Tikrit Journal of Engineering Sciences/Vol.20/No.2/March 2013, (29-36)

Parallel Bidirectional Port for a Soft-core Microprocessor Designing a System

Mazin Rejab Khalil, Asst.Prof **Aseel Thamer Ibrahem**

Department of Computer Technical Engineering, Technical College, Mosul

Abstract

This paper introduces a technique to design a n bits bidirectional port to be connected with soft core microblaze processor system that is configurable on Spartan 3E slice. A soft core processor system with peripheral is designed using embedded design technique with integrated software environment (ISE tool) supplied by Xilinx. An interrupt system controller is added to the designed system to control the incoming interrupt signal to promote the interrupt service routine to deal with incoming data. The system performance is tested by sending and transmitting packets of data from and to a MatLab program using the P.C parallel port.

Keywords: Spartan 3E slice, Soft core processor, Embedded design, MatLab program.

تصميم منفذ متوازي ثنائي الاتجاه باستخدام منظومة معالج ذو نواة قابلة للبرمجة

الخلاصة

هذا البحث يقدم تقنية تصميم منفذ 8 بت ثنائي الاتجاه ثم ربطه مع منظومة معالج ذو نواة قابلة للبرمجة يتم تطبيقه على شكل شرائح نوع Spartan-3E. منظومة المعالج ذو النواة القابلة للبرمجة مع ملحقاته تم تصميمه باستخدام تقنية الانظمة المطمورة المتكاملة مع بيئة ISE tool المجهز من قبل شركة Xilinx. نظام المقاطعة اضيفة الى النظام المصمم للسيطرة على اشارة المقاطعة وتمكين البرنامج الرئيسي من الذهاب الى البرنامج الفرعي الذي من خلاله يتم استلام البيانات القادمة من المصدر الخارجي . النظام المصمم انجز وفحص من خلال ارسال واستلام بيانات من و الى ال MATLAB باستخدام منفذ التوازي parallel port الموجود في الحاسبة PC.

الكلمات الدالة: معالج ذو نظام قابل للبرمجة ، الانظمة المطمورة ، شرائح Spartan -3E.

Introduction

In the computer world, a *port* is a set of signal lines that the microprocessor, or CPU, uses to exchange data with other components. Typical uses for ports are communicating with printers, modems, keyboards, and displays, or just about any component or device except system memory. Most computer ports are digital, where each signal, or bit, is 0 or 1. A parallel port transfers multiple bits at once, while a serial port transfers a bit at a time (though it may transfer in both directions at once).

FPGAs are truly revolutionary devices, which blend the benefits of both hardware and software. They implement circuits just like hardware, providing huge power, area, and performance benefits over software, yet can be reprogrammed cheaply and easily to implement a wide range of tasks.[1]

Embedded design techniques are used to construct a processor system with peripherals to be implemented and configured on FPGAs.

The target of the research is to design a n bits parallel birectional port with interrupt to be connected with soft core microblaze proccesser system that is configurable on Spartan 3E slice. The main goal is to creat a system that utilizes the advantage of FPGA and soft-core processors in embedded system, and with aproper parallelism of data, to increase the runtime performance compared to the software runs on a comm. P.C with serial port. The design procedure starts by explaining the hardware part and software part of the designed system. The procedure describes how to introduce the software drivers that enables the C language program to deal with the hardware peripherals. The results are recorded and discussed.

Hardware part

Figure 1, shows the components of the embedded processor system, it is composed of the following parts:

- MicroBlaze embedded soft core processor which is a reduced instruction set computer (RISC) optimized for implementation in Xilinx Field Programmable Gate Arrays (FPGAs) [2].
- Processor Local Bus (PLB) v4.6 which provides bus infrastructure for connecting an optional number of PLB masters and slaves into an overall PLB system[3].
- Multi-Port Memory Controller MPMC which is a full parameterizable memory controller that supports SDRAM/DDR/DDR2 memory [4].
- The BRAM Block which is a configurable memory module that attaches to a variety of BRAM Interface Controllers[5].
- The LMB BRAM Interface Controller which is the interface between the LMB and the bram_block peripheral. A BRAM memory subsystem consist of the controller along with the bram_block peripheral[6].
- The LMB which is a fast, local bus for connecting MicroBlaze instruction and data ports to high-speed peripherals, primarily on-chip block RAM (BRAM) [7] .
- The ChipScope™ Pro Integrated CONtroller core (ICON) provides an interface between the JTAG Boundary Scan (BSCAN) interface of the FPGA device [8] .
- The XPS Universal Asynchronous Receiver Transmitter (UART) Lite Interface connects to the PLB (Processor Local Bus) and provides the controller interface for asynchronous serial data transfer. This soft IP core is designed to interface with the PLBV46 [9] .
- A general purpose input/output interface the XPS GPIO design provides to a Processor Local Bus (PLB). The XPS GPIO that can be configured as either a single or a dual channel device. The channel width is configurable and when both channels are enabled, the channel width remains the same for both. Figure 2 exhibits the block diagram of the GPIO core which is composed of the following parts:

❖ PLB Interface module which provides an interface between the GPIO core and the PLBV4.6 bus standard. The PLB Interface module implements the basic functionality of PLB slave operation and does the necessary protocol and timing translation between the PLB and the IPIC interfaces. The PLB Interface module allows only single beat transactions.

❖ Interrupt Controller which provides interrupt capture support for the GPIO core. The Interrupt Controller is used to collect interrupts from the GPIO core, by which the GPIO core requests the attention of the microprocessor through the assertion of interrupt signals.

❖ GPIO core which provides an interface between the IPIC interface and the XPS GPIO channels. The GPIO core consists of registers and multiplexers for reading and writing the XPS GPIO channel registers. It also includes the necessary logic to identify an interrupt event when the channel input changes. [10] .

Figure .3 show the block diagram of the implemented system, table .1 display the address map of implemeanted system.

Software part

The software part includes generating libraries and introducing a C application program to control the data flow via the general purpose input/output port (GPIO). Figure 5 shows the procedure of data flow between the parallel port of P.C and the designed parallel port included in the embedded processor system.

Figure.6 represents the flow chart of the application C program that is used to program the embedded processor system. Figure.7 represents the flow chart of application MATLAB program that is used transmited or received data.

Aspecial software drivers in form of functions were used in order to let the designed GPIO core be seen by the C language and they are:

- XGpio_Initialize(XGpio *InstancePtr,u16 DeviceId) which Initialize the XGpio instance provided by the caller based on the given DeviceID.
- XGpio_SetDataDirection(XGpio*InstancePtr, unsigned channel, u32 Direction Mask) which set the input/output direction of all discrete signals for the specified GPIO channel.
- XGpio_mGetDataReg(BaseAddress,Channel) which Get the data register of the specified GPIO channel.
- XGpio_mSetDataReg(BaseAddress,Channel,Data) which Set the data register of the specified GPIO channel.

Where instance Ptr is a pointer to an XGpio instance to be worked on, channel contains the channel of the GPIO(1 or 2) to operate on, direction mask is a bit mask specifying which discretes are input and which are output(bits set to 0 are output and bits set to 1 are input), DeviceId is the unique id of the device controlled by this component. Passing in a device id associates the generic XGpio instance to a specific device, as chosen by the caller or application developer. As well as a special MATLAB functions are also used to transmit and receive data.

Practical Results

For the purpose of debugging and results exhibition 40 bytes were sent from matlab to embedded system, and introduced to the system as input samples. Figure.7 shows a sample of the data received from matlab. Matlab can send any data (floating point, signed and unsigned integers) in the form of bytes, accordingly, the embedded processor system receive these bytes with the application program interface(API) which are responsible for making the parallel port interface be seen by the C language program. The program interprets the type of the sent data according to its location in the memory.

Figure.8 shows the control signal, displayed on chipscope.The control signal are used to make synchronization between the matlab system and the designed embedded system to ensure correct transmitting and receiving operations. Figure (9) shows the data received by the embedded system which is retransmitted to matlab as shown in figure.10. .

