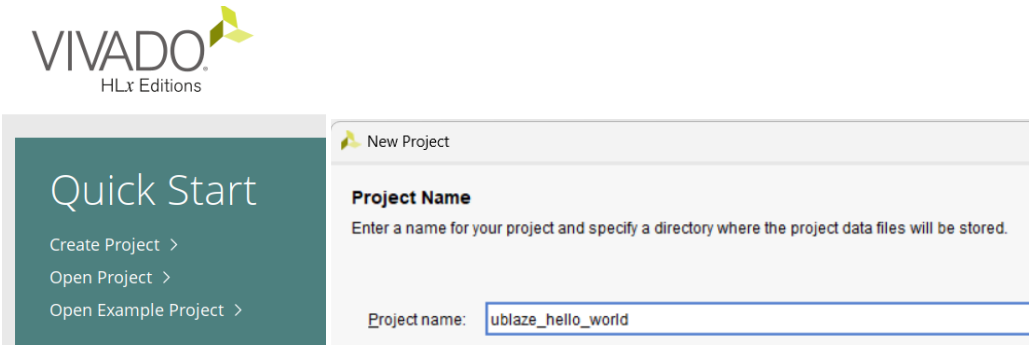
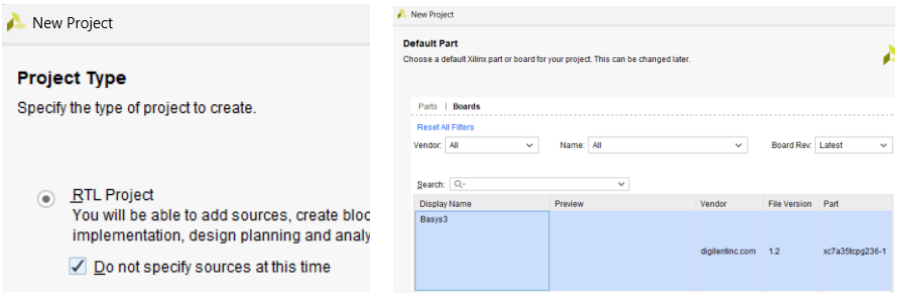
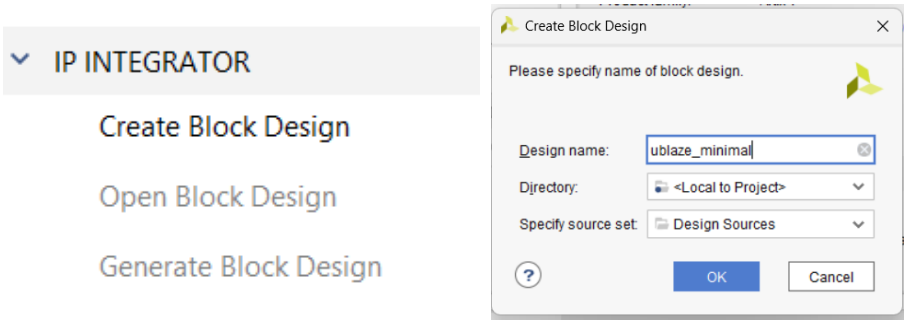
1- Create a new RTL project in Vivado and select the appropriate board (e.g. Basys3).

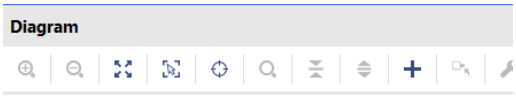


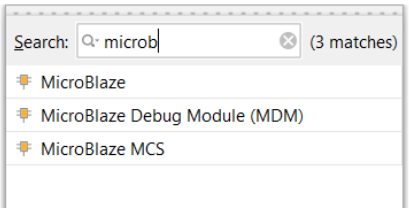


2- In the newly created project, create block design.

