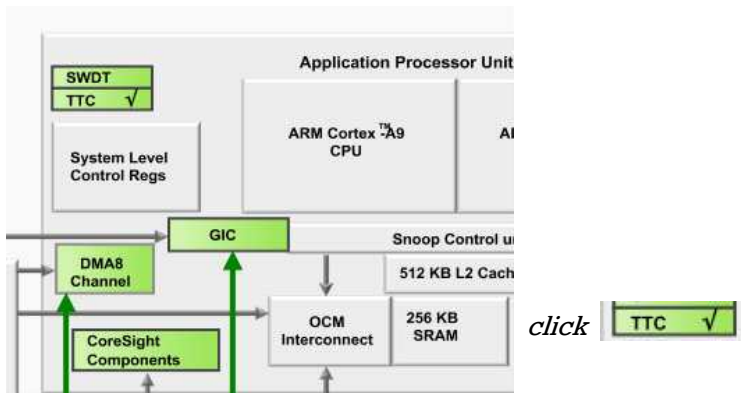
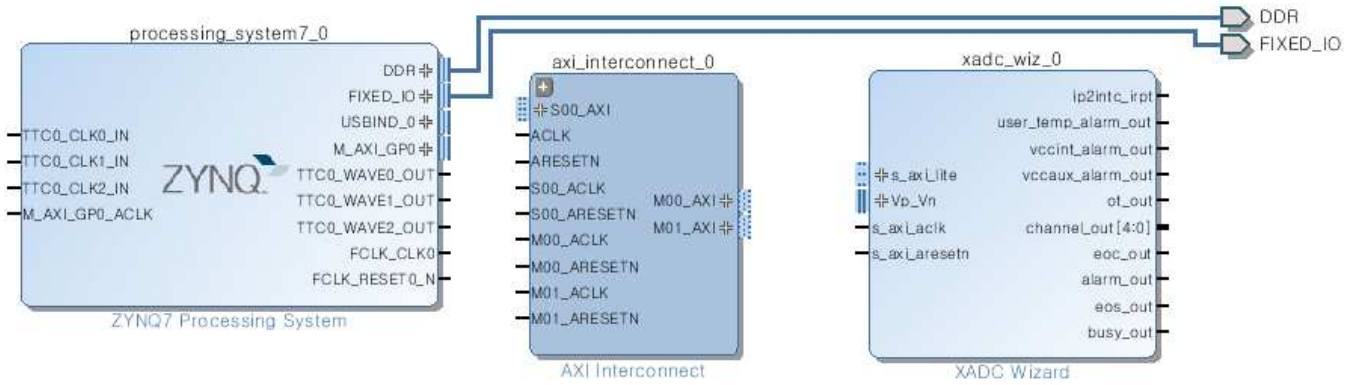


# 1. Make Project (XAdcTest1.xpr)

## 2. Create Block Design



Bank 0 IO Voltage: LVCMOS 3.3V Bank 1

Search:

Peripheral	IO	Si
Memory Interfaces		
I/O Peripherals		
Application Processor Unit		
<input checked="" type="checkbox"/> Timer 0	EMIO	
<input type="checkbox"/> Timer 1		
<input type="checkbox"/> Watchdog		
Programmable Logic Test and Debug		

*Unselect* Timer0, *OK*

*Deselect* USB0 and Ethernet0

*double click* IP, axi\_interconnec\_t0

**Top Level Settings** Slave Interfaces

Number of Slave Interfaces	1
Number of Master Interfaces	1
Interconnect Optimization Strategy	Custom

*change* Number of Master Interfaces to 1, *OK*

double click IP, xadc\_wiz\_0

Select  AXI4Lite  Continuous Mode  Channel Sequencer (Basic Tab)

Component Name XAdcSysMon\_xadc\_wiz\_0.0

Basic | ADC Setup | Alarms | Channel Sequencer | Summary

Interface Options:  AXI4Lite  DRP  None

Startup Channel Selection:  Simultaneous Selection  Independent ADC  Single Channel  Channel Sequencer

AXI4STREAM Options:  Enable AXI4Stream  
FIFO Depth: 7 [7..1020]

Control/Status Ports:  reset\_in  Temp Bus  JTAG Arbiter

Event Mode Trigger:  convst in  convstclk in

Timing Mode:  Continuous Mode  Event Mode

DRP Timing Options:  Enable DCLK  
DCLK Frequency(MHz): 100 [8.0..250.0]  
ADC Conversion Rate(KSPS): 1000 [154.0..1000.0]  
Acquisition Time (CLK): 4  
Clock divider value = 4  
ADC Clock Frequency(MHz) = 25.00

Analog Sim File Options: Sim File Selection: Default  
Analog Stimulus File: design  
Sim File Location: ./

Select Sequencer Mode Continuous Channel Averaging 16

ADC Offset and Gain Calibration  Sensor Offset and Gain Calibration  Enable CALIBRATION Averaging