Conclusions

An embedded microblaze processor system has been configured on spartan3E FPGA device and implemented to accommodate several applications, the default data transfer media is via UART. Which is comparatively slow, therefore a parallel port is built to make data transfer faster, as it deals with transferring byte by byte in steal of transferring bits serially.

The most important point that must be noticed is the ability of the system to deal with defferent types of data.

The designed system is suitable to acquire data, process it according to the target and then output results to peripheral media. As printer or screen or any other device.

References

1. Scott Hauck and Andre DeHon, "Reconfigurable Computing The theory and practice Of FPGA-Based Computation", Elsevier Inc.., 2008.
2. Xilinx Company, "Spartan-3E FPGA Starter Kit Board User Guide", User Guide UG230 (v1.1), January-20-2011. http://www.xilinx.com/support/documentation/boards_and_kits/ug230.pdf.
3. Xilinx Company , " Processor Local Bus (PLB) v4.6 (v1.02a)" data sheet DS531 December 20, 2007
4. Xilinx Company, "Multi-Port Memory Controller (MPMC) (v6.00.a) ", data sheet DS643, April 19, 2010.
5. Xilinx Company, "Block RAM (BRAM) Block (v1.00a) ", data sheet DS444, March 12, 2007. http://www.xilinx.com/support/documentation/ip_documentation/bram_block.pdf.
6. Xilinx Company, " LMB BRAM Interface Controller (v2.10b)", data sheet DS452, March 2, 2010.http://www.xilinx.com/support/documentation/ip_documentation/lmb_bram_if_cntlr.pdf.
7. Xilinx Company, " Local Memory Bus (LMB) (v1.00a) ", data sheet DS445 , April 24. 2009.http://www.xilinx.com/support/documentation/ip_documentation/lmb_v10.pdf.
8. Xilinx Company," ChipScope Pro ICON (v. 1.00a, 1.01a, 1.02a)", data sheet DS646 March 24, 2008.
9. Xilinx Company, "XPS UART Lite (v1.00a)", data sheet DS571 January 14, 2008.
10. Xilinx Company, "XPS General Purpose Input/Output (GPIO) (v1.00a)", data sheet DS569 January 14, 2008. http://www.xilinx.com/support/documentation/ip_documentation/GPIO_v10.pdf.

