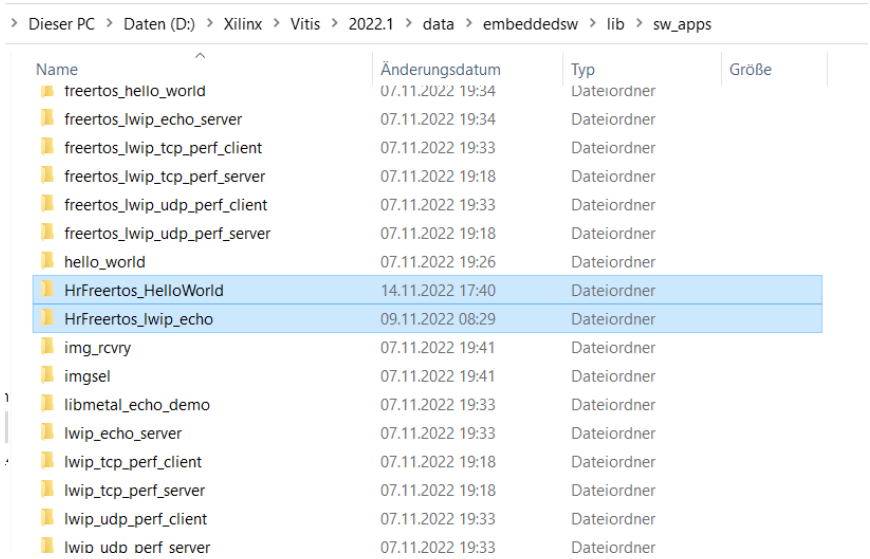
Experiments: Some Preparations ….

1) VITIS application templates

There are template for VITIS projects at:

Windows: **?:\Xilinx\Vitis\2022.1\data\embeddedsw\lib\sw_apps**
Linux: **/scratch/Xilinx/Vitis/2022.1/data/embeddedsw/lib/sw_apps**

copy: ***HrFreertos_HelloWorld***
HrFreertos_lwip_echo
into this directory.



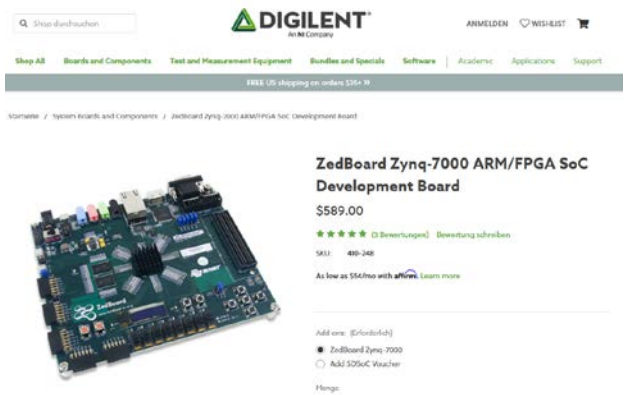
2) IP Block for PMOD_AD1

copy: ***IP_Pmod_AD1***

into your Vivado Project directory

Experiment: List of Material

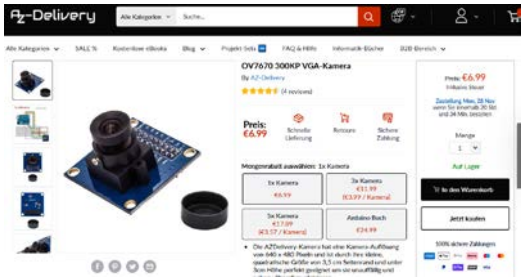
ZedBoard



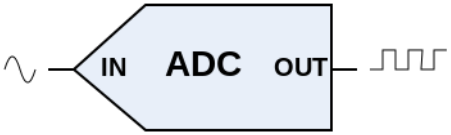
or alternative ZYBO

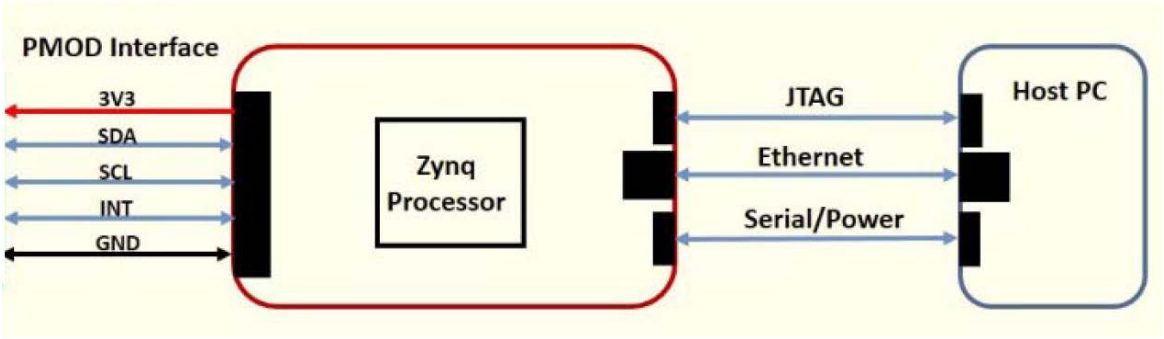
PMOD AD1

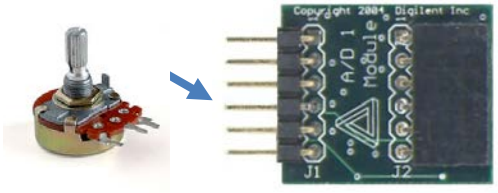
Camera Module



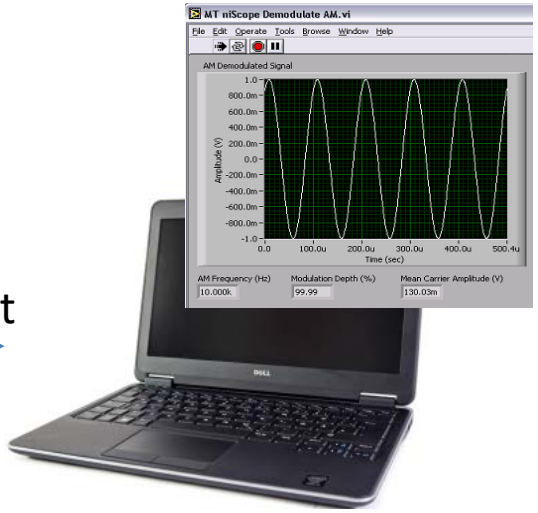
Generic Experiment Setup







Serial
Ethernet
(JTAG)

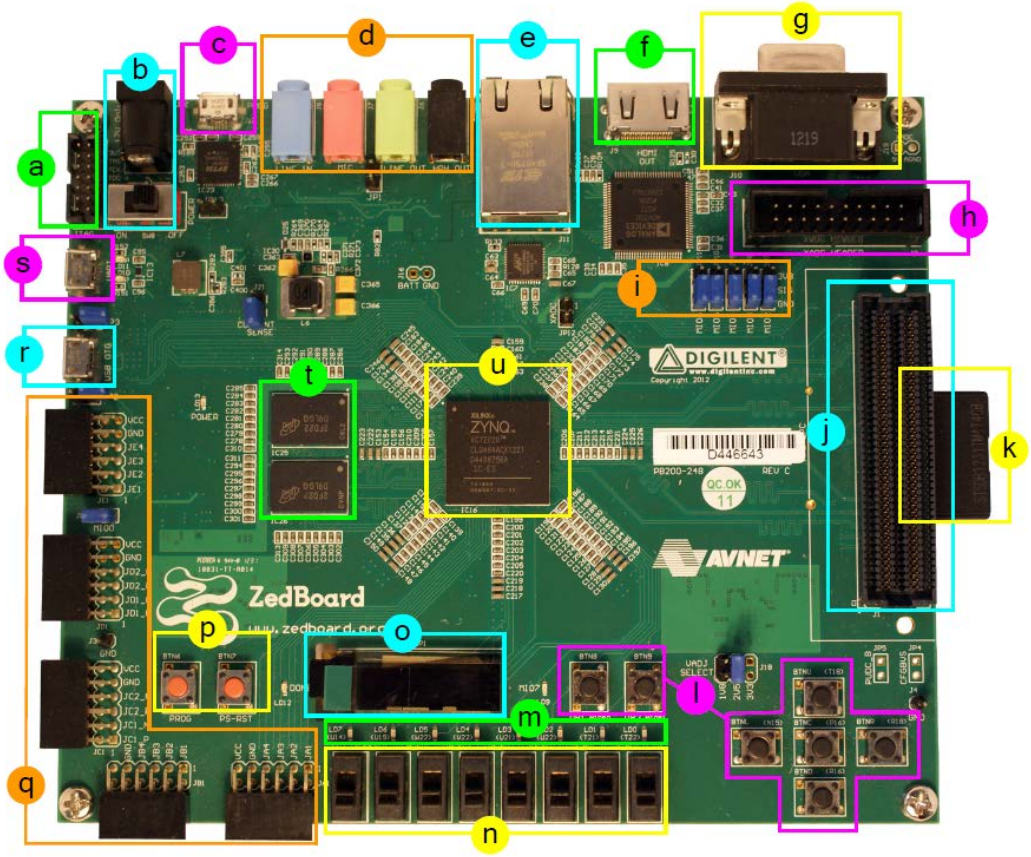


PMOD AD1 Interface
- **Create IP Block**
- **VHDL code**

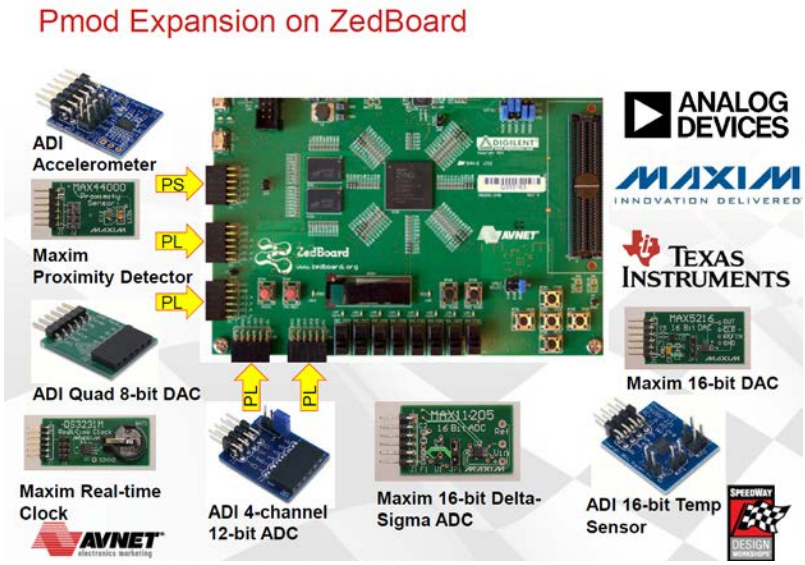
Create SOC Hardware
SDK: freeRTOS + lwIP
- **µC code for DAQ**
- **and Ethernet communication**