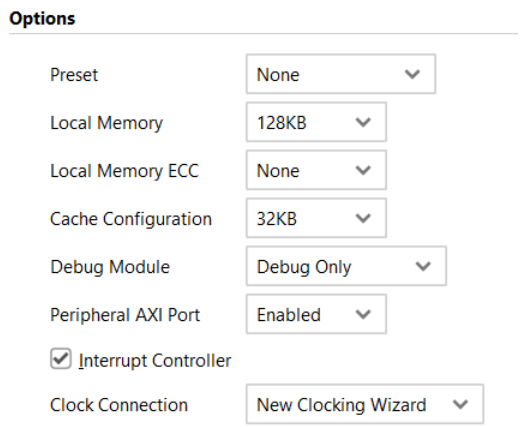


3- In the newly created "Diagram", add microblaze IP core.

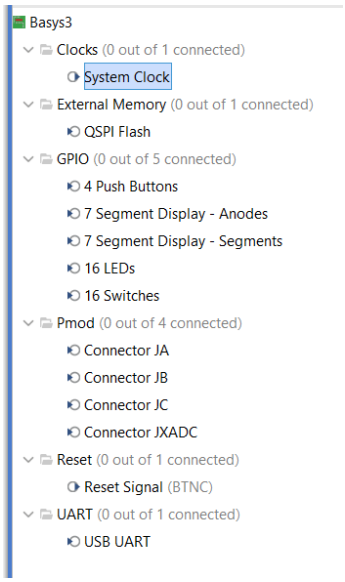




4- Click "Run block automation". Set an appropriate memory size. 128 Kb for now.

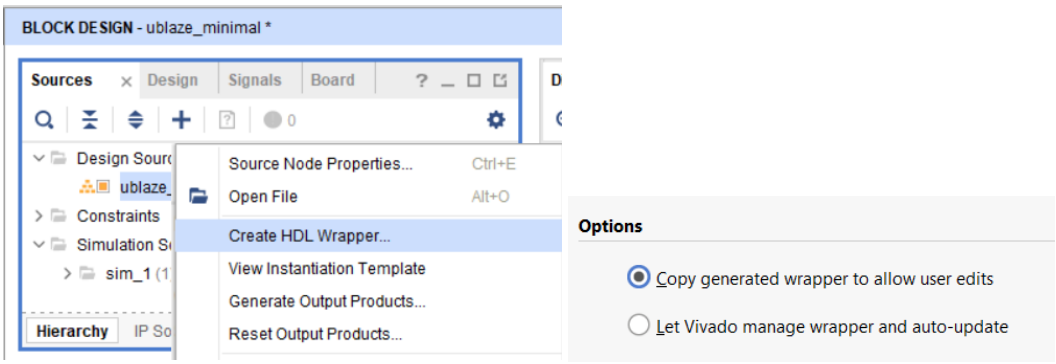


5- Enable board options one by one. It will add appropriate connections and/or IP core e.g. AXI GPIO. Don't enable PMODs and QSPI Flash for now.

