

## Lab 2: Designing Hardware IP

### 3 points

As FPGAs become larger and more complex, and as design schedules become shorter, use of third-party IP and design reuse is becoming mandatory. We use a powerful feature within the Vivado Design Suite called the Vivado IP integrator.

The Vivado IP integrator lets you create complex system designs by instantiating and interconnecting IP from the Vivado IP catalog or from other sources.

In this lab we want to analyze a DCT-IP core and synthesize it onto a FPGA. The purpose is to use the hardware implementation of DCT to perform a fast JPEG conversion. For this we use the ZedBoard, which comes with Zynq device, a SoC from Xilinx. The SoC combines an ARM dual-core Cortex-A9 Processing System with FPGA programmable logic.

You go through a workflow of how to create an IP on FPGA programmable logic and how to interface it with a C++ project running on ARM processor to finally build a hardware/software-codesign. The major goal of this codesign is performance optimization.

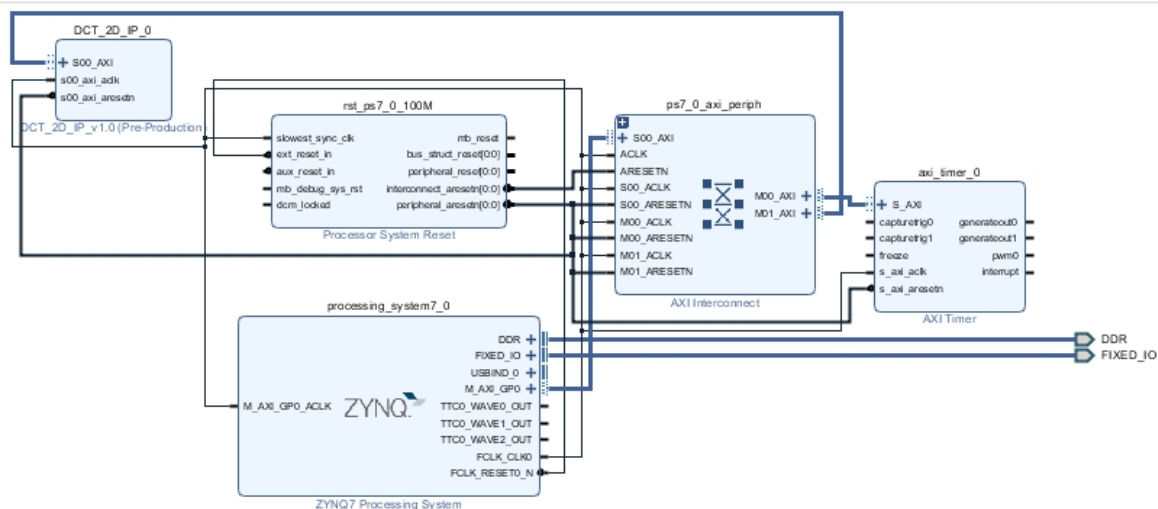


Fig. 1: JPEG-Encoder on Zynq SoC

DCT stands for discrete cosine transform and has its application in signal and image processing, especially for lossy compression of images e.g. JPEG encoding. Already, there are some implementations of the DCT for the FPGA. Within this lab session we are using a DCT-IP provided by Unicore Systems which is available for free via [opencores.org](http://opencores.org).

**Software** required for this lab:

Xilinx Vivado Design Suite 2018.2 WebPACK

**Hardware** require for this lab:

ZedBoard

2 Micro-USB cables for JTAG and UART interface

**Data-files** for this lab:

*DCT\_IP\_VHDL\_Testbench.zip*

*jpeg\_encoder\_lab\_files.zip*

**Manuals** for this lab:

*Unicore\_DCT\_IP\_user\_manual.pdf*








*ZedBoard\_HW\_UG\_v2\_2.pdf*

## Task 1: Verification of DCT-IP

### Step 1: Unzip Data-file

Copy *DCT\_IP\_VHDL\_Testbench.zip* to your project directory and unzip.

You will get the following files:

Name
 bmp_generator.vhd
 DCT8AAN1.vhd
 DCT8AAN2.vhd
 DCT_AAN.vhd
 dct_beh.vhd
 DCT_BUF.vhd
 test_dct.vhd

For the verification of the DCT-IP we need those seven VHDL-files provided by Unicore Systems.








### Step 2: Open Vivado and set up a new project

Open Vivado and select „Create Project”. Give Project name e.g. „dct\_unicore” and select a proper Project location. Select as Project Type „RTL Project”. Add Sources: all files of *DCT\_IP\_VHDL\_Testbench*. Make sure that the sources are copied into the project and the language is VHDL. Click Next.

New Project

#### Add Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those files, to add to your project on disk and add it to your project. You can also add and create sources later.

	Index	Name	Library	HDL Source For
	1	DCT8AAN1.vhd	xil_defaultlib	Synthesis & Simulation
	2	DCT8AAN2.vhd	xil_defaultlib	Synthesis & Simulation
	3	DCT_AAN.vhd	xil_defaultlib	Synthesis & Simulation
	4	DCT_BUF.vhd	xil_defaultlib	Synthesis & Simulation
	5	bmp_generator.vhd	xil_defaultlib	Synthesis & Simulation
	6	dct_beh.vhd	xil_defaultlib	Synthesis & Simulation
	7	test_dct.vhd	xil_defaultlib	Synthesis & Simulation

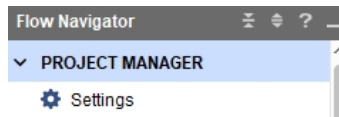
☒ Scan and add RTL include files into project  
☒ Copy sources into project  
☒ Add sources from subdirectories

Target language: VHDL Simulator language: VHDL

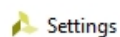
Skip the next options „Add Constraints”. To associate the project with the given hardware, select the „ZedBoard Zynq Evaluation and Development Kit” with Board Rev „c” or „d”(file version 1.4) as mentioned on your specific board. Make sure you have selected the right Board Rev. Click Next. Click Finish.

### Step 3: Change the simulation settings

In Flow Navigator click on “Settings”.



In the Tab “Simulation”, change the Simulation Run Time parameter to 20000ns to get a better testbench result overview later on. Click Apply, click OK.



Q

**Project Settings**

- General
- Simulation**
- Elaboration
- Synthesis
- Implementation
- Bitstream
- > IP

**Tool Settings**

- Project
- IP Defaults
- Source File
- Display
- WebTalk
- Help
- > Text Editor
- > 3rd Party Simulators
- > Colors
- > Selection Rules
- > Shortcuts
- > Strategies
- > Window Behavior

**Simulation**  
Specify various settings associated to Simulation

Target simulator:

Vivado Simulator

Simulator language:

VHDL

Simulation set:

sim\_1

Simulation top module name:

TEST\_DCT

Compilation

Elaboration

**Simulation**

Netlist

Advanced

xsim.simulate.tcl.post	
<b>xsim.simulate.runtime</b>	20000ns
xsim.simulate.log_all_signals	<input type="checkbox"/>
xsim.simulate.custom_tcl	
xsim.simulate.wdb	
xsim.simulate.saif_scope	
xsim.simulate.saif	
xsim.simulate.saif_all_signals	<input type="checkbox"/>
xsim.simulate.xsim.more_options	

**xsim.simulate.runtime**  
Specify simulation run time

?

