Praktikum Technische Informatik: Eingebettete Systeme

TECHNISCHE UNIVERSITÄT DARMSTADT

Prof. Dr. Andreas Koch
Dipl.-Inform. Andreas Engel

Sommersemester 2014

(Stand 12. April 2014)

I Einleitung

Ziel dieser Lehrveranstaltung ist die Vermittlung praktischer Kenntnisse beim Entwurf und dem Einsatz eingebetteter Systeme. Dabei ist das in Abbildung 1 dargestellte Messsystem zu realisieren, welches einen Analog-/Digital-Wandler (analog to digital converter, ADC) mit einer dynamischen Vorverstärkung (programmable gain amplifier, PGA) kombiniert.

Der ADC ist im Wesentlichen als Hardwaremodul innerhalb eines FPGAs realisiert und basiert darauf, eine vorverstärkte Eingangsspannung V_{scaled} mit der Spannung $V_{compare}$ über einem Kondensator C_M zu vergleichen. Für diesen Vergleich wird ein differentieller FPGA-Eingang verwendet. Je nach Polarität des `COMPARE`-Eingangs muss das ADC_CONTROL-Modul den Kondensator weiter auf- oder entladen. Dies geschieht durch das Anlegen eines entsprechenden Wertes am `CHARGE`-Ausgang. Aus der Zahl der Auf- und Entladezyklen leitet ADC_CONTROL den aktuellen Wert von $V_{compare}$ ab und gibt dessen digitale Repräsentation als `SCALED`-Signal aus, sobald der Abgleich mit V_{scaled} erfolgte [1].

Die dynamische Vorverstärkung dient dazu, die Genauigkeit des Messsystems bei schwachen Eingangssignalen zu erhöhen. Dafür konfiguriert das PGA_CONTROL-Modul den PGA über eine digitale Schnittstelle entsprechend des vom Benutzer vorgegebenen Skalierungsfaktors `GAIN`. Umgekehrt berechnet PGA_CONTROL die digitale Repräsentation `UNSCALED` des unskalierten V_{in} aus dem aktuellen Skalierungsfaktor und `SCALED`.

Zur Realisierung dieses Messsystems wird ein Erweiterungsboard für ein Microsemi Igloo FPGA [2, 3] gefertigt und die Hardwaremodule ADC_CONTROL und PGA_CONTROL in einer Hardwarebeschreibungssprache entworfen.

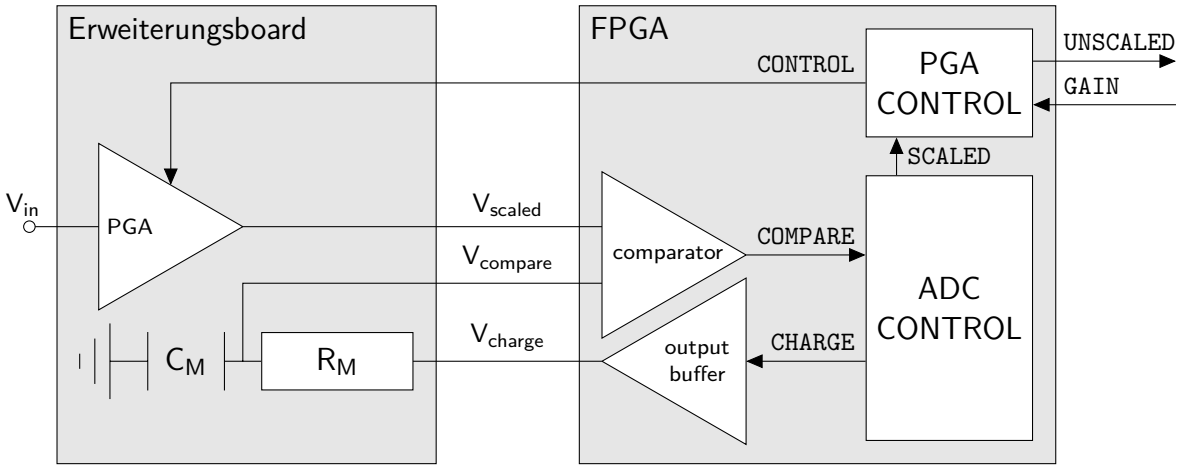

Abbildung 1: Messsystem bestehend aus PGA und ADC

II Organisation

Anmeldung

Die Anmeldung zum Praktikum als Prüfungsleistung erfolgt über TUCaN [4]. Wegen der begrenzten Ressourcen unseres Elektroniklabors ist die Teilnehmerzahl auf 20 Personen limitiert. Bei zu großer Nachfrage muss daher die Reihenfolge der Anmeldung über die Teilnahme entscheiden.

Kommunikation

Für die Diskussion eventueller Unklarheiten steht Ihnen ein Moodle-Forum [5] zur Verfügung, für welches Sie automatisch angemeldet werden. Nutzen Sie dieses als erste Anlaufstelle und posten Sie hier noch nicht beantwortete Fragen. Dadurch soll allen Praktikumsteilnehmer der gleiche Kenntnisstand zugänglich sein. Es wird empfohlen, das Aufgaben- und Nachrichtenforum des Moodle-Kurses zu abonnieren. Daneben wird eine wöchentliche Sprechstunde durch den Praktikumsbetreuer angeboten.

Ablauf und Bewertung

KW 17	Einführungsveranstaltung und Elektrotechnik-Grundlagen
KW 24 (13.06.2014)	Abgabe von Aufgabenblock 1
KW 32 (08.08.2014)	Abgabe von Aufgabenblock 2
KW 33+34	Durchführung der Kolloquien

Die Einführungsveranstaltung findet nach Absprache mit den angemeldeten Praktikumsteilnehmern in der ersten Vorlesungswoche statt und dient der Vorstellung der Aufgaben und der Einweisung in das Elektroniklabor. Dabei werden die Teilnehmer außerdem in Zweiergruppen eingeteilt. Daneben wird eine Einführung in die wichtigsten Elektrotechnischen Grundkenntnisse als freiwillige Zusatzveranstalltung angeboten.

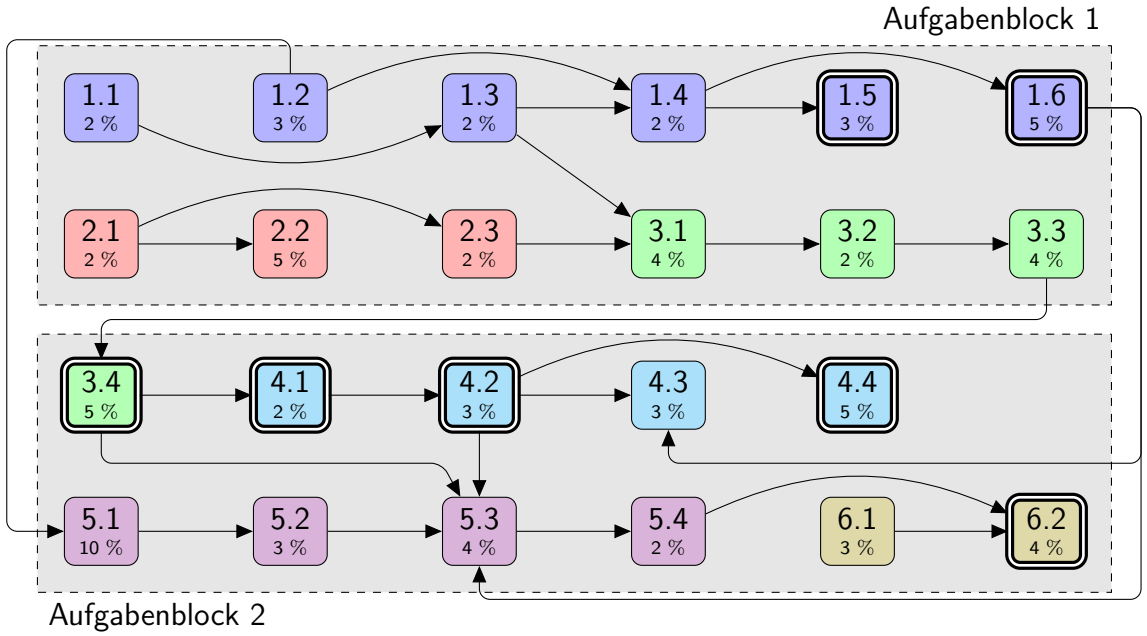

Abbildung 2: Abhängigkeiten und Wertung der Teilaufgaben, Arbeitsschritte im Labor

Die Aufgaben bauen teilweise aufeinander auf und die Bearbeitung erfolgt in zwei Blöcken. Abbildung 2 soll Sie bei Ihrer Zeit- und Gruppeneinteilung unterstützen. Sie zeigt die Bewertungsgewichtung der Teilaufgaben, die Abhängigkeiten zwischen den Teilaufgaben und die Notwendigkeit zur Ausführung von Teilaufgaben im Elektroniklabor. Zwischen der Abgabe des ersten Aufgabenblocks und der Bearbeitung von Aufgabe 3.4 können ein paar Tage für die Bestellung von Bauteilen vergehen. Sie können in dieser Zeit aber mit der Bearbeitung von Aufgabe 5 beginnen.

80 % der Bewertungspunkte können durch das Lösen der Teilaufgaben erzielt werden. Die verbleibenden 20 % werden im abschließenden Kolloquium vergeben. Die Teilnahme am Kolloquium ist *verpflichtend*.

Das Lösen der Teilaufgaben umfasst neben dem Programmieren von Hardware-Modulen auch das schriftliche Beantworten von Sachfragen und die Dokumentation von Messergebnissen. Für den schriftlichen Teil der beiden Aufgabenblöcke sind die Dateien `Block1.pdf` und `Block2.pdf` zu erstellen. Für alle anderen (Programmier-) Aufgaben wird eine Projektstruktur auf der Praktikums-Webseite vorgegeben. Diese im nächsten Abschnitt beschriebene Projektstruktur enthält alle notwendigen Quelldateien, die von Ihnen entsprechend der Aufgabenstellungen ergänzt werden sollen. **Eigener Quellcode ist zu kommentieren.** Für die Abgabe eines Aufgabenblocks senden Sie die ergänzte Projektstruktur (ohne die temporären Zwischenergebnisse, vgl. Abbildung 3) zusammen mit dem schriftlichen Teil per Email an den Praktikumsbetreuer. Der in Aufgabe 3.4 anzufertigende Prototyp wird im abschließenden Kolloquium begutachtet.

Arbeitsplätze

Im Elektroniklabor (S2|02 B014) des Fachgebiets sind zwei Arbeitsplätze mit folgender Ausstattung eingerichtet:

- Hameg HMOx524 Mixed-Signal Oszilloskop [6]
- Agilent 33522A Funktionsgenerator [7]
- Microsemi M1AGL1000 FPGA Evaluationsboard [8, 9]
- Lötstation mit Zubehör

Alle Geräte sind ausschließlich innerhalb ihrer Spezifikation zu betreiben. Im Zweifelsfall ist mit dem Praktikumsbetreuer Rücksprache zu halten. Die Ausgänge des Funktionsgenerators sind elektronisch auf 0 V bis 5 V begrenzt. Diese Begrenzung darf *nicht* deaktiviert werden.

BNC-Kabel zum Anschluss der Messgeräte befinden sich hinter dem Einzelteiledrehturm. BNC-Kupplungen zum Verzweigen sowie Klemmen zum Befestigen der Verkabelung befinden sich im Einzelteiledrehturm. Vor Verlassen des Arbeitsplatzes werden alle Versuchsanordnungen abgebaut und alle Geräte, Werkzeuge und Kabel an die dafür vorgesehenen Positionen zurück verbracht. Nicht mehr verwendbares Verbrauchsmaterial ist zu entsorgen. *Das Verzehren von Speisen und Getränken im Labor ist untersagt.*

Plagiarismus

Der Fachbereich Informatik misst der Einhaltung der Grundregeln der wissenschaftlichen Ethik großen Wert bei. Zu diesen gehört auch die strikte Verfolgung von Plagiarismus [10].

