ISSN 2278-3091

International Journal of Advanced Trends in Computer Science and Engineering, Vol. 3 , No.1, Pages : 95 – 100 (2014)
Special Issue of ICETETS 2014 - Held on 24-25 February, 2014 in Malla Reddy Institute of Engineering and Technology, Secunderabad– 14, AP, India

A Low Complexity Serial Communication Interface Design Based On FPGA

Ch.Ramakoti Reddy[1] , SK.Nagul meera[2], A.Naresh[3]
[1]Asst professor dept. of ECE, mriet, India, rkr293@gmail.com
[2]Asst professor dept. of ECE, mriet, India, nagulmeera987@gmail.com
[3]Asst professor dept. of ECE, mriet, India, nari.432@gmail.com

Abstract: A serial communication interface based on FPGA (Field Programmable Gate Array) has been designed in this paper, used for data communication with other equipment. the realization of this serial communication function under the condition of without any increasing in hardware resources. It accords with hardware equipment standardization principle. The design is adopted the Xilinx Company's Virtex-4 series FPGA chip, simulation results indicate that it satisfies protocol requirements.

Keywords-FPGA; Serial Communication; Parallel to Serial Conversion

1. INTRODUCTION

As the key processing equipment of comprehensive as processing system, mission management computer implements comprehensive control and management of the system, data processing, information processing, data decoding and so on, it needs to crosslink with many equipment's and the types of communication interface are various. Generally standard interface as RS232, RS422, RS485, ARINC429, simulation and on-off, can satisfy the requirements, but there are some equipment which has special requirements. In order to guarantee normal correspond, private communication interface design is needed module guarantees the stability and the reliability of the system in largely an reduces the design personnel repeat work and effectively improves the work efficiency. But the design requirements of mission management computer are diverse, how possible on the base of without any increasing in existing module kinds to satisfy increasingly rich design requirements is currently hardware design personnel need to consider problems. The FPGA (Field Programmable Gate Array) has the characteristics of the reconstruction, the rapidity, design flexibility and the high –density of logical resources [1]; we make full use of the programmable resources of FPGA on a great extent to module function expansion and to meet increasingly complex requirements [2] The implement mode of private Communication interface based on FPGA is presented in this paper; under the condition of without any increasing in original module hardware resources, it has realized module function expansion, shortened the development cycle, and satisfied the module standardization requirements

II. THE GENERAL STRUCTURE OF DESIGN

(A). Communication Protocol: The Mission management of computer sends control information and high-speed Information to aviation administer transponder by CLK, STB and DI signal, CLK denotes data sending clock, DI

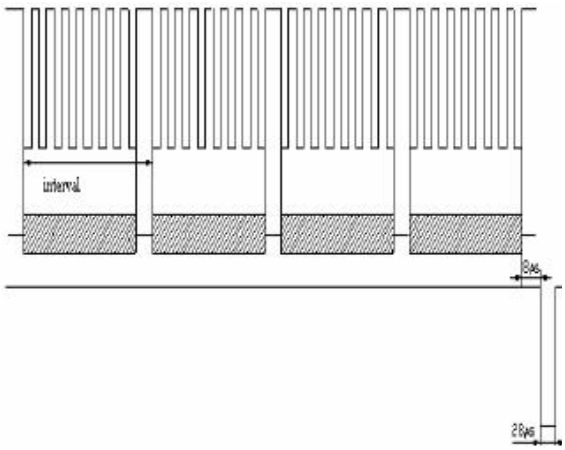

FIG1: Protocol Structure

Denotes data (including control and state data), STB denotes data sending finish symbol [4]. Figur1 and Figure 2 respectively shows the signal formats of CLK, STB and DI. Figure 2 shows the interval in figur1. T200he cycle of CLK is 13us, duty-cycle is3/10, the 32 bits of data is sent in bytes every time, it has32us interval between bytes, 4 bytes have been sent 8us later, STB is effective finished.