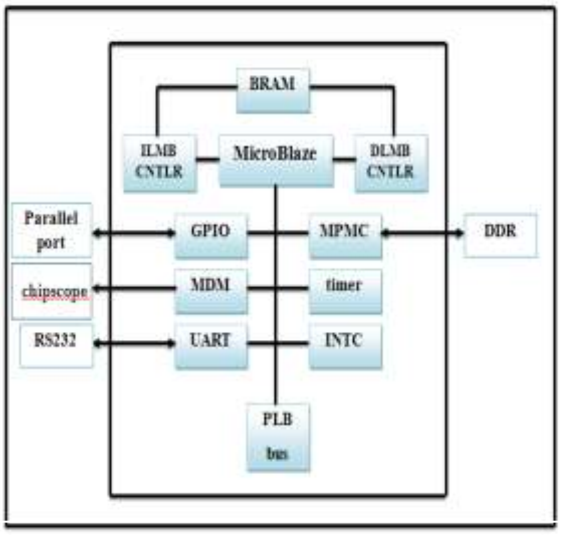


Figure.(1) the component of the imbedded processor system

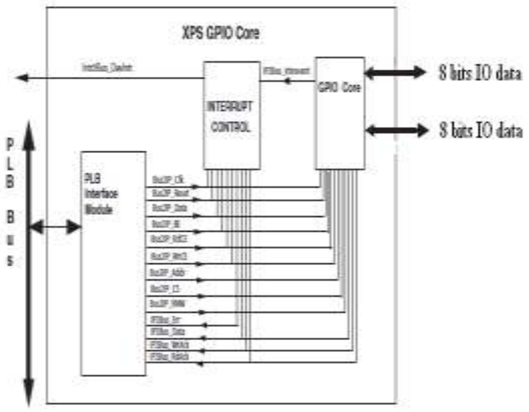


Figure.(2) XPS GPIO Block Diagram

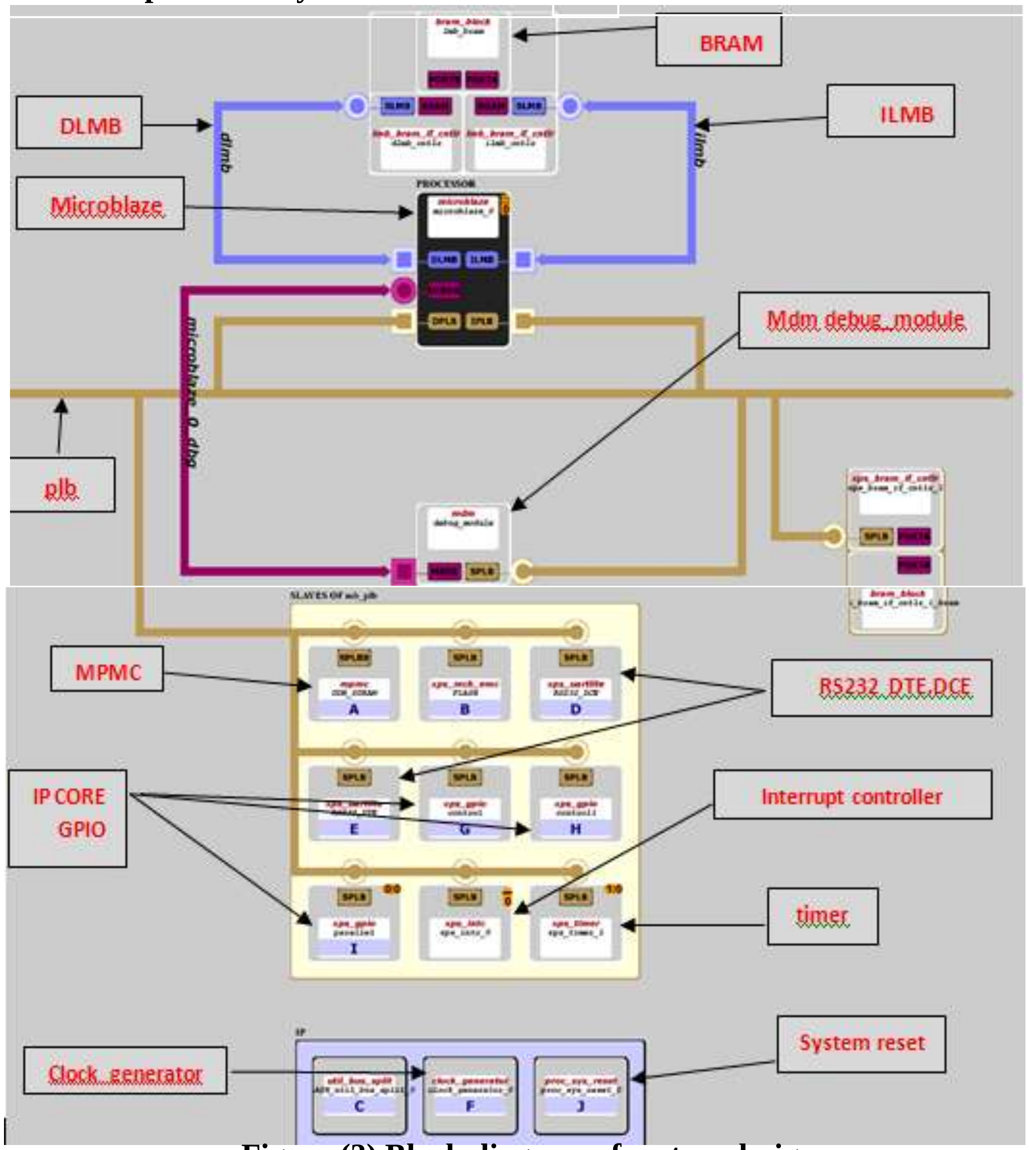


Figure.(3) Block diagram of system design

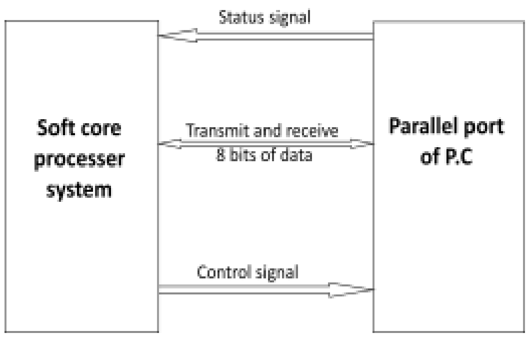

Status signal
Soft core processer system
Transmit and receive 8 bits of data
Parallel port of P.C
Control signal

Figure.(4) Transmitting and receiving data between the embedded processer system parallel port and the parallel port of the P.C

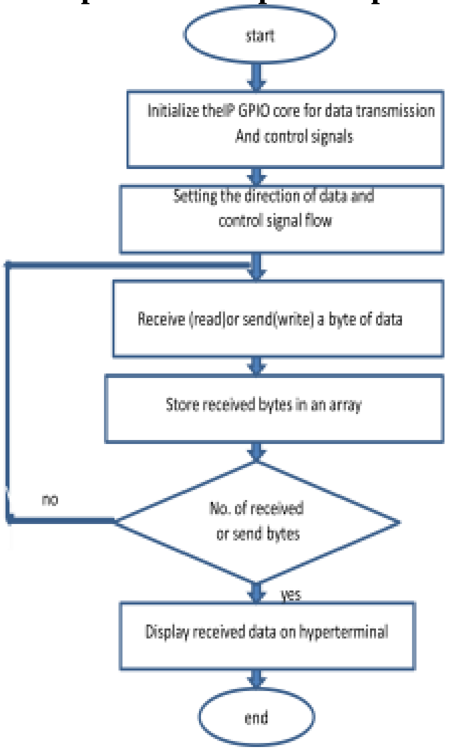

