

Anleitung zur Integration von IPs in Xilinx Zynq SoCs über Vivado unter Peta-Linux

Teil der Projektarbeit 2

Lars Koers

Matrikelnummer 7201353

Abgabedatum: 2. Februar 2023

Aufbau der Anleitung:

Der erste Abschnitt behandelt das Vorgehen zur Erstellung einer IP als Axi4-Lite Schnittstelle in Vivado. Mit Vivado wird ein Block Design erstellt und im Anschluss exportiert. Am Ende des Kapitels liegt eine .xsa-Datei für die nächsten Installationsschritte vor.

Im zweiten Kapitel wird mit der fertigen .xsa-Datei des Hardware Designs die PetaLinux-Installation durchgeführt. Wenn bereits eine .xsa-Datei vorliegt, kann zu Kapitel 2 vorgesprungen werden.

Die erstellten Dateien aus dem Installationsvorgang werden in Kapitel 3 auf einen Datenträger geladen, welcher in zwei Partitionen aufgeteilt wird. Abschließend liegt ein bootfähiges Projekt auf dem Speichermedium vor.

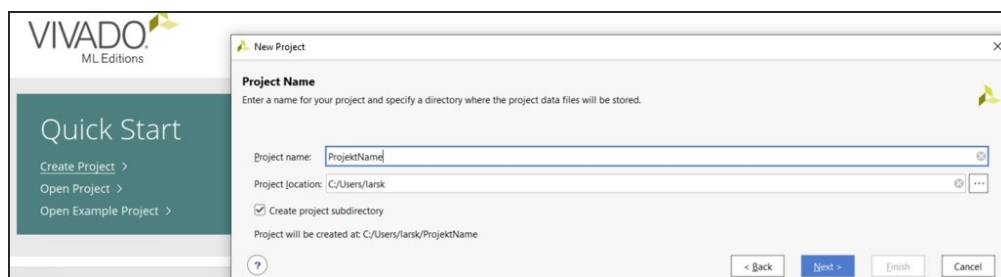
Die beiden abschließenden Kapitel 4 und 5 beinhalten Testoptionen für das Projekt und eine kurze Aufführung von Befehlen bei einer Anpassung des Quellcodes.

1 Integration einer IP mit Hardwaredesign in Vivado

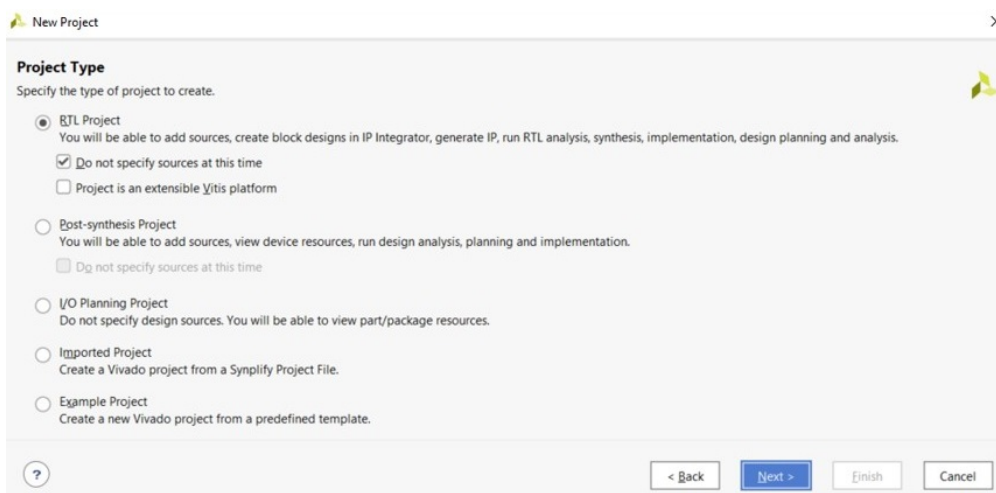
Die Erstellung findet unter Vivado 2022 statt und stellt ein Beispiel zur Erläuterung des Vorgehens dar.

1.1 Create and package new IP

1.1.1 Zu Beginn wird ein neues Projekt in Vivado erstellt und ein spezifischer Name vergeben.



1.1.2 RTL Projekt auswählen und fortfahren.

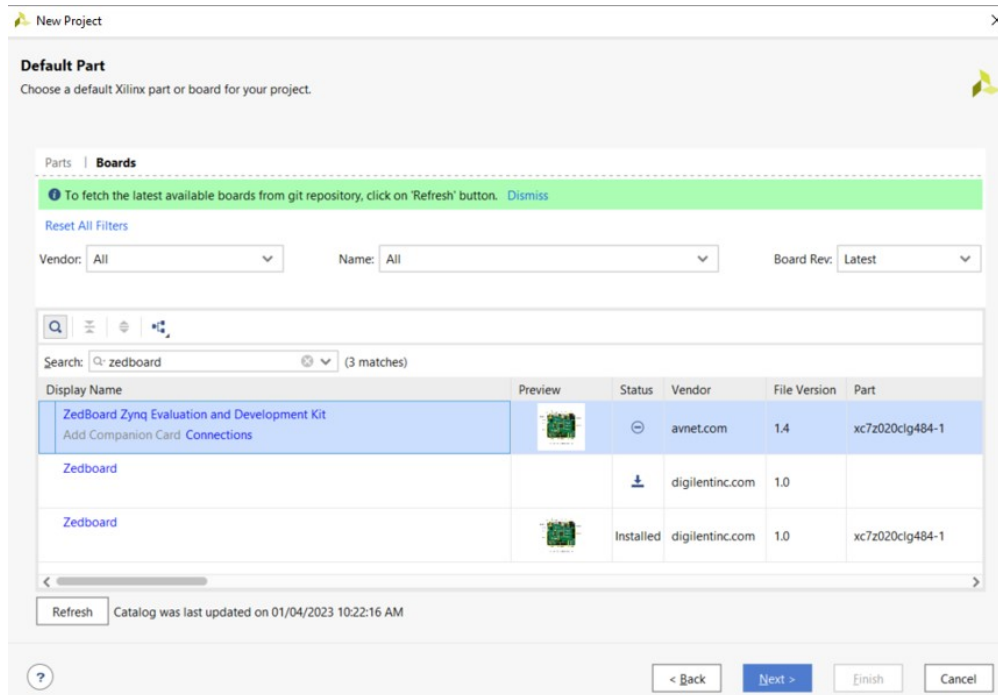


1 Integration einer IP mit Hardwaredesign in Vivado

1.1.3 Board Plattform auswählen.

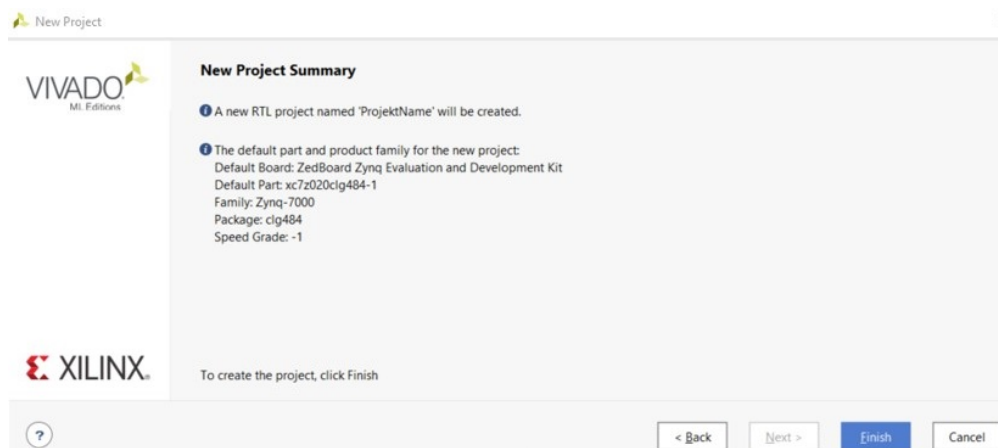
→ !Achtung! bitte aufpassen, dass das richtige Board-file ausgewählt wird.

→ Ggf. sollte der Katalog aktualisiert werden. → Refresh

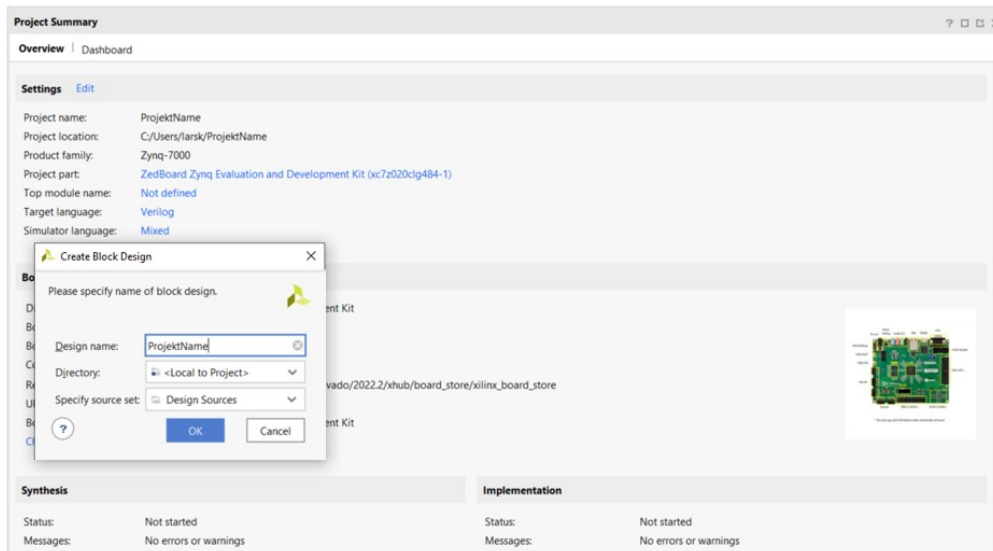


→ Es handelt sich in diesem Beispiel um ein Zedboard: ZedBoard Zynq Evaluation and Development Kit von avnet.com

1.1.4 Mit diesen Einstellungen kann das Projekt erstellt werden



1.1.5 Als nächstes wird ein Block Design erstellt.
→ Create Block Design → Projektnamen vergeben

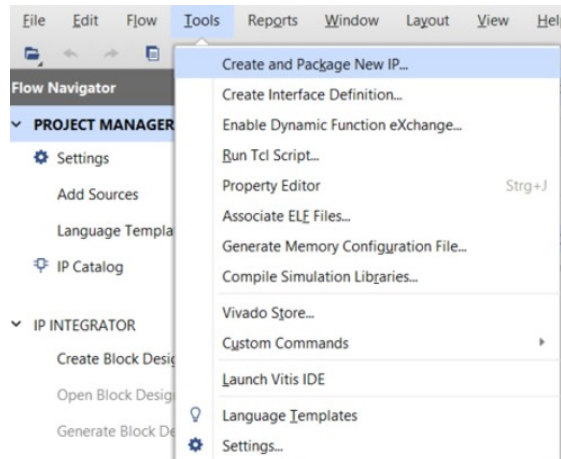


1.1.6 Nun kann theoretisch das Design erstellt werden. Wie eine eigene IP erstellt wird, wird im nächsten Abschnitt beschrieben. Wenn dies nicht benötigt wird, kann zu Abschnitt 1.3 übergegangen werden.

