

Instruction

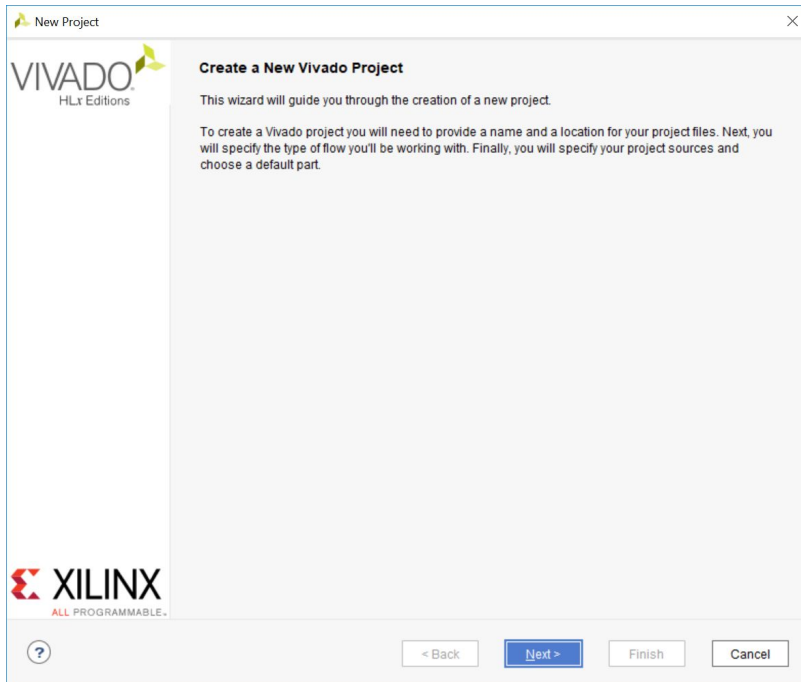
This Instruction will help to use the HAW_RISC_V CPU architecture and run a assembler program.

Create a Project	2
Generate a Clock	5
Generate a Block RAM with the Assembler program	7
Run the Synthesis	10
Run the Implementation	11
Generate a Bitstream	12
Program a Bitstream on the ZedBoard	13
Create own coe files for HAW RISC V CPU	13

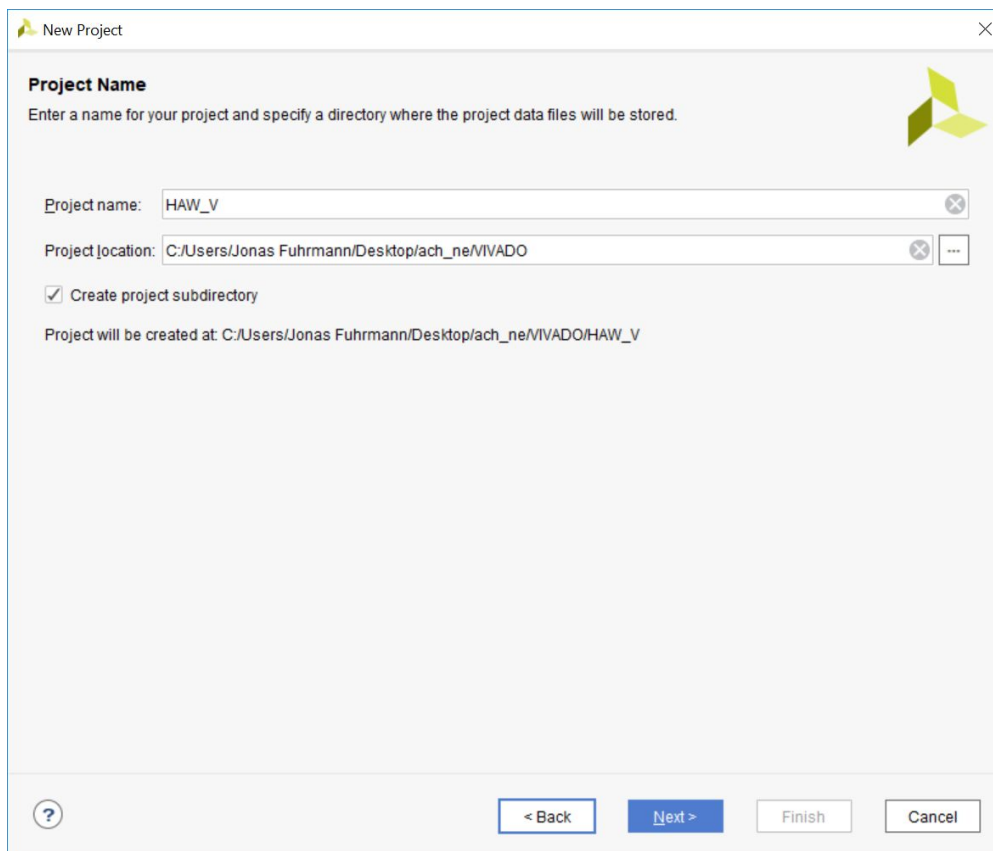
Create a Project

- Open Vivado

Create a new project and follow the wizard from Vivado as shown below:



Click on **Next>** to continue.



Enter a project name and a project location and select the checkbox to create project subdirectories. Click on **Next>** to continue.

New Project

Project Type
Specify the type of project to create.


☒ **RTL Project**
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.
☐ Do not specify sources at this time

☐ **Post-synthesis Project**: You will be able to add sources, view device resources, run design analysis, planning and implementation.
☐ Do not specify sources at this time

☐ **I/O Planning Project**
Do not specify design sources. You will be able to view part/package resources.

☐ **Imported Project**
Create a Vivado project from a Synplify, XST or ISE Project File.

☐ **Example Project**
Create a new Vivado project from a predefined template.



Choose *RTL Project* and click on **Next>** to continue.


New Project

Add Sources
Specify HDL, netlist, Block Design, and IP files, or directories containing those files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.

