International Journal of Enhanced Research in Science, Technology & Engineering
ISSN: 2319-7463, Vol. 4 Issue 8, August-2015

# Designing Embedded AXI Based Direct Memory Access System

Mazin Rejab Khalil[1], Rafal Taha Mahmood[2]

[1]Assistant Professor, Computer Engineering, Technical College, Mosul, Iraq
[2]MA Student Research Stage, Computer Engineering, Technical College, Mosul, Iraq

## ABSTRACT

**The work aims to design a soft core processor system with Advanced eXtensible Interface (AXI) processor bus which deals with different data capacities with 32, 64, 128, and 256 bits data width. The system deals with Direct Memory Access (DMA) unit to transfer data between the system memory and external peripheral. Memory Controller Block – Dual Data Rate (MCB-DDR2) external memory is introduced to act as main memory system. Embedded design techniques were applied to construct the processor system which is to be configured on Spartan-6 (SP601) Field Programmable Gate Arrays (FPGAs) slice.**

**Integrated Software Environment (ISE 13.2) is used for system development and programming the hardware with suitable application program.**

## 1. INTRODUCTION

Embedded design techniques tools are used now a day to design embedded processor system to be configured on FPGAs. Processor system usually consist a soft core processor (MicroBlaze) or hard processor (PowerPC), processor bus, memories and peripherals. The most commonly used buses in such a type of system are Processor Local Bus (PLB v4.6).

In [1] a DMA controller is designed to act with Micro blaze processor system configured on Spartan-3A FPGAs. The system is designed to perform data transfer between the internal block RAM an external peripheral.

In [2] a DMA system is depicted to act with multiprocessor connected via On-chip Processer Bus (OPB).

In [3] a DMA mode is proposed to act as a universal synchronous/ a synchronous Receiver/Transmitter (USART) IP soft core in Altera kit with AVALON bus.

In the work AXI processor bus is used instead of PLB processor bus with adding a MCB-DDR2 external memory.

AXI processor system must be adapted to deal with different data width due to application diversities to cope with this challenge the need for a flexible processor bus arose. Xilinx Company developed and advanced extensible bus interface (AXI) that is configurable on Spartan-6 FPGAs. Figure (1) shows the AXI interconnect core block diagram [4].

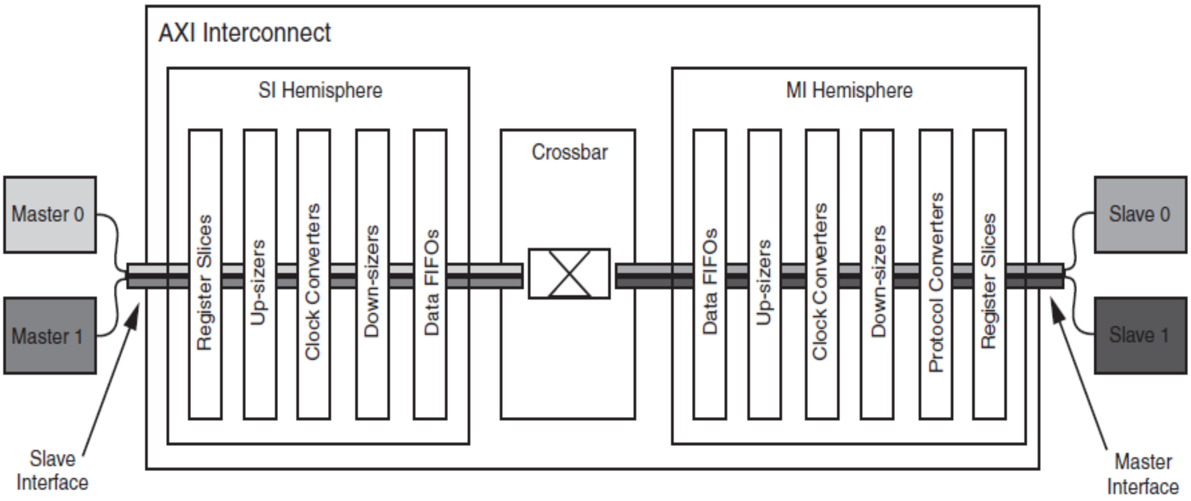


**Figure 1. AXI Interconnect Core Block Diagram**

The AXI Interconnect core consists of the SI (AXI Interconnect Slave Interface), the MI (AXI Interconnect Master Interface) and the functional units that comprise the AXI channel pathways between them. The SI accepts Write and Read transaction requests from connected master devices. The MI issues transactions to slave devices. At the center is the crossbar that routes traffic on all the AXI channels between the various devices connected to the SI and MI. The AXI Interconnect core also comprises other functional units located between the crossbar and each of the interfaces that perform various conversion and storage functions. The crossbar effectively splits the AXI Interconnect core down the middle between the SI-related functional units (SI hemisphere) and the MI-related units (MI hemisphere). [4]
The Central Direct Memory Access (AXI CDMA) core is a soft Xilinx IP core that can be introduced to the designed soft core processor system. The AXI CDMA provides high-bandwidth direct memory access (DMA) between a memory mapped source address and a memory mapped destination address using the AXI4 protocol. Initialization, status, and control registers are accessed through an AXI4-Lite slave interface, suitable for the Xilinx MicroBlaz microprocessor. Figure (2) shows the block diagram of AXI-CDMA Intellectual Property (IP) core [5].

