

# 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 *Hard-ware 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 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*.

<b>SKIT</b>	
Karleuber Institut für Technologie	
Email address	
Enter email	
Please provide your KIT email as u@student.kit.edu Password	
Enter password	
Please provide your SCC credentials Upload File	
Browse No file selected.	
Upload your exported verilog file here (*.v) Create Bitstream for which Challenge?	
Challenge 2	
Submit	

#### 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!

A Vivedo 2020.1	-	σ	×
Eile Edit Tools Reports Window Layout View Help 🕰 Quick Access			
Carlos Ar	I Default Lay	out	~
HARDWARE MANAGER - unconnected			? ×
No hardware target is open     Open target			
Image: Second Targets       Image: Second Targets         Image: Second Targets       Image: Second Targets         No context       Image: Second Targets         No context       Image: Second Targets         Image: Second Targets       Image: Second Targets         Second Targets       Image: Second Targets         Image: Second Targets       Image: Second Targets <t< th=""><th></th><th></th><th></th></t<>			
Td Censole x Messages Serial 10 Laks Serial 10 Cases		? _ 🗆	Ľ
<pre>() open_by_manager INTO: [17_ray i5-301] Refreshing IF repositories specified INTO: [17_ray i5-101] Joade Ti Prepositories specified INTO: [17_ray i5-201] Joade Ti Prepositories appendent () open_by_manager: Time (s): que = 00100107 : Ameney (00): past = 1040.144 ; gain = 0.000 () () () () () () () () () () () () ()</pre>		>	Ĵ
Automatically connect to local hardware target			_

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

A Vivado 2020.1		- 0 ×
File Edit Tools Reports Window Layout View Help Q. Quick Access		
S A A B S X O X Dashboard -		III Default Layout V
HARDWARE MANAGER - localhost/vilinx_td/Dioilent/210248A27691		? ×
There are no debug cores Program device Refresh device		
1		
Hardware C X		
Name Status V Localhost (1) Connected		
✓ ■		
@ arm_dap_0 (0) N/A		
W xc7z020_1 (1) Not programm     XADC (Contem Manifest)		
ANDC (System Monitor)		
	A Program Device X	
	Select a bitstream programming file and download it to your hardware device. You can optionally	
<	programming file.	
Hardwara Davica Dronartiae 2 D 14 V	Ζ.	
	Bitstream file:	
	Debug probes file:	
Name: xc7z020_1	Enable end of startup check	
Part: xc7z020		
ID code: 23727093		
IR length: 6	(?) Erogram Cancel	
Status: Not programmed		
Programming file:		
<		
General Properties		
Tcl Console × Messages Serial I/O Links Serial I/O Scans		? _ 0 6
INFO: [Labtoolstcl 44-466] Opening hw_target localhost:3121/xilinx_tcf/Digi:	ent/210248827691	
☐ open_hw_target: Time (s): cpu = 00:00:05 ; elapsed = 00:00:08 . Memory (MB): current hw device [get hw devices xc7z020 1]	peak = 2269.336 ; gain = 1209.172	
□ refresh_hw_device -update_hw_probes false [lindex [get_hw_devices xc7z020_1] ○ THEOL [Lindex] 27-1425] Device xc7z020_1[JIIG devices today = ]] is not normally a set of the set of	0]	
Into. [Intotal 2/-1455] Device Kontozo (dike device index = 1) is not program.	anne and an and a states - vy.	l I
<		×
Type a Tcl command here		
Hardware Device: xc7z020_1		

Abbildung 7 Laden des Bitstreams

Program device -> Select Bitstream file -> Program

## Blockdiagramm Challenge 2

Buttons	Button Up Button Down Button Left Button Right Button Center	<b>Entprelllogik</b> (Digitalzähler)	Button Up Button Down Button Left Button Right Button Center	BCD-Eingabe (Zustandsautomaten)	Sf in_a_1 [3:0] in_a_0 [3:0] in_b_1 [3:0] in_b_0 [3:0] out_2 [3:0]	1] out_1 [3:0] out_0 [3:0]
				888 888 888	top_left [6:0]           top_mid [6:0]           top_right [6:0]           mid_left [6:0]           mid_right [6:0]           bottom_left [6:0]           bottom_mid [6:0]           bottom_right [6:0]	3CD-Decoder

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



Abbildung 10 Benennung der Ein- und Ausgabeelemente