III Werkzeugfluss und Projektstruktur

Zur Entwicklung der Hardwaremodule für das FPGA wird Libero SoC v11.3 [11] von Microsemi verwendet. Grundsätzlich werden Windows und Linux-Versionen der Entwicklungsumgebung angeboten. Das zum Programmieren des FPGAs notwendige FlashPro-Tool ist allerdings nur in der Windows-Variante enthalten und funktioniert auch nicht aus einer Virtuellen Maschine heraus. Diese Praktikumsanleitung bezieht sich daher auf eine Windows-Installation der notwendigen Software. Die Verwendung alternativer Betriebssysteme ist nicht grundsätzlich untersagt, aber mit entsprechendem Mehraufwand zur Inbetriebnahme der Software verbunden.

Microsemi vergibt freie Lizenzen für registrierte Benutzer, die ein Jahr gültig sind und danach verlängert werden können. Die freie Version von Libero SoC unterstützt VHDL und Verilog und kann auch Bitstreams aus gemischten Quellen erzeugen. Die RTL-Simulation gemischt-sprachiger Quellen wird in der freien Version allerdings nicht unterstützt. Daher werden alle vorgegebenen Module und Tests in beiden Sprachen bereit gestellt.

Bei der Installation von Libero SoC ist darauf zu achten, die zugehörigen Simulations- und Synthesewerkzeuge mit zu installieren (keine standalone-Installation). Außerdem müssen die IGLOO-Bibliotheken von Modelsim bei der Installation mit ausgewählt werden.

Libero SoC verwaltet verschiedene HDL-Quellen (manuell geschrieben, graphisch editiert oder aus einer Modul-Bibliothek generiert) und erzeugt Konfigurationsskripte zum Aufruf der verschiedenen Programme des HDL-Werkzeugflusses (Simulation, Synthese, Layout, Bitstream-Generierung und -Programmierung). Dadurch können nach jeder Änderung an den HDL-Quellen alle Schritte bis zum Programmieren des FPGAs automatisiert ablaufen. Diese Automatisierung verhindert aber einen tieferen Einblick in den Werkzeugfluss und die Funktionsweise der einzelnen Werkzeuge, welche in diesem Praktikum vermittelt werden soll. Außerdem kann eine Fehlkonfiguration in der Libero-Umgebung schnell dazu führen, das nicht mehr die gewünschten Quelldateien in den Werkzeugfluss eingebunden werden.

Für dieses Praktikum werden daher alle Quelldateien und Konfigurationsskripte fest vorgegeben. Abbildung 3 zeigt die verwendete Projektstruktur und die Reihenfolge, in der die verschiedenen Werkzeuge aufgerufen werden müssen, um die implementierten Hardware-Module zu testen und auf das FPGA zu programmieren. Die einzelnen Schritte werden im Folgenden näher beschrieben. Dafür sei

- `<hdl>` ∈ {`vhdl,vlog`} die von Ihnen gewählte Hardwarebeschreibungssprache
- `<project>` ∈ {`comparator,system`} das zu realisierende Projekt aus Aufgabe 1 bzw. Aufgabe 5
- `<level>` ∈ {`presynth,postsynth,postlayout`} der Detailgrad RTL-Simulation
- `<LIBERO>` das Installationsverzeichnis von Libero SoC
- `<PrakTIES>` das Projektverzeichnis des Praktikums

① Pre-Synthese-Simulation [ModelSim, `<level> = presynth`]

Um das funktionale Verhalten Ihrer implementierten Hardware zu testen, starten Sie den RTL-Simulator *ModelSim* (`<LIBERO>/Model/win32acoem/modelsim.exe`). Falls noch nicht sichtbar,

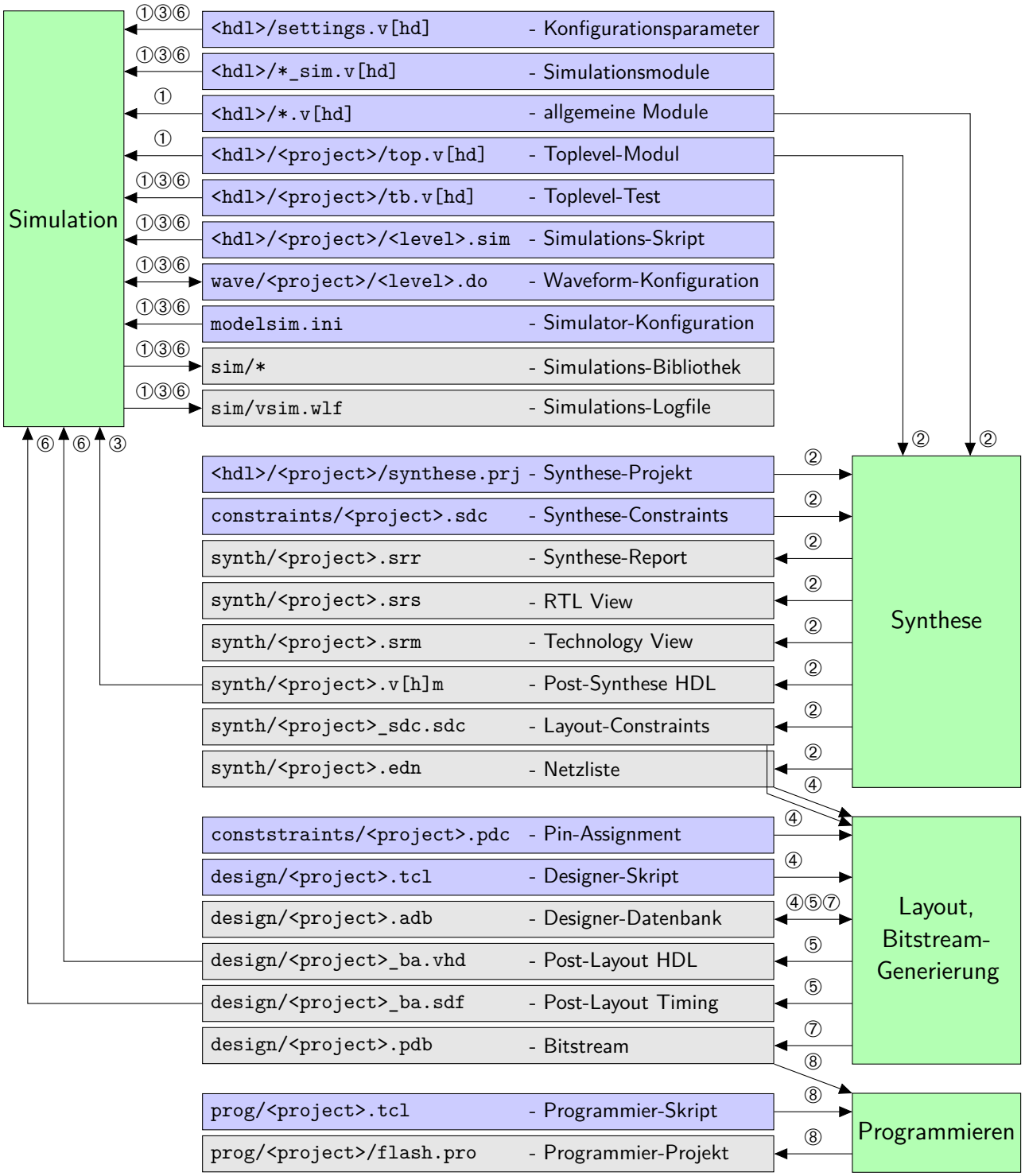

Abbildung 3: HDL-Werkzeugfluss: Die Libero-Werkzeuge generieren aus vorgegebenen Quellen (temporäre) Ergebnisse

öffnen Sie die Eingabeaufforderung (View/Transcript) und wechseln Sie darin in das Praktikumsverzeichnis (`cd <PrakTIES>`).

Rufen Sie nun das Simulations-Skript auf (`do <hdl>/<project>/<level>.sim`). Daraufhin compiliert *ModelSim* alle notwendigen Module in die Simulations-Bibliothek. Werden dabei keine syntaktischen Fehler entdeckt, dann wird die Simulation ausgeführt. Das Test-Modul erzeugt dabei die Eingabe-Signale für das Toplevel-Modul und überprüft die Korrektheit der Ausgabe-Signale. Abweichungen vom erwarteten Verhalten werden dabei als `"ERROR @ <simulationtime> : <message>."` angezeigt. Das Durchlaufen des Tests ohne eine solche Fehlermeldung ist für korrekt implementierte Hardware notwendig, aber nicht hinreichend.

Während der Simulation wird der Verlauf aller Signale im Simulations-Logfile gespeichert. Um die Ursachen evtl. aufgetretener Fehler besser analysieren zu können, ist es sinnvoll, den Verlauf ausgewählter Signale zu visualieren (Window/Wave). Die vom Simulations-Skript eingebundene Datei `<PrakTIES>/wave/<project>/<level>.do` sorgt dafür, dass alle Ein- und Ausgänge des Toplevel-Moduls angezeigt werden. Um diese Ansicht um weitere interne Signale zu erweitern, selektieren Sie in Window/Sim die gewünschte Modul-Instanz und fügen Sie danach in Window/Objects die gewünschten Signale zur Wave-Anzeige hinzu. Um die erweiterte Waveform nach dem nächsten Simulationsdurchlauf automatisch wieder anzeigen zu lassen, speichern Sie diese (File/Save Format) unter `<PrakTIES>/wave/<project>/<level>.do`.

② Synthese [Synplify Pro]

Zur Synthese des Toplevel-Moduls öffnen Sie `<PrakTIES>/<hdl>/<project>/synthese.prj` in *Synplify Pro* (`<LIBERO>/Synopsys/synplify_H201303M1/bin/synplify_pro.exe`) und starten anschließend die Synthese (Run). Wenn Ihre Implementierung syntaktisch korrekt und synthetisierbar ist, wird neben der eigentlichen Netzliste auch eine Synthese-Report erzeugt, dem Sie Timing-Information und den Ressourcen-Bedarf Ihrer Schaltung entnehmen können. Im RTL-View (HDL-Analyst/RTL/Hierarchical View) bzw. Technology View (HDL-Analyst/Technology/Hierarchical View) können Sie die erzeugte Schaltung vor bzw. nach dem Mapping auf die Zieltechnologie untersuchen. Zusätzlich wird die für die Post-Synthese Simulation benötigte Gate-Level-Beschreibung des Toplevel-Moduls generiert und die Synthese-Constraints in entsprechende Layout-Constraints übersetzt.

③ Post-Synthese Simulation [ModelSim, `<level> = postsynth`]

Wiederholen Sie Schritt ① mit `<level> = postsynth`. Dabei wird die während der Synthese erzeugte Gate-Level-Beschreibung des Toplevel-Moduls eingebunden, wodurch nun auch Bauteil-Verzögerungen in der Simulation berücksichtigt werden.

④ Erzeugen der Designer-Datenbank [Designer]

Das Layout (Place & Route) Ihrer Hardware (Schritt ⑤) und das Generieren des Bitstreams (Schritt ⑦) erfolgt mit dem *Designer* Tool (`<LIBERO>/Designer/bin/designer.exe`). Dieses arbeitet auf einer Datenbank, in welche die notwendigen Quelldateien (Netzliste, Layout-Contraints und Pin-Assignment) zunächst importiert werden müssen. Führen Sie dazu über File/Execute Script das Designer-Skript `<PrakTIES>/design/<project>.tcl` aus. Dieser Schritt muss nur ein einziges mal ausgeführt werden. Nach einer Änderung an einer der Eingabe-Dateien importiert

der Designer diese automatisch neu. Die dabei abgefragten Dialoge können unverändert bestätigt werden.

⑤ Layout [Designer]

Öffnen Sie die im Schritt ④ erzeugte Designer-Datenbank und starten Sie Tools/Back-Annotate. Die dabei abgefragten Dialoge (Compile Options, Layout Options) können unverändert bestätigt werden. Lediglich im Back-Annotate Dialog müssen Sie die von Ihnen verwendete HDL selektieren. Ergebnis dieses Durchlaufs ist die Gate-Level-Beschreibung des Toplevel-Moduls und die Timing-Information über das Routing-Delay, welche für die Post-Layout Simulation benötigt werden.

Über Tools/ChipPlanner können Sie nun auch die Verteilung der Logik auf dem FPGA untersuchen. Des Weiteren liefert Tools/SmartTime Timing Analyzer die Frequenz, mit der die erzeugte Hardware maximal getaktet werden kann.

