International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:5, No:11, 2011

A Novel Implementation of Application Specific Instruction-set Processor (ASIP) using Verilog

Kamaraju.M[1], Lal Kishore.K[2], Tilak.A.V.N[3]

***Abstract*—**The general purpose processors that are used in embedded systems must support constraints like execution time, power consumption, code size and so on. On the other hand an Application Specific Instruction-set Processor (ASIP) has advantages in terms of power consumption, performance and flexibility. In this paper, a 16-bit Application Specific Instruction-set processor for the sensor data transfer is proposed. The designed processor architecture consists of on-chip transmitter and receiver modules along with the processing and controlling units to enable the data transmission and reception on a single die. The data transfer is accomplished with less number of instructions as compared with the general purpose processor. The ASIP core operates at a maximum clock frequency of 1.132GHz with a delay of 0.883ns and consumes 569.63mW power at an operating voltage of 1.2V. The ASIP is implemented in Verilog HDL using the Xilinx platform on Virtex4.

***Keywords*—**ASIP, Data transfer, Instruction set, Processor

I. Introduction

THE ASIPs have found extensive use in many real-world applications because of their efficient instruction set and hardware. Particularly in the embedded systems like motion or vibration detectors, temperature and humidity indicators need to be run most of the time on the batteries. So the power consumption of the system plays a critical role in the operation of the system. Instead of using a microcontroller or the microprocessor in such applications the ASIPs can perform better.To transfer data from a remote location, the normal embedded systems have to use microcontroller or microprocessor and some application specific hardware. Moreover sometimes the microcontrollers will have unnecessary hardware (regarding to the application) that are integrated onto them[10]. The paper describes the architecture of ASIP particularly designed for collection of data and transfer the packetized data to the terminal station.Transmitter and receiver [1],[2],[3] designs have been proposed for transfer of data. The technique mainly concentrates on establishing communication between two systems and procedures for the data transfer. Multiple transmitter and receiver communication was described in [4].

Prof.Kamaraju.M is with the Dept.of ECE, Gudlavalleru Engineering College, Gudlavalleru, India (phone:+91 9849157394) e-mail: madduraju@yahoo.com).

Prof.Dr..Lal Kishore.K, working as Director, R&D, JNTUH,Hyderabad, India. (e-mail: lalkishorek@yahoo.com).

Prof.Dr.Tilak.A.V.N working as Dean (A&A) and professor of ECE, Gudlavalleru Engineering College, Gudlavalleru, India (e-mail: avntilak@yahoo.com).

The earlier designs[8] consists of only transmission and reception circuits and there are no processing elements included in them. Mohan and Land[11] explained interfacing logic to connect a transmitter and receiver to the external processing units like microprocessors or microcontrollers. These processing units may be 8-bit or 16-bit[12]. But the interface logic is same for both types of processing units. Interfacing separate transmitter and receiver sections externally to the processor makes the system more complex and causes increase in power consumption. ASIP is one solution to reduce the hardware for the specific application. Several ASIP architectures for various applications have been proposed in [7].

The following sections will provide a brief overview of architecture of the ASIP and explanation about the implementation. Then a brief description about the CPU, transmitter and receiver sections of the ASIP are given. The first section describes about the overall system block diagram. Later the individual modules in the system are explained. Finally few notes on simulation results and the future scope of the processor are depicted.

II. Architecture Of Asip

Fig. 1 shows the block diagram of the ASIP being described. The system consists of a 16-bit processing unit and transmitter and receiver sections which are interfaced to the CPU to support the data transfer.

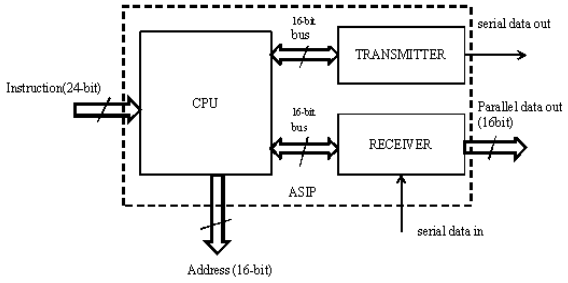

Fig. 1 Block diagram of ASIP

The central processing unit of the ASIP consists of 16-bit address and data buses[11]. The ALU (Arithmetic Logical Unit) is a fixed point integer type which performs 16-bit arithmetic and logical operations on the sensor data. The logical operations are used for testing of the sensor data and the arithmetic operations are used to manipulate the data. The CPU[8] also consists of 8 general purpose registers (R0-R7) each of 16 bits wide. There are no special purpose registers in the CPU like accumulator and there is no priority among them.

