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 **1**

1.1 Motivation 2

1.2 Abgrenzung von bestehenden Lösungen zur Logikanalyse 3

1.3 Zielsetzung 3

1.4 Aufbau der Arbeit 4

2 Design **5**

2.1 Theoretische Grundlagen 5

2.1.1 Zeit- und wertdiskrete digitale Signale 5

2.1.2 Definition “Event” 6

2.2 Überblick der benötigten Hard- und Software-Komponenten 8

2.2.1 Datenerfassung: FPGA 8

2.2.2 Datenzwischenspeicher: SRAM 9

2.2.3 Datenübertragung: SPI 9

2.2.4 Steuerung der Aufnahme und sequentielle Programmabläufe 10

2.3 Auswahl der Software-Toolchain: IceStorm 11

2.4 Auswahl der Hardware 12

2.4.1 IceZero FPGA-Shield (iCE40HX4K) 12

2.4.2 Raspberry Pi Zero W 12

2.5 Beispiel: Von der Synthese bis zum Bitstream mit der IceStorm-Toolchain . 13

3 Implementierung **14**

3.1 Portierung des Tools zum Flashen des Bitstreams (icoprog) 15

3.2 Portierung und nötige Anpassungen des Verilog-SoCs (icosoc) 16

3.3 Implementierung des Event-Recorder Moduls 17

3.3.1 Bus-Schnittstelle 17

3.3.2 Triggerlogik 17

3.4 Implementierung eines SPI-Slave-Moduls 18

3.5 Zusammenführung der Module als Icosoc-Projekt 19

3.6 Implementierung des textbasierten Benutzerinterfaces 19

4 Anwendungsfall: Jitter-Analyse von Software-generierten MIDI-Clock Signalen 20

4.1 Test-Setup: USB-Midi mit Teensy LC 21

4.2 Einrichten des Projekts 22

4.3 Konfiguration der Event-Trigger 23

4.4 Durchführen der Event-Aufnahme 23

4.5 Analyse der Ergebnisse 24

4.5.1 Software MIDI-Clock: Python-Implementierung 25

4.5.2 Software MIDI-Clock: Renoise und Reaper 26

4.5.3 Hardware MIDI-Clock: Midipal 27

5 Fazit 29

6 Aussicht 30

Abkürzungen 31

Glossar 32

Abbildungsverzeichnis 33

Tabellenverzeichnis 34

Literatur 35

A Material 37

A.1 PMOD-Pinbelegung des IceZero-Boards 37

A.2 Pinverbindungen Raspberry Pi und FPGA-Shield 38

A.3 CD 39

B Lizenz 40

B.1 Creative Commons Attribution-ShareAlike 4.0 International 40

B.2 Creative Commons Legal Code 40

B.2.1 Attribution-ShareAlike 3.0 Unported 40

B.3 ISC-Lizenz 47

1. Einführung

Mikrocontroller werden heute in Gebrauchsgegenständen aller Art verbaut und werden den Anforderungen entsprechend immer leistungsstärker und damit unter anderem auch schneller. Selbst einfache Mikrocontroller arbeiten oft mit einer Geschwindigkeit im mehrstelligen Megaherz Bereich (sprich: mehrere Millionen Takte pro Sekunde). Im Unterschied zu klassischen PC-Systemen werden an Mikrocontroller oft Echtzeit-Anforderungen gestellt, das heißt Ergebnisse müssen zuverlässig innerhalb einer vorbestimmten Zeitspanne geliefert werden[1]. Dabei wird auch die Hardware von Mikrocontrollern zunehmend komplexer und es werden vermehrt Mehrprozessor-Systeme verwendet, die angepasste und mitunter unübersichtlichere Programmiertechniken erfordern.
Dementsprechend werden für die Entwicklung von Mikrocontroller-Systemen (aber auch von digitalen Systemen im allgemeinen) Werkzeuge benötigt, mit denen Signale mit hoher zeitlicher Auflösung erfasst und analysiert werden können.
Ergänzend zu Simulations- und Software-gestützten Verfahren wird diese Aufgabe meist von Logikanalysatoren erfüllt, die die an den Eingängen anliegenden Spannungen mit einer festen Frequenz erfassen und die Daten dann zum Beispiel an einen PC übertragen, an dem sie ausgewertet werden können.
In der vorliegenden Arbeit wird eine spezielle Form von Logikanalysator entworfen, bei der ein Teil der Auswertung bereits auf dem Logikanalysator durchgeführt wird. Das Eingangssignal wird auf bestimmte - vom Benutzer definierte - Signaländerungen untersucht und nur relevante Signaländerungen (“Events”) werden an den Benutzer weitergereicht.
Dieses Vorgehen bietet sich vor allem dann an, wenn der Signalverlauf grundsätzlich bekannt ist und der Fokus der Analyse auf den exakten zeitlichen Abbildung des erwarteten Signalverlaugs liegt.
Ein Beispiel wären die Ausgänge eines Mikrocontrollers, bei dem bewusst bestimmte Kombinationen gesetzt werden um den Start und das Ende von Funktionen im Quellcode zu signalisieren. Da das Setzen von GPIO-Pins meist in einem einzigen CPU-Takt ausgeführt werden kann, können so zuverlässige Aussagen zur Laufzeit von Funktionen, oder bei periodischer Ausführung auch zur zeitlichen Fluktuation der Funktionsausführung (“Jitter-Analyse”) getroffen werden.

1.1 Motivation

Die vorliegende 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 Zielsetzung entworfen, und mit der Software-Implementierung begonnen[2].
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 Flexibilität 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 Flexibilität des Designs soll vor allem ein weiterer Grundsatz bei der Implementierung verfolgt werden:

Verwendung von 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 verfügbar gemacht.
Dieser Ansatz soll hier weiter verfolgt werden, dementsprechend steht der im Rahmen dieser Arbeit entstandene Quelltext (wie große Teile der IceStorm-Toolchain) unter der ISC-Lizenz zur Verfügung. Die Bachelorarbeit selbst wird unter Creative Commons Attribution-ShareAlike 4.0 International Lizenz veröffentlicht.
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 vorliegenden Arbeit mit Ausnahme der proprietären Komponenten des Raspberry Pi Zero).

In Kombination mit der preisgünstigen[1] Hardware bietet die IceStorm-Toolchain eine interessante Alternative zu den Angeboten der großen FPGA-Hersteller wie Xilinx oder Intel (ehemalig Altera), insbesondere für Lehrzwecke und kleinere Projekte.

[1] Der Preis des verwendeten IceZero-Boards lag zum Zeitpunkt der Erstellung der Arbeit zum Beipiel bei ca. 40€, das zur Programmierung verwendete Raspberry Pi Zero W bei unter 15€

1.2 Abgrenzung von bestehenden Lösungen zur Logikanalyse

Es ist eine Vielzahl von kommerziellen Logikanalysatoren am Markt verfügbar. Allerdings bieten selbst flexible Geräte wie zum Beispiel die Discovery Serie von Digilent nicht die gewünschte Funktionalität der Event-Filterung zur Erfassungszeit und die Möglichkeit die so gewonnenen Daten in einen Text- bzw. Kommandozeilen-basierten Workflow einzubetten [2].

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

SUMP2 ist eine Verilog-basierte Logikanalysator-Implementierung mit 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[4].

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 unter anderem deshalb interessant, da mit ihr detailliertere Triggerbedingungen definiert werden können, und so zum Beispiel 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 unterstützt, so kann zum Beispiel der Java-Client Jawi[5] oder Pulseview[6] (ein Qt-Frontend der libsigrok-Biliothek) als grafische Benutzeroberfläche verwendet werden.
Beide Varianten verwenden zur Datenübertragung eine serielle Schnittstelle (UART), die - zumindest bei Verwendung von geläufigen Baud-Raten - die Übertragungsgeschwindigkeit stark einschränkt. Ebenso sind beide Varianten konzeptionell für die Aufnahme festgelegter und relativ kurzer Sampling-Zeiten ausgelegt und unterstützen — wie die kommerziellen Produkte — keine Event-Filterung zur Erfassungszeit.

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 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.

[2] 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[3])

1.4 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 von Software-generierten MIDI-Clock Signalen 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

2.1 Theoretische Grundlagen

Zunächst sollen die grundlegenden Eigenschaften der erwarteten Eingangssignale definiert und näher beschrieben werden.

2.1.1 Zeit- und wertdiskrete digitale Signale

Bei den Eingangssignalen des Event-Recorders handelt es sich um die Ausgänge von Mikrocontrollern oder anderer digitaler Schaltungen und damit um digitale Signal. Digitale Signale sind durch zwei grundlegende Eigenschaften charakterisiert, sie sind:

- **zeitdiskret**, und
- **wertdiskret**

Wertdiskret bedeutet, dass das Signal nur genau einen Wert aus einer festgelegten Anzahl möglicher Wertezustände annehmen kann. In den meisten Fällen beschränkt sich der Wertebereich auf die Binärwerte “1” oder “0”, das heißt ein digitales Signal ist zum Beispiel bei 0,1 Volt Spannung “0” und bei 3,2 Volt “1”, wohingegen ein analoges Signal alle möglichen Werte zwischen 0 und 3,3 Volt annehmen kann.

