Hochschule
Augsburg University of
Applied Sciences

Fakultät für
Informatik

Bachelorarbeit

Desgin und Implementierung eines FPGA-Event-Recorders mithilfe der freien IceStorm-Toolchain

Studienrichtung: Technische Informatik

Domenik Müller

Hochschule für angewandte
Wissenschaften Augsburg

An der Hochschule 1
D-86161 Augsburg

Telefon +49 821 55 86-0
Fax +49 821 55 86-3222
www.hs-augsburg.de
info@hs-augsburg.de

Fakultät für Informatik
Telefon +49 821 55 86-3450
Fax +49 821 55 86-3499

Verfasser der Diplomarbeit
Domenik Müller
Am Eser 3
86150 Augsburg
Telefon +49 821 44 92 57 54
domenik.mueller@hs-augsburg.de

Prüfer: Prof. Dr. Hubert Högl
Zweitprüfer: Prof. Dr. Alexander von Bodisco

Abgabedatum: 20.06.2018

Zusammenfassung

In der vorliegenden Arbeit wird der Entwurf und die Implementierung eines FPGA-basierten Event-Recorders mithilfe der Open-Source-Toolchain IceStorm beschrieben. Ähnlich wie bei einem Logikanalysator werden digitale Signaländerungen an den Eingängen erfasst, allerdings werden die Eingangssignale nicht wie bei einem Logikanalysator kontinuierlich übertragen, sondern werden bereits zur Erfassungszeit nach relevanten Eingangskombinationen gefiltert. Ein Filterung nach vordefinierten Events bietet sich vor allem für Timinganalysen bei relativ geringer Signaldichte an und kann die zeitliche Auflösung der untersuchten Signale verbessern. Die Implementierung des Event-Recorders erfolgt auf einem Raspberry Pi Zero mit IceZero FPGA-Shield und wird vollständig mit den von der IceStorm-Toolchain zur Verfügung gestellten Open-Source-Tools und Komponenten umgesetzt.

Abstract

The work in hand describes the design and implementation of a FPGA based event recorder using the open source toolchain IceStorm. Similar to a logic analyzer it records logic level signal changes, but in contrast to a logic analyzer it does not provide a continuous stream of the input signals. Instead the signals are filtered at capture time to match only relevant input combinations. Using predefined events to filter the input stream is especially suitable for timing analysis of signals with low event density and can improve the timing resolution of the analyzed signals. The implementation of the event recorder is done on a Raspberry Pi Zero with a IceZero FPGA shield and is realized completely with the tools and components provided by the open source toolchain IceStorm.

Inhaltsverzeichnis

1. Einführung

Mikrocontroller werden heute in Gebrauchsgegenständen aller Art verbaut und werden den Anforderungen entsprechend immer leistungsstärker und damit vor allem auch schneller. Selbst einfache Mikrocontroller arbeiten oft mit einer Geschwindigkeit im mehrstelligen Megaherz Bereich (sprich: mehrere Millionen Takte pro Sekunde). Dementsprechend werden für die Entwicklung von Mikrocontroller-Systemen (und digitalen Systemen im allgemeinen) Werkzeuge benötigt, mit denen Signale mit hoher zeilticher Auflösung erfasst und analysiert werden können.

Diese Aufgabe wird meist von einem Logikanalysator erfüllt, der die an den Eingängen anliegenden Spannungen mit einer festen Frequenz erfasst und die Daten dann z.B. an einen PC überträgt, an dem sie ausgewertet werden können.

In der vorliegenden Arbeit wird eine Spezialfrom eines Logikanalysators entworfen, bei der ein Teil der Auswertung bereits auf dem Logikanalysator durchgeführt wird. Dies wird erreicht in dem das Eingangssignal auf bestimmte - vom Benutzer definierte - Signaländerungen untersucht und dementsprechend gefiltert wird.

1.1 Zielsetzung

Für die Implementierung des Event-Recorders wurden folgende technische Ziele angestrebt:

- Es soll der logische Pegel von 8 bis 16 Eingangs-Pins abgefragt werden und die Eingangsdaten sollen mit einem stabilen Zeitstempel versehen werden.
- Die zeitliche Auflösung der Aufnahme soll im Megaherz-Bereich liegen
- Bestimmte Eingangskombinationen sollen in Textform definiert, und bei der Aufnahme als Events erkannt werden
- Zur Steuerung der Aufnahme soll ein Kommandozeilentool zur Verfügung stehen, mit dem auch die aufgenommenen Daten in Textform abgespeichert werden können.