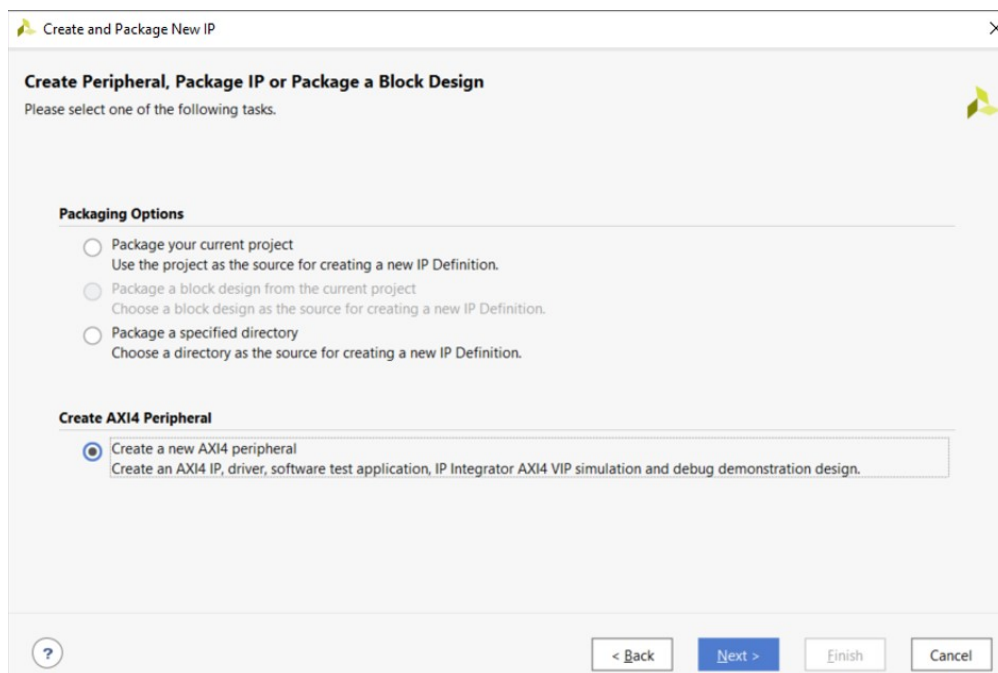
1.2 Erstellung einer IP in Vivado

1.2.1 Bei Bedarf kann eine neue IP erstellt werden. In diesem Beispiel handelt es sich um eine Axi-Lite Schnittstelle.

→ Tools → Create and Package New IP



1.2.2 Erstellen einer Axi-Schnittstelle auswählen



1 Integration einer IP mit Hardwaredesign in Vivado

1.2.3 Vergeben Sie einen sinnvollen Namen und ggf. eine kurze Einzeiler als Info zum Projekt.

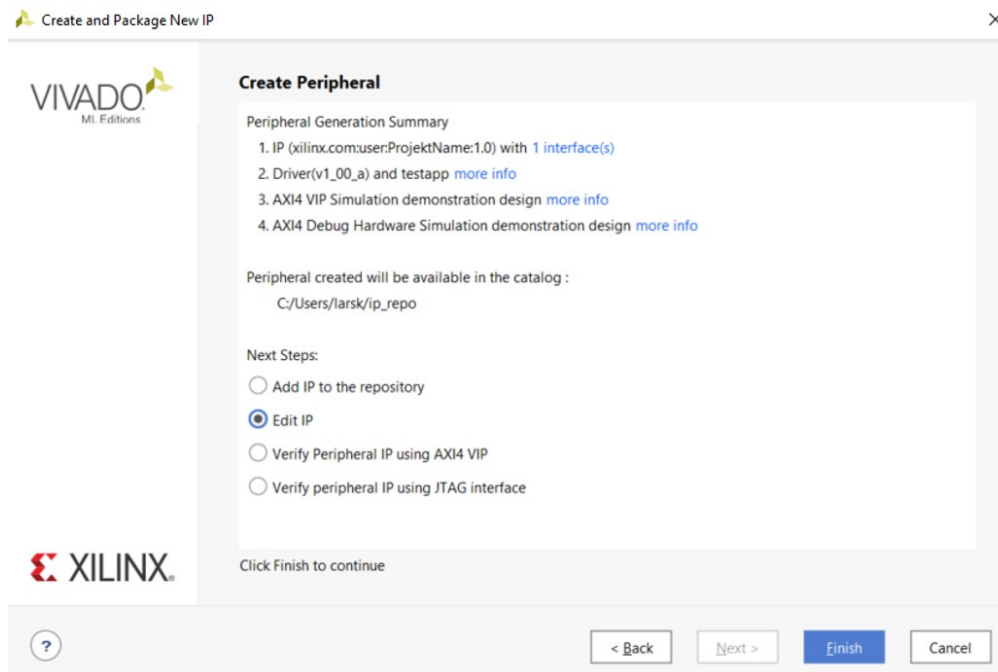
The screenshot shows the 'Peripheral Details' dialog box in Vivado. The title bar reads 'Create and Package New IP'. The main heading is 'Peripheral Details' with a subtitle 'Specify name, version and description for the new peripheral'. The dialog contains several input fields: 'Name' (ProjektName), 'Version' (1.0), 'Display name' (ProjektName_v1.0), 'Description' (Kurze Projekt Info), and 'IP location' (C:/Users/larsk/ip_repo). There is an 'Overwrite existing' checkbox which is unchecked. At the bottom, there are navigation buttons: '< Back', 'Next >', 'Finish', and 'Cancel'.

1.2.4 Die Einstellungen sind entsprechenden der Projektbedürfnisse anzupassen.
→ Schnittstellentyp, Datenbreite, Anzahl der Register

The screenshot shows the 'Add Interfaces' dialog box in Vivado. The title bar reads 'Create and Package New IP'. The main heading is 'Add Interfaces' with a subtitle 'Add AXI4 interfaces supported by your peripheral'. On the left, there is a checkbox 'Enable Interrupt Support' which is unchecked. Below it, a diagram shows a block labeled 'S00_AXI' connected to 'ProjektName_v1.0'. In the center, a tree view shows 'Interfaces' with 'S00_AXI' selected. On the right, the configuration for 'S00_AXI' is shown: 'Name' (S00_AXI), 'Interface Type' (Lite), 'Interface Mode' (Slave), 'Data Width (Bits)' (32), 'Memory Size (Bytes)' (64), and 'Number of Registers' (32) with a range of [4..512]. At the bottom, there are navigation buttons: '< Back', 'Next >', 'Finish', and 'Cancel'.

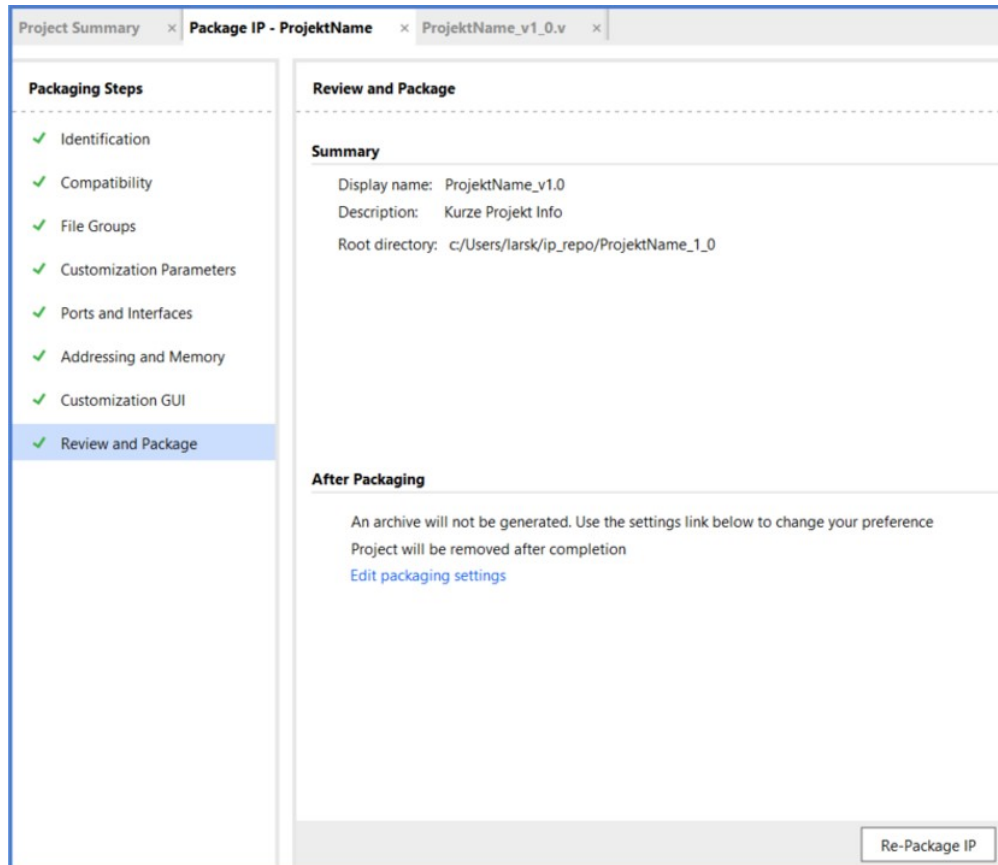
1.2.5 IP im nächsten Schritt direkt bearbeiten.

→ Next Steps: Edit IP → Finish



1.2.6 In dem sich öffnenden Bereich kann nun das Projekt angepasst werden. Dabei kann eigener Programmcode, sowie Ein- und Ausgänge eingefügt oder hinzugefügt werden. Die Schnittstellenkommunikation kann dabei bestehen bleiben, oder durch eine eigene ersetzt werden. Am einfachsten sollte sich jedoch das Anpassen des Register-Codes im zweiten Programmteil sein. Dies muss allerdings nach Bedarf und Projekt vorgenommen werden.

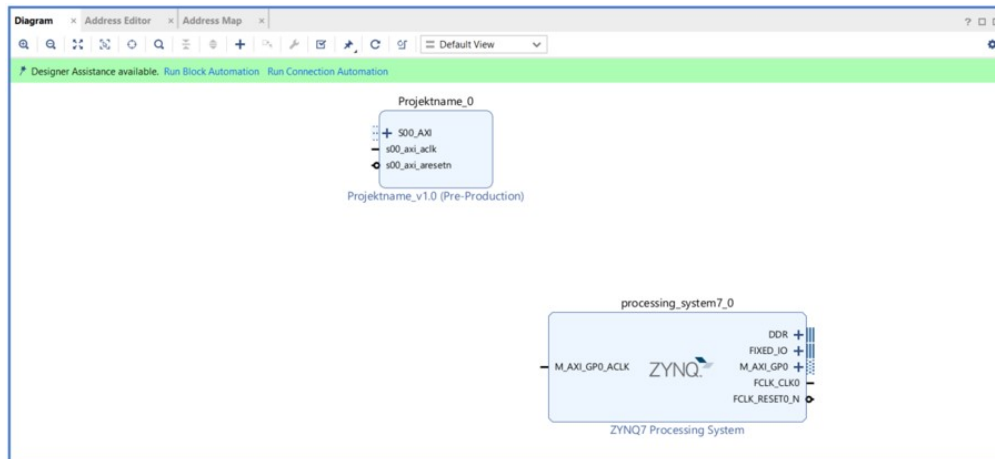
1.2.7 Wenn der Programmcode wie benötigt angepasst worden ist, muss im Package IP Fenster überprüft werden, ob alle Haken grün gesetzt sind, oder die Änderungen noch zu übernehmen sind. Anschließend wird dies mit Re-Package IP bestätigt und das Fenster der IP wird automatisch geschlossen und das Projekt gespeichert.



1.2.8 Daraufhin liegt die erstellte IP direkt zur Verfügung. Wenn dies nicht der Fall ist sollte der Pfad, an welchem das Projekt abgespeichert ist, kontrolliert oder als IP-Zugriff hinterlegt werden.

1.3 Create Block Design

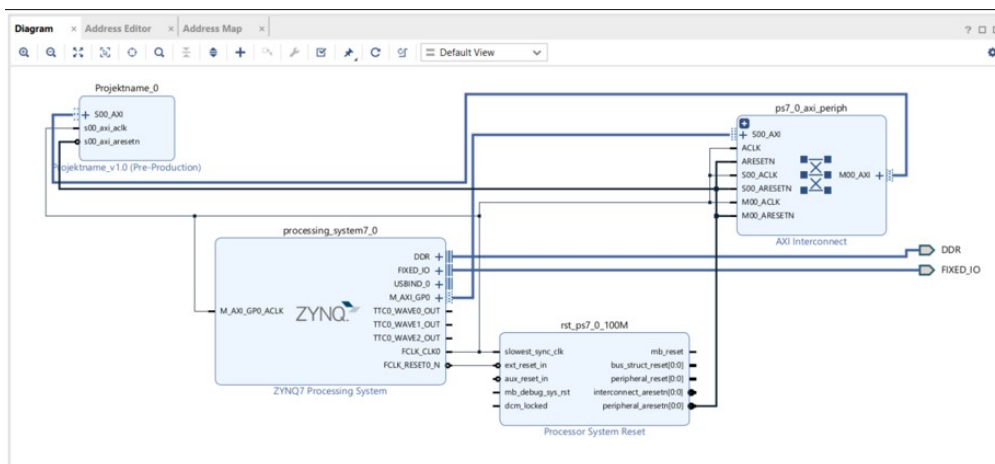
1.3.1 Nun kann das Block Design erstellt werden. Dazu werden die Module über das + Symbol hinzugefügt.



