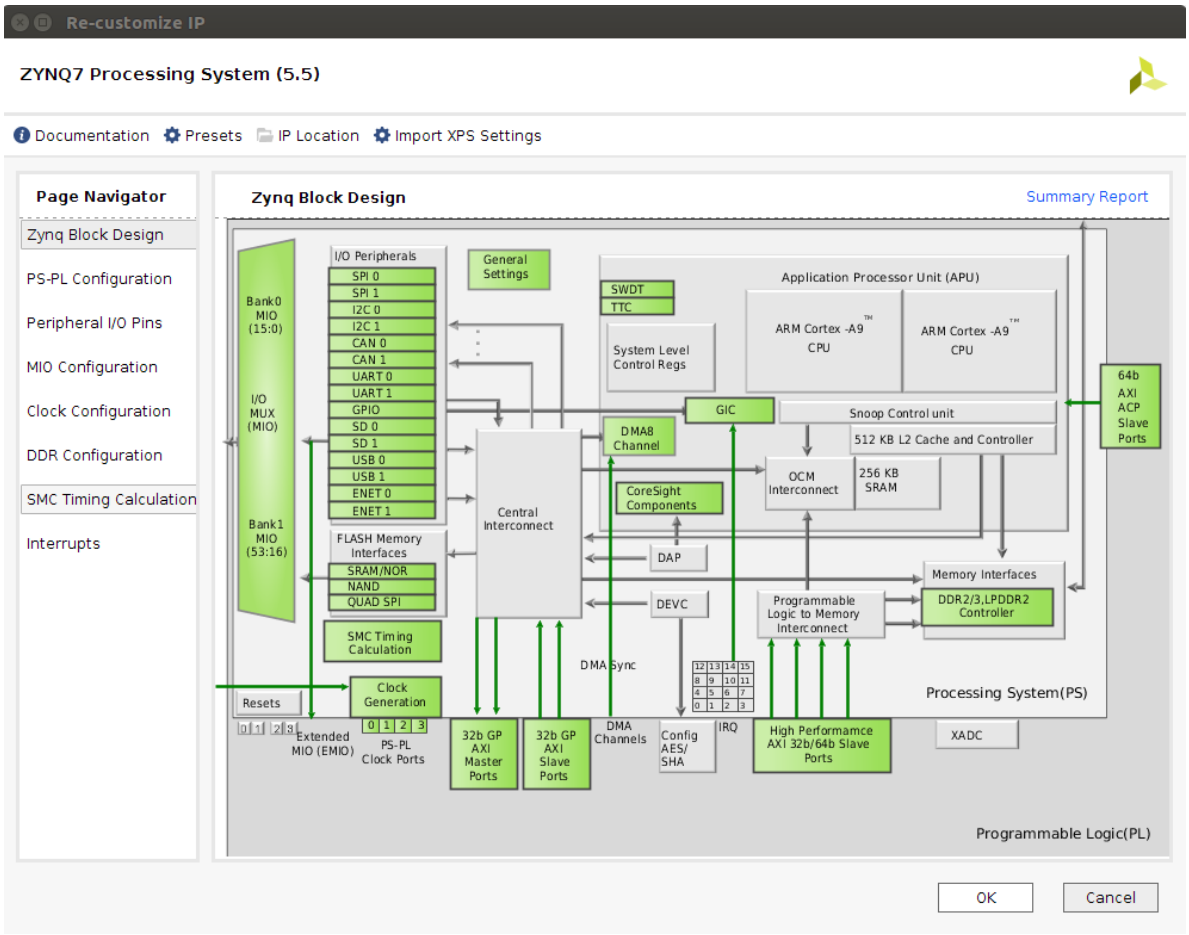
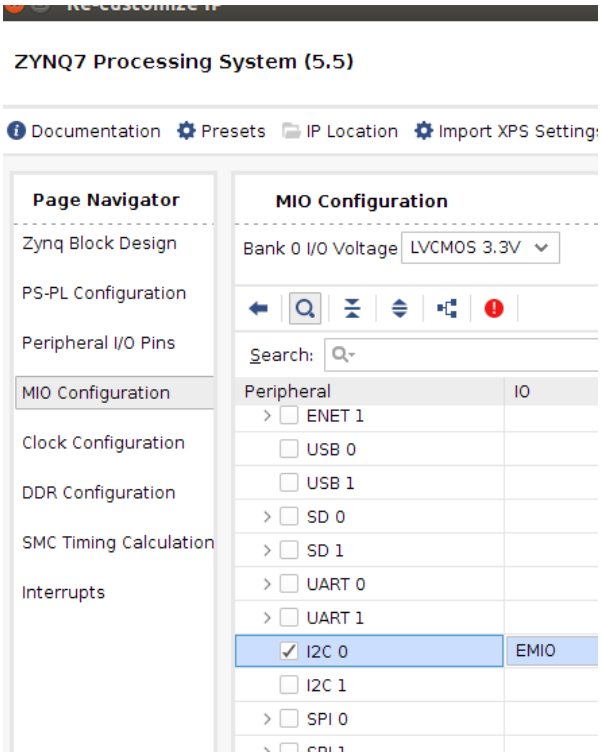
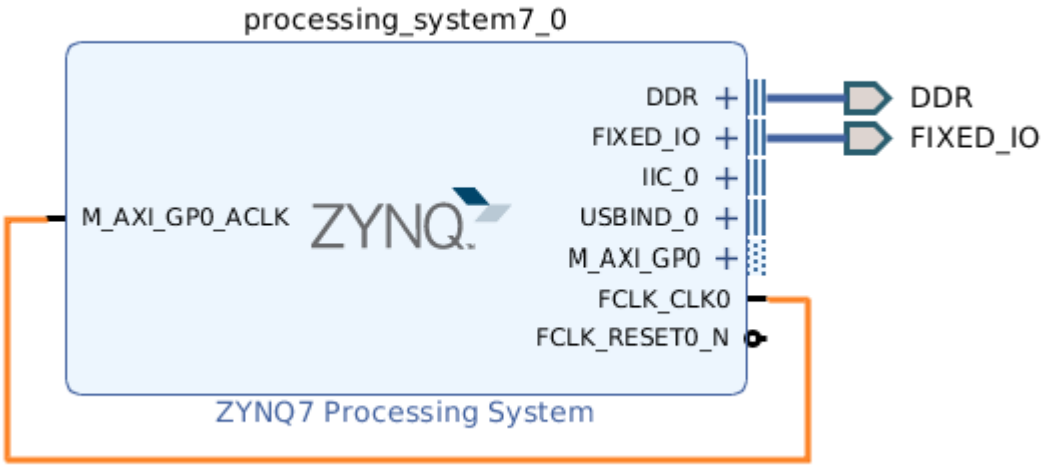
New project (project 명 : mpu6050_hw , 경로 : home/xilinx/
→ Create block → zynq → block 더블클릭



