

PetaLinux Development with the Custom VIVADO Project

[LED Controller Project]

Document Version: V1.0, Update: July 31, 2018

1. PetaLinux Development

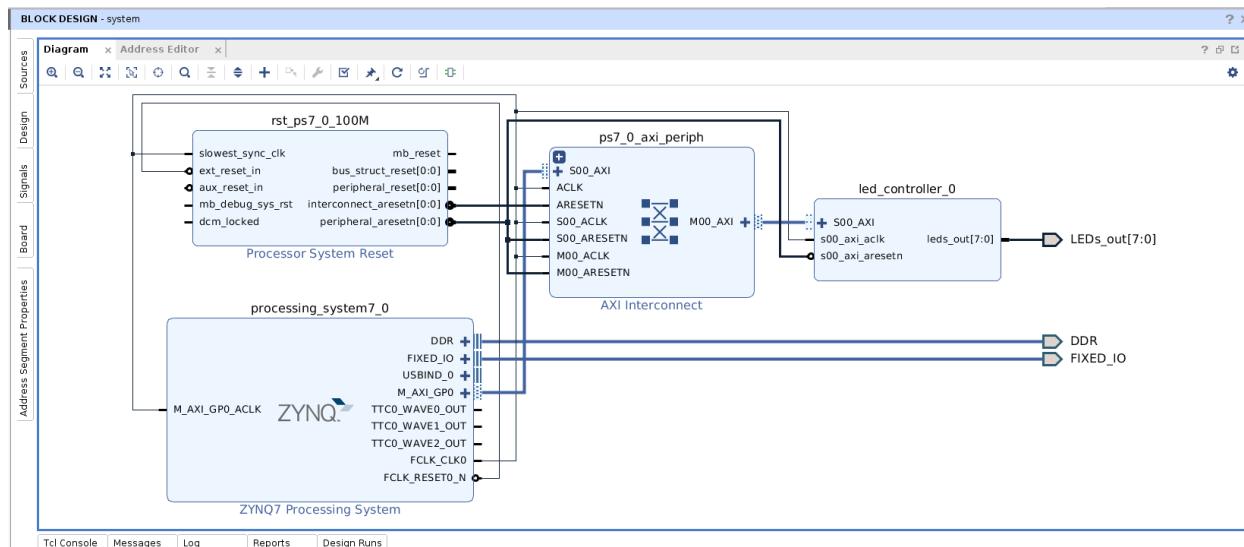
PetaLinux development for the LED controller Project. This Custom LED controller Project is explained well on our Udemy Course: [Embedded Design with Xilinx VIVADO and Zynq FPGA](#). This design is targeted for the ZedBoard FPGA Board, Rev D.

Notes:

We implement this project on the VIVADO 2018.2 and PetaLinux Version 2018.2 with [Centos OS on Windows VM](#).

For following this steps, you must have

- i. “Generate the Bistream” of the LED controller Project and exported it.
- ii. Download the PetaLinux version as of your VIVADO, follow this tutorial of ours on “[Installing PetaLinux on CentOS-LogicTronix Tutorial](#)”.
- iii. Download the ZedBoard BSP package from [Xilinx](#) Download Page: [Download Link for Petalinux 2018.3](#)



After above block design has been exported including bitstream, following steps are to compile the Petalinux build using above hardware design and running through Linux app.

1. Create a Petalinux project:

```
$petalinux-create --type project --name <PROJECT_NAME> --source
<PATH_TO_PETALINUX_PROJECT_BSP>
```

In my case it is:

```
$ petalinux-create --type project --name led_control --source avnet-digilent-
zedboard-v2018.2-final.bsp
```

2. Get Hardware description from the Exported Design

```
$ petalinux-config --get-hw-description <SDK-PROJECT-PATH> --project <PETALINUX-
PROJECT-NAME>
```

In my case it is:

```
petalinux-config --get-hw-description
Lab31_Led_Controller_project/Lab31/Lab31.sdk --project led_control
```

This will bring up a Linux menuconfig utility. We can use this to configure options for our Linux build. For now, just use arrow keys to select Exit and then press Enter.