The instruction length of the ASIP is 24-bit wide. The CPU has two flag fields called zero and carry flags. These are affected only during the execution of arithmetic and logical instructions. Along with this a stack is provided to support jump and loop instructions. The stack has 4 memory locations each having the capability to store a 16-bit return address.

To ease the transfer of data to and from the processor a special function register (SFR) of 16-bit wide is provided in the architecture. The function of the special function register is it acts as an interface between the transmitter and receiver sections to the CPU. Some application specific instructions are also provided particularly for the data transfer. The SFR will also acts as a control word register for the transmitter and receiver sections.

In the 16-bit SFR the least significant two bits are used to select the FIFO (First In First Out) and status registers and the next two least significant bits are used to select the upper byte and lower byte of the 32-bit word. The fifth least significant bit is used to enable transmitter and receiver sections. If the bit is set then the transmitter will be enabled or if the bit is reset the receiver will be enabled.

III. Tansmitter And Receiver Section Of Asip

The ASIP has independent transmitter and receivers[2],[3]. The transmitter consists of different blocks, viz. decoder, buffer, FIFO, transmit controller and a parallel to serial converter. The block diagram of the transmitter is shown in figure 2. The main block in the transmitter is FIFO which store the data to be transmitted outside. In the transmitter the FIFO consists of 16 memory blocks and each block can store a 32-bit word. So the transmitter will send the data in the form of 32-bit packets.

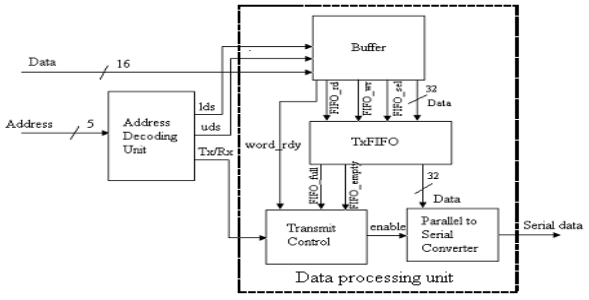

Fig. 2 Block diagram of the transmitter

The operation of the transmitter is as follows. Initially, if we want to send a 32-bit packet in serial form first we have to enable the transmitter. Since the CPU is 16-bit wide we have to send the necessary control words to the transmitter. This can be achieved by writing or loading the address into the SFR. After loading the address into the SFR the transmitter reads the address and decodes it. Decoding the address will generate the necessary control signals for the transmitter. The following procedure will explain about transmitting 32-bit word serially by combining two 16-bit words.

The 32-bit frame format for the transmitter and receiver is shown in the figure 3. The frame consists of 16-bit data (i.e lower 16 bits), source address(4 bits) and destination address(4 bits). The remaining 8 bits in the frame are kept idle(padded with zeros) or can be used as control bits like frame number. Since the source and destination addresses are of 4 bits wide we can connect a maximum of 16 different data nodes to the transmitter or the receiver.

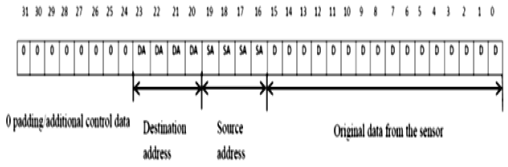

Fig. 3 Frame format for the transmitter and receiver

The receiver performs the reverse operation of the transmitter. Since the ASIP contains the independent transmitter and receiver, the receiver also consists of a FIFO similar to that of the transmitter. It can also store a maximum of 16 data packets each of having length 32 bits. The block diagram of the receiver is shown in figure 4.