Zeitdiskret bedeutet, dass ein digitales Signal “nur zu bestimmten periodischen Zeitpunkten definiert ist bzw. nur dann eine Veränderung im Signalwert aufweist”[7], das heißt dass das Signal bei der Erfassung mit einem festen “Zeitraster” abgetastet wird und zwischen den Rasterpunkten einen festen Wert behält.

Bei der Erfassung digitaler Signale ist es nötig, dass die Abtastfrequenz nach Nyquist-Shannon-Abtasttheorem mindestens doppelt so hoch als die maximale Frequenz des untersuchten Signals sein muss, damit keine Informationen aus dem Ausgangsignal verloren gehen (vgl. [7]).

Es ist zu beachten, dass es sich bei der Definition von digitalen Signalen um eine idealisierte Ansicht handelt und dass sich bei realen digitalen Signale oft – vor allem bei hohen Frequenzen – Störeffekte zeigen. Ein Beispiel für einen Störeffekt wäre das “Prellen” eines mechanischen Schalters, bei dem es durch den mechanischen Kontakt zu einem mehrfachen Signalwechsel kommen kann, bevor ein stabiles Signal anliegt. Wenn der Schalter-Zustand

dabei mit vergleichsweise langsamer Geschwindigkeit ausgelesen wird, gehen die deutlich schnelleren Signalwechsel gemäß dem Nyquist-Shannon-Abstasttheorem nicht in das Ausgangssignal ein, bei entsprechend hoher Abtastrate werden sie allerdins mit in Ausgangssignal übernommen und können dann unerwünschte Effekte auslösen.
Bei der in dieser Arbeit verwendeten Hardware sind keine speziellen Vorrichtungen (wie zum Beispiel "Schmitt-Trigger") vorhanden um solchen Effekten entgegen zu wirken, das heißt es wird am Eingang ein für die Analyse aussreichend stabiles Signal erwartet.

Damit die Ergebnisse des Event-Recorders richtig auswertet werden können, muss zu jeder Signaländerung der "diskrete" Zeitpunkt bekannt sein, das heißt das Zeitraster der Abtastung muss numeriert werden, so dass jeder Signaländerung ein eindeutiger Zeitstempel zugeordnet werden kann. Diese "Numerierung" wird umgesetzt, in dem der von einem Oszillator[1] generierte Schaltungstakt mit einer Zählerschaltung aufaddiert wird. Der Zeitstempel entspricht dann einfach dem aktuellen Zählerstand.

EVENT_ID [8 Bit]	INPUT_DATA [8 Bit]	TIMESTAMP [16 Bit]
0x02	0x01	0x0EF8

Tabelle 2.1: Beispielhafte Darstellung eines aufgenommenen Events mit 16-Bit Zeitstempel

Die Bit-Breite des Zählers legt zusammen mit der Geschwindigkeit des Takts die maximal mögliche Aufnahmelänge fest. Der verwendete Zähler hat eine Breite von 46 Bit, womit sich bei einer Taktfrequenz von 100 Mhz zum Beispiel eine Laufzeit von ca. 195 Stunden ergibt[2] (was für die meisten Anwendungsfälle mehr als aussreichend ist).

2.1.2 Definition "Event"

Für diese Arbeit soll unter dem Begriff "Event" ein vom Benutzer festgelegter Signalzustand oder eine Signaländerung an den Eingangspins des Event-Recorders verstanden werden. In den meisten Fällen macht es mehr Sinn eine Signaländerung zu definieren, als einen Zustand, da bei einem anhaltenden Zustand auch das Event kontinuierlich "ausgelöst" wird, und ein dementsprechend hohes Datenvolumen erzeugt.
Es ergeben sich folgende Definitionsmöglichkeiten für die einzelnen Eingangs-Pins:

- "0": niedriger logischer Pegel
- "1": hoher logischer Pegel
- "u": steigender Pegel
- "d": fallender Pegel
- "x": beliebiger Zustand ("don't care")

Zusätzlich wäre eine Kombination von "u" und "d" denkbar, die auf beliebige Signaländerungen reagiert. Eine Verkettung dieser Möglichkeiten bei der jedem Eingangspins ein Zustand zugewiesen wird soll als "Event-Trigger" – also als Auslöser eines bestimmten Events – bezeichnet werden.
Dies entspricht im Wesentlichen der Definition von *Trigger* die bei "traditionellen" Logikanalysatoren verwendet wird, allerdings mit dem Unterschied dass der Trigger bei einem

[1] Eigener Hardware-Baustein, der eine konstant auf- und abschwingende Spannung erzeugt und damit zur Taktung digitaler Schaltung verwendet werden kann

[2] Berechnung: $2^{46} * \frac{1}{100Mhz} = 2^{47} * 10ns$

"traditionellen" Logikanalysator die eigentliche Aufnahme einmalig auslöst, und dann bis zum Ende der Aufnahme nicht mehr von Bedeutung ist, während ein Event-Trigger als Teil der Aufnahme kontinuierlich überprüft werden muss und erst bei Erfüllung der Triggerbedingung überhaupt Ausgangsdaten generiert werden.
Es bietet sich an einem Event zusätzlich bestimme Funktionen zuweisen zu können, wie zum Beispiel das Starten[3] und Stoppen der Event-Erkennung, oder das Wechseln in einen zusätzlichen Modus, bei dem auf alle Signaländerungen reagiert wird.
Wie in der Einführung erwähnt soll die Definition der Events in Text-Form möglich sein und kann dann z.B. folgendermaßen aussehen:

```
# event configuration
events:
  - start:
          trigger: uxxxxxxx
          func: start

  - stop:
          trigger: dxxxxxxx
          func: stop

  - event1:
          trigger: 1uxxxxxx

  - event2:
          trigger: 1xuxxxxx
          func: dump_begin
...
```

[3] Die Erkennung eines Start-Events erfordet folglich, dass auch im "Ruhezustand" eine Event-Erkennung durchgeführt wird

2.2 Überblick der benötigten Hard- und Software-Komponenten

2.2.1 Datenerfassung: FPGA

Wie im vorherigen Kapitel beschrieben wird für die Datenerfassung eine Zählerschaltung benötigt, die einen stabilen Zeitstempel liefern kann. Vorraussetzung dafür ist, dass der Zähler kontinuierlich läuft und nicht durch andere Vorgänge unterbrochen werden kann. Dies ist bei PC-Systemen oder auch Mikrocontrollern nicht ohne weiteres möglich, da die Programmausführung zu jedem Zeitpunkt von Betriebssystem-Funktionen oder Interrupt-Routinen[4] pausiert werden kann.
Deswegen bietet sich hier die Verwendung einer programmierbaren logischen Schaltung wie zum Beispiel eines CPLDs (Complex Programmable Logic Device) oder FPGAs (Field Programmable Gate Array) an, bei dem eine von anderen Komponenten vollkommen unabhängige parallele Ausführung des Zählers möglich ist.
Sowohl CPLD- als auch FPGA-Chips bestehen aus einer Vielzahl von einheitlichen Blöcken ("PLBs") die einfache logische Funktionen abbilden können. Im Vergleich zu logischen Gattern sind die die Blöcke in ihrer Funktion aber frei konfigurierbar. Durch die Vernetzung der so konfigurierten Blöcke können umfangreiche digitale Schaltungen realisiert werden.
FPGAs sind etwas komplexer aufgebaut als CPLDs, dabei aber auch flexibler bei der Vernetzung und enthalten oft zusätzliche Funktionsblöcke wie den in der nachfolgenden Abbildung erkennbaren Block-RAM als Zwischenspeicher für größere Datenmengen oder die PLL-Einheit zur Erzeugung von Taktsignalen mit konfigurierbarer Geschwindigkeit (vgl. [8]).

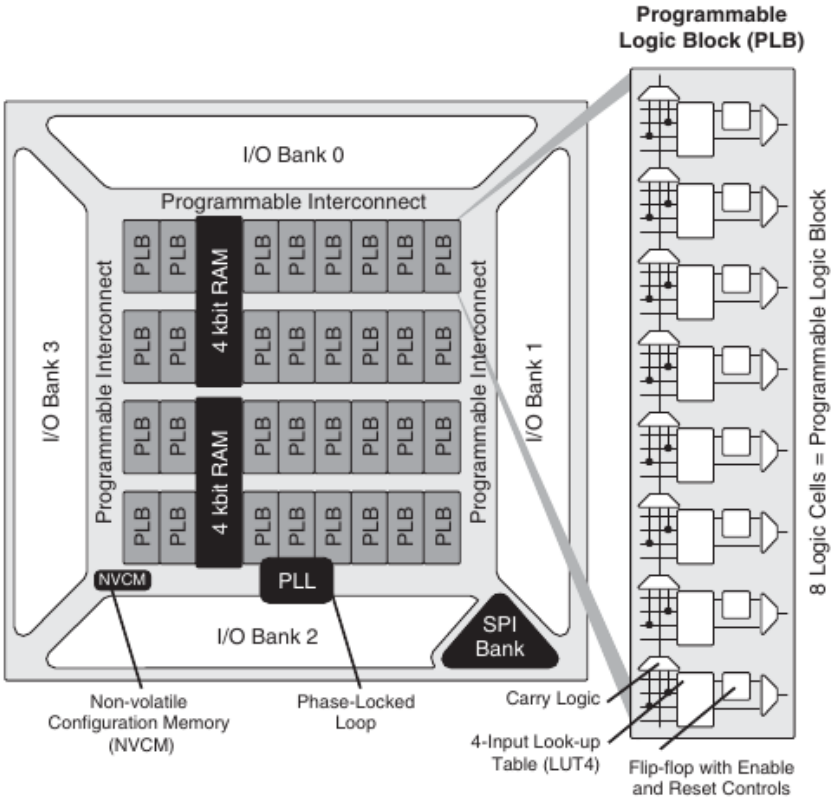


Abbildung 2.1: Aufbau des verwendeten iCE40 FPGAs (Quelle: iCE40 Datasheet[9])

Zusätzlich ist die Anzahl von Funktionsblöcken bei FPGAs üblicherweise deutlich höher als als bei CPLDs, weswegen in dieser Arbeit ein FPGA-Chip verwendet werden soll.