Component Name XAdcSysMon\_xadc\_wiz\_0.0

Basic | ADC Setup | Alarms | Channel Sequencer | Summary

Sequencer Mode: Continuous Channel Averaging: 16

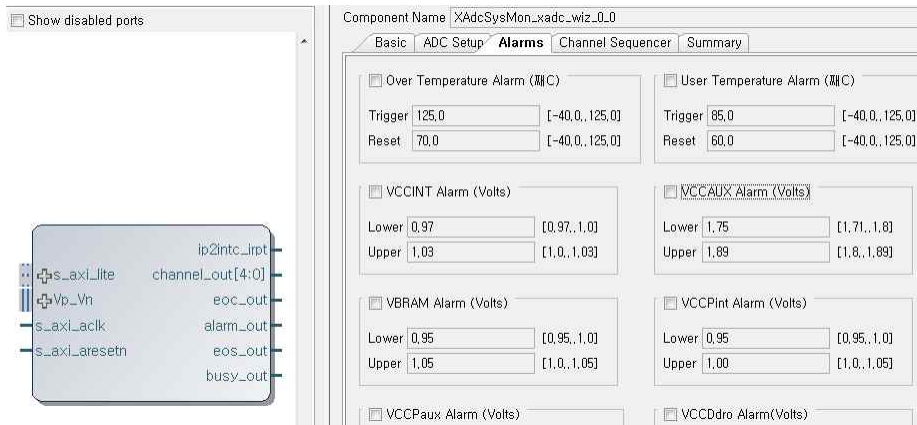
ADC Calibration:  ADC Offset Calibration  ADC Offset and Gain Calibration

Supply Sensor Calibration:  Sensor Offset Calibration  Sensor Offset and Gain Calibration