Index	Name	Library	HDL Source For	Location
1	risc_v_core.vhdl	xil_defaultlib	Synthesis & Simulation	C:/Users/Jonas Fuhrmann/Desktop
2	riscv_pack.vhd	xil_defaultlib	Synthesis & Simulation	C:/Users/Jonas Fuhrmann/Desktop
3	top_level.vhdl	xil_defaultlib	Synthesis & Simulation	C:/Users/Jonas Fuhrmann/Desktop
4	adder.vhdl	xil_defaultlib	Synthesis & Simulation	C:/Users/Jonas Fuhrmann/Desktop
5	alu.vhdl	xil_defaultlib	Synthesis & Simulation	C:/Users/Jonas Fuhrmann/Desktop
6	branch_checker.vhdl	xil_defaultlib	Synthesis & Simulation	C:/Users/Jonas Fuhrmann/Desktop
7	execute_stage.vhdl	xil_defaultlib	Synthesis & Simulation	C:/Users/Jonas Fuhrmann/Desktop
8	decode.vhd	xil_defaultlib	Synthesis & Simulation	C:/Users/Jonas Fuhrmann/Desktop
9	instruction_decode.vhd	xil_defaultlib	Synthesis & Simulation	C:/Users/Jonas Fuhrmann/Desktop

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

Target language: VHDL Simulator language: VHDL



Add all *.vhd/.vhdl* source from all subdirectories to the Vivado project. All *.vhd/* files with **_tb** are not needed. Click on **Add Files**.

Select VHDL as Target and Simulator language.

Click on **Next>** to continue.

New Project

Add Constraints (optional)

Specify or create constraint files for physical and timing constraints.

Constraint File	Location
top_level.xdc	C:\Users\Jonas Fuhrmann\Desktop\lach_ne-2017-2018\src\cpu\vivado\constraints

☐ Copy constraints files into project

Add the top_level.xdc file as an constraints file. Click on **Add Files** and then on **Next>** to continue.

New Project

Default Part









Choose a default Xilinx part or board for your project. This can be changed later.

Select: ☒ Parts ☐ Boards

Filter/Preview

Vendor: All
 Display Name: All
 Board Rev: Latest

Search:

Display Name	Vendor	Board Rev	Part	I/O Pin Count	F
 ZedBoard Zynq Evaluation and Development Kit	em.avnet.com	d	 xc7z020clg484-1	484	1
 Artix-7 AC701 Evaluation Platform	xilinx.com	1.1	 xc7a200tfg676-2	676	1
 Kintex UltraScale+ KCU116 Evaluation Platform	xilinx.com	1.0	 xc7k50p-ffvb676-2-e	676	1
 ZYNO-7 ZC702 Evaluation Board	xilinx.com	1.0	 xc7z020cln484-1	484	1

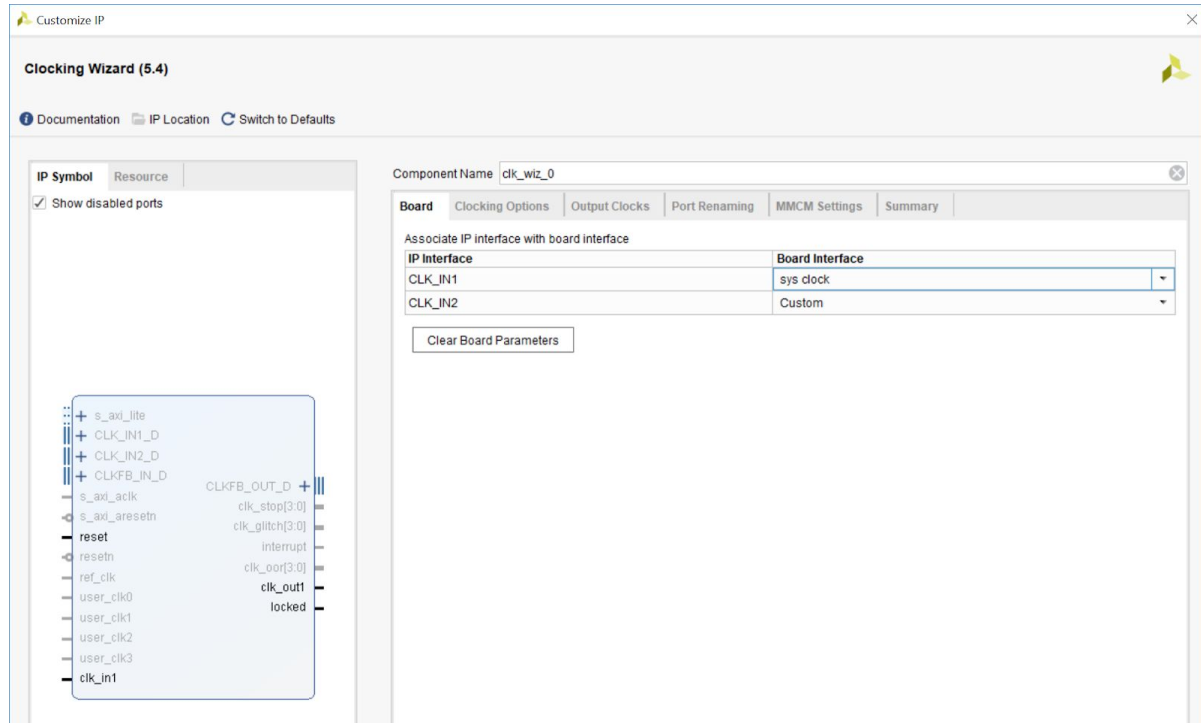
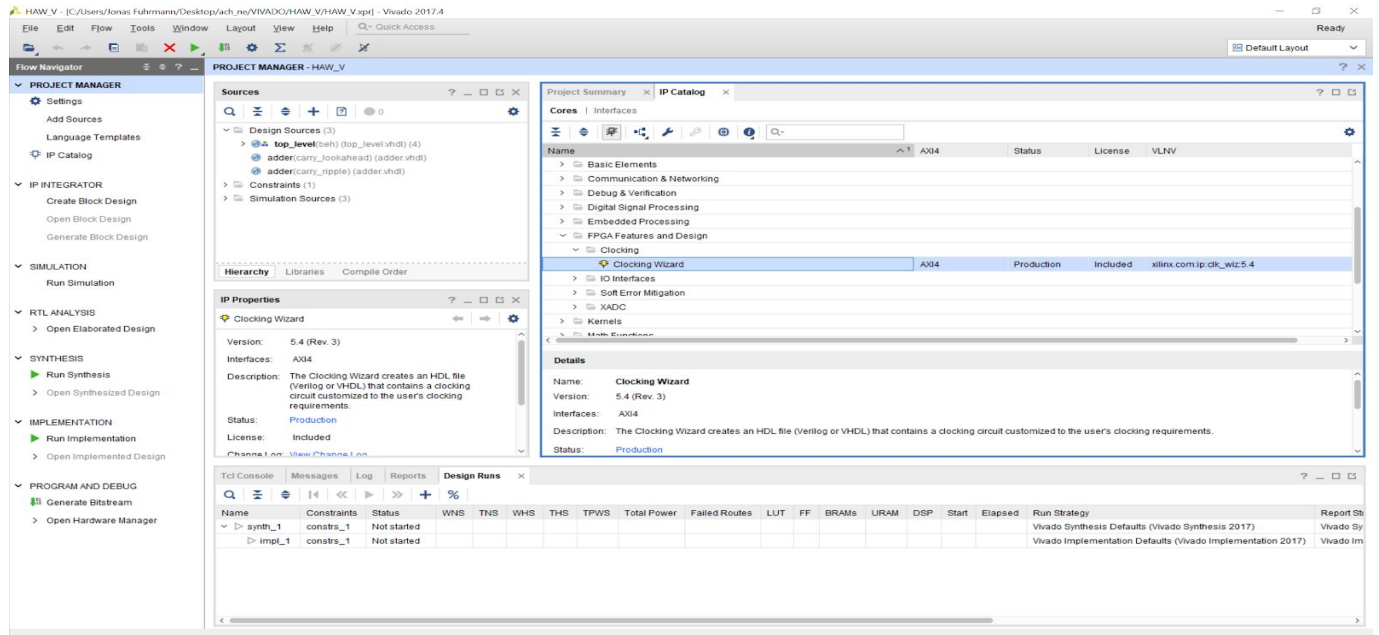
No Board Connectors