(B).General Frame:

This interface control logic is located on data processing module, FPGA communicates with PowerPC processor by internal bus [5]. Figure 3 shows the integral design diagram of this interface logic. FPGA adopts the Xilinx Company'sVirtex-4 series, the input signal includes 33.33MHz clock, data/address multiplexing bus and write enabling signals. The output signal is CLK, STB, DI which are the signal the protocol requires. The interface logic mainly includes three function modules

The data buffer unit: The data buffer unit is the functional unit that is mainly responsible for the storage control of parallel data; at the same time it receives upper software

ISSN 2278-3091

International Journal of Advanced Trends in Computer Science and Engineering, Vol. 3 , No.1, Pages : 95 – 100 (2014)
Special Issue of ICETETS 2014 - Held on 24-25 February, 2014 in Malla Reddy Institute of Engineering and Technology, Secunderabad– 14, AP, India

order; if there are sending requirements, it sends the parallel data in buffer to parallel to serial conversion unit to process.

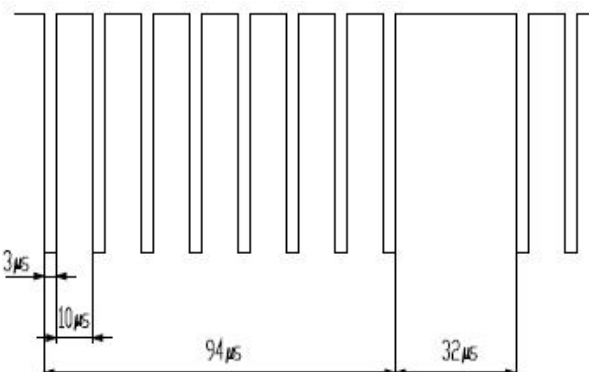

FIG2: Clock Signal Analysis

The clock generating unit: The clock generating unit is the functional unit that is mainly responsible for generating 3/10 duty-cycle clock signal according to protocol requirements; input is the interval bus clock signal which PowerPC output; frequency is 33.33MHz; at the same time it provides sampling basic clock for parallel to serial conversion unit and generates sending finish signal STB.

The parallel to serial conversion unit: The parallel to serial conversion unit is the functional unit that is mainly responsible for the conversion of parallel data is to serial data; the data is output on the CLK clock edge according to protocol requirements After the module are electrified, firstly the data which will be sent is wrote in data buffer unit; then the module receives upper software order to send data; the data buffer unit sends the data to parallel to serial conversion unit to data transform and produces serial DI data; After waiting serial data ready, clock generating unit produces CLK signal according to protocol requirements and produces STB signal after data sending has been finished for 8us, this data sending has been finished

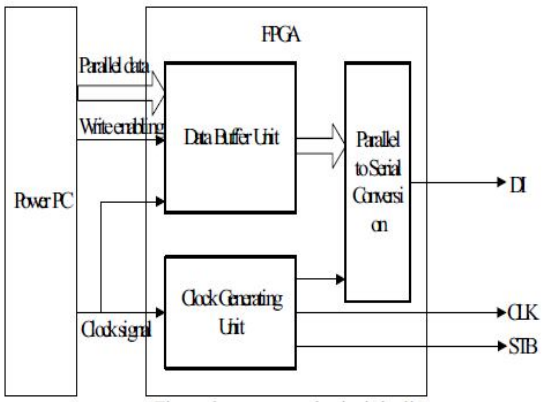

FIG3: General principle diagram

The parallel to serial conversion unit: The parallel to serial conversion unit is the functional unit that is mainly responsible for the conversion of parallel data is to serial data; the data is output on the CLK clock edge according to protocol requirements After the module are electrified, firstly the data which will be sent is wrote in data buffer unit; then the module receives upper software order to send data; the data buffer unit sends the data to parallel to serial conversion unit to data transform and produces serial DI data; After waiting serial data ready, clock generating unit produces CLK signal according to protocol requirements and produces STB signal after data sending has been finished for 8us, this data sending has been finished

III. THE REALIZATION OF THE MODULES

In this section it is shown the realization of three functional units—a data buffer unit, a clock generating unit and a parallel to serial conversion unit.