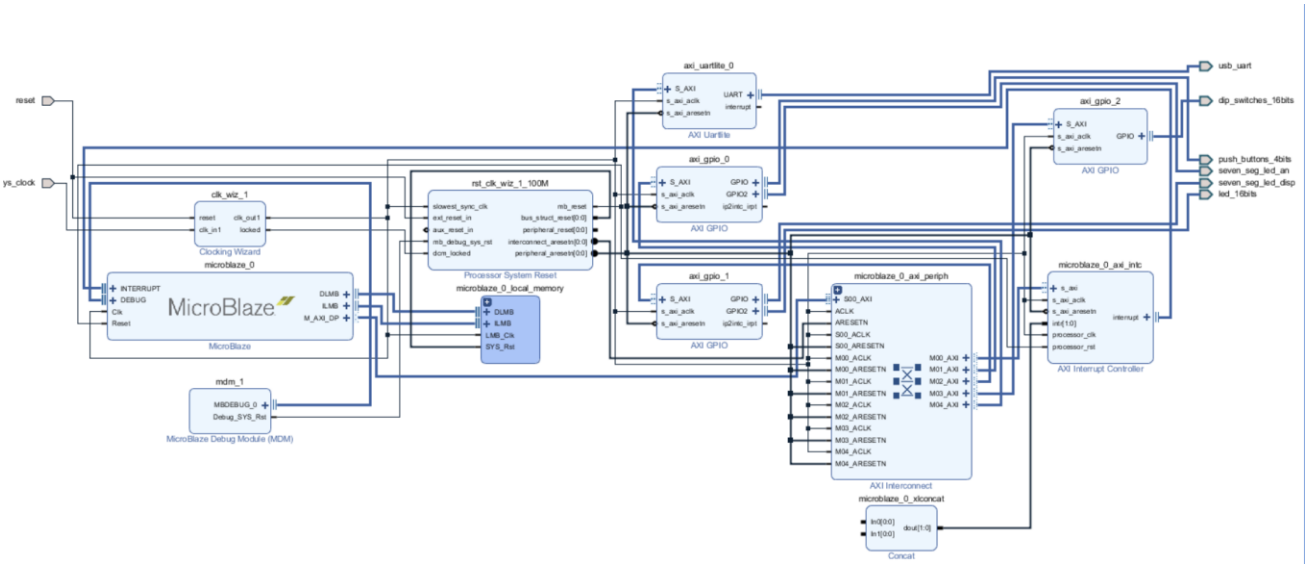


6- Run connection automation.

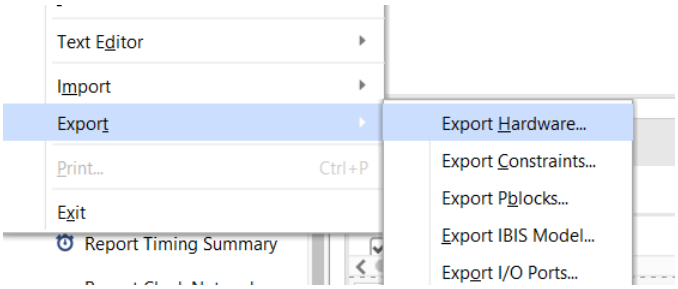
7- Tools->validate design to see if there is a problem. Disable caches in uBlaze if a warning appears.
8- Create HDL wrapper.

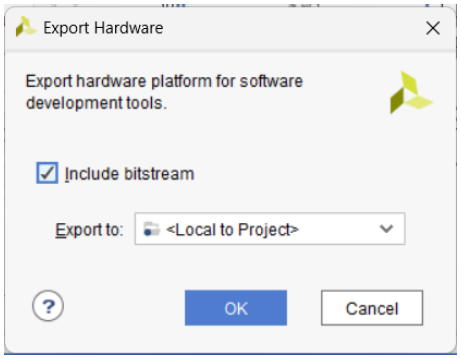


9- Open HDL wrapper to verify if all the included peripherals (GPIO and uart etc.) are present. If not, generation was not successful and previous steps may have to be repeated.
10- Manually verify the design by looking at the board schematic as well.



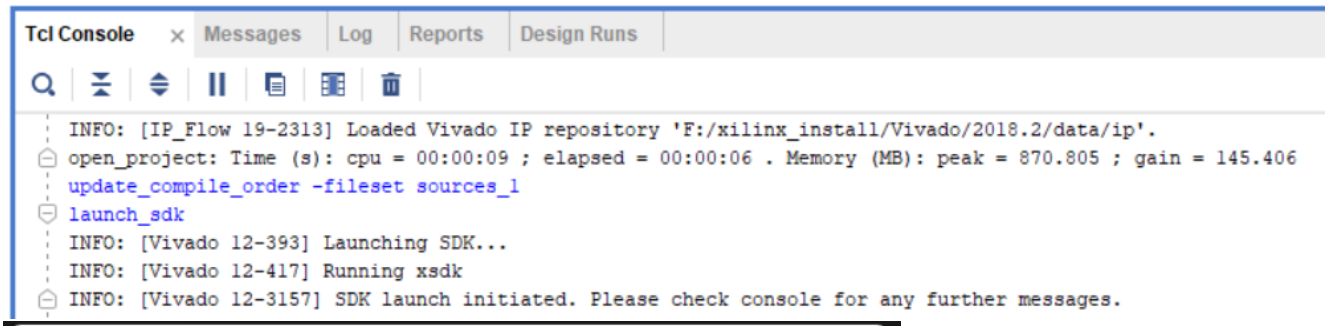
11- Press “Generate Bitstream”. It may take some time to synthesize, implement and route.
12- Export hardware including bitstream.