⑥ Post-Layout Simulation [ModelSim, `<level> = postlayout`]

Wiederholen Sie Schritt ① mit `<level> = postlayout`. Dabei wird die während der Back-Annotation erzeugte Gate-Level-Beschreibung des Toplevel-Moduls eingebunden. Zusammen mit den Leitungsverzögerungen wird somit das reale Zeitverhalten der Schaltung simuliert.

⑦ Generieren des Bitstreams [Designer]

Öffnen Sie die im Schritt ④ erzeugte Designer-Datenbank und starten Sie Tools/Generate Programming File. Die dabei abgefragten Dialoge können unverändert bestätigt werden.

⑧ Programmieren des FPGAs [Flash Pro]

Schließen Sie das FPGA über den Mini-USB Port (J1) an den Rechner an. Die mit Libero SoC installierten Treiber binden das Gerät als `USB-Controller/Actel FlashPro 3/3x Programmer` ein. Bei korrektem Anschluss leuchtet die LED D3 am FPGA gelb auf.

Öffnen Sie nun das *Flash Pro* Tool (`<LIBERO>/Designer/bin/flashpro.exe`) und führen Sie über File/Run Script das Programmier-Skript `<PrakTIES>/prog/<project>.tcl` aus. Dadurch wird das Programmier-Projekt erzeugt und der im Schritt ⑦ erzeugte Bitstream eingebunden. Der Programmiervorgang startet automatisch (LED D2 blinkt) und wird im Erfolgsfall durch das Aufleuchten der grünen LED D1 am FPGA quittiert.

Das Programmier-Skript schließt das Projekt nach dem Programmieren automatisch, da sonst eine Schreibsprerre der Bitstream-Datei ein erneutes Generieren des Bitstreams verhindert.

Aufgabe 1 Schaltverhalten des Komparators

Das IGLOO FPGA unterstützt die beiden differentiellen I/O-Standards LVDS und LVPECL. Die zentrale Komponente eines differentiellen Eingangs ist ein Komparator, also ein im Sättigungsbereich betriebener Operationsverstärker. Abbildung 4 illustriert dessen ideales Schaltverhalten qualitativ. Für $V_\mathrm{p} > V_\mathrm{n}$ gibt der Komparator seine positive Versorgungsspannung $\mathrm{V_{CC}}$ aus. Andernfalls wird die negative Versorgungsspannung $\mathrm{V_{EE}}$ ausgegeben.

Reale Komparatoren weisen dagegen einen endlichen Umschaltbereich $(V_f^l \ldots V_r^h)$ auf, in dem $V_\mathrm{p} > V_\mathrm{n}$ nicht sicher entschieden werden kann. Außerdem unterscheiden sich die Schaltschwellen bei steigender $(V_r^l \ldots V_r^h)$ und fallender $(V_f^h \ldots V_f^l)$ Spannungsdifferenz (Hysterese).

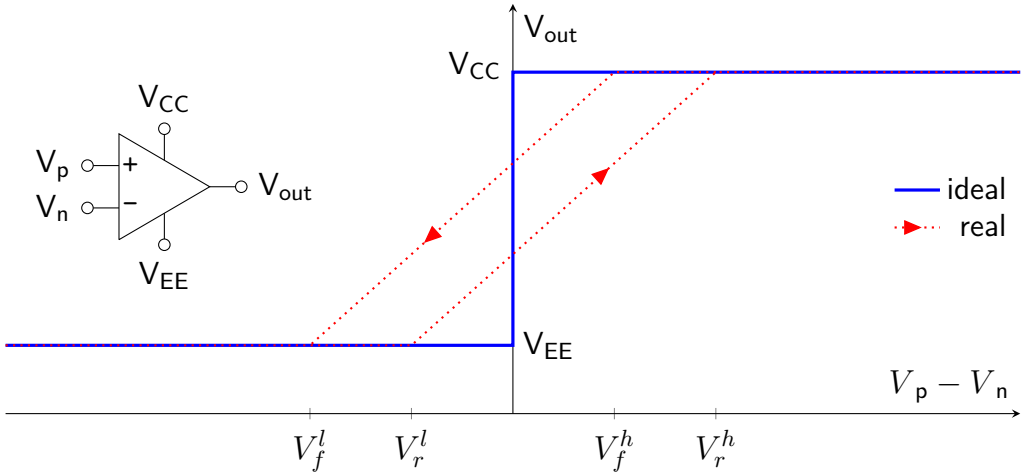

Abbildung 4: Komparator mit idealem und realem Schaltverhalten

Ziel dieser Aufgabe ist es, den für die ADC-Anwendung besser geeigneten differentiellen Eingang auszuwählen. Dafür ist unter anderem das Hystereseverhalten der beiden Komparatortypen zu vermessen.

Aufgabe 1.1 Datenblatt-Analyse

Welche für die ADC-Anwendung relevanten Informationen über die beiden differentiellen Eingänge können Sie dem IGLOO-Datenblatt [2] entnehmen? Welche Eigenschaften sprechen für LVDS, welche für LVPECL?

Aufgabe 1.2 Hardware-Modul für Komparator-Test

Der Ausgang eines Komparators ist im Umschaltbereich instabil und muss daher über ein Register für die Dauer einer Taktperiode auf einen Wert fixiert werden. Dazu ist das in Abbildung 5 dargestellte TOP-Modul in `<PrakTIES>/<hdl>/comparator/top.v[hd]` zu implementieren.

Instanziieren Sie das vorgegebene OPAMP-Modul aus `<PrakTIES>/<hdl>/opamp.v[hd]`. Je nach Konfigurationsparameter *OPAMP_TYPE* (in `<PrakTIES>/<hdl>/settings.v[hd]`) instanziiert dieses einen der differentiellen Eingangspuffer oder generiert eine simulationsfähige Ersatzschaltung. Dies ist notwendig, weil der RTL-Simulator die differentiellen Eingänge nicht unterstützt.

Das Ausgangsregister im TOP-Modul ist als vorderflankengesteuertes Flip-Flop mit synchronem reset (active low) durch eine Verhaltensbeschreibung zu implementieren.

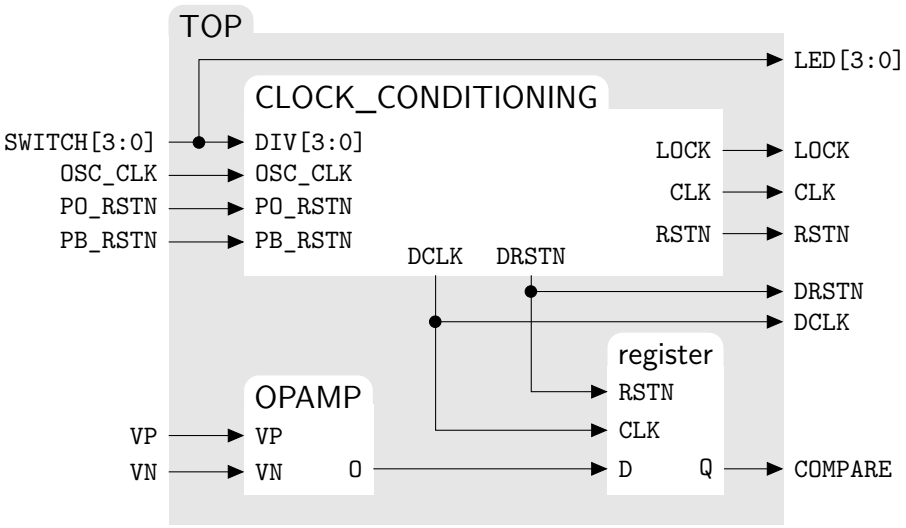

Abbildung 5: Toplevel-Modul für Messungen am Komparator

Instanziieren Sie außerdem das (noch zu implementierende) CLOCK_CONDITIONING-Modul aus `<PrakTIES>/<hdl>/clock_conditioning.v[hd]`, um das Takt- und Resetsignal für dieses Register zu erzeugen.

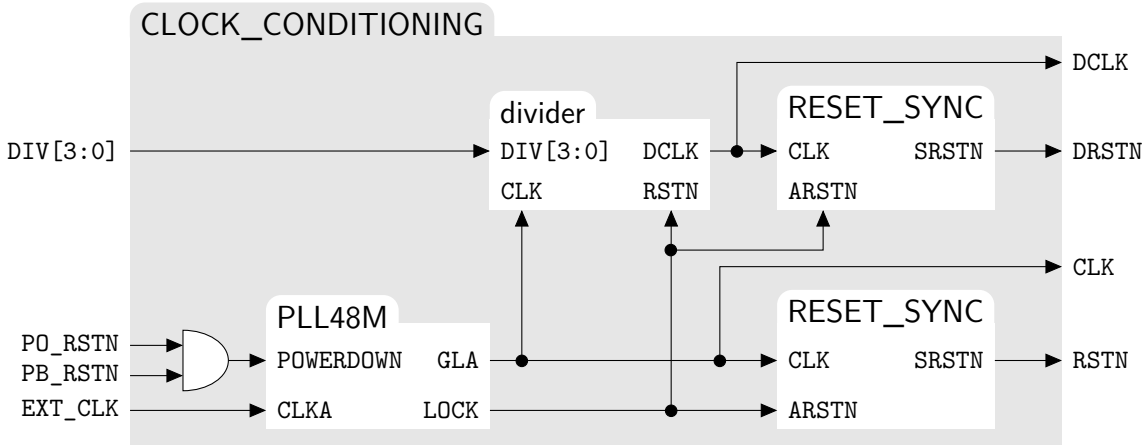

Abbildung 6: Takt-Konditionierungsmodul

Wie in Abbildung 6 gezeigt, erzeugt das CLOCK_CONDITIONING-Modul zwei verschiedene Taktsignale aus dem 48 MHz Takt `OSC_CLK`, welches von einem Oszillator auf dem Evaluationsboard generiert wird. Das `CLK`-Signal hat eine feste Frequenz von 4 MHz und wird mit dem statischen phase-locked loop PLL48M erzeugt, welches in `<PrakTIES>/<hdl>/pll48M.v[hd]` vorgegeben ist. Solange wenigstens eines der beiden externen Reset-Signale (`PO_RSTN` bzw. `PB_RSTN`) aktiv (low) ist, soll die Taktgenerierung abgeschaltet bleiben. Das PLL aktiviert sein `LOCK`-Signal (asynchron, active high), sobald der Ausgabetakt stabil anliegt.

Das `DCLK`-Signal wird aus dem `CLK`-Signal durch einen selbst zu implementierenden Taktteiler abgeleitet. Der Teilungsfaktor wird durch das als vorzeichenlose Ganzzahl zu interpretierende `DIV`-Signal beeinflusst und ist zur Laufzeit veränderbar. Das `DCLK`-Signal soll eine Frequenz von $\frac{4}{\texttt{DIV}+1}$ MHz und einen Tastgrad (duty cycle) zwischen 33 % und 66 % haben.

Zu jedem der beiden Taktsignale ist ein synchrones Reset-Signal (active low) zu erzeugen. Implementieren Sie ein entsprechendes RESET_SYNC-Modul in `<PrakTIES>/<hdl>/reset_sync.v[hd]`. Dessen `SRSTN` Ausgang soll für wenigstens einen vollständigen Takt auf 0 gehalten werden, nachdem das asynchrone `ARSTN`-Signal (active low) deaktiviert wurde.

Aufgabe 1.3 Pinzuweisung

Ordnen Sie nun jedem funktionalen Ein- und Ausgangssignal des TOP-Moduls einen physikalischen Pin des FPGAs zu. Ergänzen Sie hierfür in `<PrakTIES>/constraints/comparator.pdc` die fehlenden Angaben <?>. Beachten Sie dabei die folgende Hinweise:

- Der Schaltplan des Evaluationsboard [9] enthält die Zuordnung zwischen FPGA-Pins (BGA-Index) und deren externer Beschaltung.
- Die Pins für `OSC_CLK, PO_RSTN` und `PB_RSTN` sind eindeutig festgelegt.
- Alle weiteren Ein- und Ausgänge (außer `LED` und `SWITCH`) sollen über den Pinheader P5 erreichbar sein.
- Die OPAMP-Eingänge (`VP,VN`) sind für beide Komparator-Varianten anzugeben. Die LVDS-Konfiguration bleibt aber vorerst auskommentiert.
- (`VP,VN`) muss ein differentielles Pinpaar des FPGAs sein. Die funktionale Eigenschaft der FPGA-Pins [3, S. 206] ist im Schaltplan neben dem BGA-Index notiert. Diese Zuordnung gilt aber nur für ein M1AGL600-FPGA. Da für das im Praktikum verwendete Evaluationsboard mit dem größeren M1AGL1000-FPGA kein separater Schaltplan existiert, entnehmen Sie die funktionale Pinbeschreibung dem FPGA-Datenblatt [2, S. 230].
- Die Versorgungsspannung der I/O-Bank von (`VP,VN`) muss den Anforderungen von LVDS bzw. LVPECL genügen.
- Zwischen `VP` und `VN` darf auf dem Evaluationsboard kein Terminierungswiderstand verbaut sein.

