Karlsruher Institut für Technologie

Informationen und Anleitung

14. 01. 2023

Digitaltechnik - Challenges auf Xilinx Zedboard

Institut für Technik der Informationsverarbeitung
Prof. Dr.-Ing. Dr. h.c. Jürgen Becker
M.Sc. Julian Höfer

Wintersemester 22/23

Überblick

Die im Logiksimulator erstellten Schaltungen lassen sich auch auf Hardware testen. Wir nutzen ein Xilinx Zedboard, dessen zentraler Baustein ein sogenannter FPGA ist. Was in den einzelnen Schritten passiert und wie alles mit den in der Vorlesung Digitaltechnik vermittelten Themen zusammenhängt wird im Folgenden kurz erklärt.

Wo? Aufgebaut und verfügbar ist das Zedboard im ITIV-Poolraum 216, im abgetrennten Bereich im hinteren Teil des Raumes. Bei geschlossener Tür, einfach klingeln oder klopfen oder (auch bei Fragen) gerne kurz beim Übungsleiter vorbeischauen (Büro 127). Der für die Implementierung benötigte *Bitstream* lässt sich bereits von Zuhause generieren.

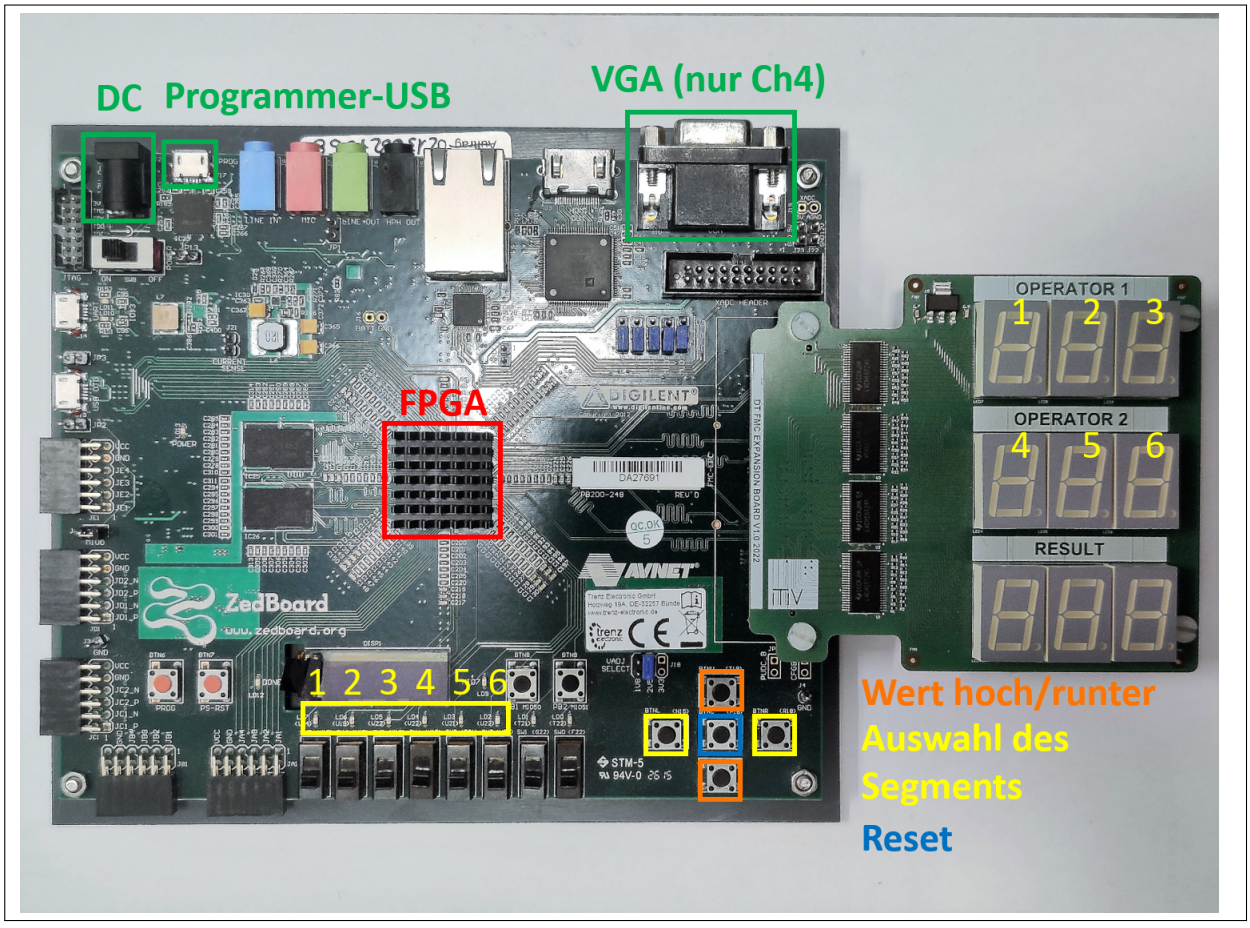


Abbildung 1 Übersicht über das Zedboard und die Steuerung

Beim Xilinx Zedboard handelt es sich um ein Entwicklungsboard für verschiedene Einsatzgebiete, von Videoverarbeitung über beschleunigte Software bis hin zu SoC Prototyping. Dafür sind zahlreiche Schnittstellen verfügbar. Für die Rechnerchallenges (2 und 3) wird die FMC-Schnittstelle benutzt, um das Erweiterungsboard mit den 7-Segment-Anzeigen anzusteuern. In Challenge 4 kommt die VGA-Schnittstelle zum Einsatz, um eine Bildschirmausgabe zu ermöglichen.

Für die Rechnerchallenges können die Buttons wie gekennzeichnet verwendet werden. Das aktuell ausgewählte Eingabesegment wird durch eine der 6 LEDs kenntlich gemacht.