Network software
- **Receive Data**
- **Visualization**

ZedBoard & PMOD extensions



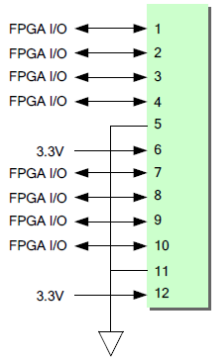
- a Xilinx JTAG connector
- b Power input and switch
- c USB-JTAG (programming)
- d Audio ports
- e Ethernet port
- f HDMI port (output)
- g VGA port
- h XADC header port
- i Configuration jumpers
- j FMC connector
- k SD card (underside)
- l User push buttons
- m LEDs
- n Switches
- o OLED display
- p Prog & reset push buttons
- q 5 x Pmod connector ports
- r USB-OTG peripheral port
- s USB-UART port
- t DDR3 memory
- u Zynq device (+ heatsink)

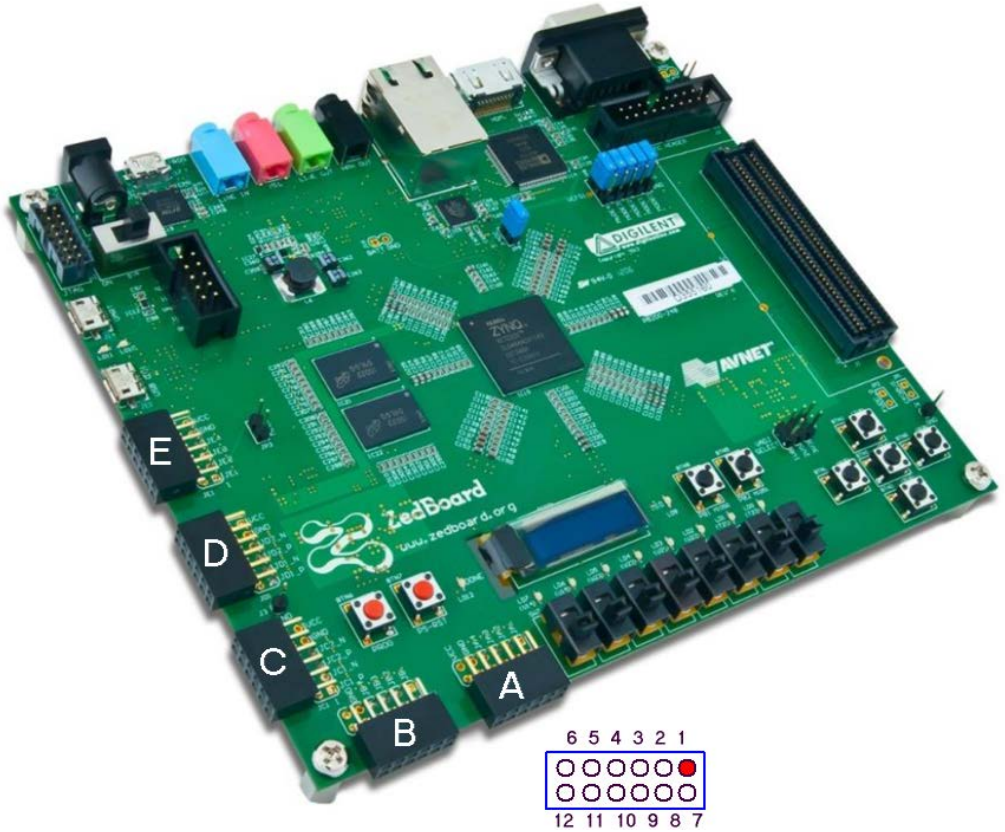


Pmod	Signal Name	Zynq pin	Pmod	Signal Name	Zynq pin
JA1	JA1	Y11	JB1	JB1	W12
	JA2	AA11		JB2	W11
	JA3	Y10		JB3	V10
	JA4	AA9		JB4	W8
	JA7	AB11		JB7	V12
	JA8	AB10		JB8	W10
	JA9	AB9		JB9	V9
	JA10	AA8		JB10	V8

Pmod	Signal Name	Zynq pin	Pmod	Signal Name	Zynq pin
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

Pmod	Signal Name	Zynq pin
JE1 MIO Pmod	JE1	A6
	JE2	G7
	JE3	B4
	JE4	C5
	JE7	G6
	JE8	C4
	JE9	B6
	JE10	E6

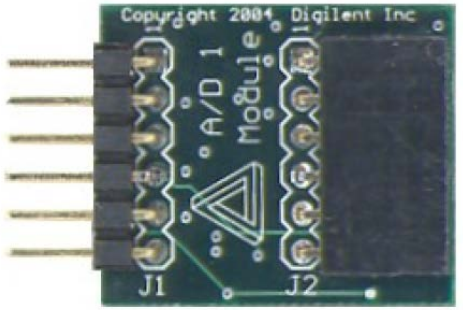


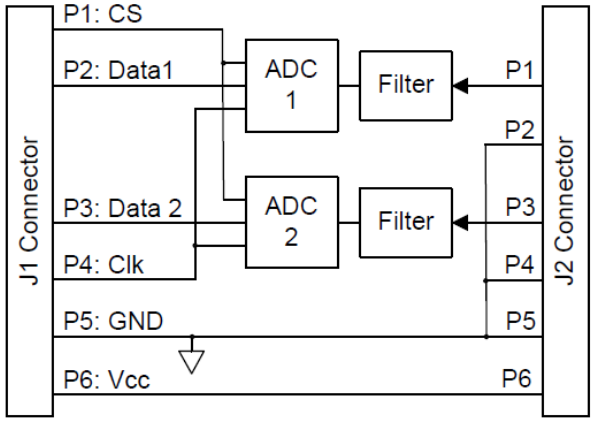


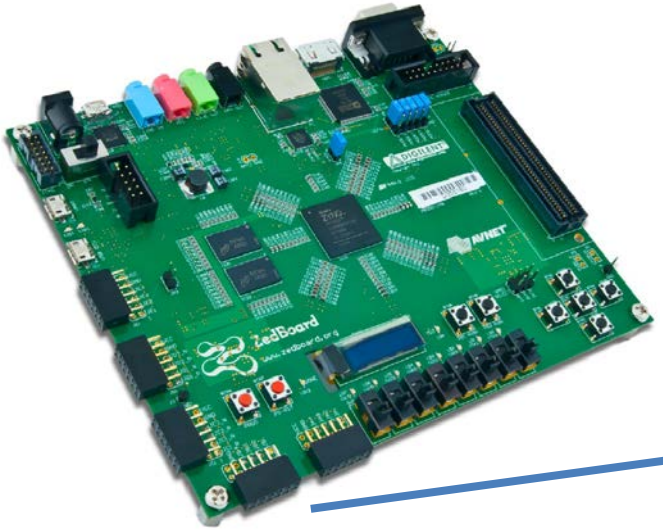
PMOD AD1 (2 channel, 12 bit, 1 MSPS)

Features include:

- two AD7476A 12-bit A/D converter chips
- a 6-pin header connector
- a 6-pin connector
- two 2-pole Sallen-Key anti-alias filters
- two simultaneous A/D conversion channels at up to one MSa per channel
- very low power consumption
- small form factor (0.95” x 0.80”).







Pmod	Signal Name	Zynq pin	
JB1	JB1	W12	CS
	JB2	W11	Data_1
	JB3	V10	Data_2
	JB4	W8	SCLK
	JB7	V12	
	JB8	W10	
	JB9	V9	
	JB10	V8	

