Hochschule Darmstadt University of Applied Sciences	Microelectronic Systems Lab1 WS18/19	Prof. Dr.-Ing. T. Schumann

Lab1: Hardware/Software Co-Design on Xilinx Zynq SoC

3 points

In this lab you will get an intro to the state-of-the-art SoC design flow using Xilinx Vivado Design Suite 2018.2. Design target is the ZedBoard, containing Xilinx Zynq SoC.
This lab includes creating a Vivado project, using the IP integrator, writing software using Xilinx SDK, generating a bitstream and downloading it on a ZedBoard for verifying your design. This lab should give you an exposure to FPGA-based SoC design and implementation.

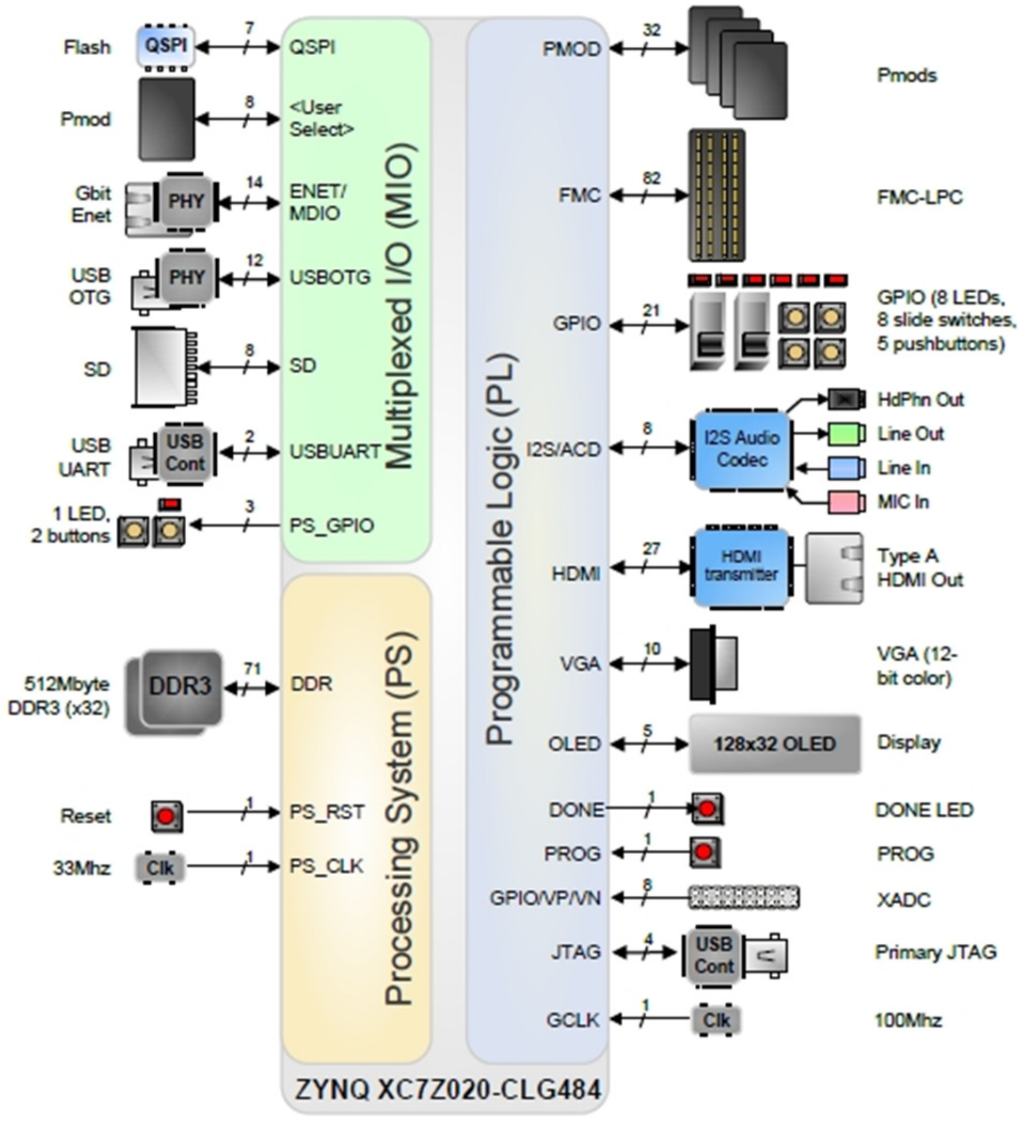
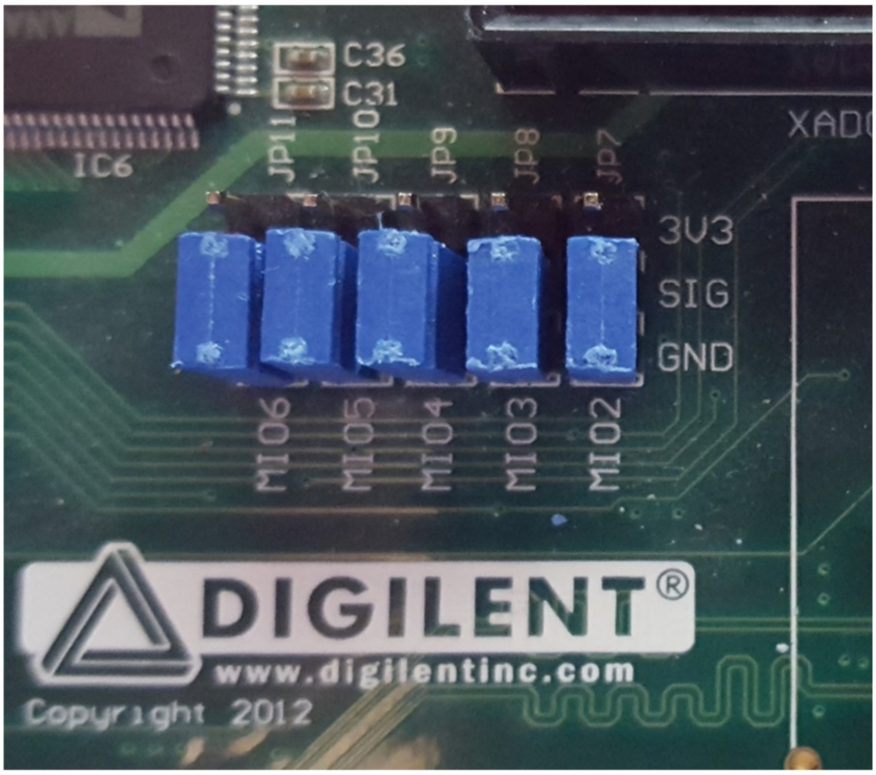


Figure 1 – ZedBoard Block Diagram

1. Hardware Setup:

- ZedBoard with its power supply.
- 2x USB micro cable for JTAG and UART.
- Jumper settings for using the JTAG interface

First, make sure that the jumpers **JP7-JP11** are in the JTAG position (shown below) and that the ZedBoard JTAG-Interface and UART are plugged into your computer via micro-USB cables. Connect also power supply to ZedBoard, but turn **power** switch **off**!



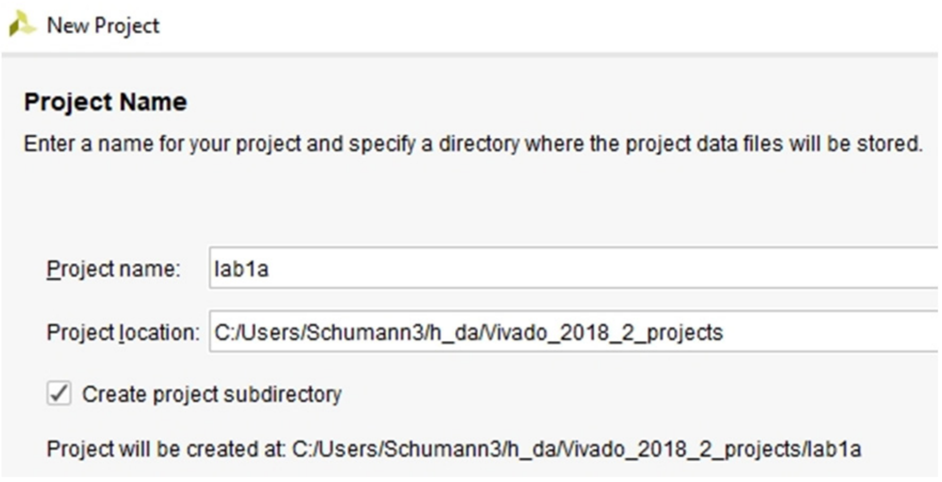
2. Software Setup:

- Xilinx Vivado 2018.2.
- Xilinx SDK 2018.2.