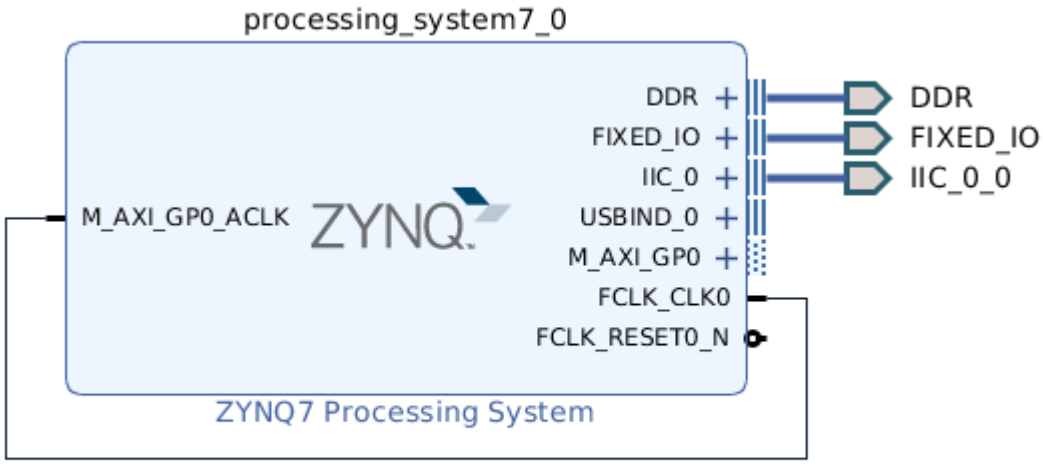


run block autumation

수동으로 클락 걸어줘야함. 아래처럼 마우스 드래그 선 잇는다.



Ii2_0 핀에서 우클릭 - make external



design source → create HDL wrapper

run synthesis

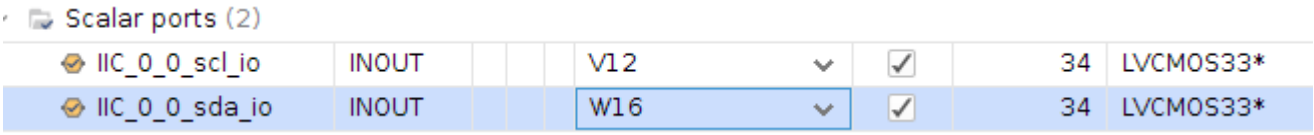
run implementation

여기서 generate 하면 안되고 open implemented 해야 함. 핀 매핑 해야 되서

I/o planning 으로 간다.

이제 레퍼런스 매뉴얼을 참조해야한다.
https://reference.digilentinc.com/reference/programmable-logic/zybo-z7/reference-manual

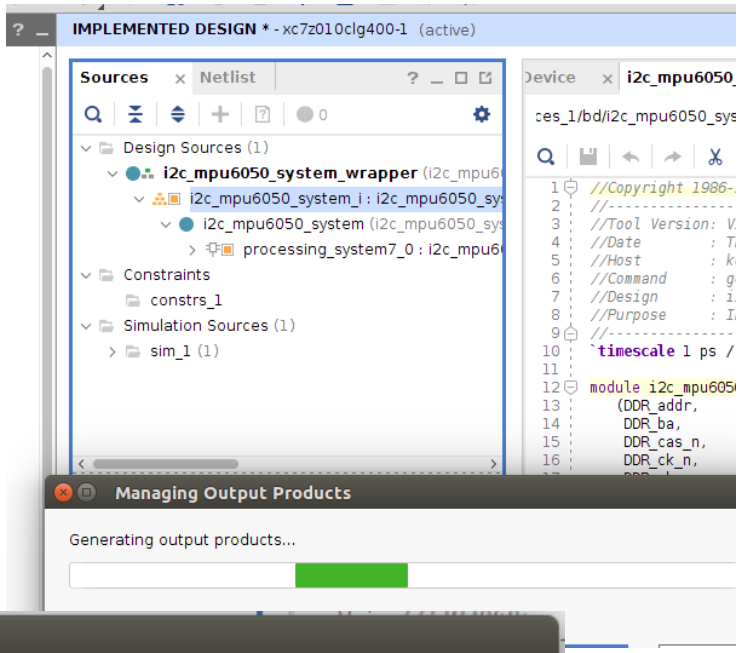
맵에 T19 없다. 사용할수없음. 바궈주자.