Was ist eine Hardwarebeschreibungssprache? In den DT-Challenges werden Logikschaltungen mithilfe einer graphischen Oberfläche und Gattern entworfen. Für komplexere Schaltungen ist das jedoch nicht mehr möglich. Hier kommen abstraktere Hardwarebeschreibungssprachen wie *Verilog* oder *VHDL* zum Einsatz, welche auf Registerebene (Register-Transfer-Level = RTL) arbeiten. Diese liefern einen Standard, mit welchem sich beliebige Hardwarearchitekturen beschreiben und gleichzeitig modellieren lassen. Das hat den Vorteil, dass sich das Zeitverhalten der Schaltung simulieren lässt und mithilfe von Testbenches eine funktionale und zeitliche Verifikation erfolgen kann.

Im Verlauf des Studiums werden Hardwarebeschreibungssprachen am ITIV in der Vorlesung *Hardware Modeling and Simulation - HMS* sowie im Praktikum *Digital Hardware Design Laboratory - DHL* vertieft und angewendet.

Was ist Hardware-Synthese? Als Hardware-Synthese wird der Schritt von der Hardwarebeschreibung zur Implementierung bezeichnet. Dieser lässt sich, wie in Abbildung 2 dargestellt, vereinfacht in zwei Stufen einteilen:

Die Logiksynthese generiert zunächst eine Netzliste. In der Vorlesung lernen wir die Minimierung kennen, um die Anzahl der Gatter zu reduzieren. Diese wird in diesem Schritt durchgeführt, natürlich automatisiert und rechnergestützt und ergänzt durch weitere Optimierungen.

Diese Netzliste wird dann in der zweiten Stufe ebenfalls automatisiert auf die Zieltechnologie abgebildet, welche meist ASICs oder FPGAs sind. Als ASIC wird mit *Application Specific Integrated Circuit* eine integrierte Schaltung bezeichnet, welche ”fest verdrahtet” ist und meist aus CMOS-Standardzellen besteht. Diese CMOS-Gatter werden im letzten Kapitel der Vorlesung eingeführt.

