COEN 317: MICROPROCESSOR-BASED SYSTEM

Section UN-X

LAB REPORT # 2: AND Array Implemented in Programmable Logic

INSTRUCTOR:
Dr. Fadi El-Hassan

Lab Coordinator:
Ted Obuchowicz

Submitted by:
Jaskirat Kaur
40138320

"I certify that this submission is my original work and meets the Faculty's Expectations of Originality"

Date Written: Monday, March 4, 2024

Objectives:

The objectives of the lab are to become familiar with the Xilinx ZC702 development board, specifically with its programmable logic. Additionally, it is also to gain experience with the PlanAhead tools, the Xilinx Platform Studio, and the SDK.

Introduction:

In this lab, we are asked to create the AND gate of two 8-bit inputs. A 16-bit wide GPIO will be used to provide the data to the AND gate. The outputs of the AND gate will be displayed on the development board's LED.

In this lab, the first step was to set up the environment and launch the PlanAhead tool. The second was designing the system in XPS. This step was crucial to understanding how components interact with each other and how to configure them correctly. The next step was to configure the port settings and add GPIO for the AND gate inputs. GPIO stands for general-purpose input and output. It is a pin on the board that can be controlled by the programmer at run time. Created and synthesized a top-level VHDL file, modifying it to include the AND gate components. Then download the bitstream generated in the previous step to Zynq Board. Writing and running the application in SDK allowing to see the direct impact of software on hardware.

For more information on the lab including a block diagram of the system, please refer to the lab manual.

Results:

Similarly to Lab 1, this lab is also directed to introduce us to the development board. The lab required us to follow the detailed steps of how to setup the programs and what to write. However, it is important to understand the procedures and the connections that we are setting. In PlanAhead, we created a RTL Project that uses a VHDL file and a UCF file.

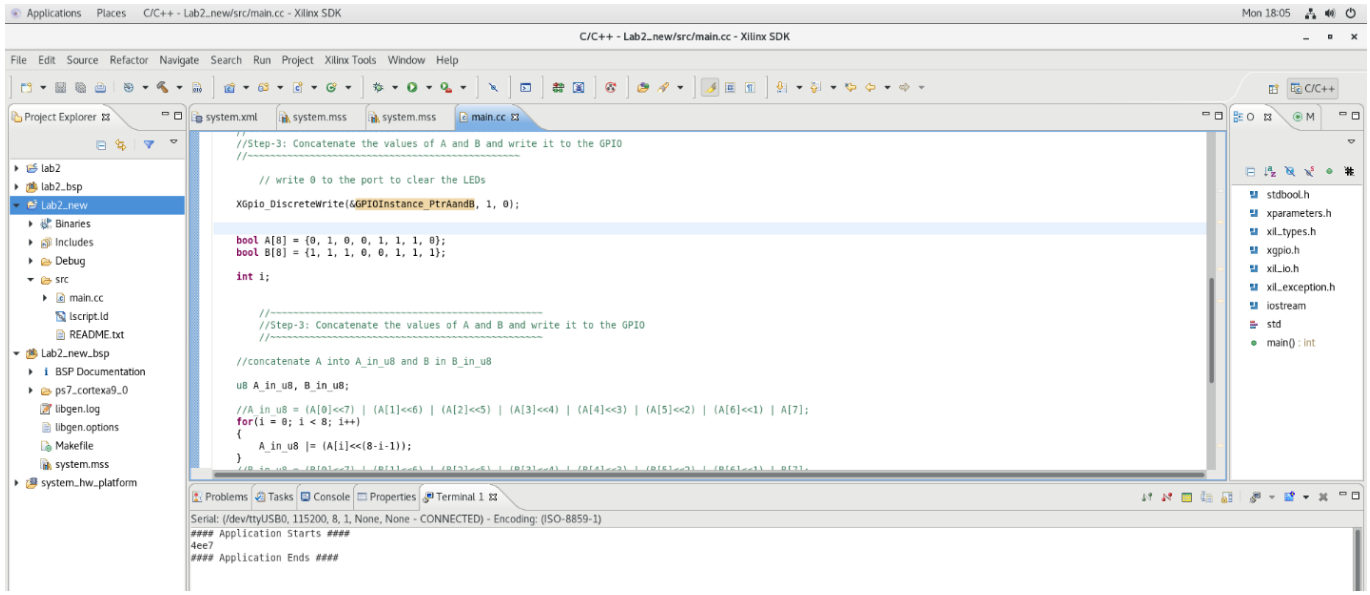
- The VDHL file will program the board to read two 8-bit inputs and AND them to an 8-bit output.
- The UCF file will tell the program which GPIO ports to use as inputs and as outputs.

