TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

Vivado + PetaLinux - MPU 9250 / Vivado + SDK - UART

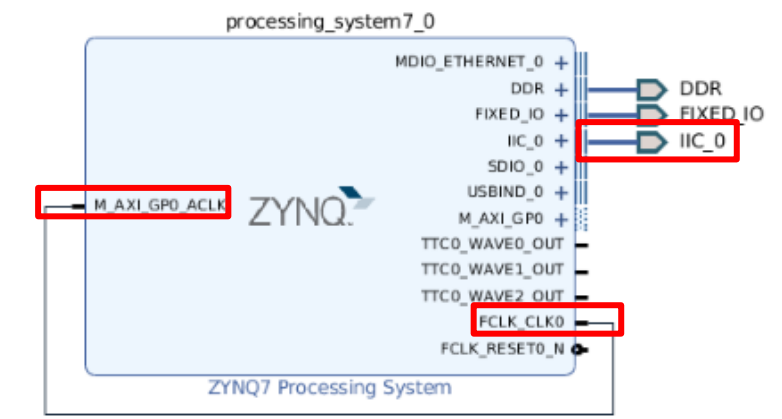
강사 : Innova Lee(이상훈)

강사 메일 : gcccompil3r@gmail.com

학생 : 문지희

학생 메일 : mjh8127@naver.com

Vivado + PetaLinux - MPU 9250



Block design Setting

Pmod JE (Std.)
JE1: V12
JE2: W16
JE3: J15
JE4: H15
JE7: V13
JE8: U17
JE9: T17
JE10: Y17

iic_0_scl_io	INOUT	J15	✓	35	LVCMOS33*
iic_0_sda_io	INOUT	H15	✓	35	LVCMOS33*

SCL - JE3 : J15
SDA - JE4 : H15

I/O Port Setting

Vivado + PetaLinux - MPU 9250

```c
#define I2C_FILE_NAME        "/dev/i2c-0"

int fd =0;

int main(int argc, char argv[])
{
        if((fd =open(I2C_FILE_NAME, O_RDWR)) < 0)
        {
                perror("Open Device Error! \n");
                return -1;
        }
        ioctl_mpu9250(fd);

        wait(10000000);
        printf("Module open Success!! \n");

        int8_t c = readByte(WHO_AM_I_MPU9250, fd);
        printf("I AM = 0x%x \n", c);

        if(c == 0x71)
        {
                ioctl_mpu9250(fd);
                calibrateMPU9250(gyroBias, accelBias, fd);
                printf("MPU9250 calibration Success!!!!!!\n\r");

                initMPU9250(fd);
                printf("MPU9250 Init Success!!!!!!\n\r");

                ioctl_ak8963(fd);
```

device_driver.c

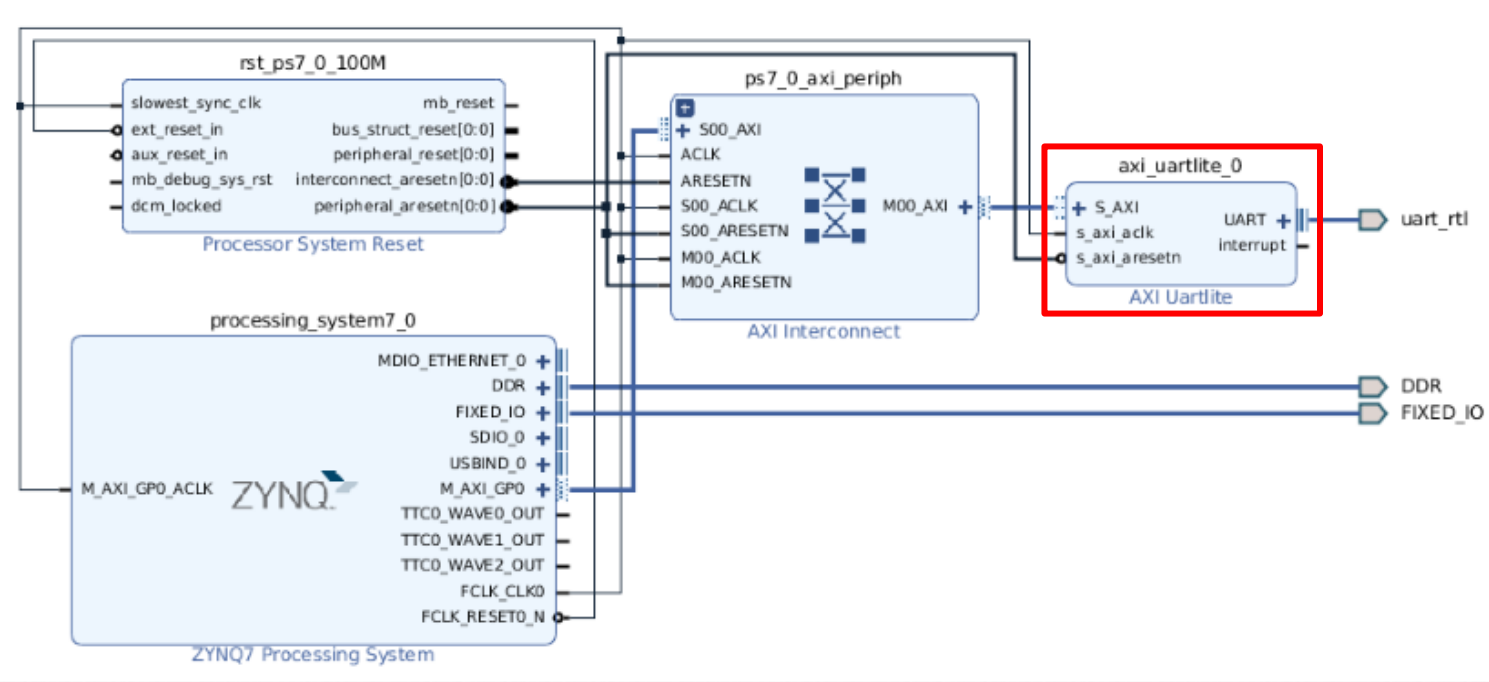
오류발생

```c
int readByte(int8_t regAddr, int fd)
{
        int i;
        int8_t buf[2] = {regAddr,0};
        //int8_t data[1] = {0};
        if(write(fd,&buf[0],1) != 1)
        {
                perror("read register error -readByte - w \n");
                return -1;
        }
        if(read(fd, &buf[1], 1) != 1)
        {
                printf("retrun 0x%x \n", buf[1]);
                perror("read register error -readByte -r  \n");
                return -1;
        }

        return buf[1];

}
```