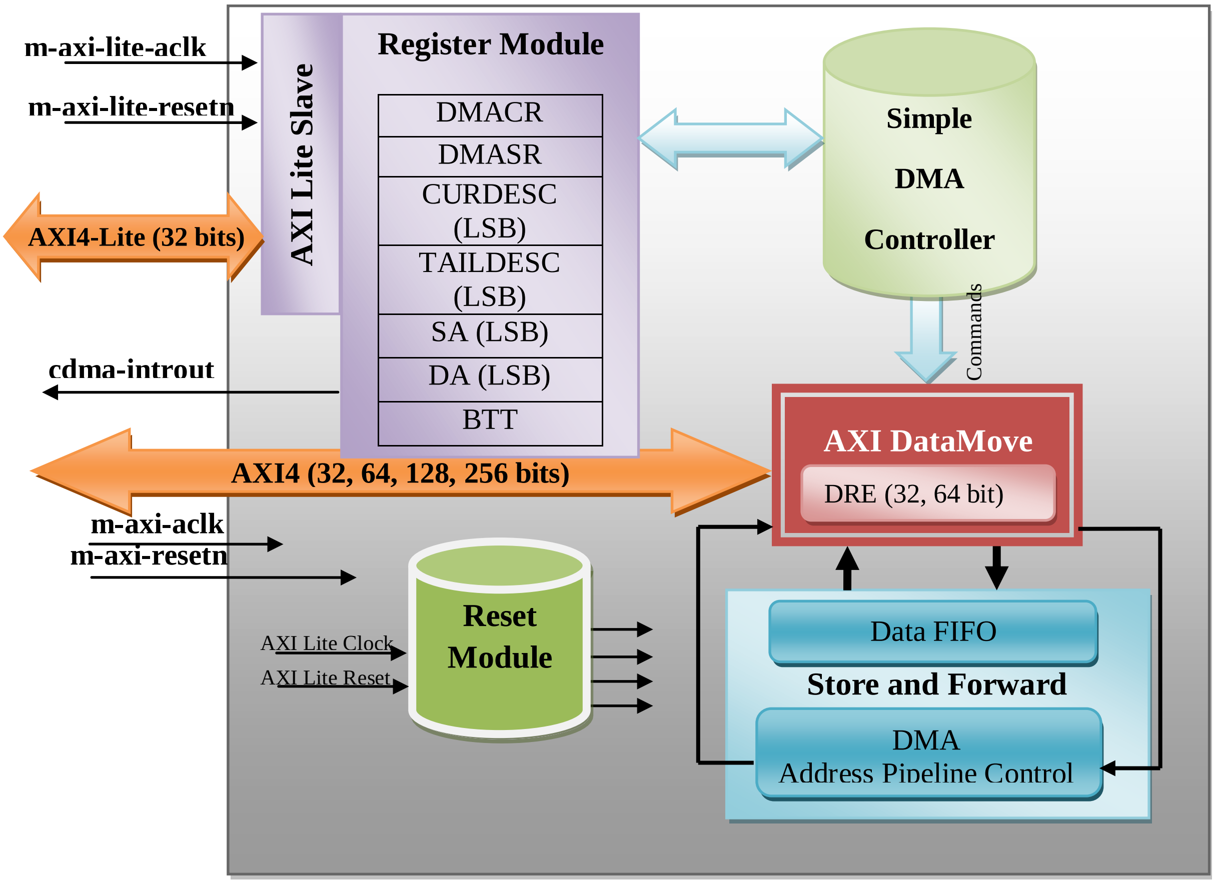


Figure 2. The Block Diagram of AXI-CDMA

The basic core design has an AXI4 Master interface for the main CDMA data transport function and an AXI4-Lite Slave interface for register accesses. The core's control and status registers are included in the Register Module. Access to these registers is provided through the AXI4-Lite Slave interface. The register module provides control and status for all CDMA operations. Primary high-speed CDMA data transport is provided by the AXI DataMover helper core. The AXI DataMover is used for high-throughput transfer of data from AXI4 to AXI4-Stream and from AXI4-Stream to AXI4. The DataMover provides CDMA operations with 4 kbyte address boundary protection, automatic burst partitioning, as well as providing the ability to queue multiple transfer requests. Furthermore, the AXI DataMover provides byte-level data realignment (for 32-bit and 64-bit data widths) allowing the CDMA to read from and write to any byte offset combination [5].
The AXI Centralized DMA provides the same simple transfer mode operation as the legacy PLBv4.6 Centralized DMA. A simple mode transfer is defined as that which the CPU programs the Centralized DMA register set for a single transfer and then initiates the transfer. The Centralized DMA:

- Performs the transfer
- Generates an interrupt when the transfer is complete
- Waits for the microprocessor to program and start the next transfer

Also, the AXI Centralized DMA includes an optional data realignment function for 32- and 64-bit bus widths. This feature allows addressing independence between the transfer source and destination addresses [6].
The suggested procedure in the work starts by constructing the embedded processor system, introducing the AXI-CDMA controller to the system and programming the resultant hardware using C language to accommodate the system to operate in CDMA mode. The results are displayed at Hyper Terminal media and real time chip scope window.

