Designing of Soft Core Processor System with Direct Memory Access (DMA) Mode

Mazin Rejab Khalil **Rafal Taha Mahmood**

Basrah University. College of Engineering

E-mail: mazin_r_khalil@yahoo.com **E-mail: rafal_1985_r@yahoo.com**

Abstract

A soft core processor system is constructed using embedded design techniques and it is configured on Field Programmable Gate Arrays (FPGAs). The system is accommodated to act with Direct Memory Access (DMA) mode using suitable Xilinx Intellectual Property (IP) core. A dual data rate synchronous dynamic random access memory (DDR_SDRAM) with 64 Mbyte capacity is introduced to the system and accessed by the DMA controller. The controller is performed to transfer programmable quantity of data from source address to destination address without intervention of the processor.

Spartan-3E slice is used and programmed using Xilinx Platform Studio (XPS) which is provided by Xilinx integrated software environment at (ISE 10.1). The system performance is tested by transferring data from matlab media to the DDR_SDRAM and vice-versa, mat lab 2012a version software is used for this type of data transfer.

تصميم نظام المعالج المصغر ذات النواة المبرمجة بتقنية الوصول المباشر للذاكرة

مازن رجب خليل **رفل طه محمود**

المستخلص :

تم تصميم نظام معالج ذات نواة مبرمجة باستخدام تقنيات الانظمة المطمورة لينفذ على البوابات المنطقية القابلة للبرمجة الحقلية وبنظام الوصول العشوائي للذاكرة . تم ربط ذاكرة للوصول العشوائي نوع DDR_SDRAM بسعة ٦٤ ميكا بايت بنظام المعالج المصغر لغرض استخدامها في نظام الوصول المباشر DMA وتم اضافة مسيطر على الذاكرة بشكل نواة قابلة للبرمجة لتمكين النظام من العمل بطريقة الوصول المباشر للذاكرة وذلك لنقل البيانات بين الذاكرة (كمصدر) والأجهزة المحيطة (كهدف) للوصول وبالعكس . تم استخدام شريحة Spartan_3e وبرمجتها باستخدام بيئة ISE10.1 الصادرة من شركة Xilinx وتم اختبار أداء النظام باستخدام بيئة matlab اصدار ٢٠١٢ لنقل البيانات بين الذاكرة وبيئة matlab وبالعكس بطريقة الوصول المباشر للذاكرة .

1. Introduction

Direct memory access (DMA) system is used usually to transfer certain quantity of data between source and destination address without processor intervention.

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 and 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 this work a DDR-SDRAM external memory is used instead of the limited capacity internal block RAM with newest version of Processor Local Bus (PLB v4.6). A communication interaction between a matlab media and the designed soft core processor system is suggested to transfer data between them according to DMA techniques.

DMA is a feature of modern computers that allows certain hardware subsystems within the computer to access system memory for reading and/or writing independently of the central processing unit. Computers that have DMA channels can transfer data to and from devices with much less CPU overhead than computers without a DMA channel [3].

The processing unit which controls the DMA process is known as DMA controller. Typically the job of the DMA controller is to setup a connection between the memory unit and the I/O device; the data can be transferred with much less processor overhead. Figure (1) shows the block diagram of DMA operation. When an interrupt signal is activated, the processer goes to idle case and open circuit its connection with buses. The buses become under the control of the DMA controller[4].

The XPS Central DMA Controller operates on the PLB using independent master and slave interfaces. It responds as a slave when its registers are being read and written. It initiates read and write transactions as a master when a DMA operation is in progress. The master and slave connections of the XPS Central DMA operate as 32-bit PLB agents. However, either the master or slave can connect to a PLB with wider data paths (64-bit or 128-bit) and conduct transactions with wider slaves or masters[5].

DMA Operation forwards fast data transfer between source and destination compared with data transfer with processor intervention.
Figure (2) shows the block diagram of the DMA controller core [5].

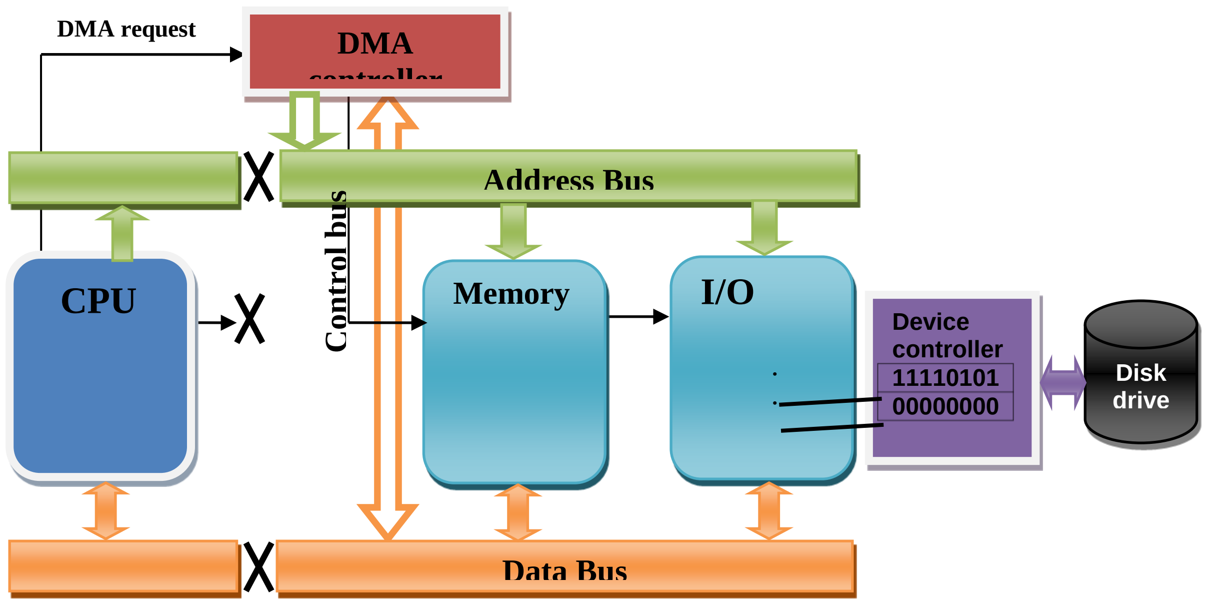


Figure 1: Operation of a DMA Controller [4]