[4]Bei einer Interrupt-Routine wird die aktuelle Programmausführung auf CPU-Ebene pausiert, zum Beispiel um auf Ereignisse externer Hardwaregeräte reagieren zu können

Die in der Abbildung erkennbaren I/O-Bänke beinhalten GPIO-Pins die als Ein- oder Ausgänge konfiguriert und direkt mit den Logikblöcken ("PLBs") im FPGA verknüpft werden können.

2.2.2 Datenzwischenspeicher: SRAM

Nach der Erfassung müssen die Daten zwischengespeichert werden. Dabei sind vor allem zwei Faktoren entscheidend:

- Die **Geschwindigkeit** des Speichers, die meist den maximalen Datendurchsatz der Gesamtschaltung bestimmt
- Die **Größe** des Speichers, die festlegt wie lange bei hohem Datendatendurchsatz aufgenommen werden kann

Die meisten nicht-flüchtigen Speicher sind aufgrund der unzureichenden Geschwindigkeit nicht für diesen Anwendungszweck geeignet, weswegen sich die Verwendung von RAM-Speicher anbietet.
Bei Mikrocontrollern und kleineren FPGA-Boards werden aus Kostengründen und wegen der unkomplizierten Ansteuerung oft SRAM-Speichereinheiten verbaut. SRAM-Speicher hat meist sehr kurze Zugriffszeiten, allerdings bei einer geringen Speichergröße von wenigen MBit. Der in dieser Arbeit verwendete SRAM-Speicher hat beispielsweise eine Größe von 4 Mbit (512 KB) und könnte damit 62500 Events zwischenspeichern[5].
Bei kontinuierlicher Erfassungs mit einer Abtastrate von 100 Mhz entspricht dies einer Aufnahmezeit von unter einer Millisekunde[6]. Da allerdings keine Eingangssignale erwartet werden, bei denen Events mit einer Frequenz von 100 Mhz auftreten und außerdem bereits bei laufender Aufnahme Events aus dem RAM-Speicher entnommen und weiter übertragen werden, kann der SRAM-Speicher trotzdem im Sinne der Aufgabenstellung als Zwischenspeicher verwendet werden.
Als Alternative könnte DRAM-Speicher verwendet werden, der ein Vielfaches der Speichergröße bietet und damit auch längere Aufnahmen bei hoher Signaldichte ermöglichen würde. DRAM-Speicher erfordert allerdings im Vergleich zu SRAM ein kontinuierliches "Auffrischen" der Speicher-Inhalte durch eine Controller-Einheit und bedeutet deshalb deutlich mehr Aufwand bei der Implementierung.

2.2.3 Datenübertragung: SPI

Nach der Datenerfassung und Zwischenspeicherung werden die Daten an ein externes System übertragen, an dem sie weiterverarbeitet oder ausgewertet werden können. Zur Datenübertragung gibt es eine Vielzahl von Schnittstellen und Protokollen. Dabei werden die Daten im Normalfall "serialisiert", das heißt wenn ein Event aus 64 Bit besteht, werden die Bits nacheinander über eine einzige Datenleitung übertragen. Geläufige serielle Datenübertragungsverfahren sind vor allem UART, I^2C und SPI.

Bei einem **UART (Universal Asynchronous Receiver Transmitter)** werden die Daten über eine Empfangs- und eine Sendeleitung ausgetauscht. Es wird kein eigenes Taktsignal übertragen, weswegen auf beiden Seiten eine feste Übertragungsgeschwindigkeit ("Baud-Rate") eingestellt werden muss. UART-Schnittstellen sind weit verbreitet, bei üblichen Baud-Raten ist die Übetragungsgeschwindigkeit allerdings relaitv gering (vgl. [10]).

[5] Bei einer angenommenen Event-Größe von 64-Bit, und unter der Annahme dass der SRAM-Speicher ausschließlich zum Zwischenspeichern von Events verdwendet wird

[6] $62500 * 10ns = 0.625ms$

I^{2}C (Inter-Integrated Circuit) ist ein synchroner Datenbus, bei dem eine eigene Leitung für den Takt und eine Datenleitung verwendet wird. I^{2}C arbeitet nach dem Master-Slave-Prinzip, das heißt es können auch mehrere Geräte miteinander kommunizieren. I^{2}C untertützt Übertragungsraten bis zu 5 Mbit/s (unidirektional, vgl. [11]).

SPI (Serial Peripheral Interface) ist wie I^{2}C ein synchroner Datenbus nach dem Master-Slave-Prinzip. Neben der Leitung für den Takt wird eine Daten-Leitung in Senderichtung und eine Datenleitung in Empfangsrichtung verwendet. Zusätzlich wird pro Gerät eine "Chip-Select" Leitung benötigt, um die Geräte adressieren zu können. SPI kann Übertragungsraten bis in den mehrstelligen Megabit-Bereich ermöglichen (vgl. [12]).

Aufgrund der hohen Übertragungsrate und der relativ unkomplizierten Implementierungsmöglichkeiten soll SPI für die Datenübetragung verwendet werden.

2.2.4 Steuerung der Aufnahme und sequentielle Programmabläufe

Neben der reinen Datenerfassung und -Übertragung werden noch Komponenten zur Steuerung und zur Kontrolle des Aufnahmevorgangs benötigt.
Grundsätzlich können die benötigten Vorgänge- und Zustände (zum Beispiel das Starten und Stoppen der Aufnahme) direkt im FPGA umgesetzt werden. Als Kommunikationsweg zur Steuerung und Konfiguration kann dann wiederum SPI verwendet werden.
Bei längeren oder komplexeren Programmabläufen die keine zeitkritische Ausführung erfordern bietet sich die Verwendung eines zusätzlichen Mikrocontrollers an. Da die meisten PC-Systeme keine programmierbaren GPIO-Pins als SPI-Schnittstelle zur Verfügung stellen, kann ein Mikrocontroller außerdem als Brücke zwischen FPGA und Anwendersystem fungieren, und zum Beispiel die vom FPGA erfassten Daten über einen USB-Port zur Verfügung stellen.
Davon abgesehen wird auch auch die entsprechende Hardware- und Software-Infrstruktur benötig um das FPGA und den Mikrokonroller zu programmieren und zu konfigieren.

2.3 Auswahl der Software-Toolchain: IceStorm

2.4 Auswahl der Hardware

2.4.1 IceZero FPGA-Shield (iCE40HX4K)

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

2.4.2 Raspberry Pi Zero W

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 von Software-generierten MIDI-Clock Signalen

Als Beispiel für ein zeitkritisches Signal sollen im Folgenden mehrere MIDI-Clock Signale mit dem Event-Recorder untersucht werden, und dabei der allgemeine Ablauf einer Event-Aufnahme und der nachfolgenden Analyse geschildert werden. Das MIDI-Clock Signal wird in den untersuchten Fällen software-basiert – auf einem normalen Anwender-System – generiert, deswegen ist von einer messbaren zeitlichen Fluktuation ("Jitter") auszugehen. Das heißt die Clock-Signale treten nicht exakt zum erwarteten Zeitpunkt sondern leicht zeitlich verfrüht oder verspätetet auf.
MIDI ist ein 1983 spezifizierter Standard für den Austausch von Steuerinformation zwischen elektronischen Musikintrumenten und wird trotz einiger signifikanter Limitierungen seit über 35 Jahren nahezu unverändert für praktisch alle elektronischen Musikinstrumente im professionellen und im Hobby-Bereich verwendet. MIDI bietet mit der "MIDI-Clock" eine Möglichkeit mehrere Geräte zeitlich zu synchronisieren. So könnte zum Beispiel ein Synthesizer mit der Musiksoftware auf einem PC synchronisiert werden, um tempo-abhängige Arpeggios[1] zu spielen.
Dabei verwendet MIDI zur Datenübertragung das gleiche Protokoll wie ein UART (ohne Parity-Bit) bei einer Übertragungsgeschwindigkeit von 31250 Bit/s. Die meisten MIDI-Nachrichten setzen sich aus einem Stautsbyte und zwei darauf folgenden Datenbytes zusammten, für die MIDI-Clock werden allerdings nur einzelne Bytes benötigt.
Zuerst wird ein Start-Byte (0xFA) übertragen, dann wird – abhängig vom Tempo – 24 mal pro Viertelnote ein "Clock-Tick" (0xF8) gesendet, und abschließend ein Stopp-Bit (0xFC).