## 2. SYSTEM DESIGN

The system under consideration is designed with in two stages:

1) The hardware part of the soft processor system is constructed, and then the AXI-CDMA controller core is added to the system.
2) In the software part the resultant hardware is programmed by C language to make the system operate in a CDMA mode.

### A. Hardware Part Development

The hardware part includes the processor (MicroBlaze), AXI bus, CDMA unit, DDR2 Interface (128 MB), External Memory Controller (EMC), BRAM, Networking, Interrupt Controller, General purpose Input/output (GPIO), Floating Point Unit (FPU) and MicroBlaze Debug Module (MDM). It is developed in the Xilinx Platform Studio (XPS) and its component is defined in Microprocessor Hardware Specification (MHS) file, figure (3) displays the block diagram of the designed hardware part [5].

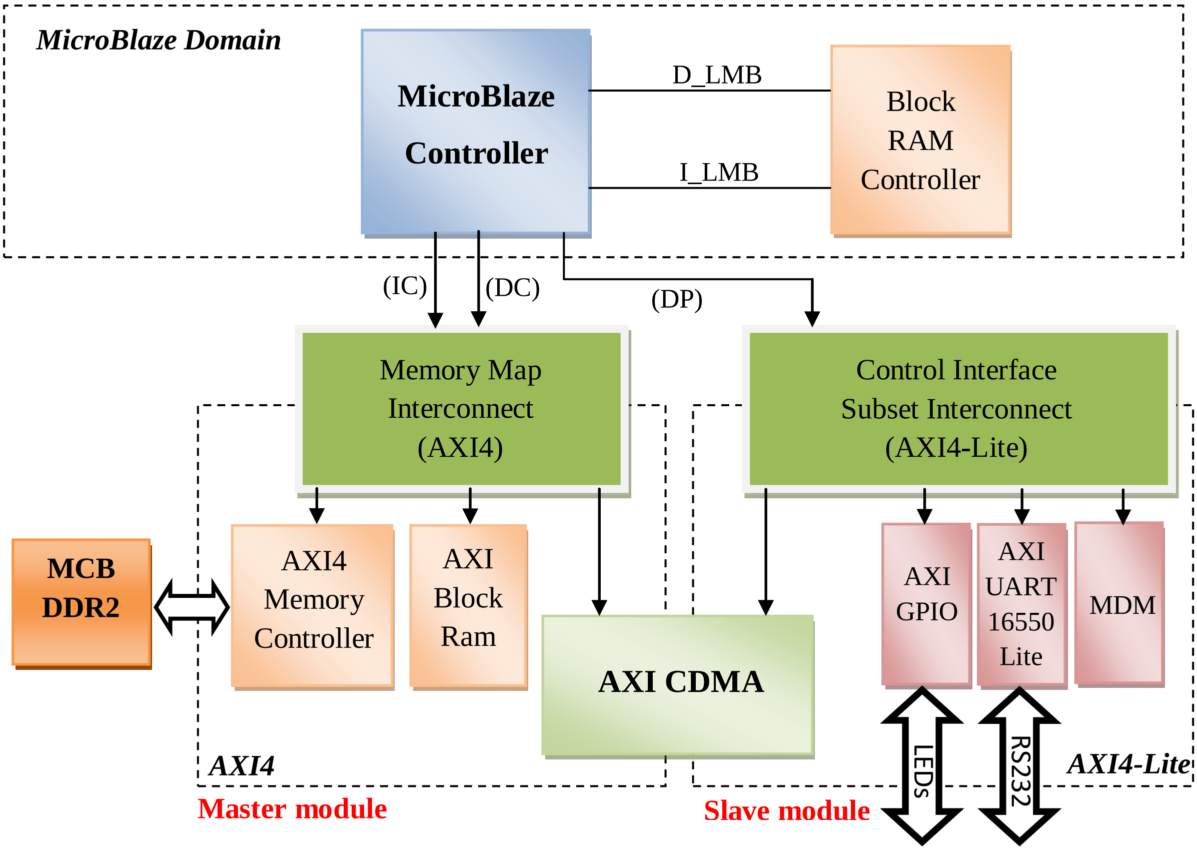


**Figure 3. The Block Diagram of the Designed System**

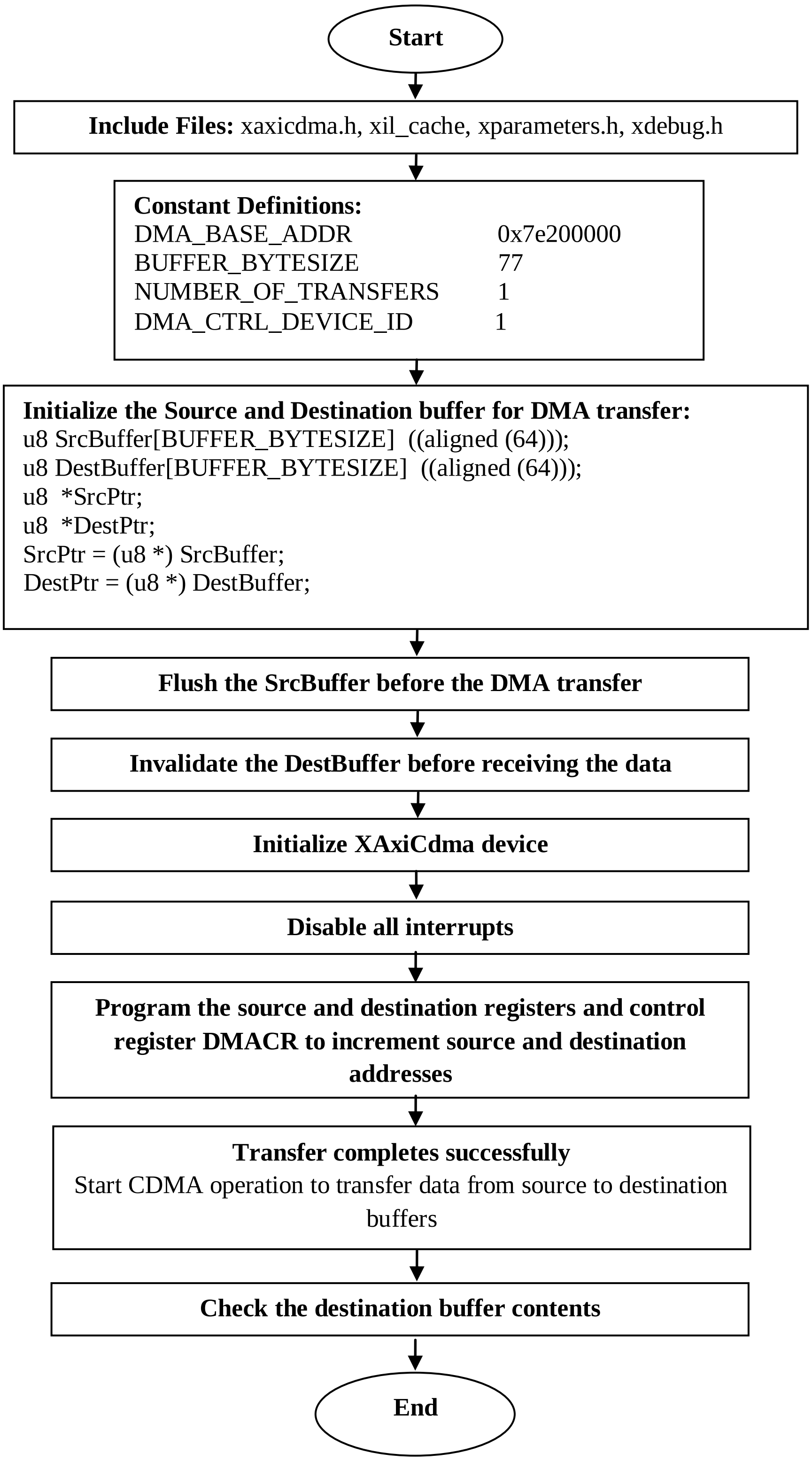
The CDMA IP core is added to the system using the following steps:

- Instantiate the IP core from Pcore files to the system.
- Making bus connections for master and slave part.
- Making port connections between the IP and the AXI bus.
- Configuring the IP module parameter to cope with the system criteria.

- Generating an address to the added IP within the system address map.
- Rescan the system to sense the newly added IP.

### B. Software Part Development

C- Program is developed to be run on the processor to make the system act in DMA mode. Figure (4) shows the flow chart of the prepared program.