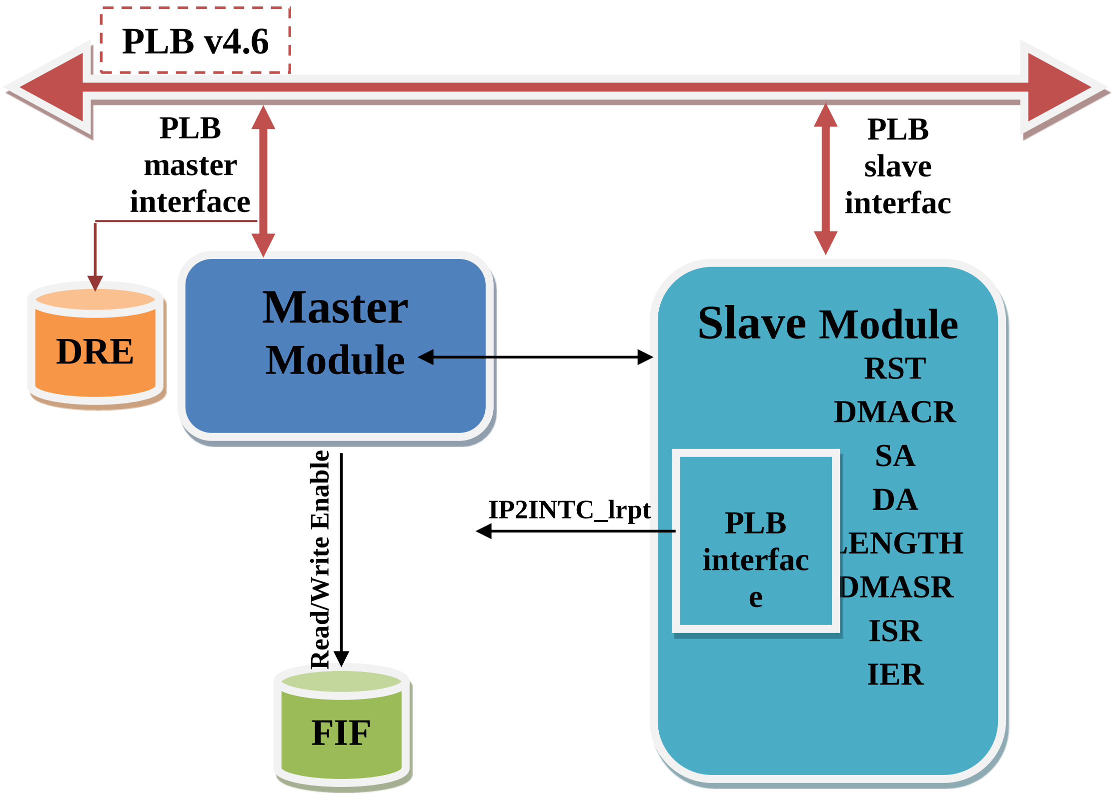


Figure (2): The Block Diagram of the DMA Controller Core

The core is composed of three modules; slave attachment module, master attachment module and memory buffer.

In the slave attachment module the DMA responds to PLB transactions to read and write the DMA registers to modify source address, destination address, length of data, DMA status and interrupt status when DMA operation proceeds. These modifications are performed by using the Source Address register (SA), the Destination Address register (DA), the Length Register (LR), the DMA Status Register (DMASR), the Interrupt Status Register (ISR), the Interrupt Enable Register (IER) and DMA Control Register (DMACR).

In the master attachment module, the DMA performs read and write transactions as a PLB master to transfer the amount of data specified in the length register from source address to destination address with updating the source, destination, length and status registers during the DMA transfer. The memory buffer is 16*32 internal data buffer that is used to support PLB burst transfer to speed up the DMA operation [5].

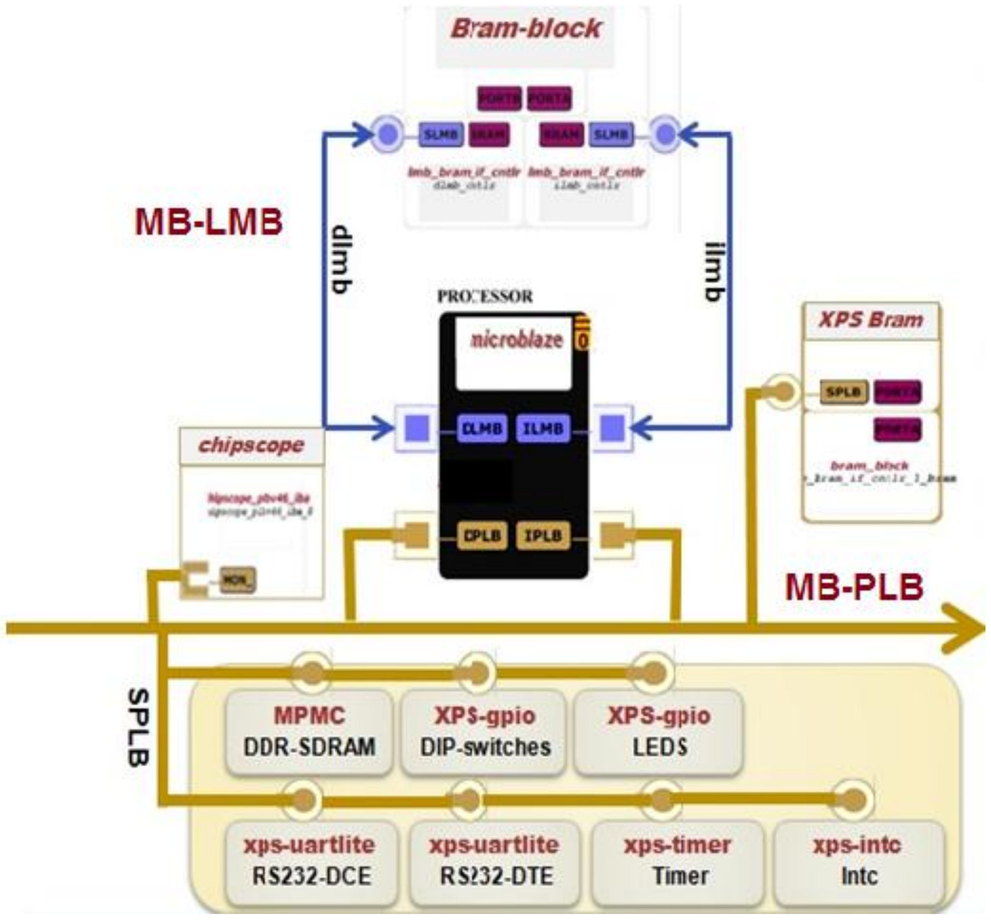
The suggested procedure in this work starts by constructing the embedded processor system, introducing the DMA controller to the system and programming the resultant hardware using C-language to accommodate the system to operate in DMA mode. The system is tested to verify its functionality by transferring data between matlab media and the designed processor system; the results are displayed at Hyper Terminal media and real time chip scope window.