OK

Cancel

Apply

### Step 4: Running a simulation

In Flow Navigator click on “Run Simulation”, select “Run Behavioral Simulation”.

A new window is opened and the simulation waveform window is available.

After finishing with tasks 2a/b/c, close project!

## Questions:

1. Analysis of the Custom IP
  - a) Draw a block diagram of the VHDL module „DCT\_AAN”! Briefly explain with the help of IP documentation how to calculate the 2D DCT of a 8x8 matrix.  
Hint: You may have to analyze its subcomponents.
  - b) A test bench for the DCT-IP core is also available. Explain the functions of the three components „TEST\_DCT“, „DCT\_BEH“ and „BMP\_Generator“!  
What function does the process „ERROR\_CALC“ have?
2. Verification of the Custom IP
  - a) Explain the meaning of the following output signals: DCT[11:0], DCT\_STD[11:0], ERROR, QUADMEAN!
  - b) Verify the functionality of the custom IP using the provided test bench.  
Simulate at least for 20µs (step 4). What is the delay (number of clock cycles) between first data input and first data output? How many clock cycles does it take to read in one input matrix? Give the min. and max. value of mean square error for this simulation! Attach simulation snapshots to verify your findings!

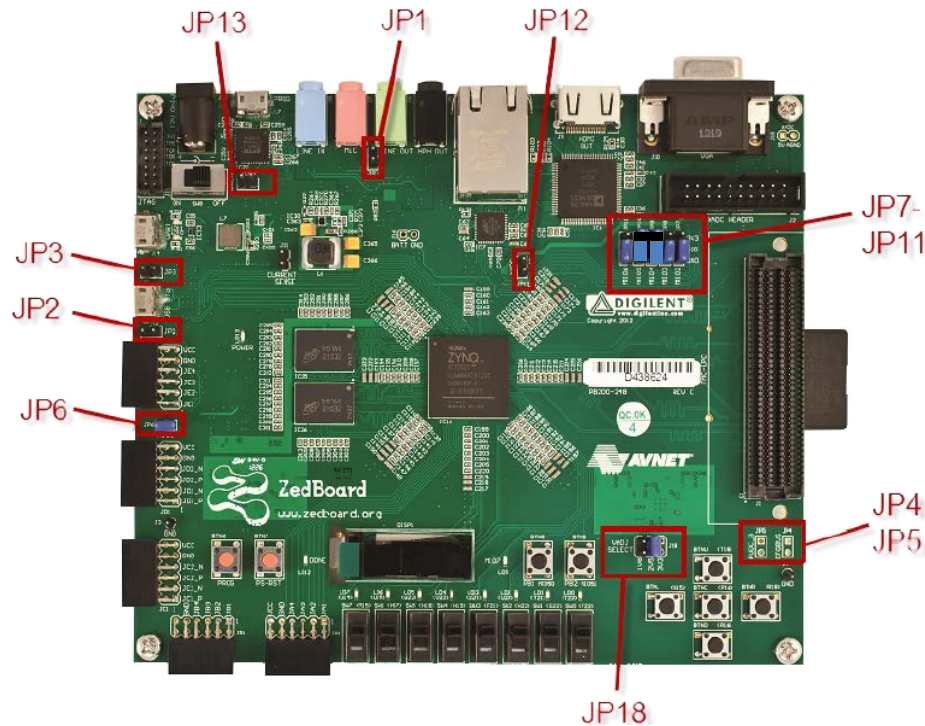
## Task 2: Integration of DCT-IP into JPEG encoder project

Aim is the integration of the hardware custom IP into an existing JPEG encoder project. Go through the workflow of creating a custom IP on FPGA and integrate the IP into a JPEG encoder project to finally build a hardware/software-codesign.

### Step 1: Set Up ZedBoard

Connect JTAG and UART interface to your host computer using Micro-USB cables.

Jumper Settings: Make sure Jumpers **JP7-JP11** are **set to GND**. Jumper JP18 is set to 2.5V.



Plug the power supply cable into the ZedBoard. Now switch on the power supply. You may see the following devices in Windows: (if no driver for USB\_UART can be detected, install the Cypress driver).