3. New Project Setup:

We shall create the hardware part of the project in the lab exercise. The aim is to understand the logic design and implementation on Programmable Logic (FPGA) of Zynq device using Xilinx Vivado tool flow.

1. Open Vivado, click on file --> project --> new
2. Click next, here give the project name (lab1a) and select the location for the project files to be stored (e.g. lab_microelectronics) and click next.
   Note: file path should not contain any white spaces.


New Project

Project Name

Enter a name for your project and specify a directory where the project data files will be stored.

Project name: lab1a

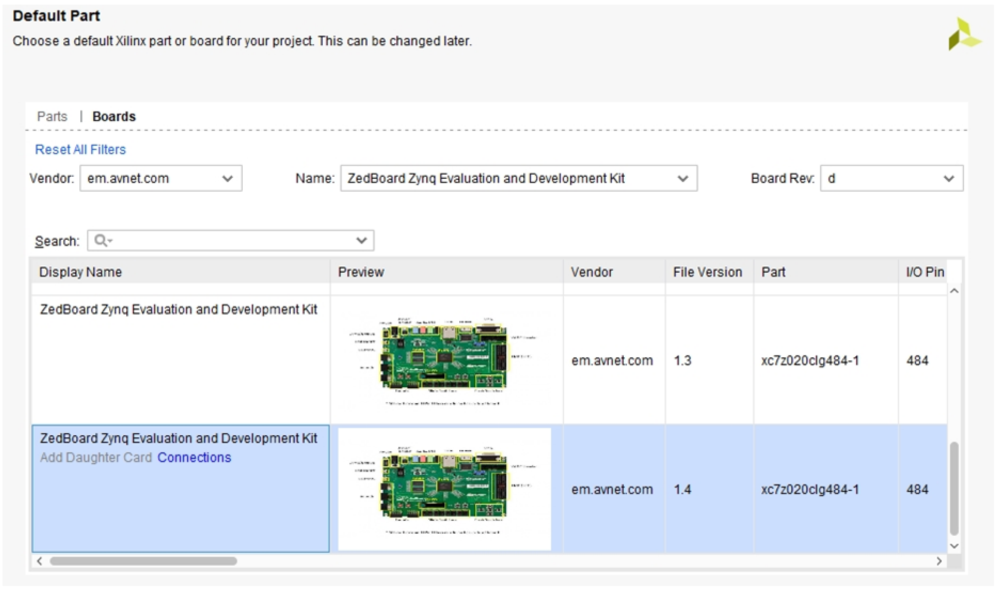
Project location: C:/Users/Schumann3/h_da/Vivado_2018_2_projects

✓ Create project subdirectory

Project will be created at: C:/Users/Schumann3/h_da/Vivado_2018_2_projects/lab1a

3. Select RTL Project and click next.

4. Click next as we do not add any files at this point of time. Click also next on "Add contraints".
5. Click on Boards and select "ZedBoard". Check your actually used ZedBoard regarding revision (c or d file version 1.4). Click next and click finish.


Default Part

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

Parts | Boards

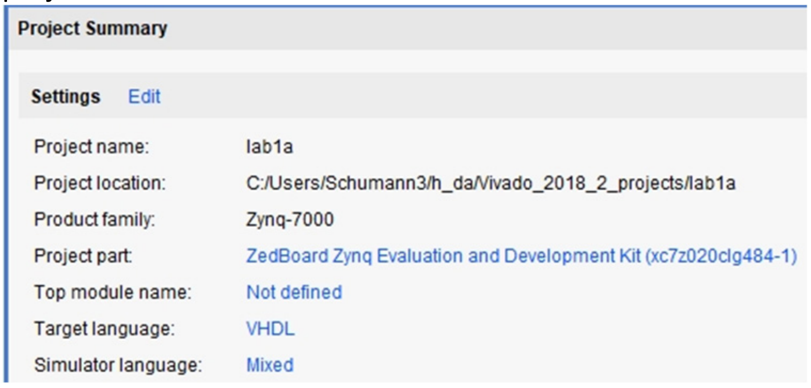
Reset All Filters

Vendor: em.avnet.com Name: ZedBoard Zynq Evaluation and Development Kit Board Rev: d

Search:

Display Name	Preview	Vendor	File Version	Part	I/O Pin
ZedBoard Zynq Evaluation and Development Kit		em.avnet.com	1.3	xc7z020clg484-1	484
ZedBoard Zynq Evaluation and Development Kit Add Daughter Card Connections		em.avnet.com	1.4	xc7z020clg484-1	484

6. Now the Vivado project summary window will be open giving the description of the project created.

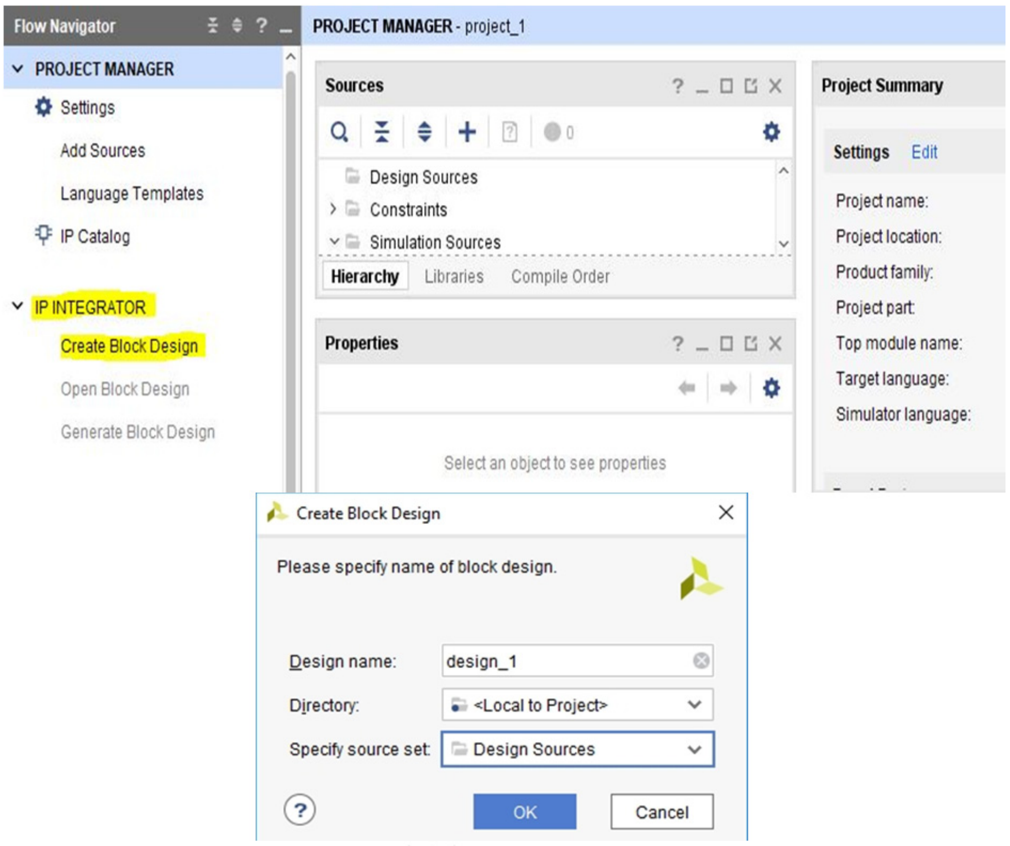

Project Summary

Settings Edit

Project name:	lab1a
Project location:	C:/Users/Schumann3/h_da/Vivado_2018_2_projects/lab1a
Product family:	Zynq-7000
Project part:	ZedBoard Zynq Evaluation and Development Kit (xc7z020clg484-1)
Top module name:	Not defined
Target language:	VHDL
Simulator language:	Mixed

4. Task 1: creating the hardware block design:

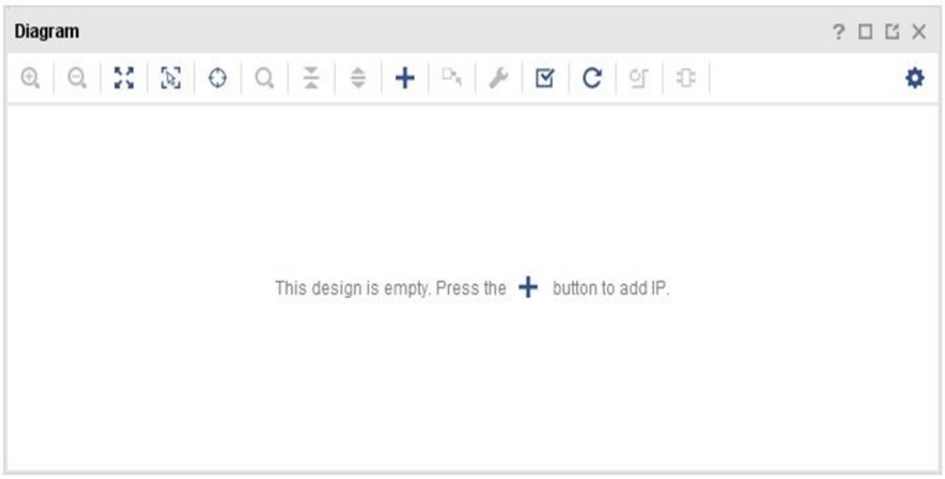
1. In Flow Navigator click on "Create Block Design" for IP Integrator and specify name of block design. Take default settings and click OK.