start
Initialize theIP GPIO core for data transmission And control signals
Setting the direction of data and control signal flow
Receive (read)or send(write) a byte of data
Store received bytes in an array
No. of received or send bytes
no
yes
Display received data on hyperterminal
end

Figure.(5) flow chart of the application C program

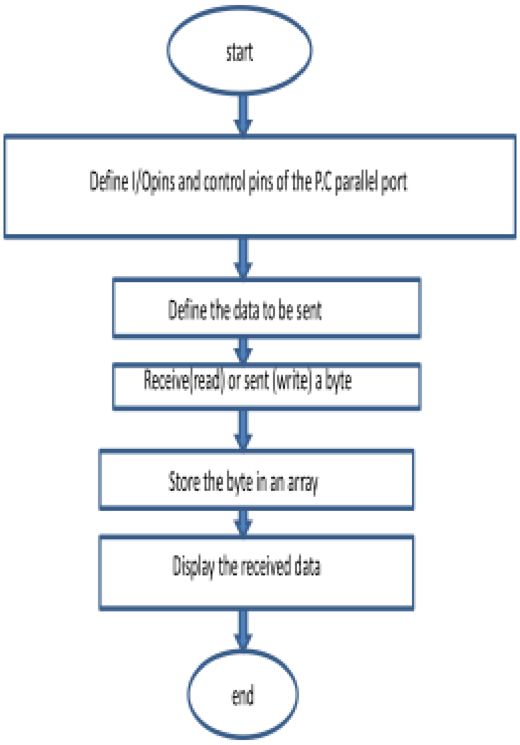

start
Define I/Opins and control pins of the P.C parallel port
Define the data to be sent
Receive(read) or sent (write) a byte
Store the byte in an array
Display the received data
end

