LAB 5**:** Direct Memory Access

SUBMITTED TO: SINA GHOFRANI MAJELAN

SUBMITTED BY: NABILA TABASSUM

STD ID: 4019039

COURSE ID: COEN\_317\_FN\_X

THIS IS TO CERTIFY, THIS IS MY ORIGINAL WORK AND MEET GINA CODY’S EXPECTATION OF ORIGINALITY

DATE: 11TH DECEMBER, 2022.

**INTRODUCTION AND OBJECTIVE**

This lab introduces the intellectual property core of AXI CDMA. The purpose of using AXI CDMA is that it will be controlling the two of the high-performance slave ports out of four high performance slave port by acting as a Master controlling device.

**IMPLEMENTATION:**

In the terminal, plan ahead was launch by using the extra space. Then, new project was created by selection the project name as

Lab 5. No sources were specified by selecting VHDL as targeted language. No specification of UCF file was needed for this lab. Afterwards, for the evaluation board, ZYNQ-7 ZC702 board was chosen, for

the project to be created.Next, an embedded processor project was created with Add source wizard, by selecting add source

under project, then just add embedded source, create sub design, and named the system as module.

Then, the system was designed in XPS, by creating a Base System using the BSB Wizard, afterwards, the

AXI System should be selected by default. Later, Verification of the Zynq Processing System 7 is selected

by Removing the GPIO\_SW and LEDs\_4Bits Peripherals. Lastly, just add the axi\_cdma 3.04a IP instance by making changes in the user tab like Data width of data transfer channel: THE1024 Maximum Burst Length: 256 and choosing the option User will make necessary connections and settings instead of selecting processing\_system7\_0 and select okay.

Then add two AXI interconnect and name it axi\_interconnect\_gp1 and axi\_interconnect\_gp1. From the n 32b GP AXI Master Port, select the Enable M\_AXI\_GP1 interface. Then, select okay to close the window. Again, double click on the green High Performance AXI 32b/64b Slave Ports block, select the check box for the Enable S\_AXI\_HP0 interface by Specifying the HP0 Base Address to 0x20000000 and the HP0 High Address to 0x2ffffffff. Moreover, select the check box for the Enable S\_AXI\_HP2 interface by specifying the HP2 Base Address to 0x30000000 and the HP2 High Address to 0x3ffffffff. From the assembly view, Connect the master and slave as specified in the lab manual. Lastly, exist the close the XPS by file -> exist.

The following step is to export the Hardware to SDK by Right-click on “system (system.xmp)” and choose

Create Top HDL. Then, Right-click on “system (system.xmp)” and choose Create Top HDL, Afterwards,

synthesize the VHDL code by clicking on Run Synthesis on the left Panel. When the implementation of

synthesis was successful, then implementation was run. When implementation was successfully done,

Bitstream was generated upon successful completion of it. The bitstream was downloaded to the Zynq

Board by attaching the power cable, the Platform Cable USB II, and the serial cable for the UART.

Moreover, the impact was launched by applying power to the board and the verification of the Platform

Cable USB II status LED is being done by illuminated in green.

Lastly, using SDK application project was created. A new software project was created and then the file

Lab5.sdk was copied into it. The program was compiled to build the executable and connection of the

SDK terminal was done to the board. Afterwards, the green connects buttons was connected and

desired connection types and port was selected. The Run the executable file on the board was run by

clicking run configuration.

**CONCLUSION**

Lastly, the desired result was obtained for this lab. Moreover, the purpose of this lab is to familiarize ourselves AXI CDMA, which was obtained.

**QUESTIONS AND RESULTS:**

**Q1:Add an instance of an AXI Timer to your design (using XPS). Use the timer to measure how long (in number of clock cycles) it takes to transfer the array without using the CDMA and how long it takes to perform transfer using the CDMA. Obtain values for different sizes of the array: 1024 integers, 4096 integers, 8192, 16384, 32767, 1048576, up to a maximum number of integers allowed in one transfer. Compute the speedup given by the use of the CDMA to perform the memory transfer.**

Size of array =1024

Graphical user interface, text, application, email

Description automatically generated

Size of array =4096

Graphical user interface, text, application, email

Description automatically generated

Size of array =8192

Graphical user interface, text, application, email

Description automatically generated

Size of array= 16384

Graphical user interface, text, application, email

Description automatically generated

Size of array: 32767

Graphical user interface, text, application, email

Description automatically generated

Size of array=1048576;Graphical user interface, text, application, email

Description automatically generated

**Q2: What will happen if we program the BTT register with a number which is greater than the maximum allowed?**

Programming the BIT register with a number from 31 to 23 will result in no effect as writing to these bits has no effect, and they are always read as zeroes.