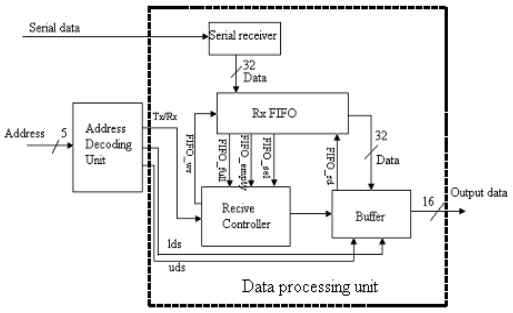

Fig. 4 Block diagram of the receiver

The receiver controller is the main block in the receiver. It generates the necessary control signals to all the blocks in the receiver. The controller sequences the flow of data through different blocks by generating necessary signals. The controller will take the *Tx/Rx word ready*, *fifo_empty* etc. signals from various blocks and generates different control signals. The operation of the receiver is opposite to that of the transmitter. In the receiver, the buffer takes the 32-bit packet from the FIFO and divides the packet into two 16-bit words. Since data bus width of the processor is only 16 bits wide it can only take 16-bit data. Again the upper and lower data select signals are used to select the 16-bit words.

To receive the serial data, first the receiver should be enabled. This can be done by writing the proper control word into the SFR. When the receiver is enabled, the serial data is received through the RXD pin. Then this serial data is converted into parallel data (32-bit) by using the serial to parallel converter. If serial to parallel conversion is completed for the received 32-bit data a *word_rdy* signal is generated. At this time the buffer will generate *FIFO write* signal. Then the

parallel data is temporarily stored in the FIFO. When *FIFO read* signal is asserted and based on the upper or lower select signals, either upper or lower 16-bit words are placed onto the data bus.

IV. Implementation

The ASIP was implemented using Xilinx platform on Virtex4 device in VerilogHDL. Application specific instructions were designed along with general purpose instruction set. These instructions will reduce the burden on the processor during the transmission and reception of data. Some of the application specific instructions can be seen in the example programs given below.

Upon execution of the code given in program1 the transmitter successfully transmitted the 32-bit word. The simulation results are given in the next section.

Program1:

```
LOAD R0,0005H  //Loading immediate data into the
                register R0//
OUT  R0,0000   //Sending the contents of the register
                R0 to the output port having address
                0000h//
SFRW 0010H     //Transmitter lower 16-bit word //
SFRW 0013H     //Transmitter upper 16-bit word //
```

By the execution of the following code the receiver successfully received the 32-bit word. The simulation results are given in section V.

Program 2:

```
SFRW 0000  //Receive lower 16-bit word //
SFRW 0003  //Receive upper 16-bit  word //
```

The flow chart for the transmitter is shown in figure 5. Initially when the reset signal is asserted then the default values are loaded into the registers and the transmitter and receiver are disabled. If the reset signal is not asserted then transmitter waits for the valid SFR address. If the valid address is available then the address will be decoded and the transmitter is enabled. If the write strobe signal is asserted and data is available on the data bus based on the *lds* and *uds* signals either upper or lower byte is loaded into the buffer. If read signal is asserted then the contents of status register are sent to the output port. When the buffer is filled *word ready* signal loads the 32-bit data into the FIFO. Then if *ptos_en* signal is asserted the 32-bit data in the FIFO is loaded into the parallel to serial converter and the data will be transmitted.

Figure 6 shows the flow chart for the receiver. The reset signal makes the processor to load default values into its registers. If the reset signal is low then receiver waits for the valid SFR address. If valid address is fed to the decoder the address will be decoded and generates necessary control signals. When the *serial to parallel enable* signal is asserted then the receiver starts receiving the serial data. Then this received serial data is loaded into the FIFO based on the FIFO condition. If the FIFO is full then the serial receiver waits till at least one position in the FIFO is vacant, if not, then the 32-bit parallel data is immediately loaded into the FIFO. When the *read strobe* signal is asserted this 32-bit data is loaded into the 32-bit buffer register. Then based on the decoded signals *lds* or *uds* the corresponding lower or upper 16-bit data is sent to the output port.