1.3.2 Wenn alle Module hinzugefügt sind, kann Vivado die Verbindung automatisch durchführen.

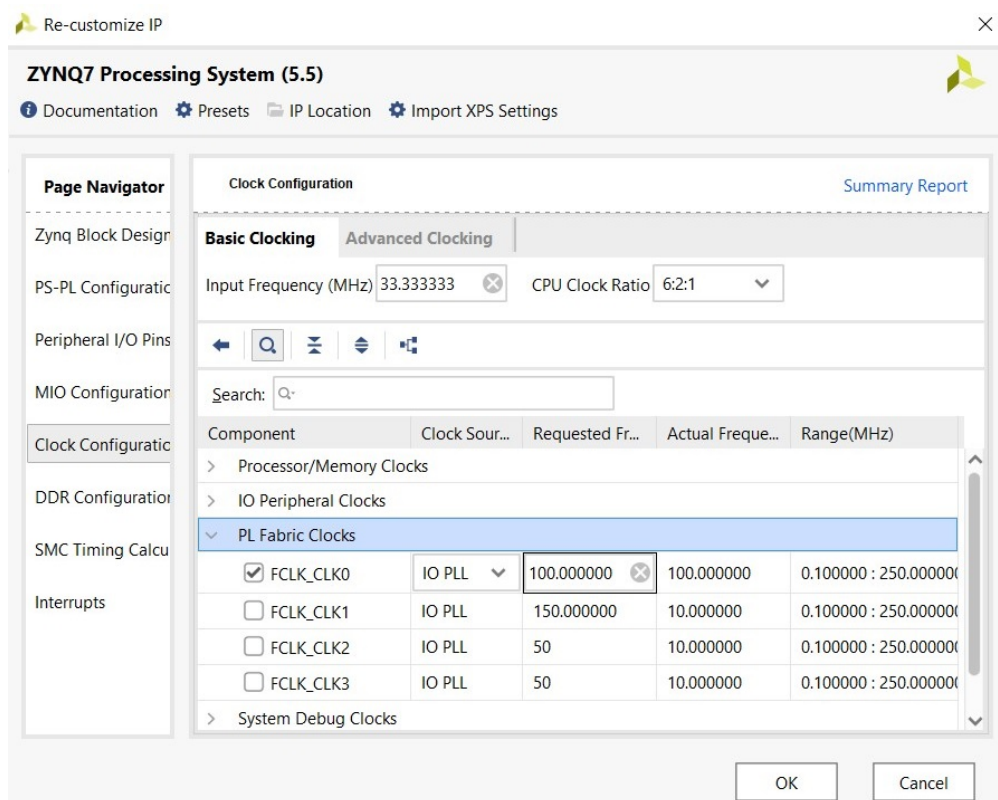
→ Run Block Automation

→ Run Connection Automation



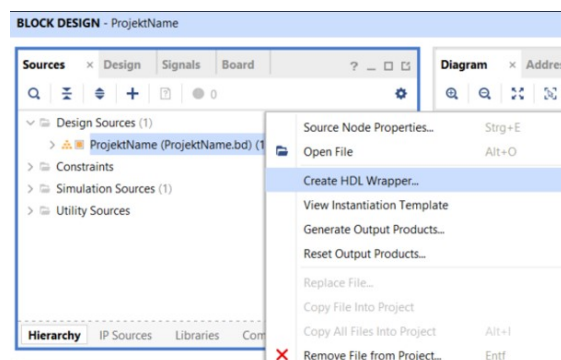
1.3.3 Im Anschluss sollten die Einstellungen des verwendeten Bausteins kontrolliert werden. Hier handelt es sich um ein Zynq-Baustein. Die verwendete Clock sollte auf 100MHz eingestellt sein.

→ Rechtsklick auf Zynq-Baustein → Customize Block



1.3.4 Sind alle Einstellungen wie benötigt, kann ein HDL-Wrapper erzeugt werden.

→ Rechtsklick auf das Projekt → Create HDL-Wrapper



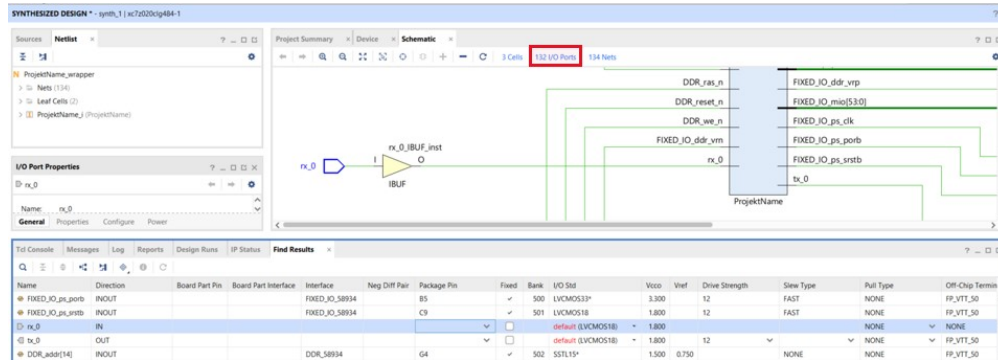
1 Integration einer IP mit Hardwaredesign in Vivado

1.3.5 Als nächstes ist die Synthese durchzuführen.

→ Run Synthese

1.3.6 Bei fehlerfreiem Durchlauf das Design öffnen und in das Schematic gehen.

→ Open Synthesized Design → Schematic öffnen mit Klick auf I/O Ports



1.3.7 Wenn Input und Output Ports vorgesehen sind, werden diese nun auf Pins des Boards gelegt. Die Belegung ist der Board Dokumentation zu entnehmen.

→ Hier werden als Beispiel die Pins JA1 und JA2 auf dem Zedboard verwendet.

Table 16 - Pmod Connections

Pmod	Signal Name	Zynq pin	Pmod	Signal Name	Zynq pin
JA1	JA1	Y11	JB1	JB1	W12
	JA2	AA11		JB2	W11
	JA3	Y10		JB3	V10
	JA4	AA9		JB4	W8
	JA7	AB11		JB7	V12
	JA8	AB10		JB8	W10
	JA9	AB9		JB9	V9
	JA10	AA8		JB10	V8
JC1 Differential	JC1_N	AB6	JD1 Differential	JD1_N	W7
	JC1_P	AB7		JD1_P	V7
	JC2_N	AA4		JD2_N	V4
	JC2_P	Y4		JD2_P	V5
	JC3_N	T6		JD3_N	W5
	JC3_P	R6		JD3_P	W6
	JC4_N	U4		JD4_N	U5
	JC4_P	T4		JD4_P	U6

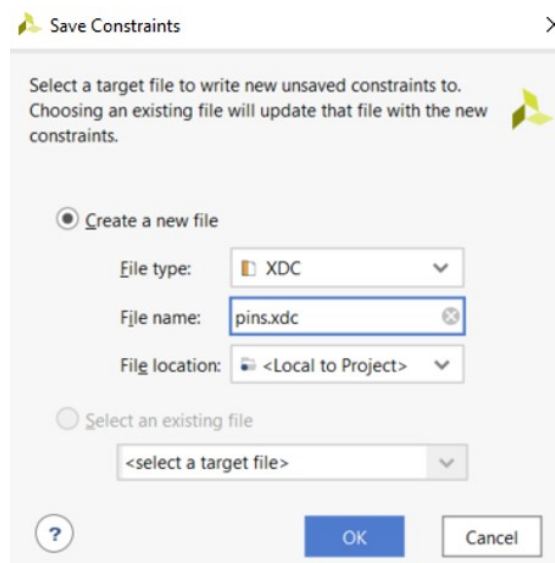
1 Integration einer IP mit Hardwaredesign in Vivado

→ Somit müssen rx und tx den Pins Y11 und AA11 des Boards zugewiesen werden. Als weitere Anpassung, aufgrund der höheren Signalspannungen, muss im Feld I/O Std LVCMOS33* eingestellt werden.

rx_0	IN				Y11	✓	13	LVCMOS33*	3.300
tx_0	OUT				AA11	✓	13	LVCMOS33*	3.300

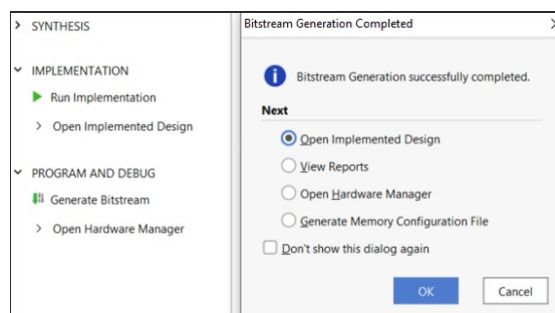
1.3.8 Die neuen Einstellungen sind zu speichern, bevor das Design geschlossen wird.