Select *ZedBoard Zynq Evaluation and Development Kit* from the given Boards and click on **Next>** to finish the wizard.

Generate a Clock

The CPU needs a clock with 50 MHz, but the Board runs with 100 MHz System Clock. Therefore we use the Ip Catalog from Vivado to GEnerate a PLL with 50 MHz.

Click on **Ip Catalog** on the Project Manager and open the **Clocking Wizard**.



Select the **sys clock** as `CLK_IN1`. The Component Name should be `clk_wiz_0`!

Click on the index tab **Clocking Options** to continue.

Re-customize IP

Clocking Wizard (5.4)

Documentation IP Location Switch to Defaults

IP Symbol Resource

☒ Show disabled ports

Component Name: `clk_wiz_0`

Board Clocking Options Output Clocks Port Renaming PLLE2 Settings Summary

Clock Monitor

☐ Enable Clock Monitoring

Primitive

☐ MMCM ☒ PLL

Clocking Features

☒ Frequency Synthesis ☐ Minimize Power

☒ Phase Alignment

☐ Dynamic Reconfig

☒ Safe Clock Startup

Jitter Optimization

☒ Balanced

☐ Minimize Output Jitter

☐ Maximize Input Jitter filtering

Dynamic Reconfig Interface Options

☒ AXI4Lite ☐ DRP

☐ Phase Duty Cycle Config ☐ Write DRP registers

Input Clock Information

Input Clock	Port Name	Input Frequency(MHz)	Jitter Options	Input Jitter	Source
<input checked="" type="checkbox"/> Primary	<code>clk_in1</code>	100.000	UI	0.010	Single ended
<input type="checkbox"/> Secondary	<code>clk_in2</code>	100.000	94.118 - 188.235	0.010	Single ended

OK Cancel

Disable *Clock Monitoring*. Set everything as seen above!

Set the Port Name as `clk_in1` and set the Jitter Options to *UI* and Input Jitter to *0.010*.

Click on the index tab **Output Clocks**.

Re-customize IP

Clocking Wizard (5.4)

Documentation IP Location Switch to Defaults

IP Symbol Resource

☒ Show disabled ports

Component Name: `clk_wiz_0`

Board Clocking Options **Output Clocks** Port Renaming PLLE2 Settings Summary

The phase is calculated relative to the active input clock.

Output Clock	Port Name	Output Freq (MHz)		Phase (degrees)		Duty Cycle (%)	
		Requested	Actual	Requested	Actual	Requested	Actual
<input checked="" type="checkbox"/> <code>clk_out1</code>	<code>clk_out1</code>	50.000	50.000	0.000	0.000	50.000	50.0
<input type="checkbox"/> <code>clk_out2</code>	<code>clk_out2</code>	100.000	N/A	0.000	N/A	50.000	N/A
<input type="checkbox"/> <code>clk_out3</code>	<code>clk_out3</code>	100.000	N/A	0.000	N/A	50.000	N/A
<input type="checkbox"/> <code>clk_out4</code>	<code>clk_out4</code>	100.000	N/A	0.000	N/A	50.000	N/A
<input type="checkbox"/> <code>clk_out5</code>	<code>clk_out5</code>	100.000	N/A	0.000	N/A	50.000	N/A
<input type="checkbox"/> <code>clk_out6</code>	<code>clk_out6</code>	100.000	N/A	0.000	N/A	50.000	N/A