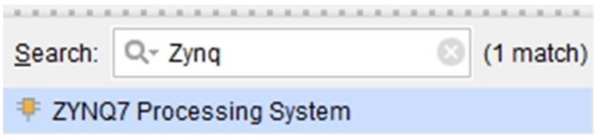
Please specify name of block design.

Design name: design_1
Directory: <Local to Project>
Specify source set: Design Sources

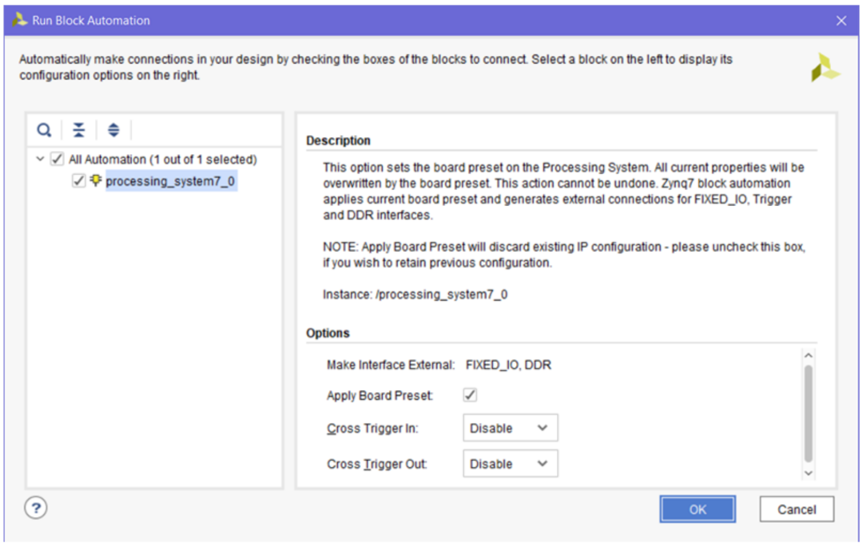
2. In the diagram window click on the + to add IPs. Alternatively, you can select "Add IP" by right click in the diagram window.



3. Add the IP "ZYNQ7 Processing System". Click Enter.



4. Click on the “Run Block Automation” appearing on top of the window. Click OK on the next window.


Run Block Automation

Automatically make connections in your design by checking the boxes of the blocks to connect. Select a block on the left to display its configuration options on the right.

All Automation (1 out of 1 selected)
processing_system7_0

Description

This option sets the board preset on the Processing System. All current properties will be overwritten by the board preset. This action cannot be undone. Zynq7 block automation applies current board preset and generates external connections for FIXED_IO, Trigger and DDR interfaces.

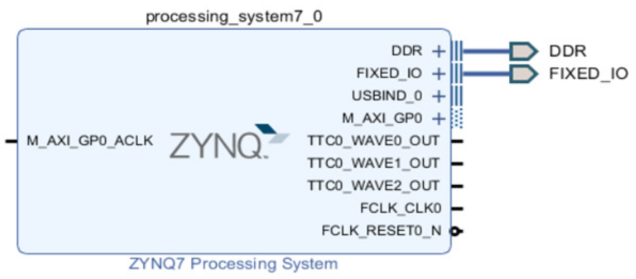
NOTE: Apply Board Preset will discard existing IP configuration - please uncheck this box, if you wish to retain previous configuration.

Instance: /processing_system7_0

Options

Make Interface External: FIXED_IO, DDR
Apply Board Preset: ✓
Cross Trigger In: Disable
Cross Trigger Out: Disable

5. Your Zynq block should now look like the picture below.



6. Now add the IP “AXI GPIO” by clicking on the + button.

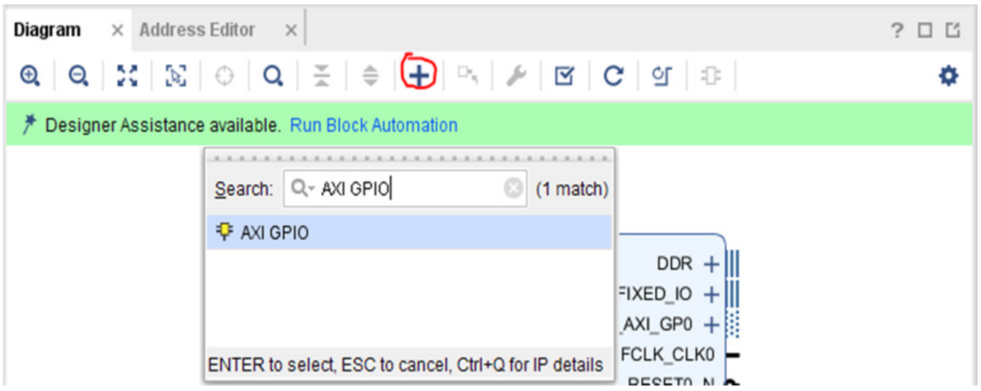
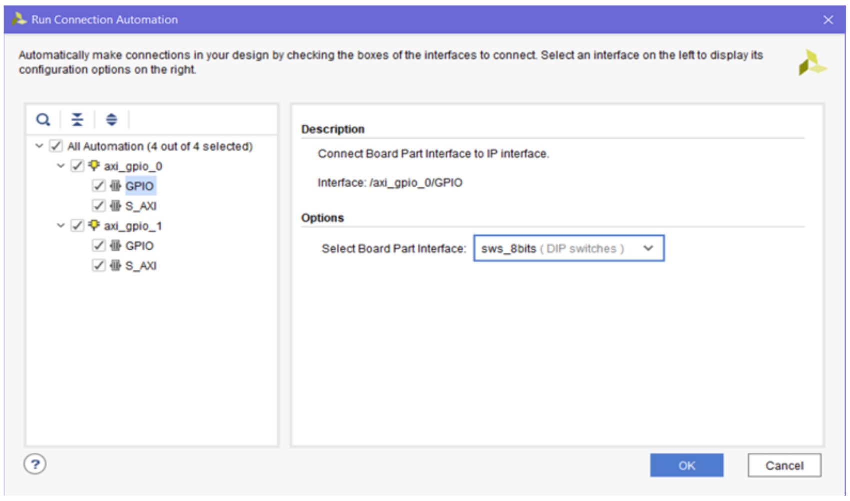


Diagram
Address Editor
Designer Assistance available. Run Block Automation
Search: AXI GPIO (1 match)
AXI GPIO
ENTER to select, ESC to cancel, Ctrl+Q for IP details
DDR
FIXED_IO
AXI_GP0
FCLK_CLK0

7. Now you see axi_gpio_0 core in Diagram. (if you need to customize an IP, you double-click on IP core)
8. Add another IP “AXI GPIO”.
9. Click on “Run connection automation” option on the top of the window.

Check “All Automation” on the left-hand side.
Click on GPIO of axi_gpio_0 and select “sws_8bits”. Similarly, for the other GPIO select “LED_8bits” and click OK.



Automatically make connections in your design by checking the boxes of the interfaces to connect. Select an interface on the left to display its configuration options on the right.