2. System Design

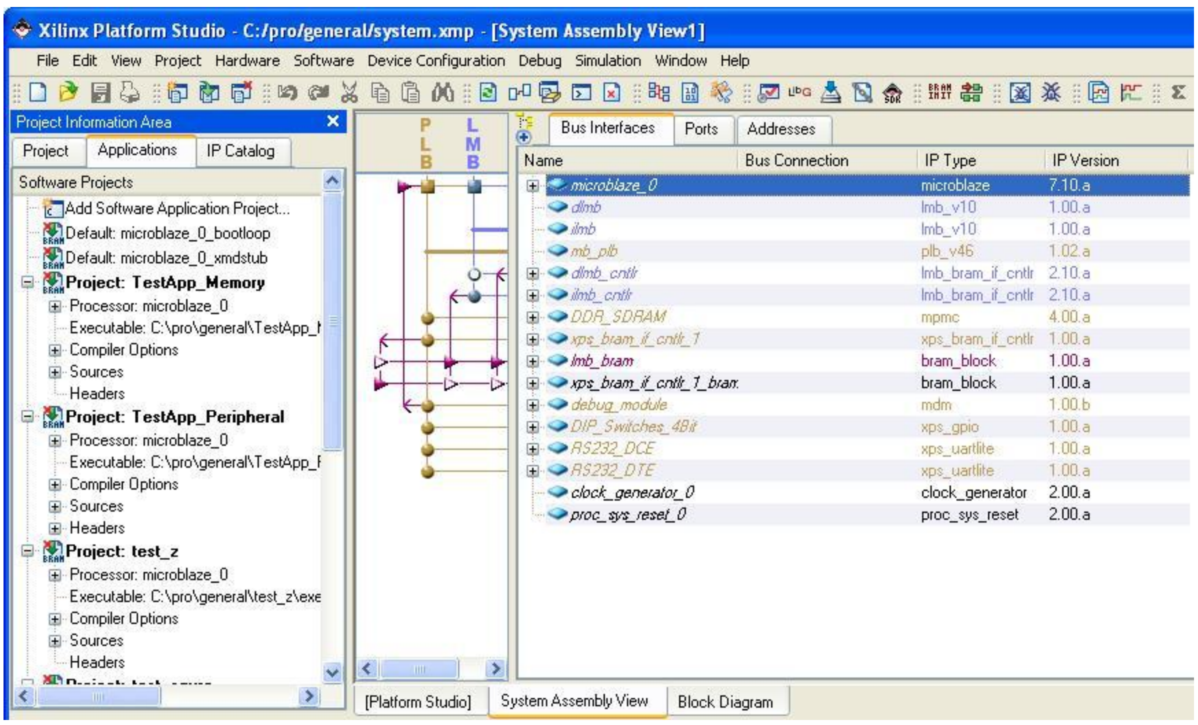
The system under consideration is designed using three stages. In the first stage the hardware part of the soft processor system is constructed. While in the second stage a DMA controller core is added to the system. Finally in the third stage the resultant system is programmed by C-language to operate in a DMA mode.

3. Soft Core Processor System Design

Using embedded design techniques [6], a soft core processor system as shown in Figure (3) is designed using the platform studio provided by Xilinx ISE (10.1) software. Figure (3-a) shows the block diagram of the hardware part , Figure (3-b) exhibits the assembly view and Figure (3-c) displays the address map of the system .



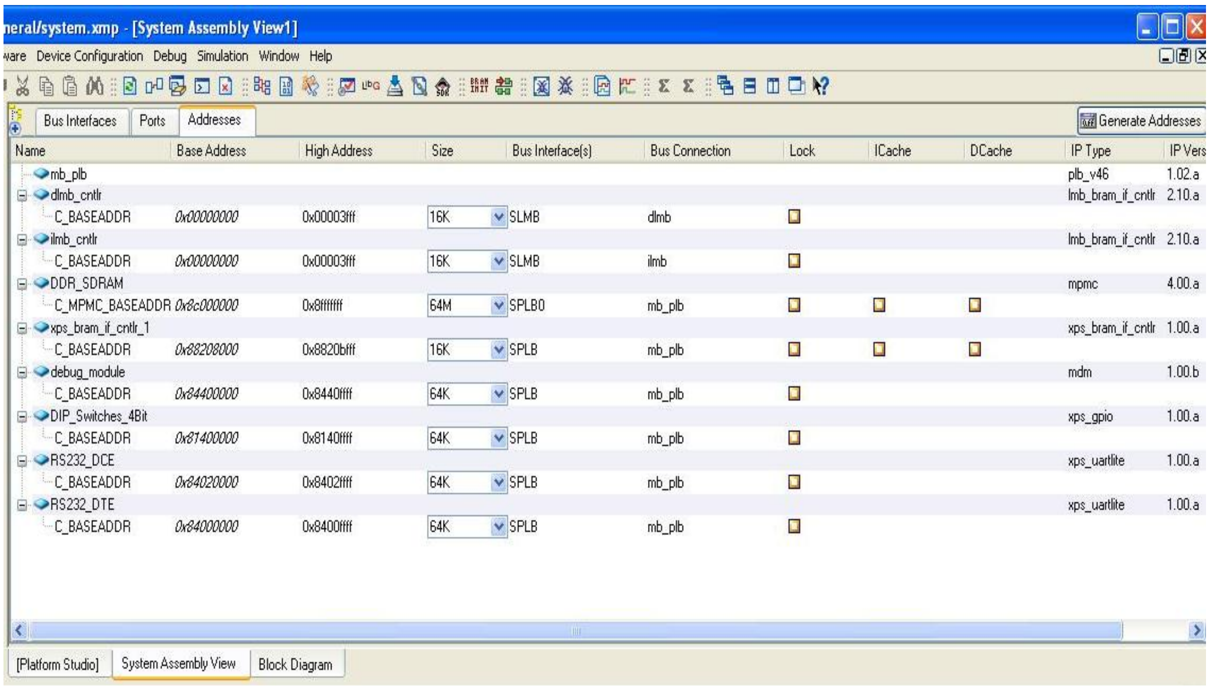
a


Xilinx Platform Studio - C:/pro/general/system.xmp - [System Assembly View1]
File Edit View Project Hardware Software Device Configuration Debug Simulation Window Help
Project Information Area
Project Applications IP Catalog
PLB LMB
Bus Interfaces Ports Addresses
[Platform Studio] System Assembly View Block Diagram

Software Projects
- Add Software Application Project...
- Default: microblaze_0_bootloop
- Default: microblaze_0_xmdstub
- Project: TestApp_Memory
  - Processor: microblaze_0
  - Executable: C:\pro\general\TestApp_I
  - Compiler Options
  - Sources
  - Headers
- Project: TestApp_Peripheral
  - Processor: microblaze_0
  - Executable: C:\pro\general\TestApp_I
  - Compiler Options
  - Sources
  - Headers
- Project: test_z
  - Processor: microblaze_0
  - Executable: C:\pro\general\test_z\exe
  - Compiler Options
  - Sources
  - Headers