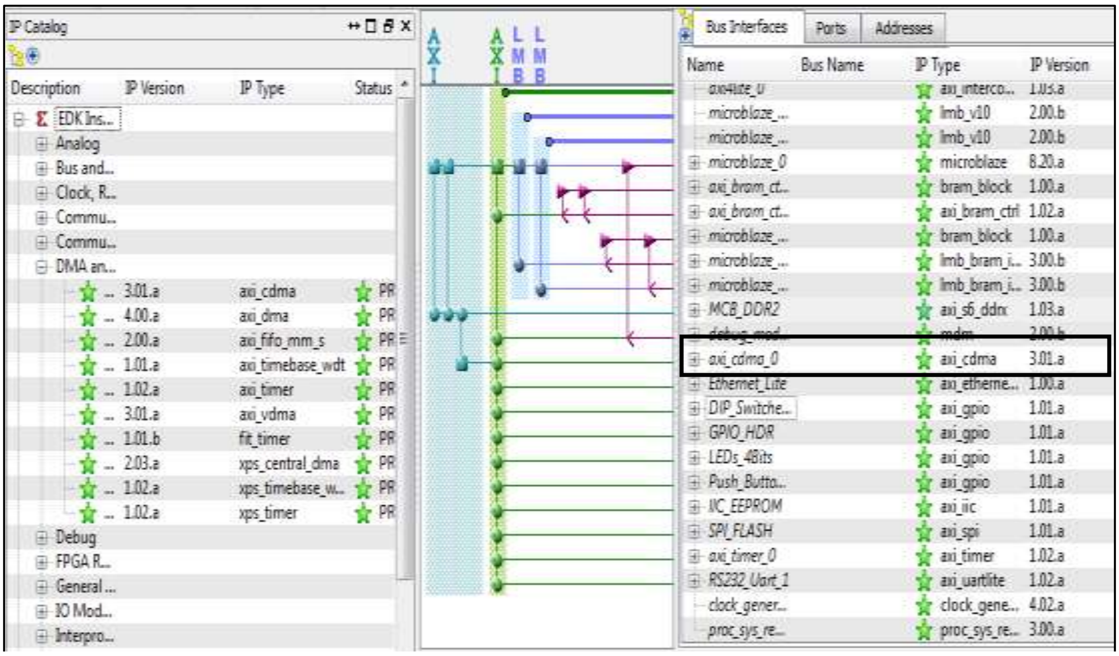

Start
Include Files: xaxicdma.h, xil_cache, xparameters.h, xdebug.h
Constant Definitions:
DMA_BASE_ADDR 0x7e200000
BUFFER_BYTESIZE 77
NUMBER_OF_TRANSFERS 1
DMA_CTRL_DEVICE_ID 1
Initialize the Source and Destination buffer for DMA transfer:
Flush the SrcBuffer before the DMA transfer
Invalidate the DestBuffer before receiving the data
Initialize XAxiCdma device
Disable all interrupts
Program the source and destination registers and control register DMACR to increment source and destination addresses
Transfer completes successfully
Start CDMA operation to transfer data from source to destination buffers
Check the destination buffer contents
End

```
u8 SrcBuffer[BUFFER_BYTESIZE]  ((aligned (64)));
u8 DestBuffer[BUFFER_BYTESIZE]  ((aligned (64)));
u8  *SrcPtr;
u8  *DestPtr;
SrcPtr = (u8 *) SrcBuffer;
DestPtr = (u8 *) DestBuffer;
```

**Figure 4. The flow chart of designed system programming**

## 3. RESULTS

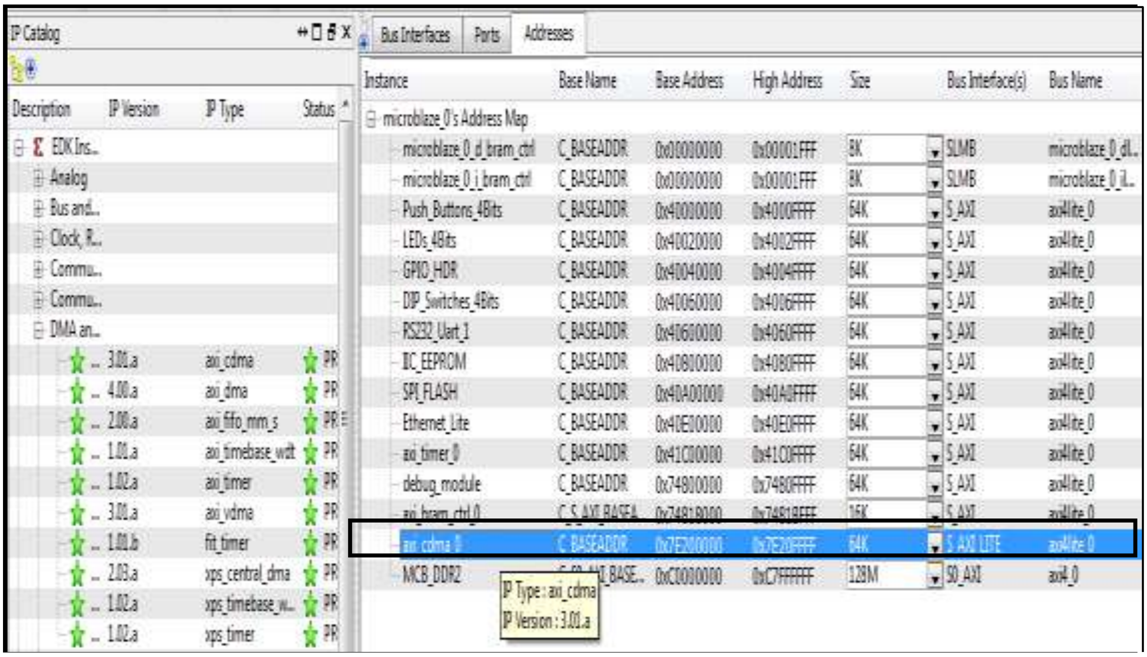
Figure (5) shown the resulting hardware part of the system, figure (5-a) shows the assembly view and figure (5-b) shows the address map of the designed system issued by the platform studio XPS.