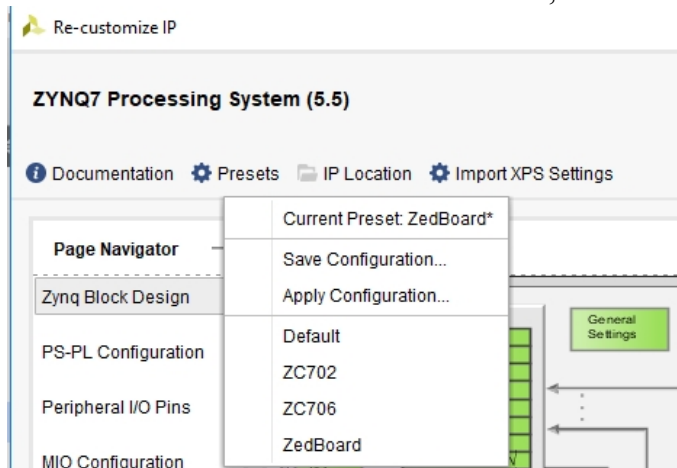


## Step 2: Create PS

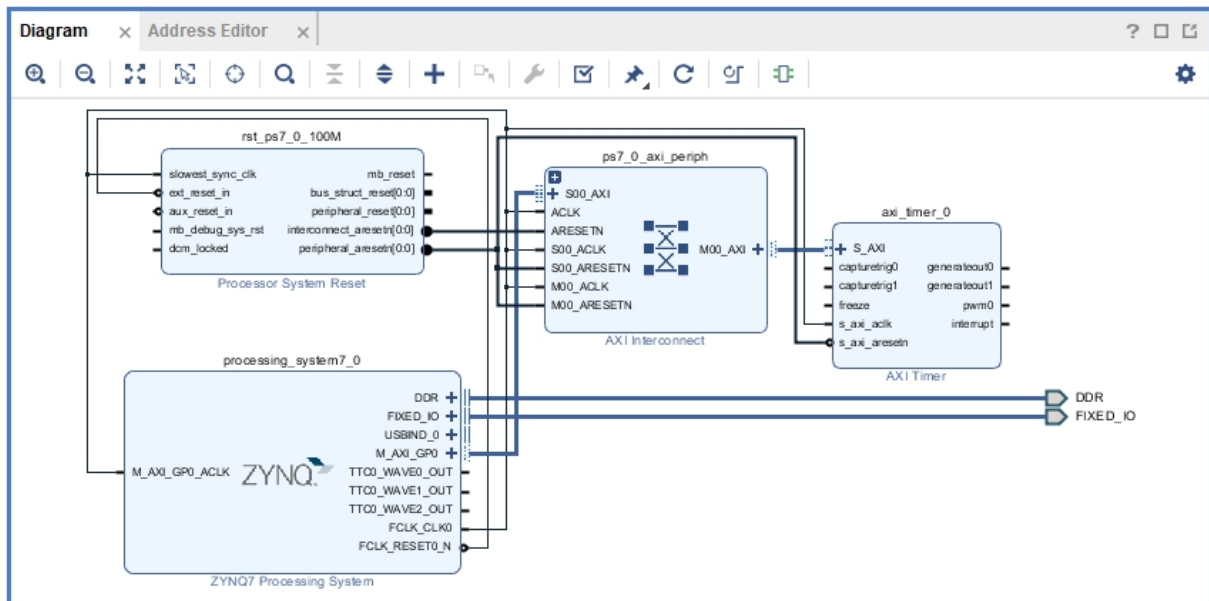
Create new Vivado Project, name: jpeg\_encoder\_for\_lab, RTL project, select ZedBoard.

Create Block Design: Add IP **Zynq7 Processing System** and IP **AXI Timer**.

Double-click on ZYNQ block: this opens a window to Re-customize IP. Click on Presets and check if current Preset is ZedBoard! If not, select ZedBoard.



Run Connection Automation.



### Step 3: Create a new DCT-IP

From the top toolbar, select „Tools” and „Create and Package New IP”. Click Next. Select “Create a new AXI4 peripheral” to allow a connection between the ARM and FPGA via the AXI4Lite Bus. Click Next.

Create and Package New IP

### Create Peripheral, Package IP or Package a Block Design

Please select one of the following tasks.

#### Packaging Options

- ☐ Package your current project  
Use the project as the source for creating a new IP Definition.
- ☐ Package a block design from the current project  
Choose a block design as the source for creating a new IP Definition.  
Select a block design:
- ☐ Package a specified directory  
Choose a directory as the source for creating a new IP Definition.

#### Create AXI4 Peripheral

- ☒ Create a new AXI4 peripheral  
Create an AXI4 IP, driver, software test application, IP Integrator AXI4 VIP simulation and debug demonstration design.

?
< Back
Next >
Finish

**Important:** Select as name „**DCT\_2D\_IP**”, description and the IP location to „..\\jpeg\_encoder\_for\_lab\\ip\_repo”.

Remark: The AXI4Lite interface allows data exchange between the ARM and FPGA by providing some mapped registers that can be accessed from the ARM and FPGA side. If you are interesting to get more knowledge about the interface the internet might help you.

Create and Package New IP

×

### Peripheral Details

Specify name, version and description for the new peripheral



Name: DCT\_2D\_IP  
Version: 1.0  
Display name: DCT\_2D\_IP\_v1.0  
Description: My new AXI IP  
IP location: C:/Users/Schumann3/h\_da/Vivado\_2018\_2\_projects/lab2/jpeg\_encoder\_for\_lab/ip\_repo

☐ Overwrite existing

Click Next.

By adding the AXI4 interface, select Interface Type Lite in Slave Mode. This will create 32 Bits registers which are mapped to both, ARM and FPGA on the SoC. For the number of registers, type 50. Click Next.

Create and Package New IP

×

### Add Interfaces

Add AXI4 interfaces supported by your peripheral



☐ Enable Interrupt Support

+ -

Interfaces  
S00\_AXI


S00\_AXI  
DCT\_2D\_IP\_v1.0

Name: S00\_AXI  
Interface Type: Lite  
Interface Mode: Slave  
Data Width (Bits): 32  
Memory Size (Bytes): 64  
Number of Registers: 50 [4..512]

Now we want to edit the IP to integrate the DCT functionality. Select „Edit IP” and click Finish.



Create and Package New IP



### Create Peripheral

Peripheral Generation Summary

1. IP (xilinx.com:user:DCT\_2D\_IP:1.0) with [1 interface\(s\)](#)
2. Driver(v1\_00\_a) and testapp [more info](#)
3. AXI4 VIP Simulation demonstration design [more info](#)
4. AXI4 Debug Hardware Simulation demonstration design [more info](#)


Peripheral created will be available in the catalog :

C:/Users/Schumann3/h\_da/Vivado\_2018\_2\_projects/lab2/jpeg\_encoder\_for\_lab/ip\_repo

Next Steps:

- ☐ Add IP to the repository
- ☒ Edit IP
- ☐ Verify Peripheral IP using AXI4 VIP
- ☐ Verify peripheral IP using JTAG interface

Click Finish to continue



< Back

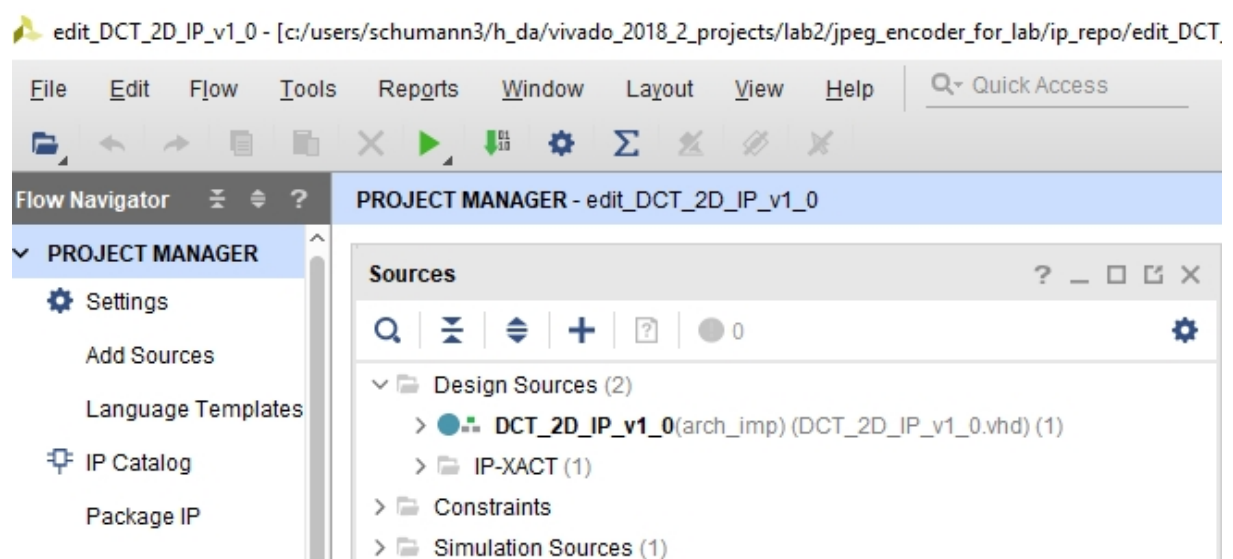
Next >

Finish

A new window “edit\_DCT\_2D\_IP\_v1\_0” pops up.