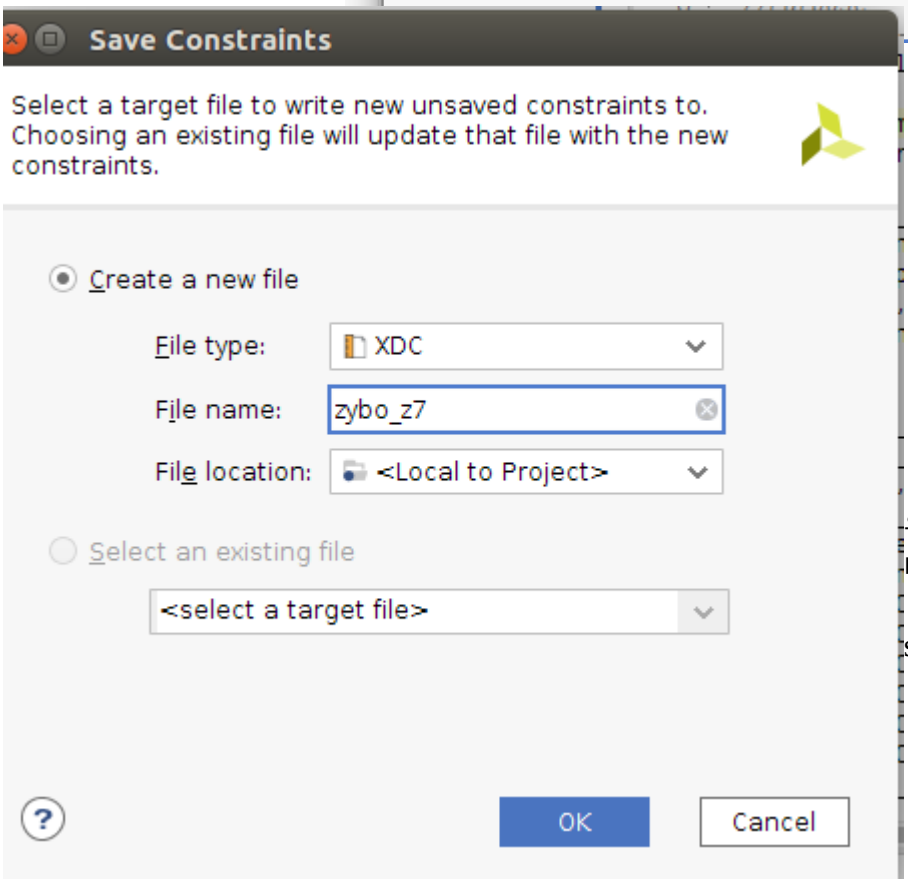


맵에 사용 가능한 핀 두개 쓰자. pin 1,2 v12 W16 로 각각 바꿔준다.

둘다 +LVCMOS33 으로바꿔준다.

SOURCES 로 가서,



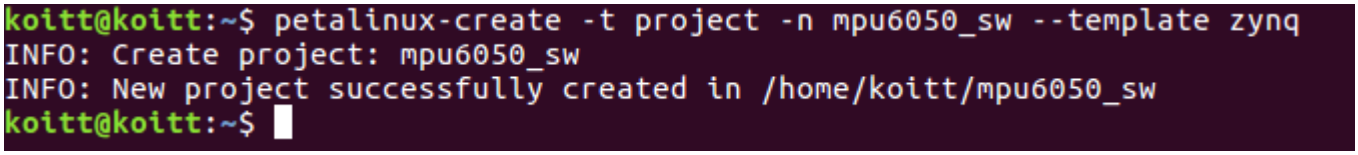


우클릭해서 generate output – generate 한다.

save 한다.

어차피 implementation 까지 똑같으니까 synthesis 만 하지말고 run implementation 누른다.
Ok ok 하고, 그다음, generate – bitstream 누른다. 오른쪽 위 작게 동글뱅이 기다린다.
끝나면, ok 누르지말고 창 닫는다.
그리고 File – export- export hardware – include bitstream 하고 ok 한다.

(리눅스 올릴거면 , sdk 안 쓰고)
(리눅스안올리면 sdk 쓴다. 그래서 지금은 sdk 안 켬.)



petalinux-create -t project -n mpu6050_sw –template zynq

한 다음에 ~/xilinx/mpu6050_sw/ 폴더안에 들어와서

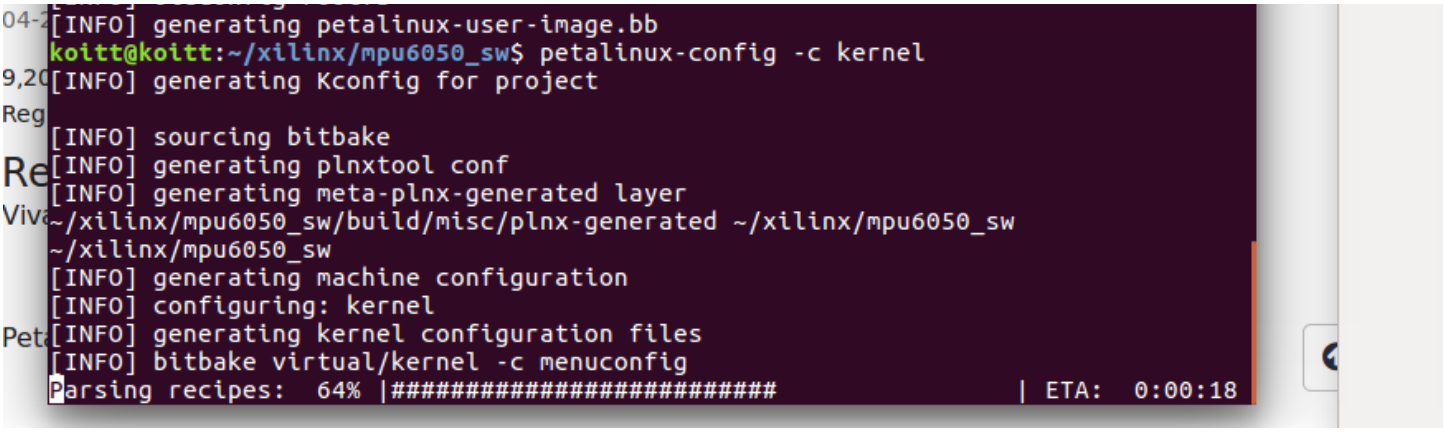
petalinux-config --get-hw-description=~/xilinx/mpu6050_hw_design/mpu6050_hw_design.sdk

하면, 파란 창 넘어간다.