| Description | IP Version | IP Type | Status |
|---|---|---|---|
| EDK Ins... | | | |
| Analog | | | |
| Bus and... | | | |
| Clock, R... | | | |
| Commu... | | | |
| Commu... | | | |
| DMA an... | | | |
| ... | 3.01.a | axi_cdma | PR |
| ... | 4.00.a | axi_dma | PR |
| ... | 2.00.a | axi_fifo_mm_s | PR |
| ... | 1.01.a | axi_timebase_wdt | PR |
| ... | 1.02.a | axi_timer | PR |
| ... | 3.01.a | axi_vdma | PR |
| ... | 1.01.b | fit_timer | PR |
| ... | 2.03.a | xps_central_dma | PR |
| ... | 1.02.a | xps_timebase_w... | PR |
| ... | 1.02.a | xps_timer | PR |
| Debug | | | |
| FPGA R... | | | |
| General ... | | | |
| IO Mod... | | | |
| Interpro... | | | |

| Name | Bus Name | IP Type | IP Version |
|---|---|---|---|
| axi4lite_0 | | axi_interco... | 1.03.a |
| microblaze_... | | lmb_v10 | 2.00.b |
| microblaze_... | | lmb_v10 | 2.00.b |
| microblaze_0 | | microblaze | 8.20.a |
| axi_bram_ct... | | bram_block | 1.00.a |
| axi_bram_ct... | | axi_bram_ctrl | 1.02.a |
| microblaze_... | | bram_block | 1.00.a |
| microblaze_... | | lmb_bram_i... | 3.00.b |
| microblaze_... | | lmb_bram_i... | 3.00.b |
| MCB_DDR2 | | axi_s6_ddrx | 1.03.a |
| debug_mod... | | mdm | 2.00.b |
| axi_cdma_0 | | axi_cdma | 3.01.a |
| Ethernet_Lite | | axi_etherne... | 1.00.a |
| DIP_Switche... | | axi_gpio | 1.01.a |
| GPIO_HDR | | axi_gpio | 1.01.a |
| LEDs_4Bits | | axi_gpio | 1.01.a |
| Push_Butto... | | axi_gpio | 1.01.a |
| IIC_EEPROM | | axi_iic | 1.01.a |
| SPI_FLASH | | axi_spi | 1.01.a |
| axi_timer_0 | | axi_timer | 1.02.a |
| RS232_Uart_1 | | axi_uartlite | 1.02.a |
| clock_gener... | | clock_gene... | 4.02.a |
| proc_sys_re... | | proc_sys_re... | 3.00.a |

**a**



| Description | IP Version | IP Type | Status |
|---|---|---|---|
| EDK Ins... | | | |
| Analog | | | |
| Bus and... | | | |
| Clock, R... | | | |
| Commu... | | | |
| Commu... | | | |
| DMA an... | | | |
| ... | 3.01.a | axi_cdma | PR |
| ... | 4.00.a | axi_dma | PR |
| ... | 2.00.a | axi_fifo_mm_s | PR |
| ... | 1.01.a | axi_timebase_wdt | PR |
| ... | 1.02.a | axi_timer | PR |
| ... | 3.01.a | axi_vdma | PR |
| ... | 1.01.b | fit_timer | PR |
| ... | 2.03.a | xps_central_dma | PR |
| ... | 1.02.a | xps_timebase_w... | PR |
| ... | 1.02.a | xps_timer | PR |

| Instance | Base Name | Base Address | High Address | Size | Bus Interface(s) | Bus Name |
|---|---|---|---|---|---|---|
| microblaze_0's Address Map | | | | | | |
| microblaze_0_d_bram_ctrl | C_BASEADDR | 0x00000000 | 0x00001FFF | 8K | SLMB | microblaze_0_dl... |
| microblaze_0_i_bram_ctrl | C_BASEADDR | 0x00000000 | 0x00001FFF | 8K | SLMB | microblaze_0_il... |
| Push_Buttons_4Bits | C_BASEADDR | 0x40000000 | 0x4000FFFF | 64K | S_AXI | axi4lite_0 |
| LEDs_4Bits | C_BASEADDR | 0x40020000 | 0x4002FFFF | 64K | S_AXI | axi4lite_0 |
| GPIO_HDR | C_BASEADDR | 0x40040000 | 0x4004FFFF | 64K | S_AXI | axi4lite_0 |
| DIP_Switches_4Bits | C_BASEADDR | 0x40060000 | 0x4006FFFF | 64K | S_AXI | axi4lite_0 |
| RS232_Uart_1 | C_BASEADDR | 0x40600000 | 0x4060FFFF | 64K | S_AXI | axi4lite_0 |
| IIC_EEPROM | C_BASEADDR | 0x40800000 | 0x4080FFFF | 64K | S_AXI | axi4lite_0 |
| SPI_FLASH | C_BASEADDR | 0x40A00000 | 0x40A0FFFF | 64K | S_AXI | axi4lite_0 |
| Ethernet_Lite | C_BASEADDR | 0x40E00000 | 0x40E0FFFF | 64K | S_AXI | axi4lite_0 |
| axi_timer_0 | C_BASEADDR | 0x41C00000 | 0x41C0FFFF | 64K | S_AXI | axi4lite_0 |
| debug_module | C_BASEADDR | 0x74800000 | 0x7480FFFF | 64K | S_AXI | axi4lite_0 |
| axi_bram_ctrl_0 | C_S_AXI_BASEA... | 0x74818000 | 0x7481BFFF | 16K | S_AXI | axi4lite_0 |
| axi_cdma_0 | C_BASEADDR | 0x7E200000 | 0x7E20FFFF | 64K | S_AXI_LITE | axi4lite_0 |
| MCB_DDR2 | C_S0_AXI_BASE... | 0xC0000000 | 0xC7FFFFFF | 128M | S0_AXI | axi4_0 |

