellt werden. Diese werden in einem Array gespeichert, was einem Bildspeicher entspricht. Da hier aber nur die ASCII-Werte der Zeichen gespeichert werden, verbraucht dies nicht zu viel Speicherplatz. Man muss darauf achten, dass alle Strings gleich lang sind, wenn man an eine Stelle abwechselnd unterschiedliche Texte schreiben will. Die Prozedur `drawString` schreibt den Text in den Bildspeicher. Wenn dieser zu kurz ist, bleibt noch das Ende der vorherigen Textes übrig. Hier muss man also noch mit Leerzeichen auffüllen.
Beim Zeichnen der aufgezeichneten Signale werden noch steigende und fallende Flanken ergänzt. Dabei wird einfach bei einem Wechsel des Signals eine senkrechte Linie dazu gezeichnet.
Die Komponente VGA hat diese Ein- und Ausgänge (vgl. Abb. 5.1.5.):

clock Eingang für den Takt des Bausteins (`std_logic`)

startAddress Adresse des ersten darzustellenden Wertes im Speicher (`std_logic_vector(14 downto 0)`)

ramAddress Ausgang für Speicheradresse. Die Daten an dieser Adresse liegen im nächsten Takt an `ramData` an (`std_logic_vector(14 downto 0)`)

ramData Eingang für Daten aus dem Speicher (`std_logic_vector(7 downto 0)`)

zoomFactor Eingang für Zoomfaktor: 1, 2 oder 4 für 100%, 200% oder 400% (`integer`)

zoomOut Eingang, der anzeigt, ob hinein oder herausgezoomt wird: 1 bedeutet herauszoomen, 0 hineinzoomen. Bei Zoomfaktor 1 ist dieser Wert egal (`boolean`)

charAddress Ausgang zur Ansteuerung der TextROMs (`character`)

charData Eingang von TextROM (`Letter`, vgl. 5.2.8.)

smState Eingang für Status der Logikanalyzers (`State`, vgl. 5.2.3.)

menuState Eingang für Status des Menüs (`Menu`, vgl. 5.2.4.)

samplingMode Eingang für verwendeten Sampling Modus (`SamplingMode`, vgl. 5.2.6.)

samplingRate Eingang für verwendete Sampling Rate (`SamplingRate`, vgl. 5.2.5.)

triggerOn Eingang, der 1 ist, wenn die Trigger aktiviert sind (`boolean`)

triggerState Eingang für Einstellungen der Trigger (`AllTriggers`, vgl. 5.2.2.)

triggerSel Eingang für im Triggermenu gewählten Kanal (0-7). 8 bedeutet, dass das Hauptmenü ausgewählt ist. (`integer range 0 to 8`)

hsync HSync-Ausgang für VGA-Signal (`std_logic`)

vsync VSync-Ausgang für VGA-Signal (`std_logic`)

red Ausgang für rote Farbkomponente des VGA-Signals (`std_logic_vector(1 downto 0)`, vgl. Kapitel 2.)

green Ausgang für grüne Farbkomponente des VGA-Signals (`std_logic_vector(1 downto 0)`)

blue Ausgang für blaue Farbkomponente des VGA-Signals (`std_logic_vector(1 downto 0)`)

5.1.6. TextROM

Der TextROM ist nur ein Speicher für die Schriftzeichen. Dabei wird für einen ASCII-Wert das Zeichen als 8x8 Pixel-Grafik gespeichert. Die Tabelle enthält aktuell nur Großbuchstaben, Zahlen und einige Sonderzeichen, um Platz zu sparen. Die Schriftzeichen wurden aus dem Linux Kernel übernommen.

Abb. 9: Ein- und Ausgänge der Komponente TextROM

Die Komponente TextROM hat diese Ein- und Ausgänge (vgl. Abb. 5.1.6.):

address Eingang mit ASCII-Zeichen (`character`)

char Ausgang mit 8x8 Pixel-Schriftzeichen (`Letter`, vgl. 5.2.8.)

5.2. Eigene Datentypen

Als Datentypen für einige Ein- und Ausgänge wurden im vorherigen Kapitel eigene Datentypen angegeben. Diese sind in der Datei GlobalTypes.vhd definiert und sollen hier erklärt werden:

5.2.1. TriggerState

`TriggerState` speichert die Einstellung des Triggers für einen Kanal. Es hat die Werte `Off`, `High`, `Low`, `Rising` und `Falling`. Die Bedeutung der Werte ist selbsterklärend.

5.2.2. AllTriggers

`AllTriggers` ist ein Array von 8 `TriggerState` Signalen. Darin werden die Triggereinstellungen für alle Kanäle gespeichert.

5.2.3. State

`State` speichert den Zustand, in dem sich der Logikanalyzer gerade befindet:

Start Der Nutzer befindet sich im Menü und kann die Einstellungen ändern.

WaitRunning Es wird auf das Auslösen des Triggers gewartet.

StartRunning Aufzeichnung wird gestartet. Hier ist der Eingang `start` des Samplers 1 (vgl. Kapitel 5.1.3.).

Running Die Aufzeichnung läuft. Der Eingang `start` des Samplers ist wieder 0.

View Der Nutzer kann durch die aufgenommenen Daten scrollen und zoomen.

Stopped Als Fehlerzustand gedacht, wird aktuell nicht verwendet. Die Aufzeichnung wird angehalten, falls sie läuft.

5.2.4. Menu

`Menu` speichert den Zustand des Menüs, d. h. die aktuell ausgewählte Option. Mögliche Werte sind `MSamplingMode`, `MSamplingRate`, `MTriggerOn`, `MTriggerSettings` und `MView`.

5.2.5. SamplingRate

`SamplingRate` speichert die Samplingrate. Mögliche Werte sind `s1`, `ms100`, `ms10`, `ms1` und `Max`.

5.2.6. SamplingMode

`SamplingMode` speichert den Aufzeichnungsmodus. Mögliche Werte sind `OneShot` und `Continuous`.

5.2.7. Timebase

Die Konstante `tb` vom Typ `Timebase` ist ein zweidimensionales Array, das jeder Zoomstufe und Samplingrate die entsprechende Timebase als String zuordnert.

5.2.8. Letter

`Letter` ist ein Bitarray mit Breite und Höhe 8. Darin werden die 8x8 Pixel großen Schriftzeichen gespeichert.

5.2.9. Font

`Font` ist ein Array aus `Letter`, das die Indizes 32 bis 90 hat. Der Index ist der ASCII-Wert eines Zeichens.

6. Schlusswort

Trotz einiger Startschwierigkeiten hat uns das ETI-Großpraktikum viel Spaß gemacht. Es war nicht nur wesentlich praxisnäher als das „normale“ Praktikum, wir haben auch einiges über VHDL und die Anwendungsmöglichkeiten der Sprache gelernt. Die Einarbeitungszeit, insbesondere das „Denken lernen in Hardware“, war recht mühsam. Nachdem allerdings die VGA-Ausgabe einmal korrekt funktioniert hatte, war der Rest eine sehr angenehme Arbeit. Einzig unser Zeitmanagement war nicht ganz ausgefeilt.
Alles in allem lässt sich sagen, dass wir das Praktikum so allen weiterempfehlen möchten, die ihre theoretischen Sprachkenntnisse aus der Vorlesung einmal „in echt“ anwenden und richtige Ergebnisse sehen wollen.

Abbildungsverzeichnis