→ Vergeben Sie einen sinnvollen Namen, z.B pins.xdc



1.3.9 Im Anschluss kann der Bitstream erzeugt werden.

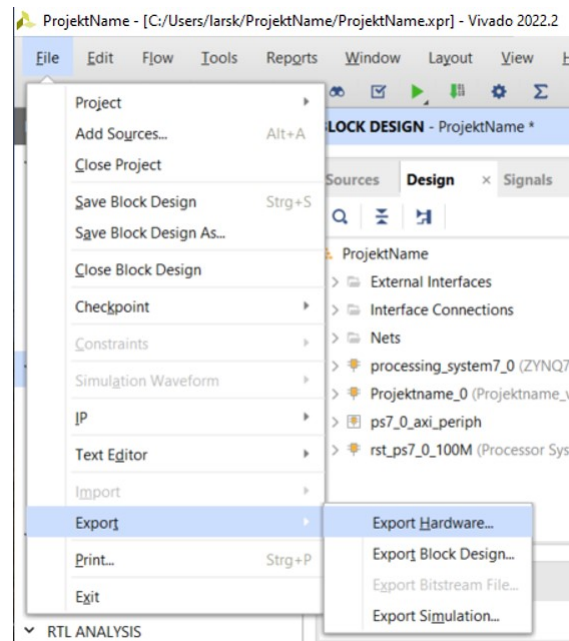
→ Generate Bitstream



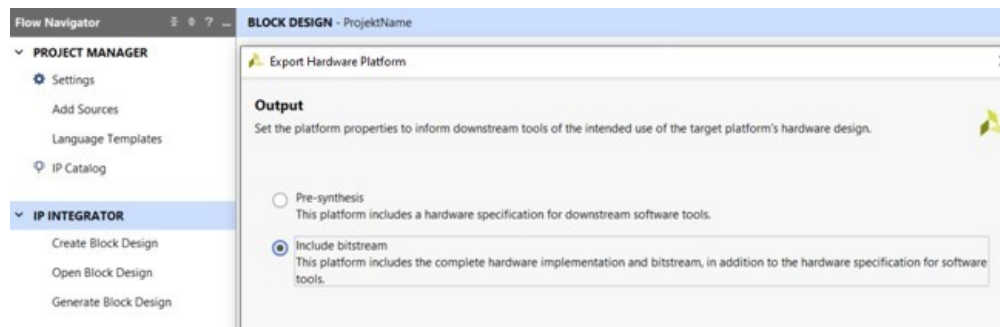
1.4 Export Hardware

1.4.1 Wenn der Bitstream fehlerfrei generiert wurde, kann das Design exportiert werden.

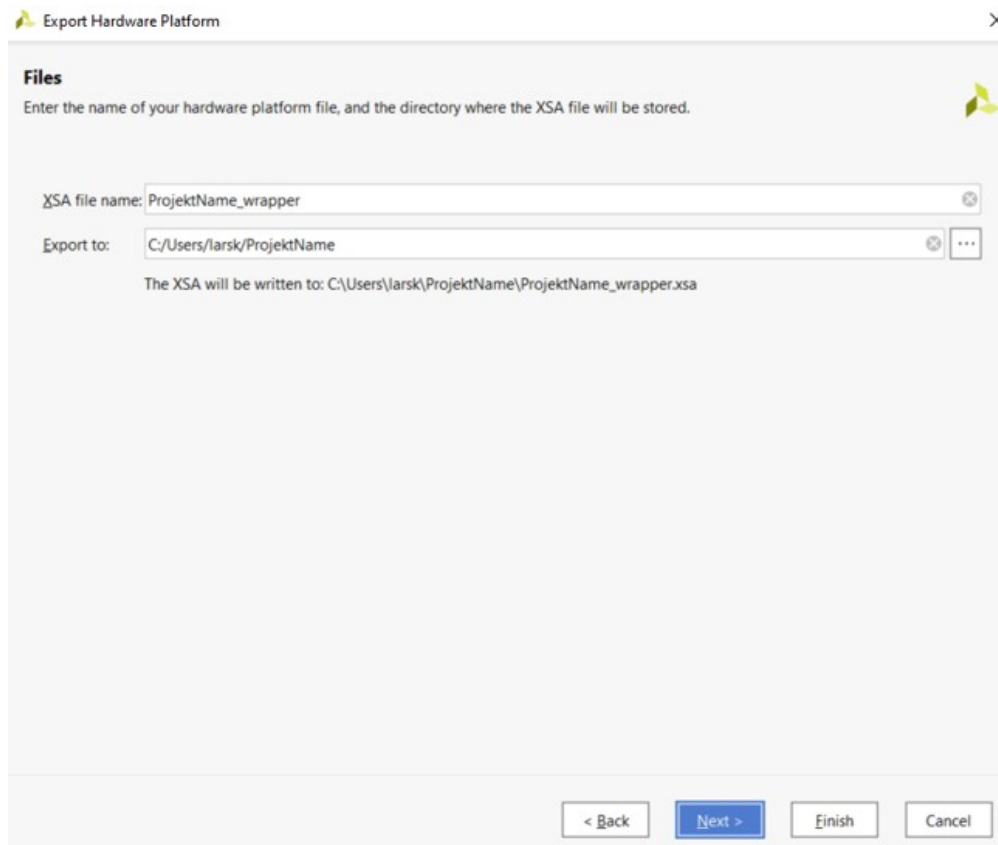
→ File → Export Hardware



1.4.2 Bitte darauf achten, dass der Bitstream mit einbezogen wird (Include bitstream).



1.4.3 Nun kann ein Name vergeben werden, unter welchem die Datei im Projektordner abgespeichert werden soll.



→ Das Projekt liegt nun als .xsa-Datei vor und kann für die PetaLinux-Installation verwendet werden.

2 PetaLinux-Installation

Der Installationsvorgang findet unter Linux auf dem Labor Computer statt. Die Befehle werden im Terminal eingegeben und ausgeführt. Die meisten Schritte können auch über Remote Desktop durchgeführt werden. Lediglich die Erstellung des Datenträgers mit Einfügung der root Datei kann zum Schluss nur direkt am PC, auf welchem die Daten liegen, durchgeführt werden. Ein späteres Abändern ist dann wieder über Remote Desktop möglich, wenn nur Änderungen im Programmcode stattfinden und nicht die sdk Datei verändert werden muss.

2.1 Installation der PetaLinux-Umgebung (Einmalige Durchführung)

2.1.1 Zuerst ist das Eingabeterminal aufzurufen und eine Verbindung zum hoerni Server herzustellen. Über diesen findet dann der Installationsvorgang statt.

→ ssh hoerni

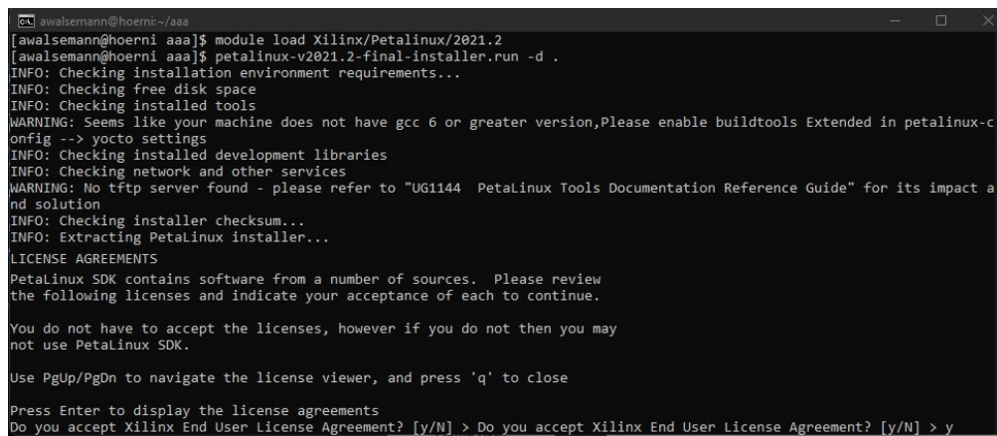
2.1.2 Login mit Nutzerpasswort vom Laboraccount

→Passwort

2.1.3 Nun muss das entsprechende Umgebungsmodul geladen, PetaLinux installiert und die License Agreements bestätigt werden.

→ module load Xilinx/Petalinux/2021.2

→ petalinux-v2021.2-final-installer.run -d



```
[awalsemann@hoerni ~]$ module load Xilinx/Petalinux/2021.2
[awalsemann@hoerni ~]$ petalinux-v2021.2-final-installer.run -d .
INFO: Checking installation environment requirements...
INFO: Checking free disk space
INFO: Checking installed tools
WARNING: Seems like your machine does not have gcc 6 or greater version, Please enable buildtools Extended in petalinux-config --> yocto settings
INFO: Checking installed development libraries
INFO: Checking network and other services
WARNING: No tftp server found - please refer to "UG1144 PetaLinux Tools Documentation Reference Guide" for its impact and solution
INFO: Checking installer checksum...
INFO: Extracting PetaLinux installer...
LICENSE AGREEMENTS
PetaLinux SDK contains software from a number of sources. Please review
the following licenses and indicate your acceptance of each to continue.

You do not have to accept the licenses, however if you do not then you may
not use PetaLinux SDK.

Use PgUp/PgDn to navigate the license viewer, and press 'q' to close

Press Enter to display the license agreements
Do you accept Xilinx End User License Agreement? [y/N] > Do you accept Xilinx End User License Agreement? [y/N] > y
```


2.1.4 Im Anschluss wird der Speicherort erstellt, in welchem die Installationsdateien abgelegt werden sollen. Dies kann zum Beispiel unter dem Pfad des Benutzerordners erfolgen. Der Name hier wird schlicht mit **petalinux** gewählt.

→ `mkdir <user_home_dir>/petalinux`

(Hierbei steht `<user_home_dir>` für den Pfad auf das eigene User-Verzeichnis)

2.1.5 Durch den nächsten Befehl stehen die nötigen PetaLinux-Befehle zur Verfügung

→ `source settings.sh`

```
Select awalsemann@hoerni:~/Documents/petalinux
[awalsemann@hoerni petalinux]$ [awalsemann@hoerni petalinux]$
[awalsemann@hoerni petalinux]$ source settings.sh
Petalinux environment set to '/user/awalsemann/Documents/petalinux'
INFO: Checking free disk space
INFO: Checking installed tools
WARNING: Seems like your machine does not have gcc 6 or greater version, Please enable buildtools Extended in petalinux-c
onfig --> yocto settings
INFO: Checking installed development libraries
INFO: Checking network and other services
WARNING: No tftp server found - please refer to "UG1144 2021.2 PetaLinux Tools Documentation Reference Guide" for its im
pact and solution
```

2.1.6 Nun muss eine Datei von der Xilinx-Webseite geladen und installiert werden. Es handelt sich um ein Board Support Package für das verwendete FPGA-Board (Zedboard).

Direkt Link zum File: Board support Package

<https://www.xilinx.com/member/forms/download/xef.html?filename=avnet-digilent-zedboard-v2021.2-final.bsp>

Link über die Webseite, zu wählende Datei rot markiert in der folgenden Abbildung.

<https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/embedded-design-tools/2021-2.html>

Zynq-7000 SoC Board Support Packages - 2021.2

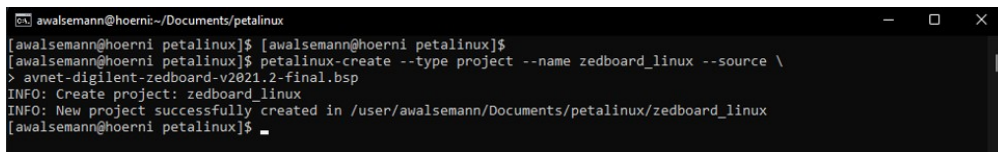
Important Information

Download only the required BSP(s) depending on the evaluation board that is being used. All BSPs have a prebuilt directory with bootable images. Hover your mouse over the download hyper-link to see a description of the BSP contents.

ZC702 BSP (BSP - 118.17 MB) MD5 SUM Value : f3ca87fe918a2f0a93c8ee0869cf341f	<div>Download Includes</div> <div>Download Type</div> <div>Last Updated</div> <div>Answers</div> <div>Documentation</div>	<div>Zynq-7000 SoC Board Support Packages</div> <div>Oct 27, 2021</div> <div>Release Notes and Known Issues</div> <div>PetaLinux Tools Documentation</div>
ZC706 BSP (BSP - 126.73 MB) MD5 SUM Value : 62bf2cbf0cec26376ec2fc112f3a34b7		
ZED BSP (BSP - 117.39 MB) MD5 SUM Value : d2ba98a3caac5c9ef5376003d2a5cf		

2.1.7 Die Datei ist im PetaLinux Ordner abzulegen und die nachfolgende Installation in diesem durchzuführen. Es wird zusätzlich ein neuer Unterordner mit `mkdir` angelegt, in welchem die folgenden Installationsdateien abgelegt werden. Dazu wird ein neuer Projektordner erstellt und das Board File mit dem folgenden Befehl geladen und Installiert. Der hier gewählte Name für den Unterordner lautet `zedboard_linux`. Die Datei enthält z.B. Infos für den Bootloader.

→ `petalinux-create --type project --name --source avnet-digilent-zedboard-v2021.2-final.bsp`



```
awalsemann@hoerni:~/Documents/petalinux
[awalsemann@hoerni petalinux]$ [awalsemann@hoerni petalinux]$
[awalsemann@hoerni petalinux]$ petalinux-create --type project --name zedboard_linux --source \
> avnet-digilent-zedboard-v2021.2-final.bsp
INFO: Create project: zedboard_linux
INFO: New project successfully created in /user/awalsemann/Documents/petalinux/zedboard_linux
[awalsemann@hoerni petalinux]$
```

2.1.8 Es müssen erneut die License agreements bestätigt werden, um fort zu fahren.

2.2 Projekt konfigurieren

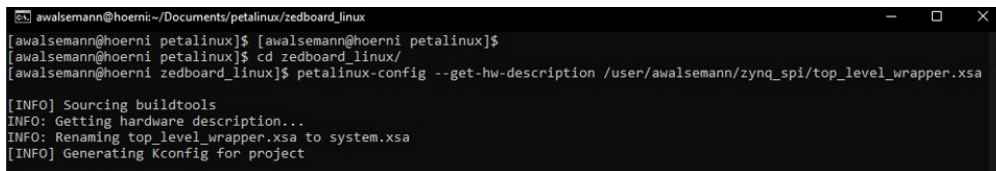
Die nachfolgenden Schritte können nun immer wiederholt werden, wenn das Projekt geändert wurde oder ein neues erstellt wird. Ggf. muss erst die Verbindung zum Server, mit `ssh hoerni` und dem Login über das eigene Passwort hergestellt werden. Beachten Sie des Weiteren, in welchem Ordner Sie sich befinden!