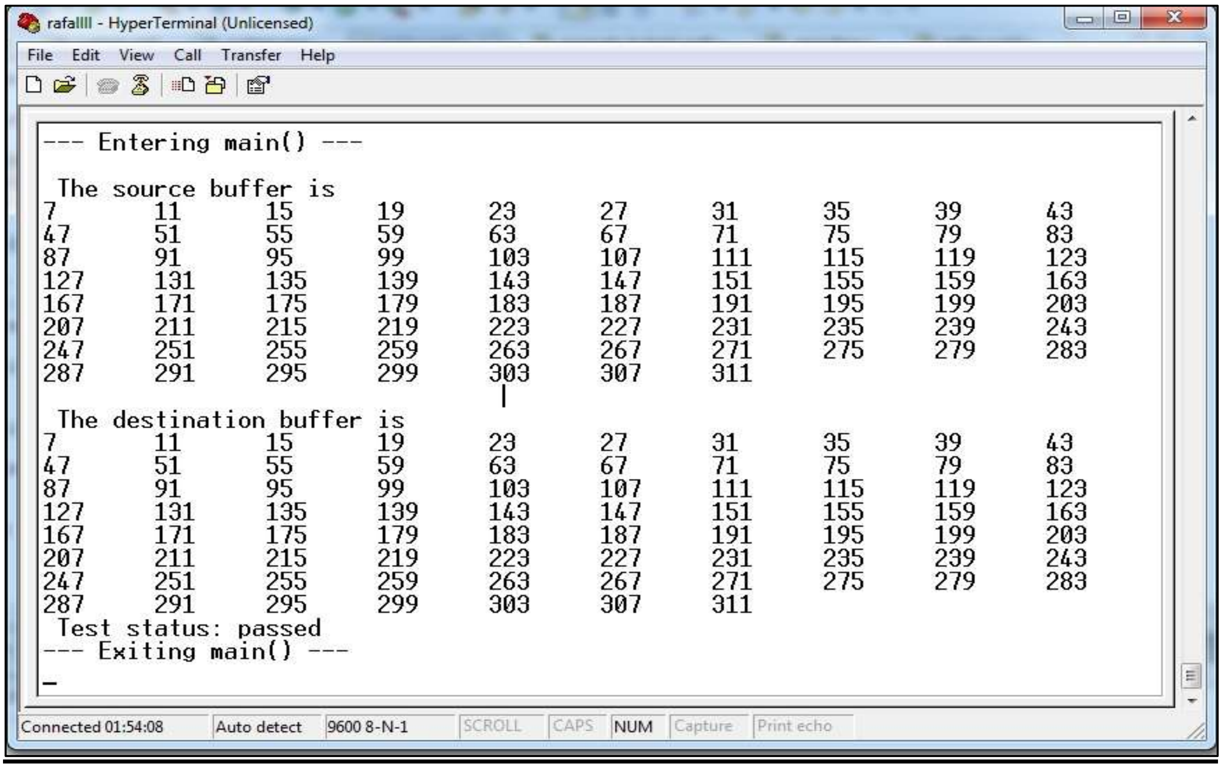
IP Type : axi_cdma
IP Version : 3.01.a

**b**

**Figure 5. The hardware part of designed soft core when adding DMA core**

a. **The assembly view of the system.**
b. **The address map of the system.**

Figure (6) shows the data read from destination buffer and the data transferred from the source buffer displayed in hyper terminal window.


rafallll - HyperTerminal (Unlicensed)
File Edit View Call Transfer Help

```
--- Entering main() ---

 The source buffer is
7       11      15      19      23      27      31      35      39      43
47      51      55      59      63      67      71      75      79      83
87      91      95      99      103     107     111     115     119     123
127     131     135     139     143     147     151     155     159     163
167     171     175     179     183     187     191     195     199     203
207     211     215     219     223     227     231     235     239     243
247     251     255     259     263     267     271     275     279     283
287     291     295     299     303     307     311

 The destination buffer is
7       11      15      19      23      27      31      35      39      43
47      51      55      59      63      67      71      75      79      83
87      91      95      99      103     107     111     115     119     123
127     131     135     139     143     147     151     155     159     163
167     171     175     179     183     187     191     195     199     203
207     211     215     219     223     227     231     235     239     243
247     251     255     259     263     267     271     275     279     283
287     291     295     299     303     307     311
 Test status: passed
--- Exiting main() ---
_
```

**Figure 6. The data transfer from source to destination buffer**

Figure (7) shows the operation of data transfer displayed on chip scope window during write data bus cycle.