- All Automation (4 out of 4 selected)
  - axi_gpio_0
    - GPIO
    - S_AXI
  - axi_gpio_1
    - GPIO
    - S_AXI

Description

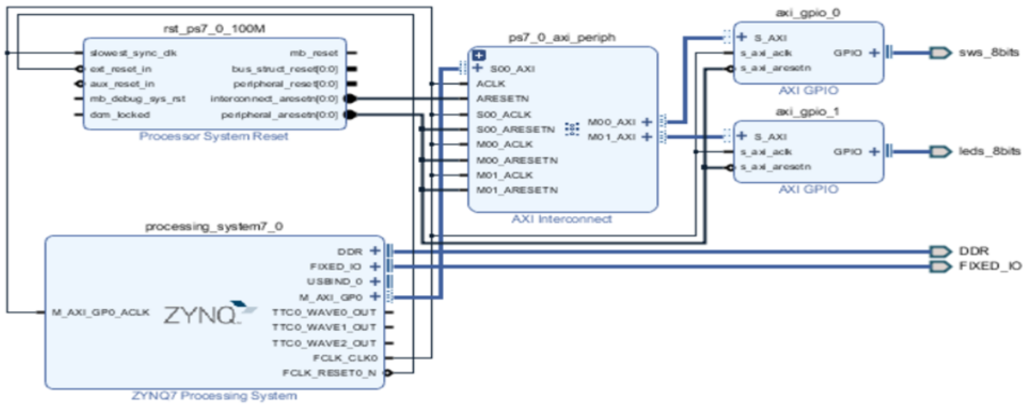
Connect Board Part Interface to IP interface.

Interface: /axi_gpio_0/GPIO

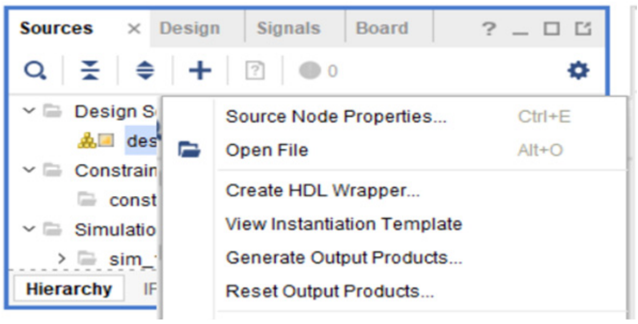
Options

Select Board Part Interface: sws_8bits (DIP switches)

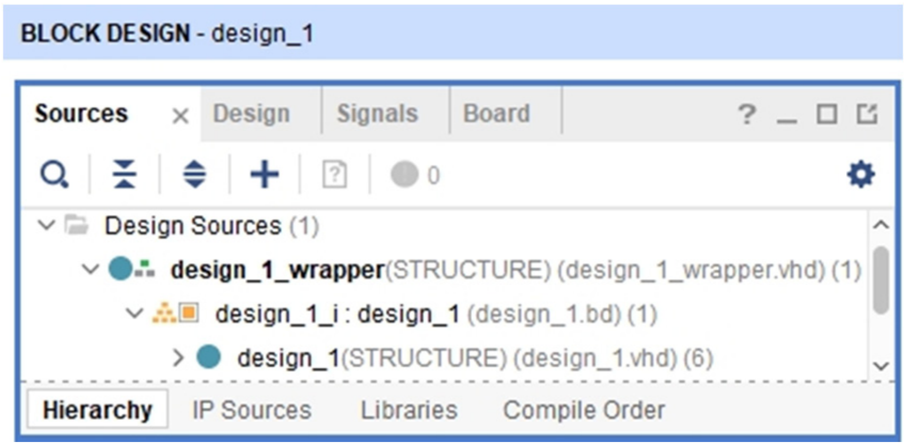
10. Click on Regenerate Layout to redraw the block diagram.
11. Now your block diagram should look as given below.



12. Select Validate Design . This will check for design and connection errors.
13. After the design validation step we will proceed with creating 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.

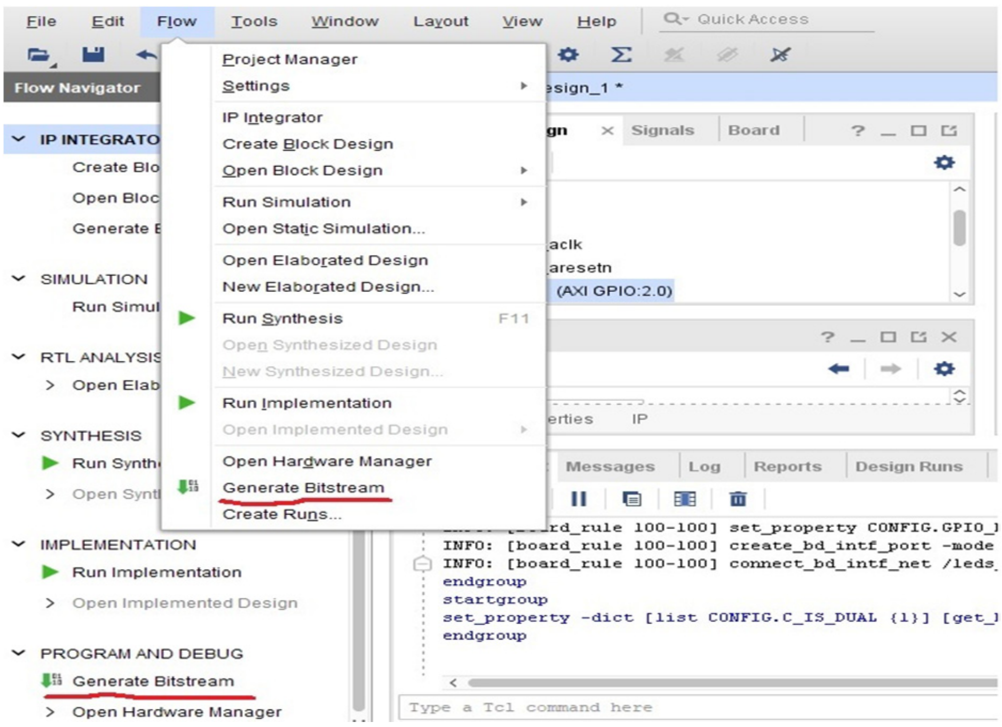


Design Sources now changes with design_1_wrapper as top entity.



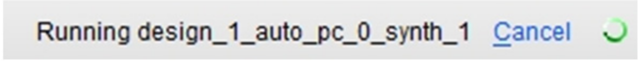
BLOCK DESIGN - design_1
Sources
Design
Signals
Board
Design Sources (1)
design_1_wrapper(STRUCTURE) (design_1_wrapper.vhd) (1)
design_1_i : design_1 (design_1.bd) (1)
design_1(STRUCTURE) (design_1.vhd) (6)
Hierarchy
IP Sources
Libraries
Compile Order

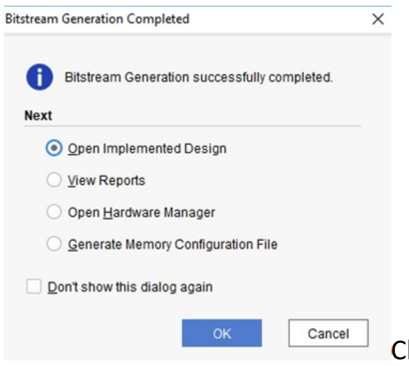
14. Click on "Generate Bitstream" (at the bottom of the Flow Navigator or Flow -> Generate Bitstream). Click Yes to launch synthesis and implementation. Click OK with default settings on next window.



```
....: [...rd_rule 100-100] set_property CONFIG.GPIO_I
INFO: [board_rule 100-100] create_bd_intf_port -mode
INFO: [board_rule 100-100] connect_bd_intf_net /leds_
endgroup
startgroup
set_property -dict [list CONFIG.C_IS_DUAL {1}] [get_]
endgroup
```

Wait for the process to complete.
As long as the process is running you see top right of Vivado window:




Bitstream Generation Completed

Bitstream Generation successfully completed.

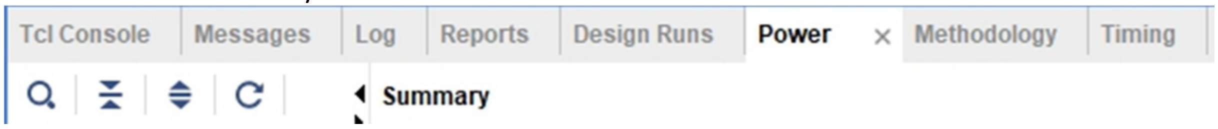
Next

- [✓ Open Implemented Design]
- View Reports
- Open Hardware Manager
- Generate Memory Configuration File

Don't show this dialog again

Click OK.