Enable CALIBRATION Averaging

External Multiplexer Setup:  External Multiplexer  
Channel for MUX: VP\_VN

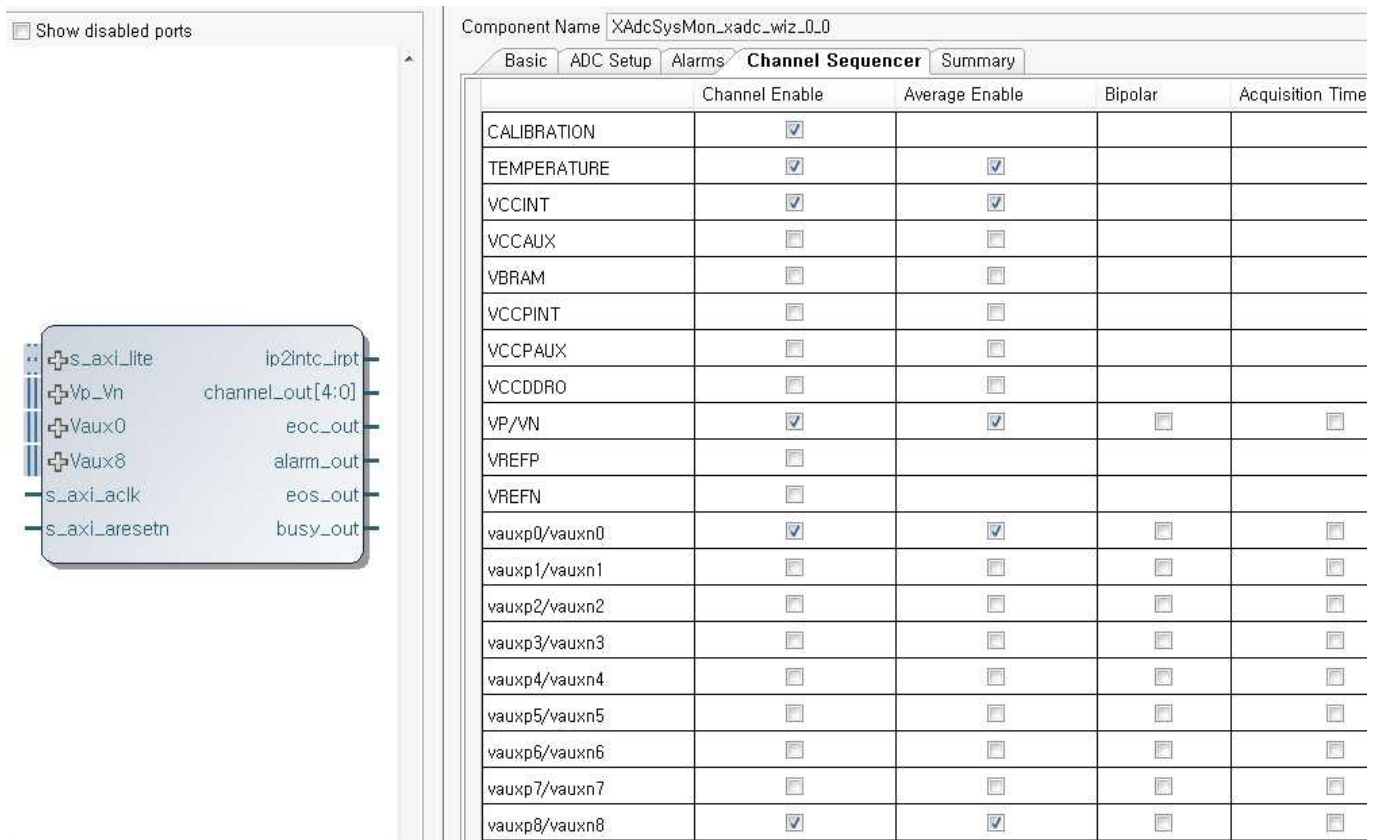
Power Down Options:  ADCB  ADCA



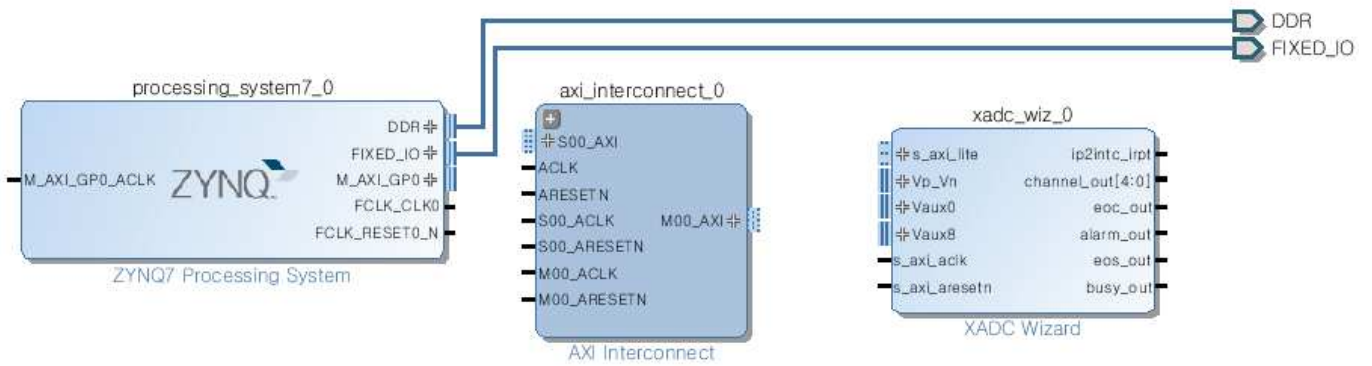
*Deselect all alarms*

**Select** TEMPERATURE, CALIBRATION, VP/VN ,VCCINT, vauxp0/vauxn0 ,vauxp8/vauxn8

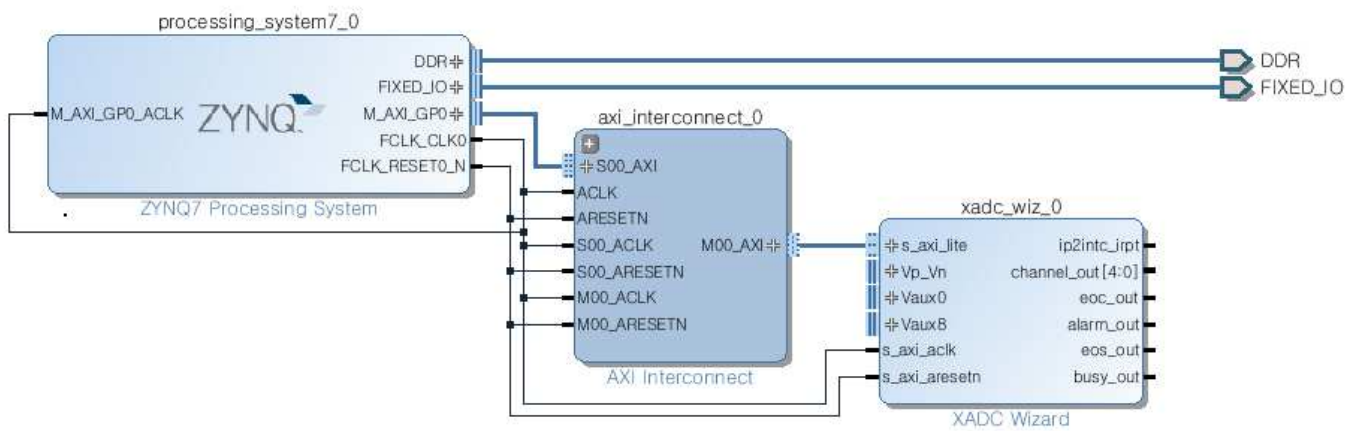