Figure.(6) Flow chart of the application MATLAB program

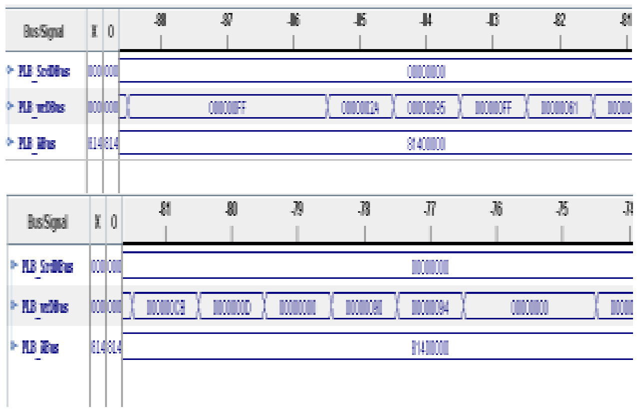


Figure.(7) Data receive from Matlab displayed on chipscope

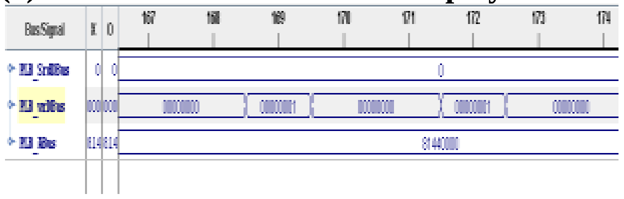
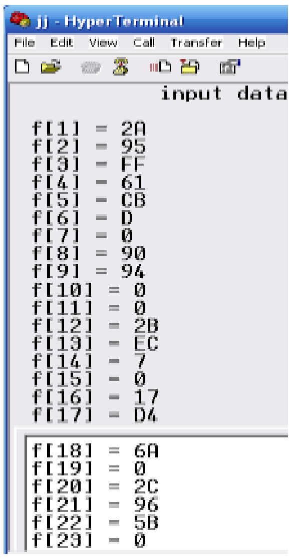


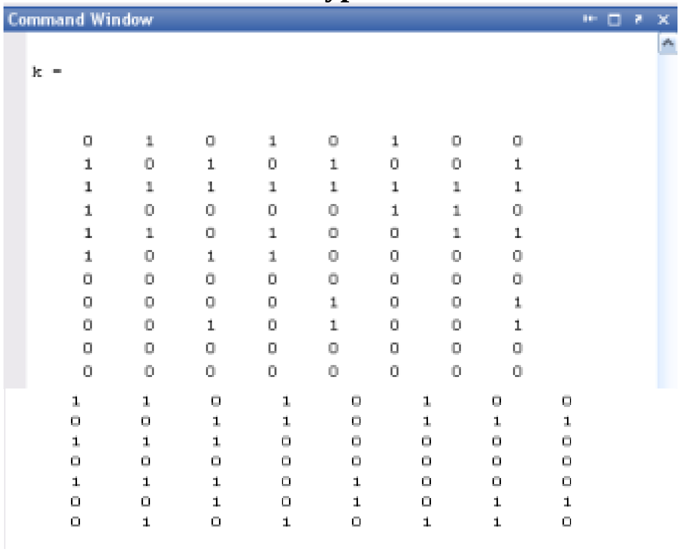
Figure.(8) control signal displayed on chipscope



```
          input data

 f[1] = 2A
 f[2] = 95
 f[3] = FF
 f[4] = 61
 f[5] = CB
 f[6] = D
 f[7] = 0
 f[8] = 90
 f[9] = 94
 f[10] = 0
 f[11] = 0
 f[12] = 2B
 f[13] = EC
 f[14] = 7
 f[15] = 0
 f[16] = 17
 f[17] = D4
 f[18] = 6A
 f[19] = 0
 f[20] = 2C
 f[21] = 96
 f[22] = 5B
 f[23] = 0
```