☐ USE CLOCK SEQUENCING

Clocking Feedback

Source

☒ Automatic Control On-Chip

☐ Automatic Control Off-Chip

☐ User-Controlled On-Chip

☐ User-Controlled Off-Chip

Signaling

☒ Single-ended

☐ Differential

Enable Optional Inputs / Outputs for MMCM/PLL

☐ reset ☐ power_down ☐ locked

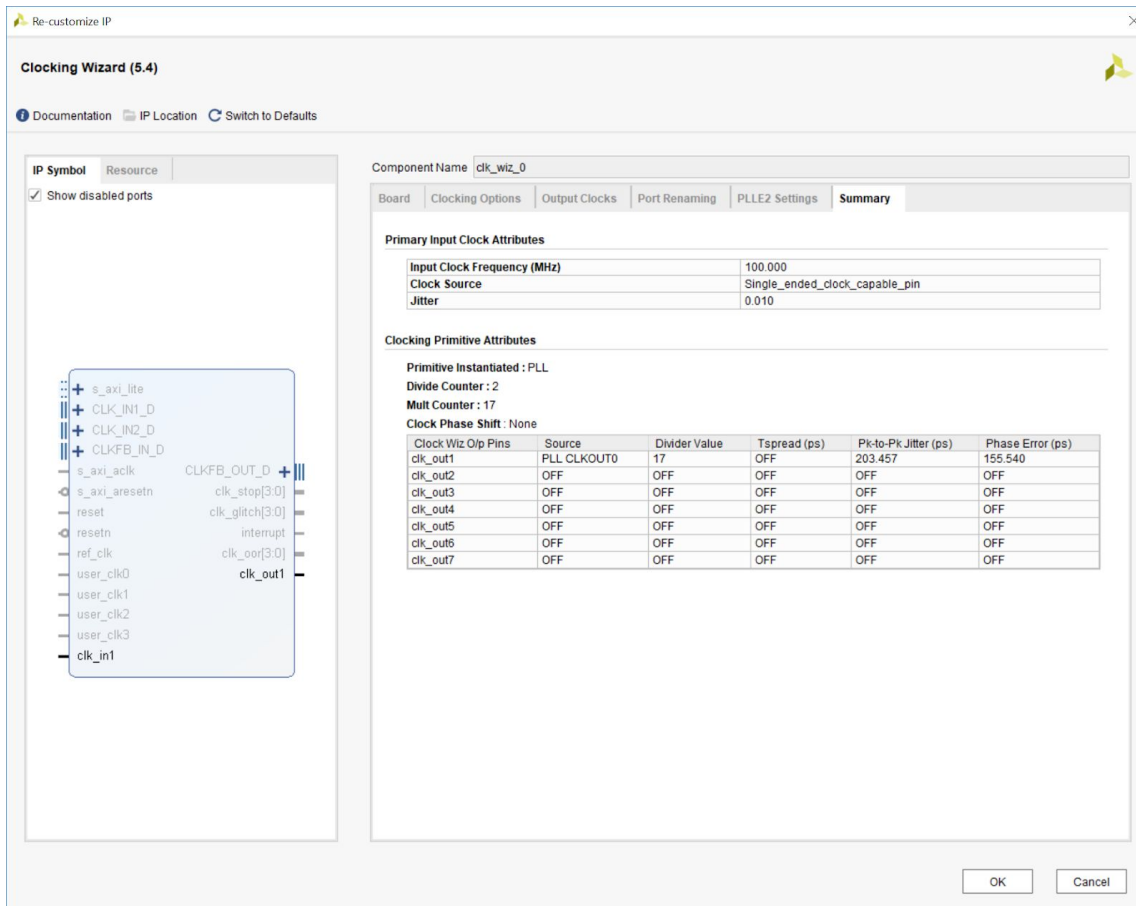
Reset Type

☒ Active High ☐ Active Low

OK Cancel

Select `clk_out1` as Output Clock and as Port Name. Set 50.000 as Output Freq (MHz) Requested and set Phase (degrees) Requested to 0.000 and Duty Cycle (%) Requested to 50.000.

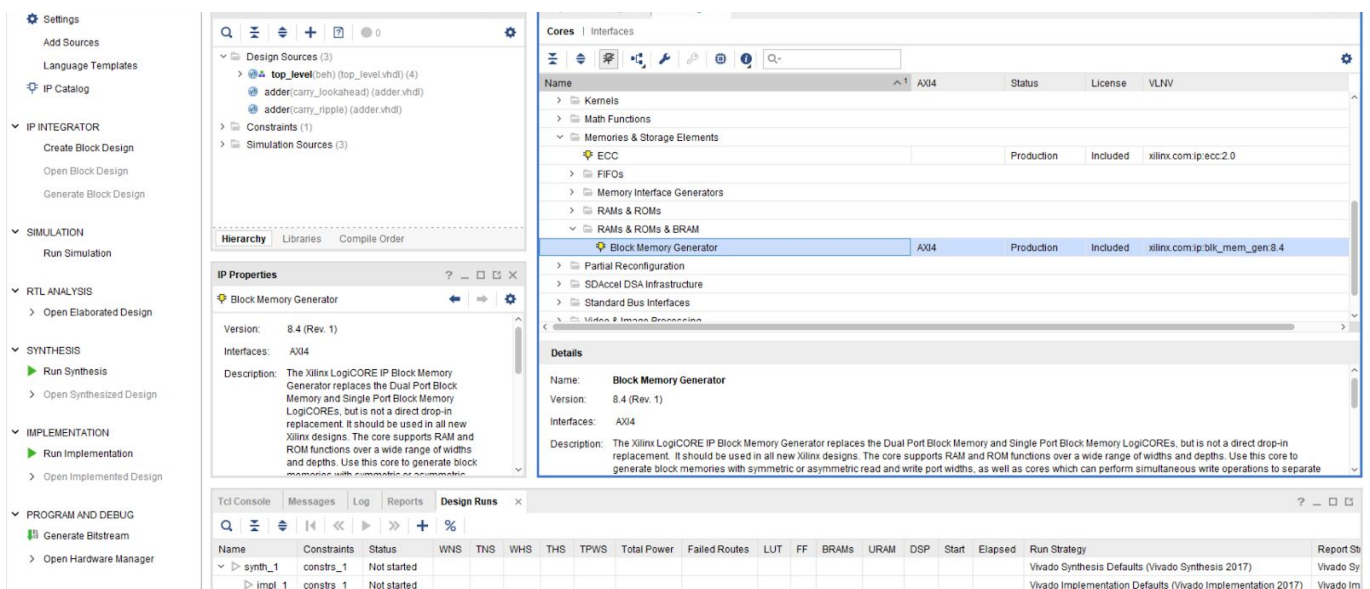
Click on the index tab **Summary** to continue.