Experiment 1: First freeRTOS application

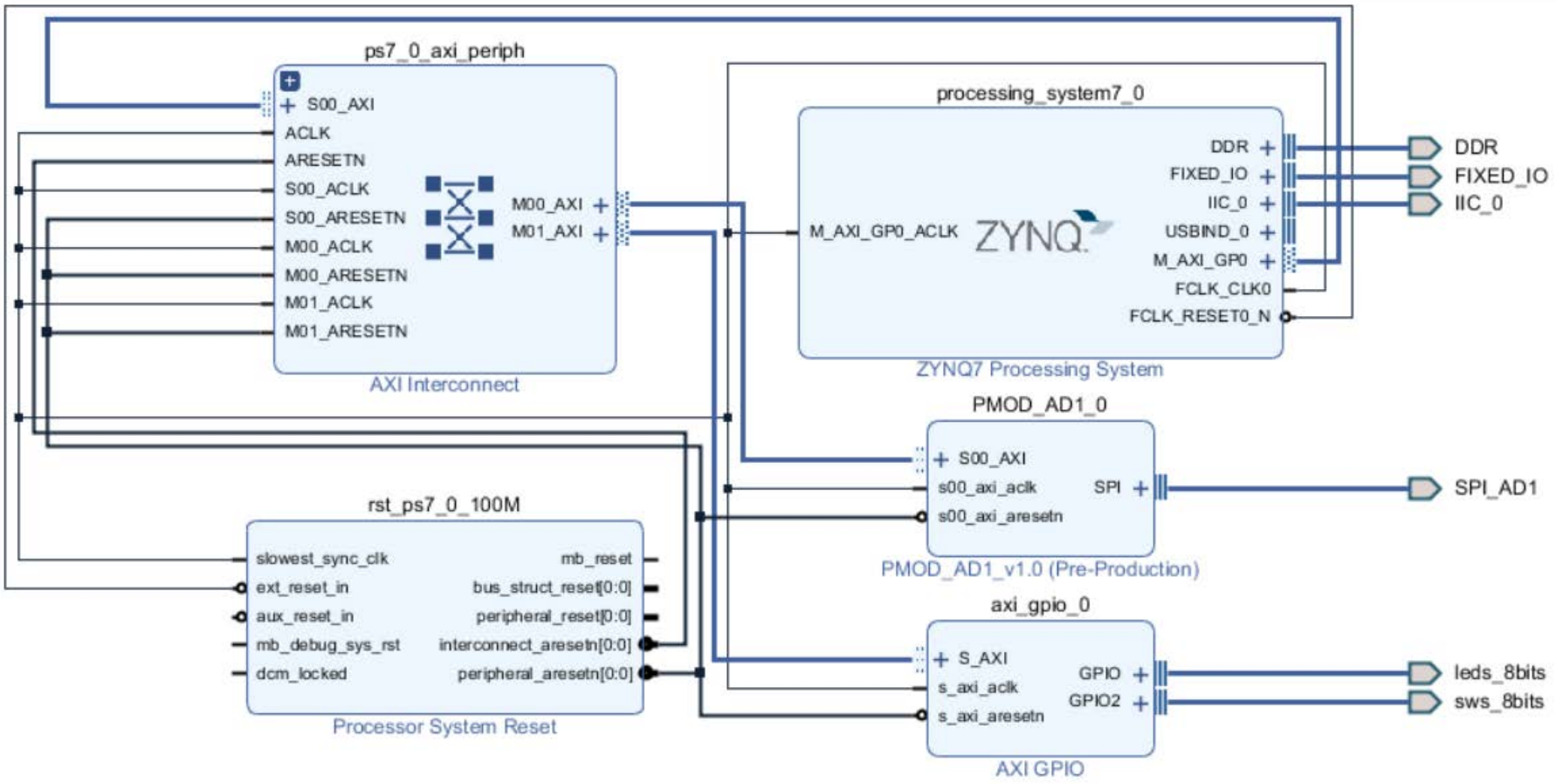
serial

PMOD AD Interface
- **Create IP Block**
- **VHDL code**

Create SOC Hardware
SDK: freeRTOS
- **µC code for DAQ**

Debug Terminal

VIVADO: Create Hardware Platform for Experiment



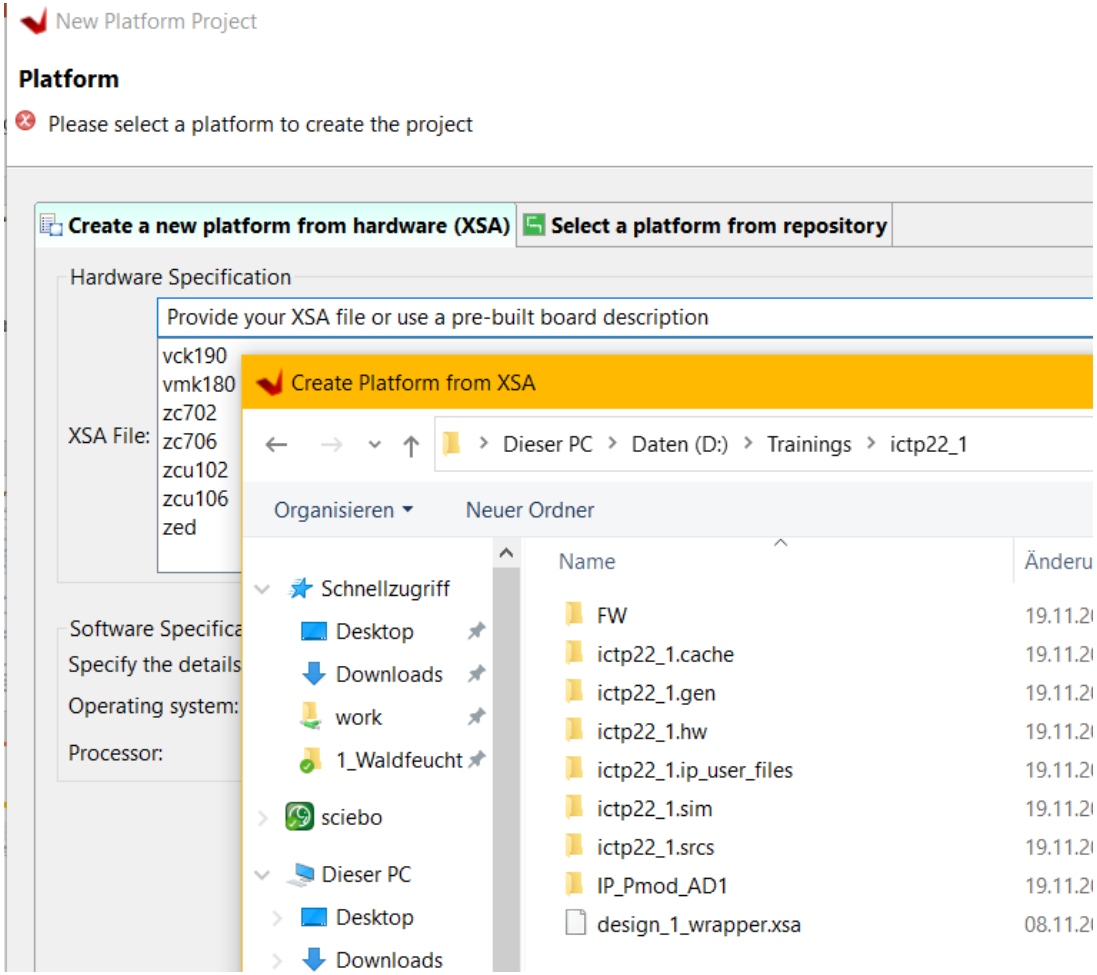
Constraints:

```
#PMOD AD1 on JB1
set_property IOSTANDARD LVCMOS33 [get_ports {SPI_AD1_*} ]

set_property PACKAGE_PIN W11 [get_ports SPI_AD1_io0_i]
set_property PACKAGE_PIN W10 [get_ports SPI_AD1_io1_i]
set_property PACKAGE_PIN W12 [get_ports SPI_AD1_ss1_o]
set_property PACKAGE_PIN W8  [get_ports SPI_AD1_sck_o]
```

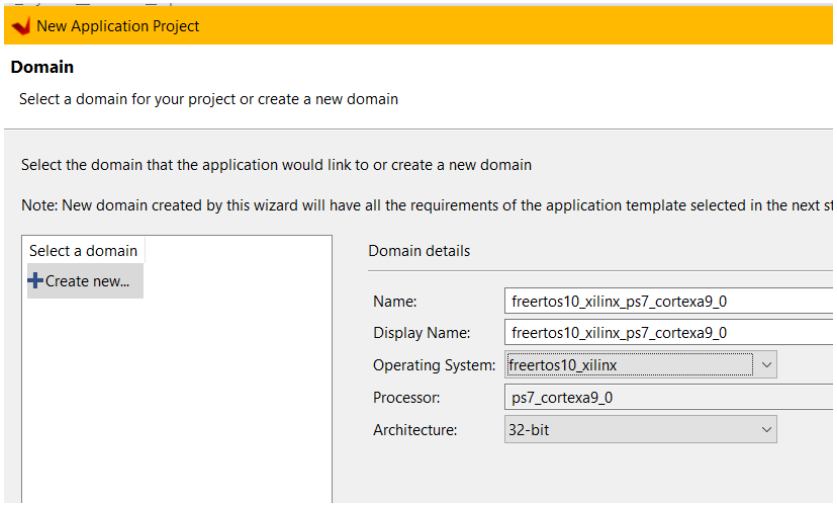
VITIS: Create Application Project

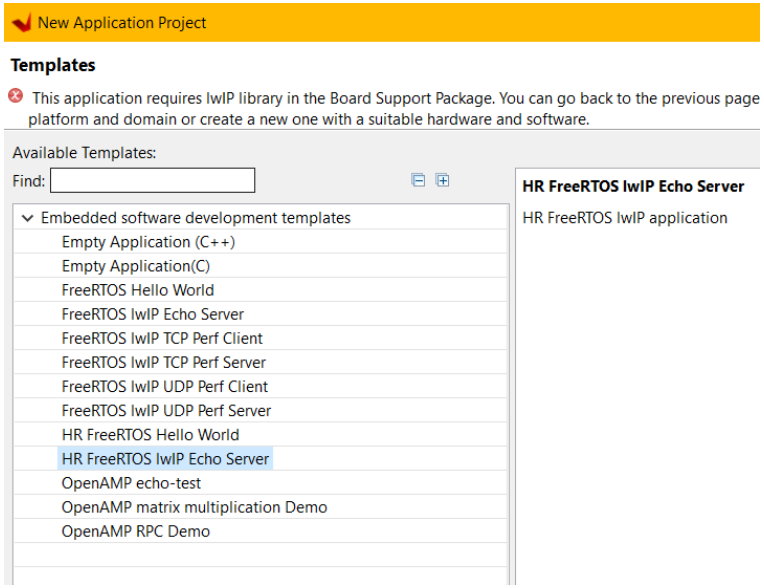
- 1) Create new Application Project
  - Platform: create from exported hardware *.XSA



VITIS: Create Application Project

- 2) Application Project
  - New Application project: Operating System: **freertos10_Xilinx**
  - Select template **'HR FreeRTOS lwIP echo server'**





VITIS:

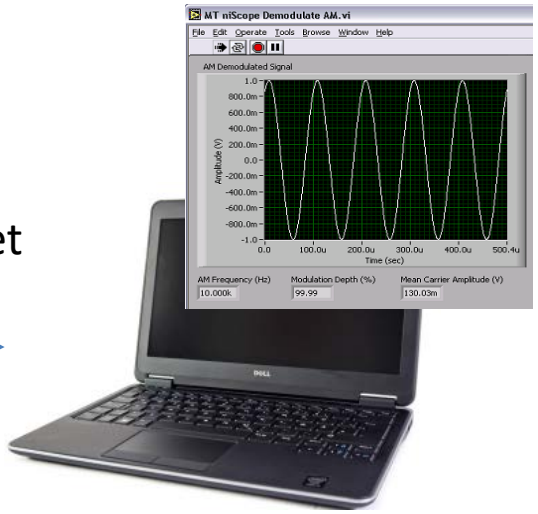
- Create a project based “Hr FreeRTOS lwIP echo server”
- Compile, make Run Configuration, download to Zedboard
- Connect a terminal to the Serial line Debug output (Teraterm, …)
- Check for “Hello World”

- Add a task for controlling the LEDs (via GPIO)

- Add a task for readout of the ADC
  - Print the ADC value

Experiment 2: Realtime Data Acquisition System based on freeRTOS & lwIP

Ethernet
TCP/IP



PMOD AD Interface
- **Create IP Block**
- **VHDL code**

Create SOC Hardware
SDK: freeRTOS + lwIP
- **µC code for DAQ**
- **and Ethernet communication**

Network software
- **Receive Data**
- **Visualization**

VITIS:

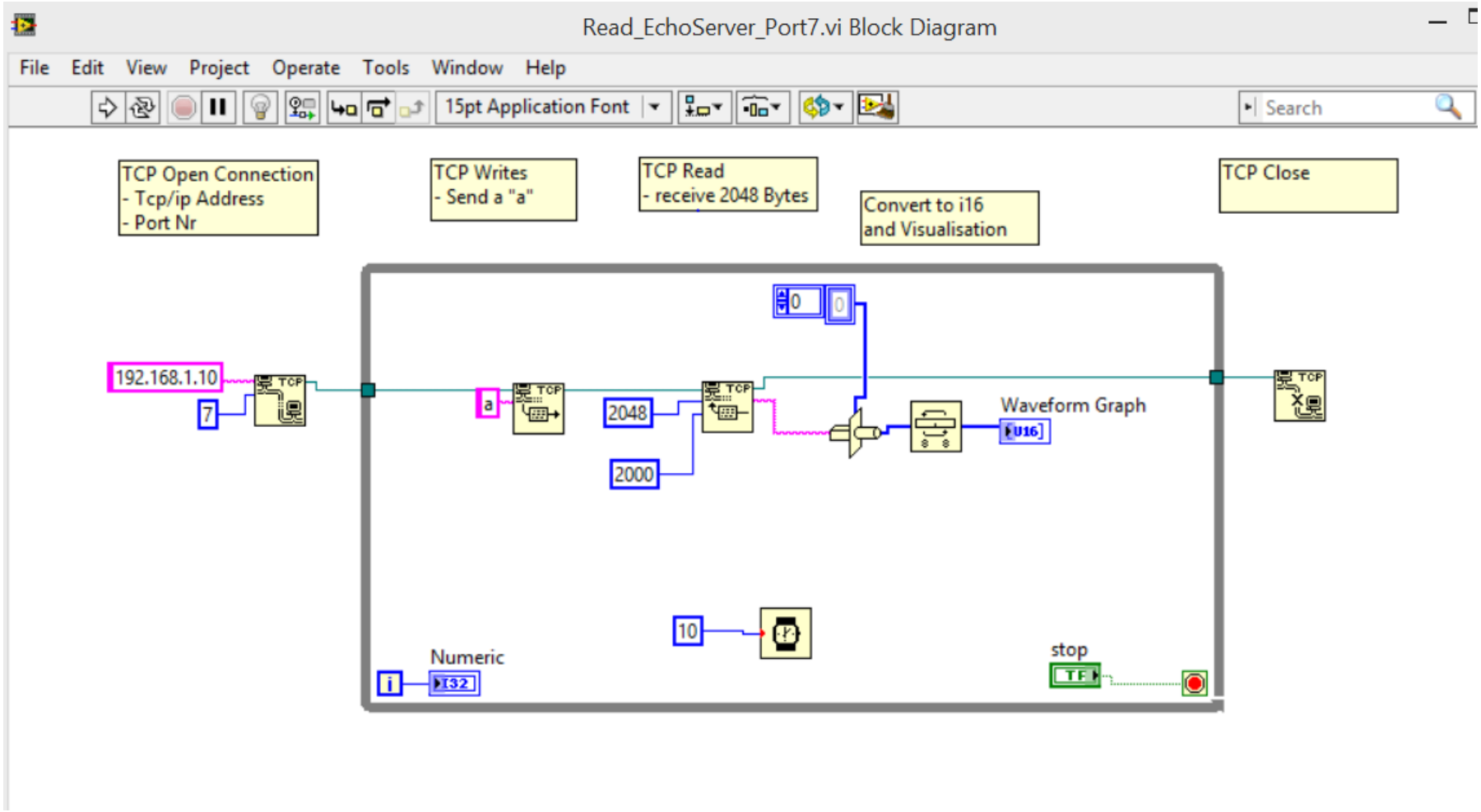
- Enable ‘task_StartEthernet‘ in main.c
- Setup the PC network to match the Zedboard IP address space (192.168.2.2)
- Check the debug serial for established Ethernet connection
- On the PC:
  - Use ‘ping’ for checking the Ethernet connection
  - Start a Brower and connect to 192.168.2.2

- Modify the ‘webserver’ to enable the ADC
  - Print the ADC value in html language
  - Check with the browser

  - Enable the html code for the visualization of the graph
  - Check with the browser

Appendix

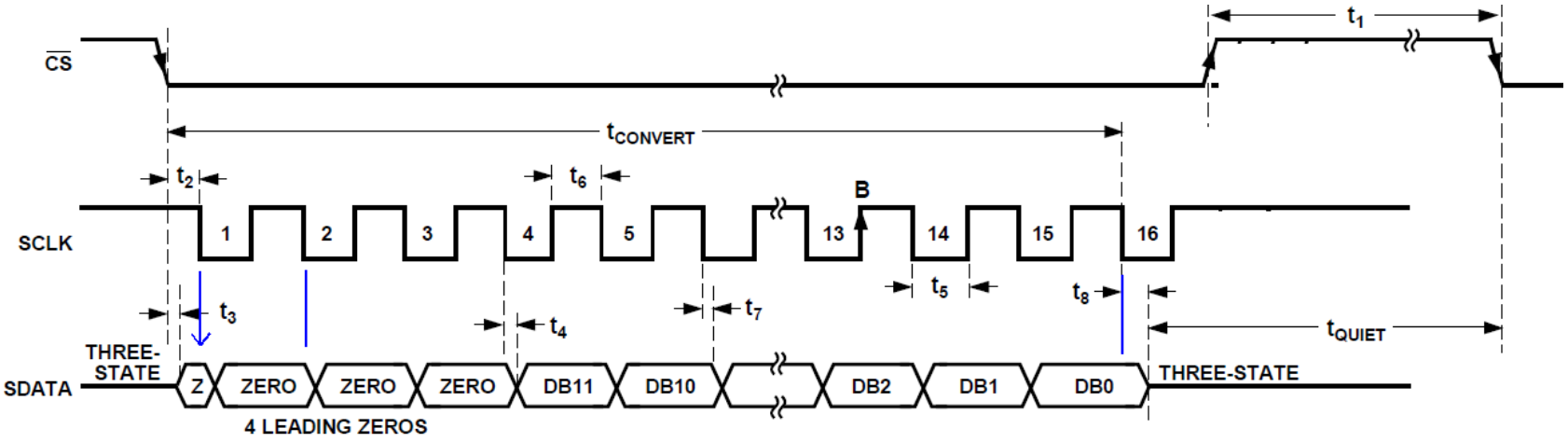
Labview Client for Visualization



AD7476 serial ADC: Timing

- Cycle time: 1 µs (=1 MHz Sample rate)
  - AXI Clock (100 MHz) divide by 100
- SCLK
  - Max. 20 MHz = AXI Clock / 5
  - Realized in a Statemachine of 5 states

0) SCLK =1 / CS = 0
1) SCLK =1
2) SCLK =0 / Read serial Bit
3) SCLK =0 / increment Bit nr
4) SCLK =0 / check for 16 bits
   Jump to 0 or IDLE



VHDL sniplet

use ieee.std_logic_unsigned.all;

```
port ( -- Users to add ports here
        CS          : out std_logic;
        SCLK        : out std_logic;
        SDATA1      : in std_logic;
        SDATA2      : in std_logic;
…
```

```
          type      TSTATES   is        (IDLE, S0, S1, S2, S3, S4 );
          signal    sState              :         TSTATES;
  signal  Pacer1M     : std_logic;

  signal  AdcData1      : std_logic_vector (15 downto 0);
  signal  AdcData2      : std_logic_vector (15 downto 0);
```

```
--*************** Add user logic here  *********************************************

p1Mhz:      process (S_AXI_ACLK)
                variable Prediv : std_logic_vector (7 downto 0);
            begin
                if rising_edge (S_AXI_ACLK) then
                    if (Prediv < 99) then
                        Prediv := Prediv +1;
                        Pacer1M <= '0';
                    else
                        Prediv := (others=>'0');
                        Pacer1M <= '1';
                    end if;
                end if;
            end process;
```

```
pAD7576t:  process (S_AXI_ACLK)
                variable BitNr : integer range 0 to 31;
                variable InDat1, InDat2 : std_logic_vector (15 downto 0);
           begin
            if rising_edge (S_AXI_ACLK) then
            case sState is
                when IDLE =>   SCLK    <= '1';
                               CS      <= '1';
                               BitNr   := 0;
                               if (Pacer1M = '1') then
                                 CS <= '0';
                                 sState <= S0;
                               end if;

                when S0    =>  SCLK <= '1';
                               sState <= S1;

                when S1    =>  SCLK <= '1';
                               sState <= S2;

                when S2    =>  SCLK <= '0';
                               InDat1 := InDat1 (14 downto 0) & SDATA1;       -- serial SHIFT IN
                               InDat2 := InDat2 (14 downto 0) & SDATA2;       -- serial SHIFT IN
                               BitNr   := BitNr +1;
                               sState <= S3;

                when S3    =>  SCLK    <= '0';
                               sState   <= S4;

                when S4    =>  SCLK <= '1';
                               if (BitNr < 16) then
                                          sState <= S0;
                               else
                                         CS <= '1';
                                         AdcData1 <= InDat1;
                                         AdcData2 <= InDat2;
                                         sState <= IDLE;
                               end if;
            end case;
            end if;
           end process;
```

Pin Constraints

```
#--- pinning for Pmod Connector B (upper level) ---------

set_property PACKAGE_PIN W12 [get_ports AD1_CS]
set_property PACKAGE_PIN W11 [get_ports AD1_SDATA1]
set_property PACKAGE_PIN V10 [get_ports AD1_SDATA2]
set_property PACKAGE_PIN W8 [get_ports  AD1_SCLK]

set_property IOSTANDARD LVCMOS33 [get_ports AD1_CS]
set_property IOSTANDARD LVCMOS33 [get_ports AD1_SCLK]
set_property IOSTANDARD LVCMOS33 [get_ports AD1_SDATA1]
set_property IOSTANDARD LVCMOS33 [get_ports AD1_SDATA2]
```

- Now Generate BitFile
- Export
- Launch SDK

Application based on ‚hrfreertos_lwip_web_tcp“
Changes in „Echo.c“

```
short AdData[1024];

void ReadADC_Fast (void)
{
int i;
  for (i=0; i<1024; i++)
  {
     AdData[i] = (short) *(int*)0x43C10000;                // direct IO access
     usleep(2);
  }
}

void process_echo_request(void *p)
{
….
while (1)
{
…
          /* handle request */
          ReadADC_Fast ();

          wrote = write(sd, AdData, 2048);
          {
…
}
```