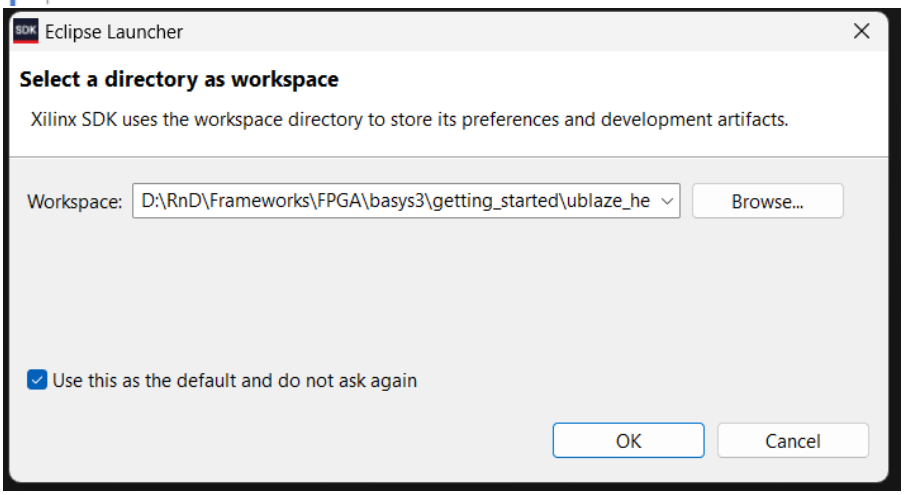




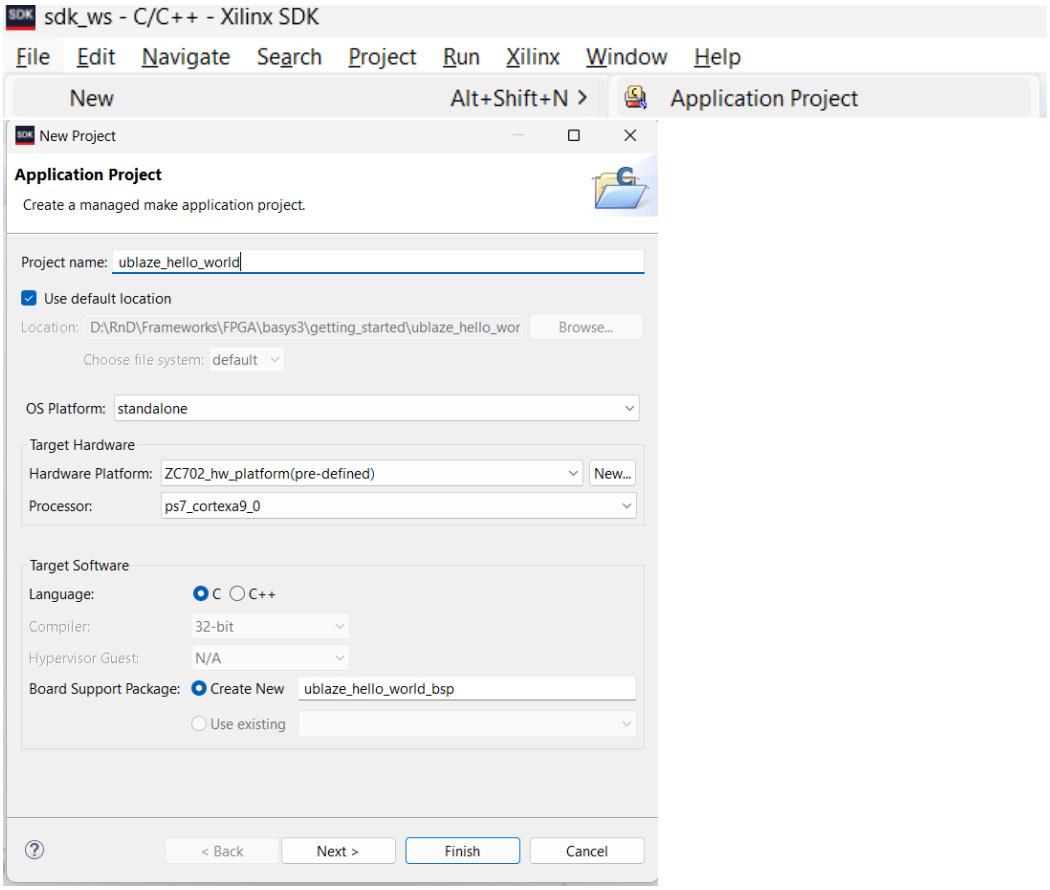
13- In the TCL console, type “launch_sdk” and select a folder as workspace.