If Everything is set correctly, click on **Ok** to start the generation of the 50 MHz PLL. This will take some time!

Generate a Block RAM with the Assembler program

The CPU needs BRAM with the Assembler program, to run it. This will be generated by Vivado as well. Therefore we use the Ip catalog again. Open the *Block Memory Generator* from the **IP Catalog**.



Re-customize IP

Block Memory Generator (8.4)

Documentation IP Location Switch to Defaults

IP Symbol Power Estimation

☒ Show disabled ports

Component Name: blk_mem_gen_0

Basic Port A Options Port B Options Other Options Summary

Interface Type: Native ☒ Generate address interface with 32 bits

Memory Type: True Dual Port RAM ☐ Common Clock

ECC Options

ECC Type: No ECC

☐ Error Injection Pins: Single Bit Error Injection

Write Enable

☒ Byte Write Enable

Byte Size (bits): 8

Algorithm Options

Defines the algorithm used to concatenate the block RAM primitives. Refer datasheet for more information.

Algorithm: Minimum Area

Primitive: 8kx2

OK Cancel

Set *Native* as Interface Type and *True Dual Port Ram* as Memory Type. Select *Generate address interface with 32 bits!* Ecc is not needed. Set everything else as shown above! Click on the index tab **Port A Options** to continue.

Re-customize IP

Block Memory Generator (8.4)

Documentation IP Location Switch to Defaults

IP Symbol Power Estimation

☒ Show disabled ports

Component Name: blk_mem_gen_0

Basic **Port A Options** Port B Options Other Options Summary

Memory Size

Write Width: 32 Range: 32 to 1024 (bits)

Read Width: 32

Write Depth: 143360 Range: 2 to 1048576

Read Depth: 143360

Operating Mode: Write First Enable Port Type: Use ENA Pin

Port A Optional Output Registers

☐ Primitives Output Register ☐ Core Output Register

☐ SoftECC Input Register ☐ REGCEA Pin

Port A Output Reset Options

☐ RSTA Pin (set/reset pin) Output Reset Value (Hex): 0

☐ Reset Memory Latch Reset Priority: CE (Latch or Register Enable)

READ Address Change A

☐ Read Address Change A

OK Cancel

Set write and read Width to 32 and the depth to 143360! Set Operating Mode to *Write First* and Enable Port Type to *Use ENA Pin*. Disable everything else and continue on tab **Port B Options**.

Re-customize IP

Block Memory Generator (8.4)

Documentation IP Location Switch to Defaults

IP Symbol Power Estimation

☒ Show disabled ports

Component Name: blk_mem_gen_0

Basic Port A Options Port B Options Other Options Summary

Memory Size

Write Width: 32
Read Width: 32
Write Depth: 143360
Read Depth: 143360

Operating Mode: Write First Enable Port Type: Use ENB Pin

Port B Optional Output Registers

☐ Primitives Output Register ☐ Core Output Register
☐ SoftECC Output Register ☐ REGCEB Pin

Port B Output Reset Options

☐ RSTB Pin (set/reset pin) Output Reset Value (Hex): 0
☐ Reset Memory Latch Reset Priority: CE (Latch or Register Enable)

READ Address Change B

☐ Read Address Change B

OK Cancel

Set everything as Port A (as shown above) and continue on tab **Other Options**.

Re-customize IP

Block Memory Generator (8.4)

Documentation IP Location Switch to Defaults

IP Symbol Power Estimation

☒ Show disabled ports

Component Name: blk_mem_gen_0

Basic Port A Options Port B Options Other Options Summary

Pipeline Stages within Mux: 0 Mux Size: 35x1

Memory Initialization

☒ Load Init File
Coe File: 017-2018/src/cpu/vivado/test_programs/gpio_full_test.coe Browse Edit
☒ Fill Remaining Memory Locations
Remaining Memory Locations (Hex): 0

Structural/UniSim Simulation Model Options

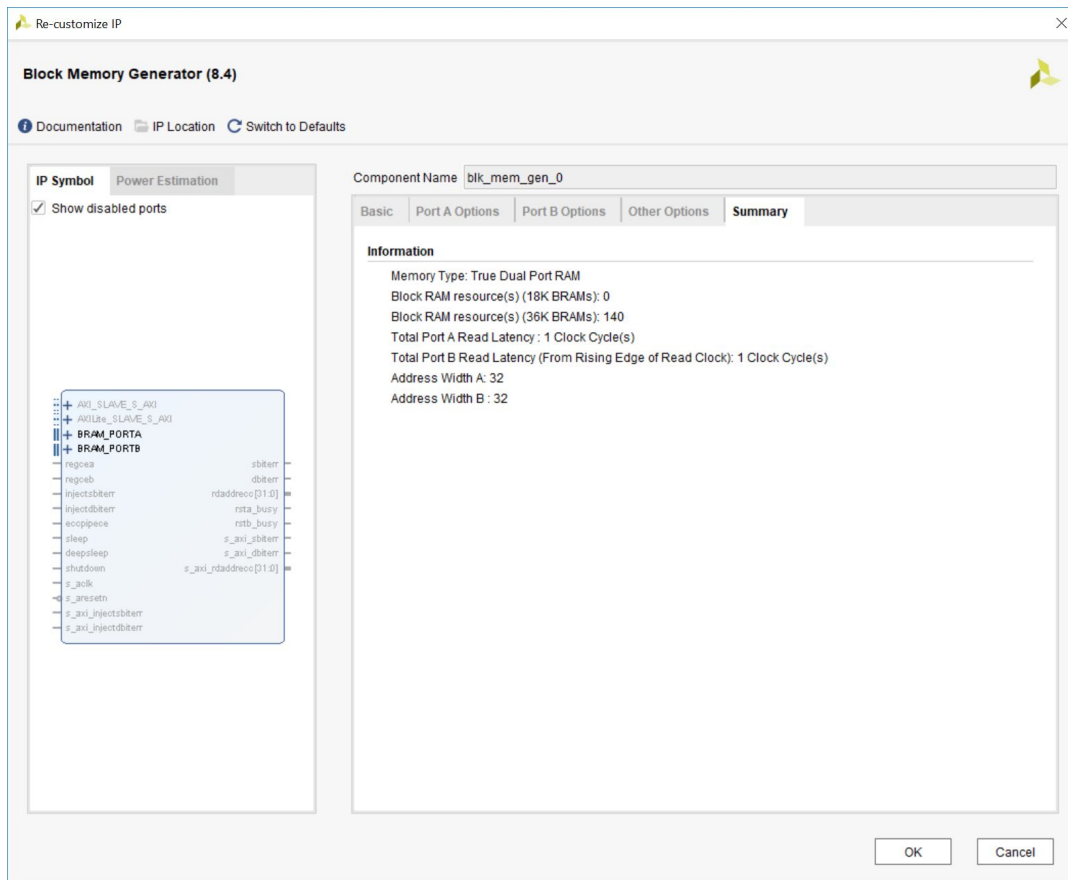
Defines the type of warnings and outputs are generated when a read-write or write-write collision occurs.
Collision Warnings: All