Annex: Accessing memory mapped Hardware-Devices

Using BSP driver functions:

```
#include "xparameters.h"
#include "xgpio.h"

#define LED_CHANNEL     1
#define SW_CHANNEL      2

XGpio Gpio;             /* The Instance of the GPIO Driver */

            // GPIO Initialisation:
            XGpio_Initialize                    (&Gpio, XPAR_AXI_GPIO_0_DEVICE_ID);    //=0
            XGpio_SetDataDirection              (&Gpio, LED_CHANNEL,    0xFFFFFF00);   // 0 = Outputs
            XGpio_SetDataDirection              (&Gpio, SW_CHANNEL,     0xFFFFFFFF);   // 1 = Inputs

            //GPIO Data:
            Data = XGpio_DiscreteRead           (&Gpio, SW_CHANNEL);
            XGpio_DiscreteWrite                 (&Gpio, LED_CHANNEL, Data);
```

Using direct I/O functions:

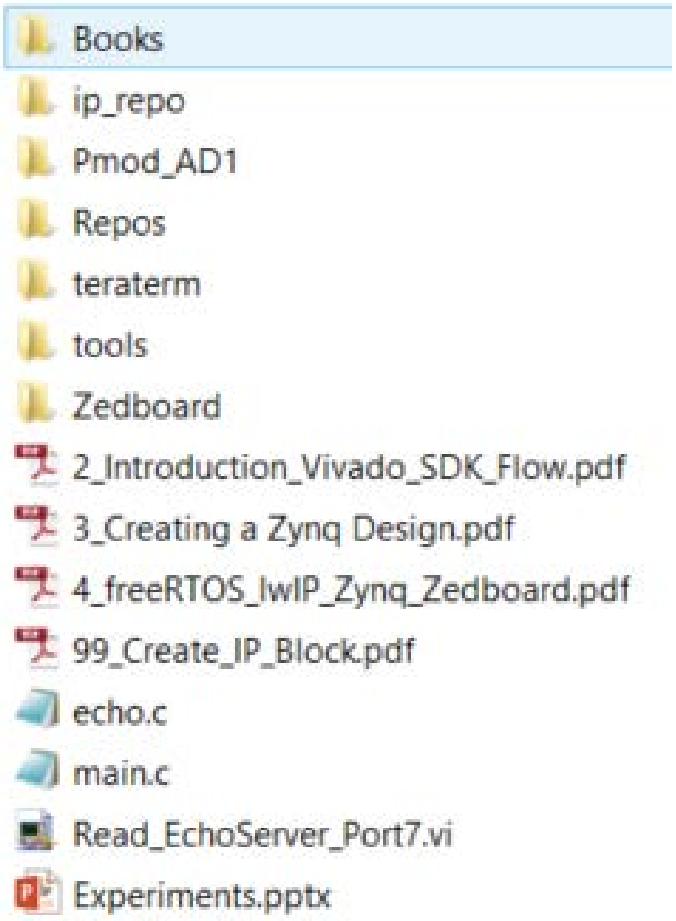
```
#include "xparameters.h"
#include “Xil_io.h”

            Xil_Out32 (Addr, Value)
            Xil_In32  (Addr)
```

Pointer usage:

```
#include "xparameters.h“
            *(u32*)Addr = Value;
```

On Stick



- Some books..
- VHDL IP Blocks for Vivado IP Integrator
- The PMOD documentation
- The Software Repository (freeRTOs & lwIP)
- A Terminal
- Some tools
- Zedboard Docu

- This file …!

Pulse Oximeter

Heart Beat Monitor

Dr. Heinz Rongen

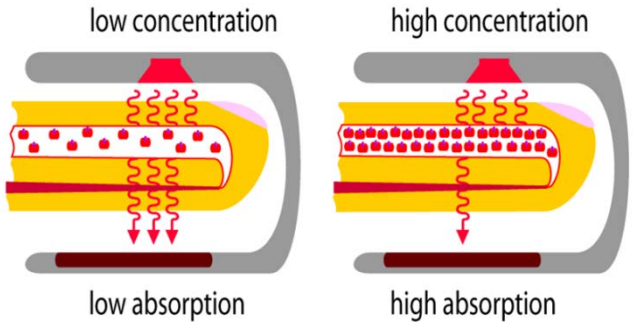
Pulse Oximeter

- medical device that indirectly monitors
  - the oxygen saturation of a patient's blood
  - and changes in blood volume in the skin

→ Monitoring of heart rate

- Infrared LED (900 to 950 nm)

- Absorption at these wavelengths differs significantly
  - between oxyhemoglobin and its deoxygenated form
  - the arterial expand and contract with each heartbeat

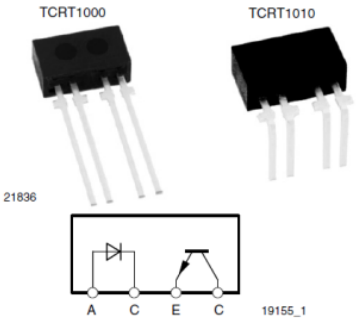


VISHAY

TCRT1000, TCRT1010

Vishay Semiconductors

Reflective Optical Sensor with Transistor Output



FEATURES

- Package type: leaded
- Detector type: phototransistor
- Dimensions (L x W x H in mm): 7 x 4 x 2.5
- Peak operating distance: 1 mm
- Operating range within > 20 % relative collector current: 0.2 mm to 4 mm
- Typical output current under test: I_C = 0.5 mA
- Daylight blocking filter
- Emitter wavelength: 950 nm
- Lead (Pb)-free soldering released
- Compliant to RoHS directive 2002/95/EC and in accordance to WEEE 2002/96/EC

Pb-free

e4

RoHS COMPLIANT

DESCRIPTION

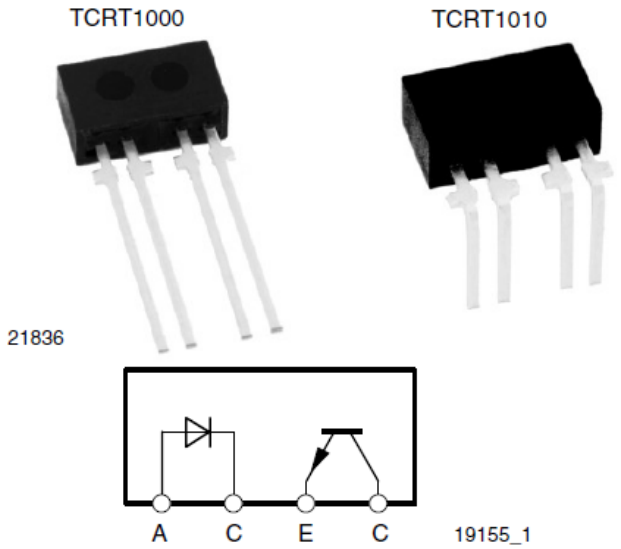
The TCRT1000 and TCRT1010 are reflective sensors which include an infrared emitter and phototransistor in a leaded package which blocks visible light.

APPLICATIONS

- Optoelectronic scanning and switching devices i.e., index sensing, coded disk scanning etc. (optoelectronic encoder assemblies for transmissive sensing).

TCRT1010
IR-LED and IR-Detector as reflective Sensor

Reflective Optical Sensor with Transistor Output



FEATURES

- Package type: leaded
- Detector type: phototransistor
- Dimensions (L x W x H in mm): 7 x 4 x 2.5
- Peak operating distance: 1 mm
- Operating range within > 20 % relative collector current: 0.2 mm to 4 mm
- Typical output current under test: I_C = 0.5 mA
- Daylight blocking filter
- Emitter wavelength: 950 nm
- Lead (Pb)-free soldering released
- Compliant to RoHS directive 2002/95/EC and in accordance to WEEE 2002/96/EC

Pb-free

e4

RoHS COMPLIANT

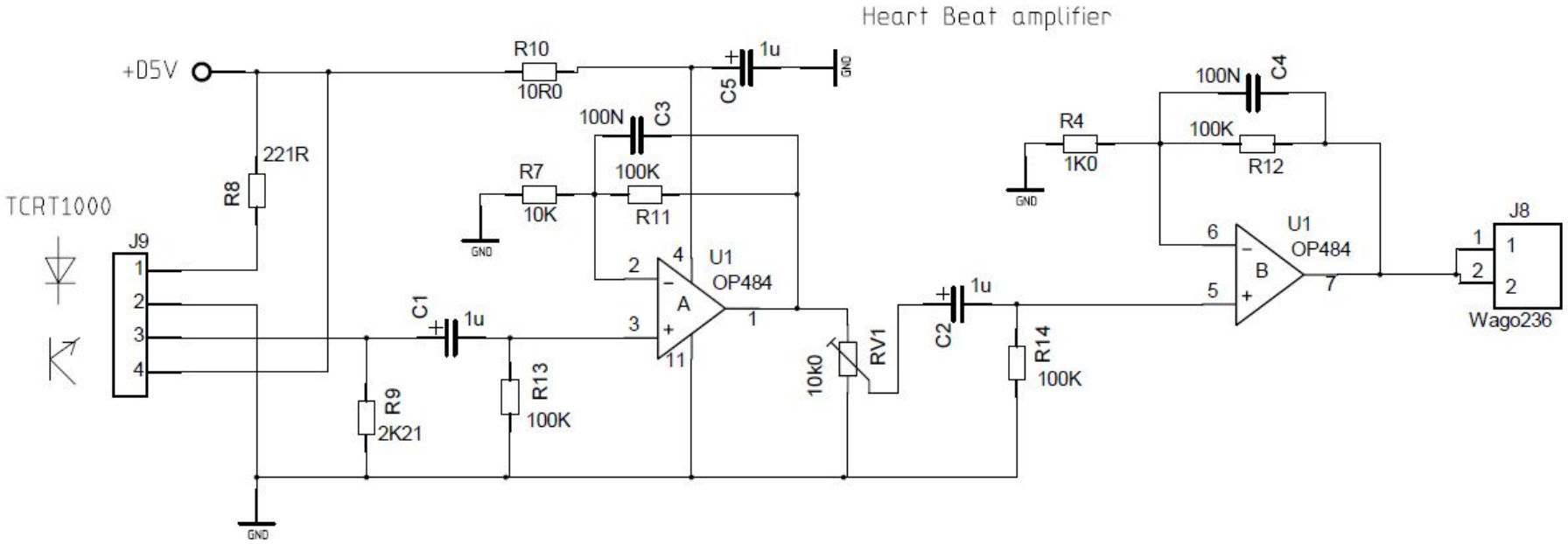
DESCRIPTION

The TCRT1000 and TCRT1010 are reflective sensors which include an infrared emitter and phototransistor in a leaded package which blocks visible light.