#### Step 4: Add DCT VHDL files to the IP peripheral

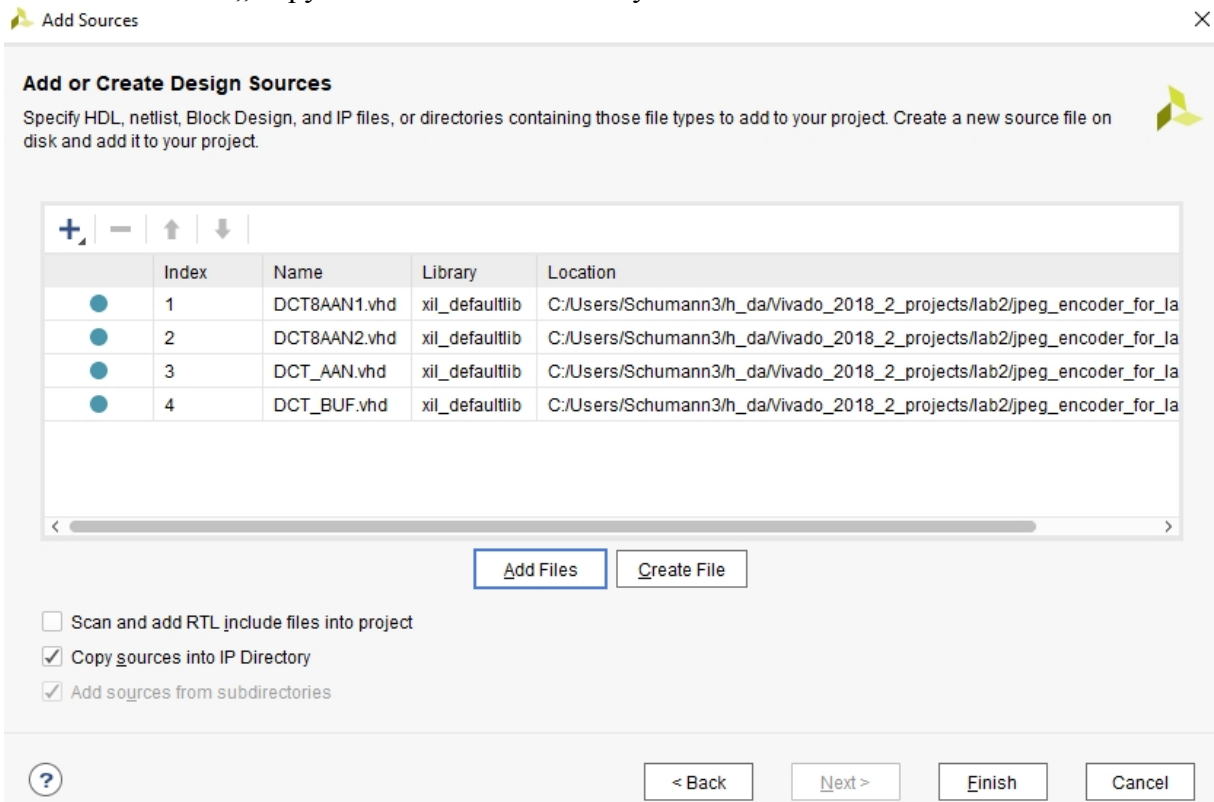
In the this new window, from the Flow Navigator, click ”Add Sources”.





Select „Add or create design sources” and click Next.

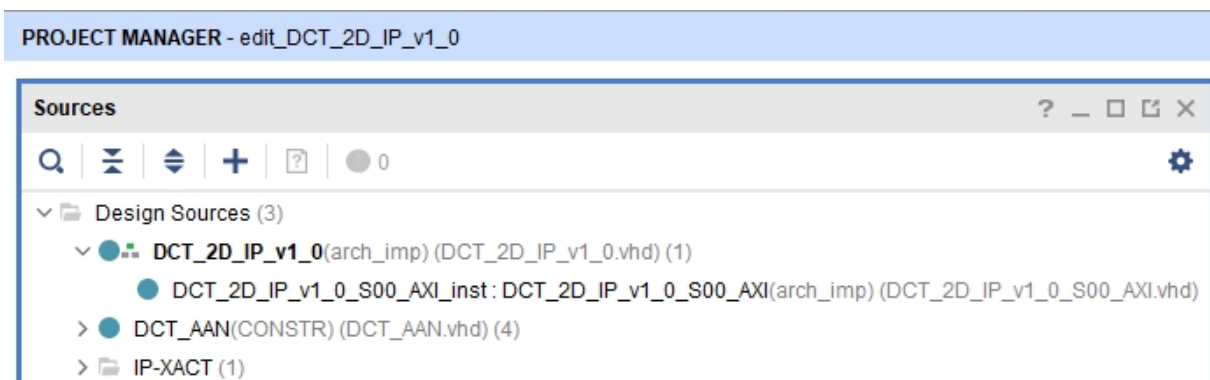
Click on Add Files: Look in *jpeg\_encoder\_for\_lab* folder, double click on *DCT\_sources* folder and select all 4 source files „DCT\_AAN”, „DCT\_BUF”, „DCT8AAN1” and „DCT8AAN2”. Make sure to tick „Copy sources into IP Directory” and then click Finish.



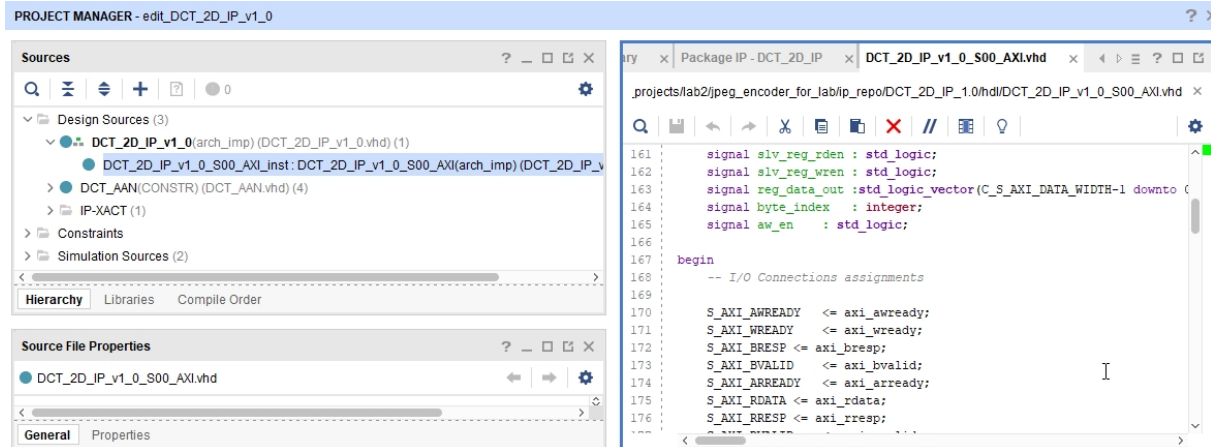
The DCT core is now added to the IP Peripheral.