Behavioral Simulation Model Options

☐ Disable Collision Warnings ☐ Disable Out of Range Warnings

OK Cancel

Enable the Option *Load init file* and select from the project folder the file *gpio_full_test.coe* as the Coe file. The Coe file contains the assembler program in the RISC V type. Fill the Remaining Memory with 0. Click on the the tab **Summary** to check the settings.

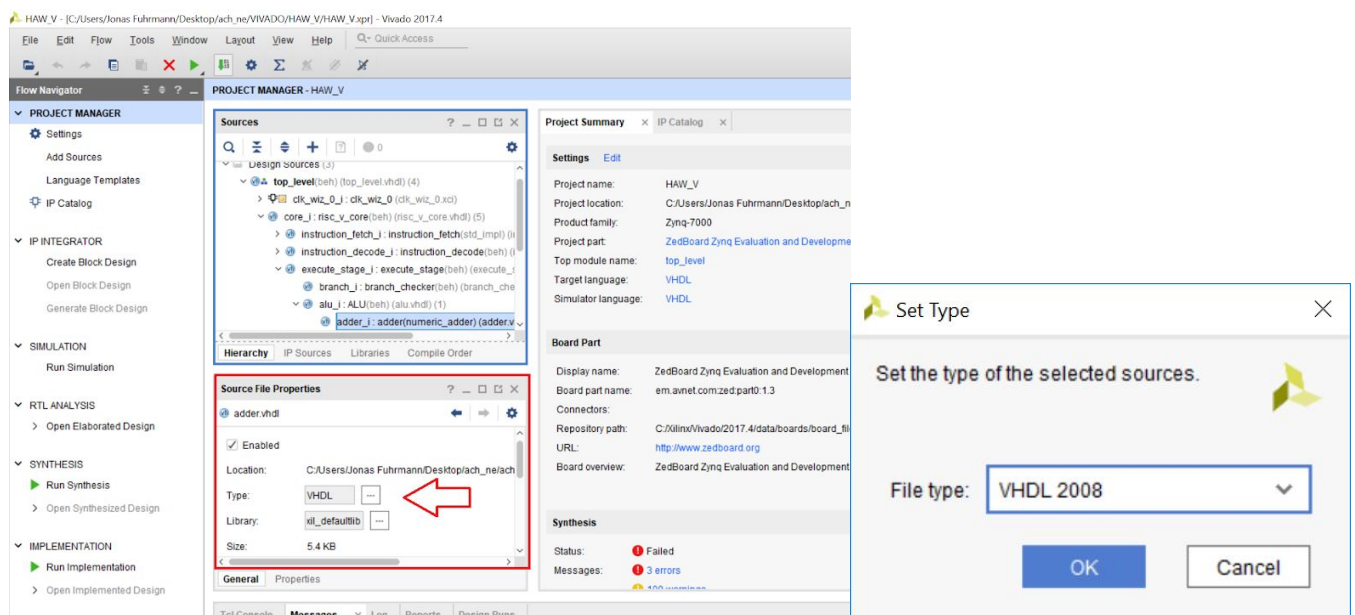


Start the generation of the BRAM by clicking on **Ok**. This will take some Time!

Run the Synthesis

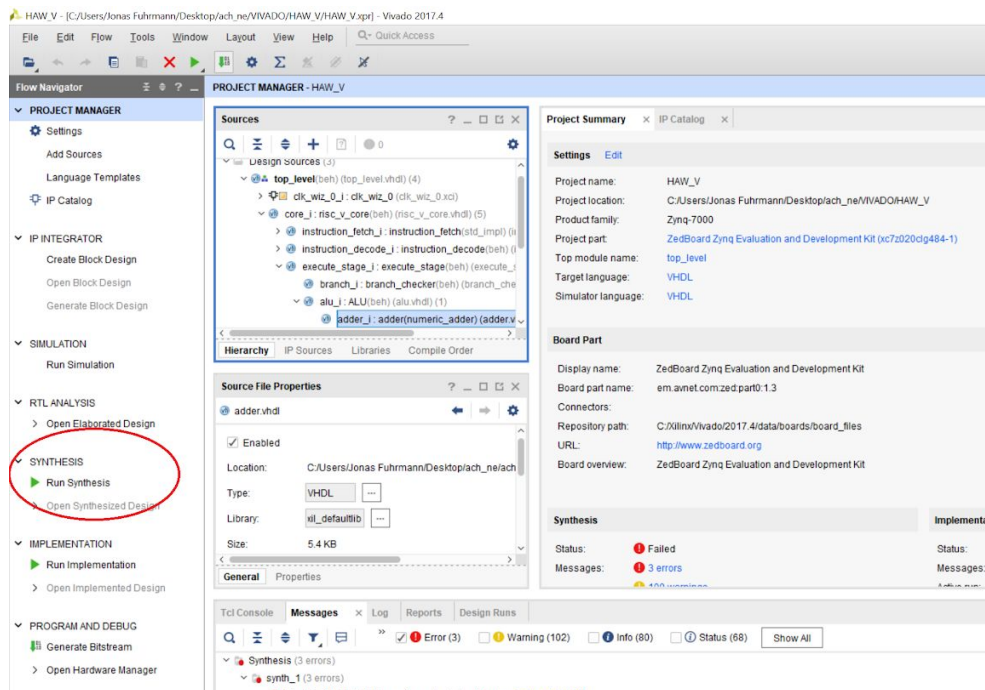
It is important, that every component is generated before the Synthesis is started (Clock and BRAM)! To run the Synthesis all vhd files should be **VHDL 2008**.