2.2.1 Das Hardware Design wird nun importiert und dazu vorher im Projektordner abgelegt. Die nächsten Schritte können in diesem Ordner durchgeführt werden. In diesem Beispiel handelt es sich um den Ordner `zedboard_linux`.

→ `cd zedboard_linux`

2.2.2 Nun wird die PetaLinux Installation mit dem Hardware Design durchgeführt und dazu die `.xsa`-Datei eingebunden.

→ `petalinux-config --get-hw-description Projektname_wrapper.xsa`



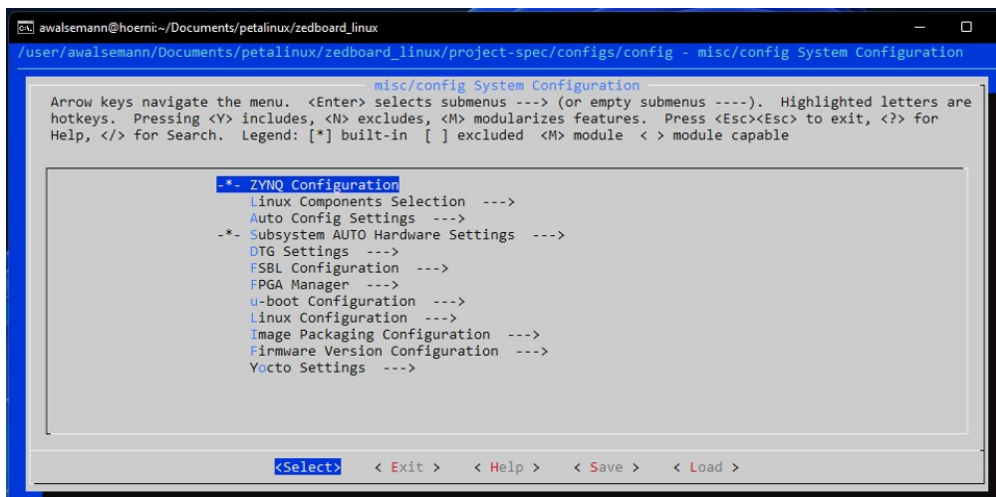
```

awalsemann@hoerni:~/Documents/petalinux/zedboard_linux
[awalsemann@hoerni petalinux]$ [awalsemann@hoerni petalinux]$
[awalsemann@hoerni petalinux]$ cd zedboard_linux/
[awalsemann@hoerni zedboard_linux]$ petalinux-config --get-hw-description /user/awalsemann/zynq_spi/top_level_wrapper.xsa

[INFO] Sourcing buildtools
INFO: Getting hardware description...
INFO: Renaming top_level_wrapper.xsa to system.xsa
[INFO] Generating Kconfig for project

```

2.2.3 Die Installation dauert einen Moment und im Anschluss wird ein Konfigurationsfenster geöffnet. Hier können Einstellungen für das System vorgenommen werden. Die nachfolgenden Einstellungen sollten ebenfalls durchgeführt werden, können jedoch auch nach Bedarf weiter angepasst werden.



```

awalsemann@hoerni:~/Documents/petalinux/zedboard_linux
/user/awalsemann/Documents/petalinux/zedboard_linux/project-spec/configs/config - misc/config System Configuration

misc/config System Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are
hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for
Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable

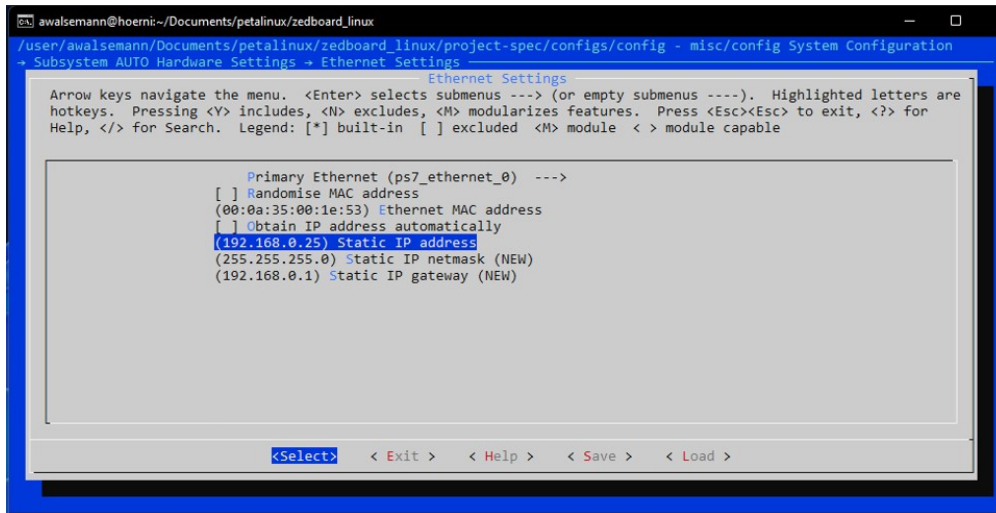
--*- ZYNQ Configuration
Linux Components Selection --->
Auto Config Settings --->
--*- Subsystem AUTO Hardware Settings --->
DTG Settings --->
FSBL Configuration --->
FPGA Manager --->
u-boot Configuration --->
Linux Configuration --->
Image Packaging Configuration --->
Firmware Version Configuration --->
Vxto Settings --->

<Select> < Exit > < Help > < Save > < Load >

```

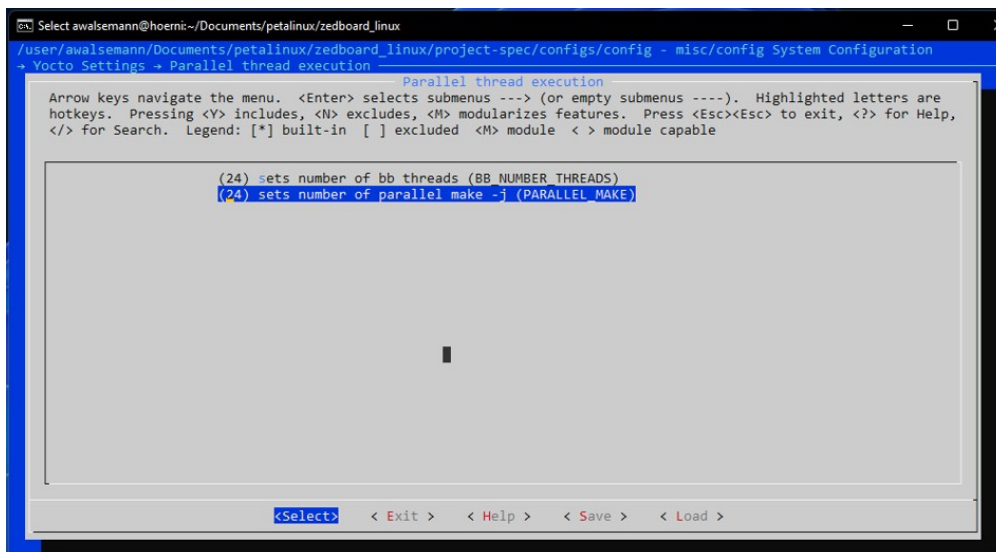
2.2.4 Festlegen der IP-Adresse

→ Subsystem AUTO Hardware Settings → Ethernet Settings → Static IP address



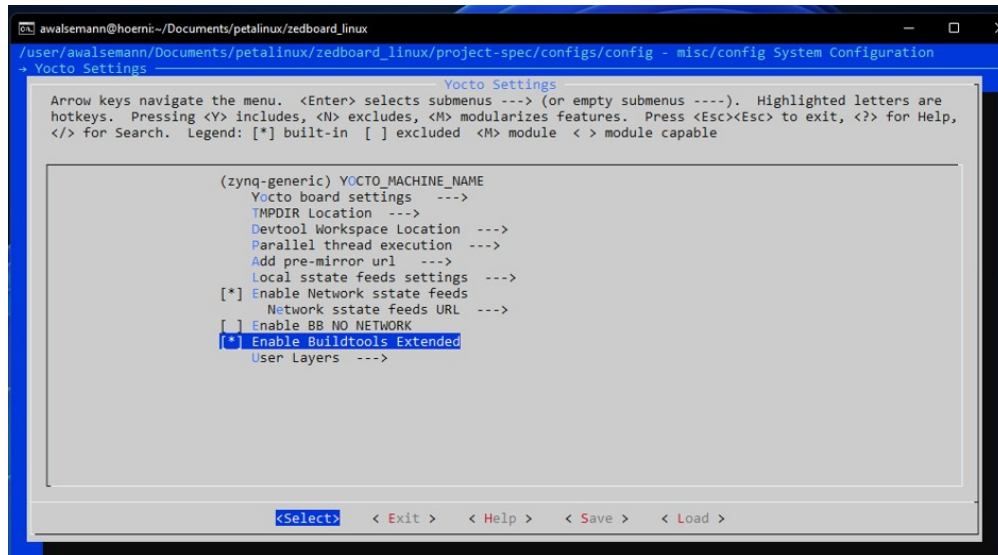
2.2.5 Die Parallele Thread Execution wird auf 24 gesetzt.

→ Yocto Settings → Parallel thread execution



2.2.6 Enable Buildtools Extended wird gesetzt. (Dies ist eine, für diesen Anwendungsfall, besser abgestimmte Version von Xilinx).

→ Yocto settings → Enable Buildtools Extended → Y zum Auswählen



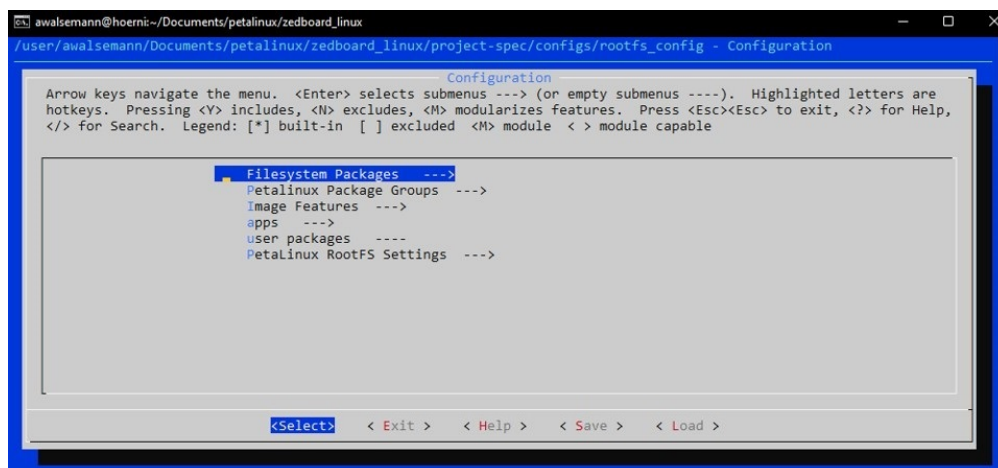
2.2.7 Einstellungen mit EXIT verlassen und mit Y speichern.

2.2.8 Nun kann die rootfs config bearbeitet werden. Mit folgender Eingabe ist diese im Terminal aufzurufen.

→ petalinux-config --c rootfs

2.2.9 Peek and Poke zum Lesen und schreiben von Registern hinzufügen.

→ apps → peekpoke → Y zum auswählen



Mit peek und poke können zu Testzwecken für eine gewünschte Adresse Daten aus dem Speicher gelesen oder in den Speicher geschrieben werden.