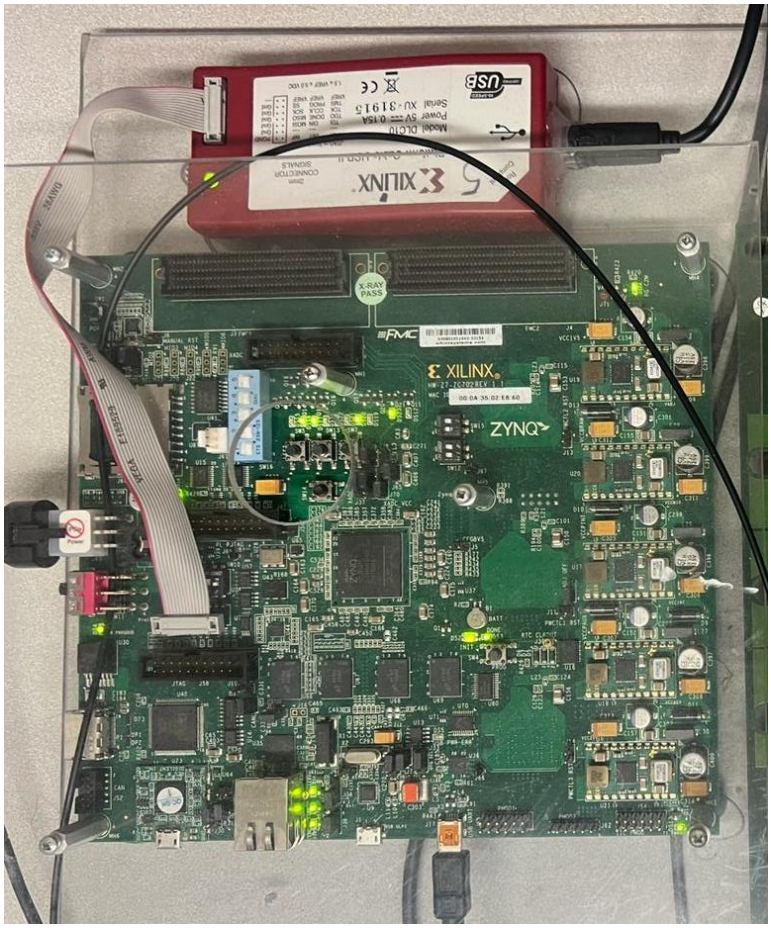
From there, we then created a system in XPS.

This system sets up our IO connections.

Lastly, before synthesizing, implementing, and generating the bitstream on PlanAhead, we modify the system_stub so that the new input and output values can be included. This is important so that the UCF ports can match.

In the end, I managed to get my “AND” program to run successfully and was able to see the 8 LEDs light according to the output in the program.


Applications Places C/C++ - Lab2_new/src/main.cc - Xilinx SDK Mon 18:05
C/C++ - Lab2_new/src/main.cc - Xilinx SDK
File Edit Source Refactor Navigate Search Run Project Xilinx Tools Window Help
Project Explorer
lab2
lab2_bsp
Lab2_new
Binaries
Includes
Debug
src
main.cc
lscript.ld
README.txt
Lab2_new_bsp
BSP Documentation
ps7_cortexa9_0
libgen.log
libgen.options
Makefile
system.mss
system_hw_platform
system.xml
system.mss
system.mss
main.cc
//Step-3: Concatenate the values of A and B and write it to the GPIO
// write 0 to the port to clear the LEDs
XGpio_DiscreteWrite(&GPIOInstance_PtrAandB, 1, 0);
bool A[8] = {0, 1, 0, 0, 1, 1, 1, 0};
bool B[8] = {1, 1, 1, 0, 0, 1, 1, 1};
int i;
//Step-3: Concatenate the values of A and B and write it to the GPIO
//concatenate A into A_in_u8 and B in B_in_u8
u8 A_in_u8, B_in_u8;
//A_in_u8 = (A[0]<<7) | (A[1]<<6) | (A[2]<<5) | (A[3]<<4) | (A[4]<<3) | (A[5]<<2) | (A[6]<<1) | A[7];
for(i = 0; i < 8; i++)
{
A_in_u8 |= (A[i]<<(8-i-1));
}
stdbool.h
xparameters.h
xil_types.h
xgpio.h
xil_io.h
xil_exception.h
iostream
std
main() : int
Problems Tasks Console Properties Terminal 1
Serial: (/dev/ttyUSB0, 115200, 8, 1, None, None - CONNECTED) - Encoding: (ISO-8859-1)
Application Starts
4ee7
Application Ends