Name	Bus Connection	IP Type	IP Version
microblaze_0		microblaze	7.10.a
dlmb		lmb_v10	1.00.a
ilmb		lmb_v10	1.00.a
mb_plb		plb_v46	1.02.a
dlmb_cntlr		lmb_bram_if_cntlr	2.10.a
ilmb_cntlr		lmb_bram_if_cntlr	2.10.a
DDR_SDRAM		mpmc	4.00.a
xps_bram_if_cntlr_1		xps_bram_if_cntlr	1.00.a
lmb_bram		bram_block	1.00.a
xps_bram_if_cntlr_1_bram		bram_block	1.00.a
debug_module		mdm	1.00.b
DIP_Switches_4Bit		xps_gpio	1.00.a
RS232_DCE		xps_uartlite	1.00.a
RS232_DTE		xps_uartlite	1.00.a
clock_generator_0		clock_generator	2.00.a
proc_sys_reset_0		proc_sys_reset	2.00.a

b


neral/system.xmp - [System Assembly View1]
ware Device Configuration Debug Simulation Window Help
Bus Interfaces
Ports
Addresses
Generate Addresses
[Platform Studio]
System Assembly View
Block Diagram

Name	Base Address	High Address	Size	Bus Interface(s)	Bus Connection	Lock	ICache	DCache	IP Type	IP Vers
mb_plb									plb_v46	1.02.a
dlmb_cntlr									lmb_bram_if_cntlr	2.10.a
C_BASEADDR	*0x00000000*	0x00003fff	16K	SLMB	dlmb	☐				
ilmb_cntlr									lmb_bram_if_cntlr	2.10.a
C_BASEADDR	*0x00000000*	0x00003fff	16K	SLMB	ilmb	☐				
DDR_SDRAM									mpmc	4.00.a
C_MPMC_BASEADDR	*0x8c000000*	0x8fffffff	64M	SPLB0	mb_plb	☐	☐	☐		
xps_bram_if_cntlr_1									xps_bram_if_cntlr	1.00.a
C_BASEADDR	*0x88208000*	0x8820bfff	16K	SPLB	mb_plb	☐	☐	☐		
debug_module									mdm	1.00.b
C_BASEADDR	*0x84400000*	0x8440ffff	64K	SPLB	mb_plb	☐				
DIP_Switches_4Bit									xps_gpio	1.00.a
C_BASEADDR	*0x81400000*	0x8140ffff	64K	SPLB	mb_plb	☐				
RS232_DCE									xps_uartlite	1.00.a
C_BASEADDR	*0x84020000*	0x8402ffff	64K	SPLB	mb_plb	☐				
RS232_DTE									xps_uartlite	1.00.a
C_BASEADDR	*0x84000000*	0x8400ffff	64K	SPLB	mb_plb	☐				

c

Figure (3): The Hardware Part of Designed Soft Core Processor System

a. **The block diagram of the system.**

b. **The assembly view of the system.**

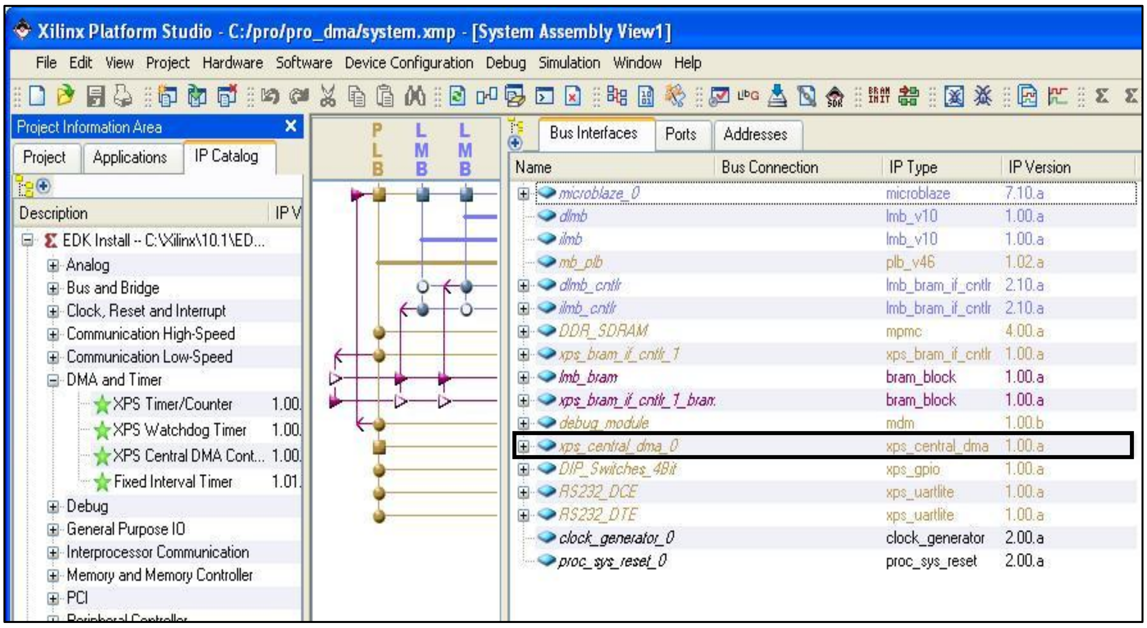
c. **The address map of the system.**

4. Adding DMA Controller

The platform studio provides an environment in which an available IP cores can be accessed. The DMA controller core is available in the form IP module that can be dragged from the IP catalogue to the system assembly view. The following steps are adopted to perform successful DMA introduction to the system:

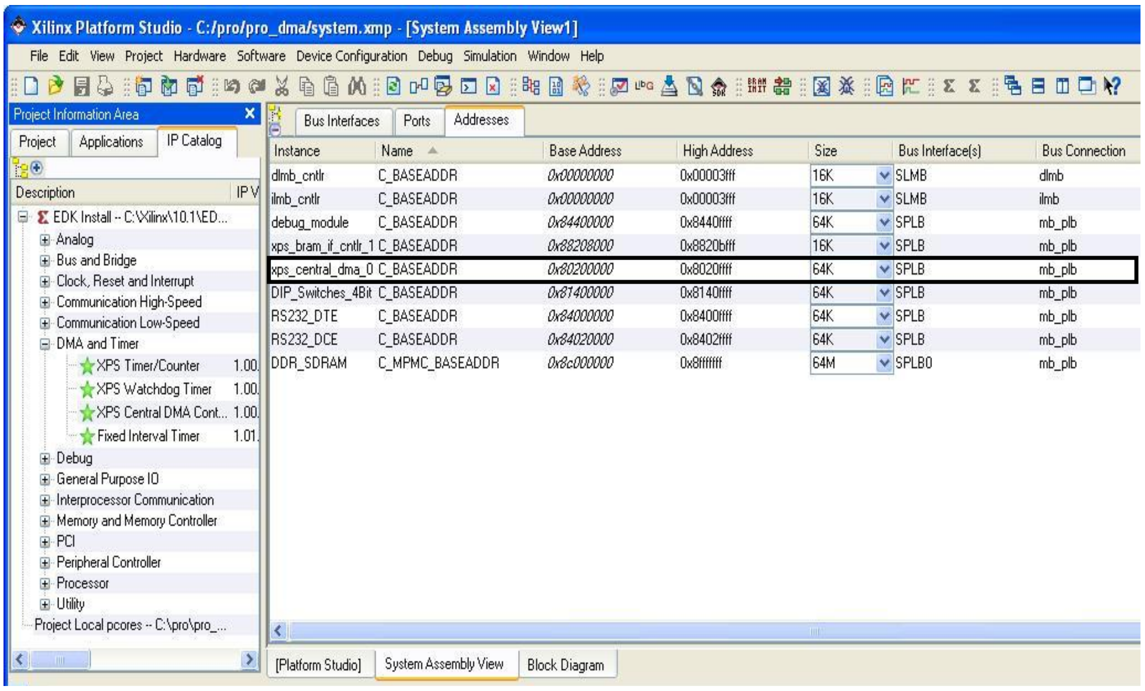
1. Using bus interface window the DMA core is connected to the PLB in the slave and master module and the core parameters are customized to adapt with the processor system.
2. Using the port window the interrupt port of the controller is connected to the interrupt port of the processor.
3. Using the address window, the address map of the system is reconfigured to take the DMA controller into consideration.