1.2 Motivation

Die Arbeit schließt thematisch an die Bachelorarbeit "Ein universales, rekonfigurierbares und freies USB-Gerät zur Timing-, Protokoll-, Logik- und Eventanalyse von digitalen Signalen" von Andreas Müller und einer darauf folgenden Projektarbeit an.
In der Bachelorarbeit wurde eine Hardware-Platine namens "USB-TPLE" mit USB-Schnittstelle, einem CPLD-Chip von Altera und einem Atmega Mikrocontroller für die selbe Zielzetzung entworfen, und mit der Software-Implementierung begonnen[1].
Im nachfolgenden Semester-Projekt "Logikanalysator mit AVR Mega32U4 und Altera MAX CPLD" im Wintersemester 2013/14 wurde die Software-Implementierung ausgebaut und eine funktionsfähige Konfiguration für den CPLD-Chip entwickelt.

Im folgenden wird allerdings ein anderer Ansatz für die Umsetzung verfolgt:

Verwendung von käuflich verfügbarer Hardware
: Anstatt der selbst entworfenen Platine soll aus Gründen der Vefügbarkeit und um die Einstiegshürde für Benutzer zu verringern ein käuflich erwerbbares Produkt verwendet werden.
Die Verwendung käuflicher Hardware soll außerdem die Gesamt-Komplexität des Projekts reduzieren einen stärkeren Fokus auf Grundfunktionalität ermöglichen.

Verwendung eines FPGAs
: Um größere Flexibiltät bei der Implementierung zu ermöglichen wird ein Field Programmable Gate Array (FPGA) anstatt des Complex Programmable Logic Device (CPLD) verwendet (eine detailliertere Erklärung findet sich im Kapitel Design).

Neben Verfügbarkeit und Flexibiliät des Designs soll vor allem ein weiterer Grundsatz bei der Implementierung verfolgt werden:

Open-Source Software und Hardware
: Bereits die Arbeit von Andreas Müller wurde unter einer Open-Source-Lizenz veröffentlicht und es wurden alle Projekt-Quellen und Ressourcen (einschließlich des Hardwaredesigns) öffentlich vefügbar gemacht.
Dieser Ansatz soll hier weiter verfolgt werden, dementsprechend werden alle im Rahmen dieser Arbeit entstandenen Dokumente unter der LGPL3-Lizenz veröffentlicht (siehe Anhang B).
Ausserdem steht mit dem Projekt "IceStorm" erstmals auch eine Open-Source Software-Toolchain zur Programmierung von FPGA-Chips zur Verfügung, wodurch eine vollständige Open-Source Implementierung möglich wird. (In der vorliegenen Arbeit mit Ausnahme der proprietären Komponenenten des Raspberry Pi Zero).

Es ist eine Vielzahl von kommerziellen Logikanalysatoren am Markt verfügbar, allerdings bieten selbst sehr flexible Geräte wie z.B. die Discovery Serie von Digilent nicht die gewünschte Funktionalität der Event-Filterung zur Erfassugszeit mit der Möglichkeit die so gewonnenen Daten in einen Text- bzw. Kommandozeilen-basierten Workflow einzubetten [1].

Davon abgesehen gibt es auch einige Open-Source Logikanalysatoren. Für diese Arbeit relevant sind hier vor allem:

[1] Geräte der Discovery-Serie können durch eine API z.B. in Python geskriptet werden, eine kontinuierliche "Event-Erkennung" scheint aber nicht ohne weiteres möglich (siehe z.B. folgender Foreneintrag[2])

SUMP2 ist eine Verilog-basierte Logikanalysator-Implementierung mit UART-Schnittstelle für die Datenübertragung und einer zugehörigen - in Python implementierten - grafischen Benutzeroberfläche. Es existieren angepasste Varianten von SUMP2 die ohne weitere Modifikationen auf dem auch in dieser Arbeit verwendeten iCE40-FPGA-Chip lauffähig sind[3].

Open Bench Logic Sniffer ist ein Open-Source Hardware-Produkt das auf einem Xilinx Spartan 3E FPGA basiert und eine weiterentwickelte Variante von SUMP2 verwendet. Die "Demon core" betitelte Weiterentwicklung ist insbesondere deshalb interessant, da mit ihr ausgefeiltere Triggerbedingungen definiert werden können, und so z.B. zeitliche und logische Abläufe von Eingangssignalen als Trigger abgebildet werden können. Hierauf soll im Kaptiel Aussicht noch einmal eingegangen werden.