[1] Ein Akkord bei dem die einzelnen Töne nicht gleichzeitig, sondern zeitversetzt nacheinander gespielt oder ausgelöst werden

Eine einfache Implementierung in Python mithilfe des `python-rtmidi`-Moduls könnte folgendermaßen aussehen:

```
# MIDI CLOCK TEMPO (beats per minute)
BPM = 80

# define clock messages
clock_start = [0xFA]
clock_tick  = [0xF8]
clock_stop  = [0xFC]
[...]
    # calculate clock period
    clock_period = 60 / (BPM * 24)

    # send start byte
    midiout.send_message(clock_start)

    # run
    while (True):
        midiout.send_message(clock_tick)
        time.sleep(clock_period)
```

4.1 Test-Setup: USB-Midi mit Teensy LC

Um das MIDI-Signal auswerten zu können wird ein Mikrokontroller ("Teensy LC") verwendet, der per USB an einen PC angeschlossen werden kann, und über die USB-Schnittstelle ein MIDI-Gerät simuliert. Das heißt am PC wird eine virtuelle MIDI-Schnittstelle erzeugt, und alle ankommenden MIDI-Daten werden direkt zum Mikrocontroller übertragen. Der Mikrocontroller wartet auf das Start-Byte einer MIDI-Clock und setzt einen GPIO-Pin auf "1" um dem Event-Recorder den Anfang der Aufnahme zu signalisieren. Danach wird bei jedem empfangenen Clock-Tick ein zweiter GPIO-Pin kurz auf "1" gesetzt und anschließend wieder auf "0". Dies soll beim Event-Recorder als Event erkannt werden. Beim Empfangen des Stopp-Bytes wird der erste GPIO-Pin auf "0" gesetzt und die Aufnahme soll beendet werden.
Der Mikrocontroller kann mit der Arduino-IDE programmiert werden und es steht eine MIDI-Bibliothek zur Verfügung, was eine kurze und unkomplizierte Umsetzung erlaubt:

```
[...]
// midi clock start handler (0xFA)
void onStart() {
  digitalWrite(0, HIGH);
}

// midi clock tick handler (0xF8)
void onClock() {
    digitalWrite(1, HIGH);
    digitalWrite(1, LOW);
}

// midi clock stop handler (0xFC)
void onStop() {
  digitalWrite(0, LOW);
```

```
}

void setup() {
  // pin setup
  [...]
  // register midi handlers
  usbMIDI.setHandleStart(onStart);
  usbMIDI.setHandleStop(onStop);
  usbMIDI.setHandleClock(onClock);
}

void loop() {
  usbMIDI.read();
}
```

4.2 Einrichten des Projekts

Die vollständige Einrichtung des Raspberry Pi Zero W soll hier aus Platzgründen nicht beschrieben werden, es steht aber eine detailliertere Anleitung im Projekt-Wiki zur Verfügung. Der grundsätzliche Ablauf ist wiefolgt:

1. Download, Kompilieren und Installation der `IceStorm`-Toolchain auf dem Anwender-PC (Linux-System)

   ```
   # siehe http://www.clifford.at/icestorm/#install
   ```

2. Download, Kompilieren und Installation der RISC-V Toolchain

   ```
   git clone git@github.com:cliffordwolf/picorv32.git
   cd picorv32
   make download-tools
   make -j\$(nproc) build-tools
   ```

   Das Kompilieren der RISC-V `gcc`-Toolchain kann je nach Rechenleistung mehrere Stunden dauern.

3. Download des Git-Projekts auf dem Anwender-PC (Linux-System)

   ```
   cd ..
   git clone https://github.com/dm7h/icozsoc.git
   ```

4. Einrichten der SSH-Verbindung zum Raspberry Pi Zero W

   ```
   # siehe zum Beispiel:
   ↪  https://www.raspberrypi.org/documentation/remote-
   ↪  access/ssh/passwordless.md
   # Exportieren des SSH-Hosts, z.B. pi@zero oder pi@192.168.1.43:
   export SSH_RASPI=pi@zero
   # eine SSH-Verbindung sollte dann ohne Passwort-Abfrage möglich sein:
   ssh $SSH_RASPI $
   ```

5. Download, Kompilieren und Installation des `icozctl` Git-Projekts
   Zur Installation des `icozctl`-Tools wird auf dem Raspberry Pi folgendes ausgeführt:

```
git clone https://github.com/dm7h/icozctl
cd icozctl
sudo make install
```

6. Generieren und Programmieren des Bitfiles mithilfe des zur Verfügung gestellten `Makefiles`
   Wenn `icozctl` erfolgreich auf dem Raspberry Pi installiert wurde, kann auf dem Anwender-PC das FPGA-Bitstream und die IcoSoc-Anwendung kompiliert und via Raspberry Pi auf die Hardware geflasht werden:

   ```
   cd ../icozsoc/examples/event-recorder/
   make prog_flash
   ```

4.3 Konfiguration der Event-Trigger

Um die Events zu konfigurieren wird zunächst per SSH auf das Raspberry Pi gewechselt. Im Ordner des `icozctl`-Tools befindet sich die Event-Konfigurations-Datei `config.yml`. Für die Aufnahme wird nicht zwangsläufig eine Event-Erkennung benötig, es kann aber aber trotzdem ein Start-, Stopp- und Clock-Event definiert werden um unnötige Aufnahmedaten zu minimieren:

```
# event configuration
events:
  - start:
          trigger: u
          funcion: start

  - stop:
          trigger: d
          function: stop

  - clk_up:
          trigger: 1u

  - clk_down:
          trigger: 1d
```

4.4 Durchführen der Event-Aufnahme

Die Auswertung soll aus Anschauungszwecken das VCD-Dateiformat verwenden. Die Aufnahme kann dann mit folgendem Befehl gestartet werden:

```
icozctl -c config.yml -o midi_clock.vcd
```

Das Programm wartet nun auf Eingangsdaten, die zum Beispiel durch das Starten der in Python implementieren Midi-Clock oder durch ein Musikprogramm geliefert werden können. Die Aufnahme kann zu jedem Zeitpunkt durch die Tastenkombination Strg+c beendet werden. Alternativ kann `icozctl` auch mit der Option “-N” gestartet werden, um die Aufnahme nach einer fesgelegten Anzahl von Samples zu beenden.

```
icozctl -N 512 -c config.yml -o midi_clock.vcd
```

Das Ergebnis der Aufnahme kann dann zum Beispiel mit dem Programm `gtkwave` oder mit `pulseview` grafisch dargestellt und überprüft werden:

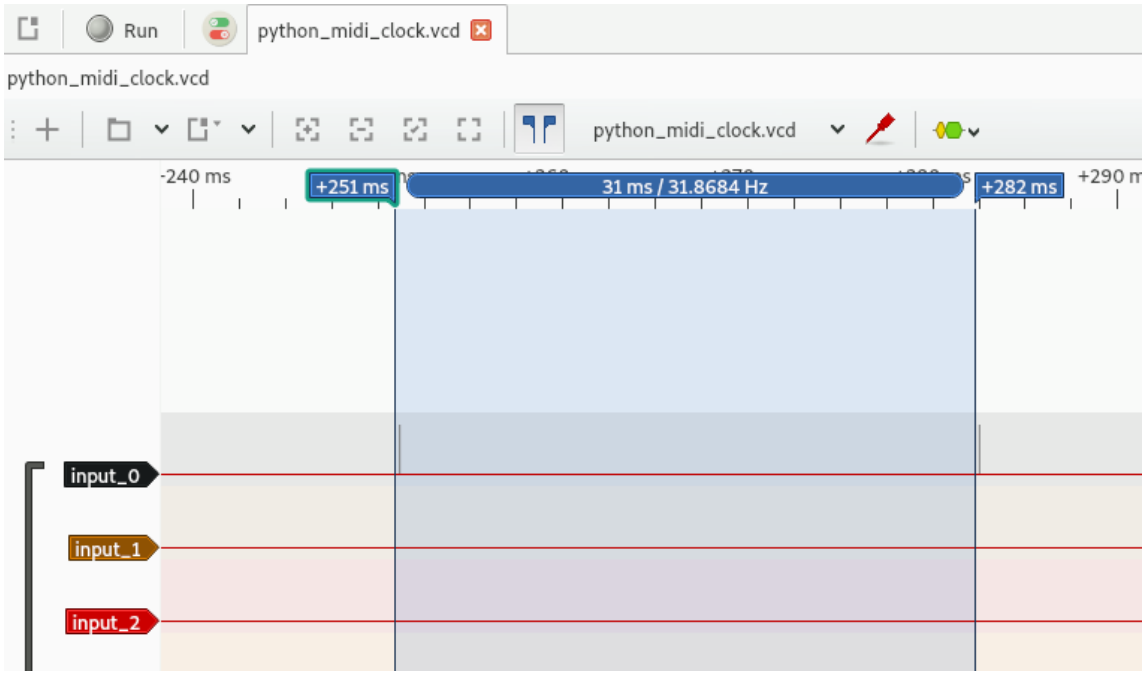


Abbildung 4.1: Darstellung eines aufgenommenen
Midi-Clock-Signals mit Pulseview (Quelle: Pulseview-Screenshot)

4.5 Analyse der Ergebnisse

Für die Analyse der Daten kann zum Beispiel ein Python-Skript verwendet werden. Um Aussagen über den Jitter des Signals machen zu können bietet sich die Generierung eines Histogramms an, bei dem auf der x-Achse die Abweichung zum erwarteten Clock-Signal und auf der y-Achse die Häufigkeit der Abweichung dargestellt wird.