**Check** Average Enable



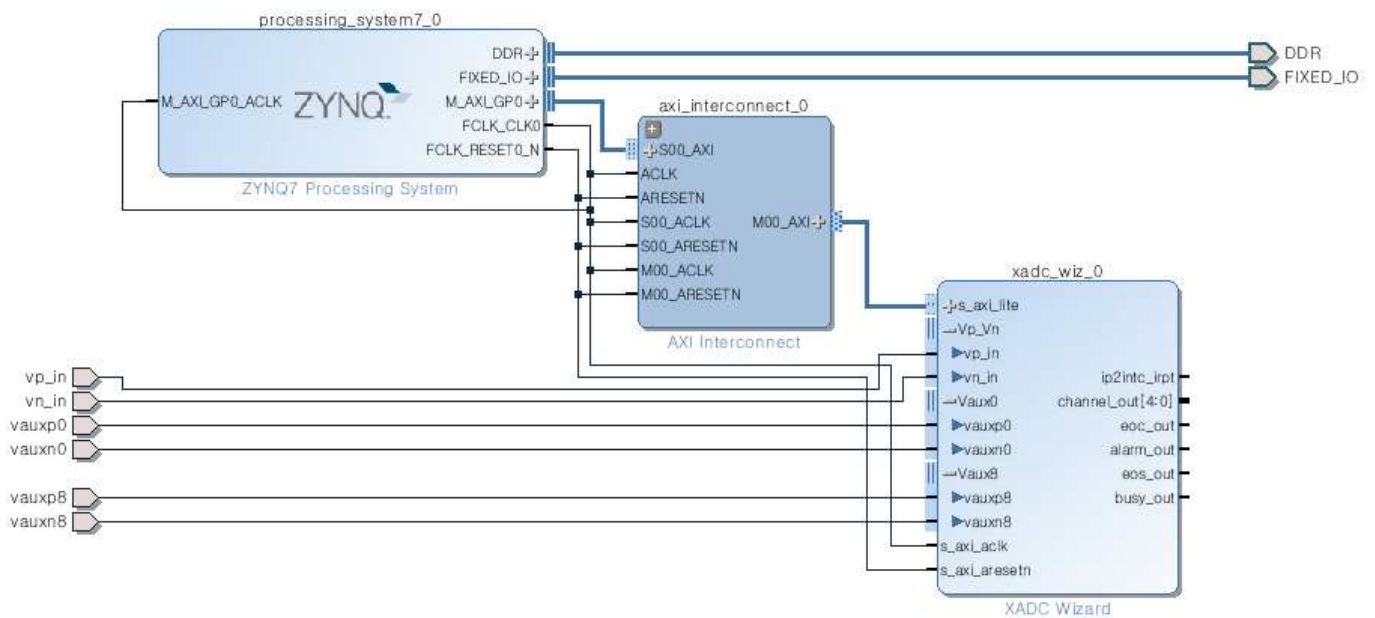
**Click** Summary, *OK*



**Make Connections**



**Make Xadc ports**

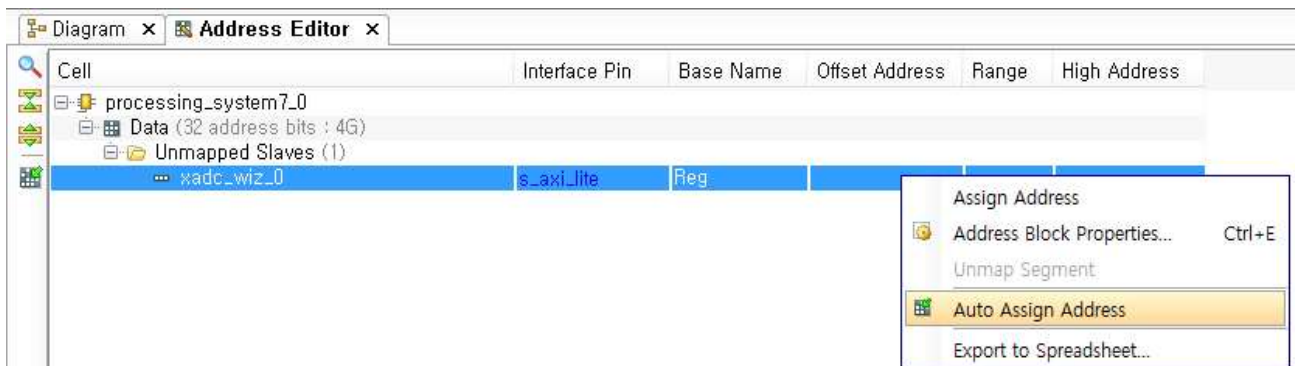


**Validate**



OK

Click **Address Editor** tab. select **Auto Assign Address**

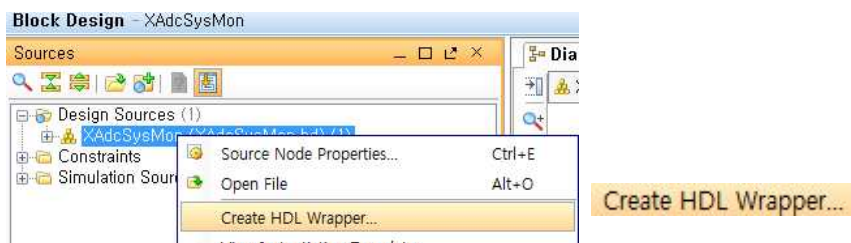


Check XAdc Base address (0x43C00000)