A. The Data Buffer Unit

The data buffer unit is the functional unit that is responsible for the storage of the parallel data in internal bus through FPGA built-in “distributed double-port RAM “resource to achieve [6]; the principle diagram is shown in Figure 4. This module’s input signal includes write enabling (WR_EN), read enabling (RD_EN), write clock (WR_CLK), and read clock (RD_CLK), write address (WR_ADDR), read address (RD_ADDR) and write data (WR_DATA), Output port uses synchronized output RD_sDATA. After the power is reset, if WR_EN is effective, then data is wrote to WR_ADDR corresponding data unit under the action of the WR_CLK clock edge; if WR_EN is ineffective, then write port is closed. When RD_EN is effective, the data in RD_ADDR address space is read under the action of the RD_CLK clock edge as the initial data in the parallel to serial conversion unit.

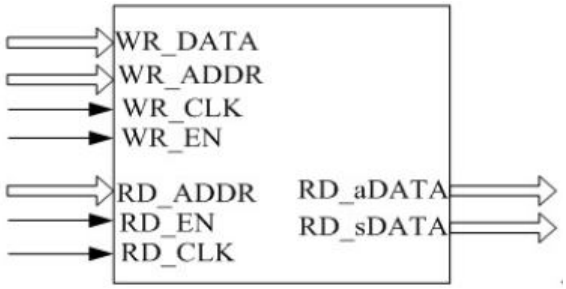

FIG4: the principle diagram of the data buffer unit

B. The Clock Generating Unit

The clock generating unit is the key part of the logic. From Figure 1 we can see that this module need to generate the various of time intervals, such as 3us, 10us, 94us, 32us,8us, 28us and so on. But the greatest common denominator of these data is 1, so we need to produce the clock whose cycle is

ISSN 2278-3091

International Journal of Advanced Trends in Computer Science and Engineering, Vol. 3 , No.1, Pages : 95 – 100 (2014)
Special Issue of ICETETS 2014 - Held on 24-25 February, 2014 in Malla Reddy Institute of Engineering and Technology, Secunderabad– 14, AP, India

1us as the basic clock of all time intervals, at the same time we need to produce the clock signal which has special cycle according to the protocol requirements. Figure3 shows the realization idea of the input clock‘s cycle. Firstly we produce 99.99MHz high frequency clock according to the 33.33MHz input clock’s three frequency doubling. 100 frequency decomposition later, 1 MHz basic clock is generated. And then we use 1MHz clock to sample, after the output control signal is effective, the counters generate the cycle which is 126us, CLK1 and CLK2 whose duty-cycles are respectively 94/32and120/6,then the counters generate CLK3 and CLK4 whose duty cycles are3/10 through CLK1 and CLK2, Figure 6 shows their time sequence. Finally CLK signal and CLK5 clock of protocol requirements are generated by door control; CLK5 is used for parallel to serial conversion sampling. The realization of STB signal is similar to it. After output control is effective,26us is used to data preparation, 94us×4+32us×3=472us is used to data sending, 8us time interval later, STB is effective, that is 26+472+8=506us later STB is effective. After this is kept 28us, STB is ineffective; the next effective output control is waited. Figure 5. the realization of clock generating unit formula of 1MHzbasic clock is 33.33MHz □ 3/100=0.9999MHz 1MHz.Frequency doubling is implemented by Virtex-4 integrated DCM (Digital Clock Manager), DCM provides powerful digital clock management function, including de skew ,frequency doubling, frequency decomposition, phase-shift and so on. In the case of speed priority and low frequency model, input clock may accept the range of 32MHz~150MHz, output clock may reach3 2MHz~210MHzThe input clock of this design is 33.33MHz, and the output clock is 99.99MHz, these satisfy the requirements. And the remaining clock signals are implemented by synchronous counter, 1MHz clock is used for sampling signal, these ensure the signal quality. Figure 6 shows the time sequence, these satisfy the protocol to clock requirements. Figure 6 shows the used clock frequency range of DCM.