Aufgabe 1.4 Simulation

Nennen Sie Ihre CLOCK_CONDITIONING-Instanz `i_clk` und Ihre `PLL48M`-Instanz `i_pll`. Dadurch wird sichergestellt, dass die Synthese-Constraints aus `<PrakTIES>/constraints/comparator.sdc` richtig angewendet werden können.

Setzten Sie *OPAMP_TYPE* auf 0 und weisen Sie die LVPECL-Pins den beiden Komparator-Eingängen zu. Führen Sie nun die Schritte ① bis ⑥ des HDL-Werkzeugflusses für `<project> = comparator` aus, bis alle Simulationen ohne Fehlermeldung durchlaufen werden.

Erweitern Sie die Waveform (`<PrakTIES>/wave/comparator/postlayout.do`) der Postlayout-Simulation so, dass alle Verzögerungen einer steigenden Signalflanke auf dem kombinatorischen Pfad vom `VP`-Pad zum `COMPARE`-Register-Eingang abgelesen werden können. Wie groß ist die Gesamtverzögerung? Welcher relative Anteil an dieser Verzögerung wird durch die Leitungslaufzeiten verursacht?

Setzen Sie *OPAMP_TYPE* auf 1 und führen die Schritte ②,⑤,⑦ und ⑧ des HDL-Werkzeugflusses durch. Die von Ihnen beschriebene Hardware wird nun auf dem FPGA ausgeführt.

Aufgabe 1.5 Vermessung der Taktgenerierung

Bauen Sie die in Abbildung 7 gezeigte Konfiguration aus FPGA-Evaluationsboard und Oszilloskop auf. Das `PB_RSTN`-Signal können Sie am Testpunkt TP5 des Evaluationsboards abgreifen. Verwenden Sie die beiden automatischen Messmodule des Oszilloskops, um sich die Frequenzen von `CLK` und `DCLK` anzeigen zu lassen. Das Oszilloskop HMO2524 kann die Frequenz der digitalen Kanäle im Automeasure-Modus nicht bestimmen. *Für diese Aufgabe ist daher das leistungsfähigere Gerät (HMO3524) zu verwenden.*

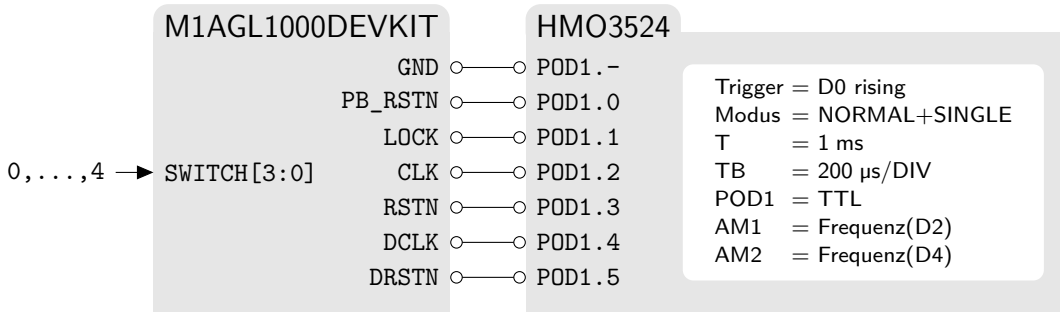

Abbildung 7: Versuchsaufbau – Synchrone Resets und Taktfrequenz

Im Folgenden soll der Start des PLL48M-Moduls einmalig erfasst und danach verschiedene Messungen daran vorgenommen werden. Setzten Sie dafür `SWITCH` $= 0$ und aktivieren Sie den NORMAL+SINGLE-Modus des Trigger-Systems am Oszilloskop. Durch das Betätigen des Reset-Buttons am Evaluationsboard werden die digitalen Signale nun an der beabsichtigten Stelle erfasst.

Nutzen Sie nun die Mess-Cursor, um die Dauer

- zwischen den steigenden Flanken von `PB_RSTN` und `LOCK`,
- der ersten `CLK`-Periode nach der steigenden Flanke von `PB_RSTN` und
- der ersten `CLK`-Periode nach der steigenden Flanke von `LOCK`

zu bestimmen. Zu jeder dieser Messungen ist eine sinnvolle Zeitauflösung zu wählen und ein Screenshot zu erstellen.

Schalten Sie nun die Mess-Cursor wieder ab. Stellen Sie die steigende `LOCK`-Flanke und die ersten 20 `CLK`-Perioden bei einer Zeitauflösung von 500 ns/DIV dar. Die Frequenzen der beiden Taktsignale werden nun von den automatischen Messmodulen erfasst. Wiederholen Sie diese Messung für $1 \leq$ `SWITCH` ≤ 4 und dokumentieren Sie Ihre Beobachtung durch entsprechende Screenshots. Entspricht der Verlauf der Takt und Reset-Signale Ihren Erwartungen?

Falls Sie sich den Unterschied zwischen dem sauberen digitalen Taktsignalen und dem realen Spannungsverlauf des Taktsignals veranschaulichen möchten, können Sie das `DCLK`-Signal zusätzlich am CH1 des Oszilloskops anzeigen lassen. Dies ist aber nicht Teil der bewerteten Aufgabenstellung.

Aufgabe 1.6 Vermessung des Komparator-Umschaltbereichs

Um den Schaltvorgang des Komparators zu untersuchen, müssen zunächst passende Eingangssignale für `VP` und `VN` erzeugt werden. Realisieren Sie hierfür den in Abbildung 8 gezeigten Versuchsaufbau und achten Sie dabei insbesondere auf die korrekte Verbindung der Bezugspotentiale (GND).

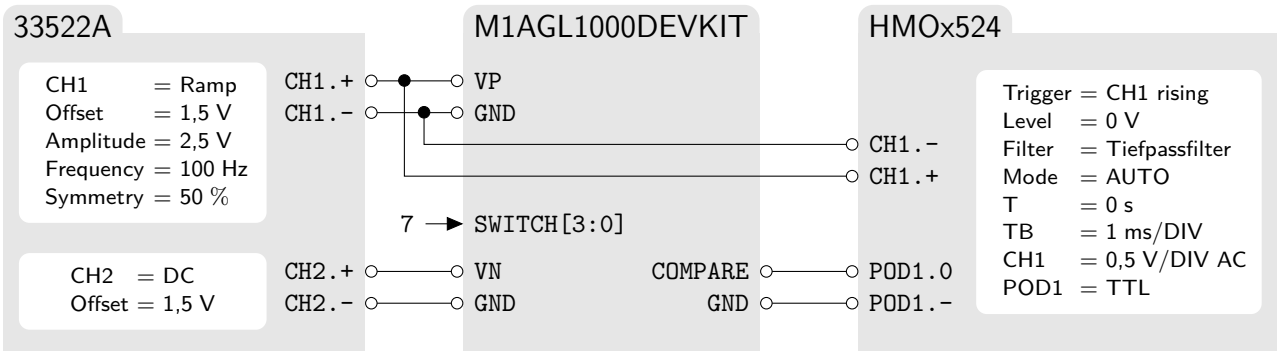

Abbildung 8: Versuchsaufbau – Umschaltbereich des Komparators

Erstellen Sie Screenshots für Zeitauflösungen von 1 ms/DIV und 20 µs/DIV. Beschreiben Sie Ihre Beobachtungen und erklären Sie, warum die an CH1 angezeigte Spannung dafür geeignet ist, den Umschaltbereich des Komparators zu vermessen. Welche Messungenauigkeit (in mV) wird durch das Ausgaberegister im Toplevel-Modul der Schaltung (vgl. Abbildung 5) verursacht?

Vergrößern Sie die horizontale und vertikale Auslösung des Oszilloskops nun soweit, bis der Umschaltbereich des Komparators gerade noch vollständig dargestellt wird. Schalten Sie das Oszilloskop nun in den Erfassungsmodus Hüllkurve. Dabei werden die minimalen und maximalen Werte von CH1 über fortlaufende Triggerfenster ermittelt. Nach einer ausreichend großen Zeitspanne zeichnet sich der Umschaltbereich des Komparators ab. Verwenden Sie die Mess-Cursor um V_r^l und V_r^h zu bestimmen. Wiederholen Sie das Verfahren bei fallender CH1-Flanke und ermitteln Sie V_f^l und V_f^h. Damit haben Sie den Umschaltbereich für den LVPECL-Komparator vermessen. Bestimmen Sie die vier Werte nun auch für den LVDS-Komparator. Die Screenshots aller Messungen sind in der Abgabe einzubinden.

Stellen Sie die Hysterese-Kurven beider Komparatoren in einem Diagramm quantitativ dar. Welcher der beiden Komparator-Typen ist nach Ihren Messungen für die ADC-Anwendung besser geeignet?

Aufgabe 2 Signalkonditionierung

In Aufgabe 1 wurde ein Operationsverstärker als Komparator verwendet, um zu entscheiden, welche der beiden Eingangsspannungen größer ist. Der Operationsverstärker befindet sich dabei die meiste Zeit in Sättigung, gibt also eine seiner beiden Versorgungsspannungen (V_{CC} bzw. V_{EE}) aus. Der schmale Umschaltbereich zwischen den beiden Zuständen ist ein ungewollter (aber eben unvermeidbarer) Effekt.

Ein Operationsverstärker kann aber auch genau umgekehrt verwendet werden. Durch eine externe Beschaltung kann der Arbeitspunkt des Operationsverstärkers in den Umschaltbereich verschoben werden, um eine Eingangsspannung in eine skalierte und evtl. verschobene Ausgangsspannung zu transformieren. Der Sättigungsbereich wird dabei i.d.R. nicht erreicht.

Aufgabe 2.1 Gegenkopplung von Operationsverstärkern

(a) invertierend (b) nicht invertierend

Abbildung 9: OPV-Verstärkerschaltungen

Abbildung 9 zeigt zwei typische Verstärkerschaltungen. Beide benutzen einen Operationsverstärker und koppeln dessen Ausgang V_{out} über einen Spannungsteiler auf den invertierenden Eingang zurück. Neben dem eigentlichen Signaleingang V_{in} verwenden diese Verstärker außerdem eine (i.d.R. konstante) Referenzspannung V_{ref}. Dies ist einer Verallgemeinerung der in der Literatur häufig beschriebenen Grundschaltung mit $V_{ref} = V_{EE}$ (= GND) [12].

Beide Verstärker realisieren eine affine Transformation $T_{G,o} : V_{in} \mapsto V_{out} = G \cdot V_{in} + o$, solange $V_{EE} \leq V_{out} \leq V_{CC}$. Leiten Sie Ausdrücke für den Offset o und den Verstärkungsfaktor G beider Schaltungen in Abhängigkeit vom R_1, R_2 und V_{ref} her. Gehen Sie dabei vom idealen Operationsverstärker mit unendlich großer Differenzenverstärkung und unendlich großem Eingangswiderstand aus.

Aufgabe 2.2 Anwendung von Verstärkerschaltungen

Ein wichtiges Anwendungsgebiet der Verstärkerschaltungen ist die Signalkonditionierung von Messsystemen. Dabei wird der Ausgabebereich $V^{\text{Sensor}} := \{V \in \mathbb{R} : V^{\text{Sensor}}_{\text{min}} \leq V \leq V^{\text{Sensor}}_{\text{max}}\}$ eines Sensors auf den Eingangsbereich $V^{\text{ADC}} := \{V \in \mathbb{R} : V^{\text{ADC}}_{\text{min}} \leq V \leq V^{\text{ADC}}_{\text{max}}\}$ eines AD-Wandlers abgebildet. Dafür werden häufig auch mehrere Verstärkerstufen in Reihe geschaltet, was einer Konkatenation der Transformationen entspricht. Es darf aber weiterhin kein Zwischenergebnis in der Transformationskette den Sättigungsbereich erreichen.