4. The resulting hardware part of the system is shown in Figure (4). Figure (4-a) shows the assembly view of the system with DMA core, Figure (4-b) shows the address map of the system with DMA core.


Xilinx Platform Studio - C:/pro/pro_dma/system.xmp - [System Assembly View1]
File Edit View Project Hardware Software Device Configuration Debug Simulation Window Help
Project Information Area
Project Applications IP Catalog
Description IP V
EDK Install -- C:\Xilinx\10.1\ED...
Analog
Bus and Bridge
Clock, Reset and Interrupt
Communication High-Speed
Communication Low-Speed
DMA and Timer
XPS Timer/Counter 1.00
XPS Watchdog Timer 1.00
XPS Central DMA Cont... 1.00
Fixed Interval Timer 1.01
Debug
General Purpose IO
Interprocessor Communication
Memory and Memory Controller
PCI
PLB
LMB
LMB
Bus Interfaces Ports Addresses

Name	Bus Connection	IP Type	IP Version
microblaze_0		microblaze	7.10.a
dlmb		lmb_v10	1.00.a
ilmb		lmb_v10	1.00.a
mb_plb		plb_v46	1.02.a
dlmb_cntlr		lmb_bram_if_cntlr	2.10.a
ilmb_cntlr		lmb_bram_if_cntlr	2.10.a
DDR_SDRAM		mpmc	4.00.a
xps_bram_if_cntlr_1		xps_bram_if_cntlr	1.00.a
lmb_bram		bram_block	1.00.a
xps_bram_if_cntlr_1_bram		bram_block	1.00.a
debug_module		mdm	1.00.b
xps_central_dma_0		xps_central_dma	1.00.a
DIP_Switches_4Bit		xps_gpio	1.00.a
RS232_DCE		xps_uartlite	1.00.a
RS232_DTE		xps_uartlite	1.00.a
clock_generator_0		clock_generator	2.00.a
proc_sys_reset_0		proc_sys_reset	2.00.a

a


Xilinx Platform Studio - C:/pro/pro_dma/system.xmp - [System Assembly View1]
File Edit View Project Hardware Software Device Configuration Debug Simulation Window Help
Project Information Area
Project Applications IP Catalog
Description IP V
EDK Install -- C:\Xilinx\10.1\ED...
Analog
Bus and Bridge
Clock, Reset and Interrupt
Communication High-Speed
Communication Low-Speed
DMA and Timer
XPS Timer/Counter 1.00
XPS Watchdog Timer 1.00
XPS Central DMA Cont... 1.00
Fixed Interval Timer 1.01
Debug
General Purpose IO
Interprocessor Communication
Memory and Memory Controller
PCI
Peripheral Controller
Processor
Utility
Project Local pcores -- C:\pro\pro_...
Bus Interfaces Ports Addresses
[Platform Studio] System Assembly View Block Diagram

Instance	Name	Base Address	High Address	Size	Bus Interface(s)	Bus Connection
dlmb_cntlr	C_BASEADDR	0x00000000	0x00003fff	16K	SLMB	dlmb
ilmb_cntlr	C_BASEADDR	0x00000000	0x00003fff	16K	SLMB	ilmb
debug_module	C_BASEADDR	0x84400000	0x8440ffff	64K	SPLB	mb_plb
xps_bram_if_cntlr_1	C_BASEADDR	0x88208000	0x8820bfff	16K	SPLB	mb_plb
xps_central_dma_0	C_BASEADDR	0x80200000	0x8020ffff	64K	SPLB	mb_plb
DIP_Switches_4Bit	C_BASEADDR	0x81400000	0x8140ffff	64K	SPLB	mb_plb
RS232_DTE	C_BASEADDR	0x84000000	0x8400ffff	64K	SPLB	mb_plb
RS232_DCE	C_BASEADDR	0x84020000	0x8402ffff	64K	SPLB	mb_plb
DDR_SDRAM	C_MPMC_BASEADDR	0x8c000000	0x8fffffff	64M	SPLB0	mb_plb

b

Figure (4): 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.

5. Programming the system

The C -language is used to program the resultant system hardware to operate in DMA mode, Figure (5) shows the flow chart of the prepared program. Application Peripheral Interfaces (API) are used to make the hardware peripheral be sensed by the C –language compiler. The APIs are software drivers constructed in the form of C -language functions. The following APIs are used in the prepared program.

```
#define XDmaCentral_mWriteReg(BaseAddress, RegOffset, Data)
```

Where:

Base address: represents the base address of the DMA controller.

Offset address: the offset address of each register in the controller. The offset address of each register is shown in Table [1].

Data: the data request to program the register.

Table (1): XPS Central DMA Controller Registers [4]

Register Name	Base Address+ Offset(hex)	Default Value(hex)	Access
Software Reset Register (RST)	C_BASEADDR + 0	NA	Write
DMA Control Register (DMACR)	C_BASEADDR + 4	80000004	R/W
Source Address (SA)	C_BASEADDR + 8	00000000	R/W
Destination Address (DA)	C_BASEADDR +C	00000000	R/W
Length (LENGTH)	C_BASEADDR + 10	00000000	R/W
DMA Status Register (DMASR)	C_BASEADDR + 14	00000000	Read
Interrupt Status Register (ISR)	C_BASEADDR + 2C	00000000	Read/TOW
Interrupt Enable Register (IER)	C_BASEADDR + 30	00000000	R/W