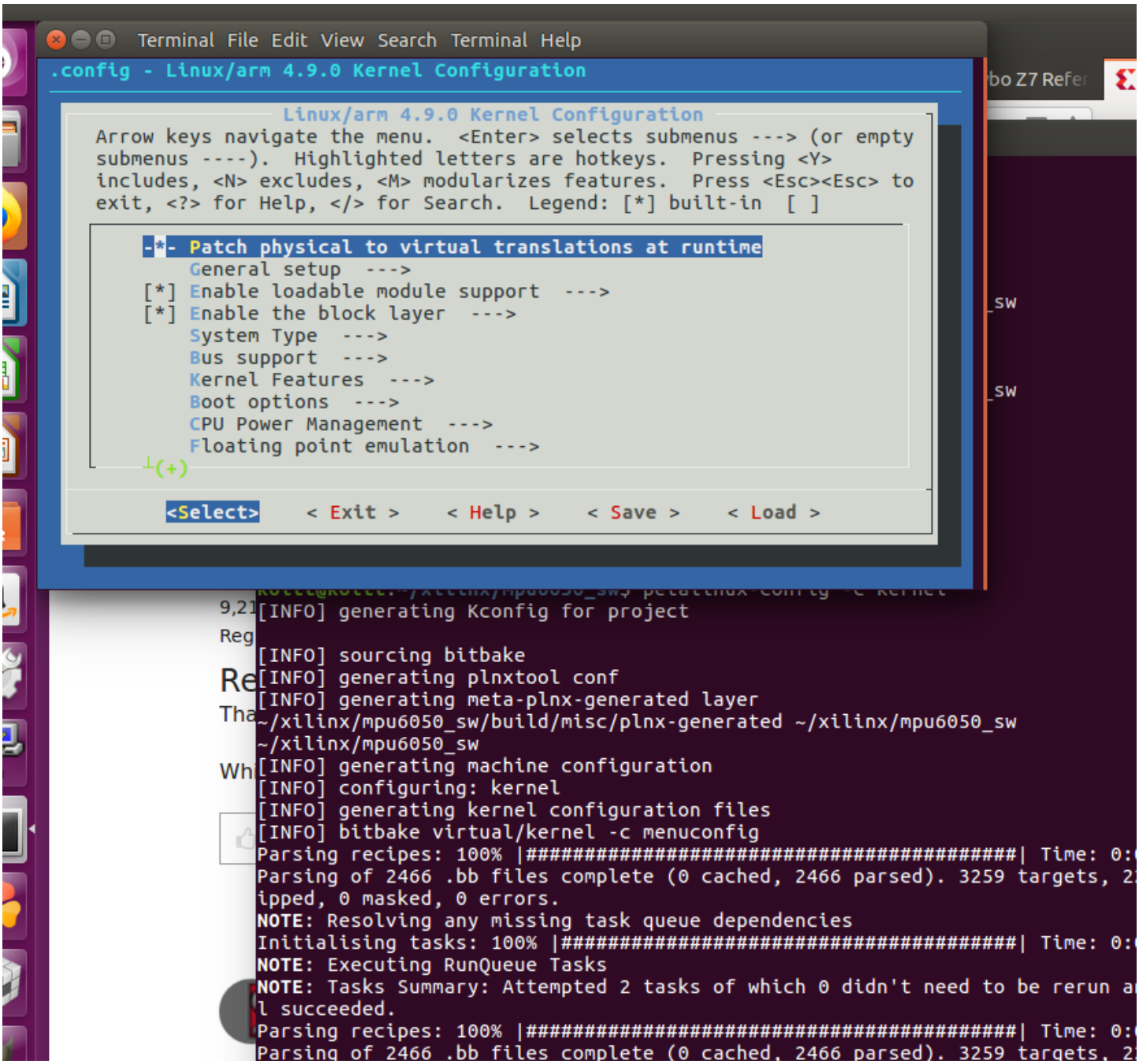
Subsystem auto - adv - uboot
부터 아래로 순서대로 다 sd card 로 바 꾸고 맨앞에와서, Image packaging config - root filesystem sype sd card 바꾸고 저장.

다하고 save 하고 나온다, 기다린다..

그다음, petalinux-config -c kernel 하면 아래처럼 진행된다.