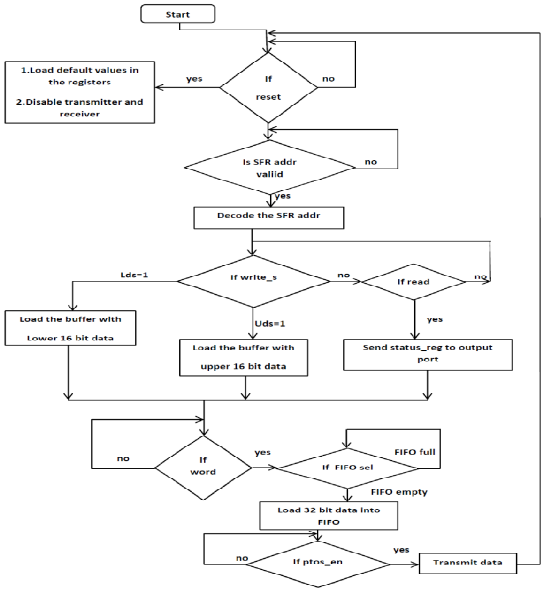

Fig. 5 Transmitter flowchart

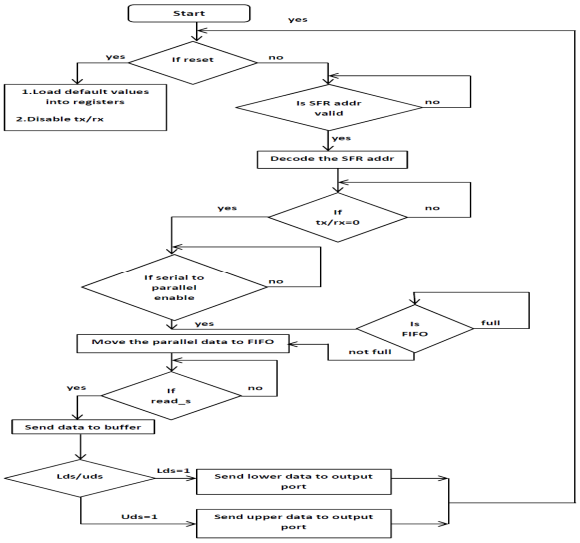

Fig. 6 Receiver flow chart

V. Results

All the modules of ASIP are simulated and verified using the Xilinx tools. The simulation results of top level transmitter section of ASIP are shown in figure 7. Initially the data(0005h) is loaded into the register r0. To verify that the data was properly loaded in the register the OUT instruction was written. The instruction sends the data to the output port(0000h indicates port address). In the simulation results

we can clearly observe the 0005h at the output port.Next, in order transmit the data it should be 32-bit format. So the processor has to send two 16-bit words to form a packet. The lower byte indicates the actual sensor data while upper byte indicates the payload attached to the data. In the above program both upper and lower bytes are 0005h. To send the upper byte and lower byte two instructions are written. The data transmission started when *ptos_en* signal is asserted. This signal asserts only when the 32-bit packet is loaded into the FIFO.

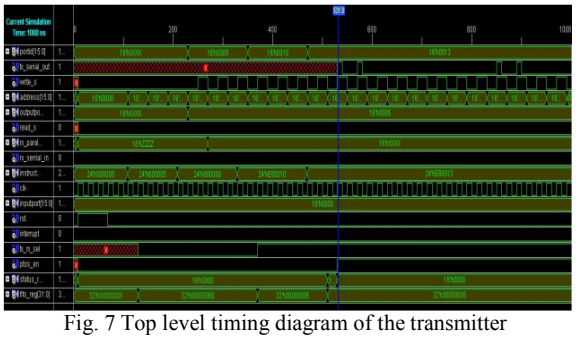

Fig. 7 Top level timing diagram of the transmitter

The simulation results of top level receiver are shown in figure 8. When the *tx_rx* signal is asserted low the receiver starts receiving the serial data. If the total 32 bits are received then the packet is moved into the FIFO. At this time if read strobe signal is asserted then the data is loaded onto the buffer. If lower or upper byte selection signals goes active high the corresponding data from the buffer is obtained at the output port. In the simulation results shown the upper data word is selected.

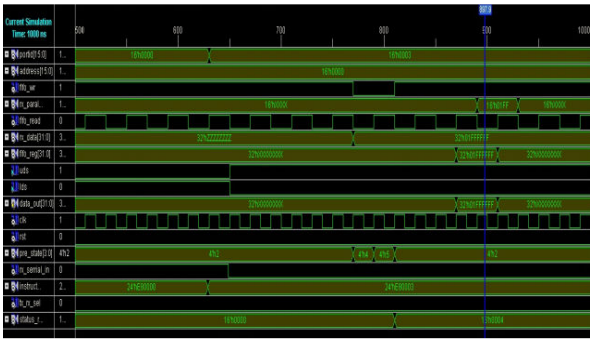

Fig. 8 Top level timing diagrams of the Receiver

The top level module with input and out signals of the ASIP is shown in Fig. 9 and the RTL schematics of the transmitter and receiver are shown in the figures 10 and 11 respectively.