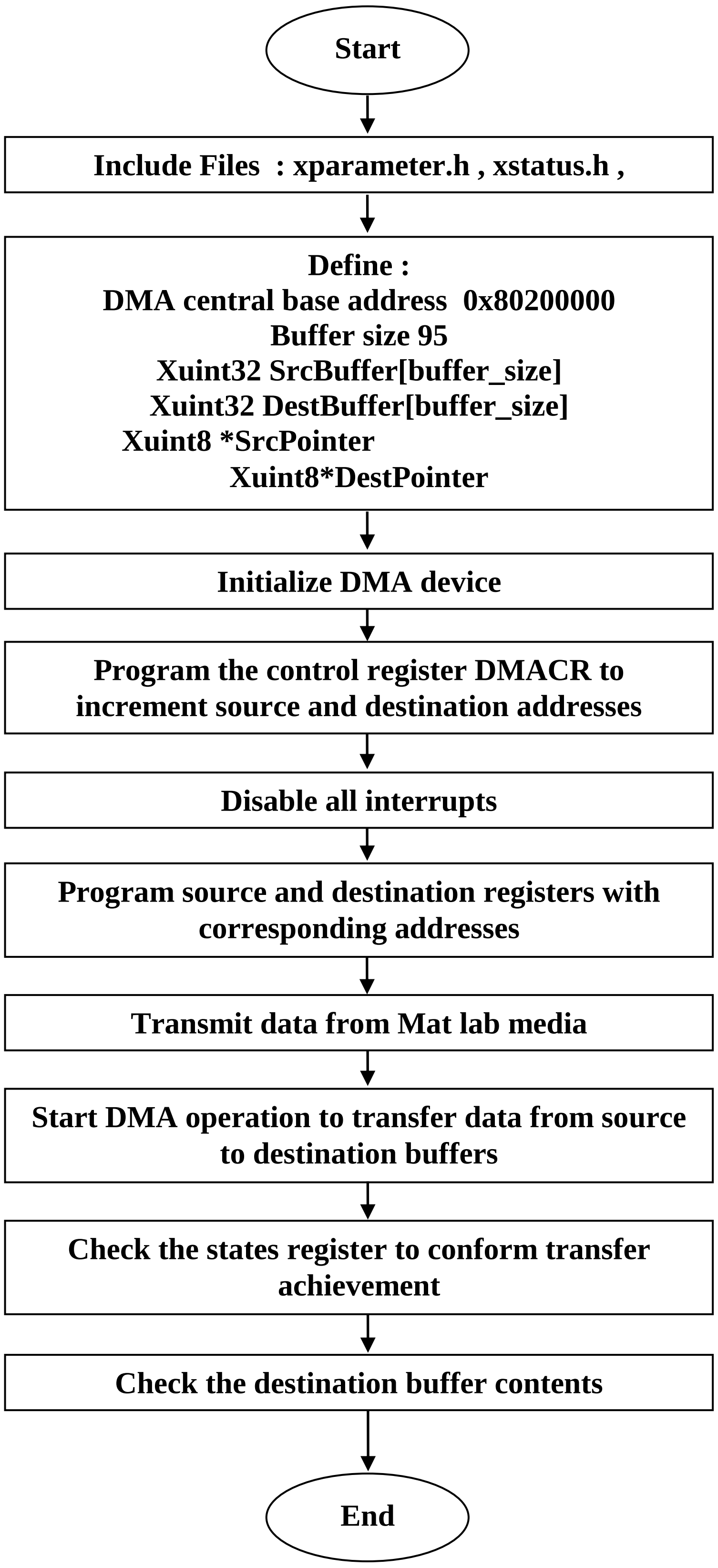
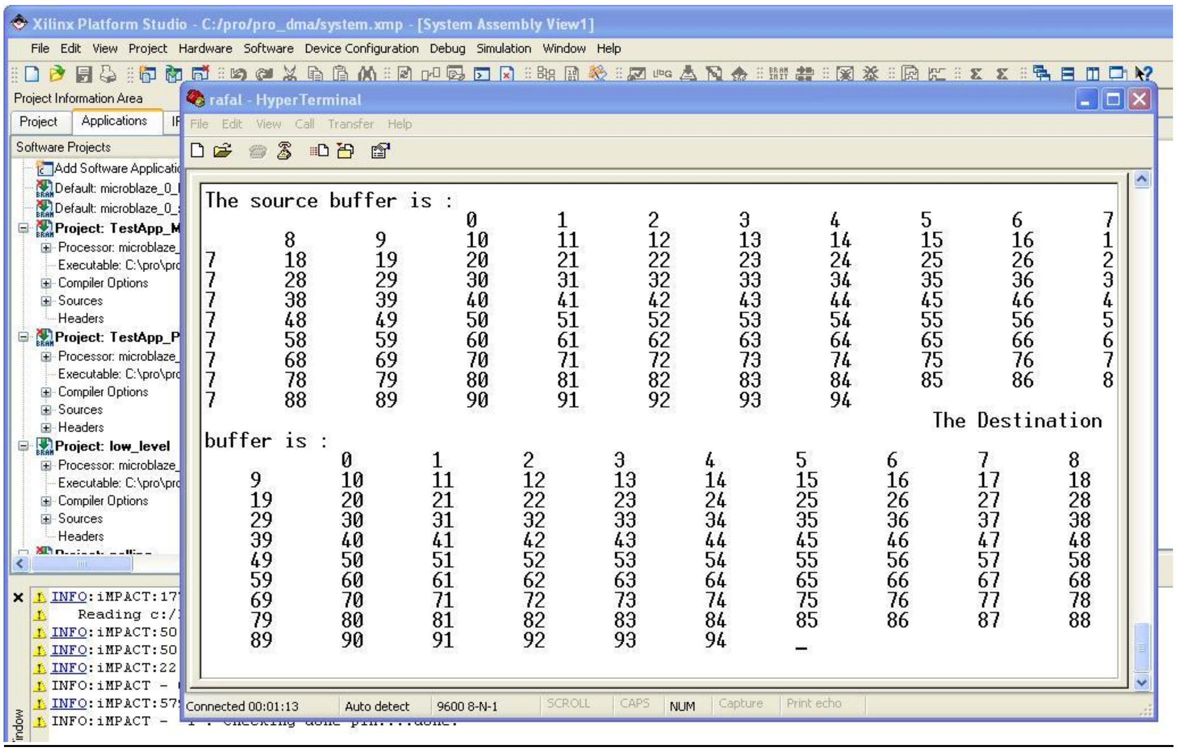

Start
Include Files : xparameter.h , xstatus.h ,
Define :
DMA central base address 0x80200000
Buffer size 95
Xuint32 SrcBuffer[buffer_size]
Xuint32 DestBuffer[buffer_size]
Xuint8 *SrcPointer
Xuint8*DestPointer
Initialize DMA device
Program the control register DMACR to increment source and destination addresses
Disable all interrupts
Program source and destination registers with corresponding addresses
Transmit data from Mat lab media
Start DMA operation to transfer data from source to destination buffers
Check the states register to conform transfer achievement
Check the destination buffer contents
End