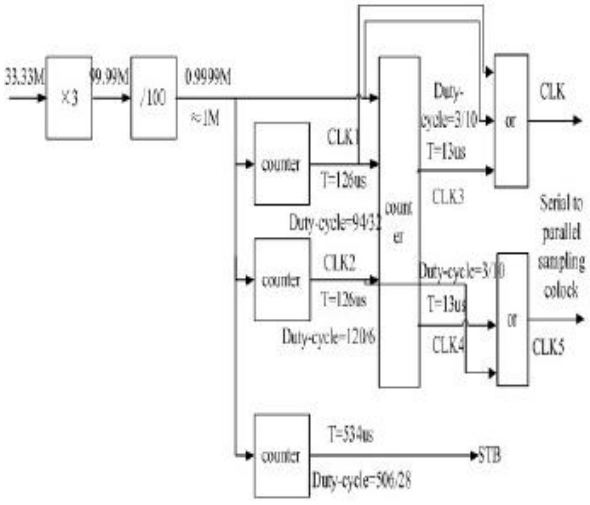

FIG5*:* The Realization of Clock Generating Unit

C. The Parallel to Serial Conversion Unit

The parallel to serial conversion Unit mainly implements the conversion of parallel data to serialdata.Virtex-4provides plenty of parallel to serial conversion resources, so it can implement 2 frequency doubling to 8 frequency doubling parallel to serial conversion [7], and this is suitable for high-speed continuous conversion situation. This module conversion rate is quite low, and the quantity of data is small. Adopting the combination of data buffer and shift register realize the conversion of parallel data to serial data. This method is simple and suitable for low-speed and little data situation, and it can transplant to implement the control of serial interface AD/DA.

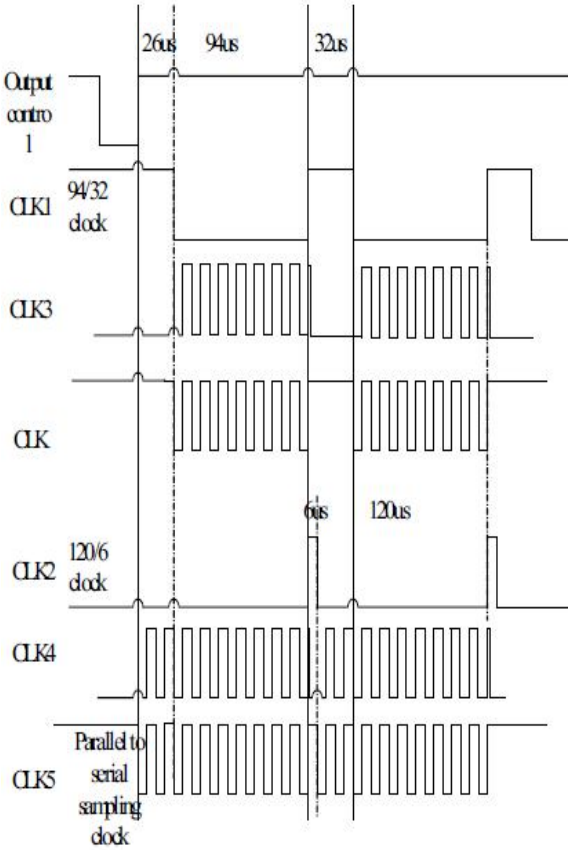

FIG 6: clock generating unit waveforms

IV. EMULATION RESULT:

After the design has been completed, we carry on the emulation to the function. The parallel input data is in turn"11111111", "00001111", "01010101", and “10101010".Figure 7 shows the waveform diagram. From this waveform diagram, we can see that this program has realized the extraction of effective data bits to input data, and carried on serial output according to certain baud rate. Data transmission is steady; data output satisfy the protocol

ISSN 2278-3091

International Journal of Advanced Trends in Computer Science and Engineering, Vol. 3 , No.1, Pages : 95 – 100 (2014)
Special Issue of ICETETS 2014 - Held on 24-25 February, 2014 in Malla Reddy Institute of Engineering and Technology, Secunderabad– 14, AP, India

requirements; the specific function and capability have been validated in system-test.

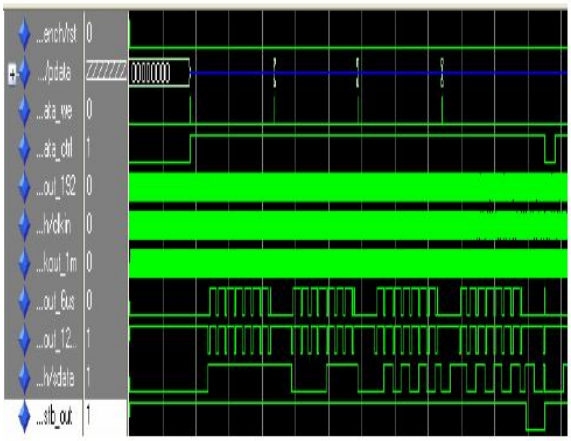

Figure 7. Emulation result

SRAM-based Devices:

As seen of SRAM configuration in Chapter 1, the majority of FPGAs are based on the use cells, which means that they can be configured over and over again. The main advantages of this programming technology are that new design ideas can be quickly implemented and tested, while evolving standards and protocols can be accommodated relatively easily. Furthermore, when the system is first powered up, the FPGA can initially be programmed to perform one function such as a self-test or board/system test, and it can then be reprogrammed to perform its main task. Another big advantage of the SRAM-based approach is that these devices are at the forefront of technology. FPGA vendors can leverage the fact that many other companies specializing in memory devices expend tremendous resources on *research and development (R & D)* in this area. Furthermore, the SRAM cells are created using exactly the same CMOS technologies as the rest of the device, so no special processing steps are required in order to create these components. Unfortunately, there's no such thing as a free lunch. One downside of SRAM based devices is that they have to be reconfigured every time the system is powered up. This either requires the use of a special external memory device (which has an associated cost and consumes real estate on the board) or of an on-board microprocessor

Security Issues:

Another consideration with regard to SRAM-based devices is that it can be difficult to protect your intellectual property, or IP, in the form of your design. This is because the configuration file used to program the device is stored in
Some form of external memory.
On the bright side, some of today's SRAM-based FPGAs support the concept of bit stream encryption. In this case, the final configuration data is encrypted before being stored in the external memory device. The encryption key itself is loaded into a special SRAM-based register in the FPGA via its JTAG port .In conjunction with some associated logic, this key allows the incoming encrypted configuration bit stream to be decrypted as it's being loaded into the device. The command/process of loading an encrypted bit stream automatically disables the FPGA's read-back capability. This means that you will typically use unencrypted configuration data during development (where you need to use read-back) and then start to use encrypted data when you move into production. (You can load an unencrypted bit stream at any time, so you can easily load a test configuration and then reload the encrypted version.)

FPGA Logic Block Fundamentals and Trade-Offs:

The purpose of a logic block in an FPGA is to provide the basic computation and storage elements used in digital logic systems. As used in the original gate arrays, the most simple and un-specific way of providing this capability is to use a transistor as the basic logic element, and build gates and storage elements from it. This approach was indeed attempted in a commercial FPGA from the now-defunct company Cross point .This kind of very fine-grained logic block, however, requires the use of large amounts of programmable interconnect to create any typical logic function. It will result in an FPGA that is bound to suffer from area-inefficiency (because programmable routing is expensive in terms of area), low performance (each routing "hop" is slow), and high power consumption (because of the higher capacitance of programmable interconnect that must be charged and discharged). At the other extreme, a logic block could be an entire processor. This approach exists in the commercial space, although processors are mixed with some more fine grained logic blocks in a device . Such a logic block on its own would not have the performance gains that come from customizable hardware. In addition, if such a block was used to implement a 2-input AND gate, it would be incredibly inefficient, which illustrates the danger of using logic blocks that are too coarse-grained. In between these extremes is a spectrum of logic block choices ranging from fine to coarse-grain logic blocks. FPGA architects over the last two decades have selected basic logic blocks made of transistors, NAND gates, an interconnection of multiplexers lookup tables, and PAL-style wide-input gates. These choices were originally driven by intuitive insights on the part of architects, typically with very little data or analysis, with a few exceptions. In addition to a basic logic block, many modern FPGAs contain a heterogeneous mixture of different blocks, some of which can only be used for very specific functions, such as dedicated memory blocks or multipliers. These structures are very efficient at implementing specific functions, yet go to waste if unused. A central issue in FPGA architecture design is the selection of specific, hard circuits for inclusion in an FPGA.
In general, we are interested in knowing the effect of an FPGA's architecture on the area-efficiency, speed, and power of a set of application circuits implemented in the FPGA. The set of applications represent the "target market" of the FPGA. Ideally, this set would include all digital hardware applications if just a single FPGA architecture could serve all possible markets, maximizing its advantage as a single

ISSN 2278-3091

International Journal of Advanced Trends in Computer Science and Engineering, Vol. 3 , No.1, Pages : 95 – 100 (2014)
Special Issue of ICETETS 2014 - Held on 24-25 February, 2014 in Malla Reddy Institute of Engineering and Technology, Secunderabad– 14, AP, India

standard device. Modern commercial practice requires the use of several architectural families to serve different market segments. It is a common practice in FPGA architecture research to employ an empirical approach to study and explore different architectures. Here, application circuits are synthesized into different architectures through a CAD flow that is able to vary the architectural elements.