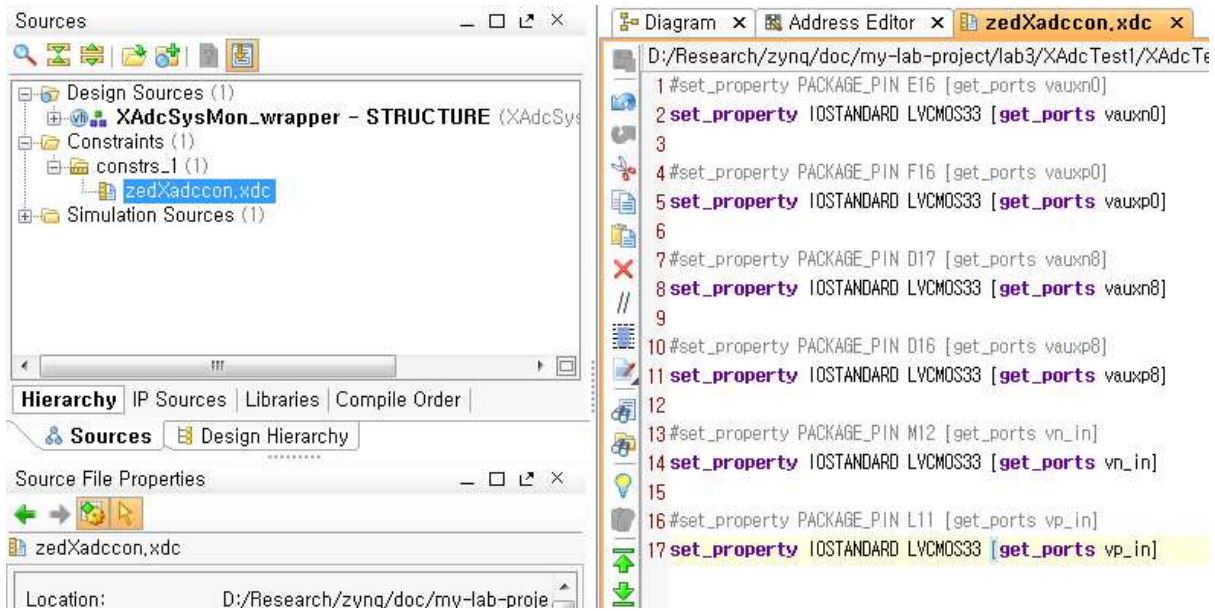


Validate again

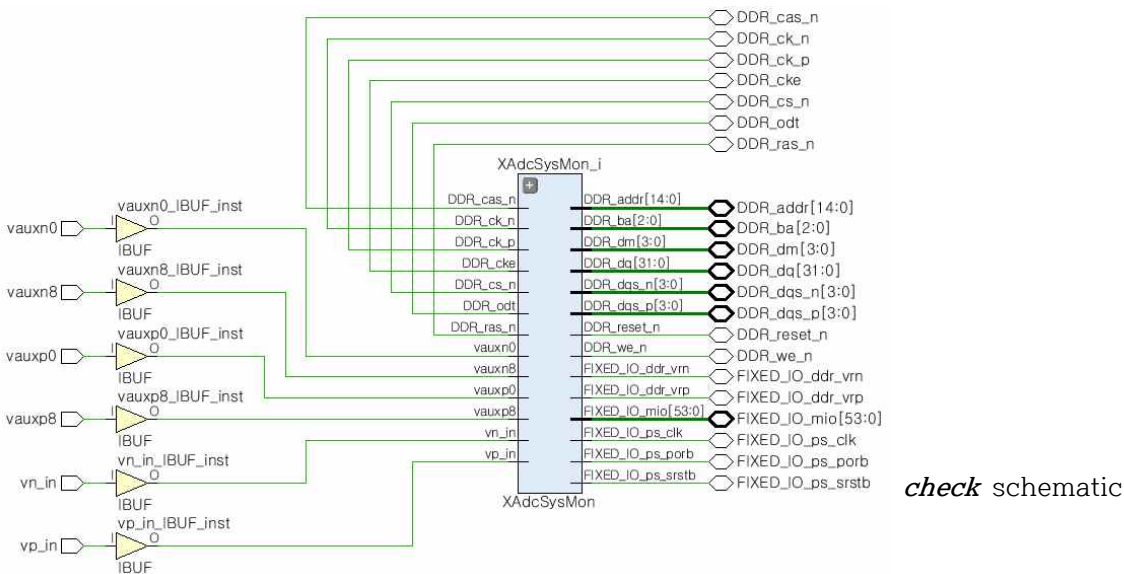
Generate Block Design



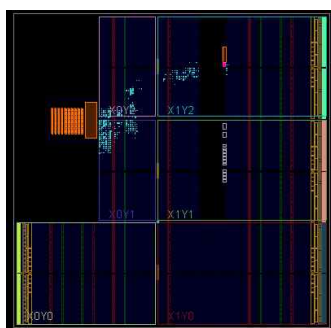
**Make** constraints file



**3. Run Synthesis**

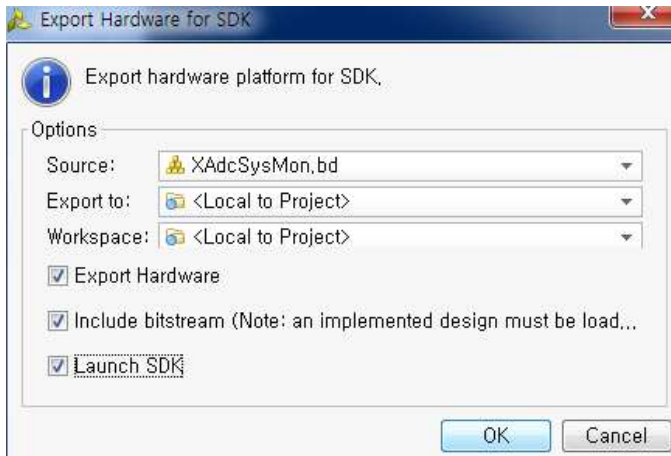


**4. Generate Bitstream**



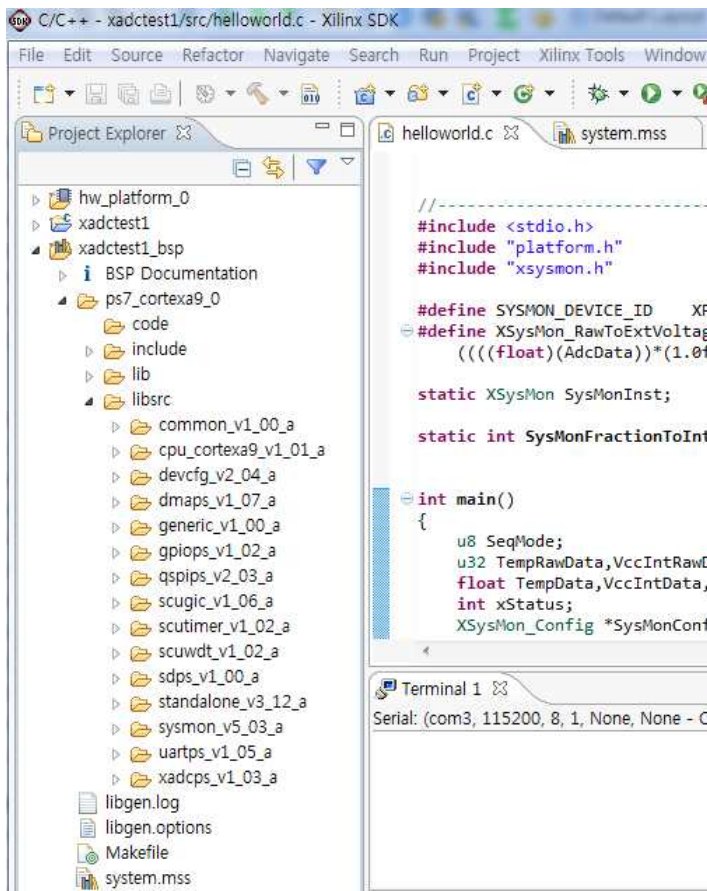
Implemented Design

## 5. Export to SDK



OK

## 6. Make an application project (xadctest1)



check if sysmon\_v5\_03\_a is in the libsrc

```

helloworld.c  system.mss
ps7_ram_0 generic
ps7_ram_1 generic
ps7_scuc_0 generic
ps7_scugic_0 scugic Documentation Examples
ps7_scutimer_0 scutimer Documentation Examples
ps7_scuwdt_0 scuwdt Documentation Examples
ps7_sd_0 sdps Documentation
ps7_slcr_0 generic
ps7_uart_1 uartps Documentation Examples
ps7_xadc_0 xadcps Documentation
xadc_wiz_0 sysmon Documentation Examples

```

refer to the example codes, [xadc\\_wiz\\_0 sysmon](#) [Documentation](#) [Examples](#)

**Edit** source code (helloworld.c)

```

//-----
#include <stdio.h>
#include "platform.h"
#include "xsysmon.h"

#define SYSMON_DEVICE_ID      XPAR_SYSMON_0_DEVICE_ID //ID of xadc_wiz_0
#define XSysMon_RawToExtVoltage(AdcData)              \
    (((float)(AdcData))*(1.0f))/65536.0f // (ADC 16bit result)/16/4096 = (ADC 16bit result)/65536 \