Figure (5): The flow chart of designed system programing

6. Results

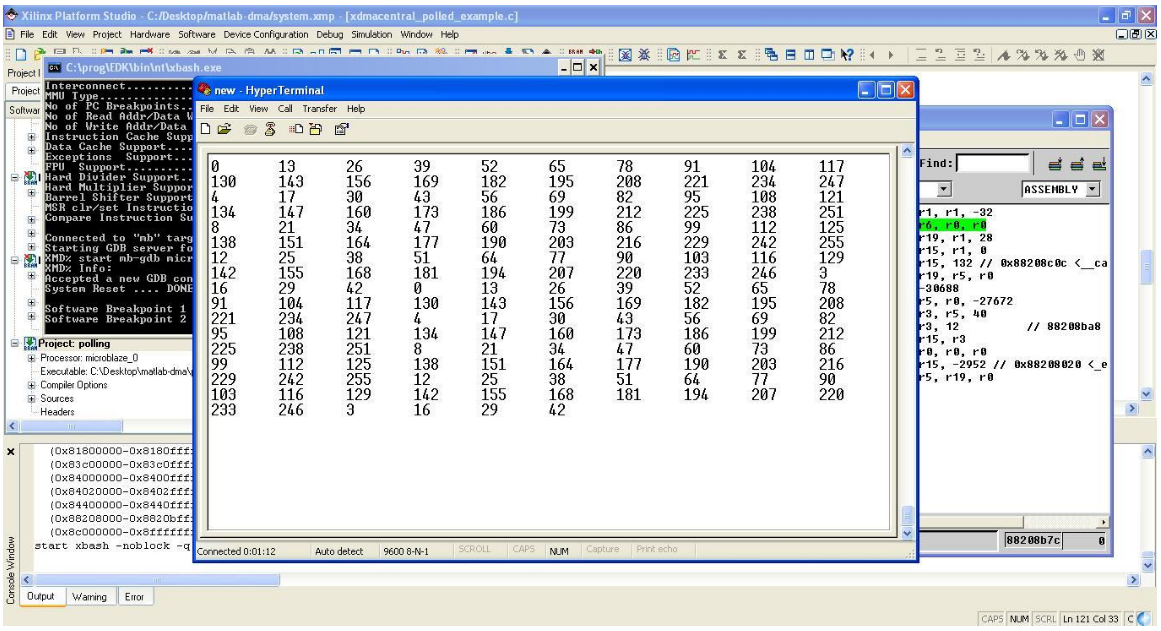
Figure (6) shows the data read from destination buffer and the data transferred from the source buffer displayed in hyper terminal window.



```
The source buffer is :
                         0      1      2      3      4      5      6      7
       8      9         10     11     12     13     14     15     16     1
7     18     19         20     21     22     23     24     25     26     2
7     28     29         30     31     32     33     34     35     36     3
7     38     39         40     41     42     43     44     45     46     4
7     48     49         50     51     52     53     54     55     56     5
7     58     59         60     61     62     63     64     65     66     6
7     68     69         70     71     72     73     74     75     76     7
7     78     79         80     81     82     83     84     85     86     8
7     88     89         90     91     92     93     94
                                                       The Destination
buffer is :
            0      1      2      3      4      5      6      7      8
     9     10     11     12     13     14     15     16     17     18
    19     20     21     22     23     24     25     26     27     28
    29     30     31     32     33     34     35     36     37     38
    39     40     41     42     43     44     45     46     47     48
    49     50     51     52     53     54     55     56     57     58
    59     60     61     62     63     64     65     66     67     68
    69     70     71     72     73     74     75     76     77     78
    79     80     81     82     83     84     85     86     87     88
    89     90     91     92     93     94     _
```

```
Connected 00:01:13    Auto detect    9600 8-N-1
```

```
INFO:iMPACT:17
   Reading c:/
INFO:iMPACT:50
INFO:iMPACT:50
INFO:iMPACT:22
INFO:iMPACT -
INFO:iMPACT:57
INFO:iMPACT -
```



```
Interconnect..........
MMU Type..............
No of PC Breakpoints..
No of Read Addr/Data W
No of Write Addr/Data
Instruction Cache Supp
Data Cache Support....
Exceptions  Support...
FPU  Support..........
Hard Divider Support..
Hard Multiplier Suppor
Barrel Shifter Support
MSR clr/set Instructio
Compare Instruction Su

Connected to "mb" targ
Starting GDB server fo
XMD% start mb-gdb micr
XMD% Info:
Accepted a new GDB con
System Reset .... DONE

Software Breakpoint 1
Software Breakpoint 2
```

```
0       13      26      39      52      65      78      91      104     117
130     143     156     169     182     195     208     221     234     247
4       17      30      43      56      69      82      95      108     121
134     147     160     173     186     199     212     225     238     251
8       21      34      47      60      73      86      99      112     125
138     151     164     177     190     203     216     229     242     255
12      25      38      51      64      77      90      103     116     129
142     155     168     181     194     207     220     233     246     3
16      29      42      0       13      26      39      52      65      78
91      104     117     130     143     156     169     182     195     208
221     234     247     4       17      30      43      56      69      82
95      108     121     134     147     160     173     186     199     212
225     238     251     8       21      34      47      60      73      86
99      112     125     138     151     164     177     190     203     216
229     242     255     12      25      38      51      64      77      90
103     116     129     142     155     168     181     194     207     220
233     246     3       16      29      42
```

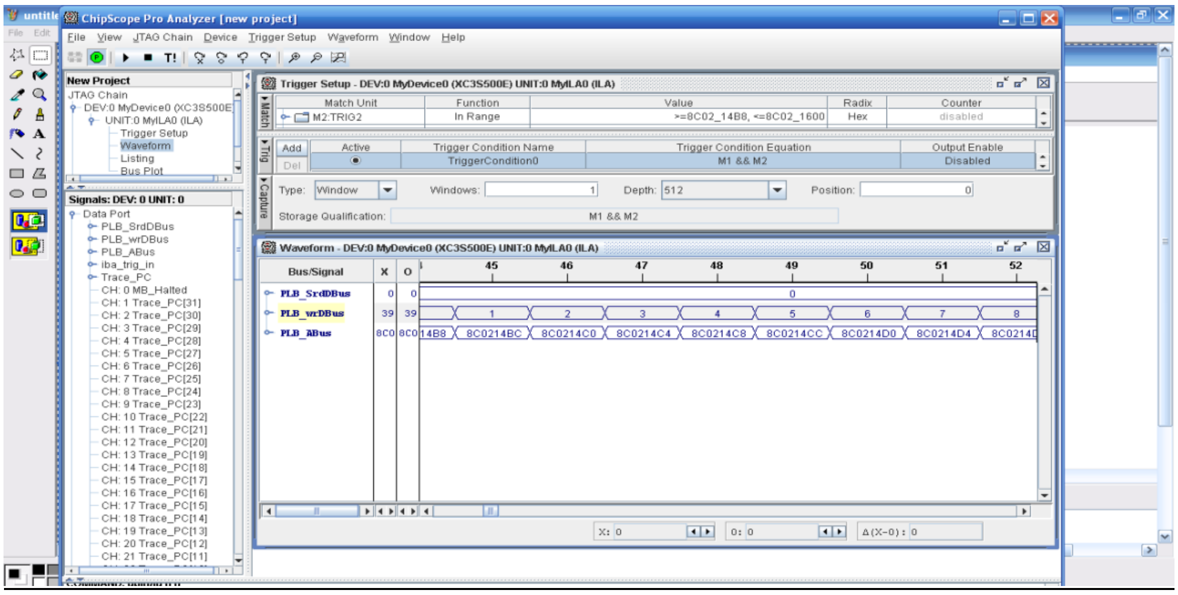
```
Connected 0:01:12    Auto detect    9600 8-N-1
```

```
r1, r1, -32
r6, r0, r0
r19, r1, 28
r15, r1, 0
r15, 132 // 0x88208c0c <__ca
r19, r5, r0
-30688
r5, r0, -27672
r3, r5, 40
r3, 12            // 88208ba8
r15, r3
r0, r0, r0
r15, -2952 // 0x88208020 <_e
r5, r19, r0
```

```
(0x81800000-0x8180fff
(0x83c00000-0x83c0fff
(0x84000000-0x8400fff
(0x84020000-0x8402fff
(0x84400000-0x8440fff
(0x88208000-0x8820bff
(0x8c000000-0x8ffffff
start xbash -noblock -q
```