3. Lets do some configurations in our build. We want to enable SSH, so we need to run petalinux-config again to configure the root filesystem to include the DropBear SSH server. Run the following command:

```
$ petalinux-config -c rootfs
```

If you are not inside the project directory use this command:

```
$ petalinux-config -c rootfs --project <PROJECT-PATH>
```

4. Press Enter to select Filesystem Packages, then select console/network, and then select dropbear. Use the arrow keys and spacebar to select dropbear. Then select Exit. Continue to exit pages until you exit out the menuconfig process. Select Yes to save.

5. Build the Linux image using following command:

```
$ petalinux-build
```

If you are not inside the project directory use this command:

```
$ petalinux-build --project <PROJECT-PATH>
```

6. Package the Linux build and FPGA:

```
$ petalinux-package --boot --format BIN --project <PROJECT-PATH> --fsbl <FIRST-
STAGE-BOOTLOADER-FILE> --fpga <BITSTREAM> --u-boot <U-BOOT-PATH>
```

In my case it is:

```
petalinux-package --boot --format BIN --project led_control/ --fsbl
led_control/images/linux/zynq_fsbl.elf --fpga
Lab31_Led_Controller_project/Lab31/Lab31.runs/impl_1/system_wrapper.bit --u-
boot led_control/images/linux/u-boot.elf
```

Copy the BOOT.BIN file containing the FSBL, FPGA image, and u-boot to the SD card you will be using on your ZedBoard. Also, copy the file led_control/images/linux/image.ub, which contains the Linux kernel and filesystem to boot the ZedBoard.

7. To control the LED controller through application in Linux compile the following C code.

```
#include <unistd.h>
#include <stdio.h>
#include <fcntl.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
#include <stdint.h>
#include <sys/types.h>
#include <sys/mman.h>
```

```

// Define maximum LED value (2^8)-1 = 255

#define LED_LIMIT 255

// Define delay length

#define DELAY 10000000

/* Define the base memory address of the led_controller IP core */
static volatile uint32_t *leds;
static const uint32_t leds_addr = 0x43C00000;

void map_gpios(void)
{
    int fd;
    if ((fd = open("/dev/mem", O_RDWR|O_SYNC)) < 0)

        if (fd < 0) {
            perror("/dev/mem");
            exit(-1);
        }

    if ((leds = mmap(0, getpagesize(), PROT_READ|PROT_WRITE, MAP_SHARED, fd,
        leds_addr))
        == MAP_FAILED) {
        perror("leds");
        exit(-1);
    }
}

static void run_leds_mode(void)
{
    /* unsigned 32-bit variables for storing current LED value */

    uint32_t led_val = 0;

    int i=0;

    printf("led_controller IP test begin.\r\n");

    printf("-----\r\n\r\n");

    /* Loop forever */

    while(1){

        while(led_val<=LED_LIMIT){

            /* Print value to terminal */

            printf("LED value: %d\r\n", led_val);

```

```

        /* Write value to led_controller IP core */

        *leds = led_val;

        /* increment LED value */

        led_val++;

        /* run a simple delay to allow changes on LEDs to be
visible */

        for(i=0;i<DELAY;i++);

    }

    /* Reset LED value to zero */

    led_val = 0;

}

/* main function */

int main(void)
{
    map_gpios();
    run_leds_mode();

    return 0;
}

```

Write the above code in file named say ***led_controller_test.c*** and compile it with following command:
\$ arm-linux-gnueabihf-gcc <C source code> -lm -o <App-Name>

In my case it is as follows:

```
$ arm-linux-gnueabihf-gcc led_controller_test.c -lm -o led_controller_test
```

8. This will build an ARM binary called ***led_controller_test***, which you can copy to the ZedBoard and run. I usually use the SSH copy command *scp* to copy the file to the file to the board. You can find out the IP address of the ZedBoard by typing *ifconfig* in the serial console window once you have logged in. From the serial console on the ZedBoard first login to the console. Use **root** as the username and password. Then issue the command *ifconfig*. This shows the IP address of my board is that for me is 192.168.1.77.