Das verwendete SUMP2 Datenübertragungsformat wird zum Teil auch von anderen Anwendungen unterützt, so kann zum Beispiel der Java-Client Jawi[4] oder Pulseview[5] als grafische Benutzeroberfläche verwendet werden.
Beide Varianten verwenden zur Datenübertragung eine serielle Schnittstelle (UART), die zumindest bei Verwendung von traditionellen Baud-Raten die Übertragungsgeschwindigkeit stark einschränkt. Ebenso sind beide Varianten konzeptionell für die Aufnahme relativ kurzer Sampling-Zeiten ausgelegt und unterstützen wie die kommerziellen Produkte keine Event-Filterung zum Erfassungszeitpunkt.

Eine Anpassung des SUMP2 Projektes wurde in Erwägung gezogen, aber aufgrund der zum Teil recht hohen Code-Komplexität und der strukturellen Unterschiede nicht durchgeführt.

1.3 Aufbau der Arbeit

Im folgenen Kapitel Design werden zunächst die nötigen technischen Grundlagen für die Umsetzung des Projekts besprochen, anschließend wird auf getroffene Desginentscheidungen bei der Auswahl der Hardware und Software eingegangen, und ein kurzes Implementierungs-Beispiel mit der IceStorm-Toolchain erläutert. Das Kaptiel Implementierung beschreibt di nötigen Anpassungen bestehender Software und die Entwicklung neuer Softwarekomponenenten bei der Durchführung des Projektes. Im Kapitel Anwendungsfall: Jitter-Analyse eines Software-generierten Clock-Signals wird die Benutzung des Event-Recorder anhand eines konkreten Beispiels besprochen. Es folgt ein Fazit in dem der Status des Projekts und die Umsetzung rekapituliert werden und abschließend wird im Kapitel Aussicht auf Optimierungsmöglichkeiten und weiteres Vorgehen eingegangen.

2. Design

Design lala ...

2.1 Technische Grundlagen

FPGA: datenaquise akurates timing - nyquist

2.2 Benötigte Hard- und Software-Komponenten

schneller Datenspeicher als Buffer: SRAM - alternativen

Mikrocontroller / PC: datenaufarbeitung und auswertung außerdem: treiber/verwaltung FPGA

Kommunikation: schnittstelle (UART, SPI, I2C, etc ..)

2.3 Auswahl der Software-Toolchain

2.4 Auswahl der Hardware

FPGA: iCE40 vielzahl boards: z.B. https://embeddedmicro.com/products/mojo-v3 -> SDRAM

-> keine Opensource-Toolchain!

iCE40: erst: https://www.olimex.com/Products/FPGA/iCE40/iCE40HX1K-EVB/open-source-hardware

dann icezero:

- formfaktor

Controller: pi: preis, vielfältigkeit, formfaktor

2.5 Beispiel: Von der Synthese bis zum Bitstream mit der IceStorm-Toolchain

3. Implementierung

...

3.1 Portierung des Tools zum Flashen des Bitstreams (icoprog)

...

3.2 Portierung und nötige Anpassungen des Verilog-SoCs (icosoc)

Bla fasel...

3.3 Implementierung des Event-Recorder Moduls

3.3.1 Bus-Schnittstelle

3.3.2 Triggerlogik

3.4 Implementierung eines SPI-Slave-Moduls

Bla fasel...

3.5 Zusammenführung der Module als Icosoc-Projekt

3.6 Implementierung des textbasierten Benutzerinterfaces

4. Anwendungsfall: Jitter-Analyse eines Software-generierten Clock-Signals

...

4.1 Einrichten des Projekts

...

4.2 Konfiguration der Event-Trigger

...

4.3 Durchführen der Event-Aufnahme

...

4.4 Analyse der Ergebnisse

...

5. Fazit

Bla fasel...

6. Aussicht

Bla fasel...