### Step 5: Modify the IP Peripheral

Click on Design Source „DCT\_VHDL\_IP\_v1\_0”.



Double click on the „DCT\_2D\_IP\_v1\_0\_S00\_AXI\_inst” to open VHDL-file of transport buffer.



Now we need to replace file content of DCT\_2D\_IP\_v1\_0\_S00\_AXI.vhd with new interface definition file dct\_2d\_axi\_step5.vhd, provided as part of jpeg\_encoder\_lab\_files.zip.

In the editor: right click -> select all, delete

Copy file content of **dct\_2d\_axi\_step5.vhd** into the editor window. Click on Save File.

### Step 6: Instantiate the DCT\_AAN core

As the data handling between the IP Core and the ARM Processor via the introduced buffer is established, your task is to instantiate the DCT\_AAN component within the VHDL-file „DCT\_VHDL\_IP\_v1\_0\_S00\_AXI.vhd”. Move to the end of the file and complete the port map section of DCT\_AAN instantiation!

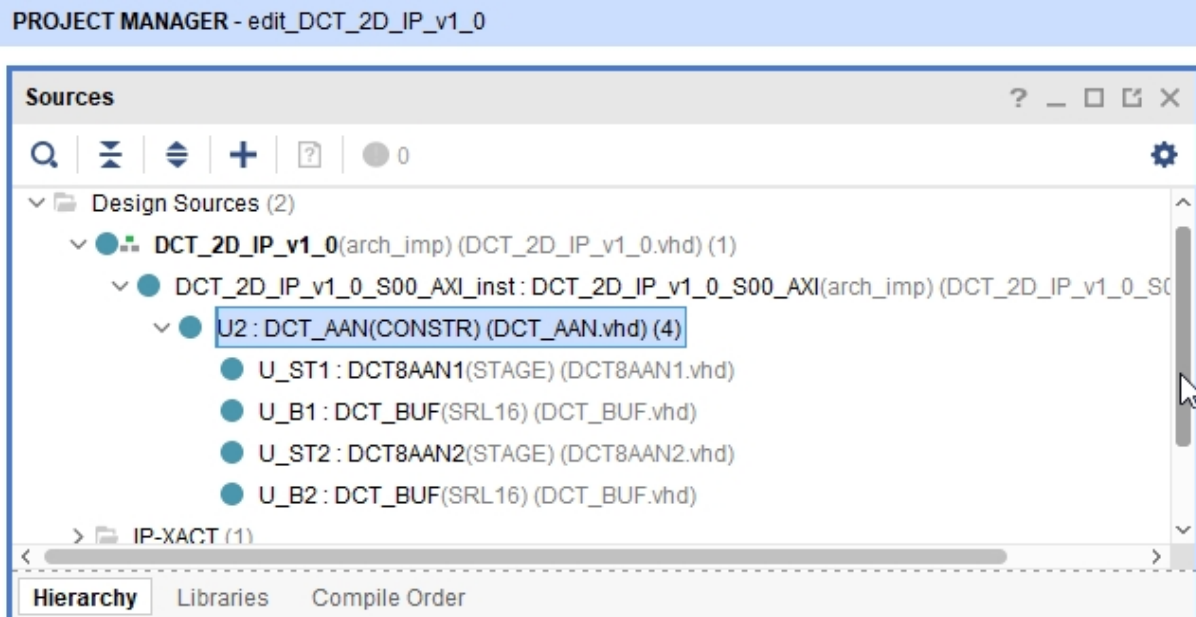
U2: DCT\_AAN

```
generic map( d_SIGNED => 0, scale_out => 1)
port map(
  CLK =>
  RST => not(S_AXI_ARESETN),
  START =>
  EN => slv_reg49(1),
  DATA_IN =>
  DATA_OUT =>
  RDY => dct_core_rdy
);

-- User logic ends
```

When you are done, click on Save File.

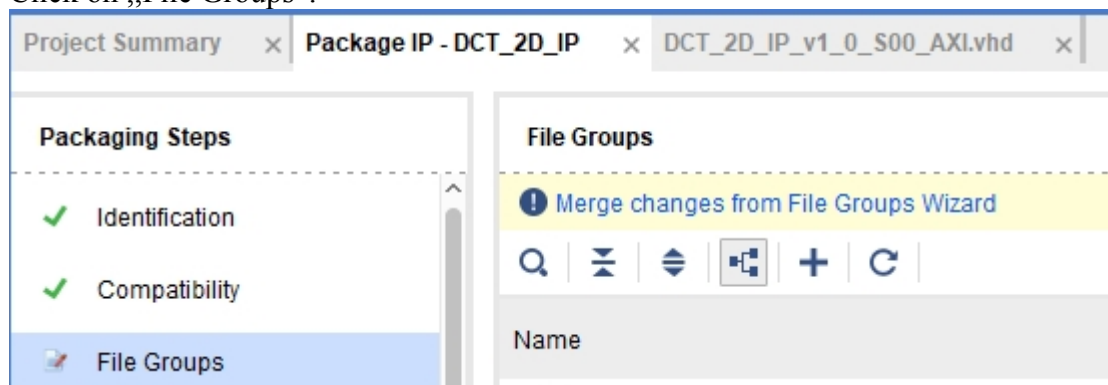
If you did this step correctly, you should notice from the Sources window that the „DCT\_AAN.vhd” file has been integrated into the hierarchy because of the instantiation within the peripheral.



### Step 7: Repackage IP

Click on **Package IP** in Flow Navigator.

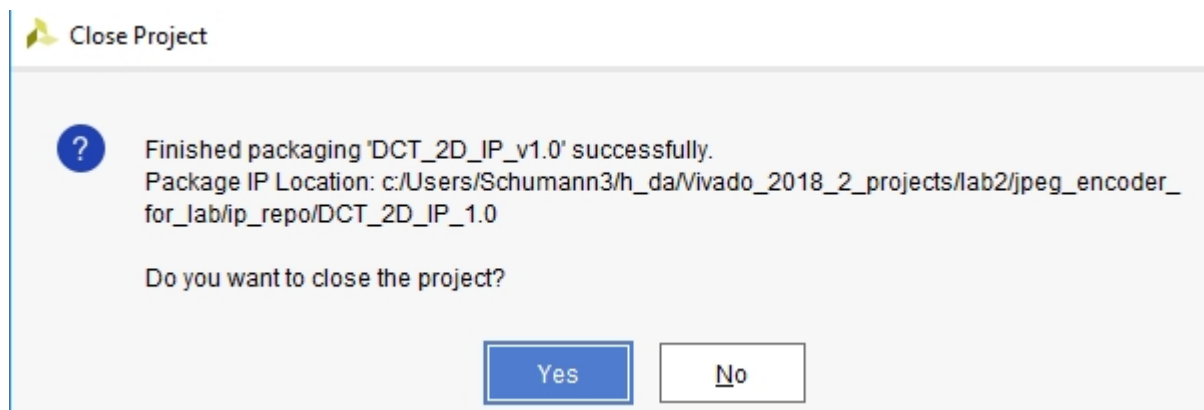
Click on „File Groups”.