If Vitis is installed then, Tools-> Launch Vitis IDE -> Create Platform Project

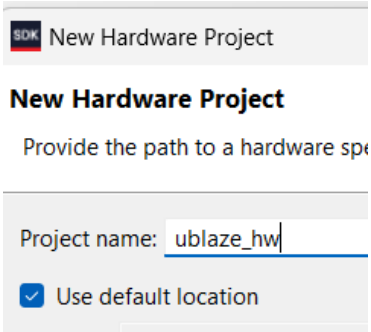


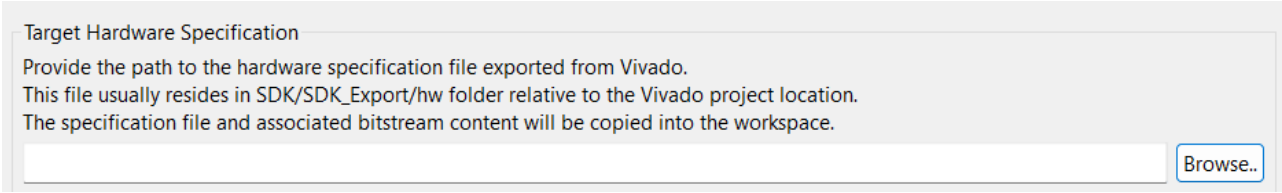


14- Create new application project.

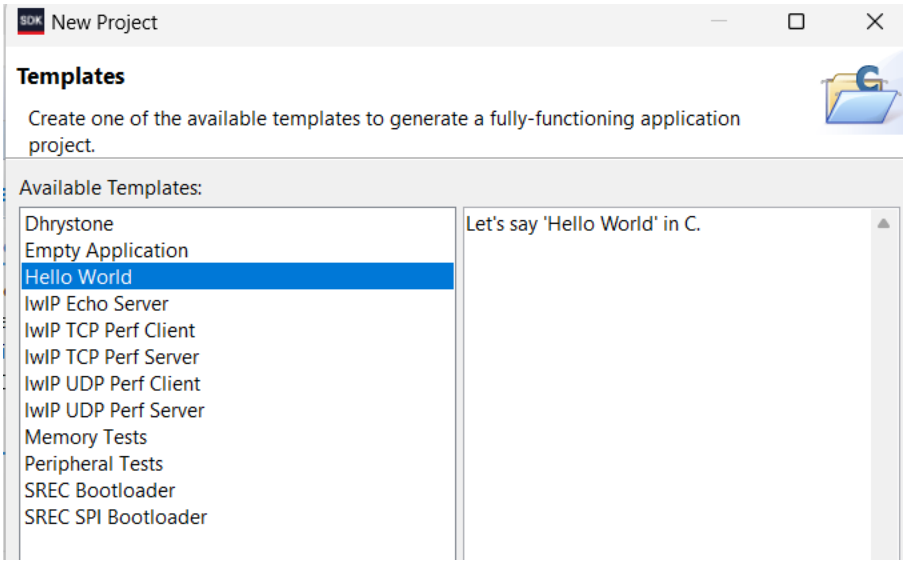


15- For the target hardware, click new. Give the project a name. Select the “hdf” file generated above in “export hardware” step. It would be located most likely in a folder with “sdk” in its name within the project. Otherwise search for files of type “*.hdf” in the vivado project directory.
For newer vivado version, “xsa” file is generated.