In den DT Challenges wird die Hardware-Synthese für den FPGA des Zedboard abstrahiert durch den DT Bitstream Generator durchgeführt. Weitere Details zu den Prozessschritten der Hardware-Synthese werden am ITIV in der Vorlesung *Hardware-Synthese und -Optimierung* vertieft. Die zum Einsatz kommenden Tools werden in den Praktika *Digital Hardware Design Laboratory - DHL* und *Praktikum System-on-Chip - PSOC* angewendet.

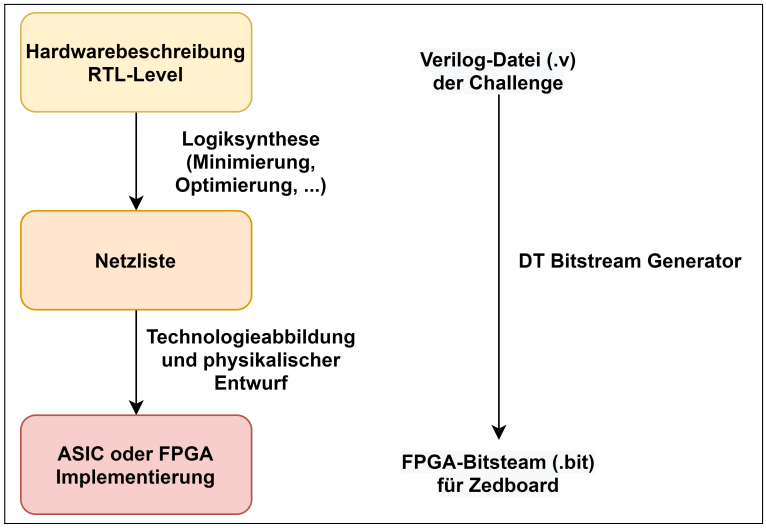


Abbildung 2 Allgemeiner Prozess der Hardware-Synthese und Umsetzung in den Challenges

Was ist ein FPGA? FPGA bedeutet *Field Programmable Gate Array* und wird auch häufig als programmierbare Logik bezeichnet. Im Gegensatz zu den fest verdrahteten ASICs handelt es sich bei FPGAs also um Bausteine, die durch Rekonfiguration nicht nur eine feste, sondern verschiedene Schaltfunktionen realisieren können. Der Hauptbestandteil eines FPGAs sind zehntausende bis hunderttausende von Multiplexern, deren ”Verdrahtung” untereinander programmiert werden kann. Häufig werden die Multiplexer zu sogenannten ”Look-Up Tables” (LUTs) erweitert. Dabei werden SRAM-Speicherzellen an den Eingängen der Multiplexer hinzugefügt.

Mit diesen Multiplexerschaltungen können beliebige Logikschaltungen schnell implementiert werden. Die Idee basiert auf dem in der Vorlesung eingeführten Entwicklungssatz der Schaltalgebra oder *Shannon-Expansion*. Dieses Verfahren liefert eine Vorschrift, wie eine beliebige Logikschaltung mithilfe von Multiplexern umgesetzt wird. Auf dieser Vorschrift basieren die Algorithmen der Technologieabbildung für FPGAs und werden automatisch und rechnergestützt durchgeführt.