    // voltage value = (ADC 16bit result)/65536 * 1Volt

static XSysMon SysMonInst: //a sysmon instance
static int SysMonFractionToInt(float FloatNum);

int main()
{
    u8 SeqMode;
    u32 TempRawData,VccIntRawData,ExtVolRawData,i;
    float TempData,VccIntData,ExtVolData;
    int xStatus;
    XSysMon_Config *SysMonConfigPtr;
    XSysMon *SysMonInstPtr = &SysMonInst;

    init_platform();
    print("Hello World\n\r");

    //----- SysMon Initialize
    SysMonConfigPtr = XSysMon_LookupConfig(SYSMON_DEVICE_ID);
    if(SysMonConfigPtr == NULL) printf("LookupConfig FAILURE\n\r");

    xStatus = XSysMon_CfgInitialize(SysMonInstPtr, SysMonConfigPtr,SysMonConfigPtr->BaseAddress);
    if(XST_SUCCESS != xStatus) printf("CfgInitialize FAILED\n\r");
    //-----
    XSysMon_GetStatus(SysMonInstPtr); // Clear the old status
    while(1)
    {
        //wait until EOS activated
        while ((XSysMon_GetStatus(SysMonInstPtr) & XSM_SR_EOS_MASK) != XSM_SR_EOS_MASK);

        TempRawData = XSysMon_GetAdcData(SysMonInstPtr, XSM_CH_TEMP);//Read the on-chip Temperature Data
        TempData = XSysMon_RawToTemperature(TempRawData);
        printf("\r\nThe Current Temperature is %0d.%03d Centigrades.\r\n",
            (int)(TempData), SysMonFractionToInt(TempData));
    }
}