Für das Einlesen der VCD-Datei wird das Python-Modul `Verilog_VCD` verwendet:

```
import Verilog_VCD
vcd = Verilog_VCD.parse_vcd('midi_clock.vcd')
```

Im VCD-Dateiformat wird jedem Signal ein Symbol als Abkürzung zugewiesen, auf die Daten des ersten Signals der Datei kann zum Beispiel mit der Abkürzung “!” zugegriffen werden. Die Zeitstempel sind in der Python-Liste unter dem Index ”tv”(“time value”) abrufbar. Der zeitliche Abstand zum jeweils vorhergehenden Event (Zeit-Delta) kann folgendermaßen berechnet werden:

```
# get the first time value
last = vcd["!"]["tv"][0][0]

# calculate time deltas
deltas = []
for item in vcd["!"]["tv"][1:]:
    deltas.append(item[0]-last)
    last = item[0]
```

Die Daten werden in ein numpy-Array umgwandelt und es wird die Differenz zum erwarteten Zeit-Delta gebildet. Anschließend werden sie in eine pandas-Zeitserie konvertiert, wodurch automatisch statistisch relevante Kennwerte wie der Mittelwert und die Standardabweichung berechnet und angezeigt werden können:

```
np_deltas = np.array(deltas)
expected_delta = 31250000 # expected clock period in nanoseconds
np_deltas -= expected_delta
print(deltas_series.describe())
```

Die Ausgabe sieht zum Beispiel folgendermaßen aus[2]

```
count                        86
mean      0 days 00:00:00.000125
std       0 days 00:00:00.000026
min       0 days 00:00:00.000053
25%       0 days 00:00:00.000109
50%       0 days 00:00:00.000117
75%       0 days 00:00:00.000131
max       0 days 00:00:00.000227
```

Abschließend wird mithilfe der matplotlib-Bibliothek ein Histogramm in Millisekunden-Auflösung erzeugt und angezeigt:

```
(deltas_series/pd.Timedelta(milliseconds = 1)).hist()
[...]
plt.show()
```

4.5.1 Software MIDI-Clock: Python-Implementierung

Für die erste Messung wurde die Python-Implementierung verwendet.

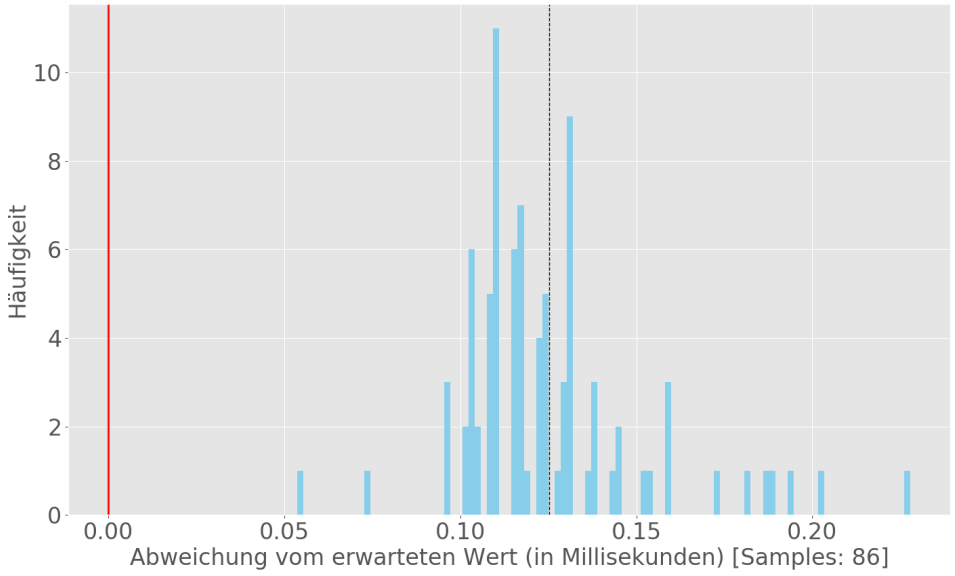


Abbildung 4.2: Histogramm der Python-generierten MIDI-Clock

[2]Die Werte stammen von der oben beschriebenen Python-Implementierung, das Zeit-Delta wird hier in Microsekunden-Auflösung angezeigt

Die erwartete Abweichung (0) ist mit der roten Linie markiert, der Mittlerwert mit der gestrichelten schwarzen Linie.
Auffällig ist hier vor allem dass die MIDI-Clock langsamer läuft als erwartet, da eine konstante positive Abweichung von ca. 0,125 Millisekunden vorliegt. Dies lässt sich dadurch erklären, dass bei jedem Ausführen des `midiout.sendmessage(clock_tick)`-Befehls zusätzlich zur berechneten Clock-Periode Zeit verbraucht wird. Der Jitter liegt bei circe 0,2 Millisekunden und ist damit vergleichsweise niedrig.

4.5.2 Software MIDI-Clock: Renoise und Reaper

Anschließend wurde die MIDI-Clock von den Musikprogrammen Renoise und Reaper aufgenommen[3].

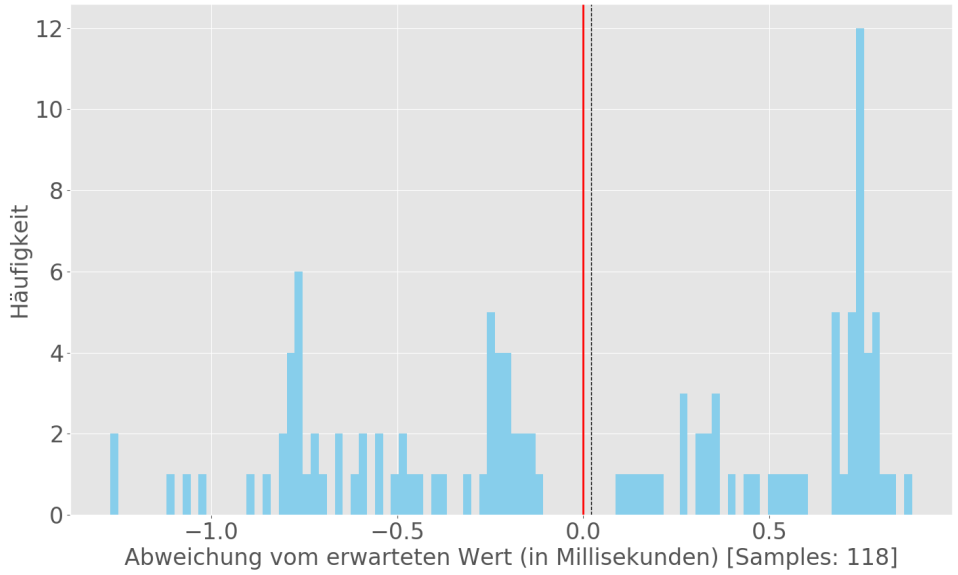


Abbildung 4.3: Histogramm der MIDI-Clock des Musikprogramms Renoise

Bei der MIDI-Clock von Renoise zeigt sich ein Jitter von fast 2 Millisekunden, ein eher schlechter Wert bei dem sogar hörbare Konsequenzen auftreten könnten.

[3]Es wurden (von den Vorlieben des Autors abgesehen) keine besonderen Auswahlkriterien verfolgt, beide Programme haben eine kostenlose Demo-Version mit der die Funktionalität geprüft werden kann

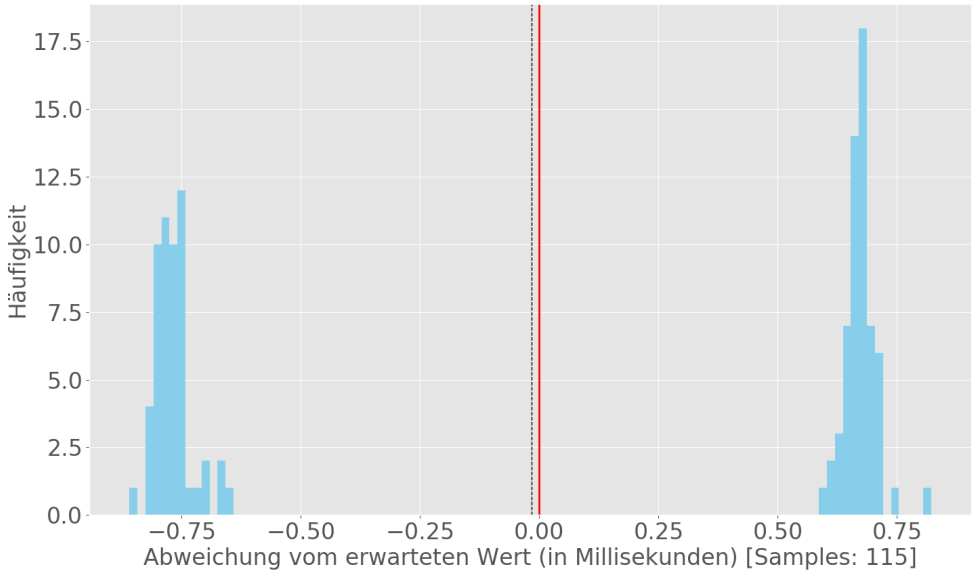


Abbildung 4.4: Histogramm der MIDI-Clock des Musikprogramms Reaper

Bei der MIDI-Clock von Reaper fällt der Jitter mit circa 1,5 Millisekunden etwas geringer aus und zeigt eine starke ausgeprägte bimodale Verteilung. Auch dies ist – inbesondere im Vergleich zur “naiven” Python-Implementierung – ein eher schlechter Wert..