1. Schritt: Generieren des Bitstreams (von Zuhause)

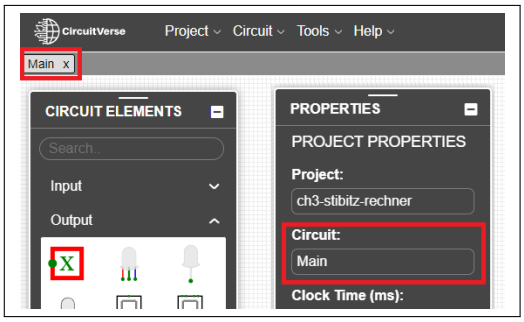


Abbildung 3 Die Hauptschaltung muss *Main* genannt werden

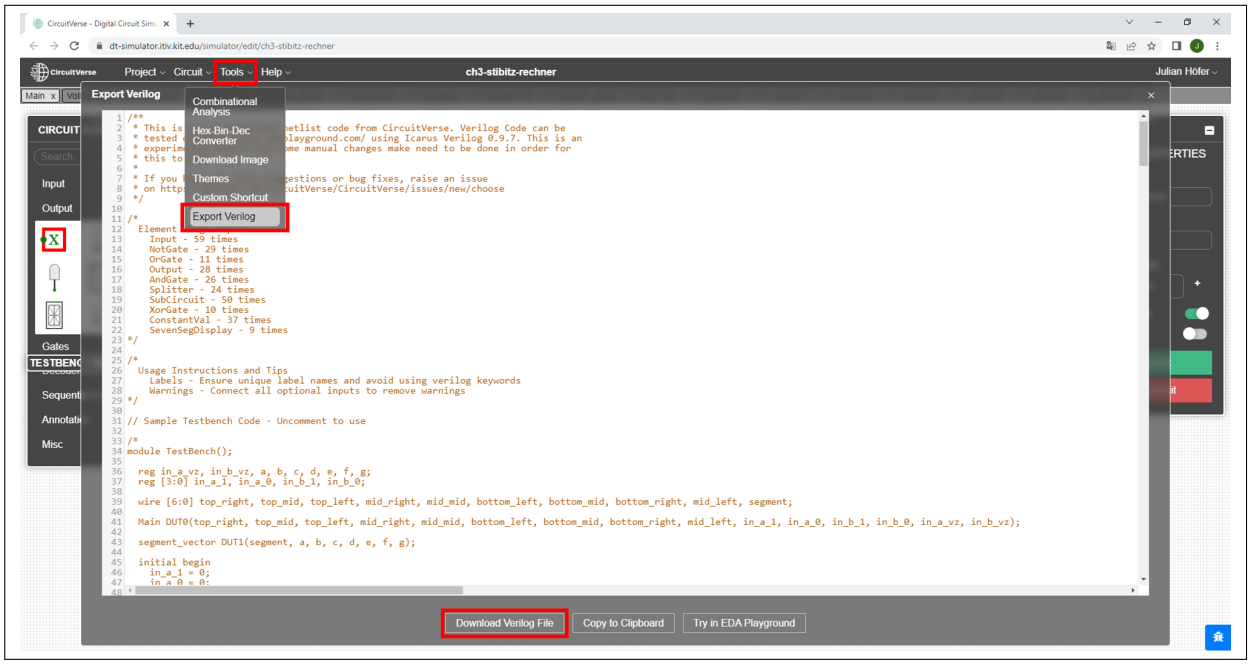


Abbildung 4 Herunterladen des Verilog-Codes der Schaltung als .v-Datei durch *Tools -> Export Verilog -> Download Verilog File*

Wer hier Interesse hat, kann gerne einmal in den Code reinschauen. Verilog bedient sich der Syntax der Programmiersprache C, es handelt sich jedoch nicht um ausführbaren C-Code. Vielmehr wird die Struktur und das Verhalten der Hardware über Module beschrieben. Die einzelnen Module haben fest definierte Schnittstellen zu den anderen Modulen, wobei sich der Systemcharakter gut beobachten lässt: *Eingabe -> Verarbeitung -> Ausgabe*.