Click on „Merge changes from File Groups Wizard”. The „File Groups” should now have a green tick.

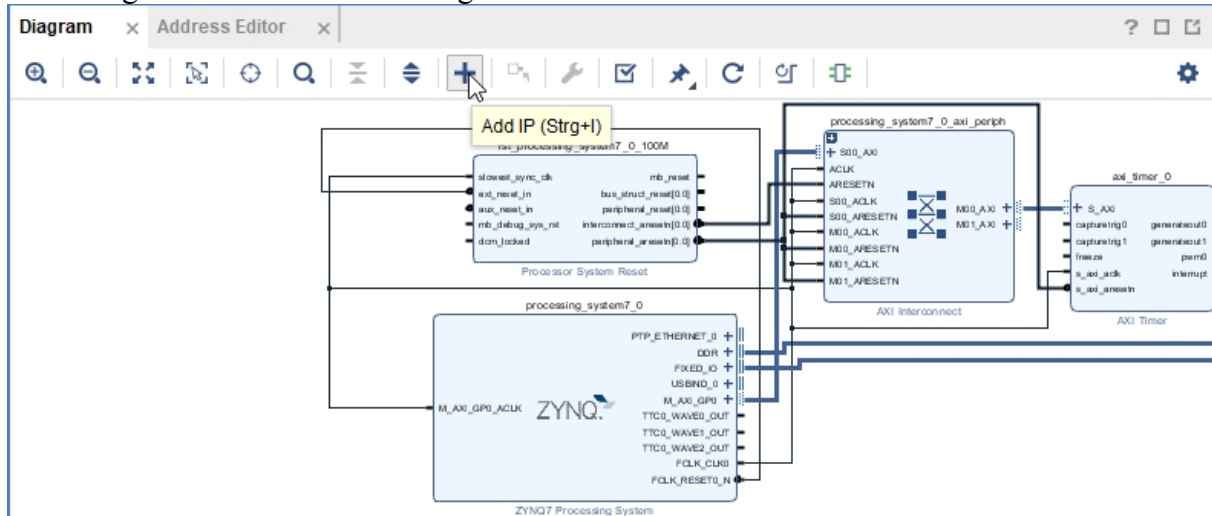
Now click „Review and Package” and click on the Button „Re-Package IP”.

DCT IP is configured and available in the IP Catalog. Click Yes to close the IP project.



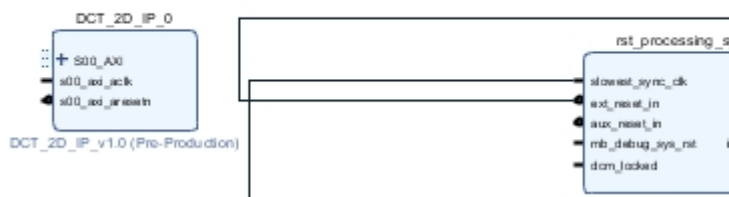
### Step 8: Add DCT IP to the design

Now we go back to our Block Design and click on “Add IP”.



Search for „DCT\_2D\_IP\_v1.0” and select it. The block should appear in the Block Diagram and you should see the message „Designer Assistance available. Run Connection Automation”. Click on Run Connection Automation.

Designer Assistance available. [Run Connection Automation](#)



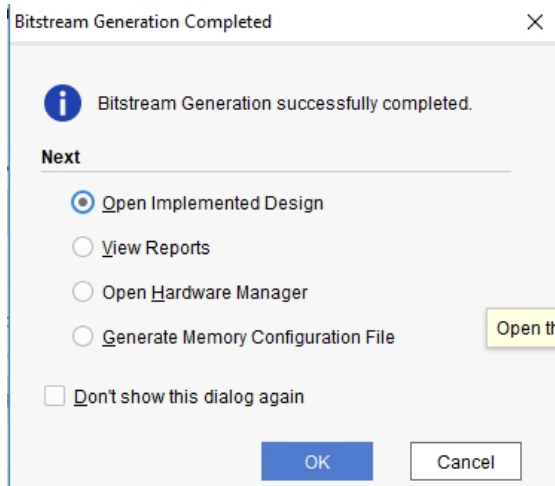
In the window that appears, set Clock source for slave interface to ”Auto” and click OK. Congratulations, our IP is successfully added! Click on Save Block Design.

Click Validate Design.

Create a HDL Wrapper: In the Block Design window, tab Sources, click on “design\_1.bd” , right click: select Create HDL Wrapper and click OK on next window. This will create a top module in VHDL and will allow you to generate a bitstream.

### Step 9: Generate the final bitstream for the FPGA

In Flow Navigator click on Generate Bitstream. Click OK for default settings.



Bitstream Generation successfully completed window opens, click Cancel.

### Step 10: Export your hardware to SDK

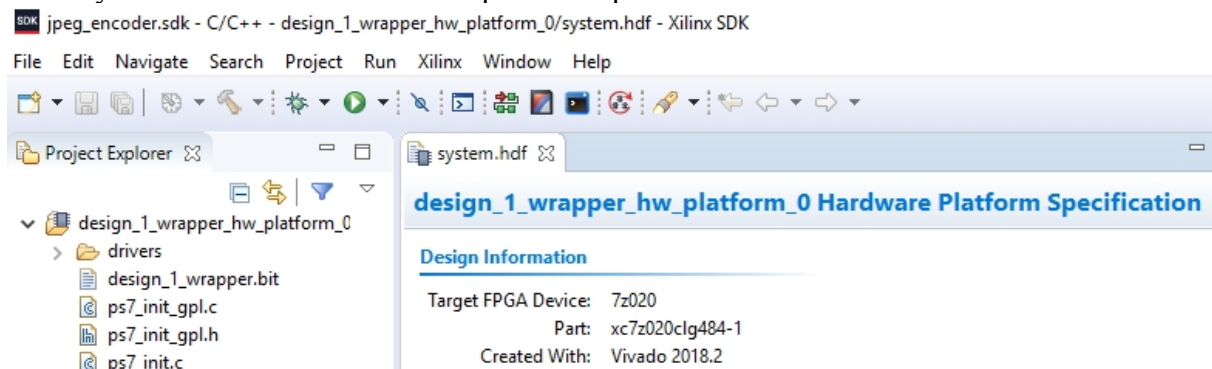
The development of hardware DCT on FPGA is done, so we can switch to the SDK to get the remaining JPEG C-code running on ARM processor.