Abkürzungen

API Application Programming Interface. 2, 19, *Glossary:* API

CPLD Complex Programmable Logic Device. 2, 19, *Glossary:* CPLD

FPGA Field Programmable Gate Array. 2, 19, *Glossary:* FPGA

SPI Serial Peripheral Interface. 19, *Glossary:* SPI

UART Universal Asynchronous Receiver Transmitter. 3, 19, *Glossary:* UART

VHDL Very High Speed Integrated Circuit Hardware Description Language. 19, *Glossary:* VHDL

Glossar

API Schnittstelle zur Anwendungsprogrammierung. Erlaubt zum Beispiel die Verwendung von Programmkomponeneten durch eine exteren Skript-Sprache wie Python[6]. 19

CPLD Im Vergleich zu FPGAs deutlich einfacher Aufgebaute programmierbare logische Schaltungen[7]. 2, 19

FPGA Ein rekonfigurierbarer Chip dessen Schaltungsstruktur in einer Hardwarebeschreibungssprache (wie VHDL oder Verilog) frei programmierbar ist[8]. 2, 19

SPI Synchroner, serieller Datenbus für Datenübertragungen nach dem Master-Slave-Prinzip mit dem vergleichsweise hohe Datendurchsätze möglich sind (vgl. [9]). 19

UART Schnittstelle zur asynchronen, seriellen Datenübertragung (vgl. [10]). 19

Verilog Wortkreuzung aus "Verification" und "Logic". Hardwarebeschreibungssparche für Programmierung und Simulation von FPGAs und CPLDs die in den USA geläufiger ist als VHDL ([11], siehe auch [12] zur Namesnherkunft). 3

VHDL Vor allem in Europa verbeitetete Hardwarebeschreibungssprache für die Simulation und Programmierung von FPGAs und CPLDs[13]. 19

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

[1] A. Müller, „Ein universales, rekonfigurierbares und freies USB-Gerät zur Timing-, Protokoll-, Logik- und Eventanalyse von digitalen Signalen“, Bachelorarbeit, Hochschule Augsburg, 2010.

[2] *Advanced triggering and buffer filling*, [Online; accessed 31. May 2018], Mai 2018. Adresse: https://forum.digilentinc.com/topic/9488-advanced-triggering-and-buffer-filling.

[3] *SUMP2 – 96 MSPS Logic Analyzer for $22*, [Online; accessed 31. May 2018], Okt. 2016. Adresse: https://blackmesalabs.wordpress.com/2016/10/24/sump2-96-msps-logic-analyzer-for-22.

[4] *Open Bench Logic Sniffer - DP*, [Online; accessed 31. May 2018], Juni 2016. Adresse: http://dangerousprototypes.com/docs/Open_Bench_Logic_Sniffer.

[5] *Openbench Logic Sniffer - sigrok*, [Online; accessed 31. May 2018], Mai 2018. Adresse: https://sigrok.org/wiki/Openbench_Logic_Sniffer.

[6] *Programmierschnittstelle – Wikipedia*, [Online; accessed 31. May 2018], Mai 2018. Adresse: https://de.wikipedia.org/wiki/Programmierschnittstelle.

[7] *Complex Programmable Logic Device – Wikipedia*, [Online; accessed 26. May 2018], Mai 2018. Adresse: https://de.wikipedia.org/wiki/Complex_Programmable_Logic_Device.

[8] *Field Programmable Gate Array – Wikipedia*, [Online; accessed 26. May 2018], Mai 2018. Adresse: https://de.wikipedia.org/wiki/Field_Programmable_Gate_Array.

[9] *Serial Peripheral Interface – Wikipedia*, [Online; accessed 31. May 2018], Mai 2018. Adresse: https://de.wikipedia.org/wiki/Serial_Peripheral_Interface.

[10] *Universal Asynchronous Receiver Transmitter – Wikipedia*, [Online; accessed 31. May 2018], Mai 2018. Adresse: https://de.wikipedia.org/wiki/Universal_Asynchronous_Receiver_Transmitter.

[11] *Verilog – Wikipedia*, [Online; accessed 26. May 2018], Mai 2018. Adresse: https://de.wikipedia.org/wiki/Verilog.

[12] S. Golson, *Oral History of Philip Raymond “Phil” Moorby*. Computer History Museum, 2013, S. 23–25. Adresse: http://archive.computerhistory.org/resources/access/text/2013/11/102746653-05-01-acc.pdf.

[13] *Very High Speed Integrated Circuit Hardware Description Language – Wikipedia*, [Online; accessed 26. May 2018], Mai 2018. Adresse: https://de.wikipedia.org/wiki/Very_High_Speed_Integrated_Circuit_Hardware_Description_Language.

A. Pinbelegung

Anhang A...