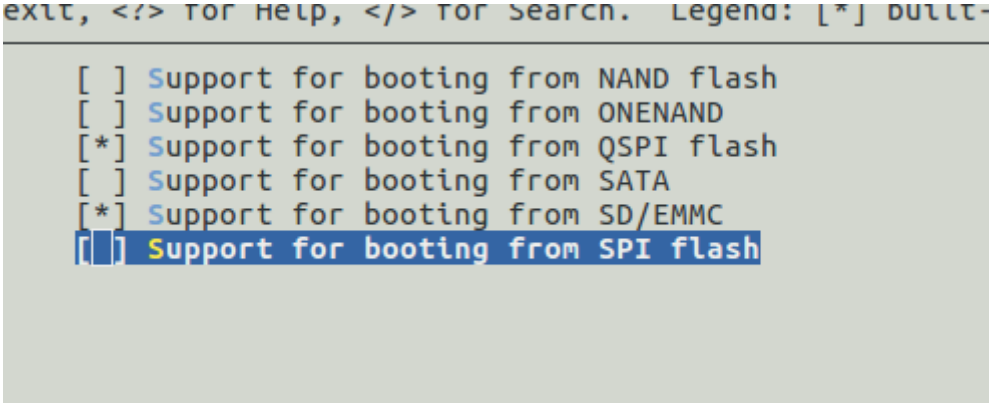
진행되는것 다 되면, 창 새로 뜨면서, 파란창 추가로 아래처럼 뜬다.


Terminal File Edit View Search Terminal Help
.config - Linux/arm 4.9.0 Kernel Configuration
Linux/arm 4.9.0 Kernel Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]
-*- Patch physical to virtual translations at runtime
General setup --->
[*] Enable loadable module support --->
[*] Enable the block layer --->
System Type --->
Bus support --->
Kernel Features --->
Boot options --->
CPU Power Management --->
Floating point emulation --->
<Select> < Exit > < Help > < Save > < Load >
[INFO] generating Kconfig for project
[INFO] sourcing bitbake
[INFO] generating plnxtool conf
[INFO] generating meta-plnx-generated layer
~/xilinx/mpu6050_sw/build/misc/plnx-generated ~/xilinx/mpu6050_sw
~/xilinx/mpu6050_sw
[INFO] generating machine configuration
[INFO] configuring: kernel
[INFO] generating kernel configuration files
[INFO] bitbake virtual/kernel -c menuconfig
Parsing recipes: 100% |###| Time: 0:
Parsing of 2466 .bb files complete (0 cached, 2466 parsed). 3259 targets, 2
ipped, 0 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies
Initialising tasks: 100% |######################################| Time: 0:
NOTE: Executing RunQueue Tasks
NOTE: Tasks Summary: Attempted 2 tasks of which 0 didn't need to be rerun a
l succeeded.
Parsing recipes: 100% |###| Time: 0:
Parsing of 2466 .bb files complete (0 cached, 2466 parsed). 3259 targets, 2

파란 창 뜬거에서,
device drivers > i2c support > i2c hardware bus support > i2c cadence 에 * 있나 확인, save exit

다시, petalinux-config -c u-boot

boot media – qspi sd/emmc 선택, (스페이스바로선택). Save 나오자



그다음,

```
koitt@koitt:~/xilinx/mpu6050_sw$ petalinux-create -t apps -n mpu6050-app --enable
INFO: Create apps: mpu6050-app
INFO: New apps successfully created in /home/koitt/xilinx/mpu6050_sw/project-spec/meta-user/recipes-apps/mpu6050-app
INFO: Enabling created component...
INFO: sourcing bitbake
INFO: oldconfig rootfs
INFO: mpu6050-app has been enabled
```

```
koitt@koitt:~/xilinx/mpu6050_sw$ cd project-spec/meta-user/recipes-app/mpu6050-app/
bash: cd: project-spec/meta-user/recipes-app/mpu6050-app/: No such file or directory
koitt@koitt:~/xilinx/mpu6050_sw$ cd project-spec/meta-user/recipes-apps/mpu6050-app/
koitt@koitt:~/xilinx/mpu6050_sw/project-spec/meta-user/recipes-apps/mpu6050-app$ ls
files  mpu6050-app.bb  README
koitt@koitt:~/xilinx/mpu6050_sw/project-spec/meta-user/recipes-apps/mpu6050-app$ vi mpu6050-app.bb
koitt@koitt:~/xilinx/mpu6050_sw/project-spec/meta-user/recipes-apps/mpu6050-app$
```

```
Terminal File Edit View Search Terminal Help
#
# This file is the mpu6050-app recipe.
#

SUMMARY = "Simple mpu6050-app application"
SECTION = "PETALINUX/apps"
LICENSE = "MIT"
LIC_FILES_CHKSUM = "file://${COMMON_LICENSE_DIR}/MIT;md5=0835ade698e0bcf8506ecda2f7b4
f302"

SRC_URI = "file://mpu6050-app.c \
       file://Makefile \
          "

S = "${WORKDIR}"

do_compile() {
         oe_runmake
}

do_install() {
         install -d ${D}${bindir}
         install -m 0755 mpu6050-app ${D}${bindir}
}
~
```

```
koitt@koitt:~/xilinx/mpu6050_sw/project-spec/meta-user/recipes-apps/mpu6050-app/files
$ ls
Makefile  mpu6050-app.c
koitt@koitt:~/xilinx/mpu6050_sw/project-spec/meta-user/recipes-apps/mpu6050-app/files
$ vi mpu6050-app.c
```

이제 커널에서 본 makefile, ~~.c 파일 두개 보인다.