Entwerfen Sie für $V^{\text{Sensor}}_{\text{min}}$ = -0,4 V, $V^{\text{Sensor}}_{\text{max}}$ = 0,4 V, $V^{\text{ADC}}_{\text{min}}$ = 0,25 V, $V^{\text{ADC}}_{\text{max}}$ = 2,25 V eine *nicht-invertierende* Schaltung, welche V^{Sensor} auf V^{ADC} abbildet. Geben Sie diese Schaltung in einen beliebigen Schaltungssimulator ein, bspw. [13, 14, 15]. Verwenden Sie dabei ausschließlich die folgenden Komponenten:

- eine 2,5 V Gleichspannungsquelle als Versorgungsspannung
- eine $0 \pm 0,4$ V Wechselspannungsquelle einer Frequenz von 1 Hz zur Simulation des Sensorsignals
- möglichst wenige Operationsverstärker mit Versorgungsspannungsanschlüssen
- möglichst wenige Wiederstände (Nennwert ausschließlich 10 kΩ)

Als Lösung dieser Aufgabe wird neben dem Schaltbild die simulierte Ein- und Ausgangsspannung für 1 s Simulationszeit erwartet.

Aufgabe 2.3 Dynamische Vorverstärkung

Nutzt das transformierte Signal den Eingangsbereich des AD-Wandlers voll aus, dann kann es mit maximaler Genauigkeit erfasst werden. Bei realen Messsystemen (bspw. der Erfassung von Schwingungen an einer Verkehrsbrücke) kann der dynamische Anteil des Signalpegels allerdings verringert werden (bspw. bei ausbleibendem Verkehr), so dass der ADC nicht mehr voll ausgesteuert wird (vgl. Abbildung 10). Um die Messgenauigkeit zu erhöhen, muss der dynamische Anteil des Sensorsignals in solchen Fällen mit einem zur Laufzeit veränderlichen Verstärkungsfaktor skaliert werden.

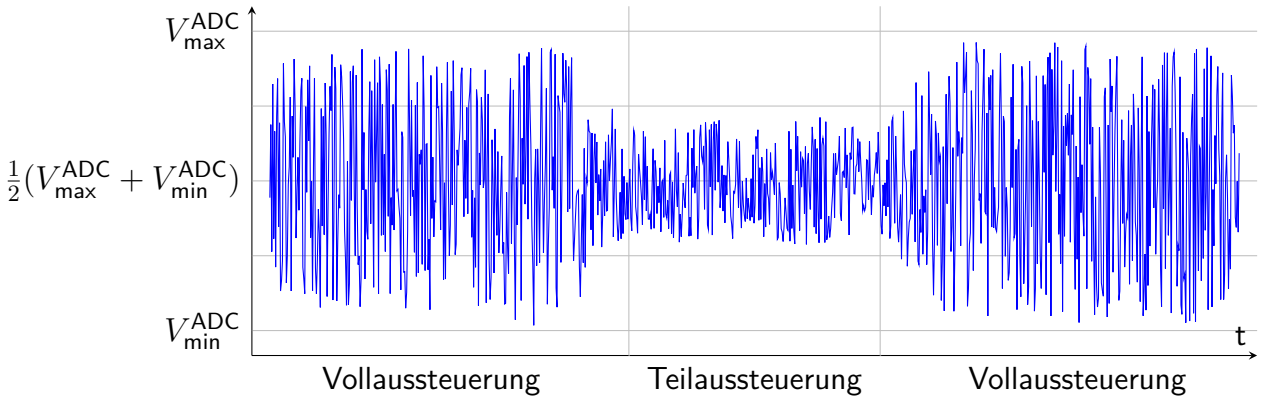

Abbildung 10: Statische und dynamische Anteile eines Messsignals

Ein PGA (*programmable gain amplifier*) ist eine Verstärkerschaltung mit einer digitalen Schnittstelle zur Konfiguration des Verstärkungsfaktors. Im Praktikum wird ein LMP8100 [16] verwendet, um das Messsignal auf den vollen ADC-Eingangsbereich ($V_{\text{min}}^{\text{ADC}} = 0{,}25$ V, $V_{\text{max}}^{\text{ADC}} = 2{,}25$ V) zu verstärken. Gehen Sie davon aus, dass der statische Anteil des Messsignals in der Mitte des Spannungsbereichs liegt, wie in Abbildung 10 gezeigt. Der LMP8100 wird mit $V^+ = 3{,}3$ V und $V^- = 0$ V vom FPGA-Evaluationsboard versorgt. Entnehmen Sie dem Datenblatt des PGAs folgende Angaben:

- Um welchen Verstärker-Typ (invertierend/nicht-invertierend) handelt es sich?
- Welche Verstärkungsfaktoren können eingestellt werden?
- Wie lange dauert ein Wechsel zwischen zwei Verstärkungsfaktoren mindestens?
- Welche Spannung muss an den beiden `GRT`-Pins anliegen, um die in Abbildung 10 dargestellte Teilaussteuerung zur Vollaussteuerung zu verstärken?
- Welche passiven Bauteile werden für die externe Beschaltung des PGAs benötigt?

Aufgabe 3 Lochraster-Prototyp der Erweiterungsplatine

In dieser Aufgabe wird das in Abbildung 11 gezeigte Erweiterungsboard für das Messsystem entworfen und als 2,54 mm-Lochraster-Prototyp realisiert. Neben dem PGA und einer Spannungsreferenz enthält das Erweiterungsboard den $C_{\mathrm{M}} = 100$ nF Messkondensator und den $R_{\mathrm{M}} = 3{,}3$ kΩ Messwiderstand. Beachten Sie, dass für die externe Beschaltung der beiden integrierten Schaltungen (LMP8100 [16] und LT6656 [17]) ggf. weitere passive Bauteile notwendig sind.

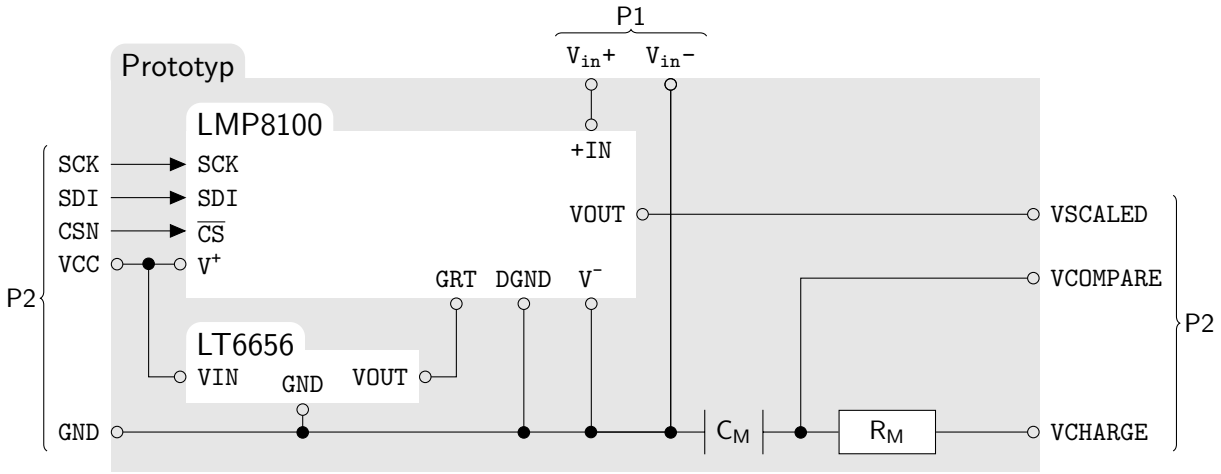

Abbildung 11: Aufbau des Erweiterungsboards (*ohne passive Komponenten für externe IC-Beschaltung*)

Die zu vermessende Eingangsspannung $\mathrm{V_{in}}$ wird dem Erweiterungsboard über einen 2×1 Pinheader P1 zugeführt. Gehen Sie davon aus, dass sich $\mathrm{V_{in}}$ wie in Aufgabe Aufgabe 2.3 beschrieben verhält. Die Schnittstelle zum FPGA-Evaluationsboard für die digitalen und analogen Signale sowie die Spannungsversorgung (`VCC` = 3,3 V) erfolgt über den 2×20 Pinheader P2. Dieser soll später direkt in den Pinheader P5 des FPGA-Evaluationsboards eingesteckt werden können.

Aufgabe 3.1 Schaltplan

Entwerfen Sie den Schaltplan `<PrakTIES>/hw/prototype.sch` für das Erweiterungsboard mit der freien Version des Schaltungseditors Eagle [18]. Verwenden Sie folgende Schaltsymbol-Bibliotheken:

- `<PrakTIES>/hw/prototype.lbr` für den LMP8100 und LT6656,
- `rcl` (EAGLE-intern) für die passiven Bauteile und
- `pinhead` (EAGLE-intern) für die Pinheader

Die konkrete Bauform (package) der passiven Bauteile ist für den Schaltplan noch nicht relevant. Deren Nennwerte sind aber zu beschriften.

Beachten Sie bei der Zuweisung der Signale zum Pinheader P2, dass

- die Positionen von `GND` und `VCC` durch das FPGA-Evaluationsboard fest vorgegeben sind,
- P2 beim Einstecken in das FPGA-Evaluationsboard auf dem Kopf steht und
- `VSCALED` am negativen Eingang des LVDS-Komparators anliegen soll.

Der Schaltplan muss nicht kompakt, sondern übersichtlich sein. Die Anordnung der Pins an einem Schaltplansymbol entspricht i.d.R. nicht der tatsächlichen Anordnung am zugehörigen IC-Gehäuse. Der Schaltplan ist somit nicht für das Platzieren von Bauteilen oder Routen von Signalleitungen geeignet.

Aufgabe 3.2 Bauteile

Erstellen Sie eine Stückliste (bill of materials) aller Bauteile (inkl. LT6656 aber exkl. LMP8100), welche Sie für die Fertigung der Erweiterungsplatine benötigen. Geben Sie zu jeder Position eine Farnell-Bestellnummer, die Anzahl der benötigten Teile, den Einzelpreis und den Verwendungszweck an. Beachten Sie dabei folgende Hinweise:

- Die Gesamtkosten Ihrer Stückliste darf 20 € (brutto, ohne Versand) nicht überschreiten.
- Fertigungswerkzeug (Lötkolben, Pinzetten) und Verbrauchsmaterial (Lötzinn, Flussmittel, Draht) müssen nicht bestellt werden.
- Alle Bauteile müssen für die Durchsteckmontage im 2,54 mm-Raster geeignet sein. Sollten selbst bestellte Bauteile nur für die Oberflächenmontage erhältlich sein, sind entsprechende Adapter auszuwählen.
- Passive Bauteile, welche die Messgenauigkeit des Systems beeinflussen, sollten eine möglichst geringe Bauteiltoleranz und Temperaturabhängigkeit aufweisen.
- Die Mindestbestellmenge eines Artikels darf die benötigte Menge übersteigen. Für die Kosten Ihrer Stückliste zählen nur die Einzelpreise.
- Alle Bauteile müssen kurzfristig lieferbar sein (nicht ab Lager USA).

Aufgabe 3.3 Board-Layout

Entwerfen Sie nun das Board-Layout `<PrakTIES>/hw/prototype.brd` der Erweiterungsplatine.

Passen Sie dazu die Bauform (package) der passiven Elemente den konkreten Bauteilen Ihrer Stückliste an. Verwenden Sie die nächst größere Bauform, wenn Sie keine exakt passende finden können. Für einen bis zu 8 mm langen, axial bedrahteten Widerstand ist bspw. die 0207/10 Bauform geeignet. Die Größe des integrierten Schaltungen wurde in der vorgegebenen Bibliothek bereits korrekt hinterlegt.

Verwenden Sie die Grid-Einstellungen des Layout-Editors, um die Komponenten und Signalpfade im 2,54 mm-Lochraster anzuordnen. Nutzen Sie die Freiheiten der konfigurierbaren FPGA-Pinbelegung, um ein möglichst einfaches und kompaktes (max. 29×18 Rasterpunkte) Layout zu erzielen. Gegebenenfalls sind hierfür noch einmal Änderungen am Schaltplan notwendig.