4.5.3 Hardware MIDI-Clock: Midipal

Abschließend wurde noch die auf einem ATMega328p-Mikrocontroller basierende Hardware-MIDI-Clock des “MIDIpal” getestet. Das Clock-Signal wurde über den MIDI-Eingang einer RME Digiface PCI-Soundkarte erfasst und mit einer Software-Loopback auf das virtuelle MIDI-Interface des Teensy LC geroutet.

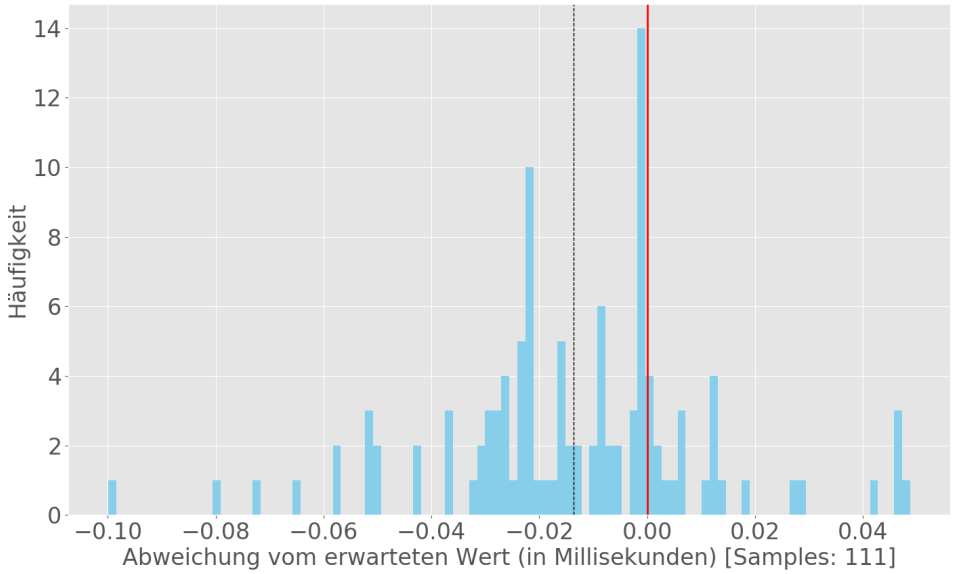


Abbildung 4.5: Histogramm einer Mikrocontroller-generierten Hardware-MIDI-Clock (Mutable Instruments MidiPal)

Die Messung liefert ein deutlich stabileres Ergebnis mit einem Jitter von circa 0,15 Millisekungen. Es zeigt sich, dass die Mikrokontrollerbasierte Lösung den beiden Software-Produkten in Hinblick auf die Stabilität des Clock-Signals überlegen ist.

Zu allen Messungen ist anzumerken, dass der Teensy-Mikrocontroller (der mit einer Taktfrequenz von 40 Mhz läuft) einen nicht näher bestimmten Einfluss auf das Endergebnis hat. Alle Messungen wurden wiederholt mit variierenden Samplezahlen durchgeführt, und die beschriebenen Charakteristika sind ohne nenneswerte Abweichungen reproduzierbar. Dementsprechend kann davon ausgegangen werden, dass die vom Teensy verursachte Verfälschung für die Messungen vernachlässigbar ist.

5. Fazit

Bla fasel...

6. Aussicht

Bla fasel...

Abkürzungen

I²C Inter-Integrated Circuit. 9, 10, 31, *Glossary:* I²C

API Application Programming Interface. 3, 31, *Glossary:* API

CPLD Complex Programmable Logic Device. 2, 8, 31, *Glossary:* CPLD

FPGA Field Programmable Gate Array. 2, 8, 31, *Glossary:* FPGA

GPIO General Purpose Input / Output. 10, 31, *Glossary:* GPIO

MIDI Musical Instrument Digital Interface. 20

PLL Phase-locked loop. 8, 31, *Glossary:* PLL

RAM Random-Access Memory. 9

SPI Serial Peripheral Interface. 9, 10, 31, *Glossary:* SPI

UART Universal Asynchronous Receiver Transmitter. 3, 9, 20, 31, *Glossary:* UART

VHDL Very High Speed Integrated Circuit Hardware Description Language. 31, *Glossary:* VHDL

Glossar

I²C Synchroner, serieller Datenbus nach dem Master-Slave-Prinzip, der Übertragunsraten bis zu 5 Mbit/s ermöglicht (vgl. [11]). 10, 31

API Schnittstelle zur Anwendungsprogrammierung. Erlaubt zum Beispiel die Verwendung von Programmkomponeneten durch eine exteren Skript-Sprache wie Python (vgl. [13]). 31

CPLD Im Vergleich zu FPGAs einfacher aufgebaute programmierbare logische Schaltungen (vgl. [14]). 2, 8, 31

FPGA Ein rekonfigurierbarer Chip dessen Schaltungsstruktur in einer Hardwarebeschreibungssprache (wie VHDL oder Verilog) frei programmierbar ist (vgl. [15]). 2, 8, 31

GPIO Frei programmierbarer Kontakt der zum Beispiel für das Ansprechen externer Hardwarekomponenten verwendet werden kann (vgl. [16]). 31

PLL Elektronische Schaltungsanordnung die die Frequenz eines veränderbaren Oszillators beeinflussen kann. Wird bei FPGAs meist zur Erzeugung von Taktsignalen mit einstellbarer Geschwindkeit verwendet (vgl. [17]). 31

SPI Synchroner, serieller Datenbus für Datenübertragungen nach dem Master-Slave-Prinzip mit dem vergleichsweise hohe Datendurchsätze möglich sind (vgl. [12]). 10, 31

UART Schnittstelle zur asynchronen, seriellen Datenübertragung (vgl. [10]). 9, 31

Verilog Wortkreuzung aus "Verification" und "Logic". Hardwarebeschreibungssparche für Programmierung und Simulation von FPGAs und CPLDs die in den USA geläufiger ist als VHDL (vgl. [18], siehe auch [19] zur Namesnherkunft). 3

VHDL Vor allem in Europa verbeitetete Hardwarebeschreibungssprache für die Simulation und Programmierung von FPGAs und CPLDs (vgl. [20]). 31

Abbildungsverzeichnis

2.1 Blockdiagramm des verwendeten iCE40 FPGAs 8

4.1 Darstellung eines aufgenommenen Midi-Clock-Signals mit Pulseview 24
4.2 Histogramm der Python-generierten MIDI-Clock 25
4.3 Histogramm der MIDI-Clock des Musikprogramms Renoise 26
4.4 Histogramm der MIDI-Clock des Musikprogramms Reaper 27
4.5 Histogramm einer Mikrocontroller-generierten Hardware-MIDI-Clock 28

A.1 Pinbelegung der PMOD-Header des IceZero-Boards 37

Tabellenverzeichnis

2.1 Beispielhafte Darstellung eines aufgenommenen Events mit 16-Bit Zeitstempel 6

A.1 Pinbelegung der PMOD-Header des Icezero-Boards 37
A.2 Pinbelegung 38
A.3 Überblick über den Inhalt des Git-Repositories 39

Literatur

[1] *Echtzeit – Wikipedia*, [Online; accessed 1. Jun. 2018], Mai 2018. Adresse: https://de.wikipedia.org/wiki/Echtzeit.

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

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

[4] *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.

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

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

[7] *Digitalsignal – Wikipedia*, [Online; accessed 4. Jun. 2018], Mai 2018. Adresse: https://de.wikipedia.org/wiki/Digitalsignal.

[8] *Programmierbare logische Schaltung – Wikipedia*, [Online; accessed 6. Jun. 2018], Mai 2018. Adresse: https://de.wikipedia.org/wiki/Programmierbare_logische_Schaltung.

[9] *iCE40 LP/HX Family Data Sheet*, Lattice Semiconductor Corp., 2017.

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

[11] *I²C – Wikipedia*, [Online; accessed 7. Jun. 2018], Juni 2018. Adresse: https://de.wikipedia.org/wiki/I%C2%B2C.

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

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

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

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

[16] *Allzweckeingabe/-ausgabe – Wikipedia*, [Online; accessed 7. Jun. 2018], Mai 2018. Adresse: https://de.wikipedia.org/wiki/Allzweckeingabe/-ausgabe.

[17] *Phasenregelschleife – Wikipedia*, [Online; accessed 6. Jun. 2018], Juni 2018. Adresse: https://de.wikipedia.org/wiki/Phasenregelschleife.

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

[19] 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.

[20] *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. Material

A.1 PMOD-Pinbelegung des IceZero-Boards

Bei der Pin-Belegung wurde das selbe Schema wie beim GPIO-Modul des IcoSoc verwendet. Die Numerierung läuft dementsprechend von rechts nach links und für die einzelnen PMOD-Header "zeilenweise" von oben nach unten.