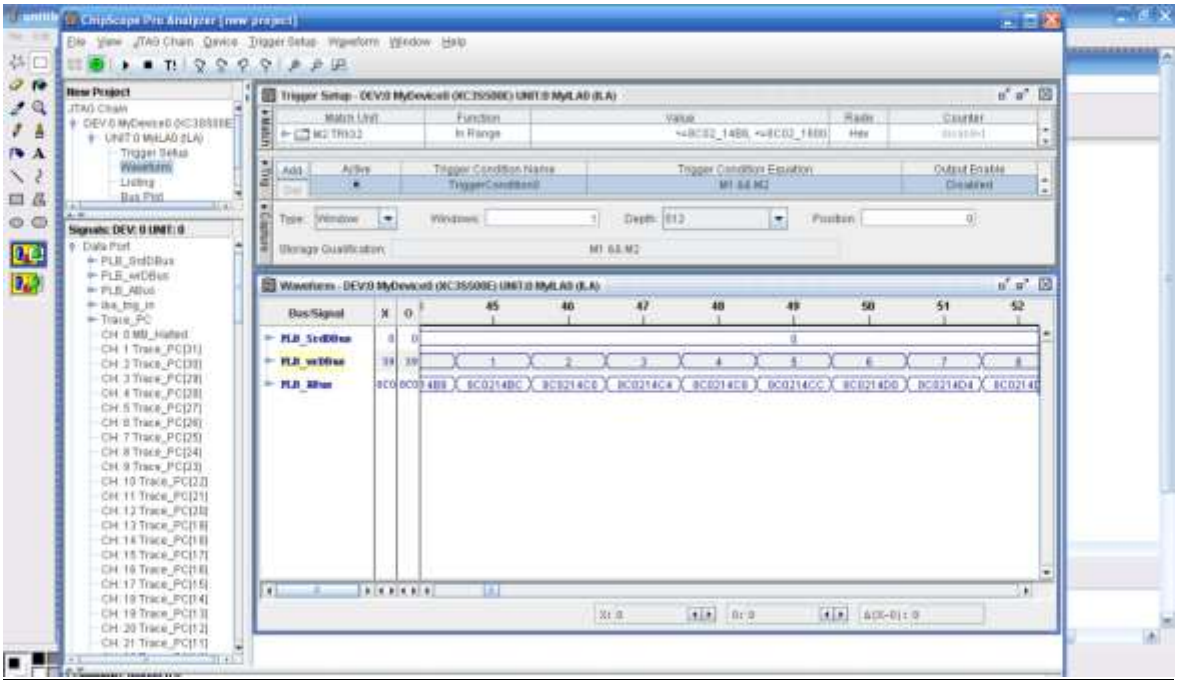


**Figure 7. Data transfer based on CDMA operation displayed on chip scope window**

## 4. Conclusions

Adding extensible processor bus (AXI) offers the DMA based processor system high flexibility as the data width between the source and destination peripherals will not be restricted to fixed widths. The data width between destination and source can be (32/32, 32/64, 32/128, 64/32, 64/64, 64/128, 128/128, 128/64, 128/32).
AXI also provides facility to use different source/destination data flow rates. This flexibility enables the processor system to be used with various applications.

## 5. References

[1]. Bakshi A. B., Burman A. B., & Chakraborty A. Ch. , 2014 , "Development of DMA controller for real time data processing in FPGA based embedded application", IOSR Journal of VLSI and Signal Processing (IOSR-JVSP) . e-ISSN: 2319-4200, p-ISSN No.: 2319-4197, www.iosrjournals.org, PP 01-08 .

[2]. Tumeo A. T., Monchiero M. M., Palermo G. P., Ferrandi F. F. & Sciuto D. S., 2008, "Lightweight DMA Management Mechanisms for Multiprocessors on FPGA", 1-4244-1898-5/08/$20.00 IEEE .

[3]. Allam P. A., 2013, "Design And Implementation Of USART IP Soft Core Based On DMA Mode", International Journal of Computer Trends and Technology (IJCTT), ISSN:2231-2803, http://www.ijcttjourmal.org, page no.:3580-3584.

[4]. XILINX, 2011, " LogiCORE IP AXI Interconnect (v1.03.a)" , web site: www.xilinx.com, DS768.

[5]. XILINX, 2011, " LogiCORE IP AXI Central Direct Memory Access (axi_cdma) (v3.01.a)" , web site: www.xilinx.com, DS792.

[6]. XILINX, 2011, " AXI Reference Guide" , web site: www.xilinx.com, UG761.