Beachten Sie folgende Hinweise, um die anschließende Prototypen-Fertigung zu vereinfachen:

- Platzieren Sie alle Komponenten auf der Unterseite der Platine (Mirror Funktion). Achten Sie aber darauf, das keine abstehenden Komponenten das Aufstecken der Erweiterungsplatine auf das FPGA-Evaluationsboard verhindern.
- Platzieren Sie die Signalleitungen bevorzugt auf der Oberseite der Platine. Nur zum Vermeiden von Überkreuzungen werden die mit nicht isoliertem Kupferdraht realisierten Signale auf der Unterseite entlang geführt.

- Führen Sie die Signalleitungen immer entlang des Lochrasters und niemals diagonal oder zwischen zwei Lötinseln hindurch.

Aufgabe 3.4 Prototyp fertigen

Lassen Sie sich vom Praktikumsbetreuer in die Löttechnik einweisen. Fertigen Sie erst *nach* dieser Einweisung den von Ihnen entworfenen Prototypen.

Schließen Sie Ihren Prototypen an das FPGA an und bestimmen Sie die exakten Werte von V_{CC} und V_{ref} an den Pins 8 und 10 des PGAs.

Aufgabe 4 Auf- und Entladen des Kondensators

Die realen Eigenschaften von passiven Bauteilen weichen in vom Hersteller angegebenen Grenzen von deren nominellen Werten ab. In dieser Aufgabe werden die realen Werte von R_M und C_M bestimmt und die Konsequenzen ihrer Dimensionierung untersucht.

Aufgabe 4.1 Vermessen des Vorwiderstands

Zur Steuerung der Vergleichsspannung $V_{compare}$ für den Komparator wird der Messkondensator (C_M) über einen 3,3 V LVCMOS-Ausgang des FPGAs (`VCHARGRE`) auf- bzw. entladen. Warum muss ein Widerstand (R_M) zwischen Kondensator und FPGA-Pin geschaltet werden? Wie groß muss der Widerstand für den von Ihnen in Aufgabe 3.1 gewählten `VCHARGRE`-Pin mindestens sein? Warum ist es sinnvoller, einen deutlich größeren Widerstand zu verwenden? Wodurch wird die Größe des Widerstands nach oben begrenzt (es ist kein konkreter Wert zu berechnen)?

Bestimmen Sie den realen Wert $\hat{R}_M$ des Messwiderstands auf Ihrem Lochraster-Prototyp mit einem Handmultimeter. Liegt der Wert im spezifizierten Toleranzbereich?

Aufgabe 4.2 Vermessen des Kondensators

Die realen Bauteileigenschaften eines Kondensators lassen sich nicht direkt mit dem Handmultimeter bestimmen. Um die reale Kapazität $\hat{C}_M$ Ihres Messkondensators zu ermitteln, soll dessen Auf- und Entladecharakteristik

$$V_{\text{compare}}(t + \Delta t) = V_{\text{compare}}(t) + (V_{\text{charge}} - V_{\text{compare}}(t)) \cdot (1 - e^{-\Delta t / \hat{R}_M \hat{C}_M}) \tag{1}$$

untersucht werden. Realisieren Sie hierfür den in Abbildung 12 gezeigten Versuchsaufbau.

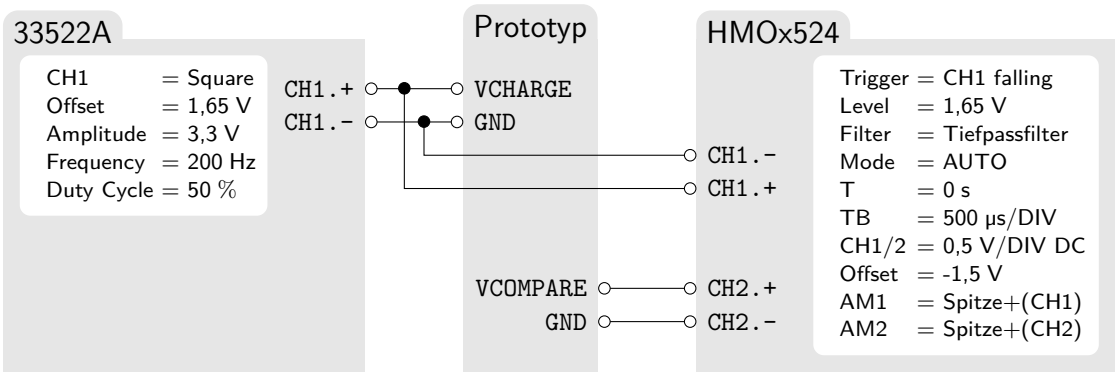

Abbildung 12: Versuchsaufbau – Auf-/Entladeverhalten des Kondensators

Erstellen Sie einen Screenshot, auf dem ein Auf- und ein Entladevorgang des Kondensators zu sehen ist und bestimmen Sie die maximale Spannung über $\hat{C}_M$.

Vergrößern Sie die Zeitauflösung des Oszilloskops nun auf 50 µs/DIV und stellen Sie den Beginn des Aufladevorgangs dar. Positionieren Sie die Mess-Cursor im Modus V-Marker so, dass die Zeitkonstante $\hat{\tau} = \hat{R}_M \cdot \hat{C}_M$ direkt abgelesen werden kann. Wiederholen Sie die Messung für den Entladevorgang und bestimmen Sie $\hat{C}_M$ aus dem Mittelwert der beiden gemessenen Zeitkonstanten. Liegt dieser im vom Hersteller angegebenen Toleranzbereich?

Aufgabe 4.3 Auswirkungen der Zeitkonstante

V_{compare} soll im V^{ADC}-Bereich ($0{,}25\ \text{V} \leq V_{\text{compare}} \leq 2{,}25\ \text{V}$) variiert werden. Welche Zeiten werden benötigt, um zwischen den beiden Randwerten zu wechseln? Welche Eigenschaft des ADC wird durch diese Zeit limitiert?

Durch die getaktete Arbeitsweise des ADC wird V_{charge} für die Dauer $T_{\texttt{DCLK}}$ einen Arbeitstakts konstant gehalten. Dadurch kann V_{compare} die Zielspannung V_{scaled} unter- oder überschreiten, ohne dass ADC_CONTROL dies bemerkt. $T_{\texttt{DCLK}}$ beeinflusst somit die Genauigkeit des ADC. Geben Sie die maximale Spannungsänderung $\Delta V_{\text{compare}}$ innerhalb eines Taktes für $V_{\text{compare}} \in V^{\text{ADC}}$ in Abhängigkeit von $T_{\texttt{DCLK}}$ an. Stellen Sie diese Abhängigkeit für alle mit dem CLOCK_CONDITIONING-Modul (vgl. Aufgabe 1) generierbaren Taktperioden graphisch dar. Erklären Sie anhand Ihrer Ergebnisse aus Aufgabe 1.6, warum die Wahl von $T_{\texttt{DCLK}} < 2\ \mu\text{s}$ nicht sinnvoll ist.

Aufgabe 4.4 Variation im Arbeitsbereich

Abschließend soll das Verhalten des Kondensators unter den realen Rahmenbedingungen der AD-Wandlung analysiert werden. Dazu ist die Spannungsänderung $\Delta V_{\text{compare}}$ während eines 2 µs dauernden Auf- und Entladevorgangs für verschiedene V_{compare} zu ermitteln.

Geben Sie zunächst an, wie groß die Frequenz eines Rechteck-Signals mit Duty-Cycle $\alpha \in [0, 1]$ sein muss, um eine 2 µs lange high-Phase ($f_h(\alpha)$) bzw. low-Phase ($f_l(\alpha)$) zu erreichen.

Aktivieren Sie nun die Bandbreitenbegrenzung (BWL) für CH2 am Oszilloskop und erhöhen Sie die Zeitauflösung auf 200 ns/DIV. Stellen Sie nacheinander $f_h(\alpha)$ und $f_l(\alpha)$ für $\alpha \in \{20\ \%, 30\ \%, 40\ \%, 50\ \%, 60\ \%\}$ am Funktionsgenerator ein. Stellen Sie für jede dieser Konfigurationen die 2 µs dauernde High- bzw. Low-Phase von CH1 am Oszilloskop dar. Bestimmen Sie jeweils den Wert von CH2 zu Beginn dieser Phasen mit einem Mess-Cursor. Um nun die Spannungsänderung an CH2 innerhalb dieser Phase möglichst genau bestimmen zu können, unterdrücken Sie den Gleichspannungsanteil von CH2 und erhöhen Sie dessen Auflösung auf 5 mV/DIV. Für alle 20 Messungen sind Screenshots zu erstellen.

Erstellen Sie ein Diagramm mit zwei Graphen für die theoretisch zu erwartende Spannungsänderung innerhalb von 2 µs beim Auf- bzw. Entladevorgang in Abhängigkeit von der Kondensatorspannung vor dem Auf- bzw. Entladevorgang. Verwenden Sie dabei Ihre experimentell ermittelte Zeitkonstante. Tragen Sie Ihre tatsächlich ermittelten Messwerte in das Diagramm ein. Entspricht das beobachtete Verhalten im untersuchten Arbeitsbereich dem erwarteten Verhalten?

Aufgabe 5 Beschreibung und Verifikation der Steuerung des Messsystems

In dieser Aufgabe werden die in Abbildung 13 gezeigten Hardware-Module des Messsystems implementiert und deren funktionales Verhalten verifiziert.

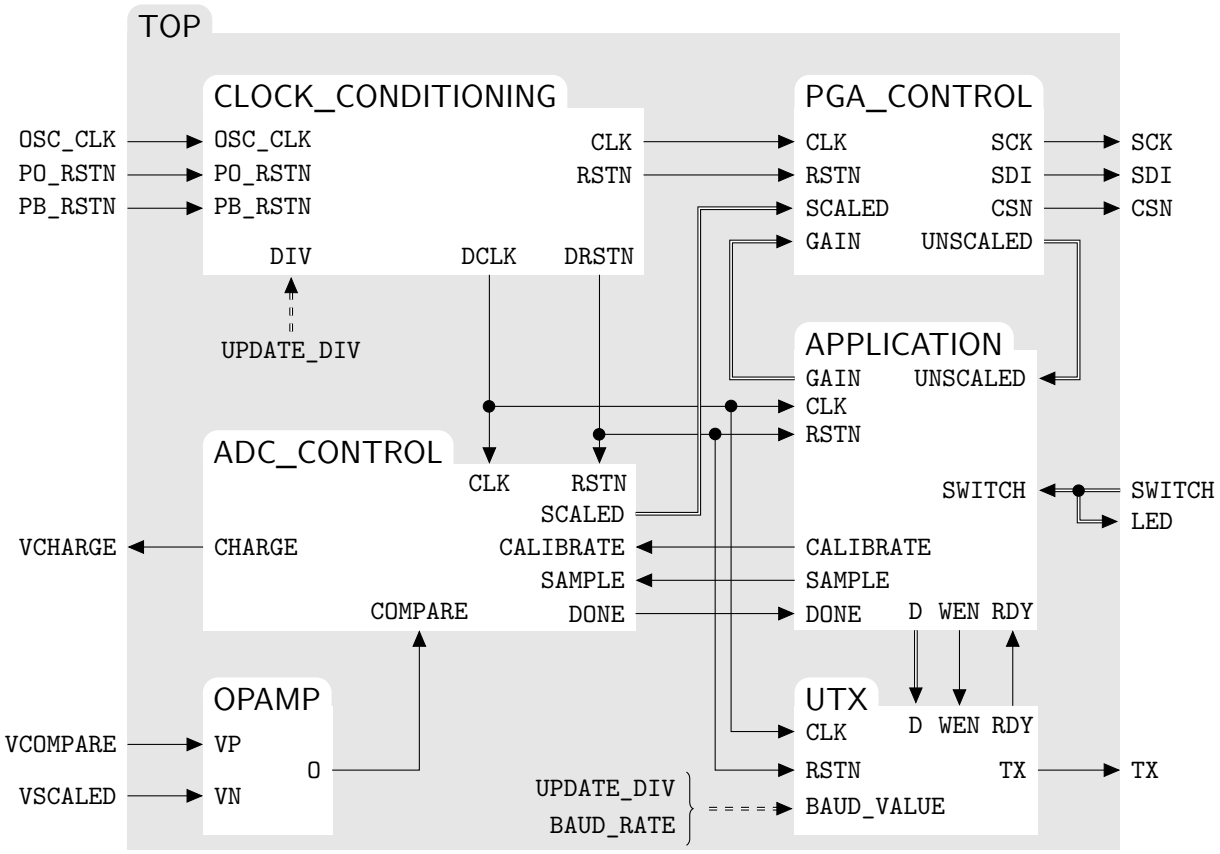

Abbildung 13: Toplevel-Modul des Messsystems

Der Komparator vergleicht die Spannung $V_{compare}$ über dem Messwiderstand mit der vom PGA verstärkten Spannung V_{scaled}. Das ADC_CONTROL-Modul verwendet diese Information, um das Auf- und Entladen des Messkondensators über V_{charge} zu steuern und gleichzeitig den aktuellen Wert von $V_{compare}$ aus der Abfolge der Auf- und Entladezyklen abzuschätzen. Sobald $V_{compare}$ nur noch um V_{scaled} schwankt, enthält das `SCALED`-Signal die digitale Repräsentation von V_{scaled}. Das PGA_CONTROL-Modul berechnet daraus das `UNSCALED`-Signal als digitale Repräsentation von V_{in}. Dieses wiederum wird vom APPLICATION-Modul byteweise an den UART-Sender UTX übertragen, um die abgetasteten Werte später am PC analysieren zu können.