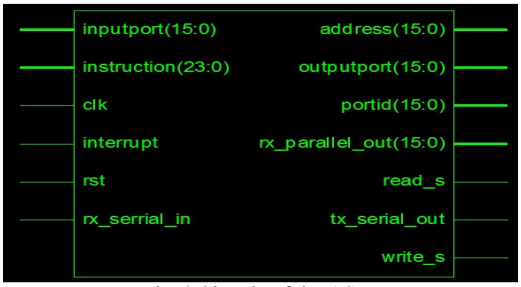

Fig. 9 Signals of the ASIP

The transmitter consists of address decoder, buffer, FIFO, transmit controller, and parallel to serial converter. Interconnection between the blocks and the input- output signals of the transmitter module are also shown in the figure. 10.

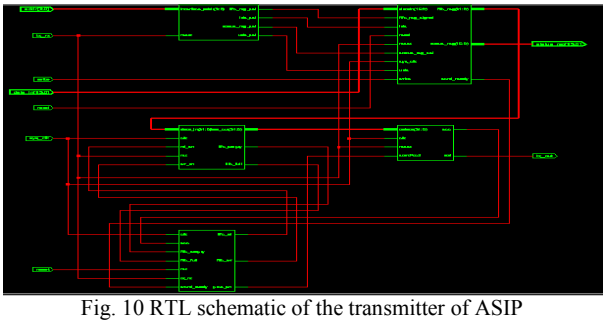

Fig. 10 RTL schematic of the transmitter of ASIP

The receiver is also similar to that of the transmitter except that the receiver will perform the reverse operation of the transmitter. The receiver consists of address decoder, buffer, FIFO, receive controller, and serial to parallel converter. Interconnection between the blocks and the input-output signals of the transmitter module are also shown in the figure 11.

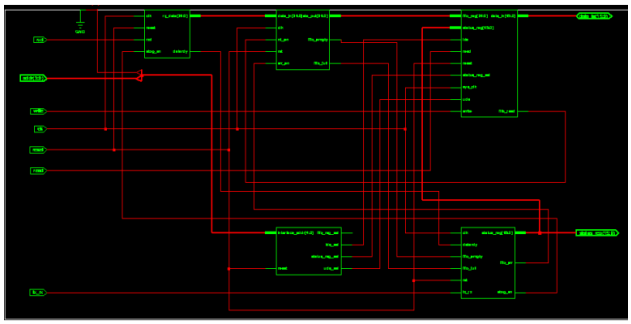

Fig. 11 RTL schematic of the receiver

Figure 12 represents the layout of the ASIP core when implemented onto the FPGA Virtex4. In figure 13, the colored area represents the components of the ASIP that are placed on the FPGA and the place and rout diagram of the ASIP core is shown in figure 13.

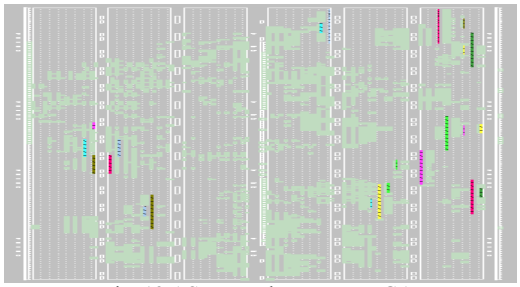

Fig. 12 ASIP core layout on FPGA

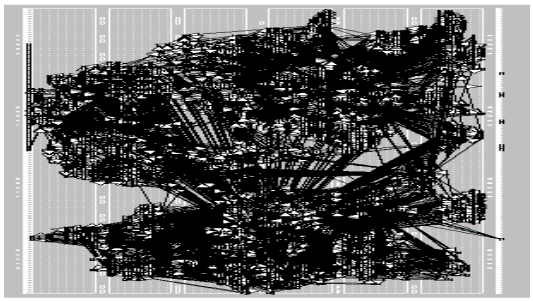

Fig. 13 Place and route diagram of ASIP core

In figure 14, the device utilization of ASIP core and CORE1[5] were specified. The graph clearly shows that the number of slices and LUTs utilized by the ASIP is less than that of the CORE1. The graph also indicates that the area occupied by ASIP on the FPGA is less.