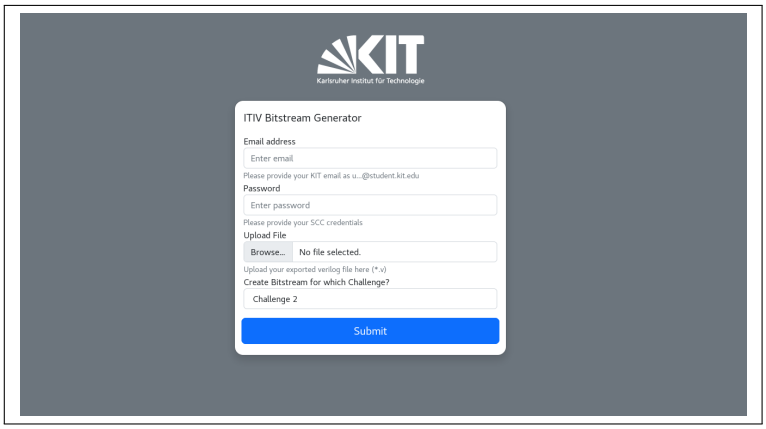


Abbildung 5 Unter dt-simulator.itiv.kit.edu :
Hochladen des Verilog-Codes und Eingabe von E-Mail, Passwort und Auswahl der Challenge. Der fertige Bitstream wird automatisch nach ca. 30 Minuten per E-Mail zugesandt.

Sollte die Bitstreamgenerierung fehlschlagen, wird per E-Mail eine Fehlermeldung zugesandt. In diesem Fall könnte folgendes die Ursache sein (Liste wird fortlaufend aktualisiert):

- Ist die Schaltung als *Main* benannt?
- Sind die Ein- und Ausgänge der Schaltung richtig benannt? Es dürfen keine zusätzlichen Schnittstellen vorhanden sein. Siehe Tabelle oder Blockdiagramm.
- Ist etwas mit *input* oder *output* gelabelt? Das sind reservierte Keywords von Verilog und dürfen daher in Circuitverse nicht verwendet werden, was auch für Subcircuits gilt.
- ...

Sollte keiner der obigen Punkte helfen, die E-Mail gerne an den Übungsleiter weiterleiten und den URL-Link zum Circuitverse Dashboard des Users mit einfügen.

2. Schritt: Ausprobieren der Schaltung (im Poolraum)

Im Poolraum 216 am ITIV (Gebäude 30.10) ist ein Arbeitsplatz eingerichtet. Die notwendigen Anschlüsse sind bereits verbunden, wie in Abbildung 1 dargestellt. Es muss nur noch mithilfe von *Vivado* der Bitstream auf das Board geladen werden. Hierfür startet die Software Vivado startet automatisch nach dem Login. Nun muss der Hardware Manager geöffnet werden *Open Hardware Manager*. Wenn offen, zeigen Abbildung 6 und Abbildung 7 die letzten Schritte. Dann kann losgelegt werden!