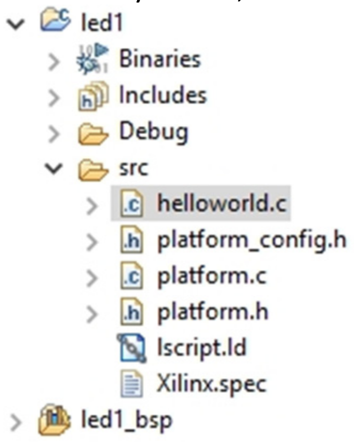
15. Check on Power Summary:



Question1: Which block of design consumes most power? How much? % of total dynamic power dissipation?

5. Task 2: Exporting hardware to SDK:

1. In Vivado: File→Export→Export Hardware. Make sure to check the box for Include bitstream then click OK.
2. In Vivado: File→Launch SDK and click OK. SDK is starting and importing your hardware specification.
3. In SDK: File→New→Application Project.
4. Provide the Project Name (e.g. led1), take default settings and click Next. Select "Hello World" and click on Finish. This process will add two directories to the Project Explorer.
5. In directory "led1", double-click on "helloworld .c" . This opens a window with C-code.



6. Replace this code with the code below and click on File -> Save. This should automatically build the software.

```
#include <stdio.h>
#include "platform.h"
#include <xgpio.h>
#include "xparameters.h"
#include "sleep.h"
```

```
int main()
{
   XGpio input, output;
//   int button_data = 0;
   int switch_data = 0;

   XGpio_Initialize(&input, XPAR_AXI_GPIO_0_DEVICE_ID);    //initialize input
XGpio variable
   XGpio_Initialize(&output, XPAR_AXI_GPIO_1_DEVICE_ID);   //initialize output
XGpio variable

   XGpio_SetDataDirection(&input, 1, 0xF);                 //set first channel
tristate buffer to input
//   XGpio_SetDataDirection(&input, 2, 0xF);               //set second channel
tristate buffer to input

   XGpio_SetDataDirection(&output, 1, 0x0);         //set first channel tristate
buffer to output

   init_platform();

   while(1){

      switch_data = XGpio_DiscreteRead(&input, 1);  //get switch data

      XGpio_DiscreteWrite(&output, 1, switch_data); //write switch data to the
LEDs
     usleep(200000);                   //delay

   }
   cleanup_platform();
   return 0;
}
```

7. After Build is complete, make sure that the ZedBoard is connected to the host PC via the JTAG und UART USB Ports and that JP7-JP11 is set to JTAG mode. Switch on power supply of ZedBoard.
8. To program the FPGA, on the top toolbar, click the Program FPGA button . And then click Program. The blue LED on ZedBoard turn on!
9. Right click within C-code window and select Run As → Launch on Hardware (System Debugger). This writes the code to ARM-processor of Zynq.
10. Now you should see LED turning ON and OFF depending on the switch position.
11. Exit SDK.

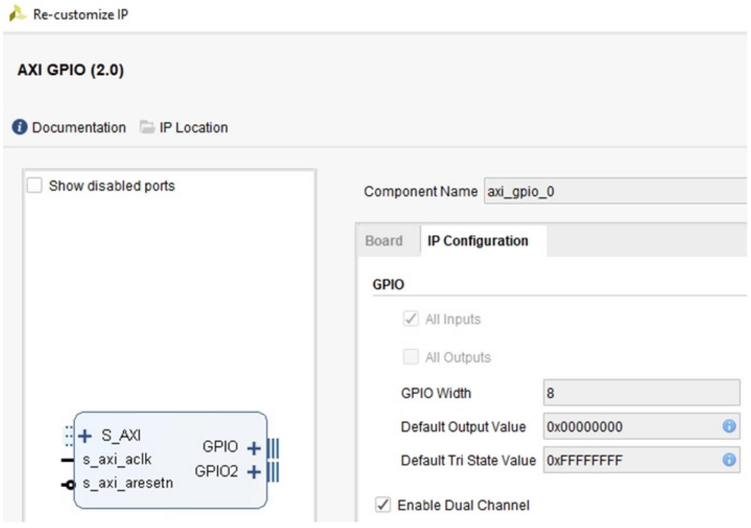
Question2: What is the meaning of directory led1_bsp?

6. Task 3: UART communication:

Now we want to use UART communication. Aim is to identify if one of the five buttons of ZedBoard is pressed. So we need first to add another GPIO for the buttons in Vivado, then we need to change C-Code in SDK.

1. Save your current Vivado project as lab1b. File -> Project -> Save As, Project Name: lab1b
2. Within IP Integrator click on Open Block Design

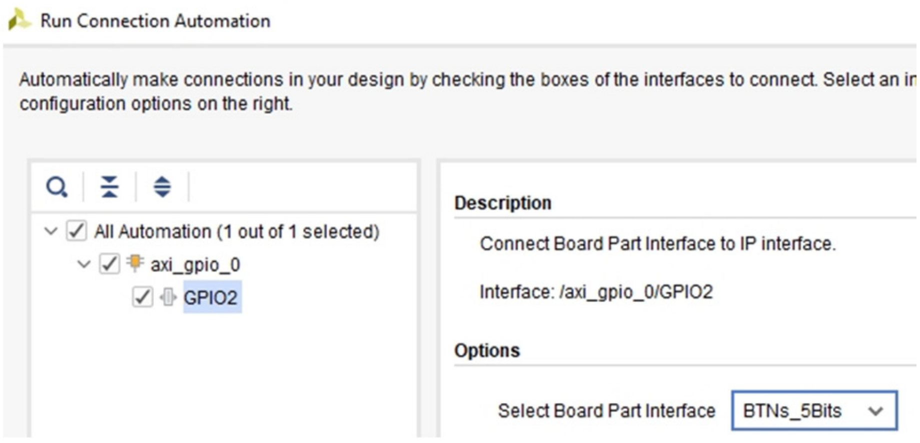
3. Double-click on axi_gpio_0, select “Enable Dual Channel” under IP Configuration. Click OK. Now we can use GPIO2 for our buttons.



Component Name axi_gpio_0

✓ All Inputs
All Outputs
GPIO Width 8
Default Output Value 0x00000000
Default Tri State Value 0xFFFFFFFF
✓ Enable Dual Channel

4. Click on “Run Connection Automation”



Automatically make connections in your design by checking the boxes of the interfaces to connect. Select an in configuration options on the right.

✓ All Automation (1 out of 1 selected)
✓ axi_gpio_0
✓ GPIO2

Description

Connect Board Part Interface to IP interface.

Interface: /axi_gpio_0/GPIO2

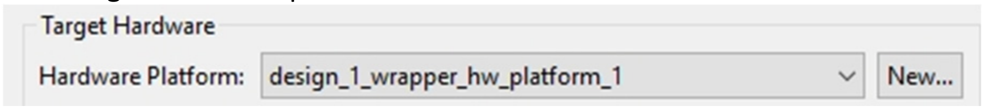
Options

Select Board Part Interface BTNs_5Bits

Click OK.

5. Select Validate Design . File -> Save Block Design. Click on “Generate Bitstream”! Bitstream Generation Completed window pops up: Click Cancel.

6. Export to SDK: File→Export→Export Hardware. Make sure to check the box for Include bitstream then click OK.
7. File→Launch SDK and click OK
   In Project Explorer you get new hardware platform “design_1_wrapper_hw_platform_1”.
8. Delete led_1 and led_1_bsp from Project Explorer.
9. File→New→Application Project
   Project Name: buttons1
   Select right hardware platform



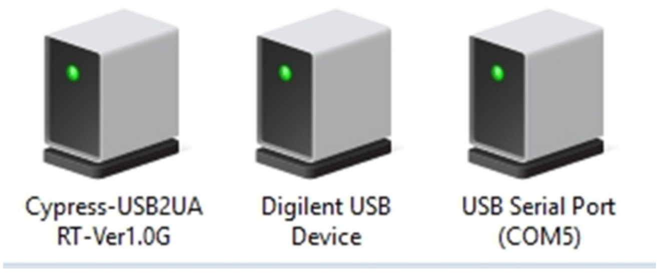
Hardware Platform: design_1_wrapper_hw_platform_1

   Click Finish. If SDK closes unexpected, just open again from Vivado: File -> Launch SDK
10. Replace helloworld.c with below given code for UART communication. Click File -> Save!