Figure.(9) the received data shown on HyperTerminal

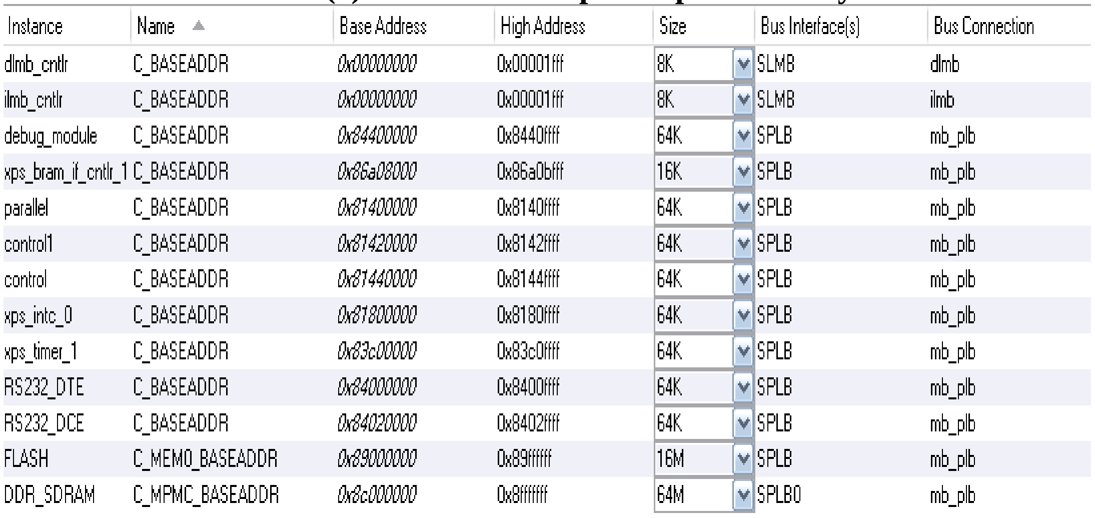


```
k =

     0     1     0     1     0     1     0     0
     1     0     1     0     1     0     0     1
     1     1     1     1     1     1     1     1
     1     0     0     0     0     1     1     0
     1     1     0     1     0     0     1     1
     1     0     1     1     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     1     0     0     1
     0     0     1     0     1     0     0     1
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
    1     1     0     1     0     1     0     0
    0     0     1     1     0     1     1     1
    1     1     1     0     0     0     0     0
    0     0     0     0     0     0     0     0
    1     1     1     0     1     0     0     0
    0     0     1     0     1     0     1     1
    0     1     0     1     0     1     1     0
```

Figure.(10) the data sent from embedded system to MATLAB and displayed on workspace.

Table.(1)The address map of implemented system



Instance	Name	Base Address	High Address	Size	Bus Interface(s)	Bus Connection
dlmb_cntlr	C_BASEADDR	0x00000000	0x00001fff	8K	SLMB	dlmb
ilmb_cntlr	C_BASEADDR	0x00000000	0x00001fff	8K	SLMB	ilmb
debug_module	C_BASEADDR	0x84400000	0x8440ffff	64K	SPLB	mb_plb
xps_bram_if_cntlr_1	C_BASEADDR	0x86a08000	0x86a0bfff	16K	SPLB	mb_plb
parallel	C_BASEADDR	0x81400000	0x8140ffff	64K	SPLB	mb_plb
control1	C_BASEADDR	0x81420000	0x8142ffff	64K	SPLB	mb_plb
control	C_BASEADDR	0x81440000	0x8144ffff	64K	SPLB	mb_plb
xps_intc_0	C_BASEADDR	0x81800000	0x8180ffff	64K	SPLB	mb_plb
xps_timer_1	C_BASEADDR	0x83c00000	0x83c0ffff	64K	SPLB	mb_plb
RS232_DTE	C_BASEADDR	0x84000000	0x8400ffff	64K	SPLB	mb_plb
RS232_DCE	C_BASEADDR	0x84020000	0x8402ffff	64K	SPLB	mb_plb
FLASH	C_MEM0_BASEADDR	0x89000000	0x89ffffff	16M	SPLB	mb_plb
DDR_SDRAM	C_MPMC_BASEADDR	0x8c000000	0x8fffffff	64M	SPLB0	mb_plb