2.2.10 Mit dem nächsten Befehl im Terminal wird das Projekt erstellt und die Programmdateien abgespeichert.

→ petalinux-build

→ petalinux-package --boot --u-boot --format BIN --force

```
awalsemann@hoerni:~/Documents/petalinux/zedboard_linux
[awalsemann@hoerni zedboard_linux]$
[awalsemann@hoerni zedboard_linux]$ petalinux-build
[INFO] Sourcing buildtools
[INFO] Building project
[INFO] Sourcing buildtools extended
[INFO] Sourcing build environment
[INFO] Generating workspace directory
INFO: bitbake petalinux-image-minimal
NOTE: Started PRServer with DBfile: /user/awalsemann/Documents/petalinux/zedboard_linux/build/cache/prserv.sqlite3, IP: 127.0.0.1, PORT: 46342, PID: 36460
Loading cache: 100% |#####| Time: 0:00:01
Loaded 5128 entries from dependency cache.
Parsing recipes: 100% |#####| Time: 0:00:01
Parsing of 3476 .bb files complete (3471 cached, 5 parsed). 5133 targets, 268 skipped, 0 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies
Initialising tasks: 100% |#####| Time: 0:00:03
Checking sstate mirror object availability: 100% |#####| Time: 0:00:04
Sstate summary: Wanted 1171 Found 946 Missed 225 Current 82 (80% match, 82% complete)
NOTE: Executing Tasks
NOTE: Tasks Summary: Attempted 3861 tasks of which 2860 didn't need to be rerun and all succeeded.
INFO: Failed to copy built images to tftp dir: /tftpboot
[INFO] Successfully built project
[awalsemann@hoerni zedboard_linux]$ petalinux-package --boot --u-boot --format BIN --force
[INFO] Sourcing buildtools
[INFO] Getting system flash information...
INFO: File in BOOT BIN: "/user/awalsemann/Documents/petalinux/zedboard_linux/images/linux/zyng_fsb1.elf"
INFO: File in BOOT BIN: "/user/awalsemann/Documents/petalinux/zedboard_linux/project-spec/hw-description/top_level_wrapper.bit"
INFO: File in BOOT BIN: "/user/awalsemann/Documents/petalinux/zedboard_linux/images/linux/u-boot.elf"
INFO: File in BOOT BIN: "/user/awalsemann/Documents/petalinux/zedboard_linux/images/linux/system.dtb"
INFO: Generating zynq binary package BOOT.BIN...

***** Xilinx Bootgen v2021.2
**** Build date : Sep 30 2021-06:08:18
** Copyright 1986-2021 Xilinx, Inc. All Rights Reserved.

[INFO] : Bootimage generated successfully

INFO: Binary is ready.
WARNING: Unable to access the TFTPBOOT folder /tftpboot!!!
WARNING: Skip file copy to TFTPBOOT folder!!!
[awalsemann@hoerni zedboard_linux]$
```

Die erstellten Dateien liegen im Ordner zedboard_linux/images.

2.2.11 Als letztes muss noch die SDK Generierung für die spätere Verwendung in Vitis im Ordner Linux durchgeführt werden.

→ `cd images/linux/`

→ `petalinux-build --sdk`

```

[awalsemann@hoerni ~/Documents/petalinux/zedboard_linux/images/linux]
[awalsemann@hoerni zedboard_linux]$ [awalsemann@hoerni zedboard_linux]$
[awalsemann@hoerni zedboard_linux]$ cd images/linux/
[awalsemann@hoerni linux]$ petalinux-build --sdk
[INFO] Sourcing buildtools
[INFO] Building project
[INFO] Sourcing buildtools extended
[INFO] Sourcing build environment
[INFO] Generating workspace directory
INFO: bitbake petalinux-image-minimal -c do_populate_sdk
NOTE: Started PRServer with DBfile: /user/awalsemann/Documents/petalinux/zedboard_linux/build/cache/prserv.sqlite3, IP: 127.0.0.1,
PORT: 44732, PID: 129540
Loading cache: 100% ##### Time: 0:00:01
Loaded 5128 entries from dependency cache.
Parsing recipes: 100% ##### Time: 0:00:01
Parsing of 3476 .bb files complete (3471 cached, 5 parsed). 5133 targets, 268 skipped, 0 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies
Initialising tasks: 100% ##### Time: 0:00:02
Checking sstate mirror object availability: 100% ##### Time: 0:02:01
Sstate summary: Wanted 514 Found 315 Missed 199 Current 660 (61% match, 83% complete)
NOTE: Executing Tasks
NOTE: Tasks Summary: Attempted 3521 tasks of which 3502 didn't need to be rerun and all succeeded.
[INFO] Copying SDK Installer...
[INFO] Successfully built project
[awalsemann@hoerni linux]$ ls
BOOT.BIN      config      rootfs.cpio      rootfs.ext4      rootfs.tar.gz    system.dtb      u-boot-dtb.elf  vmlinux
bootgen.bif  image.ub   rootfs.cpio.gz   rootfs.jffs2     sdk.sh           u-boot.bin      u-boot.elf      zImage
boot.scr     pxelinux.cfg rootfs.cpio.gz.u-boot rootfs.manifest  system.bit       u-boot-dtb.bin  uImage          zynq_fsbl.elf
[awalsemann@hoerni linux]$ ./sdk.sh
Petalinux SDK Installer version 2021.2
#####
Enter target directory for SDK (default: /opt/petalinux/2021.2): sdk
You are about to install the SDK to "/user/awalsemann/Documents/petalinux/zedboard_linux/images/linux/sdk". Proceed [Y/n]? y
Extracting SDK.....done
Setting it up...done
SDK has been successfully set up and is ready to be used.
Each time you wish to use the SDK in a new shell session, you need to source the environment setup script e.g.
$ . /user/awalsemann/Documents/petalinux/zedboard_linux/images/linux/sdk/environment-setup-cortexa9t2hf-neon-xilinx-linux-gnueabi
[awalsemann@hoerni linux]$

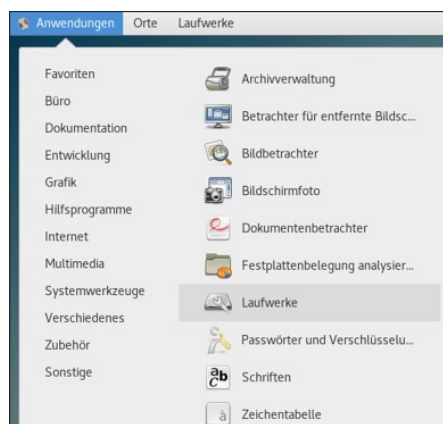
```

2.2.12 Die `sdk`-Datei muss später nur bei Bedarf neu geladen werden, z.B. bei neuen Treibern. Wenn sich nur der Programmcode unabhängig vom weiteren Bedarf für Vitis geändert hat, reicht es aus die `BOOT.BIN`, `BOOT.src` und `u_boot.ub` Dateien neu auf die SD-Karte zu laden.

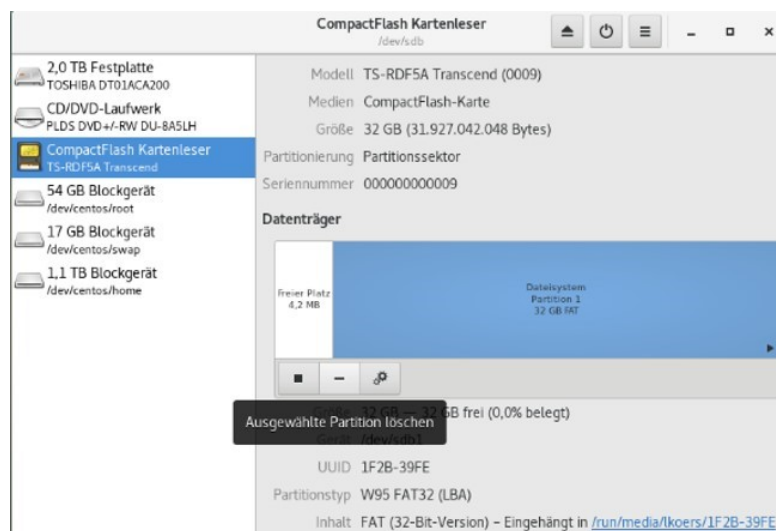
3 SD-Karte vorbereiten für Bootdateien

3.1.1 Die SD-Karte muss in zwei Partitionen unterteilt werden. Die Aufteilung ist relativ beliebig, jedoch sollte die **boot** Partition stets größer gewählt werden als die **root** Partition.

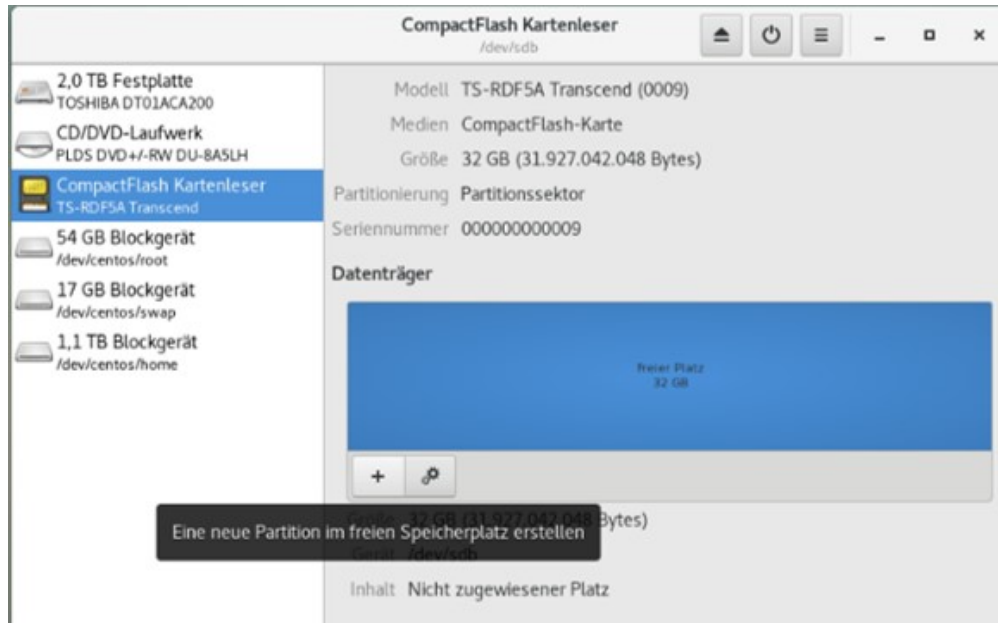
→ Laufwerke aufrufen



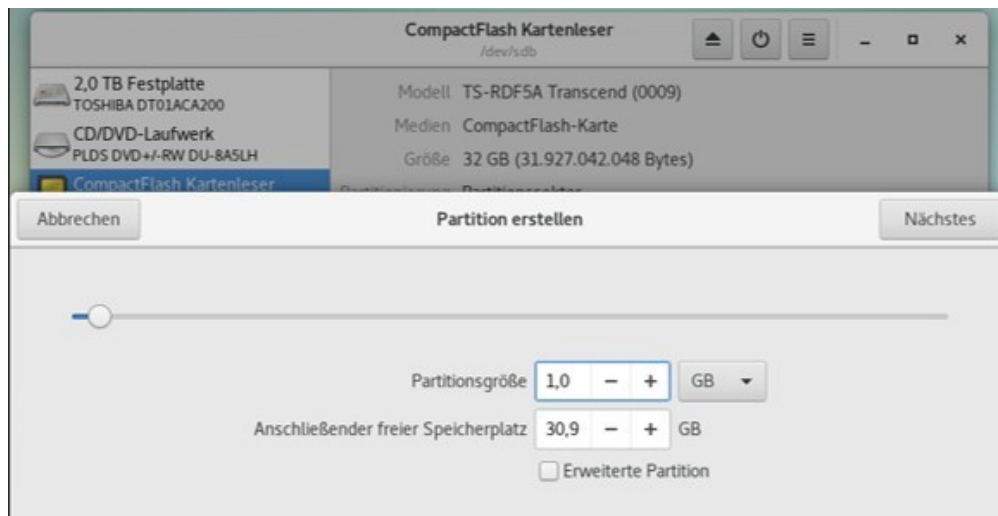
3.1.2 Unterteilen in zwei Partitionen, ggf. muss eine vorhandene mit - gelöscht werden.