XILINX
ZYNQ

Question:

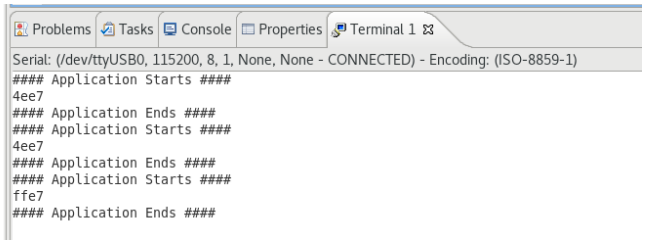
We are asked to change the values in the two input arrays and observe the resulting changes in the LEDs. To assure that the program output and what we are observing is correct, we can perform the AND of A and B ourselves. The output is 1 if and only if the two input bits are a 1. Therefore, I decided to make all my input A bits 1 and check to check which the LEDs are lit. Results are as expected:

A = {1, 1, 1, 1, 1, 1, 1, 1}

B = {1, 1, 1, 0, 0, 1, 1, 1}

Output of A AND B should be {1, 1, 1, 0, 0, 1, 1, 1}

The following image is the terminal that illustrates how '4ee7' changes to 'ffe7' and output from the LEDs.



lab2_gpio_for_C.ucf

```
# output LEDS on ZC702 base board - verified
NET  output<0>     LOC = E15;  # | IOSTANDARD=LVCMOS18; # Bank  13 VCCO - VADJ     - IO_L23P_T3_13
NET  output<1>     LOC = D15;  # | IOSTANDARD=LVCMOS18; # Bank  13 VCCO - VADJ     - IO_L3N_T0_DQS_13
NET  output<2>     LOC = W17;  #  | IOSTANDARD=LVCMOS18; # Bank  34 VCCO - VADJ     - IO_L20N_T3_34
NET  output<3>     LOC = W5;   #  | IOSTANDARD=LVCMOS18; # Bank  34 VCCO - VADJ     - IO_L20P_T3_34
NET  output<4>     LOC = V7;   #  | IOSTANDARD=LVCMOS18; # Bank  35 VCCO - VADJ     - IO_L3P_T0_DQS_AD1P_35
NET  output<5>     LOC = W10;  #  | IOSTANDARD=LVCMOS18; # Bank  35 VCCO - VADJ     - IO_L3N_T0_DQS_AD1N_35
NET  output<6>     LOC = P18;  #  | IOSTANDARD=LVCMOS18; # Bank  33 VCCO - VADJ     - IO_L13P_T2_MRCC_33
NET  output<7>     LOC = P17;  #  | IOSTANDARD=LVCMOS18; # Bank  13 VCCO - VADJ     - IO_L24N_T3_13
NET  output[*] IOSTANDARD = LVCMOS25;
```