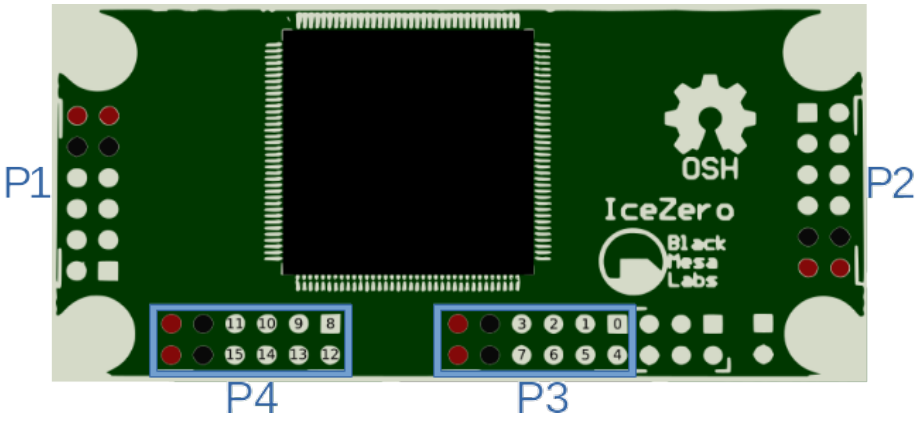


Abbildung A.1: Pinbelegung der PMOD-Header des IceZero-Boards (Quelle: Eigene Abbildung)

3.3V	GND	pin_11	pin_10	pin_9	pin_8
3.3V	GND	pin_15	pin_14	pin_13	pin_12

PMOD - P4

3.3V	GND	pin_3	pin_2	pin_1	pin_0
3.3V	GND	pin_7	pin_6	pin_5	pin_4

PMOD - P3

Tabelle A.1: Pinbelegung der PMOD-Header des Icezero-Boards

A.2 Pinverbindungen Raspberry Pi und FPGA-Shield

Tabelle A.2: Pinbelegung

ice40	WiringP	Name	Physical		Name	WiringPi	ice40
		3.3V	1	2	5V		
	8	SDA.1	3	4	5V		
	9	SCL.1	5	6	GND		
	7	1-Wire	7	8	TxD	15	
		GND	9	10	RxD	16	
	0	GPIO. 0	11	12	GPIO.1	1	
	2	GPIO. 2	13	14	GND		
	3	GPIO. 3	15	16	GPIO. 4	4	
		3.3V	17	18	GPIO. 5	5	
	12	MOSI	19	20	GND		
	13	MISO	21	22	GPIO. 6	6	
	14	SCLK	23	24	CE0	10	
		GND	25	26	CE1	11	
	30	SDA.0	27	28	SCL.0	31	
	21	GPIO.21	29	30	GND		
	22	GPIO.22	31	32	GPIO.26	26	
	23	GPIO.23	33	34	GND		
	24	GPIO.24	35	36	GPIO.27	27	
	25	GPIO.25	37	38	GPIO.28	28	
		GND	39	40	GPIO.29	29	

A.3 CD

Die beiliegende CD enthält den Inhalt des Github-Repositories

`https://github.com/dm7h/fpga-event-recorder`

zum Zeitpunkt der Abgabe.

Folgende Tabelle entält einen Überblick über die Inhalte des Repositories:

Verz.	Unterverzeichnis	Inhaltsbeschreibung
doc/	ref/	iCE40 Manuals und relevante Hardwaredokumentation
thesis/		Finale Version der Bachelorarbeit als PDF
thesis/	src/	Latex-Sourcen der Bachelorarbeit
src/		
	Logikanalysator/	Dateien des Semesterprojekts "Logikanalysator mit AVR Mega32U4 und Altera MAX CPLD" inkl. Dateien der Bachelorarbeit von Andreas Müller (USB-TPLE)
	icotools/	Portierung des icotools Projekts für das IceZero-Board. Original-Repository: https://github.com/cliffordwolf/icotools. Relevant sind hier vor allem die Unterverzeichnisse "icosoc" und "icoprog"
	icozctl/	Fork des icoprog-Tools (icotools/icoporg) mit zusätzlichen Funktionen zur Steuerung des Event-Recorders
	icozero/	Vom IcoSoc unabhängiges Beispiel-Projekt für das IceZero-Board mit leichten Modifikationen (nicht projekt-relevant)
	picosoc/	Portierung des PicoSoc-Projekts auf das IceZero-Board (nicht direkt projekt-relevant). PicoSoc ist eine minimale Variante des IcoSocs. Original-Repository: https://github.com/cliffordwolf/picorv32/tree/master/picosoc
	sump2/	SUMP2 Variante für das IceZero-Board (nicht für die Icestorm-Toolchain)
	sump2_pipistrello_ftdi_fifo/	SUMP2 Vairante für das Pipistrello-Board bei dem der Versuch unternommen wurde den UART durch einen FTDI-Fifo zu ersetzen, um einen Höheren Datenduchsatz zu ermöglichen. Quelle: http://forum.gadgetfactory.net/topic/1748-open-bench-logic-sniffer-with-64mb-capture-buffer/
	demon-core-import/	SUMP2 Weiterenwicklung für den Open Bench Logic Sniffer. Quelle: https://github.com/jhol/demon-core-import

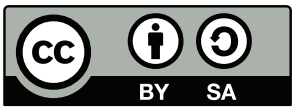
Tabelle A.3: Überblick über den Inhalt des Git-Repositories

B. Lizenz

Alle im Rahmen dieser Arbeit erstellten Text-Dokumente stehen unter der folgenden Creative Commons BY-SA 4.0 Lizenz.

B.1 Creative Commons Attribution-ShareAlike 4.0 International

Dieses Werk ist lizenziert unter einer Creative Commons „Namensnennung – Weitergabe unter gleichen Bedingungen 3.0 Deutschland“ Lizenz.



B.2 Creative Commons Legal Code

B.2.1 Attribution-ShareAlike 3.0 Unported

> CREATIVE COMMONS CORPORATION IS NOT A LAW FIRM AND DOES NOT PROVIDE LEGAL SERVICES. DISTRIBUTION OF THIS LICENSE DOES NOT CREATE AN ATTORNEY-CLIENT RELATIONSHIP. CREATIVE COMMONS PROVIDES THIS INFORMATION ON AN “AS-IS” BASIS. CREATIVE COMMONS MAKES NO WARRANTIES REGARDING THE INFORMATION PROVIDED, AND DISCLAIMS LIABILITY FOR DAMAGES RESULTING FROM ITS USE.

B.2.1.1 *License*

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE (“CCPL” OR “LICENSE”). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

(a) **"Adaptation"** means a work based upon the Work, or upon the Work and other pre-existing works, such as a translation, adaptation, derivative work, arrangement of music or other alterations of a literary or artistic work, or phonogram or performance and includes cinematographic adaptations or any other form in which the Work may be recast, transformed, or adapted including in any form recognizably derived from the original, except that a work that constitutes a Collection will not be considered an Adaptation for the purpose of this License. For the avoidance of doubt, where the Work is a musical work, performance or phonogram, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered an Adaptation for the purpose of this License.

(b) **"Collection"** means a collection of literary or artistic works, such as encyclopedias and anthologies, or performances, phonograms or broadcasts, or other works or subject matter other than works listed in Section 1(f) below, which, by reason of the selection and arrangement of their contents, constitute intellectual creations, in which the Work is included in its entirety in unmodified form along with one or more other contributions, each constituting separate and independent works in themselves, which together are assembled into a collective whole. A work that constitutes a Collection will not be considered an Adaptation (as defined below) for the purposes of this License.

(c) **"Creative Commons Compatible License"** means a license that is listed at https://creativecommons.org/compatiblelicenses that has been approved by Creative Commons as being essentially equivalent to this License, including, at a minimum, because that license: (i) contains terms that have the same purpose, meaning and effect as the License Elements of this License; and, (ii) explicitly permits the relicensing of adaptations of works made available under that license under this License or a Creative Commons jurisdiction license with the same License Elements as this License.

(d) **"Distribute"** means to make available to the public the original and copies of the Work or Adaptation, as appropriate, through sale or other transfer of ownership.

(e) **"License Elements"** means the following high-level license attributes as selected by Licensor and indicated in the title of this License: Attribution, ShareAlike.

(f) **"Licensor"** means the individual, individuals, entity or entities that offer(s) the Work under the terms of this License.

(g) **"Original Author"** means, in the case of a literary or artistic work, the individual, individuals, entity or entities who created the Work or if no individual or entity can be identified, the publisher; and in addition (i) in the case of a performance the actors, singers, musicians, dancers, and other persons who act, sing, deliver, declaim, play in, interpret or otherwise perform literary or artistic works or expressions of folklore; (ii) in the case of a phonogram the producer being the person or legal entity who first fixes the sounds of a performance or other sounds; and, (iii) in the case of broadcasts, the organization that transmits the broadcast.