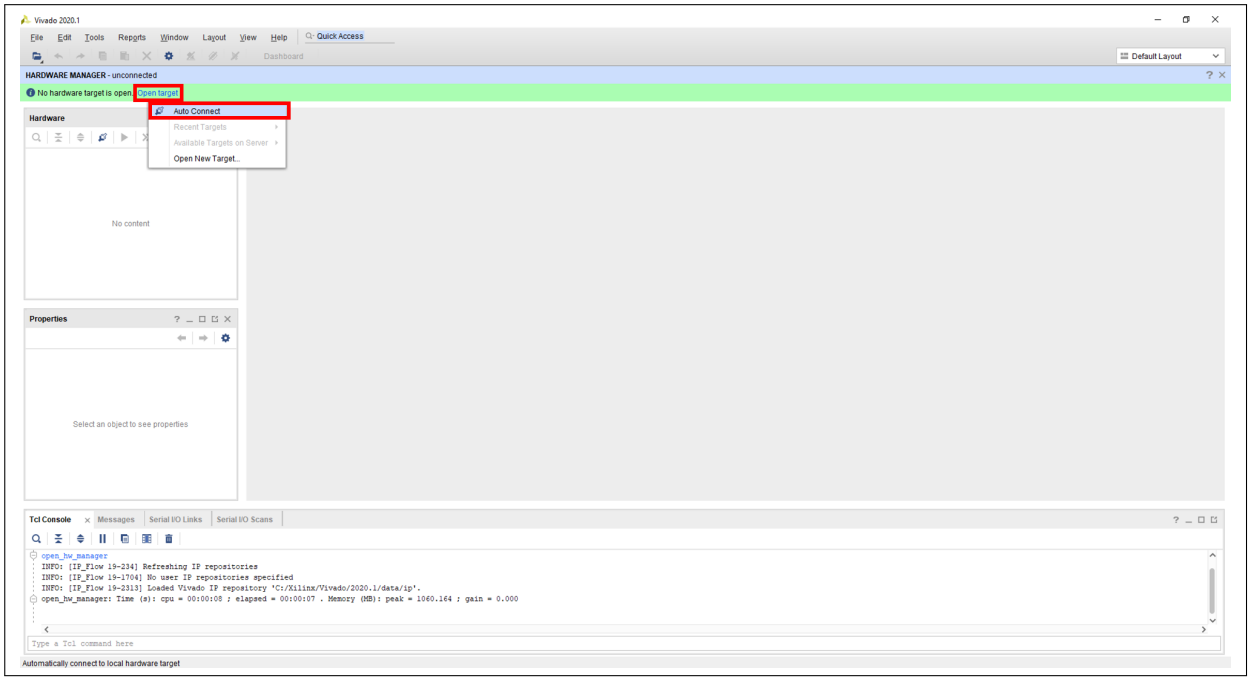


Abbildung 6 Verbinden des Zedboards, falls nicht schon erfolgt
Open Target -> Auto Connect