```



```

VccIntRawData = XSysMon_GetAdcData(SysMonInstPtr, XSM_CH_VCCINT); //Read the on-chip Vccint Data
VccIntData = XSysMon_RawToVoltage(VccIntRawData);
printf("The Current VCCINT is %0d.%03d Volts. \r\n",
      (int)(VccIntData), SysMonFractionToInt(VccIntData));

ExtVolRawData = XSysMon_GetAdcData(SysMonInstPtr,XSM_CH_VPVN); //Read the external Vpn Data
ExtVolData = XSysMon_RawToExtVoltage(ExtVolRawData);
printf("The Current VpVn is %0d.%03d Volts. \r\n",
      (int)(ExtVolData), SysMonFractionToInt(ExtVolData));

ExtVolRawData = XSysMon_GetAdcData(SysMonInstPtr,XSM_CH_AUX_MIN); //Read the external Vaux0 Data
ExtVolData = XSysMon_RawToExtVoltage(ExtVolRawData);
printf("The Current Vaux0 is %0d.%03d Volts. \r\n",
      (int)(ExtVolData), SysMonFractionToInt(ExtVolData));

ExtVolRawData = XSysMon_GetAdcData(SysMonInstPtr,XSM_CH_AUX_MIN+8);//Read the external Vaux8 Data
ExtVolData = XSysMon_RawToExtVoltage(ExtVolRawData);
printf("The Current Vaux8 is %0d.%03d Volts. \r\n",
      (int)(ExtVolData), SysMonFractionToInt(ExtVolData));

usleep(500000); //wait 500ms

}

return 0;
}


//-----
int SysMonFractionToInt(float FloatNum)
{
    float Temp;

    Temp = FloatNum;
    if (FloatNum < 0) {
        Temp = -(FloatNum);
    }

    return( ((int)((Temp -(float)((int)Temp)) * (1000.0f))));
}
//-----

```


## 7. Run application program

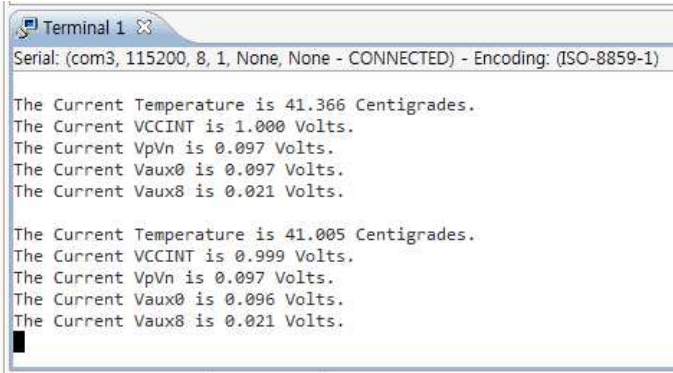
*Turn-on* and *Program* (  ) the ZED board

*Connect* Terminal



*Input* voltages into Vpn, Vaux0, Vaux8 terminals.

*Run* program (  )



*Check* terminal display

---

## 8. Plotting ADC results on the Serial Oscilloscope

*Modify* source program (*separate* Temperature, Vccint, Vpn by comma(,) in printf command) :

```
while(1)
{
    while ((XSysMon_GetStatus(SysMonInstPtr) & XSM_SR_EOS_MASK) != XSM_SR_EOS_MASK); //wait until EOS activated

    TempRawData = XSysMon_GetAdcData(SysMonInstPtr, XSM_CH_TEMP); //Read the on-chip Temperature Data
    TempData = XSysMon_RawToTemperature(TempRawData);

    VccIntRawData = XSysMon_GetAdcData(SysMonInstPtr, XSM_CH_VCCINT); //Read the on-chip Vccint Data
    VccIntData = XSysMon_RawToVoltage(VccIntRawData)*100;

    ExtVolRawData = XSysMon_GetAdcData(SysMonInstPtr, XSM_CH_VPVN); //Read the external Vpn Data
    ExtVolData = XSysMon_RawToExtVoltage(ExtVolRawData)*100;

    //printf("\r\nTemperature: %0d.%03d[deg], VCCINT: %0d.%03d[V], Vpn: %0d.%03d[V]",
    printf("\r\n%0d.%03d,%0d.%03d,%0d.%03d",
        (int)(TempData), SysMonFractionToInt(TempData),
        (int)(VccIntData), SysMonFractionToInt(VccIntData),
        (int)(ExtVolData), SysMonFractionToInt(ExtVolData)
    );

    //usleep(1000); //wait 1ms
```

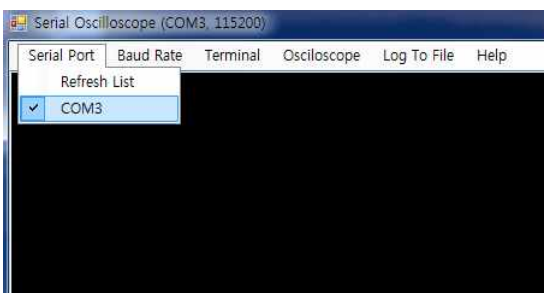
---

*Download* serial oscilloscope program,  Serial-Oscilloscope-v1.5 , *unzip*

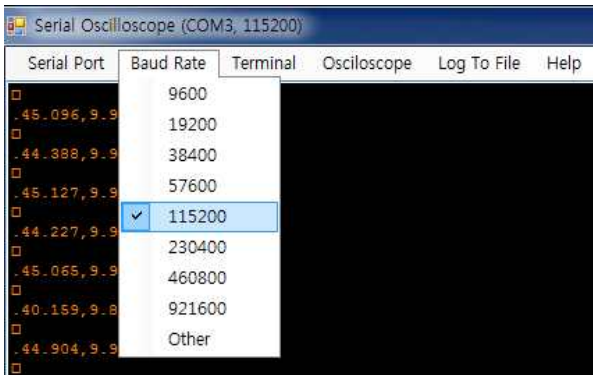
*Open* setting file,  Oscilloscope\_settings

*Change* the OscBeam2Color to yellow(0000FFFF), `OscBeam2Color=0000FFFF` , *Save* file