(h) **"Work"** means the literary and/or artistic work offered under the terms of this License including without limitation any production in the literary, scientific and artistic domain, whatever may be the mode or form of its expression including digital form, such as a book, pamphlet and other writing; a lecture, address, sermon or other work of the same nature; a dramatic or dramatico-musical work; a choreographic work or entertainment in dumb show; a musical composition with or without words; a cinematographic work to which are assimilated works expressed by a process analogous

to cinematography; a work of drawing, painting, architecture, sculpture, engraving or lithography; a photographic work to which are assimilated works expressed by a process analogous to photography; a work of applied art; an illustration, map, plan, sketch or three-dimensional work relative to geography, topography, architecture or science; a performance; a broadcast; a phonogram; a compilation of data to the extent it is protected as a copyrightable work; or a work performed by a variety or circus performer to the extent it is not otherwise considered a literary or artistic work.

(i) **"You"** means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

(j) **"Publicly Perform"** means to perform public recitations of the Work and to communicate to the public those public recitations, by any means or process, including by wire or wireless means or public digital performances; to make available to the public Works in such a way that members of the public may access these Works from a place and at a place individually chosen by them; to perform the Work to the public by any means or process and the communication to the public of the performances of the Work, including by public digital performance; to broadcast and rebroadcast the Work by any means including signs, sounds or images.

(k) **"Reproduce"** means to make copies of the Work by any means including without limitation by sound or visual recordings and the right of fixation and reproducing fixations of the Work, including storage of a protected performance or phonogram in digital form or other electronic medium.

2. Fair Dealing Rights. Nothing in this License is intended to reduce, limit, or restrict any uses free from copyright or rights arising from limitations or exceptions that are provided for in connection with the copyright protection under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

(a) to Reproduce the Work, to incorporate the Work into one or more Collections, and to Reproduce the Work as incorporated in the Collections;

(b) to create and Reproduce Adaptations provided that any such Adaptation, including any translation in any medium, takes reasonable steps to clearly label, demarcate or otherwise identify that changes were made to the original Work. For example, a translation could be marked "The original work was translated from English to Spanish," or a modification could indicate "The original work has been modified.";

(c) to Distribute and Publicly Perform the Work including as incorporated in Collections; and,

(d) to Distribute and Publicly Perform Adaptations.

(e) For the avoidance of doubt:

i. **Non-waivable Compulsory License Schemes**. In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme cannot be waived, the Licensor reserves the exclusive right to

collect such royalties for any exercise by You of the rights granted under this License;

ii. **Waivable Compulsory License Schemes**. In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme can be waived, the Licensor waives the exclusive right to collect such royalties for any exercise by You of the rights granted under this License; and,

iii. **Voluntary License Schemes**. The Licensor waives the right to collect royalties, whether individually or, in the event that the Licensor is a member of a collecting society that administers voluntary licensing schemes, via that society, from any exercise by You of the rights granted under this License.

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. Subject to Section 8(f), all rights not expressly granted by Licensor are hereby reserved.

4. Restrictions. The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

(a) You may Distribute or Publicly Perform the Work only under the terms of this License. You must include a copy of, or the Uniform Resource Identifier (URI) for, this License with every copy of the Work You Distribute or Publicly Perform. You may not offer or impose any terms on the Work that restrict the terms of this License or the ability of the recipient of the Work to exercise the rights granted to that recipient under the terms of the License. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties with every copy of the Work You Distribute or Publicly Perform. When You Distribute or Publicly Perform the Work, You may not impose any effective technological measures on the Work that restrict the ability of a recipient of the Work from You to exercise the rights granted to that recipient under the terms of the License. This Section 4(a) applies to the Work as incorporated in a Collection, but this does not require the Collection apart from the Work itself to be made subject to the terms of this License. If You create a Collection, upon notice from any Licensor You must, to the extent practicable, remove from the Collection any credit as required by Section 4(c), as requested. If You create an Adaptation, upon notice from any Licensor You must, to the extent practicable, remove from the Adaptation any credit as required by Section 4(c), as requested.

(b) You may Distribute or Publicly Perform an Adaptation only under the terms of: (i) this License; (ii) a later version of this License with the same License Elements as this License; (iii) a Creative Commons jurisdiction license (either this or a later license version) that contains the same License Elements as this License (e.g., Attribution-ShareAlike 3.0 US)); (iv) a Creative Commons Compatible License. If you license the Adaptation under one of the licenses mentioned in (iv), you must comply with the terms of that license. If you license the Adaptation under the terms of any of the licenses mentioned in (i), (ii) or (iii) (the "Applicable License"), you must comply with the terms of the Applicable License generally and the following provisions: (I) You must include a copy of, or the URI for, the Applicable License with every copy of each Adaptation You Distribute or Publicly Perform; (II) You may not offer or impose any terms on the Adaptation that restrict the terms of the Applicable License or the ability of the recipient of the Adaptation to exercise the rights granted to that recipient under the terms of the Applicable License; (III) You must keep intact all

notices that refer to the Applicable License and to the disclaimer of warranties with every copy of the Work as included in the Adaptation You Distribute or Publicly Perform; (IV) when You Distribute or Publicly Perform the Adaptation, You may not impose any effective technological measures on the Adaptation that restrict the ability of a recipient of the Adaptation from You to exercise the rights granted to that recipient under the terms of the Applicable License. This Section 4(b) applies to the Adaptation as incorporated in a Collection, but this does not require the Collection apart from the Adaptation itself to be made subject to the terms of the Applicable License.

(c) If You Distribute, or Publicly Perform the Work or any Adaptations or Collections, You must, unless a request has been made pursuant to Section 4(a), keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or if the Original Author and/or Licensor designate another party or parties (e.g., a sponsor institute, publishing entity, journal) for attribution ("Attribution Parties") in Licensor's copyright notice, terms of service or by other reasonable means, the name of such party or parties; (ii) the title of the Work if supplied; (iii) to the extent reasonably practicable, the URI, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and (iv) , consistent with Ssection 3(b), in the case of an Adaptation, a credit identifying the use of the Work in the Adaptation (e.g., "French translation of the Work by Original Author," or "Screenplay based on original Work by Original Author"). The credit required by this Section 4(c) may be implemented in any reasonable manner; provided, however, that in the case of a Adaptation or Collection, at a minimum such credit will appear, if a credit for all contributing authors of the Adaptation or Collection appears, then as part of these credits and in a manner at least as prominent as the credits for the other contributing authors. For the avoidance of doubt, You may only use the credit required by this Section for the purpose of attribution in the manner set out above and, by exercising Your rights under this License, You may not implicitly or explicitly assert or imply any connection with, sponsorship or endorsement by the Original Author, Licensor and/or Attribution Parties, as appropriate, of You or Your use of the Work, without the separate, express prior written permission of the Original Author, Licensor and/or Attribution Parties.

(d) Except as otherwise agreed in writing by the Licensor or as may be otherwise permitted by applicable law, if You Reproduce, Distribute or Publicly Perform the Work either by itself or as part of any Adaptations or Collections, You must not distort, mutilate, modify or take other derogatory action in relation to the Work which would be prejudicial to the Original Author's honor or reputation. Licensor agrees that in those jurisdictions (e.g. Japan), in which any exercise of the right granted in Section 3(b) of this License (the right to make Adaptations) would be deemed to be a distortion, mutilation, modification or other derogatory action prejudicial to the Original Author's honor and reputation, the Licensor will waive or not assert, as appropriate, this Section, to the fullest extent permitted by the applicable national law, to enable You to reasonably exercise Your right under Section 3(b) of this License (right to make Adaptations) but not otherwise.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR

WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

(a) This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Adaptations or Collections from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

(b) Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

(a) Each time You Distribute or Publicly Perform the Work or a Collection, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.

(b) Each time You Distribute or Publicly Perform an Adaptation, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.

(c) If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

(d) No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.

(e) This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with

respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

(f) The rights granted under, and the subject matter referenced, in this License were drafted utilizing the terminology of the Berne Convention for the Protection of Literary and Artistic Works (as amended on September 28, 1979), the Rome Convention of 1961, the WIPO Copyright Treaty of 1996, the WIPO Performances and Phonograms Treaty of 1996 and the Universal Copyright Convention (as revised on July 24, 1971). These rights and subject matter take effect in the relevant jurisdiction in which the License terms are sought to be enforced according to the corresponding provisions of the implementation of those treaty provisions in the applicable national law. If the standard suite of rights granted under applicable copyright law includes additional rights not granted under this License, such additional rights are deemed to be included in the License; this License is not intended to restrict the license of any rights under applicable law.

B.2.1.2 Creative Commons Notice

Creative Commons is not a party to this License, and makes no warranty whatsoever in connection with the Work. Creative Commons will not be liable to You or any party on any legal theory for any damages whatsoever, including without limitation any general, special, incidental or consequential damages arising in connection to this license. Notwithstanding the foregoing two (2) sentences, if Creative Commons has expressly identified itself as the Licensor hereunder, it shall have all rights and obligations of Licensor.

Except for the limited purpose of indicating to the public that the Work is licensed under the CCPL, Creative Commons does not authorize the use by either party of the trademark "Creative Commons" or any related trademark or logo of Creative Commons without the prior written consent of Creative Commons. Any permitted use will be in compliance with Creative Commons' then-current trademark usage guidelines, as may be published on its website or otherwise made available upon request from time to time. For the avoidance of doubt, this trademark restriction does not form part of the License.

Creative Commons may be contacted at https://creativecommons.org/.

B.3 ISC-Lizenz

Alle im Rahmen dieser Arbeit erstellten oder modifizierten Quelltext-Dokumente stehen soweit nicht anders vermerkt unter der nachfolgenden ISC-Lizenz.