Verschiedene Aspekte des Messsystems können mit den in `<PrakTIES>/<hdl>/settings.v[hd]` definierten Parametern konfiguriert werden. Der Wertebereich mancher Parameter (bspw. *GAIN*) ist eingeschränkt, andere Parameter (*SAMPLE_WIDTH*) können beliebige Werte annehmen. Änderungen der Konfigurationsparameter sollen ohne Anpassung der Hardwarebeschreibungen der einzelnen Module möglich sein.

Aufgabe 5.1 Implementierung der Hardware-Module

ADC_CONTROL: `<PrakTIES>/<hdl>/adc_control.v[hd]`
Dieses Modul setzt das in [1] beschriebene Konzept um und leitet aus der Folge von Auf- und

Entladezyklen die aktuelle Spannung über dem Kondensator ab. Hierfür sind Berechnungen im Festkommaformat notwendig, da die Synthese von Gleitkommaoperationen zu aufwendig wäre. Ein Festkommawert x wird dabei als eine um n_x Bits nach rechts verschobene Ganzzahl X interpretiert: $x = X \cdot 2^{-n_x}$. Die Multiplikation und Addition von Festkommazahlen wird auf die entsprechenden Operationen auf den Ganzzahlen mit Normierungsshifts zurückgeführt:

$$z = x \cdot y = X \cdot 2^{-n_x} \cdot Y \cdot 2^{-n_y} = Z \cdot 2^{-n_z} \quad \Leftrightarrow \quad Z = X \cdot Y \cdot 2^{n_z - n_x - n_y}$$
$$z = x + y = X \cdot 2^{-n_x} + Y \cdot 2^{-n_y} = Z \cdot 2^{-n_z} \quad \Leftrightarrow \quad Z = X \cdot 2^{n_z - n_x} + Y \cdot 2^{n_z - n_y}$$

Für die digitale Repräsentation werden alle Spannungswerte auf den Maximalwert VCC normiert. In der Festkommadarstellung entfallen dadurch die Stellen vor dem Komma. Das `SCALED`-Signal ist bspw. $SAMPLE\_WIDTH$ Bits breit und trägt den Wert $\frac{V_{\text{scaled}}}{VCC} \cdot 2^{SAMPLE\_WIDTH}$ (sobald eine Messung abgeschlossen ist). Überlegen Sie sich, welche Vereinfachung von Gleichung 1 durch diese Normierung erreicht werden kann.

Das Verfahren zur Abschätzung von V_{compare} nach Gleichung 1 ist iterativ, da die Spannungsänderung innerhalb eines Taktes vom aktuellen Wert abhängt. Um das Aufakkumulieren von Rundungsfehlern zu reduzieren, soll die ADC_CONTROL-interne Repräsentation von V_{compare} mit einer Genauigkeit von $SAMPLE\_WIDTH + RC\_ACCURACY$ Bits arbeiten. Die Laufzeit-Konstante $e^{-\Delta t/\hat{R}_{\text{M}}\hat{C}_{\text{M}}}$ soll mit derselben Genauigkeit repräsentiert und aus den Parametern TAU und $UPDATE\_DIV$ abgeleitet werden.

Das `CHARGE`-Signal kann aus verschiedenen Gründen nicht direkt mit dem `COMPARE`-Signal des Komparators beschaltet werden. Zum einen wird ein Register zwischen den beiden Signalen benötigt, um die Auf- und Entladezeit exakt festzulegen. Außerdem muss es möglich sein, das Angleichen von V_{compare} an V_{scaled} zeitweise zu unterbrechen. Hierdurch wird der Stromverbrauch des Systems gesenkt, wenn die Anwendung gerade keine Auskunft über die Eingangsspannung benötigt. Außerdem ist dies die einzige Möglichkeit, das Messsystem in einen definierten Ausgangszustand zu versetzen, da die initiale Kondensatorladung zum Zeitpunkt des FPGA-Resets nicht bekannt ist.

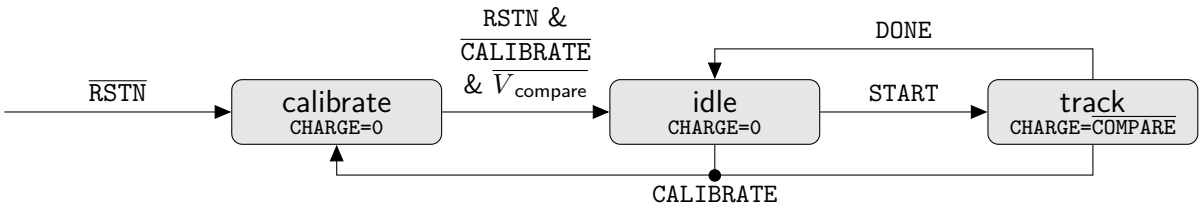

Abbildung 14: Zustandsautomat für Kalibrierung und Messung

Implementieren Sie die in Abbildung 14 dargestellte Kontrolllogik: Nach dem Reset des FPGAs oder der Aktivierung von `CALIBRATE` (synchron, active high) befindet sich das System im Zustand `calibrate`, in welchem der Kondensator permanent entladen wird. In jedem Kalibrierungszyklus wird eine vollständige Entladung des Kondensators nachgestellt. Sobald der (virtuelle) Kondensator vollständig entladen ist, wechselt das System in den `idle` Zustand, in welchem der Kondensator ebenfalls nicht aufgeladen wird. ADC_CONTROL soll dabei selbständig erkennen, wann die Kalibrierung abgeschlossen ist. Das `CALIBRATE`-Signal dient also nur zum Auslösen einer Kalibrierung.

Erst mit der Anforderung einer Messung über das `SAMPLE`-Signal (synchron, active high) wechselt das System in den `track`-Zustand, in welchem die Kondensatorspannung gezielt der Eingangsspannung angenähert wird. Sobald ein gültiger Schätzwert für V_{scaled} gefunden wurde, muss dies der

Anwendung signalisiert werden. Setzen Sie hierfür das DONE-Signal für genau einen Takt auf 1. Das System wechselt dabei wieder zurück in den idle-Zustand, bis der nächste Messwert angefordert wird.

Es bleibt noch zu klären, wann ein Schätzwert als gültig einzustufen ist. Berechnen Sie dazu den gleitenden Mittelwert über die jeweils letzten 2^{AVG_SIZE} Werte von SCALED. Sobald SCALED diesen Mittelwert erreicht bzw. über- oder unterschreitet, wird die Schätzung als gültig angenommen. Die Mittelwertberechnung soll mit jedem track-Zyklus neu gestartet werden und eine konstante Laufzeitkomplexität aufweisen.

PGA_CONTROL: <PrakTIES>/<hdl>/pga_control.v[hd]
Implementieren Sie die SPI-Ansteuerung des LMP8100 im PGA_CONTROL-Modul, *ohne* den CoreSPI aus dem Libero Core-Katalog oder einen anderen vorgefertigten SPI-Core einzubinden. Beachten Sie insbesondere die Timing-Anforderungen des PGAs. Der serielle Takt SCK am PGA soll 4 MHz betragen.

Das von der Anwendung vorgegebene GAIN-Signal repräsentiert den gewünschten Verstärkungsfaktor G im LMP8100-kompatiblen Format ($\texttt{GAIN} = G - 1$). Sorgen Sie dafür, dass jede Änderung von GAIN an den PGA übertragen, dabei aber $G = MAX\_GAIN$ nicht überschritten wird. Konfigurieren Sie den PGA für maximale Bandbreiten-Kompensation (vgl. Tabelle 3 in [16]).

Mit dem aktuell *im PGA* konfigurierten Verstärkungsfaktor muss nun noch die unverstärkte Eingangsspannung UNSCALED aus SCALED berechnet werden. Verwenden Sie hierfür ausschließlich kombinatorische Logik. Um die durch die analoge Vorverstärkung gewonnene Genauigkeit nicht wieder zu verlieren, ist UNSCALED vier Bit breiter als SCALED. Beachten Sie, dass der PGA die Abweichung von V_{in} vom V^{ADC}-Mittelwert *VREF* verstärkt.

APPLICATION: <PrakTIES>/<hdl>/application.v[hd]
Die Anwendung nimmt die Laufzeitkonfiguration über das SWITCH-Signal entgegen und spaltet es in GAIN (Bit 3 bis 0) und CALIBRATE (Bit 4) auf.

Daneben erzeugt das APPLICATION-Modul alle *SAMPLE_DIV* DCLK-Takte einen SAMPLE-Impuls (ein Takt high), um eine neue Messung anzustoßen. Sobald der Abschluss der Messung mit dem DONE-Flag signalisiert wurde, soll UNSCALED byteweise (least significant byte first) über die UART-Schnittstelle UTX (<PrakTIES>/<hdl>/utx.v[hd]) ausgegeben werden. Diese nimmt das am D-Signal anliegende Byte entgegen, sobald die beiden Steuersignale WEN (write enable, active low) und RDY (ready, active high) aktiv sind.

TOP: <PrakTIES>/<hdl>/system/top.v[hd]
Setzen Sie nun alle Submodule zum Toplevelmodul zusammen. Leiten Sie die Beschaltung des DIV-Eingangs des CLOCK_CONDITIONING-Moduls vom *UPDATE_DIV*-Parameter ab.

Die Übertragungsgeschwindigkeit (Baudrate, b_r) der UART-Schnittstelle hängt von der Taktrate f des DCLK-Takts und der Beschaltung des b_v BAUD_VALUE-Signals ab: $b_r = \frac{f}{2 \cdot b_v + 16}$. Die Beschaltung von BAUD_VALUE ist daher aus den Parametern *BAUD_RATE* und *UPDATE_DIV* zu bestimmen.

Aufgabe 5.2 Funktionale Verifikation

Implementieren Sie einen UART-Empfänger URX in <PrakTIES>/<hdl>/urx.v[hd], der aus seinem seriellen Eingang RX das übertragene BYTE ermittelt. Ein zusätzliches VALID-Flag soll während

der Übertragung eines Bytes auf 0 gesetzt werden und erst nach Abschluss der Übertragung auf 1 schalten. Verwenden Sie für die Implementierung dieses Moduls *keinen* vorgefertigten UART-Core. Dieses Test-Modul muss *nicht* synthetisierbar sein.

Weisen Sie nun jedem Toplevel-Port des Messystems einen FPGA-Pin zu, indem Sie die fehlenden Angaben in `<PrakTIES>/constraints/system.pdc` ergänzen. Die UART-Ausgabe `TX` soll über den Mini-USB-Port J2 erfolgen. Beachten Sie dabei auch die Randbedingungen, die durch Ihren Prototyp aus Aufgabe 3 vorgegeben werden.

Setzten Sie *OPAMP_TYPE* auf 0. Führen Sie nun die Schritte ① bis ⑥ des HDL-Werkzeugflusses für `<project> = system` aus, bis alle Simulationen ohne Fehlermeldung durchlaufen werden.