A.1 Tex Beispiele

A.1.1 Zitieren

Quellen**li00**, **jackson91**, **lakhina04a**, **netflow**, **rfc2386** nicht vergessen. Dazu verwendet ihr bibtex.

A.1.2 Ein Bild skaliert

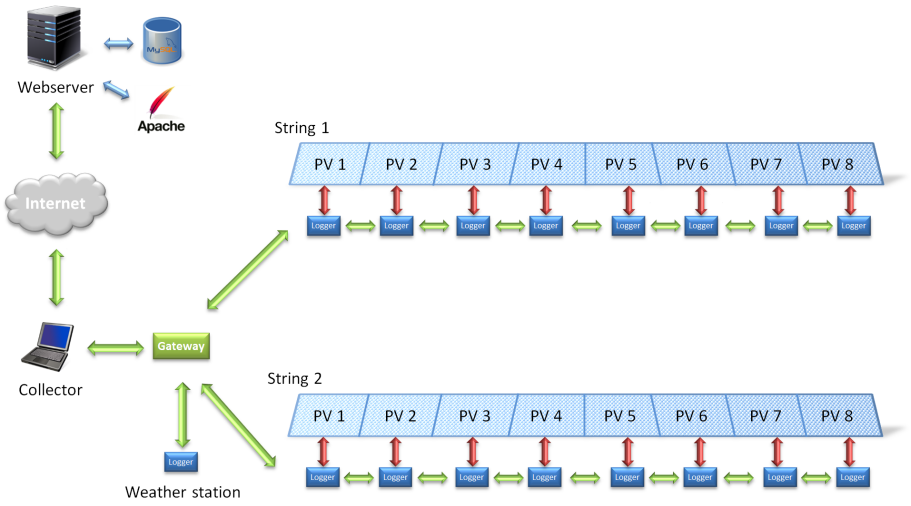
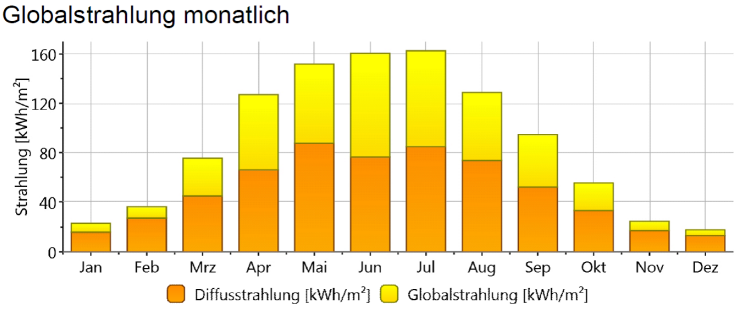


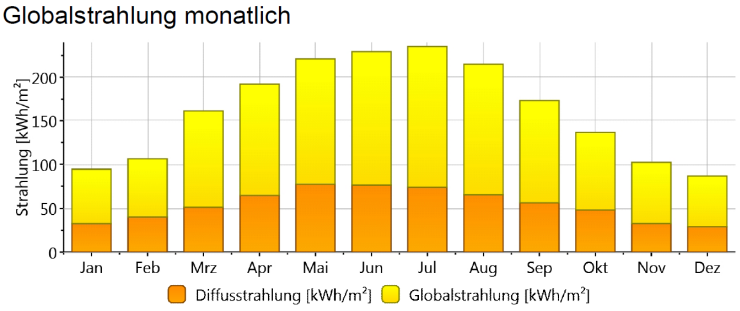
Abbildung A.1: Beschriftungstext

A.1.3 Zwei Bilder nebeneinander oder untereinander

A.2 Tabellen



(a) Beschriftung Bild links



(b) Beschriftung Bild rechts

Abbildung A.2: Beschriftung beide Bilder

Tabelle A.1: Single-hop Scenario - Traffic Pattern

Pattern	Parameter	Distribution	Range/Values
Burst	Burst IAT	uniform	[9.9; 10.1] s
	Packets per Burst	constant	100
	Packet IAT	constant	0.02 s
	Packet Size	constant	1024 bit
	# Sources	-	2
	Offset	uniform	[0; 1] s
Single	Packet IAT	uniform	[0.9; 1.1] s
	Packet Size	constant	1024 bit
	# Sources	-	[10;20;30;40;50; 60;70;80;90;100]
	Offset	uniform	[0; 1] s

B. GPL

Anhang B ...