V. **FUTURE FPGA DEVELOPMENTS**

Super-fast I/O:

FPGA chips typically sport one or more of these transceiver blocks, each of which has multiple channels. Each channel can carry 2.5 Gbps of real data; So four channels have to be combined to achieve 10 Gbps. Furthermore, an external device has to be employed to convert an incoming optical signal into the four channels of electrical data that are passed to the FPGA. Conversely, this device will accept four channels of electrical data from the FPGA and convert them into a single outgoing optical signal. Some FPGAs today can accept and generate these 10 Gbps optical signals internally.

Super-fast Confi guration:

The vast majority of today's FPGAs are configured using a serial bit-stream or a parallel stream only 8 bits wide. This severely limits the way in which **192** FPGAs: Instant Access these devices can be used in reconfigurable computing-type applications. Quite some time ago (somewhere around the mid-1990s), a team at Pilkington Microelectronics (PMEL) in the United Kingdom came up with a novel FPGA architecture in which the device's primary I/O pins were also used to load the configuration data. This provided a super wide bus (256 or more pins/bits) that could program the device in a jiffy. As an example of where this sort of architecture might be applicable, consider the fact that there is a wide variety of compressor/decompressor (CODEC) algorithms that can be used to compress and decompress audio and data. If you have a system that needs to decompress different files that were compressed using different algorithms, then you are going to need to support a variety of different CODECs. Assuming that you wished to perform this decompression in hardware using an FPGA, then with traditional devices you would either have to implement each CODEC in its own device or as a separate area in a larger device. You wouldn't wish to reprogram the FPGA to perform the different algorithms on the fly because this would take from 1 to 2.5 seconds with a large component, which is too long for an end user to wait. By comparison, in the case of the PMEL architecture, the reconfiguration data could be appended to the front of the file to be processed .The idea was that the configuration data would flood through the wide bus, program the device in a fraction of a second, and be immediately followed by the main audio or video data file to be decompressed. If the next file to be processed required a different CODEC, then the appropriate configuration file could be used to reprogram the device. This concept was applicable to a wide variety of applications. Unfortunately, the original incarnation of this technology fell by the wayside, but it's not beyond the bounds of possibility that something like this could reappear in the not-so-distant future.

More Hard IP:

In the case of technology nodes of 90 nm and below, it's possible to squeeze so many transistors onto a chip that we are almost certainly going to see an increased amount of hard IP blocks for such things as communications functions, special-purpose processing functions, microprocessor peripherals, and the like. Traditional digital FPGA vendors have a burning desire to grab as many of the functions on a circuit board as possible and to suck these functions into their devices. In the short term, this might mean that FPGAs start to include hard IP blocks with analog content such as analog-to-digital (A/D) and digital to- analog (D/A) converters. Such blocks would be programmable with regard to such things as the number of quanta (width) and the dynamic range of the analog signals they support. They might also include amplification and some Filtering and signal conditioning functions. Furthermore, over the years a number of companies have promoted different, flavors of field-programmable analog arrays (FPAAs). Thus, there is more than a chance that predominantly digital FPGAs Will start to include areas of truly programmable analog functionality similar to that provided in pure FPAA devices.

Embedding FPGA Cores in ASIC Fabric:

The cost of developing a modern ASIC at the 90-nm technology node is horrendous. This problem is compounded by the fact that, once you've completed a design and built the chip, your algorithms and functions are effectively "frozen in silicon. " This means that if you have to make any changes in the future, you're going to have to regenerate the design, create a new set of photo-masks (costing around $1million), and build a completely new chip. To address these issues, some users are interested in creating ASICs with FPGA cores embedded into the fabric. Apart from anything else, this means that you can use the same design for multiple end applications without having to create new mask sets. I also think that we are going to see increased deployment of structured ASICs and that these will lend themselves to sporting embedded FPGA cores because their design styles and tools will exhibit a lot of commonality.FPGA area-efficiency is a key metric because the size of the FPGA die dictates a significant portion of its cost, particularly for devices with a large logic capacity. (For smaller devices, I/O and packaging also become significant in the cost of the devices.