This can be set here:



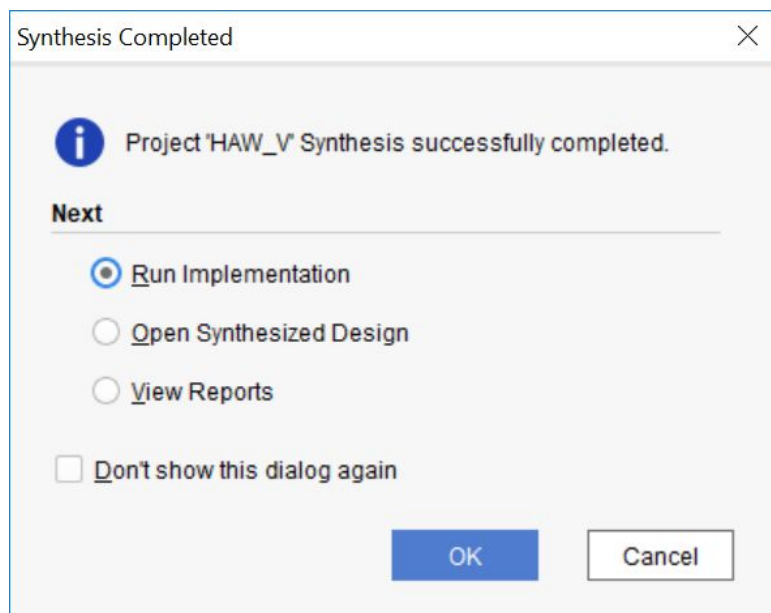
Click on the ... and select **VHDL 2008**. Click on **OK** to continue. This must be done for **all vhd Files**!

If this is done click on *Run Synthesis*.

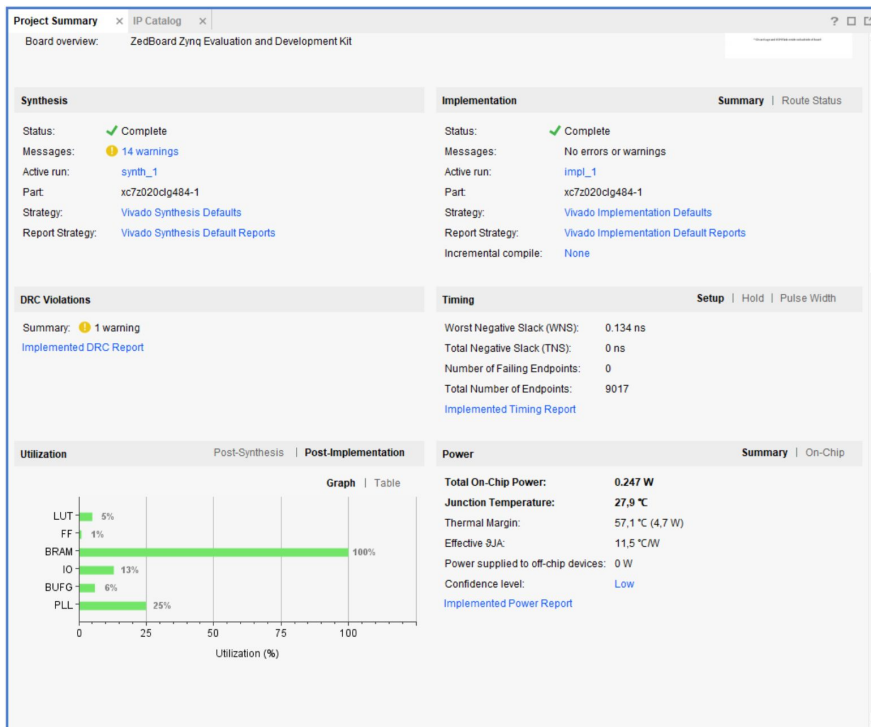


Run the Implementation

After the Synthesis is finished successfully , Vivado opens a window as shown below. Click on *Start Implementation*.



After the Implementation there is an Report (Project Summary) which shows different kind of Information about used hardware, power, critical path etc..



Generate a Bitstream

After the implementation we need a bitstream. Click on **Generate Bitstream**.