Main.cc:

```
#include "stdbool.h"

#include "xparameters.h"
#include "xil_types.h"
#include "xgpio.h"
#include "xil_io.h"
#include "xil_exception.h"

#include <iostream>
using namespace std;

int main()
{

   static XGpio GPIOInstance_PtrAandB;
   int xStatus;

   cout << "#### Application Starts ####" << endl;

   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   //Step-1: AXI GPIO Initialization
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   xStatus = XGpio_Initialize(&GPIOInstance_PtrAandB,
XPAR_AXI_GPIO_FOR_A_AND_B_DEVICE_ID);
   if(xStatus != XST_SUCCESS)
   {
   cout << "GPIO A Initialization FAILED" << endl;
   return 1;
   }

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//Step-2: AXI GPIO Set the Direction
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//XGpio_SetDataDirection(XGpio *InstancePtr, unsigned Channel, u32 DirectionMask);
//we use only channel 1, and 0 is the the parameter for output
```

```
XGpio_SetDataDirection(&GPIOInstance_PtrAandB, 1, 0);

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//Step-3: Concatenate the values of A and B and write it to the GPIO
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

      // write 0 to the port to clear the LEDs

XGpio_DiscreteWrite(&GPIOInstance_PtrAandB, 1, 0);


bool A[8] = {1, 1, 1, 1, 1, 1, 1, 1};
bool B[8] = {1, 1, 1, 0, 0, 1, 1, 1};

int i;


      //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      //Step-3: Concatenate the values of A and B and write it to the GPIO
      //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

//concatenate A into A_in_u8 and B in B_in_u8

u8 A_in_u8, B_in_u8;

//A_in_u8 = (A[0]<<7) | (A[1]<<6) | (A[2]<<5) | (A[3]<<4) | (A[4]<<3) | (A[5]<<2) | (A[6]<<1) | A[7];
for(i = 0; i < 8; i++)
{
A_in_u8 |= (A[i]<<(8-i-1));
}
//B_in_u8 = (B[0]<<7) | (B[1]<<6) | (B[2]<<5) | (B[3]<<4) | (B[4]<<3) | (B[5]<<2) | (B[6]<<1) | B[7];
for(i = 0; i < 8; i++)
{
B_in_u8 |= (B[i]<<(8-i-1));
}

      // shift the concatenated  A array 8 bits to the right and bitwise OR it with the concatenated B array
      // to form the 16 bit data to be written to the GPIO port

      // send it to cout so we see it on the terminal also
  // note we we use the ::hex  to specify that the output should be
  // displayed in hex

     cout << ::hex <<  ((A_in_u8<<8) | B_in_u8)  << endl;

XGpio_DiscreteWrite(&GPIOInstance_PtrAandB, 1, ((A_in_u8<<8) | B_in_u8));

   cout << "#### Application Ends ####" << endl;

   return 0;
}
```

System_stub.vhd:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

library UNISIM;
use UNISIM.VCOMPONENTS.ALL;

entity system_stub is
  port (
    processing_system7_0_MIO : inout std_logic_vector(53 downto 0);
    processing_system7_0_PS_SRSTB : in std_logic;
    processing_system7_0_PS_CLK : in std_logic;
    processing_system7_0_PS_PORB : in std_logic;
    processing_system7_0_DDR_Clk : inout std_logic;
    processing_system7_0_DDR_Clk_n : inout std_logic;
    processing_system7_0_DDR_CKE : inout std_logic;
    processing_system7_0_DDR_CS_n : inout std_logic;
    processing_system7_0_DDR_RAS_n : inout std_logic;
    processing_system7_0_DDR_CAS_n : inout std_logic;
    processing_system7_0_DDR_WEB_pin : out std_logic;
    processing_system7_0_DDR_BankAddr : inout std_logic_vector(2 downto 0);
    processing_system7_0_DDR_Addr : inout std_logic_vector(14 downto 0);
    processing_system7_0_DDR_ODT : inout std_logic;
    processing_system7_0_DDR_DRSTB : inout std_logic;
    processing_system7_0_DDR_DQ : inout std_logic_vector(31 downto 0);
```

```
    processing_system7_0_DDR_DM : inout std_logic_vector(3 downto 0);
    processing_system7_0_DDR_DQS : inout std_logic_vector(3 downto 0);
    processing_system7_0_DDR_DQS_n : inout std_logic_vector(3 downto 0);
    processing_system7_0_DDR_VRN : inout std_logic;
    processing_system7_0_DDR_VRP : inout std_logic;
    axi_gpio_for_A_and_B_GPIO_IO_O_pin : out std_logic_vector(0 to 15);
    output                          : out std_logic_vector(0 to 7)
  );
end system_stub;

architecture STRUCTURE of system_stub is

  component system is
    port (
      processing_system7_0_MIO : inout std_logic_vector(53 downto 0);
      processing_system7_0_PS_SRSTB : in std_logic;
      processing_system7_0_PS_CLK : in std_logic;
      processing_system7_0_PS_PORB : in std_logic;
      processing_system7_0_DDR_Clk : inout std_logic;
      processing_system7_0_DDR_Clk_n : inout std_logic;
      processing_system7_0_DDR_CKE : inout std_logic;
      processing_system7_0_DDR_CS_n : inout std_logic;
      processing_system7_0_DDR_RAS_n : inout std_logic;
      processing_system7_0_DDR_CAS_n : inout std_logic;
      processing_system7_0_DDR_WEB_pin : out std_logic;
      processing_system7_0_DDR_BankAddr : inout std_logic_vector(2 downto 0);
```