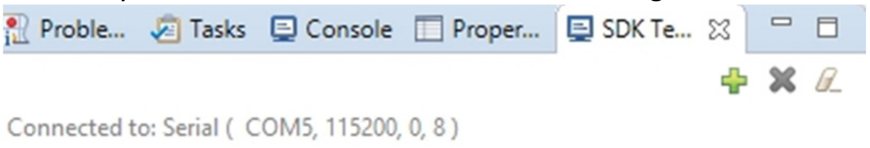
```
#include <stdio.h>
#include "platform.h"
#include <xgpio.h>
#include "xparameters.h"
#include "sleep.h"

int main()
{
   XGpio input, output;
   int button_data = 0;
   int switch_data = 0;

   XGpio_Initialize(&input, XPAR_AXI_GPIO_0_DEVICE_ID);    //initialize input
XGpio variable
   XGpio_Initialize(&output, XPAR_AXI_GPIO_1_DEVICE_ID);   //initialize output
XGpio variable

   XGpio_SetDataDirection(&input, 1, 0xF);                 //set first channel
tristate buffer to input
   XGpio_SetDataDirection(&input, 2, 0xF);                 //set second channel
tristate buffer to input

   XGpio_SetDataDirection(&output, 1, 0x0);         //set first channel tristate
buffer to output

   init_platform();

   while(1){
      button_data = XGpio_DiscreteRead(&input, 1);  //get button data

      //print message dependent on whether one or more buttons are pressed
      if(button_data == 0b00000){} //do nothing

      else if(button_data == 0b00001)
         xil_printf("button 0 pressed\n\r");

      else if(button_data == 0b00010)
         xil_printf("button 1 pressed\n\r");

      else if(button_data == 0b00100)
         xil_printf("button 2 pressed\n\r");

      else if(button_data == 0b01000)
         xil_printf("button 3 pressed\n\r");

      else if(button_data == 0b10000)
              xil_printf("button 4 pressed\n\r");
      else
         xil_printf("multiple buttons pressed\n\r");
usleep(200000);                  //delay

   }
   cleanup_platform();
   return 0;
}
```

11. Click button: Program FPGA Select right Hardware Platform. Click Program.

12. Right click within C-code window and select Run As → Launch on Hardware (System Debugger).
13. Check on Windows Devices which COM port is used for UART (Serial Port)



Connect your SDK Terminal to this Serial Port using the + button.



Connected to: Serial (COM5, 115200, 0, 8)

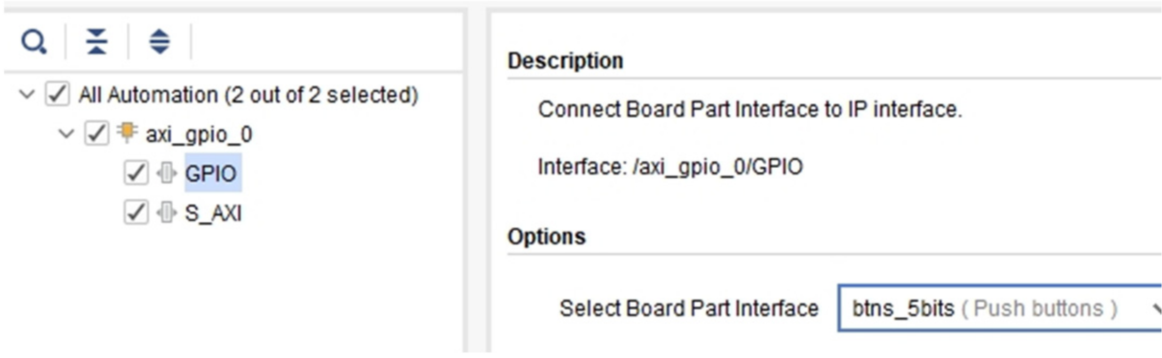
14. Press button on ZedBoard and check on UART terminal output! Take screenshot!

7. Task 4: Hardware/Software Co-Design:

a) Modify the above SDK project: Add a new application project: Led_2
Aim is to run blinking LEDs, by implementing a ring counter or a Johnson counter. Keep the identification of pressed button. Test the same on the ZedBoard! Get a "Done++" from lab tutor!

b) Now we want to change our hardware block design in Vivado: Add a 4-bit adder in VHDL as RTL module! Inputs of the adder are the slide switches and outputs are the LEDs!

i) First exit SDK. Save your Vivado project as lab1c!
ii) Open Block Design, delete the 2 AXI GPIO-blocks together with their ports!
iii) Add one AXI GPIO. Click on "Run Connection Automation": select push-buttons for this GPIO.



Description

Connect Board Part Interface to IP interface.

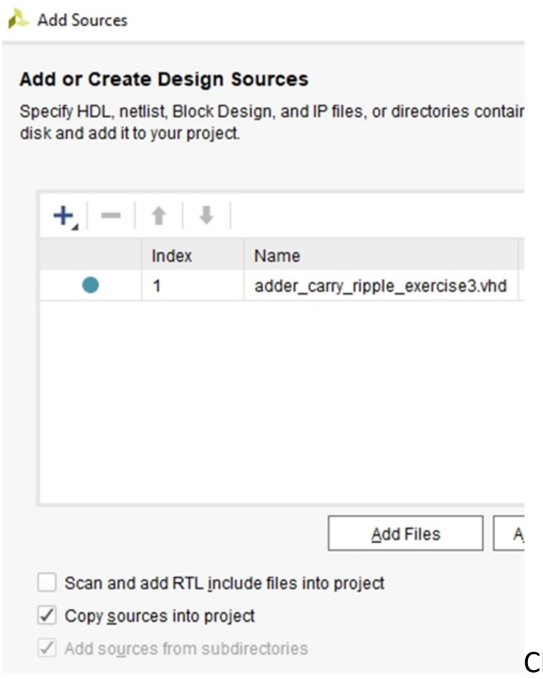
Interface: /axi_gpio_0/GPIO

Options

Select Board Part Interface: btns_5bits (Push buttons)

Click on Regenerate Layout. File -> Save Block Design.

iv) Add the VHDL code (4-bit carry ripple adder of exercise 3) to sources of the Vivado project. File -> Add Sources, tick copy sources into project


Add Sources

Add or Create Design Sources

Specify HDL, netlist, Block Design, and IP files, or directories contair disk and add it to your project.

	Index	Name
●	1	adder_carry_ripple_exercise3.vhd

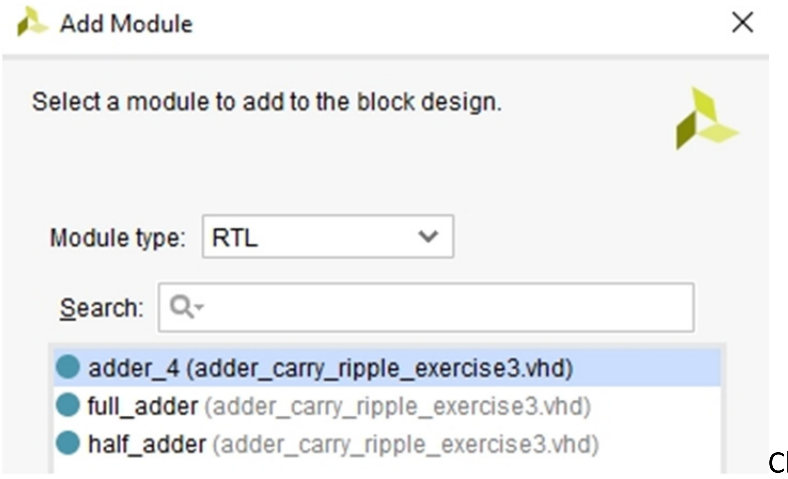
☐ Scan and add RTL include files into project

☑ Copy sources into project

☑ Add sources from subdirectories

Click finish!

v) Within Block Diagram window right click: Add Module


Add Module

Select a module to add to the block design.