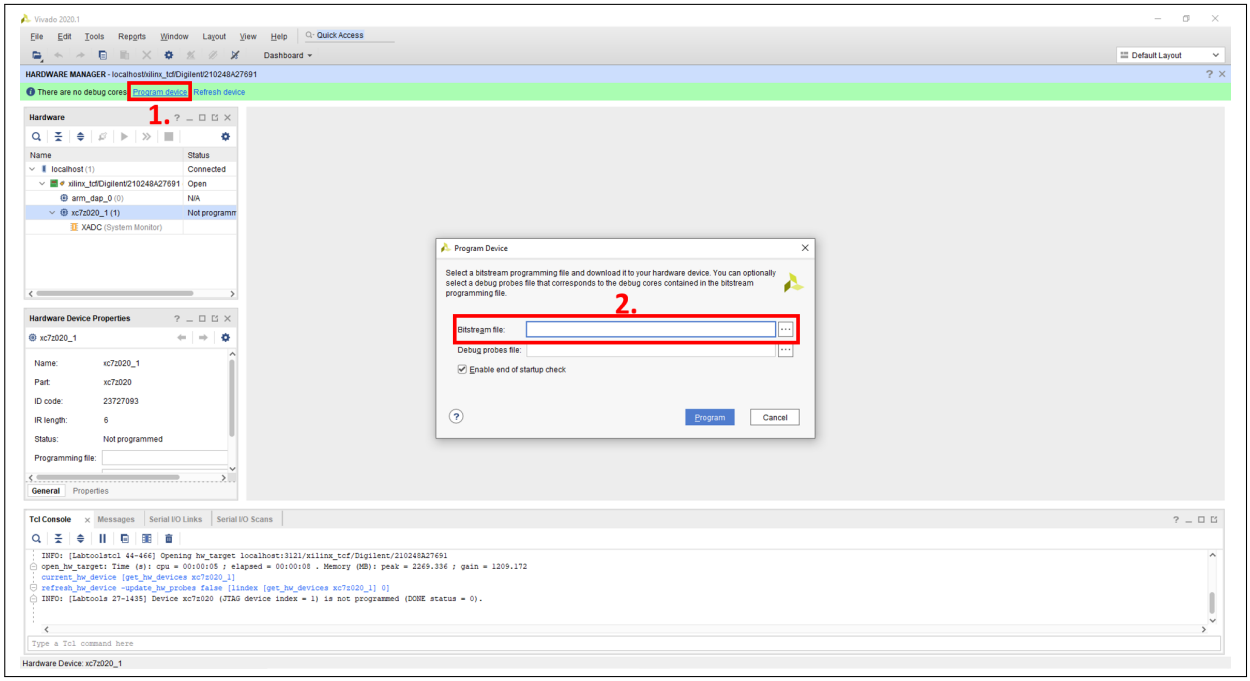


Abbildung 7 Laden des Bitstreams
Program device -> Select Bitstream file -> Program

Blockdiagramm Challenge 2

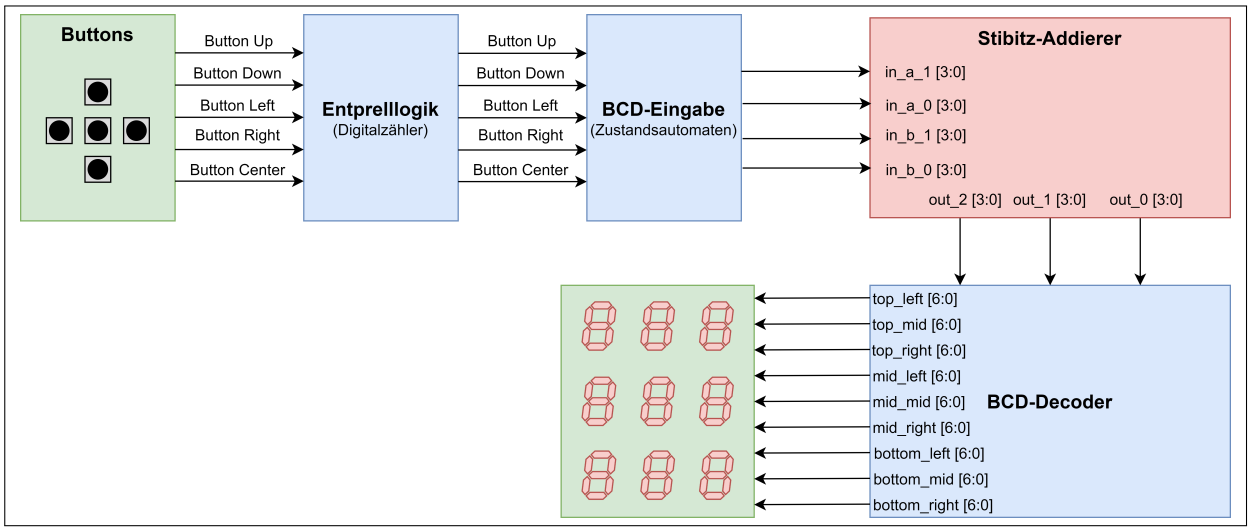


Abbildung 8 Blockdiagramm der zweiten Challenge

Ein- / Ausgabeelement	Richtung	Bitbreite	Label
1. Summand, Einerstelle	in	4	in_a_0
1. Summand, Zehnerstelle	in	4	in_a_1
2. Summand, Einerstelle	in	4	in_b_0
2. Summand, Zehnerstelle	in	4	in_b_1
Summe Einerstelle	out	4	out_0
Summe Zehnerstelle	out	4	out_1
Summe Hunderterstelle	out	4	out_2