**Terminal:**

Table

Description automatically generated

**Reference:**

[1]. LogiCORE IP AXI Central direct Memory Access v3.03a Product Guide, Xilinx Inc., PG034 October 16, 2012, p.5

[2]. Zynq-7000 All Programmable SoC: Concepts, Tools, and Techniques (CTT), Xilinx Inc., UG873 (v14.4) December 18 (Keith’s birthday), 2012, p.52

**APPENDIX:**

#include "xil\_exception.h"

#include "xil\_cache.h"

#include "xparameters.h"

#include <iostream>

#include "xtmrctr.h"

using namespace std;

int main()

{

XTmrCtr TimerInstancePtr;// timer pointer

u32\* cdma\_ptr = (u32\*) XPAR\_AXI\_CDMA\_0\_BASEADDR; //pointer for cdma base address

u32\* source\_ptr = (u32\*) XPAR\_PS7\_DDR\_0\_S\_AXI\_HP0\_BASEADDR; // pointer for source address

u32\* destination\_ptr = (u32\*) XPAR\_PS7\_DDR\_0\_S\_AXI\_HP2\_BASEADDR; //pointer for destinationaddress

int xStatus;

int size= 1048576;

for(int i =0; i <= size; i++) //initializing the contents of the source array

{

\*(source\_ptr + i) = i;

}

for(int i =0; i <= size; i++)//initializing the contents of the destination array

{

\*(destination\_ptr + i) = -i;

}

xStatus = XTmrCtr\_Initialize(&TimerInstancePtr, XPAR\_AXI\_TIMER\_0\_DEVICE\_ID);

if(xStatus != XST\_SUCCESS)

{

cout << "TIMER INIT FAILED" << endl;

return 0;

}

//setting the timer reset value

XTmrCtr\_SetResetValue(&TimerInstancePtr, 0, 0);

//setting the timer option

XTmrCtr\_SetOptions(&TimerInstancePtr, XPAR\_AXI\_TIMER\_0\_DEVICE\_ID, XTC\_CAPTURE\_MODE\_OPTION);

//reset the cdma

\*(cdma\_ptr) = 0x00000004;

//configure the cdma

\*(cdma\_ptr) = 0x00000020;

//load address of source array

\*(cdma\_ptr + 6) = 0x20000000;

//load address of destination array

\*(cdma\_ptr + 8) = 0x30000000;

//flush the cash

Xil\_DCacheFlush();

//number of bytes to transfer

\*(cdma\_ptr + 10) = size\*4;

//start the timer

XTmrCtr\_Start(&TimerInstancePtr, 0);

//begin transfer

int idle\_bit\_in\_status\_register ;

idle\_bit\_in\_status\_register = \*(cdma\_ptr + 1) & 2;

while(idle\_bit\_in\_status\_register== 0)

{

idle\_bit\_in\_status\_register = \*(cdma\_ptr + 1) & 2; //isolate the idle bit

if(idle\_bit\_in\_status\_register == 2){

break;

}

}

XTmrCtr\_Stop(&TimerInstancePtr, 0); // stop timer

unsigned int clock\_timer;

clock\_timer = XTmrCtr\_GetValue(&TimerInstancePtr, 0);

bool are\_equal = false;

for(int i=0 ; i <= 64; i++)

if ( \*(destination\_ptr + i) = \*(source\_ptr + i)){

are\_equal = true;

cout <<"same vale for teh destination and source : " << \*(destination\_ptr + i) << " " << \*(source\_ptr + i) << endl;

}

else

{

cout <<"values are diffrent for destination and source " << \*(destination\_ptr + i) << " " << \*(source\_ptr + i) << endl;

cout<< "app ends"<<endl;

}

if (!(are\_equal)){

cout << "values are different for source and destination reg!" << endl;

cout<< "app ends"<<endl;

}

else {

cout << "values for source and destination register are equal!" << endl;

}

cout << "it takes "<< clock\_timer <<" to perform"<<endl;

return 0;

}