3.1.3 Als nächstes mit + eine neue Partition hinzufügen.

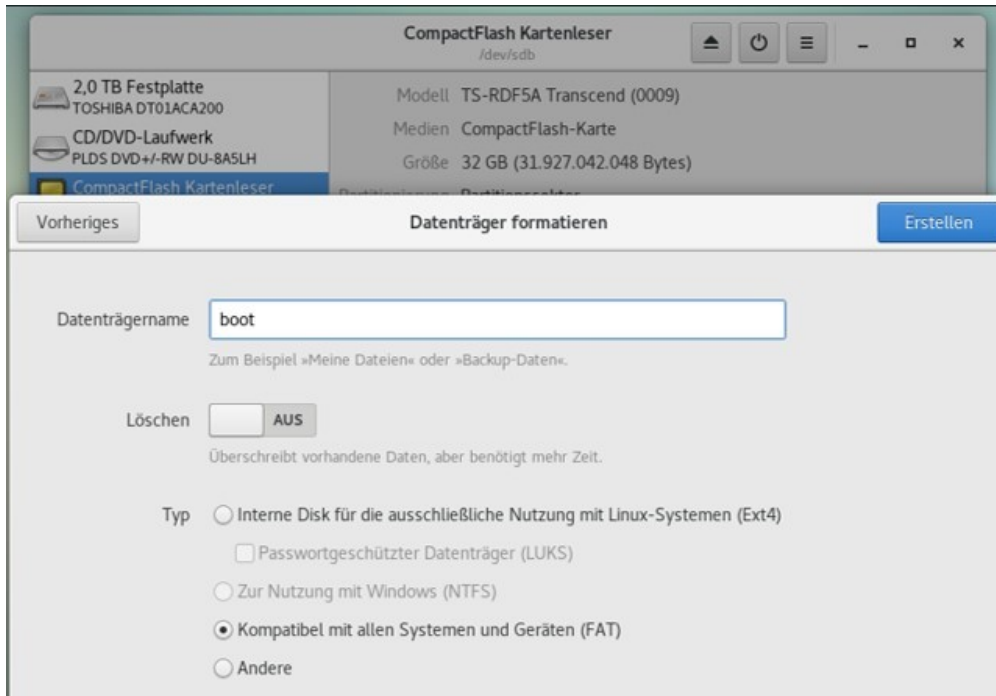


3.1.4 Speicheraufteilung festlegen, z.B. 512MB oder 1GB für die erste Partition.

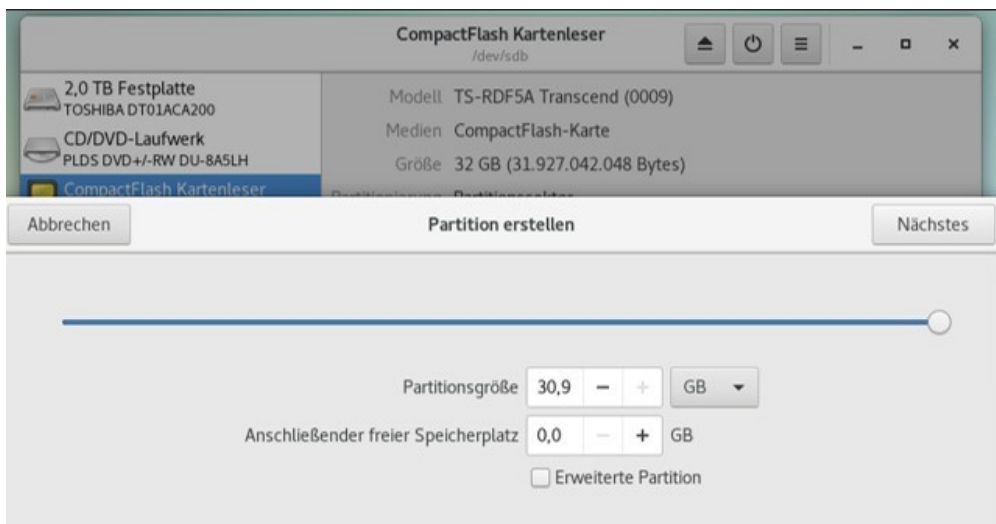


3 SD-Karte vorbereiten für Bootdateien

3.1.5 Namen der Partition mit boot festlegen und Typ FAT ausgewählt lassen.



3.1.6 Nächste Partition mit + erstellen und verfügbaren Restspeicher der Partition zuweisen.



3 SD-Karte vorbereiten für Bootdateien

3.1.7 Name root vergeben und Typ Ext4 auswählen.

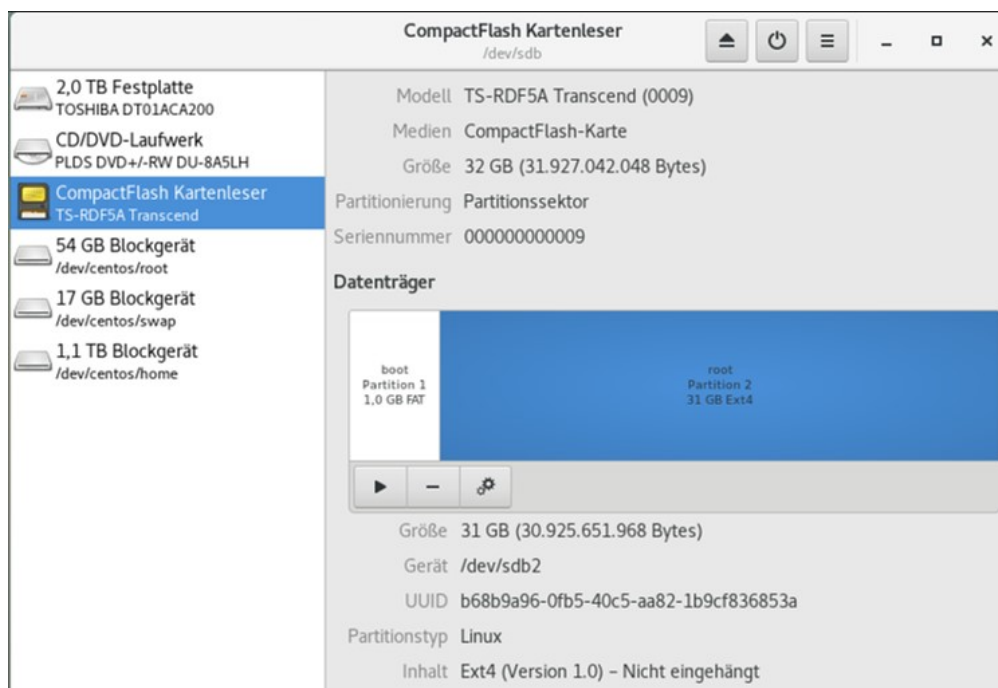
Vorheriges **Datenträger formatieren** Erstellen

Datenträgername
Zum Beispiel »Meine Dateien« oder »Backup-Daten«.

Löschen
Überschreibt vorhandene Daten, aber benötigt mehr Zeit.

Typ ☒ Interne Disk für die ausschließliche Nutzung mit Linux-Systemen (Ext4)
☐ Passwortgeschützter Datenträger (LUKS)
☐ Zur Nutzung mit Windows (NTFS)
☐ Kompatibel mit allen Systemen und Geräten (FAT)
☐ Andere

→ Fertig erstellte Partitionen auf der SD-Karte.



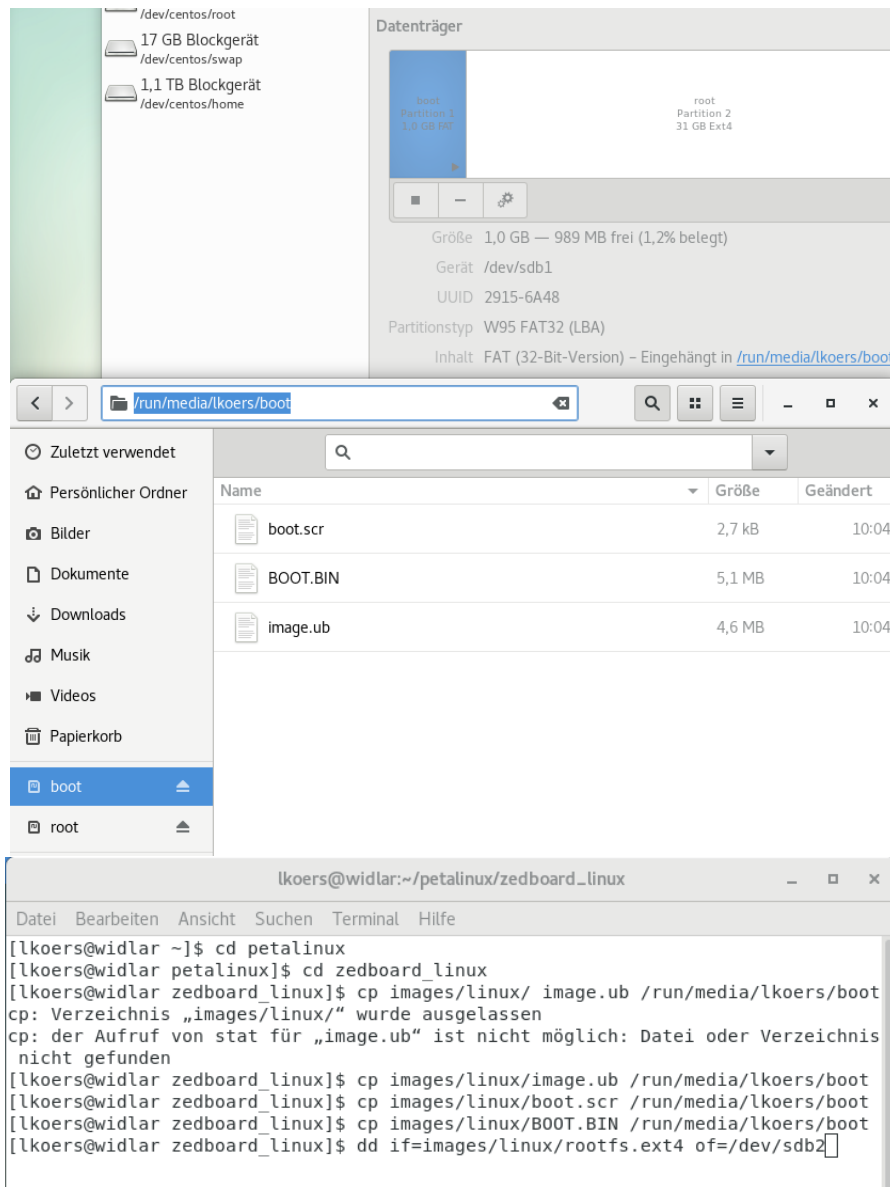
3 SD-Karte vorbereiten für Bootdateien

3.1.8 Im Anschluss können die zuvor erstellten Dateien auf die SD-Karte geladen werden. Dabei genügt es bei drei Dateien diese in die boot Partition zu kopieren.

→ Entweder mit Copy and Paste

Oder durch einen Befehl in Linux mit Dateiname und Pfad der Speicherkarte(Blau).