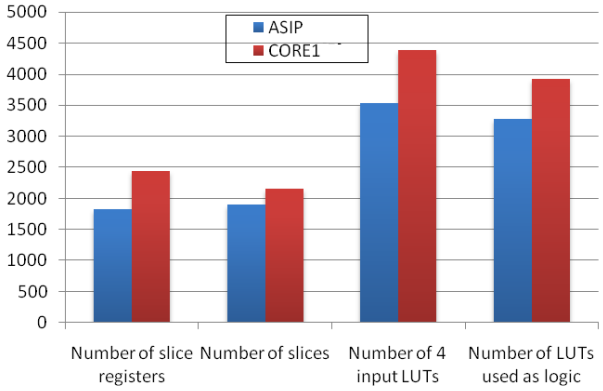

Fig. 14 Device utilization parameters of ASIP core and CORE1[5]

The comparison of different parameters related to the ASIP's signal delays are shown in the figure 15.

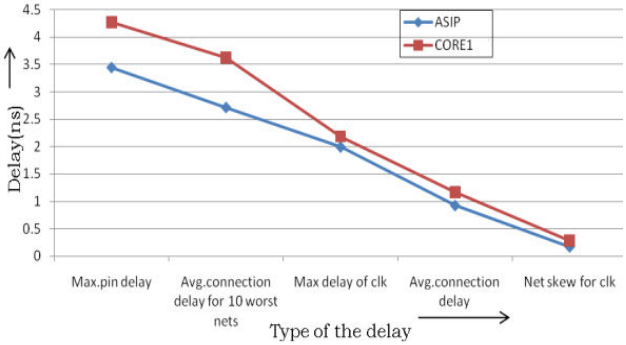

Fig. 15 Various delays of ASIP and CORE1

The figures 16 and 17 gives comparative view of the power consumption of the ASIP and CORE1. In the figure 16, the deviation of the curves indicates that as the operating voltage is increasing the power consumption of the CORE1 increases more rapidly when compared with ASIP power. The variation of power with respect to frequency for the ASIP and CORE1 is shown in figure 17.

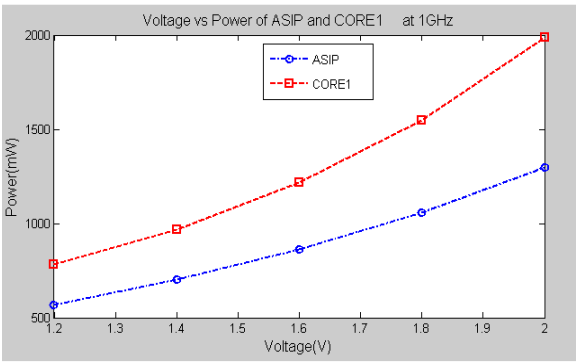

Fig. 16 Voltage vs power trend of ASIP and CORE1

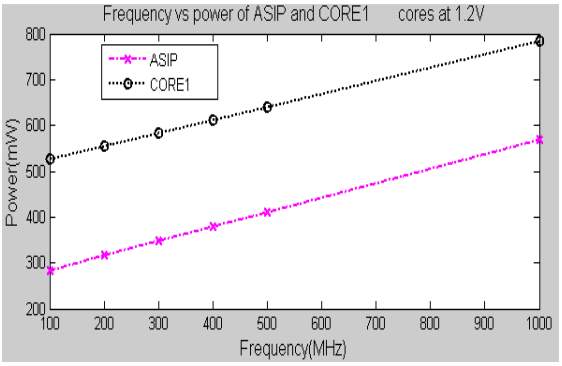

Fig. 17 Power consumption of ASIP and CORE1

A comparison of different parameters of ASIP and CORE1 **is** shown in TABLE I. The table also gives the specifications of the ASIP.

TABLE I
COMPARISON OF ASIP AND CORE1

Parameter	ASIP core	CORE1
Total number of gates	33.561k	44.871k
Maximum delay for clock	1.992ns	1.999 ns
Maximum frequency	1.132 GHz	1.083GHz
Power consumption(@1.2V,1GHz)	569.83mW	783.15mW
Peak memory usage for synthesis	301MB	308MB
Setup time	4.667ns	5.117ns
Hold time	0.441ns	0.455ns

VI. CONCLUSION

The ASIP designed for the data transfer integrates a processing unit and individual transmitter and receiver onto the single chip. The results shows that the ASIP design occupies less space and consumes less power. The advantage of the designed ASIP is that it can transfer serial data at high speeds which is the key requirement for the present communication technologies. The application specific instructions designed for transfer of data can transmit and receive data with less number of instructions thus reducing the

burden on the processor. Moreover an integrated CPU in the design performs the necessary processing for the transmission and reception of data. The scope for adding multiple transmitters and receivers is also provided in the ASIP design.