Copy the executable to the ZedBoard using the *scp* command.

```
$scp led_controller_test root@192.168.1.77:
```

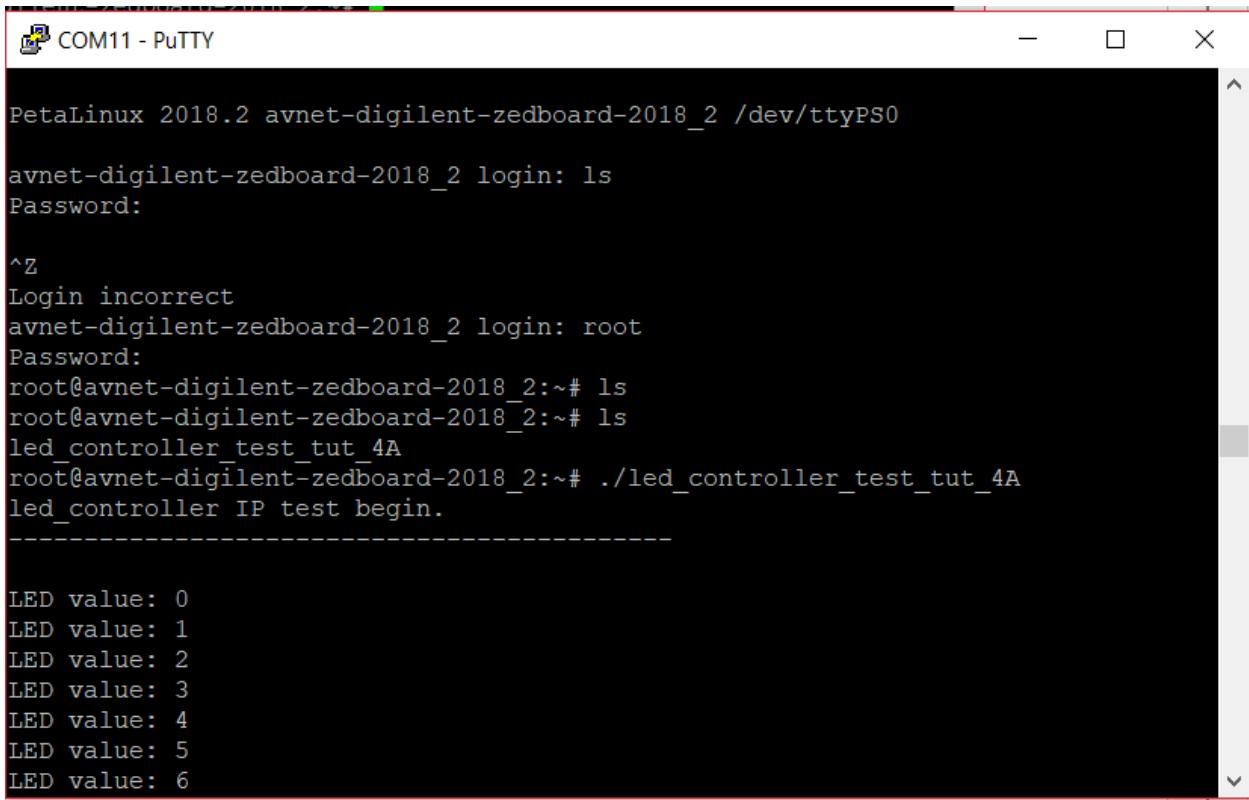
You will need to give the root password **root**.

Then from the console window check if the program has been transferred using command *ls*.

Then run the app like this:

```
# ./led_controller_test
```

The program should then run and you will see LED values increasing on the screen as well as on the ZedBoard.



COM11 - PuTTY

```
PetaLinux 2018.2 avnet-digilent-zedboard-2018_2 /dev/ttys0
avnet-digilent-zedboard-2018_2 login: ls
Password:

^Z
Login incorrect
avnet-digilent-zedboard-2018_2 login: root
Password:
root@avnet-digilent-zedboard-2018_2:~# ls
root@avnet-digilent-zedboard-2018_2:~# ls
led_controller test_tut_4A
root@avnet-digilent-zedboard-2018_2:~# ./led_controller test_tut_4A
led_controller IP test begin.

-----
LED value: 0
LED value: 1
LED value: 2
LED value: 3
LED value: 4
LED value: 5
LED value: 6
```

Thank You!

Reference:

- i. LogicTronix Petalinux Tutorial Series: <https://logictronix.com/our-resources/petalinux-development/>