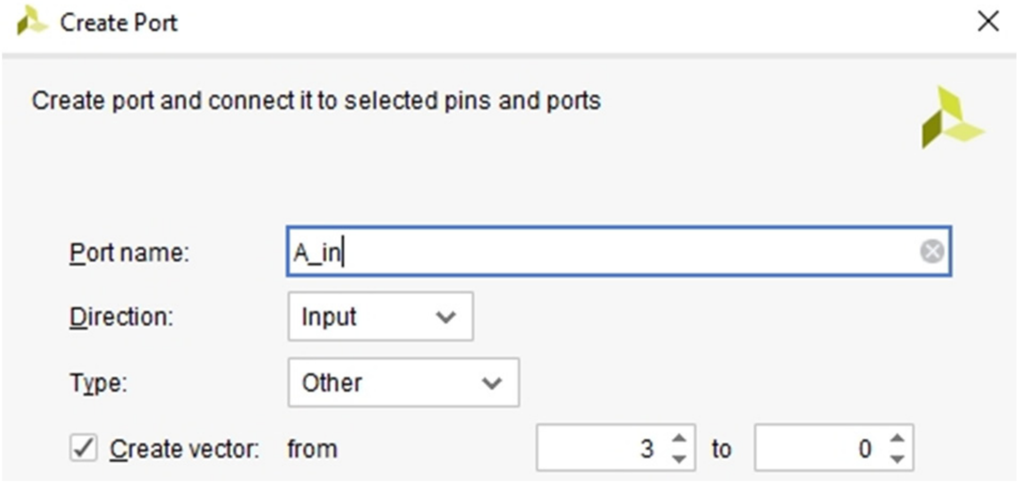
Module type: RTL

Search:

- adder_4 (adder_carry_ripple_exercise3.vhd)
- full_adder (adder_carry_ripple_exercise3.vhd)
- half_adder (adder_carry_ripple_exercise3.vhd)

Click OK.

vi) Within Block Diagram window right click: Create Port


Create Port

Create port and connect it to selected pins and ports

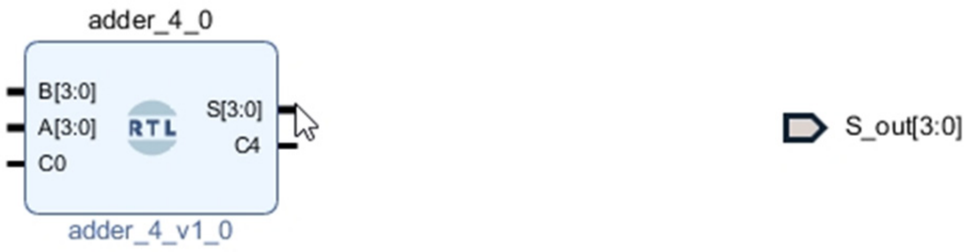
Port name: A_in

Direction: Input

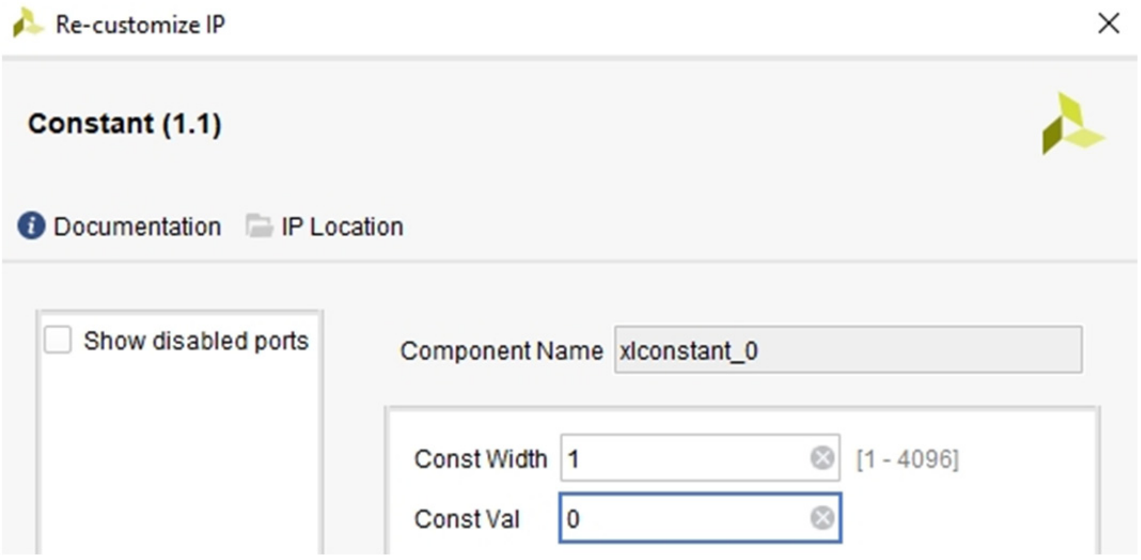
Type: Other

☑ Create vector: from 3 to 0

Create Port B_in, S_out, C4_out (no vector)! Use mouse to wire ports to block adder_4_0!



vii) Add IP Constant. Double-click on block of Constant and set Constant Value to 0.



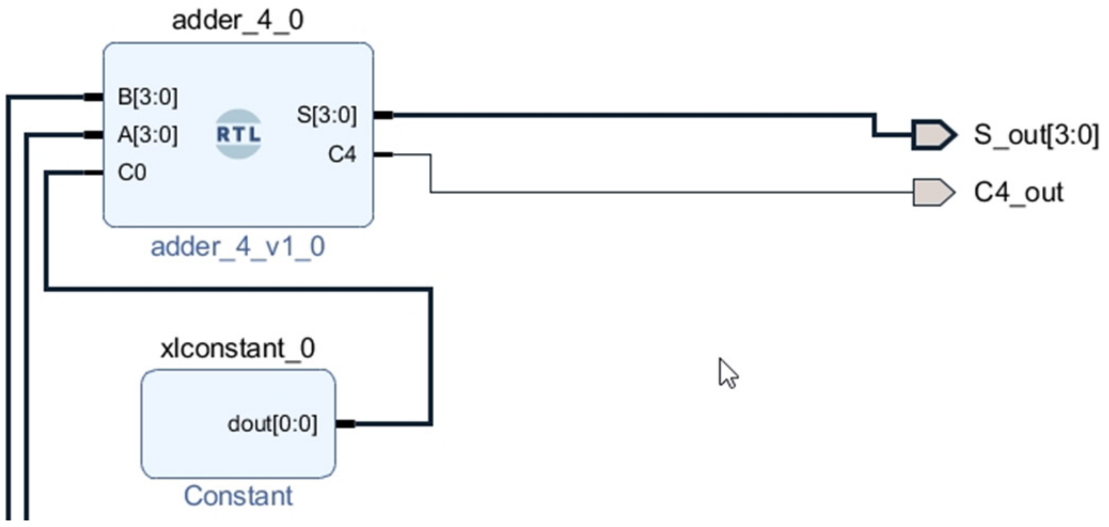
Constant (1.1)

Component Name xlconstant_0

Const Width 1 [1 - 4096]

Const Val 0

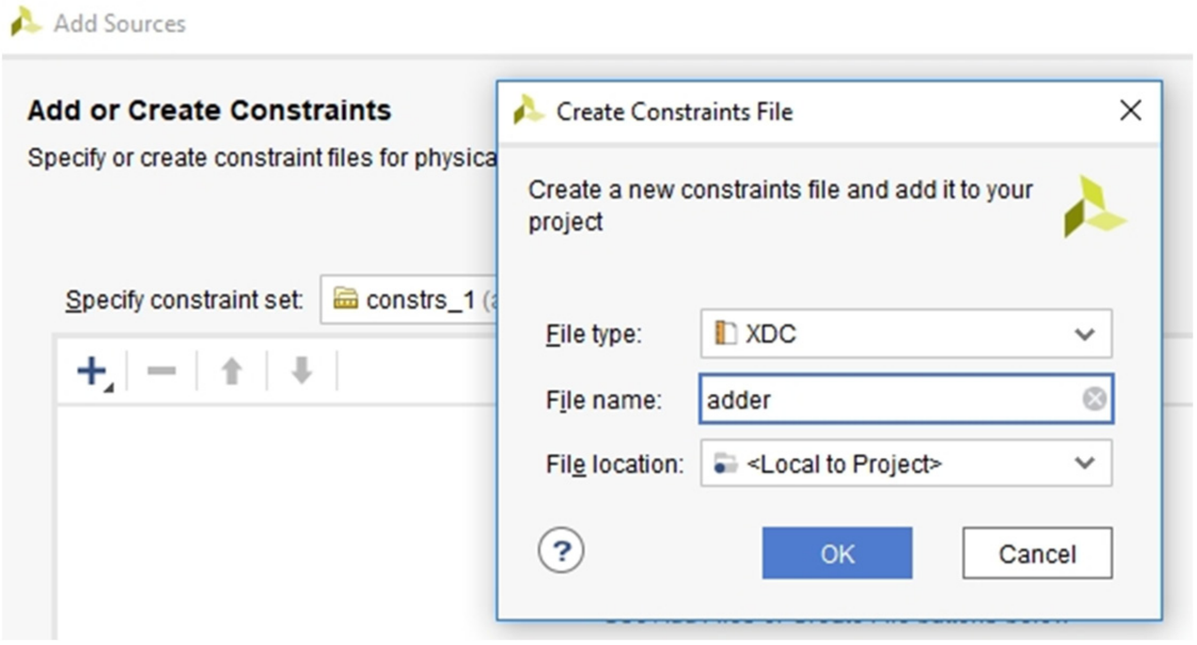
Now wire this block to C0 port of adder_4_0!