HAW_V - [C:/Users/Jonas Fuhrmann/Desktop/ach_ne/VIVADO/HAW_V/HAW_V.xpr] - Vivado 2017.4

File Edit Flow Tools Window Layout View Help Quick Access

Flow Navigator PROJECT MANAGER - HAW_V

PROJECT MANAGER

- Settings
- Add Sources
- Language Templates
- IP Catalog
- IP INTEGRATOR
 - Create Block Design
 - Open Block Design
 - Generate Block Design
- SIMULATION
 - Run Simulation
- RTL ANALYSIS
 - Open Elaborated Design
- SYNTHESIS
 - Run Synthesis
 - Open Synthesized Design
- IMPLEMENTATION
 - Run Implementation
 - Open Implemented Design
- PROGRAM AND DEBUG**
 - Generate Bitstream** (highlighted with a red box and arrow)
 - Open Hardware Manager

Sources

Design sources (3)

- top_level (beh) (top_level.vhdl) (4)
 - clk_wiz_0_1: clk_wiz_0 (clk_wiz_0.xci)
 - core_1: risc_v_core (beh) (risc_v_core.vhdl) (5)
 - instruction_fetch_1: instruction_fetch (std_impl) (1)
 - instruction_decode_1: instruction_decode (beh) (1)
 - execute_stage_1: execute_stage (beh) (execute_1)
 - branch_1: branch_checker (beh) (branch_che)
 - alu_1: ALU (beh) (alu.vhdl) (1)
 - adder_1: adder (numeric_adder) (adder.v)

Source File Properties

adder.vhdl

Enabled

Location: C:/Users/Jonas Fuhrmann/Desktop/ach_ne/ach

Type: VHDL

Library: xil_defaultlib

Size: 5.4 KB

Project Summary x IP Catalog x

Settings Edit

Project name: HAW_V
 Project location: C:/Users/Jonas Fuhrmann/Desktop/ach_ne/VIVADO/HAW_V
 Product family: Zynq-7000
 Project part: ZedBoard Zynq Evaluation and Development Kit (xc7z020dpg484-1)
 Top module name: top_level
 Target language: VHDL
 Simulator language: VHDL

Board Part

Display name: ZedBoard Zynq Evaluation and Development Kit
 Board part name: em.avnet.com:zed:part0:1.3
 Connectors:
 Repository path: C:/Xilinx/Vivado/2017.4/data/boards/board_files
 URL: <http://www.zedboard.org>
 Board overview: ZedBoard Zynq Evaluation and Development Kit

Synthesis

Status: ● Failed
 Messages: ● 3 errors

Tcl Console Messages x Log Reports Design Runs

Messages

3 Errors (3) 102 Warnings (102) 80 Info (80) 68 Status (68) Show All

Synthesis (3 errors)

synth_1 (3 errors)

[Synth 8-944] 0 definitions of operator "*" match here [adder.vhdl:149]

Program a Bitstream on the ZedBoard

After the generation of the bitstream is successfully completed, Vivado opens a window. Select *Open Hardware Manager* and click on **OK**.

Before plugging in the Zedboard, there are Jumper settings that need to be checked.

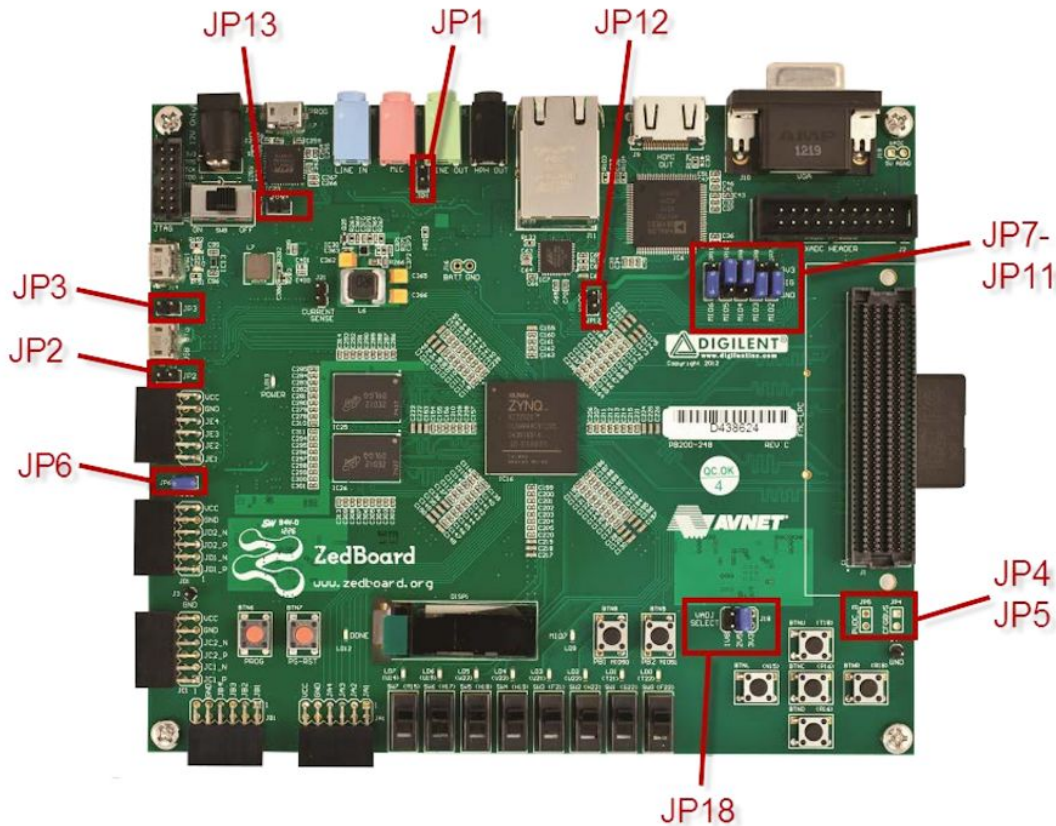


Figure 20 - ZedBoard Jumper Map

Set Jumper **JP7- JP11** all to 0 to set the correct Mode!

After this connect the Zedboard with power and with the USB cable to the Pc with Vivado to upload the program (Use the USB-PORT right next to the Power connection). Turn on the ZedBoard.

The Hardware Manager from Vivado should show the ZedBoard. If not, update the Hardware Manager. Click on it and program the Bitstream.

The program *gpio_full_test.coe* uses the switches and the buttons to trigger the LEDs.

Create own coe files for HAW RISC V CPU

To create own coe files out of assembler programs, the RISC V toolchain is needed. After that there is a script which can be found in *vivado/helper_scripts*. It is important to know, that the compiler doesn't link addresses from the *.data* section. The *.data* section can be put behind the *.text* section. Then the offset has to be calculated and insert into the *.coe* file.