Device Tree

1) device 는 Node 로 표현하며 각각의 node 는 다양한 속성 정보를 갖는다.

2) node@"숫자" 노드 뒤에 숫자는 Unit Address 로 장치에 접근하기 위해 사용되는 1 차주소.

3) node 앞 부분에 alias 를 붙일 수 있으며 다른 node 에서는 주로 이 alias 를 활용하여 node 를 참조

4) compatible 이란 속성을 이용하여 우리가 표현 하고자 하는 보드에 대하여 기술하고 있다.

5) copatible 속성은 항상 "제조사, 모델" 순으로 문자열 형태로 표현한다.


Label
Node Name
Unit Address
속성 이름
Devicetree
axi_gpio_0: gpio@40000000 {
#gpio-cells = <2>;
compatible = "xlnx,xps-gpio-1.00.a";
속성값
gpio-controller ;
interrupt-parent = <&&microblaze_0_intc>;
interrupts = < 6 2 >;
reg = < 0x40000000 0x10000 >;
xlnx,all-inputs = <0x0>;
xlnx,all-inputs-2 = <0x0>;
xlnx,dout-default = <0x0>;
xlnx,dout-default-2 = <0x0>;
xlnx,gpio-width = <0x2>;
xlnx,gpio2-width = <0x2>;
xlnx,interrupt-present = <0x1>;
xlnx,is-dual = <0x1>;
xlnx,tri-default = <0xffffffff>;
xlnx,tri-default-2 = <0xffffffff>;
} ;

Kernel Configuration Options for Driver

To enable GPIO in the kernel, the following configuration options need to be enabled:

```
CONFIG_GPIO_SYSFS=y
CONFIG_SYSFS=y
CONFIG_GPIO_XILINX=y (for axi_gpio)
```

커널 구성

project-spec/meta-plnx-generated/recipes-kernel/linux/configs/plnx_kernel.cfg 여기에 삽입

(확실치 않음)

Expected Output

```
Note from the boot log what the mappings of the 2 AXI GPIO units are :
[    1.354448] XGpio: /amba_pl@0/gpio@80000000: registered, base is 504
[    1.354761] XGpio: /amba_pl@0/gpio@80010000: registered, base is 496
The AXI GPIO driving the LEDs is at 0x80000000 so its base is 504.
The AXI GPIO reading the DIP switches is at 0x80010000 so its base is 496.
Note the nodes in place :
root@plnx_aarch64:/sys/class/gpio# ls /sys/class/gpio
export       gpiochip306  gpiochip496  unexport
gpiochip290  gpiochip322  gpiochip504
Activate LEDs using sysfs
Each created node controls a single bit of GPIO
root@plnx_aarch64:~# echo 504 > /sys/class/gpio/export
root@plnx_aarch64:/sys/class/gpio# ls /sys/class/gpio
export       gpiochip290  gpiochip322  gpiochip504
gpio504      gpiochip306  gpiochip496  unexport
root@plnx_aarch64:~# echo out > /sys/class/gpio/gpio504/direction
root@plnx_aarch64:~# echo 1 > /sys/class/gpio/gpio504/value - note the respective LED lights
root@plnx_aarch64:~# echo 505 > /sys/class/gpio/export
root@plnx_aarch64:~# echo out > /sys/class/gpio/gpio505/direction
root@plnx_aarch64:~# echo 1 > /sys/class/gpio/gpio505/value - note the respective LED lights
Repeat for 506 - 511
Read DIP Switches using sysfs
Each created node controls a single bit of GPIO
root@plnx_aarch64:~# echo 496 > /sys/class/gpio/export
root@plnx_aarch64:~# echo in > /sys/class/gpio/gpio496/direction
root@plnx_aarch64:~# cat /sys/class/gpio/gpio496/value - try each switch position
```

GPIO 제어 (sysfs 방식)
시도하다 포기

Device Driver 소스 분석

```c
/* 버스에 디바이스 추가 */
int bus_add_device(struct device *dev)
{
    struct bus_type *bus = bus_get(dev->bus);
    int error = 0;
    /* 버스에 추가될 디바이스인 경우 해당 버스에 소속된 디바이스 속성들 추가
        divier_probe, driver_autoprobe 속성이 추가 */
    if (bus) {
        pr_debug("bus: '%s': add device %s\n", bus->name, dev_name(dev));
        error = device_add_attrs(bus, dev);
        if (error)
            goto out_put;
        error = device_add_groups(dev, bus->dev_groups);
        if (error)
            goto out_id;
    /* 해당 버스의 디바이스 디렉토리에 디바이스명으로 심볼링크를 생성해 해당
        error = sysfs_create_link(&bus->p->devices_kset->kobj,
                        &dev->kobj, dev_name(dev));
        if (error)
            goto out_groups;
    /* 디바이스 디렉토리에 "subsystem"심볼링크를 생성하여 해당 버스 디렉토리
        error = sysfs_create_link(&dev->kobj,
                &dev->bus->p->subsys.kobj, "subsystem");
        if (error)
            goto out_subsys;
    /* 버스가 관리하는 디바이스 리스트에 디바이스를 추가  */
        klist_add_tail(&dev->p->knode_bus, &bus->p->klist_devices);
    }
    return 0;
```