Block Design is done!
Regenerate Layout, Validate Design, File -> Save Block Design

viii) Now you need to add design contraints to connect input ports to 8 slide switches and output ports to 5 LEDS!

In Flow Navigator click on Add Sources, tick on add contraints, click on Create File, provide File name, click OK. Click Finish.


Add Sources

Add or Create Constraints

Specify or create constraint files for physica

Specify constraint set: constrs_1

Create Constraints File

Create a new constraints file and add it to your project

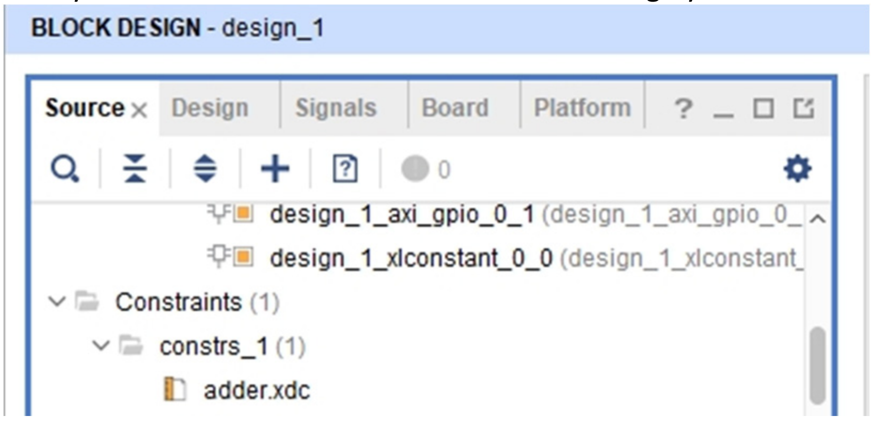
File type: XDC

File name: adder

File location: <Local to Project>

OK | Cancel

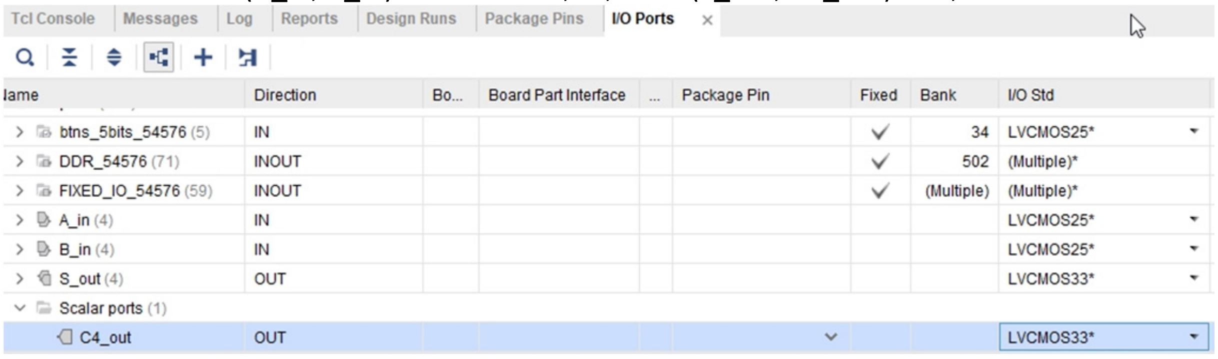
ix) Now you see in the Sources window of Block Design your contraint file


BLOCK DESIGN - design_1

Source | Design | Signals | Board | Platform

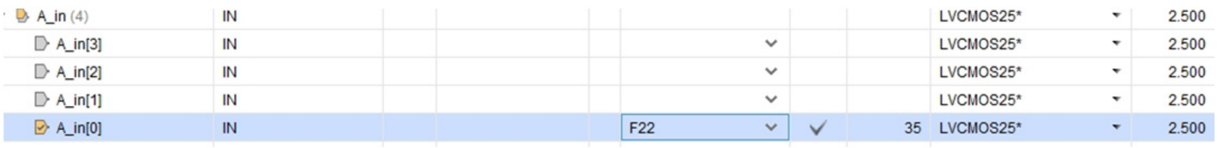
```
design_1_axi_gpio_0_1 (design_1_axi_gpio_0_
design_1_xlconstant_0_0 (design_1_xlconstant_
Constraints (1)
  constrs_1 (1)
    adder.xdc
```

Last step is to define this contraints file.
In Flow Navigator: RTL Analysis, click on Open Eleborated Design
First define the voltage levels I/O Std of our adder ports: slide switches (A_in, B_in) connect to 2,5V, LEDs (S_out, C4_out) to 3,3V!


Tcl Console | Messages | Log | Reports | Design Runs | Package Pins | I/O Ports

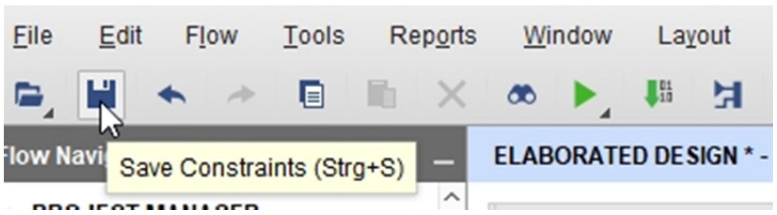
lame	Direction	Bo...	Board Part Interface	...	Package Pin	Fixed	Bank	I/O Std
btns_5bits_54576 (5)	IN					✓	34	LVCMOS25*
DDR_54576 (71)	INOUT					✓	502	(Multiple)*
FIXED_IO_54576 (59)	INOUT					✓	(Multiple)	(Multiple)*
A_in (4)	IN							LVCMOS25*
B_in (4)	IN							LVCMOS25*
S_out (4)	OUT							LVCMOS33*
Scalar ports (1)								
C4_out	OUT							LVCMOS33*

Now we define the Package Pins . For this you need the ZedBoard manual. Check for the Package Pins of DIP switches and LEDs!
Hint: A_in[0] needs to be connected to DIP switch SW0 using Zynq pin F22!

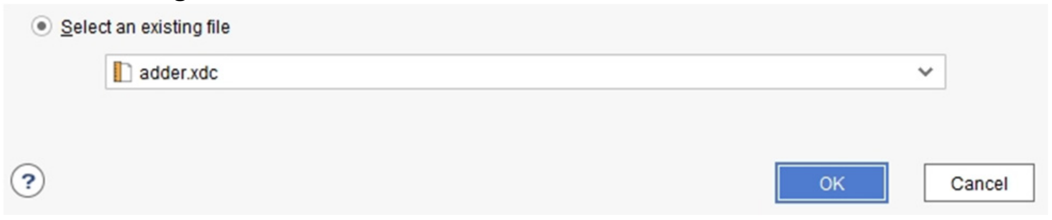


A_in (4)	IN			LVCMOS25*	2.500
A_in[3]	IN			LVCMOS25*	2.500
A_in[2]	IN			LVCMOS25*	2.500
A_in[1]	IN			LVCMOS25*	2.500
A_in[0]	IN	F22 ✓	35	LVCMOS25*	2.500

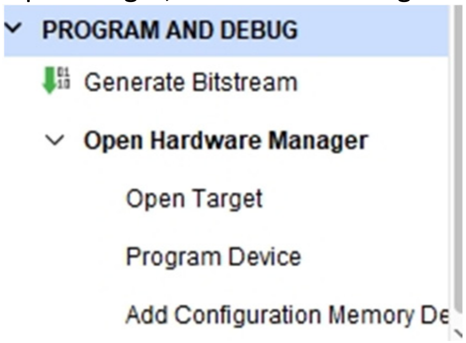
Do this for all switches SW0-SW7 and LEDs LD0-LD4!
Then Save Contraints!



Select existing file



x) Generate Bitstream! Since we want to check on hardware only, click on Open Target, then click on Program Device.



Now set binary addition 12+7 with DIP switches and check on LEDs output!

<u>Lab preparation</u>: Task 1! Lab tutor will check on your preparation!

<u>Lab report:</u> deadline is **Nov.19**! Submit your lab report as hardcopy (group of 2 students)!
Include:

- Answer to questions 1+2.
- Task 3: screenshot
- Task4a: code of ring counter or Johnson counter,
- Task4b: screenshot of I/O Ports definition and photo of LEDs for binary addition 12+7