In Vivado: File -> Export -> Export Hardware. In the new window, check the „Include bitstream” and click „OK”.

File -> Launch SDK. Use the default settings and click OK. The SDK shall now open.

### Step 11: Define hardware platform and board support package

Window “Load Hardware Specification in Background” pops up: Click OK. Project Explorer shows you one folder with hardware platform specification.



You have to create the board support package for your design.

In SDK: File -> New -> Board Support Package

**New Board Support Package Project**

Create a Board Support Package.

Project name:

☒ Use default location

Location:

Choose file system:

**Target Hardware**

Hardware Platform:

CPU:

Compiler:

**Board Support Package OS**

Standalone is a simple, low-level software layer. It provides access to basic processor features such as caches, interrupts and exceptions as well as the basic features of a hosted environment, such as standard input and output, profiling, abort and exit.

Don't change default settings. Click on Finish.

In the Board Support Package Settings, include the "Generic Fat File System Library" to support SD card access.

**Board Support Package Settings**

Control various settings of your Board Support Package.

**Overview**

- standalone
  - xilffs
  - drivers
    - ps7\_cortexa9\_0

**standalone\_bsp\_1**

OS Type:  Standalone is a simple, low-level software layer. It provides access to basic process caches, interrupts and exceptions as well as the basic features of a hosted environ input and output, profiling, abort and exit.

OS Version:

**Target Hardware**

Hardware Specification:

**Supported Libraries**

Check the box next to the libraries you want included in your Board Support Package. You can configure the library in th

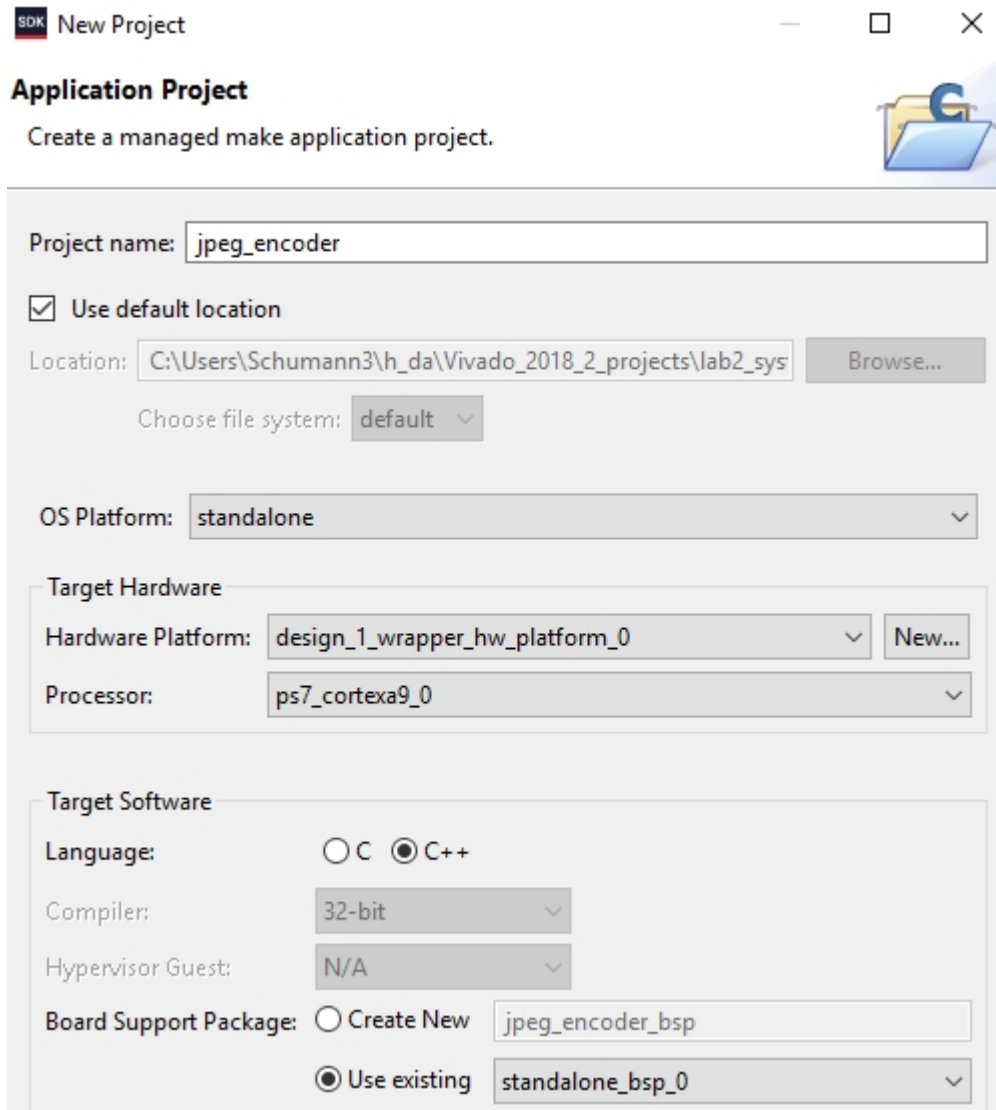
Name	Version	Description
<input type="checkbox"/> libmetal	1.4	Libmetal Library
<input type="checkbox"/> lwip202	1.1	Lwip202 library: lwIP (light weight IP) is an open sour...
<input type="checkbox"/> openamp	1.5	OpenAmp Library
<input checked="" type="checkbox"/> xilffs	3.9	Generic Fat File System Library

Finally, click "OK".

## Step 12: Get C-Program of JPEG-encoder

DCT is done on Programmable Logic (FPGA/PL), but remaining algorithms for JPEG-encoding is running as software on Processing System (ARM/PS).

File -> New -> Application Project



**Application Project**  
Create a managed make application project.

Project name:

☒ Use default location

Location:

Choose file system:

OS Platform:

Target Hardware

Hardware Platform:

Processor:

Target Software

Language: ☐ C ☒ C++

Compiler:

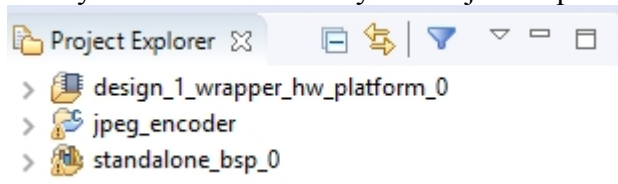
Hypervisor Guest:

Board Support Package: ☐ Create New

☒ Use existing

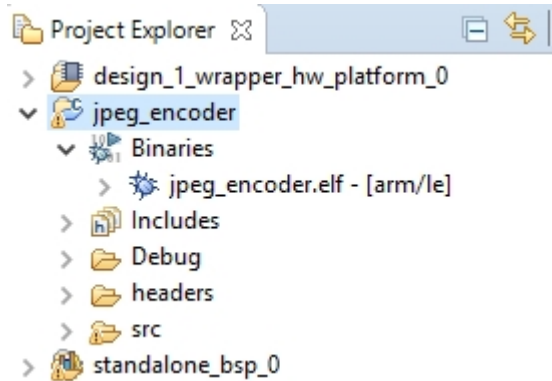
Click Finish.