```
    processing_system7_0_DDR_Addr : inout std_logic_vector(14 downto 0);
    processing_system7_0_DDR_ODT : inout std_logic;
    processing_system7_0_DDR_DRSTB : inout std_logic;
    processing_system7_0_DDR_DQ : inout std_logic_vector(31 downto 0);
    processing_system7_0_DDR_DM : inout std_logic_vector(3 downto 0);
    processing_system7_0_DDR_DQS : inout std_logic_vector(3 downto 0);
    processing_system7_0_DDR_DQS_n : inout std_logic_vector(3 downto 0);
    processing_system7_0_DDR_VRN : inout std_logic;
    processing_system7_0_DDR_VRP : inout std_logic;
    axi_gpio_for_A_and_B_GPIO_IO_O_pin : out std_logic_vector(15 downto 0)
  );
 end component;

 component and_gate8 is
   port(    x: in std_logic_vector(0 to 7);
            y: in std_logic_vector(0 to 7);
            F: out std_logic_vector(0 to 7)
   );
 end component;



attribute BOX_TYPE : STRING;
attribute BOX_TYPE of system : component is "user_black_box";

signal signal_for_A_and_B : std_logic_vector(0 to 15);
signal output_signal      : std_logic_vector(0 to 7);
```

```
begin


  system_i : system
    port map (
      processing_system7_0_MIO => processing_system7_0_MIO,
      processing_system7_0_PS_SRSTB => processing_system7_0_PS_SRSTB,
      processing_system7_0_PS_CLK => processing_system7_0_PS_CLK,
      processing_system7_0_PS_PORB => processing_system7_0_PS_PORB,
      processing_system7_0_DDR_Clk => processing_system7_0_DDR_Clk,
      processing_system7_0_DDR_Clk_n => processing_system7_0_DDR_Clk_n,
      processing_system7_0_DDR_CKE => processing_system7_0_DDR_CKE,
      processing_system7_0_DDR_CS_n => processing_system7_0_DDR_CS_n,
      processing_system7_0_DDR_RAS_n => processing_system7_0_DDR_RAS_n,
      processing_system7_0_DDR_CAS_n => processing_system7_0_DDR_CAS_n,
      processing_system7_0_DDR_WEB_pin => processing_system7_0_DDR_WEB_pin,
      processing_system7_0_DDR_BankAddr => processing_system7_0_DDR_BankAddr,
      processing_system7_0_DDR_Addr => processing_system7_0_DDR_Addr,
      processing_system7_0_DDR_ODT => processing_system7_0_DDR_ODT,
      processing_system7_0_DDR_DRSTB => processing_system7_0_DDR_DRSTB,
      processing_system7_0_DDR_DQ => processing_system7_0_DDR_DQ,
      processing_system7_0_DDR_DM => processing_system7_0_DDR_DM,
      processing_system7_0_DDR_DQS => processing_system7_0_DDR_DQS,
      processing_system7_0_DDR_DQS_n => processing_system7_0_DDR_DQS_n,
      processing_system7_0_DDR_VRN => processing_system7_0_DDR_VRN,
      processing_system7_0_DDR_VRP => processing_system7_0_DDR_VRP,
```

```
    axi_gpio_for_A_and_B_GPIO_IO_O_pin => signal_for_A_and_B
  );

 and8: and_gate8
  port map( x => signal_for_A_and_B(0 to 7),
            y => signal_for_A_and_B(8 to 15),
            F => output_signal
  );

 output <= output_signal;


end architecture STRUCTURE;
```

Lab2_AND.vhd

```
-- 8-bit AND gate


library ieee;
use ieee.std_logic_1164.all;

----------------------------------------------------

entity and_gate8 is
port( x: in std_logic_vector(0 to 7);
   y: in std_logic_vector(0 to 7);
   F: out std_logic_vector(0 to 7)
);
end and_gate8;


architecture Behavioral of and_gate8 is
begin
F <= x and y;

end Behavioral;
```

Conclusion:

In conclusion, the objectives of the lab were met. I managed to learn how to use the programmable logic within the Xilinx ZC702 board. I also managed to gain experience and practice with PlanAhead and Xilinx Platform Studio.