mpu6050-app.c
열어보면,

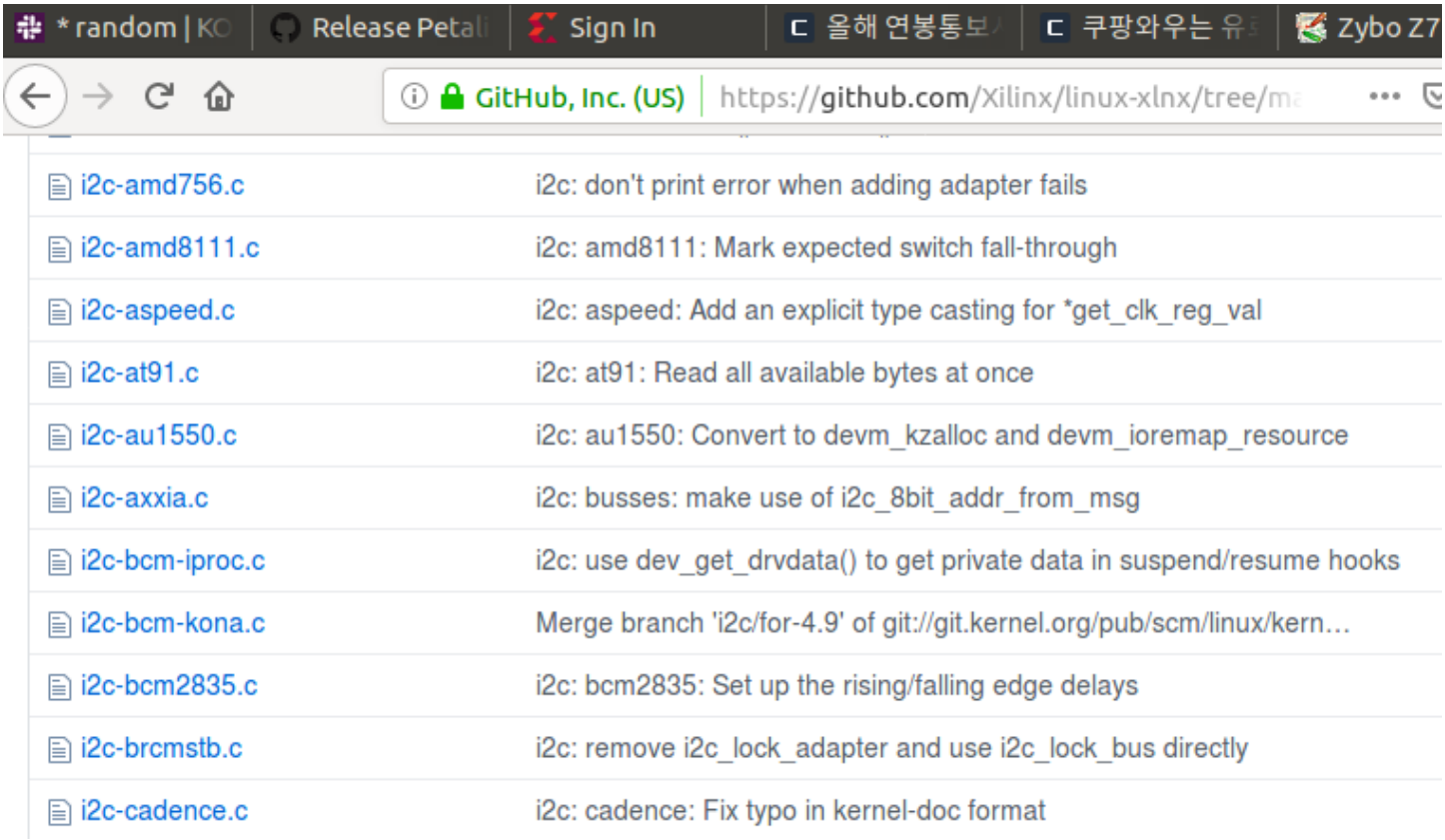
```
#include <stdio.h>

int main(int argc, char **argv)
{
    printf("Hello World!\n");

    return 0;
}
```

hello world 만 들어있다. 여기에 이제 여기에 i2c 드라이버 짜야 함.

Xilinx driver code 보려면 깃 가야함. 깃 xilinx
https://github.com/Xilinx/linux-xlnx/tree/master/drivers/i2c

<참고 파일>
https://github.com/Digilent/vivado-boards/archive/master.zip?_ga=2.107667584.3089703.1553043368-2028910606.1543203947
https://github.com/Digilent/Petalinux-Zybo-Z7-10/releases?_ga=2.51880425.1762548192.1552140278-1400676129.1550250933

linux-xlnx/drivers/i2c/busses/ i2c-cadence.c 이거에대한 커널 코드를 싹 다 분석해야 드라이버 코드 짤 수 있다.



내용 잘모르겠다이러면, 최상단에, documentation – driver api -i2c 가 올라와있는게있다. 여긴 별 내용이 없지만..
documentation -i2c/i2c-stub 여기에 팁들이있다.