*Run* application  Serial Oscilloscope



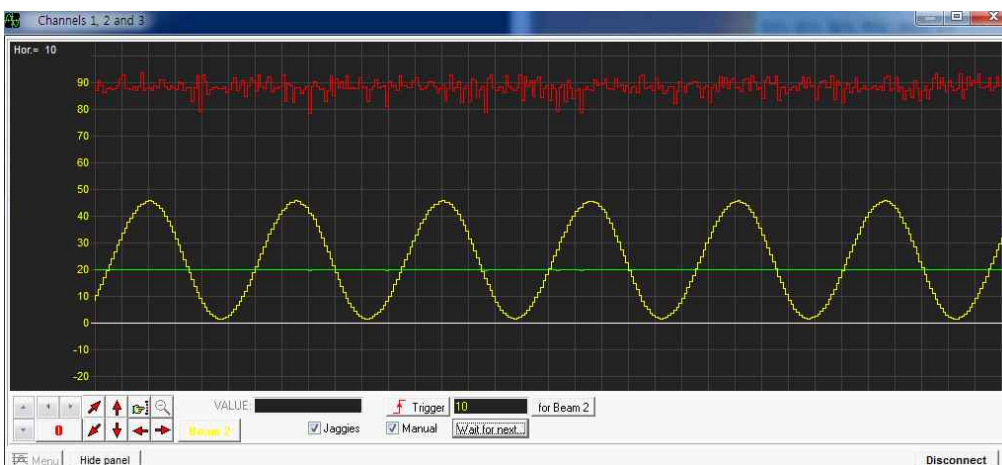
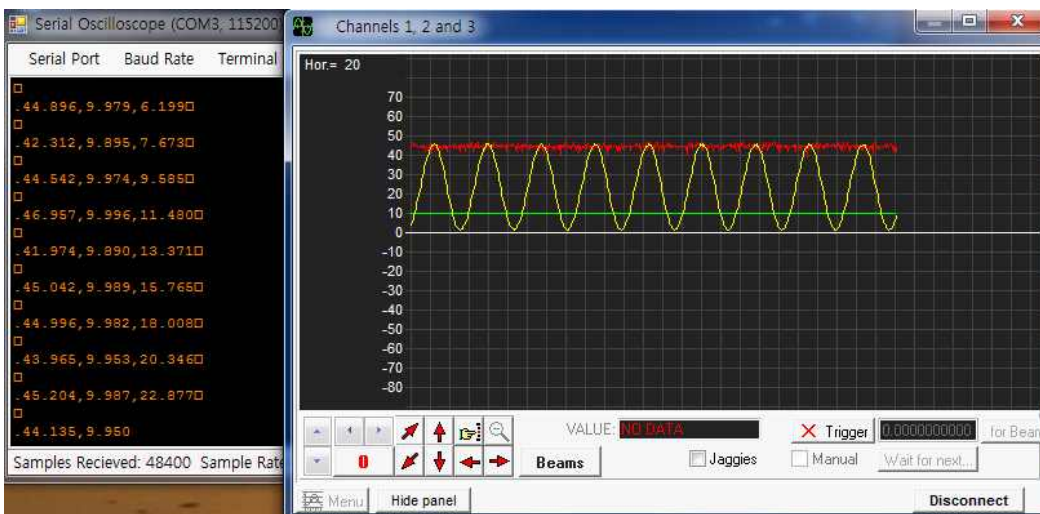
*check* COM3



*select* baud rate



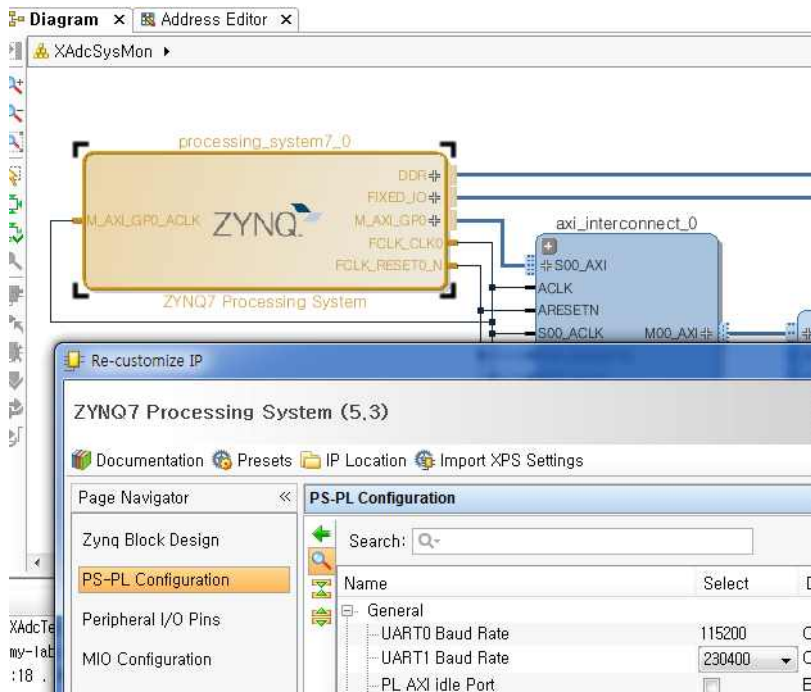
Select Channels 1, 2 and 3



*tune* scope parameters

---

*Appendix. Setting other uart baudrate*



---

*See you next Lab4.*