Tabelle 1 Benennung der Ein- und Ausgabeelemente

Blockdiagramm Challenge 3

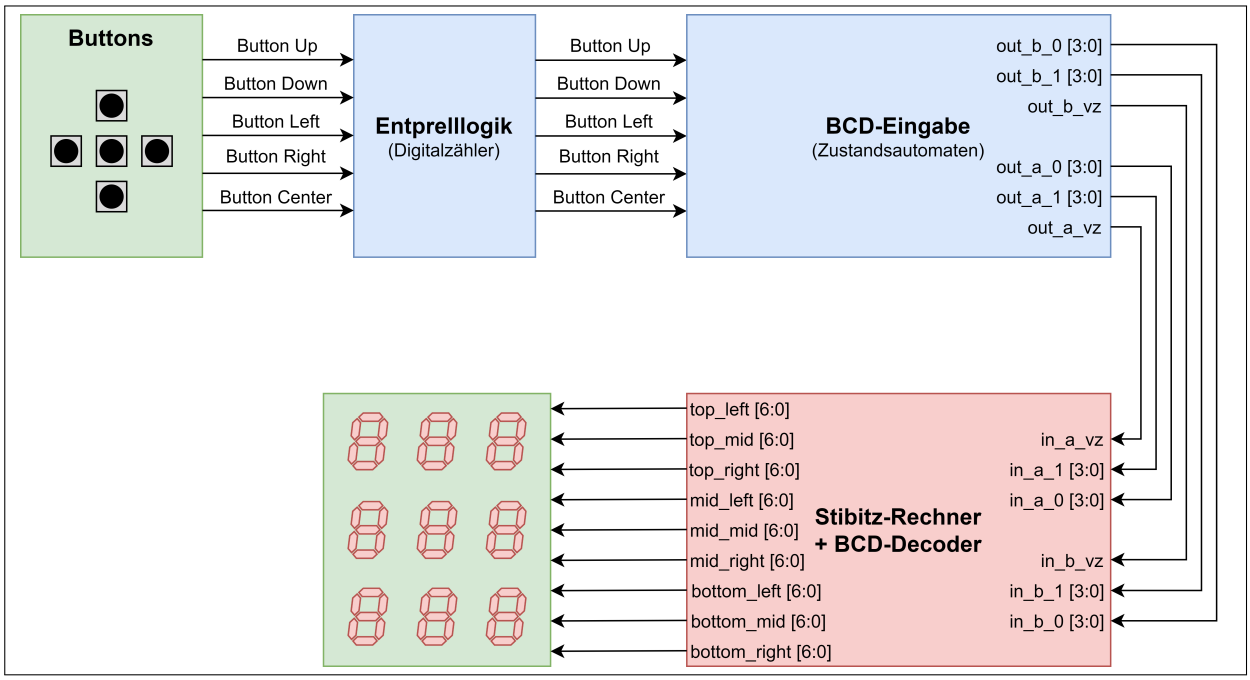
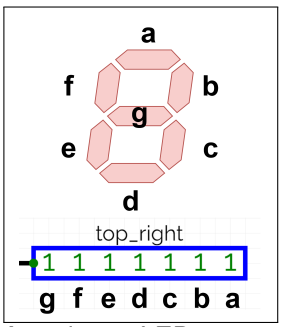


Abbildung 9 Blockdiagramm der dritten Challenge

Ein- / Ausgabeelement	Richtung	Bitbreite	Label
1. Operand, Vorzeichen	in	1	in_a_vz
1. Operand, Einer	in	4	in_a_0
1. Operand, Zehner	in	4	in_a_1
2. Operand, Vorzeichen	in	1	in_b_vz
2. Operand, Einer	in	4	in_b_0
2. Operand, Zehner	in	4	in_b_1
7-Segment oben links	out	7	top_left
7-Segment oben mitte	out	7	top_mid
7-Segment oben rechts	out	7	top_right
7-Segment mitte links	out	7	mid_left
7-Segment mitte mitte	out	7	mid_mid
7-Segment mitte rechts	out	7	mid_right
7-Segment unten links	out	7	bottom_left
7-Segment unten mitte	out	7	bottom_mid
7-Segment unten rechts	out	7	bottom_right



Anordnung LEDs

Abbildung 10 Benennung der Ein- und Ausgabeelemente