REFERENCES

[1] ARINC Digital Data System Compendium, ARINC Report 419-3,November 5, 1984.

[2] ARINC Specification 429P1-15, Sep. 1, 1995.

[3] ARINC Specification 429P2-15, March 6, 1996.

[4] "ARINC 629 P1-4 Multi-Transmitter Data Bus","Part1, Technical Description", December 1995.

[5] ARINC Airborne Computer Data Loader,ARINC Report 615-2, June 1, 1991.

[6] Baburao.K, Apparao.T, Prabu.A.V, RamBabu.E, "VHDL Implementation and Verification of ARINC-429 Core",International Journal of engineering, technology and science,vol . 2,2011,pp 174-178.

[7] Jin Soo Kim Sunwoo, Myung H. Sunwoo, "Three low power ASIP processor designs for communications, video, and audio applications", International Conference on Design & Technology of Integrated Systems in Nanoscale Era, 2007. DTIS , pp. 241-244.

[8] Kamaraju.M, Tilak.A.V.N, Lalkishore.K, Baburao.K, "VHDL implementation and verification of ARINC-429 core", CoRR International Journal, vol.1,2010,pp1-5.

[9] Kane.G., and Heinrich.J,MIPS RISC Architecture: reference for the R2000, R3000, R6000 and the new R4000 instruction set computer architecture. Prentice-Hall, EnglewoodCliffs, NJ, 1992.

[10] Krall.A. "An extended Prolog instruction set for RISC processors", in VLSI for Artificial Intelligence and Neural Networks (New York, NY, 1991), J. G. Delgado-Frias and W. R. Moore, Eds., Plenum Press, pp. 101–108.

[11] Mohan.R, Land.I, "Building Integrated ARINC 429 Interfaces using an FPGA". Actel Corporation, Mountain View, California,2005.

[12] Motorola Inc.,"M68000 8-/16-/32-Bit Microprocessors User's Manual,"[Onlinedocument],1993(Ninth Edition), available at http://www.freescale.com/files/32bit/doc/ref_manual/MC68000UM.pdf.

[13] Rose.J, GamaL.A.E, and A. Sangiovanni Vincentelli.A, "Architecture of Field-Programmable Gate Arrays," Proc. IEEE, vol. 81, no. 7, July 1993, pp.1013-102.

[14] Sato.J, Alomary,A. Y., Honma,Y., Nakata, T., Shiomi, A., Hikichi, N., and Imai, M. PEAS-I: "A hardware/software codesign system for ASIP development", IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science E77-A, 3 (March 1994),pp 483–491.

[15] Van prate.J, Goossens.G, Lanneer.D, and De man.H " Instruction set definition and instruction set selection for ASIPs", In Proc. of the 7th International Symposium on High-Level Synthesis 1994, pp. 11–16.

Kamaraju.M obtained his bachelor and master degree from Andhra university, Visakhapatnam, currently working as professor in the dept.of ECE, Gudlavalleru Engineering College,Gudlavalleru. He is a fellow of IETE and member of IEEE and VSI. His interests are Microprocessors , Microcontrollers, Embedded system Design, Low Power VLSI Design, He had an experience of 18 years in the fileld of teaching and 5 years of research experience in the field of VLSI design.

Dr. Lal Kishore.K obtained his Master's degree and Ph.D. from Indian Institute of science (IISC) Bangalore. He had published 114 research papers in international/Natioanl journals and presented papers in International/ National Conferences. He wrote books on Electronic Devices and circuits,Electronic circuit Analysis, Linear IC Applications, Electronic Measurements & Instrumentation and VLSI Design.

Dr. Tilak.A.V.N obtained his Master's degree from Indian Institute of Technology, Kanpur and Ph.D. from Indian Institute of Technology,Madras during 1984 and 1997 respectively. He is a Fellow of Institution of Electronics And Telecommunication Engineers(IETE), India and Life Member of ISTE.