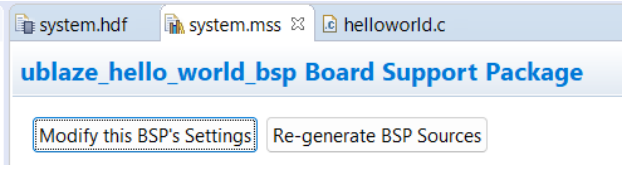




16- In the next step, select the “hello world” template.



17- Select Project -> Build All.
18- Select BSP from left side menu and click system.mss



19- Click Modify BSP Settings. Verify that stdin and stdout use the uart ip core. If not, then select appropriately.

Board Support Package Settings

Control various settings of your Board Support Package.

- Overview
  - standalone
  - drivers
    - microblaze_0

Configuration for OS: standalone

Name	Value
clocking	false
hypervisor_guest	false
lockstep_mode_debug	false
sleep_timer	none
stdin	axi_uartlite_0
stdout	axi_uartlite_0
ttc_select_cntr	2
zynqmp_fsbl_bsp	false
> microblaze_exceptions	false
> enable_sw_intrusive_profili	false

20- Hardware-related information can be seen from system.hdf file.

system.hdf | helloworld.c

ublaze_hw Hardware Platform Specification

Design Information

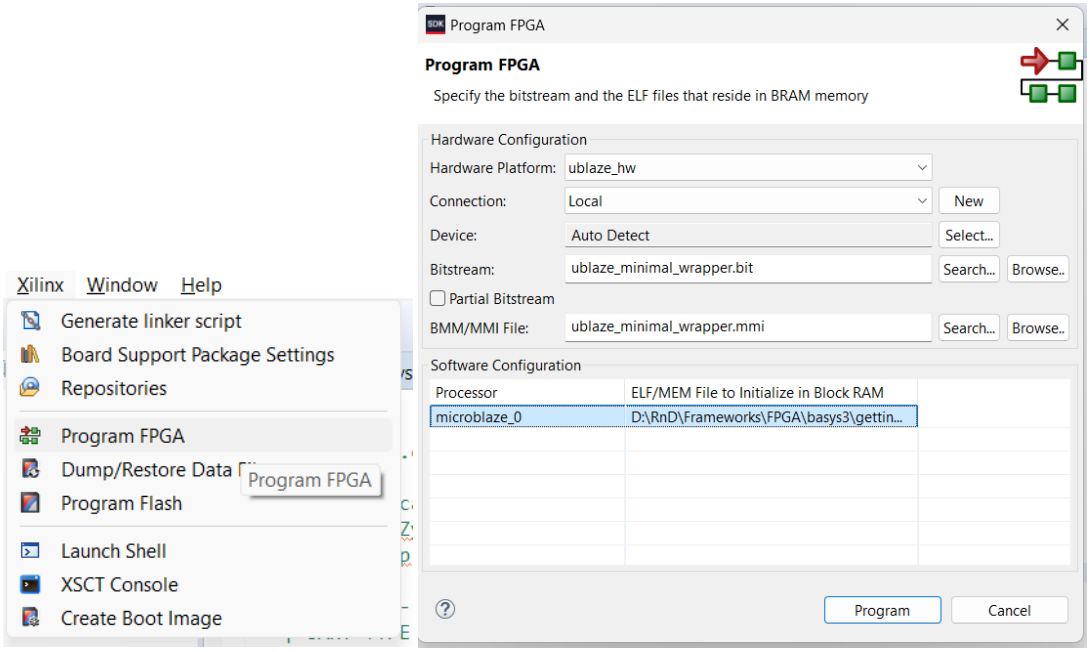
Target FPGA Device: 7a35t
Part: xc7a35tcpg236-1
Created With: Vivado 2018.2
Created On: Mon Sep 23 13:47:11 2024

Address Map for processor microblaze_0

Cell	Base Addr	High Addr	Slave I/f	Mem/Reg
axi_gpio_1	0x40010000	0x4001ffff	S_AXI	REGISTER
microblaze_0_axi_intc	0x41200000	0x4120ffff	s_axi	REGISTER
axi_gpio_0	0x40000000	0x4000ffff	S_AXI	REGISTER
axi_uartlite_0	0x40600000	0x4060ffff	S_AXI	REGISTER
microblaze_0_local_memor...	0x00000000	0x0001ffff	SLMB	MEMORY
axi_gpio_2	0x40020000	0x4002ffff	S_AXI	REGISTER

21- Right click the newly created application project and “Build”. This will create an executable “elf” file.