Aufgabe 5.3 Parameter-Optimierung

Die Vergrößerung von *SAMPLE_WIDTH* und *RC_ACCURACY* erhöht die Messgenauigkeit, aber auch den Ressourcen-Bedarf des Messsystems. Durch den endlichen Umschaltbereich des Komparators ist die erreichbare Messgenauigkeit aber begrenzt. In dieser Aufgabe sollen daher sinnvolle Obergrenzen für die Bitbreiten der ermittelten Abtastwerte bestimmt werden.

Setzen Sie zunächst die Konfigurationsparamter *OPAMP_VFH*, *OPAMP_VFL*, *OPAMP_VRH* und *OPAMP_VRL* entsprechend der Hysterese-Kennwerte, die Sie in Aufgabe 1.6 für den LVDS-Komparator bestimmt haben. Setzen Sie ferner *TAU* auf die in Aufgabe 4.2 ermittelte Zeitkonstante sowie *VCC* und *VREF* auf die in Aufgabe 3.4 bestimmten Werte.

Überlegen Sie sich ein sinnvolles Vorgehen bei der Variation von *SAMPLE_WIDTH* und *RC_ACCURACY*. Führen Sie für jeden Parametersatz eine Presynthese-Simulation durch und notieren Sie den minimalen und maximalen Fehler sowie den mittleren Fehler (rms) für die vier simulierten Kombinationen aus `GAIN` und `VIN`. Stellen Sie den Zusammenhang zwischen diesen Fehlern und den variierten Konfigurationsparametern graphisch dar. Welche Kombination von *SAMPLE_WIDTH* und *RC_ACCURACY* führt zu einer hohen Genauigkeit bei niedrigem Ressourcenbedarf?

Aufgabe 5.4 Synthese und Layout

Setzen Sie:

- *OPAMP_TYPE* = -1,
- *SAMPLE_WIDTH* und *RC_ACCURACY* auf die in Aufgabe 5.3 bestimmten Werte und
- *SAMPLE_DIV* = 5000

Führen Sie nun die Schritte ②,⑤,⑦ und ⑧ des HDL-Werkzeugflusses durch. Stellen Sie die Post-Synthese und Post-Layout Statistiken für Ressourcenverbrauch und erreichbare Taktraten an `CLK` und `DCLK` gegenüber. Welche Operationen bestimmen die kritischen Pfade der beiden Taktsignale?

Aufgabe 6 Evaluation des Messsystems

In dieser Aufgabe soll nun abschließend die Funktionalität und die Genauigkeit des entwickelten ADCs untersucht werden.

Aufgabe 6.1 Auslesen der Abtastwerte

Zum Auslesen der vom FPGA über die UART-Schnittstelle ausgegebenen `UNSCALED`-Samples müssen die einzelne Bytes von der COM-Schnittstelle des PCs gelesen und in entsprechende Spannungswerte umgerechnet werden. Die Laufzeitbibliotheken der meisten Programmiersprachen stellen die notwendige Funkionalität zum Auslesen der COM-Schnittstelle bereit [19, 20, 21].

Implementieren Sie ein Skript `<PrakTIES>/read.*` in der Programmiersprache Ihrer Wahl, welches folgende Kommandozeilenparameter entgegen nimmt:

- *COM* – der auszulesende COM-Port
- *BAUD* – die Baudrate de seriellen Schnittstelle
- *SAMPLES* – die Anzahl der zu lesenden Samples
- *BITS* – die Anzahl der Bits pro Sample
- *VCC* – die Spannung, auf welche die digitalen Samples normiert wurden

Neben den ausgelesenen Spannungswerten soll das Skript noch den Mittelwert und die Stichprobenstandardabweichung der Abtastwerte ausgeben.

Aufgabe 6.2 Statische und dynamische Messungen

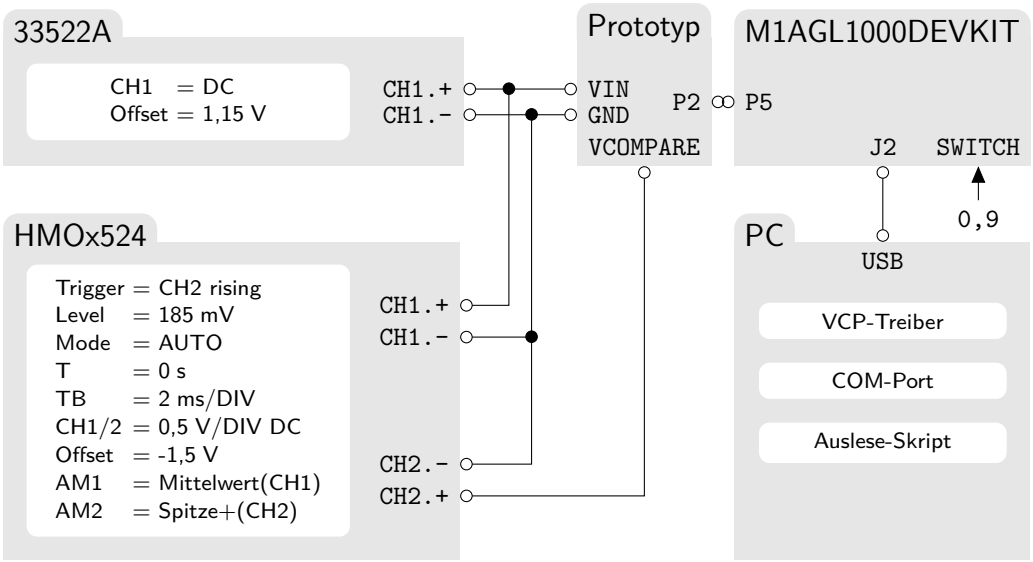

Abbildung 15: Versuchsaufbau – Genauigkeit des ADC

Installieren Sie den Virtual COM Port Treiber [22], welcher die serielle Schnittstelle des FPGAs als COM-Port bereit stellt. Verbinden Sie nun den Mini-USB-Port J2 des FPGA-Evaluationsboards mit einem USB-Port des PCs. Bei korrekt installierten Treibern wird das Gerät als `Silicon`

`Labs CP210x USB to UART Bridge` erkannt und der zugeordnete COM-Port im Gerätemanager angezeigt.

Realisieren Sie den in Abbildung 15 gezeigten Versuchsaufbau. Dabei wird die zu vermessende Eingangsspannung mit dem Funktionsgenerator erzeugt und am CH1 des Oszilloskops angezeigt. Der Prototyp des ADC-Erweiterungsboards wird an das FPGA-Evaluationsboard angeschlossen. Zur Veranschaulichung der Funktionsweise wird die Spannung über dem Messkondensator am CH2 des Oszilloskops angezeigt.

Erstellen Sie für alle Kombinationen aus $V_{\text{in}} \in \{1{,}15\ \text{V}; 1{,}35\ \text{V}\}$ und `SWITCH` $\in \{0, 9\}$ Screenshots am Oszilloskop und erfassen Sie jeweils 1000 Samples mit ihrem Auslese-Skript aus Aufgabe 6.1. Geben Sie den Mittelwert und die Stichprobenstandardabweichung der Samples für alle vier Messungen an und stellen Sie die Streuung der Messwerte als Histogramme dar.

Konfigurieren Sie nun eine sinusförmige Eingangsspannung (Offset = 1.25 V, Amplitude = 200 mV, Frequenz = 1 Hz) am Funktionsgenerator ein. Fertigen Sie für `SWITCH` $\in \{0, 9\}$ zwei Screenschots bei einer Zeitauflösung von 100 ms/DIV an und erfassen Sie jeweils 1000 Samples mit Ihrem Auslese-Skript. Stellen Sie die beiden gemessenen Wertereihen zusammen mit der erwarteten (idealen) V_{in}-Kurve in einem Diagramm dar. Gleichen Sie hierfür die Phasenverschiebung der gemessenen Kurven an die Referenz-Kurve an.

Literatur

[1] David Kessner. *ADC In An FPGA*. 2011. URL: http://davidkessner.wordpress.com/2011/05/01/adc-in-an-fpga (besucht am 12. 04. 2014).

[2] Microsemi. *IGLOO Low Power Flash FPGAs with Flash Freeze Technology*. 2012. URL: http://www.actel.com/documents/IGLOO_DS.pdf (besucht am 12. 04. 2014).

[3] Microsemi. *IGLOO FPGA Fabric User's Guide*. 2012. URL: http://www.actel.com/documents/IGLOO_UG.pdf (besucht am 12. 04. 2014).

[4] TUCaN. *20-00-0647-pr Praktikum zu Technischer Informatik*. 2014. URL: https://www.tucan.tu-darmstadt.de/scripts/mgrqcgi?APPNAME=CampusNet&PRGNAME=COURSEDETAILS&ARGUMENTS=-N000000000000001,-N000334,-N0,-N346324795549005,-N346324795585006,-N0,-N0,-N0 (besucht am 12. 04. 2014).

[5] Moodle. *PrakTIES*. 2012. URL: https://moodle.informatik.tu-darmstadt.de/course/view.php?id=221 (besucht am 12. 04. 2014).

[6] Hameg. *250/350 MHz Digital Oszilloskop HMO Serie*. 2012. URL: http://www.hameg.com/manuals.0.html?&L=1&tx_hmdownloads_pi1[mode]=download&tx_hmdownloads_pi1[uid]=5115 (besucht am 12. 04. 2014).

[7] Agilent. *Wellenformgenerator der Reihe 33500*. 2012. URL: http://cp.literature.agilent.com/litweb/pdf/33500-90911.pdf (besucht am 12. 04. 2014).

[8] Microsemi. *ARM Cortex-M1-Enabled IGLOO Development Kit User's Guide*. 2012. URL: http://www.actel.com/documents/M1IGLOO_DevKit_UG.pdf (besucht am 12. 04. 2014).

[9] SoC Solutions. *IGLOO Development Kit Schematic*. 2009. URL: http://www.actel.com/download/rsc/?f=M1AGL_DEV_KIT_SCS_SS (besucht am 12. 04. 2014).

[10] Andreas Koch und Wolfgang Heenes. *Grundregeln der wissenschaftlichen Ethik am Fachbereich Informatik*. 2011. URL: www.informatik.tu-darmstadt.de/plagiarism (besucht am 12. 04. 2014).

[11] Microsemi. *Libero SoC v11.3*. 2014. URL: http://www.microsemi.com/products/fpga-soc/design-resources/design-software/libero-soc (besucht am 12. 04. 2014).

[12] Thomas Schaerer. *Operationsverstärker I*. 2002. URL: http://www.elektronik-kompendium.de/public/schaerer/opa1.htm (besucht am 12. 04. 2014).

[13] PartSim. *Circuit Simulation Made Easy*. URL: http://www.partsim.com/ (besucht am 12. 04. 2014).

[14] ubuntuusers. *Oregano*. 2014. URL: http://wiki.ubuntuusers.de/Oregano (besucht am 12. 04. 2014).

[15] NGSpice. *Mixed Mode / Mixed Level Circuit Simulator*. URL: http://ngspice.sourceforge.net/ (besucht am 12. 04. 2014).

[16] Texas Instruments. *LMP8100 Programmable Gain Amplifier*. 2008. URL: http://www.ti.com/lit/ds/symlink/lmp8100.pdf (besucht am 12. 04. 2014).

[17] Linear Technology. *LT6656 1 µA Precision Series Voltage Reference*. 2008. URL: http://cds.linear.com/docs/en/datasheet/6656fc.pdf (besucht am 12. 04. 2014).

[18] Cadsoft. *EAGLE PCB Design Software*. 2014. URL: http://www.cadsoft.de/ (besucht am 12. 04. 2014).

[19] Jonas Bähr u. a. *Ruby/SerialPort*. 2008. URL: http://ruby-serialport.rubyforge.org/ (besucht am 12. 04. 2014).

[20] Oracle. *Class SerialPort*. URL: http://docs.oracle.com/cd/E17802_01/products/products/javacomm/reference/api/javax/comm/SerialPort.html (besucht am 12. 04. 2014).

[21] Microsoft. *SerialPort-Klasse*. 2014. URL: http://msdn.microsoft.com/de-de/library/system.io.ports.serialport.aspx (besucht am 12. 04. 2014).

[22] Silicon Labs. *CP210x USB to UART Bridge VCP Drivers*. 2014. URL: http://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx (besucht am 12. 04. 2014).