Now you have 3 folders in your Project Explorer:



Expand jpeg\_encoder: delete folder src! Copy folder **src** from your unzipped JPEG\_encoder file. Copy also folder **headers**! Now a build process is starting and you get a *jpeg\_encoder.elf* file in your Binaries!





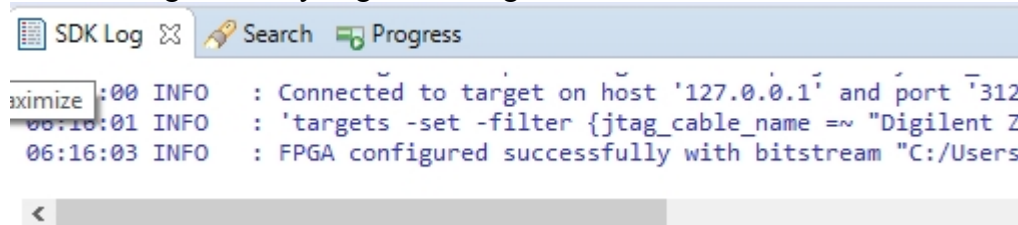
### Step 13: Run the project

If not yet done: **Switch on power supply** of ZedBoard! JTAG and UART cable should be connected to host computer.

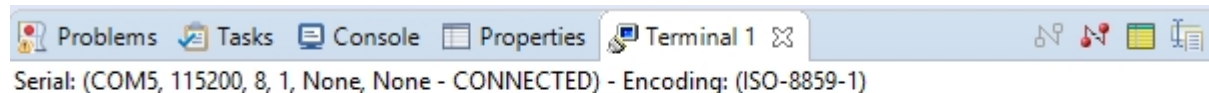
Click on „Program FPGA” within the SDK.


This will program the FPGA on the ZedBoard with our previously generated bitstream which was exported to the SDK. Note: The DONE LED on the board turns **blue**.

On SDK Log window you get a message:

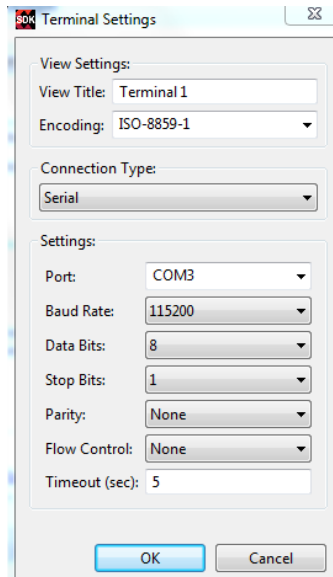


Now, make sure to use the SDK build-in UART terminal, as the JPEG conversion gives some status information to the UART interface.



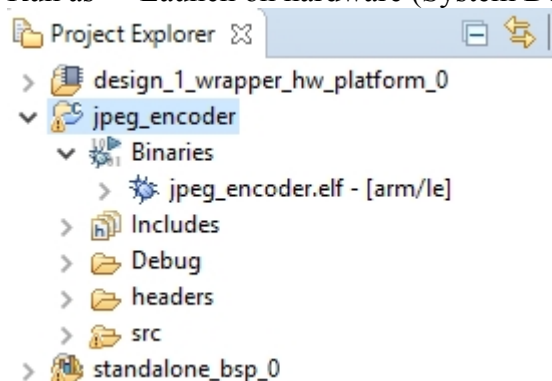
Click on . Set the connection type to **serial**. For proper COM port configuration, check your device manager on your computer. Set Baud Rate to 115200. Click on OK. Click on

 to connect Terminal 1.

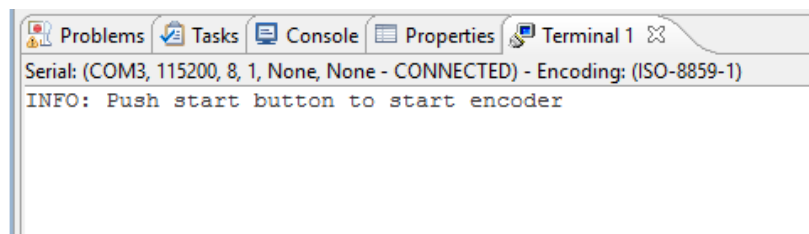


Insert the SD card to your computer and copy *bmp* folder to SD card. Also create a folder *jpg* on SD card. Put SD card into ZedBoard SD card slot.

Ensure the BMP files are located in the “bmp” folder in the root directory of the SD card. Now run the C++ project: In Project Explorer select *jpeg\_encoder* and right click: Run as -> Launch on hardware (System Debugger)



After loading the application, you should notice the message from the UART terminal:



Click on **PB1** on the ZedBoard. This push button is located just above the DIP switches.

The JPEG conversion will now start. All BMP files located in the *bmp* folder will be converted to JPEG and getting saved in *jpg* folder.

The JPEG conversion is finished after getting the measurement statistics.

```
Serial: (COM3, 115200, 8, 1, None, None - CONNECTED) - Encoding: (ISO-8859-1)
INFO: Converting image row: 128 of 128
INFO: JPEG Image conversion for: /JPG/IMAGE01.JPG finished.
INFO: JPEG File size: 422544 byte
INFO: JPEG compression rate: 500
INFO: JPEG image /BMP/IMAGE01.BMP was encoded in: 18952ms
INFO: I/O time amount: 3660ms
INFO: Bitmap Block extraction time amount: 178ms
INFO: DCT time amount: 6481ms
INFO: Color Conversion time amount: 2041ms
INFO: Quantization time amount: 2389ms
INFO: Huffman encoding time amount: 4134ms
```

**Step 14:** Do a screen shot of statistics shown in serial terminal!

Insert the SD card to your computer and verify the JPEG pictures: Compare BMP to JPEG pictures regarding size and quality!

If the results are not as expected, make sure you have done Step 5 correctly.

**Prepare before lab:** Task 1, question 1 a+b

**Note:** Tutor will check on your preparation at the beginning of lab! No preparation means you need to leave lab with no points!

Lab report: deadline is **Dec.10!** Submit your lab report as hardcopy (group of 2 students).

Include:

1. Answers to question 1 a+b!
2. Answers to question 2 a+b+c!
3. VHDL-code for instantiation of the DCT core!
4. Screen shot of statistics in Terminal 1 (step14)! Compare BMP vs JPEG regarding size and quality!