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-a) presents the data flow with address during write bus cycle, Figure (7-b) presents the data flow with address during write bus cycle zoomed out.


Trigger Setup - DEV:0 MyDevice0 (XC3S500E) UNIT:0 MyILA0 (ILA)

Match Unit	Function	Value	Radix	Counter
M2:TRIG2	In Range	>=8C02_14B8, <=8C02_1600	Hex	disabled

Active	Trigger Condition Name	Trigger Condition Equation	Output Enable
●	TriggerCondition0	M1 && M2	Disabled

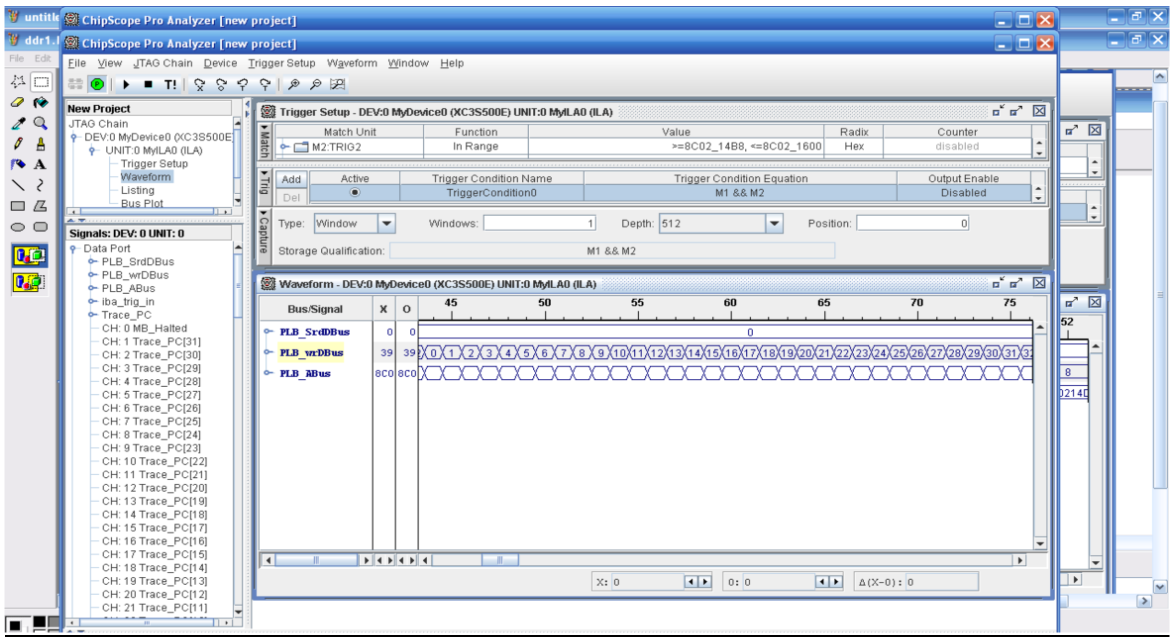
Type: Window Windows: 1 Depth: 512 Position: 0

Storage Qualification: M1 && M2

Waveform - DEV:0 MyDevice0 (XC3S500E) UNIT:0 MyILA0 (ILA)

```
Bus/Signal   X    O    45        46        47        48        49        50        51        52
PLB_SrdDBus  0    0    0
PLB_wrDBus   39   39   1         2         3         4         5         6         7         8
PLB_ABus     8C0  8C0  8C0214BC  8C0214C0  8C0214C4  8C0214C8  8C0214CC  8C0214D0  8C0214D4  8C0214D
```

a


Trigger Setup - DEV:0 MyDevice0 (XC3S500E) UNIT:0 MyILA0 (ILA)

Match Unit	Function	Value	Radix	Counter
M2:TRIG2	In Range	>=8C02_14B8, <=8C02_1600	Hex	disabled

Active	Trigger Condition Name	Trigger Condition Equation	Output Enable
●	TriggerCondition0	M1 && M2	Disabled

Type: Window Windows: 1 Depth: 512 Position: 0

Storage Qualification: M1 && M2

Waveform - DEV:0 MyDevice0 (XC3S500E) UNIT:0 MyILA0 (ILA)

```
Bus/Signal   X    O    45   50   55   60   65   70   75
PLB_SrdDBus  0    0    0
PLB_wrDBus   39   39   0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
PLB_ABus     8C0  8C0
```

b

Figure (7): Data transfer based on DMA operation displayed on chip scope window

a- **data flow with address during write bus cycle**

b- **data flow during write bus cycle zoomed out**

7. Conclusions

A soft core processor system is designed using embedded design techniques and configured on FPGA slice. The system is accommodated to act in DMA mode to transfer data from a peripheral to external DDR-SDRAM memory by adding a DMA IP core to the system and programming the resultant hardware using C-language with suitable API. The transferred data width is 32-bit which is adaptable the PLB data width. The system can operate with (40K) internal Block RAM and external (64M byte) DDR-SDRAM. The designed system can be used efficiently with video graphic arrays (VGA) to display graphics on a screen since the speed of data transfer between memory and the VGA controller is sufficient to capture all the pixels of the image frame(640columns x 480 rows) in 60 screen/second display mode, this rate of data flow could not be attained with processor systems without DMA operation mode.

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", 1-4244-1998 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- http://www.talktoanit.com/A+/aplus-website/lessons-io-principles.html

5- XILINX, 2010, "LogiCORE IP XPS Central DMA Controller (v2.03a)" , Web Site: www.xilinx.com, DS579.

6- Schmidt A. G., 2010, "Embedded Systems Design with Platform FPGAs", TK7895.E42S27 2010, diacriTech, India, Web Site: www.elsevter.com/permissions.