VI. CONCLUSION

With the enhancement of the comprehensive mission management system integration rate, the equipment with which mission management computer needs to cross-link are more and more. In this paper the design method of the private

serial interface based on FPGA is shown, it has realized the new function, shortened the development cycle, reduced manpower investment and adhered to the principle of the module standardization in the case of without increasing original module kind. This design method is worth promoting in future design.

REFERENCES

[1] A. Aggarwal and D. Lewis, "Routing architectures for hierarchical field programmable gate arrays," in IEEE International Conference on Computer Design, pp. 475–478, October 1994.

[2] E. Ahmed, The Effect of Logic Block Granularity on Deep-Submicron FPGA Performance and Density. Master's thesis, University of Toronto, Department of Electrical and Computer Engineering, 2001.

[3] E. Ahmed and J. Rose, "The effect of LUT and cluster size on deep-submicron FPGA performance and density," in Proceedings of the 2000 ACM/SIGDA Eighth International Symposium on Field Programmable Gate Arrays, pp. 3–12, ACM Press, 2000.

[4] E. Ahmed and J. Rose, "The effect of LUT and cluster size on deep-submicron FPGA performance and density," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 12, no. 3, pp. 288–298, March 2004.

[5] J. Anderson and F. Najm, "A novel low-power FPGA routing switch," in Proceedings of the IEEE Custom Integrated Circuits Conference, pp. 719–722,October 2004.

[6] H. B. Bakaglu, Circuits, Interconnection, and Packaging for VLSI. Reading, MA: Addison Wesley, 1990.

[7] R. Baumann, "Soft errors in advanced computer systems," IEEE Design and Test of Computers, vol. 22, no. 3, pp. 258–266, 2005.

[8] M. J. Beauchamp, S. Hauck, K. D. Underwood, and K. S. Hemmer, "Embedded floating-point units in FPGAs," in FPGA'06: Proceedings of the International Symposium on Field Programmable Gate Arrays, pp. 12–20, USA, New York, NY: ACM Press, 2006.

[9] V. Betz and J. Rose, "Improving FPGA performance via the use of architecture families," in 3rd ACM Intl. Symposium on Field-Programmable Gate Arrays, pp. 10–16, 1995.

[10] V. Betz and J. Rose, "How much logic should go in an FPGA logic block?,"IEEE Design and Test of Computers, vol. 15, no. 1, pp. 10–15, January–March1998.

[11] V. Betz and J. Rose, "Circuit design, transistor sizing and wire layout of FPGA interconnect," in Proceedings of the IEEE Custom Integrated Circuits Conference, pp. 171–174, 1999.

[12] V. Betz and J. Rose, "FPGA routing architecture: Segmentation and buffering to optimize speed and density," in Proceeding: ACM/SIGDA International Symposium on Field Programmable Gate Arrays, pp. 140–149, February 1999.

[13] C. Bolchini, D. Quarta, and M. D. Santambrogio, "SEU mitigation for SRAM based FPGAs through dynamic partial reconfiguration," in GLSVLSI '07: Proceedings of the 17th Great Lakes Symposium on VLSI, pp. 55–60, USA, New York, NY: ACM Press, 2007.

[14] K. A. Bowman, S. G. Duvall, and J. D. Meindl, "Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for giga scale integration,"

AUTHORS:

CH.RAMA KOTI REDDY, received M.tech degree from JNTU Ananthapuram, presently working as an assistant professor in malla reddy institute of engineering and technology Hyderabad

SK.NAGULMEERA, received M.tech degree from JNTU Hyderabad, presently working as an assistant professor in malla reddy institute of engineering and technology Hyderabad.

A.NARESH, received M.tech degree from HU Chennai, presently working as an assistant professor in malla reddy institute of engineering and technology Hyderabad.