→ `cp images/linux/ Dateiname /run/media/user/boot`



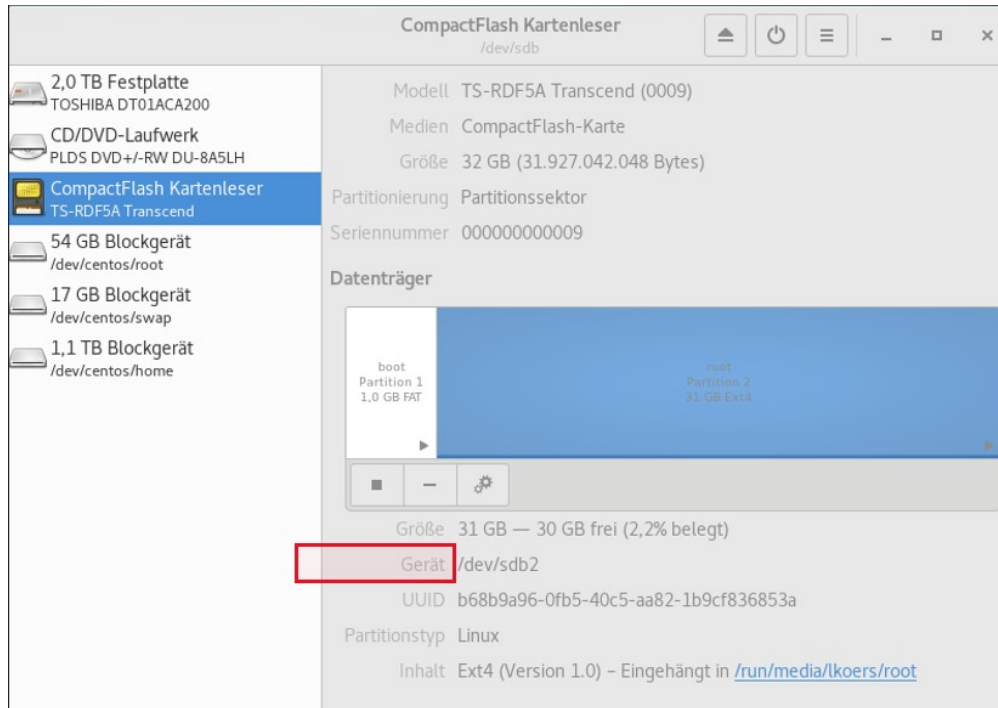
The screenshot displays the process of preparing an SD card for booting. At the top, the 'Datenträger' (Storage) window shows the 'boot' partition (1,0 GB, 989 MB free) selected. Below it, the file manager window shows the path '/run/media/lkoers/boot' and a list of files: 'boot.scr' (2,7 kB), 'BOOT.BIN' (5,1 MB), and 'image.ub' (4,6 MB). At the bottom, the terminal window shows the following commands and output:

```
[lkoers@widlar ~]$ cd petalinux
[lkoers@widlar petalinux]$ cd zedboard linux
[lkoers@widlar zedboard_linux]$ cp images/linux/ image.ub /run/media/lkoers/boot
cp: Verzeichnis „images/linux/“ wurde ausgelassen
cp: der Aufruf von stat für „image.ub“ ist nicht möglich: Datei oder Verzeichnis
nicht gefunden
[lkoers@widlar zedboard_linux]$ cp images/linux/image.ub /run/media/lkoers/boot
[lkoers@widlar zedboard_linux]$ cp images/linux/boot.scr /run/media/lkoers/boot
[lkoers@widlar zedboard_linux]$ cp images/linux/BOOT.BIN /run/media/lkoers/boot
[lkoers@widlar zedboard_linux]$ dd if=images/linux/rootfs.ext4 of=/dev/sdb2
```

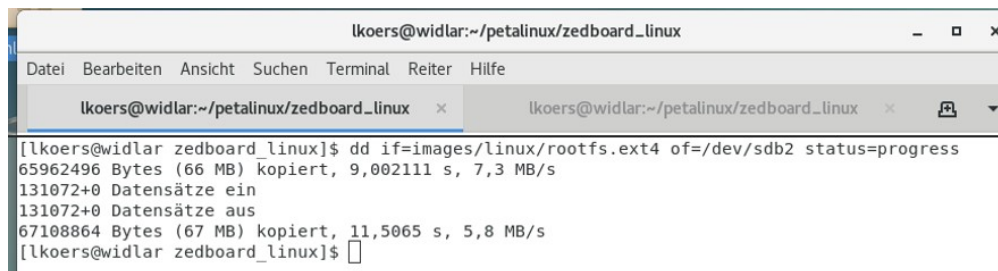
3 SD-Karte vorbereiten für Bootdateien

3.1.9 Die rootfs Datei **muss** wie im nächsten Schritt beschrieben verschoben werden!!
Befehl im Linux Terminal ausführen!

!!!Achtung auf richtigen Pfad verschieben, root Ordner → Siehe SD-Karte neben Gerät
(ROT markiert)



→ `dd if=images/linux/rootfs.ext4 of=/dev/sdb2`



3.1.10 Nach der Ausführung des Befehls befinden sich die Daten auf der SD-Karte. Bitte überprüfen Sie, ob auch wirklich etwas in den jeweiligen Partitionen vorhanden ist. Ggf. die SD-Karte einmal aus- und wieder einstecken.

4 Test der fertigen Installation am SoC-Board

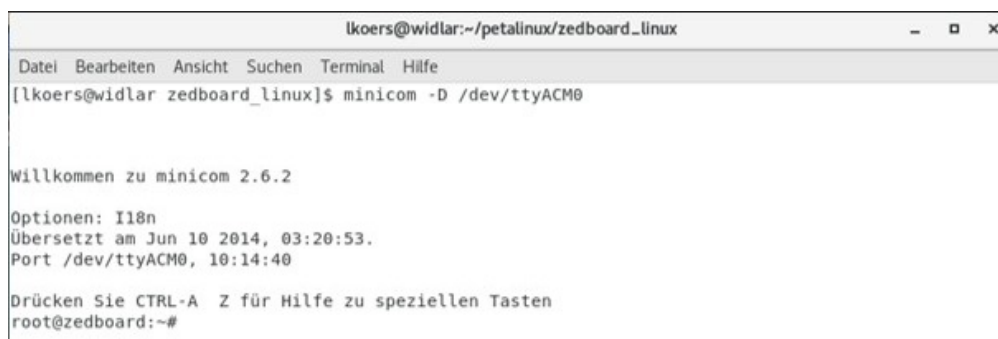
4.1 Test unter Linux

4.1.1 Nachdem alle Schritte durchgeführt worden sind, kann die SD-Karte in das Zed-board gesteckt, das Board an den PC angeschlossen und eingeschaltet werden. Bei richtigem Installationsvorgang bootet das Projekt und kann verwendet werden.

→ Einschalten → Boot startet automatisch

4.1.2 Verbindung unter Linux prüfen. Dazu das Terminalfenster öffnen und folgenden Befehl eingeben:

→ `minicom -D /dev/ttyACM0`



```
lkoers@widlar:~/petalinux/zedboard_linux
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
[lkoers@widlar zedboard_linux]$ minicom -D /dev/ttyACM0

Willkommen zu minicom 2.6.2

Optionen: I18n
Übersetzt am Jun 10 2014, 03:20:53.
Port /dev/ttyACM0, 10:14:40

Drücken Sie CTRL-A Z für Hilfe zu speziellen Tasten
root@zedboard:~#
```

4.1.3 Hier können unter anderem peek und poke Befehle zum Testen ausgeführt werden.

4.1.4 Wenn das Programm beendet werden soll, kann dies wie folgt verlassen werden.

→ Strg a → dann q zum verlassen der Anwendung



```
lkoers@widlar:~/petalinux/zedboard_linux
Datei Bearbeiten Ansicht Suchen Terminal Hilfe

root@zedboard:~#

+-----+
| Minicom ohne Reset verlassen? |
|      Ja      Nein      |
+-----+
```

4.2 Test unter Windows mit Tera Term

Über ein Programm wie zum Beispiel Tera Term können Befehle an das Board gesendet und die Antworten gelesen werden.

4.2.1 Dazu das Programm öffnen und den COM-Port auswählen, an dem das Board angeschlossen ist. Anschließend kann wie im Beispiel hier der Bootvorgang eingesehen werden und am Ende Befehle wie peek und poke verwendet werden.

```
COM6 - Tera Term VT
File Edit Setup Control Window Help
of_cfs_init: OK
ALSA device list:
No soundcards found.
Waiting for root device /dev/mmcblk0p2...
mmc0: new high speed SDHC card at address 5048
mmcblk0: mmc0:5048 S0326 29.7 GiB
mmcblk0: p1 p2
EXT4-fs (mmcblk0p2): mounted filesystem with ordered data mode. Opts: (null)
VFS: Mounted root (ext4 filesystem) on device 179:2.
devtmpfs: mounted
Freeing unused kernel memory: 1024K
Run /sbin/init as init process
INIT: version 2.97 booting
random: fast init done
Starting udev
udev[72]: starting version 3.2.9
random: udevd: uninitialized urandom read (16 bytes read)
random: udevd: uninitialized urandom read (16 bytes read)
random: udevd: uninitialized urandom read (16 bytes read)
udev[73]: starting udev-3.2.9
FAT-fs (mmcblk0p1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
EXT4-fs (mmcblk0p2): re-mounted. Opts: (null)
bootlogd: /dev/ttyFS0huclock: can't open '/dev/misc/rtc': No such file or directory
Fri Mar 9 12:34:56 UTC 2018
huclock: can't open '/dev/misc/rtc': No such file or directory
urandom read: 2 callbacks suppressed
random: dd: uninitialized urandom read (512 bytes read)
Configuring packages on first boot....
(This may take several minutes. Please do not power off the machine.)
Running postinst /etc/rpm-postinsts/100-sysvinit-inittab...
update-rc.d: /etc/init.d/run-postinsts exists during rc.d purge (continuing)
Removing any system startup links for run-postinsts ...
/etc/rcS.d/S99run-postinsts
INIT: Entering runlevel: 5
Configuring network interfaces... macb e000b000.ethernet eth0: PHY [Marvell 88E1510] (irq=POLL)
macb e000b000.ethernet eth0: configuring for phy/rnii-id link mode
done.
Starting haveged: haveged: command socket is listening at fd 3
haveged: haveged starting up

Starting Dropbear SSH server: Waiting for kernel randomness to be initialized...
haveged: haveged: ver: 1.9.13; arch: generic; vend: ; build: (gcc 10.2.0 CTV); collect: 128K
haveged: haveged: cpu: (VC); data: 16K (D); inst: 16K (D); idx: 12/40; sz: 15006/57790
haveged: haveged: tot tests(BR8): A:1/1 B:1/1 continuous tests(B): last entropy estimate 7.99994
haveged: haveged: fills: 0, generated: 0

random: crng init done
Generating 2048 bit rsa key, this may take a while...
Public key portion is:
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCMIMhKvqN1j7HuiVpNisly73SpMSq69aUbxz40q3cWES6SkTU0aTSKZze6kYT2bH60Gdb7/BtTbTQ9q3EB/Vi+C5PLFJ/+hc0K6SZHNdqRyIz+2TrxHtJL
xkTt/29u38rrrEoF.IgIzKBURu+YJUH/XCoJzeorHfZHPF1TGNY63dMoH0510Mt3d9px8oxZS3F root@zedboard
Fingerpr.int: shall! cb:cf4a8:88:da:47:17:f3:ed:c9:b3:2f:7c:e9:e1:f9:41:6a:ab:2e
dropbear.
huclock: can't open '/dev/misc/rtc': No such file or directory
Starting internet superserver: inetd.
Starting syslogd/klogd: done
Starting tcf-agent: OK

root@zedboard:~#
root@zedboard:~#
```

5 Erneute Installation bei z.B Anpassungen im Quellcode

5.1.1 Wenn sich Fehler eingeschlichen haben sollten, kann die Installation jeder Zeit auf dem Server erneut durchgeführt werden. Hier werden noch einmal kurz die hierfür benötigten Befehle aufgeführt. Diese genügen, um eine Änderung, die sich nicht auf das Image oder die sdk-Datei auswirkt, aufzuspielen. Nachdem das Vivado Projekt wieder als .xsa File exportiert wurde, kann die Datei in den PetaLinux Ordner gelegt werden.

Dieser Vorgang kann nun auch beispielsweise über den Remote Desktop ausgeführt werden.

5.1.2 Im Linux Terminal sind nun folgende Eingaben zu tätigen:

!Bitte beachten, dass die kursiven Texte projektspezifisch sind!

→ssh hoerni

Login mit Passwort

→cd *petalinux*

→source settings.sh

→cd *Ordner-der-.xsa-Datei*

→petalinux-config --get-hw-description *Projektname_wrapper.xsa*

Exit wenn Konfigurationseinstellungen geöffnet werden

→petalinux-build

→petalinux-package --boot --u-boot --format BIN --force

5.1.3 Im Anschluss erneut die drei Dateien (BOOT.BIN, BOOT.src und u_boot.ub) für den Boot Ordner kopieren und auf die SD-Karte ziehen.