I2c 드라이버 코드

```
#include <stdio.h>
#include <errno.h>
#include <fcntl.h>
#include <stdint.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <linux/i2c.h>
#include <linux/i2c-dev.h>
#define I2C_FILE_NAME   "/dev/i2c-0" //vivodo 에서 설정한 것
#define X_H                0
#define X_L                1
#define Y_H                2
#define Y_L                3
#define Z_H                4
#define Z_L                5
#define ACCEL_CONFIG       0x1c
#define INI_STATUS         0x3A
#define ACCEL_XOUT_H       0x3B
#define ACCEL_XOUT_L       0x3C
#define ACCEL_YOUT_H       0x3D
#define ACCEL_YOUT_L       0x3E
#define ACCEL_ZOUT_H       0x3F
#define ACCEL_ZOUT_L       0x40
#define TEMP_OUT_H         0x41
#define TEMP_OUT_L         0x42
#define GYRO_XOUT_H        0x43
#define GYRO_XOUT_L        0x44
#define GYRO_YOUT_H        0x45
#define GYRO_YOUT_L        0x46
#define GYRO_ZOUT_H        0x47
#define GYRO_ZOUT_L        0x48
#define I2C_SLV0_D0        0X63
#define I2C_SLV1_D0        0X64
#define I2C_SLV2_D0        0X65
#define I2c_SLV3_D0        0X66
#define PWR_MGMT_1         0x6B
#define FIFO_COUNTH        0x72
#define FIFO_COUNTL        0x73
#define WHO_AM_I           0x75
#define MPU6050_ADDR       0x68
#define AF_SEL_ACCEL_RANGE_2G   0 << 3
#define AF_SEL_ACCEL_RANGE_4G   1 << 3
#define AF_SEL_ACCEL_RANGE_8G   2 << 3
#define AF_SEL_ACCEL_RANGE_16G  3 << 3

int mpu6050_init(int addr)
{
```

```
    int fd;
    if((fd = open(I2C_FILE_NAME, O_RDWR))<0)
    {
        perror("Failed to open I2C Bus\n");
        return -1;
    }
    //slave 로 구동시킬 것으로 명시
    if(ioctl(fd, I2C_SLAVE, MPU6050_ADDR) < 0)
    {
        perror("Failed to acquire Bus Access\n");
        return -1;
    }
    return fd;
}

inline void mpu6050_deinit(int fd)
{
    close(fd);
}

void mpu6050_write_reg(int fd, uint8_t reg, uint8_t val)
{
    uint8_t data[2];
    data[0] = reg;
    data[1] = val;
    if(write(fd, data, 2)!=2)
        perror("mpu6050 write error!");
}
uint16_t mpu6050_select_range(int fd, uint16_t range)
{
    int sense;
    mpu6050_write_reg(fd, ACCEL_CONFIG, range);
    switch(range)
    {
        case AF_SEL_ACCEL_RANGE_2G:
            sense = 0x4000;
            break;
        case AF_SEL_ACCEL_RANGE_4G:
            sense = 0x2000;
            break;
        case AF_SEL_ACCEL_RANGE_8G:
            sense = 0x1000;
            break;
        case AF_SEL_ACCEL_RANGE_16G:
            sense = 0x0800;
            break;
    }
    return sense;
}

uint8_t mpu6050_read_reg(int fd, uint8_t reg)
{
    unsigned char data[3];
    if(write(fd, data, 1)!=1)
    {
        perror("Error Sending read request via i2c\n");
```

```
        return -1;
    }
    if(read(fd, data, 1)!=1)
    {
        perror("Error Getting read request via i2c\n");
    }
    return data[0];
}

inline void mpu6050_power_on(int fd)
{
    mpu6050_write_reg(fd, PWR_MGMT_1, 0x0);
}

int16_t mpu6050_read_reg_pair(int fd, uint8_t reg)
{
    uint8_t data[3];
    data[0] = reg;
    if(write(fd, data, 1)!=1)
   {
        perror("Error Sending read request via i2c\n");
        return -1;
    }

    if(read(fd, data, 2)!=2)
    {
        perror("Error Getting read request via i2c\n");
        return -1;
    }

    return (data[0] << 8) | data[1];
}

inline float mpu6050_get_temp(int fd)
{
    return (float)(mpu6050_read_reg_pair(fd, TEMP_OUT_H) / 340 + 36.53);
}

int main(int argc, char **argv)
{
    float temper, acc_x, acc_y, acc_z;
    int16_t gyro_x, gyro_y, gyro_z;
    int fd, whoami = 0, sense;
    if((fd = mpu6050_init(MPU6050_ADDR))<0)
    {
        printf("Can't Init MPU6050\n");
        return -1;
    }

    sense = mpu6050_select_range(fd, AF_SEL_ACCEL_RANGE_16G);
    mpu6050_power_on(fd);
    whoami = mpu6050_read_reg(fd, WHO_AM_I);
    printf("Sensor ID: 0x%x\n", whoami);
    for(;;)
    {
        temper = mpu6050_get_temp(fd);
```

```
        acc_x = ((float)(mpu6050_read_reg_pair(fd, ACCEL_XOUT_H)))/sense;
        acc_y = ((float)(mpu6050_read_reg_pair(fd, ACCEL_YOUT_H)))/sense;
        acc_z = ((float)(mpu6050_read_reg_pair(fd, ACCEL_ZOUT_H)))/sense;
        printf("Temper:%.3f\tacc(x):%.3f\tacc(y):%.3f\tacc(z):%.3f\n", temper, acc_x, acc_y, acc_z);
        usleep(200000);
    }
    mpu6050_deinit(fd);
    return 0;
}
```

하고 나와서 **petalinux-build**

하는데 만약 여기서 에러 나면 다시 돌아가서 처음부터해야한다.
Vivado 했던거 다음 petalinux create 부분.

최악의경우는 vivado project 경로 설정이 잘못되었을수도있음 → 파일을 지우고, 처음부터 다시 시작.

---

petalinux-build 성공 시 (error 없이)

cd images/linux/ 온다.

```
koitt@koitt:~/xilinx/mpu6050_sw$ ls
build  components  config.project  images  project-spec
koitt@koitt:~/xilinx/mpu6050_sw$ cd images/linux
koitt@koitt:~/xilinx/mpu6050_sw/images/linux$
```

petalinux-package --boot --force --fsbl zynq_fsbl.elf --fpga i2c_mpu6050_wrapper.bit --u-boot

```
koitt@koitt:~/xilinx/mpu6050_sw/images/linux$ petalinux-package --boot --force --fsbl zynq_
fsbl.elf --fpga i2c_mpu6050_wrapper.bit --u-boot
INFO: File in BOOT BIN: "/home/koitt/xilinx/mpu6050_sw/images/linux/zynq_fsbl.elf"
INFO: File in BOOT BIN: "/home/koitt/xilinx/mpu6050_sw/images/linux/i2c_mpu6050_wrapper.bit
"
INFO: File in BOOT BIN: "/home/koitt/xilinx/mpu6050_sw/images/linux/u-boot.elf"
INFO: Generating zynq binary package BOOT.BIN...
INFO: Binary is ready.
WARNING: Unable to access the TFTPBOOT folder /tftpboot!!!
WARNING: Skip file copy to TFTPBOOT folder!!!
```