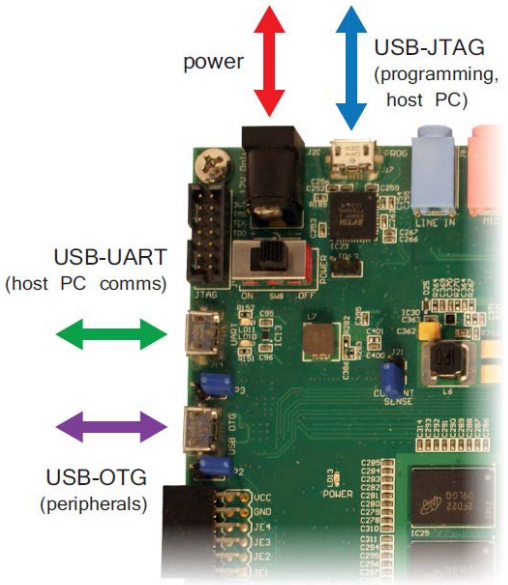
APPLICATIONS

- Optoelectronic scanning and switching devices i.e., index sensing, coded disk scanning etc. (optoelectronic encoder assemblies for transmissive sensing).

Preamplifier (with Filter)



ZYNQ Stand-Alone Booting (SD-Card or QSPI Memory)

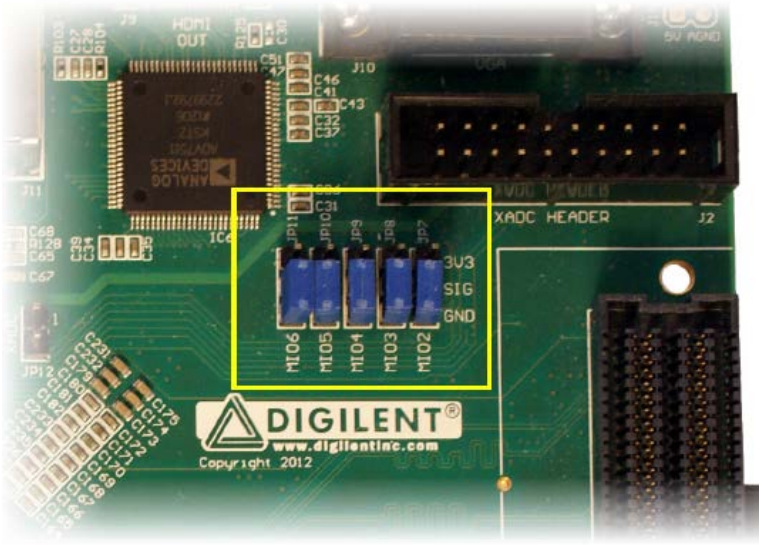


Programming the ZedBoard

- ***USB-JTAG*** — Default method for prototyping
- ***Traditional JTAG*** — A Xilinx JTAG connector.
  Requires a *Xilinx Platform USB* cable

- ***SD card*** — can be used to program the ZYNQ with files stored on the SD card.

- ***Quad-SPI flash memory*** — Non-volatile flash-memory.



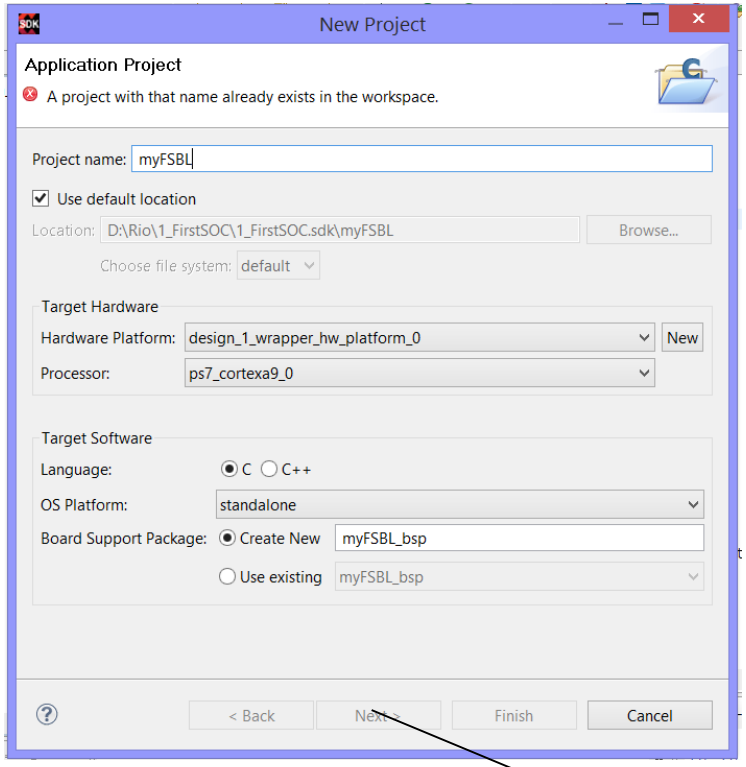
- **Boot mode latched at POR using MIO pins by BootROM**

Mode	MIO[5] (JP10)	MIO[4] (JP9)	MIO[3] (JP8)
JTAG	0	0	0
NOR	0	0	1
NAND	0	1	0
Quad-SPI	1	0	0
SD Card	1	1	0

Prepare for a SD card Image

Create a First Step Boot Loader (FSBL)

- File → New → Application

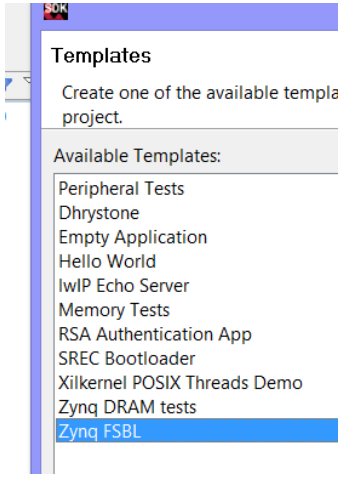


- Name is "**_FSBL"

- Let him create a new BSP or the FSBL

Click **NEXT**

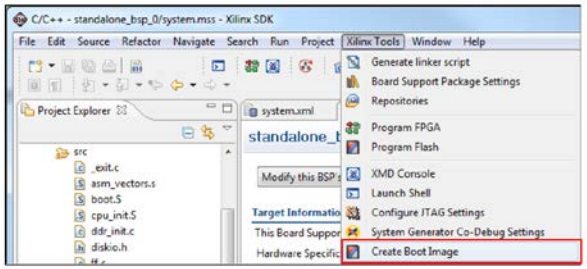
and select the **"Zynq FSBL"**

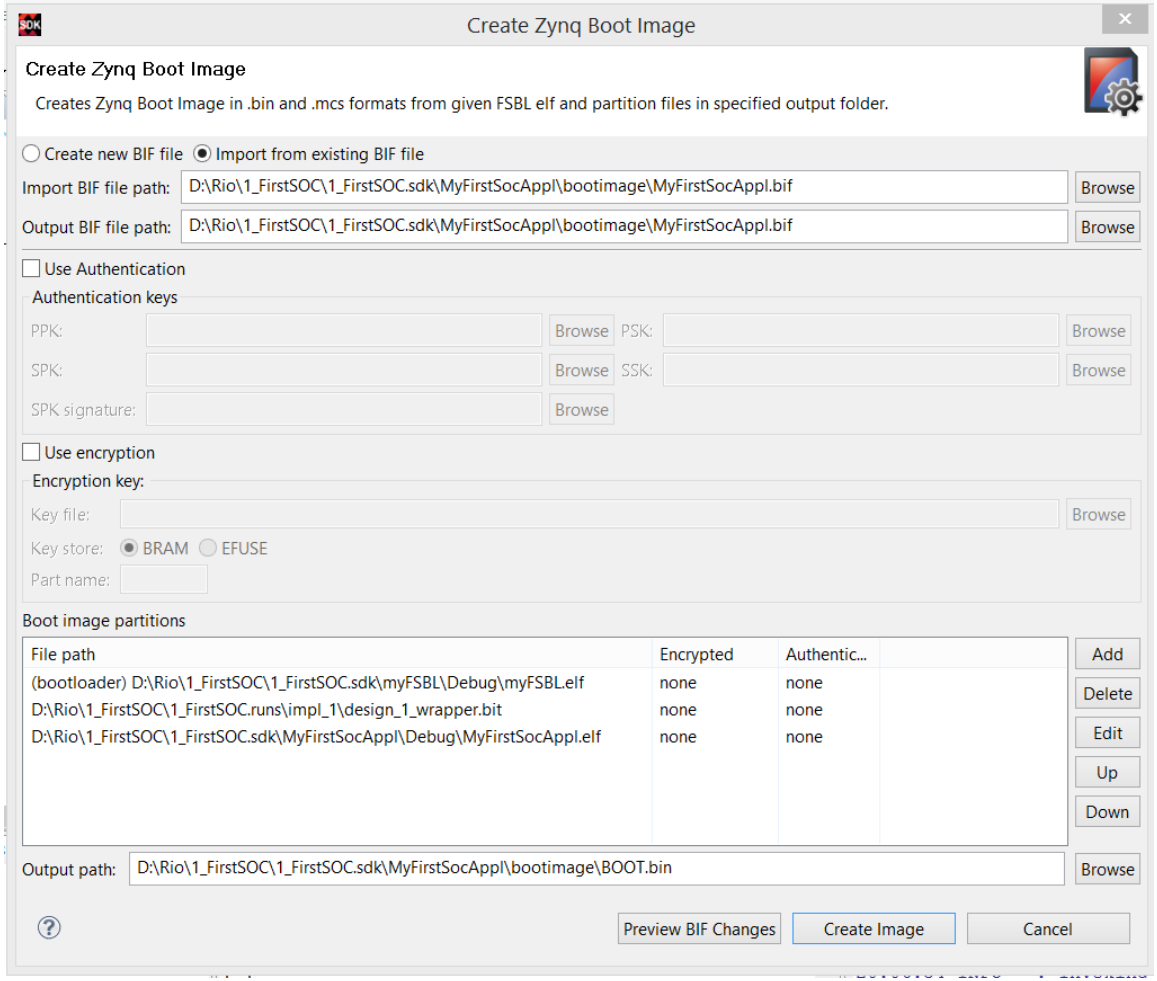


The FSBL is now created

Create a Zynq Boot Image for SD Card

Xilinx Tools → Create Zynq Boot Image





Add the following files:

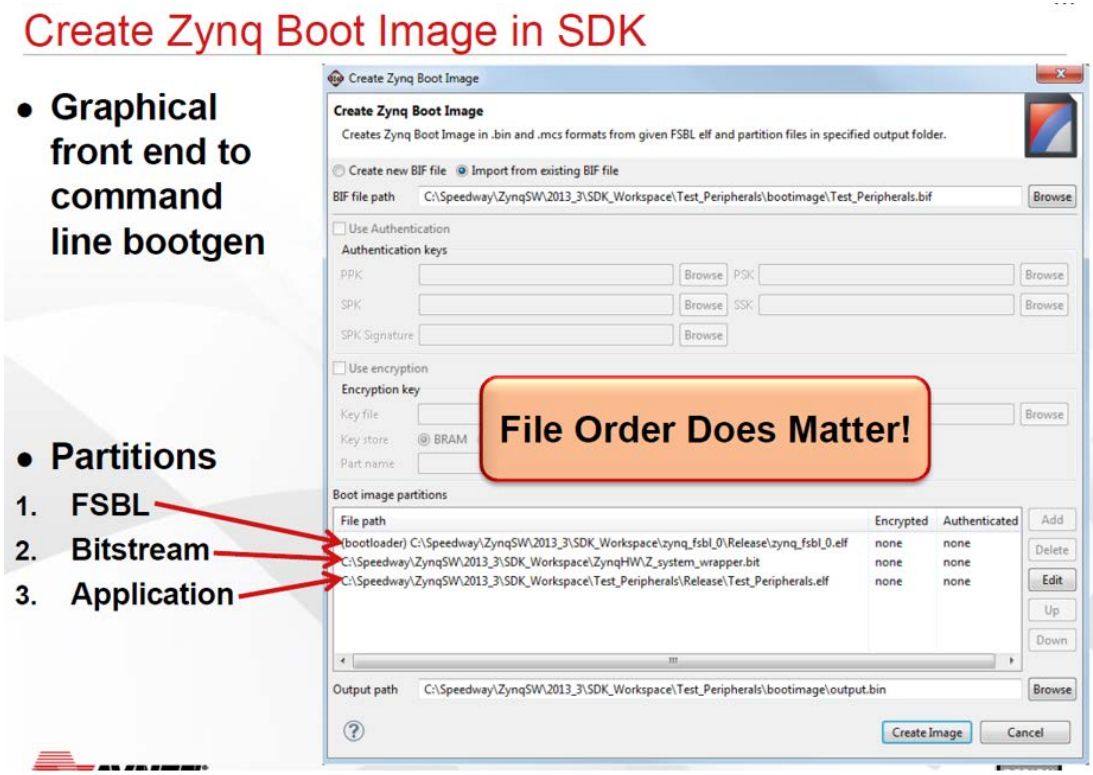
- the FSBL
- the BIT file
- the ELF file

- Click “Create Image”

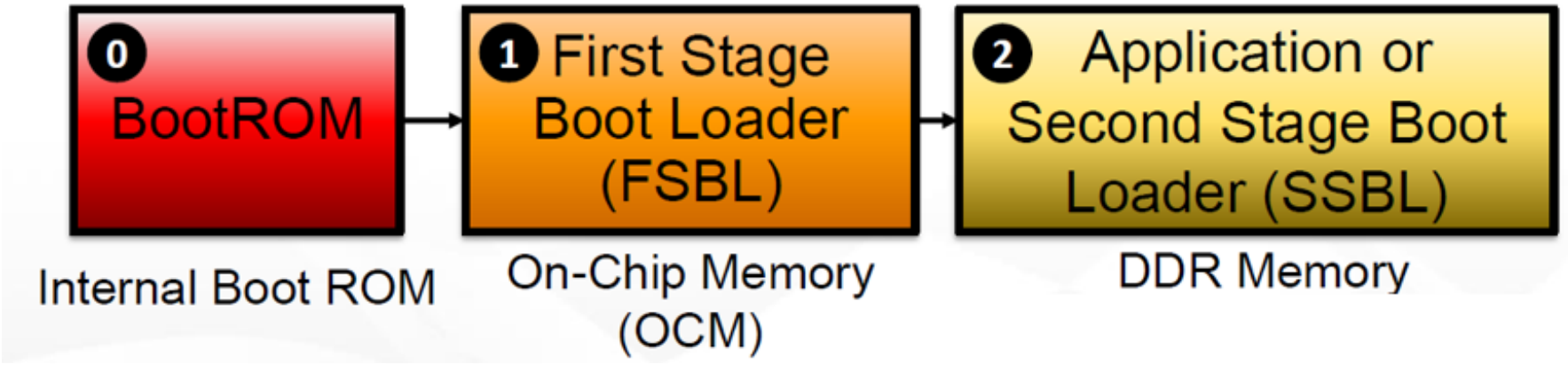
- BOOT.BIN is created

Directories to find the components

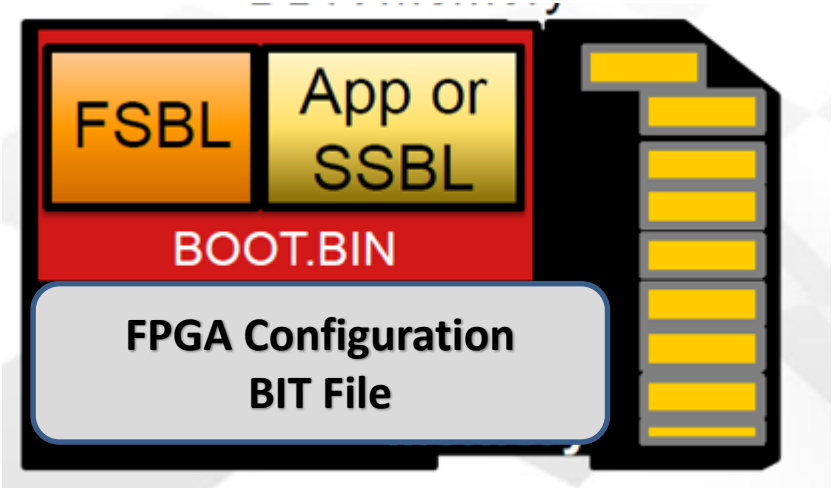
- **First Step Boot Loader**
  ***.sdk\myFSBL\Debug**
- **FPGA configuration**
  ***.runs\impl_1**
- **Application**
  ***.sdk\MyFirstSocAppl\Debug**



- **Is the order of the boot images critical? If so, list the order.**
  - YES! FSBL ELF, then bitstream, then application ELF

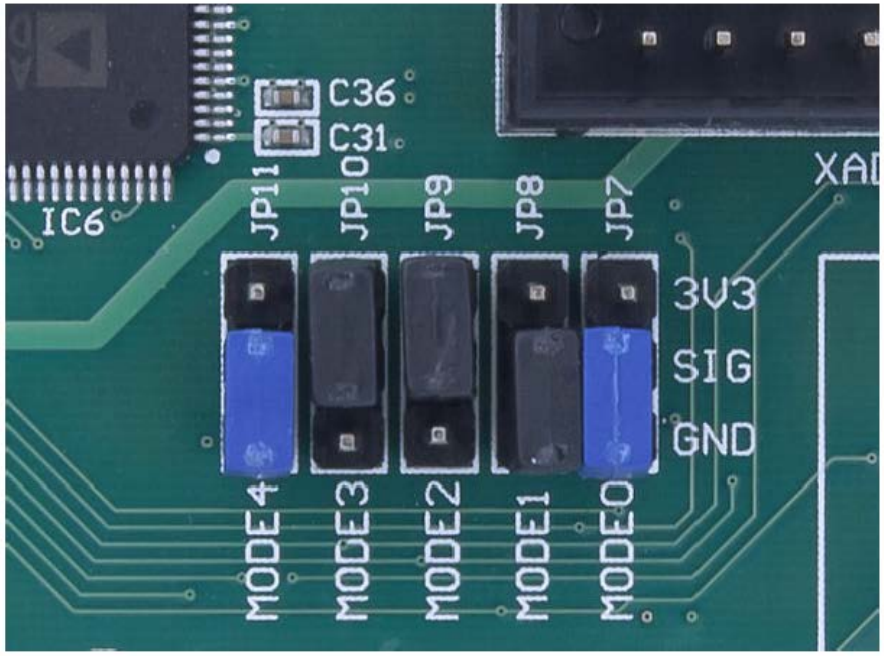


- all in **BOOT.BIN**



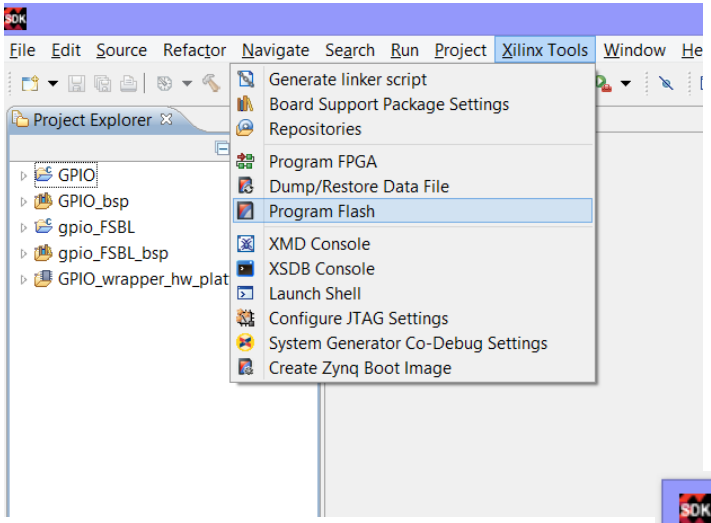
Booting from SD Card

- Copy BOOT.BIN on a SD card
- Set Jumper according
- Insert CD card
- Power Cycle the ZedBoard