mpu9250.h



Block Design Setting

Vivado + SDK - UART

Pmod JA (XADC)	Pmod JB (Hi-Speed)	Pmod JC (Hi-Speed)	Pmod JD (Hi-Speed)	Pmod JE (Std.)	Pmod JF (MIO)
JA1: N15	JB1: T20	JC1: V15	JD1: T14	JE1: V12	JF1: MIO-13
JA2: L14	JB2: U20	JC2: W15	JD2: T15	JE2: W16	JF2: MIO-10
JA3: K16	JB3: V20	JC3: T11	JD3: P14	JE3: J15	JF3: MIO-11
JA4: K14	JB4: W20	JC4: T10	JD4: R14	JE4: H15	JF4: MIO-12
JA7: N16	JB7: Y18	JC7: W14	JD7: U14	JE7: V13	JF7: MIO-0
JA8: L15	JB8: Y19	JC8: Y14	JD8: U15	JE8: U17	JF8: MIO-9
JA9: J16	JB9: W18	JC9: T12	JD9: V17	JE9: T17	JF9: MIO-14
JA10: J14	JB10: W19	JC10: U12	JD10: V18	JE10: Y17	JF10: MIO-15

uart_rtl_rxd	IN	V12	✓	34	LVCMOS33*
uart_rtl_txd	OUT	W16	✓	34	LVCMOS33*

I/O Port Setting

Standard Pmod

200ohm 저항과 zynq의 PL에 연결되어 쇼트를 방지한다. 데이터의 최대 스위칭 속도를 제한할 수 있다.(고속 액세스를 제한)

MIO Pmod

200ohm 저항과 PS의 MIO버스와 연결되어 Std Pmod와 같은 특징이 있다.
PS장치 컨트롤러 코어에서만 액세스 할 수 있고 GPIO, UART, I2C, SPI 코어를 사용 가능하다.
(But, Uart와 I2C는 일반 pmod핀 배치가 일치하지 않아 외부 핀으로 교체해야할 수 있다.)

XADC(Dual Analog/Digital) Pmod

아날로그 신호를 ADC에 입력할 수 있다.

High-Speed Pmod

쇼트 보호기능이 없다. High speed differential signaling(고속 차동 신호)가 필요할 때에만 사용한다.

Vivado + SDK - UART

```c
#include "xparameters.h"
#include "xuartps.h"
#include "xil_printf.h"

#define UART_DEVICE_ID                  XPAR_XUARTPS_0_DEVICE_ID

XUartPs Uart_Ps;

int UartPsHelloWorldExample(u16 DeviceId);
void wait(int delay);

int main(void)
{
    int Status;

    while(1)
    {
        Status = UartPsHelloWorldExample(UART_DEVICE_ID);
    }
}
```

```c
int UartPsHelloWorldExample(u16 DeviceId)
{
    u8 HelloWorld[] = "Hello World\n\r";
    int SentCount = 0;
    int Status;
    XUartPs_Config *Config;

    Config = XUartPs_LookupConfig(DeviceId);
    if (NULL == Config) {
        return XST_FAILURE;
    }

    Status = XUartPs_CfgInitialize(&Uart_Ps, Config, Config->BaseAddress);

    if (Status != XST_SUCCESS) {
        return XST_FAILURE;
    }

    XUartPs_SetBaudRate(&Uart_Ps, 115200);

    while (SentCount < (sizeof(HelloWorld) - 1)) {
        /* Transmit the data */
        SentCount += XUartPs_Send(&Uart_Ps,
                        &HelloWorld[SentCount], 1);
    }

    return SentCount;
}
```

```
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
```

/dev/ttyUL 장치파일을 만들기 위해 한 일들 목록

1. 헤더파일 수정
open함수에서 O_CREAT 설정 추가
-> 생성x

2. 디바이스 트리 수정
system_top.dts에서
&axi_uartlilte_0{
    compatible="generic-ttyUL"
};
추가 -> 생성x

3. 헤더파일 수정 + 디바이스 트리 수정
system_top.dts에서
&axi_uartlilte_0{
compatible="generic-uio"
};
헤더파일에서 장치이름을 "dev/uio-0" 으로 수정
-> uio-0파일은 생성되지만 file open 실패

4. Putty 내에서 mknod로 장치파일 생성
-> 만들어지지만 소스코드 실행했을 때 file open 실패

5. config.project 파일 수정
http://www.wiki.xilinx.com/Uartlite%20Driver 참조

6. config.project 파일 수정 + 디바이스 트리 수정
system_config.dtsi 에서 uartlite_0@42C00000{ ... } 추가