일단 다음 넘어와서,

sd 카드 파티션 설정 다 한다음에, (how_to_boot_zybo_with_sd_card.pdf)

BOOTS 폴더에는 3 가지 파일 복사해야한다.
BOOT.BIN image.ub system.dtb

```
koitt@koitt:~/xilinx/mpu6050_sw/images/linux$ sudo cp rootfs.cpio /media/koitt/rootfs
[sudo] password for koitt:
koitt@koitt:~/xilinx/mpu6050_sw/images/linux$
```

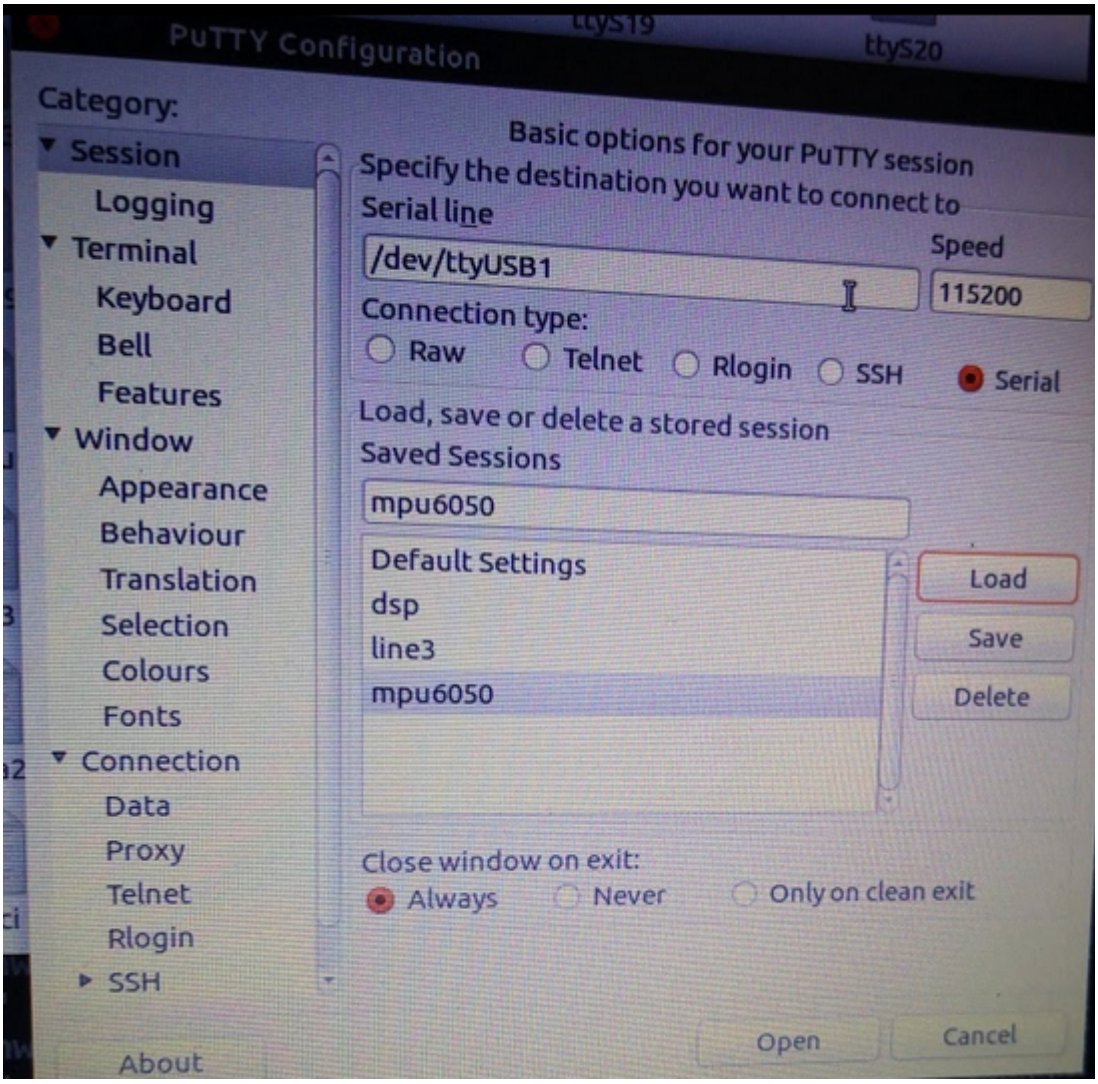
folder rootfs 에 들어가서, rootfs.cpio 를 rootfs 폴더에 복사해서 **sudo pax -rvf rootfs.cpio** 한다.
(pat 없으면 sudo apt-get install pat 해서 설치한다)

그러면, bin/home 등 쫙 폴더 생성된다.

```
pax: sv4cpio vol 1, 954 files, 11972608 bytes read, 0 bytes written.
koitt@koitt:/media/koitt/rootfs$ ls
bin   dev  home  lib         media  proc        run   sys  usr
boot  etc  init  lost+found  mnt    rootfs.cpio  sbin  tmp  var
koitt@koitt:/media/koitt/rootfs$
```

Xilinx z7 보드 부팅 점퍼 - > sd 로 해주고 카드끼우고 전원넣으면 부팅성공.

Sudo chmod 777 /dev/ttyUSB1 권한 설정 준다.
Putty 설정은 아래와 같다.



putty 로 연결 후,

엔터 치면, id pw 입력 뜬다. → root root

이제 통신창에 값이 쫙 뜬다.