22- Click “Program FPGA”. Select the appropriate “elf” file. Only one elf file would be present in this project normally. Click “Program”.



23- The sample “helloworld.c” file only prints “Hello World” once on UART. Default baudrate is 9600. Open a UART program e.g. putty to see this message. Change the c code to print this in loop. Program FPGA again and see the repeating statement on UART.

```
int main()
{
    init_platform();

    while(1){
        print("Hello from Basys3......\n\r");
    }
    cleanup_platform();
    return 0;
}
```

24- Press the center push button to reset the system.

25- Run the following program to test 7-segment. The addresses for each hardware (e.g. push buttons, leds etc.) have to be carefully written. Use the system.hdf and vivado board file to match the addresses. For two-channel GPIO, the address offset is 8. E.g. if axi_gpio_0 has the address ‘0x40000000’, then GPIO can be accessed using this address while GPIO2 can be accessed using ‘0x40000008’.

```
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"


int* switches        = (int*) 0x40000000; // only 16 bits
int* leds            = (int*) 0x40010000; // only 16 bits
int* push_buttons    = (int*) 0x40020000; // only 4 bits [Down Right Left Up]
int* segment_data    = (int*) 0x40030000; // only 8 bits
int* segment_enable  = (int*) 0x40040000; // only 4 bits
int counter = 0;

const int svn_sg_code [] = {
                        0b11000000,  // 0---------- > index 0
                        0b11111001,  // 1---------- > index 1
                        0b10100100,  // 2---------- > index 2
                        0b10110000,  // 3---------- > index 3
                        0b10011001,  // 4---------- > index 4
                        0b10010010,  // 5---------- > index 5
                        0b10000010,  // 6---------- > index 6
                        0b11111000,  // 7---------- > index 7
                        0b10000000,  // 8---------- > index 8
                        0b10010000,  // 9---------- > index 9
};

void delay(int n) {
                        volatile int x = 0;

                        for (int i = 0; i < n; i++) {
                                        for (int j = 0; j < n; j++) {
                                                        x++;
                                        }
                        }
}

void display() {
                        int count_tmp = counter;
                        int first_digit = count_tmp%10;
                        count_tmp =  count_tmp/10;

                        int second_digit = count_tmp%10;
                        count_tmp =  count_tmp/10;

                        int third_digit = count_tmp%10;
                        count_tmp =  count_tmp/10;

                        int forth_digit = count_tmp%10;

                        *segment_data = svn_sg_code[first_digit];
                        *segment_enable = 0b1110;
                        delay(100);
                        *segment_data = svn_sg_code[second_digit];
                        *segment_enable = 0b1101;
                        delay(100);
                        *segment_data = svn_sg_code[third_digit];
                        *segment_enable = 0b1011;
                        delay(100);
                        *segment_data = svn_sg_code[forth_digit];
                        *segment_enable = 0b0111;
                        delay(100);
}



int main()
{
   init_platform();

   print("Hello Counter\n\r");




   int up;
   int down;

   for (;;) {
                        *leds = *switches;

                        if (*push_buttons & 0b0010) { // initialization
                                        counter = *switches;
                        }
                        if (*push_buttons & 0b0001) { // up
                                        up   = 1;
                                        down = 0;
                                        for (int i = 0; i < 50; i++) {
                                                        display();
                                        }
                        }
                        if (*push_buttons & 0b1000) { // down
                                        up   = 0;
                                        for (int i = 0; i < 50; i++) {
                                                        display();
                                        }
                                        down = 1;
                        }

                        if (up == 1) {
                                        counter++;
                                        if (counter > 9999)
                                                        counter = 0;
                                        up = 0;
                        }

                        if (down == 1) {
                                        counter--;
                                        if (counter < 0)
                                                        counter = 9999;
                                        down = 0;
                        }

                        display();

   }



   cleanup_platform();
   return 0;
}
```