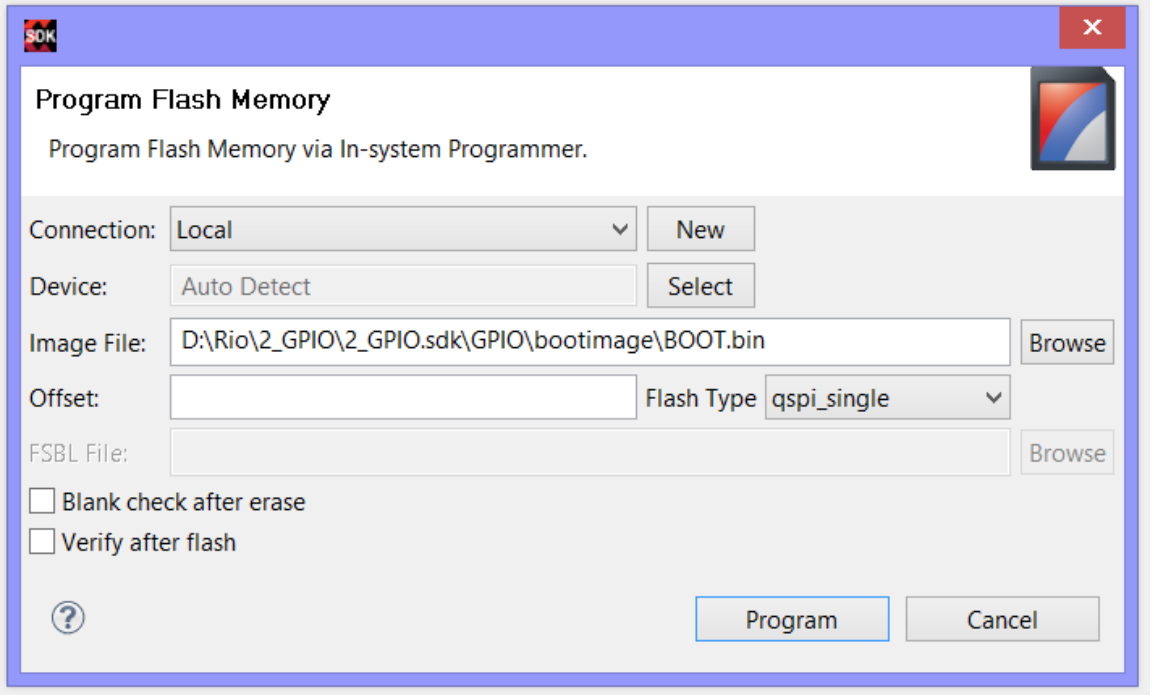
Booting from FLASH (QSPI Memory)

- Xilinx Tools → Program Flash



- Browse the BOOT.BIN
- → Program

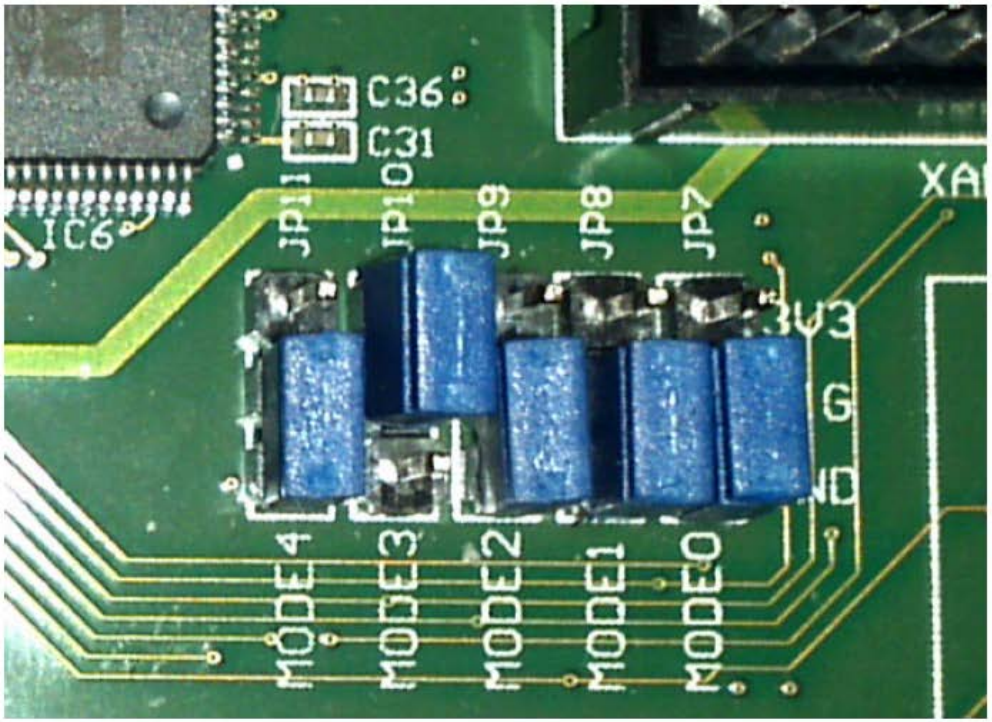
- Needs some time …



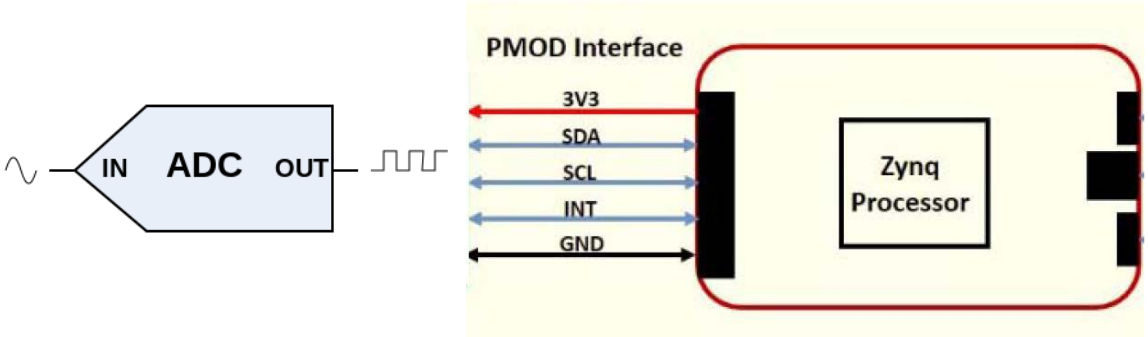
Booting from FLASH (QSPI Memory)

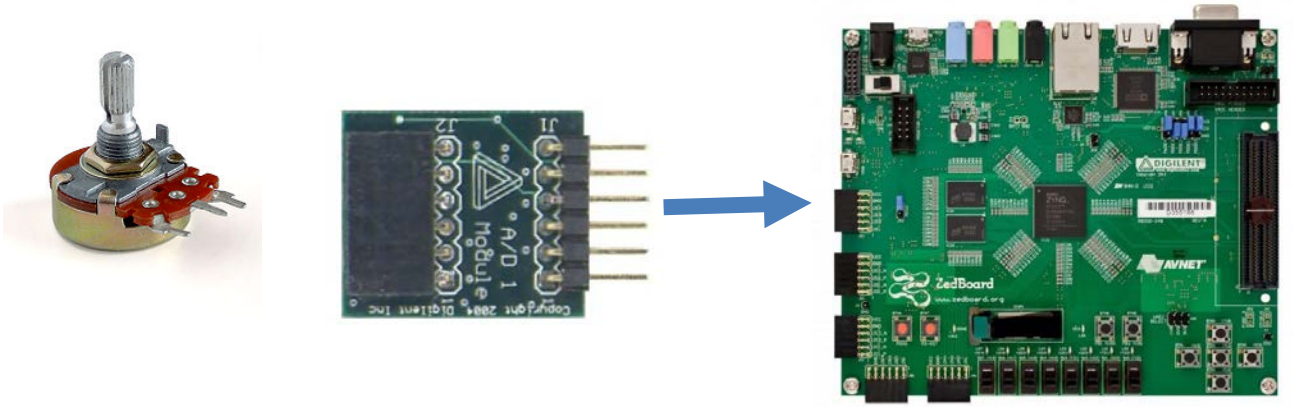
. Turn off the ZedBoard. Verify the Configuration Mode jumpers are set for QSPI boot mode as described and in the figure below:

- MODE3 (JP10) shunted to 3.3V
- All other MODE pins shunted to GND



Generic Experiment Setup

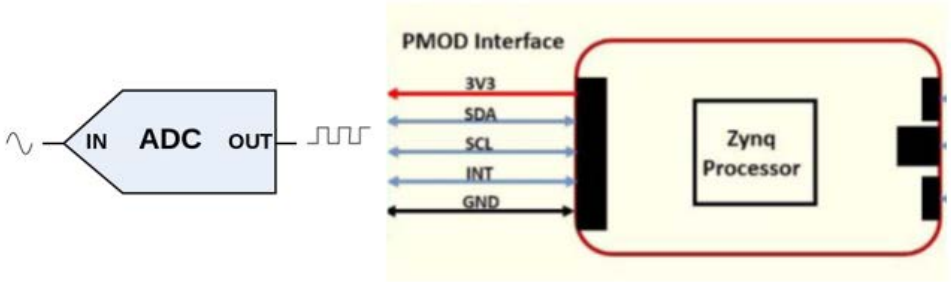


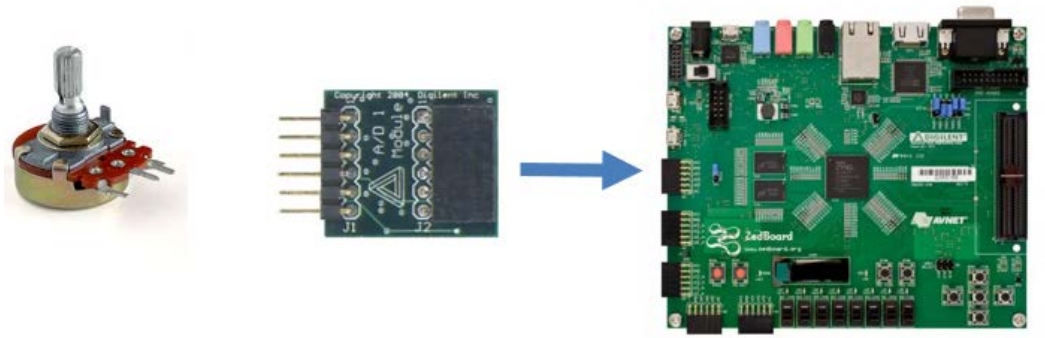


PMOD AD1 Interface
- **Create IP Block**
- **VHDL code**

Vivado: Create SOC Hardware
SDK: freeRTOS based Application
- **µC code for DAQ**

Demonstration 1





PMOD AD1 Interface
- **Create IP Block**
- **VHDL code**

Vivado: Create SOC Hardware
SDK: freeRTOS based Application
- **µC code for DAQ**